A bash solution for docker and iptables conflict

Lorenzo Garuti · October 14, 2021

Docker Docker Swarm Firewall Iptables Ubuntu Linux

If you've ever tried to setup firewall rules on the same machine where docker daemon is running you may have noticed that docker (by default) manipulate your iptables chains. If you want the full control of your iptables rules this might be a problem.

Docker and iptables

Docker is utilizing the iptables "nat" to resolve packets from and to its containers and "filter" for isolation purposes, by default docker creates some chains in your iptables setup:

```
sudo iptables -L

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
DOCKER-USER  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-1  all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain DOCKER (1 references)
target     prot opt source               destination

Chain DOCKER-INGRESS (0 references)
target     prot opt source               destination
```

```
Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target     prot opt source               destination
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-USER (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

now for example we have the need to expose our nginx container to the world:

```
docker run --name some-nginx -d -p 8080:80 nginx:latest
47a12adff13aa7609020a1aa0863b0dff192fbcf29507788a594e8b098ffe47a

docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS
47a12adff13a   nginx:latest   "/docker-entrypoint.…"   27 seconds ago   Up 24 se
```

and now we can reach our nginx default page:

```
curl -v http://192.168.25.200:8080

*   Trying 192.168.25.200:8080...
* TCP_NODELAY set
* Connected to 192.168.25.200 (192.168.25.200) port 8080 (#0)
> GET / HTTP/1.1
> Host: 192.168.25.200:8080
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.21.1
< Date: Thu, 14 Oct 2021 10:31:38 GMT
< Content-Type: text/html
< Content-Length: 612
< Last-Modified: Tue, 06 Jul 2021 14:59:17 GMT
< Connection: keep-alive
< ETag: "60e46fc5-264"
< Accept-Ranges: bytes
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
...
* Connection #0 to host 192.168.25.200 left intact
```

NOTE the connection test is made using an external machine, not the same machine where the docker container is running.

The “magic” iptables rules added also allow our containers to reach the outside world:

```
docker run --rm nginx curl ipinfo.io/ip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    15  100    15    0     0     94      0 --:--:-- --:--:-- --:--:--    94

1.2.3.4
```

Now check what happened to our iptables rules:

```
iptables -L

...
Chain DOCKER (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             172.17.0.2           tcp dpt:http
...
```

a new rule is appeared, but is not the only rule added to our chains.

To get a more detailed view of our iptables chain we can dump the full iptables rules with *iptables-save*:

```
# Generated by iptables-save v1.8.4 on Thu Oct 14 12:32:46 2021
*mangle
:PREROUTING ACCEPT [33102:3022248]
:INPUT ACCEPT [33102:3022248]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [32349:12119113]
:POSTROUTING ACCEPT [32357:12120329]
COMMIT
# Completed on Thu Oct 14 12:32:46 2021
# Generated by iptables-save v1.8.4 on Thu Oct 14 12:32:46 2021
*nat
:PREROUTING ACCEPT [1:78]
:INPUT ACCEPT [1:78]
:OUTPUT ACCEPT [13:1118]
:POSTROUTING ACCEPT [13:1118]
:DOCKER - [0:0]
:DOCKER-INGRESS - [0:0]
-A PREROUTING -m addrtype --dst-type LOCAL -j DOCKER
-A OUTPUT ! -d 127.0.0.0/8 -m addrtype --dst-type LOCAL -j DOCKER
-A POSTROUTING -s 172.17.0.0/16 ! -o docker0 -j MASQUERADE
-A POSTROUTING -s 172.17.0.2/32 -d 172.17.0.2/32 -p tcp -m tcp --dport 80 -j MAS
-A DOCKER -i docker0 -j RETURN
-A DOCKER ! -i docker0 -p tcp -m tcp --dport 8080 -j DNAT --to-destination 172.1
COMMIT
# Completed on Thu Oct 14 12:32:46 2021
# Generated by iptables-save v1.8.4 on Thu Oct 14 12:32:46 2021
*filter
:INPUT ACCEPT [4758:361293]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [4622:357552]
:DOCKER - [0:0]
:DOCKER-INGRESS - [0:0]
:DOCKER-ISOLATION-STAGE-1 - [0:0]
:DOCKER-ISOLATION-STAGE-2 - [0:0]
:DOCKER-USER - [0:0]
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o docker0 -j DOCKER
```

```
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A DOCKER -d 172.17.0.2/32 ! -i docker0 -o docker0 -p tcp -m tcp --dport 80 -j A
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -j RETURN
-A DOCKER-ISOLATION-STAGE-2 -o docker0 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -j RETURN
-A DOCKER-USER -j RETURN
COMMIT
# Completed on Thu Oct 14 12:32:46 2021
```

in our dump we can see some other rules added by docker:

DOCKER-INGRESS (nat table)

```
-A POSTROUTING -s 172.17.0.0/16 ! -o docker0 -j MASQUERADE
-A POSTROUTING -s 172.17.0.2/32 -d 172.17.0.2/32 -p tcp -m tcp --dport 80 -j MAS
-A DOCKER -i docker0 -j RETURN
-A DOCKER ! -i docker0 -p tcp -m tcp --dport 8080 -j DNAT --to-destination 172.1
```

DOCKER-USER (filter table)

```
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A DOCKER -d 172.17.0.2/32 ! -i docker0 -o docker0 -p tcp -m tcp --dport 80 -j A
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -j RETURN
-A DOCKER-ISOLATION-STAGE-2 -o docker0 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -j RETURN
-A DOCKER-USER -j RETURN
```

to explore in detail how iptables and docker work:

- Docker docs
- Docker forum question
- gist from x-yuri

- argus-sec.com post

The problem

But what happen if we stop and restart our firewall?

```
systemctl stop ufw|firewalld # <- the service (ufw or firewalld) may change from
systemctl stop ufw|firewalld


curl -v http://192.168.25.200:8080
*   Trying 192.168.25.200:8080...
* TCP_NODELAY set


docker run --rm nginx curl ipinfo.io/ip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:06 --:--:--     0
```

we can see that:

- our container is not reachable from the outside world
- our container is not able to reach internet

The solution

The solution for this problem is a simple bash script (combined to an awk script) to manage our iptables rules. In short the script parse the output of the *iptables-save* command and preserve a set of chains. The chains preserved are:

for table nat:

- POSTROUTING
- PREROUTING
- DOCKER
- DOCKER-INGRESS
- OUTPUT

for table filter:

- FORWARD
- DOCKER-ISOLATION-STAGE-1
- DOCKER-ISOLATION-STAGE-2
- DOCKER
- DOCKER-INGRESS
- DOCKER-USER

Install iptables-docker

The first step is to clone this repository

Local install (sh)

NOTE this kind of install use a static file (src/iptables-docker.sh). By default **only** ssh access to local machine is allowd. To allow specific traffic you have to edit manually this file with your own rules:

```
    # Other firewall rules
    # insert here your firewall rules
    $IPT -A INPUT -p tcp --dport 1234 -m state --state NEW -s 0.0.0.0/0 -j ACCEP
```

NOTE2 if you use a swarm cluster uncomment the lines under *Swarm mode - uncomment to enable swarm access (adjust source lan)* and adjust your LAN subnet

To install iptables-docker on a local machine, clone this repository and run *sudo sh install.sh*

```
sudo sh install.sh

Set iptables to iptables-legacy
Disable ufw,firewalld
Synchronizing state of ufw.service with SysV service script with /lib/systemd/sy
Executing: /lib/systemd/systemd-sysv-install disable ufw
Failed to stop firewalld.service: Unit firewalld.service not loaded.
Failed to disable unit: Unit file firewalld.service does not exist.
Install iptables-docker.sh
Create systemd unit
Enable iptables-docker.service
Created symlink /etc/systemd/system/multi-user.target.wants/iptables-docker.serv
start iptables-docker.service
```

Automated install (ansible)

You can also use ansible to deploy iptables-docker everywhere. To do this adjust the settings under group_vars/main.yml.

Label	Default	Description
`docker_preserve`	`yes`	Preserve docker iptables rules
`swarm_enabled`	`no`	Tells to ansible to open the required ports for the swarm cluster
`ebable_icmp_messages`	`yes`	Enable response to ping requests
`swarm_cidr`	`192.168.1.0/24`	Local docker swarm subnet
`ssh_allow_cidr`	`0.0.0.0/0`	SSH alloed subnet (default everywhere)
`iptables_allow_rules`	`[]`	List of dict to dynamically open ports. Each dict has the following key: desc, proto, from, port. See group_vars/all.yml for examples
`iptables_docker_uninstall`	`no`	Uninstall iptables-docker

Now create the inventory (hosts.ini file) or use an inline inventory and run the playbook:

```
ansible-playbook -i hosts.ini site.yml
```

Usage

To start the service use:

```
sudo systemctl start iptables-docker

or

sudo iptables-docker.sh start
```

To stop the srevice use:

```
sudo systemctl stop iptables-docker

or

sudo iptables-docker.sh stop
```

Test iptables-docker

Now if you turn off the firewall with *sudo systemctl stop iptables-docker* and if you check the iptable-save output, you will see that the docker rules are still there:

```
sudo iptables-save

# Generated by iptables-save v1.8.4 on Thu Oct 14 15:52:30 2021
*mangle
:PREROUTING ACCEPT [346:23349]
:INPUT ACCEPT [346:23349]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [340:24333]
:POSTROUTING ACCEPT [340:24333]
COMMIT
# Completed on Thu Oct 14 15:52:30 2021
# Generated by iptables-save v1.8.4 on Thu Oct 14 15:52:30 2021
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:DOCKER - [0:0]
:DOCKER-INGRESS - [0:0]
-A PREROUTING -m addrtype --dst-type LOCAL -j DOCKER
-A OUTPUT ! -d 127.0.0.0/8 -m addrtype --dst-type LOCAL -j DOCKER
-A POSTROUTING -s 172.17.0.0/16 ! -o docker0 -j MASQUERADE
-A POSTROUTING -s 172.17.0.2/32 -d 172.17.0.2/32 -p tcp -m tcp --dport 80 -j MAS
-A DOCKER -i docker0 -j RETURN
-A DOCKER ! -i docker0 -p tcp -m tcp --dport 8080 -j DNAT --to-destination 172.1
COMMIT
# Completed on Thu Oct 14 15:52:30 2021
# Generated by iptables-save v1.8.4 on Thu Oct 14 15:52:30 2021
*filter
```

```
:INPUT ACCEPT [357:24327]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [355:26075]
:DOCKER - [0:0]
:DOCKER-INGRESS - [0:0]
:DOCKER-ISOLATION-STAGE-1 - [0:0]
:DOCKER-ISOLATION-STAGE-2 - [0:0]
:DOCKER-USER - [0:0]
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A DOCKER -d 172.17.0.2/32 ! -i docker0 -o docker0 -p tcp -m tcp --dport 80 -j A
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -j RETURN
-A DOCKER-ISOLATION-STAGE-2 -o docker0 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -j RETURN
-A DOCKER-USER -j RETURN
COMMIT
# Completed on Thu Oct 14 15:52:30 2021
```

our container is still accesible form the outside:

```
 curl -v http://192.168.25.200:8080
*   Trying 192.168.25.200:8080...
* TCP_NODELAY set
* Connected to 192.168.25.200 (192.168.25.200) port 8080 (#0)
> GET / HTTP/1.1
> Host: 192.168.25.200:8080
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.21.1
< Date: Thu, 14 Oct 2021 13:53:33 GMT
< Content-Type: text/html
< Content-Length: 612
< Last-Modified: Tue, 06 Jul 2021 14:59:17 GMT
< Connection: keep-alive
< ETag: "60e46fc5-264"
< Accept-Ranges: bytes
```

and our container can reach internet:

```
docker run --rm nginx curl ipinfo.io/ip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    15  100    15    0     0     94      0 --:--:-- --:--:-- --:--:--    94
my-public-ip-address
```

Important notes

Before install iptables-docker please read this notes:

- both local instal and ansible install configure your system to use **iptables-legacy**
- by default **only** port 22 is allowed
- ufw and firewalld will be permanently **disabled**
- filtering on all docker interfaces is disabled

Docker interfaces are:

- vethXXXXXX interfaces
- br-XXXXXXXXXXX interfaces
- docker0 interface
- docker_gwbridge interface

Extending iptables-docker

You can extend or modify iptables-docker by editing:

- src/iptables-docker.sh for the local install (sh)
- roles/iptables-docker/templates/iptables-docker.sh.j2 template file for the automated install (ansible)

Uninstall

Local install (sh)

Run uninstall.sh

Automated install (ansible)

set the variable “iptables_docker_uninstall” to “yes” into group_vars/all.yml and run the playbook.

Share: Twitter, Facebook