МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ

УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Механизмы безопасности в Linux

Утверждено Редакционно-издательским советом университета

в качестве методических указаний к лабораторной работе

С А М А Р А

Издательство СГАУ

2010

УДК 004.451

Составитель А.М. С у х о в

Рецензент: к.т. н., доц. Попов С.Б.

Механизмы безопасности в Linux: Методические указания к лабораторной работе/ Сост. А.М. Сухов. - Самара: Изд-во Самарского государственного аэрокосмического университета, 2010. 24 с.

В настоящих методических указаниях приведен материал, необходимый для выполнения лабораторных работ по дисциплинам «Операционная система Linux на высокопроизводительных кластерах», «Перспективные информационные технологии»

Целью лабораторной работы является изучение основных возможностей по обеспечению безопасности информационных систем и путей ее достижения в операционной системе Linux.

Предназначено для слушателей ФПК СГАУ и студентов специальностей 010500, 010501

1. Цель лабораторной работы

- изучение основных возможностей по обеспечению безопасности информационных систем
- получение опыта работы с основными утилитами, обеспечивающими различные аспекты безопасности информационные систем в операционной системе Linux
- классификация основных угроз безопасности информационных систем

2. Основные сведения

В данной лабораторной работе будет проведен краткий экскурс в наиболее распространенные средства, связанные с безопасностью Linux. Информация предоставлена в сжатом виде, и если какое-то средство вас заинтересует, можно пройтись по ссылкам и прочитать более подробно. Будут рассмотрены следующие средства: ssh, sudo, firewall (iptables).

sudo: выполнение программ от чужого имени

Описание: Выполнение программ от своего и/или чужого имени.

Механизм работы: При вызове команды sudo/sudoedit система считывает файл /etc/sudoers, и на его основе определяет, какие команды может вызывать пользователь.

Пример использования: Вся конфигурация определяется в файле /etc/sudoers. Например, можно разрешать выполнять только определенные команды и только от определенного пользователя:

WEBMASTERS www = (www) ALL, (root) /usr/bin/su www

Данная строчка говорит о том, что пользователи, определенные в алиасе *WEBMASTERS* могут выполнять все команды от имени пользователя *www*, или делать *su* только в *www*.

Пакет sudo позволяет системному администратору давать права определенным пользователям (или группам) на исполнение конкретных программ с правами другого пользователя (и записывать эти действия в журнал). Возможна привязка списка допустимых команд к имени хоста, что позволяет использовать один файл настройки на нескольких хостах с различными полномочиями. Обычно требует аутентификации пользователя (например, ввода пароля). Позволяет избегать слишком частого ввода пароля (по умолчанию - 5 минут). Для борьбы с подменой динамических библиотек из окружения удаляются переменные типа LD_* и т.п., а также IFS, ENV, BASH_ENV, KRB_CONF, KRB5_CONFIG, LOCALDOMAIN, RES_OPTIONS, HOSTALIASES. Можно также удалять текущую директорию из PATH.

Состоит из файла настройки */etc/sudoers*, программы его редактирования visudo и клиентской программы sudo. В лабораторной работе также описывается процедура установки из исходных текстов и ключи configure.

visudo

visudo позволяет осуществить безопасное редактирование sudoers, она проверяет файл на корректность после выхода из текстового редактора. Ключи:

- -c [-q] (только проверить существующий файл без вызова редактора)
- -f имя-файла (указать другой файл sudoers)
- -s (более строгая проверка)

Описание прав пользователей

В файле /etc/sudoers описываются права пользователей на выполнение команд с помощью sudo. Состоит из операторов трех типов: определение синонимов (Alias), переопределение конфигурационных параметров и описание прав пользователей. Если для одного пользователя подходит несколько описаний, то действует последнее. Комментарии начинаются с символа '#', если это не uid. Продолжение команды на следующей строке обозначается символом '\' в конце строки.

Синонимы (предопределен синоним ALL):

- User_Alias имя = список-пользователей-через-запятую
- Runas_Alias имя = список-пользователей-через-запятую
- Host_Alias имя = список-хостов-через-запятую
- Cmnd_Alias имя = список-команд-через-запятую

Имя может состоять из прописных латинских букв, цифр и подчеркивания. Предопределены синонимы 'ALL' для каждого типа. На одной строке может размещать несколько описаний синонимов, разделяя их символом ':'. netgroup - это NIS.

Пользователь может определяться (восклицательный знак перед именем означает отрицание):

- *имя*
- *%имя-группы*
- *+netgroup*
- *User_Alias*

Эффективный пользователь может определяться (восклицательный знак перед именем означает отрицание):

- *имя*
- *#uid*
- *%имя-группы*
- *+netgroup*
- *Runas_Alias*

Хост может определяться (восклицательный знак перед именем означает отрицание; сетевая маска записывается в точечной записи или CIDR; при указании имени хоста можно использовать шаблоны в стиле shell, но лучше при этом использовать опцию fqdn; не стоит использовать имя localhost):

- *имя-хоста*
- *IP-адрес*
- *network/netmask*
- *+netgroup*
- *Host_Alias*

Команда может определяться (восклицательный знак перед именем означает отрицание; можно использовать шаблоны в стиле shell для имени файла и аргументов; необходимо защищать обратным слешом от интерпретации разборщика символы запятой, двоеточия, равенства и обратного слеша; пустой список аргументов обозначается как ""; полное имя каталога должно оканчиваться на "/", разрешает выполнить любую команду из данного каталога, но не из подкаталога):

- *полное-имя-файла*
- *полное-имя-файла аргументы*
- *каталог*
- *Cmnd_Alias*

Конфигурационные параметры можно переопределить для всех пользователей, для определенных пользователей, для определенных хостов, для команд, выполняемых от имени указанного пользователя:

- Defaults список-параметров-через-запятую
- Defaults:имя-пользователя список-параметров-через-запятую
- Defaults@имя-хоста список-параметров-через-запятую
- Defaults>имя-пользователя список-параметров-через-запятую

Параметр задаётся в виде "имя = значение", "имя += значение", "имя -= значение", "имя" или "!имя" из следующего списка (необходимо использовать символы "" и '\' в значениях при необходимости; += и -= добавляют и удаляют элементы списка):

- флаги
  - authenticate (on; пользователь должен вводить свой пароль)
  - env_editor (off; visudo будет использовать переменную окружения VISUAL или EDITOR; очень не рекомендуется)
  - env_reset (сбросить переменные окружения, кроме HOME, LOGNAME, PATH, SHELL, TERM, USER)
  - fqdn (off; помещать в журнал и sudoers полное доменное имя хоста с использованием запросов DNS вместо простого `hostname`)
  - ignore_dot (off; игнорировать "." в $PATH; не реализовано)
  - log_host (off; записывать имя хоста в файл)
  - log_year (off; записывать в файл год в четырёхзначном формате

- mail_always (off; посылать письмо при каждом выполнении sudo)
- mail_badpass (off; посылать письмо при вводе неправильного пароля)
- mail_no_user (on; посылать письмо, если пользователя нет в sudoers)
- mail_no_host (off; пользователь есть в sudoers, но не для этого хоста)
- mail_no_perms (off; пользователь есть в sudoers, но нет прав на эту команду)
- noexec (off;)
- path_info (off; уточнять причину отказа выполнения команды)
- preserve_groups (off; не менять список групп)
- runaspw (off; запрашивать пароль указанного в runas_default пользователя вместо пароля текущего пользователя)
- requiretty (off; выполнять команду только при запуске с реального tty)
- root_sudo (on; очень рекомендуется запретить использование sudo для root)
- rootpw (off; запрашивать пароль root вместо пароля пользователя)
- shell_noargs (off; разрешать запуск sudo без параметров - выход в shell)
- targetpw (off; запрашивать пароль указанного ключом -u пользователя вместо пароля текущего пользователя)
- tty_tickets (off; отдельный интервал истечения времени действия пароля для каждого терминала)

- числа

- loglinelen (80; длина строк в файле; 0 - бесконечная)
- timestamp_timeout (5 минут; интервал действия пароля)
- umask (022; umask для выполняемой команды)

- строки

- editor (/bin/vi; список редакторов через ":", используемых visudo)
- exempt_group (; пользователи из данной группы не должны вводить пароль)
- lecture (закатывать лекцию при первом вызове: never, once, always; раньше это был флаг)
- lecture_file
- listpw (any; запрашивать ли пароль при использовании ключа "-l": all, any, never, always)
- logfile (/var/log/sudo.log)
- mailerflags (-t)
- mailerpath
- mailsub ("*** SECURITY information for %h ***"; текст темы письма)
- mailto (root; кому посылать письма; рекомендуется заключать в кавычки)
- passprompt ("Password:" (%h расширяется в имя хоста, %u - в имя пользователя)
- runas_default (root)
- syslog (local2; syslog facilities: authpriv, auth, daemon, user, local0-7)
- syslog_badpri (alert; какой приоритет syslog использовать при неудачах)
- syslog_goodpri (notice; какой приоритет syslog использовать при удачном выполнении)

- timestampdir (/var/run/sudo)
- verifypw (all; запрашивать ли пароль при использовании ключа "-v": all, any, never, always)

- список
  - env_check (переменные окружения удаляются, если содержат '%' или '/')
  - env_delete
  - env_keep (список переменных, не сбрасываемых по env_reset)

Описание прав пользователей (списки через запятую, теги и имена целевых пользователей наследуются):

список-пользователей список-хостов = список-описаний-команд [: список-хостов = список-описаний-команд]

описание-команды ::= [**(***список-целевых-пользователей***)**] {*тег*:} *команда*

Теги: NOPASSWD (не запрашивать пароль текущего пользователя), PASSWD, NOEXEC, EXEC.

Команда sudo

Команда sudo выполняет указанную в качестве параметра команду от имени другого пользователя в зависимости от настроек. При отсутствии команды от имени другого пользователя выполняется редактор (-e) или командная оболочка (-s) или имитируется начальный вход в систему (-i). Устанавливается требуемый реальный идентификатор пользователя и группы, euid, egid и список групп. Может запрашиваться пароль текущего пользователя.

Ключи:

- -H (установить HOME как описано в /etc/passwd для целевого пользователя)
- -L (показать список возможных параметров с оисанием)
- -P (не устанавливать список групп)
- -S (читать пароль с stdin вместо tty)
- -V (показать версию sudo; для root выдаются также параметры установки)
- -k (обнулить допустимое время использования sudo без повторного введения пароля)
- -K (аналогично, но более действенным способом)
- -b (выполнить команду в фоновом режиме)
- -e имя-файла (редактировать указанный файл, в настройках должно быть разрешено выполнять команду sudoedit)
- -h (вывести описание команды)
- -i (имитируется начальный вход в систему)
- -l (показать список разрешенных команд)
- -p формат (задаёт текст приглашения при вводе пароля)
- -s (запустить командную оболочку)

- -u имя-пользователя (с правами какого пользователя исполнить команду; по умолчанию - root)
- -u #uid
- -v (продлевает допустимое время использования sudo без повторного введения пароля на очередные 5 минут)
- -- (не интерпретировать остальные параметры)

По умолчанию, простая аккредитация пользователя означает, что при попытке исполнить команду от имени другого ему будет предложено ввести свой пароль - свой собственный, обратите внимание, а не пароль того, от чьего имени он собирается действовать. Вдобавок, "эффект памяти" после первого ввода пароля составляет по тому же умолчанию всего пять минут, и если команду sudo вновь отдать по истечении этого срока, она снова затребует пароль. Однако изменением настроек в файле /etc/sudoers можно преодолеть эти ограничения, уместные для больших многопользовательских систем, но никак не для домашних компьютеров.

Кстати, если sudo попытается воспользоваться неаккредитованный пользователь, сообщение об этом в виде электронного письма незамедлительно отправится на локальный адрес root, и если даже машина не подключена к Интернету, суперпользователь получит это письмо. Кроме того, запись о нарушителе появится в системном журнале.

Чтобы определить, какими полномочиями наделён тот или иной пользователь для исполнения команды sudo, достаточно от его имени сделать запрос sudo -l

```
[root@nefertem root]#
[root@nefertem root]# sudo -l
User root may run the following commands on this host:
    (ALL) ALL
[root@nefertem root]#
[root@nefertem root]#
```

Изначально только root имеет право вершить при помощи sudo всё, что угодно, - но суперпользователю эта утилита ни к чему, он и так способен на всё. Владельцам же прочих учётных записей на данном компьютере следует позаботиться о своей аккредитации. Заведение в */etc/sudoers* новых правил следует производить - внимание! - не при помощи привычного вам редактора, а посредством особой утилиты *visudo*. Её аналоги, кстати, имеются и для редактирования файлов паролей */etc/passwd* и групп */etc/group* - *vipw* и *vigr*, соответственно. Главная их особенность - в том, что перед запуском самого редактора они блокируют (lock) редактируемый файл, так что ни один другой пользователь не сможет в то же самое время его править. Vi ведь оперирует не с самим файлом, а с его копией в памяти, как мы помним, и до самого момента явной записи информации на диск в исходный файл не вносятся изменения. Значит, возможна ситуация, когда один и тот же файл открывают на редактирование два пользователя (в случае */etc/sudo*, например, им обоим должнен быть известен пароль *root*). Тогда в итоге файл останется таким, каким сохранит его на диск последний из завершивших работу пользователей. Если же файл блокирован, открыть его кому-то ещё в то же самое время будет непросто.

Утилита sudo - мощный и гибкий инструмент, и с его помощью системный администратор может существенно облегчить свою жизнь. Скажем, можно аккредитовать в */etc/sudoers* своего заместителя для выполнения некоторых рутинных обязанностей и не терзаться мыслью о том, что пароль root'а знает кто-то ещё: теперь это ни к чему. Однако вряд ли для домашнего компьютера потребуется вся потаённая мощь *sudo* - всё-таки уровень подозрительности к окружающим, если это члены семьи, даже у самого параноидального сисадмина должен быть понижен. Так что наиболее распространённая опция в sudo "для дома, для семьи" - это разрешение на запуск той или иной полезной утилиты от имени непривилегированного пользователя без дополнительного введения пароля.

Аккредитация в этом случае выглядит чрезвычайно просто (если вас интересуют более глубокие подробности, к вашим услугам man sudo и man 5 sudoers). Запустите команду visudo, и в самой последней строке открывшегося файла допишите строчку:

[имя пользователя] localhost.localdomail = NOPASSWD: [полный путь исполнения]

Все множество средств обеспечения безопасности можно разделить на следующие группы или категории:

- Средства управления доступом к системе (доступ с консоли, доступ по сети) и разграничения доступа
- Обеспечение контроля целостности и неизменности программного обеспечения (сюда же я отношу средства антивирусной защиты, поскольку внедрение вируса есть изменение ПО)
- Средства криптографической защиты
- Средства защиты от вторжения извне (внешнего воздействия)
- Средства протоколирования действий пользователей, которые тоже служат обеспечению безопасности (хотя и не только)
- Средства обнаружения вторжений
- Средства контроля состояния безопасности системы (обнаружения уязвимостей)

Как перезагрузить компьютер, находясь за пару тысяч километров от него или выполнить другие действия удаленно и безопасно? Ответ на эти вопросы - протокол SSH.

SSH (Secure Shell) - это сетевой протокол, используемый для удаленного управления компьютером и для передачи файлов. SSH похож по функциональности на протоколы telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли.

SSH позволяет передавать через безопасный канал любой другой сетевой протокол, таким образом, можно не только удаленно работать на компьютере, но и передавать по шифрованному каналу звуковой поток или видео (например, с веб-камеры). Также, SSH может использовать сжатие передаваемых данных для последующей их шифрации.

Большинство хостинг-провайдеров за определенную плату предоставляют клиентам доступ к их домашнему каталогу по SSH. Это очень удобно как для работы в командной строке, так и для удаленного запуска графических приложений. Через SSH можно работать и в локальной сети. Если вам лень ходить к серверам - администрируйте их удаленно, используя любой SSH-клиент.

SSH-клиенты и SSH-сервера написаны под большинство платформ: Windows, Linux, Mac OS X, Java, Solaris, Symbian OS, Windows Mobile и даже Palm OS.

Первая версия протокола, SSH-1, была разработана в 1995 году исследователем Tatu Yl'nen из Технологического университета Хельсинки, Финляндия. SSH-1 был написан для обеспечения большей конфиденциальности, чем протоколы rlogin, telnet и rsh. В 1996 году была разработана более безопасная версия протокола, SSH-2, уже несовместимая с SSH-1. Протокол приобрел еще большую популярность, и к 2000 году его использовало уже порядка двух миллионов пользователей. В 2006 году протокол был утвержден рабочей группой IETF в качестве Интернет- стандарта.

Однако, до сих пор в некоторых странах (Франция, Россия, Ирак и Пакистан) требуется специальное разрешение в соответствующих структурах для использования определенных методов шифрования, включая SSH. См. закон Российской Федерации "О федеральных органах правительственной связи и информации".

Распространены две реализации SSH: коммерческая (закрытая) и свободная (открытая). Свободная реализация называется OpenSSH. К 2006 году 80% компьютеров Интернет использовало именно OpenSSH. Коммерческая реализация разрабатывается организацией SSH Inc., http://ssh.com/ - закрытая реализация, бесплатная для некоммерческого использования. Свободная и коммерческая реализации SSH содержат практически одинаковый набор команд.

Существуют две версии протокола SSH: SSH-1 и SSH-2. В первой версии протокола есть существенные недостатки, поэтому в настоящее время SSH-1 практически нигде не применяется.

Многие взломщики сканируют сеть в поиске открытого порта SSH, особенно - адреса хостинг-провайдеров. Так что, в целях безопасности - запрещайте доступ по ssh для суперпользователя. Обычно злоумышленники подбирают именно пароль суперпользователя. См. ниже рекомендации по безопасности использования SSH.

Протокол SSH-2 устойчив в атакам "man-in-middle", в отличие от протокола telnet. То есть, прослушивание трафика, "снифинг", ничего не дает злоумышленнику. Протокол SSH-2 также устойчив к атакам путем присоединения посредине (session hijacking) и обманом сервера имен (DNS spoffing).

Далее по тексту, мы будем иметь ввиду под SSH вторую версию протокола, SSH-2.

SSH-сервера

- Debian GNU/Linux: dropbear, lsh-server, openssh-server, ssh
- MS Windows: freeSSHd, OpenSSH sshd, WinSSHD, ProSHHD, Dropbear SSH Server

SSH-клиенты и оболочки

- Debian GNU/Linux: kdessh, lsh-client, openssh-client, putty, ssh
- MS Windows: PuTTY, SecureCRT, ShellGuard, Axessh, ZOC, SSHWindows, ProSSHD
- Mac OS: NiftyTelnet SSH
- Symbian OS: PuTTY
- Java: MindTerm, AppGate Security Server

Для работы по SSH нужен SSH-сервер и SSH-клиент. Сервер прослушивает соединения от клиентских машин и при установлении связи производит аутентификацию, после чего начинает обслуживание клиента. Клиент используется для входа на удаленную машину и выполнения команд.

Предположим, что сервер у нас настроен и работает. Для подключения клиента требуется сгенерировать пару из открытого и закрытого ключей. Если Вы используете PuTTY под Windows - это делается утилитой puttygen.exe.

Под Linux обычно используется команда puttygen (для PuTTY) или ssh-keygen (для OpenSSH). Далее указываем клиенту - где находится закрытый ключ, и соединяемся с вводом пароля. Возможно также беспарольное соединение, а чем упомянуто ниже.

Рекомендации по безопасности использования SSH:

1. Запрещение удаленного root-доступа.
2. Запрещение подключения с пустым паролем или отключение входа по паролю.
3. Выбор нестандартного порта для SSH-сервера.
4. Использование длинных SSH2 RSA-ключей (2048 бит и более). По состоянию на 2006 год система шифрования на основе RSA считалась надёжной, если длина ключа не менее 1024 бит.
5. Ограничение списка IP-адресов, с которых разрешен доступ. Например, настройкой файрвола.
6. Запрещение доступа с некоторых, потенциально опасных адресов.
7. Отказ от использования распространенных или широко известных системных логинов для доступа по SSH.
8. Регулярный просмотр сообщений об ошибках аутентификации.
9. Установка детекторов атак (IDS, Intrusion Detection System).
10. Использование ловушек, подделывающих SSH-сервис (honeypots)

Как подключиться к удаленному серверу из Linux или FreeBSD? Команда подключения к локальному SSH-серверу из командной строки для пользователя pacify (сервер слушает нестандартный порт 30000):

$ ssh -p30000 pacify@127.0.0.1

Под Windows можно воспользоваться программой PuTTY. Она имеет простой графический интерфейс, через который удобно настраивать подключения. На вкладке SSH\Auth надо выбрать закрытый ключ, который будет использоваться PuTTY.

К удаленному компьютеру можно подключиться по SSH, не вводя пароль, а используя открытый ключ. Разумеется, открытый ключ лучше передавать на сервер по защищенному каналу.

Генерация пары SSH-2 RSA-ключей длиной 4096 бита программой puttygen под Linux/UNIX надо выполнить команду:

$ puttygen -t rsa -b 4096 -o sample

Под Windows у этой программы (puttygen.exe) есть пользовательский интерфейс.

Установка SSH в Linux на примере Debian

Итак, всё, что нам нужно для установки полного комплекта удалённого управления компьютером (**SSH-клиент** и **SSH-сервер**) давным-давно лежит в репозитории. Лёгким движением ставим пакет:

 # apt-get install ssh

и ждём несколько мгновений, когда оно настроится. После этого мы получим возможность **SSH** доступа в систему и управления ей. Так как технология эта кросс-платформенная, то можно управлять по SSH Linux или FreeBSD можно и из Windows. Для этого есть **putty,** SSH Windows клиент.

На стороне клиента теперь надо поправить настройки, которые лежат в каталоге /etc/ssh - конфиг для клиента называется ssh-config, конфиг для сервера, соответственно, sshd-config. На своей, клиентской, стороне, настраиваем возможность приёма X11Forward, ищем и меняем ключи на:

ForwardX11 yes
ForwardX11Trusted yes

Клиентская машина теперь может запускать удалённо графические приложения на сервере. Настройка **SSH** на стороне клиента закончена, теперь идём к админу далёкого сервера. В принципе, можно на клиентской стороне ничего не менять, а логиниться на удалённую машину так:

*$ ssh –**X** user@server1.mydomain.com*

Или

*$ ssh –**X** user@192.168.x.x*

На стороне сервера пеперь нужно настроить SSH сервер: в конфигах машины-сервера, к которой будем подсоединяться (у вас ведь есть её рутовый пароль, так ведь?) в /etc/ssh/sshd-config ищем и меняем ключи на:

X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes

Этим мы разрешаем серверу запускать удалённо графические приложения и отправлять их на клиентскую машину. Перестартуем сервис:

$sudo /etc/init.d/ssh restart

Теперь мы сможем заходить на машину не только в консольном режиме, но и с запуском иксовых приложений. Если хочется разрешить вход только с определённых машин, нужно подправить строки в конфиге */etc/ssh/sshd_config*

AllowUsers hacker@*

AllowUsers *@192.168.1.*

SSH в действии

Всё готово, и теперь будут приведены несколько команд SSH для примера. Открываем консольку и пишем в ней:

$ ssh имя_пользователя_удалённой_машины@ip_адрес_или_сетевое_имя_удалённой_машины

После этого нас могут спросить: данный айпишник ещё не опознан, как доверительный, стоит ему доверять? Пишем yes, стоит, конечно! Далее вводим пароль пользователя удалённой машины, под которым мы заходим и, если он правильный, попадаем в консоль ***удалённой*** машины. В процессе набора пароля вы его не увидите - набирайте всё равно; даётся три попытки - потом соединение обрывается.

Итак, нас поприветствуют как-то вроде этого:

penta4@penta4rce:~$ ssh beast@192.168.1.5
Password:
Last login: Tue Oct 10 19:23:57 2006 from 192.168.1.1
beast@notebeast:~$

Теперь, в окошке терминала, который на нашей машине, мы рулим компьютером, к которому мы подключились. Не перепутайте терминалы, а то вырубите не тот компьютер. Здесь всё просто и логично, но нам бы хотелось ещё и запускать графические приложения на удалённой системе.

Запуск графических приложений удалённо. Вводим, как обычно, логин и пароль ***удалённой*** машины. И вот перед нами та же самая консоль. Как вызвать графическое приложение? Просто наберите имя вызываемой программы и поставьте после имени знак амперсанд:

$ gimp &;

Это запустит на ***удалённо*** машине GiMP в фоне и вернёт вам консоль для дальнейших действий. Если вы не поставите амперсанд после имени приложения, то управление в консоль будет возвращено только после завершения приложения.

Принцип работы файрвол

Брандмауэр (файрвол) предназначен для фильтрации и обработки пакетов, проходящих через сеть. Когда пакет прибывает, брандмауэр анализирует заголовки пакета и принимает решение, “выбросить” пакет (DROP), принять пакет (ACCEPT, пакет может пройти дальше) или сделать с ним что-то еще более сложное.

В Linux брандмауэр является модулем ядра, его неотъемлемой частью. С его помощью мы можем делать с пакетами множество хитроумных вещей, но основной принцип манипуляции трафиком сохраняется: просматриваются заголовки пакетов и решается их дальнейшая судьба. Интерфейсом для модификации правил, по которым брандмауэр обрабатывает пакеты, служит iptables.

Итак, пребывающий пакет проходит по цепочке правил. Каждое правило содержит *условие* и *цель* (действие). Если пакет *удовлетворяет условию* то он *передается на цель*, в противном случае к пакету применяется следующее правило в цепочке. Если пакет не удовлетворил ни одному из условий в цепочке, то к нему применяется *действие по умолчанию.*

Например, к нам пришел пакет от адреса 192.168.164.84 для 128.3.4.5, применим цепочку из трех правил к этому пакету:

номер правила	**условие**	**действие**	**результат**
1	пакет от адреса 127.2.4.5	пропустить (ACCEPT)	переход к следующему правилу
2	пакет для адреса 234.2.*.5	пропустить (ACCEPT)	переход к следующему правилу
3	пакет от адреса 192.168.*	выбросить пакет (DROP)	пакет выброшен

Как видно, пакет не удовлетворил первым двум условиям цепочки, зато удовлетворил третьему правилу, которое заставило брандмауэр выбросить этот пакет.

Цепочки собраны в три основные таблицы (при желании можно добавить свои):

- **filter** - Таблица, используемая по-умолчанию,
- **nat** - Эта таблица используется, когда встречается пакет, устанавливающий новое соединение.

- **mangle** - Эта таблица используется для специальных изменений пакетов.

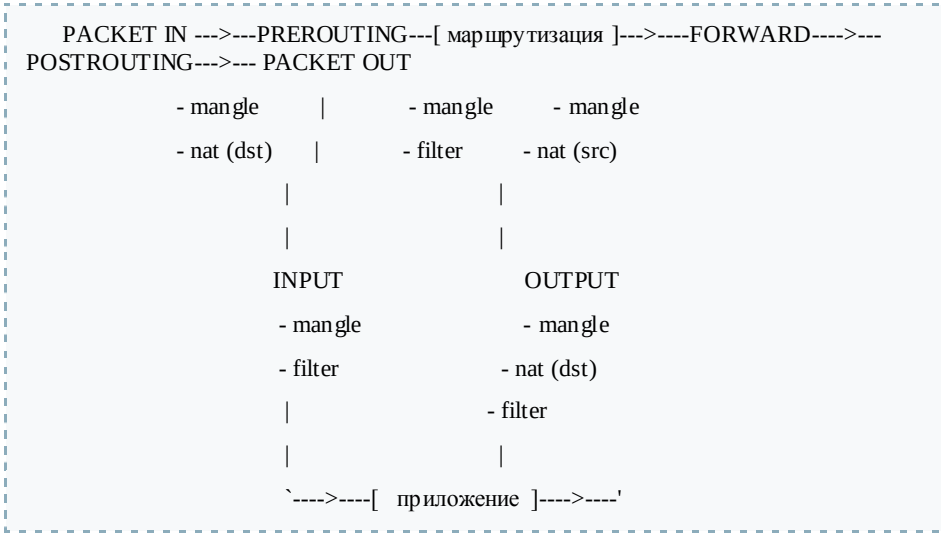
Основными цепочками являются следующие:

- **FORWARD**: для проходящих (пересылаемых) пакетов
- **INPUT**: для входящих/получаемых пакетов
- **OUTPUT**: для исходящих/отправляемых пакетов

Действие может быть именем цепочки, определенной пользователем, или одной из специальных целей: ACCEPT, DROP, QUEUE, RETURN или MASQUERADE.

ACCEPT означает принять пакет. DROP означает проигнорировать (выбросить) пакет. MASQUERADE означает скрыть (маскировать) IP. (ниже будет рассмотрено подробно)

Схематично обработку пакета можно изобразить следующим образом:



Входящий пакет начинает обрабатываться брандмауэром с цепочки PREROUTING в таблице **mangle**. Затем он обрабатывается правилами цепочки PREROUTING таблицы **nat**. На этом этапе проверяется, не требуется ли модификация назначения пакета (DNAT). Важно сменить назначение сейчас, потому что маршрут пакета определяется сразу после того, как он покинет цепочку PREROUTING. После этого он будет отправлен на цепочку INPUT (если целью пакета является этот компьютер) или FORWARD (если его целью является другой компьютер в сети).

Если целью пакета является другой компьютер, то пакет фильтруется правилами цепочки FORWARD таблиц **mangle** и **filter**, а затем к нему применяются правила цепочки POSTROUTING. На данном этапе можно использовать

SNAT/MASQUARADE (подмена источника/маскировка). После этих действий пакет (если выжил) будет отправлен в сеть

Если назначением пакета является сам компьютер с брандмауэром, то, после маршрутизации, он обрабатывается правилами цепочек INPUT таблиц **mangle** и **filter.** В случае прохождения цепочек пакет передается приложению.

Когда приложение, на машине с брандмауэром, отвечает на запрос или отправляет собственный пакет, то он обрабатывается цепочкой OUTPUT таблицы **filter**. Затем к нему применяются правила цепочки OUTPUT таблицы **nat**, для определения, требуется-ли использовать DNAT (модификация назначения), пакет фильтруется цепочкой OUTPUT таблицы **filter** и выпускается в цепочку POSTROUTING которая может использовать SNAT и QoS. В случае успешного прохождения POSTROUTING пакет выходит в сеть.

Для добавления правила в цепочку используется ключ **-A**

```
#iptables -A INPUT правило
```

добавит правило в цепочку INPUT таблицы **filter** (по умолчанию). Для указания таблицы, в цепочку которой следует добавить правило, используйте ключ **-t**:

```
#iptables -t nat -A INPUT правило
```

добавит правило в цепочку INPUT таблицы **nat**.

Цель по умолчанию задается с помощью ключа **-P**:

```
#iptables -P INPUT DROP
```

Условия для отбора пакетов

Теперь мы знаем как пакеты проходят сквозь различные таблицы и цепочки iptables. Пришло время разъяснить принципы, по которым строятся условия накладываемые на пакеты:

Немного о протоколе TCP/IP

TCP/IP является протоколом, в котором соединение устанавливается в 3 фазы. Если компьютер А пытается установить соединение с компьютером Б они обмениваются специальными TCP пакетами.

После чего соединение считается **установленным** (ESTABLISHED).

iptables различает эти состояния как NEW и ESTABLISHED.

Опции отбора пакетов

опция	описание	пример
—protocol (сокращено -	Определяет протокол. Опции tcp, udp, icmp, или любой другой протокол определенный	iptables -A INPUT —protocol tcp

p)	в /etc/protocols	
—source	IP адрес источника пакетаs. Может быть определен несколькими путями. • Одиночный хост: host.domain.tld, или IP адрес: 10.10.10.3 • Пул-адресов (подсеть): 10.10.10.3/24 или 10.10.10.3/255.255.255.0	iptables -A INPUT —source 10.10.10.3
—destination	IP адрес назначения пакета. Может быть определен несколькими путями - смотри —source	iptables -A INPUT —destination 192.168.1.0/24
—source-port	Порт источник, возможно только для протоколов —protocol tcp, или —protocol udp	iptables -A INPUT —protocol tcp —source-port 25
—destination-port	Порт назначения, возможно только для протоколов —protocol tcp, или —protocol udp	iptables -A INPUT —protocol udp —destination-port 67
—state	Состояние соединения. Доступно, если модуль ‘state’ загружен с помощью ‘-m state’. Доступные опции: **NEW** Все пакеты устанавливающие новое соединение **ESTABLISHED** Все пакеты принадлежащие установленному соединению **RELATED** Пакеты, не принадлежащие установленному соединению, но связанные с ним. Например - FTP в активном режиме использует разные соединения для передачи данных. Эти соединения связанны. **INVALID**	iptables -A INPUT -m state —state NEW,ESTABLISHED

	Пакеты, которые не могут быть по тем или иным причинам идентифицированны. Например ICMP ошибки не принадлежащие существующим соединениям	
—in-interface (сокращенно -i)	Определяет интерфейс, на который прибыл пакет. Полезно для NAT и машин с несколькими сетевыми интерфейсами	iptables -t nat -A PREROUTING —in-interface eth0
—out-interface (сокращенно -o)	Определяет интерфейс, с которого уйдет пакет. Полезно для NAT и машин с несколькими сетевыми интерфейсами	iptables -t nat -A POSTROUTING —in-interface eth1
—tcp-flags	Определяет TCP флаги пакета. Содержит 2 параметра: Список флагов которые следует проверить и список флагов которые должны быть установлены	
—syn	Сокращение для '—tcp-flags SYN,RST,ACK SYN'. Поскольку проверяет TCP флаги, используется с —protocol tcp. В примере показан фильтр для пакетов с флагом NEW, но без флага SYN. Обычно такие пакеты должны быть выброшены (DROP).	iptables -A INPUT —protocol tcp ! —syn -m state —state NEW

Определение цели

Чтобы определить действие, которое брандмауэр выполнит, если пакет в одной из цепочек удовлетворяет **условию**, iptables использует цели, устанавливаемые с помощью ключа —jump (или просто -j).

Целью может быть любая другая цепочка, куда будет передан пакет, или одно из следующих действий:

- **ACCEPT** Пропускает удовлетворяющий условию пакет (пакет покидает данную цепочку и передается дальше).

- **DROP** Выбрасывает удовлетворяющий условию пакет (молча, как будто он и не приходил).

- **REJECT** Отклоняет пакет, сообщая об этом передавшему. Имеет дополнительный параметр '—reject-with' определяющий сообщение которое будет передано отправившему пакет, это может быть: ***icmp-net-unreachable***, ***icmp-host-***

unreachable, icmp-port-unreachable (по умолчанию), ***icmp-proto-unreachable, icmp-net-prohibitedor*** и ***icmp-host-prohibited***. Сообщение (по умолчанию “порт недоступен”) возвращается отправившему пакет компьютеру.

- **LOG** Заносит в журнал информацию о пакете. Полезно в комбинации с DROP и REJECT для отладки.

- **RETURN** Возвращает пакет в ту цепочку, из которой он прибыл.

- **SNAT** Применяет source NAT ко всем удовлетворяющим условию пакетам. Может использоваться только в цепочках POSTROUTING и OUTPUT таблицы **nat.**

- **DNAT** Применяет destination NAT ко всем удовлетворяющим условию пакетам. Может использоваться только в цепочке POSTROUTING таблицы **nat.**

- **MASQUERADE** Может применяться только в цепочке POSTROUTING таблицы **nat**. Используется при наличии соединения с динамическим IP. Похож на SNAT, однако имеет свойство забывать про все активные соединения при потере интерфейса. Это полезно при наличии соединения, на котором время от времени меняется IP-адрес, но при наличии постоянного IP это только доставит неудобства. В том числе поэтому рекомендуется для статических IP использовать SNAT.

Примеры

Настроим простейший шлюз для раздачи интернета для дома или малого офиса (SOHO). Допустим, что внутрь сети смотрит интерфейс **eth1** с ip адресом 192.168.1.1, а в интернет интерфейс **eth0**:

Разрешим пропуск трафика через шлюз (в противном случае трафик не проходит цепочку FORWARDING):

```
#sysctl -w net.ipv4.ip_forward="1"
```

Добавим правило, для маскировки ip в цепочку POSTROUTING таблицы **nat**

```
#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Теперь клиенты внутренней сети могут получить возможность доступа в интернет установив в качестве шлюза адрес 192.168.1.1 .

Теперь попробуем нечто более интересное: запретим возможность установки новых соединений с маршрутизатором из интернета, а также форвардинг из сетей отличных от 192.168.1.0/24 .

Установим политики по умолчанию для цепочек в DROP, запретив все соединения кроме тех, которые будут разрешены:

```
#iptables -P INPUT DROP
#iptables -P FORWARD DROP
#iptables -P OUTPUT DROP
```

Разрешим входящие соединения на маршрутизатор с внутренней сети (для управления)

#iptables -A INPUT -i eth1 --source 192.168.1.0/24 --match state --state NEW,ESTABLISHED -j ACCEPT

Разрешим маршрутизатору отвечать компьютерам во внутренней сети:

#iptables -A OUTPUT -o eth1 --destination 192.168.1.0/24 --match state --state NEW,ESTABLISHED -j ACCEPT

Разрешим перенаправление пакетов из внутренней сети во внешнюю для установки соединений и установленных соединений:

#iptables -A FORWARD -i eth1 --source 192.168.1.0/24 --destination 0.0.0.0/0 --match state --state NEW,ESTABLISHED -j ACCEPT

Разрешим перенаправление пакетов из интернета во внутреннюю сеть только для установленных соединений:

#iptables -A FORWARD -i eth0 --destination 192.168.1.0/24 --match state --state ESTABLISHED -j ACCEPT

Итого:

- Маршрутизатор разрешает клиентам внутренней сети устанавливать соединение с узлами в интернет.
- Маршрутизатор имеет возможность удаленного управления из внутренней сети и только из внутренней сети.
- Маршрутизатор блокирует все попытки установить новое соединение **из** сети интернет.
- Маршрутизатор не принимает пакеты из сети интернет сам, только *перенаправляет* пакеты во внутреннюю сеть и только принадлежащие *установленным* соединениям.

Список контрольных вопросов

1. При помощи каких основных утилит обеспечивается безопасность в операционной системе Linux?
2. При помощи какой утилиты производится выполнение программ от своего и/или чужого имени?
3. Какой файл отвечает за настройки прав для утилиты ***sudo***?
4. Каковы основные отличия удаленного доступа ***telnet*** от сетевого протокола ***ssh***?
5. Назовите основные программные пакеты, реализующие ***ssh*** в различных операционных системах?
6. Какая утилита реализует механизм генерации открытого ключа ***ssh***?
7. Каковы основные принципы работы файрвол?
8. Приведите примеры фильтрации пакетов при помощи ***iptables***

Задания по лабораторной работе

1. Проверьте, установлена ли на Вашем рабочем месте утилита ***sudo*** и, при необходимости, инсталлируйте ее.
2. Отредактируйте файл с конфигурацией sudo, довив своего пользователя, и запустите команду *ifconfig* от своего имени.
3. Проверьте, какими правами наделен Ваш пользовтель?
4. Установите серверный пакет Open SSH и настройте его
5. Попробуйте зайти по протоколу ssh на рабочее место соседнего компьютера и откройте свое место для соседа по лабораторной работе.
6. Сгенерируйте и установите открытый ключ для доступа, попробуйте беспарольный вход.
7. Установите правило, блокирующее пакеты от соседнего рабочего места и проверьте работоспособность фильтра. По окончании теста отмените это правило.

Список литературы

1. *Уэлш. М. и~др.,* Руководство по установке и использованию системы **Linux**. М.: ИЛКиРЛ, 1999
2. Александр Боковой, Александр Колотов, Александр Прокудин, Алексей Новодворский, Алексей Смирнов,Анатолий Якушин, Антон Бояршинов, Антон Ионов, Вадим Виниченко, Виталий Липатов, Георгий Курячий, Даниил Смирнов, Дмитрий Аленичев, Дмитрий Левин, Илья Трунин, Кирилл Маслинский, Максим Отставнов, Мэтт Уэлш, Олег Власенко, Сергей Турчин, Станислав Иевлев, Юрий Коновалов и другие; ALT Linux снаружи. ALT Linux изнутри, ISBN 5-9706-0029-6, Издатель: ДМК пресс, 2006 г. Москва
3. Марк Г. Собелл, Практическое руководство по Red Hat Linux: Fedora Core и Red Hat Enterprise Linux, 2-е издание (Practical Guide to Red Hat Linux: Fedora Core and Red Hat Enterprise Linux), 1072 стр., с ил.; ISBN 5-8459-0841-8, 0-13-147024-8; формат 70x100/16;твердый переплет DVD-ROM; 2005, 2 кв.; Вильямс.
4. Разработка приложений в среде Linux. Программирование для linux, 2-е издание, Майкл К. Джонсон, Эрик В. Троан
5. Руководство администратора Linux. Установка и настройка. 2-е издание, Эви Немет, Гарт Снайдер, Трент Хейн
6. Linux. Библия пользователя, Кристофер Негус
7. Linux для чайников , 6-е издание, Ди-Анн Лебланк
8. Разработка ядра Linux, 2-е издание, Роберт Лав
9. Библиотека Qt 4. Программирование прикладных приложений в среде Linux., Чеботарев Арсений Викторович
10. Red Hat Linux Fedora 4. Полное руководство, Пол Хадсон, Эндрю Хадсон, Билл Болл, Хойт Дафф
11. Искусство программирования для Unix, Эрик С. Реймонд
12. Linux для "чайников", 5-е издание, Ди-Анн Лебланк
13. Red Hat Linux. Секреты профессионала, Наба Баркакати
14. Использование Linux, Apache, MySQL и PHP для разработки Web-приложений, Джеймс Ли, Брент Уэр
15. Секреты хакеров. Безопасность сетей - готовые решения, 4-е издание, Стюарт Мак-Клар, Джоэл Скембрей, Джордж Курц
16. FreeBSD: полный справочник., Родерик Смит

17. Секреты хакеров. Безопасность Linux — готовые решения, 2-е издание, Брайан Хатч, Джеймс Ли, Джордж Курц
18. Red Hat Linux 8. Библия пользователя, Кристофер Негус
19. Серверы Linux. Самоучитель, Птицын Константин Александрович
20. Безопасность Linux, 2-е издание, Скотт Манн, Эллен Л. Митчелл, Митчелл Крелл
21. Сетевые средства Linux, Родерик Смит
22. Руководство администратора Linux, Эви Немет, Гарт Снайдер, Трент Хейн
23. Сети TCP/IP, том 3. Разработка приложений типа клиент/сервер для Linux/POSIX, Дуглас Камер, Дэвид Л. Стивенс
24. Секреты хакеров. Безопасность Linux — готовые решения, Брайан Хатч, Джеймс Ли, Джордж Курц
25. Программирование для Linux. Профессиональный подход, Марк Митчелл, Джеффри Оулдем, Алекс Самьюэл
26. Использование Linux, 6-е издание. Специальное издание, Дэвид Бендел, Роберт Нейпир
27. Создание сетевых приложений в среде Linux, Шон Уолтон
28. Освой самостоятельно Linux за 24 часа, 3-е издание,
29. Система электронной почты на основе Linux. Руководство администратора, Ричард Блам
30. Системное администрирование Linux, М. Карлинг, Стефан Деглер, Джеймс Деннис

Учебное издание

Механизмы безопасности в Linux

Методические указания

Составитель: Сухов Андрей Михайлович

Изд-во Самарского государственного
аэрокосмического университета.
443086 Самара, Московское шоссе, 34.