Защищаем Линукс - файрвол за 10 минут.

Антон Малащенко aka Rygoravich, Среда, 19 Январь 2005, 16:42

Благодарности.

Прежде всего хочу сказать спасибо Игорю Майорову и Алексею Горбунову, которые помогли мне разобраться с Iptables. Без них эта статья возможно и не появилась бы на свет.

Вступление.

Как известно, в наше время компьютеры все чаще и чаще подвергаются сетевым атакам. Излюбленная цель хакеров - компьютеры под управлением ОС Windows, которые сравнительно несложно заразить вирусом или трояном. Однако пользователи юникс-систем также не всегда остаются в стороне. Причина большинства взломов компьютеров с установленной ОС Linux - неправильно настроенные сервисы. Встречается такое достаточно часто, поскольку во многих дистрибутивах сервисы по умолчанию автоматически запускаются после установки. Что делать? Ответ прост - настроить файрвол.

Когда я столкнулся с необходимостью настроить файрвол, то прежде всего, конечно, стал обшаривать интернет в поисках документации. Результат был ошеломляющим - за исключением нескольких статей об общих принципах работы и замечательного (но требующего некоторого количества специальный знаний, очень большого и потому - отнимающего много времени на его изучение) мануала Iptables Tutorial русскоязычной документации нет вообще. Что же делать домашним пользователям, которым недосуг тратить долгие часы своего времени на чтение документации, но которые, тем не менее, хотят чувствовать себя защищенными? Не найдя ответа на этот вопрос я все же изучил вышеупомянутое руководство, проконсультировлся в почтовых листах с более опытными пользователями и, дабы облегчить жизнь тем, кому еще придется столкнуться с этим, пишу данную статью. Следует учесть, что она ориентирована на домашних пользователей - администраторамсерверов настоятельно рекомендуется изучить Iptables Tutorial.

Немного об устройстве файрвола в Линукс. Значительная часть его встроена в ядро и в подавляющем большинстве сборок от популярных дистрибьюторов подключается в виде модулей. Для управления файрволом в современных ядрах (ветки 2.4 и 2.6) используется утилита iptables. При включенном файрволе каждый пакет данных последовательнопроходит через несколько таблиц. Рассматривать их все мы не будем, нас интересует только таблица filter, в которой, собственно и происходит фильтрация трафика. Итак, таблица filter имеет три встроенные цепочки - INPUT, OUTPUT и FORWARD, предназначенные соответственно, для входящих, исходящих и перенаправляемых пакетов. Для каждой цепочки мы можем создавать свои правила. Все пакеты, присылаемые из сети и предназначенныедля вашего компьютера последовательнопроходят через правила в цепочке INPUT до тех пор, пока одно

из правил не определит, принять пакет или заблокировать. Существуют и другие действия, например, запись в лог, но здесь мы их рассматривать не будем. Пакеты, посылаемые вашим компьютером аналогичным образом проходят по цепочке OUTPUT. И наконец, пакеты, проходящие через ваш компьютер из одной сети в другую используют цепочку FORWARD.

Часть I - Базовая настройка.

Итак, начнем. Прежде всего - настройка файрвола - привелегия исключительно root-пользователя. Предполагается, что все команды вы будете исполнять с правами root (хотя команду ping можно запускать от любого пользователя). Зарегистрируйтеь в системе как root командой "su" и включите файрвол. Для этого загрузите соответствующий модуль ядра:

```
# modprobe iptable_filter
```

Реально эта команда загружает еще и модуль ip_tables, который необходим для iptable_filter. Теперь посмотрите текущие установки файрвола командой iptables-save:

```
# iptables-save
```

Вы увидите примерно следующее:

```
# Generated by iptables-save v1.2.8 on Fri Jan 14 05:27:49 2005
*filter
:INPUT ACCEPT [3:128]
:FORWARD ACCEPT [0:0]
 :OUTPUT ACCEPT [1:48]
COMMIT
# Completed on Fri Jan 14 05:27:49 2005
```

Это значит, что никаких правил пока не установлено. Обратите внимание на строки, содержащие имена цепочек и слово ACCEPT. Это значит, что любой пакет, для которого в цепочке не нашлось подходящего правила, автоматически принимается. Разумеется, это совсем не то, чего мы добиваемся. Поэтому выполните три команды:

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

Этими командами мы устанавливаем политику DROP по умолчанию. Это значит, что любой пакет, для которого явно не задано правило, которое его разрешает, автоматически отбрасывается. Поскольку пока еще у нас не задано ни одно правило - будут отвергнуты все пакеты, которые придут на ваш компьютер, равно как и те, которые вы попытаетесь отправить в сеть. В качестве демонстрации можно попробовать пропинговать свой компьютер через интерфейс обратной петли:

```
# ping -c 5 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- localhost ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 3999ms
```

Как мы видим, ни один из отправленных пакетов не достиг цели. Это, конечно, самый безопасный вариант, но зачем нам сеть, которой нельзя воспользоваться? Создадим правила, позволяющие отправлять пакеты в сеть и принимать их.

```
# iptables -A INPUT -i lo -j ACCEPT
```

Рассмотрим эту команду подробнее. Итак, ключ -A (от слова add) означает, что мы добавляем новое правило в цепочку, название которой следует после этого ключа. В данном случае это цепочка INPUT, т.е. правило будет действовать для всех пакетов, направляющихся на ваш компьютер. Ключ -i определяет сетевой интерфейс, с которого пришел пакет. В вышеупомянутом правиле это lo, соответствующий локальному интерфейсу обратной петли. Ключ -j задает действие, которое применяется ко всем пакетам, соответствующим перечисленным критериям, в нашем примере - ACCEPT, т.е. разрешить. Таким образом одной этой командой мы разрешаем получение всех пакетов с заданного интерфейса. Однако отправить их пока что не получится - так как это действие запрещено, а точнее, не разрешено. Разрешаем:

```
# iptables -A OUTPUT -o lo -j ACCEPT
```

Несложно догадаться, что здесь мы добавляем новое правило в цепочку OUTPUT (как вы помните, через нее проходят пакеты, отправляемые с нашего компьютера). Обратите внимание, что здесь вместо -i lo используется -o lo - это значит, что мы определяем сетевой

интерфейс, на который будут отправлены пакеты. Теперь можно проверить работу интерфейса обратной петли командой ping:

```
# ping -c 5 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.226 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.123 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.125 ms
64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64 time=0.124 ms
64 bytes from localhost (127.0.0.1): icmp_seq=5 ttl=64 time=0.123 ms

--- localhost ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.123/0.144/0.226/0.041 ms
```

Итак, интерфейс loopback работает и позволяет прохождение любых пакетов. Переходим к следующей стадии.
[newpage]

Часть II - Настройка для работы приложений-клиентов в сети Интернет.

Для начала разрешим прохождение служебных пакетов (вроде генерируемых командой ping, сообщений о недоступности хоста и т.п.).

```
# iptables -A INPUT -p ICMP -j ACCEPT
# iptables -A OUTPUT -p ICMP -j ACCEPT
```

Здесь -p ICMP указыват, что данное правило применимо только к пакетам протокола (-p от protocol) ICMP, т.е. к служебным пакетам. Интерфейс явно не указываем - это значит, что правило подходит для всех сетевых интерфейсов, имеющихся в системе. Однако, одних только служебных пакетов недостаточно...

```
# iptables -A INPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A INPUT -p UDP -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Здесь мы разрешаем пакеты, принимаемые по протоколам TCP и UDP (соответственно, в первой и второй командах), имеющие статус ESTABLISHED или RELATED. Ключ -m определяет тип критерия (т.е. в нашем случае - определять по статусу), а --state явно указывает, к пакетам с каким статусом применять правило. Всего статусов существует четыре: INVALID, ESTABLISHED, NEW и RELATED, но нас интересуют только эти два: ESTABLISHED означает, что

пакет принадлежит установленному соединению, а RELATED порождает новое соединение из уже существующего (используется, например, протоколом ftp). В упрощенной форме можно сказать, что эти два правила разрешают попадание на ваш компьютер только тех TCP- и UDP-пакетов, которые были запрошены приложениями с вашего компьютера.

Однако пока еще запросить эти пакеты не представляется возможным, ибо в интернет могут уйти пакеты только по ICMP-протоколу. Для клиентских приложений делаем следующее:

```
# iptables -A OUTPUT -p TCP --sport 32768:65535 -j ACCEPT
# iptables -A OUTPUT -p UDP --sport 32768:65535 -j ACCEPT
```

Здесь --sport (source port) означает диапазон портов, с которых можно отправить пакеты. Как правило, именно указанные нами порты используют клиентские приложения, а порты с адресами ниже 32768 могут использоваться серверными приложениями. Проверим созданный набор правил командой iptables-save:

```
# iptables-save
# Generated by iptables-save v1.2.8 on Fri Jan 14 06:47:52 2005
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
 :OUTPUT DROP [20:1572]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 32768:65535 -j ACCEPT
-A OUTPUT -p udp -m udp --sport 32768:65535 -j ACCEPT
COMMIT
# Completed on Fri Jan 14 06:47:52 2005
```

Теперь посмотрим, что же делать, если вы где-то ошиблись в наборе и ввели неверное правило. Попробуйте сделать следующее:

```
# iptables -A OUTPUT -p TCP -j ACCEPT
# iptables-save
# Generated by iptables-save v1.2.8 on Wed Jan 19 06:15:32 2005
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
 :OUTPUT DROP [0:0]
```

```
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 32768:65535 -j ACCEPT
-A OUTPUT -p udp -m udp --sport 32768:65535 -j ACCEPT
-A OUTPUT -p tcp -j ACCEPT
COMMIT
# Completed on Wed Jan 19 06:15:32 2005
```

Как видим, появилось новое правило, позволяющее отправку любых TCP-пакетов. Это нам совсем ни к чему, и не помешает его удалить. Выполняем исходную команду, заменив -A ключом -D:

```
# iptables -D OUTPUT -p TCP -j ACCEPT
# iptables-save
# Generated by iptables-save v1.2.8 on Wed Jan 19 06:18:32 2005
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
 :OUTPUT DROP [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 32768:65535 -j ACCEPT
-A OUTPUT -p udp -m udp --sport 32768:65535 -j ACCEPT
COMMIT
# Completed on Wed Jan 19 06:18:32 2005
```

На сей раз в выводе iptables-save этого правила нет. Таким образом, удаление правил из цепочек можно производить точно так же, как и добавление, заменив ключ -A (add) на -D (delete).

[newpage]

Часть III - Настройка доступа из локальной сети.

В настоящее время все более популярными становятся домашние локальные сети. Если у вас ее нет, вы можете смело пропустить эту часть статьи и сразу перейти к следующей. Если есть

- читайте далее.

Итак, все правила, которые мы задали для интернета будут справедливы и для локальной сети - вы уже можете зайти на сервера вашей сети. Однако достаточно часто требуется большее, например, предоставить какие-либо ресурсы другим компьютерам в сети. К тому же, если в сети нет контроллера домена, то вы не сумеете просмотреть список доступных компьютеров, если их адреса не прописаны в /etc/hosts.

Будем исходить из предположения, что ваша сеть основана на технологиях Microsoft - на сегодняшний день это самые распространенны сети. Обмен данными в таких сетях основан на протоколах SMB и NetBEUI (расширение Microsoft для протокола NetBIOS). Мы не будем рассматривать тонкости настройки этих протоколов - ограничимся рассмотрением аспектов их работы при использовании файрвола. Обратите внимание - далее во всех примерах предполагается, что вы подключены к локальной сети через интерфейс eth0. Если используется другой интерфейс - замените все вхождения eth0 на интерфейс вашей локальной сети. Ну что ж, приступим.

```
# iptables -A INPUT -i eth0 -p TCP --dport 137:139 -j ACCEPT
# iptables -A INPUT -i eth0 -p UDP --dport 137:139 -j ACCEPT
```

Эти команды добавляют в цепочку INPUT правила, разрешающие прием пакетов по протоколам TCP и UDP, пришедших с сетевого интерфейса eth0 на порты 137, 138 и 139 (--dport от "destination port"). Именно эти порты используются протоколом NetBIOS. Аналогичным образом делаем для цепочки OUTPUT, не забыв, естественно, заменить ключи -i и --dport на -o и --sport соответственно (т.е. интерфейс назначения вместо интерфейса источника и порт источника вместо порта назначения):

```
# iptables -A OUTPUT -o eth0 -p TCP --sport 137:139 -j ACCEPT
# iptables -A OUTPUT -o eth0 -p UDP --sport 137:139 -j ACCEPT
```

Теперь ваш компьютер может быть не только клиентом, но и smb-сервером, разумеется, если вы настроили соответствующий сервис. Однако остается нерешенным один вопрос - определение адресов по протоколу NetBIOS:

```
# nmblookup server
querying server on 192.168.255.255
name_query failed to find name server
```

Здесь предполагается, что server соответствует имени какого-либо компьютера, подключенного сети. Дело в том, что для определения адреса компьютера команда nmblookup открывает порт в верхнем диапазоне адресов и посылает с него один запрос на порт 137 всех компьютеров сети сразу (broadcast) по протоколу UDP, после чего компьютер,

чье имя соответствует запрошенному в исходном пакете отвечает конкретно вашему. При этом ответный пакет не может получить статус ESTABLISHED, т.к. адрес отправителя (например 192.168.5.1) не соответствует адресу, на который был послан запрос (например 192.168.255.255). Однако зная механизм определения адреса несложно создать соответствующее правило:

```
# iptables -A INPUT -i eth0 -p UDP --sport 137 --dport 32768:65535 -j ACCEPT
```

Проверяем результат:

```
# nmblookup server
querying server on 192.168.255.255
192.168.5.1 server<00>
```

Работает. Итак, у нас получился следующий набор правил:

```
# iptables-save
# Generated by iptables-save v1.2.8 on Sat Jan 15 07:08:29 2005
*filter
:INPUT DROP [714:58115]
:FORWARD DROP [0:0]
 :OUTPUT DROP [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 137:139 -j ACCEPT
-A INPUT -i eth0 -p udp -m udp --dport 137:139 -j ACCEPT
-A INPUT -i eth0 -p udp -m udp --sport 137 --dport 32768:65535 -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 32768:65535 -j ACCEPT
-A OUTPUT -p udp -m udp --sport 32768:65535 -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --sport 137:139 -j ACCEPT
-A OUTPUT -o eth0 -p udp -m udp --sport 137:139 -j ACCEPT
COMMIT
# Completed on Sat Jan 15 07:08:29 2005
```

А что делать, если вам захочется раздать какой-нибудь сервис на всю сеть? Возможно, у вас настроен http- или ftp-сервер и вы хотите, чтобы он был доступен всем компьютерам в локальной сети. Общий рецепт прост - откройте TCP-порт, который обслуживает данный сервер. Напрмер, для http-сервера это можно сделать так:

```
# iptables -A INPUT -i eth0 -p TCP --dport 80 -j ACCEPT
# iptables -A OUTPUT -o eth0 -p TCP --sport 80 -j ACCEPT
```

T.e. для любого сервиса нужно добавить по одному правилу в цепочки INPUT и OUTPUT, разрешающему соответственно прием и отправку пакетов с использованием этого порта для конкретного сетевого интерфейса. Список соответствия портов конкретным сервисам находится в файле /etc/services. Для некоторых сервисов также имеет смысл открыть UDP-порт, что может привести к некоторому ускорению обслуживания, однако вполне можно обойтись и без него. Кроме того, нужно учесть еще один момент - для использования некоторых сервисов (например, ftp или irc) требуется загрузка дополнительных модулей. Причины этого описаны в Iptables Tutorial, здесь же укажем только команды, необходимые для работы этих протоколов.

```
# modprobe ip_conntrack_ftp
```

Для работы сервера FTP.

```
# modprobe ip_conntrack_irc
```

Для работы сервера IRC.

Итак, все правила созданы. Однако впереди нас ждет еще один подводный камень - они будут работать только до первой перезагрузки... А после нее опять будут настройки по умолчанию, то есть разрешение для всех действий. Что делать? Читать следующую часть статьи.

[newpage]

Часть IV - Сохранение и восстановление конфигурации файрвола.

Вряд ли кому-нибудь захочется каждый раз вручную вводить все эти команды при каждом запуске компьютера. Я даже не говорю о том, что пока мы не зададим нужные правила, нас вполне могут успеть взломать... Соответственно, необходимо предусмотреть установку заданного набора правил при каждой загрузке компьютера. Причем очень желательно, чтобы установка правил файрвола происходила до активизации сетевых интерфейсов. Можно, конечно, создать шелл-скрипт, который бы запускал нужные команды при загрузки, но есть способ проще... Вспомните, мы ведь неоднократно для просмотра списка правил использовали команду iptables-save. По ее названию несложно догадаться, что основной функцией программы является совсем не это.

Сохраняем список правил:

```
# iptables-save >/root/iptables.save
```

где /root/iptables.save - произвольное имя файла. Теперь в этом файле находится полный список правил, которые мы задали и в любой момент мы можем восстановить его. Восстановление производится командой iptables-restore:

```
# iptables-restore /root/iptables.save
```

Эту команду можно прописать в стартовые скрипты, однако скорее всего за вас это уже сделал производитель дистрибутива. Проверьте, есть ли у вас сервис iptables:

```
# chkconfig --list iptables
iptables        0:выкл  1:выкл  2:выкл  3:выкл  4:выкл  5:выкл  6:выкл
```

Если его нет - будет выведено сообщение об ошибке. В этом случае пропишите в файл /etc/rc.d/rc.local две строки:

```
modprobe iptable_filter
iptables-restore /root/iptables.save
```

Где /root/iptables.save - файл с вашим набором правил, созданный при помощи iptables-save. Если же скрипт для запуска iptables есть в вашей системе, то скопируйте файл /root/iptables.save в /etc/sysconfig/iptables (важное предупреждение: в вашей системе этот путь может отличаться - посмотрите значение переменной IPTABLES_CONFIG в файле /etc/rc.d/init.d/iptables) и активизируйте старт iptables при загрузке:

```
# chkconfig --add iptables
```

На этом настройка завершена. Однако имейте в виду, что файрвол не является панацеей от всех бед - это только еще один (но зачастую весьма эффективный) способ защиты вашей системы.

Заключение.

Приведенные в этой статье настройки файрвола не являются оптимальными для всех систем - очень часто, особенно на серверах, нужно вводить некоторые коррективы. Некоторые из предложенных действий можно рассматривать как далеко не самые удачные. Например, запуск файрвола из файла /etc/rc.d/rc.local вряд ли можно назвать оптимальным решением, т.к. файрвол в этом случае будет запускаться после поднятия сетевых интерфейсов, а кроме того, настройки системы не принято хранить в каталоге /root. Однако они вполне подойдут для большинства домашних компьютеров, а кроме того просты, логичны и не требуют специальных знаний. Рассматривайте эту статью только в качестве пошагового руководства базовой конфигурации файрвола, но если у вас есть потребность в более тонкой настройке - обратитесь к Iptables Tutorial.

Надежной вам защиты!