SDN-Based Stateless Firewall

Student Name: Hrishikesh Dahale

Email: hdahale@asu.edu

Submission Date: 03/06/2022

Class Name and Term: CSE548 Spring 2022

I. Project Overview

This lab explores setting up a software-defined environment based on mininet and containernet. I also got to practice setting up an OpenFlow-based flow-level firewall on SDN. Finally, I need to set up and practice flow-based firewall filtering policies such as enabling the ability to accept, drop, or reject the incoming flows, thus ensuring the system's safety from malicious attacking network traffic.

All the configurations used for this lab have been uploaded on GitHub; references are provided throughout the text and in Appendix A at the bottom of the File.

II. Software

For this lab, the following software has been used:

- Various network tools (specifically, tcpdump, ping, traceroute, hping3, and nc – netcat)
- POX (GitHub link: https://noxrepo.github.io/pox-doc/html/
- Open vSwitch: http://www.openvswitch.org/
- Open vSwitch Cheat Sheet: https://therandomsecurityguy.com/openvswitch-cheatsheet/
- Containernet: https://containernet.github.io/
- Containernet tutorial: https://github.com/containernet/containernet/wiki/Tutorial:-Getting-Started

III. Project Description

A. *Lab "CS-CNS-00101" – OpenFlow Based Stateless firewall*

In this lab, the students must verify the working of a stateless firewall and try adding different rules using config files.

1) Create a mininet based topology with 4 container hosts and one controller switches and run it.

- *Add link from controller1 to switch 1.*
- *Add link from controller2 to switch 1.*
- *Add link from switch 1 to container 1.*
- *Add link from switch 1 to container 2.*
- *Add link from switch 1 to container 3.*
- *Add link from switch 1 to container 4.*

```
root@ubuntu:~# mn --topo=single,4 --controller=remote,port=6633 --controller=remote,port=6655 --switch=ovsk --mac
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6655
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0 c1
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet>
```

Figure 1 Create 4 container host and 1 controller switch

2) *Make the interfaces up and assign IP addresses to interfaces of container hosts.*
   - *Assign IP address 192.168.2.10 to container host #1.*
   - *Assign IP address 192.168.2.20 to container host #2.*
   - *Assign IP address 192.168.2.30 to container host #3.*
   - *Assign IP address 192.168.2.40 to container host #4.*

The following commands are running from the mininet CLI to assign them the addresses requested by the lab:

```
*** Starting CLI:
containernet> h1 ifconfig h1-eth0 192.168.2.10
containernet> h2 ifconfig h2-eth0 192.168.2.20
containernet> h3 ifconfig h3-eth0 192.168.2.30
containernet> h4 ifconfig h4-eth0 192.168.2.40
containernet>
```

Figure 2 Change IP Addresses

The following screenshot documents that the containers have all assumed the correct IP addresses.

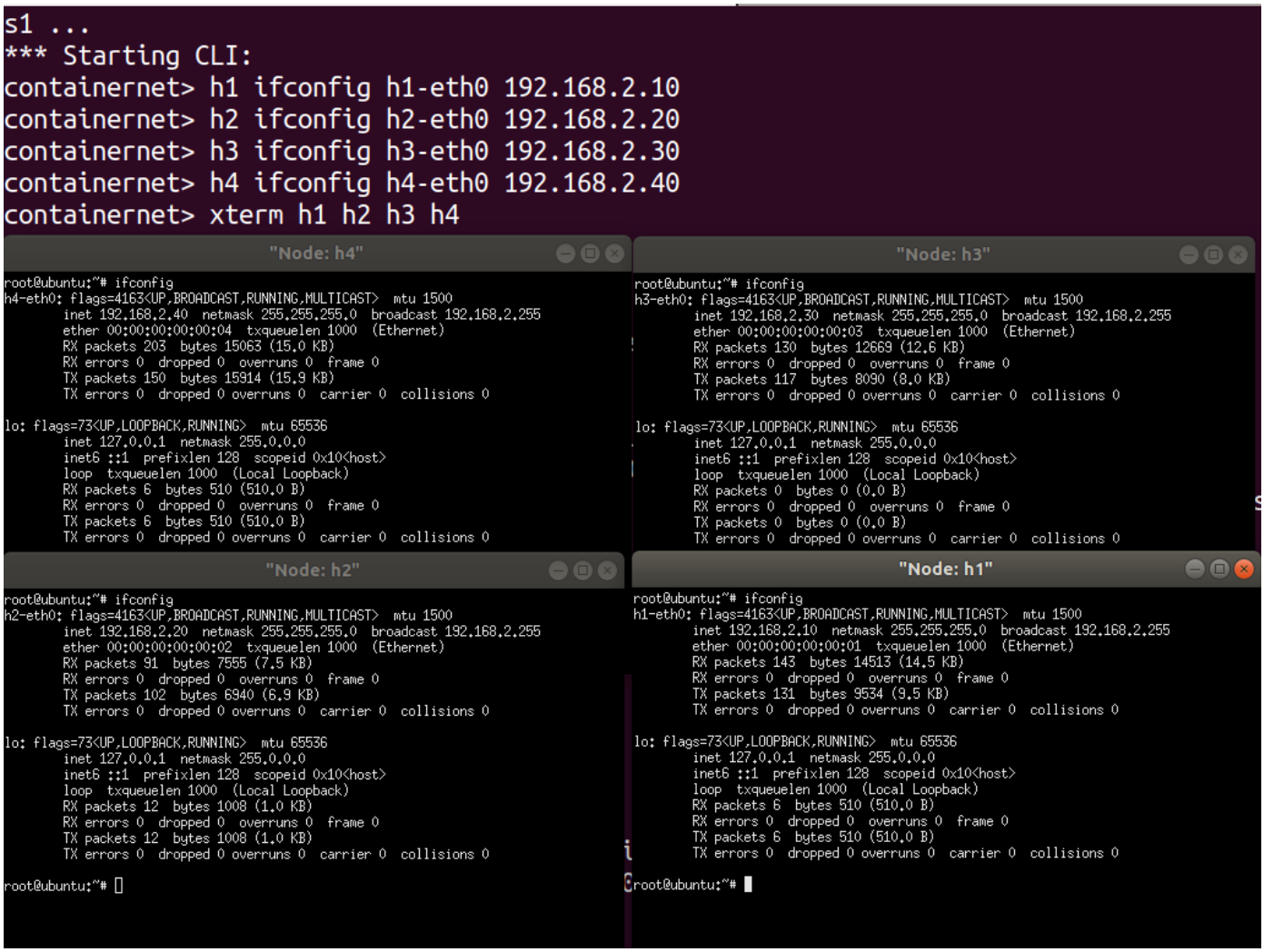

Figure 3 CS-CNS-00101 – Lab 2

The following screenshot documents the l3firewall.config:

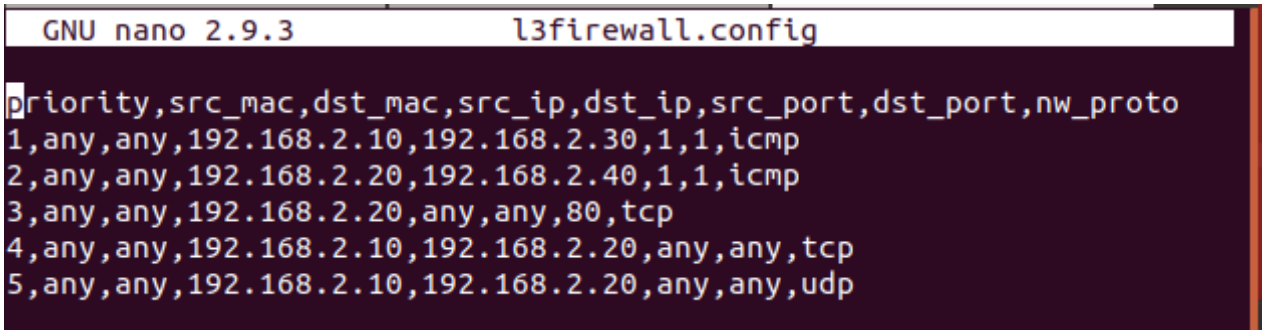

Figure 4 l3firewall.config

The following screenshot documents the l2firewall.config:

```
 GNU nano 2.9.3          l2firewall.config

id,mac_0,mac_1
1,00:00:00:00:00:02,00:00:00:00:00:04
```

Figure 5 l2firewall.config

At this point, I restart both pox and mininet.

```
root@ubuntu:/home/ubuntu/pox# ./pox.py openflow.of_01 --port=6633 po
x.forwarding.l2_learning pox.forwarding.L3Firewall --l2config="l2fir
ewall.config" --l3config="l3firewall.config"
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
src_ip, dst_ip, src_port, dst_port 192.168.2.10 192.168.2.30 1 1
src_ip, dst_ip, src_port, dst_port 192.168.2.20 192.168.2.40 1 1
src_ip, dst_ip, src_port, dst_port 192.168.2.20 any any 80
src_ip, dst_ip, src_port, dst_port 192.168.2.10 192.168.2.20 any any
src_ip, dst_ip, src_port, dst_port 192.168.2.10 192.168.2.20 any any
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
00:00:00:00:00:02 00:00:00:00:00:04
```

Figure 6 checking pox output

3) Add new rule to l3config file for blocking ICMP traffic from source IP 192.168.2.10 and destination IP 192.168.2.30.

Let us now test out our rules to see if they work. Let us start by pinging h3 from h1. The ping should fail.

```
containernet> h1 ping h3
PING 192.168.2.30 (192.168.2.30) 56(84) bytes of data.
^C
--- 192.168.2.30 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2056ms

containernet>
```

Figure 7 ping from h1 to h3 are blocked

4) Add new rule to l3config file for blocking ICMP traffic from source IP 192.168.2.20 and destination IP 192.168.2.40.

Pinging h4 from h2 also fails

```
containernet> h2 ping h4
PING 192.168.2.40 (192.168.2.40) 56(84) bytes of data.
^C
--- 192.168.2.40 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1019ms

containernet>
```

Figure 8 ping from h2 to h4 are blocked

5) Add new rule to l3config file for blocking HTTP traffic from source IP 192.168.2.20.

```
containernet> h1 nmap -p 80 h2

Starting Nmap 7.60 ( https://nmap.org ) at 2022-03-05 14:44 MST
Nmap scan report for 192.168.2.20
Host is up (0.00020s latency).

PORT   STATE    SERVICE
80/tcp filtered http
MAC Address: 00:00:00:00:00:02 (Xerox)

Nmap done: 1 IP address (1 host up) scanned in 13.61 seconds
containernet>
```

Figure 9 port 80 is filtered

From the screenshot, it is evidence that port 80 is filtered indicating that there is a firewall installed

6) Add new rule to l2config file for blocking traffic from MAC address 00:00:00:00:00:02 to destination MAC address 00:00:00:00:00:04.

I start the Python SimpleHTTPServer on port 443 of node h4 and test fetching a page from the node h1, which works (as expected):

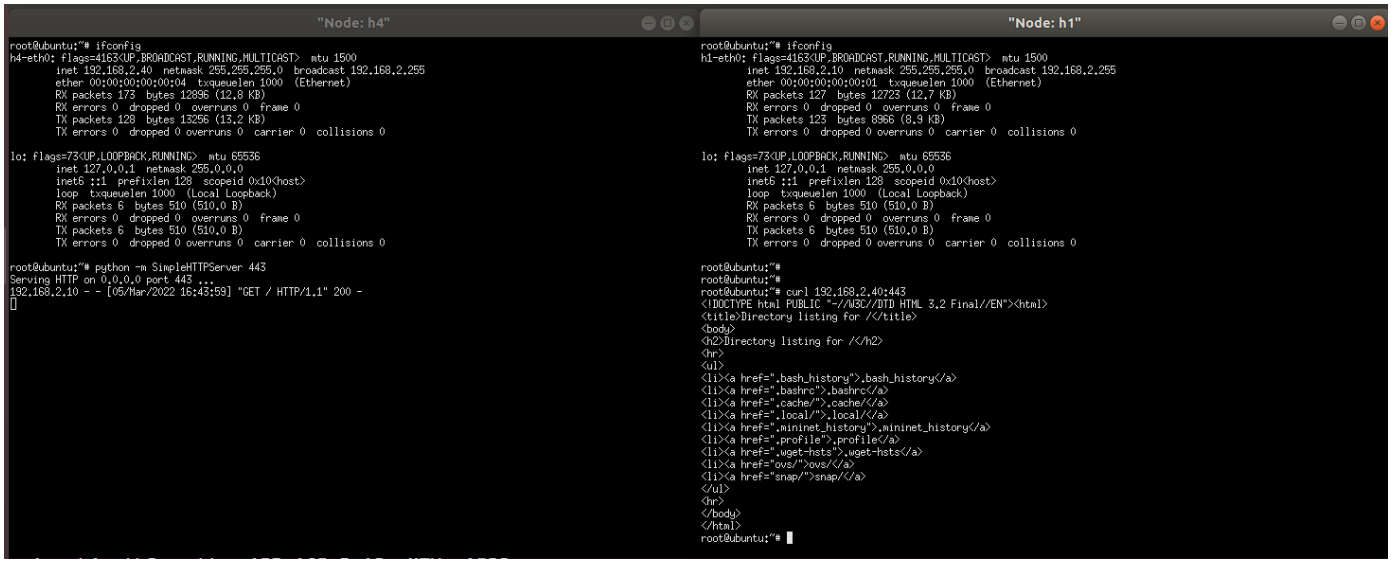

Figure 10 h1 can reach web server on h4

However, trying to “browse” the web page at port 443 from node h2 fails as the connection is dropped and, as we can see, it never reaches the node h4 – confirming the Layer2 rule works as intended:

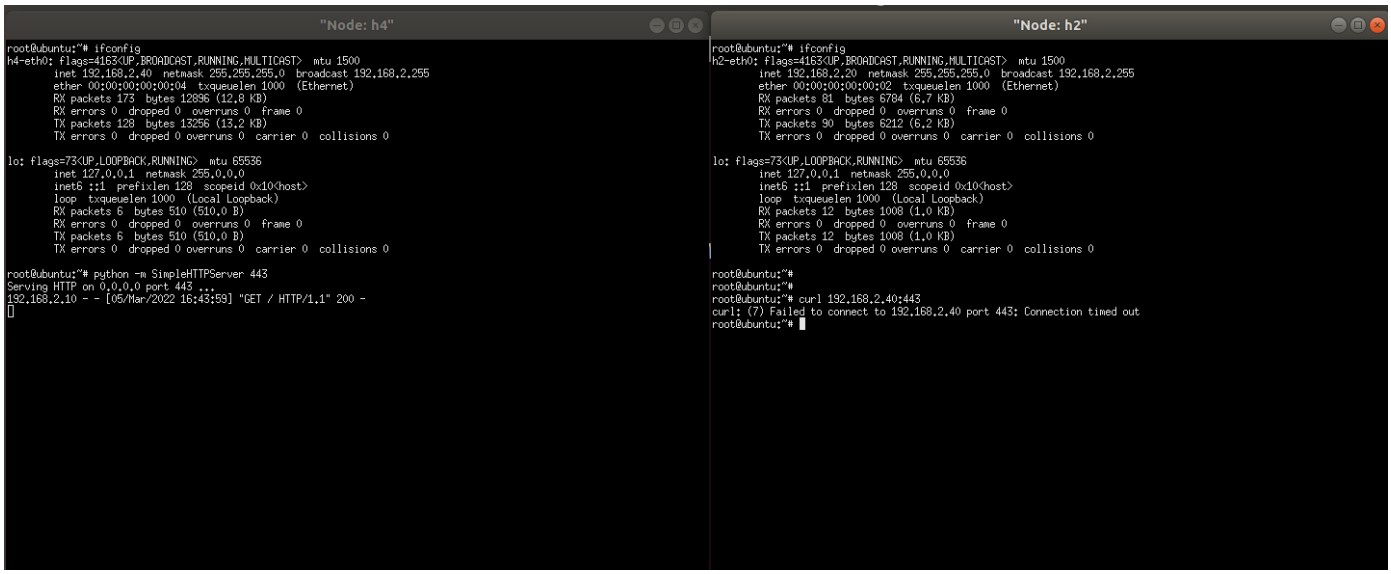

Figure 11 h2 cannot reach web server on h4

Another test against the webserver on h4, this time from h3, shows that effectively only h2 is blocked from reaching web pages.

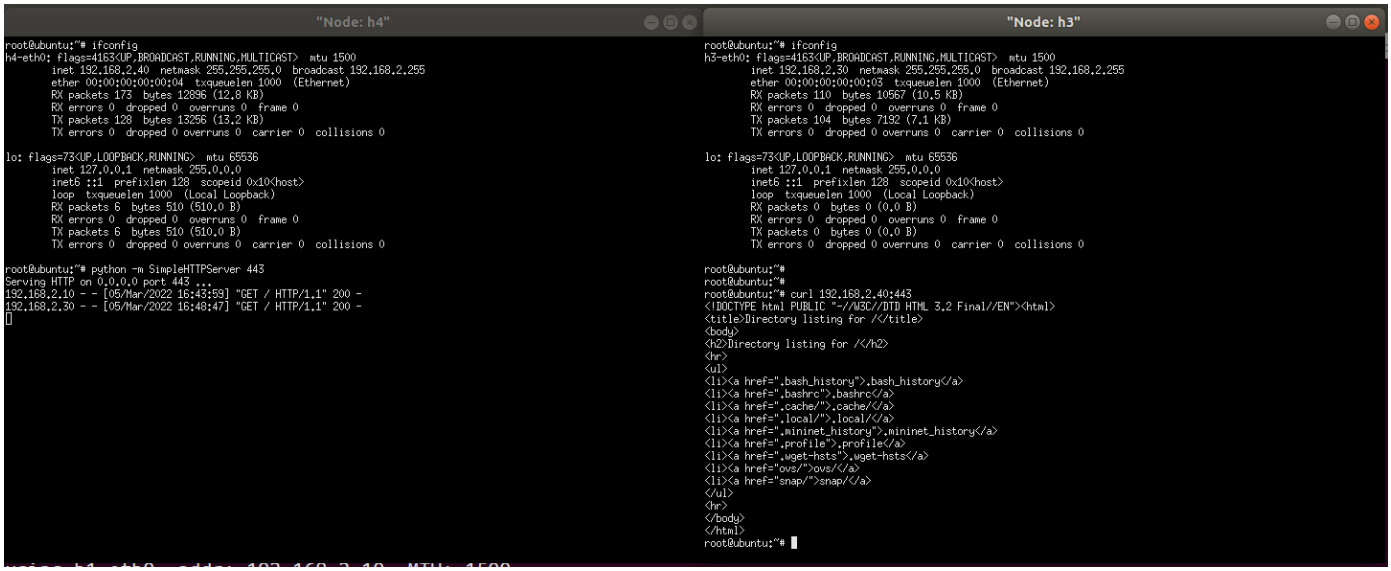

Figure 12 h3 can reach web server on h4

7) Add new rule to l3config file for blocking tcp traffic from 192.168.2.10 to 192.168.2.20.

```
containernet> h1 hping3 -c 5 h2 -V --tcp-timestamp
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): NO FLAGS are set, 40 headers + 0 data bytes

--- 192.168.2.20 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
containernet>
```

Figure 13 TCP blocked

8) Add new rule to l3config file for blocking udp traffic from 192.168.2.10 to 192.168.2.20.

```
containernet> h1 hping3 -c 5 h2 -V --udp
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): udp mode set, 28 headers + 0 data bytes

--- 192.168.2.20 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
containernet>
```

Figure 14 UDP blocked

IV. Conclusion

By completing this assignment, I have learned how to set up a firewall on SDN, flow-based firewall filtering policies such as enabling the ability to accept, drop, or reject the incoming flows, thus ensuring the system's safety from malicious attacking network traffic.

V. Appendix B: Attached files

Please find the list of files created for this lab and mentioned throughout this document, plus their GitHub link for download.

The overall GitHub directory for the project is:

Project_2_SDN-Based_Stateless Firewall.pdf	https://github.com/HD-Codes/CSE-548/tree/main/Project_2_SDN-Based_Stateless_Firewall.pdf
l2firewall.config	https://github.com/HD-Codes/CSE-548/blob/main/Project_2_SDN-Based_Stateless_Firewall/l2firewall.config
l3firewall.config	https://github.com/HD-Codes/CSE-548/blob/main/Project_2_SDN-Based_Stateless_Firewall/l3firewall.config
Video	https://youtu.be/9ayAjw04QGA

VI. References

[1] Linux NAT Tutorial: https://www.karlrupp.net/en/computer/nat_tutorial

[2] Ubuntu “Basic Iptables HOWTO”: https://help.ubuntu.com/community/IptablesHowTo

[3] “Iptables Tutorial: Ultimate Guide to Linux Firewall”: https://phoenixnap.com/kb/iptables-tutorial-linux-firewall