На защиту сервера от внешних угроз в первую очередь встает межсетевой экран, который фильтрует входящий и исходящий траффик. Настройкой iptables — частного случая фаервола на CentOS я хочу заняться в данной статье, а также рассказать о его установке и отключении. Мое руководство не будет исчерпывающим, я рассмотрю лишь те аспекты, которые считаю наиболее важными и сам использую в своей работе.

Содержание:

- 1 Вступление
- 2 Отключение firewalld
- 3 Установка iptables
- 4 Настройка фаервола
- 5 Открытие портов
- 6 Проброс (forward) порта
- 7 Включение логов
- 8 Как отключить iptables
- 9 Заключение
- 10 Видео

Вступление

Iptables в настоящее время является стандартом де-факто в среде современных linux дистрибутивов. Я даже сходу не могу припомнить, что еще используют в качестве фаервола. Так что любому администратору линукс приходится сталкиваться в своей работе с настройкой этого межсетевого экрана.

К этому фаерволу существуют разные обвязки, которые используются для более «удобной» настройки. В ubuntu есть **ufw**, в centos — **firewalld**, с другими не знаком. Лично я не вижу никакого удобства в использовании этих инструментов. Я привык настраивать линуксовый фаервол по-старинке, как научился в самом начале работы. Мне кажется это наиболее простым и удобным способом, которым я с вами и поделюсь. Суть его сводится к тому, что создается скрипт с правилами фаервола. Этот скрипт можно легко редактировать под свои нужды и переносить с сервера на сервер.

Отключение firewalld

Вопрос отключения firewalld я уже касался в теме по настройке сервера. Первым делом отключим firewalld, который присутствует в centos 7 по-умолчанию сразу после установки:

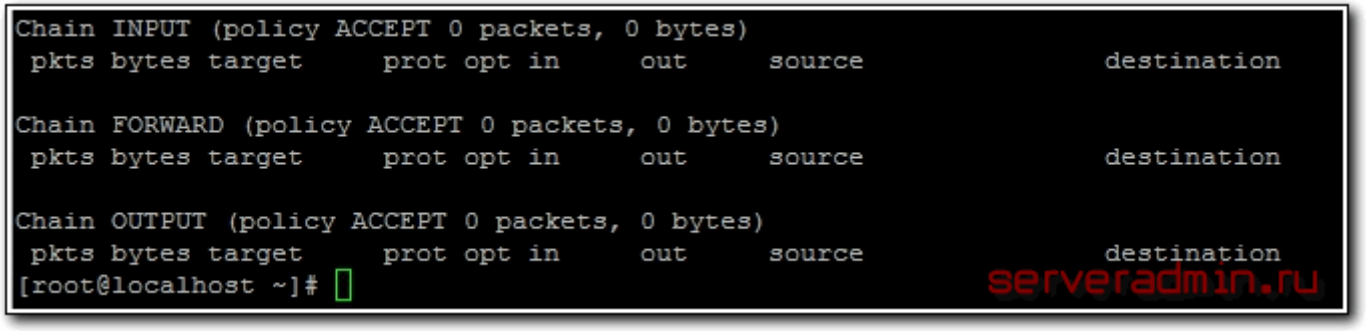
```
# systemctl stop firewalld
```

Теперь удалим его из автозагрузки, чтобы он не включился снова после рестарта:

```
# systemctl disable firewalld
```

После этого на сервере настройки сетевого экрана становятся полностью открытыми. Посмотреть правила iptables можно командой:

```
# iptables -L -v -n
```



Дальше пойдет информация исключительно по конфигурированию только iptables. Темы firewalld я больше касаться не буду.

Установка iptables

На самом деле фаервол у нас на сервере уже стоит и работает, просто нет никаких правил, все открыто. Установить нам нужно будет дополнительные утилиты управления, без которых конфигурировать iptables невозможно. Например, нельзя будет перезапустить фаервол:

```
# systemctl restart iptables.service
Failed to issue method call: Unit iptables.service failed to load: No such file or directory.
```

Или добавить в автозапуск не получится:

```
# systemctl enable iptables.service
Failed to issue method call: No such file or directory
```

Чтобы подобных ошибок не было, установим необходимый пакет с утилитами:

```
# yum -y install iptables-services
```

Теперь можно добавить iptables в автозагрузку и запустить:

```
# systemctl enable iptables.service
# systemctl start iptables.service
```

Настройка фаервола

Для управления правилами фаервола я использую скрипт. Создадим его:

```
# mcedit /etc/iptables.sh
```

Далее будем наполнять его необходимыми правилами. Я буду разбирать все значимые части скрипта, а **полностью его приведу в виде текстового файла в конце статьи**. Правила сделаны в виде картинок, чтобы запретить копирование и вставку. Это может привести к ошибкам в работе правил, с чем я сам столкнулся во время подготовки статьи.

Мы рассмотрим ситуацию, когда сервер является шлюзом в интернет для локальной сети.

Первым делом зададим все переменные, которые будем использовать в скрипте. Это не обязательно делать, но рекомендуется, потому что удобно переносить настройки с сервера на сервер. Достаточно будет просто переназначить переменные.

```
#!/bin/bash
export IPT="iptables"
# Внешний интерфейс
export WAN=eth0
export WAN_IP=85.31.203.127
# Локальная сеть
export LAN1=eth1
export LAN1_IP_RANGE=10.1.3.0/24
```

Перед применением новых правил, очищаем все цепочки:

```
$IPT -F
$IPT -F -t nat
$IPT -F -t mangle
$IPT -X
$IPT -t nat -X
$IPT -t mangle -X
```

Блокируем весь трафик, который не соответствует ни одному из правил:

```
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
```

Разрешаем весь трафик локалхоста и локалки:

```
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A INPUT -i $LAN1 -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT
$IPT -A OUTPUT -o $LAN1 -j ACCEPT
```

Разрешаем делать ping:

```
$IPT -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
$IPT -A INPUT -p icmp --icmp-type destination-unreachable -j ACCEPT
$IPT -A INPUT -p icmp --icmp-type time-exceeded -j ACCEPT
$IPT -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Если вам это не нужно, то не добавляйте разрешающие правила для icmp.

Открываем доступ в инет самому серверу:

```
$IPT -A OUTPUT -o $WAN -j ACCEPT
```

Если вы хотите открыть все входящие соединения сервера, то добавляйте дальше правило:

```
$IPT -A INPUT -i $WAN -j ACCEPT
```

Делать это не рекомендуется, привожу просто для примера, если у вас появится такая необходимость.

Дальше разрешим все установленные соединения и дочерние от них. Так как они уже установлены, значит прошли через цепочки правил, фильтровать их еще раз нет смысла:

```
$IPT -A INPUT -p all -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A OUTPUT -p all -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A FORWARD -p all -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Теперь добавим защиту от наиболее распространенных сетевых атак. Сначала отбросим все пакеты, которые не имеют никакого статуса:

```
$IPT -A INPUT -m state --state INVALID -j DROP
$IPT -A FORWARD -m state --state INVALID -j DROP
```

Блокируем нулевые пакеты:

```
$IPT -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
```

Закрываемся от syn-flood атак:

```
$IPT -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
$IPT -A OUTPUT -p tcp ! --syn -m state --state NEW -j DROP
```

Следом за этими правилами рекомендуется поставить правила на запрет доступа с определенных IP, если у вас имеется такая необходимость. Например, вас задолбал адрес 84.122.21.197 брутом ssh. Блокируем его:

```
$IPT -A INPUT -s 84.122.21.197 -j REJECT
```

Если вы не ставите ограничений на доступ из локальной сети, то разрешаем всем выход в интернет:

```
$IPT -A FORWARD -i $LAN1 -o $WAN -j ACCEPT
```

Следом запрещаем доступ из инета в локальную сеть:

```
$IPT -A FORWARD -i $WAN -o $LAN1 -j REJECT
```

Чтобы наша локальная сеть пользовалась интернетом, включаем nat:

```
$IPT -t nat -A POSTROUTING -o $WAN -s $LAN1_IP_RANGE -j MASQUERADE
```

Чтобы не потерять доступ к серверу, после применения правил, разрешаем подключения по ssh:

```
$IPT -A INPUT -i $WAN -p tcp --dport 22 -j ACCEPT
```

И в конце записываем правила, чтобы они применились после перезагрузки:

```
/sbin/iptables-save  > /etc/sysconfig/iptables
```

Мы составили простейший конфиг, который блокирует все входящие соединения, кроме ssh и разрешает доступ из локальной сети в интернет. Попутно защитились от некоторых сетевых атак.

Сохраняем скрипт, делаем исполняемым и запускаем:

```
# chmod 0740 /etc/iptables.sh
# /etc/iptables.sh
```

Выполним просмотр правил и проверим, все ли правила на месте:

```
# iptables -L -v -n
```

> Обращаю ваше внимание — применять правила нужно лишь в том случае, если у вас имеется доступ к консоли сервера. При ошибке в настройках вы можете потерять доступ. Убедитесь, что в нештатной ситуации вы сможете отключить фаервол и скорректировать настройки.

Открытие портов

Теперь немного расширим нашу конфигурацию и откроем в iptables порты для некоторых сервисов. Допустим, у нас работает веб-сервер и необходимо открыть к нему доступ из интернета. Добавляем правила для веб-трафика:

```
#$IPT -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
#$IPT -A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
```

Было добавлено разрешение на входящие соединения по 80-му и 443-му портам, которые использует web сервер в своей работе.

Если у вас установлен почтовый сервер, то нужно разрешить на него входящие соединения по всем используемым портам:

```
$IPT -A INPUT -p tcp -m tcp --dport 25 -j ACCEPT
$IPT -A INPUT -p tcp -m tcp --dport 465 -j ACCEPT
$IPT -A INPUT -p tcp -m tcp --dport 110 -j ACCEPT
$IPT -A INPUT -p tcp -m tcp --dport 995 -j ACCEPT
$IPT -A INPUT -p tcp -m tcp --dport 143 -j ACCEPT
$IPT -A INPUT -p tcp -m tcp --dport 993 -j ACCEPT
```

Для корректной работы DNS сервера, нужно открыть UDP порт 53

```
$IPT -A INPUT -i $WAN -p udp --dport 53 -j ACCEPT
```

И так далее. По аналогии можете открыть доступ для всех необходимых сервисов.

Проброс (forward) порта

Рассмотрим ситуацию, когда необходимо выполнить проброс портов с внешнего интерфейса на какой-то компьютер в локальной сети. Допустим, вам необходимо получить rdp доступ к компьютеру 10.1.3.50 из интернета. Делаем проброс TCP порта 3389:

```
$IPT -t nat -A PREROUTING -p tcp --dport 3389 -i ${WAN} -j DNAT --to 10.1.3.50
```

Если вы не хотите светить снаружи известным портом, то можно сделать перенаправление с нестандартного порта на порт rdp конечного компьютера:

```
$IPT -t nat -A PREROUTING -p tcp --dport 23543 -i ${WAN} -j DNAT --to 10.1.3.50:3389
```

Если вы пробрасываете порт снаружи внутрь локальной сети, то обязательно закомментируйте правило, которое блокирует доступ из внешней сети во внутреннюю. В моем примере это правило:

```
$IPT -A FORWARD -i $WAN -o $LAN1 -j REJECT
```

Либо перед этим правилом создайте разрешающее правило для доступа снаружи к внутреннему сервису, например вот так:

```
$IPT -A FORWARD -i $WAN -d 10.1.3.50 -p tcp -m tcp --dport 3389 -j ACCEPT
```

Включение логов

Во время настройки полезно включить логи, чтобы мониторить заблокированные пакеты и выяснять, почему отсутствует доступ к необходимым сервисам, которые мы вроде бы уже открыли. Я отправляю все заблокированные пакеты в отдельные цепочки (block_in, block_out, block_fw), соответствующие направлению трафика и маркирую в логах каждое направление. Так удобнее делать разбор полетов. Добавляем следующие правила в самый конец скрипта, перед сохранением настроек:

```
$IPT -N block_in
$IPT -N block_out
$IPT -N block_fw

$IPT -A INPUT -j block_in
$IPT -A OUTPUT -j block_out
$IPT -A FORWARD -j block_fw

$IPT -A block_in -j LOG --log-level info --log-prefix "--IN--BLOCK"
$IPT -A block_in -j DROP
$IPT -A block_out -j LOG --log-level info --log-prefix "--OUT--BLOCK"
$IPT -A block_out -j DROP
$IPT -A block_fw -j LOG --log-level info --log-prefix "--FW--BLOCK"
$IPT -A block_fw -j DROP
```

Все заблокированные пакеты вы сможете отследить в файле /var/log/messages.

После того, как закончите настройку, закомментируйте эти строки, отключив логирование. Обязательно стоит это сделать, так как логи очень быстро разрастаются. Практического смысла в хранении подобной информации лично я не вижу.

Как отключить iptables

Если вы вдруг решите, что firewall вам больше не нужен, то отключить его можно следующим образом:

```
# systemctl stop iptables.service
```

Эта команда останавливает фаервол. А следующая удаляет из автозагрузки:

```
# systemctl disable iptables.service
```

Отключив сетевой экран, мы разрешили все соединения.

Заключение

Как и обещал, выкладываю готовый скрипт с основным набором правил, которые мы рассмотрели iptables.sh

Хочу еще раз обратить внимание, что при настройке iptables необходимо быть предельно внимательным. Не начинайте это дело, если не имеете доступа к консоли сервера. Даже во время написания этой статьи я потерял доступ к серверу из-за нелепой ошибке в правилах. Ошибка эта возникла из-за копирования и потери двойного тире — оно заменилось на одинарное.

А представляете, если это был бы удаленный сервер? Ко мне так обратился один знакомый, который попросил настроить firewall на веб-сервере. Предыдущий админ заставил его 2 раза приезжать к размещению машины и сбрасывать настройки неправильно сконфигурированного экрана, что в конечном счете и привело к прекращению сотрудничества. Это было давно, но случай мне запомнился. Сейчас практически все хостинги предлагают платно или бесплатно удаленный KVM доступ. Лучше озаботиться об этом заранее.

Видео

Помогла статья? Есть возможность отблагодарить автора