

Assignment 3

SDN Homework

Garuda Suma Pranavi

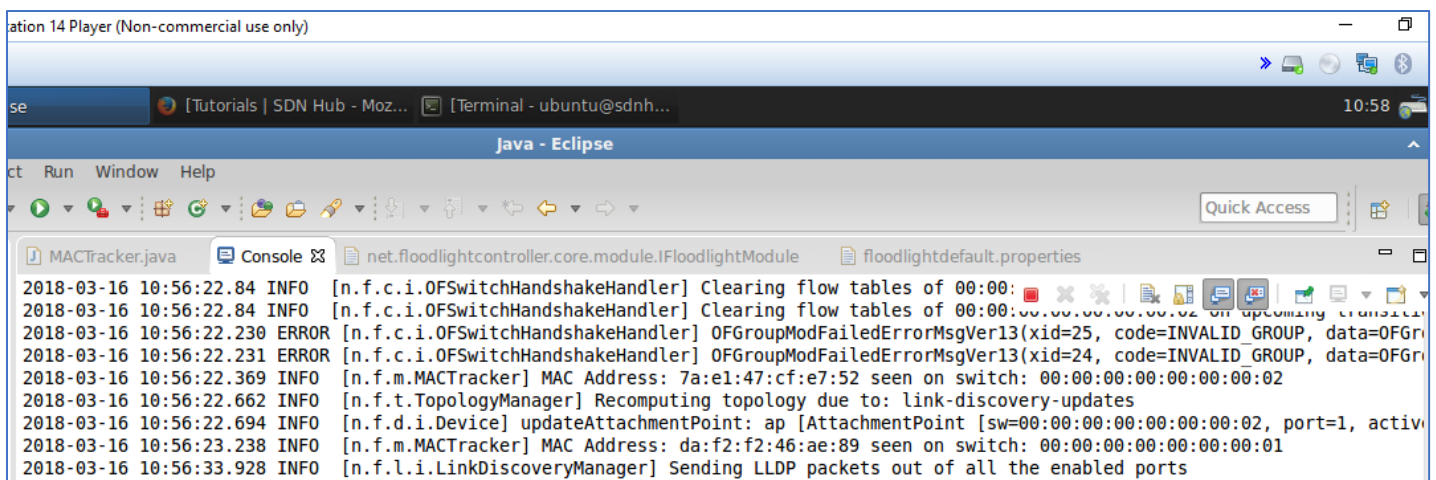
UIN 926009146

Task one : Your first App!

Running mactracker: This program relies on the fact that when the host first sends a ping, an ARP request is sent to the switch, where it checks if it has the MAC address specified in the packet, if not it sends a PACKET_IN message to the controller. The receive function then takes this packet and adds it as an entry on the switch. The logger is then used to printout the MAC addresses that we got on the packets and also the corresponding switch.

We simulated a linear topology on Mininet: `sudo mn -topo=linear,2 -controller=remote,ip=127.0.0.1,port=6653` (i.e. 2 hosts individually connected to switches and switches connected together).

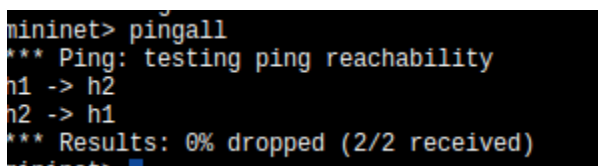
We made a ping request from h1 to h2 and observed the following results on the console:



```
2018-03-16 10:56:22.84 INFO [n.f.c.i.OFSwitchHandshakeHandler] Clearing flow tables of 00:00:00:00:00:00
2018-03-16 10:56:22.84 INFO [n.f.c.i.OFSwitchHandshakeHandler] Clearing flow tables of 00:00:00:00:00:00
2018-03-16 10:56:22.230 ERROR [n.f.c.i.OFSwitchHandshakeHandler] OFGroupModFailedErrorMsgVer13(xid=25, code=INVALID_GROUP, data=OFGr
2018-03-16 10:56:22.231 ERROR [n.f.c.i.OFSwitchHandshakeHandler] OFGroupModFailedErrorMsgVer13(xid=24, code=INVALID_GROUP, data=OFGr
2018-03-16 10:56:22.369 INFO [n.f.m.MACTracker] MAC Address: 7a:e1:47:cf:e7:52 seen on switch: 00:00:00:00:00:00:02
2018-03-16 10:56:22.662 INFO [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
2018-03-16 10:56:22.694 INFO [n.f.d.i.Device] updateAttachmentPoint: ap [AttachmentPoint [sw=00:00:00:00:00:00:02, port=1, activ
2018-03-16 10:56:23.238 INFO [n.f.m.MACTracker] MAC Address: da:f2:f2:46:ae:89 seen on switch: 00:00:00:00:00:00:01
2018-03-16 10:56:33.928 INFO [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
```

Task Two: Simple Switch

In this task we were asked to create a switch which is able to do forwarding even after we remove the forwarding class from floodlightdefault.properties. Most of the functionality is taken from the source code of the learning switch. We basically form a hash map where the MAC address is the key and the port number it's value. If we notice a new MAC and port number, it is added to the hash map and Flooding/unicasting is performed according to the availability of the key in the hashmap. For the first packet, flooding is performed and the destination MAC address and port number is found and added to the hash map. In this way we fill the table and we send a packet_out command to the switch.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

I wasn't sure what else could be added as indication that our program was running even after we remove forwarding. I am also attaching the code for the learning switch as a separate doc.

Task Three: Run FRESCO App

Without blacklist.fre:

After following the installation procedure mentioned in the SUCCESS website, I first ran the fresco application without any apps, with the following Mininet topology: `sudo mn -topo linear,3 -controller=remote,ip=127.0.0.1,port=6653`. (i.e. 3 hosts, each connected to their own individual switch).

```
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

We can see that we have connectivity between each of the hosts. Individual ping commands also attest to the same:

```
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=10.1 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.196/10.196/10.196/0.000 ms
mininet> h2 ping -c 1 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=7.14 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.142/7.142/7.142/0.000 ms
mininet> h3 ping -c 1 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=9.31 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 9.310/9.310/9.310/0.000 ms
mininet>
```

//from h1 to h2, h2 to h3 and h3 to h1 (we can transmit and receive the packet successfully)

With blacklist.fre:

After placing the blacklist.fre application in the fresco_apps/enable folder with the same Mininet configuration: `sudo mn -topo=linear,3 -controller=remote,ip=127.0.0.1,port=6653` (i.e. 3 hosts, each connected to their individual switches), we get the following result:

```
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X h3
h3 -> X h2
*** Results: 66% dropped (2/6 received)
```

We blacklist the IP of h1 i.e. 10.0.0.1, hence we do not get any response from the ping from h1 to h2 or h1 to h3. According to the results, ping from h2 to h1 or from h3 to h1 also does not work. However connectivity between h2 and h3 is not affected by the blacklist.

I understand the need to block any incoming or outgoing traffic from/to the attacker. However, according to the modules used in our blacklist application, we just match the flow-source IP and if there is a match, those flows are dropped. In this case, I did not understand why the flow from h2 to h1/from h3 to h1 is being dropped (since the source IP is not 10.0.0.1). I've also added a screenshot of the individual ping attempts:

```

mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

mininet> h1 ping -c 1 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

mininet> h2 ping -c 1 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=5.97 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5.978/5.978/5.978/0.000 ms
mininet> h3 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.90 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.904/2.904/2.904/0.000 ms
mininet> h3 ping -c 1 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 10.0.0.3 icmp_seq=1 Destination Host Unreachable

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

```

Task Four: Write FRESCO App

First we run our program without the port scan application enabled and we get the following output:

```

mininet> h1 nmap h2

Starting Nmap 6.40 ( http://nmap.org ) at 2018-03-18 14:56 PDT
Nmap scan report for 10.0.0.2
Host is up (0.0045s latency).
All 1000 scanned ports on 10.0.0.2 are closed
MAC Address: DA:84:B0:C0:99:C2 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 16.69 seconds

```

We can see that all the 1000 scanned ports on 10.0.0.2 are shown as closed and we do not get any information about applications listening on this port (because there aren't any).

However, on running the nmap 10.0.0.2 command we can see:

```

ubuntu@sdnhubvm:~$ nmap 10.0.0.2

Starting Nmap 6.40 ( http://nmap.org ) at 2018-03-18 15:03 PDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds

```

```

ubuntu@sdnhubvm:~$ nmap -PN 10.0.0.2

Starting Nmap 6.40 ( http://nmap.org ) at 2018-03-18 14:53 PDT
Nmap scan report for 10.0.0.2
Host is up (0.17s latency).
All 1000 scanned ports on 10.0.0.2 are filtered

```

After we put the port_scan.fre application in our enable folder we can see:

```
mininet> h1 nmap h2

Starting Nmap 6.40 ( http://nmap.org ) at 2018-03-18 15:31 PDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 0.62 seconds
mininet> h1 nmap -Pn h2

Starting Nmap 6.40 ( http://nmap.org ) at 2018-03-18 15:31 PDT
Nmap done: 1 IP address (0 hosts up) scanned in 0.59 seconds
mininet>
```

Host 2 is now blocking our ping probes and seems down to host 1, earlier we could see that the 1000 ports on the device h2 (MAC address also specified) were identified as closed. Now we do not get any information about the host, thus the port scanning attack has been successfully avoided.

One more thing we can check is that host 2 may seem closed because it's firewall settings may have such a provision, but we verify that using the -Pn command and still don't see any response. Thus all nmap commands have been blocked and we can see no scanning result from nmap.

Observation and Thinking: We must have a firewall at the end device to prevent port scanning attacks (so that nmap shows all ports as filtered and does not convey any information). If certain ports are open and the attacker is able to access the applications enabled on them and their versions, they also know the inherent vulnerabilities in the system and it becomes much easier for them to perform attacks.

Note: I wasn't able to work on developing any new application with the multitude of reports that we had to submit over the past week. (Every alternate day).