Simple stateful firewall (Русский)


Эта статья или раздел нуждается в переводе


Примечания: Перевод устарел. (обсуждение: Talk:Simple stateful firewall (Русский)#)


Эта страница нуждается в сопроводителе


Статья не гарантирует актуальность информации. Помогите русскоязычному сообществу поддержкой подобных страниц. См. Команда переводчиков ArchWiki

Это руководство объяснит как настроить простой полноценный firewall (брандмауэр), используя всего лишь iptables. Также пытается показать правила и объяснить зачем они нужны. Это руководство делится на две части. Первая часть для одной системы, вторая часть для NAT шлюза.

Важно: Это руководство нужно применять только войдя в систему локально. Если же Вы войдете в систему удаленно (по SSH, например), то скорее всего потеряете соединение с удаленной системой. Вы были предупреждены!

Contents

[hide]

Требования

До начала настройки, вам необходимо удостоверится, что все необходимые инструменты у вас доступны:

```
$ pacman -Q iptables
iptables 1.4.5-1
```

Если же пакет **iptables** уже установлен, то вы должны быть уверены что ваше ядро поддерживает iptables. Все сборки ядра Arch Linux по умолчанию имеют поддержку iptables.

Данное руководство предполагает что на данный момент в iptables правила не установлены. Чтобы проверить это, выполните команду

```
# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination

Chain OUTPUT (policy ACCEPT 0K packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination
```

Если вывод такой же как указано выше (исключая счетчики пакетов), правила не установлены, можно двигаться дальше. Если же нет, сбросьте таблицы следующим образом:

```
# iptables -P INPUT ACCEPT
# iptables -P FORWARD ACCEPT
# iptables -P OUTPUT ACCEPT
# iptables -F
# iptables -X
```

Настрока для одной системы

Созданием необходимых цепочек

Для базовой настройки, мы создадим две обычных цепочки (chains), о которых расскажем позже:

```
# iptables -N open
# iptables -N interfaces
```

Цепочка INPUT

Все пакеты, которые принимаются через любой сетевой интерфейс и имеют один локальный IP адрес хоста в заголовке назначения пройдут через цепоку **INPUT** сначала. Через эту цепочку должны проходить только те пакеты, которые мы хотим принимать.

Первое правило, будет принимать все **ICMP** сообщения. **ICMP** означает **Internet Control Message Protocol**. Некоторые **ICMP** сообщения очень важны, некоторые менее важны (как эхо-запросы (Pings), но никто из за них не пострадал, так что в целом это хорошая идея, не блокировать их:

```
# iptables -A INPUT -p icmp -j ACCEPT
```

Следующее правило позволит убедится в том, что никакая часть трафика, принадлежащего к уже установленным соединениям не будет отброшена (dropped). This can be done with the **state** match. A package can have one of the four states **ESTABLISHED**, **RELATED**, **NEW** and **INVALID**. So, we want to accept all packets that are in state **ESTABLISHED** or **RELATED**, hence the name "stateful firewall":

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

В большинстве случаев, мы не хотим отбрасывать все входящие соединения, и именно поэтому мы настраиваем две выборочные цепочки **open** и **interfaces**. Поэтому, мы добавим правило для каждой из них:

```
# iptables -A INPUT -j interfaces
# iptables -A INPUT -j open
```

Теперь, последними двумя правилами, мы отбрасываем все, что не было явно разрешено выше. Для **TCP** пакетов, мы запрещаем подключения с **tcp-reset**. Входящим **UDP** пакетам отвечают **ICMP** сообщениями.

```
# iptables -A INPUT -p tcp -j REJECT --reject-with tcp-reset
# iptables -A INPUT -p udp -j REJECT --reject-with icmp-port-unreachable
```

Все другие протоколы, кроме **TCP**, **UDP** and **ICMP** будут отброшены (если они не соответствовали правилам выше). Мы делаем это путем установки к цепочке **INPUT**политики **DROP**

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

Цепочка FORWARD

Если вы настраиваете шлюз NAT , пожалуйста обратитесь ко второй части данного руководства. Если нет то, мы просто устанавливаем к цепочке **FORWARD** политику **DROP**:

```
# iptables -P FORWARD DROP
```

Цепочка OUTPUT

В общем случае не требуется фильтровать весь исходящий трафик, поскольку это сделает бы настройку намного более сложной и потребует дополнительных вычислительных мощностей. Поэтому устанавливаем к цепочке **OUTPUT** политику **ACCEPT**.

```
# iptables -P OUTPUT ACCEPT
```

Цепочка interfaces

Мы будем использовать цепочку **interfaces** для всего трафика из надежных сетевых интерфейсов. Первое необходимое правило:

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

Это правило принимает весь трафик с интерфейса loopback, который необходим для нормальной работы многих приложений. Вы можете добавить сюда больше интерфейсов. Для примера, если вы хотите принимать весь входящий трафик с интерфейса **eth0**, добавьте это правило:

```
# iptables -A interfaces -i eth0 -j ACCEPT
```

Входящие соединения на остальные интерфейсы будут запрещены, до тех пор пока они не попадут под другие исключения в цепочке **open**.

Цепочка open

Цепочка **open** содержит правила для того, чтобы принимать входящие подключения на определенных портах или протоколах. Например, если Вы хотите принимать SSH соединения на любых сетевых интерфейсах, то добавьте это правило:

```
# iptables -A open -p tcp --dport 22 -j ACCEPT
```

Однако, это не очень хорошая идея, т.к. Вы открываете всему миру доступ к вашей машине через 22 порт. Поэтому, Вы можете ограничить, какие машины могут подключаться по 22 порту, изменяя файл /etc/hosts.allow :

```
# Позволяем локальным пользователям подключаться через ssh
sshd: 127.0.0.1

# Разрешить эти адреса для соединения через SSH
sshd: 192.168.0.1
sshd: 172.272.0.32
```

Разрешаем HTTP соединения через сетевой интерфейс ppp0:

```
# iptables -A open -i ppp0 -p tcp --dport 80 -j ACCEPT
```

Принимать все входящие tcp подключения с назначением портов от 65000 до 65005 на интерфейс foo:

```
# iptables -A open -i foo -p tcp --dport 65000:65005 -j ACCEPT
```

То же самое конечно возможно с udp:

```
# iptables -A open -i foo -p udp --dport 65000:65005 -j ACCEPT
```

или с другими протоколами, отличными от tcp и udp:

```
# iptables -A open -i foo -p 123 -j ACCEPT
```

смотрите руководство к iptables для других различных правил, как соответствие multiple ports or protocols.

Защита от основных атак

Force SYN packets check

Make sure NEW incoming tcp connections are SYN packets; otherwise, we need to drop them:

```
  iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```

Force Fragments packets check

Packets with incoming fragments. Drop them.

```
  iptables -A INPUT -f -j DROP
```

XMAS пакеты

Incoming malformed XMAS packets. Drop them:

```
  iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
```

Drop all NULL packets

Incoming malformed NULL packets:

```
  iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
```

Spoofing attack

Blocking reserved private networks incoming from the internet

```
  iptables -I INPUT -i eth0 -s 10.0.0.0/8 -j DROP
  iptables -I INPUT -i eth0 -s 172.16.0.0/12 -j DROP
  iptables -I INPUT -i eth0 -s 192.168.0.0/16 -j DROP
  iptables -I INPUT -i eth0 -s 127.0.0.0/8 -j DROP
```

You can also add the following line to `/etc/sysctl.d/90-firewall.conf` to enable source address verification which is built into Linux kernel itself.

```
net.ipv4.conf.all.rp_filter = 1
```

"Прячем" ваш компьютер

Если вы работаете на настольной машине, это могла бы быть хорошая идея блокировать некоторые входящие запросы.

Блокирование Ping запросов

A 'Ping' request is an ICMP packet sent to the destination address to ensure connectivity between the devices. If your network works well, you can safely block all ping requests.

```
iptables -A INPUT -p icmp --icmp-type echo-request -i eth0 -j DROP
```

You can also add the following line to `/etc/sysctl.d/90-firewall.conf` file:

```
net.ipv4.icmp_echo_ignore_all = 1
```

ICMP type match blocking

Если Ваш компьютер не маршрутизатор (как большинство настольных компьютеров):

```
iptables -I INPUT -p icmp --icmp-type redirect -j DROP
iptables -I INPUT -p icmp --icmp-type router-advertisement -j DROP
iptables -I INPUT -p icmp --icmp-type router-solicitation -j DROP
iptables -I INPUT -p icmp --icmp-type address-mask-request -j DROP
iptables -I INPUT -p icmp --icmp-type address-mask-reply -j DROP
```

Block nmap's uptime detection

Prevent uptime detection from port scanners like nmap

Add the following line to `/etc/sysctl.d/90-firewall.conf`

```
net.ipv4.tcp_timestamps = 0
```

Прочие настройки ядра

Add the following lines to `/etc/sysctl.d/90-firewall.conf` to prevent certain kinds of attacks:

```
net.ipv4.conf.all.accept_source_route=0
net.ipv4.icmp_echo_ignore_broadcasts=1
net.ipv4.icmp_ignore_bogus_error_responses=1
```

Сохранение правил

Now, the rules are ready and should be saved to your hard drive. First, we need edit the configuration file **/etc/conf.d/iptables**:

```
# Конфигурация для iptables правил
```

```
IPTABLES=/usr/sbin/iptables

IPTABLES_CONF=/etc/iptables/iptables.rules
IPTABLES_FORWARD=0  # disable IP forwarding!!!
```

You can specify another filename than **iptables.rules** if you want.

Сейчас сохраним правила командой:

```
# /etc/rc.d/iptables save
```

and make sure your rules are loaded when you boot by editing **/etc/rc.conf**, iptables should be added preferably before 'network'.

```
 DAEMONS=(... iptables network ...)
```

Настройка NAT шлюза

Эта секция руководства описывает настройку NAT шлюза. Преполагается, что Вы уже прочитали первую часть руководства и настроили
цепочки **INPUT**, **OUTPUT**, **open** и **interfaces** как описано было выше . Все выше описанные правила были созданы в таблице **filter**, а в этой секции мы будем использовать еще и таблицу **nat**.

Настройка таблицы filter

Перед тем как начнем, убедитесь, что цепочка **FORWARD** пуста:

```
# iptables -F FORWARD
```

Создание цепочки necessary

В нашей настройке, мы будем использовать еще две цепочки в таблице filter: **fw-interfaces** и **fw-open**. Создадим их с помощью команд:

```
# iptables -N fw-interfaces
# iptables -N fw-open
```

Настройка цепочки FORWARD

Настройка цепочки **FORWARD** аналогична настройке цепочки **INPUT** в первой части данного руководства.

Некоторые сети и сервера блокируют сообщения **ICMP** и препятствуют **path MTU detection**. If your outgoing interface's MTU is lower than the MTU on the local network (like with PPPoE connections), this may prevent you from communicating with such servers. Это правило помагает решить проблему по крайней мере для TCP соединений:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-
to-pmtu
```

Now we set up a rule with the **state** match, identical to the one in the **INPUT** chain:

```
# iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Наш следующий шаг будет включением перенаправления для надежных сетевых интерфейсов и to make all packets pass the цепочек **fw-open**.

```
# iptables -A FORWARD -j fw-interfaces
# iptables -A FORWARD -j fw-open
```

The remaining packets are denied with an **ICMP** message:

```
# iptables -A FORWARD -j REJECT --reject-with icmp-host-unreachable
# iptables -P FORWARD DROP
```

Настройка цепочек fw-interfaces и fw-open

The meaning of the **fw-interfaces** and **fw-open** chains is explained later, when we deal with the **POSTROUTING** and **PREROUTING** chains in the **nat** table, respectively.

Настройка таблицы nat

All over this section, we assume that the outgoing interface (the one with the public internet IP) is **ppp0**. Keep in mind that you have to change the name in all following rules if your outgoing interface has another name.

Настройка цепочки POSTROUTING

Теперь, мы должны определиться, кому разрешать соединяться с Интернетом. Предположим, что у нас есть подсеть '"192.168.0.0/255.255.255.0"' (что означает все адреса, которые имеют форму 192.168.0. *) на '"eth0"'. Мы сначала должны принять машины на этом интерфейсе в FORWARD таблице, именно поэтому мы создали **fw-interfaces** цепочку выше:

```
# iptables -A fw-interfaces -i eth0 -j ACCEPT
```

Теперь, мы должны изменить все исходящие пакеты так, чтобы у них был наш внешний адрес IP как исходный адрес, вместо внутреннего адреса локальной сети. Чтобы сделать это, мы используем средство **MASQUERADE**:

```
# iptables -t nat -A POSTROUTING -s 192.168.0.0/255.255.255.0 -o ppp0 -j
MASQUERADE
```

Не забывайте **-o ppp0** параметр заданные выше, если Вы опустите это, your network will be screwed up.

Let's assume we have another subnet, **10.3.0.0/255.255.0.0** (which means all addresses 10.3.*.*) on interface **eth1**. Мы добавляем те же самые правила как выше снова:

```
# iptables -A fw-interfaces -i eth1 -j ACCEPT
# iptables -t nat -A POSTROUTING -s 10.3.0.0/255.255.0.0 -o ppp0 -j
MASQUERADE
```

Последний шаг включить IP Forwarding (если это еще не сделано):

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Машины от этих подсетей могут теперь использовать вашу новую машину как NAT шлюз. Заметьте что вы также можете установить DNS и DHCP сервер, например **dnsmasq** или комбинацию из **bind** и **dhcpd**, что упрощает настройку сетевых параметров на клиентских машинах. Но это тема не этого руководства.

Настройка цепочки PREROUTING

Sometimes, we want to forward an incoming connection from the gateway to a LAN machine. To do this, we use the **fw-open** chain defined above, as well as the **PREROUTING**chain in the **nat** table

Я приведу два простых примера: Первый пример, мы хотим перенаправить все входящие SSH соединения (порт 22) на SSH сервер машины **192.168.0.5**:

```
# iptables -A fw-open -d 192.168.0.5 -p tcp --dport 22 -j ACCEPT
# iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 22 -j DNAT --to
192.168.0.5
```

The second example will show you how to forward packets to a different port than the incoming port. We want to forward any incoming connection on port **8000** to our web server on **192.168.0.6**, port **80**:

```
# iptables -A fw-open -d 192.168.0.6 -p tcp --dport 80 -j ACCEPT
# iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 8000 -j DNAT --to
192.168.0.6:80
```

The same setup also works with udp packets.

Сохранение правил

Теперь, вы должны сохранить правила. Только в этот раз мы должны включить пересылку IP в **/etc/conf.d/iptables**:

```
# Конфигурация для правил iptables

IPTABLES=/usr/sbin/iptables

IPTABLES_CONF=/etc/iptables/iptables.rules
IPTABLES_FORWARD=1  # включение пересылки IP!!!
```

Сохраняем правила

```
# /etc/rc.d/iptables save
```

и убедитесь, что ваши правила загрузятся при старте системы

```
DAEMONS=(... iptables ...)
```

Работаем с knockd

knockd is a **port knocking** daemon that can provide an added layer of security to your network. The knockd **wiki** provides three example port knocking configurations. These configs can be easily altered to intergrate properly with firewall described here. You should simply substitue the `INPUT` chain specification, with the custom `open` chain used in the firewall.

Для примера:

```
 [options]
       logfile = /var/log/knockd.log
 [opencloseSSH]
       sequence      = 2222:udp,3333:tcp,4444:udp
       seq_timeout   = 15
       tcpflags      = syn,ack
       start_command = /usr/sbin/iptables -A open -s %IP% -p tcp --syn --
dport 22 -j ACCEPT
       cmd_timeout   = 10
       stop_command  = /usr/sbin/iptables -D open -s %IP% -p tcp --syn --
dport 22 -j ACCEPT
```

It is wise to randomly select the ports that you use for the knock sequence. Random.org **link** can help you generate a selection of ports between 1 and 65535. Once you have selected your port range check that you haven't inadvertantly selected a commonly used port; this **port database** can help you check.

Также Смотрите

- **Sharing ppp connection with wlan interface**
- **Internet Share (Русский)**
- **NAT'ing firewall - Share your broadband connection**