Workshop : Introduction to Network Functions

Abstract

In this workshop, you will get introduced to creating Firewall and Network Address Translator (NAT) network functions in software. You will learn how network functions are typically deployed in an enterprise network.

1 Expected outcome

In this workshop, the student would learn about:

1. Using iptables and route Linux utilities to implement Firewall and NAT

2 Specification

This workshop is divided into two parts : (1) firewall , and (2) NAT. For both parts, the topology has 2 end hosts that are connected together through a network function (also called *middlebox*). In the first part, the middlebox is a Firewall, while in the second part it is a NAT. End-hosts and middlebox are supposed to perform different functions, hence we create two different images for them.

2.1 Create Docker image of end hosts

The endpoint hosts will be implemented as Docker containers. The Dockerfile for their image can be found here : Endpoint Docker[1]. Use the following command to create the Docker image named "endpoint".

```
$ docker build -t endpoint:latest .
```

2.2 Create Docker image for the middlebox

The network function will also be implemented as a Docker container. For this workshop, we will use the same Docker image for both the network functions and selectively configure the container to behave either as a NAT or a firewall.

The Dockerfile for their image can be found here : Middlebox Docker[2]. Use the following command to create the Docker image named "nf".

```
$ docker build -t nf:latest .
```

2.3 Firewall

Implement the network topology shown in Figure 1.

1. Create Open vSwitch virtual switches (bridges).

   ```
   $ sudo ovs-vsctl add-br ovs-br1
   ```

2. Create Docker containers for the endpoints

   ```
   $ docker run -d --privileged --name=int --net=none endpoint:latest tail -f /dev/null
   $ docker run -d --privileged --name=ext --net=none endpoint:latest tail -f /dev/null
   ```

3. Create Docker container for the network function

   ```
   $ docker run -d --privileged --name=fw --net=none nf:latest tail -f /dev/null
   ```

4. Connect the endpoints to OVS bridges

   ```
   $ sudo ovs-docker add-port ovs-br1 eth0 int --ipaddress=192.168.1.2/24
   ```

   Similarly connect the container *ext* (that represents external host) to the switch *ovs-br2* and assign it the IP address as shown in Figure 1. Similarly create 2 ports on fw container with the appropriate IP addresses as shown in Figure 1.

5. Now setup the routes on the endpoint hosts.

   ```
   $ docker exec int route add -net 145.12.131.0/24 gw 192.168.1.1 dev eth0
   $ docker exec ext route add -net 192.168.1.0/24 gw 145.12.131.74 dev eth0
   ```

[1] https://github.com/harshitgupta1337/sicc_nfv/blob/master/workshop_1/Dockerfile
[2] https://github.com/harshitgupta1337/sicc_nfv/blob/master/workshop_2/Dockerfile

These routes enable to endpoint hosts to know how to send packets to the other endpoint.

6. Enable network forwarding on network function container

```
$ docker exec fw sysctl net.ipv4.ip_forward=1
```

7. Configure Firewall to block incoming packets with destination port 22 (incoming SSH connections)

```
$ docker exec fw iptables -A FORWARD -i eth1 -p tcp --destination-port 22 -j DROP
```

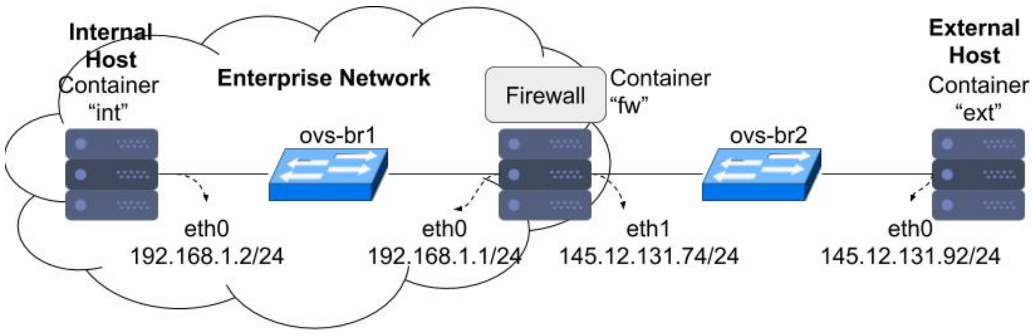

Figure 1: Schematic of network topology for the Firewall exercise.

2.4 Network Address Translation

Implement the network topology shown in Figure 2. Perform the following steps after steps 1-5 from the Firewall exercise. Note that in this topology, the middlebox container is named *nat*.

6. Enable network forwarding on network function container

```
$ docker exec nat sysctl net.ipv4.ip_forward=1
```

7. Use *iptables* to setup NAT

```
$ docker exec nat iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
$ docker exec nat iptables -A FORWARD -i eth0 -o eth1 -m state --state \
  ESTABLISHED,RELATED -j ACCEPT
$ docker exec nat iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

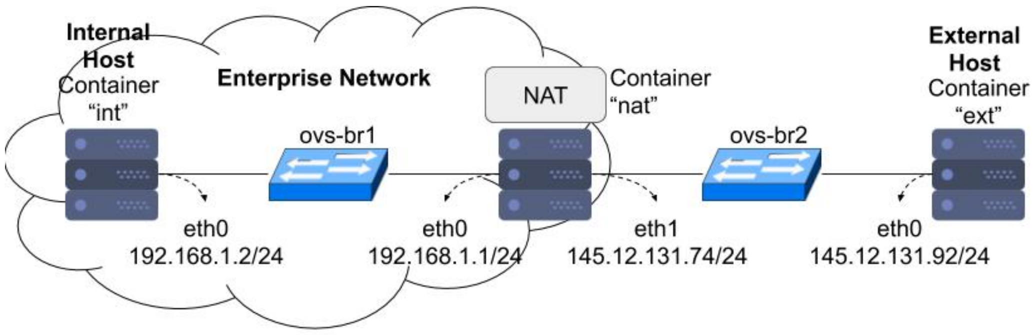

Figure 2: Schematic of network topology for the NAT exercise.

3 Requirements

Verify that the following requirements are met.

- The topology needs to correctly implemented. Both the endpoints (internal and external hosts) should be able to ping (ICMP) each other. You can use the Linux ping utility to test this.
- Use *tcpdump* on all network interfaces on all Docker containers to record packets being sent from and received on that interface. In order to generate traffic from and to specific ports, use the *nc* command-line tool (usage of *nc* can be found in Section 5)

  For the Firewall exercise, record traffic for two types of activity.

  1. Traffic originating from External Host to Internal Host for port 22. This flow will be blocked by the Firewall.
  2. Traffic originating from External Host to Internal Host for port 80. This flow will not be blocked by the Firewall.

  For the NAT exercise, record traffic for one activity.

  1. Traffic originating from Internal Host to External Host for port 80. Make sure to record the transformation of IP addresses by the NAT middlebox.

4 Deliverables

- Two executable scripts *setup_fw_topology.sh* and *setup_nat_topology.sh* that set up the topology for each workshop part. The script should also perform all necessary configuration of the middlebox, so that no further commands executions are needed before starting reachability tests.
- A demo of the working topology for both Firewall and NAT.
- A document containing the packet traces from tcpdump for each scenario explained in Section 3. The document should mention the various types of protocols seen in the packet trace and the role of each protocol. Special attention should be given to ARP packets as ARP will be used in the next workshops.

5 Useful References

5.1 Usage of *nc*

1. Start listening on the destination port (port 80 in this example)

   ```
   $ nc -l 80
   ```

2. Initiate connection to the destination port (destination host IP is 192.168.1.2)

   ```
   $ nc 192.168.1.2 80
   ```

3. Send data on the TCP connection by typing plaintext
4. Terminate the TCP connection by hitting Ctrl+C on either terminals

5.2 Other useful references

- Introduction to Firewall[3]
- How NAT works[4]

[3]https://www.geeksforgeeks.org/introduction-to-firewall/
[4]https://computer.howstuffworks.com/nat.htm