Базовая настройка iptables

Правильно настроенный брандмауэр – один из основных аспектов безопасности сервера. Брандмауэр позволяет выбрать для сервера индивидуальные правила и политику, которые ограничат трафик. Брандмауэры, такие как iptables, позволяют определять структурные рамки, в которых будут применяться эти правила.

Данное руководство поможет вам составить список базовых правил брандмауэра для сервера Ubuntu 20.04. Такой список легко расширить, и в дальнейшем вы сможете использовать этот шаблон для создания более сложных правил.

Установка iptables-persistent

Начнем с обновления локального кеша пакетов:

```
sudo apt update
```

Теперь нужно установить пакет iptables-persistent, который позволяет сохранять наборы правил брандмауэра и автоматически использовать их в дальнейшем:

```
sudo apt install iptables-persistent
```

Во время установки пакет предложит вам сохранить текущий набор правил брандмауэра. Чтобы сделать это, выберите yes. Обратите внимание, что нужно выполнить команду netfilter-persistent для запуска службы брандмауэра iptables. Потом мы отредактируем сгенерированные файлы правил.

Команда iptables обрабатывает только трафик IPv4. Для обработки трафика IPv6 существует отдельный инструмент, ip6tables. Правила для IPv4 и IPv6 хранятся в отдельных таблицах и цепочках. Пакет iptables-persistent записывает правила IPv4 в /etc/iptables/rules.v4, а правила IPv6 – в /etc/iptables/rules.v6.

Основные команды iptables

Вывод текущих правил в табличном виде выполняется при помощи вызова команды *iptables* с ключом *-L*:

```
sudo iptables -L
```

Вывод:

```
Output:
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

В выводе представлены 3 стандартных цепочки (*INPUT, OUTPUT, FORWARD*) и действие по умолчанию (*default policy*) для каждой цепочки – *ACCEPT*. Как видим, у нас пока нет боевых правил. По умолчанию iptables идет без предустановленных правил и пропускает весь трафик.
Вывод текущих настроенных правил в виде строк выполняется при помощи вызова команды *iptables* с ключом *-S*:

```
sudo iptables -S
```

Вывод:

```
Output:
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

Ключ *-P* указывает на действие, применяемое к пакету по умолчанию.
Построчный вывод полезен тем, что каждая строка вывода — это полноценная команда в iptables (например, мы можем получить такой вывод уже на настроенном сервере и использовать его для настройки другого сервера, сделав небольшие правки).
Для очистки всех правил используется ключ *-F*:

```
sudo iptables -F
```

Тут стоит отметить важность политик по умолчанию, так как они не удалятся после применения этой команды. Поясним подробней. Допустим у нас есть только удаленный доступ к серверу и в политике iptables по умолчанию выставлено действие *DROP*. В результате чего после очистки всех правил повторное подключение к серверу будет невозможно. Поэтому, если у нас нет физического доступа или консольного подключения к серверу, то перед сбросом всех правил необходимо убедиться, что действие по умолчанию – *ACCEPT*. Это позволит нам удаленно подключиться к серверу, создать разрешающее удаленную сессию первое правило. После этого для повышения безопасности можно уже выставить политику по умолчанию – *DROP*.

Чтобы задать действие по умолчанию *ACCEPT* и выполнить последующую очистку правил вводим:

```
sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -F
```

Создание правил в iptables

Создадим правило, которое позволит принимать текущее SSH-соединение между нами и сервером

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED, RELATED -j ACCEPT
```

- `-A INPUT`: флаг *-A* (append) – добавляет правило в конец цепочки *INPUT*
- `-m conntrack`: этот параметр вызывает модуль *conntrack* для отслеживания информации о соединениях.
- `–ctstate ESTABLISHED, RELATED`: выделяем все соединения в состоянии *ESTABLISHED* (выделяется трафик по уже существующим соединениям) и *RELATED* (трафик по новым соединениям, но связанных с уже открытыми).
- `-j ACCEPT`: действие (target) – то, что будет произведено с пакетом, если он попал под критерий. В нашем случае это *ACCEPT*.

Добавим еще два правила, разрешающие SSH-соединения (по умолчанию на 22 порт) и WEB-подключения на 80 порт.
Приведем синтаксис этих правил:

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Разберем новые параметры:

- `-p tcp`: фильтруем пакеты, использующие протокол TCP
- `–dport`: выделяем TCP-пакеты с портом назначения 22 и 80.

Также есть еще одно правило, которое связано с понятием “петли” (loopback-интерфейса). Используется loopback-интерфейс для взаимодействия служб и приложений в пределах одной локальной системы, без надобности отправления пакетов на сетевой интерфейс.
Правило выглядит так:

```
sudo iptables -I INPUT 1 -i lo -j ACCEPT
```

Разберем его:

- Ключ `-I` (insert) указывает вставить правило в цепочку на 1-ое место. Порядковый номер вставки указывается за именем цепочки *INPUT* (напоминаем, что флаг -A добавляет правило в конец)
- `-i lo` – указываем интерфейс *loopback* и разрешаем трафик через него.

Мы составили 4 правила, на основании которых iptables будет разрешать трафик на сервер. Но поскольку мы не создали ни одного запрещающего правила, то все соединения с сервером будут по-прежнему разрешены.
Есть два способа реализации блокирования нежелательного трафика:

1. Политика запрета по умолчанию.

Достигается написанием правила:

```
sudo iptables -P INPUT DROP
```

Тут надо помнить об опасности потери удаленного доступа к серверу. Желательно его применять, когда есть консольный доступ к серверу.

2. Добавление запрещающего правила в конец цепочки.

```
sudo iptables -A INPUT -j DROP
```

Тут уже нет опасности потери удаленного подключения к серверу, но есть особенность добавления правил. Новые правила необходимо добавлять в цепочку перед запрещающим правилом. Достигается это путем ввода набора из трех правил: удаление запрещающего правила, добавление нового правила, добавление запрещающего правила в конец.

```
sudo iptables -D INPUT -j DROP
sudo iptables -A INPUT "MY_NEW_RULE"
sudo iptables -A INPUT -j DROP
```

Также можно вставить новое правило до запрещающего правила, используя порядковый номер. Для этого сперва узнаем нумерацию имеющихся правил:

```
sudo iptables -L --line-numbers
```

Потом на основании вывода вставляем правило под нужным порядковым номером:

```
sudo iptables -I INPUT 4 "MY_NEW_RULE"
```

Удаление правил iptables

Приведем различные способы удаления правил iptables.

Удаление по имени правила.

В этом случае используется ключ -D после которого следует имя правила в виде строки. Имя правила узнаем при помощи команды, рассмотренной выше:

```
sudo iptables -S
```

Допустим в данном выводе мы нашли правило

```
-A INPUT -i eth0 -p tcp --dport 443 -j ACCEPT
```

и которое подлежит удалению.
Для этого введем:

```
iptables -D INPUT -i eth0 -p tcp --dport 443 -j ACCEPT
```

Обратите внимание, что ключ -A при удалении правила не указывается.

Удаление по номеру правила в цепочке.

Для удаления этим способом нам необходимо знать название цепочки, к которой принадлежит правило и его порядковый номер в ней. Все это определяется командой:

```
sudo iptables -L --line-numbers
```

Вывод:

```
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
2    ACCEPT     all  --  anywhere             anywhere
3    DROP       all  --  anywhere             anywhere             ctstate INVALID
4    UDP        udp  --  anywhere             anywhere             ctstate NEW
5    TCP        tcp  --  anywhere             anywhere             tcp flags:FIN,SYN,RST,ACK/SYN ctstate NEW
6    ICMP       icmp --  anywhere             anywhere             ctstate NEW
7    REJECT     udp  --  anywhere             anywhere             reject-with icmp-port-unreachable
```

И само удаление:

```
sudo iptables -D ИМЯ ЦЕПОЧКИ НОМЕР
```

Например, удалим привило с номером 5 в цепочке INPUT:

```
sudo iptables -D INPUT 5
```

Удаление всех правил iptables

Мы можем удалить все правила в пределах одной цепи. Для этого используется ключ -F. Например, удалим все правила цепочки INPUT:

```
sudo iptables -F INPUT
```

Для удаления правил во всех цепочках вводим команду:

```
sudo iptables -F
```

Сохранение правил iptables

По умолчанию правила iptables не хранятся в постоянной памяти и сбрасываются после перезагрузки.Один из способов сохранения правил — использование пакета iptables-persistent, который мы установили ранее.

Для сохранения правил введите команду:

```
sudo service netfilter-persistent save
```

Система при следующей загрузке использует последние сохраненные правила

В процессе настройки брандмауэра, по разным причинам, возникает необходимость вернуться к заведомо рабочим, испытанным правилам. Сервис *netfilter-persistent* сохраняет их в файле */etc/iptables/rules.v4.*

Восстанавливаем правила следующей командой:

```
iptables-restore < /etc/iptables/rules.v4
```

Примеры настройки iptables

Как заблокировать IP-адрес в iptables

```
sudo iptables -A INPUT -s 192.168.1.100 -j DROP
```

Как разрешить IP-адрес в iptables

Необходимо разрешить весь трафик к серверу для клиента с IP-адресом 192.168.1.100:

```
iptables -A INPUT -s 192.168.1.100 -j ACCEPT
```

Как открыть порт в iptables

Предположим, что политика по умолчанию — блокировать все, что явно не разрешено. Откроем порты веб-сервера для обеспечения работы HTTP протокола — порт 80, и поддержки HTTPS протокола совместно с SSL — порт 443. Также для доступа к серверу по SSH откроем порт 22. Эту задачу можно решить как минимум двумя способами: создать однострочное правило, либо прописать правила по каждому из портов, рассмотрим оба. В одну строку:

```
iptables -A INPUT -p tcp -m multiport --dports 22,80,443 -j ACCEPT
```

При использовании расширения multiport, всегда необходимо использовать критерий *-p tcp* или *-p udp*, таким образом, одной строкой можно указать до 15 разных портов через запятую. Важно не путать критерии *—-dport* и *—-dports*. Первый из них используется для указания одного порта, второй сразу для нескольких, аналогично с *—sport* и *—sports*.

Вариант многострочной записи — для каждого порта свое правило:

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Как закрыть порт в iptables

Необходимость в закрытии порта может возникнуть, когда используется политика по умолчанию **ACCEPT** и доступ к определенному сервису нужно ограничить. Рассмотрим несколько ситуаций.

Закроем доступ к FTP серверу, работающему на 21 порту:

```
iptables -A INPUT -p tcp --dport 21 -j DROP
```

Оставим доступ только себе к SSH серверу, для остальных запретим:

```
iptables -A INPUT -p tcp ! -s 192.168.1.100 --dport 22 -j DROP
```

Здесь *192.168.1.100* — IP-адрес доверенной машины, знак *!* перед ключом *-s* используется для инверсии, т.е. всем, кроме этого адреса доступ закрыт.

Как разрешить или запретить ICMP ping трафик

Разрешить *ping* хоста:

```
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
```

Запретить входящие icmp-пакеты:

```
iptables -A INPUT -p icmp --icmp-type 8 -j DROP
```

Как разрешить трафик на локальном узле

Трафик на локальном сетевом интерфейсе lo должен быть разрешен для корректной работы сервисов, использующих для обмена данными интерфейс локальной петли, например, базы данных, прокси-сервера squid. Поэтому рекомендуется разрешить трафик на вход и выход:

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

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

Версия #2
Кирилл создал 8 февраля 2024 14:09:54
Кирилл обновил 8 февраля 2024 14:51:12