TOP10 ошибок конфигурации Linux/BSD

крис касперски ака мыщъх, no-email

воздвигнуть Linux/BSD – не проблема, инсталлятор все сделает за нас, а вот правильно настроить систему, чтобы ее тут же не атаковали хакеры, удается далеко не каждому. проанализировав ситуацию, мыщъх отобрал десяток наиболее распространенных ошибок, допускаемых не только начинающими Linux'идами, но и "матерыми" пользователями.

введение

Логотип на главной странице OpenBSD все видели? "Всего лишь две удаленных уязвимости в конфигурации по умолчанию за десять лет промышленной эксплуатации". Означает ли это, что установив OpenBSD на свою машину, мы можем пить пиво (курить траву, сушить грибы) и ничего не опасаться? Нет и еще раз нет!

Несмотря на то, что в xBSD и, особенно, в Linux'е имеется достаточное количество дыр под которые написано множество exploit'ов, большинство атак совершается не через них (хотя и через них тоже), а "благодаря" грубым ошибкам конфигурации, допущенных администратором.

Это справедливо как для серверов, так и для рабочих станций, однако, сервера имеют свою специфику, в которой доминируют дыры в PHP/Perl-скриптах, SQL-injecting и т. д., о чем уже неоднократно писалась на страницах нашего журнала, так что оставим сервера в покое (о них есть кому позаботиться) и сосредоточимся на рабочих станциях, воздвигаемых на домашних компьютерах неопытными пользователями, обучающихся методом тыка, и совершенно незнакомых с тактикой ведения боя против хакеров.

Рисунок 1 легендарная надежность OpenBSD – всего лишь две удаленных дыры в конфигурации по умолчанию за десять лет промышленной эксплуатации

1. использование одинаковых паролей

Как ни печально, но большинство пользователей, выбрав себе пароль, используют его везде где только возможно: на вход в систему, для доступа к почтовому ящику, при регистрации на различных форумах и других сетевых ресурсах, забывая о том, что во всех этих случаях пароли передаются в открытом виде и могут быть выловлены любым sniffer'ом или путем отправки фальшивого ответа от имени DNS-сервера, перенаправляющего жертву на узел злоумышленника (подробнее об этом рассказывается в пункте 4). Так же не стоит забывать и о том, что хакер может заманить вас на свою страничку, под тем или иным предлогом требующую регистрации (например, для записи в гостевой книге) или предложить вам бесплатный почтовый сервис.

Заполучив пароль, используемый более чем на одном ресурсе, хакер сорвет банк и хорошо еще если тут же не сменит пароли, лишая вас законного доступа к своим аккаунтам. Практически все знают, что постоянно сидеть под root'ом нехорошо, но при этом сплошь и рядом назначают одинаковые пароли как на root'а, так и на простого пользователя.

С другой стороны, удержать в голове целую кучу паролей практически невозможно, особенно, если они не вводятся с клавиатуры каждый раз, а автоматически "подставляются"

программой. Но за это удобство приходится платить и через которое время пароли начисто забывается. Что делать? Как быть? Записывать пароли?! Так ведь это не выход. Если листок со списком паролей спрятать в секретном месте, то при выходе в сеть с чужой машины нам он все равно не поможет, а хранить пароли в записной книжке слишком рискованно. Мир не без любопытствующих товарищей! Никому доверять нельзя! А бумаге — тем более.

Некоторые используют довольно хитрый трюк — слегка видоизменяя пароли или включая в пароль имя ресурса, например, используем в качестве базового пароля: rfn3g1k-h, добавляя к нему: 1nb0x, при регистрации почтового ящика на inbox.ru и 030n, при создании аккаунта в интетнет-магазине www.ozon.ru.

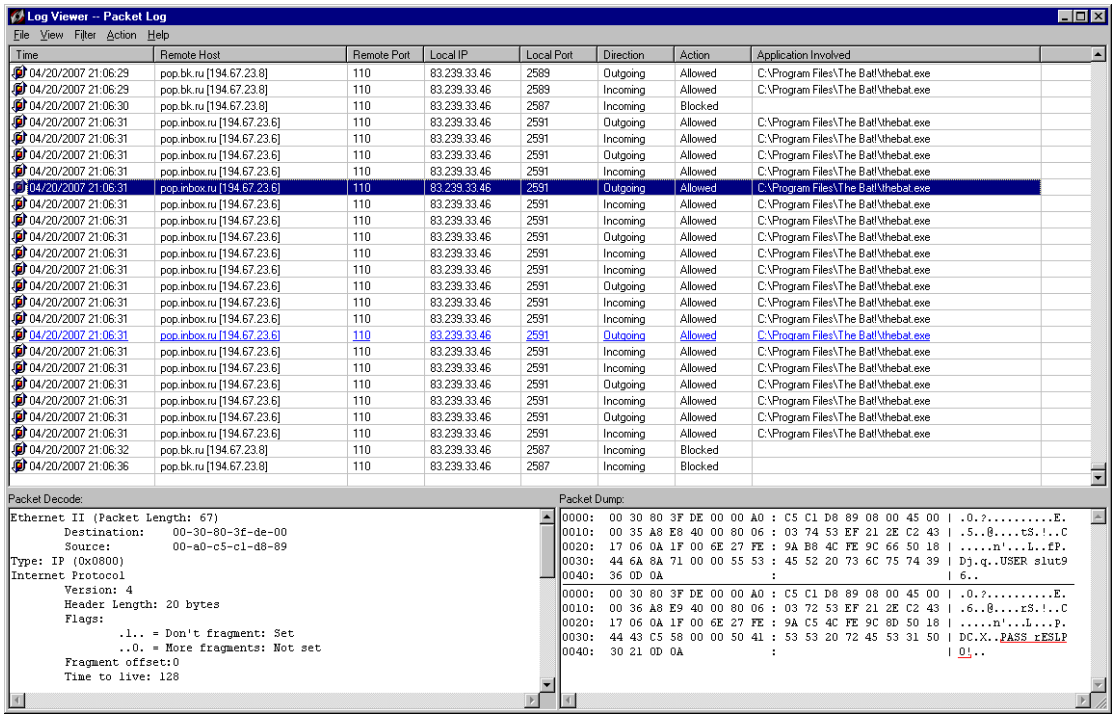

Рисунок 2 пароль "rESLP0!", выловленный sniffer'ом в POP3 сессии, использовался владельцем машины не только для доступа к почтовому ящику, но и во многих других местах

2. установка открытого proxy

Proxy-сервера на рабочих станциях встречаются намного чаще, чем можно подумать. Во-первых, они справляются с кэшированием web-страничек намного лучше, чем браузеры. К тому же при использовании нескольких браузеров (равно, как и браузеров запускаемых из-под разных пользователей) каждый из них ведет свой кэш, что не только нецелесообразно, но и неэкономично!

Локальный proxy позволяет взять кэширование на себя, отключив его в настройках браузеров (кстати говоря, работа Горящего Лиса после этого заметно ускоряется). Во-вторых, proxy решает проблему совместного доступа в Интернет для всех членов семьи и виртуальных машин (типа VM Ware), делая это намного эффективнее, чем NAT. В-третьих, многие устанавливают proxy-сервер просто "на всякий случай", даже не разобравшись что это за штука и нужна ли она им или нет.

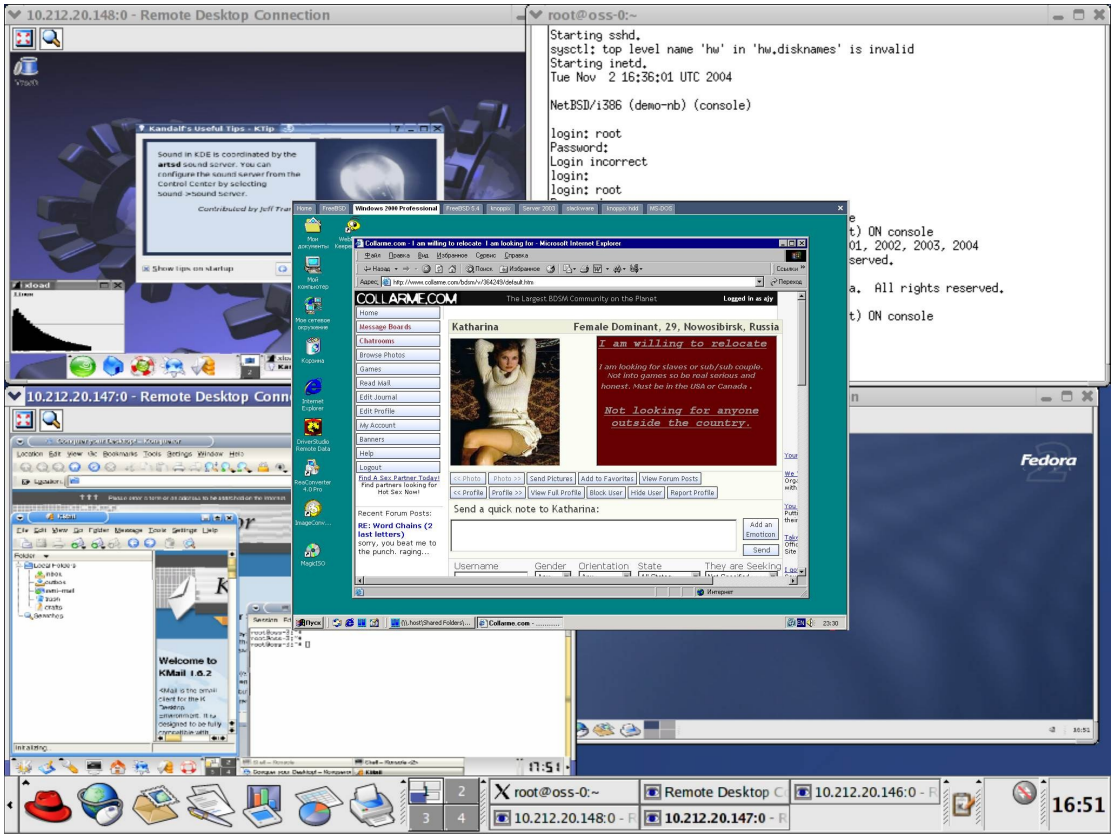

Рисунок 3 proxy, установленный на локальной машине, позволяет блуждать по сети из-под Windows 2000, запущенной на VM Ware

При этом, по умолчанию, proxy-сервер обычно доступен всем желающим и такие желающие находятся достаточно быстро. Какой резон в использовании чужого proxy? Начнем с того, что большинство интернет-провайдеров либо вообще не тарифицирует внутрисетевой трафик, либо продают его значительнее дешевле, чем внешний. Таким образом, отыскав свободный proxy внутри сети провайдера, хакер кидает жертву на бабки. Или, что еще хуже, совершает через него атаку, оставляя в логах чужой IP.

Некоторые переводят proxy на нестандартные порты, надеясь, что "там" хакеры его не найдут. Наивные! Хакеры ищут не вручную, а через сканеры и им, в общем-то совершенно безразлично сколько продлиться сканирование.

Другие предпочитают закрывать доступ на proxy паролем, что в смысле защищенности выглядит блестящим решением, но, к сожалению, далеко не все прикладные программы поддерживают функцию авторизации, причем, даже если сегодня таких программ у нас нет, они могут появится в будущем, так что лучше сразу настраивать proxy на века.

Для отсечения всех "левых" хакеров достаточно использовать привязку к интерфейсам, IP- и/или MAC-адресам. Поясним, что это такое. Каждое коммуникационное устройство трактуется операционной системой как сетевой интерфейс, которому может быть присвоен IP-адрес, после чего оно приобретает возможность посылать и отправлять пакеты во "внешний мир", действуя согласно правилам, прописанным в таблицах маршрутизации. Грубо говоря, если у нас есть сетевая карта, в которую воткнут кабель от свича, обслуживающего домашнюю сеть, мы можем "привязать" proxy-сервер к ее интерфейсу, заблокировав все остальные и пользоваться proxy смогут только члены локальной сети, однако, если в свич воткнут кабель от DSL-модема с Ethernet-портом, proxy-сервер становится "общенародным" достоянием всего Интеннета, что, очевидно, не входит в наши планы.

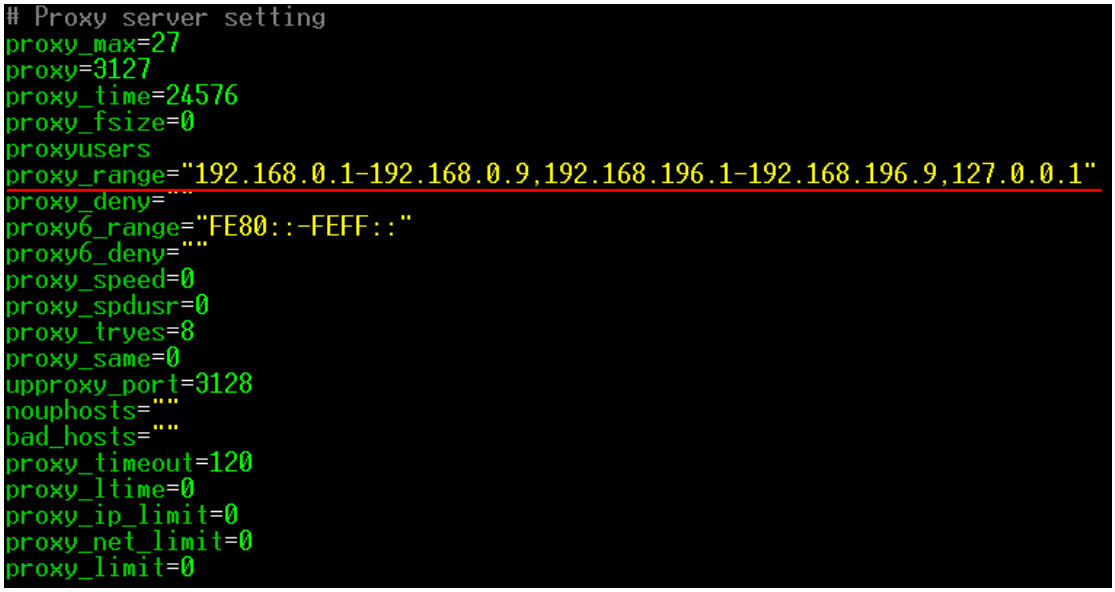

Рисунок 4 привязка proxy-сервера к IP-адресам, с которых разрешен доступ

Привязка к IP-адресам разрешает доступ только с тех узлов, чей IP входит в перечисленный диапазон. Это достаточно надежный способ защиты, и хотя существует ряд атак, позволяющих ее обойти и надежнее привязываться к MAC-адресам узлов своей домашней сети, главное в воздвижении защиты — это не переусердствовать. Если пренебречь угрозой целенаправленной атаки, привязки к IP-адресам вполне достаточно.

Привязка может выполняться как на proxy-сервере, так и на брандмауэре. В чем разница? Закрытие доступа на брандмауэре экономит немного трафика и потому более целесообразно. С другой стороны, поскольку брандмауэр можно обойти, лучше продублировать привязку и на proxy-сервере (если, конечно, он это позволяет).

3. включение поддержки IPv6

Поддержка IPv6 в BSD и Linux появилась не вчера и даже не позавчера, между тем, IPv6 стек все еще остается сырым и подверженным целому спектру атак: от отказа в обслуживании до захвата управления машиной, причем, реально IPv6 никому не нужен. Сегодня с ним можно разве что проиграться, да и то в основном на серверах, а не на рабочих станциях. Пройдет немало лет, прежде чем протокол IPv6 окажется востребованным, но и тогда останется возможность работы через древний IPv4, так что нет никаких оснований держать IPv6 на своей машине, подвергая ее ненужному и совершенно неоправданному риску хакерской атаки.

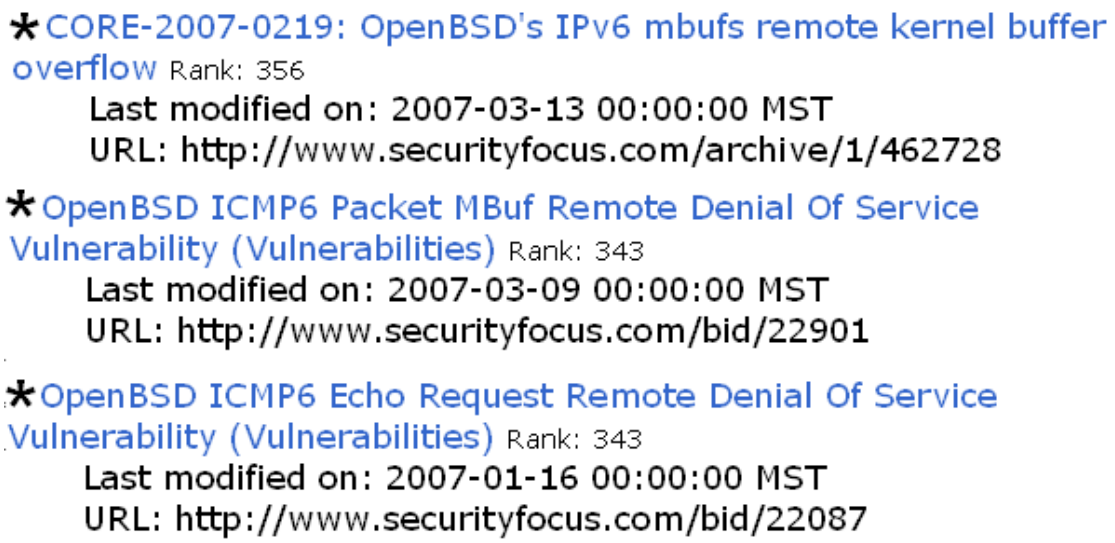

Рисунок 5 примеры дыр, найденных в IPv6 стеке

Выбор протокола IPv6 осуществляется на стадии установки и в дальнейшем отказаться от него без перекомпиляции ядра не так-то просто, однако, существует более простой путь — заблокировать весь IPv6 трафик на брандмауэре, для этого на xBSD-системах необходимо выполнить следующую последовательность действий.

```
# добавить следующую строку в файл /etc/pf.conf:
block in quick inet6 all

# загрузить обновленный pf.conf
# внутрь запущенного PF посредством утилиты pfctl
pfctl -f /etc/pf.conf

# разрешить его использование
pfctl -e -f /etc/pf.conf

# посмотреть текущий статус
# на предмет проверки успешности принятия нового правила
pfctl -s rules
```

Листинг 1 блокирование всего IPv6 трафика на встроенном брандмауэре

4. DNS на UDP

DNS-протокол, по умолчанию работающий на UDP, небезопасен и хакер может запросто перенаправить нас на свой узел, просто отправив подложенный ответ от имени DNS-сервера. Чтобы этого не происходило, необходимо общаться с DNS-сервером только по TCP-протоколу. Это чуть медленнее, зато _намного_ надежнее, поскольку в отличии от UDP, TCP работает с установкой соединения, включающей в себя операцию "рукопожатия", то есть просто так отправить TCP-пакет с поддельным IP в заголовке нельзя, как минимум требуется угадать идентификатор последовательности, чтобы подделать все остальные пакеты.

Проще всего это сделать путем блокировки всего UDP трафика на 53 порте, однако, если DNS-сервер провайдера не поддерживает работу через TCP (как, например, djbdns), то мы не сможем разрешать доменные имена вообще, что очень нехорошо.

А почему бы не установить свой собственный локальный DNS-сервер?! Как показывает практика, DNS-сервера большинства провайдеров тормозят со страшной силой и гораздо выгоднее общаться к корневым DNS-серверам, к тому же это еще и безопаснее. DNS-сервер входит в поставку практически любого дистрибутива (см. "man named") и уже содержит все необходимое, в том числе и адреса корневых DNS-серверов, прописанные в конфигурационных файлов. Все, что нужно — это указать инсталлятору, что мы хотим установить DNS.

Для большей надежности рекомендуется задействовать цифровую подпись, установив параметр auth-nxdomain (как правило, находящийся в /etc/bind/named.conf.options файле в значение "yes".

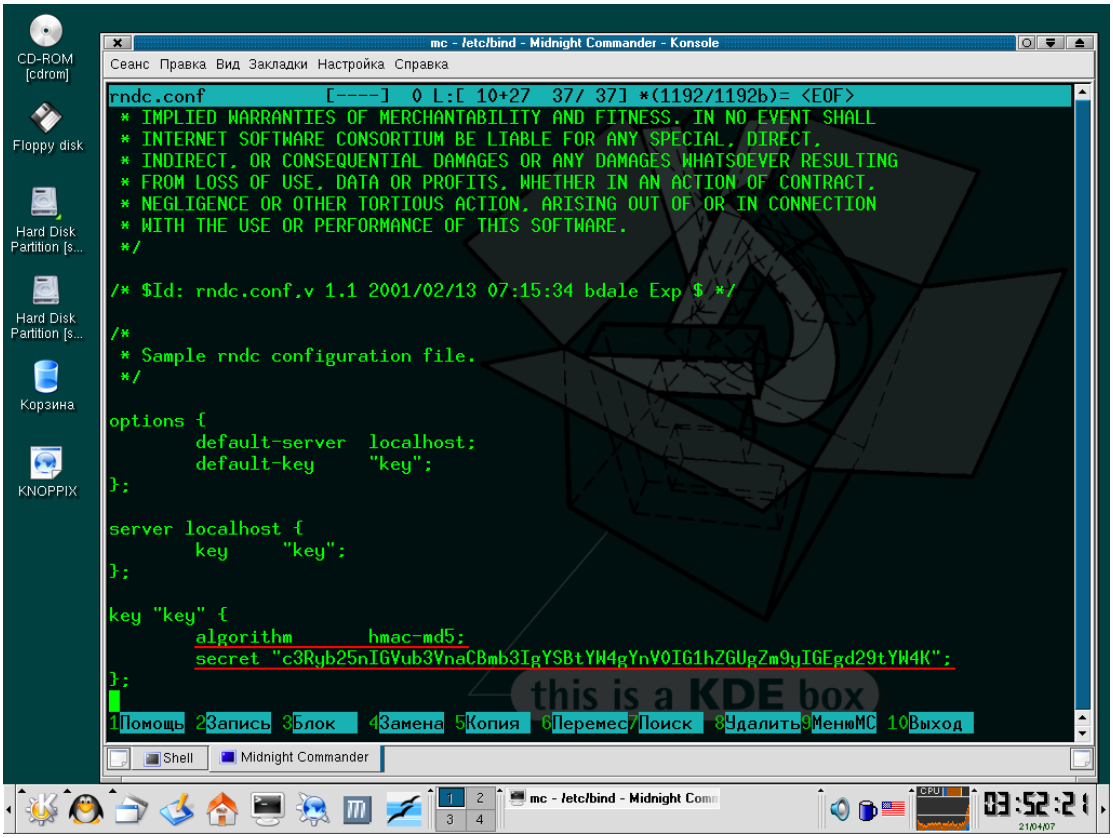

Рисунок 6 использование цифровой подписи предотвращает посылку подложных DNS-ответов хакером

5. запуск подозрительных программ под root'ом

Считается, что вирусов под Linux/BSD не существует. Это неверно. Вирусы есть и там, просто они не получили большого распространения в силу низкой распространенности самого Linux/BSD, а так же того факта, что нормальные люди сидят не под root'ом, а под простым пользователем, не имеющим права модификации уже установленных исполняемых файлов. Тем не менее, если запустить вируса под root'ом, он сможет _такого_ натворить, что потом не разгребешь за всю оставшуюся жизнь.

Причем, это относится не только к запуску, но и… к компиляции! А точнее к сборке исходных текстов утилитой make, обрабатывающей файл makefile, который может включать в себя команды операционной системы, дающие хакеру неограниченную власть над машиной.

Если программа получена из ненадежных источников, то необходимо либо выполнить аудит исходных текстов, либо — запускать ее от имени специального пользователя, не имеющего доступа ни к каким файлам, кроме тех, которые явно необходимы для работы программы. В отличии от Windows, операционные системы семейства UNIX действительно являются многопользовательскими, так что грех не использовать эту возможность.

Кстати говоря, набирая "make install" не забудьте перекреститься. Далеко не все пакеты устанавливаются "гладко" и многие из них потом приходится "выковыривать" из системы, зачастую путем полной переустановки или капитальных работ по восстановлению. Программы, запущенные от имени простого пользователя, могут разрушить только конфигурацию этого самого пользователя, что не является проблемой, поскольку, для установки программ всегда можно создать специального пользователя с конфигурацией по умолчанию и только, убедившись, что программа работает правильно, устанавливать ее от своего имени.

Однако, если программа тянет за собой загружаемые модули ядра или нуждается в изменении системных конфигурационных файлов, ей приходится предоставлять права root'а, идя на сознательный риск. Или же… отправить такую программу в /dev/nul, где ей и место.

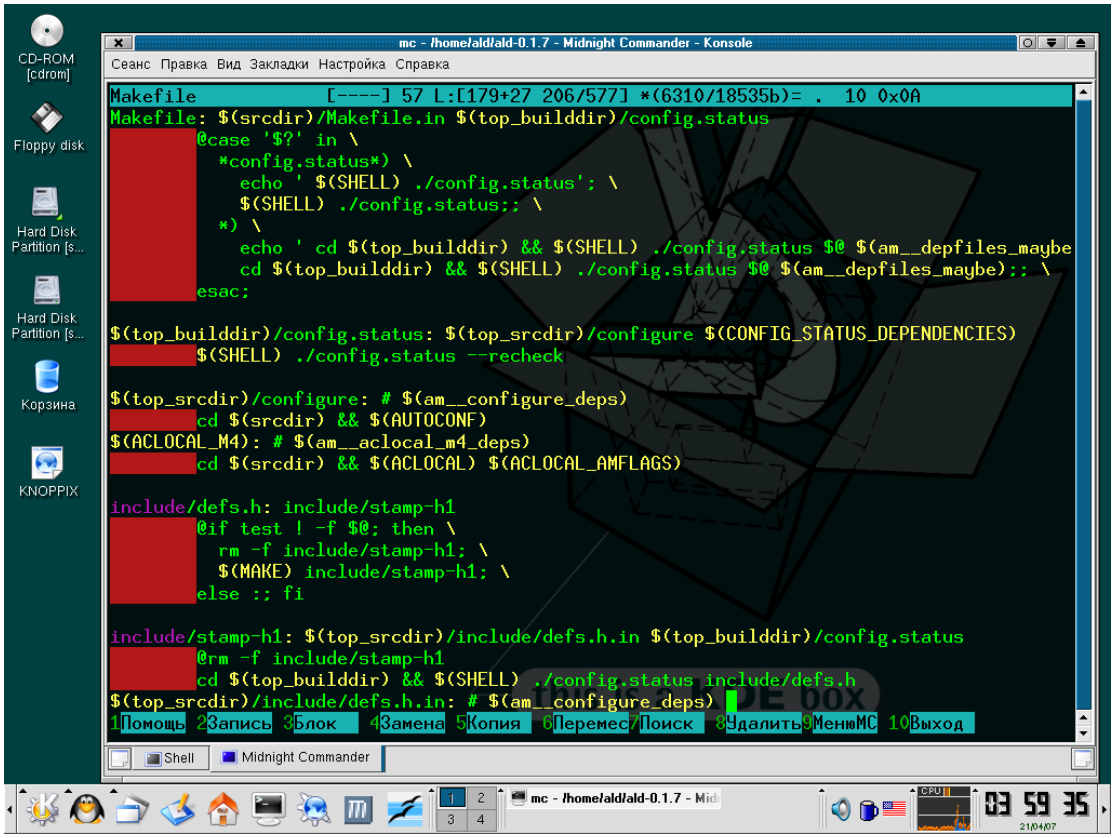

Рисунок 7 команды операционной системы в makefile-файле

6. использование Горящего Лиса

Лис считался надежным браузером лишь до тех пор, пока на него никто не обращал внимания, но теперь… по количеству обнаруженных дыр он вплотную приближается к печально известному IE, и хотя массовых атак на Лиса пока не отмечено, это не значит, что можно и дальше бродить по сети. Беспечная жизнь закончилась. Лис уже отхватил солидную долю рынка, что делает его весьма соблазнительной мишенью в хакерских глазах.

Даже оперативная установка самых свежих заплатка не гарантирует безопасности! К счастью, помимо Лиса есть и другие браузеры, например, Conquer, интегрированный в KDE, а так же текстовой браузер Lynx (входящий в большинство дистрибутивов по умолчанию), которым очень любит пользоваться мыщъх.

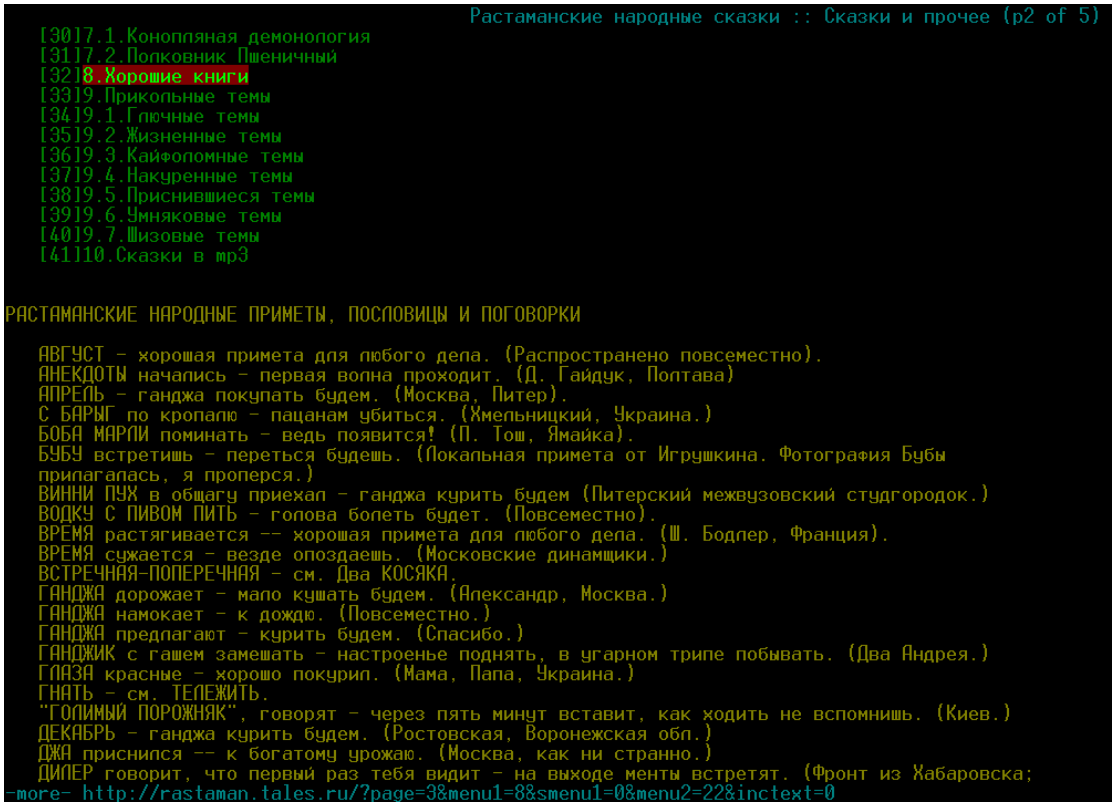

Рисунок 8 Lynx – надежный, быстрый, безопасный текстовой браузер

7. использование готового ядра без перекомпиляции

Большинство exploit'ов, эксплуатирующих дыры в ядре Linux/BSD, содержат в себе жесткого прошитые (hardcoded) адреса машинных команд, меняющиеся от версии к версии и, естественно, при перекомпиляции.

Ядро из коробки довольно предсказуемо и атакующий может без труда установить какой именно байт передаваемых данных затирает адрес возврата и чем его необходимо заменить, чтобы передать управление на shell-код. В большинстве случаев, его заменяют на адрес машинной инструкции jmp esp, а если выполнение в стеке запрещено, то выделяют блок памяти посредством вызова malloc с последующей установкой атрибутов исполнения через функцию mprotect и копирования shell-кода на новое место обитания функцией memcpy. Естественно, адреса всех этих функций атакующий должен знать заранее, иначе у него ничего не получится. Уязвимая программа выбросит исключение, которое будет отловлено ядром и, если программист не предусмотрел специальной обработки критических ситуаций, программа завершиться в аварийном режиме. То есть, дальше банального DOS'a хакер не продвинется.

Атаковать систему с перекомпилированным ядром и всеми стандартными библиотеками — практически нереально и это по плечу только настоящим профессионалом, а не подросткам, научившихся скачивать exploit'ы из сети.

8. не удаленный map-файл

Файл System.map (обычно расположенный в каталоге /boot) включает в себя символьную информацию о глобальных переменных и функциях, экспортируемых ядром, и широко используется rootkit'ами, прячущих от глаз администратора враждебные файлы, процессы и сетевые соединения. И хотя некоторые (между прочим, достаточно многие) rootkit'ы могут находить необходимые им функции и без System.map'а, его удаление существенно уменьшает вероятность атаки.

В "мирных целях", System.map нужен разве что отладчикам, да некоторым низкоуровневым программам. На всякий случай, чтобы потом не перекомпилировать ядро (а system.map создается именно при перекомпиляции ядра), скопируйте в надежное место (например, на внешний носитель) или просто переименуйте во что-то менее напряженное.

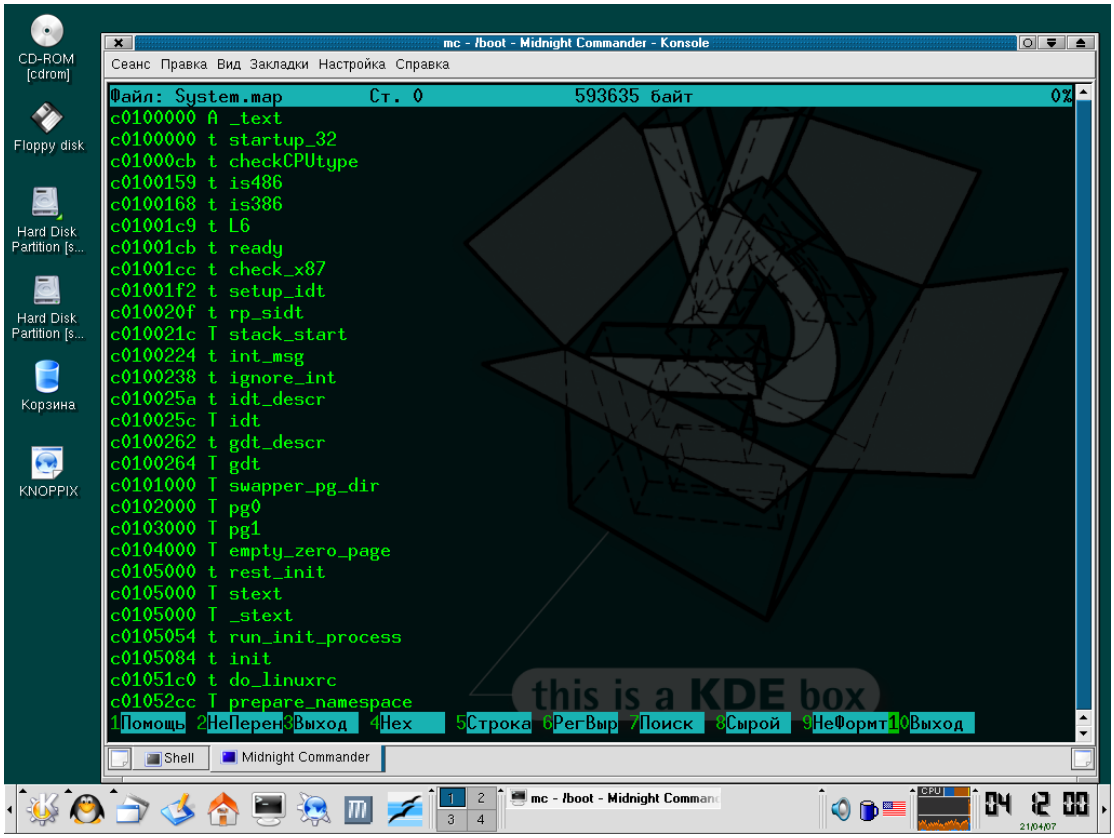

Рисунок 9 фрагмент файла System.map

9. отсутствующие директории в lib

Порядок поиска динамических библиотек задается системной переменной LD_LIBRARY_PATH, значение которой берется из конфигурационного файла /etc/ld.so.config, перечисляющего директории с динамическими библиотеками. В правильно установленной системе, право создания новых файлов в этих директориях имеет только root, что логично. Поскольку, в противном случае любой желающий смог бы добавить свою зловредную библиотеку в вышестоящую директорию, так чтобы она загружалась вместо оригинала (кстати, проверьте свою систему, вдруг она ведет себя не так?!).

Некоторые инсталляторы (например, установщик KNOPPIX'a) прописывают в файле /etc/ld.so.config пути к несуществующим директориям. Казалось бы, ну что тут такого? Мелочь… На самом деле, это _огромная_ дыра в безопасности, поскольку для создания директорий иметь права root'a совершенно необязательно и в них можно размещать библиотеки-спутники, работающие по принципу вирусов-спутников известных еще со времен MS-DOS. Откройте файл /etc/ld.so.config и удалите из него все несуществующие пути, если таковые там присутствуют.

10. игнорирование битов NX/XD

Долгое время x86-процессоры поддерживали только два атрибута защиты на уровне страниц — атрибут доступности и атрибут записи. Атрибут исполнения кода поддерживался только на уровне селекторов и практически все UNIX-подобные системы размещали стек, код, данные и кучу в едином адресном пространстве, доступном для исполнения.

Несмотря на то, что функция mprotect поддерживает четыре атрибута защиты: PROT_NONE, PROT_READ, PROT_WRITE и PROT_EXEC, на аппаратном уровне атрибут PROT_EXEC является синонимом атрибута PROT_READ, то есть, если страницу можно прочитать, с тем же успехом ее можно исполнять. Этой дырой воспользовались хакеры, размещая исполняемый код в стеке и хотя было предложено множество защитных комплексов, размещающих стек в неисполняемой области памяти все они были глючными и ненадежными.

Настоящая революция наступила только с появлением новых атрибутов защиты в каталогах страниц называемых NX (Not eXecutable) и XD (eXecutable Disabled) в процессорах

Intel и AMD соответственно. Последние версии Linux/BSD поддерживают эти биты в том или ином виде, спрашивая пользователя при установке, нужен ли ему неисполняемый стек или нет? Однако, поскольку ряд честных программ (и прежде всего runtime-компиляторов, транслирующих код налету прямо в память) нуждаются в исполняемом стеке, по умолчанию защита выключена во всех системах, кроме OpenBSD.

И хотя, неисполняемые биты отнюдь не панацея (хакеры уже давно научились обходить их), они, тем не менее, чрезвычайно затрудняют атаку, а в сочетании с перекомпилированным ядром и стандартными библиотеками, делают ее практически невозможной, так что смысл в этой защите все-таки есть и лучше держать ее на взводе.