Основы Linux для хакеров

Русский перевод книги Linux Basics for Hackers.

Больше крутых материалов в моем телеграм-канале @grogrigs

LINUX BASICS FOR HACKERS

GETTING STARTED WITH *NETWORKING*, *SCRIPTING*, AND *SECURITY* IN KALI

OCCUPYTHEWEB

Титульная страница

Основы Linux для хакеров

Начало работы с сетями, сценариями и безопасностью в Kali

OccupyTheWeb

San Francisco

Авторские права

Основы Linux для хакеров. Авторское право © 2019, OccupyTheWeb.

Все права защищены. Никакая часть этой работы не может быть воспроизведена или передана в какой-либо форме или какими-либо средствами, электронными или механическими, включая фотокопирование, запись или любую систему хранения и извлечения информации, без предварительного письменного разрешения владельца авторских прав и издателя.

ISBN-10: 1-59327-855-1

ISBN-13: 978-1-59327-855-7

Издатель: William Pollock

Редакторы производства: Serena Yang и Meg Sneeringer

Иллюстрация обложки: Josh Ellingson

Внутреннее оформление: Octopod Studios

Редактор развития: Liz Chadwick

Технический рецензент: Cliff Janzen

Литературный редактор: Barton D. Reed

Верстальщики: Serena Yang и Meg Sneeringer

Корректор: Paula L. Fleming

Составитель указателя: JoAnne Burek

Для информации о распространении, переводах или оптовых продажах, пожалуйста, свяжитесь с No Starch Press, Inc. напрямую:

No Starch Press, Inc.

245 8th Street, San Francisco, CA 94103

телефон: 1.415.863.9900; info@nostarch.com

www.nostarch.com

Каталогизационные данные Библиотеки Конгресса

Имена: OccupyTheWeb, автор.

Название: Основы Linux для хакеров: начало работы с сетями, сценариями и безопасностью в Kali / OccupyTheWeb.

Описание: первое издание. | San Francisco : No Starch Press, Inc., [2018].

Идентификаторы: LCCN 2018030544 (печатное издание) | LCCN 2018032646 (электронная книга) | ISBN 9781593278564 (EPUB) | ISBN 159327856X (EPUB) | ISBN 9781593278557 (печатное издание) | ISBN 1593278551 (печатное издание) | ISBN 9781593278564 (электронная книга) | ISBN 159327856X (электронная книга)

Темы: LCSH: тестирование на проникновение (компьютерная безопасность) | Kali Linux. | Хакеры. | Операционные системы (компьютеры)

Классификация: LCC QA76.9.A25 (электронная книга) | LCC QA76.9.A25 O325 2018 (печатное издание) | DDC 005.8--dc23

Запись LC доступна по адресу https://lccn.loc.gov/2018030544

No Starch Press и логотип No Starch Press являются зарегистрированными товарными знаками No Starch Press, Inc. Другие названия продуктов и компаний, упомянутые здесь, могут быть товарными знаками их соответствующих владельцев. Вместо использования символа товарного знака при каждом появлении товарного имени мы используем эти имена только в редакционном стиле и в интересах владельца товарного знака, без намерения нарушить права на товарный знак.

Информация в этой книге распространяется на условиях "как есть", без гарантии. Хотя при подготовке этой работы были приняты все меры предосторожности, ни автор, ни No Starch Press, Inc. не несут ответственности перед каким-либо лицом или организацией за какие-либо потери или ущерб, вызванные или предположительно вызванные прямо или косвенно информацией, содержащейся в ней.

Введение

Хакинг - важнейший набор навыков XXI века! Я говорю это не легкомысленно. События последних лет, кажется, каждое утро подтверждают это заголовками новостей. Государства шпионят друг за другом, чтобы получить секреты; киберпреступники крадут миллиарды долларов; выпускаются цифровые черви, требующие выкуп; противники влияют на выборы друг друга; участники конфликтов выводят из строя коммунальные системы друг друга. Все это - работа хакеров, и их влияние на наш все более цифровой мир только начинает ощущаться.

Я решил написать эту книгу после работы с десятками тысяч начинающих хакеров через Null-Byte, https://www.hackers-arise.com/, и почти со всеми родами войск и разведывательными ведомствами США (NSA, DIA, CIA и FBI). Этот опыт показал мне, что у многих начинающих хакеров почти нет или совсем нет опыта работы с Linux, и именно отсутствие такого опыта становится главным препятствием в начале пути к профессиональному хакингу. Почти все лучшие хакерские инструменты написаны для Linux, поэтому базовые навыки Linux - обязательное условие для того, чтобы стать профессиональным хакером. Я написал эту книгу, чтобы помочь начинающим хакерам преодолеть этот барьер.

Хакинг - элитная профессия в сфере IT. Поэтому он требует широкого и детального понимания концепций и технологий IT. На самом фундаментальном уровне Linux является необходимостью. Если вы хотите сделать хакинг и информационную безопасность своей карьерой, я настоятельно рекомендую вкладывать время и силы в использование и понимание Linux.

Эта книга не предназначена для опытного хакера или опытного администратора Linux. Напротив, она предназначена для тех, кто хочет начать увлекательный путь хакинга, кибербезопасности и пентестинга. Кроме того, это не всеобъемлющий трактат о Linux или хакинге, а отправная точка в эти миры. Книга начинается с основ Linux и переходит к базовым сценариям на bash и Python. Там, где это уместно, я старался использовать примеры из мира хакинга, чтобы объяснять принципы Linux.

В этом введении мы рассмотрим рост этического хакинга в информационной безопасности, а также пройдем процесс установки виртуальной машины, чтобы вы могли установить Kali Linux на свою систему, не затрагивая операционную систему, которая у вас уже работает.

Что в этой книге

В первой группе глав вы освоитесь с основами Linux: глава 1 познакомит вас с файловой системой и терминалом и даст несколько базовых команд. Глава 2 покажет, как манипулировать текстом, чтобы находить, изучать и изменять программы и файлы.

В главе 3 вы будете управлять сетями. Вы просканируете сети, найдете сведения о подключениях и замаскируете себя, скрывая сетевую и DNS-информацию. Глава 4 научит добавлять, удалять и обновлять программы, а также поддерживать систему в аккуратном состоянии. В главе 5 вы будете управлять правами доступа к файлам и каталогам, чтобы контролировать, кто к чему имеет доступ. Вы также изучите некоторые приемы повышения привилегий.

Глава 6 научит управлять службами, включая запуск и остановку процессов и распределение ресурсов, чтобы дать вам больший контроль. В главе 7 вы будете управлять переменными окружения ради оптимальной производительности, удобства и даже скрытности. Вы будете находить и фильтровать переменные, менять переменную `PATH` и создавать новые переменные окружения.

Глава 8 познакомит вас со сценариями bash - обязательным инструментом любого серьезного хакера. Вы изучите основы bash и построите сценарий для сканирования целевых портов, в которые позднее сможете проникать.

Главы 9 и 10 дадут вам важные навыки управления файловой системой: вы узнаете, как сжимать и архивировать файлы, чтобы поддерживать систему в порядке, копировать целые устройства хранения и получать информацию о файлах и подключенных дисках.

Поздние главы глубже погружаются в темы хакинга. В главе 11 вы будете использовать систему журналирования и манипулировать ею, чтобы получать сведения об активности цели и заметать собственные следы. Глава 12 покажет, как использовать и злоупотреблять тремя основными службами Linux: веб-сервером Apache, OpenSSH и MySQL. Вы создадите веб-сервер, построите удаленный видеошпион и узнаете о базах данных и их уязвимостях. Глава 13 покажет, как оставаться защищенным и анонимным с помощью прокси-серверов, сети Tor, VPN и зашифрованной электронной почты.

Глава 14 посвящена беспроводным сетям. Вы изучите базовые сетевые команды, затем взломаете точки доступа Wi-Fi и научитесь обнаруживать сигналы Bluetooth и подключаться к ним.

Глава 15 глубже погружается в сам Linux и дает общий обзор того, как работает ядро и как его драйверы можно использовать для доставки вредоносного программного обеспечения. В главе 16 вы изучите важные навыки планирования задач, чтобы автоматизировать свои хакерские сценарии. Глава 17 научит вас основным концепциям Python, и вы напишете два хакерских инструмента: сканер для наблюдения за TCP/IP-подключениями и простой взломщик паролей.

Что такое этический хакинг

В последние годы вместе с ростом области информационной безопасности резко выросла и область этического хакинга, также известного как хакинг "белой шляпы", то есть хакинг "хороших парней". Этический хакинг - это практика попыток проникновения в систему и ее эксплуатации, чтобы найти слабые места и лучше защитить ее. Я делю область этического хакинга на два основных компонента: тестирование на проникновение для законной компании в сфере информационной безопасности и работу на военные или разведывательные ведомства своей страны. Обе области быстро растут, и спрос на них высок.

Тестирование на проникновение

По мере того как организации все сильнее задумываются о безопасности, а стоимость нарушений безопасности растет экспоненциально, многие крупные организации начинают передавать услуги безопасности внешним подрядчикам. Одна из ключевых таких услуг - тестирование на проникновение. По сути, тест на проникновение - это законный заказной взлом, предназначенный для демонстрации уязвимости сети и систем компании.

Обычно организации сначала проводят оценку уязвимостей, чтобы найти потенциальные уязвимости в своей сети, операционных системах и службах. Я подчеркиваю слово "потенциальные", потому что такое сканирование уязвимостей включает значительное число ложноположительных результатов: вещей, определенных как уязвимости, хотя на самом деле ими они не являются. Роль тестировщика на проникновение состоит в том, чтобы попытаться взломать эти уязвимости, то есть проникнуть через них. Только после этого организация может узнать, реальна ли уязвимость, и решить, стоит ли вкладывать время и деньги в ее закрытие.

Военная сфера и шпионаж

Почти каждое государство на земле теперь занимается кибершпионажем и кибервойной. Достаточно бегло просмотреть заголовки, чтобы увидеть: кибероперации стали выбранным методом шпионажа и атак на военные и промышленные системы.

Хакинг играет ключевую роль в этих военных и разведывательных действиях, и со временем это будет становиться только более очевидным. Представьте войну будущего, где хакеры смогут получить доступ к военным планам противника и вывести из строя его электросеть, нефтеперерабатывающие заводы и системы водоснабжения. Такие действия происходят уже сейчас, каждый день. Поэтому хакер становится ключевым компонентом обороны своей страны.

Почему хакеры используют Linux

Так почему же хакеры используют Linux, а не другие операционные системы? Главным образом потому, что Linux дает гораздо более высокий уровень контроля несколькими разными способами.

Linux имеет открытый исходный код

В отличие от Windows, Linux имеет открытый исходный код, то есть исходный код операционной системы доступен вам. Поэтому вы можете менять его и манипулировать им по своему усмотрению. Если вы пытаетесь заставить систему работать способами, для которых она не была предназначена, возможность манипулировать исходным кодом необходима.

Linux прозрачен

Чтобы эффективно взламывать, вы должны знать и понимать свою операционную систему и, в значительной степени, операционную систему, которую атакуете. Linux полностью прозрачен: мы можем видеть все его рабочие части и манипулировать ими.

С Windows все иначе. Microsoft очень старается сделать так, чтобы внутреннее устройство ее операционных систем было как можно труднее понять, поэтому вы никогда по-настоящему не знаете, что происходит "под капотом". В Linux же прожектор направлен прямо на каждый компонент операционной системы. Это делает работу с Linux более эффективной.

Linux дает тонкий контроль

Linux гранулярен. Это означает, что у вас есть почти бесконечный контроль над системой. В Windows вы можете контролировать только то, что Microsoft позволяет вам контролировать. В Linux всем можно управлять из терминала - как на мельчайшем уровне, так и на самом крупном. Кроме того, Linux делает сценарии на любых скриптовых языках простыми и эффективными.

Большинство хакерских инструментов написано для Linux

Гораздо более 90 процентов всех хакерских инструментов написано для Linux. Конечно, есть исключения, такие как Cain and Abel и Wikto, но эти исключения только подтверждают правило. Даже когда такие хакерские инструменты, как Metasploit или `nmap`, портируют на Windows, не все возможности переносятся из Linux.

Будущее принадлежит Linux/Unix

Это может показаться радикальным заявлением, но я твердо верю, что будущее информационных технологий принадлежит системам Linux и Unix. Microsoft пережила свой расцвет в 1980-е и 1990-е годы, но ее рост замедляется и застаивается.

С самого появления интернета Linux/Unix был предпочтительной операционной системой для веб-серверов благодаря своей стабильности, надежности и устойчивости. Даже сегодня Linux/Unix используется на двух третях веб-серверов и доминирует на рынке. Встроенные системы в маршрутизаторах, коммутаторах и других устройствах почти всегда используют ядро Linux, а мир виртуализации находится под доминированием Linux: и VMware, и Citrix построены на ядре Linux.

Более 80 процентов мобильных устройств работают под управлением Unix или Linux (iOS - это Unix, а Android - Linux). Поэтому если вы считаете, что будущее вычислений лежит в мобильных устройствах, таких как планшеты и телефоны (а спорить с этим было бы трудно), тогда будущее - за Unix/Linux. У Microsoft Windows всего 7 процентов рынка мобильных устройств. Хотите ли вы впрягаться именно в эту повозку?

Загрузка Kali Linux

Прежде чем начать, нужно загрузить и установить Kali Linux на свой компьютер. Именно с этим дистрибутивом Linux мы будем работать на протяжении всей книги. Linux был впервые разработан Линусом Торвальдсом в 1991 году как альтернатива Unix с открытым исходным кодом. Поскольку он открыт, добровольные разработчики пишут код ядра, утилит и приложений. Это означает, что нет единой корпоративной структуры, надзирающей за разработкой, и в результате часто не хватает соглашений и стандартизации.

Kali Linux был разработан Offensive Security как хакерская операционная система, построенная на дистрибутиве Linux под названием Debian. Существует много дистрибутивов Linux, и Debian - один из лучших. Вероятно, вам больше всего знаком Ubuntu как популярный настольный дистрибутив Linux. Ubuntu также построен на Debian. Среди других дистрибутивов - Red Hat, CentOS, Mint, Arch и SUSE. Хотя все они используют одно и то же ядро Linux (сердце операционной системы, которое управляет процессором, оперативной памятью и так далее), у каждого есть собственные утилиты, приложения и выбор графического интерфейса (GNOME, KDE и другие) для разных целей. Поэтому каждый из этих дистрибутивов Linux выглядит и ощущается немного по-разному. Kali был разработан для тестировщиков на проникновение и хакеров и поставляется со значительным набором хакерских инструментов.

Я настоятельно рекомендую использовать Kali для этой книги. Хотя вы можете использовать другой дистрибутив, вам, скорее всего, придется загрузить и установить разные инструменты, которые мы будем использовать, а это может означать много часов загрузки и установки. Кроме того, если этот дистрибутив не построен на Debian, могут быть и другие небольшие отличия. Загрузить и установить Kali можно с https://www.kali.org/.

На домашней странице щелкните ссылку загрузок в верхней части страницы. На странице загрузок перед вами будет несколько вариантов загрузки. Важно выбрать правильный. В левой части таблицы вы увидите имя образа - название версии, которую загружает ссылка. Например, первое имя образа, которое я вижу, - 64-битная версия Kali Linux; это означает, что это полный Kali Linux и он подходит для 64-битных систем. Большинство современных систем используют 64-битный CPU Intel или AMD. Чтобы определить, какой тип CPU установлен в вашей системе, перейдите в панель управления, затем в раздел системы и безопасности, а затем в раздел системы: он должен быть указан там. Если ваша система 64-битная, загрузите и установите 64-битную версию полного Kali (не облегченную и не любой другой альтернативный вариант).

Если у вас старый компьютер с 32-битным CPU, вам нужно установить 32-битную версию, которая находится ниже на странице.

У вас есть выбор между загрузкой через HTTP и Torrent. Если выбрать HTTP, Kali загрузится прямо в вашу систему, как любой другой файл, и будет помещен в папку загрузок. Загрузка через torrent - это одноранговая загрузка, используемая многими сайтами обмена файлами. Для нее понадобится приложение для torrent-загрузок, например BitTorrent. Файл Kali затем будет загружен в папку, где это приложение хранит свои загрузки.

Есть и другие версии для других типов CPU, например для широко используемой архитектуры ARM, встречающейся во множестве мобильных устройств. Если вы используете Raspberry Pi, планшет или другое мобильное устройство (пользователи телефонов, скорее всего, предпочтут Kali NetHunter), обязательно загрузите и установите версию Kali для архитектуры ARM, прокрутив страницу до образов ARM и щелкнув образы Kali для ARM.

Kali загружен, но прежде чем что-либо устанавливать, я хочу немного поговорить о виртуальных машинах. Как правило, для новичка установка Kali в виртуальную машину - лучшее решение для обучения и практики.

Виртуальные машины

Технология виртуальных машин (VM) позволяет запускать несколько операционных систем на одном аппаратном устройстве, например ноутбуке или настольном компьютере. Это означает, что вы можете продолжать использовать привычную операционную систему Windows или macOS и запускать виртуальную машину с Kali Linux внутри этой операционной системы. Чтобы изучать Linux, не нужно перезаписывать уже установленную ОС.

Существует множество приложений виртуальных машин от VMware, Oracle, Microsoft и других поставщиков. Все они хороши, но здесь я покажу, как загрузить и установить бесплатный VirtualBox от Oracle.

Установка VirtualBox

VirtualBox можно загрузить с https://www.virtualbox.org/, как показано на рис. 1. Щелкните ссылку загрузок в левом меню и выберите пакет VirtualBox для текущей операционной системы вашего компьютера, которая будет хостом для VM VirtualBox. Обязательно загрузите последнюю версию.

Рис. 1: Домашняя страница VirtualBox

Когда загрузка завершится, щелкните установочный файл, и перед вами появится знакомый мастер установки, показанный на рис. 2.

Рис. 2: Диалог мастера установки

Щелкните "Далее", и вы должны увидеть экран выборочной настройки, как на рис. 3.

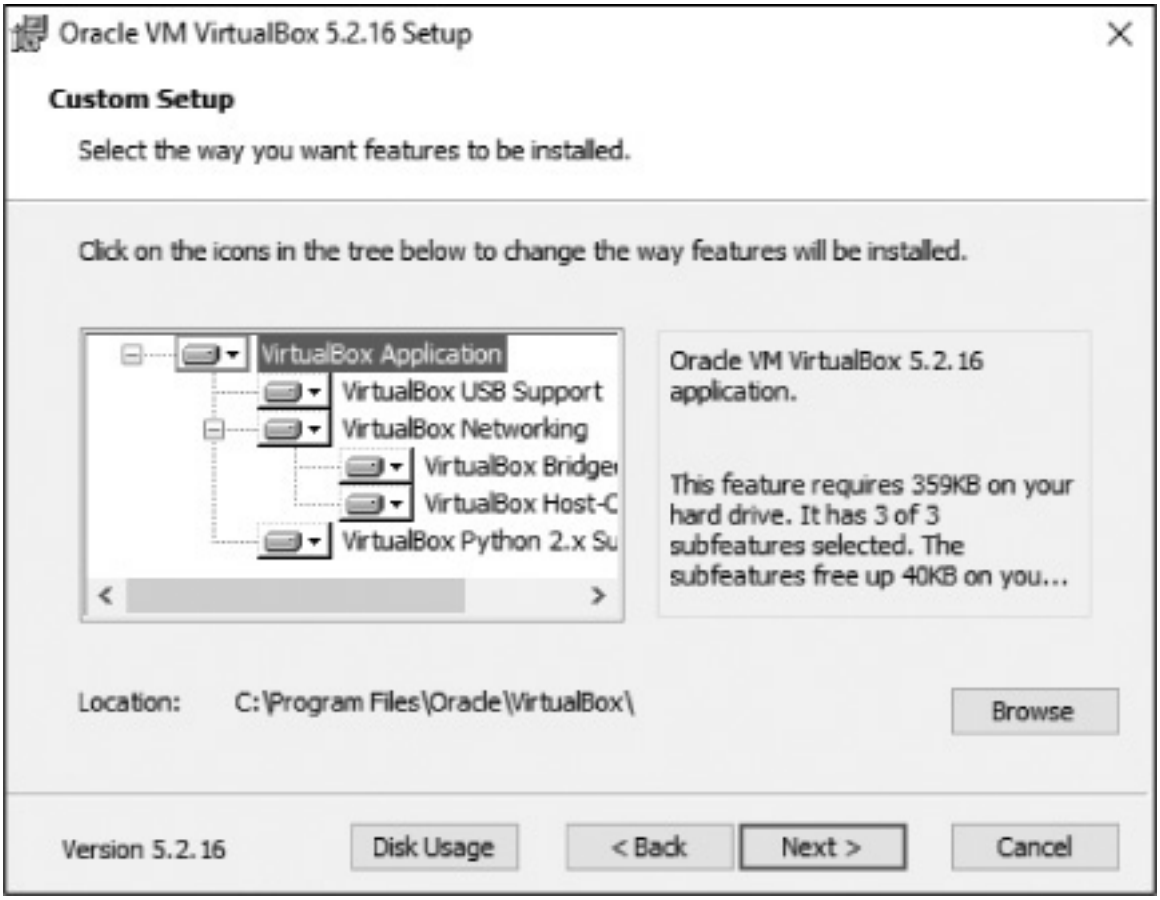

Рис. 3: Диалог выборочной настройки

На этом экране просто щелкните "Далее". Продолжайте щелкать "Далее", пока не дойдете до предупреждения о сетевых интерфейсах, а затем щелкните "Да".

Щелкните "Установить", чтобы начать процесс. Во время этого процесса вам, вероятно, несколько раз предложат установить программное обеспечение устройств. Это виртуальные сетевые устройства, необходимые для связи ваших виртуальных машин. Каждый раз щелкайте "Установить".

Когда установка завершится, щелкните "Готово".

Настройка виртуальной машины

Теперь начнем работу с вашей виртуальной машиной. VirtualBox должен открыться после установки; если этого не произошло, откройте его, и вы должны увидеть менеджер VirtualBox, как на рис. 4.

Рис. 4: Менеджер VirtualBox

Поскольку мы будем создавать новую виртуальную машину с Kali Linux, щелкните "Создать" в левом верхнем углу. Откроется диалог создания виртуальной машины, показанный на рис. 5. Дайте машине имя (подойдет любое, но я просто использовал Kali), затем выберите Linux в раскрывающемся меню типа. Наконец, выберите Debian (64-bit) в третьем раскрывающемся меню (если только вы не используете 32-битную версию Kali; в этом случае выберите 32-битную версию Debian). Щелкните "Далее", и вы увидите экран, похожий на рис. 6. Здесь нужно выбрать, сколько оперативной памяти вы хотите выделить этой новой виртуальной машине.

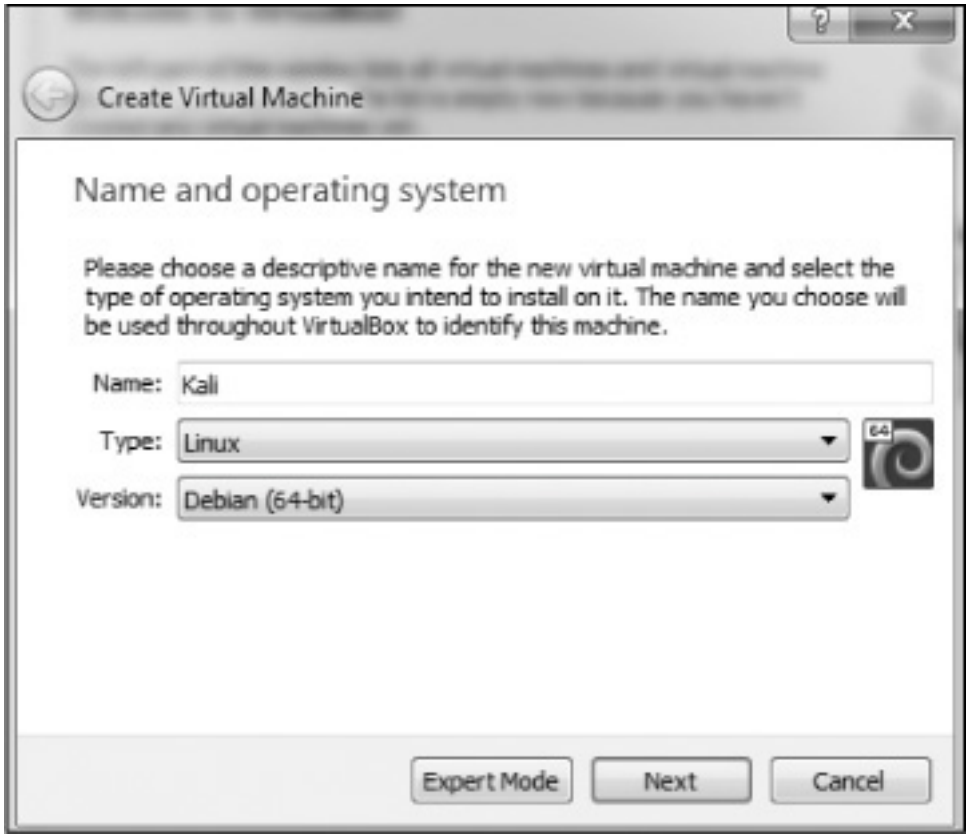

Рис. 5: Диалог создания виртуальной машины

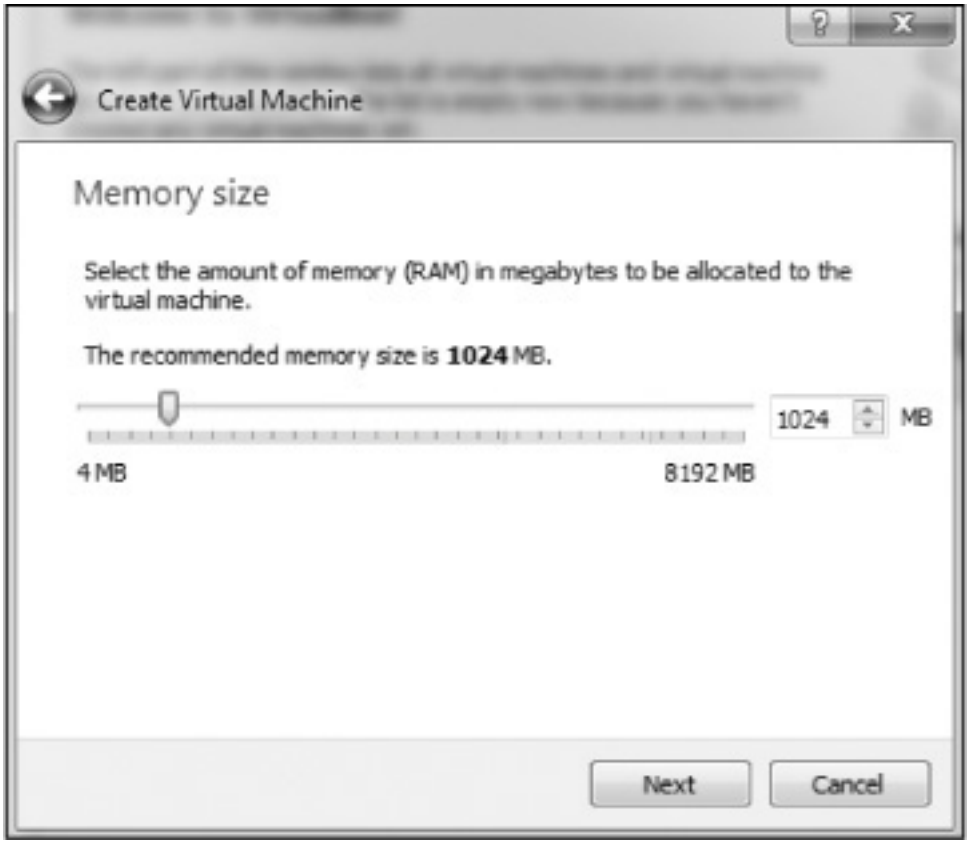

Рис. 6: Выделение памяти

Как правило, я не рекомендую использовать более 25 процентов общего объема оперативной памяти вашей системы. Это означает: если на вашей физической, или хостовой, системе

установлено 4 ГБ, выберите для виртуальной машины всего 1 ГБ; если на физической системе 16 ГБ, выберите 4 ГБ. Чем больше оперативной памяти вы дадите виртуальной машине, тем лучше и быстрее она будет работать, но вы также должны оставить достаточно оперативной памяти для хостовой операционной системы и любых других виртуальных машин, которые можете захотеть запускать одновременно. Ваши виртуальные машины не используют оперативную память, когда вы ими не пользуетесь, но они занимают место на жестком диске.

Щелкните "Далее", и вы перейдете к экрану жесткого диска. Выберите создание виртуального жесткого диска и щелкните "Создать".

На следующем экране можно решить, хотите ли вы, чтобы создаваемый жесткий диск выделялся динамически или имел фиксированный размер. Если выбрать динамическое выделение, система не займет весь максимальный объем, выделенный для виртуального жесткого диска, пока он вам не понадобится; так будет сохранено больше неиспользуемого места на жестком диске хостовой системы. Я предлагаю выбрать динамическое выделение.

Щелкните "Далее", и вы выберете объем места на жестком диске для VM и расположение VM (см. рис. 7).

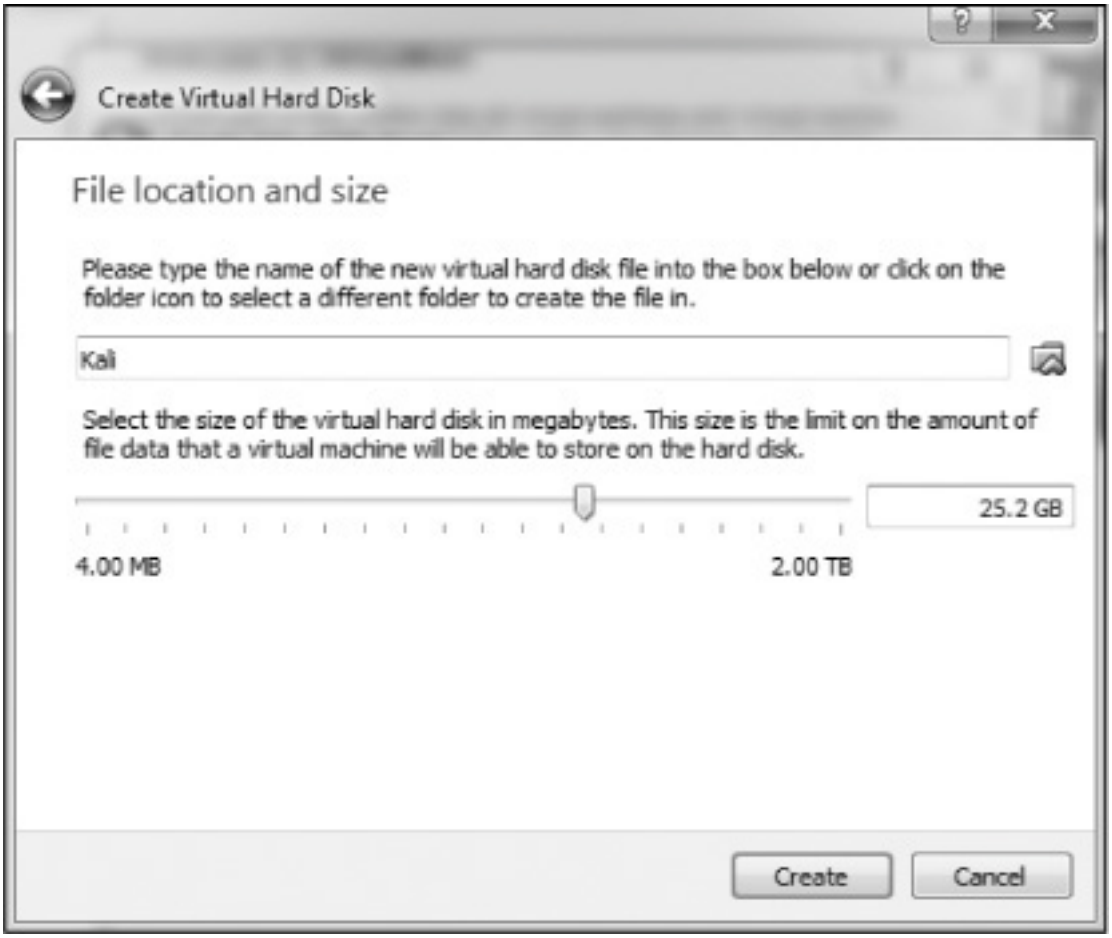

Рис. 7: Выделение места на жестком диске

Значение по умолчанию - 8 ГБ. Обычно я считаю его немного малым и рекомендую выделить минимум 20-25 ГБ. Помните: если вы выбрали динамическое выделение места на жестком диске, оно не будет использовать это место, пока оно вам не понадобится; а расширение жесткого диска после того, как он уже был выделен, может быть непростой задачей, так что лучше ошибиться в большую сторону.

Щелкните "Создать", и вы готовы двигаться дальше!

Установка Kali на VM

На этом этапе вы должны увидеть экран, похожий на рис. 8. Теперь нужно установить Kali. Обратите внимание: слева в менеджере VirtualBox должно быть указано, что VM Kali выключена. Щелкните кнопку запуска (значок с зеленой стрелкой).

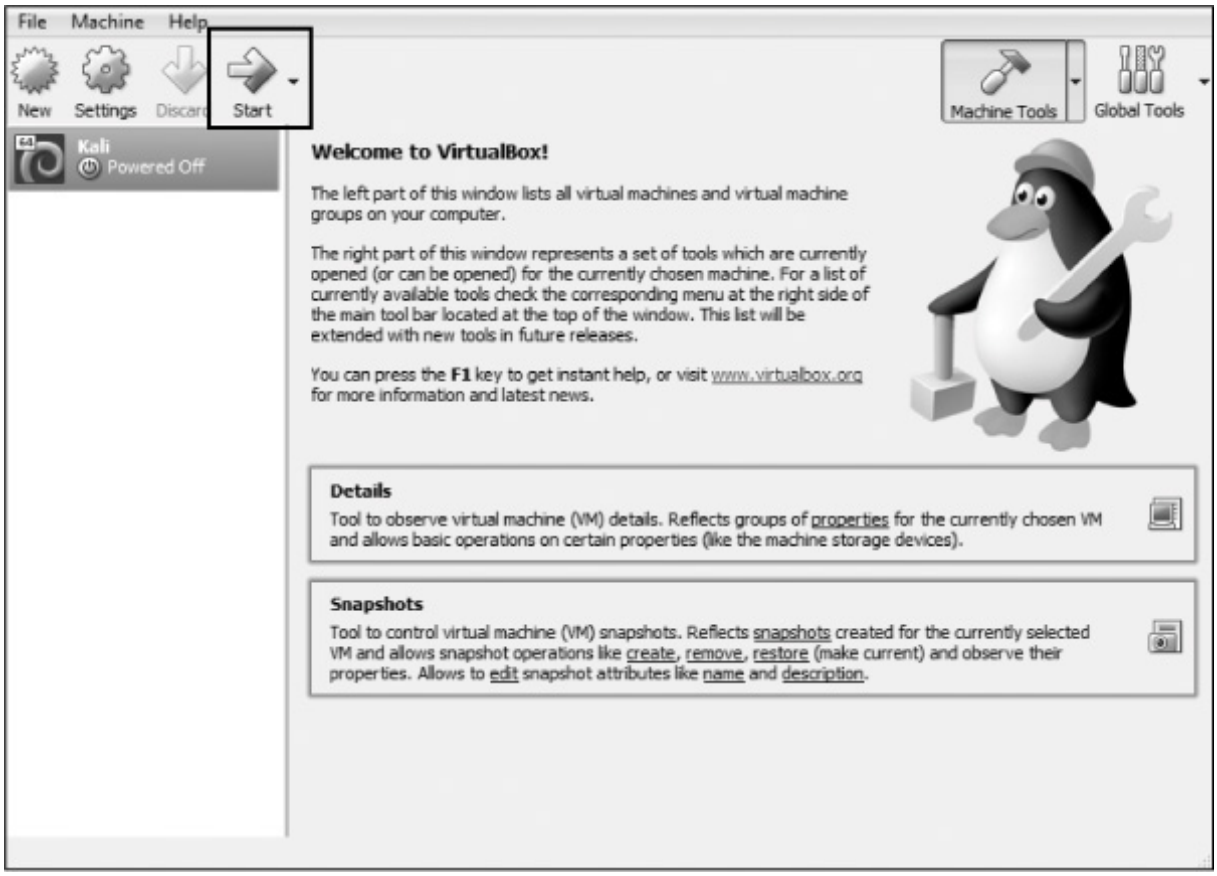

Рис. 8: Экран приветствия VirtualBox

Затем менеджер VirtualBox спросит, где найти загрузочный диск. Вы уже загрузили образ диска с расширением `.iso`, который должен находиться в папке загрузок (если вы загружали Kali через torrent, файл `.iso` будет в папке загрузок вашего приложения для torrent-загрузок). Щелкните значок папки справа, перейдите в папку загрузок и выберите файл образа Kali (см. рис. 9).

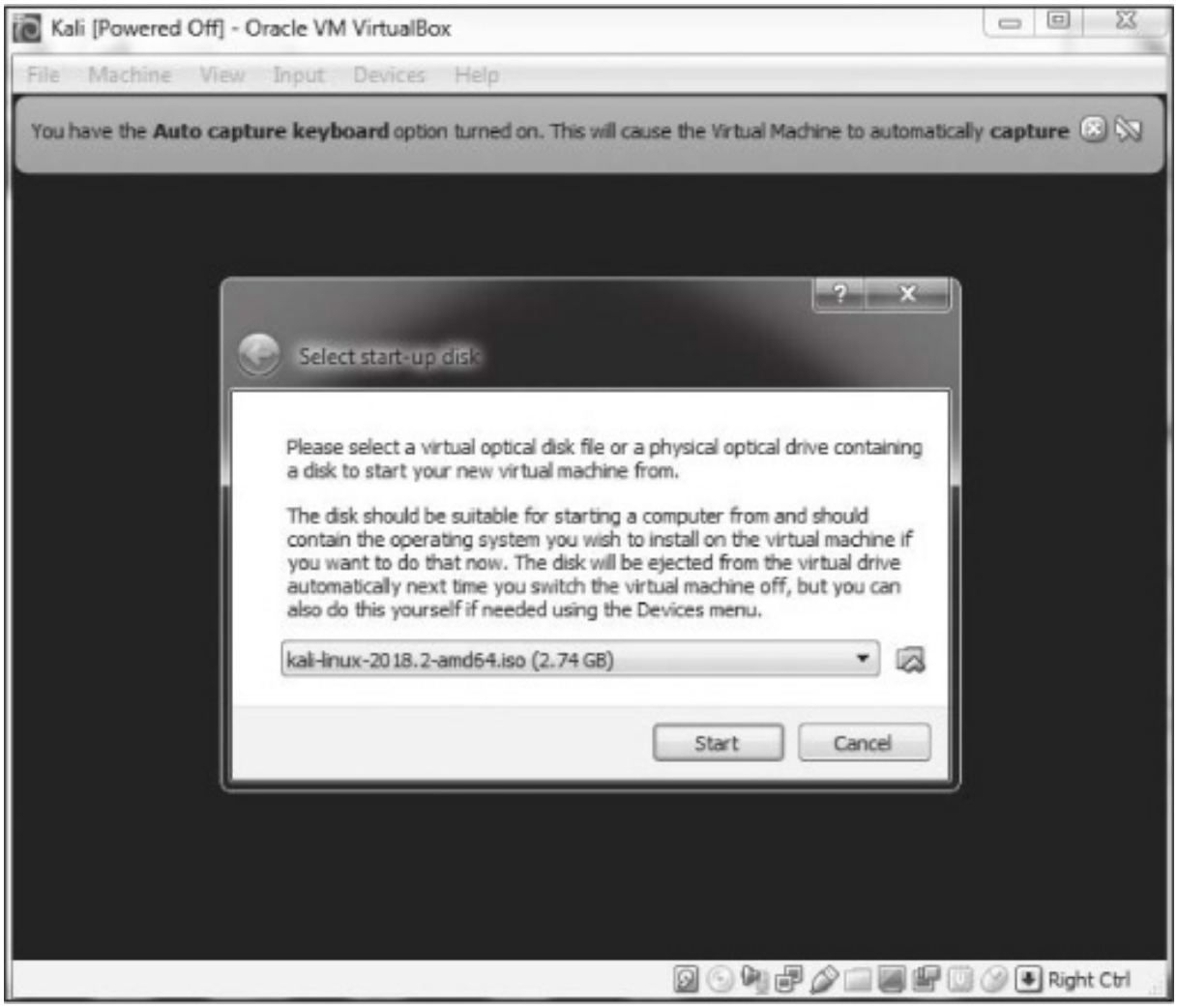

Рис. 9: Выбор загрузочного диска

Затем щелкните "Запустить". Поздравляю, вы только что установили Kali Linux на виртуальную машину!

Настройка Kali

Теперь Kali откроет экран, похожий на рис. 10, и предложит несколько вариантов запуска. Я советую новичкам использовать графическую установку. Для перемещения по меню используйте клавиши клавиатуры.

Если при установке Kali в VirtualBox вы получите ошибку, вероятно, в BIOS вашей системы не включена виртуализация. Каждая система и ее BIOS немного отличаются, поэтому уточните у производителя или поищите в интернете решения для вашей системы и BIOS. Кроме того, в системах Windows, скорее всего, нужно отключить конкурирующее программное обеспечение виртуализации, например Hyper-V. И снова поиск в интернете по вашей системе должен помочь это сделать.

Рис. 10: Выбор метода установки

Далее вам предложат выбрать язык. Обязательно выберите язык, на котором вам удобнее всего работать, а затем щелкните "Продолжить". Затем выберите свое местоположение, щелкните "Продолжить" и выберите раскладку клавиатуры.

Когда вы щелкнете "Продолжить", VirtualBox пройдет процесс обнаружения вашего оборудования и сетевых адаптеров. Просто терпеливо подождите, пока он это сделает. В конце концов вы увидите экран с просьбой настроить сеть, как на рис. 11.

Рис. 11: Ввод имени хоста

Первый пункт, который он запрашивает, - имя вашего хоста. Вы можете назвать его как угодно, но я оставил значение по умолчанию - `kali`.

Затем вам предложат ввести доменное имя. Здесь вводить ничего не нужно. Щелкните "Продолжить". Следующий экран, показанный на рис. 12, очень важен. Здесь вас просят указать пароль, который вы хотите использовать для пользователя root.

Рис. 12: Выбор пароля

Пользователь root в Linux - всесильный системный администратор. Вы можете использовать любой пароль, который считаете надежным. Если бы это была физическая система, которую мы используем в интернете, я бы предложил очень длинный и сложный пароль, чтобы ограничить возможность злоумышленника взломать его. Поскольку это виртуальная машина, к которой люди не могут получить доступ, не получив сначала доступ к вашей хостовой операционной системе, парольная аутентификация на этой виртуальной машине менее важна, но все же выбирайте осмотрительно.

Щелкните "Продолжить", и вам предложат установить часовой пояс. Сделайте это и продолжайте.

Следующий экран спрашивает о разметке дисков (раздел - это именно то, что звучит: часть или сегмент вашего жесткого диска). Выберите управляемое использование всего диска, и Kali обнаружит ваши жесткие диски и автоматически настроит разметчик.

Затем Kali предупредит, что все данные на выбранном диске будут стерты... но не волнуйтесь! Это виртуальный диск, он новый и пустой, поэтому на самом деле ничего не произойдет. Щелкните "Продолжить".

Теперь Kali спросит, хотите ли вы хранить все файлы в одном разделе или использовать отдельные разделы. Если бы это была производственная система, вы, вероятно, выбрали бы отдельные разделы для `/home`, `/var` и `/tmp`, но поскольку мы будем использовать ее как учебную систему в виртуальной среде, можно просто выбрать вариант хранения всех файлов в одном разделе.

Теперь вас спросят, нужно ли записать изменения на диск. Выберите завершение разметки и запись изменений на диск. Kali еще раз спросит, хотите ли вы записать изменения на диск; выберите "Да" и щелкните "Продолжить" (см. рис. 13).

Рис. 13: Запись изменений на диск

Теперь Kali начнет устанавливать операционную систему. Это может занять некоторое время, так что наберитесь терпения. Самое время сходить в ванную и взять любимый напиток.

Когда установка завершится, вам предложат указать, хотите ли вы использовать сетевое зеркало. На самом деле это не обязательно, поэтому щелкните No.

Затем Kali спросит, хотите ли вы установить GRUB (Grand Unified Bootloader), показанный на рис. 14. Загрузчик позволяет выбирать разные операционные системы для загрузки; это означает, что при запуске машины вы можете загрузиться в Kali или в другую операционную систему. Выберите "Да" и щелкните "Продолжить".

Рис. 14: Установка GRUB

На следующем экране вам предложат выбрать, хотите ли вы установить загрузчик GRUB автоматически или вручную. По пока неясным причинам, если выбрать второй вариант, Kali после установки склонен зависать и показывать пустой экран. Выберите ручной ввод устройства, как показано на рис. 15.

Рис. 15: Ввод устройства вручную

На следующем экране выберите диск, куда должен быть установлен загрузчик GRUB (скорее всего, это будет что-то вроде `/dev/sda`). Пройдите до следующего экрана, который должен сообщить, что установка завершена.

Поздравляю! Вы установили Kali. Щелкните "Продолжить". Kali попытается перезагрузиться, и вы увидите множество строк кода на пустом черном экране, прежде чем в итоге перед вами появится экран входа Kali 2018, показанный на рис. 16.

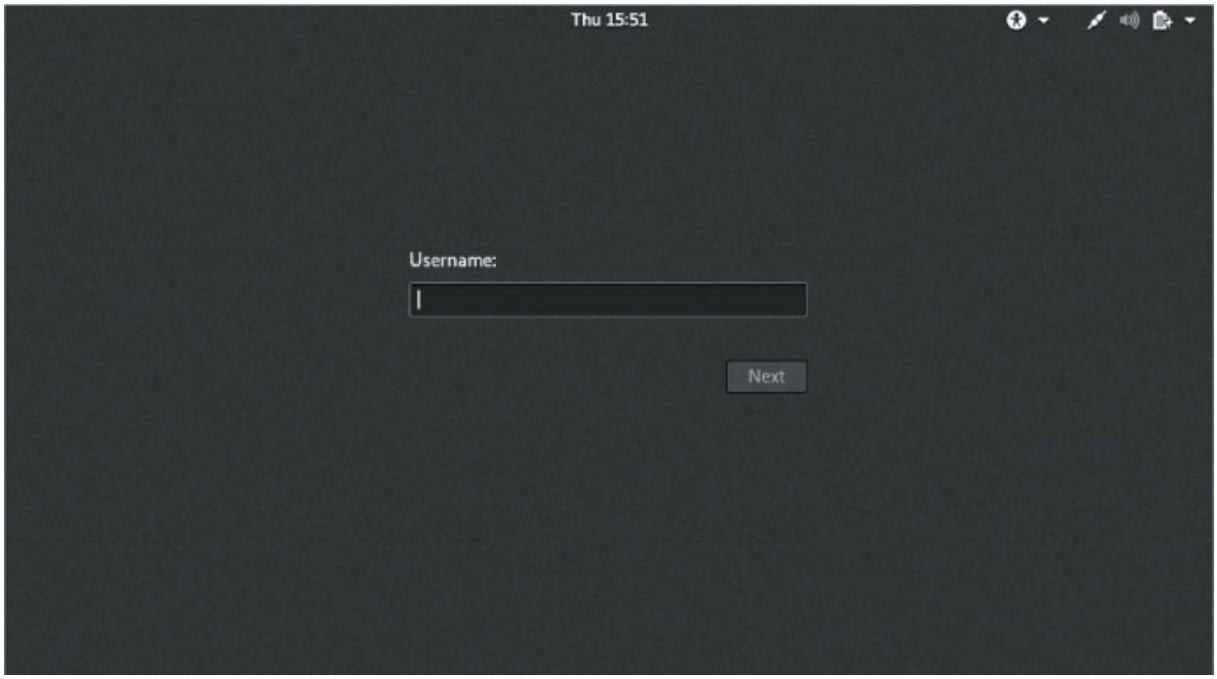

Рис. 16: Экран входа Kali

Войдите как root, и вас попросят ввести пароль. Введите тот пароль, который выбрали для пользователя root.

После входа как root перед вами появится рабочий стол Kali Linux, как на рис. 17.

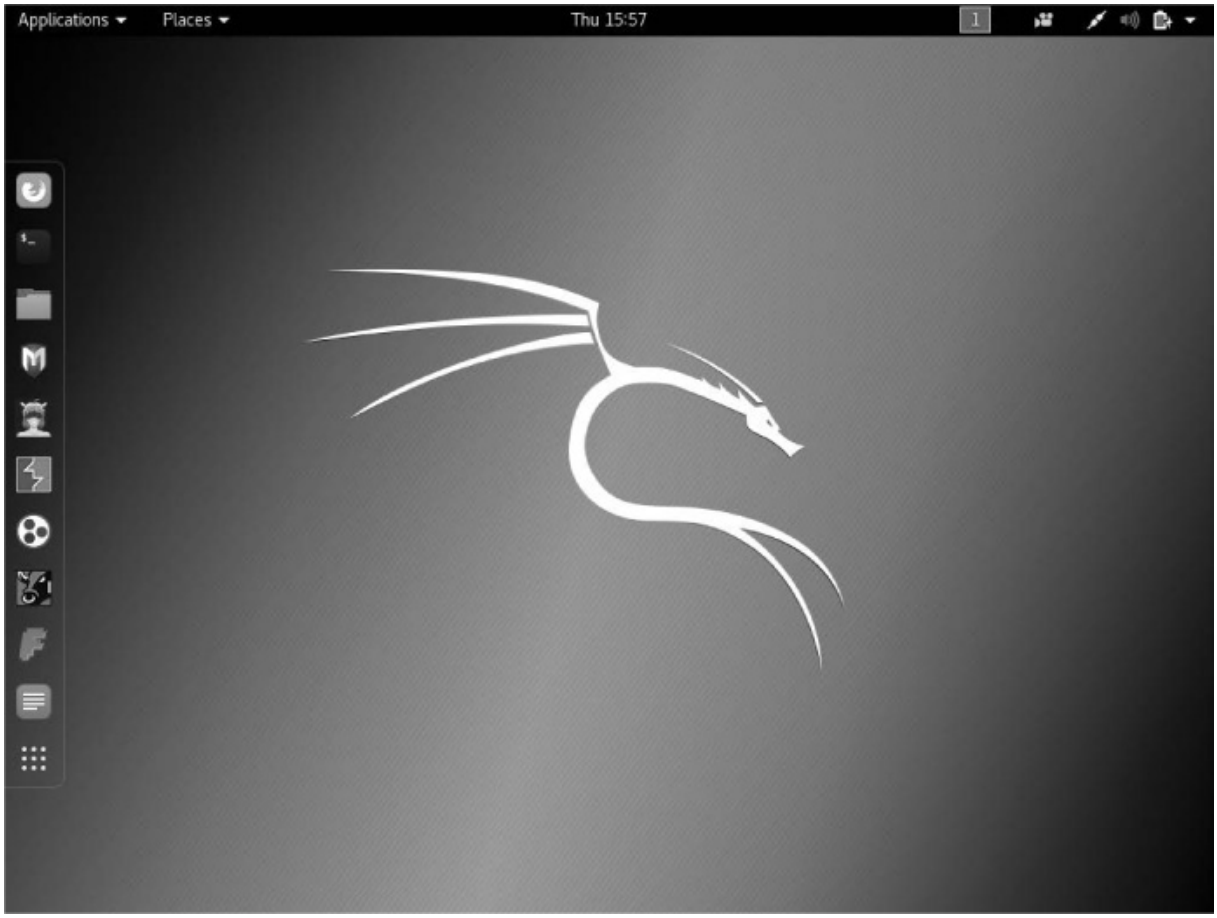

Рис. 17: Домашний экран Kali

Теперь вы готовы начать свое путешествие в увлекательную область хакинга! Добро пожаловать!

Глава 1. Начало работы с основами

По самой своей природе хакеры - люди действия. Мы хотим трогать вещи и играть с ними. Мы также хотим создавать и иногда ломать. Немногие из нас хотят читать длинные тома теории информационных технологий, прежде чем смогут заняться тем, что любят больше всего: хакингом. С учетом этого эта глава задумана так, чтобы дать вам фундаментальные навыки, достаточные для запуска и работы в Kali... прямо сейчас!

В этой главе мы не будем разбирать ни одну концепцию в больших подробностях: мы охватим ровно столько, чтобы вы могли играть и исследовать операционную систему хакеров - Linux. Более глубокие обсуждения оставим для следующих глав.

Вводные термины и понятия

Прежде чем мы начнем путешествие по замечательному миру "Основ Linux для хакеров", я хочу ввести несколько терминов, которые помогут прояснить некоторые понятия, обсуждаемые далее в этой главе.

Бинарные файлы. Этот термин относится к файлам, которые можно выполнять, подобно исполняемым файлам в Windows. Бинарные файлы обычно находятся в каталогах `/usr/bin` или `/usr/sbin` и включают утилиты вроде `ps`, `cat`, `ls` и `cd` (в этой главе мы коснемся всех четырех), а также приложения вроде инструмента для взлома беспроводных сетей `aircrack-ng` и системы обнаружения вторжений (IDS) Snort.

Чувствительность к регистру. В отличие от Windows, Linux чувствителен к регистру. Это означает, что `Desktop` отличается от `desktop`, а тот отличается от `DeskTop`. Каждое из этих имен будет обозначать другой файл или каталог. Многих людей, пришедших из среды Windows, это раздражает. Если вы получаете сообщение об ошибке "файл или каталог не найден" и уверены, что файл или каталог существует, скорее всего, нужно проверить регистр символов.

Каталог. Это то же самое, что папка в Windows. Каталог дает способ организовывать файлы, обычно иерархически.

Домашний каталог. У каждого пользователя есть собственный каталог `/home`, и обычно именно там по умолчанию сохраняются создаваемые вами файлы.

Kali. Kali Linux - это дистрибутив Linux, специально созданный для тестирования на проникновение. В нем предустановлены сотни инструментов, что экономит часы, которые пришлось бы потратить

на их самостоятельную загрузку и установку. Я буду использовать последнюю на момент написания версию Kali: Kali 2018.2, впервые выпущенную в апреле 2018 года.

root. Как почти в любой операционной системе, в Linux есть учетная запись администратора, или суперпользователя, предназначенная для доверенного человека, который может делать в системе почти все. Это включает перенастройку системы, добавление пользователей и изменение паролей. В Linux такая учетная запись называется `root`. Как хакер или пентестер вы часто будете использовать учетную запись `root`, чтобы получить контроль над системой. Более того, многие хакерские инструменты требуют использования учетной записи `root`.

Сценарий. Это серия команд, выполняемых в интерпретируемой среде, которая преобразует каждую строку в исходный код. Многие хакерские инструменты - это просто сценарии. Сценарии можно запускать с помощью интерпретатора bash или интерпретаторов других скриптовых языков, таких как Python, Perl или Ruby. Сейчас Python - самый популярный интерпретатор среди хакеров.

Оболочка. Это среда и интерпретатор для выполнения команд в Linux. Самая широко используемая оболочка - bash, что означает "снова Bourne", но среди других популярных оболочек есть оболочки C и Z. В этой книге я буду использовать исключительно оболочку bash.

Терминал. Это интерфейс командной строки (CLI).

Разобравшись с этими основами, мы попытаемся методично развить важнейшие навыки Linux, которые понадобятся вам, чтобы стать хакером или тестировщиком на проникновение. В этой первой главе я проведу вас через первые шаги работы с Kali Linux.

Экскурсия по Kali

После запуска Kali вас встретит экран входа, показанный на рис. 1-1. Войдите, используя имя пользователя учетной записи `root` и пароль по умолчанию `toor`.

Рис. 1-1: Вход в Kali с использованием учетной записи root

Теперь у вас должен быть доступ к рабочему столу Kali (см. рис. 1-2). Мы быстро рассмотрим два самых базовых аспекта рабочего стола: интерфейс терминала и файловую структуру.

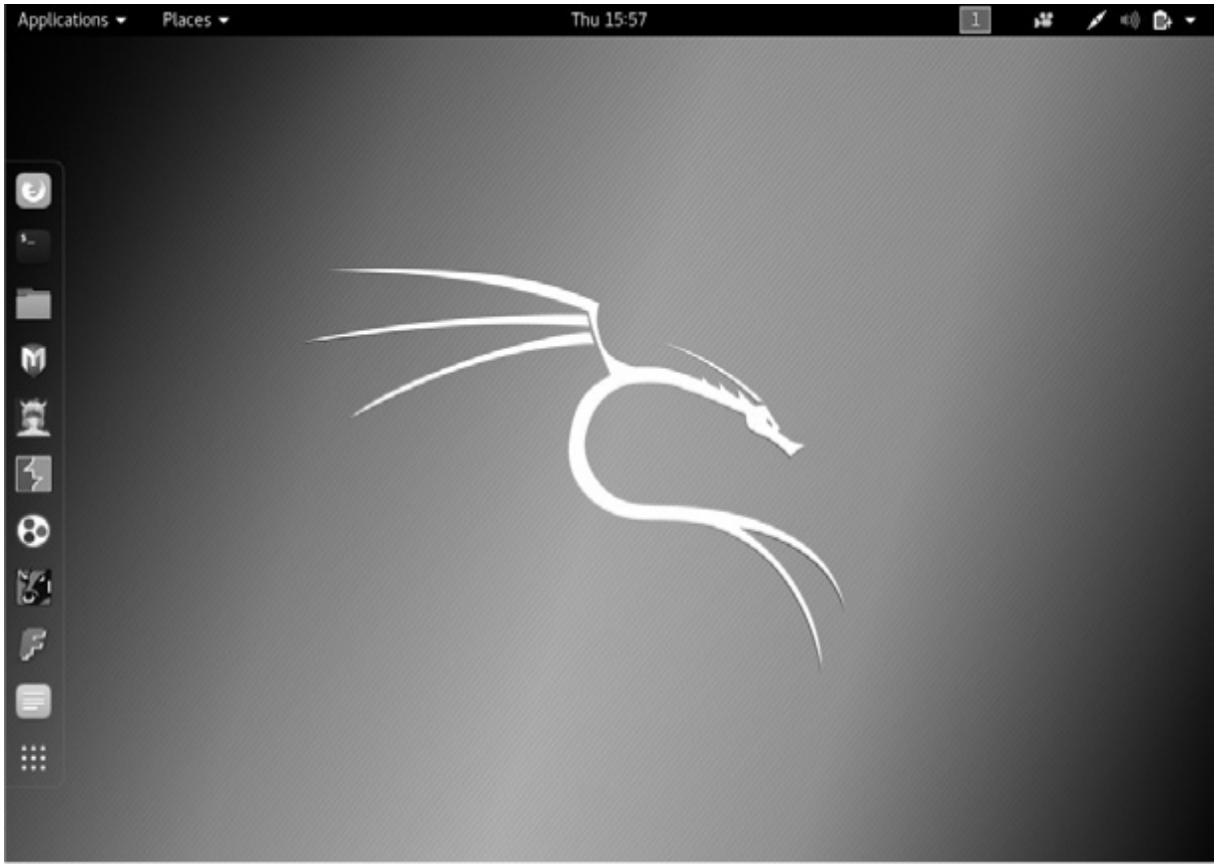

Рис. 1-2: Рабочий стол Kali

Терминал

Первый шаг в использовании Kali - открыть терминал, то есть интерфейс командной строки, который мы будем использовать в этой книге. В Kali Linux значок терминала находится внизу рабочего стола. Дважды щелкните этот значок, чтобы открыть терминал, или нажмите `CTRL-ALT-T`. Ваш новый терминал должен выглядеть так, как показано на рис. 1-3.

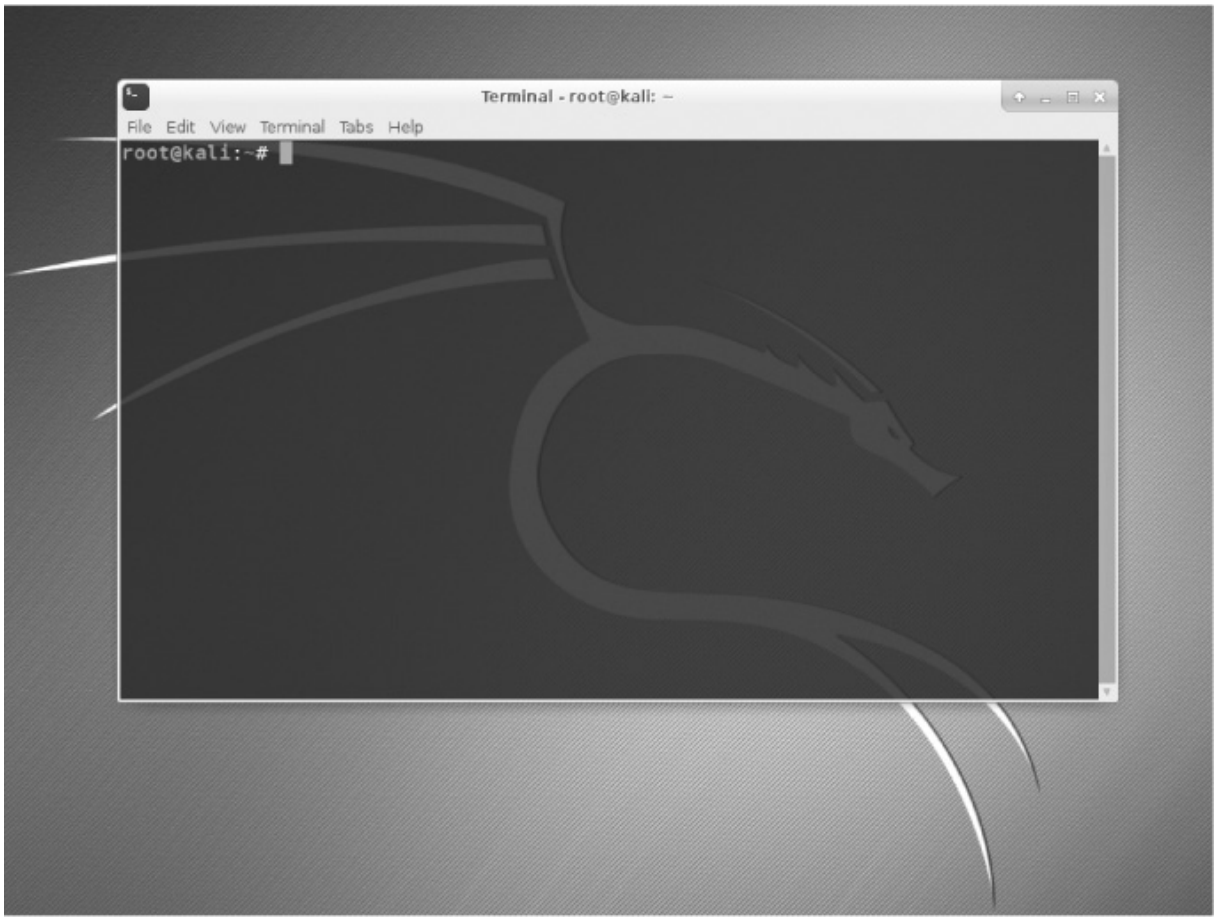

Рис. 1-3: Терминал Kali

Этот терминал открывает среду командной строки, известную как оболочка, которая позволяет выполнять команды в базовой операционной системе и писать сценарии. Хотя в Linux много разных оболочек, самая популярная - оболочка bash; она же является оболочкой по умолчанию в Kali и многих других дистрибутивах Linux.

Чтобы изменить пароль, можно использовать команду `passwd`.

Файловая система Linux

Структура файловой системы Linux несколько отличается от структуры Windows. В основе файловой системы Linux нет физического диска (например, диска `C:`), вместо этого используется логическая файловая система. В самом верху структуры файловой системы находится `/`, который часто называют корнем файловой системы, как если бы это было перевернутое дерево (см. рис. 1-4). Имейте в виду, что это не то же самое, что пользователь `root`. Сначала эти термины могут казаться запутанными, но их станет легче различать, когда вы привыкнете к Linux.

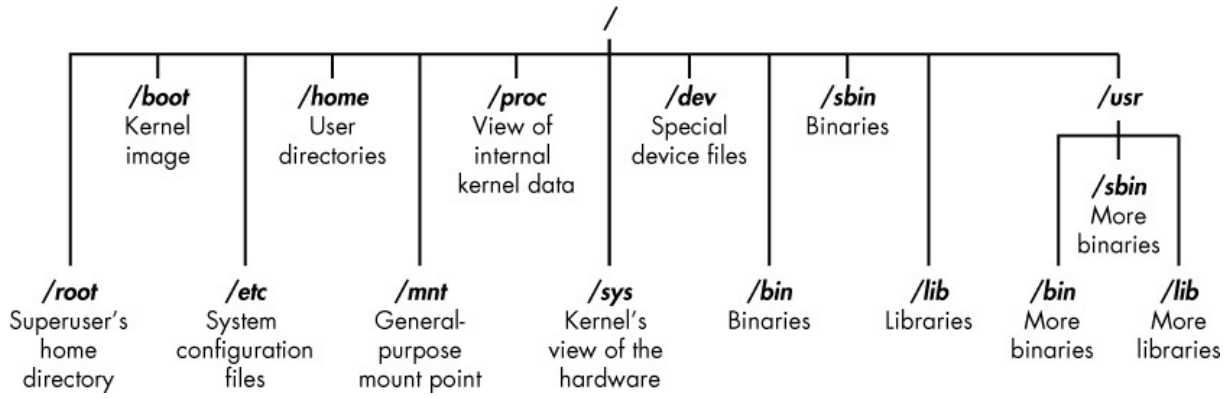

Рис. 1-4: Файловая система Linux

Корень (`/`) файловой системы находится на вершине дерева, а ниже перечислены самые важные подкаталоги, которые нужно знать:

- `/root` - домашний каталог всемогущего пользователя `root`;
- `/etc` - обычно содержит конфигурационные файлы Linux, то есть файлы, управляющие тем, когда и как запускаются программы;
- `/home` - домашний каталог пользователя;
- `/mnt` - место, где другие файловые системы присоединяются, или монтируются, к файловой системе;
- `/media` - место, где обычно присоединяются, или монтируются, CD и USB-устройства;
- `/bin` - место, где находятся бинарные файлы приложений (эквивалент исполняемых файлов в Microsoft Windows);
- `/lib` - место, где находятся библиотеки (общие программы, похожие на Windows DLL).

Мы будем подробнее работать с этими ключевыми каталогами на протяжении всей книги. Понимание этих каталогов первого уровня важно для навигации по файловой системе из командной строки.

Также важно заранее знать: не следует входить как `root` при выполнении обычных задач, потому что любой, кто взломает вашу систему (да, хакеров иногда тоже взламывают), пока вы вошли как `root`, сразу получит привилегии `root` и тем самым "завладеет" вашей системой. Входите как обычный пользователь, когда запускаете обычные приложения, просматриваете веб, используете инструменты вроде Wireshark и так далее.

Базовые команды в Linux

Для начала рассмотрим несколько базовых команд, которые помогут вам быстро приступить к работе в Linux.

Определение текущего местоположения с помощью pwd

В отличие от работы в среде с графическим пользовательским интерфейсом (GUI), такой как Windows или macOS, командная строка Linux не всегда явно показывает, в каком каталоге вы сейчас находитесь. Чтобы перейти в новый каталог, обычно нужно знать, где вы находитесь в данный момент. Команда `pwd`, сокращение от "показать текущий рабочий каталог", возвращает ваше местоположение внутри структуры каталогов.

Введите pwd в терминале, чтобы увидеть, где вы находитесь:

```
kali >pwd
/root
```

В этом случае Linux вернул /root, сообщая, что я нахожусь в каталоге пользователя root. А поскольку при запуске Linux вы вошли как root, вы тоже должны находиться в каталоге пользователя root, который расположен на один уровень ниже верхушки структуры файловой системы (/).

Если вы находитесь в другом каталоге, pwd вернет вместо этого имя того каталога.

Проверка учетной записи с помощью whoami

В Linux единственный "всемогущий" суперпользователь, или системный администратор, называется root; у него есть все системные привилегии, необходимые для добавления пользователей, изменения паролей, изменения привилегий и так далее. Очевидно, вы не хотите, чтобы кто угодно мог вносить такие изменения: вам нужен тот, кому можно доверять и кто хорошо знает операционную систему. Как хакеру вам обычно нужны все эти привилегии для запуска нужных программ и команд (многие хакерские инструменты не будут работать без привилегий root), поэтому вы захотите войти как root.

Если вы забыли, вошли ли вы как root или как другой пользователь, можно использовать команду whoami, чтобы увидеть, под каким пользователем вы вошли:

```
kali >whoami
root
```

Если бы я вошел как другой пользователь, например под своей личной учетной записью, whoami вернула бы вместо этого мое имя пользователя:

```
kali >whoami
OTW
```

Навигация по файловой системе Linux

Навигация по файловой системе из терминала - важнейший навык Linux. Чтобы что-то сделать, нужно уметь перемещаться и находить приложения, файлы и каталоги, расположенные в других каталогах. В системе на основе GUI каталоги видны визуально, но при использовании интерфейса командной строки структура полностью текстовая, и навигация по файловой системе означает использование команд.

Смена каталогов с помощью cd

Чтобы сменить каталог из терминала, используйте команду cd, сокращение от "сменить каталог". Например, вот как перейти в каталог /etc, используемый для хранения конфигурационных файлов:

```
kali >cd /etc
root@kali:/etc#
```

Приглашение меняется на `root@kali:/etc`, показывая, что мы находимся в каталоге `/etc`. Можно подтвердить это, введя `pwd`:

```
root@kali:/etc# pwd
/etc
```

Чтобы подняться на один уровень в файловой структуре (к корню файловой структуры, то есть `/`), используйте `cd`, за которым следуют две точки (`..`), как показано здесь:

```
root@kali:/etc# cd ..
root@kali:/# pwd
/
root@kali:/#
```

Это переносит нас на один уровень вверх, из `/etc` в корневой каталог `/`, но вы можете подниматься на столько уровней, сколько нужно. Просто используйте столько пар двойных точек, сколько уровней хотите пройти:

- `..` - подняться на один уровень;
- `../..` - подняться на два уровня;
- `../../..` - подняться на три уровня и так далее.

Например, чтобы подняться на два уровня, введите `cd`, а затем два набора двойных точек, разделенных косой чертой:

```
kali >cd ../..
```

Также можно перейти на корневой уровень файловой структуры из любого места, введя `cd /`, где `/` представляет корень файловой системы.

Просмотр содержимого каталога с помощью ls

Чтобы увидеть содержимое каталога (файлы и подкаталоги), можно использовать команду `ls` (сокращение от "список"). Она очень похожа на команду `dir` в Windows.

```
kali >ls
bin initrd.img media   run   var
boot initrd.img.old mnt sbin  vmlinuz
dev lib     opt    srv   vmlinuz.old
etc lib64    proc   tmp
home lost+found  root   usr
```

Эта команда перечисляет и файлы, и каталоги, содержащиеся в каталоге. Ее также можно использовать для любого конкретного каталога, а не только для текущего: укажите имя каталога после команды. Например, `ls /etc` показывает, что находится в каталоге `/etc`.

Чтобы получить больше информации о файлах и каталогах, например их права доступа, владельца, размер и время последнего изменения, можно добавить ключ `-l` после `ls` (`l` означает long). Это часто называют длинным листингом. Попробуем:

```
kali >ls -l
total 84
drw-r--r--    1    root     root    4096    Dec    5  11:15    bin
drw-r--r--    2    root     root    4096    Dec    5  11:15    boot
drw-r--r--    3    root     root    4096    Dec    9  13:10    dev
drw-r--r--    18   root     root    4096    Dec    9  13:43    etc
--snip--
drw-r--r--    1    root     root    4096    Dec    5  11:15    var
```

Как видите, `ls -l` дает значительно больше сведений: является ли объект файлом или каталогом, число ссылок, владелец, группа, размер, время создания или изменения и имя.

Я обычно добавляю ключ `-l` при любом листинге в Linux, но тут каждому свое. Подробнее о `ls -l` мы поговорим в главе 5.

Некоторые файлы в Linux скрыты и не будут показаны простой командой `ls` или `ls -l`. Чтобы показать скрытые файлы, добавьте ключ `-a` в нижнем регистре:

```
kali >ls -la
```

Если вы не видите файл, который ожидали увидеть, стоит попробовать `ls` с флагом `a`.

Получение справки

Почти у каждой команды, приложения или утилиты в Linux есть отдельный файл справки, который дает рекомендации по использованию. Например, если мне нужна помощь по лучшему инструменту взлома беспроводных сетей `aircrack-ng`, я могу просто набрать команду `aircrack-ng`, а затем ключ `--help`:

```
kali >aircrack-ng --help
```

Обратите внимание на двойной дефис. В Linux принято использовать двойной дефис (`--`) перед словесными параметрами, такими как `help`, и одиночный дефис (`-`) перед однобуквенными параметрами, такими как `-h`.

Когда вы введете эту команду, вы должны увидеть краткое описание инструмента и рекомендации по его использованию. В некоторых случаях для доступа к файлу справки можно использовать `-h` или `-?`. Например, если мне нужна помощь по лучшему инструменту хакера для сканирования портов, `nmap`, я введу следующее:

```
kali >nmap -h
```

К сожалению, хотя многие приложения поддерживают все три варианта (`--help`, `-h` и `-?`), нет гарантии, что выбранное приложение будет их поддерживать. Поэтому если один вариант не работает, попробуйте другой.

Обращение к страницам руководства с помощью man

Помимо ключа справки, у большинства команд и приложений есть страница руководства с дополнительной информацией, например описанием и краткой формой команды или приложения. Чтобы посмотреть страницу руководства, достаточно набрать `man` перед командой, утилитой или приложением. Например, чтобы увидеть страницу руководства для `aircrack-ng`, нужно ввести следующее:

```
kali >man aircrack-ng
NAME
       aircrack-ng - a 802.11 WEP / WPA-PSK key cracker
SYNOPSIS
       aircrack-ng [options] <.cap / .ivs file(s)>
DESCRIPTION
       aircrack-ng is an 802.11 WEP and WPA/WPA2-PSK key cracking program.
       It can recover the WEP key once enough encrypted packets have been
       captured with airodump-ng. This part of the aircrack-ng suite deter-
       mines the WEP key using two fundamental methods. The first method is
       via the PTW approach (Pyshkin, Tews, Weinmann). The main advantage
       of the PTW approach is that very few data packets are required to
       crack the WEP key. The second method is the FMS/KoreK method. The
       FMS/KoreK method incorporates various statistical attacks to dis-
       cover the WEP key and uses these in combination with brute forcing.
       Additionally, the program offers a dictionary method for determining
       the WEP key. For cracking WPA/WPA2 pre-shared keys, a wordlist (file
       or stdin) or an airolib-ng has to be used.
```

Это открывает руководство для `aircrack-ng` и дает более подробную информацию, чем экран справки. Прокручивать файл руководства можно клавишей `ENTER`, а перемещаться по страницам вверх и вниз - клавишами `PG DN` и `PG UP` соответственно. Чтобы выйти, просто введите `q`, и вы вернетесь к приглашению командной строки.

Поиск всякого

Пока вы не привыкнете к Linux, ориентироваться в нем может быть неприятно, но знание нескольких базовых команд и приемов во многом сделает командную строку дружелюбнее. Следующие команды помогают находить нужное из терминала.

Поиск с помощью locate

Наверное, самая простая команда для поиска - `locate`. Если после нее указать ключевое слово, обозначающее то, что вы хотите найти, команда пройдет по всей файловой системе и найдет каждое вхождение этого слова.

Например, чтобы найти `aircrack-ng`, введите следующее:

```
kali >locate aircrack-ng
/usr/bin/aircrack-ng
/usr/share/applications/kali-aircrack-ng.desktop
/usr/share/desktop-directories/05-1-01-aircrack-ng.directory
--snip--
/var/lib/dpkg/info/aircrack-ng.md5sums
```

Однако команда `locate` не идеальна. Иногда ее результаты могут быть избыточными и давать слишком много информации. Кроме того, `locate` использует базу данных, которая обычно обновляется только раз в день, поэтому если вы создали файл несколько минут или часов назад, он может не появиться в этом списке до следующего дня. Стоит знать недостатки этих базовых

команд, чтобы лучше решать, когда какую из них использовать.

Поиск бинарных файлов с помощью whereis

Если вы ищете бинарный файл, можно использовать команду `whereis`, чтобы найти его. Эта команда возвращает не только расположение бинарного файла, но также его исходный код и страницу руководства, если они доступны. Вот пример:

```
kali >whereis aircrack-ng
aircrack-ng: /usr/bin/aircrack-ng /usr/share/man/man1/aircrack-ng.1.gz
```

В этом случае `whereis` вернула только бинарные файлы `aircrack-ng` и страницу руководства, а не каждое вхождение слова `aircrack-ng`. Гораздо эффективнее и нагляднее, правда?

Поиск бинарных файлов в переменной PATH с помощью which

Команда `which` еще более конкретна: она возвращает только расположение бинарных файлов в переменной `PATH` Linux. Подробнее переменную `PATH` мы рассмотрим в главе 7, но сейчас достаточно знать, что `PATH` содержит каталоги, в которых операционная система ищет команды, выполняемые вами в командной строке. Например, когда я ввожу `aircrack-ng` в командной строке, операционная система обращается к переменной `PATH`, чтобы понять, в каких каталогах искать `aircrack-ng`:

```
kali >which aircrack-ng
/usr/bin/aircrack-ng
```

Здесь `which` смогла найти один бинарный файл в каталогах, перечисленных в переменной `PATH`. Как минимум эти каталоги обычно включают `/usr/bin`, но могут также включать `/usr/sbin` и, возможно, еще несколько.

Более мощный поиск с помощью find

Команда `find` - самая мощная и гибкая из поисковых утилит. Она может начинать поиск в любом указанном каталоге и искать по разным параметрам, включая, конечно, имя файла, но также дату создания или изменения, владельца, группу, права доступа и размер.

Базовый синтаксис `find` выглядит так:

```
find directory options expression
```

Итак, если бы я хотел найти файл с именем `apache2` (веб-сервер с открытым исходным кодом), начиная с корневого каталога, я ввел бы следующее:

```
kali >find / (1) -type f (2) -name apache2 (3)
```

Сначала я указываю каталог, в котором нужно начать поиск, в данном случае `/` (1). Затем задаю тип файла, который нужно искать, в данном случае `f` для обычного файла (2). Наконец, я указываю имя файла, который ищу, в данном случае `apache2` (3).

Результаты этого поиска показаны здесь:

```
kali >find / -type f -name apache2
/usr/lib/apache2/mpm-itk/apache2
/usr/lib/apache2/mpm-event/apache2
/usr/lib/apache2/mpm-worker/apache2
/usr/lib/apache2/mpm-prefork/apache2
/etc/cron.daily/apache2
/etc/logrotate.d/apache2
/etc/init.d/apache2
/etc/default/apache2
```

Команда `find` начала с вершины файловой системы (`/`), прошла по каждому каталогу, ища `apache2` в имени файла, а затем перечислила все найденные экземпляры.

Как можно представить, поиск по всем каталогам может быть медленным. Один способ ускорить его - искать только в каталоге, где вы ожидаете найти нужные файлы. В данном случае мы ищем конфигурационный файл, поэтому могли бы начать поиск в каталоге `/etc`, и Linux искал бы только в нем и его подкаталогах. Попробуем:

```
kali >find /etc -type f -name apache2
/etc/init.d/apache2
/etc/logrotate.d/apache2
/etc/cron.daily/apache2
```

Этот намного более быстрый поиск нашел вхождения `apache2` только в каталоге `/etc` и его подкаталогах. Также важно отметить: в отличие от некоторых других команд поиска, `find` показывает только точные совпадения имени. Если у файла `apache2` есть расширение, например `apache2.conf`, поиск не найдет совпадение. Это ограничение можно устранить с помощью подстановочных знаков, которые позволяют сопоставлять несколько символов. Подстановочные знаки бывают разных форм: `*`, `.`, `?` и `[]`.

Поищем в каталоге `/etc` все файлы, которые начинаются с `apache2` и имеют любое расширение. Для этого можно написать команду `find` со следующим подстановочным знаком:

```
kali >find /etc -type f --name apache2.*
/etc/apache2/apache2.conf
```

Когда мы выполняем эту команду, оказывается, что в каталоге `/etc` есть один файл, соответствующий шаблону `apache2.*`. Когда мы используем точку, за которой следует подстановочный знак `*`, терминал ищет любое расширение после имени файла `apache2`. Это может быть очень полезным приемом для поиска файлов, когда вы не знаете расширение.

Когда я выполняю эту команду, я нахожу два файла, которые начинаются с `apache2` в каталоге `/etc`, включая файл `apache2.conf`.

> **Краткий взгляд на подстановочные знаки.** Допустим, мы выполняем поиск в каталоге, где есть файлы `cat`, `hat`, `what` и `bat`. Подстановочный знак `?` используется для обозначения одного символа, поэтому поиск `?at` найдет `hat`, `cat` и `bat`, но не `what`, потому что в этом имени файла перед `at` стоят две буквы. Подстановочный знак `[]` используется для совпадения с символами внутри квадратных скобок. Например, поиск `[c,b]at` совпадет с `cat` и `bat`, но не с `hat` или `what`. Среди самых широко используемых подстановочных знаков - звездочка (`*`), которая совпадает с любыми символами любой длины, от нуля до неограниченного количества символов. Например, поиск `*at` найдет `cat`, `hat`, `what` и `bat`.

Фильтрация с помощью grep

Очень часто при работе в командной строке вам нужно искать определенное ключевое слово. Для этого можно использовать команду `grep` как фильтр для поиска ключевых слов.

Команду `grep` часто используют, когда вывод передается по каналу из одной команды в другую. Каналы я рассматриваю в главе 2, но пока достаточно сказать, что Linux (и Windows, если уж на то пошло) позволяет взять вывод одной команды и отправить его как ввод другой команды. Это называется передачей по каналу, и для этого используется команда | (клавиша | обычно находится над клавишей `ENTER` на клавиатуре).

Команда `ps` используется для отображения информации о процессах, выполняющихся на машине. Подробнее мы рассмотрим ее в главе 6, но для этого примера предположим, что я хочу увидеть все процессы, выполняющиеся в моей системе Linux. В этом случае я могу использовать команду `ps`, за которой следуют ключи `aux`, уточняющие, какую информацию о процессах показать:

```
kali >ps aux
```

Это дает мне список всех процессов, выполняющихся в системе. Но что, если я хочу найти только один процесс, чтобы проверить, запущен ли он?

Я могу сделать это, передав вывод `ps` в `grep` и выполнив поиск по ключевому слову. Например, чтобы узнать, запущена ли служба `apache2`, я ввел бы следующее:

```
kali >ps aux | grep apache2
root 4851 0.2 0.7 37548 7668 ? Ss 10:14 0:00 /usr/sbin/apache2 -k start
root 4906 0.0 0.4 37572 4228 ? S 10:14 0:00 /usr/sbin/apache2 -k start
root 4910 0.0 0.4 37572 4228 ? Ss 10:14 0:00 /usr/sbin/apache2 -k start
--snip--
```

Эта команда говорит Linux показать все мои службы, а затем отправить этот вывод в `grep`, который просмотрит вывод в поисках ключевого слова `apache2` и покажет только соответствующий вывод, сэкономив мне немало времени и зрения.

Изменение файлов и каталогов

Когда вы нашли файлы и каталоги, нужно уметь выполнять с ними действия. В этом разделе мы рассмотрим, как создавать файлы и каталоги, копировать файлы, переименовывать файлы и удалять файлы и каталоги.

Создание файлов

В Linux много способов создавать файлы, но сейчас мы рассмотрим только два простых метода. Первый - `cat`, сокращение от "соединять части вместе". Команда `cat` обычно используется для отображения содержимого файла, но ее также можно использовать для создания небольших файлов. Для создания больших файлов лучше вводить код в текстовом редакторе, таком как `vim`, `emacs`, `leafpad`, `gedit` или `kate`, а затем сохранять его как файл.

Конкатенация с помощью cat

Команда `cat`, за которой следует имя файла, покажет содержимое этого файла. Но чтобы создать файл, мы указываем после `cat` перенаправление, обозначаемое символом >, и имя файла, который хотим создать. Вот пример:

```
kali >cat > hackingskills
Хакинг - самый ценный набор навыков XXI века!
```

Когда вы нажмете `ENTER`, Linux перейдет в интерактивный режим и будет ждать, пока вы начнете вводить содержимое файла. Это может сбивать с толку, потому что приглашение исчезает, но если вы просто начнете печатать, все введенное попадет в файл (в данном случае `hackingskills`). Здесь я ввел `Хакинг - самый ценный набор навыков XXI века!`. Чтобы выйти и вернуться к приглашению, я нажимаю `CTRL-D`. Затем, когда я хочу увидеть, что находится в файле `hackingskills`, я ввожу следующее:

```
kali >cat hackingskills
Хакинг - самый ценный набор навыков XXI века!
```

Если вы не используете символ перенаправления, Linux выплюнет обратно содержимое вашего файла.

Чтобы добавить больше содержимого в конец файла, можно использовать команду `cat` с двойным перенаправлением (>>), после которого следует то, что вы хотите добавить. Вот пример:

```
kali >cat >> hackingskills
Каждому стоит изучать хакинг
```

Linux снова переходит в интерактивный режим и ждет содержимое, которое нужно добавить к файлу. Когда я ввожу `Каждому стоит изучать хакинг` и нажимаю `CTRL-D`, я возвращаюсь к приглашению. Теперь, когда я показываю содержимое этого файла с помощью `cat`, видно, что к файлу добавлена строка `Каждому стоит изучать хакинг`:

```
kali >cat hackingskills
Хакинг - самый ценный набор навыков XXI века! Каждому стоит
изучать хакинг
```

Если я хочу перезаписать файл новой информацией, я могу снова использовать команду `cat` с одиночным перенаправлением:

```
kali >cat > hackingskills
Специалист по IT-безопасности без навыков хакинга остается в темноте
kali >cat hackingskills
Специалист по IT-безопасности без навыков хакинга остается в темноте
```

Как видите, Linux переходит в интерактивный режим, я ввожу новый текст, а затем возвращаюсь к приглашению. Когда я снова использую `cat`, чтобы увидеть содержимое файла, я вижу, что мои прежние слова были перезаписаны последним текстом.

Создание файла с помощью touch

Вторая команда для создания файлов - `touch`. Изначально эта команда была разработана, чтобы пользователь мог просто "коснуться" файла и изменить некоторые его свойства, например дату создания или изменения. Однако если файл еще не существует, эта команда по умолчанию создает его.

Создадим newfile с помощью touch:

```
kali >touch newfile
```

Теперь, когда я использую ls -l, чтобы увидеть длинный список каталога, я вижу, что создан новый файл с именем newfile. Обратите внимание: его размер равен 0, потому что в newfile нет содержимого.

Создание каталога

Команда для создания каталога в Linux - mkdir, сокращение от "создать каталог". Чтобы создать каталог с именем newdirectory, введите следующую команду:

```
kali >mkdir newdirectory
```

Чтобы перейти в этот только что созданный каталог, просто введите:

```
kali >cd newdirectory
```

Копирование файла

Для копирования файлов используется команда cp. Она создает дубликат файла в новом расположении и оставляет старый файл на месте.

Здесь мы создадим файл oldfile в корневом каталоге с помощью touch и скопируем его в /root/newdirectory, переименовав в процессе и оставив исходный oldfile на месте:

```
kali >touch oldfile
kali >cp oldfile /root/newdirectory/newfile
```

Переименование файла необязательно и выполняется просто добавлением имени, которое вы хотите ему дать, в конец пути к каталогу. Если при копировании файл не переименовать, он по умолчанию сохранит исходное имя.

Когда затем мы переходим в newdirectory, мы видим, что там есть точная копия oldfile под названием newfile:

```
kali >cd newdirectory
kali >ls
newfile oldfile
```

Переименование файла

К сожалению, в Linux нет команды, предназначенной исключительно для переименования файла, как в Windows и некоторых других операционных системах, но есть команда mv.

Команду `mv` можно использовать, чтобы переместить файл или каталог в новое место или просто дать существующему файлу новое имя. Чтобы переименовать `newfile` в `newfile2`, нужно ввести следующее:

```
kali >mv newfile newfile2
kali >ls
oldfile newfile2
```

Теперь, когда вы выводите список (`ls`) этого каталога, вы видите `newfile2`, но не `newfile`, потому что файл был переименован. То же самое можно делать с каталогами.

Удаление файла

Чтобы удалить файл, можно просто использовать команду `rm`:

```
kali >rm newfile2
```

Если теперь выполнить длинный листинг каталога, можно подтвердить, что файл удален.

Удаление каталога

Команда для удаления каталога похожа на команду `rm` для удаления файлов, но к ней добавляется `dir` ("каталог"):

```
kali >rmdir newdirectory
rmdir: failed to remove 'newdirectory': Directory not empty
```

Важно отметить: `rmdir` не удалит каталог, который не пуст, а выдаст предупреждение, что каталог не пуст, как видно в этом примере. Перед удалением каталога нужно сначала удалить все его содержимое. Это сделано, чтобы вы случайно не удалили объекты, которые не собирались удалять.

Если вы действительно хотите удалить каталог и все его содержимое за один раз, можно использовать ключ `-r` после `rm`:

```
kali >rm -r newdirectory
```

Но небольшое предупреждение: будьте осторожны с использованием параметра `-r` вместе с `rm`, по крайней мере поначалу, потому что очень легко по ошибке удалить ценные файлы и каталоги. Например, использование `rm -r` в вашем домашнем каталоге удалило бы каждый файл и каталог там, что, вероятно, совсем не то, что вы намеревались сделать.

Идите и играйте!

Теперь, когда у вас есть базовые навыки навигации по файловой системе, можно немного поиграть со своей системой Linux, прежде чем двигаться дальше. Лучший способ освоиться с терминалом - прямо сейчас попробовать новые навыки. В следующих главах мы будем исследовать нашу хакерскую игровую площадку дальше и глубже.

Упражнения

Прежде чем переходить к главе 2, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Используйте команду `ls` из корневого каталога (`/`), чтобы изучить структуру каталогов Linux. Перейдите в каждый из каталогов командой `cd` и выполните `pwd`, чтобы проверить, где вы находитесь в структуре каталогов.
2. Используйте команду `whoami`, чтобы проверить, под каким пользователем вы вошли.
3. Используйте команду `locate`, чтобы найти списки слов, которые можно использовать для взлома паролей.
4. Используйте команду `cat`, чтобы создать новый файл, а затем добавить данные в этот файл. Помните, что > перенаправляет ввод в файл, а >> добавляет данные в файл.
5. Создайте новый каталог `hackerdirectory` и создайте в нем новый файл с именем `hackedfile`. Теперь скопируйте этот файл в каталог `/root` и переименуйте его в `secretfile`.

Глава 2. Манипуляция текстом

В Linux почти все, с чем вы работаете напрямую, является файлом, и чаще всего это текстовые файлы. Например, все конфигурационные файлы в Linux являются текстовыми. Поэтому, чтобы перенастроить приложение, вы просто открываете конфигурационный файл, меняете текст, сохраняете файл, а затем перезапускаете приложение - перенастройка завершена.

При таком количестве текстовых файлов манипуляция текстом становится критически важной для управления Linux и приложениями Linux. В этой главе вы будете использовать несколько команд и приемов для манипуляции текстом в Linux.

Для иллюстрации я буду использовать файлы лучшей в мире системы обнаружения сетевых вторжений (NIDS) Snort, которая была впервые разработана Марти Роешем и теперь принадлежит Cisco. NIDS обычно применяют для обнаружения вторжений хакеров, поэтому если вы хотите стать успешным хакером, вы должны знать, как NIDS может сдерживать атаки и как ею можно злоупотреблять, чтобы избежать обнаружения.

> **Примечание.** Если версия Kali Linux, которую вы используете, не поставляется с предустановленным Snort, вы можете загрузить файлы из репозитория Kali, введя `apt-get install snort`.

Просмотр файлов

Как было показано в главе 1, самая базовая команда отображения текста, вероятно, `cat`, но у нее есть ограничения. Используйте `cat`, чтобы показать конфигурационный файл Snort (`snort.conf`), находящийся в `/etc/snort` (см. листинг 2-1).

```
kali >cat /etc/snort/snort.conf
```

Листинг 2-1: Отображение `snort.conf` в окне терминала

Теперь на экране должен появиться весь файл `snort.conf`, который будет прокручиваться, пока не дойдет до конца файла, как показано здесь. Это не самый удобный или практичный способ просматривать этот файл и работать с ним.

```
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/exploit.rules
--snip--
# команды порогов событий или подавления...
kali >
```

В следующих двух разделах я покажу вам команды `head` и `tail` - два способа вывести только часть содержимого файла, чтобы легче просматривать ключевое содержимое.

Берем голову

Если вы хотите посмотреть только начало файла, можно использовать команду head. По умолчанию эта команда отображает первые 10 строк файла. Например, следующая команда показывает первые 10 строк snort.conf:

```
kali >head /etc/snort/snort.conf
#---------------------------------------------------------------
# Пакеты правил VRT Snort.conf
#
# Подробнее см. здесь:
--snip--
#Ошибки Snort:bugs@snort.org
```

Если вы хотите увидеть больше или меньше стандартных 10 строк, укажите нужное количество с ключом дефиса (-) после вызова head и перед именем файла. Например, если вы хотите увидеть первые 20 строк файла, нужно ввести команду, показанную в начале листинга 2-2.

```
kali >head -20 /etc/snort/snort.conf
#-----------------------------------------------
#Пакеты правил VRT Snort.conf
#
#Подробнее см. здесь:
#.
#.
#.
#Параметры : --enable-gre --enable-mpls --enable-targetbased
--enable-ppm --enable-perfprofiling enable-zlib --enable-act
live-response --enable-normalizer --enable-reload --enable-react
```

Листинг 2-2: Отображение первых 20 строк snort.conf в окне терминала

В окне терминала должны быть показаны только первые 20 строк snort.conf.

Хватаем хвост

Команда tail похожа на head, но используется для просмотра последних строк файла. Используем ее для snort.conf:

```
kali >tail /etc/snort/snort.conf
#include $SO_RULE_PATH/smtp.rules
#include $SO_RULE_PATH/specific-threats.rules
#include $SO_RULE_PATH/web-activex.rules
#include $SO_RULE_PATH/web-client.rules
#include $SO_RULE_PATH/web-iis.rules
#include $SO_RULE_PATH/web-miscp.rules
#Команды порогов событий и подавления. См. threshold.conf
```

Обратите внимание: эта команда отображает несколько последних строк include файлов правил, но не все, потому что, как и у head, по умолчанию tail показывает 10 строк. Можно вывести больше строк, взяв последние 20 строк snort.conf. Как и с командой head, можно сообщить tail, сколько строк отображать, введя дефис (-) и затем число строк между командой и именем файла, как показано в листинге 2-3.

```
kali >tail -20 /etc/snort/snort.conf
#include $SO_RULE_PATH/chat.rules
#include $SO_RULE_PATH/chat.rules
#include $SO_RULE_PATH/chat.rules
--snip--
#Команды порогов событий или подавления. См. theshold.conf
```

Листинг 2-3: Отображение последних 20 строк snort.conf в окне терминала

Теперь мы можем увидеть почти все строки include файлов правил на одном экране.

Нумерация строк

Иногда, особенно с очень длинными файлами, нам нужно отображать файл с номерами строк. Поскольку в `snort.conf` больше 600 строк, номера строк здесь были бы полезны. Так проще ссылаться на изменения и возвращаться к одному и тому же месту в файле.

Чтобы показать файл с номерами строк, используется команда `nl`. Просто введите команду, показанную в листинге 2-4.

```
kali >nl /etc/snort/snort.conf
612
###################################################################
613 #dynamic library rules
614 #include $SO_RULE_PATH/bad-traffic.rules
615 #include $SO_RULE_PATH/chat.rules
--snip--
630 #include $SO_RULE_PATH/web-iis.rules
631 #include $SO_RULE_PATH/web-misc.rules
```

Листинг 2-4: Отображение номеров строк в выводе терминала

Теперь у каждой строки есть номер, что значительно упрощает ссылки.

Фильтрация текста с помощью grep

Команда `grep`, вероятно, самая широко используемая команда манипуляции текстом. Она позволяет фильтровать содержимое файла для отображения. Например, если вы хотите увидеть все строки, содержащие слово `output` в файле `snort.conf`, можно использовать `cat` и попросить его показать только эти строки (см. листинг 2-5).

```
kali >cat /etc/snort/snort.conf | grep output
# 6) Настройка плагинов вывода
# Шаг #6: Настройка плагинов вывода
# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types,
vlan_event_types
output unified2: filename merged.log, limit 128, nostamp, mpls_event_types,
vlan_event_types
# output alert_unified2: filename merged.log, limit 128, nostamp
# output log_unified2: filename merged.log, limit 128, nostamp
# output alert_syslog: LOG_AUTH LOG_ALERT
# output log_tcpdump: tcpdump.log
```

Листинг 2-5: Отображение строк с вхождениями ключевого слова или фразы, указанной `grep`

Эта команда сначала просмотрит `snort.conf`, а затем с помощью канала (`|`) отправит его в `grep`; `grep` примет файл как ввод, найдет строки с вхождениями слова `output` и покажет только эти строки. Команда `grep` очень мощная и важная для работы в Linux, потому что она может сэкономить часы поиска каждого вхождения слова или команды в файле.

Хакерская задача: использование grep, nl, tail и head

Допустим, вы хотите вывести пять строк непосредственно перед строкой `# Шаг #6: Настройка плагинов вывода`, используя как минимум четыре команды, которые только что изучили. Как бы вы это сделали? Подсказка: у этих команд есть гораздо больше параметров, чем мы обсудили. Узнать больше можно с помощью встроенной команды Linux `man`. Например, `man tail` покажет файл справки для команды `tail`.

У этой задачи много решений; здесь я показываю, какие строки изменить для одного из способов, а ваша задача - найти другой метод.

Шаг 1

```
kali >nl /etc/snort/snort.conf | grep output
34 # 6) Настройка плагинов вывода
512 # Шаг #6: Настройка плагинов вывода
518 # output unified2: filename merged.log, limit 128, nostamp, mpls_event_types,
vlan_event_types
521 # output alert_unified2: filename snort.alert, limit 128, nostamp
522 # output log_unified2: filename snort.log, limit 128, nostamp
525 # output alert_syslog: LOG_AUTH LOG_ALERT
528 # output log_tcpdump: tcpdump.log
```

Мы видим, что строка `# Шаг #6: Настройка плагинов вывода` - это строка 512, и знаем, что нам нужны пять строк перед строкой 512, а также сама строка 512 (то есть строки 507-512).

Шаг 2

```
kali >tail -n +507 /etc/snort/snort.conf | head -n 6
nested_ip inner, \
whitelist $WHITE_LIST_PATH/white_list.rules, \
blacklist $BLACK_LIST_PATH/black_list.rules
##################################################
# Шаг #6: Настройка плагинов вывода
```

Здесь мы используем `tail`, чтобы начать со строки 507, затем передаем вывод в `head` и возвращаем только верхние шесть строк, получая пять строк перед строкой шага #6 вместе с самой этой строкой.

Использование sed для поиска и замены

Команда `sed` позволяет искать вхождения слова или текстового шаблона, а затем выполнять с ним некоторое действие. Название команды - сокращение от "потоковый редактор", потому что она следует той же концепции. В самой базовой форме `sed` работает как функция поиска и замены в Windows.

Найдите слово mysql в файле snort.conf с помощью grep:

```
kali >cat /etc/snort/snort.conf | grep mysql
include $RULE_PATH/mysql.rules
#include $RULE_PATH/server-mysql.rules
```

Вы должны увидеть, что команда grep нашла два вхождения mysql.

Допустим, вы хотите, чтобы sed заменил каждое вхождение mysql на MySQL (помните, Linux чувствителен к регистру), а затем сохранил новый файл как snort2.conf. Это можно сделать, введя команду из листинга 2-6.

```
kali >sed s/mysql/MySQL/g /etc/snort/snort.conf > snort2.conf
```

Листинг 2-6: Использование sed для поиска и замены ключевых слов или фраз

Команда s выполняет поиск: сначала вы задаете термин, который ищете (mysql), затем термин, которым хотите его заменить (MySQL), разделяя их косой чертой (/). Команда g сообщает Linux, что замену нужно выполнить глобально. Затем результат сохраняется в новый файл с именем snort2.conf.

Теперь, когда вы используете grep с snort2.conf для поиска mysql, вы увидите, что вхождений не найдено, но при поиске MySQL увидите два вхождения.

```
kali >cat snort2.conf | grep MySQL
include $RULE_PATH/MySQL.rules
#include $RULE_PATH/server-MySQL.rules
```

Если бы вы хотели заменить только первое вхождение термина mysql, нужно было бы убрать завершающую команду g.

```
kali >sed s/mysql/MySQL/ snort.conf > snort2.conf
```

Также можно использовать команду sed, чтобы найти и заменить конкретное по порядку вхождение слова, а не все вхождения и не только первое. Например, если вы хотите заменить только второе вхождение слова mysql, просто поместите номер вхождения (в данном случае 2) в конец команды:

```
kali >sed s/mysql/MySQL/2 snort.conf > snort2.conf
```

Эта команда воздействует только на второе вхождение mysql.

Просмотр файлов с помощью more и less

Хотя cat - хорошая утилита для отображения файлов и создания небольших файлов, у нее явно есть ограничения при отображении больших файлов. Когда вы используете cat с snort.conf, файл прокручивается через все страницы до конца, что не очень практично, если вы хотите извлечь из него какую-либо информацию.

Для работы с более крупными файлами у нас есть две другие утилиты просмотра: more и less.

Управление отображением с помощью more

Команда more отображает файл по одной странице за раз и позволяет перелистывать его вниз клавишей ENTER. Именно эту утилиту используют страницы man, поэтому рассмотрим ее первой. Откройте snort.conf командой more, как показано в листинге 2-7.

```
kali >more /etc/snort/snort.conf
--snip--
# Параметры сборки Snort:
```

```
# Параметры: --enable-gre --enable-mpls --enable-targetbased
--enable-ppm --enable-perfprofiling enable-zlib --enable-active
-response --enable-normalizer --enable-reload --enable-react
--enable-flexresp3
#
--More--(2%)
```

Листинг 2-7: Использование `more` для постраничного отображения вывода терминала

Обратите внимание: `more` показывает только первую страницу, затем останавливается и сообщает в левом нижнем углу, какая часть файла показана (в данном случае 2 процента). Чтобы увидеть дополнительные строки или страницы, нажмите `ENTER`. Чтобы выйти из `more`, введите `q`.

Отображение и фильтрация с помощью less

Команда `less` очень похожа на `more`, но имеет дополнительные возможности; отсюда распространенная шутка поклонников Linux: "`less` оказывается больше, чем `more`". С `less` можно не только прокручивать файл в своем темпе, но и фильтровать его по терминам. Как в листинге 2-8, откройте `snort.conf` с помощью `less`.

```
kali >less /etc/snort/snort.conf
--snip--
# Параметры сборки Snort:
# Параметры: --enable-gre --enable-mpls --enable-targetbased
--enable-ppm --enable-perfprofiling enable-zlib --enable-active
-response --enable-normalizer --enable-reload --enable-react
/etc/snort/snort.conf
```

Листинг 2-8: Использование `less` для постраничного отображения вывода терминала и фильтрации результатов

Обратите внимание: в левом нижнем углу экрана `less` выделил путь к файлу. Если нажать клавишу косой черты (`/`), `less` позволит искать термины в файле. Например, при первой настройке Snort нужно определить, как и куда отправлять вывод предупреждений о вторжениях. Чтобы найти этот раздел конфигурационного файла, можно просто выполнить поиск по слову `output`:

```
# Параметры сборки Snort:
# Параметры: --enable-gre --enable-mpls --enable-targetbased
--enable-ppm --enable-perfprofiling enable-zlib --enable-active
-response --enable-normalizer --enable-reload --enable-react
/output
```

Это сразу перенесет вас к первому вхождению `output` и выделит его. Затем можно искать следующее вхождение `output`, набрав `n`.

```
# Шаг #6: Настройка плагинов вывода
# Подробнее см. руководство Snort, настройка Snort - плагины вывода
##################################################################
#unified2
# Рекомендуется для большинства установок
# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types,
vlan_event_types
output unified2: filename snort.log, limit 128, nostamp, mpls_event_types,
vlan_event_types
# Дополнительная конфигурация для отдельных типов установок
# output alert_unified2: filename snort.alert, limit 128, nostamp
# output log_unified2: filename snort.log, limit 128, nostamp
# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT
:
```

Как видите, `less` перенес вас к следующему вхождению слова `output` и выделил все поисковые термины. В данном случае он перешел прямо к разделу вывода Snort. Как удобно!

Итоги

В Linux есть множество способов манипулировать текстом, и у каждого способа есть свои сильные и слабые стороны. В этой главе мы коснулись нескольких самых полезных методов, но я предлагаю вам попробовать каждый и выработать собственные ощущения и предпочтения. Например, я считаю `grep` незаменимым и широко использую `less`, но вы можете думать иначе.

Упражнения

Прежде чем переходить к главе 3, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Перейдите в `/usr/share/wordlists/metasploit`. Это каталог с несколькими списками слов, которые можно использовать для перебора паролей на разных защищенных паролем устройствах с помощью Metasploit, самого популярного фреймворка для пентестинга и хакинга.
2. Используйте команду `cat`, чтобы просмотреть содержимое файла `passwords.lst`.
3. Используйте команду `more`, чтобы отобразить файл `passwords.lst`.
4. Используйте команду `less`, чтобы просмотреть файл `passwords.lst`.
5. Теперь используйте команду `nl`, чтобы поставить номера строк перед паролями в `passwords.lst`. В нем должно быть 88 396 паролей.
6. Используйте команду `tail`, чтобы увидеть последние 20 паролей в `passwords.lst`.
7. Используйте команду `cat`, чтобы отобразить `passwords.lst`, и передайте вывод по каналу, чтобы найти все пароли, содержащие `123`.

Глава 3. Анализ и управление сетями

Понимание сетей критически важно для любого начинающего хакера. Во многих ситуациях вы будете взламывать что-то через сеть, а хороший хакер должен знать, как подключаться к этой сети и взаимодействовать с ней. Например, вам может понадобиться подключиться к компьютеру так, чтобы ваш адрес интернет-протокола (IP) был скрыт от просмотра, или может понадобиться перенаправить запросы системы доменных имен (DNS) цели на свою систему; такие задачи относительно просты, но требуют небольшого знания сетей Linux. Эта глава покажет вам несколько важнейших инструментов Linux для анализа и управления сетями во время ваших приключений в сетевом хакинге.

Анализ сетей с помощью ifconfig

Команда `ifconfig` - один из самых базовых инструментов для изучения активных сетевых интерфейсов и взаимодействия с ними. Ее можно использовать для запроса активных сетевых подключений, просто введя `ifconfig` в терминале. Попробуйте сами, и вы должны увидеть вывод, похожий на листинг 3-1.

```
kali >ifconfig
(1) eth0 Link encap:Ethernet HWaddr 00:0c:29:ba:82:0f
(2) inet addr:192.168.181.131 (3) Bcast:192.168.181.255 (4) Mask:255.255.255.0
--snip--
(5) lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
--snip--
(6) wlan0 Link encap:Ethernet HWaddr 00:c0:ca:3f:ee:02
```

Листинг 3-1: Использование `ifconfig` для получения сетевой информации

Как видите, команда `ifconfig` показывает полезные сведения об активных сетевых интерфейсах системы. В верхней части вывода находится имя первого обнаруженного интерфейса, `eth0` (1), что означает Ethernet0 (Linux начинает счет с 0, а не с 1). Это первое проводное сетевое подключение. Если бы проводных Ethernet-интерфейсов было больше, они появились бы в выводе в том же формате (`eth1`, `eth2` и так далее).

Далее указан тип используемой сети (Ethernet), за ним `HWaddr` и адрес; это глобально уникальный адрес, прошитый в каждом сетевом устройстве. В данном случае это сетевая интерфейсная карта (NIC), обычно называемая адресом управления доступом к среде (MAC).

Вторая строка содержит информацию об IP-адресе, сейчас назначенном этому сетевому интерфейсу (в данном случае `192.168.181.131` (2)); `Bcast` (3), или широковещательном адресе, который используется для отправки информации всем IP в подсети; и, наконец, сетевой маске (`Mask` (4)), которая используется для определения того, какая часть IP-адреса связана с локальной сетью. В этом разделе вывода вы также найдете более технические сведения, но они выходят за рамки этой главы по основам сетей Linux.

Следующий раздел вывода показывает еще одно сетевое подключение под названием `lo` (5), сокращение от "петлевой адрес"; иногда его называют локальным хостом. Это специальный программный адрес, который подключает вас к собственной системе. Программы и службы, не работающие на вашей системе, не могут его использовать. `lo` применяется для тестирования чего-либо на вашей системе, например собственного веб-сервера. Обычно локальный хост представлен IP-адресом `127.0.0.1`.

Третье подключение - интерфейс `wlan0` (6). Он появляется только при наличии беспроводного интерфейса или адаптера, как у меня здесь. Обратите внимание, что он также показывает MAC-адрес этого устройства (`HWaddr`).

Эта информация из `ifconfig` позволяет подключаться к настройкам локальной сети (LAN) и манипулировать ими, а это важный навык для хакинга.

Проверка беспроводных сетевых устройств с помощью iwconfig

Если у вас есть беспроводной адаптер, можно использовать команду `iwconfig`, чтобы собрать важные для беспроводного хакинга сведения: IP-адрес адаптера, его MAC-адрес, режим работы и многое другое. Информация, которую можно получить этой командой, особенно важна при использовании инструментов беспроводного хакинга вроде `aircrack-ng`.

В терминале посмотрим на беспроводные устройства с помощью `iwconfig` (см. листинг 3-2).

```
kali >iwconfig
wlan0 IEEE 802.11bg ESSID:off/any
Mode:Managed Access Point: Not Associated Tx-Power=20 dBm
--snip--
lo no wireless extensions
eth0 no wireless extensions
```

Листинг 3-2: Использование `iwconfig` для получения информации о беспроводных адаптерах

Этот вывод сообщает, что единственный сетевой интерфейс с беспроводными расширениями - `wlan0`, чего мы и ожидали. Ни у `lo`, ни у `eth0` беспроводных расширений нет.

Для `wlan0` мы узнаем, какие беспроводные стандарты IEEE 802.11 поддерживает устройство: `b` и `g`, два ранних стандарта беспроводной связи. Сейчас большинство беспроводных устройств также включает `n` (`n` - самый новый стандарт).

Из `iwconfig` мы также узнаем режим беспроводного расширения (в данном случае `Mode:Managed`, в отличие от режима мониторинга или неразборчивого режима). Для взлома беспроводных паролей нам понадобится неразборчивый режим.

Далее видно, что беспроводной адаптер не подключен (`Not Associated`) к точке доступа (AP), а его мощность равна `20 dBm`, что представляет силу сигнала. В главе 14 мы проведем больше времени с этой информацией.

Изменение сетевой информации

Умение менять свой IP-адрес и другую сетевую информацию - полезный навык, потому что он поможет получать доступ к другим сетям, выглядя как доверенное устройство в этих сетях. Например, при атаке отказа в обслуживании (DoS) можно подделать свой IP так, чтобы казалось, будто атака идет из другого источника; это помогает избежать фиксации IP во время криминалистического анализа. В Linux это относительно простая задача, и выполняется она командой `ifconfig`.

Изменение IP-адреса

Чтобы изменить IP-адрес, введите `ifconfig`, затем интерфейс, который хотите переназначить, и новый IP-адрес, который хотите назначить этому интерфейсу. Например, чтобы назначить IP-адрес `192.168.181.115` интерфейсу `eth0`, нужно ввести следующее:

```
kali >ifconfig eth0 192.168.181.115
kali >
```

Если все сделано правильно, Linux просто вернет приглашение командной строки и ничего не скажет. Это хорошо!

Затем, когда вы снова проверите сетевые подключения с помощью `ifconfig`, вы должны увидеть, что IP-адрес изменился на только что назначенный новый IP-адрес.

Изменение сетевой маски и широковещательного адреса

Сетевую маску (`netmask`) и широковещательный адрес также можно изменить командой `ifconfig`. Например, если вы хотите назначить тому же интерфейсу `eth0` сетевую маску `255.255.0.0` и широковещательный адрес `192.168.1.255`, введите следующее:

```
kali >ifconfig eth0 192.168.181.115 netmask 255.255.0.0 broadcast
192.168.1.255
kali >
```

И снова, если все сделано правильно, Linux отвечает новым приглашением командной строки. Теперь снова введите `ifconfig`, чтобы убедиться, что каждый параметр изменился соответствующим образом.

Подделка MAC-адреса

Также можно использовать `ifconfig`, чтобы изменить свой MAC-адрес (или `HWaddr`). MAC-адрес глобально уникален и часто используется как мера безопасности, чтобы не пускать хакеров в сети или отслеживать их. Изменение MAC-адреса с подделкой другого MAC-адреса почти тривиально и нейтрализует эти меры безопасности. Поэтому это очень полезный прием для обхода контроля доступа к сети.

Чтобы подделать MAC-адрес, сначала используйте параметр `down` команды `ifconfig`, чтобы отключить интерфейс (в данном случае `eth0`). Затем введите команду `ifconfig`, за которой следуют имя интерфейса (`hw` для аппаратного адреса, `ether` для Ethernet) и новый поддельный MAC-адрес. Наконец, снова включите интерфейс параметром `up`, чтобы изменение вступило в силу. Вот пример:

```
kali >ifconfig eth0 down
kali >ifconfig eth0 hw ether 00:11:22:33:44:55
kali >ifconfig eth0 up
```

Теперь, когда вы проверите настройки с помощью `ifconfig`, вы должны увидеть, что `HWaddr` изменился на ваш новый поддельный IP-адрес!

Назначение новых IP-адресов с DHCP-сервера

В Linux есть сервер протокола динамической настройки узла (DHCP), который запускает демон - процесс, работающий в фоне, - под названием `dhcpd`, или DHCP-демон. DHCP-сервер назначает IP-адреса всем системам в подсети и ведет журналы того, какой IP-адрес какой машине выделен в каждый момент времени. Поэтому после атаки это отличный ресурс для криминалистов, позволяющий отследить хакеров. По этой причине полезно понимать, как работает DHCP-сервер.

Обычно для подключения к интернету из LAN нужно иметь IP, назначенный DHCP. Поэтому после установки статического IP-адреса нужно вернуться и получить новый IP-адрес, назначенный DHCP. Для этого всегда можно перезагрузить систему, но я покажу, как получить новый DHCP-адрес без выключения и перезапуска системы.

Чтобы запросить IP-адрес у DHCP, просто вызовите DHCP-сервер командой `dhclient`, за которой следует интерфейс, которому нужно назначить адрес. Разные дистрибутивы Linux используют разные DHCP-клиенты, но Kali построен на Debian, который использует `dhclient`. Поэтому назначить новый адрес можно так:

```
kali >dhclient eth0
```

Команда `dhclient` отправляет запрос `DHCPDISCOVER` с указанного сетевого интерфейса (здесь `eth0`). Затем она получает предложение (`DHCPOFFER`) от DHCP-сервера (в данном случае `192.168.181.131`) и подтверждает назначение IP DHCP-серверу с помощью DHCP-запроса.

```
kali >ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:ba:82:0f
inet addr:192.168.181.131 Bcast:192.168.181.131 Mask:255.255.255.0
```

В зависимости от конфигурации DHCP-сервера назначаемый IP-адрес в каждом случае может отличаться.

Теперь, когда вы вводите `ifconfig`, вы должны увидеть, что DHCP-сервер назначил новый IP-адрес, новый широковещательный адрес и новую сетевую маску вашему сетевому интерфейсу `eth0`.

Манипуляция системой доменных имен

Хакеры могут найти кладезь информации о цели в ее системе доменных имен (DNS). DNS - критический компонент интернета, и хотя она предназначена для преобразования доменных имен в IP-адреса, хакер может использовать ее для сбора информации о цели.

Изучение DNS с помощью dig

DNS - это служба, которая преобразует доменное имя вроде `hackers-arise.com` в соответствующий IP-адрес, чтобы ваша система знала, как до него добраться. Без DNS нам всем пришлось бы помнить тысячи IP-адресов любимых сайтов - немалая задача даже для человека с выдающейся памятью.

Одна из самых полезных команд для начинающего хакера - `dig`, которая дает способ собрать DNS-информацию о целевом домене. Сохраненная DNS-информация может быть ключевым фрагментом ранней разведки, который нужно получить перед атакой. Такая информация может включать IP-адрес сервера имен цели (сервера, который преобразует имя цели в IP-адрес), почтовый сервер цели и потенциально любые поддомены и IP-адреса.

Например, введите `dig hackers-arise.com` и добавьте параметр `ns` (сокращение от "сервер имен"). Сервер имен для `hackers-arise.com` отображается в `ANSWER SECTION` листинга 3-3.

```
kali >dig hackers-arise.com ns
--snip--
;; QUESTION SECTION:
;hackers-arise.com. IN NS
;; ANSWER SECTION:
hackers-arise.com. 5 IN NS ns7.wixdns.net.
hackers-arise.com. 5 IN NS ns6.wixdns.net.
;; ADDITIONAL SECTION:
ns6.wixdns.net. 5 IN A 216.239.32.100
--snip--
```

Листинг 3-3: Использование `dig` и его параметра `ns` для получения информации о сервере имен домена

Также обратите внимание: в `ADDITIONAL SECTION` этот запрос `dig` раскрывает IP-адрес (`216.239.32.100`) DNS-сервера, обслуживающего `hackers-arise.com`.

Команду `dig` также можно использовать для получения информации о почтовых серверах, связанных с доменом, добавив параметр `mx` (`mx` - сокращение от "сервер почтового обмена"). Эта информация критична для атак на почтовые системы. Например, сведения о почтовых серверах `www.hackers-arise.com` показаны в `AUTHORITY SECTION` листинга 3-4.

```
kali >dig hackers-arise.com mx
--snip--
;; QUESTION SECTION:
;hackers-arise.com. IN MX
;; AUTHORITY SECTION:
hackers-arise.com. 5 IN SOA ns6.wixdns.net. support.wix.com 2016052216 10800
3600 604 800 3600
--snip--
```

Листинг 3-4: Использование `dig` и его параметра `mx` для получения информации о сервере почтового обмена домена

Самый распространенный DNS-сервер Linux - доменная система имен Беркли (BIND). В некоторых случаях пользователи Linux будут называть DNS словом BIND, но не путайтесь: и DNS, и BIND сопоставляют отдельные доменные имена с IP-адресами.

Изменение DNS-сервера

В некоторых случаях может понадобиться использовать другой DNS-сервер. Для этого нужно отредактировать простой текстовый файл `/etc/resolv.conf` в системе. Откройте этот файл в текстовом редакторе; я использую Leafpad. Затем в командной строке введите точное имя редактора, за которым следуют расположение файла и имя файла. Например:

```
kali >leafpad /etc/resolv.conf
```

Эта команда откроет файл `resolv.conf` в каталоге `/etc` в указанном мной графическом текстовом редакторе Leafpad. Файл должен выглядеть примерно как на рис. 3-1.

Рис. 3-1: Типичный файл `resolv.conf` в текстовом редакторе

Как видно в строке 3, мой сервер имен установлен на локальный DNS-сервер по адресу `192.168.181.2`. Это работает нормально, но если я хочу добавить или заменить этот DNS-сервер, скажем, публичным DNS-сервером Google по адресу `8.8.8.8`, я добавил бы следующую строку в файл `/etc/resolv.conf`, чтобы указать сервер имен:

```
nameserver 8.8.8.8
```

Затем мне нужно было бы просто сохранить файл. Однако того же результата можно добиться исключительно из командной строки, введя следующее:

```
kali >echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

Эта команда выводит строку `nameserver 8.8.8.8` и перенаправляет ее (`>`) в файл `/etc/resolv.conf`, заменяя текущее содержимое. Теперь ваш файл `/etc/resolv.conf` должен выглядеть как на рис. 3-2.

Рис. 3-2: Изменение файла `resolv.conf` для указания DNS-сервера Google

Если теперь открыть файл `/etc/resolv.conf`, вы должны увидеть, что он направляет DNS-запросы к DNS-серверу Google, а не к вашему локальному DNS-серверу. Теперь ваша система будет обращаться к публичному DNS-серверу Google, чтобы преобразовывать доменные имена в IP-адреса. Это может означать, что доменные имена будут разрешаться чуть дольше (вероятно, на миллисекунды). Поэтому, чтобы сохранить скорость, но оставить возможность использования публичного сервера, можно оставить локальный DNS-сервер в файле `resolv.conf` и после него указать публичный DNS-сервер. Операционная система опрашивает каждый DNS-сервер, перечисленный в `/etc/resolv.conf`, в порядке появления, поэтому система обратится к публичному DNS-серверу только в том случае, если доменное имя не найдено на

локальном DNS-сервере.

> **Примечание.** Если вы используете DHCP-адрес и DHCP-сервер предоставляет настройку DNS, DHCP-сервер заменит содержимое файла при обновлении DHCP-адреса.

Сопоставление собственных IP-адресов

Специальный файл в вашей системе, называемый файлом hosts, также выполняет преобразование доменного имени в IP-адрес. Файл `hosts` находится в `/etc/hosts`, и, в некотором смысле как с DNS, его можно использовать для указания собственного сопоставления IP-адреса и доменного имени. Другими словами, вы можете определить, на какой IP-адрес перейдет ваш браузер, когда вы введете `www.microsoft.com` (или любой другой домен), вместо того чтобы позволять DNS-серверу решать это. Как хакеру, это может быть полезно для перехвата TCP-соединения в вашей локальной сети, чтобы направить трафик на вредоносный веб-сервер с помощью инструмента вроде `dnsspoof`.

В командной строке введите следующую команду (вы можете заменить `leafpad` предпочитаемым текстовым редактором):

```
kali >leafpad /etc/hosts
```

Теперь вы должны увидеть свой файл `hosts`, который будет выглядеть примерно как на рис. 3-3.

Рис. 3-3: Файл `hosts` Kali Linux по умолчанию

По умолчанию файл `hosts` содержит только сопоставление для вашего локального хоста по адресу `127.0.0.1` и имени хоста вашей системы (в данном случае Kali, по адресу `127.0.1.1`). Но вы можете добавить любой IP-адрес, сопоставленный с любым нужным доменом. В качестве примера использования можно сопоставить `www.bankofamerica.com` с вашим локальным веб-сайтом по адресу `192.168.181.131`.

```
127.0.0.1 localhost
127.0.1.1 kali
192.168.181.131 bankofamerica.com
# Следующие строки желательны для узлов с поддержкой IPv6
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Обязательно нажмите `TAB` между IP-адресом и ключом домена, а не пробел.

По мере того как вы будете глубже вовлекаться в свои хакерские занятия и узнаете об инструментах вроде `dnsspoof` и Ettercap, вы сможете использовать файл `hosts`, чтобы направлять любой трафик в своей LAN, посещающий `www.bankofamerica.com`, на ваш веб-сервер по адресу `192.168.181.131`.

Довольно просто, правда?

Итоги

Любому хакеру нужны базовые сетевые навыки Linux для подключения, анализа и управления сетями. По мере продвижения эти навыки будут становиться все полезнее для разведки, подделки и подключения к целевым системам.

Упражнения

Прежде чем переходить к главе 4, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Найдите информацию об активных сетевых интерфейсах.
2. Измените IP-адрес на `eth0` на `192.168.1.1`.
3. Измените аппаратный адрес на `eth0`.
4. Проверьте, есть ли у вас активные доступные беспроводные интерфейсы.
5. Сбросьте IP-адрес на адрес, назначенный DHCP.
6. Найдите сервер имен и почтовый сервер своего любимого сайта.
7. Добавьте DNS-сервер Google в файл `/etc/resolv.conf`, чтобы ваша система обращалась к нему, когда не может разрешить запрос доменного имени через локальный DNS-сервер.

Глава 4. Добавление и удаление программ

Одна из самых фундаментальных задач в Linux - как и в любой операционной системе - это добавление и удаление программ. Вам часто придется устанавливать программы, которые не поставлялись с вашим дистрибутивом, или удалять ненужные программы, чтобы они не занимали место на жестком диске.

Некоторым программам для работы нужны другие программы, и иногда вы обнаружите, что можете загрузить все необходимое сразу в виде программного пакета, то есть группы файлов - обычно библиотек и других зависимостей, - которые нужны программе для успешной работы. При установке пакета все файлы внутри него устанавливаются вместе, а также добавляется сценарий, упрощающий загрузку программы.

В этой главе мы рассмотрим три ключевых метода добавления новых программ: менеджер пакетов `apt`, установщики с графическим интерфейсом и `git`.

Использование apt для работы с программами

В дистрибутивах Linux на основе Debian, к которым относятся Kali и Ubuntu, менеджер программ по умолчанию - расширенный инструмент управления пакетами, или `apt`; его основная команда - `apt-get`. В самой простой и распространенной форме `apt-get` можно использовать для загрузки и установки новых программных пакетов, но с его помощью также можно обновлять программы и выполнять полное обновление пакетов.

Поиск пакета

Перед загрузкой программного пакета можно проверить, доступен ли нужный пакет в вашем репозитории, то есть месте, где операционная система хранит информацию. У инструмента `apt` есть функция поиска, которая может проверить, доступен ли пакет. Синтаксис прямолинеен:

```
apt-cache search keyword
```

Обратите внимание, что мы используем команду `apt-cache` для поиска в кэше `apt`, то есть в месте, где он хранит имена пакетов. Например, если бы вы искали систему обнаружения вторжений Snort, нужно было бы ввести команду, показанную в листинге 4-1.

```
kali >apt-cache search snort
fwsnort - Snort-to-iptables rule translator
ippl - IP protocols logger
--snip--
snort - flexible Network Intrusion Detection System
snort-common - flexible Network Intrusion Detection System - common files
--snip--
```

Листинг 4-1: Поиск Snort в системе с помощью `apt-cache`

Как видите, множество файлов содержит ключевое слово `snort`, но примерно в середине вывода мы видим `snort - flexible Network Intrusion Detection System`. Это именно то, что мы ищем!

Добавление программ

Теперь, когда вы знаете, что пакет `snort` существует в вашем репозитории, можно использовать `apt-get`, чтобы загрузить программу.

Чтобы установить программу из репозитория операционной системы по умолчанию в терминале, используйте команду `apt-get`, затем ключевое слово `install`, а затем имя пакета, который хотите установить. Синтаксис выглядит так:

```
apt-get install packagename
```

Попробуем установить Snort в вашей системе. Введите `apt-get install snort` как команду, показанную в листинге 4-2.

```
kali >apt-get install snort
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Предлагаемые пакеты:
snort-doc
Будут установлены следующие НОВЫЕ пакеты:
snort
--snip--
Установить эти пакеты без проверки [Y/n]?
```

Листинг 4-2: Установка Snort с помощью `apt-get install`

Вывод сообщает, что именно устанавливается. Если все выглядит правильно, введите y в ответ на приглашение, и установка программы продолжится.

Удаление программ

При удалении программ используйте `apt-get` с параметром `remove`, за которым следует имя удаляемой программы (см. листинг 4-3).

```
kali >apt-get remove snort
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Следующие пакеты были установлены автоматически и больше не требуются:
libdaq0 libprelude2 oinkmaster snort-common-libraries snort-rules-default
--snip--
Хотите продолжить [Y/n]?
```

Листинг 4-3: Удаление Snort с помощью `apt-get remove`

И снова вы увидите выполняемые задачи в реальном времени, а система спросит, хотите ли вы продолжить. Можно ввести y, чтобы удалить программу, но, возможно, вы захотите оставить Snort, потому что мы еще будем использовать его снова. Команда `remove` не удаляет конфигурационные

файлы; это значит, что в будущем можно переустановить тот же пакет без повторной настройки.

Если вы действительно хотите удалить конфигурационные файлы одновременно с пакетом, можно использовать параметр `purge`, как показано в листинге 4-4.

```
kali >apt-get purge snort
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Следующие пакеты были установлены автоматически и больше не требуются:
libdaq0 libprelude2 oinkmaster snort-common-libraries snort-rules-default
--snip--
Хотите продолжить [Y/n]?
```

Листинг 4-4: Удаление Snort и сопутствующих конфигурационных файлов с помощью `apt-get purge`

Просто введите `Y` в приглашении, чтобы продолжить purge программного пакета и конфигурационных файлов.

Возможно, вы заметили в выводе строку о том, что автоматически установленные пакеты больше не требуются. Чтобы все оставалось небольшим и модульным, многие пакеты Linux разбиты на программные единицы, которые могут использовать разные программы. Когда вы установили Snort, вместе с ним были установлены несколько зависимостей или библиотек, которые нужны Snort для работы. Теперь, когда вы удаляете Snort, эти другие библиотеки или зависимости больше не нужны, поэтому они тоже удаляются.

Обновление списков пакетов

Репозитории программ периодически обновляются новым программным обеспечением или новыми версиями существующих программ. Эти обновления не доходят до вас автоматически, поэтому нужно запросить их, чтобы применить обновления к собственной системе. Обновление списка пакетов - это не то же самое, что обновление самих пакетов: первое просто обновляет список пакетов, доступных для загрузки из репозитория, тогда как второе обновляет пакет до последней версии в репозитории.

Можно обновить свою отдельную систему, введя команду `apt-get`, за которой следует ключевое слово `update`. Это просмотрит все пакеты в вашей системе и проверит, доступны ли обновления. Если да, обновления будут загружены (см. листинг 4-5).

```
kali >apt-get update
Получено:1 http://mirrors.ocf.berkeley.edu/kali kali-rolling InRelease [30.5kb]
Получено:2 http://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 Packages
[14.9MB]
Получено:3 http://mirrors.ocf.berkeley.edu/kali kali-rolling non-free amd64 Packages
[163kb]
Получено:4 http://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib amd64 Packages [107
kB]
Получено 15.2 MB за 1 мин 4 с (236 kB/s)
Чтение списков пакетов... Готово
```

Листинг 4-5: Обновление всех устаревших списков пакетов с помощью `apt-get update`

Список доступных программ в репозитории вашей системы будет обновлен. Если обновление прошло успешно, терминал сообщит `Чтение списков пакетов... Готово`, как видно в листинге 4-5. Обратите внимание, что имя репозитория и значения - время, размер и так далее - в вашей системе могут отличаться.

Обновление установленных пакетов

Чтобы обновить существующие пакеты в системе, используйте `apt-get upgrade`. Поскольку обновление пакетов может внести изменения в ваши программы, нужно войти как `root` или использовать команду `sudo` перед вводом `apt-get upgrade`. Эта команда обновит каждый пакет в вашей системе, о котором знает `apt`, то есть только пакеты, хранящиеся в репозитории (см. листинг 4-6). Полное обновление может занять много времени, поэтому какое-то время вы, возможно, не сможете пользоваться системой.

```
kali >apt-get upgrade
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Расчет обновления... Готово
Следующие пакеты были установлены автоматически и больше не требуются:
--snip--
Следующие пакеты будут обновлены:
--snip--
1101 обновлен, 0 вновь установлено, 0 удалено и 318 не обновлено.
Нужно получить 827 MB архивов.
После этой операции будет освобождено 408 MB дискового пространства.
Хотите продолжить? [Y/n]
```

Листинг 4-6: Обновление всех устаревших пакетов с помощью `apt-get upgrade`

В выводе вы должны увидеть, что система оценивает объем места на жестком диске, необходимый для программного пакета. Введите `Y`, если хотите продолжить и у вас достаточно места на жестком диске для обновления пакетов.

Добавление репозиториев в файл sources.list

Серверы, на которых хранится программное обеспечение для конкретных дистрибутивов Linux, называются репозиториями. Почти у каждого дистрибутива есть собственные репозитории программ - разработанных и настроенных для этого дистрибутива, - которые могут плохо работать или вообще не работать с другими дистрибутивами. Хотя эти репозитории часто содержат одинаковое или похожее программное обеспечение, они не идентичны, и иногда в них разные версии одной и той же программы или совершенно разные программы.

Конечно, вы будете использовать репозиторий Kali, в котором много программ для безопасности и хакинга. Но поскольку Kali специализируется на безопасности и хакинге, в нем нет некоторых специализированных программ и инструментов, а также даже некоторых вполне обычных программ. Стоит добавить один-два резервных репозитория, в которых ваша система сможет искать, если не найдет конкретную программу в репозитории Kali.

Репозитории, в которых ваша система будет искать программы, хранятся в файле `sources.list`, и вы можете изменить этот файл, чтобы определить, из каких репозиториев хотите загружать программы. Я часто добавляю репозитории Ubuntu после репозиториев Kali в своем файле `sources.list`; так, когда я прошу загрузить новый программный пакет, система сначала смотрит в репозитории Kali, а если пакета там нет, смотрит в репозитории Ubuntu.

Файл `sources.list` находится в `/etc/apt/sources.list`, и его можно открыть любым текстовым редактором. Я снова буду использовать Leafpad. Чтобы открыть файл `sources.list`, введите в терминале следующее, заменив `leafpad` именем своего редактора:

```
kali >leafpad /etc/apt/sources.list
```

После ввода этой команды вы должны увидеть окно, похожее на рис. 4-1, со списком репозиториев Kali по умолчанию.

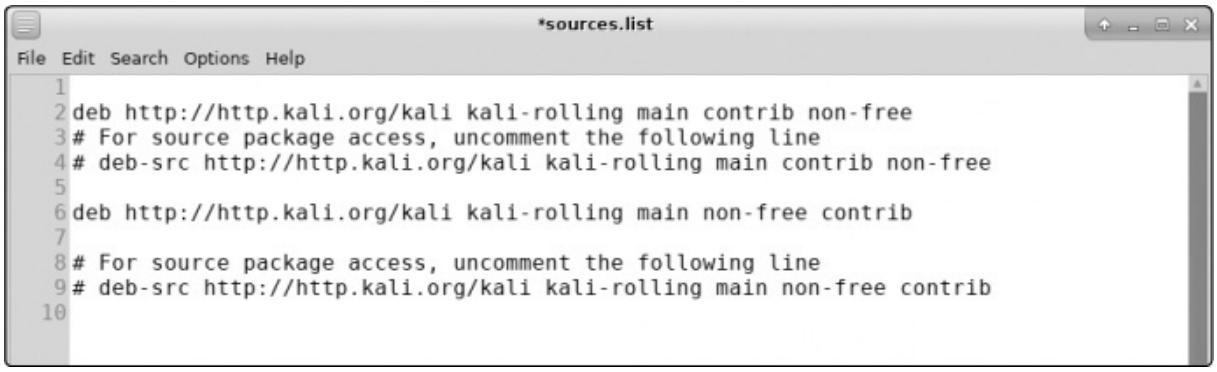

Рис. 4-1: Репозитории Kali по умолчанию в `sources.list`

Многие дистрибутивы Linux делят репозитории на отдельные категории. Например, Ubuntu выделяет такие категории репозиториев:

- `main` - содержит поддерживаемое программное обеспечение с открытым исходным кодом;
- `universe` - содержит программное обеспечение с открытым исходным кодом, поддерживаемое сообществом;
- `multiverse` - содержит программное обеспечение, ограниченное авторским правом или другими юридическими вопросами;
- `restricted` - содержит проприетарные драйверы устройств;
- `backports` - содержит пакеты из более поздних выпусков.

Я не рекомендую использовать репозитории `testing`, `experimental` или `unstable` в вашем `sources.list`, потому что они могут загрузить в систему проблемные программы. Программное обеспечение, которое не полностью протестировано, может сломать систему.

Когда вы просите загрузить новый программный пакет, система последовательно просматривает репозитории, перечисленные в `sources.list`, и останавливается, когда находит нужный пакет. Сначала проверьте, совместим ли репозиторий с вашей системой. Kali, как и Ubuntu, построен на Debian, поэтому эти репозитории довольно хорошо работают с каждой из этих систем.

Чтобы добавить репозиторий, просто отредактируйте файл `sources.list`, добавив имя репозитория в список, а затем сохраните файл. Допустим, например, вы хотите установить Oracle Java 8 на Kali. Пакет `apt` для Oracle Java 8 недоступен в стандартных источниках Kali, но быстрый поиск в интернете показывает, что замечательные люди из WebUpd8 создали такой пакет. Если добавить их репозиторий в источники, затем можно установить Oracle Java 8 командой `apt-get install oracle-java8-installer`. На момент написания, чтобы добавить необходимые репозитории, нужно было добавить в `sources.list` следующие расположения репозиториев:

```
deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main
deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu precise main
```

Использование установщика с графическим интерфейсом

Новые версии Kali больше не включают графический инструмент установки программ, но вы всегда можете установить такой инструмент с помощью команды `apt-get`. Два самых распространенных графических инструмента установки - Synaptic и Gdebi. Установим Synaptic и используем его для установки пакета Snort:

```
kali >apt-get install synaptic
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
--snip--
Обработка триггеров для меню (2.1.47)...
kali >
```

После установки Synaptic можно запустить его из меню `Настройки > Менеджер пакетов Synaptic`; должно открыться окно, похожее на рис. 4-2.

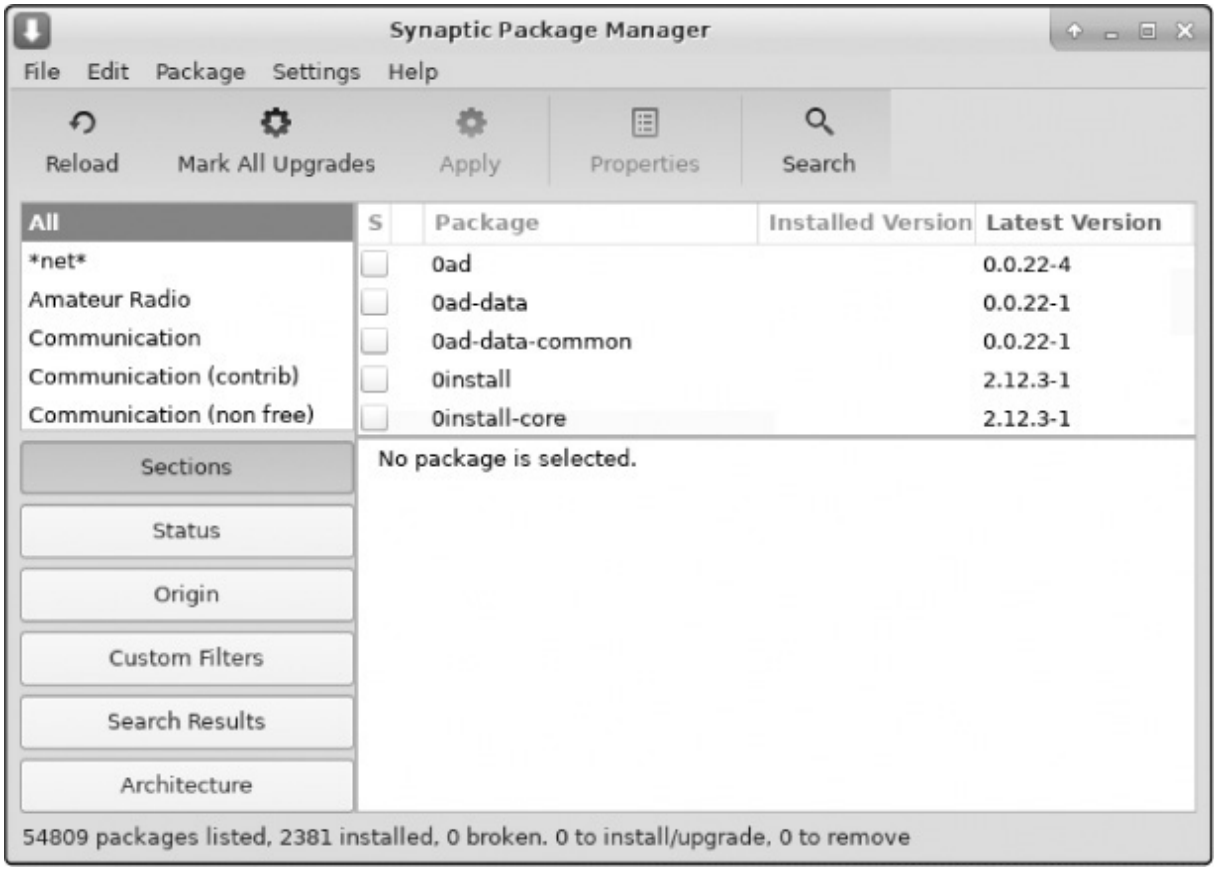

Рис. 4-2: Интерфейс менеджера пакетов Synaptic

Теперь можно искать нужный пакет. Просто щелкните вкладку Search, чтобы открыть окно поиска. Поскольку вы снова ищете Snort, введите `snort` в окно поиска и щелкните Search. Прокрутите результаты поиска вниз, чтобы найти нужный пакет. Установите флажок рядом с `snort`, а затем щелкните вкладку Apply, как показано на рис. 4-3. Теперь Synaptic загрузит и установит Snort из репозитория вместе со всеми необходимыми зависимостями.

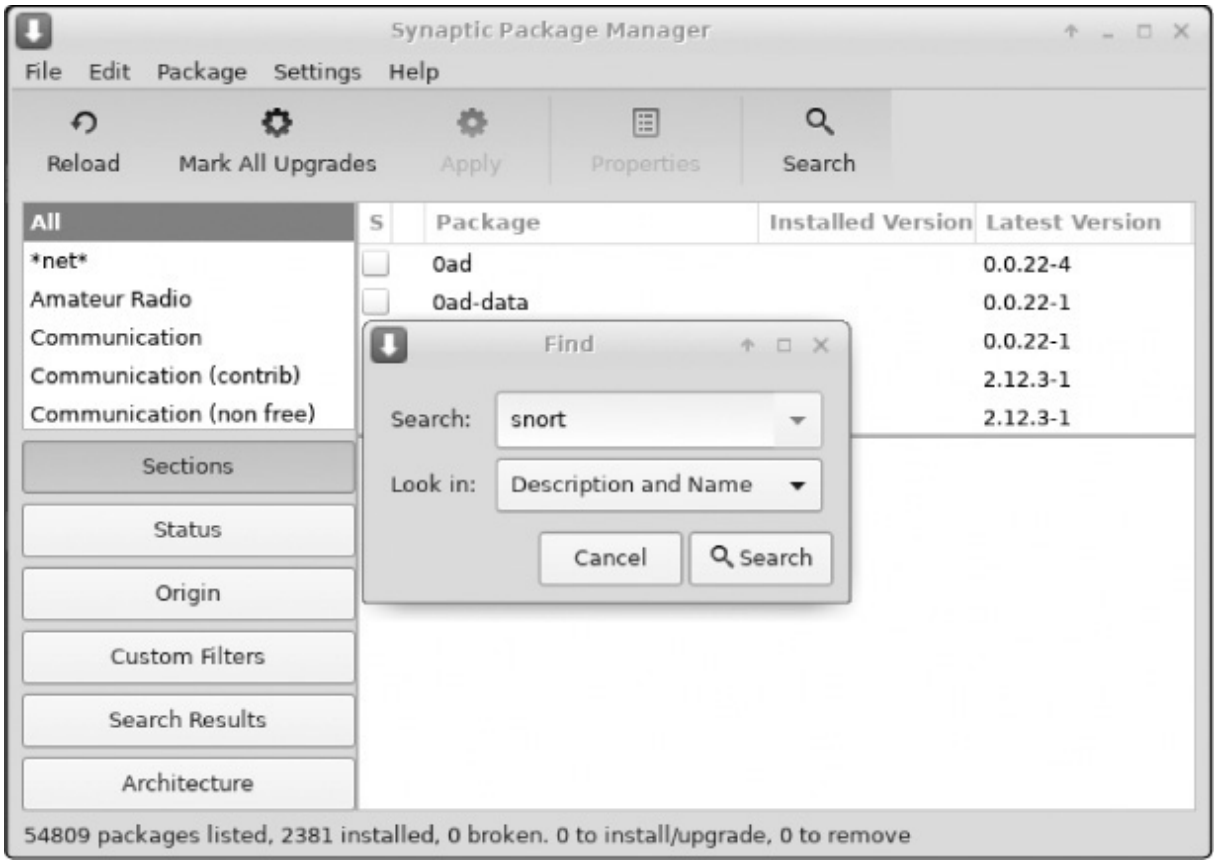

Рис. 4-3: Загрузка Snort из менеджера пакетов Synaptic

Установка программ с помощью git

Иногда нужная вам программа недоступна ни в одном из репозиториев - особенно если она совсем новая, - но может быть доступна на github (https://www.github.com/), сайте, который позволяет разработчикам делиться своим программным обеспечением с другими, чтобы те могли загружать его, использовать и оставлять отзывы. Например, если вам нужен `bluediving`, набор для Bluetooth-хакинга и пентестинга, и вы не можете найти его в репозитории Kali, можно поискать программу на github, введя `bluediving` в строку поиска. Если она есть на github, вы должны увидеть ее репозиторий в результатах поиска.

Когда вы нашли программу на github, можно установить ее из терминала, введя команду `git clone`, за которой следует ее URL на github. Например, `bluediving` находится по адресу https://www.github.com/balle/bluediving.git. Чтобы клонировать его в свою систему, введите команду, показанную в листинге 4-7.

```
kali >git clone https://www.github.com/balle/bluediving.git
Cloning into 'bluediving'...
remote: Counting objects: 131, Done.
remote: Total 131 (delta 0), reused 0 (delta 0), pack-reused 131
Receiving objects: 100% (131/131), 900.81 KiB | 646.00 KiB/s, Done.
Resolving deltas: 100% (9/9), Done.
Checking connectivity... Done.
```

Листинг 4-7: Клонирование `bluediving` с помощью `git clone`

Команда `git clone` копирует все данные и файлы из этого расположения в вашу систему. Можно проверить, успешно ли они загружены, использовав команду длинного листинга `ls -l` в целевом каталоге:

```
kali >ls -l
```

Если вы успешно клонировали `bluediving` в свою систему, вы должны увидеть следующий вывод:

```
total 80
drwxr-xr-x 7 root root 4096 Jan 10 22:19 bluediving
drwxr-xr-x 2 root root 4096 Dec 5 11:17 Desktop
drwxr-xr-x 2 root root 4096 Dec 5 11:17 Documents
drwxr-xr-x 2 root root 4096 Dec 5 11:17 Downloads
drwxr-xr-x 2 root root 4096 Dec 5 11:17 Music
--snip--
```

Как видите, `bluediving` был успешно клонирован в систему, и для его файлов создан новый каталог с именем `bluediving`.

Итоги

В этой главе вы изучили несколько из многих способов загрузки и установки новых программ в системе Linux. Менеджеры программных пакетов (такие как `apt`), установщики с графическим интерфейсом и клоны `git` - самые распространенные и важные методы, которые должен знать начинающий хакер. Скоро вы обнаружите, что освоились с каждым из них.

Упражнения

Прежде чем переходить к главе 5, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Установите новый программный пакет из репозитория Kali.
2. Удалите тот же программный пакет.
3. Обновите свой репозиторий.
4. Выполните обновление своих программных пакетов.
5. Выберите новую программу на github и клонируйте ее в свою систему.

Глава 5. Управление правами файлов и каталогов

Не каждый пользователь одной операционной системы должен иметь одинаковый уровень доступа к файлам и каталогам. Как любая профессиональная или корпоративная операционная система, Linux имеет методы защиты доступа к файлам и каталогам. Эта система безопасности позволяет системному администратору - пользователю `root` - или владельцу файла защищать свои файлы от нежелательного доступа или вмешательства, выдавая выбранным пользователям права на чтение, запись или выполнение файлов.

Для каждого файла и каталога можно указать состояние прав для владельца файла, для определенных групп пользователей и для всех остальных пользователей. Это необходимость в многопользовательской операционной системе корпоративного уровня. Альтернатива была бы довольно хаотичной.

В этой главе я покажу, как проверять и изменять права на файлы и каталоги для выбранных пользователей, как задавать права по умолчанию для файлов и каталогов и как устанавливать специальные права. Наконец, вы увидите, как понимание прав доступа может помочь хакеру эксплуатировать систему.

Разные типы пользователей

Как вы знаете, в Linux пользователь `root` всемогущ. Пользователь `root` в основном может делать в системе что угодно. Другие пользователи системы имеют более ограниченные возможности и права и почти никогда не обладают доступом, который есть у пользователя `root`.

Этих других пользователей обычно собирают в группы, которые в целом выполняют похожую функцию. В коммерческой организации такими группами могут быть финансы, инженерный отдел, продажи и так далее. В IT-среде такие группы могут включать разработчиков, сетевых администраторов и администраторов баз данных. Идея состоит в том, чтобы поместить людей с похожими потребностями в группу, которой выдаются соответствующие права; затем каждый член группы наследует групповые права. Это делается прежде всего для удобства администрирования прав и, следовательно, безопасности.

Пользователь `root` по умолчанию входит в группу `root`. Каждый новый пользователь системы должен быть добавлен в группу, чтобы наследовать права этой группы.

Выдача прав

Каждому файлу и каталогу должен быть назначен определенный уровень прав для разных идентичностей, которые его используют. Три уровня прав таковы:

- `r` - право на чтение. Оно дает разрешение только открыть и просмотреть файл.
- `w` - право на запись. Оно позволяет пользователям просматривать и редактировать файл.
- `x` - право на выполнение. Оно позволяет пользователям выполнять файл, но не обязательно просматривать или редактировать его.

Таким образом пользователь `root` может выдавать пользователям уровень прав в зависимости от того, для чего им нужны файлы. Когда файл создается, обычно пользователь, создавший его, становится владельцем файла, а владеющей группой становится текущая группа пользователя. Владелец файла может выдавать ему различные привилегии доступа. Давайте посмотрим, как менять права, передавая владение отдельным пользователям и группам.

Передача владения отдельному пользователю

Чтобы передать владение файлом другому пользователю, чтобы он мог управлять правами, можно использовать команду `chown` (change owner):

```
kali >chown (1) bob (2) /tmp/bobsfile
```

Здесь мы указываем команду, имя пользователя, которому передаем владение, а затем расположение и имя соответствующего файла. Эта команда выдает учетной записи пользователя Bob (1) владение файлом `bobsfile` (2).

Передача владения группе

Чтобы передать владение файлом от одной группы другой, можно использовать команду `chgrp` (change group).

Хакеры чаще работают в одиночку, чем в группах, но ситуация, когда несколько хакеров или пентестеров вместе работают над проектом, вполне возможна; в таком случае использование групп необходимо. Например, у вас может быть группа пентестеров и группа участников команды безопасности, работающих над одним проектом. Пентестеры в этом примере - группа `root`, то есть у них есть все права и доступ. Группе `root` нужен доступ к хакерским инструментам, тогда как специалистам по безопасности нужен доступ только к защитным инструментам, таким как система обнаружения вторжений (IDS). Допустим, группа `root` загружает и устанавливает программу с именем `newIDS`; группе `root` нужно изменить владение на группу `security`, чтобы группа `security` могла свободно использовать ее. Для этого группа `root` просто ввела бы следующую команду:

```
kali >chgrp (1) security (2) newIDS
```

Эта команда передает группе `security` (1) владение `newIDS` (2).

Теперь нужно узнать, как проверить, сработали ли эти назначения. Вы сделаете это, проверив права файла.

Проверка прав

Когда нужно выяснить, какие права на файл или каталог выданы каким пользователям, используйте команду `ls` с ключом `-l` (long), чтобы отобразить содержимое каталога в длинном формате: этот список будет содержать права. В листинге 5-1 я использую команду `ls -l` для файла `/usr/share/hashcat` (одного из моих любимых инструментов для взлома паролей), чтобы увидеть, что можно узнать о находящихся там файлах.

```
kali >ls -l /usr/share/hashcat
total 32952
(1) (2)   (3)  (4)     (5)    (6)     (7)
drwxr-xr-x 5 root root 4096 Dec 5 10:47 charsets
-rw-r--r-- 1 root root 33685504 June 28 2018 hashcat.hcstat
-rw-r--r-- 1 root root 33685504 June 28 2018 hashcat.hctune
drwxr-xr-x 2 root root 4096 Dec 5 10:47 masks
drwxr-xr-x 2 root root 4096 Dec 5 10:47 OpenCL
drwxr-xr-x 3 root root 4096 Dec 5 10:47 rules
```

Листинг 5-1: Проверка прав файла командой длинного листинга

В каждой строке мы получаем информацию о следующем:

• (1) тип файла;
• (2) права на файл для владельца, групп и пользователей соответственно;
• (3) число ссылок (эта тема выходит за рамки книги);
• (4) владелец файла;
• (5) размер файла в байтах;
• (6) когда файл был создан или изменен в последний раз;
• (7) имя файла.

Пока сосредоточимся на кажущихся непонятными строках букв и дефисов у левого края каждой строки. Они сообщают, является ли объект файлом или каталогом и какие права, если они есть, на него установлены.

Первый символ сообщает тип файла: `d` означает каталог, а дефис (`-`) указывает на файл. Это два самых распространенных типа файлов.

Следующая часть определяет права на файл. Есть три набора по три символа, составленные из некоторого сочетания read (`r`), write (`w`) и execute (`x`) именно в таком порядке. Первый набор представляет права владельца; второй - права группы; последний - права всех остальных пользователей.

Независимо от того, на какой набор из трех букв вы смотрите, если первым стоит `r`, этот пользователь или группа пользователей имеет право открыть и прочитать этот файл или каталог. `w` как средняя буква означает, что они могут записывать в файл или каталог, то есть изменять его, а `x` в конце означает, что они могут выполнить, или запустить, файл или каталог. Если любой из символов `r`, `w` или `x` заменен дефисом (`-`), соответствующее право не выдано. Обратите внимание, что пользователи могут иметь право выполнения только для бинарных файлов или сценариев.

Используем третью строку вывода в листинге 5-1 как пример:

```
-rw-r--r-- 1 root root 33685504 June 28 2018 hashcat.hcstat
```

Файл, как мы знаем по правому концу строки, называется `hashcat.hcstat`. После начального `-`, который указывает, что это файл, права `rw-` сообщают, что у владельца есть права чтения и записи, но нет права выполнения.

Следующий набор прав (`r--`) представляет права группы и показывает, что у группы есть право чтения, но нет прав записи или выполнения. И наконец, мы видим, что у остальных пользователей также есть только право чтения (`r--`).

Эти права не высечены в камне. Как пользователь `root` или владелец файла вы можете их изменять. Дальше мы как раз это сделаем.

Изменение прав

Для изменения прав можно использовать команду Linux `chmod` (change mode). Изменять права может только пользователь `root` или владелец файла.

В этом разделе мы используем `chmod`, чтобы изменить права на `hashcat.hcstat` двумя разными способами. Сначала используем числовое представление прав, а затем символическое.

Изменение прав в десятичной записи

Можно использовать сокращение для обозначения прав: одно число представляет один набор прав `rwx`. Как и все, что лежит в основе операционной системы, права представлены в двоичном виде, поэтому переключатели ON и OFF представлены соответственно 1 и 0. Можно думать о правах `rwx` как о трех переключателях ON/OFF, так что когда выданы все права, это равно `111` в двоичном виде.

Такой двоичный набор затем легко представить одной цифрой, преобразовав его в octal, восьмеричную систему счисления, которая начинается с 0 и заканчивается 7. Одна восьмеричная цифра представляет набор из трех двоичных цифр, то есть весь набор `rwx` можно представить одной цифрой. Таблица 5-1 содержит все возможные сочетания прав и их восьмеричные и двоичные представления.

Таблица 5-1: Восьмеричные и двоичные представления прав

Двоичное	Восьмеричное	rwx
000	0	`---`
001	1	`--x`
010	2	`-w-`
011	3	`-wx`
100	4	`r--`
101	5	`r-x`
110	6	`rw-`
111	7	`rwx`

Используя эту информацию, пройдем несколько примеров. Сначала, если мы хотим установить только право чтения, можно обратиться к таблице 5-1 и найти значение для чтения:

```
r w x
4 - -
```

Далее, если мы хотим установить право `wx`, можно использовать тот же метод и посмотреть, что устанавливает `w` и что устанавливает `x`:

```
r w x
- 2 1
```

Обратите внимание в таблице 5-1: восьмеричное представление для `-wx` равно 3, что совсем не случайно совпадает со значением, которое мы получаем, складывая два значения для установки `w` и `x` по отдельности: 2 + 1 = 3.

Наконец, когда включены все три права, это выглядит так:

```
r w x
4 2 1
```

А 4 + 2 + 1 = 7. Здесь мы видим, что в Linux, когда все переключатели прав включены, они представлены восьмеричным эквивалентом 7.

Итак, если бы мы хотели представить все права для владельца, группы и всех пользователей, можно было бы записать это так:

```
7 7 7
```

Вот где появляется сокращение. Передав `chmod` три восьмеричные цифры (по одной для каждого набора `rwx`), а затем имя файла, можно изменить права на этот файл для каждого типа пользователя. Введите в командной строке следующее:

```
kali >chmod 774 hashcat.hcstat
```

Глядя на таблицу 5-1, мы видим, что эта инструкция дает владельцу все права, группе все права, а всем остальным (`other`) только право чтения.

Теперь можно увидеть, изменились ли эти права, запустив `ls -l` для каталога и посмотрев на строку `hashcat.hcstat`. Перейдите в каталог и выполните эту команду:

```
kali >ls -l
total 32952
drwxr-xr-x 5 root root 4096 Dec 5 10:47 charsets
(1) -rwxrwxr-- 1 root root 33685504 June 28 2018 hashcat.hcstat
-rw-r--r-- 1 root root 33685504 June 28 2018 hashcat.hctune
drwxr-xr-x 2 root root 4096 Dec 5 10:47 masks
drwxr-xr-x 2 root root 4096 Dec 5 10:47 OpenCL
drwxr-xr-x 3 root root 4096 Dec 5 10:47 rules
```

Слева в строке `hashcat.hcstat` (1) вы должны увидеть `-rwxrwxr--`. Это подтверждает, что вызов `chmod` успешно изменил права на файл, дав и владельцу, и группе возможность выполнять файл.

Изменение прав с помощью UGO

Хотя числовой метод, вероятно, самый распространенный способ изменять права в Linux, некоторым людям символический метод `chmod` кажется более интуитивным. Оба метода работают одинаково хорошо, так что просто найдите тот, который подходит вам. Символический метод часто называют синтаксисом UGO: user (или owner), group и others.

Синтаксис UGO очень прост. Введите команду `chmod`, а затем пользователей, для которых хотите изменить права: `u` для user, `g` для group или `o` для others, после чего укажите один из трех операторов:

- `-` - удаляет право;
- `+` - добавляет право;
- `=` - устанавливает право.

После оператора включите право, которое хотите добавить или удалить (`rwx`), и, наконец, имя файла, к которому нужно применить изменение.

Итак, если вы хотите удалить право записи у пользователя, которому принадлежит файл `hashcat.hcstat`, можно ввести следующее:

```
kali >chmod u-w hashcat.hcstat
```

Эта команда говорит удалить (`-`) право записи (`w`) у пользователя (`u`) для `hashcat.hcstat`.

Теперь, когда вы снова проверите права с помощью `ls -l`, вы должны увидеть, что у файла `hashcat.hcstat` больше нет права записи для пользователя:

```
kali >ls -l
total 32952
drwxr-xr-x 5 root root 4096 Dec 5 10:47 charsets
-r-xr-xr-- 1 root root 33685504 June 28 2018 hashcat.hcstat
-rw-r--r-- 1 root root 33685504 June 28 2018 hashcat.hctune
drwxr-xr-x 2 root root 4096 Dec 5 10:47 masks
drwxr-xr-x 2 root root 4096 Dec 5 10:47 OpenCL
drwxr-xr-x 3 root root 4096 Dec 5 10:47 rules
```

Также можно изменить несколько прав одной командой. Если вы хотите дать и пользователю, и другим пользователям (не включая группу) право выполнения, можно ввести следующее:

```
chmod u+x, o+x hashcat.hcstat
```

Эта команда говорит Linux добавить право выполнения для пользователя, а также право выполнения для others для файла `hashcat.hcstat`.

Выдача root права выполнения для нового инструмента

Как хакеру вам часто придется загружать новые хакерские инструменты, но Linux автоматически назначает всем файлам и каталогам права по умолчанию 666 и 777 соответственно. Это означает, что по умолчанию вы не сможете выполнить файл сразу после загрузки. Если попытаться, обычно появится сообщение вроде "Permission denied". В таких случаях вам понадобится с помощью `chmod` дать себе права `root` и выполнения, чтобы выполнить файл.

Например, допустим, мы загрузили новый хакерский инструмент под названием `newhackertool` и поместили его в каталог пользователя `root` (`/`).

```
kali >ls -l
total 80
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Desktop
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Documents
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Downloads
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Music
-rw-r--r-- 1 root root 1072 Dec 5 11.17 newhackertool (1)
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Pictures
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Public
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Templates
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Videos
```

Мы видим `newhackertool` в (1) вместе с остальным содержимым каталога `root`. Мы видим, что у нашего `newhackertool` нет права выполнения ни для кого. Это делает его невозможным для

использования. Может показаться странным, что Linux по умолчанию не дает выполнять загруженный файл, но в целом такая настройка делает систему безопаснее.

Мы можем дать себе право выполнять `newhackertool`, введя следующее:

```
kali >chmod 766 newhackertool
```

Теперь, когда мы выполняем длинный листинг каталога, видно, что у нашего `newhackertool` есть право выполнения для владельца:

```
kali >chmod 766 newhackertool
kali >ls -l
total 80
--snip--
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Music
-rwxrw-rw- 1 root root 1072 Dec 5 11.17 newhackertool
drwxr-xr-x 7 root root 4096 Dec 5 11.17 Pictures
--snip--
```

Как вы теперь понимаете, это дает нам, как владельцу, все права, включая выполнение, а группе и всем остальным дает только права чтения и записи (4 + 2 = 6).

Задание более безопасных прав по умолчанию с помощью масок

Как вы видели, Linux автоматически назначает базовые права: обычно 666 для файлов и 777 для каталогов. Можно изменить права по умолчанию, выделяемые файлам и каталогам, создаваемым каждым пользователем, с помощью метода `umask` (или unmask). Метод `umask` представляет права, которые вы хотите удалить из базовых прав на файл или каталог, чтобы сделать их более безопасными.

`umask` - это трехзначное десятичное число, соответствующее трем цифрам прав, но число `umask` вычитается из числа прав, чтобы получить новое состояние прав. Это означает, что при создании нового файла или каталога его права устанавливаются как значение по умолчанию минус значение `umask`, как показано на рис. 5-1.

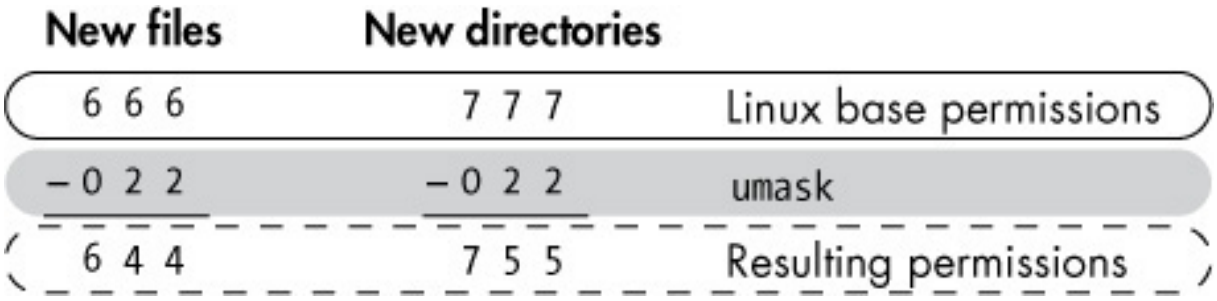

Рис. 5-1: Как значение `umask` 022 влияет на права новых файлов и каталогов

Например, если `umask` установлен в `022`, новый файл с исходными правами по умолчанию `666` теперь будет иметь права `644`, то есть владелец имеет права чтения и записи, а группа и все остальные пользователи имеют только право чтения.

В Kali, как и в большинстве систем Debian, `umask` предварительно настроен на `022`, что означает: значение Kali по умолчанию - `644` для файлов и `755` для каталогов.

Значение `umask` не универсально для всех пользователей системы. Каждый пользователь может задать личное значение `umask` по умолчанию для файлов и каталогов в своем личном файле `.profile`. Чтобы увидеть текущее значение, войдя как пользователь, просто введите команду `umask` и обратите внимание на возвращенное значение. Чтобы изменить значение `umask` для пользователя, отредактируйте файл `/home/username/.profile` и, например, добавьте `umask 007`,

чтобы настроить права так, что только пользователь и члены группы пользователя будут иметь права.

Специальные права

В дополнение к трем правам общего назначения, `rwx`, Linux имеет три специальных права, которые немного сложнее. Это установка идентификатора пользователя (SUID), установка идентификатора группы (SGID) и sticky bit. В следующих трех разделах я рассмотрю каждое по очереди.

Временная выдача прав root с помощью SUID

Как вы уже должны знать, пользователь может выполнить файл только в том случае, если у него есть право выполнять именно этот файл. Если у пользователя есть только права чтения и/или записи, он не может выполнять файл. Это может казаться очевидным, но у правила есть исключения.

Возможно, вы сталкивались со случаем, когда файлу во время выполнения нужны права пользователя `root` для всех пользователей, даже для тех, кто не является `root`. Например, файлу, который позволяет пользователям менять свой пароль, нужен доступ к файлу `/etc/shadow` - файлу, который хранит пароли пользователей в Linux, - а для выполнения требуется привилегия пользователя `root`. В таком случае можно временно выдать привилегии владельца для выполнения файла, установив бит SUID на программу.

По сути, бит SUID говорит, что любой пользователь может выполнить файл с правами владельца, но эти права не распространяются за пределы использования этого файла.

Чтобы установить бит SUID, введите `4` перед обычными правами, так что файл с новым итоговым правом `644` при установленном бите SUID будет представлен как `4644`.

Установка SUID на файл - не то, что обычно делает рядовой пользователь, но если вы хотите это сделать, используйте команду `chmod`, например `chmod 4644 filename`.

Выдача прав группы пользователя root с помощью SGID

SGID также выдает временные повышенные права, но он выдает права группы владельца файла, а не права владельца файла. Это означает, что при установленном бите SGID тот, у кого нет права выполнения, может выполнить файл, если владелец принадлежит группе, у которой есть право выполнять этот файл.

Бит SGID работает немного иначе, когда применяется к каталогу: когда бит установлен на каталог, владение новыми файлами, созданными в этом каталоге, переходит к группе создателя каталога, а не к группе создателя файла. Это очень полезно, когда каталог используется несколькими пользователями совместно. Все пользователи этой группы могут выполнять файлы, а не только один пользователь.

Бит SGID представлен как `2` перед обычными правами, поэтому новый файл с итоговыми правами `644` при установленном бите SGID будет представлен как `2644`. И снова для этого используется команда `chmod`, например `chmod 2644 filename`.

Устаревший sticky bit

Sticky bit - это бит прав, который можно установить на каталог, чтобы позволить пользователю удалять или переименовывать файлы внутри этого каталога. Однако sticky bit - наследие старых систем Unix, и современные системы, такие как Linux, игнорируют его. Поэтому дальше я не буду его здесь обсуждать, но вы должны знать этот термин, потому что можете услышать его в мире Linux.

Специальные права, повышение привилегий и хакер

Как хакер вы можете использовать эти специальные права для эксплуатации систем Linux через повышение привилегий, при котором обычный пользователь получает привилегии `root` или системного администратора и связанные с ними права. С правами `root` в системе можно делать что угодно.

Один способ сделать это - эксплуатировать бит SUID. Системный администратор или разработчик программы может установить бит SUID на программу, чтобы позволить этой программе получать доступ к файлам с привилегиями `root`. Например, у сценариев, которым нужно менять пароли, часто установлен бит SUID. Вы, хакер, можете использовать это право, чтобы получить временные привилегии `root` и сделать что-то вредоносное, например получить доступ к паролям в `/etc/shadow`.

Поищем файлы с установленным битом SUID в нашей системе Kali, чтобы попробовать это. В главе 1 я познакомил вас с командой `find`. Мы используем ее мощь, чтобы найти файлы с установленным битом SUID.

Как вы помните, команда `find` мощная, но ее синтаксис немного сложнее, чем у некоторых других команд поиска, таких как `locate` и `which`. Если нужно, потратьте минуту и повторите синтаксис `find` в главе 1.

В этом случае мы хотим найти файлы в любом месте файловой системы, принадлежащие пользователю `root` или другому системному администратору, с правами `4000`. Для этого можно использовать следующую команду `find`:

```
kali >find / -user root -perm -4000
```

Этой командой мы просим Kali начать поиск с вершины файловой системы, используя синтаксис `/`. Затем она ищет везде ниже `/` файлы, принадлежащие `root`, что указано через `user root`, и имеющие установленный бит прав SUID (`-perm -4000`).

Когда мы выполняем эту команду, получаем вывод, показанный в листинге 5-2.

```
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/pkexec
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/kismet_capture
--snip--
```

Листинг 5-2: Поиск файлов с установленным битом SUID

Вывод раскрывает множество файлов, у которых установлен бит SUID. Перейдем в каталог `/usr/bin`, где находятся многие из этих файлов, затем выполним длинный листинг этого каталога и прокрутим вниз до файла `sudo`, как показано в листинге 5-3.

```
kali >cd /usr/bin
kali >ls -l
--snip--
```

```
-rwxr-xr-x 1 root root 176272 Jul 18 2018 stunnel4
-rwxr-xr-x 1 root root 26696 Mar 17 2018 sucrack
(1) -rwsr-xr-x 1 root root 140944 Jul 5 2018 sudo
--snip--
```

Листинг 5-3: Определение файлов с установленным битом SUID

Обратите внимание: в (1) первый набор прав - для владельца - имеет `s` на месте `x`. Так Linux показывает, что установлен бит SUID. Это означает, что любой, кто запускает файл `sudo`, получает привилегии пользователя `root`, что может быть проблемой безопасности для системного администратора и потенциальным вектором атаки для хакера. Например, некоторым приложениям для успешного выполнения задач нужен доступ к файлу `/etc/shadow`. Если атакующий сможет получить контроль над таким приложением, он сможет использовать доступ этого приложения к паролям в системе Linux.

В Linux есть хорошо развитая система безопасности, защищающая файлы и каталоги от несанкционированного доступа. Начинающему хакеру нужно базовое понимание этой системы не только для защиты своих файлов, но и для выполнения новых инструментов и файлов. В некоторых случаях хакеры могут эксплуатировать права SUID и SGID, чтобы повысить привилегии от обычного пользователя до пользователя `root`.

Итоги

Использование прав в Linux для защиты файлов и каталогов пользователя или группы от других пользователей системы можно применять и в наступательных, и в защитных целях. Теперь вы должны знать, как управлять этими правами и как эксплуатировать слабые места в этой системе безопасности, особенно биты SUID и SGID.

Упражнения

Прежде чем переходить к главе 6, проверьте знания, полученные в этой главе, выполнив следующие упражнения:

1. Выберите каталог и выполните для него длинный листинг. Отметьте права на файлы и каталоги.
2. Выберите файл, который у вас нет прав выполнять, и выдайте себе права выполнения с помощью команды `chmod`. Попробуйте использовать и числовой метод (`777`), и метод UGO.
3. Выберите другой файл и измените его владельца с помощью `chown`.
4. Используйте команду `find`, чтобы найти все файлы с установленным битом SGID.

Глава 6. Управление процессами

В любой момент времени в системе Linux обычно одновременно выполняются сотни, а иногда даже тысячи процессов. Процесс - это просто программа, которая выполняется и использует ресурсы. К процессам относятся терминал, веб-сервер, любые выполняющиеся команды, любые базы данных, интерфейс GUI и многое другое. Любой хороший администратор Linux - и особенно хакер - должен понимать, как управлять своими процессами, чтобы оптимизировать систему. Например, когда хакер получает контроль над целевой системой, он может захотеть найти и остановить определенный процесс, такой как антивирусное приложение или файрвол. Для этого хакеру сначала нужно знать, как найти процесс. Хакер также может захотеть настроить периодический запуск сканирующего сценария, чтобы искать уязвимые системы, поэтому мы также посмотрим, как планировать такой сценарий.

В этой главе вы научитесь управлять этими процессами. Сначала вы научитесь просматривать и находить процессы, а также узнавать, какие процессы используют больше всего ресурсов. Затем вы научитесь управлять процессами: запускать их в фоне, назначать им приоритет и при необходимости завершать их (без крови). Наконец, вы научитесь планировать запуск процессов в указанные дни и даты и в конкретное время.

Просмотр процессов

В большинстве случаев первый шаг в управлении процессами - посмотреть, какие процессы выполняются в вашей системе. Основной инструмент для просмотра процессов - и один из лучших друзей администратора Linux - это команда `ps`. Запустите ее в командной строке, чтобы увидеть активные процессы:

```
kali >ps
PID TTY   TIME   CMD
39659 pts/0 00:00:01 bash
39665 pts/0 00:00:00 ps
```

Ядро Linux, внутреннее ядро операционной системы, управляющее почти всем, назначает каждому процессу уникальный process ID (PID) последовательно, по мере создания процессов. При работе с этими процессами в Linux часто нужно указывать их PID, поэтому гораздо важнее отметить PID процесса, чем имя процесса.

Сама по себе команда ps на самом деле не дает много информации. Запуск команды ps без параметров перечисляет процессы, запущенные текущим вошедшим пользователем (в нашем случае root), и процессы, выполняющиеся в этом терминале. Здесь она просто сообщает, что оболочка bash открыта и работает, а также что мы запустили команду ps. Нам нужно гораздо больше информации, особенно о процессах, запущенных другими пользователями и системой в фоне. Без этой информации мы очень мало знаем о том, что на самом деле происходит в нашей системе.

Запуск команды ps с параметрами aux покажет все процессы, выполняющиеся в системе для всех пользователей, как показано в листинге 6-1. Обратите внимание, что перед этими параметрами не ставится дефис (-) и что все написано в нижнем регистре; поскольку Linux чувствителен к регистру, использование параметров в верхнем регистре даст значительно другие результаты.

```
kali >ps aux
USER PID %CPU  %MEM    VSZ RSS TTY  STAT START TIME  COMMAND
root 1  0.0 0.4 202540 6396 ? Ss Apr24 0:46 /sbin/init
root 2  0.0 0.0   0  0 ? S  Apr24 0:00 [kthreadd]
root 3  0.0 0.0   0  0 ? S  Apr24 0:26 [ksoftirqd/0]
--snip--
root 39706 0.0 0.2 36096 3204 pts/0 R+ 15:05 0:00 ps aux
```

Листинг 6-1: Использование параметров aux для просмотра процессов всех пользователей

Как видите, теперь эта команда перечисляет так много процессов, что они, вероятно, уходят за нижний край экрана. Первый процесс - init, указанный в последнем столбце, а последний процесс - команда, которую мы запустили для отображения, ps aux. Многие детали (PID, %CPU, TIME, COMMAND и так далее) могут отличаться в вашей системе, но формат должен быть тем же.

Для наших целей самые важные столбцы в этом выводе таковы:

• USER - пользователь, который запустил процесс;
• PID - process ID;
• %CPU - процент CPU, который использует этот процесс;
• %MEM - процент памяти, который использует этот процесс;
• COMMAND - имя команды, запустившей процесс.

В целом, чтобы выполнить какое-либо действие над процессом, нужно указать его PID. Давайте посмотрим, как использовать этот идентификатор в своих интересах.

Фильтрация по имени процесса

Когда мы запрашиваем сведения о процессах или выполняем над ними действие, обычно нам не нужно выводить на экран все процессы. Это просто проблема слишком большого количества информации. Чаще всего нам нужно найти информацию об одном процессе. Для этого можно использовать команду фильтрации grep, с которой я познакомил вас в главе 1.

Для демонстрации используем фреймворк эксплуатации Metasploit, самый широко используемый фреймворк эксплуатации и хороший друг почти каждого хакера. Он установлен в вашей системе Kali, поэтому запустите Metasploit следующим образом:

```
kali >msfconsole
```

Когда фреймворк эксплуатации запущен, посмотрим, сможем ли мы найти его в списке процессов. Для этого используйте команду `ps aux`, а затем передайте ее по каналу (`|`) в `grep`, ища строку `msfconsole`, как в листинге 6-2.

```
kali >ps aux | grep msfconsole
root 39756 0.0 0.0 4304 716 pts/2 Ss+ 15:13 0:00 sh -c service
postgresql start && msfdb init & msfconsole
root 39759 35.1 15.2 4304 227888 pts/2 Sl+ 15:13 1:36 ruby /usr/bin/
msfconsole
root 39892 0.0 0.0 4304 940 pts/2 S+ 15:18 0:00 grep msfconsole
```

Листинг 6-2: Фильтрация поиска `ps`, чтобы найти конкретный процесс

В отфильтрованном выводе этого листинга вы должны увидеть все процессы, соответствующие термину `msfconsole`. Сначала показана база данных PostgreSQL, которую использует Metasploit, затем сама программа `msfconsole` из `/usr/bin/msfconsole`. Наконец, вы должны увидеть команду `grep`, которую использовали для поиска `msfconsole`. Обратите внимание, что вывод не включает список заголовков столбцов из `ps`. Поскольку ключевого слова `msfconsole` нет в заголовке, он не отображается. Тем не менее результаты показаны в том же формате.

Из этого можно узнать важную информацию. Например, если нужно узнать, сколько ресурсов использует Metasploit, можно обратиться к третьему столбцу (столбцу CPU) и увидеть, что он использует 35,1 процента вашего CPU, а затем к четвертому столбцу, чтобы увидеть, что он использует 15,2 процента системной памяти. Это немало. Требовательный зверь!

Поиск самых прожорливых процессов с помощью top

Когда вы вводите команду `ps`, процессы отображаются в порядке запуска, и поскольку ядро назначает PID в порядке запуска, вы видите процессы, упорядоченные по номеру PID.

Во многих случаях нам нужно знать, какие процессы используют больше всего ресурсов. Здесь полезна команда `top`, потому что она отображает процессы, упорядоченные по используемым ресурсам, начиная с крупнейших. В отличие от команды `ps`, которая дает разовый снимок процессов, `top` обновляет список динамически - по умолчанию каждые 10 секунд. Вы можете наблюдать и отслеживать эти ресурсоемкие процессы, как показано в листинге 6-3.

```
kali >top
top - 15:31:17 up 2 days, ^;50, 4 users, load average: 0.00, 0.04, 0.09
Tasks: 176 total, 1 running, 175 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.3 us, 0.7 sy, ).) ni, 97.4 id, 0.0 wa, 0.0 hi 0.0 si 0.0
KiB Mem : 1491220 total, 64848 free, 488272 used, 938100 buff/cache
KiB Swap : 1046524 total, 1044356 free, 2168 used. 784476 avail MEM
PID USER PR  NI VIRT  RES  SHR  S %CPU  %MEM   TIME+  COMMAND
39759 root 20 0 893180 247232 11488 S 0.7 16.6 1:47.88 ruby
39859 root 20 0 27308 16796 14272 S 0.3 1.2 1:47.88 postgres
39933 root 20 0 293936 61500 29108 S 0.7 4.1 1:47.88 Xorg
--snip--
```

Листинг 6-3: Поиск самых прожорливых процессов с помощью `top`

Системные администраторы часто держат `top` запущенным в терминале, чтобы отслеживать использование ресурсов процессами. Как хакер вы можете захотеть делать то же самое, особенно если в вашей системе выполняется несколько задач. Пока `top` запущен, нажатие клавиши `H` или `?` выведет список интерактивных команд, а нажатие `Q` завершит `top`. Скоро вы снова используете `top`

для управления процессами в разделах "Изменение приоритета процесса с помощью `nice`" и "Завершение процессов".

Управление процессами

Хакерам часто нужно выполнять несколько процессов одновременно, и такая операционная система, как Kali, для этого идеальна. Хакер может запускать сканер портов одновременно со сканером уязвимостей и эксплойтом. Это требует, чтобы хакер эффективно управлял этими процессами, наилучшим образом используя системные ресурсы и завершая задачу. В этом разделе я покажу, как управлять несколькими процессами.

Изменение приоритета процесса с помощью nice

Вы нечасто слышите слово nice в контексте хакеров, но здесь услышите. Команда `nice` используется, чтобы влиять на приоритет процесса для ядра. Как вы видели при запуске команды `ps`, в системе одновременно выполняется множество процессов, и все они конкурируют за доступные ресурсы. Окончательное решение о приоритете процесса принимает ядро, но можно использовать `nice`, чтобы предложить повысить приоритет процесса.

Идея термина nice в том, что при его использовании вы определяете, насколько "любезны" будете к другим пользователям: если ваш процесс использует большую часть системных ресурсов, вы ведете себя не слишком nice.

Значения `nice` находятся в диапазоне от -20 до +19, где ноль - значение по умолчанию (см. рис. 6-1). Высокое значение `nice` означает низкий приоритет, а низкое значение `nice` означает высокий приоритет, то есть вы не так уж любезны к другим пользователям и процессам. Когда процесс запускается, он наследует значение `nice` родительского процесса. Владелец процесса может снизить приоритет процесса, но не может повысить его. Конечно, суперпользователь, или пользователь `root`, может произвольно установить значение `nice` каким угодно.

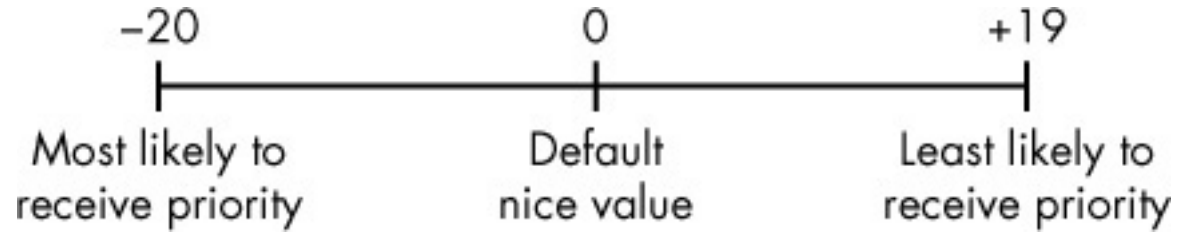

Рис. 6-1: Значения приоритета niceness

Когда вы запускаете процесс, уровень приоритета можно задать командой `nice`, а после того как процесс начал выполняться, изменить приоритет можно командой `renice`. Синтаксис этих двух команд немного отличается и может сбивать с толку. Команда `nice` требует увеличить значение `nice`, тогда как команда `renice` хочет абсолютное значение niceness. Рассмотрим пример, чтобы показать это.

Задание приоритета при запуске процесса

Для демонстрации предположим, что у нас есть процесс с именем `slowprocess`, расположенный в `/bin/slowprocess`. Если мы хотим ускорить его завершение, можно запустить процесс командой `nice`:

```
kali >nice -n -10 /bin/slowprocess
```

Эта команда увеличит значение `nice` на `-10`, повысив его приоритет и выделив ему больше ресурсов.

С другой стороны, если мы хотим быть любезными к другим пользователям и процессам и дать `slowprocess` более низкий приоритет, можно положительно увеличить его значение `nice` на 10:

```
kali >nice -n 10 /bin/slowprocess
```

Попробуйте это на процессе, который сейчас выполняется, а затем запустите `ps`, чтобы увидеть, как он изменился, если вообще изменился.

Изменение приоритета выполняющегося процесса с помощью renice

Команда `renice` принимает абсолютные значения между -20 и 19 и устанавливает приоритет на этот конкретный уровень, а не повышает или снижает его относительно уровня, с которым процесс был запущен. Кроме того, `renice` требует PID целевого процесса, а не его имя. Итак, если `slowprocess` использует чрезмерное количество ресурсов в вашей системе и вы хотите дать ему более низкий приоритет, тем самым позволяя другим процессам получить более высокий приоритет и больше ресурсов, можно выполнить `renice` для `slowprocess`, PID которого равен 6996, и дать ему гораздо более высокое значение `nice`:

```
kali >renice 20 6996
```

Как и с `nice`, только пользователь `root` может выполнить `renice` процесса до отрицательного значения, чтобы дать ему более высокий приоритет, но любой пользователь может быть nice и снизить приоритет с помощью `renice`.

Также можно использовать утилиту `top`, чтобы изменить значение `nice`. Когда утилита `top` запущена, просто нажмите клавишу `R`, а затем укажите PID и значение `nice`. Листинг 6-4 показывает работающую утилиту `top`. Когда я нажимаю клавишу `R` и указываю PID и значение `nice`, получаю следующий вывод:

```
top - 21:36:56 up 21:41,  2 users, load average: 0.60, 0.22, 0.11
Tasks: 128 total,  1 running, 127 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.5 us, 0.7 sy, 0.0 ni, 96.7 id, 1.1 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem:  511864 total,  500780 used, 11084 free,  152308 buffers
KiB Swap:  901116 total,  14444 used,  886672 free,  171376 cached
❶ PID to renice
  |
PID   USER  PR  NI  VIRT   RES   SHR   S  %CPU  %MEM  TIME      COMMAND
5451  root  20  0   1577m  19m   14m   S  5.3   3.9   42:46.26  OLLYDBG.EXE
2766  root  20  0   55800  20m   5480  S  2.6   4.0   1:01.42   Xorg
5456  root  20  0   6356   4272  1780  S  1.3   0.8   13:21.69  wineserver
7     root  20  0   0      0     0     S  0.3   0.0   0:30.12   rcu_sched
5762  root  20  0   174m   20m   17m   S  0.3   4.1   0:04.74   gnome-terminal
```

Листинг 6-4: Изменение значения `nice` при использовании `top`

Когда я нажимаю клавишу R, меня просят указать PID (1) текстом `renice PID [value] to value`. Затем вывод должен измениться, чтобы отразить новые приоритеты.

Завершение процессов

Иногда процесс потребляет слишком много системных ресурсов, демонстрирует необычное поведение или, в худшем случае, зависает. Процесс, демонстрирующий такое поведение, часто называют zombie process. Для вас самым проблемным симптомом, вероятно, будет напрасная трата ресурсов, используемых зомби, которые можно было бы лучше выделить полезным процессам.

Когда вы определили проблемный процесс, можно остановить его командой `kill`. Существует много разных способов "убить" программу, и у каждого есть собственный номер kill.

У команды `kill` есть 64 разных kill signals, и каждый делает что-то немного другое. Здесь мы сосредоточимся на нескольких, которые, скорее всего, окажутся самыми полезными. Синтаксис команды `kill` - `kill-signal PID`, где ключ signal необязателен. Если вы не предоставите флаг сигнала, по умолчанию используется `SIGTERM`. Таблица 6-1 перечисляет распространенные сигналы `kill`.

Таблица 6-1: Часто используемые сигналы `kill`

Имя сигнала	Номер параметра	Описание
`SIGHUP`	1	Известен как сигнал Hangup (HUP). Он останавливает указанный процесс и перезапускает его с тем же PID.
`SIGINT`	2	Сигнал Interrupt (INT). Это слабый сигнал завершения, который не гарантированно сработает, но работает в большинстве случаев.
`SIGQUIT`	3	Известен как дамп памяти. Он завершает процесс, сохраняет информацию о процессе в памяти, а затем сохраняет эту информацию в текущем рабочем каталоге в файл с именем `core`. Причины для этого выходят за рамки книги.
`SIGTERM`	15	Сигнал Termination (TERM). Это сигнал завершения по умолчанию для команды `kill`.
`SIGKILL`	9	Абсолютный сигнал завершения. Он принудительно останавливает процесс, отправляя ресурсы процесса в специальное устройство `/dev/null`.

С помощью команды `top` можно определить, какие процессы используют слишком много ресурсов. Часто эти процессы будут легитимными, но могут быть и вредоносные процессы, забирающие ресурсы, которые вы захотите завершить.

Если вы просто хотите перезапустить процесс сигналом HUP, введите параметр `-1` с `kill`:

```
kali >kill -1 6996
```

В случае зомби-процесса или вредоносного процесса вы, вероятно, захотите отправить процессу сигнал `kill -9`, абсолютный сигнал завершения. Это гарантирует, что процесс будет завершен.

```
kali >kill -9 6996
```

Если вы не знаете PID процесса, можно использовать команду `killall`, чтобы завершить процесс. Эта команда принимает имя процесса, а не PID, как аргумент.

Например, гипотетический `zombieprocess` можно завершить так:

```
kali >killall -9 zombieprocess
```

Наконец, процесс также можно завершить в команде `top`. Просто нажмите клавишу `K`, а затем введите PID нарушающего процесса.

Запуск процессов в фоне

В Linux, работаете ли вы из командной строки или GUI, вы работаете внутри оболочки. Все запускаемые команды выполняются внутри этой оболочки, даже если они запускаются из графического интерфейса. Когда вы выполняете команду, оболочка ждет, пока команда завершится, прежде чем предложить новое приглашение командной строки.

Иногда вам может понадобиться, чтобы процесс выполнялся в фоне, а не заставлял ждать завершения в этом терминале. Например, допустим, мы хотим работать над сценарием в текстовом редакторе и поэтому вызвали текстовый редактор (`leafpad`), введя следующее:

```
kali >leafpad newscript
```

В этом случае оболочка bash откроет текстовый редактор `leafpad`, чтобы создать `newscript`. Пока мы работаем в текстовом редакторе, терминал занят запуском текстового редактора. Если вернуться к терминалу, мы увидим, что он выполняет наш текстовый редактор и что у нас нет нового приглашения, позволяющего вводить другие команды.

Конечно, можно открыть еще один терминал для запуска дополнительных команд, но лучший вариант для экономии ресурсов и места на экране - запустить текстовый редактор в фоне. Запуск процесса в фоне просто означает, что он продолжит выполняться, не нуждаясь в терминале. Так терминал освобождается для других задач.

Чтобы запустить текстовый редактор в фоне, просто добавьте амперсанд (`&`) в конец команды:

```
kali >leafpad newscript &
```

Теперь, когда текстовый редактор откроется, терминал вернет новое приглашение командной строки, так что мы сможем вводить другие команды в системе и одновременно редактировать `newscript`. Это эффективно для любого процесса, который может выполняться значительное время, когда вам нужен терминал. Как хакеру вам это пригодится для запуска нескольких терминалов с несколькими задачами, чтобы экономить ресурсы и пространство экрана.

Перемещение процесса на передний план

Если вы хотите переместить процесс, выполняющийся в фоне, на передний план, можно использовать команду `fg` (foreground). Команда `fg` требует PID процесса, который вы хотите вернуть на передний план, как показано далее.

```
kali >fg 1234
```

Если вы не знаете PID, можно использовать команду `ps`, чтобы найти его.

Планирование процессов

И системным администраторам Linux, и хакерам часто нужно планировать запуск процессов в определенное время дня. Например, системный администратор может захотеть запланировать резервное копирование системы на каждую субботу в 2:00 ночи. Хакер может захотеть настроить сценарий на регулярный запуск разведки, поиск открытых портов или уязвимостей. В Linux этого можно добиться как минимум двумя способами: с помощью `at` и `crond`.

Команда `at` - это демон, фоновый процесс, полезный для планирования однократного запуска задания в будущем. `crond` больше подходит для планирования задач, которые должны выполняться каждый день, неделю или месяц, и мы подробно рассмотрим его в главе 16.

Мы используем демон `at`, чтобы запланировать выполнение команды или набора команд в будущем. Синтаксис прост: команда `at`, за которой следует время выполнения процесса. Аргумент времени можно передавать в разных форматах. Таблица 6-2 содержит самые распространенные форматы времени `at`.

Таблица 6-2: Форматы времени, принимаемые командой `at`

Формат времени	**Значение**
`at 7:20pm`	Запланировано на 19:20 текущего дня
`at 7:20pm June 25`	Запланировано на 19:20 25 июня
`at noon`	Запланировано на полдень текущего дня
`at noon June 25`	Запланировано на полдень 25 июня
`at tomorrow`	Запланировано на завтра
`at now + 20 minutes`	Запланировано через 20 минут от текущего времени
`at now + 10 hours`	Запланировано через 10 часов от текущего времени
`at now + 5 days`	Запланировано через пять дней от текущей даты
`at now + 3 weeks`	Запланировано через три недели от текущей даты
`at 7:20pm 06/25/2019`	Запланировано на 19:20 25 июня 2019 года

Когда вы вводите демон `at` с указанным временем, `at` переходит в интерактивный режим, и вас встречает приглашение `at>`. Здесь вы вводите команду, которую хотите выполнить в указанное время:

```
kali >at 7:20am
at> /root/myscanningscript
```

Этот фрагмент кода запланирует выполнение `myscanningscript` сегодня в 7:20 утра.

Итоги

Управление процессами в Linux - ключевой навык для каждого пользователя Linux и хакера. Вы должны уметь просматривать, находить, завершать, приоритизировать и планировать процессы, чтобы оптимально управлять своим экземпляром Linux. Хакеру часто потребуется находить на цели процессы, которые он хочет завершить, например антивирусное ПО или файрвол. Ему также нужно будет управлять несколькими процессами в атаке и назначать им приоритеты.

Упражнения

Прежде чем переходить к главе 7, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Выполните команду `ps` с параметрами `aux` в своей системе и отметьте, какой процесс первый, а какой последний.
2. Выполните команду `top` и отметьте два процесса, использующих наибольший объем ресурсов.
3. Используйте команду `kill`, чтобы завершить процесс, который использует больше всего ресурсов.
4. Используйте команду `renice`, чтобы снизить приоритет выполняющегося процесса до `+19`.
5. Создайте сценарий с именем `myscanning` (содержимое не важно) в текстовом редакторе, а затем запланируйте его запуск на следующую среду в 1:00 ночи.

Глава 7. Управление пользовательскими переменными окружения

Чтобы получить максимум от своей хакерской системы Linux, нужно понимать переменные окружения и уметь управлять ими ради оптимальной производительности, удобства и даже скрытности. Однако среди областей, которые вызывают трудности у новичков в Linux, управление пользовательскими переменными окружения может быть самой сложной для освоения.

Технически есть два типа переменных: переменные оболочки и переменные окружения. Переменные окружения - это общесистемные переменные, встроенные в вашу систему и интерфейс; они управляют тем, как система выглядит, действует и "ощущается" пользователем, и наследуются любыми дочерними оболочками или процессами. Переменные оболочки, с другой стороны, обычно перечисляются в нижнем регистре и действительны только в той оболочке, где они заданы. Чтобы не перегружать объяснениями, в этой главе я охвачу только несколько самых базовых и полезных навыков работы с переменными окружения и переменными оболочки и не буду слишком глубоко погружаться в различия между ними.

Переменные - это просто строки в парах ключ-значение. Обычно каждая пара выглядит как `KEY=value`. В случаях, где есть несколько значений, они выглядят как `KEY=value1:value2`. Как и в большинстве вещей в Linux, если в значении есть пробелы, его нужно заключить в кавычки. В Kali Linux ваше окружение - это ваша оболочка bash. У каждого пользователя, включая `root`, есть набор переменных окружения по умолчанию, которые определяют, как система выглядит, действует и ощущается. Вы можете менять значения этих переменных, чтобы заставить систему работать эффективнее, настроить рабочую среду под свои индивидуальные нужды и потенциально заметать следы, если это понадобится.

Просмотр и изменение переменных окружения

Вы можете просмотреть все свои переменные окружения по умолчанию, введя `env` в терминале из любого каталога:

```
kali >env
XDG_VTNR=7
SSHAGENT_PID=922
XDG_SESSION_ID=2
XDG_GREETER_DATA_DIR=/var/lib/lightdm/data/root
GLADE_PIXMAP_PATH=:echo
TERM=xterm
SHELL=/bin/bash
--snip--
USER=root
--snip--
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
--snip--
HOME=/root
--snip--
```

Переменные окружения всегда пишутся в верхнем регистре, как `HOME`, `PATH`, `SHELL` и так далее. Это только переменные окружения по умолчанию, которые поставляются с вашей системой. Пользователь также может создавать собственные переменные, и, как вы увидите, чтобы включить их в вывод, нужна другая команда.

Просмотр всех переменных окружения

Чтобы просмотреть все переменные окружения, включая переменные оболочки, локальные переменные и функции оболочки, такие как любые пользовательские переменные и псевдонимы команд, используйте команду `set`. Эта команда перечислит все переменные окружения, уникальные для вашей системы, что в большинстве случаев даст настолько длинный вывод, что вы не сможете просмотреть его целиком на одном экране. Можно запросить просмотр каждой переменной построчно и в более доступном виде, используя `set` и передавая его по каналу в команду `more`:

```
kali >set | more
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdlist:complete_fullquote:expand_aliases:extglob.....
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
--snip--
```

Теперь список переменных будет заполнять один экран, строка за строкой, а затем останавливаться. Когда вы нажимаете `ENTER`, терминал продвигается к следующей строке, переводя вас к следующей переменной, так что можно прокручивать список, нажимая или удерживая `ENTER`. Как вы, возможно, помните из главы 2, всякий раз когда вы используете команду `more` для вывода, можно ввести `q`, чтобы выйти и вернуться к приглашению командной строки.

Фильтрация конкретных переменных

Хотя использование `set` с `more` дает более управляемые результаты, чем просмотр огромного блока имен переменных, который вы получаете только с `set`, это все равно может быть довольно утомительно, если вы ищете конкретную переменную. Вместо этого можно использовать команду фильтрации `grep`, чтобы найти интересующую переменную.

Возьмем в качестве примера переменную `HISTSIZE`. Эта переменная содержит максимальное количество команд, которое будет хранить файл истории команд. Эти команды - все те, которые вы ранее вводили в командную строку в этом сеансе и которые можно вызвать клавишами со стрелками вверх и вниз. Обратите внимание, что `HISTSIZE` хранит не сами команды, а только их количество, которое может быть сохранено.

Передайте вывод `set` по каналу в `grep`, чтобы найти переменную `HISTSIZE`:

```
kali >set | grep HISTSIZE
HISTSIZE=1000
```

Как видите, эта команда находит переменную `HISTSIZE` и отображает ее значение. Значение этой переменной по умолчанию, вероятно, установлено в `1000` в вашей системе. Это указывает, что терминал по умолчанию будет хранить последние 1000 команд.

Изменение значений переменных на время сеанса

Теперь посмотрим, как изменить значение переменной. Как отмечалось, переменная `HISTSIZE` содержит значение количества команд, которые нужно хранить в файле истории. Иногда вы не захотите, чтобы система сохраняла прошлые команды, возможно, потому что не хотите оставлять никаких доказательств своей активности в собственной системе или целевой системе. В таком случае можно установить переменную `HISTSIZE` в `0`, чтобы система не хранила ни одну из ваших прошлых команд.

Поскольку у этой переменной одно значение, чтобы изменить ее, вы присваиваете ей новое значение привычным способом, показанным в листинге 7-1.

```
kali >HISTSIZE=0
```

Листинг 7-1: Изменение значения `HISTSIZE`

Теперь, когда вы попытаетесь использовать клавиши со стрелками вверх и вниз для вызова своих команд, ничего не произойдет, потому что система больше их не хранит. Это скрытно, хотя и может быть неудобно.

Постоянное сохранение изменений значений переменных

Когда вы меняете переменную окружения, это изменение происходит только в конкретном окружении; в данном случае это сеанс оболочки bash. Это означает, что когда вы закрываете терминал, все внесенные изменения теряются, а значения возвращаются к значениям по умолчанию. Если вы хотите сделать изменения постоянными, нужно использовать команду `export`. Эта команда экспортирует новое значение из вашего текущего окружения (оболочки bash) в остальную систему, делая его доступным в каждом окружении, пока вы снова не измените и не экспортируете его.

Переменные - это строки, поэтому, если вы склонны к осторожности, неплохая идея - сохранить содержимое переменной в текстовый файл, прежде чем изменять ее. Например, поскольку мы собираемся изменить переменную `PS1`, которая управляет информацией, отображаемой в приглашении, сначала выполните следующую команду, чтобы сохранить существующие значения в текстовый файл в домашнем каталоге текущего пользователя:

```
kali >echo $HISTSIZE > ~/valueofHISTSIZE.txt
```

Так вы всегда сможете отменить свои изменения. Если хотите быть еще осторожнее и создать текстовый файл со всеми текущими настройками, можно сохранить вывод команды `set` в текстовый файл такой командой:

```
kali >set > ~/valueofALLon01012017.txt
```

После изменения переменной, как мы сделали в листинге 7-1, можно сделать изменение постоянным, введя `export`, а затем имя измененной переменной:

```
kali >export HISTSIZE
```

Теперь переменная `HISTSIZE` по-прежнему будет установлена в `0`, когда вы покинете это окружение и войдете в другое. Если вы хотите сбросить переменную `HISTSIZE` до `1000`, просто введите:

```
kali >HISTSIZE=1000
kali >export HISTSIZE
```

Этот фрагмент кода установит значение переменной `HISTSIZE` в `1000` и экспортирует его во все ваши окружения.

Изменение приглашения оболочки

Приглашение оболочки, еще одна переменная окружения, предоставляет полезную информацию, например пользователя, под которым вы работаете, и каталог, в котором сейчас находитесь. Приглашение оболочки по умолчанию в Kali имеет следующий формат:

```
username@hostname:current_directory
```

Если вы работаете как пользователь `root`, это преобразуется в следующее приглашение по умолчанию:

```
root@kali:current_directory
```

Можно изменить имя в приглашении оболочки по умолчанию, задав значение переменной `PS1`. У переменной `PS1` есть набор заполнителей для информации, которую вы хотите отображать в приглашении, включая следующие:

- `\u` - имя текущего пользователя;
- `\h` - hostname;
- `\W` - базовое имя текущего рабочего каталога.

Это очень полезно, если у вас открыты оболочки на нескольких системах или вы вошли под несколькими учетными записями. Задавая разные значения `\u` и `\h` для разных оболочек или учетных записей, можно с первого взгляда понять, кто вы и какая у вас текущая система.

Давайте немного развлечемся и изменим приглашение в терминале. Например, можно ввести следующее:

```
kali >PS1="Лучший хакер мира: #"
Лучший хакер мира: #
```

Теперь каждый раз при использовании этого терминала вам будут напоминать, что вы "лучший хакер мира". Но любой следующий терминал, который вы откроете, по-прежнему будет иметь приглашение командной строки по умолчанию, потому что переменная `PS1` хранит значения только для вашего терминального сеанса. Помните: пока вы не экспортируете переменную, она действительна только для этого сеанса. Если вам очень нравится это новое приглашение командной строки и вы хотите видеть его в каждом терминале, нужно экспортировать его:

```
kali >export PS1
```

Это сделает изменение постоянным во всех сеансах.

Как насчет еще небольшого развлечения? Допустим, вы действительно хотите, чтобы терминал выглядел как приглашение Windows `cmd`. В этом случае можно изменить имя приглашения на `C:` и сохранить `\w`, чтобы приглашение показывало текущий каталог, как показано в листинге 7-2.

```
kali >export PS1='C:\w> '
C:/tmp>
```

Листинг 7-2: Изменение приглашения и отображение текущего каталога

В целом отображение текущего каталога в приглашении может быть полезным, особенно для новичка, так что это стоит учитывать при изменении переменной `PS1`.

Изменение PATH

Одна из самых важных переменных в вашем окружении - переменная PATH, которая управляет тем, где в системе оболочка будет искать вводимые вами команды, такие как `cd`, `ls` и `echo`. Большинство команд расположены в подкаталоге `sbin` или `bin`, например `/usr/local/sbin` или `/usr/local/bin`. Если оболочка bash не найдет команду ни в одном из каталогов переменной PATH, она вернет ошибку `command not found`, даже если эта команда действительно существует в каталоге, не входящем в PATH.

Можно узнать, какие каталоги хранятся в переменной PATH, используя `echo` для ее содержимого:

```
kali >echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Это каталоги, в которых терминал будет искать любую команду. Например, когда вы вводите `ls`, система знает, что нужно искать команду `ls` в каждом из этих каталогов, а когда находит `ls`, выполняет ее.

Каждый каталог отделяется двоеточием (`:`), и не забывайте добавлять символ содержимого `$` к PATH.

Добавление к переменной PATH

Вы, вероятно, видите, почему важно знать, что находится в переменной PATH: если вы загрузили и установили новый инструмент, скажем `newhackingtool`, в каталог `/root/newhackingtool`, вы могли бы использовать команды этого инструмента только тогда, когда находитесь в этом каталоге, потому что этот каталог не входит в переменную PATH. Каждый раз, когда вы захотите использовать этот инструмент, сначала придется переходить в `/root/newhackingtool`, что немного неудобно, если вы хотите часто использовать инструмент.

Чтобы иметь возможность использовать этот новый инструмент из любого каталога, нужно добавить каталог, содержащий этот инструмент, в переменную PATH.

Чтобы добавить `newhackingtool` в переменную PATH, введите следующее:

```
kali >PATH=$PATH:/root/newhackingtool
```

Это присваивает новой переменной PATH исходную переменную PATH плюс каталог `/root/newhackingtool`, так что переменная содержит все, что содержала раньше, плюс каталог нового инструмента.

Если снова изучить содержимое переменной PATH, вы должны увидеть, что этот каталог добавлен в конец PATH:

```
kali >echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root/newhackingtool
```

Теперь можно выполнять приложения `newhackingtool` из любого места системы, а не переходить в его каталог. Оболочка bash будет искать ваш новый инструмент во всех перечисленных каталогах!

> **Примечание.** Добавление к PATH может быть полезным приемом для каталогов, которые вы часто используете, но будьте осторожны и не добавляйте слишком много каталогов в переменную PATH. Поскольку системе придется просматривать каждый каталог в PATH, чтобы найти команды, добавление большого количества каталогов может замедлить терминал и ваш хакинг.

Как не надо добавлять к переменной PATH

Одна ошибка, которую часто совершают новые пользователи Linux, - назначать новый каталог, например `/root/newhackingtool`, непосредственно переменной `PATH` следующим образом:

```
kali >PATH=/root/newhackingtool
kali >echo $PATH
/root/newhackingtool
```

Если использовать эту команду, переменная `PATH` будет содержать только каталог `/root/newhackingtool` и больше не будет содержать каталоги системных бинарных файлов, такие как `/bin`, `/sbin` и другие, где находятся критически важные команды. Когда затем вы попытаетесь использовать любую из системных команд, получите ошибку `command not found`, как показано далее, если только сначала не перейдете в каталог системных бинарных файлов при выполнении команды:

```
kali >cd
bash: cd: command not found
```

Помните, что нужно добавлять к переменной `PATH`, а не заменять ее. Если сомневаетесь, сохраните содержимое переменной где-нибудь перед изменением.

Создание пользовательской переменной

В Linux можно создавать собственные пользовательские переменные, просто присваивая значение новой переменной с выбранным вами именем. Это может быть полезно, когда вы занимаетесь более продвинутым написанием сценариев оболочки или часто используете длинную команду, которую надоело вводить снова и снова.

Синтаксис прямолинеен: введите имя переменной, затем символ присваивания (=), а затем значение, которое нужно поместить в переменную:

```
kali >MYNEWVARIABLE="Хакинг - самый ценный набор навыков в XXI
веке"
```

Это присваивает строку переменной `MYNEWVARIABLE`. Чтобы увидеть значение этой переменной, используйте команду `echo` и символ содержимого `$` с именем переменной, как мы делали раньше:

```
kali >echo $MYNEWVARIABLE
Хакинг - самый ценный набор навыков в XXI веке
```

Как и наши системные переменные окружения, пользовательские переменные нужно экспортировать, чтобы они сохранялись в новых сеансах.

Если вы хотите удалить эту новую переменную или любую переменную, используйте команду `unset`. Однако всегда думайте, прежде чем удалять системную переменную, потому что после этого ваша система, вероятно, будет работать совершенно иначе.

```
kali >unset MYNEWVARIABLE
kali >echo $MYNEWVARIABLE
kali >
```

Как видите, когда вы вводите `unset MYNEWVARIABLE`, вы удаляете переменную вместе с ее значением. Если использовать `echo` для той же переменной, Linux теперь вернет пустую строку.

Итоги

Переменные окружения могут казаться чуждыми, но стоит с ними познакомиться. Они управляют тем, как ваша рабочая среда в Linux выглядит, действует и ощущается. Вы можете управлять этими переменными, чтобы настроить окружение под свои нужды: изменять их, экспортировать и даже создавать собственные. В некоторых случаях они могут быть полезны для заметания следов хакера.

Упражнения

Прежде чем переходить к главе 8, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Просмотрите все свои переменные окружения с помощью команды `more`.
2. Используйте команду `echo`, чтобы посмотреть переменную `HOSTNAME`.
3. Найдите способ заменить косую черту (`/`) на обратную косую черту (`\`) в примере фальшивого Microsoft `cmd` для `PS1` (см. листинг 7-2).
4. Создайте переменную с именем `MYNEWVARIABLE` и поместите в нее свое имя.
5. Используйте `echo`, чтобы посмотреть содержимое `MYNEWVARIABLE`.
6. Экспортируйте `MYNEWVARIABLE`, чтобы она была доступна во всех окружениях.
7. Используйте команду `echo`, чтобы посмотреть содержимое переменной `PATH`.
8. Добавьте свой домашний каталог в переменную `PATH`, чтобы любые бинарные файлы в вашем домашнем каталоге можно было использовать из любого каталога.
9. Измените переменную `PS1` на `Величайший хакер мира:`.

Глава 8. Сценарии bash

Любой уважающий себя хакер должен уметь писать сценарии. Если уж на то пошло, любой уважающий себя администратор Linux тоже должен уметь писать сценарии. Хакерам часто нужно автоматизировать команды, иногда из нескольких инструментов, и эффективнее всего это делается с помощью коротких программ, которые они пишут сами.

В этой главе мы построим несколько простых сценариев оболочки bash, чтобы вы начали работать со сценариями. По мере продвижения мы будем добавлять возможности и функции, в итоге построив сценарий, способный находить потенциальные цели атаки в диапазоне IP-адресов.

Чтобы стать элитным хакером, вам также нужна способность писать сценарии на одном из широко используемых скриптовых языков, таких как Ruby (эксплойты Metasploit написаны на Ruby), Python (многие хакерские инструменты - это сценарии Python) или Perl (Perl - лучший скриптовый язык для манипуляции текстом). Краткое введение в сценарии Python я даю в главе 17.

Ускоренный курс bash

Оболочка - это интерфейс между пользователем и операционной системой, который позволяет манипулировать файлами и запускать команды, утилиты, программы и многое другое. Преимущество оболочки в том, что вы выполняете эти задачи непосредственно с компьютера, а не через абстракцию вроде GUI, что позволяет подстраивать задачу под свои нужды. Для Linux доступен ряд разных оболочек, включая оболочку Korn, оболочку Z, оболочку C и оболочку Bourne-again, более известную как bash.

Поскольку оболочка bash доступна почти во всех дистрибутивах Linux и UNIX, включая macOS и Kali, мы будем использовать исключительно оболочку bash.

Оболочка bash может запускать любые системные команды, утилиты или приложения, которые может запускать обычная командная строка, но также включает собственные встроенные команды. Таблица 8-1 дальше в главе даст вам справку по некоторым полезным командам, находящимся внутри оболочки bash.

В предыдущих главах вы использовали команды `cd`, `pwd`, `set` и `umask`. В этом разделе вы будете использовать еще две команды: `echo`, впервые использованную в главе 7, которая отображает сообщения на экране, и `read`, которая считывает данные и сохраняет их в другом месте. Одно только освоение этих двух команд позволит вам построить простой, но мощный инструмент.

Для создания сценариев оболочки вам понадобится текстовый редактор. Вы можете использовать любой текстовый редактор Linux, который вам больше нравится, включая `vi`, `vim`, `emacs`, `gedit`, `kate` и так далее. В этих уроках я буду использовать Leafpad, как и в предыдущих главах. Использование другого редактора не должно влиять на ваш сценарий или его функциональность.

Ваш первый сценарий: "Hello, Hackers-Arise!"

Для первого сценария мы начнем с простой программы, которая возвращает на экран сообщение `Hello, Hackers-Arise!`. Откройте текстовый редактор, и начнем.

Сначала нужно сообщить операционной системе, какой интерпретатор вы хотите использовать для сценария. Для этого введите shebang - сочетание знака решетки и восклицательного знака:

```
#!
```

Затем после shebang (`#!`) следует `/bin/bash`, чтобы указать, что вы хотите, чтобы операционная система использовала интерпретатор оболочки bash. Как вы увидите в следующих главах, shebang можно также использовать для других интерпретаторов, таких как Perl или Python. Здесь вы хотите использовать интерпретатор bash, поэтому введите следующее:

```
#!/bin/bash
```

Далее введите команду `echo`, которая говорит системе просто повторить, или echo, на монитор все, что следует за командой.

В этом случае мы хотим, чтобы система вывела нам `Hello, Hackers-Arise!`, как сделано в листинге 8-1. Обратите внимание, что текст или сообщение, которое мы хотим вывести, должно быть в двойных кавычках.

```
#!/bin/bash
# Это мой первый сценарий bash. Пожелайте мне удачи.
echo "Hello, Hackers-Arise!"
```

Листинг 8-1: Ваш сценарий "Hello, Hackers-Arise!"

Здесь вы также видите строку, перед которой стоит знак решетки (#). Это комментарий, то есть заметка, которую вы оставляете себе или любому другому читателю кода, чтобы объяснить, что делает сценарий. Программисты используют комментарии в каждом языке программирования. Эти комментарии не читаются и не выполняются интерпретатором, поэтому вам не нужно бояться испортить ими код. Они видны только людям. Оболочка bash понимает, что строка является комментарием, если она начинается с символа #.

Теперь сохраните этот файл как `HelloHackersArise` без расширения и выйдите из текстового редактора.

Установка прав выполнения

По умолчанию только что созданный сценарий bash не является исполняемым даже для вас, владельца. Посмотрим на права нашего нового файла в командной строке: используйте `cd`, чтобы перейти в каталог, а затем введите `ls -l`. Это должно выглядеть примерно так:

```
kali >ls -l
--snip--
-rw-r--r-- 1 root root 42 Oct 22 14:32 HelloHackersArise
--snip--
```

Как видите, у нового файла права `rw-r--r--` (644). Как вы узнали в главе 5, это означает, что у владельца файла есть только права чтения (`r`) и записи (`w`), но нет права выполнения (`x`). У группы и всех остальных пользователей есть только права чтения. Нам нужно выдать себе права выполнения, чтобы запустить этот сценарий. Мы меняем права командой `chmod`, как вы видели в главе 5. Чтобы дать владельцу, группе и всем остальным права выполнения, введите следующее:

```
kali >chmod 755 HelloHackersArise
```

Теперь, когда мы выполняем длинный листинг файла, видно, что у нас есть права выполнения:

```
kali >ls -l
--snip--
-rwx r-x r-x 1 root root 42 Oct 22 14:32 HelloHackersArise
--snip--
```

Сценарий теперь готов к выполнению!

Запуск HelloHackersArise

Чтобы запустить наш простой сценарий, введите следующее:

```
kali >./HelloHackersArise
```

`./` перед именем файла говорит системе, что мы хотим выполнить этот сценарий в файле `HelloHackersArise` из текущего каталога. Это также говорит системе: если в другом каталоге есть другой файл с именем `HelloHackersArise`, игнорировать его и запускать только `HelloHackersArise` в текущем каталоге. Может показаться маловероятным, что в системе есть другой файл с таким именем, но хорошая практика - использовать `./` при выполнении файлов, потому что это локализует выполнение файла в текущем каталоге, а во многих каталогах будут повторяющиеся имена файлов, такие как `start` и `setup`.

Когда мы нажимаем `ENTER`, наш очень простой сценарий возвращает сообщение на монитор:

```
Hello, Hackers-Arise!
```

Успех! Вы только что завершили свой первый сценарий оболочки!

Добавление функциональности с помощью переменных и пользовательского ввода

Итак, теперь у нас есть простой сценарий. Все, что он делает, - выводит сообщение в стандартный вывод. Если мы хотим создавать более продвинутые сценарии, скорее всего, понадобится добавить переменные.

Переменная - это область хранения, которая может удерживать что-то в памяти. Этим "чем-то" могут быть буквы или слова (строки) или числа. Она называется переменной потому, что удерживаемые в ней значения можно изменять; это чрезвычайно полезная возможность для добавления функциональности в сценарий.

В следующем сценарии мы добавим функциональность, которая попросит пользователя ввести имя, поместит все, что он введет, в переменную, затем попросит пользователя указать главу этой книги, на которой он находится, и поместит этот ввод с клавиатуры в переменную. После этого мы выведем пользователю приветственное сообщение, включающее его имя и номер главы.

Откройте новый файл в текстовом редакторе и введите сценарий, показанный в листинге 8-2.

```
(1) #!/bin/bash
(2) # Это ваш второй сценарий bash. В нем вы запрашиваете ввод /
```

```
# пользователя, помещаете ввод в переменную и /
# отображаете содержимое переменной в строке.
(3) echo "What is your name?"
read name
(4) echo "На какой вы главе в Основах Linux для хакеров?"
read chapter
(5) echo "Добро пожаловать," $name ", в главу" $chapter "Основ Linux для хакеров!"
```

Листинг 8-2: Простой сценарий, использующий переменные

Мы начинаем с `#!/bin/bash`, чтобы сообщить системе, что хотим использовать интерпретатор bash для этого сценария (1). Затем добавляем комментарий, описывающий сценарий и его функциональность (2). После этого мы запрашиваем имя пользователя и просим интерпретатор прочитать ввод и поместить его в переменную, которую называем `name` (3). Затем мы просим пользователя ввести номер главы, которую он сейчас проходит в этой книге, и снова читаем ввод с клавиатуры в переменную, на этот раз с именем `chapter` (4).

В последней строке мы строим строку вывода, которая приветствует читателя по имени в главе, на которой он находится (5). Мы используем команду `echo` и передаем текст, который хотим отобразить на экране, в двойных кавычках. Затем, чтобы подставить имя и номер главы, введенные пользователем, мы добавляем переменные в те места, где они должны появиться в сообщении. Как отмечалось в главе 7, чтобы использовать значения, содержащиеся в переменных, нужно поставить перед именем переменной символ `$`.

Сохраните этот файл как `WelcomeScript.sh`. Расширение `.sh` - соглашение для файлов сценариев. Возможно, вы заметили, что ранее мы не включали расширение; оно не строго обязательно, и если оставить расширение, файл по умолчанию сохранится как файл сценария оболочки.

Теперь запустим этот сценарий. Не забудьте сначала выдать себе право выполнения с помощью `chmod`; иначе операционная система отругает вас сообщением `Permission denied`.

```
kali >./WelcomeScript.sh
What is your name?
OccupytheWeb
На какой вы главе в Основах Linux для хакеров?
8
Добро пожаловать, OccupytheWeb, в главу 8 Основ Linux для хакеров!
```

Как видите, сценарий принял ввод от пользователя, поместил его в переменные, а затем использовал этот ввод для создания приветствия пользователя.

Это простой сценарий, но он научил вас использовать переменные и принимать ввод с клавиатуры. Оба эти понятия критически важны в написании сценариев, и вам понадобится использовать их в более сложных сценариях в будущем.

Ваш самый первый хакерский сценарий: сканирование открытых портов

Теперь, когда у вас есть базовые навыки написания сценариев, перейдем к немного более продвинутым сценариям, имеющим реальное применение в хакинге. Мы используем пример из мира хакинга "черной шляпы". Хакеры "черной шляпы" - это те, у кого злонамеренные намерения, например кража номеров кредитных карт или порча сайтов. Хакеры "белой шляпы" - это те, у кого хорошие намерения, например помощь разработчикам программ или системным администраторам в повышении безопасности систем. Хакеры "серой шляпы" - это те, кто склонен перемещаться между этими двумя крайностями.

Прежде чем продолжать, нужно познакомиться с простым, но важным инструментом nmap, который установлен в Kali по умолчанию. Вы, вероятно, уже слышали это имя; nmap используется для исследования системы, чтобы понять, подключена ли она к сети, и выяснить, какие порты открыты. По найденным открытым портам можно предположить, какие службы работают на целевой системе. Это критически важный навык для любого хакера или системного администратора.

В самой простой форме синтаксис запуска сканирования nmap выглядит так:

```
nmap <type of scan><target IP><optionally, target port>
```

Не слишком сложно. Самое простое и надежное сканирование nmap - сканирование TCP-подключением, обозначаемое ключом -sT в nmap. Поэтому, если бы вы хотели просканировать IP-адрес 192.168.181.1 с помощью TCP-сканирования, нужно было бы ввести следующее:

```
nmap -sT 192.168.181.1
```

Если пойти на шаг дальше и выполнить TCP-сканирование адреса 192.168.181.1, проверяя, открыт ли порт 3306 (порт MySQL по умолчанию), можно ввести:

```
nmap -sT 192.168.181.1 -p 3306
```

Здесь -p обозначает порт, который вы хотите сканировать. Попробуйте выполнить это сейчас в своей системе Kali.

Наша задача

На момент написания этой книги в федеральной тюрьме США отбывает срок хакер по имени Макс Батлер, также известный в хакерском мире как Max Vision. Макс был чем-то вроде хакера "серой шляпы". Днем он был специалистом по IT-безопасности в Кремниевой долине, а ночью крал и продавал номера кредитных карт на черном рынке. В какой-то момент он управлял крупнейшим в мире черным рынком кредитных карт CardersMarket. Теперь Макс отбывает 13-летний срок и одновременно помогает группе реагирования на компьютерные инциденты (CERT) в Питтсбурге защищаться от хакеров.

За несколько лет до того, как Макса поймали, он понял, что в системе Aloha Point of Sale (POS), используемой многими небольшими ресторанами, встроен технический черный ход для поддержки. В этом случае черный ход позволял технической поддержке помогать клиентам. Техподдержка Aloha могла получать доступ к системе конечного пользователя через порт 5505, чтобы оказать помощь, когда пользователь обращался за поддержкой. Макс понял, что если найдет систему, подключенную к интернету и использующую Aloha POS, то сможет получить доступ к системе с привилегиями системного администратора через порт 5505. Макс смог войти во многие такие системы и украсть десятки тысяч номеров кредитных карт.

В конце концов Макс захотел найти каждую систему, у которой открыт порт 5505, чтобы перейти от кражи тысяч номеров кредитных карт к краже миллионов. Макс решил написать сценарий, который сканировал бы миллионы IP-адресов в поисках систем с открытым портом 5505. Конечно, у большинства систем порт 5505 не открыт, поэтому если он был открыт, они, вероятно, запускали обреченную Aloha POS. Он мог запускать этот сценарий днем на работе, а ночью взламывать системы, определенные как имеющие открытый порт 5505.

Наша задача - написать сценарий, который будет почти идентичен сценарию Макса, но вместо сканирования порта 5505, как делал Макс, наш сценарий будет искать системы, подключенные к вездесущей онлайн-базе данных MySQL. MySQL - база данных с открытым исходным кодом, используемая за миллионами веб-сайтов; мы будем работать с MySQL в главе 12. По умолчанию MySQL использует порт 3306. Базы данных - это "золотое руно", за которым охотится почти каждый хакер "черной шляпы", поскольку они часто содержат номера кредитных карт и персональные

идентифицирующие данные (PII), имеющие большую ценность на черном рынке.

Простой сканер

Прежде чем написать сценарий для сканирования публичных IP по всему интернету, возьмемся за куда меньшую задачу. Вместо сканирования всего мира сначала напишем сценарий для сканирования порта `3306` в локальной сети, чтобы понять, работает ли наш сценарий вообще. Если работает, мы легко отредактируем его для гораздо более крупной задачи.

В текстовом редакторе введите сценарий, показанный в листинге 8-3.

```
(1) #!/bin/bash
(2) # Этот сценарий предназначен для поиска хостов с установленной MySQL
nmap (3) -sT 192.168.181.0/24 (4) -p 3306 (5) >/dev/null (6) -oG MySQLscan
(7) cat MySQLscan | grep open > MySQLscan2 (8)
cat MySQLscan2
```

Листинг 8-3: Упрощенный сценарий сканера

Мы начинаем с shebang и интерпретатора, который нужно использовать (1). Затем добавим комментарий, объясняющий, что делает сценарий (2).

Теперь используем команду `nmap`, чтобы запросить TCP-сканирование (3) нашей LAN в поисках порта `3306` (4). Обратите внимание, что ваши IP-адреса могут отличаться; в терминале используйте команду `ifconfig` в Linux или команду `ipconfig` в Windows, чтобы определить свой IP-адрес. Чтобы оставаться скрытными, мы также отправляем стандартный вывод `nmap`, который обычно появляется на экране, в специальное место в Linux, где он исчезает (5). Мы делаем это на локальной машине, поэтому это не так важно, но если бы вы использовали сценарий удаленно, вам захотелось бы скрыть вывод `nmap`. Затем мы отправляем вывод сканирования в файл с именем `MySQLscan` в grep-able-формате (6), то есть формате, с которым может работать `grep`.

Следующая строка отображает файл `MySQLscan`, в котором мы сохранили вывод, а затем передает этот вывод по каналу в `grep`, чтобы отфильтровать строки, включающие ключевое слово `open` (7). Затем мы помещаем эти строки в файл с именем `MySQLscan2` (8).

Наконец, вы отображаете содержимое файла `MySQLscan2`. Этот финальный файл должен включать только строки вывода `nmap` с хостами, у которых открыт порт `3306`. Сохраните этот файл как `MySQLscanner.sh` и выдайте себе права выполнения с помощью `chmod 755`.

Выполните сценарий:

```
kali >./MySQLscanner.sh
host: 192.168.181.69 () Ports: 3306/open/tcp//mysql///
```

Как видно, этот сценарий смог определить единственный IP-адрес в моей LAN с запущенной MySQL. Конечно, ваши результаты могут отличаться в зависимости от того, есть ли в локальной сети открытые порты с установками MySQL.

Улучшение сканера MySQL

Теперь мы хотим адаптировать этот сценарий, чтобы он был применим не только к вашей локальной сети. Этот сценарий было бы гораздо удобнее использовать, если бы он мог спрашивать у пользователя диапазон IP-адресов, который нужно сканировать, и порт, который нужно искать, а затем использовать этот ввод. Помните, в разделе "Добавление функциональности с помощью переменных и пользовательского ввода" вы научились запрашивать данные у пользователя и помещать его ввод с клавиатуры в переменную. Давайте посмотрим, как можно использовать переменные, чтобы сделать этот сценарий более гибким и эффективным.

Добавление приглашений и переменных в наш хакерский сценарий

В текстовом редакторе введите сценарий, показанный в листинге 8-4.

```
#!/bin/bash
(1) echo "Введите начальный IP-адрес : "
(2) read FirstIP
(3) echo "Введите последний октет последнего IP-адреса : "
read LastOctetIP
(4) echo "Введите номер порта, который хотите сканировать : "
read port
(5) nmap -sT $FirstIP-$LastOctetIP -p $port >/dev/null -oG MySQLscan
(6) cat MySQLscan | grep open > MySQLscan2
(7) cat MySQLscan2
```

Листинг 8-4: Ваш продвинутый сканер порта MySQL

Первое, что нужно сделать, - заменить указанную подсеть диапазоном IP-адресов. Мы создадим переменную с именем `FirstIP` и вторую переменную с именем `LastOctetIP`, чтобы создать диапазон, а также переменную `port` для номера порта. Последний октет - это последняя группа цифр после третьей точки в IP-адресе. В IP-адресе `192.168.1.101` последний октет - `101`.

> **Примечание.** Имя переменной неважно, но хорошая практика - использовать имя переменной, которое помогает помнить, что она хранит.

Нам также нужно запросить эти значения у пользователя. Это можно сделать с помощью команды `echo`, которую мы использовали в листинге 8-1.

Чтобы получить значение для переменной `FirstIP`, выводим на экран `echo "Введите начальный IP-адрес : "`, запрашивая у пользователя первый IP-адрес, который он хочет сканировать (1). Увидев это приглашение на экране, пользователь введет первый IP-адрес, поэтому нам нужно захватить этот ввод от пользователя.

Это можно сделать командой `read`, за которой следует имя переменной, в которой мы хотим сохранить ввод (2). Эта команда поместит IP-адрес, введенный пользователем, в переменную `FirstIP`. Затем мы сможем использовать это значение `FirstIP` во всем сценарии.

То же самое мы сделаем для переменных `LastOctetIP` (3) и `port` (4), запросив у пользователя информацию и затем использовав команду `read`, чтобы захватить ее.

Далее нужно отредактировать команду `nmap` в нашем сценарии, чтобы она использовала переменные, которые мы только что создали и заполнили. Чтобы использовать значение, хранящееся в переменной, мы просто ставим перед именем переменной `$`, например `$port`. Итак, в (5) мы сканируем диапазон IP-адресов, начиная с первого IP, введенного пользователем, и до второго IP, введенного пользователем, и ищем конкретный порт, введенный пользователем. Мы использовали переменные вместо подсети для сканирования и порта, чтобы определить, что сканировать. Символ перенаправления `>` говорит стандартному выводу `nmap`, который обычно идет

на экран, вместо этого перейти в /dev/null (/dev/null - это просто место, куда отправляют вывод, чтобы он исчез). Затем мы отправляем вывод в grep-able-формате в файл с именем MySQLscan.

Следующая строка остается такой же, как в нашем простом сканере: она выводит содержимое файла MySQLscan, передает его по каналу в grep, где оно фильтруется по строкам, включающим ключевое слово open, а затем отправляет этот вывод в новый файл с именем MySQLscan2 (6). Наконец, мы отображаем содержимое файла MySQLscan2 (7).

Если все работает как ожидается, этот сценарий просканирует IP-адреса от первого введенного адреса до последнего введенного адреса, ища введенный порт, а затем вернет только IP-адреса, у которых указанный порт открыт. Сохраните файл сценария как MySQLscannerAdvanced, не забыв выдать себе право выполнения.

Пример запуска

Теперь мы можем запускать простой сценарий сканера с переменными, которые определяют, какой диапазон IP-адресов и какой порт сканировать, без необходимости редактировать сценарий каждый раз, когда хотим выполнить сканирование:

```
kali >./MySQLscannerAdvanced.sh
Введите начальный IP-адрес :
192.168.181.0
Введите последний IP-адрес :
192.168.181.255
Введите номер порта, который хотите сканировать :
3306
Host: 192.168.181.254 ()Ports:3306/open/tcp//mysql//
```

Сценарий запрашивает у пользователя первый IP-адрес, последний IP-адрес, а затем порт для сканирования. Собрав эту информацию, сценарий выполняет сканирование nmap и создает отчет обо всех IP-адресах в диапазоне, у которых открыт указанный порт.

Как видите, даже самое простое написание сценариев может создать мощный инструмент. Еще больше о написании сценариев вы узнаете в главе 17.

Распространенные встроенные команды bash

Как и обещано, таблица 8-1 дает список некоторых полезных команд, встроенных в bash.

Таблица 8-1: Встроенные команды bash

Команда	Функция
:	Возвращает 0 или true
.	Выполняет сценарий оболочки
bg	Помещает задание в фон
break	Выходит из текущего цикла
cd	Меняет каталог
continue	Возобновляет текущий цикл
echo	Отображает аргументы команды
eval	Вычисляет следующее выражение
exec	Выполняет следующую команду без создания нового процесса

Команда	Функция
`exit`	Завершает оболочку
`export`	Делает переменную или функцию доступной другим программам
`fg`	Переносит задание на передний план
`getopts`	Разбирает аргументы сценария оболочки
`jobs`	Перечисляет фоновые (`bg`) задания
`pwd`	Отображает текущий каталог
`read`	Читает строку из стандартного ввода
`readonly`	Объявляет переменную доступной только для чтения
`set`	Перечисляет все переменные
`shift`	Сдвигает параметры влево
`test`	Вычисляет аргументы
`[`	Выполняет условный test
`times`	Печатает пользовательское и системное время
`trap`	Перехватывает сигнал
`type`	Показывает, как каждый аргумент будет интерпретирован как команда
`umask`	Меняет права по умолчанию для нового файла
`unset`	Удаляет значения из переменной или функции
`wait`	Ожидает завершения фонового процесса

Итоги

Написание сценариев - важнейший навык для любого хакера или системного администратора. Оно позволяет автоматизировать задачи, которые обычно отнимали бы часы вашего времени, а после сохранения сценарий можно использовать снова и снова. Сценарии bash - самая базовая форма написания сценариев, а в главе 17 вы перейдете к сценариям на Python с еще большими возможностями.

Упражнения

Прежде чем переходить к главе 9, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Создайте собственный сценарий приветствия, похожий на наш сценарий `HelloHackersArise`.
2. Создайте сценарий, похожий на `MySQLscanner.sh`, но спроектируйте его для поиска систем с базой данных Microsoft SQL Server на порту `1433`. Назовите его `MSSQLscanner`.
3. Измените этот сценарий `MSSQLscanner`, чтобы он запрашивал у пользователя начальный и конечный IP-адреса и порт для поиска. Затем отфильтруйте все IP-адреса, у которых эти порты закрыты, и отобразите только те, у которых они открыты.

Глава 9. Сжатие и архивирование

Хакерам часто нужно загружать и устанавливать новое программное обеспечение, а также отправлять и загружать несколько сценариев и большие файлы. Эти задачи проще выполнять, если такие файлы сжаты и объединены в один файл. Если вы пришли из мира Windows, то, вероятно, узнаете эту идею по формату `.zip`, который объединяет и сжимает файлы, чтобы сделать их меньше для передачи через интернет или съемные носители. В Linux есть много способов делать это, и в этой главе мы рассмотрим несколько самых распространенных инструментов для таких задач. Мы также рассмотрим команду `dd`, которая позволяет копировать целые диски, включая удаленные файлы на этих дисках.

Что такое сжатие?

Интересная тема сжатия сама по себе могла бы заполнить целую книгу, но для этой книги нам нужно только базовое понимание процесса. Сжатие, как следует из названия, делает данные меньше, тем самым требуя меньше места для хранения и упрощая передачу данных. Для ваших целей начинающего хакера достаточно разделять сжатие на два вида: с потерями и без потерь.

Сжатие с потерями очень эффективно уменьшает размер файлов, но целостность информации теряется. Другими словами, файл после сжатия не является точной копией исходного. Этот тип сжатия отлично подходит для графических, видео- и аудиофайлов, где небольшая разница в файле почти незаметна: `.mp3`, `.mp4`, `.png` и `.jpg` являются алгоритмами сжатия с потерями. Если в файле `.png` изменится один пиксель или в файле `.mp3` одна нота, ваш глаз или слух вряд ли заметит разницу, хотя, конечно, ценители музыки скажут, что они определенно слышат разницу между `.mp3` и несжатым файлом `.flac`. Сильные стороны сжатия с потерями - его экономичность и эффективность. Коэффициент сжатия очень высок, то есть получившийся файл значительно меньше исходного.

Однако сжатие с потерями неприемлемо, когда вы отправляете файлы или программное обеспечение и целостность данных критически важна. Например, если вы отправляете сценарий или документ, целостность исходного файла должна сохраняться при распаковке. Эта глава посвящена сжатию без потерь, которое доступно через ряд утилит и алгоритмов. К сожалению, сжатие без потерь не так эффективно, как сжатие с потерями, что вполне ожидаемо, но для хакера целостность часто гораздо важнее коэффициента сжатия.

Объединение файлов в tar-архив

Обычно первое, что вы делаете при сжатии файлов, - объединяете их в архив. В большинстве случаев при архивировании файлов вы будете использовать команду `tar`. Название `tar` означает "ленточный архив", что отсылает к доисторическим дням вычислительной техники, когда системы использовали ленту для хранения данных. Команда `tar` создает один файл из множества файлов; затем такой файл называют архивом или tar-файлом.

Например, предположим, что у вас есть три файла сценариев, похожих на те, что мы использовали в главе 8, с именами `hackersarise1`, `hackersarise2` и `hackersarise3`. Если вы перейдете в каталог, где они находятся, и выполните подробный листинг, вы ясно увидите файлы и ожидаемые сведения о них, включая размер файлов, как показано здесь:

```
kali >ls -l
-rwxr-xr-x 1 root root 22311 Nov 27 2018 13:00 hackersarise1.sh
-rwxr-xr-x 1 root root 8791 Nov 27 2018 13:00 hackersarise2.sh
-rwxr-xr-x 1 root root 3992 Nov 27 2018 13:00 hackersarise3.sh
```

Предположим, вы хотите отправить все три файла другому хакеру, с которым работаете над проектом. Вы можете объединить их и создать один архивный файл с помощью команды из листинга 9-1.

```
kali >tar -cvf HackersArise.tar hackersarise1 hackersarise2 hackersarise3
hackersarise1
hackersarise2
hackersarise3
```

Листинг 9-1. Создание tar-файла из трех файлов

Разберем эту команду, чтобы лучше ее понять. Команда архивирования - `tar`, и здесь мы используем ее с тремя параметрами. Параметр `c` означает создание, `v` означает подробный вывод (он необязателен) и выводит список файлов, с которыми работает `tar`, а `f` означает запись в следующий указанный файл. Этот последний параметр также работает при чтении из файлов. Затем мы задаем новому архиву имя файла, который хотим создать из трех сценариев: `HackersArise.tar`.

В целом эта команда возьмет все три файла и создаст из них один файл, `HackersArise.tar`. Когда вы снова выполните подробный листинг каталога, вы увидите, что теперь в нем также есть новый файл `.tar`, как показано далее:

```
kali >ls -l
--snip--
-rw-r--r-- 1 root root 40960 Nov 27 2018 13:32 HackersArise.tar
--snip--
kali >
```

Обратите внимание на размер tar-файла: 40 960 байт. Когда три файла архивируются, `tar` использует значительные накладные расходы для выполнения этой операции: если сумма размеров трех файлов до архивирования составляла 35 094 байта, то после архивирования tar-файл вырос до 40 960 байт. Другими словами, процесс архивирования добавил более 5 000 байт. Хотя такие накладные расходы могут быть существенными для небольших файлов, они становятся все менее и менее значимыми по мере увеличения размеров файлов.

Мы можем вывести эти файлы из tar-файла, не извлекая их, используя команду `tar` с ключом `-t` для списка содержимого, как показано далее:

```
kali >tar -tvf HackersArise.tar
-rwxr-xr-x 1 root root 22311 Nov 27 2018 13:00 hackersarise1.sh
-rwxr-xr-x 1 root root 8791 Nov 27 2018 13:00 hackersarise2.sh
-rwxr-xr-x 1 root root 3992 Nov 27 2018 13:00 hackersarise3.sh
```

Здесь мы видим три исходных файла и их исходные размеры. Затем вы можете извлечь эти файлы из tar-файла с помощью команды `tar` с ключом `-x` (извлечь), как показано далее:

```
kali >tar -xvf HackersArise.tar
hackersarise1.sh
hackersarise2.sh
hackersarise3.sh
```

Поскольку вы все еще используете ключ `-v`, эта команда покажет в выводе, какие файлы извлекаются. Если вы хотите извлечь файлы "тихо", то есть без вывода чего-либо на экран, можно просто убрать ключ `-v` (подробный вывод), как показано здесь:

```
kali >tar -xf HackersArise.tar
```

Файлы были извлечены в текущий каталог; чтобы перепроверить это, вы можете выполнить подробный листинг каталога. Обратите внимание, что по умолчанию, если извлекаемый файл уже существует, `tar` удалит существующий файл и заменит его извлеченным файлом.

Сжатие файлов

Теперь у нас есть один архивный файл, но этот файл больше суммы исходных файлов. Что, если вы хотите сжать эти файлы для удобной передачи? В Linux есть несколько команд, способных создавать сжатые файлы. Мы рассмотрим следующие:

`gzip`, которая использует расширение `.tar.gz` или `.tgz`

`bzip2`, которая использует расширение `.tar.bz2`

`compress`, которая использует расширение `.tar.z`

Все они способны сжимать наши файлы, но используют разные алгоритмы сжатия и имеют разные коэффициенты сжатия. Поэтому мы рассмотрим каждую из них и то, на что она способна.

В целом `compress` является самой быстрой, но итоговые файлы получаются крупнее; `bzip2` самая медленная, но итоговые файлы получаются самыми маленькими; а `gzip` находится где-то посередине. Главная причина, по которой вам, как начинающему хакеру, следует знать все три метода, состоит в том, что при работе с другими инструментами вам будут встречаться разные типы сжатия. Поэтому в этом разделе показано, как работать с основными методами сжатия.

Сжатие с помощью gzip

Сначала попробуем `gzip` (GNU zip), поскольку это самая часто используемая утилита сжатия в Linux. Вы можете сжать файл `HackersArise.tar`, введя следующее (убедившись, что вы находитесь в каталоге, где лежит архивный файл):

```
kali >gzip HackersArise.*
```

Обратите внимание, что мы использовали подстановочный знак * для расширения файла; это сообщает Linux, что команда должна применяться к любому файлу, который начинается с `HackersArise` и имеет любое расширение. Похожую запись вы будете использовать в следующих

примерах. Когда мы выполняем подробный листинг каталога, видно, что HackersArise.tar был заменен файлом HackersArise.tar.gz, а размер файла сжался всего до 3 299 байт!

```
kali >ls -l
--snip--
-rw-r--r-- 1 root root 3299 Nov 27 2018 13:32 HackersArise.tar.gz
--snip--
```

Затем мы можем распаковать тот же файл с помощью команды gunzip, сокращенно от GNU unzip.

```
kali >gunzip HackersArise.*
```

После распаковки файл больше не сохраняется с расширением .tar.gz, а снова имеет расширение .tar. Также обратите внимание, что он вернулся к исходному размеру 40 960 байт. Попробуйте выполнить подробный листинг, чтобы подтвердить это. Стоит отметить, что gzip также можно использовать для извлечения файлов .zip.

Сжатие с помощью bzip2

Еще одна из широко используемых утилит сжатия в Linux - bzip2, которая работает похожим образом с gzip, но имеет лучшие коэффициенты сжатия, то есть итоговый файл будет еще меньше. Вы можете сжать файл HackersArise.tar, введя следующее:

```
kali >bzip2 HackersArise.*
```

Когда вы выполните подробный листинг, вы увидите, что bzip2 сжала файл всего до 2 081 байта! Также обратите внимание, что расширение файла теперь .tar.bz2.

Чтобы распаковать сжатый файл, используйте bunzip2, вот так:

```
kali >bunzip2 HackersArise.*
kali >
```

После этого файл возвращается к исходному размеру, а его расширение снова становится .tar.

Сжатие с помощью compress

Наконец, для сжатия файла можно использовать команду compress. Вероятно, это наименее часто используемая утилита сжатия, но ее легко запомнить. Чтобы воспользоваться ей, просто введите команду compress, а затем имя файла, вот так:

```
kali >compress HackersArise.*
kali >ls -l
--snip--
-rw-r--r-- 1 root root 5476 Nov 27 2018 13:32 HackersArise.tar.Z
```

Обратите внимание, что утилита compress уменьшила размер файла до 5 476 байт, что более чем вдвое больше результата bzip2. Также обратите внимание, что расширение файла теперь .tar.Z (с прописной Z).

Чтобы распаковать тот же файл, используйте `uncompress`:

```
kali >uncompress HackersArise.*
```

Вы также можете использовать команду `gunzip` с файлами, которые были сжаты с помощью `compress`.

Создание побитовых или физических копий устройств хранения

В мире информационной безопасности и хакинга одна команда Linux для архивирования по своей полезности стоит выше остальных. Команда `dd` делает побитовую копию файла, файловой системы или даже целого жесткого диска. Это означает, что копируются даже удаленные файлы (да, важно знать, что ваши удаленные файлы могут быть восстановлены), что упрощает их обнаружение и восстановление. Большинство утилит логического копирования, таких как `cp`, удаленные файлы не копируют.

После того как хакер завладел целевой системой, команда `dd` позволит ему скопировать весь жесткий диск или устройство хранения на свою систему. Кроме того, люди, чья работа состоит в поимке хакеров, а именно специалисты по компьютерной криминалистике, скорее всего, будут использовать эту команду, чтобы сделать физическую копию жесткого диска с удаленными файлами и другими артефактами, которые могут быть полезны для поиска доказательств против хакера.

Критически важно отметить, что команду `dd` не следует использовать для обычного повседневного копирования файлов и устройств хранения, потому что она очень медленная; другие команды выполняют эту работу быстрее и эффективнее. Тем не менее она превосходна, когда вам нужна копия устройства хранения без файловой системы или других логических структур, например при криминалистическом расследовании.

Базовый синтаксис команды `dd` выглядит следующим образом:

```
dd if=inputfile of=outputfile
```

Итак, если бы вы хотели сделать физическую копию своей флешки, предполагая, что флешка - это `sdb` (мы подробнее обсудим это обозначение в главе 10), вы ввели бы следующее:

```
kali >dd if=/dev/sdb of=/root/flashcopy
1257441=0 records in
1257440+0 records out
7643809280 bytes (7.6 GB) copied, 1220.729 s, 5.2 MB/s
```

Разберем эту команду: `dd` - это ваша физическая команда "копирования"; `if` обозначает входной файл, где `/dev/sdb` представляет вашу флешку в каталоге `/dev`; `of` обозначает выходной файл; а `/root/flashcopy` - это имя файла, в который вы хотите записать физическую копию. (Более полное объяснение обозначений дисков в системе Linux внутри каталога `/dev` см. в главе 10.)

С командой dd доступно множество параметров, и вы можете самостоятельно немного их изучить, но среди наиболее полезных - параметр noerror и параметр bs ("размер блока"). Как следует из названия, параметр noerror продолжает копирование даже при возникновении ошибок. Параметр bs позволяет определить размер блока (число байтов, читаемых/записываемых за один блок) копируемых данных. По умолчанию он установлен на 512 байт, но его можно изменить, чтобы ускорить процесс. Обычно его задают равным размеру сектора устройства, чаще всего 4 КБ (4 096 байт). С этими параметрами ваша команда выглядела бы так:

```
kali >dd if=/dev/sdb of=/root/flashcopy bs=4096 conv=noerror
```

Как уже упоминалось, стоит самостоятельно провести еще небольшое исследование, но это хорошее введение в команду и ее типичные способы применения.

Итоги

В Linux есть ряд команд, которые позволяют объединять и сжимать файлы для более удобной передачи. Для объединения файлов предпочтительная команда - tar, а для сжатия файлов у вас есть как минимум три утилиты: gzip, bzip2 и compress, причем у всех разные коэффициенты сжатия. Команда dd идет дальше. Она позволяет делать физическую копию устройств хранения без логических структур, таких как файловая система, что дает возможность восстанавливать такие артефакты, как удаленные файлы.

Упражнения

Прежде чем перейти к главе 10, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Создайте три сценария для объединения, похожие на те, что мы делали в главе 8. Назовите их Linux4Hackers1, Linux4Hackers2 и Linux4Hackers3.
2. Создайте tar-файл из этих трех файлов. Назовите tar-файл L4H. Отметьте, как меняется размер суммы трех файлов, когда они объединяются в tar-архив.
3. Сожмите tar-файл L4H с помощью gzip. Отметьте, как меняется размер файла. Исследуйте, как можно управлять перезаписью существующих файлов. Затем распакуйте файл L4H.
4. Повторите упражнение 3, используя как bzip2, так и compress.
5. Сделайте физическую, побитовую копию одной из своих флешек с помощью команды dd.

Глава 10. Управление файловой системой и устройствами хранения

Если вы пришли из среды Windows, то способ, которым Linux представляет устройства хранения и управляет ими, покажется вам довольно другим. Вы уже видели, что файловая система не имеет физического представления диска, как система `C:`, `D:` или `E:` в Windows, а вместо этого имеет древовидную структуру файлов с `/` наверху, то есть в корне. В этой главе рассматривается, как Linux представляет устройства хранения, такие как жесткие диски, флешки и другие накопители.

Сначала мы посмотрим, как дополнительные диски и другие устройства хранения монтируются в эту файловую систему, ведущую к каталогу `/` (root). Монтирование в этом контексте просто означает присоединение дисков или накопителей к файловой системе, чтобы сделать их доступными операционной системе (ОС). Вам как хакеру необходимо понимать систему управления файлами и устройствами хранения как на собственной системе, так и часто на системе вашей цели. Хакеры обычно используют внешние носители для загрузки данных, хакерских инструментов или даже своей ОС. Когда вы оказываетесь в целевой системе, вам нужно понимать, с чем вы работаете, где искать конфиденциальные или другие критически важные файлы, как смонтировать диск на цель и можно ли и куда поместить эти файлы в вашей системе. Все эти темы, а также управление и мониторинг устройств хранения, мы рассмотрим в этой главе.

Мы начнем с каталога, известного как `/dev`, который вы, вероятно, уже заметили в структуре каталогов: `dev` - сокращение от "устройство", и каждое устройство в Linux представлено своим собственным файлом внутри каталога `/dev`. Начнем с работы с `/dev`.

Каталог устройств /dev

В Linux есть специальный каталог, содержащий файлы, которые представляют каждое подключенное устройство: удачно названный каталог `/dev`. Для первого знакомства перейдите в каталог `/dev`, а затем выполните для него подробный листинг. Вы должны увидеть что-то похожее на листинг 10-1.

```
kali >cd /dev
kali >ls -l
total 0
crw------- 1 root root 10,175 May 16 12:44 agpgart
crw------- 1 root root 10,235 May 16 12:44 autofs
drwxr-xr-x 1 root root 160 May 16 12:44 block
--snip--
lrwxrwxrwx 1 root root 3 May 16 12:44 cdrom -> sr0
--snip--
drwxr-xr-x 2 root root 60 May 16 12:44 cpu
--snip--
```

Листинг 10-1. Подробный листинг каталога `/dev`

По умолчанию устройства отображаются в алфавитном порядке. Некоторые устройства, например `cdrom` и `cpu`, вы можете узнать, но у других имена довольно загадочные. Каждое устройство в вашей системе представлено файлом в каталоге `/dev`, включая устройства, которыми вы, вероятно, никогда не пользовались или даже не подозревали об их существовании. На случай, если оно вам когда-нибудь понадобится, для него уже ждет файл устройства.

Если вы немного прокрутите этот экран вниз, то увидите еще больше записей устройств. Особый интерес представляют устройства `sda1`, `sda2`, `sda3`, `sdb` и `sdb1`: это жесткий диск и его разделы, а также USB-флешка и ее разделы.

```
--snip--
brw-rw---- 1 root root 8, 0 May 16 12:44 sda
brw-rw---- 1 root root 8, 1 May 16 12:44 sda1
brw-rw---- 1 root root 8, 2 May 16 12:44 sda2
brw-rw---- 1 root root 8, 5 May 16 12:44 sda5
brw-rw---- 1 root root 8, 16 May 16 12:44 sdb
brw-rw---- 1 root root 8, 17 May 16 12:44 sdb1
--snip--
```

Рассмотрим их подробнее.

Как Linux представляет устройства хранения

Linux использует для дисков логические метки, которые затем монтируются в файловую систему. Эти логические метки будут различаться в зависимости от того, куда смонтированы диски; это означает, что один и тот же жесткий диск может иметь разные метки в разное время, в зависимости от того, где и когда он смонтирован.

Изначально Linux представлял дисководы гибких дисков (помните такие?) как `fd0`, а жесткие диски как `hda`. Иногда вы все еще будете видеть такие представления дисков в устаревших Linux-системах, но сегодня большинство дисководов гибких дисков исчезло (и слава богу). Даже так старые унаследованные жесткие диски, использовавшие интерфейс IDE или E-IDE, все еще представляются в форме `hda`. Более новые диски с интерфейсом последовательного ATA (SATA) и жесткие диски с интерфейсом малых компьютерных систем (SCSI) представляются как `sda`. Диски иногда разбиваются на части, известные как разделы, которые в системе обозначений представлены числами, как вы увидите далее.

Когда в системе больше одного жесткого диска, Linux просто именует их последовательно, увеличивая последнюю букву в алфавитном порядке: первый диск - `sda`, второй - `sdb`, третий - `sdc` и так далее (см. таблицу 10-1). Последнюю букву после `sd` часто называют major number, старшим номером.

Таблица 10-1. Система именования устройств

Файл устройства	Описание
`sda`	Первый жесткий диск SATA
`sdb`	Второй жесткий диск SATA
`sdc`	Третий жесткий диск SATA
`sdd`	Четвертый жесткий диск SATA

Разделы диска

Некоторые диски можно разделять на разделы, чтобы управлять информацией и отделять ее. Например, вы можете захотеть разделить жесткий диск так, чтобы файл подкачки, домашний каталог и каталог `/` находились на отдельных разделах; причин для этого может быть несколько, включая совместное использование ресурсов и ослабление разрешений по умолчанию.

Linux помечает каждый раздел младшим номером, который идет после обозначения диска. Таким образом, первый раздел на первом SATA-диске будет `sda1`. Второй раздел будет `sda2`, третий - `sda3` и так далее, как показано в таблице 10-2.

Таблица 10-2. Система обозначения разделов

Раздел	Описание
`sda1`	Первый раздел (1) на первом (a) SATA-диске
`sda2`	Второй (2) раздел на первом (a) диске
`sda3`	Третий (3) раздел на первом (a) диске
`sda4`	Четвертый (4) раздел на первом (a) диске

Иногда вам может понадобиться просмотреть разделы в вашей Linux-системе, чтобы увидеть, какие из них у вас есть и сколько емкости доступно в каждом. Это можно сделать с помощью утилиты `fdisk`. Использование ключа `-l` с `fdisk` выводит список всех разделов всех дисков, как показано в листинге 10-2.

```
kali >fdisk -l
Disk /dev/sda: 20GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x7c06cd70
Device Boot  Start End  Sectors Size Id Type
/dev/sda1 *  2048 39174143 39172096 18.7G 83 Linux
/dev/sda2  39176190 41940991 2764802 1.3G 5 Extended
/dev/sda5  39176192 41940991 2764800 1.3G 82 Linux swap / Solaris
Disk /dev/sdb: 29.8 GiB, 31999393792 bytes, 62498816 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xc3072e18
Device Boot Start End Sectors Size Id Type
/dev/sdb1   32 62498815 62498784 29.8G 7 HPFS/NTFS/exFAT
```

Листинг 10-2. Вывод списка разделов с помощью `fdisk`

Как видно в листинге 10-2, устройства `sda1`, `sda2` и `sda5` перечислены в первой строфе. Эти три устройства составляют виртуальный диск моей виртуальной машины: это диск на 20 ГБ с тремя

разделами, включая раздел подкачки (sda5), который действует как виртуальная оперативная память, подобно файлам подкачки в Windows, когда емкость оперативной памяти превышена.

Если просмотреть листинг 10-2 ниже, до третьей строфы, вы увидите вывод второго устройства, обозначенного sdb1: метка b говорит нам, что этот диск отделен от первых трех устройств. Это моя флешка на 64 ГБ. Обратите внимание, что fdisk указывает тип файловой системы HPFS/NTFS/ExFAT. Эти типы файлов - высокопроизводительная файловая система (HPFS), файловая система новой технологии (NTFS) и расширенная таблица размещения файлов (exFAT) - не являются родными для Linux-систем, а относятся к системам macOS и Windows. При расследовании полезно уметь распознавать типы файловых систем, родные для разных систем. Файловая система может указать, на машине какого типа был отформатирован диск, а это может быть ценной информацией. Kali способна использовать USB-флешки, созданные во многих разных операционных системах.

Как вы видели в главе 1, файловая система Linux устроена значительно иначе, чем Windows и другие проприетарные операционные системы. Кроме того, способ хранения файлов и управления ими в Linux тоже отличается. Новые версии Windows используют файловую систему NTFS, тогда как старые Windows-системы используют системы таблицы размещения файлов (FAT). Linux использует несколько разных типов файловых систем, но самые распространенные - ext2, ext3 и ext4. Все они являются итерациями файловой системы ext (или расширенной), причем ext4 - самая новая.

Символьные и блочные устройства

Еще одна вещь, которую следует отметить в именовании файлов устройств в каталоге /dev: в первой позиции находится либо c, либо b. Вы можете видеть это в листинге 10-1 в начале большинства записей, и выглядит это примерно так:

```
crw------- 1 root root 10,175 May 16 12:44 agpgart
```

Эти буквы представляют два способа, которыми устройства передают данные в систему и из нее. c означает character, и такие устройства, как и можно ожидать, известны как символьные устройства. Внешние устройства, которые взаимодействуют с системой, отправляя и принимая данные символ за символом, например мыши или клавиатуры, являются символьными устройствами.

b означает второй тип: блочные устройства. Они обмениваются блоками данных (несколькими байтами за раз) и включают такие устройства, как жесткие диски и DVD-приводы. Этим устройствам требуется более высокая пропускная способность передачи данных, поэтому они отправляют и принимают данные блоками (много символов или байтов за раз). Когда вы знаете, является ли устройство символьным или блочным, вы можете легко получить о нем больше информации, как увидите далее.

Вывод списка блочных устройств и сведений о них с помощью lsblk

Команда Linux lsblk, сокращенно от list block, выводит некоторые базовые сведения о каждом блочном устройстве, перечисленном в /dev. Результат похож на вывод fdisk -l, но она также показывает устройства с несколькими разделами в виде своего рода дерева, отображая каждое устройство с его разделами как ветвями, и для запуска не требует привилегий root. В листинге 10-3, например, мы видим sda с его ветвями sda1, sda2 и sda5.

```
kali >lsblk
Name   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
fd0   2:0  1 4K  0 disk
```

```
sda1  8:0  0 20G  0 disk
|-sda1 8:1 0 18.7G 0 part /
|-sda2 8:2  0 1K  0 part
|-sda5 8:5  0 1.3G 0 part [SWAP]
sdb   8:16 1 29.8G 0 disk
|-sdb1 8.17 1 29.8G 0 disk /media
sr0   11:0 1 2.7G 0 rom
```

Листинг 10-3. Вывод сведений о блочных устройствах с помощью `lsblk`

Вывод включает дисковод гибких дисков как `fd0` и DVD-привод как `sr0`, хотя в моей системе нет ни того ни другого: это просто наследие старых систем. Мы также можем видеть информацию о точке монтирования диска - это место, в котором диск был присоединен к файловой системе. Обратите внимание, что жесткий диск `sda1` смонтирован в `/`, а флешка смонтирована в `/media`. Значение этого вы увидите подробнее в следующем разделе.

Монтирование и размонтирование

Большинство современных операционных систем, включая большинство новых версий Linux, автоматически монтируют устройства хранения при их подключении: это означает, что новая флешка или жесткий диск автоматически присоединяется к файловой системе. Для тех, кто только знакомится с Linux, монтирование может быть непривычной темой.

Устройство хранения сначала должно быть физически подключено к системе, а затем логически присоединено к файловой системе, чтобы данные стали доступны операционной системе. Другими словами, даже если устройство физически подключено к системе, оно не обязательно логически присоединено и доступно операционной системе. Термин mount - наследие ранних дней вычислительной техники, когда ленты хранения (до появления жестких дисков) нужно было физически монтировать в компьютерную систему; представьте те большие компьютеры с вращающимися ленточными приводами, которые вы могли видеть в старых научно-фантастических фильмах.

Как упоминалось, точка в дереве каталогов, где присоединяются устройства, называется точкой монтирования. Две основные точки монтирования в Linux - `/mnt` и `/media`. Как правило, внутренние жесткие диски монтируются в `/mnt`, а внешние USB-устройства, такие как флешки и внешние USB-жесткие диски, монтируются в `/media`, хотя технически можно использовать любой каталог.

Самостоятельное монтирование устройств хранения

В некоторых версиях Linux вам нужно смонтировать диск вручную, чтобы получить доступ к его содержимому, так что этому навыку стоит научиться. Чтобы смонтировать диск в файловую систему, используйте команду `mount`. Точка монтирования для устройства должна быть пустым каталогом; если вы смонтируете устройство в каталог, где есть подкаталоги и файлы, смонтированное устройство закроет содержимое каталога, сделав его невидимым и недоступным. Итак, чтобы смонтировать новый жесткий диск `sdb1` в каталог `/mnt`, вы ввели бы следующее:

```
kali >mount /dev/sdb1 /mnt
```

После этого жесткий диск должен стать доступным. Если вы хотите смонтировать флешку `sdc1` в каталог `/media`, вы ввели бы это:

```
kali >mount /dev/sdc1 /media
```

Файловые системы, смонтированные в системе, хранятся в файле `/etc/fstab` (сокращенно от filesystem table), который система читает при каждой загрузке.

Размонтирование с помощью umount

Если вы пришли из среды Mac или Windows, вы, вероятно, уже размонтировали диск, даже не зная этого. Перед тем как извлечь флешку из системы, вы "извлекаете" ее, чтобы не повредить файлы, хранящиеся на устройстве. Eject - это просто другое слово для unmount, размонтировать.

Подобно команде `mount`, вы можете размонтировать второй жесткий диск, введя команду `umount`, а затем файловую запись устройства в каталоге `/dev`, например `/dev/sdb`. Обратите внимание, что команда пишется не `unmount`, а именно `umount` (без `n`).

```
kali >umount /dev/sdb1
```

Вы не можете размонтировать занятое устройство, поэтому если система читает с устройства или пишет на него, вы просто получите ошибку.

Мониторинг файловых систем

В этом разделе мы рассмотрим некоторые команды для мониторинга состояния файловой системы - навык, необходимый любому хакеру или системному администратору. Мы получим сведения о смонтированных дисках, а затем проверим и исправим ошибки. Устройства хранения особенно подвержены ошибкам, поэтому этому навыку стоит научиться.

Получение сведений о смонтированных дисках

Команда `df` (от disk free) предоставит нам базовую информацию о любых жестких дисках или смонтированных устройствах, таких как CD, DVD и флешки, включая то, сколько места используется и сколько доступно (см. листинг 10-4). Без параметров `df` по умолчанию обращается к первому диску в вашей системе (в данном случае `sda`). Если вы хотите проверить другой диск, просто укажите после команды `df` представление диска, который хотите проверить (например, `df sdb`).

```
kali >df
Filesystem  1K-Blocks Used Available Use% Mounted on
rootfs    19620732 17096196 1504788 92%  /
udev        10240   0   10240 0%  /dev
--snip--
/dev/sdb1    29823024 29712544 110480 99%  /media/USB3.0
```

Листинг 10-4. Получение сведений о дисках и смонтированных устройствах с помощью `df`

Первая строка вывода здесь показывает заголовки категорий, а затем мы получаем информацию. Дисковое пространство указано в блоках по 1 КБ. Во второй строке мы видим, что `rootfs` имеет 19 620 732 блока по одному килобайту, из которых использует 17 096 196 (или около 92 процентов), оставляя доступными 1 504 788. Команда `df` также сообщает нам, что эта файловая система смонтирована в верхней части файловой системы, `/`.

В последней строке вы можете видеть мою USB-флешку. Обратите внимание, что она обозначена как `/dev/sdb1`, почти на 100 процентов заполнена и смонтирована в `/media/USB3.0`.

Подытожим: мой виртуальный диск в этой системе обозначен как `sda1`, что раскладывается следующим образом:

`sd` - жесткий диск SATA

`a` - первый жесткий диск

`1` - первый раздел на этом диске

Моя флешка на 64 ГБ обозначена как `sdb1`, а мой внешний диск - как `sdc1`.

Проверка ошибок

Команда `fsck` (сокращенно от filesystem check) проверяет файловую систему на ошибки и исправляет повреждение, если возможно, или помещает плохой участок в таблицу bad blocks, чтобы отметить его как неисправный. Чтобы запустить команду `fsck`, нужно указать тип файловой системы (по умолчанию `ext2`) и файл устройства для проверки. Важно отметить, что перед запуском проверки файловой системы вы должны размонтировать диск. Если вы не размонтируете смонтированное устройство, получите сообщение об ошибке, показанное в листинге 10-5.

```
kali >fsck
fsck from util-linux 2.20.1
e2fsck 1.42.5 (29-Jul-2012)
/dev/sda1 is mounted
e2fsck: Cannot continue, aborting.
```

Листинг 10-5. Попытка (и неудача) запустить проверку ошибок на смонтированном диске

Итак, первый шаг при выполнении проверки файловой системы - размонтировать устройство. В этом случае я размонтирую свою флешку, чтобы выполнить проверку файловой системы:

```
kali >umount /dev/sdb1
```

Я могу добавить параметр `-p`, чтобы `fsck` автоматически исправляла любые проблемы с устройством, вот так:

```
kali >fsck -p /dev/sdb1
```

Теперь, когда устройство размонтировано, я могу проверить его на плохие сектора или другие проблемы следующим образом:

```
kali >fsck -p /dev/sdb1
fsck from util-linux 2.30.2
exfatfsck 1.2.7
Проверка файловой системы на /dev/sdb1.
Версия файловой системы 1.0
Размер сектора    512 байт
Размер кластера    32 КБ
Размер тома     7648 МБ
Использовано     1265 МБ
Доступное место 6383 МБ
Всего 20 каталогов и 111 файлов.
Проверка файловой системы завершена. Ошибок не найдено.
```

Итоги

Понимание того, как Linux обозначает свои устройства и управляет ими, критически важно для любого пользователя Linux и хакера. Хакерам нужно знать, какие устройства подключены к системе и сколько места доступно. Поскольку устройства хранения часто получают ошибки, мы можем проверять и исправлять эти ошибки с помощью `fsck`. Команда `dd` способна делать физическую копию устройства, включая любые удаленные файлы.

Упражнения

Прежде чем перейти к главе 11, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Используйте команды `mount` и `umount`, чтобы смонтировать и размонтировать свою флешку.
2. Проверьте количество свободного дискового пространства на своем основном жестком диске.
3. Проверьте свою флешку на ошибки с помощью `fsck`.
4. Используйте команду `dd`, чтобы скопировать все содержимое одной флешки на другую, включая удаленные файлы.
5. Используйте команду `lsblk`, чтобы определить базовые характеристики своих блочных устройств.

Глава 11. Система журналирования

Для любого пользователя Linux крайне важно хорошо разбираться в использовании файлов журналов. Файлы журналов хранят информацию о событиях, которые происходят при работе операционной системы и приложений, включая любые ошибки и предупреждения безопасности. Ваша система будет автоматически записывать информацию в журнал на основе набора правил, которые я покажу вам, как настраивать в этой главе.

Для хакера файлы журналов могут быть следом к действиям и личности цели. Но они также могут быть следом к вашим собственным действиям в чужой системе. Поэтому хакеру нужно знать, какую информацию он может собрать, а также что может быть собрано о его собственных действиях и методах, чтобы скрыть эти доказательства.

С другой стороны, любой, кто защищает Linux-системы, должен знать, как управлять функциями журналирования, чтобы определить, была ли система атакована, а затем расшифровать, что на самом деле произошло и кто это сделал.

Эта глава показывает, как изучать и настраивать файлы журналов, а также как удалять доказательства вашей активности и даже полностью отключать журналирование. Сначала мы рассмотрим демон, который выполняет журналирование.

Демон журналирования rsyslog

Linux использует демон с именем `syslogd`, чтобы автоматически записывать события на вашем компьютере. В разных дистрибутивах Linux используются несколько вариантов syslog, включая `rsyslog` и `syslog-ng`, и хотя они работают очень похоже, между ними есть небольшие различия. Поскольку Kali Linux построен на Debian, а Debian по умолчанию поставляется с `rsyslog`, в этой главе мы сосредоточимся на этой утилите. Если вы хотите использовать другие дистрибутивы, стоит немного изучить их системы журналирования.

Давайте посмотрим на rsyslog в вашей системе. Мы найдем все файлы, связанные с rsyslog. Сначала откройте терминал в Kali и введите следующее:

```
kali >locate rsyslog
/etc/rsyslog.conf
/etc/rsyslog.d
/etc/default/rsyslog
/etc/init.d/rsyslog
/etc/logcheck/ignore.d.server/rsyslog
/etc/logrotate.d/rsyslog
/etc/rc0.d/K04rsyslog
--snip--
```

Как видите, множество файлов содержит ключевое слово rsyslog; некоторые из них полезнее других. Файл, который мы хотим изучить, - это конфигурационный файл rsyslog.conf.

Конфигурационный файл rsyslog

Как почти каждое приложение в Linux, rsyslog управляется и настраивается с помощью текстового конфигурационного файла, расположенного, как обычно бывает в Linux, в каталоге /etc. В случае rsyslog конфигурационный файл находится по пути /etc/rsyslog.conf. Откройте этот файл любым текстовым редактором, и мы изучим, что внутри (здесь я использую Leafpad):

```
kali >leafpad /etc/rsyslog.conf
```

Вы должны увидеть что-то похожее на листинг 11-1.

```
/etc/rsyslog.conf Конфигурационный файл для rsyslog.
# Подробнее см.
# /usr/share/doc/rsyslog-doc/html/rsyslog_conf.html

#################
#### МОДУЛИ  ####
#################
module(load="imuxsock") # обеспечивает поддержку локального системного журналирования
module(load="imklog") # обеспечивает поддержку журналирования ядра
#module(load="immark") # обеспечивает возможность сообщений --MARK--
# обеспечивает прием syslog по UDP
#module(load="imudp")
#input(type="imudp" port="514")
# обеспечивает прием syslog по TCP
#module(load="imtcp")
#input(type="imtcp" port="514")
###########################
#### ГЛОБАЛЬНЫЕ ДИРЕКТИВЫ ####
###########################
```

Листинг 11-1. Фрагмент файла rsyslog.conf

Как видите, файл rsyslog.conf хорошо документирован множеством комментариев, объясняющих его использование. Большая часть этой информации сейчас вам не пригодится, но если вы перейдете ниже строки 50, то найдете раздел Rules. Именно здесь можно задать правила того, что ваша Linux-система будет автоматически записывать в журнал.

Правила журналирования rsyslog

Правила `rsyslog` определяют, какая информация записывается в журнал, сообщения каких программ журналируются и где хранится этот журнал. Как хакеру, это позволяет вам узнать, что журналируется и куда записываются эти журналы, чтобы вы могли удалить или скрыть их. Прокрутите до строки 50, и вы должны увидеть что-то похожее на листинг 11-2.

```
###############
#### ПРАВИЛА ####
###############
#
# Сначала несколько стандартных файлов журналов. Журналирование по facility.
#
auth,authpriv.*  /var/log/auth.log
*.*;auth,authpriv.none -/var/log/syslog
#cron.*       /var/log/cron.log
daemon.*        -/var/log/daemon.log
kern.*        -/var/log/kern.log
lpr.*        -/var/log/lpr.log
mail.*        -/var/log/mail.log
user.*       -/var/log/user.log
#
# Журналирование для почтовой системы. Разделите его так,
# чтобы было легко писать сценарии для разбора этих файлов.
#
mail.info      -/var/log/mail.info
mail.warn      -/var/log/mail.warn
mail.err      /var/log/mail.err
```

Листинг 11-2. Поиск правил журналирования в `rsyslog.conf`

Каждая строка - это отдельное правило журналирования, которое говорит, какие сообщения записывать и куда их записывать. Базовый формат этих правил выглядит так:

```
facility.priority  action
```

Ключевое слово `facility` указывает на программу, например `mail`, kernel или `lpr`, сообщения которой записываются в журнал. Ключевое слово `priority` определяет, какие типы сообщений для этой программы записывать. Ключевое слово `action`, находящееся справа, указывает место, куда будет отправлен журнал. Давайте подробнее рассмотрим каждый раздел, начиная с ключевого слова `facility`, которое относится к тому программному обеспечению, которое создает журнал, будь то ядро, почтовая система или пользователь.

Ниже приведен список допустимых кодов, которые можно использовать вместо ключевого слова `facility` в правилах нашего конфигурационного файла:

Код	Описание
`auth/authpriv`	Сообщения безопасности/авторизации
`cron`	Демоны часов
`daemon`	Другие демоны
`kern`	Сообщения ядра
`lpr`	Система печати
`mail`	Почтовая система
`user`	Общие сообщения пользовательского уровня

Подстановочный знак звездочки (*) вместо слова относится ко всем facilities. Вы можете выбрать больше одного facility, перечислив их через запятую.

`priority` сообщает системе, какие типы сообщений записывать в журнал. Коды перечислены от самого низкого приоритета, начиная с `debug`, до самого высокого, заканчивая `panic`. Если priority равно *, журналируются сообщения всех приоритетов. Когда вы указываете приоритет, журналируются сообщения этого приоритета и выше. Например, если вы укажете код приоритета `alert`, система будет записывать сообщения, классифицированные как `alert` и более высокого приоритета, но не будет записывать сообщения, помеченные как `crit`, или любые приоритеты ниже `alert`.

Вот полный список допустимых кодов для `priority`:

```
debug
info
notice
warning
warn
error
err
crit
alert
emerg
panic
```

Коды `warning`, `warn`, `error`, `err`, `emerg` и `panic` все считаются устаревшими и не должны использоваться.

`action` обычно является именем файла и местом, куда должны отправляться журналы. Обратите внимание, что обычно файлы журналов отправляются в каталог `/var/log` с именем файла, которое описывает facility, создавшую их, например `auth`. Это означает, например, что журналы, созданные facility `auth`, будут отправлены в `/var/log.auth.log`.

Рассмотрим несколько примеров правил журналирования:

```
mail.* /var/log/mail
```

Этот пример будет записывать почтовые события всех (`*`) приоритетов в `/var/log/mail`.

```
kern.crit /var/log/kernel
```

Этот пример будет записывать события ядра критического (`crit`) приоритета или выше в `/var/log/kernel`.

```
*.emerg *
```

Этот последний пример будет записывать все события аварийного (`emerg`) приоритета всем вошедшим в систему пользователям. С помощью этих правил хакер может определить, где расположены файлы журналов, изменить приоритеты или даже отключить конкретные правила журналирования.

Автоматическая очистка журналов с помощью logrotate

Файлы журналов занимают место, поэтому если не удалять их периодически, в конце концов они заполнят весь жесткий диск. С другой стороны, если вы удаляете файлы журналов слишком часто, у вас не будет журналов для расследования в какой-то будущий момент времени. Вы можете использовать `logrotate`, чтобы определить баланс между этими противоположными требованиями путем ротации журналов.

Ротация журналов - это процесс регулярного архивирования файлов журналов путем перемещения их в другое место, после чего у вас остается свежий файл журнала. Затем это архивное место очищается через заданный период времени.

Ваша система уже ротирует файлы журналов с помощью задания cron, которое использует утилиту `logrotate`. Вы можете настроить утилиту `logrotate`, чтобы выбрать регулярность ротации журналов, через текстовый файл `/etc/logrotate.conf`. Давайте откроем его в текстовом редакторе и посмотрим:

```
kali >leafpad /etc/logrotate.conf
```

Вы должны увидеть что-то похожее на листинг 11-3.

```
# подробности см. в "man logrotate"
# ротировать файлы журналов еженедельно
(1) weekly
# хранить журналы за 4 недели
(2) rotate 4
(3) # создавать новые (пустые) файлы журналов после ротации старых
create
(4) # раскомментируйте это, если хотите сжимать файлы журналов
#compress
# пакеты помещают информацию о ротации журналов в этот каталог
include /etc/logrotate.d
# пакеты не владеют wtmp или btmp -- будем ротировать их здесь
/var/log/wtmp {
missingok
monthly
create 0664 root utmp
rotate 1
}
```

Листинг 11-3. Конфигурационный файл `logrotate`

Сначала вы можете задать единицу времени, к которой относятся ваши числа `rotate` (1). Здесь по умолчанию указано `weekly`, то есть любое число после ключевого слова `rotate` всегда относится к неделям.

Ниже можно увидеть настройку того, как часто ротировать журналы: значение по умолчанию - ротировать журналы каждые четыре недели (2). Эта конфигурация по умолчанию подойдет большинству людей, но если вы хотите хранить журналы дольше для целей расследования или меньше, чтобы очищать их быстрее, именно эту настройку следует изменить. Например, если вы проверяете файлы журналов каждую неделю и хотите сэкономить место, можно изменить эту настройку на `rotate 1`. Если у вас много места для журналов и вы хотите хранить полупостоянную запись для последующего криминалистического анализа, можно изменить эту настройку на `rotate 26`, чтобы хранить журналы шесть месяцев, или на `rotate 52`, чтобы хранить их один год.

По умолчанию новый пустой файл журнала создается, когда старые ротируются наружу (3). Как советуют комментарии в конфигурационном файле, вы также можете выбрать сжатие ротированных файлов журналов (4).

В конце каждого периода ротации файлы журналов переименовываются и продвигаются к концу цепочки журналов, а новый файл журнала создается вместо текущего. Например, `/var/log.auth` станет `/var/log.auth.1`, затем `/var/log.auth.2` и так далее.

Если вы ротируете журналы каждые четыре недели и храните четыре набора резервных копий, у вас будет `/var/log.auth.4`, но не будет `/var/log.auth.5`, то есть `/var/log.auth.4` будет удален, а не продвинут в `/var/log/auth.5`. Это можно увидеть, используя команду `locate`, чтобы найти файлы журналов `/var/log/auth.log` с подстановочным знаком, как показано здесь:

```
kali >locate /var/log/auth.log.*
/var/log/auth.log.1
/var/log/auth.log.2
/var/log/auth.log.3
/var/log/auth.log.4
```

Подробнее о множестве способов настраивать и использовать утилиту `logrotate` см. на странице `man logrotate`. Это отличный ресурс, чтобы узнать о функциях, которые можно использовать, и переменных, которые можно изменить, настраивая обработку своих журналов. Когда вы лучше освоитесь с Linux, вы лучше поймете, как часто вам нужно журналировать и какие параметры вы предпочитаете, поэтому стоит вернуться к файлу `logrotate.conf`.

Сохранение скрытности

После того как вы скомпрометировали Linux-систему, полезно отключить журналирование и удалить любые доказательства вашего вторжения в файлах журналов, чтобы снизить вероятность обнаружения. Есть много способов сделать это, и каждый несет свои риски и свой уровень надежности.

Удаление доказательств

Сначала вам нужно удалить любые журналы вашей активности. Вы могли бы просто открыть файлы журналов и точно удалить любые записи, подробно описывающие вашу активность, строка за строкой, используя техники удаления файлов, которые вы изучили в главе 2. Однако это может занять много времени и оставить временные разрывы в файлах журналов, что выглядело бы подозрительно. Кроме того, удаленные файлы обычно может восстановить квалифицированный специалист по криминалистике.

Лучшее и более безопасное решение - уничтожить файлы журналов с помощью `shred`. При других системах удаления файлов квалифицированный исследователь все еще способен восстановить удаленные файлы, но предположим, что существует способ удалить файл и перезаписать его несколько раз, что делает восстановление гораздо сложнее. К счастью для нас, в Linux есть встроенная команда с подходящим именем `shred` именно для этой цели.

Чтобы понять, как работает команда `shred`, быстро посмотрите на экран справки, введя следующую команду:

```
kali >shred --help
Usage: shred [OPTION]...FILE...
Overwrite the specified FILE(s) repeatedly in order to make it harder
for even very expensive hardware probing to recover data
--snip--
```

Как видно из полного вывода на вашем экране, у команды `shred` много параметров. В самой базовой форме синтаксис прост:

```
shred <FILE>
```

Сама по себе `shred` удалит файл и перезапишет его несколько раз: по умолчанию `shred` перезаписывает четыре раза. Обычно чем больше раз перезаписан файл, тем труднее его восстановить, но имейте в виду, что каждая перезапись занимает время, поэтому для очень больших файлов уничтожение может стать длительным.

Два полезных параметра, которые стоит включить, - это параметр `-f`, который изменяет разрешения на файлы, чтобы разрешить перезапись, если требуется изменение разрешений, и параметр `-n`, который позволяет выбрать, сколько раз перезаписывать файлы. В качестве примера мы уничтожим файлы журналов в `/var/log/auth.log` 10 раз с помощью следующей команды:

```
kali >shred -f -n 10 /var/log/auth.log.*
```

Нам нужен параметр `-f`, чтобы получить разрешение на уничтожение файлов `auth`, а после параметра `-n` мы указываем желаемое число перезаписей. После пути к файлу, который хотим уничтожить, мы включаем подстановочную звездочку, так что уничтожаем не только файл `auth.log`, но и любые журналы, созданные с помощью `logrotate`, например `auth.log.1`, `auth.log.2` и так далее.

Теперь попробуйте открыть файл журнала:

```
kali >leafpad /var/log/auth.log.1
```

После того как вы уничтожили файл с помощью `shred`, вы увидите, что содержимое стало неразборчивой бессмыслицей, как показано на рисунке 11-1.

Рисунок 11-1. Уничтоженный файл журнала

Теперь, если инженер безопасности или специалист по криминалистике изучит файлы журналов, он не найдет ничего полезного, потому что ничего из этого не подлежит восстановлению!

Отключение журналирования

Еще один вариант замести следы - просто отключить журналирование. Когда хакер получает контроль над системой, он может немедленно отключить журналирование, чтобы система не отслеживала его действия. Это, конечно, требует привилегий root.

Чтобы отключить все журналирование, хакер может просто остановить демон `rsyslog`. Остановка любой службы в Linux использует один и тот же синтаксис, показанный здесь (подробнее вы увидите это в главе 12):

```
service servicename start|stop|restart
```

Итак, чтобы остановить демон журналирования, можно просто ввести следующую команду:

```
kali >service rsyslog stop
```

Теперь Linux перестанет создавать любые файлы журналов до перезапуска службы, позволяя вам действовать, не оставляя после себя никаких доказательств в файлах журналов!

Итоги

Файлы журналов отслеживают почти все, что происходит в вашей Linux-системе. Они могут быть бесценным ресурсом при попытке проанализировать, что произошло, будь то сбой или взлом. Для хакера файлы журналов могут быть доказательством его действий и личности. Однако предусмотрительный хакер может удалить и уничтожить эти файлы и полностью отключить журналирование, тем самым не оставив доказательств.

Упражнения

Прежде чем перейти к главе 12, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Используйте команду `locate`, чтобы найти все файлы `rsyslog`.
2. Откройте файл `rsyslog.conf` и измените ротацию журналов на одну неделю.
3. Отключите журналирование в своей системе. Исследуйте, что записывается в файл `/var/log/syslog`, когда вы отключаете журналирование.
4. Используйте команду `shred`, чтобы уничтожить и удалить все файлы журналов `kern`.

Глава 12. Использование служб и злоупотребление ими

В терминологии Linux служба - это приложение, которое работает в фоне, ожидая, когда вы им воспользуетесь. В вашей Linux-системе предустановлены десятки служб. Самая известная из них - повсеместный веб-сервер Apache, который используется для создания, управления и развертывания веб-серверов, но есть и множество других. Для целей этой главы о службах я выбрал всего четыре, которые особенно важны для хакера: веб-сервер Apache, OpenSSH, MySQL и PostgreSQL.

В этой главе вы узнаете, как настроить веб-сервер с помощью Apache, физически шпионить с помощью OpenSSH, получать доступ к данным с помощью MySQL и хранить свою хакерскую информацию с помощью PostgreSQL.

Запуск, остановка и перезапуск служб

Прежде чем мы начнем работать с этими четырьмя важнейшими службами, сначала рассмотрим, как запускать, останавливать и перезапускать службы в Linux.

Некоторые службы в Kali Linux можно останавливать и запускать через GUI, почти так же, как вы делали бы это в операционной системе вроде Windows или Mac. Однако некоторые службы требуют использования командной строки, и именно это мы здесь рассмотрим. Вот базовый синтаксис для управления службами:

```
service servicename start|stop|restart
```

Чтобы запустить службу `apache2` (веб-сервер или HTTP-службу), вы ввели бы следующее:

```
kali >service apache2 start
```

Чтобы остановить веб-сервер Apache, введите:

```
kali >service apache2 stop
```

Обычно, когда вы меняете конфигурацию приложения или службы, изменяя ее текстовый конфигурационный файл, вам нужно перезапустить службу, чтобы она подхватила новую конфигурацию. Поэтому вы ввели бы следующее:

```
kali >service apache2 restart
```

Теперь, когда вы понимаете, как запускать, останавливать и перезапускать службы из командной строки, перейдем к четырем Linux-службам, наиболее критичным для хакеров.

Создание HTTP-веб-сервера с помощью веб-сервера Apache

Веб-сервер Apache, вероятно, самая часто используемая служба в Linux-системах. Apache установлен более чем на 60 процентах мировых веб-серверов, поэтому любой уважающий себя администратор Linux должен быть с ним знаком. Как хакеру, стремящемуся взламывать веб-сайты, вам критически важно понимать внутреннюю работу Apache, веб-сайтов и серверных баз данных этих сайтов. Вы также можете использовать Apache, чтобы настроить собственный веб-сервер, с которого можно отдавать вредоносное ПО через межсайтовый скриптинг (XSS) любому, кто посетит ваш сайт, или можно клонировать веб-сайт и перенаправить трафик на свой сайт через злоупотребление системой доменных имен (DNS). В любом из этих случаев требуется базовое знание Apache.

Начало работы с Apache

Если на вашей системе запущен Kali, Apache уже установлен. Во многих других дистрибутивах Linux он также установлен по умолчанию. Если Apache у вас не установлен, вы можете загрузить и установить его из репозиториев, введя следующее:

```
kali >apt-get install apache2
```

Веб-сервер Apache часто связан с базой данных MySQL (которую мы рассмотрим в следующем разделе), и эти две службы очень часто объединяют со скриптовым языком, таким как Perl или PHP, для разработки веб-приложений. Эта комбинация Linux, Apache, MySQL и PHP или Perl образует мощную и надежную платформу для разработки и развертывания веб-приложений, известную вместе как LAMP. Это самые широко используемые инструменты для разработки веб-сайтов в мире Linux, и они очень популярны и в мире Microsoft, где их обычно называют WAMP, где `W` означает Windows.

Первый шаг, конечно, - запустить наш демон Apache. В Kali перейдите в Applications > Services > HTTPD и щелкните Apache start. То же самое можно сделать из командной строки, введя следующее:

```
kali >service apache2 start
```

Теперь, когда Apache работает, он должен быть способен отдать свою веб-страницу по умолчанию. Введите `http://localhost/` в вашем любимом веб-браузере, чтобы открыть веб-страницу, которая должна выглядеть примерно как на рисунке 12-1.

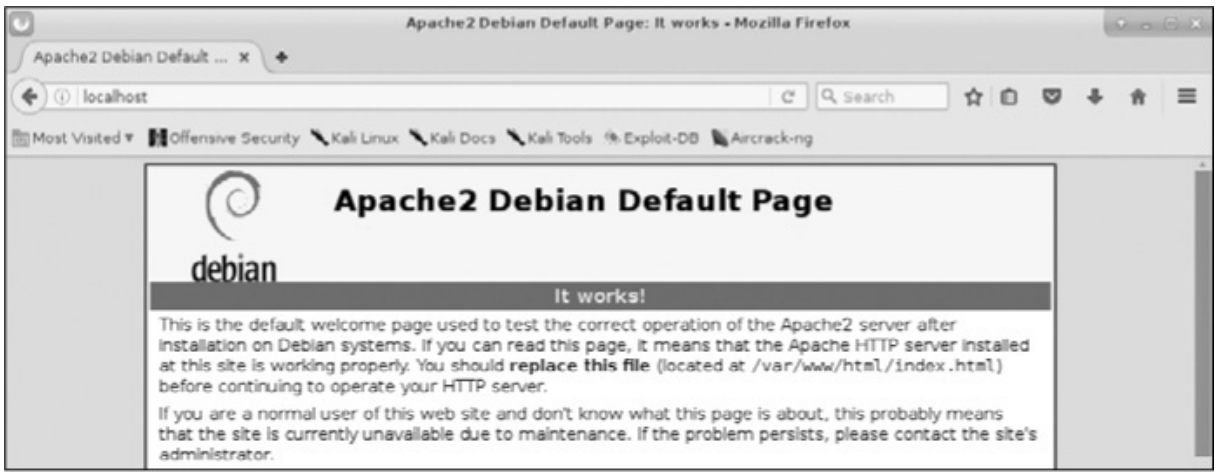

Рисунок 12-1. Страница по умолчанию веб-сервера Apache2

Как видите, Apache отображает "Работает" как свою веб-страницу по умолчанию. Теперь, когда вы знаете, что ваш веб-сервер Apache работает, давайте настроим его под себя!

Редактирование файла index.html

Веб-страница Apache по умолчанию находится по пути `/var/www/html/index.html`. Вы можете отредактировать файл `index.html`, чтобы отдавать любую информацию, которую захотите, так что давайте создадим свою. Для этого можно использовать любой текстовый редактор по вашему выбору; я буду использовать Leafpad. Откройте `/var/www/html/index.html`, и вы должны увидеть что-то похожее на листинг 12-1.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
(1) <title>Страница Apache2 Debian по умолчанию: работает</title>
<style type="text/css" media="screen">
* {
margin: 0px 0px 0px 0px;
padding: 0px 0px 0px 0px;
}
body, html {
padding: 3px 3px 3px 3px;
background-color: #D8DBE2;
font-family: Verdana, sans-serif;
font-size: 11pt;
text-align: center;
}
div.main_page {
position: relative;
display: table;
```

Листинг 12-1. Файл `index.html` *веб-сервера Apache*

Обратите внимание, что веб-страница по умолчанию содержит ровно тот текст, который отображался, когда мы открыли в браузере локальный хост (`localhost`), но в формате HTML (1). Все, что нам нужно сделать, - отредактировать или заменить этот файл, чтобы наш веб-сервер отображал нужную нам информацию.

Добавление HTML

Теперь, когда веб-сервер запущен и работает, а файл `index.html` открыт, мы можем добавить любой текст, который хотели бы отдавать через веб-сервер. Мы создадим несколько простых HTML-блоков.

Давайте создадим эту страницу. В новом файле вашего текстового редактора введите код, показанный в листинге 12-2.

```
<html>
<body>
<h1>Hackers-Arise - лучший! </h1>
<p> Если вы хотите изучать хакинг, Hackers-Arise.com </p>
<p> - лучшее место для изучения хакинга!</p>
</body>
</html>
```

Листинг 12-2. Простой HTML для добавления в файл `index.html`

После того как вы ввели текст точно так, как он показан в листинге 12-2, сохраните этот файл как `/var/www/html/index.html` и закройте текстовый редактор. Затем ваш текстовый редактор сообщит, что файл уже существует. Это нормально. Просто перезапишите существующий файл `/var/www/html/index.html`.

Что получится

Сохранив наш файл `/var/www/html/index.html`, мы можем проверить, что будет отдавать Apache. Снова перейдите в браузере по адресу `http://localhost`, и вы должны увидеть что-то похожее на рисунок 12-2.

Рисунок 12-2. Новый сайт Hackers-Arise

Apache отдал нашу веб-страницу именно так, как мы ее создали!

OpenSSH и шпионская Raspberry Pi

SSH - это аббревиатура от защищенной оболочки, и по сути это то, что позволяет нам безопасно подключаться к терминалу на удаленной системе; это замена небезопасному telnet, который был так распространен много лет назад. Когда мы строим веб-сервер, SSH позволяет нам создать список доступа (список пользователей, которые могут пользоваться этой службой), аутентифицировать пользователей с помощью зашифрованных паролей и шифровать всю связь. Это снижает вероятность того, что нежелательные пользователи будут использовать удаленный терминал (благодаря добавленному процессу аутентификации) или перехватят нашу коммуникацию (благодаря шифрованию). Вероятно, самая широко используемая Linux-служба SSH - OpenSSH, которая установлена почти в каждом дистрибутиве Linux, включая Kali.

Системные администраторы часто используют SSH для управления удаленными системами, а хакеры часто используют SSH для подключения к скомпрометированным удаленным системам, так что мы сделаем то же самое здесь. В этом примере мы используем SSH, чтобы настроить удаленную систему Raspberry Pi для шпионажа, то, что я называю "шпионской Raspberry Pi". Для этого вам понадобится Raspberry Pi и соответствующий модуль камеры Raspberry Pi.

Прежде чем мы сделаем это, запустите OpenSSH в своей Kali-системе уже знакомой командой:

```
kali >service ssh start
```

Мы будем использовать SSH для создания и управления удаленной шпионской Raspberry Pi. Если вы еще не знакомы с ней, Raspberry Pi - это крошечный, но мощный компьютер размером с кредитную карту, который отлично подходит как удаленный инструмент наблюдения. Мы задействуем Raspberry Pi с модулем камеры, чтобы использовать ее как удаленное шпионское устройство. Raspberry Pi можно купить почти в любом магазине электроники, включая Amazon, менее чем за 50 долларов, а модуль камеры - примерно за 15 долларов.

Здесь мы будем использовать шпионскую Raspberry Pi в той же сети, что и наша Kali-система, что позволяет нам использовать частные внутренние IP-адреса. Конечно, при взломе в реальном мире вы, вероятно, захотели бы настроить ее в другой удаленной сети, но это было бы немного сложнее и выходит за рамки этой книги.

Настройка Raspberry Pi

Убедитесь, что на вашей Raspberry Pi работает операционная система Raspbian; это просто еще один дистрибутив Linux, специально портированный для процессора Raspberry Pi. Инструкции по загрузке и установке Raspbian можно найти по адресу `https://www.raspberrypi.org/downloads/raspbian/`. Почти все, что вы узнали в этой книге, применимо к ОС Raspbian на Raspberry Pi так же, как к Kali, Ubuntu и другим дистрибутивам Linux.

После того как вы загрузили и установили ОС Raspbian, вам нужно подключить Raspberry Pi к монитору, мыши и клавиатуре, а затем подключить ее к интернету. Если все это для вас новое, ознакомьтесь с инструкциями по адресу `https://www.raspberrypi.org/learning/hardware-guide/`. Когда все настроено, войдите с именем пользователя `pi` и паролем `raspberry`.

Создание шпионской Raspberry Pi

Первый шаг - убедиться, что SSH работает и включен на шпионской Raspberry Pi. SSH обычно выключен по умолчанию, поэтому, чтобы включить его, перейдите в меню настроек и запустите конфигурацию Raspberry Pi. Затем перейдите на вкладку интерфейсов и рядом с SSH щелкните "Включено" (если он еще не отмечен), а затем нажмите OK.

Когда SSH включен, вы можете запустить его на шпионской Raspberry Pi, открыв терминал и введя следующее:

```
kali >service ssh start
```

Далее вам нужно подключить модуль камеры. Если вы используете плату Raspberry Pi версии 3, подключить его можно только в одно место. Выключите Pi, подключите модуль к порту камеры, а затем снова включите ее. Обратите внимание, что камера очень хрупкая и никогда не должна соприкасаться с контактами ввода-вывода общего назначения (GPIO); иначе может произойти короткое замыкание, и она выйдет из строя.

Теперь, когда служба SSH запущена и работает, поместите шпионскую Raspberry Pi где-нибудь в вашем доме, школе или другом месте, за которым вы хотите наблюдать. Она, конечно, должна быть подключена к локальной сети либо Ethernet-кабелем, либо, в идеале, через Wi-Fi. (Новые Raspberry Pi 3 и Raspberry Pi Zero обе имеют встроенный Wi-Fi.)

Теперь вам нужно получить IP-адрес вашей Raspberry Pi. Как вы узнали в главе 3, IP-адрес Linux-устройства можно получить с помощью `ifconfig`:

```
pi >ifconfig
```

IP-адрес моей Pi - `192.168.1.101`, но убедитесь, что везде, где в этой главе появляется мой адрес, вы используете IP-адрес своей шпионской Raspberry Pi. Теперь из вашей Kali-системы вы должны быть способны напрямую подключаться к своей шпионской Raspberry Pi, управлять ей и использовать ее как удаленную шпионскую систему. В этом простом примере ваша система должна находиться в той же сети, что и Pi.

Чтобы подключиться к удаленной шпионской Raspberry Pi по SSH из вашей Kali-системы, введите следующее, не забыв использовать IP-адрес собственной Pi:

```
kali >ssh pi@192.168.1.101
pi@192.168.1.101's password:
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, the extent
permitted by applicable law
last login: Tues Jan. 1 12:01:01 2018
pi@raspberyypi:: $
```

Затем Spy Pi запросит пароль. В этом случае пароль по умолчанию - `raspberry`, если вы его не меняли.

Настройка камеры

Далее нам нужно настроить камеру. Для этого запустите инструмент конфигурации Raspberry Pi, введя следующую команду:

```
pi >sudo raspi-config
```

Это должно открыть графическое меню, похожее на показанное на рисунке 12-3.

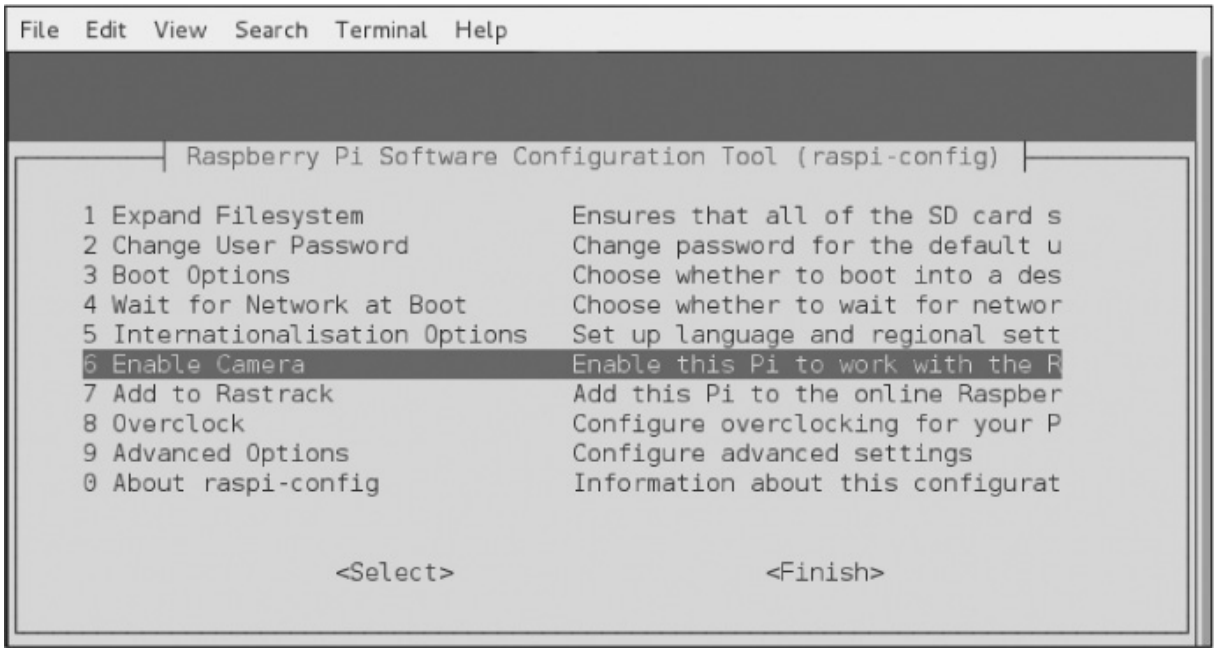

Рисунок 12-3. Инструмент настройки Raspberry Pi

Прокрутите вниз до `6 Enable Camera` и нажмите ENTER. Теперь прокрутите меню вниз до конца, выберите "Готово" и нажмите ENTER, как показано на рисунке 12-4.

Рисунок 12-4. Завершение настройки

Когда инструмент конфигурации спросит, хотите ли вы перезагрузиться, как показано на рисунке 12-5, выберите Yes и снова нажмите ENTER.

Рисунок 12-5. Перезагрузите Pi, чтобы включить изменения.

Теперь камера вашей шпионской Raspberry Pi должна быть включена и готова к наблюдению!

Начало наблюдения

После того как ваша шпионская Raspberry Pi перезагрузилась и вы вошли в нее по SSH из терминала Kali, вы готовы начать использовать ее для наблюдения, делая неподвижные снимки.

В операционной системе Raspbian есть приложение с именем `raspistill`, которое мы будем использовать для получения снимков с нашей маленькой шпионской Raspberry Pi. Введите `raspistill` в терминале, чтобы увидеть экран справки инструмента и все его параметры:

```
pi@raspberrypi: raspistill
raspistill Camera App v1.3.8
Runs camera for specific time, and takes JPG capture at end if requested
usage: raspistill [options]
Image parameter commands
--snip--
```

Теперь используем шпионскую Raspberry Pi, чтобы сделать несколько удаленных шпионских снимков! У команды `raspistill` есть множество параметров, которые вам стоит изучить, но здесь мы просто используем значения по умолчанию. Чтобы сделать снимок и сохранить его как JPEG, введите следующее:

```
pi@raspberrypi: raspistill -v -o firstpicture.jpg
raspistill Camera App v1.3.8
width 2592, Height 1944, quality 85, filename firstpicture.jpg
Time delay 5000, Raw no
--snip--
```

Мы используем параметр `-v`, чтобы получить подробный вывод, и параметр `-o`, чтобы сообщить `raspistill`, что сейчас зададим имя файла для использования; затем мы указываем имя файла. Когда мы выполняем подробный листинг на шпионской Raspberry Pi, мы видим файл `firstpicture.jpg`, как показано здесь:

```
pi@raspberrypi: ls -l
total 2452
drwxr-xr-x 2 pi pi 4096 Mar 18 2019 Desktop
drwxr-xr-x 2 pi pi 4096 Mar 18 2019 Documents
drwxr-xr-x 2 pi pi 4096 Mar 18 2019 Downloads
-rw-r--r-- 1 pi pi 2472219 Mar 18 2019 firstpicture.jpg
drwxr-xr-x 2 pi pi 4096 Mar 18 2019 Music
drwxr-xr-x 2 pi pi 4096 Mar 18 2019 Pictures
--snip--
```

Мы сделали свой самый первый шпионский снимок на удаленной шпионской Raspberry Pi с помощью SSH! Можете свободно изучать это универсальное оружие дальше.

Извлечение информации из MySQL

MySQL - самая широко используемая база данных за веб-приложениями, управляемыми базами данных. В нашу современную эпоху технологий Web 2.0, когда почти каждый веб-сайт управляется базой данных, это означает, что MySQL хранит данные большей части веба.

Базы данных - это "золотое руно" для хакеров. Они содержат критически важную информацию о пользователях, а также конфиденциальную информацию, такую как номера кредитных карт. По этой причине хакеры чаще всего нацеливаются на базы данных.

Как и Linux, MySQL имеет открытый исходный код и распространяется по универсальной общественной лицензии (GPL), и вы найдете ее предустановленной почти в каждом дистрибутиве

Linux.

Будучи бесплатной, открытой и мощной, MySQL стала предпочтительной базой данных для многих веб-приложений, включая популярные веб-сайты, такие как WordPress, Facebook, LinkedIn, Twitter, Kayak, Walmart.com, Wikipedia и YouTube.

Другие популярные системы управления содержимым (CMS), такие как Joomla, Drupal и Ruby on Rails, тоже используют MySQL. Идея понятна. Если вы хотите разрабатывать или атаковать серверные базы данных веб-приложений, вам следует знать MySQL. Давайте начнем.

Запуск MySQL

К счастью, в Kali MySQL уже установлена (если вы используете другой дистрибутив, вы можете загрузить и установить MySQL из программного репозитория или напрямую с `https://www.mysql.com/downloads/`).

Чтобы запустить службу MySQL, введите в терминал следующее:

```
kali >service mysql start
```

Далее вам нужно аутентифицироваться, войдя в систему. Введите следующее и, когда будет запрошен пароль, просто нажмите ENTER:

```
kali >mysql -u root -p
Enter password:
Welcome to MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.6.30-1 (Debian)
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
mysql >
```

В конфигурации MySQL по умолчанию пароль пользователя root пустой. Очевидно, это серьезная уязвимость безопасности, и вам следует исправить ее, добавив пароль после первого входа. Обратите внимание, что имена пользователей и пароли вашей операционной системы и MySQL являются отдельными и различными. Давайте сейчас изменим пароль для пользователя MySQL `root`, чтобы быть в безопасности.

> Прошлое и будущее MySQL
>
> MySQL была впервые разработана компанией MySQL AB из Швеции в 1995 году, затем была приобретена Sun Microsystems в 2008 году, которую, в свою очередь, приобрела Oracle в 2009 году, так что MySQL теперь принадлежит Oracle. Oracle - крупнейший в мире издатель программного обеспечения для баз данных, поэтому у сообщества открытого исходного кода есть серьезные опасения относительно приверженности Oracle сохранению MySQL открытой. В результате теперь существует форк программного обеспечения базы данных MySQL под названием "Maria", который привержен сохранению этого программного обеспечения и его последующих версий открытыми. Как администратору Linux или хакеру, вам следует следить за Maria.

Взаимодействие с MySQL

SQL - это интерпретируемый язык программирования для взаимодействия с базой данных. База данных часто является реляционной, что означает, что данные хранятся в нескольких таблицах, которые взаимодействуют между собой, и каждая таблица имеет значения в одном или нескольких столбцах и строках.

Есть несколько реализаций SQL, каждая со своими командами и синтаксисом, но вот несколько распространенных команд:

Команда	Назначение
`select`	Используется для извлечения данных
`union`	Используется для объединения результатов двух или более операций `select`
`insert`	Используется для добавления новых данных
`update`	Используется для изменения существующих данных
`delete`	Используется для удаления данных

К каждой команде можно добавлять условия, чтобы точнее указать, что вы хотите сделать. Например, строка

```
select user, password from customers where user='admin';
```

вернет значения полей `user` и `password` для любого пользователя, чье значение `user` равно "admin" в таблице `customers`.

Настройка пароля MySQL

Давайте посмотрим, какие пользователи уже есть в нашей системе MySQL, введя следующее. (Обратите внимание, что команды в MySQL завершаются точкой с запятой.)

```
mysql >select user, host, password from mysql.user;
+-------------------------------------------------------------------------
| user      | host        | password
+-------------------------------------------------------------------------
|root       |localhost     |
|root       |aphrodite.kali.org |
|root       |127.0.0.1     |
--snip--
```

Это показывает, что для пользователей `root` пароль не задан. Давайте назначим пароль для root. Для этого сначала выберем базу данных, с которой будем работать. MySQL в вашей системе поставляется с уже настроенными базами данных. Используйте команду `show databases;`, чтобы увидеть все доступные базы данных:

```
mysql >show databases;
+------------------------------+
| Database       |
+------------------------------+
| information_schema  |
| mysql         |
| performance_schema  |
+------------------------------+
3 rows in set (0.23 sec)
```

MySQL по умолчанию поставляется с тремя базами данных, две из которых (`information_schema` и `performance_schema`) являются административными базами данных, которые мы здесь не будем использовать. Мы будем использовать неадминистративную базу данных `mysql`, включенную для ваших собственных целей. Чтобы начать использовать базу данных `mysql`, введите:

```
mysql >use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
```

Эта команда подключает нас к `mysql`. Теперь мы можем установить пароль для пользователя `root` равным `hackers-arise` с помощью следующей команды:

```
mysql >update user set password = PASSWORD("hackers-arise") where user =
'root';
```

Эта команда обновит пользователя, задав пароль пользователя root равным `hackers-arise`.

Доступ к удаленной базе данных

Чтобы получить доступ к базе данных MySQL на локальном хосте (`localhost`), мы используем следующий синтаксис:

```
kali >mysql -u <username> -p
```

Эта команда по умолчанию использует экземпляр MySQL на локальном хосте (`localhost`), если ей не передано имя хоста или IP-адрес. Чтобы получить доступ к удаленной базе данных, нам нужно указать имя хоста или IP-адрес системы, на которой размещена база данных MySQL. Вот пример:

```
kali >mysql -u root -p 192.168.1.101
```

Это подключит нас к экземпляру MySQL по адресу `192.168.1.101` и запросит пароль. Для демонстрационных целей я подключаюсь к экземпляру MySQL в моей локальной сети (LAN). Если в вашей сети есть система с установленной MySQL, используйте здесь ее IP-адрес. Я буду предполагать, что вам удалось обойти пароль и войти в систему как root (вы уже знаете, что по умолчанию у базы данных `mysql` нет пароля).

Это открывает интерфейс командной строки MySQL, который предоставляет нам приглашение `mysql >`. Помимо этого интерфейса командной строки у MySQL есть графические интерфейсы - как родные (MySQL Workbench), так и сторонние (Navicat и TOAD for MySQL). Для вас как хакера интерфейс командной строки может быть лучшей возможностью для эксплуатации базы данных MySQL, поэтому здесь мы сосредоточимся на нем. Маловероятно, что как неавторизованному вошедшему в базу данных пользователю вам предоставят удобный графический интерфейс.

> Примечание
>
> Этот экран напоминает нам, что все команды должны заканчиваться точкой с запятой или `\g` (в отличие от Microsoft SQL Server), и что мы можем получить справку, введя `help;` или `\h`.

Теперь, когда мы вошли как системный администратор, мы можем беспрепятственно перемещаться по базе данных. Если бы мы вошли как обычный пользователь, наша навигация была бы ограничена разрешениями, предоставленными системным администратором этому пользователю.

Подключение к базе данных

Получив доступ к системе, мы хотим осмотреться. Наш следующий шаг - выяснить, есть ли там какие-либо базы данных, к которым стоит получить доступ. Вот команда, чтобы узнать, какие базы данных есть в доступной системе:

```
mysql >show databases;
+------------------------------+
| Database       |
+------------------------------+
| information schema |
| mysql         |
| creditcardnumbers  |
| performance_schema  |
+------------------------------+
4 rows in set (0.26 sec)
```

Ага! Мы нашли базу данных, которую стоит исследовать, с именем `creditcardnumbers`. Давайте подключимся к ней.

В MySQL, как и в других системах управления базами данных (DBMS), мы можем подключиться к интересующей нас базе данных, введя `use databasename;`.

```
mysql >use creditcardnumbers;
Database changed
```

Ответ `Database changed` указывает, что теперь мы подключены к базе данных `creditcardnumbers`.

Конечно, должно быть само собой понятно, что администратор базы данных вряд ли будет настолько услужлив, чтобы назвать базу данных чем-то так легко распознаваемым, как `creditcardnumbers`, поэтому вам может понадобиться немного исследовать систему, чтобы найти интересующую базу данных.

Таблицы базы данных

Теперь мы подключены к базе данных `creditcardnumbers` и можем немного исследовать, какую информацию она может содержать. Данные в базе данных организованы в таблицы, и каждая таблица может содержать другой набор связанных данных. Мы можем узнать, какие таблицы есть в этой базе данных, введя следующую команду:

```
mysql >show tables;
+----------------------------------+
| Tables_in_creditcardnumbers |
+----------------------------------+
| cardnumbers        |
+----------------------------------+
1 row in set (0.14 sec)
```

Здесь видно, что в этой базе данных есть только одна таблица, называемая `cardnumbers`. Обычно в базах данных много таблиц, поэтому, скорее всего, вам придется еще немного поразведать. В этой примерной базе данных нам повезло: мы можем сосредоточить внимание на этой единственной таблице, чтобы извлечь хакерское золотое руно!

Теперь, когда у нас есть таблица, которую мы хотим изучить, нам нужно понять структуру этой таблицы. Когда мы узнаем, как устроена таблица, мы сможем извлечь нужную информацию.

Структуру таблицы можно увидеть с помощью оператора `describe`, вот так:

```
mysql >describe cardnumbers;
+--------------+-------------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| customers | varchar(15) | YES | | NULL |   |
| address | varchar(15) | YES | | NULL |    |
| city  | varchar(15) | YES | | NULL |    |
| state | varchar(15) | YES | | NULL  |   |
| cc   | int(12) | NO |    | 0   |   |
+--------------+-------------+------+-----+---------+-------+
```

MySQL отвечает критически важной информацией о структуре интересующей нас таблицы. Мы видим имя каждого поля, а также тип данных, который оно хранит (часто текстовый тип `varchar` или целочисленный тип `int`). Мы также видим, будет ли оно принимать значения `NULL`; ключ, если он существует (ключ связывает таблицы); любые значения по умолчанию, которые может иметь поле; и любую дополнительную информацию в конце, например примечания.

Изучение данных

Чтобы действительно увидеть данные в таблице, мы используем команду `SELECT`. Команда `SELECT` требует, чтобы вы знали следующую информацию:

Таблица, которая содержит данные, которые вы хотите просмотреть

Столбцы внутри этой таблицы, которые содержат данные, которые вы хотите просмотреть

Мы задаем это в следующем формате:

```
SELECT columns FROM table
```

В качестве удобного сокращения, чтобы посмотреть данные из всех столбцов, можно использовать звездочку как подстановочный знак вместо того, чтобы вводить каждое имя столбца, который мы хотим посмотреть. Итак, чтобы увидеть дамп всех данных из таблицы `cardnumbers`, мы вводим следующее:

```
mysql >SELECT * FROM cardnumbers;
+-----------+--------------+-------------+-------+------------+
| customers | address | city | state | cc |
+-----------+--------------+-------------+-------+------------+
| Jones | 1 Wall St | NY | NY  | 12345678 |
| Sawyer | 12 Piccadilly | London | UK | 234567890 |
| Doe  | 25 Front St | Los Angeles | CA | 4567898877 |
+-----------+--------------+-------------+-------+------------+
```

Как видите, MySQL вывела всю информацию из таблицы `cardnumbers` на наш экран. Мы нашли хакерское золотое руно!

PostgreSQL с Metasploit

PostgreSQL, или просто Postgres, - еще одна реляционная база данных с открытым исходным кодом, часто используемая в очень крупных интернет-приложениях благодаря своей способности легко масштабироваться и справляться с тяжелыми рабочими нагрузками. Она была впервые выпущена в июле 1996 года и поддерживается большой группой разработчиков, известной как PostgreSQL Global Development Group.

PostgreSQL также установлена в Kali по умолчанию, но если вы используете другой дистрибутив Linux, она, скорее всего, будет в вашем репозитории, и вы можете установить ее, введя следующую команду:

```
kali >apt-get postgres install
```

Как хакер, вы найдете PostgreSQL особенно важной, потому что это база данных по умолчанию для самого широко используемого фреймворка тестирования на проникновение и хакинга, Metasploit. Metasploit использует PostgreSQL, чтобы хранить свои модули, а также результаты сканирований и эксплуатации, для удобства использования в тесте на проникновение или взломе. По этой причине мы будем использовать PostgreSQL здесь в контексте Metasploit.

Как почти все службы в Linux, мы можем запустить PostgreSQL, введя `service имя_приложения start`, вот так:

```
kali >service postgresql start
```

Когда PostgreSQL запущена и работает, давайте запустим Metasploit:

```
kali >msfconsole
```

Обратите внимание, что когда Metasploit завершит запуск, вы увидите приглашение `msf >`.

Обучение использованию Metasploit для целей хакинга и эксплуатации выходит за рамки этой книги, но здесь мы настроим базу данных, в которой Metasploit будет хранить свою информацию.

Когда Metasploit работает, мы можем настроить PostgreSQL следующей командой, чтобы она хранила данные любой активности Metasploit в вашей системе:

```
msf >msfdb init
[*] exec :msfdb init
Создание базы данных для использования 'msf'
Введите пароль для новой роли
Введите его снова:
Создание баз данных 'msf' и 'msf_test'
Создание конфигурационного файла /usr/share/metasploit-framework/config/database.yml
Создание начальной схемы базы данных
```

Далее нам нужно войти в Postgres как root. Здесь перед командой мы ставим `su`, команду "switch user", чтобы получить привилегии root:

```
msf >su postgres
[*] su postgres
postgres@kali:/root$
```

Когда вы входите в Postgres, вы увидите, что приглашение изменилось на `postgres@kali:/root$`, представляя приложение, имя хоста и пользователя.

На следующем шаге нам нужно создать пользователя и пароль, вот так:

```
postgres@kali:/root$ createuser msf_user -P
Введите пароль для новой роли:
Введите его снова:
```

Мы создаем имя пользователя `msf_user`, используя параметр `-P` с командой `createuser`. Затем дважды введите желаемый пароль. Далее нужно создать базу данных и предоставить разрешения для `msf_user`. Назовите базу данных `hackers_arise_db`, как показано здесь:

```
postgres@kali:/root$ createdb --owner=msf_user hackers_arise_db
postgres@kali:/root$ exit
```

Когда вы выходите из Postgres командой `exit`, терминал вернется к приглашению `msf >`.

Далее нам нужно подключить нашу консоль Metasploit, `msfconsole`, к нашей базе данных PostgreSQL, определив следующее:

• пользователь;
• пароль;
• хост;
• имя базы данных.

В нашем случае мы можем подключить `msfconsole` к нашей базе данных следующей командой:

```
msf >db_connect msf_user:password@127.0.0.1/hackers_arise_db
```

Конечно, вам нужно указать пароль, который вы использовали ранее. IP-адрес - это адрес вашей локальной системы (локального хоста, `localhost`), поэтому можно использовать `127.0.0.1`, если только вы не создали эту базу данных на удаленной системе.

Наконец, мы можем проверить статус базы данных PostgreSQL, чтобы убедиться, что она подключена:

```
msf >db_status
[*] postgresql connected to msf
```

Как видите, Metasploit отвечает, что база данных PostgreSQL подключена и готова к использованию. Теперь, когда мы выполняем сканирование системы или запускаем эксплуатации с Metasploit, результаты будут храниться в нашей базе данных PostgreSQL. Кроме того, Metasploit теперь хранит свои модули в нашей базе данных Postgres, что делает поиск нужного модуля намного проще и быстрее!

Итоги

В Linux есть множество служб, которые работают в фоне, пока пользователь не будет нуждаться в них. Веб-сервер Apache используется наиболее широко, но хакеру также следует быть знакомым с MySQL, SSH и PostgreSQL для разных задач. В этой главе мы рассмотрели самые основы начала работы с этими службами. Когда вы освоитесь со своей Linux-системой, я настоятельно советую вам самостоятельно изучить каждую из этих служб глубже.

Упражнения

Прежде чем перейти к главе 13, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Запустите службу `apache2` через командную строку.
2. Используя файл `index.html`, создайте простой веб-сайт, объявляющий о вашем вступлении в захватывающий мир хакинга.
3. Запустите службу SSH через командную строку. Теперь подключитесь к своей Kali-системе с другой системы в вашей LAN.
4. Запустите службу базы данных MySQL и измените пароль пользователя root на `hackers-arise`. Перейдите к базе данных `mysql`.
5. Запустите службу базы данных PostgreSQL. Настройте ее, как описано в этой главе, для использования Metasploit.

Глава 13. Становимся защищенными и анонимными

Сегодня почти все, что мы делаем в интернете, отслеживается. Кто бы ни занимался отслеживанием - будь то Google, отслеживающий наши онлайн-поиски, посещения сайтов и электронную почту, или Агентство национальной безопасности (NSA), каталогизирующее все наши действия, - каждый наш шаг в сети записывается, индексируется, а затем добывается для чьей-то выгоды. Обычный человек, и особенно хакер, должен понимать, как ограничить это отслеживание и оставаться относительно анонимным в вебе, чтобы ограничить эту повсеместную слежку.

В этой главе мы рассмотрим, как можно перемещаться по Всемирной паутине анонимно (или настолько близко к этому, насколько возможно), используя четыре метода:

• сеть луковой маршрутизации (Tor);
• прокси-серверы;
• виртуальные частные сети;
• частная зашифрованная электронная почта.

Ни один метод не гарантирует, что ваши действия будут защищены от любопытных глаз, и при достаточном времени и ресурсах можно отследить что угодно. Однако эти методы, вероятно, значительно усложнят работу отслеживающего.

Как интернет нас выдает

Для начала обсудим на высоком уровне некоторые способы, которыми отслеживаются наши действия в интернете. Мы не будем углубляться во все методы отслеживания или слишком подробно рассматривать какой-либо один метод, поскольку это выходит за рамки этой книги. В самом деле, такое обсуждение само по себе могло бы занять целую книгу.

Во-первых, ваш IP-адрес идентифицирует вас, пока вы перемещаетесь по интернету. Данные, отправляемые с вашей машины, обычно помечены вашим IP-адресом, что упрощает отслеживание ваших действий. Во-вторых, Google и другие почтовые сервисы будут "читать" вашу электронную почту, ища ключевые слова, чтобы эффективнее показывать вам рекламу. Хотя существует множество более сложных методов, которые требуют гораздо больше времени и ресурсов, именно эти методы мы пытаемся предотвратить в этой главе. Давайте начнем с рассмотрения того, как IP-адреса выдают нас в интернете.

Когда вы отправляете пакет данных через интернет, он содержит IP-адреса источника и назначения данных. Так пакет знает, куда он идет и куда возвращать ответ. Каждый пакет перескакивает через несколько интернет-маршрутизаторов, пока не найдет свое назначение, а затем перескакивает обратно к отправителю. Для обычного веб-серфинга каждый переход - это маршрутизатор, через который проходит пакет, чтобы добраться до назначения. Между отправителем и назначением может быть до 20-30 переходов, но обычно любой пакет находит путь к назначению менее чем за 15 переходов.

Пока пакет проходит через интернет, любой, кто перехватывает пакет, может увидеть, кто его отправил, где он был и куда направляется. Это один из способов, которым веб-сайты могут понять, кто вы, когда вы приходите, и автоматически войти в систему за вас, и это также способ, которым кто-то может отследить, где вы были в интернете.

Чтобы увидеть, какие переходы может сделать пакет между вами и назначением, можно использовать команду `traceroute`, как показано далее. Просто введите `traceroute` и IP-адрес или домен назначения, и команда отправит пакеты к назначению и отследит маршрут этих пакетов.

```
kali >traceroute google.com
traceroute to google.com (172.217.1.78), 30 hops max, 60 bytes packets
1 192.168.1.1 (192.168.1.1) 4.152 ms 3.834 ms 32.964 ms
2 10.0.0.1 (10.0.0.1) 5.797 ms 6.995 ms 7.679 ms
3 96.120.96.45 (96.120.96.45) 27.952 ms 30.377 ms 32.964 ms
--snip--
18 lgal15s44-in-f14.le100.net (172.217.1.78) 94.666 ms 42.990 ms 41.564 ms
```

Как видите, `www.google.com` находится от меня через 18 переходов по интернету. Ваши результаты, скорее всего, будут другими, потому что ваш запрос будет исходить из другого места и потому что у Google много серверов по всему миру. Кроме того, пакеты не всегда идут по одному и тому же маршруту через интернет, поэтому вы можете отправить другой пакет со своего адреса на тот же сайт и получить другой маршрут. Давайте посмотрим, как мы можем скрыть все это с помощью сети Tor.

Система луковой маршрутизации

В 1990-х Управление военно-морских исследований США (ONR) поставило задачу разработать метод анонимного перемещения по интернету для целей шпионажа. План состоял в том, чтобы настроить сеть маршрутизаторов, отдельную от интернет-маршрутизаторов, которая могла бы шифровать трафик и хранила бы только незашифрованный IP-адрес предыдущего маршрутизатора; это означало, что все остальные адреса маршрутизаторов по пути были зашифрованы. Идея состояла в том, что любой, кто наблюдает за трафиком, не смог бы определить источник или назначение данных. Это исследование стало известно как проект луковой маршрутизации (Tor) в 2002 году, и теперь оно доступно любому для относительно безопасного и анонимного перемещения по вебу.

Как работает Tor

Пакеты, отправленные через Tor, не отправляются через обычные маршрутизаторы, которые так тщательно отслеживаются многими, а идут через сеть из более чем 7 000 маршрутизаторов по всему миру благодаря добровольцам, которые позволяют Tor использовать их компьютеры. Помимо использования полностью отдельной сети маршрутизаторов, Tor шифрует данные, назначение и IP-адрес отправителя каждого пакета. На каждом переходе информация шифруется, а затем расшифровывается следующим переходом при получении. Таким образом, каждый пакет содержит информацию только о предыдущем переходе по пути, а не IP-адрес источника. Если кто-то перехватывает трафик, он может увидеть только IP-адрес предыдущего перехода, а владелец веб-сайта может увидеть только IP-адрес последнего маршрутизатора, отправившего трафик (см. рисунок 13-1). Это обеспечивает относительную анонимность в интернете.

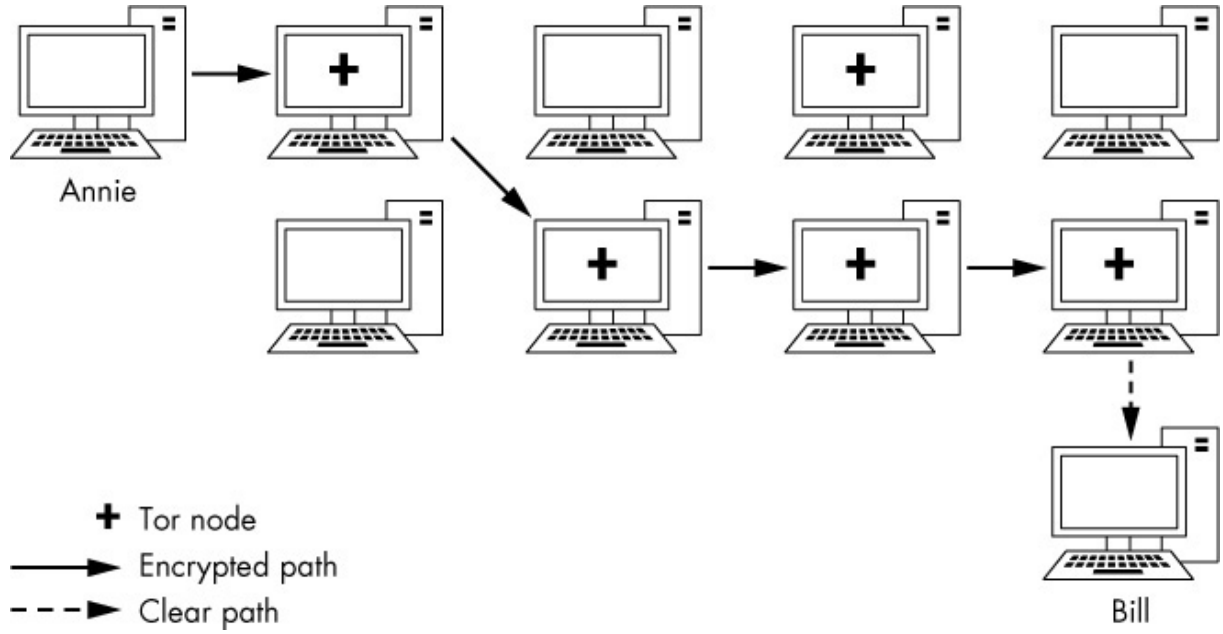

Рисунок 13-1. Как Tor использует зашифрованные данные трафика

Чтобы включить использование Tor, просто установите Tor Browser с `https://www.torproject.org/`. После установки он будет выглядеть примерно как на рисунке 13-2, и вы сможете использовать его как любой обычный интернет-браузер. Используя этот браузер, вы будете перемещаться по интернету через отдельный набор маршрутизаторов и сможете посещать сайты без отслеживания со стороны Большого Брата. К сожалению, компромисс в том, что серфинг через Tor Browser может быть гораздо медленнее; поскольку маршрутизаторов значительно меньше, пропускная способность этой сети ограничена.

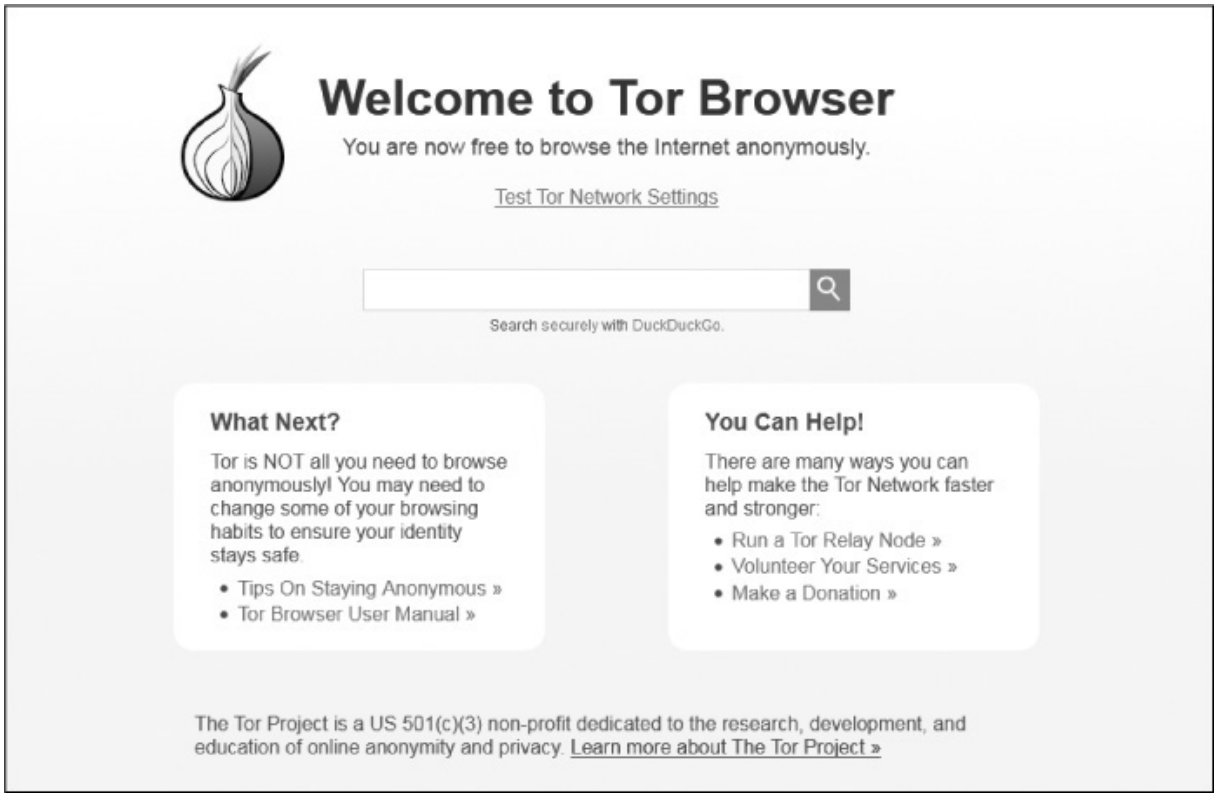

Рисунок 13-2. Стартовая страница Tor Browser

Помимо способности обращаться почти к любому веб-сайту в традиционном интернете, Tor Browser способен получать доступ к dark web. Веб-сайты, составляющие dark web, требуют анонимности, поэтому они разрешают доступ только через Tor Browser и имеют адреса, оканчивающиеся на `.onion` в качестве top-level domain (TLD). Dark web печально известен незаконной деятельностью, но там также доступен ряд легитимных сервисов. Однако одно предупреждение: при доступе к dark web вы можете столкнуться с материалами, которые многие сочтут оскорбительными.

Вопросы безопасности прокси

Разведывательные и шпионские службы Соединенных Штатов и других стран считают сеть Tor угрозой национальной безопасности, полагая, что такая анонимная сеть позволяет иностранным правительствам и террористам общаться без наблюдения. В результате ряд мощных и амбициозных исследовательских проектов работает над нарушением анонимности Tor.

Анонимность Tor уже раньше нарушалась этими органами и, вероятно, будет нарушена снова. NSA, например, запускает собственные маршрутизаторы Tor, а это означает, что ваш трафик может проходить через маршрутизаторы NSA, когда вы используете Tor. Если ваш трафик выходит через маршрутизаторы NSA, это еще хуже, потому что выходной маршрутизатор всегда знает ваше назначение. У NSA также есть метод, известный как traffic correlation, который включает поиск шаблонов во входящем и исходящем трафике и уже был способен нарушать анонимность Tor. Хотя эти попытки взломать Tor не повлияют на эффективность Tor при сокрытии вашей личности от коммерческих сервисов, таких как Google, они могут ограничить эффективность браузера в сохранении вашей анонимности от шпионских агентств.

Прокси-серверы

Другая стратегия достижения анонимности в интернете - использовать прокси, то есть промежуточные системы, которые действуют как посредники для трафика: пользователь подключается к прокси, и трафику присваивается IP-адрес прокси, прежде чем он передается дальше (см. рисунок 13-3). Когда трафик возвращается от назначения, прокси отправляет трафик обратно источнику. Таким образом, трафик выглядит так, будто он исходит от прокси, а не от исходного IP-адреса.

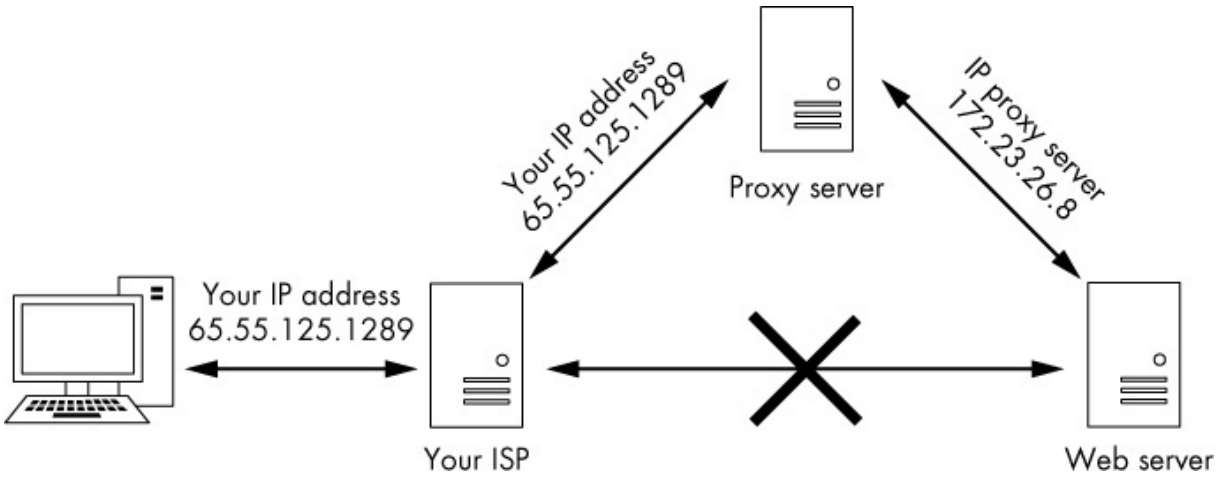

Рисунок 13-3. Пропуск трафика через прокси-сервер

Конечно, прокси, скорее всего, будет журналировать ваш трафик, но следователю пришлось бы получить повестку или ордер на обыск, чтобы получить журналы. Чтобы сделать ваш трафик еще труднее отслеживаемым, можно использовать больше одного прокси в стратегии, известной как цепочка прокси, которую мы рассмотрим немного позже в этой главе.

В Kali Linux есть отличный инструмент проксирования под названием `proxychains`, который можно настроить, чтобы скрыть ваш трафик. Синтаксис команды `proxychains` прост, как показано здесь:

```
kali >proxychains <команда, которую нужно пустить через прокси> <аргументы>
```

Аргументы, которые вы предоставляете, могут включать IP-адрес. Например, если вы хотели бы использовать `proxychains`, чтобы анонимно сканировать сайт с помощью `nmap`, вы ввели бы следующее:

```
kali >proxychains nmap -sT -Pn <IP-адрес>
```

Это отправило бы команду скрытного сканирования `nmap -sS` к указанному IP-адресу через прокси. Затем инструмент сам строит цепочку прокси, так что вам не нужно об этом беспокоиться.

Настройка прокси в конфигурационном файле

В этом разделе мы зададим прокси, который будет использовать команда `proxychains`. Как почти у каждого приложения в Linux/Unix, конфигурация `proxychains` управляется конфигурационным файлом, конкретно `/etc/proxychains.conf`. Откройте конфигурационный файл в выбранном текстовом редакторе следующей командой (заменив `leafpad` на выбранный редактор, если нужно):

```
kali >leafpad /etc/proxychains.conf
```

Вы должны увидеть файл, похожий на показанный в листинге 13-1.

```
# proxychains.conf VER 3.1
# Проксификатор туннелирования HTTP, SOCKS4, SOCKS5 с DNS.
```

```
# Параметр ниже определяет, как обрабатывается ProxyList.
# Одновременно должен быть раскомментирован только один параметр,
# иначе будет принят последний встретившийся параметр
#
# dynamic_chain
#
# Динамическая цепочка - каждое соединение будет выполнено через цепочку прокси
# все прокси соединяются в цепочку в порядке появления в списке
# хотя бы один прокси должен быть онлайн, чтобы участвовать в цепочке
# (неработающие прокси пропускаются)
# иначе приложению возвращается EINTR
# Строгая цепочка - каждое соединение будет выполнено через цепочку прокси
# все прокси соединяются в цепочку в порядке появления в списке
# все прокси должны быть онлайн, чтобы участвовать в цепочке
# иначе приложению возвращается EINTR
--snip--
```

Листинг 13-1. Файл `proxychains.conf`

Прокрутите этот файл вниз до строки 61, и вы должны увидеть раздел `ProxyList`, как показано в листинге 13-2.

```
[ProxyList]
# добавьте прокси здесь...
# пока
# значения по умолчанию заданы для "tor"
socks4 127.0.0.1 9050
```

Листинг 13-2. Раздел конфигурационного файла для добавления прокси

Мы можем добавить прокси, введя в этот список IP-адреса и порты прокси, которые хотим использовать. Пока мы будем использовать несколько бесплатных прокси. Бесплатные прокси можно найти, погуглив "free proxies" или используя сайт `http://www.hidemy.name`, как показано на рисунке 13-4. Однако обратите внимание, что использовать бесплатные прокси в реальной хакерской деятельности - плохая идея. Я подробнее расскажу об этом позже в главе. Пример, используемый здесь, приведен только для образовательных целей.

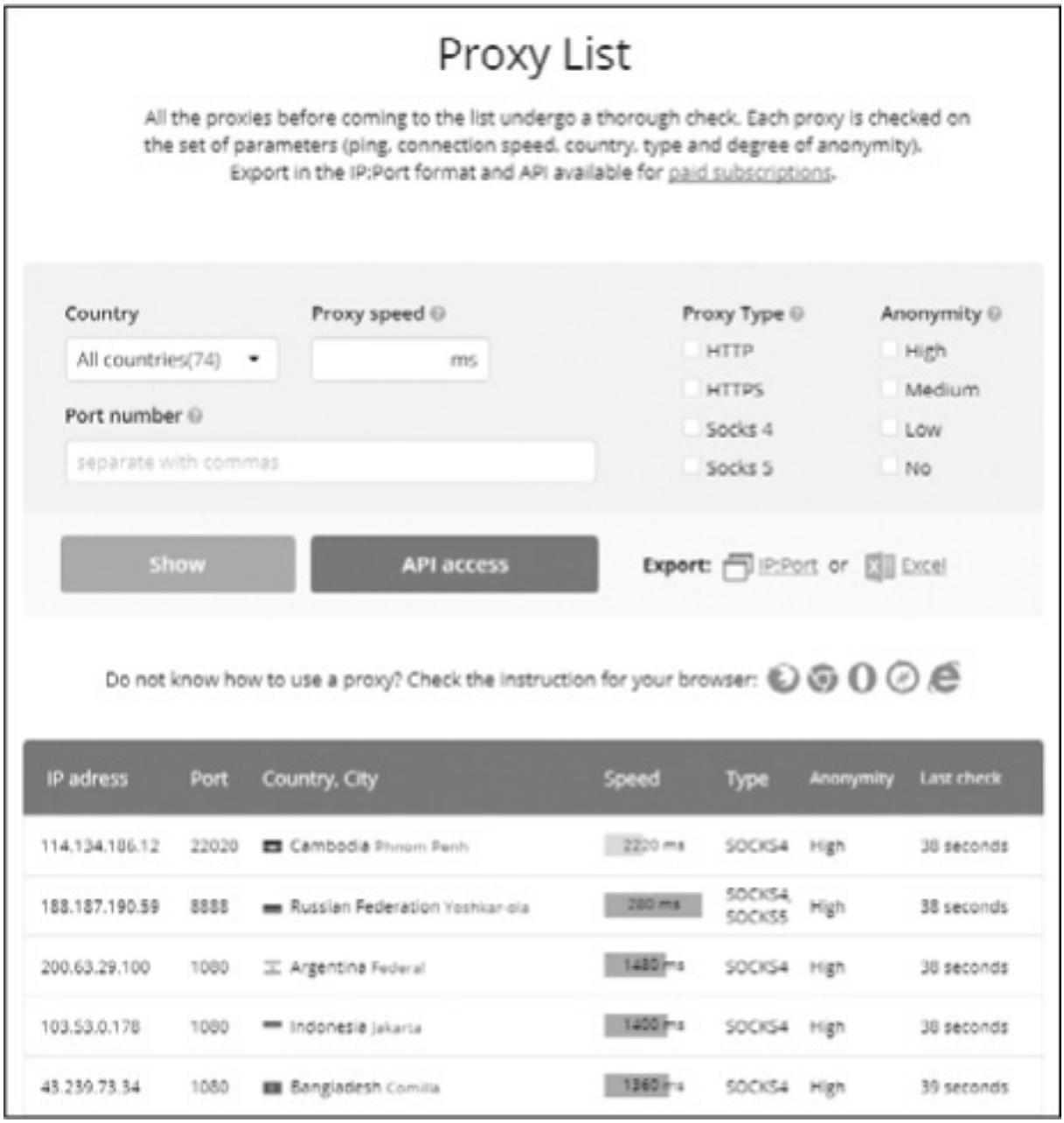

Рисунок 13-4. Бесплатные прокси с `http://www.hidemy.name`

Заполните данные в форме или просто щелкните search; затем добавьте один из полученных прокси в свой файл `proxychains.conf`, используя следующий формат:

```
Type IPaddress Port
```

Вот пример:

```
[ProxyList]
# добавьте прокси здесь...
socks4 114.134.186.12 22020
# пока
# значения по умолчанию заданы для "tor"
# socks4 127.0.0.1 9050
```

Важно отметить, что `proxychains` по умолчанию использует Tor, если вы не вводите собственные прокси. Последняя строка в листинге 13-2 указывает `proxychains` сначала отправлять трафик через хост `127.0.0.1` на порту `9050` (конфигурация Tor по умолчанию). Если вы не добавляете собственные прокси и хотите использовать Tor, оставьте это как есть. Если вы не используете Tor,

вам нужно закомментировать эту строку (добавить перед ней #).

Как бы мне ни нравился Tor, как упоминалось, он обычно очень медленный. Кроме того, поскольку NSA нарушило анонимность Tor, я гораздо менее склонен полагаться на него для анонимности. Поэтому я закомментирую эту строку и добавлю свой собственный набор прокси.

Давайте проверим это. В этом примере я собираюсь открыть браузер Firefox и заставить его анонимно перейти на `https://www.hackers-arise.com/`, отправив трафик через прокси.

Команда выглядит так:

```
kali >proxychains firefox www.hackers-arise.com
```

Это успешно открывает `https://www.hackers-arise.com/` в Firefox через выбранный мной прокси и возвращает результаты мне. Для любого, кто отслеживает этот трафик, выглядит так, будто именно мой прокси перешел на `https://www.hackers-arise.com/`, а не мой IP-адрес.

Еще несколько интересных параметров

Теперь, когда `proxychains` у нас работает, давайте рассмотрим некоторые другие параметры, которые можно настроить через файл `proxychains.conf`. В текущей настройке мы просто используем один прокси. Однако мы можем добавить несколько прокси и использовать их все, можем использовать ограниченное количество из списка или можем заставить `proxychains` случайно менять порядок. Давайте попробуем все эти варианты.

Добавление большего количества прокси

Сначала добавим еще несколько прокси в наш список. Вернитесь на `http://www.hidemy.name` и найдите еще несколько IP-адресов прокси. Затем добавьте еще несколько таких прокси в свой файл `proxychains.conf`, вот так:

```
[ProxyList]
# добавьте прокси здесь...
socks4 114.134.186.12 22020
socks4 188.187.190.59 8888
socks4 181.113.121.158 335551
```

Теперь сохраните этот конфигурационный файл и попробуйте выполнить следующую команду:

```
kali >proxychains firefox www.hackers-arise.com
```

Вы не заметите разницы, но ваш пакет теперь проходит через несколько прокси.

Динамическая цепочка

С несколькими IP в нашем файле `proxychain.conf` мы можем настроить динамическую цепочку, которая проводит наш трафик через каждый прокси в нашем списке и, если один из прокси не работает или не отвечает, автоматически переходит к следующему прокси в списке, не выдавая ошибку. Если бы мы это не настроили, один отказавший прокси сломал бы наш запрос.

Вернитесь в конфигурационный файл `proxychains`, найдите строку `dynamic_chain` (строка 10) и раскомментируйте ее, как показано далее. Также убедитесь, что вы закомментировали строку `strict_chain`, если она еще не закомментирована.

```
# dynamic_chain
#
```

```
# Динамическая цепочка - каждое соединение будет выполнено через цепочку прокси
# все прокси соединяются в цепочку в порядке появления в списке
# хотя бы один прокси должен быть онлайн, чтобы участвовать в цепочке
--snip--
```

Это включит динамическую цепочку наших прокси, тем самым обеспечивая большую анонимность и беспроблемный хакинг. Сохраните конфигурационный файл и можете попробовать.

Случайная цепочка

Наш последний трюк с прокси - параметр случайной цепочки, при котором `proxychains` случайным образом выбирает набор IP-адресов из нашего списка и использует их, чтобы создать нашу цепочку прокси. Это означает, что каждый раз, когда мы используем `proxychains`, прокси будет выглядеть для цели иначе, что усложняет отслеживание нашего трафика от его источника. Этот параметр также считается "динамическим", потому что если один из прокси не работает, он перейдет к следующему.

Вернитесь внутрь файла `/etc/proxychains.conf` и закомментируйте строки `dynamic_chain` и `strict_chain`, добавив # в начале каждой строки; затем раскомментируйте строку `random_chain`. Мы можем использовать только один из этих трех параметров одновременно, поэтому убедитесь, что закомментировали остальные параметры перед использованием `proxychains`.

Далее найдите и раскомментируйте строку с `chain_len`, а затем задайте ей разумное число. Эта строка определяет, сколько IP-адресов в вашей цепочке будет использовано при создании случайной цепочки прокси.

```
# dynamic_chain
#
# Динамическая цепочка - каждое соединение будет выполнено через цепочку прокси
# все прокси соединяются в цепочку в порядке появления в списке
# хотя бы один прокси должен быть онлайн, чтобы участвовать в цепочке
#
# strict_chain
#
# Строгая цепочка - каждое соединение будет выполнено через цепочку прокси
# все прокси соединяются в цепочку в порядке появления в списке
# все прокси должны быть онлайн, чтобы участвовать в цепочке
# иначе приложению возвращается EINTR
#
random_chain
# Случайная цепочка - каждое соединение будет выполнено через случайный прокси
# (или цепочку прокси, см. chain_len) из списка.
# этот параметр хорошо подходит для проверки вашей IDS :)
# имеет смысл только при random_chain
chain_len = 3
```

Здесь я раскомментировал `chain_len` и задал ему значение `3`, что означает, что `proxychains` теперь будет использовать три прокси из моего списка в файле `/etc/proxychains.conf`, выбирая их случайным образом и переходя к следующему, если прокси не работает. Обратите внимание, что хотя этот метод, безусловно, усиливает вашу анонимность, он также увеличивает задержку ваших онлайн-действий.

Теперь, когда вы знаете, как использовать `proxychains`, вы можете заниматься хакингом с относительной анонимностью. Я говорю "относительной", потому что нет безотказного способа оставаться анонимным, когда NSA и FSB следят за нашими онлайн-действиями, но с помощью `proxychains` мы можем значительно усложнить обнаружение.

Вопросы безопасности

В качестве последнего замечания о безопасности прокси обязательно выбирайте прокси с умом: `proxychains` настолько хорош, насколько хороши используемые вами прокси. Если вы намерены оставаться анонимным, не используйте бесплатный прокси, как упоминалось ранее. Хакеры используют платные прокси, которым можно доверять. На самом деле бесплатные прокси, вероятно, продают ваш IP-адрес и историю браузера. Как однажды сказал Брюс Шнайер, знаменитый криптограф и эксперт по безопасности: "Если что-то бесплатно, вы не клиент; вы продукт". Другими словами, любой бесплатный продукт, скорее всего, собирает ваши данные и продает их. Зачем еще предлагать прокси бесплатно?

Хотя IP-адрес вашего трафика, выходящего из прокси, будет анонимным, у служб наблюдения есть и другие способы идентифицировать вас. Например, владелец прокси будет знать вашу личность и, если на него достаточно надавят шпионские или правоохранительные органы с юрисдикцией, может выдать вашу личность, чтобы защитить свой бизнес. Важно понимать ограничения прокси как источника анонимности.

Виртуальные частные сети

Использование virtual private network (VPN) может быть эффективным способом сохранить ваш веб-трафик относительно анонимным и защищенным. VPN используется для подключения к промежуточному интернет-устройству, например маршрутизатору, который отправляет ваш трафик к его конечному назначению, помечая его IP-адресом маршрутизатора.

Использование VPN, безусловно, может усилить вашу безопасность и приватность, но это не гарантия анонимности. Интернет-устройство, к которому вы подключаетесь, должно записывать или журналировать ваш IP-адрес, чтобы правильно отправлять данные обратно вам, поэтому любой, кто сможет получить доступ к этим записям, сможет раскрыть информацию о вас.

Красота VPN в том, что они просты и легки в работе. Вы можете открыть учетную запись у VPN-провайдера, а затем без усилий подключаться к VPN каждый раз, когда входите в свой компьютер. Вы будете использовать браузер как обычно для перемещения по вебу, но любому наблюдающему будет казаться, что ваш трафик исходит от IP-адреса и местоположения интернет-устройства VPN, а не от вашего собственного. Кроме того, весь трафик между вами и устройством VPN зашифрован, так что даже ваш internet service provider не сможет видеть ваш трафик.

Среди прочего, VPN может быть эффективной для обхода контролируемых государством цензоров контента и информации. Например, если ваше национальное правительство ограничивает ваш доступ к веб-сайтам с определенным политическим сообщением, вы, вероятно, можете использовать VPN, расположенную за пределами вашей страны, чтобы получить доступ к этому контенту. Некоторые медиакорпорации, такие как Netflix, Hulu и HBO, ограничивают доступ к своему контенту IP-адресами, происходящими из их собственной страны. Использование VPN, расположенной в стране, которую эти сервисы разрешают, часто может помочь обойти эти ограничения доступа.

Некоторые из лучших и самых популярных коммерческих VPN-сервисов, по данным CNET, следующие:

• IPVanish
• NordVPN
• ExpressVPN
• CyberGhost
• Golden Frog VPN
• Hide My Ass (HMA)
• Private Internet Access
• PureVPN
• TorGuard
• Buffered VPN

Большинство этих VPN-сервисов берут 50-100 долларов в год, и многие предлагают бесплатный 30-дневный пробный период. Чтобы узнать больше о настройке VPN, выберите один из списка и посетите веб-сайт. Там вы должны найти инструкции по загрузке, установке и использованию, которым довольно легко следовать.

Сильная сторона VPN в том, что весь ваш трафик шифруется, когда покидает ваш компьютер, тем самым защищая вас от подглядывания, а ваш IP-адрес маскируется IP-адресом VPN, когда вы посещаете сайт. Как и в случае с прокси-сервером, владелец VPN имеет ваш исходный IP-адрес (иначе он не смог бы отправлять ваш трафик обратно вам). Если на него надавят шпионские агентства или правоохранительные органы, он может выдать вашу личность. Один способ предотвратить это - использовать только VPN, которые обещают не хранить и не журналировать такую информацию (и надеяться, что они говорят правду). Таким образом, если кто-то потребует, чтобы поставщик VPN-сервиса передал данные о своих пользователях, данных не будет.

Зашифрованная электронная почта

Бесплатные коммерческие почтовые сервисы, такие как Gmail, Yahoo! и Outlook Web Mail (ранее Hotmail), бесплатны по одной причине: они являются средствами отслеживания ваших интересов и показа рекламы. Как уже упоминалось, если сервис бесплатен, вы продукт, а не клиент. Кроме того, серверы поставщика электронной почты (например, Google) имеют доступ к незашифрованному содержимому вашей электронной почты, даже если вы используете HTTPS.

Один способ предотвратить подслушивание вашей электронной почты - использовать зашифрованную электронную почту. ProtonMail, показанный на рисунке 13-5, шифрует вашу электронную почту end to end, или от браузера до браузера. Это означает, что ваша электронная почта зашифрована на серверах ProtonMail; даже администраторы ProtonMail не могут читать вашу почту.

ProtonMail была основана группой молодых ученых на объекте суперколлайдера CERN в Швейцарии. Швейцарцы имеют долгую и богатую историю защиты секретов (помните те швейцарские банковские счета, о которых вы так много слышали?), а серверы ProtonMail расположены в Европейском союзе, где действуют гораздо более строгие законы о передаче персональных данных, чем в Соединенных Штатах. ProtonMail не взимает плату за базовую учетную запись, но предлагает премиум-учетные записи за символическую плату. Важно отметить, что при обмене электронной почтой с пользователями не ProtonMail существует вероятность, что часть или вся переписка не будет зашифрована. Полные сведения см. в базе знаний поддержки ProtonMail.

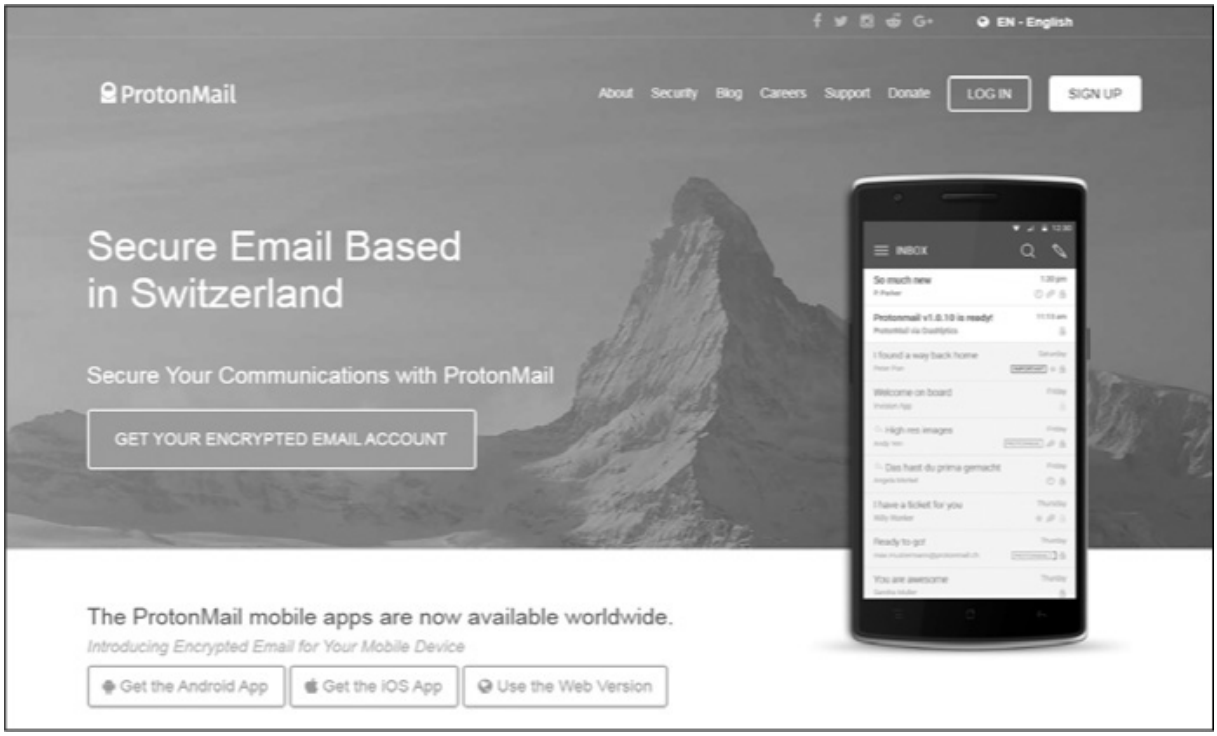

Рисунок 13-5. Экран входа ProtonMail

Итоги

За нами постоянно наблюдают коммерческие фирмы и национальные разведывательные агентства. Чтобы сохранить ваши данные и веб-путешествия защищенными, вам нужно внедрить по крайней мере одну из мер безопасности, рассмотренных в этой главе. Используя их в сочетании, вы можете минимизировать свой след в вебе и сохранить свои данные гораздо более защищенными.

Упражнения

Прежде чем перейти к главе 14, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Запустите `traceroute` до вашего любимого веб-сайта. Сколько переходов появляется между вами и вашим любимым сайтом?
2. Загрузите и установите Tor Browser. Теперь анонимно перемещайтесь по вебу так же, как вы делали бы это с любым другим браузером, и посмотрите, заметите ли разницу в скорости.
3. Попробуйте использовать `proxychains` с браузером Firefox, чтобы перейти на ваш любимый веб-сайт.
4. Изучите коммерческие VPN-сервисы некоторых поставщиков, перечисленных в этой главе. Выберите один и протестируйте бесплатный пробный период.
5. Откройте бесплатную учетную запись ProtonMail и отправьте защищенное приветствие на `occupytheweb@protonmail.com`.

Глава 14. Понимание и исследование беспроводных сетей

Способность сканировать другие сетевые устройства и подключаться к ним из вашей системы критически важна для того, чтобы стать успешным хакером, а поскольку беспроводные технологии, такие как Wi-Fi (IEEE 802.11) и Bluetooth, становятся стандартом, поиск и управление Wi-Fi- и Bluetooth-соединениями являются ключевыми. Если кто-то может взломать беспроводное соединение, он может получить вход на устройство и доступ к конфиденциальной информации. Первый шаг, конечно, - научиться находить эти устройства.

В главе 3 мы рассмотрели несколько базовых сетевых команд в Linux, включая некоторые основы беспроводных сетей, с обещанием вернуться к беспроводным сетям подробнее в главе 14. Как и обещано, здесь мы изучим две самые распространенные беспроводные технологии в Linux: Wi-Fi и Bluetooth.

Wi-Fi-сети

Начнем с Wi-Fi. В этом разделе я покажу, как находить, исследовать и подключаться к точкам доступа Wi-Fi. Прежде чем это сделать, потратим немного времени на некоторые базовые термины и технологии Wi-Fi, чтобы вам было легче понимать вывод множества запросов, которые мы будем выполнять в этой главе:

AP (точка доступа) - устройство, к которому беспроводные пользователи подключаются для доступа в интернет.

ESSID (расширенный идентификатор набора служб) - то же самое, что SSID, который мы обсуждали в главе 3, но он может использоваться для нескольких AP в беспроводной LAN.

BSSID (базовый идентификатор набора служб) - уникальный идентификатор каждой AP; он совпадает с MAC-адресом устройства.

SSID (идентификатор набора служб) - имя сети.

Каналы - Wi-Fi может работать на любом из 14 каналов (1-14). В Соединенных Штатах Wi-Fi ограничен каналами 1-11.

Мощность - чем ближе вы к Wi-Fi AP, тем больше мощность и тем легче взломать соединение.

Безопасность - протокол безопасности, используемый на Wi-Fi AP, с которой считывается информация. Существует три основных протокола безопасности для Wi-Fi. Изначальный протокол конфиденциальности, эквивалентной проводной сети (WEP), был сильно ошибочным и легко взламывался. Его замена, защищенный доступ Wi-Fi (WPA), была немного безопаснее. Наконец, WPA2-PSK, который намного безопаснее и использует предварительно разделенный ключ (PSK), общий для всех пользователей, теперь используется почти всеми Wi-Fi AP (кроме корпоративного Wi-Fi).

Каналы - Wi-Fi может работать в одном из трех режимов: управляемом, ведущем или режиме мониторинга. Что означают эти режимы, вы узнаете в следующем разделе.

Дальность беспроводной связи - в Соединенных Штатах Wi-Fi AP по закону должна передавать сигнал с верхним пределом 0,5 ватта. При такой мощности нормальная дальность составляет около 300 футов (100 метров). Антенны с высоким усилением могут увеличить эту дальность до 20 миль.

Каналы - Wi-Fi рассчитан на работу на 2,4 ГГц и 5 ГГц. Современные Wi-Fi AP и беспроводные сетевые карты часто используют оба диапазона.

Базовые беспроводные команды

В главе 3 вы познакомились с базовой сетевой командой Linux `ifconfig`, которая перечисляет каждый активированный сетевой интерфейс в вашей системе вместе с некоторой базовой статистикой, включая (самое важное) IP-адрес каждого интерфейса. Давайте еще раз посмотрим на результаты запуска `ifconfig` и на этот раз сосредоточимся на беспроводных соединениях.

```
kali >ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:ba:82:0f
inet addr:192.168.181.131 Bcast:192.168.181.255 Mask:255.255.255.0
--snip--
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
--snip--
(1) wlan0 Link encap:Ethernet HWaddr 00:c0:ca:3f:ee:02
```

Интерфейс Wi-Fi здесь показан как `wlan0` (1). В Kali Linux интерфейсы Wi-Fi обычно обозначаются как `wlanX`, где `X` представляет номер этого интерфейса. Другими словами, первый Wi-Fi-адаптер в вашей системе будет помечен `wlan0`, второй - `wlan1` и так далее.

Если вы хотите увидеть только свои Wi-Fi-интерфейсы и их статистику, в Linux есть специальная команда, похожая на `ifconfig`, но предназначенная для беспроводных интерфейсов. Эта команда - `iwconfig`. Когда вы вводите ее, отображаются только ваши беспроводные интерфейсы и их ключевые данные:

```
kali >iwconfig
lo no wireless extensions
wlan0 IEEE 802.11bg ESSID:off/any
Mode:Managed Access Point:Not-Associated Tx-Power=20 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:off
eth0 no wireless extensions
```

Здесь мы видим только беспроводные интерфейсы, также известные как сетевые карты, и ключевые данные о них, включая используемый беспроводной стандарт, выключен ли ESSID и режим. У режима есть три настройки: `managed`, то есть интерфейс готов присоединиться к AP или уже присоединился; `master`, то есть он готов действовать как AP или уже является AP; и `monitor`, режим мониторинга, который мы обсудим немного позже в этой главе. Мы также можем увидеть, ассоциировался ли с ним какой-либо клиент, и какова его мощность передачи, среди прочего. По

этому примеру можно понять, что `wlan0` находится в режиме, необходимом для подключения к Wi-Fi-сети, но пока не подключен ни к одной. Мы еще вернемся к этой команде после того, как беспроводной интерфейс будет подключен к Wi-Fi-сети.

Если вы не уверены, к какой Wi-Fi AP хотите подключиться, можно увидеть все беспроводные точки доступа, до которых может дотянуться ваша сетевая карта, используя команду `iwlist`. Синтаксис `iwlist` выглядит так:

```
iwlist interface action
```

С `iwlist` можно выполнять несколько действий. Для наших целей мы используем действие `scan`, чтобы увидеть все Wi-Fi AP в вашей области. (Обратите внимание, что со стандартной антенной ваша дальность будет 300-500 футов, но ее можно увеличить недорогой антенной с высоким усилением.)

```
kali >iwlist wlan0 scan
wlan0  Scan completed:
Cell 01 - Address:88:AD:43:75:B3:82
Channel:1
Frequency:2.412GHz (Channel 1)
Quality=70/70 Signal level =-38 dBm
Encryption key:off
ESSID:"Hackers-Arise"
--snip--
```

Вывод этой команды должен включать все Wi-Fi AP в зоне действия вашего беспроводного интерфейса, вместе с ключевыми данными о каждой AP, такими как MAC-адрес AP, канал и частота, на которых она работает, качество, уровень сигнала, включен ли ключ шифрования, и ESSID.

Вам понадобится MAC-адрес целевой AP (BSSID), MAC-адрес клиента (другой беспроводной сетевой карты) и канал, на котором работает AP, чтобы выполнить любой вид хакинга, так что это ценная информация.

Еще одна команда, очень полезная при управлении Wi-Fi-соединениями, - `nmcli` (или интерфейс командной строки менеджера сети). Демон Linux, который предоставляет высокоуровневый интерфейс для сетевых интерфейсов (включая беспроводные), известен как менеджер сети. Обычно пользователи Linux знакомы с этим демоном по его графическому интерфейсу пользователя (GUI), но его также можно использовать из командной строки.

Команду `nmcli` можно использовать, чтобы просматривать Wi-Fi AP рядом с вами и их ключевые данные, как мы делали с `iwlist`, но эта команда дает нам немного больше информации. Мы используем ее в формате `nmcli dev networktype`, где `dev` - сокращение от "устройства", а тип (в данном случае) - `wifi`, вот так:

```
kali >nmcli dev wifi
* SSID    MODE   CHAN  RATE     SIGNAL BARS  SECURITY
Hackers-Arise Infra 1 54 Mbits/s 100  WPA1 WPA2
Xfinitywifi Infra 1 54 Mbits/s 75   WPA2
TPTV1    Infra 11 54 Mbits/s 44     WPA1 WPA2
--snip--
```

Помимо отображения Wi-Fi AP в зоне действия и ключевых данных о них, включая SSID, режим, канал, скорость передачи, силу сигнала и протоколы безопасности, включенные на устройстве, `nmcli` можно использовать для подключения к AP. Синтаксис подключения к AP выглядит так:

```
nmcli dev wifi connect AP-SSID password APpassword
```

Итак, исходя из результатов нашей первой команды, мы знаем, что есть AP с SSID `Hackers-Arise`. Мы также знаем, что у нее безопасность WPA1 WPA2 (это означает, что AP способна использовать как более старый WPA1, так и более новый WPA2), а значит, для подключения к сети нам нужно предоставить пароль. К счастью, поскольку это наша AP, мы знаем, что пароль `12345678`, так что можем ввести следующее:

```
kali >nmcli dev wifi connect Hackers-Arise password 12345678
Device 'wlan0' successfully activated with '394a5bf4-8af4-36f8-49beda6cb530'.
```

Попробуйте это в сети, которую знаете, а затем, когда вы успешно подключитесь к этой беспроводной AP, снова выполните `iwconfig`, чтобы увидеть, что изменилось. Вот мой вывод после подключения к `Hackers-Arise`:

```
kali >iwconfig
lo no wireless extensions
wlan0 IEEE 802.11bg ESSID:"Hackers-Arise"
Mode:Managed Frequency:2.452GHz Access Point:00:25:9C:97:4F:48
Bit Rate=12 Mbs Tx-Power=20 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:off
Link Quality=64/70 Signal level=-46 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive reties:0 Invalid misc:13 Missed beacon:0
eth0 no wireless extensions
```

Обратите внимание, что теперь `iwconfig` указала, что ESSID - `"Hackers-Arise"` и что AP работает на частоте 2,452 ГГц. В Wi-Fi-сети несколько AP могут быть частью одной и той же сети, поэтому может быть много AP, составляющих сеть `Hackers-Arise`. MAC-адрес `00:25:9C:97:4F:48`, как можно ожидать, является MAC той AP, к которой я подключен. Какой тип безопасности использует Wi-Fi-сеть, работает ли она на 2,4 ГГц или 5 ГГц, каков ее ESSID и каков MAC-адрес AP - все это критически важные сведения, необходимые для Wi-Fi-хакинга. Теперь, когда вы знаете базовые команды, перейдем к некоторому хакингу.

Wi-Fi-разведка с aircrack-ng

Один из самых популярных эксплойтов, которые пробуют новые хакеры, - взлом точек доступа Wi-Fi. Как упоминалось, прежде чем даже думать об атаке Wi-Fi AP, вам нужны MAC-адрес целевой AP (BSSID), MAC-адрес клиента и канал, на котором работает AP.

Мы можем получить всю эту информацию и больше, используя инструменты набора `aircrack-ng`. Я уже несколько раз упоминал этот набор инструментов Wi-Fi-хакинга, и теперь пора действительно его использовать. Этот набор инструментов включен в каждую версию Kali, так что вам не нужно ничего загружать или устанавливать.

Чтобы эффективно использовать эти инструменты, сначала нужно перевести вашу беспроводную сетевую карту в режим мониторинга, чтобы карта могла видеть весь трафик, проходящий мимо нее. Обычно сетевая карта захватывает только трафик, предназначенный специально для этой карты. Режим мониторинга похож на неразборчивый режим в проводных сетевых картах.

Чтобы перевести беспроводную сетевую карту в режим мониторинга, используйте команду `airmon-ng` из набора `aircrack-ng`. Синтаксис этой команды прост:

```
airmon-ng start|stop|restart interface
```

Итак, если вы хотите перевести свою беспроводную сетевую карту (обозначенную `wlan0`) в режим мониторинга, вы ввели бы следующее:

```
kali >airmon-ng start wlan0
Найдены три процесса, которые могут вызвать проблемы
Если airodump-ng, aireplay-ng или airtun-ng перестанут работать через
короткое время, можно выполнить 'airmon-ng check kill'
--snip--
PHY    INTERFACE     DRIVER   Chipset
phy0   wlan0     rtl8187 Realtek Semiconductor Corp. RTL8187
(mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
--snip--
```

Команды `stop` и `restart` соответственно останавливают режим мониторинга и перезапускают его, если у вас возникнут проблемы.

Когда ваша беспроводная карта находится в режиме мониторинга, вы можете получать доступ ко всему беспроводному трафику, проходящему мимо вас в пределах дальности вашего беспроводного сетевого адаптера и антенны (стандартно около 300-500 футов). Обратите внимание, что `airmon-ng` переименует ваш беспроводной интерфейс: мой был переименован в `wlan0mon`, хотя у вас он может называться иначе. Обязательно отметьте новое назначенное имя вашего беспроводного интерфейса, потому что эта информация понадобится на следующем шаге.

Теперь мы используем еще один инструмент из набора `aircrack-ng`, чтобы найти ключевые данные из беспроводного трафика. Команда `airodump-ng` захватывает и отображает ключевые данные от вещающих AP и любых клиентов, подключенных к этим AP или находящихся поблизости. Синтаксис здесь прямой: просто введите `airdump-ng`, а затем имя интерфейса, которое вы получили при только что выполненном `airmon-ng`. Когда вы выполните эту команду, ваша беспроводная карта получит важную информацию (перечисленную далее) из всего беспроводного трафика ближайших AP:

• BSSID - MAC-адрес AP или клиента;
• PWR - сила сигнала;
• ENC - шифрование, используемое для защиты передачи;
• #Data - скорость передачи данных;
• CH - канал, на котором работает AP;
• ESSID - имя AP.

```
kali >airodump-ng wlan0mon
CH 9][ Elapsed: 28 s ][ 2018-02-08 10:27
BSSID      PWR Beacons #Data #/s CH MB ENC CIPHER AUTH ESSID
01:01:AA:BB:CC:22 -1 4 26 0 10 54e WPA2 CCMP PSK Hackers-Arise
--snip--
BSSID      Station    PWR  Rate Lost Frames Probe
(not associated) 01:01:AA:BB:CC:22
01:02:CC:DD:03:CF A0:A3:E2:44:7C:E5
```

Обратите внимание, что `airodump-ng` делит экран вывода на верхнюю и нижнюю части. Верхняя часть содержит информацию о вещающих AP, включая BSSID, мощность AP, сколько beacon-кадров было обнаружено, скорость передачи данных, сколько пакетов прошло через беспроводную карту, канал (1-14), теоретический предел пропускной способности, протокол шифрования, шифр, используемый для шифрования, тип аутентификации и ESSID (обычно называемый SSID). В клиентской части вывод сообщает нам, что один клиент не ассоциирован, то есть он был обнаружен, но не подключен ни к одной AP, а другой ассоциирован со станцией, то есть подключен к AP по этому адресу.

Теперь у вас есть вся информация, необходимая для взлома AP! Хотя это выходит за рамки этой книги, для взлома беспроводной AP нужны MAC-адрес клиента, MAC-адрес AP, канал, на котором работает цель, и список паролей.

Итак, чтобы взломать Wi-Fi-пароль, вы открыли бы три терминала. В первом терминале вы ввели бы команды, похожие на следующую, подставив MAC-адреса клиента и AP, а также канал:

```
airodump-ng -c 10 --bssid 01:01:AA:BB:CC:22 -w Hackers-ArisePSK wlan0mon
```

Эта команда захватывает все пакеты, проходящие через AP на канале 10, используя параметр `-c`.

В другом терминале можно использовать команду `aireplay-ng`, чтобы выбить (деаутентифицировать) любого, кто подключен к AP, и заставить его заново аутентифицироваться на AP, как показано далее. Когда он повторно аутентифицируется, вы можете захватить хеш его пароля, который передается во время четырехстороннего рукопожатия WPA2-PSK. Хеш пароля появится в правом верхнем углу терминала `airodump-ng`.

```
aireplay-ng --deauth 100 -a 01:01:AA:BB:CC:22 -c A0:A3:E2:44:7C:E5 wlan0mon
```

Наконец, в последнем терминале можно использовать список паролей (`wordlist.dic`), чтобы найти пароль в захваченном хеше (`Hackers-ArisePSK.cap`), как показано здесь:

```
aircrack-ng -w wordlist.dic -b 01:01:AA:BB:CC:22 Hackers-ArisePSK.cap
```

Обнаружение Bluetooth и подключение к нему

В наши дни почти каждый гаджет, мобильное устройство и система имеют встроенный Bluetooth, включая наши компьютеры, смартфоны, iPod, планшеты, колонки, игровые контроллеры, клавиатуры и множество других устройств. Способность взламывать Bluetooth может привести к компрометации любой информации на устройстве, управлению устройством и возможности отправлять нежелательную информацию на устройство и с него, среди прочего.

Чтобы эксплуатировать технологию, нужно понимать, как она работает. Глубокое понимание Bluetooth выходит за рамки этой книги, но я дам вам базовые знания, которые помогут сканировать Bluetooth-устройства и подключаться к ним при подготовке к их взлому.

Как работает Bluetooth

Bluetooth - универсальный протокол для маломощной связи ближнего поля, работающий на 2,4-2,485 ГГц с использованием расширенного спектра и скачкообразной перестройки частоты на 1 600 переходов в секунду (такая перестройка частоты является мерой безопасности). Он был разработан в 1994 году Ericsson Corp. из Швеции и назван в честь датского короля X века Harald Bluetooth (обратите внимание, что Швеция и Дания были одной страной в X веке).

Спецификация Bluetooth имеет минимальную дальность 10 метров, но нет верхнего ограничения дальности, которую производители могут реализовать в своих устройствах. У многих устройств дальность достигает 100 метров. С помощью специальных антенн эту дальность можно увеличить еще больше.

Подключение двух Bluetooth-устройств называется сопряжением. Практически любые два Bluetooth-устройства могут подключиться друг к другу, но они могут выполнить сопряжение только если находятся в режиме обнаружения. Bluetooth-устройство в режиме обнаружения передает следующую информацию:

• адрес;
• имя;
• класс;
• службы.

Когда два устройства выполняют сопряжение, они обмениваются секретом, или ключом связи. Каждое хранит этот ключ связи, чтобы при будущих сопряжениях идентифицировать другое устройство.

У каждого устройства есть уникальный 48-битный идентификатор (адрес, похожий на MAC) и обычно имя, назначенное производителем. Это будут полезные данные, когда мы захотим идентифицировать устройство и получить к нему доступ.

Сканирование и разведка Bluetooth

В Linux есть реализация стека протоколов Bluetooth под названием BlueZ, которую мы будем использовать для сканирования Bluetooth-сигналов. Большинство дистрибутивов Linux, включая Kali Linux, устанавливают ее по умолчанию. Если в вашем случае она не установлена, обычно ее можно найти в репозитории с помощью следующей команды:

```
kali >apt-get install bluez
```

В BlueZ есть ряд простых инструментов, которые можно использовать для управления Bluetooth-устройствами и их сканирования, включая следующие:

`hciconfig` - этот инструмент работает очень похоже на `ifconfig` в Linux, но для Bluetooth-устройств. Как видно в листинге 14-1, я использовал его, чтобы поднять Bluetooth-интерфейс и запросить характеристики устройства.

`hcitool` - этот инструмент запросов может предоставить нам имя устройства, идентификатор устройства, класс устройства и информацию о тактовом смещении, что позволяет устройствам работать синхронно.

`hcidump` - этот инструмент позволяет sniff Bluetooth-коммуникацию, то есть мы можем захватывать данные, отправленные по Bluetooth-сигналу.

Первый шаг сканирования и разведки с Bluetooth - проверить, распознан ли Bluetooth-адаптер в системе, которую мы используем, и включен ли он, чтобы мы могли использовать его для сканирования других устройств. Это можно сделать встроенным инструментом BlueZ `hciconfig`, как показано в листинге 14-1.

```
kali >hciconfig
hci0: Type: BR/EDR Bus: USB
BD Address: 10:AE:60:58:F1:37 ACL MTU: 310:10 SCO MTU: 64:8
UP RUNNING  PSCAN INQUIRY
RX bytes:131433 acl:45 sco:0 events:10519 errors:0
TX bytes:42881 acl:45 sco:0 commands:5081 errors:0
```

Листинг 14-1. Сканирование Bluetooth-устройства

Как видите, мой Bluetooth-адаптер распознан с MAC-адресом `10:AE:60:58:F1:37`. Этот адаптер получил имя `hci0`. Следующий шаг - проверить, что соединение включено; это также можно сделать с помощью `hciconfig`, передав имя и команду `up`:

```
kali >hciconfig hci0 up
```

Если команда выполнится успешно, мы не увидим вывода, только новое приглашение.

Хорошо, `hci0` поднят и готов! Давайте заставим его работать.

Сканирование Bluetooth-устройств с помощью hcitool

Теперь, когда мы знаем, что наш адаптер поднят, можно использовать другой инструмент из набора BlueZ под названием `hcitool`, который используется для сканирования других Bluetooth-устройств в зоне действия.

Сначала используем функцию сканирования этого инструмента, чтобы найти Bluetooth-устройства, которые отправляют свои маяки обнаружения, то есть находятся в режиме обнаружения, простой командой `scan`, показанной в листинге 14-2.

```
kali >hcitool scan
Scanning...
72:6E:46:65:72:66 ANDROID BT
22:C5:96:08:5D:32 SCH-I535
```

Листинг 14-2. Сканирование Bluetooth-устройств в режиме обнаружения

Как видите, в моей системе `hcitool` нашел два устройства, `ANDROID BT` и `SCH-I535`. У вас вывод, скорее всего, будет другим в зависимости от того, какие устройства есть рядом. Для тестовых целей попробуйте перевести свой телефон или другое Bluetooth-устройство в режим обнаружения и посмотрите, попадет ли оно в сканирование.

Теперь соберем больше информации об обнаруженных устройствах с помощью функции inquiry `inq`:

```
kali >hcitool inq
Inquiring...
24:C6:96:08:5D:33 clock offset:0x4e8b class:0x5a020c
76:6F:46:65:72:67 clock offset:0x21c0 class:0x5a020c
```

Это дает нам MAC-адреса устройств, тактовое смещение и класс устройств. Класс указывает, какой тип Bluetooth-устройства вы нашли, и можно посмотреть код и узнать тип устройства, перейдя на сайт Bluetooth SIG по адресу `https://www.bluetooth.org/en-us/specification/assigned-numbers/service-discovery/`.

Инструмент `hcitool` - мощный интерфейс командной строки к стеку Bluetooth, который может делать очень многое. Листинг 14-3 показывает страницу справки с некоторыми командами, которые можно использовать. Посмотрите страницу справки самостоятельно, чтобы увидеть полный список.

```
kali >hcitool --help
hcitool - HCI Tool ver 4.99
Использование:
hcitool [параметры] <команда> [параметры команды]
Параметры:
--help Показать справку
-i dev HCI-устройство
Команды
dev Показать локальные устройства
inq Опросить удаленные устройства
scan Сканировать удаленные устройства
name  Получить имя удаленных устройств
--snip--
```

Листинг 14-3. Некоторые команды `hcitool`

Многие инструменты Bluetooth-хакинга, которые вы встретите, просто используют эти команды в сценарии, и вы легко можете создать собственный инструмент, используя эти команды в своем bash- или Python-сценарии; мы рассмотрим сценарии в главе 17.

Сканирование служб с помощью sdptool

Протокол обнаружения служб (SDP) - это Bluetooth-протокол для поиска Bluetooth-служб (Bluetooth - это набор служб), и, к счастью, BlueZ предоставляет инструмент `sdptool` для просмотра устройства и служб, которые оно предоставляет. Также важно отметить, что устройство не обязано находиться в режиме обнаружения, чтобы его можно было просканировать. Синтаксис выглядит так:

```
sdptool browse MACaddress
```

Листинг 14-4 показывает, как я использую `sdptool` для поиска служб на одном из устройств, обнаруженных ранее в листинге 14-2.

```
kali >sdptool browse 76:6E:46:63:72:66
Browsing 76:6E:46:63:72:66...
Service RecHandle: 0x10002
Service Class ID List:
""(0x1800)
Protocol Descriptor List:
"L2CAP" (0x0100)
PSM: 31
"ATT" (0x0007)
uint16: 0x1
uint16: 0x5
--snip--
```

Листинг 14-4. Сканирование с помощью `sdptool`

Здесь мы видим, что инструмент `sdptool` смог извлечь информацию обо всех службах, которые это устройство способно использовать. В частности, мы видим, что это устройство поддерживает протокол ATT, то есть протокол атрибутов с низким энергопотреблением. Это может дать нам больше подсказок о том, что это за устройство, и, возможно, о потенциальных путях дальнейшего взаимодействия с ним.

Проверка достижимости устройств с помощью l2ping

После того как мы собрали MAC-адреса всех ближайших устройств, можно отправить ping этим устройствам, независимо от того, находятся они в режиме обнаружения или нет, чтобы увидеть, находятся ли они в пределах досягаемости. Это позволяет нам узнать, активны ли они и находятся ли в зоне действия. Чтобы отправить ping, мы используем команду `l2ping` со следующим синтаксисом:

```
l2ping MACaddress
```

Листинг 14-5 показывает, как я выполняю ping Android-устройства, обнаруженного в листинге 14-2.

```
kali >l2ping 76:6E:46:63:72:66 -c 4
Ping: 76:6E:46:63:72:66 from 10:AE:60:58:F1:37 (data size 44)...
44 bytes 76:6E:46:63:72:66 id 0 time 37.57ms
44 bytes 76:6E:46:63:72:66 id 1 time 27.23ms
44 bytes 76:6E:46:63:72:66 id 2 time 27.59ms
--snip--
```

Листинг 14-5. Ping Bluetooth-устройства

Этот вывод указывает, что устройство с MAC-адресом `76:6E:46:63:72:66` находится в зоне действия и достижимо. Это полезное знание, потому что мы должны знать, достижимо ли устройство, прежде чем даже задумываться о его взломе.

Итоги

Беспроводные устройства представляют будущее подключений и хакинга. Linux разработал специализированные команды для сканирования Wi-Fi AP и подключения к ним как первый шаг к взлому этих систем. Набор инструментов беспроводного хакинга `aircrack-ng` включает как `airmon-ng`, так и `airodump-ng`, которые позволяют нам сканировать и собирать ключевую информацию от беспроводных устройств в зоне действия. Набор BlueZ включает `hciconfig`, `hcitool` и другие инструменты, способные выполнять сканирование и сбор информации, необходимые для взлома Bluetooth-устройств в зоне действия. Он также включает множество других инструментов, которые стоит изучить.

Упражнения

Прежде чем перейти к главе 15, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Проверьте свои сетевые устройства с помощью `ifconfig`. Отметьте любые wireless extensions.
2. Запустите `iwconfig` и отметьте любые беспроводные сетевые адаптеры.
3. Проверьте, какие Wi-Fi AP находятся в зоне действия, с помощью `iwlist`.
4. Проверьте, какие точки доступа Wi-Fi находятся в зоне действия, с помощью `nmcli`. Что вы считаете более полезным и интуитивным, `nmcli` или `iwlist`?
5. Подключитесь к своей Wi-Fi AP с помощью `nmcli`.
6. Поднимите свой Bluetooth-адаптер с помощью `hciconfig` и просканируйте ближайшие обнаруживаемые Bluetooth-устройства с помощью `hcitool`.
7. Проверьте, находятся ли эти Bluetooth-устройства на достижимом расстоянии, с помощью `l2ping`.

Глава 15. Управление ядром Linux и загружаемыми модулями ядра

Все операционные системы состоят как минимум из двух основных компонентов. Первый и самый важный из них - ядро. Ядро находится в центре операционной системы и управляет всем, что делает операционная система, включая управление памятью, управление CPU и даже управление тем, что пользователь видит на экране. Второй элемент операционной системы часто называют пользовательским пространством, и он включает почти все остальное.

Ядро спроектировано как защищенная или привилегированная область, к которой может получать доступ только root или другие привилегированные учетные записи. Для этого есть веская причина, поскольку доступ к ядру может предоставить почти неограниченный доступ к операционной системе. В результате большинство операционных систем предоставляет пользователям и службам доступ только к пользовательскому пространству, где пользователь может получить доступ почти ко всему, что ему нужно, не беря операционную систему под контроль.

Доступ к ядру позволяет пользователю изменить то, как операционная система работает, выглядит и ощущается. Он также позволяет обрушить операционную систему, сделав ее неработоспособной. Несмотря на этот риск, в некоторых случаях системный администратор должен очень осторожно получать доступ к ядру по эксплуатационным причинам и причинам безопасности.

В этой главе мы изучим, как изменить то, как работает ядро, и добавить в ядро новые модули. Вероятно, само собой понятно, что если хакер может изменить ядро цели, он может контролировать систему. Более того, атакующему может понадобиться изменить функции ядра для некоторых атак, таких как man-in-the-middle (MITM), где хакер помещает себя между клиентом и сервером и может подслушивать или изменять коммуникацию. Сначала мы подробнее рассмотрим структуру ядра и его модули.

Что такое модуль ядра?

Ядро - центральная нервная система вашей операционной системы, контролирующая все, что она делает, включая управление взаимодействиями между аппаратными компонентами и запуск необходимых служб. Ядро работает между пользовательскими приложениями, которые вы видите, и аппаратным обеспечением, на котором все работает, например CPU, памятью и жестким диском.

Linux - монолитное ядро, которое позволяет добавлять модули ядра. Поэтому модули можно добавлять в ядро и удалять из него. Ядро иногда нужно обновлять, что может включать установку новых драйверов устройств (например, видеокарт, Bluetooth-устройств или USB-устройств), драйверов файловых систем и даже системных расширений. Эти драйверы должны быть встроены в ядро, чтобы быть полностью функциональными. В некоторых системах, чтобы добавить драйвер, приходится пересобирать, компилировать и перезагружать все ядро, но Linux способен добавлять некоторые модули в ядро без прохождения всего этого процесса. Такие модули называются загружаемыми модулями ядра, или LKMs.

LKMs по необходимости имеют доступ к самым низким уровням ядра, что делает их невероятно уязвимой целью для хакеров. Определенный тип вредоносного ПО, известный как руткит, встраивается в ядро операционной системы, часто через эти LKMs. Если вредоносное ПО внедряется в ядро, хакер может получить полный контроль над операционной системой.

Если хакер может заставить администратора Linux загрузить новый модуль в ядро, хакер не только может получить контроль над целевой системой, но и, поскольку он действует на уровне ядра операционной системы, может контролировать, что целевая система сообщает о процессах, портах, службах, пространстве на жестком диске и почти обо всем, что вы можете представить.

Итак, если хакер может успешно соблазнить администратора Linux установить видеодрайвер или другой драйвер устройства, в который встроен руткит, хакер может получить полный контроль над системой и ядром. Именно так некоторые из самых коварных руткитов используют Linux и другие операционные системы.

Понимание LKMs абсолютно необходимо, чтобы быть эффективным администратором Linux и очень эффективным и скрытным хакером.

Давайте посмотрим, как ядром можно управлять во благо и во вред.

Проверка версии ядра

Первый шаг к пониманию ядра - проверить, какое ядро работает в вашей системе. Есть как минимум два способа сделать это. Сначала можно ввести следующее:

```
kali >uname -a
Linux Kali 4.6.0-kalil-amd64 #1 SMP Debian 4.6.4-lkalil (2016-07-21) x86_64
```

Ядро отвечает, сообщая нам, что дистрибутив, на котором работает наша ОС, - Linux Kali, сборка ядра - 4.6.4, а архитектура, для которой оно собрано, - x86_64. Оно также сообщает, что имеет возможности симметричной многопроцессорности (SMP) (то есть может работать на машинах с несколькими ядрами или процессорами) и было собрано на Debian 4.6.4 21 июля 2016 года. Ваш вывод может отличаться в зависимости от того, какое ядро использовалось в вашей сборке, и от CPU в вашей системе. Эта информация может потребоваться при установке или загрузке драйвера ядра, поэтому полезно понимать, как ее получить.

Еще один способ получить эту информацию, а также другую полезную информацию, - использовать команду `cat` для файла `/proc/version`, вот так:

```
kali >cat /proc/version
Linux version 4.6.0-kalil-amd64 (devel@kali.org) (gcc version 5.4.0 20160909
(Debian 5.4.0-6) ) #1 SMP Debian 4.6.4-lkalil (2016-07-21)
```

Здесь видно, что файл `/proc/version` вернул ту же информацию.

Настройка ядра с помощью sysctl

С правильными командами можно настраивать свое ядро, то есть менять выделение памяти, включать сетевые функции и даже укреплять ядро против внешних атак.

Современные ядра Linux используют команду `sysctl` для настройки параметров ядра. Все изменения, которые вы делаете с помощью `sysctl`, остаются в силе только до перезагрузки системы. Чтобы сделать любые изменения постоянными, нужно напрямую редактировать конфигурационный файл для `sysctl` по пути `/etc/sysctl.conf`.

Предупреждение: нужно быть осторожным при использовании `sysctl`, потому что без должных знаний и опыта можно легко сделать систему незагружаемой и непригодной к использованию. Убедитесь, что вы тщательно обдумали то, что делаете, прежде чем вносить постоянные изменения.

Давайте теперь посмотрим на содержимое `sysctl`. К этому моменту вы уже должны узнавать параметры, которые мы передаем команде, показанной здесь:

```
kali >sysctl -a | less
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
dev.cdrom.check_media = 0
dev.cdrom.debug = 0
--snip--
```

В выводе вы должны увидеть сотни строк параметров, которые администратор Linux может редактировать для оптимизации ядра. Здесь есть несколько строк, полезных вам как хакеру. В качестве примера того, как можно использовать `sysctl`, мы рассмотрим включение пересылки пакетов.

В атаке man-in-the-middle (MITM) хакер помещает себя между взаимодействующими хостами, чтобы перехватывать информацию. Трафик проходит через систему хакера, так что он может просматривать и, возможно, изменять коммуникацию. Один способ добиться такой маршрутизации - включить пересылку пакетов.

Если вы прокрутите вывод на несколько страниц вниз или отфильтруете по `ipv4` (`sysctl -a | less | grep ipv4`), вы должны увидеть следующее:

```
net.ipv4.ip_dynaddr = 0
net.ipv4.ip_early_demux = 0
net.ipv4.ip_forward = 0
net.ipv4.ip_forward_use_pmtu = 0
--snip--
```

Строка `net.ipv4.ip_forward = 0` - это параметр ядра, который позволяет ядру пересылать полученные пакеты. Другими словами, полученные пакеты оно отправляет обратно наружу. Значение по умолчанию - `0`, что означает, что пересылка пакетов отключена.

Чтобы включить пересылку IP, измените `0` на `1`, введя следующее:

```
kali >sysctl -w net.ipv4.ip_forward=1
```

Помните, что изменения `sysctl` вступают в силу во время выполнения, но теряются при перезагрузке системы. Чтобы сделать изменения `sysctl` постоянными, нужно редактировать конфигурационный файл `/etc/sysctl.conf`. Давайте изменим то, как ядро обрабатывает пересылку IP для MITM-атак, и сделаем это изменение постоянным.

Чтобы включить пересылку IP, откройте файл `/etc/sysctl.conf` в любом текстовом редакторе и раскомментируйте строку для `ip_forward`. Откройте `/etc/sysctl.conf` в любом текстовом редакторе и посмотрите:

```
#/etc/sysctl.conf - конфигурационный файл для настройки системных переменных
# Дополнительные системные переменные см. в /etc/sysctl.d/.
# Информацию см. в sysctl.conf (5).
#
#kernel.domainname = example.com
# Раскомментируйте следующую строку, чтобы остановить низкоуровневые сообщения в консоли.
#kernel.printk = 3 4 1 3
###################################################################
# Функции, ранее находившиеся в netbase
#
# Раскомментируйте следующие две строки, чтобы включить защиту от подмены (reverse-path
# Включите проверку адреса источника на всех интерфейсах,
# чтобы предотвратить некоторые атаки с подменой.
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
# Раскомментируйте следующую строку, чтобы включить SYN cookies TCP/IP
#
# Примечание: это также может повлиять на TCP-сеансы IPv6
#net.ipv4.tcp_syncookies=1
# См. http://lwn.net/Articles/277146/
# Раскомментируйте следующую строку, чтобы включить пересылку пакетов для IPv4
(1) #net.ipv4.ip_forward=1
```

Релевантная строка находится у (1); просто удалите здесь комментарий (#), чтобы включить пересылку IP. С точки зрения укрепления операционной системы можно использовать этот файл, чтобы отключить эхо-запросы ICMP, добавив строку `net.ipv4.icmp_echo_ignore_all=1`, чтобы хакерам было труднее, но не невозможно, найти вашу систему. После добавления строки вам нужно будет выполнить команду `sysctl -p`.

Управление модулями ядра

В Linux есть как минимум два способа управлять модулями ядра. Более старый способ - использовать группу команд, построенных вокруг набора `insmod`; `insmod` означает "вставить модуль" и предназначен для работы с модулями. Второй способ, с использованием команды `modprobe`, мы применим немного позже в этой главе. Здесь мы используем команду `lsmod` из набора `insmod`, чтобы перечислить установленные в ядре модули:

```
kali >lsmod
Module         Size  Used by
nfnetlink_queue   20480  0
nfnetlink_log    201480 0
nfnetlink      16384  2 nfnetlink_log, nfnetlink_queue
bluetooth      516096  0
rfkill       0    2 bluetooth
--snip--
```

Как видите, команда `lsmod` перечисляет все модули ядра, а также сведения об их размере и о том, какие другие модули могут их использовать. Например, модуль `nfnetlink` - протокол на основе

сообщений для связи между ядром и пользовательским пространством - имеет размер 16 384 байта и используется как модулем nfnetlink_log, так и модулем nf_netlink_queue.

Из набора insmod мы можем загружать или вставлять модуль с помощью insmod и удалять модуль с помощью rmmod, что означает "удалить модуль". Эти команды несовершенны и могут не учитывать зависимости модулей, поэтому их использование может оставить ядро нестабильным или непригодным к использованию. В результате современные дистрибутивы Linux теперь добавили команду modprobe, которая автоматически загружает зависимости и делает загрузку и удаление модулей ядра менее рискованными. Мы рассмотрим modprobe через минуту. Сначала посмотрим, как получить больше информации о наших модулях.

Получение дополнительной информации с помощью modinfo

Чтобы узнать больше о любом из модулей ядра, можно использовать команду modinfo. Синтаксис этой команды прямой: modinfo, за которым следует имя модуля, о котором вы хотите узнать. Например, если вы хотели бы получить базовую информацию о модуле ядра bluetooth, который видели при запуске команды lsmod ранее, вы могли бы ввести следующее:

```
kali >modinfo bluetooth
filename: /lib/modules/4.6.0-kali-amd64/kernel/net/bluetooth/bluetooth.ko
alias: net-pf-31
license: GPL
version: 2.21
description: Bluetooth Core ver 2.21
author: Marcel Holtman <marcel@holtmann.org>
srcversion: FCFDE98577FEA911A3DAFA9
depends: rfkill, crc16
intree: Y
vermagic: 4.6.0-kali1-amd64 SMP mod_unload modversions
parm:  disable_esco: Disable eSCO connection creation (bool)
parm:  disable_ertm: Disable enhanced retransmission mode (bool)
```

Как видите, команда modinfo раскрывает значительный объем информации об этом модуле ядра, необходимом для использования Bluetooth в вашей системе. Обратите внимание, что среди многого другого она перечисляет зависимости модуля: rfkill и crc16. Зависимости - это модули, которые должны быть установлены, чтобы модуль bluetooth работал правильно.

Обычно это полезная информация при устранении причин, по которым конкретное аппаратное устройство не работает. Помимо таких вещей, как зависимости, можно получить информацию о версии модуля и версии ядра, для которой модуль был разработан, а затем убедиться, что они соответствуют версии, которую вы запускаете.

Добавление и удаление модулей с помощью modprobe

Большинство новых дистрибутивов Linux, включая Kali Linux, включают команду modprobe для управления LKM. Чтобы добавить модуль в ядро, вы использовали бы команду modprobe с ключом -a (add), вот так:

```
kali >modprobe -a <module name>
```

Чтобы удалить модуль, используйте ключ `-r` с `modprobe`, а затем имя удаляемого модуля:

```
kali >modprobe -r <модуль для удаления>
```

Главное преимущество использования `modprobe` вместо `insmod` состоит в том, что `modprobe` понимает зависимости, параметры, процедуры установки и удаления и учитывает все это перед внесением изменений. Поэтому добавлять и удалять модули ядра с помощью `modprobe` проще и безопаснее.

Вставка и удаление модуля ядра

Давайте попробуем вставить и удалить тестовый модуль, чтобы помочь вам познакомиться с этим процессом. Представим, что вы только что установили новую видеокарту и вам нужно установить для нее драйверы. Помните, драйверы устройств обычно устанавливаются напрямую в ядро, чтобы дать им необходимый доступ для правильной работы. Это также делает драйверы плодородной почвой для вредоносных хакеров, которые могут установить руткит или другое прослушивающее устройство.

Предположим для демонстрационных целей (не выполняйте эти команды на самом деле), что мы хотим добавить новый видеодрайвер с именем `HackersAriseNewVideo`. Вы можете добавить его в свое ядро, введя следующее:

```
kali >modprobe -a HackersAriseNewVideo
```

Чтобы проверить, правильно ли загрузился новый модуль, можно выполнить команду `dmesg`, которая печатает буфер сообщений от ядра, а затем отфильтровать по `video` и посмотреть, есть ли предупреждения, указывающие на проблему:

```
kali >dmesg | grep video
```

Если есть какие-либо сообщения ядра со словом `video`, они будут отображены здесь. Если ничего не появляется, сообщений, содержащих это ключевое слово, нет.

Затем, чтобы удалить этот же модуль, можно ввести ту же команду, но с ключом `-r`:

```
kali >modprobe -r HackersAriseNewVideo
```

Помните, загружаемые модули ядра - удобство для пользователя/администратора Linux, но они также являются крупной слабостью безопасности, с которой профессиональные хакеры должны быть знакомы. Как я говорил раньше, LKMs могут быть идеальным транспортом, чтобы поместить ваш руткит в ядро и устроить хаос!

Итоги

Ядро критически важно для общей работы операционной системы, и поэтому оно является защищенной областью. Все, что непреднамеренно добавлено в ядро, может нарушить работу операционной системы и даже взять ее под контроль.

LKMs позволяют системному администратору добавлять модули напрямую в ядро без необходимости пересобирать все ядро каждый раз, когда он хочет добавить модуль.

Если хакер может убедить системного администратора добавить вредоносный LKM, хакер может получить полный контроль над системой, часто даже без того, чтобы системный администратор это осознал.

Упражнения

Прежде чем перейти к главе 16, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Проверьте версию своего ядра.
2. Перечислите модули в своем ядре.
3. Включите пересылку IP с помощью команды `sysctl`.
4. Отредактируйте файл `/etc/sysctl.conf`, чтобы включить пересылку IP. Теперь отключите пересылку IP.
5. Выберите один модуль ядра и узнайте о нем больше с помощью `modinfo`.

Глава 16. Автоматизация задач с планированием заданий

Как и у любого, кто использует Linux, у хакера часто есть задания, сценарии или другие задачи, которые он хочет запускать периодически. Например, вы можете захотеть запланировать автоматическое регулярное резервное копирование файлов вашей системы, или, возможно, захотите ротировать файлы журналов, как мы делали в главе 11. Хакер, с другой стороны, может также захотеть, чтобы его система запускала сценарий `MySQLscanner.sh` из главы 8 каждую ночь или пока он на работе или в школе.

Все это примеры планирования автоматических заданий. Планирование заданий позволяет запускать задачи без необходимости думать об этом, и можно запланировать задания на время, когда вы иначе не используете систему, чтобы у вас было много свободных ресурсов.

Администратор Linux, а если уж на то пошло, и хакер, может также захотеть настроить определенные сценарии или службы на автоматический запуск при загрузке системы. В главе 12 мы рассмотрели использование базы данных PostgreSQL совместно с хакерским/пентест-фреймворком Metasploit. Вместо того чтобы каждый раз вручную запускать базу данных PostgreSQL перед запуском Metasploit, можно заставить PostgreSQL или любую службу либо сценарий запускаться автоматически при загрузке системы.

В этой главе вы узнаете больше о том, как использовать демон `cron` и `crontab`, чтобы настроить автоматический запуск сценариев, даже когда система оставлена без присмотра. Вы также узнаете, как настраивать сценарии запуска, которые автоматически запускаются всякий раз, когда система загружается, что предоставит вам необходимые службы, которые понадобятся в течение вашего напряженного дня хакинга.

Планирование события или задания для автоматического запуска

Демон `cron` и таблица cron (`crontab`) - самые полезные инструменты для планирования регулярных задач. Первый, `crond`, - демон, который работает в фоне. Демон `cron` проверяет таблицу cron, чтобы узнать, какие команды запускать в заданное время. Мы можем изменять таблицу cron, чтобы запланировать задачу или задание на регулярное выполнение в определенный день или дату, в определенное ежедневное время либо каждые несколько недель или месяцев.

Чтобы запланировать такие задачи или задания, внесите их в файл таблицы cron, расположенный по пути `/etc/crontab`. У таблицы cron семь полей: первые пять используются для планирования времени запуска задачи, шестое поле указывает пользователя, а седьмое поле используется для абсолютного пути к команде, которую вы хотите выполнить. Если бы мы использовали таблицу cron для планирования сценария, мы могли бы просто поместить абсолютный путь к сценарию в седьмое поле.

Каждое из пяти временных полей представляет отдельный элемент времени: минуту, час, день месяца, месяц и день недели, именно в таком порядке. Каждый элемент времени должен быть представлен численно, поэтому март представлен как `3` (нельзя просто ввести "March"). Дни недели начинаются с `0`, что означает воскресенье, и заканчиваются `7`, что тоже означает воскресенье. Таблица 16-1 суммирует это.

Таблица 16-1. Представления времени для использования в `crontab`

Поле	Единица времени	Представление
1	Минута	0-59
2	Час	0-23
3	День месяца	1-31
4	Месяц	1-12
5	День недели	0-7

Итак, если бы мы написали сценарий для сканирования всего мира на уязвимые открытые порты и хотели запускать его каждую ночь в 2:30, с понедельника по пятницу, мы могли бы запланировать его в файле `crontab`. Мы пройдем процесс внесения этой информации в `crontab` чуть позже, но сначала обсудим формат, которому нужно следовать, показанный в листинге 16-1.

```
M H DOM  MON  DOW   USER COMMAND
30 2 * * 1-5 root /root/myscanningscript
```

Листинг 16-1. Формат для планирования команд

Файл `crontab` заботливо подписывает столбцы для вас. Обратите внимание, что первое поле задает минуту (`30`), второе поле задает час (`2`), пятое поле задает дни (`1-5`, или с понедельника по пятницу), шестое поле определяет пользователя (`root`), а седьмое поле - путь к сценарию. Третье и четвертое поля содержат звездочки (`*`), потому что мы хотим, чтобы этот сценарий запускался каждый день с понедельника по пятницу независимо от дня месяца или месяца.

В листинге 16-1 пятое поле определяет диапазон для дня недели, используя дефис (`-`) между числами. Если вы хотите выполнять сценарий в несколько несмежных дней недели, можно разделить эти дни запятыми (`,`). Таким образом, вторник и четверг будут `2,4`.

Чтобы редактировать `crontab`, можно выполнить команду `crontab` с параметром `-e` (edit):

```
kali >crontab -e
Select an editor. To change later, run 'select-editor'.
1. /bin/nano <----easiest
2. /usr/bin/mcedit
3. /usr/bin/vim.basic
4. /usr/bin/vim.gtk
5. /usr/bin/vim.tiny
Choose 1-5 [1]:
```

При первом запуске этой команды она спросит, какой редактор вы хотели бы использовать. По умолчанию используется `/bin/nano`, вариант `1`. Если выбрать этот вариант, он откроется прямо в `crontab`.

Другой вариант, и часто лучший для новичка в Linux, - открыть `crontab` напрямую в любимом текстовом редакторе, что можно сделать так:

```
kali >leafpad /etc/crontab
```

Я использовал эту команду, чтобы открыть `crontab` в Leafpad. Фрагмент файла можно увидеть в листинге 16-2.

```
# /etc/crontab: общесистемный crontab
# В отличие от любого другого crontab, вам не нужно запускать команду
# 'crontab', чтобы установить новую версию после редактирования этого файла
# и файлов в /etc/cron.d. В этих файлах также есть поля имени пользователя,
# которых нет в других crontab.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# м ч день_мес мес день_нед пользователь команда
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts
#
```

Листинг 16-2. Файл `crontab`, используемый в текстовом редакторе

Теперь, чтобы настроить новую регулярно запланированную задачу, вам просто нужно ввести новую строку и сохранить файл.

Планирование задачи резервного копирования

Давайте сначала посмотрим на эту утилиту с точки зрения системного администратора. Как системный администратор, вы часто хотели бы запускать резервное копирование всех своих файлов в нерабочее время, когда система не используется и ресурсы легко доступны. (Резервное копирование системы, как правило, требует системных ресурсов, которых не хватает в рабочее время.) Идеальное время может быть в середине ночи на выходных. Вместо того чтобы входить в систему в 2:00 ночи с субботы на воскресенье (уверен, в это время у вас есть другие приоритеты), можно запланировать автоматический запуск резервного копирования на это время, даже если вас нет за компьютером.

Обратите внимание, что поле часа использует 24-часовой формат, поэтому, например, 13:00 записывается как `13`. Также обратите внимание, что дни недели (DOW) начинаются с воскресенья (`0`) и заканчиваются субботой (`6`).

Чтобы создать задание, вам просто нужно отредактировать файл `crontab`, добавив строку в предписанном формате. Итак, допустим, вы хотите создать регулярное задание резервного копирования с учетной записью пользователя с именем `backup`. Вы написали бы сценарий для резервного копирования системы и сохранили его как `systembackup.sh` в каталоге `/bin`, а затем запланировали бы это резервное копирование на каждую ночь с субботы на воскресенье в 2:00, добавив следующую строку в `crontab`:

```
00 2 * * 0 backup /bin/systembackup.sh
```

Обратите внимание, что подстановочный знак * используется для обозначения "любой", и использование его вместо цифры для дня месяца, месяца или дня недели читается как "все" дни или месяцы. Если читать эту строку слева направо, она говорит:

1. В начале часа (`00`),
2. второго часа (`2`),
3. любого дня месяца (`*`),
4. любого месяца (`*`),
5. в воскресенье (`0`),
6. от имени пользователя `backup`,
7. выполнить сценарий по пути `/bin/systembackup.sh`.

После этого демон `cron` будет выполнять этот сценарий каждое воскресное утро в 2:00, каждый месяц.

Если бы вы хотели, чтобы резервное копирование выполнялось только 15-го и 30-го числа каждого месяца, независимо от того, на какие дни недели приходятся эти даты, можно было бы изменить запись в `crontab`, чтобы она выглядела следующим образом:

```
00 2 15,30 * * backup /root/systembackup.sh
```

Обратите внимание, что поле дня месяца (DOM) теперь содержит `15,30`. Это говорит системе запускать сценарий только 15-го и 30-го числа каждого месяца, то есть примерно каждые две недели. Когда вы хотите указать несколько дней, часов или месяцев, их нужно перечислить через запятую, как мы сделали здесь.

Далее предположим, что компания требует от вас быть особенно внимательным с резервными копиями. Она не может позволить себе потерять даже день данных в случае отключения питания или сбоя системы. Тогда вам нужно было бы создавать резервную копию данных каждый будний вечер, добавив следующую строку:

```
00 23 * * 1-5 backup /root/systembackup.sh
```

Это задание будет выполняться в 23:00 (час `23`), каждый день месяца, каждый месяц, но только с понедельника по пятницу (дни `1-5`). Особенно обратите внимание, что мы обозначили дни с понедельника по пятницу, указав интервал дней (`1-5`), разделенный дефисом (`-`). Это также можно было бы обозначить как `1,2,3,4,5`; оба варианта работают совершенно нормально.

Использование crontab для планирования MySQLscanner

Теперь, когда вы понимаете основы планирования задания с помощью команды `crontab`, давайте запланируем сценарий `MySQLscanner.sh`, который ищет открытые порты MySQL и который вы построили в главе 8. Этот сканер ищет системы, на которых работает MySQL, проверяя открытый порт 3306.

Чтобы внести MySQLscanner.sh в файл crontab, отредактируйте файл, указав подробности этого задания, как мы делали с резервными копиями системы. Мы запланируем его запуск днем, пока вы на работе, чтобы он не занимал ресурсы, когда вы используете домашнюю систему. Для этого введите следующую строку в crontab:

```
00 9 * * * user /usr/share/MySQLscanner.sh
```

Мы настроили задание на запуск в 00 минут, в девятый час, каждый день месяца (*), каждый месяц (*), каждый день недели (*), и запускаем его как обычный пользователь. Нам нужно просто сохранить этот файл crontab, чтобы запланировать задание.

Теперь предположим, что вы хотите быть особенно осторожным и запускать этот сканер только по выходным и в 2:00 ночи, когда менее вероятно, что кто-то наблюдает за сетевым трафиком. Вы также хотите, чтобы он запускался только летом, с июня по август. Теперь ваше задание будет выглядеть так:

```
00 2 * 6-8 0,6 user /usr/share/MySQLscanner.sh
```

Вы добавили бы это в свой crontab вот так:

```
# /etc/crontab: общесистемный crontab
# В отличие от любого другого crontab, вам не нужно запускать команду
# 'crontab', чтобы установить новую версию после редактирования этого файла
# и файлов в /etc/cron.d. В этих файлах также есть поля имени пользователя,
# которых нет в других crontab.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# м ч день_мес мес день_нед пользователь команда
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily
)
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.monthly )
00 2 * 6-8 0,6 user /usr/share/MySQLscanner.sh
```

Теперь ваш MySQLscanner.sh будет запускаться только по выходным в июне, июле и августе в 2:00 ночи.

Сокращения crontab

В файле crontab есть несколько встроенных сокращений, которые можно использовать вместо указания времени, дня и месяца каждый раз. Они включают следующее:

```
@yearly
@annually
@monthly
@weekly
@daily
@midnight
@noon
@reboot
```

Итак, если вы хотели бы, чтобы сканер MySQL запускался каждую ночь в полночь, можно добавить в файл `crontab` следующую строку:

```
@midnight user /usr/share/MySQLscanner.sh
```

Использование rc-сценариев для запуска заданий при старте

Всякий раз, когда вы запускаете Linux-систему, выполняется ряд сценариев, чтобы настроить для вас окружение. Они известны как rc scripts. После того как ядро инициализировалось и загрузило все свои модули, ядро запускает демон, известный как `init` или `init.d`. Затем этот демон начинает выполнять ряд сценариев, найденных в `/etc/init.d/rc`. Эти сценарии включают команды для запуска многих служб, необходимых, чтобы ваша Linux-система работала так, как вы ожидаете.

Уровни запуска Linux

В Linux есть несколько уровней запуска, которые указывают, какие службы должны запускаться при загрузке. Например, уровень запуска 1 - это однопользовательский режим, и такие службы, как сеть, не запускаются на уровне запуска 1. Сценарии `rc` настроены на запуск в зависимости от выбранного уровня запуска:

Уровень запуска	Назначение
0	Остановить систему
1	Однопользовательский/минимальный режим
2-5	Многопользовательские режимы
6	Перезагрузить систему

Добавление служб в rc.d

Вы можете добавлять службы для запуска сценарием `rc.d` при старте, используя команду `update-rc.d`. Эта команда позволяет добавлять или удалять службы из сценария `rc.d`. Синтаксис `update-rc.d` прямой; вы просто указываете команду, затем имя сценария, а затем действие для выполнения, вот так:

```
kali >update-rc.d <имя сценария или службы>
<remove|defaults|disable|enable>
```

В качестве примера использования `update-rc.d` предположим, что вы всегда хотите, чтобы база данных PostgreSQL запускалась при загрузке системы, чтобы ваш фреймворк Metasploit мог использовать ее для хранения результатов тестирования на проникновение и хакинга. Вы использовали бы `update-rc.d`, чтобы добавить строку в сценарий `rc.d`, чтобы база была поднята и работала каждый раз, когда вы загружаете систему.

Прежде чем это сделать, давайте проверим, работает ли PostgreSQL в вашей системе уже сейчас. Это можно сделать с помощью команды `ps`, передав ее через pipe в фильтр, который ищет PostgreSQL с помощью `grep`, вот так:

```
kali >ps aux | grep postgresql
root 3876 0.0 0.0 12720 964pts/1 S+ 14.24 0.00 grep postgresql
```

Этот вывод говорит нам, что единственный процесс, который `ps` нашла запущенным для PostgreSQL, был самой командой, которую мы выполнили для его поиска, так что сейчас в этой системе база данных PostgreSQL не работает.

Теперь обновим наш `rc.d`, чтобы PostgreSQL запускалась автоматически при загрузке:

```
kali >update-rc.d postgresql defaults
```

Это добавляет строку в файл `rc.d`. Чтобы изменение вступило в силу, нужно перезагрузить систему. После этого давайте снова используем команду `ps` с `grep`, чтобы найти процесс PostgreSQL:

```
kali >ps aux | grep postgresql
postgresql 757 0.0 0.1 287636 25180 ? S March 14
0.00 /usr/lib/postgresql/9.6/bin/postgresql -D
/var/lib/postgresql/9.6/main
-c config_file=/etc/postgresql/9.6/main/postgresql.conf
root 3876 0.0 0.0 12720 964pts/1 S+ 14.24 0.00 grep postgresql
```

Как видите, PostgreSQL работает без того, чтобы вы вручную вводили какие-либо команды. Она автоматически запускается при загрузке системы, готовая и ожидающая использования с вашим Metasploit!

Добавление служб в загрузку через GUI

Если вам удобнее работать из GUI, чтобы добавлять службы при старте, можно загрузить простой GUI-инструмент `rcconf` из репозитория Kali, вот так:

```
kali >apt-get install rcconf
```

После завершения установки можно запустить `rcconf`, введя следующее:

```
kali >rcconf
```

Это откроет простой GUI, как на рисунке 16-1. Затем можно прокрутить доступные службы, выбрать те, которые вы хотите запускать при загрузке, и щелкнуть OK.

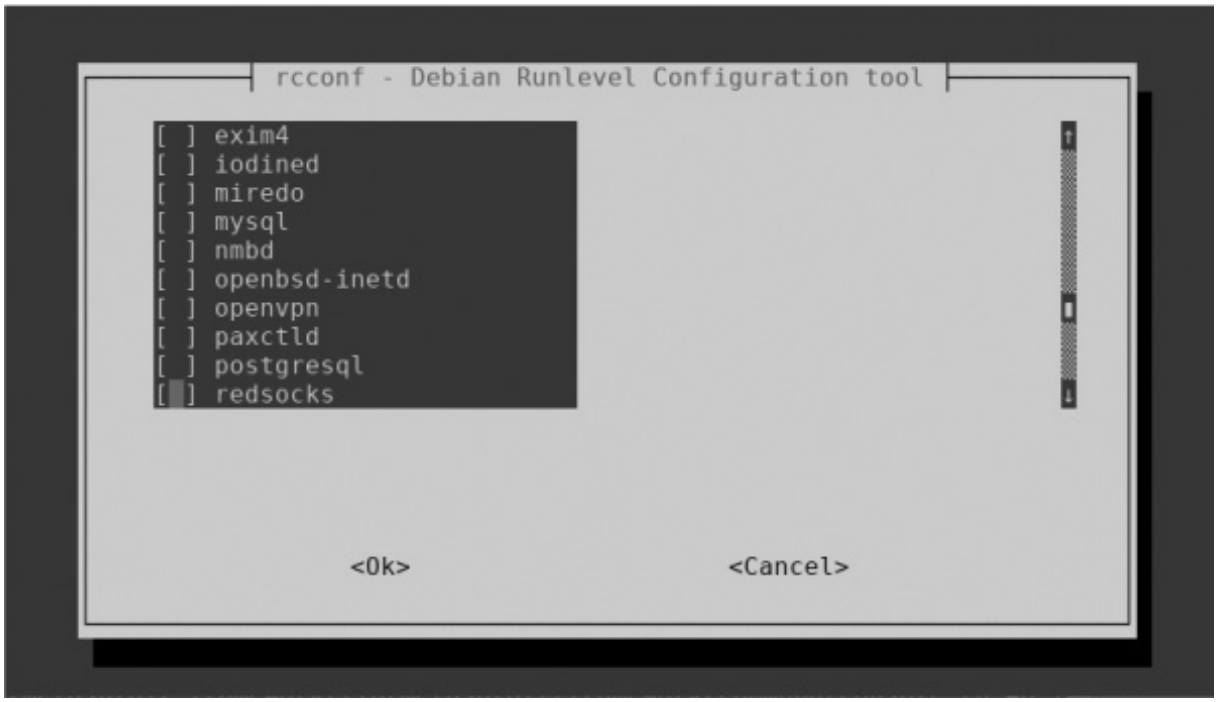

Рисунок 16-1. GUI `rcconf` для добавления служб в автозапуск

На этом рисунке можно видеть службу PostgreSQL второй снизу. Нажмите пробел, чтобы выбрать эту службу, нажмите TAB, чтобы выделить `<Ok>`, а затем нажмите ENTER. При следующей загрузке Kali PostgreSQL запустится автоматически.

Итоги

И системным администраторам, и хакерам часто нужно планировать запуск служб, сценариев и утилит через регулярные интервалы. Linux позволяет планировать почти любой сценарий или утилиту для регулярного запуска с помощью демона `cron`, который выполняет эти задания из таблицы cron. Кроме того, можно заставить службы запускаться автоматически при загрузке, используя команду `update-rc.d` или GUI-инструмент `rcconf`, чтобы обновить сценарии `rc.d`.

Упражнения

Прежде чем перейти к главе 17, попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Запланируйте запуск сценария `MySQLscanner.sh` каждую среду в 15:00.
2. Запланируйте запуск сценария `MySQLscanner.sh` каждый 10-й день месяца в апреле, июне и августе.
3. Запланируйте запуск сценария `MySQLscanner.sh` каждый вторник-четверг в 10:00.
4. Запланируйте ежедневный запуск сценария `MySQLscanner.sh` в полдень, используя сокращения.
5. Обновите свой сценарий `rc.d`, чтобы запускать PostgreSQL каждый раз при загрузке системы.
6. Загрузите и установите `rcconf` и добавьте базы данных PostgreSQL и MySQL для запуска при загрузке.

Глава 17. Основы сценариев Python для хакеров

Базовые навыки написания сценариев критически важны, чтобы стать мастером-хакером. Без развитых базовых навыков сценариев начинающий хакер, который просто использует инструменты, созданные кем-то другим, будет обречен на царство скрипт-кидди. Это означает, что вы будете ограничены использованием инструментов, разработанных кем-то другим, что уменьшает вероятность успеха и увеличивает вероятность обнаружения антивирусным (AV) программным обеспечением, системами обнаружения вторжений (IDS) и правоохранительными органами. Имея некоторые навыки сценариев, вы можете подняться в высший эшелон мастеров-хакеров!

В главе 8 мы рассмотрели основы bash-сценариев и построили несколько простых сценариев, включая `MySQLScanner.sh`, который находит системы, где работает повсеместная система баз данных MySQL. В этой главе мы начинаем рассматривать язык сценариев, наиболее широко используемый хакерами: Python. Многие из самых популярных хакерских инструментов написаны на Python, включая `sqlmap`, `scapy`, Social-Engineer Toolkit (SET), `w3af` и многие другие.

У Python есть несколько важных особенностей, делающих его особенно хорошо подходящим для хакинга, но, вероятно, самое важное - это огромное разнообразие библиотек, то есть заранее построенных модулей кода, которые можно импортировать извне и повторно использовать, и которые предоставляют мощную функциональность. Python поставляется с более чем 1 000 встроенных модулей, и еще многие доступны в различных других репозиториях.

Создание хакерских инструментов возможно и на других языках, таких как bash, Perl и Ruby, но модули Python делают создание таких инструментов гораздо проще.

Добавление модулей Python

Когда вы устанавливаете Python, вы также устанавливаете его набор стандартных библиотек и модулей, которые предоставляют широкий диапазон возможностей, включая встроенные типы данных, обработку исключений, числовые и математические модули, работу с файлами, криптографические службы, обработку интернет-данных и взаимодействие с интернет-протоколами (IP).

Несмотря на всю мощь, предлагаемую этими стандартными библиотеками и модулями, вам могут понадобиться или захотеться дополнительные сторонние модули. Сторонние модули, доступные для Python, обширны и, вероятно, являются причиной, по которой большинство хакеров

предпочитают Python для сценариев. Полный список сторонних модулей можно найти в PyPI (индекс пакетов Python, показан на рисунке 17-1) по адресу `http://www.pypi.org/`.

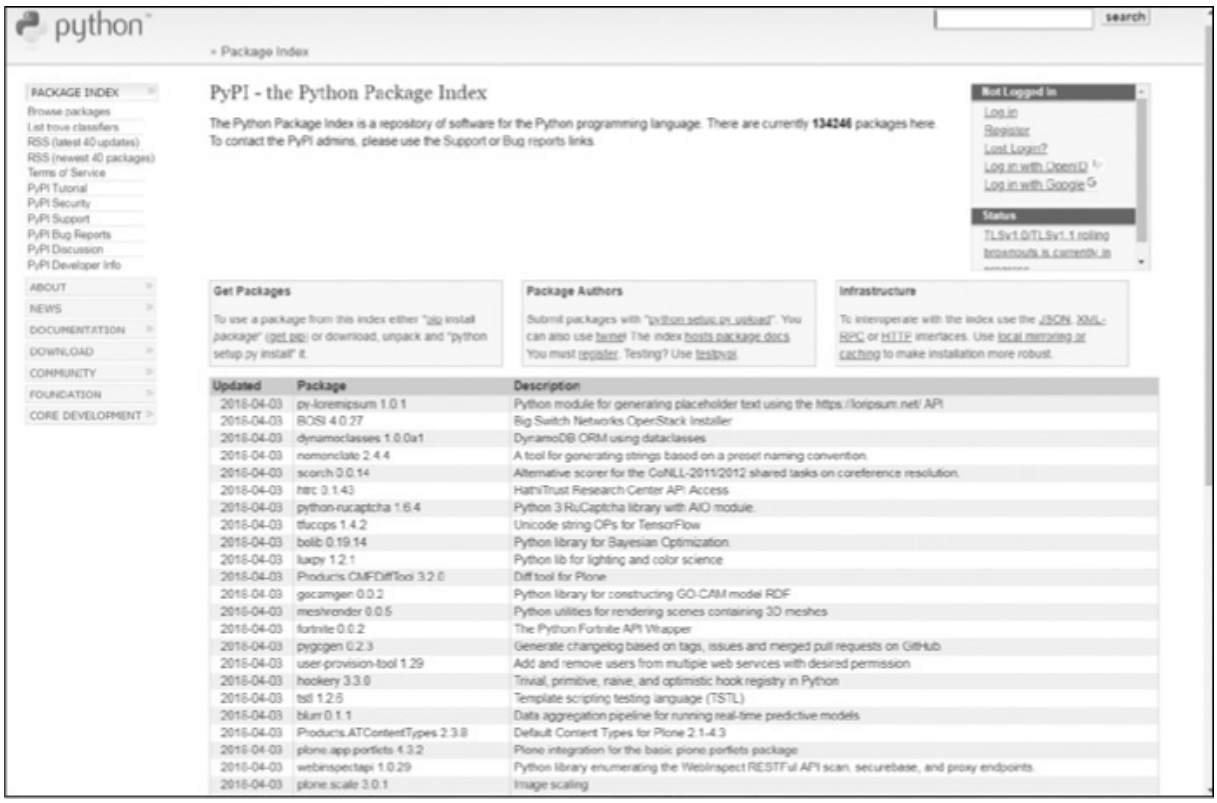

Рисунок 17-1. индекс пакетов Python

Использование pip

У Python есть менеджер пакетов специально для установки и управления пакетами Python, известный как `pip` (Pip устанавливает пакеты). Поскольку здесь мы работаем с Python 3, вам понадобится `pip` для Python 3, чтобы загружать и устанавливать пакеты. Вы можете загрузить и установить `pip` из репозитория Kali, введя следующее:

```
kali >apt-get install python3-pip
```

Теперь, чтобы загружать модули из PyPI, можно просто ввести это:

```
kali >pip3 install <package name>
```

Когда вы загружаете эти пакеты, они автоматически помещаются в каталог `/usr/local/lib/<python-version>/dist-packages`. Так, например, если вы использовали `pip`, чтобы установить Python-реализацию протокола SNMP для Python 3.6, вы найдете ее в `/usr/local/lib/python3.6/pysnmp`. Если вы не уверены, куда пакет был помещен в вашей системе (иногда разные дистрибутивы Linux используют разные каталоги), можно ввести `pip3`, затем `show` и имя пакета, как показано здесь:

```
kali >pip3 show pysnmp
Name: pysnmp
Version: 4.4.4
Summary: SNMP library for Python
Home-page: https://github.com/etingof/pysnmp
Author: Ilya Etingof <etingof@gmail.com>
Author-email: etingof@gmail.com
License: BSD
Location: /usr/local/lib/python3.6/dist-packages
Requires: ptsmi, pyansl, pycryptodomex
```

Видно, что это дает много информации о пакете, включая каталог, в котором он находится.

В качестве альтернативы использованию `pip` можно загрузить пакет напрямую с сайта (убедитесь, что он загружен в правильный каталог), распаковать его (см. главу 9 о том, как распаковывать программное обеспечение), а затем выполнить следующее:

```
kali >python setup.py install
```

Это установит любые распакованные пакеты, которые еще не были установлены.

Установка сторонних модулей

Чтобы установить сторонний модуль, созданный другим участником Python-сообщества (в отличие от официально выпущенного Python-пакета), можно просто использовать `wget`, чтобы загрузить его из места, где он хранится онлайн, распаковать модуль, а затем выполнить команду `python setup.py install`.

В качестве примера давайте загрузим и установим Python-модуль для инструмента сканирования портов, который мы использовали в главе 8, `nmap`, из его онлайн-репозитория по адресу `https://xael.org`.

Сначала нам нужно загрузить модуль с `xael.org`:

```
kali >wget http://xael.org/norman/python/python-nmap/python-nmap-0.3.4.tar.gz
--2014-12-28 17:48:32-- http://xael.org/norman/python/python-nmap/python-nmap-
0.3.4.tar.gz
Resolving xael.org (xael.org)...194.36.166.10
Connecting to xael.org (xael.org)|194.36.166.10|:80...connected.
# --snip--
2014-12-28 17:48:34 (113 KB/s) - 'python-nmap-0.3.4.tar.gz' saved
[40307/40307]
```

Здесь видно, что мы используем команду `wget` и полный URL пакета. После загрузки пакета нужно распаковать его с помощью `tar`, как вы узнали в главе 9:

```
kali >tar -xzf python-nmap-0.3.4.tar.gz
```

Затем перейдите в новый созданный каталог:

```
kali >cd python-nmap-0.3.4/
```

Наконец, в этом каталоге установите новый модуль, введя следующее:

```
kali >python setup.py install
running install
running build
running build_py
creating build
--snip--
running install_egg_info
writing /usr/local/lib/python2.7/dist-packages/python_nmap-0.3.4.egg.info
```

Бесчисленное множество других модулей можно получить таким же образом. После установки этого модуля `nmap` его можно использовать в ваших Python-сценариях, импортировав модуль. Подробнее об этом позже. Теперь давайте начнем писать сценарии.

Начало написания сценариев на Python

Теперь, когда вы знаете, как устанавливать модули в Python, я хочу охватить некоторые базовые концепции и терминологию Python, затем базовый синтаксис. После этого вы напишете несколько сценариев, которые будут полезны хакерам повсюду и, надеюсь, продемонстрируют мощь Python.

Как и с bash или любым другим языком сценариев, мы можем создавать Python-сценарии с помощью любого текстового редактора. Для этой главы, чтобы все было просто, я советую использовать простой текстовый редактор, такой как Leafpad, но полезно знать, что для Python доступен ряд интегрированных сред разработки, или IDE. IDE похожа на текстовый редактор со встроенными дополнительными возможностями, такими как цветовая подсветка, отладка и компиляция. В Kali встроена IDE PyCrust, но доступно много других IDE для загрузки, лучшей из которых, пожалуй, является PyCharm от JetBrains. Это отличная IDE с множеством улучшений, которые делают изучение Python проще и быстрее. Есть профессиональная платная версия и бесплатная редакция сообщества. Их можно найти по адресу `https://www.jetbrains.com/pycharm/`.

Когда вы завершите эту главу, если захотите продолжить изучение Python, PyCharm будет отличным инструментом, который поможет вам в разработке. Пока же мы будем использовать базовый текстовый редактор вроде Leafpad, чтобы сохранять простоту.

Обратите внимание, что изучение любого языка программирования требует времени и большого труда. Будьте терпеливы к себе: попробуйте освоить каждый из небольших сценариев, которые я предоставляю, прежде чем двигаться дальше.

> Форматирование в Python
>
> Одно различие между Python и некоторыми другими языками сценариев состоит в том, что форматирование в Python критически важно. Интерпретатор Python использует форматирование, чтобы определить, как сгруппирован код. Конкретные особенности форматирования менее важны, чем простая последовательность, особенно с уровнями отступов.

Если у вас есть группа строк кода, которую вы начинаете с двойного отступа, например, вы должны сохранять двойной отступ во всем блоке, чтобы Python распознал, что эти строки кода принадлежат друг другу. Это отличается от написания сценариев на других языках программирования, где форматирование необязательно и является хорошей практикой, но не требуется. Вы заметите это, когда будете проходить и практиковаться; это то, о чем всегда нужно помнить!

Переменные

Теперь перейдем к более практическим концепциям Python. Переменная - один из самых базовых типов данных в программировании, и вы уже встречали ее ранее в главе 8 при написании bash-сценариев. Простыми словами, переменная - это имя, связанное с определенным значением, так что всякий раз, когда вы используете это имя в программе, оно вызывает связанное значение.

Работает это так: имя переменной указывает на данные, хранящиеся в области памяти, которая может содержать любое значение, например целое число, вещественное число, строку, число с плавающей точкой, булево значение (истинное или ложное утверждение), список или словарь. Мы кратко рассмотрим все это в этой главе.

Чтобы познакомиться с основами, давайте создадим простой сценарий, показанный в листинге 17-1, в Leafpad и сохраним его как `hackers-arise_greetings.py`.

```
#!/usr/bin/python3
name="OccupyTheWeb"
print ("Привет, " + name + ", от Hackers-Arise. Лучшее место для изучения хакинга!")
```

Листинг 17-1. Ваша первая программа на Python

Первая строка просто говорит вашей системе, что вы хотите использовать интерпретатор Python для запуска этой программы, а не какой-либо другой язык. Вторая строка определяет переменную с именем `name` и присваивает ей значение (в данном случае `"OccupyTheWeb"`). Вам следует изменить это значение на собственное имя. Значение этой переменной находится в формате строковых символьных данных, то есть содержимое заключено в кавычки и обрабатывается как текст. В строки также можно помещать числа, и они будут обрабатываться как текст, то есть вы не сможете использовать их в числовых вычислениях.

Третья строка создает оператор `print()`, конкатенирующий `Привет,` со значением в переменной `name`, за которым следует текст `от Hackers-Arise. Лучшее место для изучения хакинга!` Оператор `print()` отобразит на вашем экране все, что вы передадите ему внутри круглых скобок.

Теперь, прежде чем запускать этот сценарий, нужно дать себе разрешение на его выполнение. Для этого нам нужна команда `chmod`. (Подробнее о разрешениях Linux см. главу 5.)

```
kali >chmod 755 hackers-arise_greetings.py
```

Как вы делали в главе 8 с bash-сценариями, чтобы выполнить сценарий, поставьте перед именем сценария точку и прямой слеш. Ваш текущий каталог не находится в переменной `$PATH` по причинам безопасности, поэтому нужно поставить перед именем сценария `./`, чтобы сказать системе искать имя файла в текущем каталоге и выполнить его.

Чтобы запустить этот конкретный сценарий, введите следующее:

```
kali >./hackers-arise_greetings.py
Привет, OccupyTheWeb, от Hackers-Arise. Лучшее место для изучения хакинга!
```

В Python каждый тип переменной обрабатывается как класс. Класс - это своего рода шаблон для создания объектов. Подробнее см. "объектно-ориентированное программирование (OOP)" на странице 192. В следующем сценарии я попытался продемонстрировать несколько типов переменных. Переменные могут содержать не только строки. Листинг 17-2 показывает несколько переменных, содержащих разные типы данных.

```
#!/usr/bin/python3
HackersAriseStringVariable = "Hackers-Arise - лучшее место для изучения хакинга"
HackersAriseIntegerVariable = 12
HackersAriseFloatingPointVariable = 3.1415
HackersAriseList = [1,2,3,4,5,6]
HackersAriseDictionary = {'name' : 'OccupyTheWeb', 'value' : 27}
print (HackersAriseStringVariable)
print (HackersAriseIntegerVariable)
print (HackersAriseFloatingPointVariable)
```

Листинг 17-2. Ряд структур данных, связанных с переменными

Это создает пять переменных, содержащих разные типы данных: строку, обрабатываемую как текст; целое число, числовой тип без десятичных знаков, который можно использовать в числовых операциях; число с плавающей точкой, числовой тип с десятичными знаками, который также можно использовать в числовых операциях; список, ряд значений, хранящихся вместе; и словарь, неупорядоченный набор данных, где каждое значение связано с ключом, то есть каждое значение в словаре имеет уникальный идентифицирующий ключ. Это полезно, когда вы хотите обратиться к значению или изменить его, ссылаясь на имя ключа. Например, предположим, у вас есть словарь с именем `fruit_color`, настроенный следующим образом:

```
fruit_color = {'apple' : 'red', 'grape' : 'green', 'orange' : 'orange'}
```

Если позже в сценарии вы захотите получить `fruit_color` для grape, вы просто вызываете его по ключу:

```
print (fruit_color['grape'])
```

Вы также можете изменять значения для конкретных ключей; например, здесь мы меняем цвет apple:

```
fruit_color['apple'] = 'green'
```

Мы подробнее обсудим списки и словари позже в главе.

Создайте этот сценарий в любом текстовом редакторе, сохраните его как `secondpythonscript.py`, а затем дайте себе разрешение на его выполнение, вот так:

```
kali >chmod 755 secondpythonscript.py
```

Когда мы запускаем этот сценарий, он печатает значения строковой переменной, переменной с целым числом и переменной с числом с плавающей точкой, вот так:

```
kali >./secondpythonscript.py
Hackers-Arise - лучшее место для изучения хакинга
12
3.1415
```

> Примечание
>
> В Python не нужно объявлять переменную перед присваиванием ей значения, как в некоторых других языках программирования.

Комментарии

Как и любой другой язык программирования и сценариев, Python имеет возможность добавлять комментарии. Комментарии - это просто части вашего кода, слова, предложения и даже абзацы, которые объясняют, что код должен делать. Python распознает комментарии в вашем коде и игнорирует их. Хотя комментарии не обязательны, они невероятно полезны, когда вы возвращаетесь к своему коду два года спустя и не можете вспомнить, что он должен делать. Программисты часто используют комментарии, чтобы объяснить, что делает определенный блок кода, или объяснить логику выбора конкретного метода кодирования.

Комментарии игнорируются интерпретатором. Это означает, что любые строки, обозначенные как комментарии, пропускаются интерпретатором, который просто продолжает работу, пока не встретит легитимную строку кода. Python использует символ #, чтобы обозначить начало однострочного комментария. Если вы хотите написать многострочные комментарии, можно использовать три двойные кавычки (""") в начале и конце секции комментария.

Как видно в следующем сценарии, я добавил короткий многострочный комментарий в наш простой сценарий `hackers-arise_greetings.py`.

```
#!/usr/bin/python3
"""
Это мой первый сценарий Python с комментариями. Комментарии помогают объяснять код
нам самим
и коллегам-программистам. В этом случае простой сценарий создает приветствие для
пользователя.
"""
name = "OccupyTheWeb"
print ("Привет, "+name+", от Hackers-Arise. Лучшее место для изучения хакинга!")
```

Когда мы снова выполняем сценарий, ничего не меняется по сравнению с прошлым запуском, как видно здесь:

```
kali >./hackers-arise_greetings.py
Привет, OccupyTheWeb, от Hackers-Arise. Лучшее место для изучения хакинга!
```

Он выполняется точно так же, как в листинге 17-1, но теперь у нас есть некоторая информация о нашем сценарии, когда мы вернемся к коду позже.

Функции

Функции в Python - это фрагменты кода, выполняющие определенное действие. Оператор `print()`, который вы использовали ранее, например, является функцией, которая отображает любые значения, которые вы ей передаете. В Python есть ряд встроенных функций, которые можно сразу импортировать и использовать. Большинство из них доступно в вашей установке Python по умолчанию в Kali Linux, хотя многие другие доступны из загружаемых библиотек. Давайте посмотрим всего на несколько из тысяч доступных вам функций:

`exit()` выходит из программы.

`float()` возвращает свой аргумент как число с плавающей точкой. Например, `float(1)` вернет `1.0`.

`help()` отображает справку по объекту, указанному его аргументом.

`int()` возвращает целочисленную часть своего аргумента (усекает).

`len()` возвращает количество элементов в списке или словаре.

`max()` возвращает максимальное значение из своего аргумента (списка).

`open()` открывает файл в режиме, указанном его аргументами.

`range()` возвращает список целых чисел между двумя значениями, указанными его аргументами.

`sorted()` принимает список как аргумент и возвращает его с элементами по порядку.

`type()` возвращает тип своего аргумента (например, `int`, `file`, `method`, `function`).

Вы также можете создавать собственные функции для выполнения пользовательских задач. Поскольку очень многие уже встроены в язык, всегда стоит проверить, существует ли функция уже, прежде чем тратить силы на ее создание самостоятельно. Есть много способов сделать такую проверку. Один из них - посмотреть официальную документацию Python, доступную по адресу `https://docs.python.org`. Выберите версию, с которой работаете, а затем выберите справочник библиотеки.

Списки

Многие языки программирования используют массивы как способ хранения нескольких отдельных объектов. Массив - это список значений, которые можно получать, удалять, заменять или обрабатывать разными способами, обращаясь к определенному значению в массиве по его позиции в списке, известной как индекс. Важно отметить, что Python, как и многие другие среды программирования, начинает считать индексы с `0`, поэтому первый элемент в списке имеет индекс `0`, второй - индекс `1`, третий - индекс `2` и так далее. Так, например, если бы мы хотели получить доступ к третьему значению в массиве, мы могли бы сделать это с помощью `array[2]`. В Python есть несколько реализаций массивов, но, вероятно, самая распространенная реализация известна как списки.

Списки в Python являются итерируемыми, что означает, что список может предоставлять последовательные элементы, когда вы проходите его целиком (см. "Циклы" на странице 198). Это полезно, потому что довольно часто, когда мы используем списки, мы просматриваем их, чтобы найти определенное значение, напечатать значения одно за другим или взять значения из одного списка и поместить их в другой список.

Итак, представим, что нам нужно отобразить четвертый элемент в нашем списке `HackersAriseList` из листинга 17-2. Мы можем получить доступ к этому элементу и напечатать его, вызвав имя списка, `HackersAriseList`, за которым следует индекс нужного элемента в квадратных скобках.

Чтобы проверить это, добавьте следующую строку в конец сценария `secondpythonscript.py`, чтобы напечатать элемент с индексом `3` в `HackersAriseList`:

```
# пропущенный код
print (HackersAriseStringVariable)
print (HackersAriseIntegerVariable)
print (HackersAriseFloatingPointVariable)
print (HackersAriseList[3])
```

Когда мы снова запускаем этот сценарий, видно, что новый оператор `print` печатает 4 рядом с остальным выводом:

```
kali >./secondpythonscript.py
Hackers-Arise - лучшее место для изучения хакинга
12
3.1415
4
```

Модули

Модуль - это просто секция кода, сохраненная в отдельный файл, чтобы вы могли использовать ее столько раз, сколько нужно в программе, без необходимости вводить ее заново целиком. Если вы хотите использовать модуль или любой код из модуля, его нужно импортировать. Как обсуждалось ранее, использование стандартных и сторонних модулей - одна из ключевых особенностей, делающих Python таким мощным для хакера. Если бы мы хотели использовать модуль `nmap`, установленный ранее, мы добавили бы в свой сценарий следующую строку:

```
import nmap
```

Позже в этой главе мы используем два очень полезных модуля: `socket` и `ftplib`.

Объектно-ориентированное программирование (OOP)

Прежде чем мы углубимся в Python, вероятно, стоит потратить несколько минут на обсуждение концепции объектно-ориентированного программирования (OOP). Python, как и большинство современных языков программирования (C++, Java и Ruby, если назвать несколько), следует модели OOP. Рисунок 17-2 показывает базовую концепцию OOP: главный инструмент языка - объект, у которого есть свойства в форме атрибутов и состояний, а также методы, то есть действия, выполняемые объектом или над объектом.

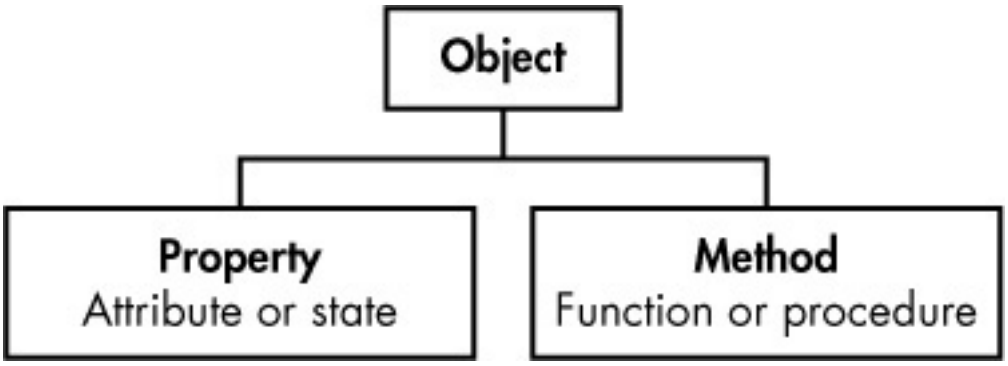

Рисунок 17-2. Иллюстрация объектно-ориентированного программирования

Идея языков программирования на основе OOP состоит в создании объектов, которые действуют как вещи в реальном мире. Например, автомобиль - это объект, у которого есть свойства, такие как колеса, цвет, размер и тип двигателя; у него также есть методы, то есть действия, которые автомобиль выполняет, такие как ускорение и запирание дверей. С точки зрения естественного человеческого языка объект - это существительное, свойство - прилагательное, а метод обычно глагол.

Объекты являются членами класса, который по сути является шаблоном для создания объектов с общими начальными переменными, свойствами и методами. Например, предположим, у нас есть

класс с именем `cars`; наш автомобиль (BMW) был бы членом класса автомобилей. Этот класс также включал бы другие объекты/автомобили, такие как Mercedes и Audi, как показано на рисунке 17-3.

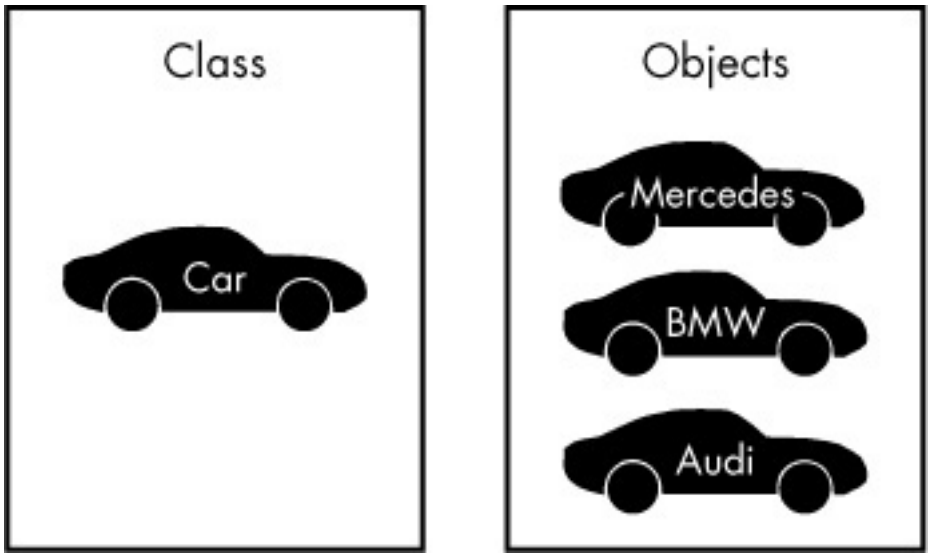

Рисунок 17-3. OOP-классы и объекты

У классов также могут быть подклассы. У нашего класса `car` есть подкласс BMW, а объект этого подкласса может быть моделью 320i.

У каждого объекта будут свойства (`make`, `model`, `year` и `color`) и методы (`start`, `drive` и `park`), как показано на рисунке 17-4.

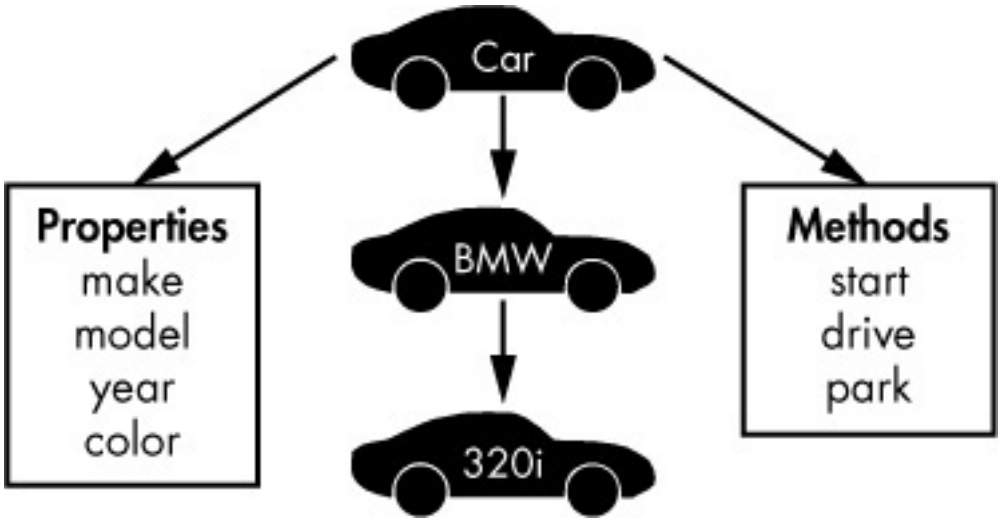

Рисунок 17-4. OOP-свойства и методы

В OOP-языках объекты наследуют характеристики своего класса, поэтому BMW 320i унаследовала бы методы `start`, `drive` и `park` от класса `car`.

Эти концепции OOP критически важны для понимания того, как работают Python и другие OOP-языки, как вы увидите в сценариях в следующих разделах.

Сетевые коммуникации в Python

Прежде чем перейти к новым концепциям Python, давайте используем то, что вы уже узнали, чтобы написать пару хакерских сценариев, связанных с сетевыми соединениями.

Создание TCP-клиента

Мы создадим сетевое соединение в Python с помощью модуля `socket`. Я уже упоминал, что Python поставляется с библиотекой модулей для множества задач. В этом случае нам понадобится модуль `socket`, чтобы создать TCP-соединение. Посмотрим его в действии.

Взгляните на сценарий в листинге 17-3 с именем `HackersAriseSSHBannerGrab.py` (я знаю, имя длинное, но потерпите). Баннер - это то, что приложение представляет, когда кто-то или что-то подключается к нему. Это похоже на то, как приложение отправляет приветствие, объявляя, чем оно является. Хакеры используют технику, известную как получение баннеров, чтобы узнать критически важную информацию о том, какое приложение или служба работает на порту.

```
#!/usr/bin/python3
(1) import socket
(2) s = socket.socket()
(3) s.connect(("192.168.1.101", 22))
(4) answer = s.recv(1024)
(5) print (answer)
s.close()
```

Листинг 17-3. Python-сценарий для получения баннеров

Сначала мы импортируем модуль `socket` (1), чтобы использовать его функции и инструменты. Здесь мы будем использовать сетевые инструменты из модуля `socket`, чтобы он взял на себя взаимодействие соединения по сети. Сокет предоставляет способ двум компьютерным узлам общаться друг с другом. Обычно один является сервером, а другой клиентом.

Затем мы создаем новую переменную `s` и связываем ее с классом `socket` из модуля `socket` (2). Так нам не нужно ссылаться на полный синтаксис `socket.socket()` всякий раз, когда мы хотим использовать класс `socket`; можно просто использовать имя переменной `s`.

Затем мы используем метод `connect()` из модуля `socket` (3), чтобы создать сетевое соединение с конкретным IP и портом. Помните, что методы - это функции, доступные для конкретного объекта. Синтаксис - `object.method` (например, `socket.connect`). В этом случае я подключаюсь к IP-адресу `192.168.1.101`, который является IP-адресом машины в моей сети, и порту `22`, который является SSH-портом по умолчанию. Вы можете проверить это на другом экземпляре Linux или Kali. У большинства порт 22 открыт по умолчанию.

После создания соединения можно сделать ряд вещей. Здесь мы используем метод получения `recv`, чтобы прочитать 1024 байта данных из сокета (4) и сохранить их в переменной с именем `answer`; эти 1024 байта будут содержать информацию баннера. Затем мы печатаем содержимое этой переменной на экран функцией `print()` (5), чтобы увидеть, какие данные были переданы через этот сокет, что позволяет нам подсмотреть их! В последней строке мы закрываем соединение.

Сохраните этот сценарий как `HackersAriseSSHBannerGrab.py`, а затем измените его разрешения командой `chmod`, чтобы можно было его выполнить.

Давайте запустим этот сценарий, чтобы подключиться к другой Linux-системе (вы можете использовать Ubuntu-систему или даже другую Kali-систему) на порту 22. Если SSH работает на этом порту, мы должны быть способны прочитать баннер в переменную `answer` и напечатать его на экран, как показано здесь:

```
kali >./HackersAriseSSHBannerGrab.py
SSH-2.0-OpenSSH_7.3p1 Debian-1
```

Мы только что создали простой Python-сценарий для получения баннеров! Этот сценарий можно использовать, чтобы узнать, какое приложение, версия и операционная система работают по этому IP-адресу и порту. Это дает нам ключевую информацию, которая нужна хакеру перед атакой на систему. По сути, это то, что сайт Shodan.io делает почти для каждого IP-адреса на планете, каталогизируя и индексируя эту информацию, чтобы мы могли ее искать.

Создание TCP-слушателя

Мы только что создали TCP-клиент, который может установить соединение с другим TCP/IP-адресом и портом, а затем подсмотреть передаваемую информацию. Этот сокет также можно использовать, чтобы создать TCP-слушатель, принимающий соединения от внешних систем к вашему серверу. Давайте попробуем сделать это далее.

В Python-сценарии, показанном в листинге 17-4, вы создадите сокет на любом порту вашей системы, который, когда кто-то подключится к этому сокету, соберет ключевую информацию о системе подключающегося. Введите сценарий и сохраните его как `tcp_server.py`. Обязательно дайте себе разрешения на выполнение с помощью `chmod`.

```
#!/usr/bin/python3
import socket
(1) TCP_IP = "192.168.181.190"
TCP_PORT = 6996
BUFFER_SIZE = 100
(2) s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
(3) s.bind((TCP_IP, TCP_PORT))
(4) s.listen (1)
(5) conn, addr = s.accept()
print ('Connection address: ', addr )
while 1:
    data = conn.recv(BUFFER_SIZE)
    if not data:
        break
    print("Received data: ", data)
    conn.send(data) # echo
conn.close()
```

Листинг 17-4. Python-сценарий TCP-слушателя

Мы объявляем, что хотим запускать сценарий с интерпретатором Python, а затем импортируем модуль `socket`, как раньше, чтобы использовать его возможности. Затем мы определяем переменные для хранения информации о TCP/IP-адресе, порте, который нужно слушать, и размере буфера данных, которые хотим захватить от подключающейся системы (1).

Мы определяем сокет (2) и привязываем сокет к IP-адресу и порту (3), используя только что созданные переменные. Мы говорим сокету слушать, используя метод `listen()` из библиотеки `socket` (4).

Затем мы захватываем IP-адрес и порт подключающейся системы с помощью метода `accept` библиотеки `socket`, и печатаем эту информацию на экран, чтобы пользователь мог ее увидеть (5). Обратите внимание на синтаксис `while 1:` здесь; мы подробнее обсудим это позже в главе, но пока просто знайте, что он используется для бесконечного запуска кода с отступом, который идет

после него, то есть Python продолжает проверять данные, пока программа не будет остановлена.

Наконец, мы помещаем информацию от подключающейся системы в буфер, печатаем ее, а затем закрываем соединение.

Теперь перейдите на другой компьютер в вашей сети и используйте браузер, чтобы подключиться к порту `6996`, указанному в нашем сценарии. Запустите сценарий `tcp_server.py`, и вы должны быть способны подключиться и собрать ключевую информацию об этой системе, включая IP-адрес и порт подключающейся системы, как показано здесь:

```
kali >./tcp_server.py
Connection Address: ('192.168.181.190', 45368)
Received data: Get /HTTP/1.1
Host:192.168.181.190:6996
User-Agent:Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gec
# --snip--
```

Это критически важная информация, которую хакеру нужно собрать, прежде чем решать, какой эксплойт использовать. Эксплойты (или хаки) очень специфичны для операционной системы, приложения и даже используемого языка, поэтому хакеру нужно знать как можно больше информации о цели, прежде чем продолжать. Этот акт сбора информации перед взломом часто называют разведкой. Вы только что разработали инструмент, который будет собирать ключевую разведывательную информацию о потенциальной цели, очень похожий на популярный хакерский инструмент `p0F`!

Словари, циклы и управляющие операторы

Давайте продолжим расширять ваше понимание Python, а затем используем все, что вы уже узнали, чтобы построить взломщик паролей для FTP-сервера.

Словари

Словари хранят информацию как неупорядоченные пары, где каждая пара содержит ключ и связанное с ним значение. Мы можем использовать словарь, чтобы хранить список элементов и дать каждому элементу метку, чтобы использовать этот элемент и ссылаться на него индивидуально. Можно использовать словарь, например, чтобы хранить идентификаторы пользователей и связанные с ними имена или хранить известные уязвимости, связанные с конкретным хостом. Словари в Python действуют как ассоциативные массивы в других языках.

Как и списки, словари являются итерируемыми, то есть мы используем управляющую структуру, такую как оператор `for`, чтобы пройти весь словарь, присваивая каждый элемент словаря переменной, пока не дойдем до конца словаря.

Среди прочего, вы можете использовать эту структуру при создании взломщика паролей, который проходит каждый пароль, хранящийся в словаре, пока один не сработает или пока взломщик не дойдет до конца словаря.

Синтаксис создания словаря выглядит так:

```
dict = {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

Обратите внимание, что для словарей используются фигурные скобки, а элементы разделяются запятой. Можно включать столько пар ключ-значение, сколько хотите.

Управляющие операторы

Управляющие операторы позволяют вашему коду принимать решения на основе некоторого условия. В Python есть ряд способов контролировать поток сценария.

Давайте рассмотрим некоторые из этих структур в Python.

Оператор if

Структура `if` в Python, как и во многих других языках программирования, включая bash, используется для проверки, истинно условие или нет, и запуска разных наборов кода для каждого сценария. Синтаксис выглядит так:

```
if условное выражение:
    выполнить этот код, если выражение истинно
```

Оператор `if` содержит условие, которое может быть чем-то вроде `if variable < 10`, например. Если условие выполнено, выражение оценивается как `true`, а затем выполняется следующий код, известный как управляющий блок. Если оператор оценивается как `false`, инструкции в управляющем блоке пропускаются и не выполняются.

В Python управляющий блок должен иметь отступ. Этот отступ идентифицирует управляющий блок для интерпретатора. Следующий оператор без отступа находится вне управляющего блока и поэтому не является частью оператора `if`; именно так Python понимает, куда пропустить, если условие не выполнено.

if...else

Структура `if...else` в Python выглядит так:

```
if условное выражение:
    # выполнить этот код, когда условие выполнено
else:
    # выполнить этот код, когда условие не выполнено
```

Как и раньше, сначала интерпретатор проверяет условие в выражении `if`. Если оно оценивается как `true`, интерпретатор выполняет инструкции в управляющем блоке. Если условный оператор оценивается как `false`, вместо этого выполняется управляющий блок, следующий за оператором `else`.

Например, здесь у нас фрагмент кода, который проверяет значение идентификатора пользователя; если оно равно `0` (root-пользователь в Linux всегда имеет UID 0), то мы печатаем сообщение `Вы пользователь root.` Иначе, если это любое другое значение, мы печатаем сообщение `Вы НЕ пользователь root.`

```
if userid == 0:
    print("Вы пользователь root")
else:
    print("Вы НЕ пользователь root")
```

Циклы

Циклы - еще одна очень полезная структура в Python. Циклы позволяют программисту повторять блок кода несколько раз, в зависимости от значения или условия. Два наиболее широко используемых - `while` и `for`.

Цикл while

Цикл `while` оценивает булево выражение (выражение, которое может оцениваться только как `true` или `false`) и продолжает выполнение, пока выражение оценивается как `true`. Например, мы могли бы создать фрагмент кода, который печатает каждое число от 1 до 10, а затем выходит из цикла, вот так:

```
count = 1
while (count <= 10):
    print(count)
    count += 1
```

Затем управляющий блок с отступом выполняется до тех пор, пока условие истинно.

Цикл for

Цикл `for` может присваивать значения из списка, строки, словаря или другой итерируемой структуры индексной переменной при каждом проходе цикла, позволяя нам использовать каждый элемент в структуре один за другим. Например, можно использовать цикл `for`, чтобы пробовать пароли, пока не найдем совпадение, вот так:

```
for password in passwords:
    attempt = connect(username, password)
    if attempt == "230":
        print("Пароль найден: " + password)
        sys.exit(0)
```

В этом фрагменте кода мы создаем оператор `for`, который проходит список предоставленных нами паролей и пытается подключиться с именем пользователя и паролем. Если попытка подключения получает код `230`, который является кодом успешного соединения, программа печатает `Пароль найден:`, а затем пароль. Затем она выходит. Если она не получает `230`, она продолжит перебирать каждый из оставшихся паролей, пока не получит `230` или пока не исчерпает список паролей.

Улучшение наших хакерских сценариев

Теперь, имея немного больше фона по структурам циклов Python и условным операторам, вернемся к нашему сценарию получения баннеров и добавим некоторые возможности.

Мы добавим список портов, с которых хотим получить баннер, вместо того чтобы слушать только один порт, а затем пройдем список циклом, используя оператор `for`. Таким образом, мы можем искать и получать баннеры для нескольких портов и отображать их на экран.

Сначала создадим список и поместим в него дополнительные порты. Откройте `HackersAriseSSHBannerGrab.py`, и мы будем работать оттуда. Листинг 17-5 показывает полный

код. Обратите внимание, что серые строки остались прежними; черные строки - те, которые нужно изменить или добавить. Мы попробуем получить баннеры для портов 21 (ftp), 22 (ssh), 25 (smtp) и 3306 (mysql).

```
#!/usr/bin/python3
import socket
Ports = [21, 22, 25, 3306]  # (1)
for i in range(0, 4):  # (2)
    s = socket.socket()
    Port = Ports[i]  # (3)
    print("Это баннер для порта")
    print(Port)
    s.connect(("192.168.1.101", Port))  # (4)
    answer = s.recv(1024)
    print(answer)
    s.close()
```

Листинг 17-5. Улучшение инструмента получения баннеров

Мы создаем список с именем `Ports` (1) и добавляем четыре элемента, каждый из которых представляет порт. Затем мы создаем оператор `for`, который проходит этот список четыре раза, поскольку в нем четыре элемента (2).

Помните, что когда вы используете цикл `for`, код, связанный с циклом, должен быть с отступом под оператором `for`.

Нам нужно изменить программу, чтобы отражать использование переменной из списка при каждой итерации. Для этого мы создаем переменную с именем `Port` и присваиваем ей значение из списка на каждой итерации (3). Затем мы используем эту переменную в нашем соединении (4).

Когда интерпретатор доходит до этого оператора, он попытается подключиться к тому порту, который назначен переменной на IP-адресе.

Теперь, если вы запустите этот сценарий на системе, где все перечисленные порты открыты и включены, вы должны увидеть что-то похожее на листинг 17-6.

```
kali >./HackersArisePortBannerGrab.py
Это баннер для порта
21
220 (vsFTPd 2.3.4)
Это баннер для порта
22
SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
Это баннер для порта
25
220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
Это баннер для порта
3306
5.0.51a-3ubuntu5
```

Листинг 17-6. Вывод инструмента получения баннеров по портам

Обратите внимание, что сценарий нашел порт 21 открытым с работающим на нем vsFTPd 2.3.4, порт 22 открытым с OpenSSH 4.7, порт 25 с Postfix и порт 3306 с MySQL 5.0.51a.

Мы только что успешно построили многопортовый инструмент получения баннеров на Python для выполнения разведки целевой системы. Инструмент сообщает нам, какая служба работает на порту и версию этой службы! Это ключевая информация, которая нужна хакеру перед продолжением атаки.

Исключения и взломщики паролей

Любой код, который вы пишете, будет подвержен риску ошибок или исключений. В терминах программирования исключение - это все, что нарушает нормальный поток вашего кода, обычно ошибка, вызванная неправильным кодом или вводом. Чтобы работать с возможными ошибками, мы используем обработку исключений, то есть просто код, который обрабатывает конкретную проблему, показывает сообщение об ошибке или даже использует исключение для принятия решений. В Python у нас есть структура `try/except`, чтобы обрабатывать такие ошибки или исключения.

Блок `try` пытается выполнить некоторый код, и если возникает ошибка, оператор `except` обрабатывает эту ошибку. В некоторых случаях мы можем использовать структуру `try/except` для принятия решений, похожую на `if...else`. Например, можно использовать `try/except` во взломщике паролей, чтобы попробовать пароль и, если возникает ошибка из-за несовпадения пароля, перейти к следующему паролю с оператором `except`. Давайте попробуем это сейчас.

Введите код из листинга 17-7 и сохраните его как `ftpcracker.py`; мы разберем его через минуту. Этот сценарий спрашивает у пользователя номер FTP-сервера и имя пользователя любой FTP-учетной записи, которую он хочет взломать. Затем он читает внешний текстовый файл, содержащий список возможных паролей, и пробует каждый из них в попытке взломать FTP-учетную запись. Сценарий делает это, пока либо не добьется успеха, либо не закончатся пароли.

```
#!/usr/bin/python3
import ftplib
server = input("FTP-сервер: ")  # (1)
user = input("имя пользователя: ")  # (2)
Passwordlist = input("Путь к списку паролей > ")  # (3)
try:  # (4)
    with open(Passwordlist, 'r') as pw:
        for word in pw:
            word = word.strip('\r').strip('\n')  # (5)
            try:  # (6)
                ftp = ftplib.FTP(server)
                ftp.login(user, word)
                print('Успех! Пароль: ' + word)  # (7)
            except:  # (8)
                print('пробуем дальше...')
except:
    print('Ошибка списка слов')
```

Листинг 17-7. Python-сценарий взломщика FTP-пароля

Мы будем использовать инструменты из модуля `ftplib` для протокола FTP, поэтому сначала импортируем его. Затем создаем переменную с именем `server` и другую переменную с именем `user`, которые будут хранить команды для пользовательского ввода. Ваш сценарий попросит пользователя ввести IP-адрес FTP-сервера (1) и имя пользователя учетной записи (2), которую пользователь пытается взломать.

Затем мы спрашиваем у пользователя путь к списку паролей (3). В Kali Linux можно найти множество списков паролей, введя `locate wordlist` в терминале.

Затем мы начинаем блок кода `try`, который будет использовать список паролей, предоставленный пользователем, чтобы попытаться взломать пароль для имени пользователя, предоставленного пользователем.

Обратите внимание, что мы используем новую Python-функцию `strip()` (5). Эта функция удаляет первый и последний символ строки (в данном случае `Passwordlist`). Это необходимо, если пароли в этом списке имеют предшествующий пробел или запятую. Функция `strip()` удаляет их и оставляет только строку символов потенциального пароля. Если мы не удалим пробельные символы, мы можем получить ложноотрицательный результат.

Затем мы используем второй блок `try` (6). Здесь мы используем модуль `ftplib`, чтобы сначала подключиться к серверу, используя IP-адрес, предоставленный пользователем, а затем попробовать следующий пароль из списка паролей на этой учетной записи.

Если комбинация имени пользователя и пароля приводит к ошибке, блок завершается и переходит к предложению `except` (8), где печатает `пробуем дальше`, а затем возвращается к началу предложения `for` и берет следующий пароль из списка паролей для попытки.

Если комбинация успешна, успешный пароль печатается на экран (7). Последняя строка перехватывает любые другие ситуации, которые иначе привели бы к ошибкам. Примером было бы, если пользователь ввел что-то, что программа не может обработать, например плохой путь к списку слов или отсутствующий список слов.

Теперь запустим этот сценарий против FTP-сервера по адресу `192.168.1.101` и посмотрим, можем ли мы взломать пароль пользователя root. Я использую список паролей с именем `bigpasswordlist.txt` в своем рабочем каталоге. Возможно, вам нужно будет указать полный путь к тому списку паролей, который вы используете, если он не находится в вашем рабочем каталоге (например, `/usr/share/bigpasswordlist.txt`).

```
kali >./ftpcracker.py
FTP-сервер: 192.168.1.101
имя пользователя: root
Путь к списку паролей >bigpasswordlist.txt
пробуем дальше...
пробуем дальше...
пробуем дальше...
--snip--
Успех! Пароль: toor
```

Как видите, `ftpcracker.py` успешно нашел пароль для пользователя root и вывел его на экран.

Итоги

Чтобы выйти за пределы статуса скрипт-кидди, хакер должен освоить язык сценариев, и Python обычно является хорошим первым выбором благодаря своей универсальности и относительно небольшой кривой обучения. Большинство хакерских инструментов написано на Python, включая `sqlmap`, `scapy` и многие другие. Здесь вы изучили некоторые основы Python, которые можно использовать для создания полезных, хотя и простых хакерских инструментов, включая инструмент получения баннеров и FTP-взломщик паролей. Чтобы узнать больше о Python, я настоятельно рекомендую отличную книгу No Starch Press "Автоматизируйте скучные задачи с Python" (2015) Эла Свейгарта.

Упражнения

Попробуйте применить навыки, полученные в этой главе, выполнив следующие упражнения:

1. Постройте инструмент SSH для получения баннеров из листинга 17-5, а затем отредактируйте его, чтобы получать баннер на порту 21.
2. Вместо жестко заданного IP-адреса в сценарии отредактируйте свой инструмент получения баннеров так, чтобы он запрашивал IP-адрес у пользователя.
3. Отредактируйте `tcp_server.py`, чтобы он запрашивал у пользователя порт для прослушивания.
4. Постройте FTPcracker из листинга 17-7, а затем отредактируйте его, чтобы использовать список слов для переменной пользователя (похоже на то, что мы сделали с паролем), вместо того чтобы запрашивать ввод у пользователя.
5. Добавьте предложение `except` в инструмент получения баннеров, который печатает `no answer`, если порт закрыт.

Задняя обложка

Начинающий хакер? Начните здесь.

Охватывает Kali Linux и Python 3

Если вы начинаете захватывающий путь хакинга, кибербезопасности и тестирования на проникновение, "Основы Linux для хакеров" - отличный первый шаг. Используя Kali Linux, продвинутый дистрибутив Linux для тестирования на проникновение, вы изучите основы использования операционной системы Linux и получите инструменты и техники, которые понадобятся, чтобы взять под контроль Linux-окружение.

Сначала вы узнаете, как установить Kali на виртуальную машину, и получите введение в базовые концепции Linux. Затем вы возьметесь за более широкие темы Linux, такие как манипуляция текстом, управление правами файлов и каталогов и управление пользовательскими переменными окружения. Затем вы сосредоточитесь на фундаментальных концепциях хакинга, таких как безопасность и анонимность, и изучите навыки написания сценариев с bash и Python.

Практические руководства и упражнения на протяжении всей книги будут закреплять и проверять ваши навыки, пока вы учитесь:

• Заметать следы, изменяя свою сетевую информацию и манипулируя утилитой журналирования `rsyslog`

• Писать инструмент для сканирования сетевых соединений, а также подключаться к беспроводным сетям и слушать их

• Сохранять свою интернет-активность скрытной, используя Tor, прокси-серверы, VPN и зашифрованную электронную почту

• Писать bash-сценарий для сканирования открытых портов в поисках потенциальных целей

• Использовать службы вроде MySQL, Apache веб-сервер и OpenSSH и злоупотреблять ими

• Создавать собственные хакерские инструменты, такие как удаленная шпионская видеокамера и взломщик паролей

Хакинг сложен, и единственного пути внутрь не существует. Почему бы не начать с начала с "Основ Linux для хакеров"?

Об авторе

OccupyTheWeb - консультант по информационной безопасности, специалист по компьютерной криминалистике и преподаватель с более чем 20-летним опытом в отрасли. Он поддерживает обучающий сайт Hackers-Arise (`https://www.hackers-arise.com/`) и обучает персонал армии США, подрядчиков Министерства обороны и федеральных сотрудников информационной безопасности и хакингу.

Лучшее развлечение для гиков™

www.nostarch.com