Net Security HW3 Report

Task1 MACTracker

```
2020-03-17 14:27:18.277 INFO  [n.f.m.MACTracker] MAC Address: 0e:41:e3:c7:6e:42 s
MAC Address:0e:41:e3:c7:6e:42
receive in myswitch
2020-03-17 14:27:18.321 INFO  [n.f.m.MACTracker] MAC Address: 8a:78:5b:61:0c:4a s
MAC Address:8a:78:5b:61:0c:4a
receive in myswitch
```

Fig1.1 Print MAC address when new MAC found

In MACTracker task, we built a simple network topo on mininet by the command “sudo mn --topo linear,2 --controller=remote,ip=127.0.0.1,port=6653”.
The controller is implemented by floodlight whose function is to check the MAC addr pair, that if it’s new. If so, then add it to the map and print the new found MAC. The print result in Eclipse console is shown above in Fig 1.1.

Task2 SimpleSwitch

In this task, I built a network whose topo is single which means that the hosts in it cannot ping each other successfully at first(as shown in Fig 2.1).

```
mn: error: no such option: --protocols
ubuntu@sdnhubvm:~[14:18]$ sudo mn --topo single,8 --controller
.1,port=6653 --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X X X
h2 -> X X X X X X X
h3 -> X X X X X X X
h4 -> X X X X X X X
h5 -> X X X X X X X
h6 -> X X X X X X X
h7 -> X X X X X X X
h8 -> X X X X X X X
*** Results: 100% dropped (0/56 received)
mininet>
```

Fig 2.1 ping failed

Then I build a hash map whose key is the MAC address, and the value is the port in switch. At first the switch receives a packet and doesn't know where to send it (no idea about which port), the controller requires the switch to send a flood and waits for the ARP replies. When ARP replies arrive at the switch, match the port number and the source MAC address and add them into the Hash map.

After that, when a new packet arrives at the switch, the controller will know where to send it and tell the switch to re-send it even when there is no link between them. Here is the result after implementation of learning switch in Fig 2.2

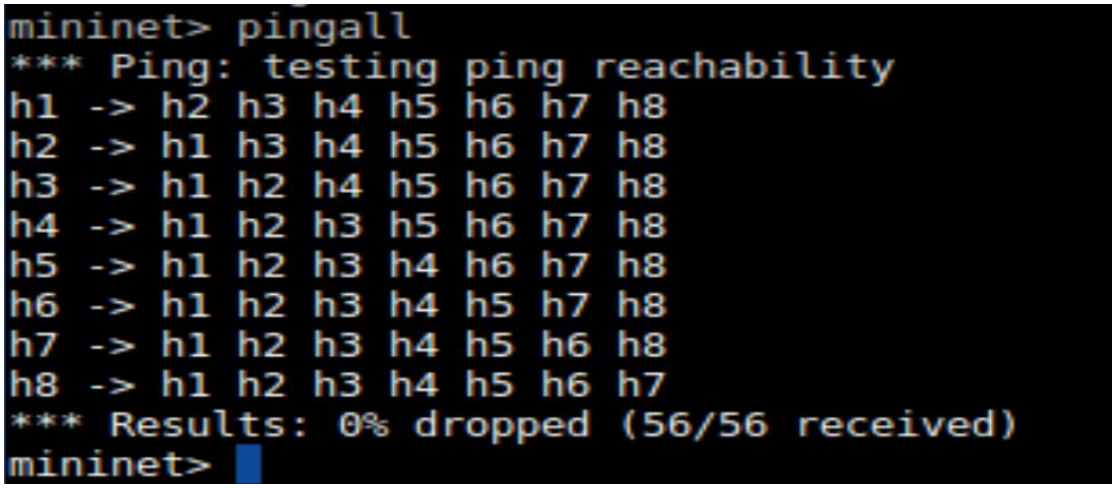

Fig 2.2 ping successfully

Task3 FRESCO

In this task, I built a network of linear topo, where three hosts under one switch can ping each other successfully at first, as shown in Fig 3.1.

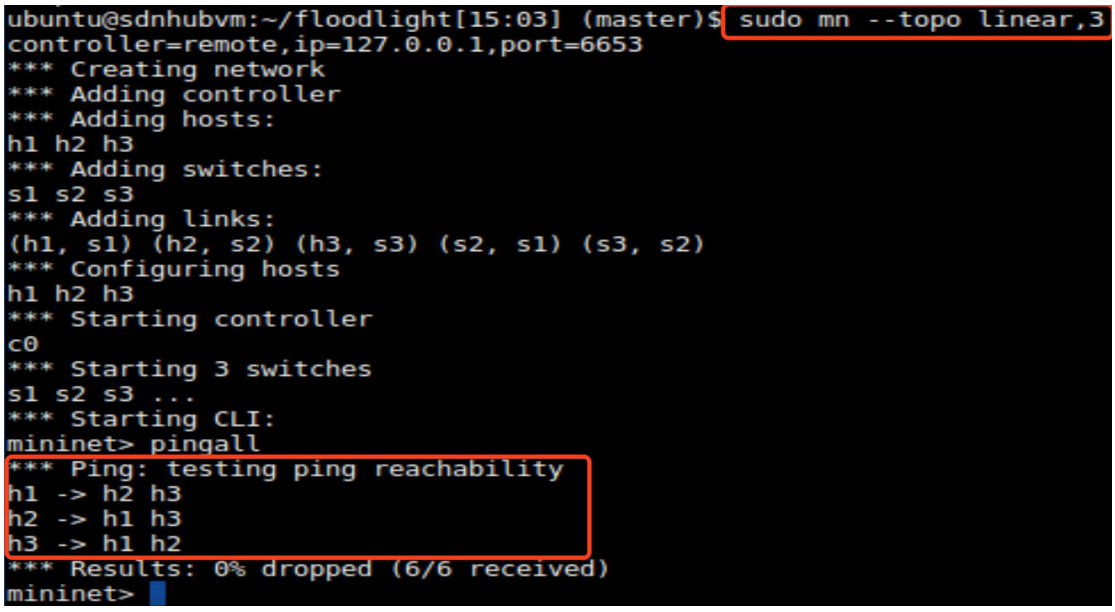

Fig 3.1 ping successfully

Then we modify the blacklist.fre to make it block any packet concerning with ‘h2’ whose IP is “10.0.0.2”.

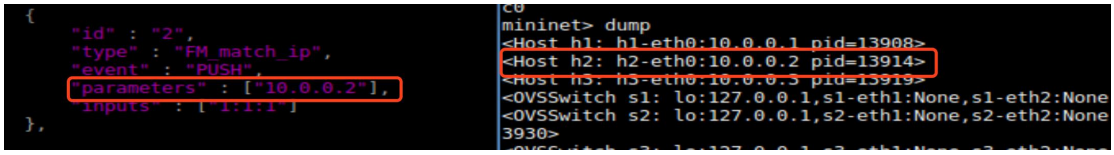

Fig 3.2 block ‘h2’ packets

Then we re-run mininet and floodlight with FRECO again. The ‘ping’ result shows that the packets from or to ‘h2’ are blocked in Fig 3.3

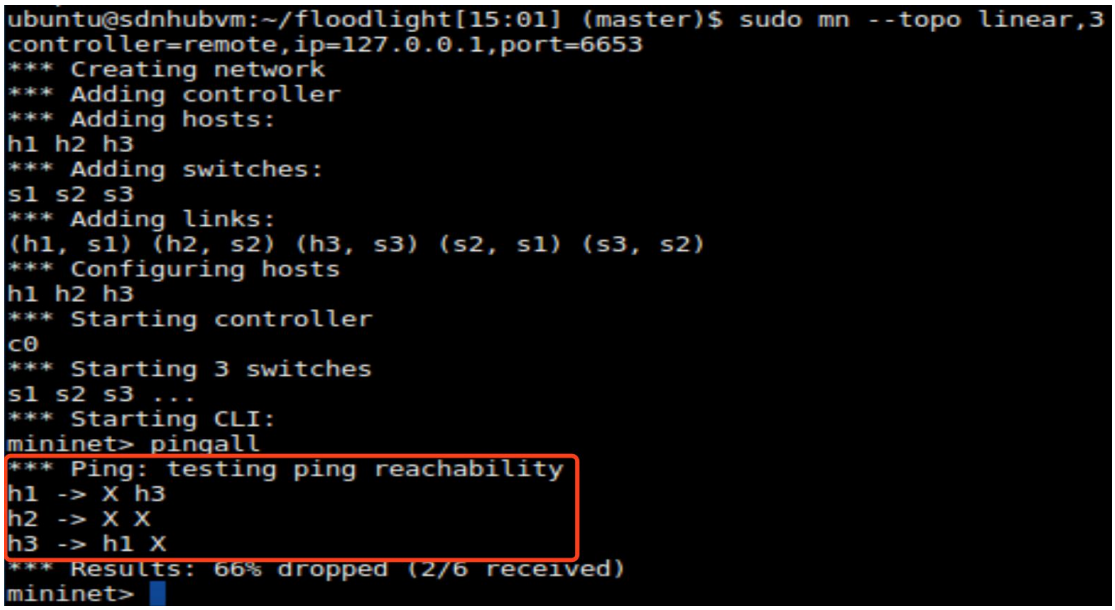

Fig 3.3 ping failed

Task4 Firewall prevents port scan

In mininet network, we can assume any host as a running VM who can run most Linux commands on it. In this task, I built a network like Task3, 3 hosts, 1 switch. Let h1 be a server(also the victim), h2 be an port scan attacker towards h1. The command is:

“h1 python -m SimpleHTTPServer 80”

“h2 nmap -PU h1”

Without the firewall, h2 will get the port 80 is open on h1, as shown in Fig 4.1.

```
ubuntu@sdnhubvm:~/floodlight[23:13] (master)$ sudo mn --topo linear,3 -
controller=remote,ip=127.0.0.1,port=6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> h1 python -m SimpleHTTPServer 80 1>/dev/null 2>/dev/null &
mininet> h2 nmap -PU h1

Starting Nmap 6.40 ( http://nmap.org ) at 2020-03-17 23:14 PDT
Nmap scan report for 10.0.0.1
Host is up (0.0030s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE
80/tcp open  http
MAC Address: F2:94:51:CE:10:FE (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 28.55 seconds
mininet> exit
```

Fig 4.1 result before firewall running

Then I write a new FRESCO app, “findscan.fre”.

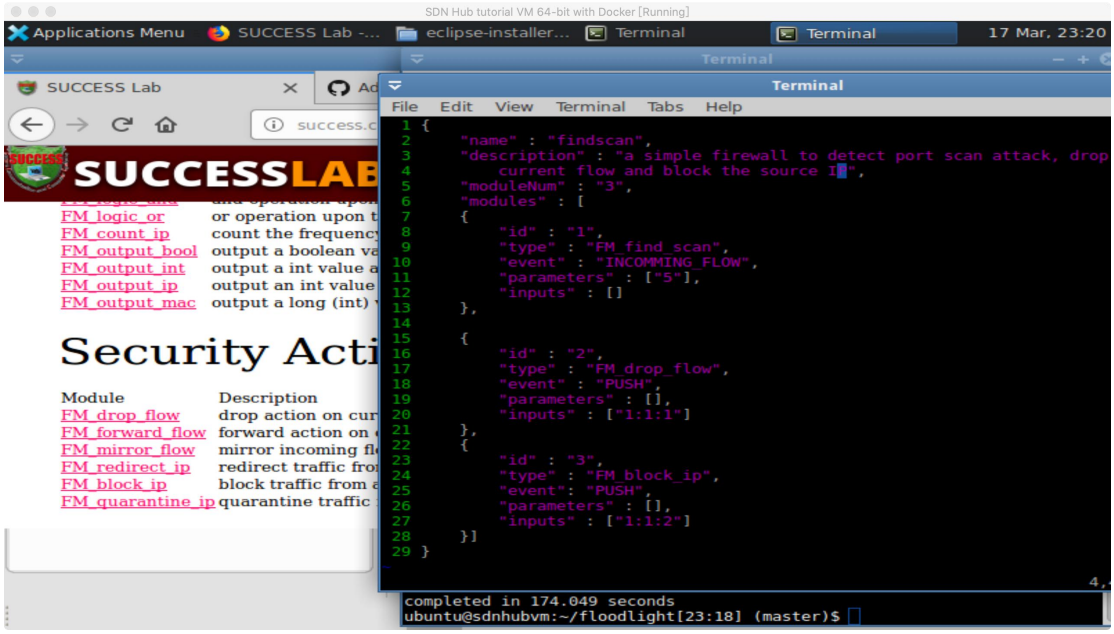

Fig 4.2 “findscan.fre”

3 modules implemented this function:

1st module detects any port scan flow from the switch, and pass it to module 2 and 3

2nd module does the action that drops current flow.

3rd module does the action that blocks all packets from the source IP after detection.

We can see that after running this firewall, the scan port action will fail, as shown in Fig 4.2.

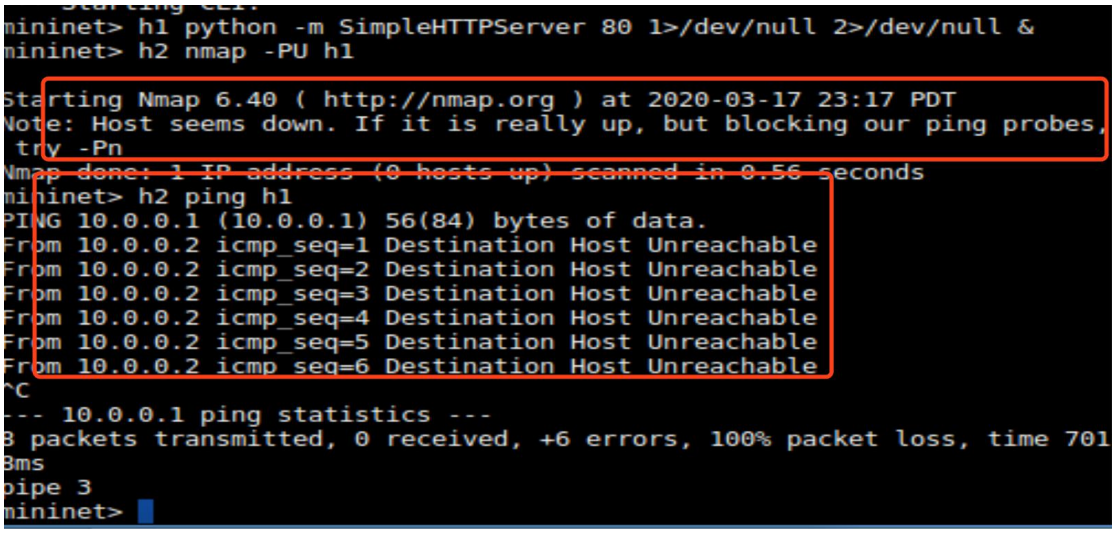

Fig 4.2 result after firewall running

Plus due to module3, even h2 is not trying to do port scan any more. The normal ping packets will fail as well. As in normal case, the network is linear, h1 and h2 can ping each other.

Task5 New FRESCO APP

In this task, I wrote a FRESCO app to filter flow transferred from fixed port(like port 23 to forbid Telnet request). The code is shown below:

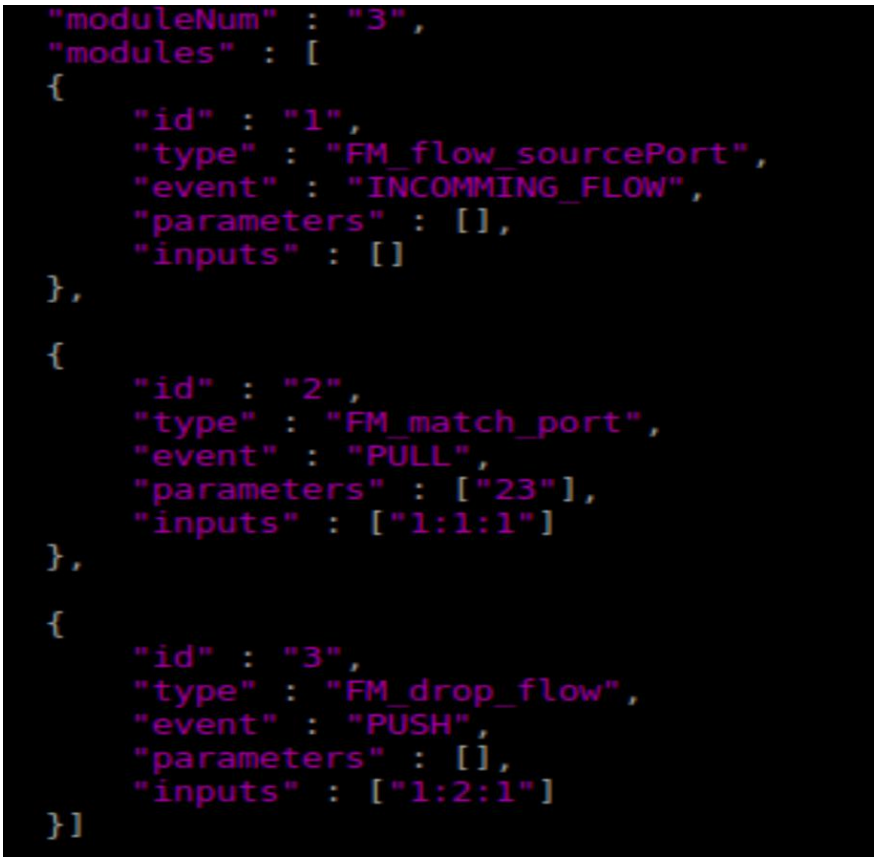

Fig 5.1 Portfilter.fre

Environment: we still use linear net topo for this task. “sudo mn --topo=linear,3”
Before running the port filter app, due to the port is forbidden by mininet at first. The refused response will be transferred at once.

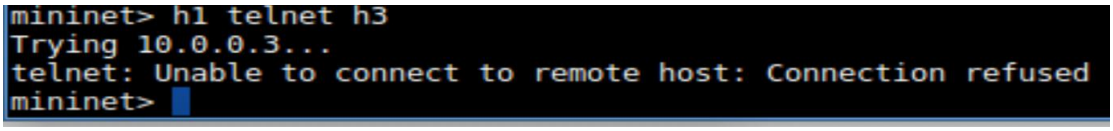

Fig 5.2 refused request

While running port filter app, the telnet request using port 23 will be dropped once received, at transferring end, there will be no response.

Fig 5.3 no response

This port filter app can be used when protecting some internal port from attackers.