1 Firewall Lab Writeup

Ethical Hacking 2021/22, University of Padua

Eleonora Losiouk, Alessandro Brighente, Denis Donadel, Gabriele Orazi

---

General advise: you can use `dmesg -wH` in a terminal to have a real-time monitor of the log file.

1.1 Task 1.B

1. Make the file, add to the kernel and look at how the DNS request is blocked.
2. To trigger each action you have to use the right command, otherwise you will not be able to see the filter working.
   - `NF_INET_PRE_ROUTING`: a ping is enough and you can see packets in a pre-routing stage.
   - `NF_INET_LOCAL_IN`: a ping is enough and you can see reply from the remote host you are pinging if it is up. If you are pinging an host which is down (try with `1.2.3.4`), you will not have any response and so no packets will be logged.
   - `NF_INET_FORWARD`: to see some results with the forwarding you have to act as a router.
   - `NF_INET_LOCAL_OUT`: ping is enough and you can see packets that you are sending to someone else.
   - `NF_INET_POST_ROUTING`: a ping is enough and you can see packets in a post-routing stage.
3. You can adapt previous code to block all the packets directed to TCP/23 and all the ICMP request. To test that the filters are working, you can use the containers as presented, or you can simply try to `ping` and `telnet` the `localhost`.

```
[  +4.302023] *** LOCAL_OUT
[  +0.000007]     127.0.0.1  --> 127.0.0.1 (TCP)
[  +0.000001] *** LOCAL_OUT
[  +0.000002]     127.0.0.1  --> 127.0.0.1 (TCP)
[  +0.000022] *** POST_ROUTING
[  +0.000002]     127.0.0.1  --> 127.0.0.1 (TCP)
[  +0.000026] *** Dropping Telnet request: 127.0.0.1 (TCP), port 23
[  +9.133565] *** LOCAL_OUT
[  +0.000007]     127.0.0.1  --> 127.0.0.1 (ICMP)
[  +0.000002] *** LOCAL_OUT
[  +0.000002]     127.0.0.1  --> 127.0.0.1 (ICMP)
[  +0.000020] *** POST_ROUTING
[  +0.000002]     127.0.0.1  --> 127.0.0.1 (ICMP)
[  +0.000025] *** Dropping ICMP 127.0.0.1
```

1.2 Task 2.A

If you run all the commands you will not be able to `telnet` into the router but you can still `ping` it.

The first two rules whitelist the `icmp` protocol (used for `ping`) both in input and output, while the second two dismiss all in/out packets for all the rest of the protocols.

1.3 Task 2.B

```
# accept ping from internal to external (3), you have to specify both request and reply
iptables -A FORWARD -i eth1 -o eth0 -p icmp --icmp-type echo-request -j ACCEPT
iptables -A FORWARD -o eth1 -i eth0 -p icmp --icmp-type echo-reply -j ACCEPT

# drop all the rest
iptables -A FORWARD -p icmp -j DROP
```

No actions required for 2. since pinging the router do not involve the FORWARD.

Here the FORWARD table:

```
$ iptables -L FORWARD -nv
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    3   252 ACCEPT     icmp --  eth1   eth0    0.0.0.0/0            0.0.0.0/0           icmptype 8
    3   252 ACCEPT     icmp --  eth0   eth1    0.0.0.0/0            0.0.0.0/0           icmptype 0
    3   252 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0
```

As a proof of 1. and 2., here you can find ping attempts from hostA (`10.9.0.11`):

```
root@81b43d6ea197:/# ping 192.168.60.6
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
^C
--- 192.168.60.6 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2029ms

root@81b43d6ea197:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.141 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.179 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.138 ms
^C
--- 192.168.60.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.138/0.152/0.179/0.018 ms
```

For point 3. and 4., you can see how internal `192.168.60.6` tries to reach `10.9.0.5`:

```
root@601b246cd285:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.239 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.168 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.166 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.166/0.191/0.239/0.033 ms
```

1.4 Task 2.C

```
iptables -A FORWARD -i eth0 -p tcp -d 192.168.60.5 --dport 23 -j ACCEPT # Accept request to only one te
iptables -A FORWARD -i eth0 -p tcp --dport 23 -j DROP # ... and drop all the others (2.)
# Internal hosts can access all the internal servers works by default (3.)
iptables -A FORWARD -o eth0 -p tcp --dport 23 -j DROP # Drop requests to external servers (4.)
```

Here the iptables FORWARD:

```
root@0f5a858aee5c:/# iptables -L FORWARD -nv
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   18  1021 ACCEPT     tcp  --  eth0   *       0.0.0.0/0            192.168.60.5        tcp dpt:23
    2   120 DROP       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:23
    2   120 DROP       tcp  --  *      eth0    0.0.0.0/0            0.0.0.0/0           tcp dpt:23
```

1.5 Task 3.A

1. It is possible to see the connection as: `icmp 1 29 src=10.9.0.5 dst=192.168.60.6 type=8 code=0 id=106 src=192.168.60.6 dst=10.9.0.5 type=0 code=0 id=106 mark=0 use=1` where the third field (29) is a countdown to the deletion of the connection (in seconds). Since, while executing a ping the values is 29 we can assume that the connection will last 30 seconds from the reception of the last ICMP packet. See more on this here.
2. As before: `udp 17 27 src=10.9.0.5 dst=192.168.60.5 sport=43068 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=43068 mark=0 use=1` Even here we can see that the third field indicates the duration of the connection. Also here we can assume to have 30 seconds.
3. Also for TCP: `tcp 6 431999 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=38628 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=38628 [ASSURED] mark=0 use=1` But in this case we have a bigger countdown before removing the connection (432000 seconds).

1.6 Task 3.B

This task can be done easily using the connection tracking mechanisms to track the whole connections which starts from inside the network:

```
# Accept request to only one telnet server... (1. and 2.)
iptables -A FORWARD -i eth0 -p tcp -d 192.168.60.5 --dport 23 -j ACCEPT
# ... and drop all the others (2.)
iptables -A FORWARD -i eth0 -p tcp --dport 23 -j DROP
# Internal hosts can access all the internal servers works by default (3.)
# activate conntrack
iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
# accept syn from the outside to start connections on port 23
iptables -A FORWARD -p tcp -i eth0 -o eth1 --dport 23 --syn -m conntrack --ctstate NEW -j ACCEPT
# Drop all the request starting from the outside
iptables -A FORWARD -i eth0 -p tcp --dport 23 -j DROP
```

2 Task 4

With only the first rule nothing happens since we do not generate any rule to drop packets which exceed the limit. Instead, with the second rule we set a rule which drop all the packets which have not been accepted by the preceding rule (e.g., all the packets which exceed the rule of `10/minutes`).

We can see an example of ~2 minutes of ping in the following:

```
root@18af2fe0e0eb:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.340 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.262 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.250 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.268 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.311 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=63 time=0.306 ms <--
64 bytes from 10.9.0.5: icmp_seq=13 ttl=63 time=0.266 ms <--
64 bytes from 10.9.0.5: icmp_seq=19 ttl=63 time=0.330 ms
64 bytes from 10.9.0.5: icmp_seq=25 ttl=63 time=0.636 ms
64 bytes from 10.9.0.5: icmp_seq=31 ttl=63 time=0.259 ms
```

```
64 bytes from 10.9.0.5: icmp_seq=37 ttl=63 time=0.352 ms
64 bytes from 10.9.0.5: icmp_seq=43 ttl=63 time=0.299 ms
64 bytes from 10.9.0.5: icmp_seq=48 ttl=63 time=0.268 ms
64 bytes from 10.9.0.5: icmp_seq=54 ttl=63 time=0.251 ms
64 bytes from 10.9.0.5: icmp_seq=60 ttl=63 time=0.117 ms
64 bytes from 10.9.0.5: icmp_seq=66 ttl=63 time=0.270 ms
64 bytes from 10.9.0.5: icmp_seq=72 ttl=63 time=0.257 ms
64 bytes from 10.9.0.5: icmp_seq=78 ttl=63 time=0.254 ms
64 bytes from 10.9.0.5: icmp_seq=84 ttl=63 time=0.266 ms
64 bytes from 10.9.0.5: icmp_seq=89 ttl=63 time=0.286 ms
^C
--- 10.9.0.5 ping statistics ---
93 packets transmitted, 20 received, 78.4946% packet loss, time 94206ms
```

As expected, after the first packets got a response, only some of the following ones got a reply to meet the limit rule.

3 Task 5

1. You can add more rule similar to the one presented but changing the `--to-destination` paramenter to the new IPs.
2. You can use the same role, changing the destination IP address and setting a good probability. You have to take into consideration that the rules will follow the order of insertion. In other world, if you set 0.33 the percentage to get the first redirection, you will have a probability of 0.67 to get to the second rule.

3.1 Extra

- This graph from Wikipedia is a good resource to understand how chains are traversed.

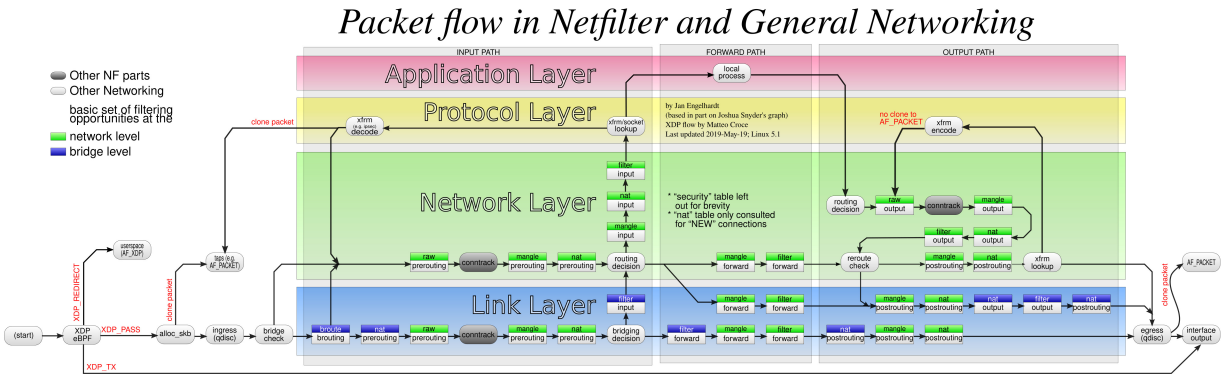

Figure 1: netfilter

- More informations on the `conntrack -L` entries here.