Ко мне иногда обращаются разработчики с просьбами заблокировать доступ к какому-нибудь контейнеру из интернета. Для тех, кто не знает, скажу, что Docker по-умолчанию поднимает свои контейнеры с доступом к ним отовсюду. Никаких ограничений нет и настроить их не очень просто без понимания работы docker и iptables.

Docker облегчил работу современным разработчикам и усложнил жизнь админам. Такая простая задача, как блокировка доступа к какому-то сервису с помощью iptables резко усложнилась. Например, есть у вас одиночный сервер, на нем работают несколько контейнеров docker. Iptables вы не настраивали вообще, своих правил нет.

Тем не менее, набор правил iptables на хосте будет примерно такой.

```
root@      :~# iptables -L -v -n
Chain INPUT (policy ACCEPT 1426K packets, 144M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
 288K  187M DOCKER-USER  all  --  *      *       0.0.0.0/0            0.0.0.0/0
 288K  187M DOCKER-ISOLATION-STAGE-1  all  --  *      *       0.0.0.0/0            0.0.0.0/0
 243K  182M ACCEPT     all  --  *      br-ba0fefda20d6  0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
  206 10764 DOCKER     all  --  *      br-ba0fefda20d6  0.0.0.0/0            0.0.0.0/0
44497 5093K ACCEPT     all  --  br-ba0fefda20d6 !br-ba0fefda20d6  0.0.0.0/0            0.0.0.0/0
   50  3000 ACCEPT     all  --  br-ba0fefda20d6 br-ba0fefda20d6  0.0.0.0/0            0.0.0.0/0
 3243  191K ACCEPT     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
  727 36467 DOCKER     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0
 3551 1761K ACCEPT     all  --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  docker0 docker0  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 1314K packets, 137M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER (2 references)
 pkts bytes target     prot opt in     out     source               destination
    3   172 ACCEPT     tcp  --  !docker0 docker0  0.0.0.0/0            172.17.0.2           tcp dpt:3000
   38  2432 ACCEPT     tcp  --  !br-ba0fefda20d6 br-ba0fefda20d6  0.0.0.0/0            192.168.32.2         tcp dpt:9090
  113  5044 ACCEPT     tcp  --  !br-ba0fefda20d6 br-ba0fefda20d6  0.0.0.0/0            192.168.32.2         tcp dpt:8080
    5   288 ACCEPT     tcp  --  !br-ba0fefda20d6 br-ba0fefda20d6  0.0.0.0/0            192.168.32.3         tcp dpt:5432

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
44497 5093K DOCKER-ISOLATION-STAGE-2  all  --  br-ba0fefda20d6 !br-ba0fefda20d6  0.0.0.0/0            0.0.0.0/0
 3551 1761K DOCKER-ISOLATION-STAGE-2  all  --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
1379M  397G RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain DOCKER-ISOLATION-STAGE-2 (2 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      br-ba0fefda20d6  0.0.0.0/0            0.0.0.0/0
    0     0 DROP       all  --  *      docker0  0.0.0.0/0            0.0.0.0/0
 953M   59G RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain DOCKER-USER (1 references)
 pkts bytes target     prot opt in     out     source               destination
```

Docker создает бриджи, свои цепочки, правила и т.д. Причем управляет всем этим **динамически**. Нельзя просто взять и настроить iptables, как это делаю я в своей статье с помощью скрипта. Такой подход уже не работает.

В крупных проектах эта проблема решается просто — все хосты с контейнерами закрыты внешним шлюзом с фаерволом, который может из себя представлять все, что угодно, но не хост с контейнерами. Реализации могут быть любыми. Важно, что это отдельная сущность.

Тем, у кого контейнеры работают на отдельных хостах, смотрящих напрямую в интернет, надо что-то делать у себя. Я покажу простой пример, как заблокировать доступ к какому-то контейнеру Docker из интернет.

Докер создает отдельную цепочку DOCKER-USER. Она проверяется раньше основной динамической цепочки DOCKER. Если вы хотите настроить свои правила доступа к контейнерам, добавляйте их в цепочку DOCKER-USER. Сервис не будет их трогать и изменять при рестарте или изменениях в контейнерах. Подробнее об этом читайте в документации.

Допустим, у нас есть контейнер с postgresql, который забинден на порт хоста 5432. По-умолчанию, к нему будет доступ из интернета. Чтобы запретить доступ к контейнеру, необходимо применить следующее правило.

```
/sbin/iptables -I DOCKER-USER -i eth0 -p tcp – dport 5432 -j DROP
```

В данном примере eth0 — внешний интерфейс, который смотрит в интернет. Усложним задачу. Вам надо запретить доступ из интернета к контейнеру, но при этом разрешить доступ с определенных ip адресов. Нам нужно применить несколько правил и следить за тем, чтобы разрешающие правила были выше запрещающих. Это можно сделать обычной нумерацией правил.

```
/sbin/iptables -I DOCKER-USER 1 -i eth0 -p tcp – dport 5432 -s 1.2.3.4 -j ACCEPT
/sbin/iptables -I DOCKER-USER 2 -i eth0 -p tcp – dport 5432 -s 5.6.7.8 -j ACCEPT
/sbin/iptables -I DOCKER-USER 3 -i eth0 -p tcp – dport 5432 -j DROP
```

Мы разрешили доступ с ip 1.2.3.4 и 5.6.7.8, всем остальным запретили, причем разрешающие правила поставили выше.

Посмотреть список правил конкретной цепочки с нумерацией можно командой:

```
# iptables -L DOCKER-USER — line-numbers -v -n
```

```
root@y          :~# iptables -L DOCKER-USER --line-numbers -v -n
Chain DOCKER-USER (1 references)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     tcp  --  eth0   *       [redacted]194        0.0.0.0/0            tcp dpt:5432
2        0     0 ACCEPT     tcp  --  eth0   *       [redacted]194        0.0.0.0/0            tcp dpt:9090
3       38  3952 ACCEPT     tcp  --  eth0   *       [redacted]0          0.0.0.0/0            tcp dpt:5432
4       19  1351 ACCEPT     tcp  --  eth0   *       [redacted]0          0.0.0.0/0            tcp dpt:9090
5       10   640 DROP       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0            tcp dpt:5432
6        0     0 DROP       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0            tcp dpt:9090
7    44864   14M RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
root@y          :~#
```


Если вам надо очистить конкретную цепочку с правилами, используйте команду.

```
/sbin/iptables -F DOCKER-USER
```

Если захотите заодно заблокировать посторонним людям доступ по ssh, добавьте следующие правила в цепочку INPUT.

```
/sbin/iptables -I INPUT 1 -p all -m state — state ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -I INPUT 2 -i eth0 -p tcp — dport 22 -s 1.2.3.4 -j ACCEPT
/sbin/iptables -I INPUT 3 -i eth0 -p tcp — dport 22 -s 5.6.7.8 -j ACCEPT
/sbin/iptables -I INPUT 4 -i eth0 -p tcp — dport 22 -j DROP
```

Я рекомендую не лениться и закрывать доступ посторонним ко всему, что им не нужно. Важно делать это в том числе на dev серверах. Эта статья в основном для них. Разработчики часто создают тестовые серверы и особо не следят за ними, хотя там могут быть продуктовые базы и прочая приватная информация. Iptables для них темный лес.

Лучше сразу после создания dev сервера закрыть от посторонних глаз все, что на нем есть, используя простые правила, которые можно применить прямо в

консоли.

Я предложил простое и топорное решение в лоб. Данную задачу можно решить различными способами. Например, правила iptables можно настраивать сразу вместе с контейнером и применять их при запуске. Можно вообще отключить динамическое управление правилами со стороны docker и писать их все самостоятельно. Но это все требует больше времени на настройку.

Чтобы после ребута сервера правила автоматически применились, можно оформить их в простой bash скрипт и запускать его с помощью cron, используя в качестве расписания параметр @reboot. Нужно только учесть один момент. Cron будет запускать скрипт раньше, чем успеет стартануть докер и создать свои цепочки. Надо каким-то образом его подождать. Самый простой способ поставить sleep на 5-10 секунд в скрипте перед применением правил для DOCKER-USER.

Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!

Помогла статья? Подписывайся на telegram канал автора

Анонсы всех статей, плюс много другой полезной и интересной информации, которая не попадает на сайт.