

Ch3 - Firewall Exploration Lab

3.2 Task 1.B: Implement a Simple Firewall Using Netfilter

2. Using your experiment results to help explain at what condition will each of the hook function be invoked:

When a network packet is received on a network device, it first passes through the Prerouting hook. This is where the routing decision takes place. The kernel decides whether the packet is destined for a local process (e.g., a listening socket on a server in this system) or whether to forward it (system operates as a router). In the first case, the packet passes the Input hook and is then handed over to the local process. If the packet is destined to be forwarded, it traverses the Forward hook and then a final Postrouting hook before being sent out on a network device. For packets that are generated locally (e.g., by a client or server process that likes sending things out), they must first pass the Output hook and then the Postrouting hook before being sent out on a network device.

By the following output logs of the LKM, we can check the ordered flow of hooks just described.

```
[ 5576.528370] *** PRE_ROUTING
[ 5576.528372] 127.0.0.1 --> 127.0.0.53 (UDP)
[ 5576.528375] *** LOCAL_IN
[ 5576.528377] 127.0.0.1 --> 127.0.0.53 (UDP)
[ 5576.528621] *** LOCAL_OUT
[ 5576.528626] 10.0.2.15 --> 10.0.2.3 (UDP)
[ 5576.528635] *** POST_ROUTING
[ 5576.528637] 10.0.2.15 --> 10.0.2.3 (UDP)
[ 5576.591324] *** PRE_ROUTING
[ 5576.591361] 10.0.2.3 --> 10.0.2.15 (UDP)
[ 5576.591381] *** LOCAL_IN
[ 5576.591394] 10.0.2.3 --> 10.0.2.15 (UDP)
[ 5576.591584] *** LOCAL_OUT
[ 5576.591585] 127.0.0.53 --> 127.0.0.1 (UDP)
[ 5576.591633] *** POST_ROUTING
[ 5576.591635] 127.0.0.53 --> 127.0.0.1 (UDP)
[ 5576.591641] *** PRE_ROUTING
[ 5576.591643] 127.0.0.53 --> 127.0.0.1 (UDP)
[ 5576.591646] *** LOCAL_IN
[ 5576.591648] 127.0.0.53 --> 127.0.0.1 (UDP)
[ 5576.591873] *** LOCAL_OUT
[ 5576.591877] 10.0.2.15 --> 152.199.19.160 (TCP)
[ 5576.591885] *** POST_ROUTING
```

3. LKM blocks ping and Telnet packets:

```
[ 852.359451] 10.0.2.4 --> 20.189.173.12 (TCP)
[ 852.360034] *** LOCAL_OUT
[ 852.360038] 10.0.2.4 --> 20.189.173.12 (TCP)
[ 852.360044] *** POST_ROUTING
[ 852.360046] 10.0.2.4 --> 20.189.173.12 (TCP)
[ 852.360232] *** PRE_ROUTING
[ 852.360236] 20.189.173.12 --> 10.0.2.4 (TCP)
[ 852.360242] *** LOCAL_IN
[ 852.360244] 20.189.173.12 --> 10.0.2.4 (TCP)
[ 852.387922] *** PRE_ROUTING
[ 852.387925] 20.189.173.12 --> 10.0.2.4 (TCP)
[ 852.387936] *** LOCAL_IN
[ 852.387938] 20.189.173.12 --> 10.0.2.4 (TCP)
[ 852.387951] *** LOCAL_OUT
[ 852.387952] 10.0.2.4 --> 20.189.173.12 (TCP)
[ 852.387955] *** POST_ROUTING
[ 852.387957] 10.0.2.4 --> 20.189.173.12 (TCP)
[ 857.214320] *** Dropping 10.9.0.1 (ICMP)
[ 858.245832] *** Dropping 10.9.0.1 (ICMP)
[ 866.176153] *** Dropping 10.9.0.1 (TCP-Telnet), port 23
[ 867.205811] *** Dropping 10.9.0.1 (TCP-Telnet), port 23
```

```
root@3d5e0e71ea26:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
^C
--- 10.9.0.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1032ms

root@3d5e0e71ea26:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
root@3d5e0e71ea26:/#
```

5.1 Task 2.A: Protecting the Router

Please report your observation and explain the purpose for each rule:

```
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

The previous command append the rule to the end of the INPUT chain, filter table. The protocol of the rule is the ICMP, the type of ICMP packets to check is the echo-request one. The target of the rule, i.e., what to do if the packet matches it, is to leave them to pass.

```
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

The previous command append the rule to the end of the OUTPUT chain, filter table. The protocol of the rule is the ICMP, the type of ICMP packets to check is the echo-reply one. The target of the rule, i.e., what to do if the packet matches it, is to leave them to pass.

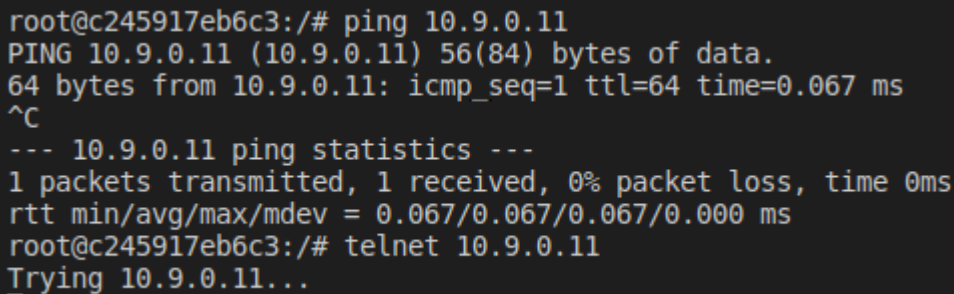
```
iptables -P OUTPUT DROP
```

The previous command set the policy for the built-in (non-user-defined) OUTPUT chain to the given target, filter table. The policy target is to DROP.

```
iptables -P INPUT DROP
```

The previous command set the policy for the built-in (non-user-defined) INPUT chain to the given target, filter table. The policy target is to DROP.

1. Can you ping the router? Yes, I can. The firsts two commands launched on the router create two rules for the INPUT and OUTPUT chains that allow ICMP echo-requests and ICMP echo-replies packets respectively to pass. Since iptables entries are applied in order, these two specified rules will be applied before the default behavior.
2. Can you telnet into the router? No, I can't. Since the last two commands launched on the router set as default behaviour for the INPUT and OUTPUT chains to drop every kind of packet that doesn't match any of the previously specified rules in the tables.



```
root@c245917eb6c3:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.067 ms
^C
--- 10.9.0.11 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.067/0.067/0.067/0.000 ms
root@c245917eb6c3:/# telnet 10.9.0.11
Trying 10.9.0.11...
```

5.2 Task 2.B: Protecting the Internal Network

1. Outside hosts cannot ping internal hosts.
2. Internal hosts can ping outside hosts.

```
iptables -A FORWARD -i eth0 -o eth1 -p icmp --icmp-type echo-reply -j
ACCEPT

iptables -A FORWARD -i eth1 -o eth0 -p icmp --icmp-type echo-request -j
ACCEPT
```

4. All other packets between the internal and external networks should be blocked.

```
iptables -P FORWARD DROP
```

5.3 Task 2.C: Protecting Internal Server

1. All the internal hosts run a telnet server (listening to port 23). Outside hosts can only access the telnet server on 192.168.60.5, not the other internal hosts.

```
iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 23 -d 192.168.60.5 -j ACCEPT
```

2. Outside hosts cannot access other internal servers.
3. Internal hosts can access all the internal servers.
4. Internal hosts cannot access external servers.

```
iptables -A FORWARD -p tcp --dport 23 -j DROP
```

5. In this task, the connection tracking mechanism is not allowed. It will be used in a later task.

6.1 Task 3.A: Experiment with the Connection Tracking (Optional)

1. ICMP experiment: Run the following command and check the connection tracking information on the router. Describe your observation. How long is the ICMP connection state be kept? After 29 seconds, the ICMP state is removed from the tracking mechanism. The remaining time is written as the 2nd value of the output returned. The first value is the id of the protocol used, while the other values describe the fields of the ICMP packets exchanged.

```
icmp      1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=50  
src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=50 mark=0 use=1  
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

2. UDP experiment: Run the following command and check the connection tracking information on the router. Describe your observation. How long is the UDP connection state be kept? If 10.9.0.5 sends various UDP packets, but 192.168.60.5 doesn't send anything, then after 29 seconds the UDP state is removed from the tracking mechanism. Also, in this case a writing UNREPLIED will compare in the tracking system for this state.

```
udp       17 27 src=10.9.0.5 dst=192.168.60.5 sport=35944 dport=9090  
[UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=35944 mark=0  
use=1  
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

If 10.9.0.5 sends various UDP packets and also 192.168.60.5 sends some UDP packets to it, then after 119 seconds the UDP state is removed from the tracking mechanism.

```
udp      17 115 src=10.9.0.5 dst=192.168.60.5 sport=54197 dport=9090
src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=54197 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

3. TCP experiment: Run the following command and check the connection tracking information on the router. Describe your observation. How long is the TCP connection state be kept? After 431999 seconds that the TCP connection is established, the state is removed from the tracking mechanism.

```
tcp      6 431997 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=37564
dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=37564 [ASSURED]
mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

6.2 Task 3.B: Setting Up a Stateful Firewall (Optional)

Please rewrite the firewall rules in Task 2.C, but this time, we will add a rule allowing internal hosts to visit any external server (this was not allowed in Task 2.C).

1. All the internal hosts run a telnet server (listening to port 23). Outside hosts can only access the telnet server on 192.168.60.5, not the other internal hosts.

```
iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j
ACCEPT

iptables -A FORWARD -p tcp -i eth0 -d 192.168.60.5 --dport 23 --syn -m
conntrack --ctstate NEW -j ACCEPT
```

2. Outside hosts cannot access other internal servers.

```
iptables -A FORWARD -p tcp -i eth0 --dport 23 -j DROP
```

3. Internal hosts can access all the internal servers.
4. Internal hosts can access external servers.

```
iptables -A FORWARD -p tcp -i eth1 --dport 23 --syn -m conntrack --ctstate
NEW -j ACCEPT
```

5. In this task, the connection tracking mechanism is allowed.

After you write the rules using the connection tracking mechanism, think about how to do it without using the connection tracking mechanism (you do not need to actually implement them). Based on these two sets

of rules, compare these two different approaches, and explain the advantage and disadvantage of each approach. When you are done with this task, remember to clear all the rules.

The only way to allow packets of an already established connection in a stateless firewall is to allow all kinds of different packets inside the connections just for specific hosts in the network and/or for specific ports. But in this way the administrator has to insert a lot of rules for each specific host if he wants to maintain the security, otherwise he should allow every kind of packet sacrificing security. In the first case, we have a lot of rules to insert in the router w.r.t. the statefull firewall, while in the last case everyone can send any kind of packet toward the router, so it is no more useful.

7 Task 4: Limiting Network Traffic (Optional)

Please run the following commands on router, and then ping 192.168.60.5 from 10.9.0.5. Describe your observation. Please conduct the experiment with and without the second rule, and then explain whether the second rule is needed or not, and why.

```
iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT
```

```
iptables -A FORWARD -s 10.9.0.5 -j DROP
```

- Using just the first rule, is like not really using it, since the effect is the same when the router has no filter rules. Indeed, the default behaviour is to let every packet to pass the router.
- Using both the commands, the first time this rule is reached by the ICMP packet, the packet will be accepted; in fact, since the default burst is 5, the first five packets will be accepted. After this, it will be 6 seconds (60/10 seconds) before a packet will be accepted from this rule. The other packets that reach the router before this time will be dropped thanks to the second command. Also, every 6 seconds which passes without matching a packet, one of the burst will be regained; if no packets hit the rule for 30 seconds (6*5 seconds), the burst will be fully recharged; back where we started.

8 Task 5: Load Balancing (Optional)

1. Using the nth mode (round-robin). Please provide some explanation for the rules.

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth -  
-every 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080
```

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth -  
-every 3 --packet 1 -j DNAT --to-destination 192.168.60.6:8080
```

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth -  
-every 3 --packet 2 -j DNAT --to-destination 192.168.60.7:8080
```

- With the 1st rule in place, if you send a UDP packet to the router's 8080 port, you will see that the 1st of the three packets gets to 192.168.60.5.
- With the 2nd rule in place, if you send a UDP packet to the router's 8080 port, you will see that the 2nd of the three packets gets to 192.168.60.6.

- With the 3rd rule in place, if you send a UDP packet to the router's 8080 port, you will see that the 3rd of the three packets gets to 192.168.60.7.

2. Using the random mode. Please provide some explanation for the rules.

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.33 -j DNAT --to-destination 192.168.60.5:8080
```

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.50 -j DNAT --to-destination 192.168.60.6:8080
```

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 1.0 -j DNAT --to-destination 192.168.60.7:8080
```

We have to take into consideration that the rules will follow the order of insertion. So:

- With the 1st rule in place, if you send a UDP packet to the router's 8080 port, select a matching packet with probability 0.33 and forward it to 192.168.60.5.
- With the 2nd rule in place, if you send a UDP packet to the router's 8080 port, select a matching packet with probability 0.50 and forward it to 192.168.60.6.
- With the 3rd rule in place, if you send a UDP packet to the router's 8080 port, select a matching packet with probability 1.0 and forward it to 192.168.60.7.