Student Name: Honglin Ma  
Email: mhl970128@gmail.com  
Submission Date: 06/03/2022  
Class Name and Term: CSE548 spring 2022

Project 2: SDN-Based Stateless Firewall

# Project Overview

In this project, the main goal was to understand learn and build a software-defined environment based on mininet as well as containerernet (a kind of dockers) in a Linux system environment. In this environment, after creating virtual hosts through mininet, I practiced how to build an OpenFlow-based stateless flow firewall on SDN. By setting up the required firewall filtering rules in this environment, the function of blocking malicious attacking network traffic is achieved.

# Network Setup

* In this experiment, a total of one virtual machine was configured on Oracle VM VirtualBox 6.0.24,it will be referred to as project23 in the following. The network was configured for this machine with the help of a virtual NAT device and a virtual DHCP service (Client-side Net 1: 10.0.2.0/24). The network was selected as shown in the figure below.initial reachability among network nodes



II-1 VM project23 net set

* In the Virtual machine project23, we need to create four containerernet hosts h1-h4 via mininet. These four hosts

will be linked to a switch, and the switch will be linked to two controllers, as will be shown in the following implementation.

# Software

For this project, the following software has been used:

* Various network tools ( tcpdump, ping, traceroute, hping3, curl etc.)
* Oracle VM VirtualBox 6.0.24
* Pox Controller
* OpenVirtual Switch
* Mininet
* Containernet

# Project Description

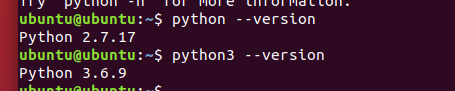
## Lab CS-CNS-00101 – OpenFlow Based Stateless firewall

The Youtube video demonstration address is: https://www.youtube.com/watch?v=I-PvRwsgm-U

Next I will show step by step the implementation process for the Lab CS-CNS-00101 (SDN Firewall):

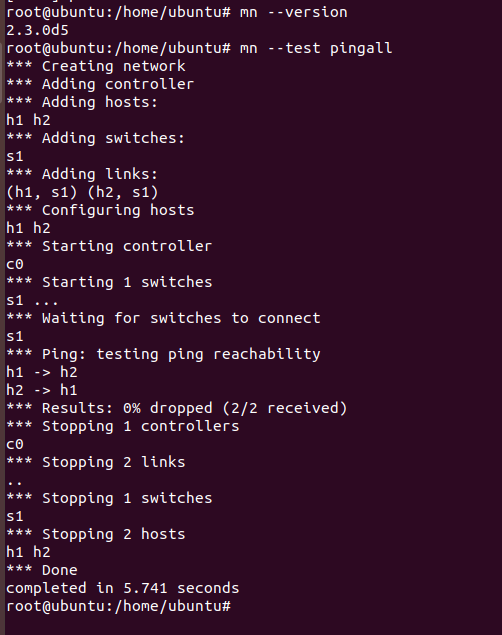
First we need to check the installation of all required services and software in the host.

1. Check Python



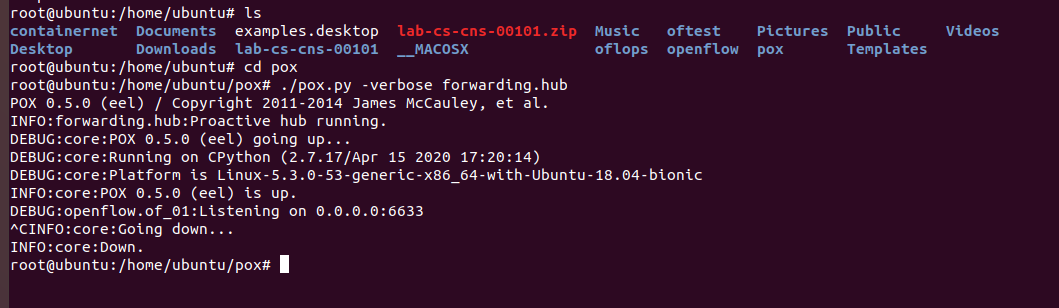
IV-1 Python check

1. Check Mininet



IV-2 Install Mininet

1. Check Pox controller



IV-3 Install Pox

1. Check OVS(OpenVirtual Switch)



IV-4 Install OVS

Lab Assessment 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.*

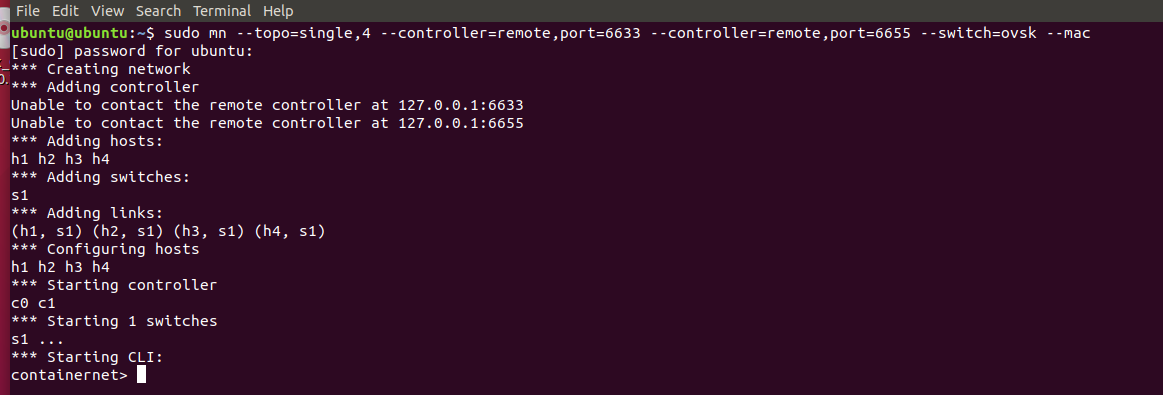
*• Add link from switch 1 to container 2.*

*• Add link from switch 1 to container 3.*

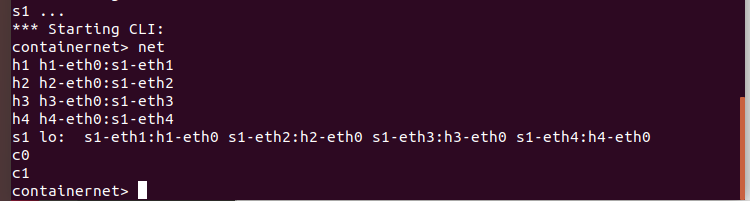
*• Add link from switch 1 to container 4.*

This problem requires us to create 4 container hosts, 1 switch, 2 controllers and create links among them.

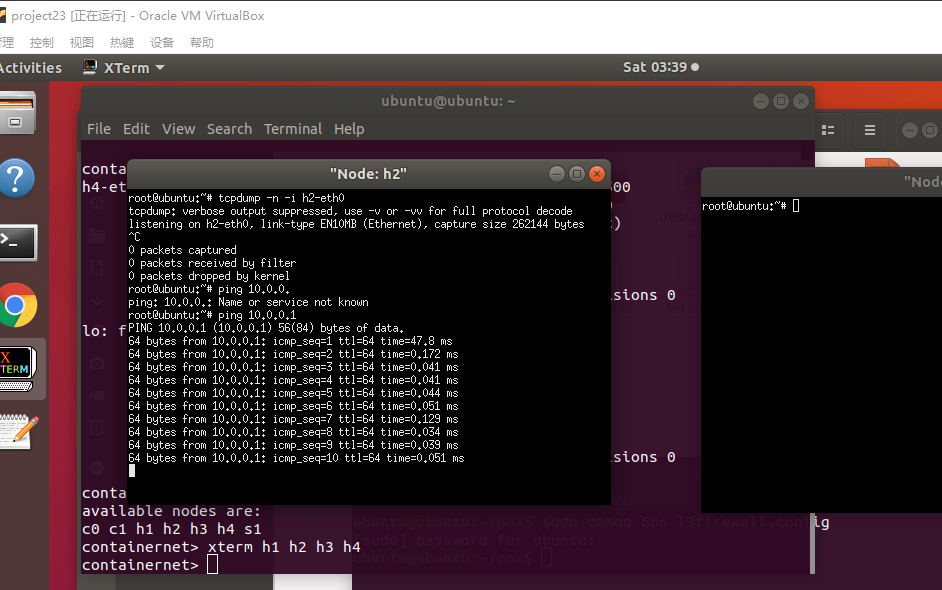
Here we use the mininet command: *sudo mn --topo=single,4 --controller=remote,port=6633 --controller=remote,port=6655 --switch=ovsk --mac*



IV-5 Lab CS-CNS-00101 Assessment 1



IV-6 Info of links



IV-7 Test 10.0.0.2 ping 10.0.0.1

In the figure IV-8 we can see that the required hosts, switches and controllers have been created and the links have been added among them. And a simple test of the data flow between hosts shows that it is possible to ping fron h2 to h1.

Lab Assessment 2):

*Make the interfaces up and assign IP addresses to interfaces of container hosts.*

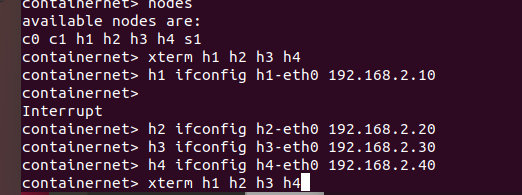
*Assign IP address 192.168.2.10 to container host #1.*

*Assign IP address 192.168.2.20 to container host #2.*

*Assign IP address 192.168.2.30 to container host #3.*

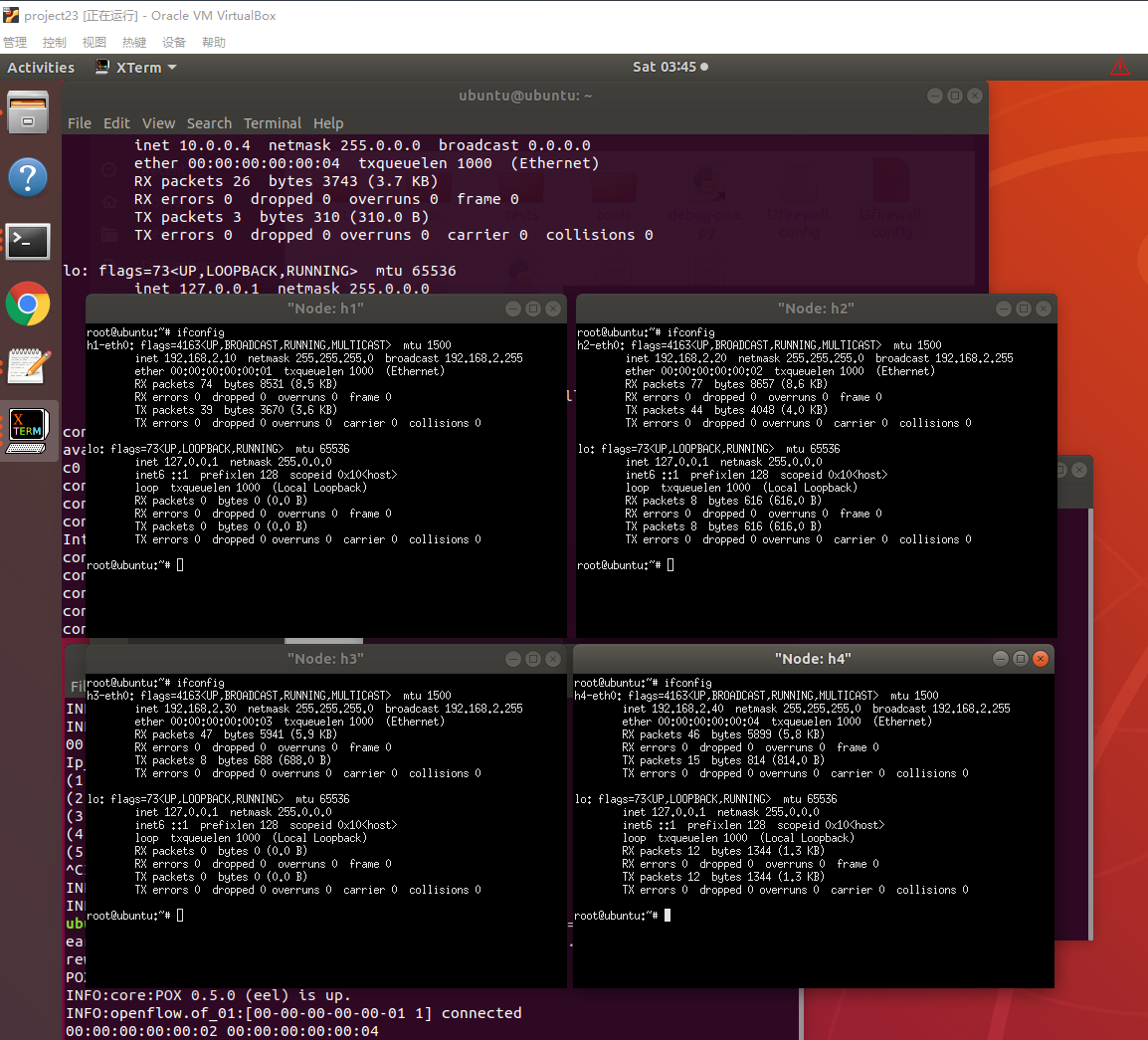
*Assign IP address 192.168.2.40 to container host #4.*

Here we need to assign new IP addresses to the four container network hosts created (since the initial IPs created by system default are 10.0.0.1-10.0.0.4) with the command(in mininet terminal): *h1 ifconfig h1-eth0 192.168.2.10 etc.*



IV-9 Assign IP addresses to container hosts

Then we can turn on all of the hosts and check the ip of each of them.

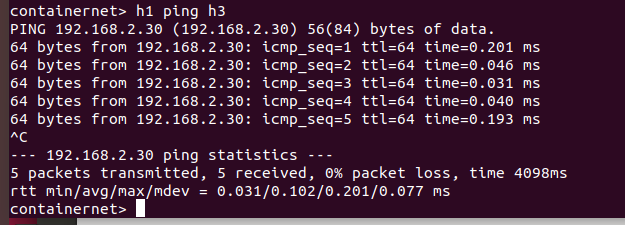


IV-10 Net information of each host

Lab Assessment 3):

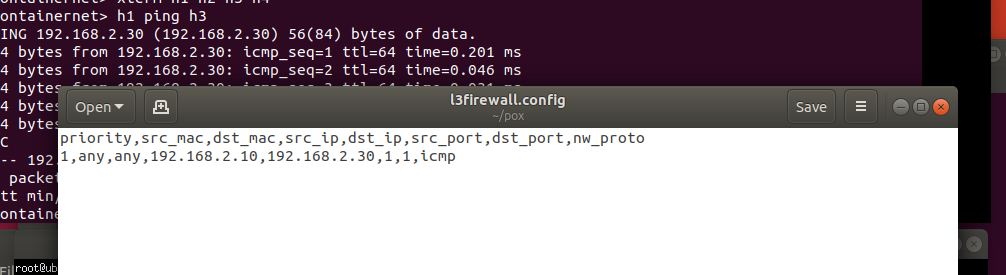
*Add new rule to l3config file for blocking ICMP traffic from source IP 192.168.2.10 and destination IP 192.168.2.30.*

First, i tested the ICMP traffic from source IP 192.168.2.10 and destination IP 192.168.2.30,and here we see the trafiic is not blocked.



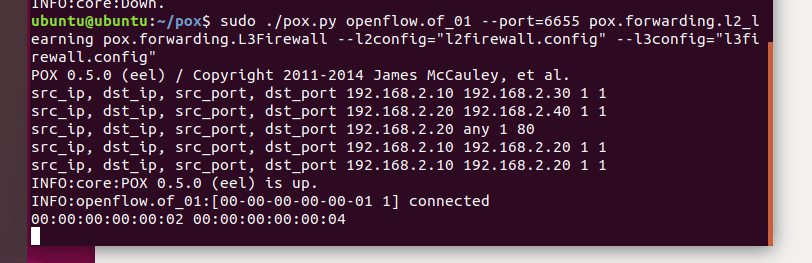
IV-11 Traffic status before rule added

Then we add the rule: *1,any,any,192.168.2.10,192.168.2.30,1,1,icmp* in l3firewall.config,which means the layer 3 level added.

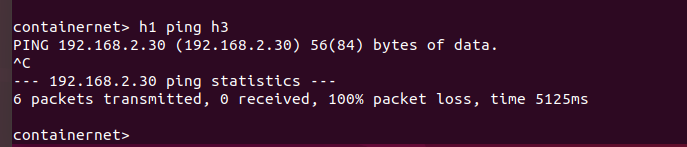


IV-12 layer 3 rule added for assessment 3

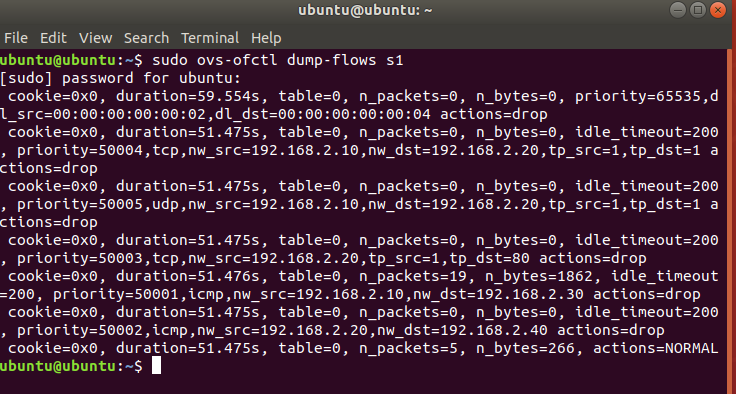
Then i invoked pox and start these rules files and test the icmp traffic between h1 and h3.



IV-13 Invoke the rules



IV-14 ICMP traffic between h1 and h3



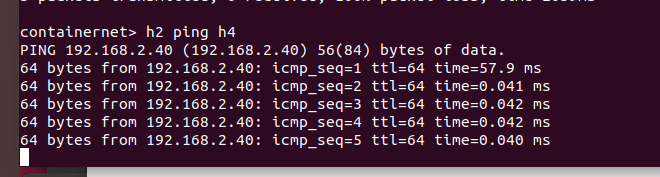
IV-15 dumpflows detail

Here the ICMP traffic from h1 to h3 has been blocked.

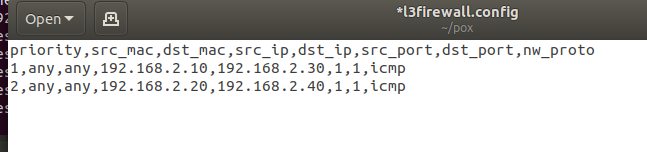
Lab Assessment 4):

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

The operation of this question is almost the same as the previous one. First test the ICMP traffic before the rule is added. Then we add the rule 2 in layer 3 level.

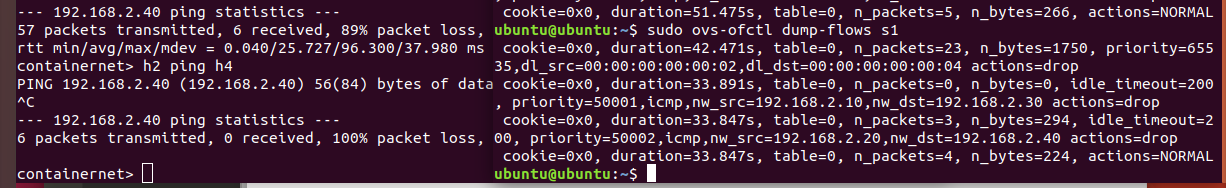


IV-16 Traffic status before rule 2 added



IV-17 Rule 2 be added

Retest h2 ping h4 and its traffic after activating the rule and find that the rule is enabled successfully.



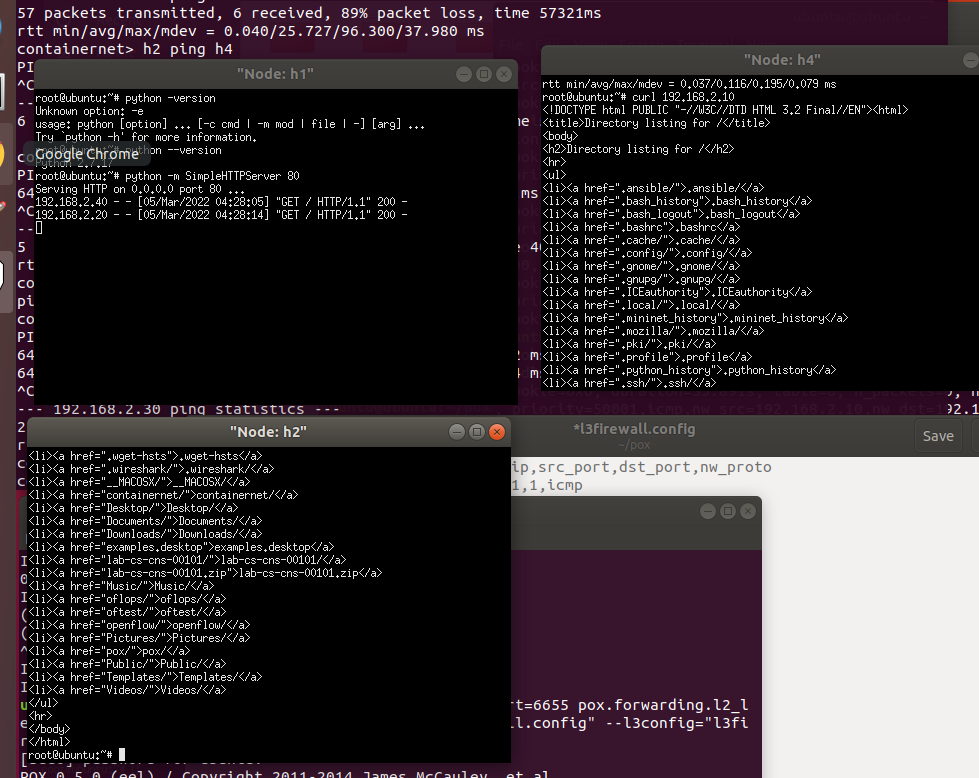
IV-18 ICMP traffic and flow of rule 2

Lab Assessment 5):

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

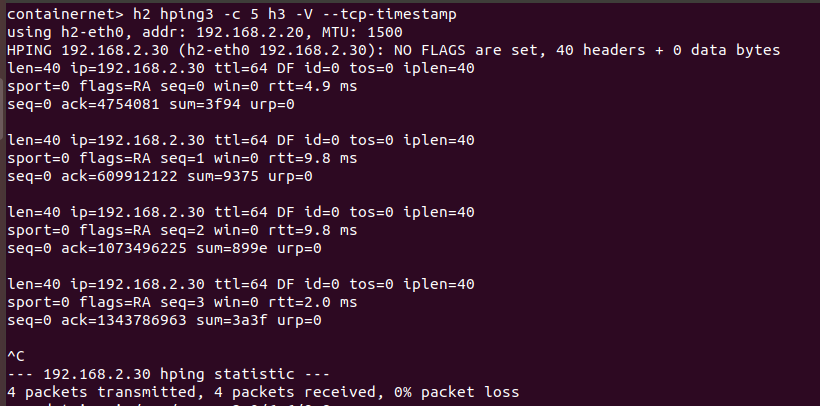
This question requires us to block all http traffic from the source address 192.168.2.20. This means we need to block the host with source IP 192.168.2.20 from accessing all other hosts on port 80 of the tcp protocol (HTTP). So let's test h2 and other machines for http traffic at the beginning as usual.

Here we have created a python-based http service in h1 and opened its port 80. Another way to test this is to use the hping3 command in a mininet window. The result is the same. Here you can see that h2 can access the http service of h1.

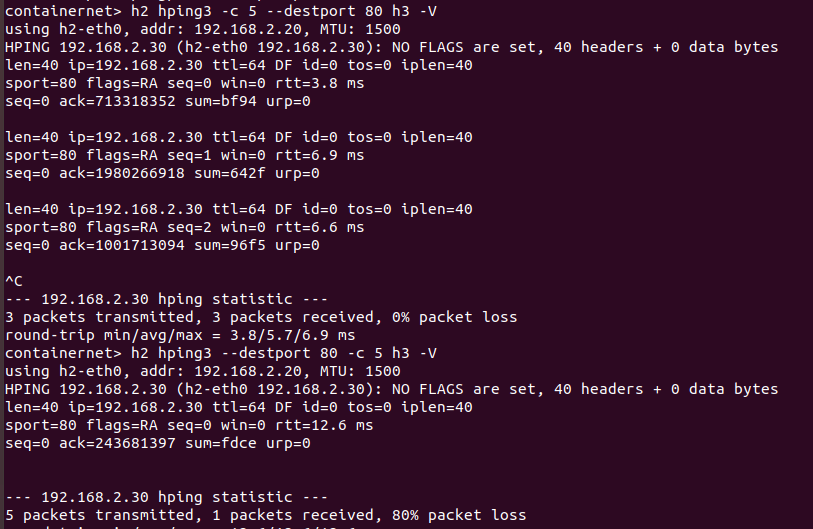


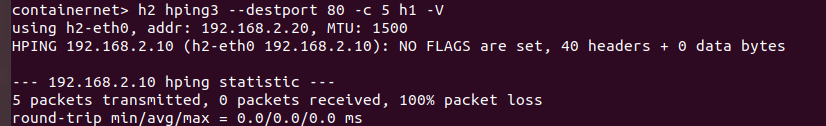
IV-19 HTTP traffic test before rule added

Then we start a new rule: *3,any,any,192.168.2.20,any,any,80,tcp* , after the new rule is started we first test the default tcp traffic for h2 and h3 and find that it is successful, then we test the traffic for h2 and h3 based on port 80 (http) and find that it fails. This means that our new rule was successfully used to block HTTP traffic from h2.



IV-20 Other port traffic was not blocked



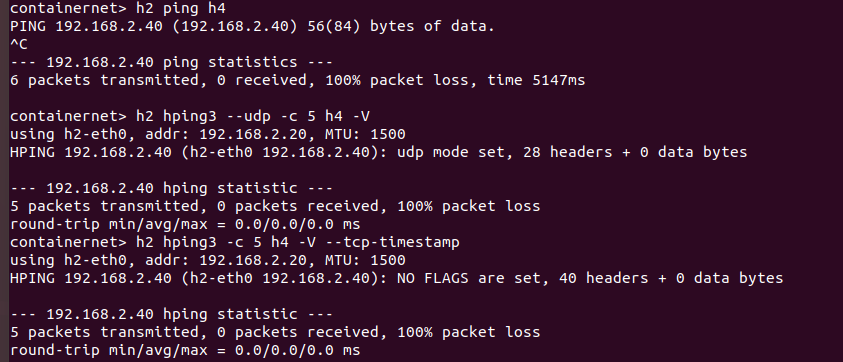


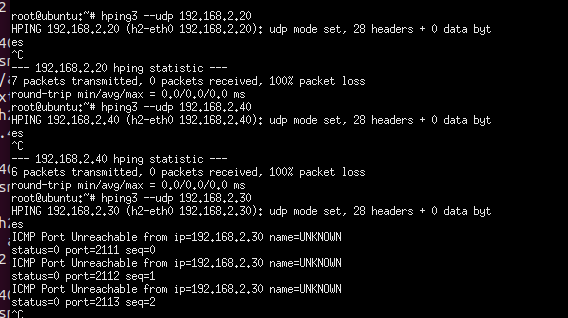
IV-21 Only HTTP traffic from h2 has been blocked

Lab Assessment 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.*

To complete this question, I need to create a new mac address-based policy in the layer 2 level. From the question we can see that we need to block all information from h2 to h4. So we can test the ICMP, TCP and UDP traffic after the new rule is implemented. The rule in layer 2 can be: *1,00:00:00:00:00:02,00:00:00:00:00:04*





IV-22 ICMP,UDP,TCP traffic are all be blocked from h2 to h4

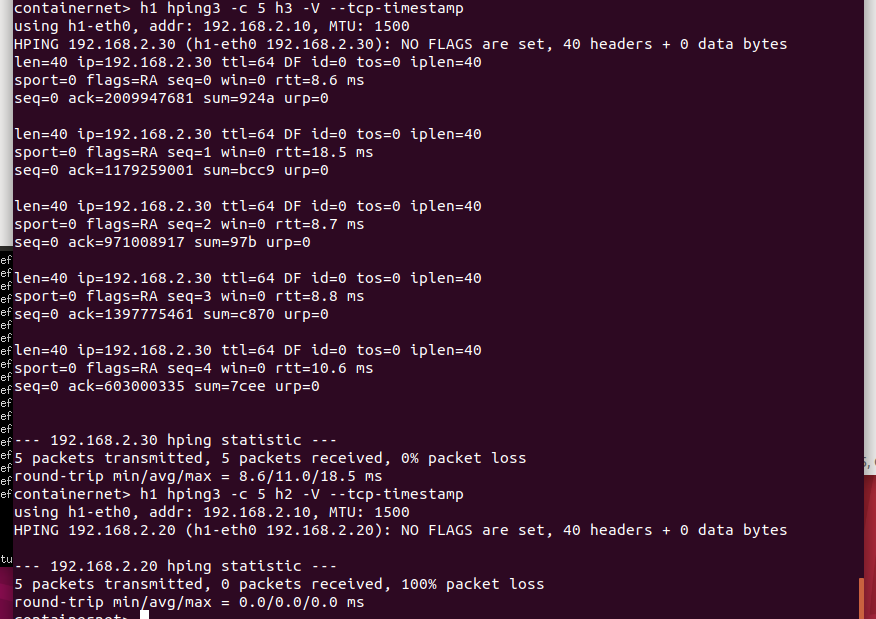
Lab Assessment 7):

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

Here we need to block all TCP traffic from h1 to h2, so it is all ports in the new rule.

New rule is: *4,any,any,192.168.2.10,192.168.2.20,any,any,tcp*

Then we test traffic by command: *h1 hping3 -c 5 h2 -V --tcp-timestamp*. The test shows that h1 can go to h3 for tcp traffic but tcp traffic with h2 is blocked.



IV-23 TCP traffic from h1 to h3 and h1 to h2

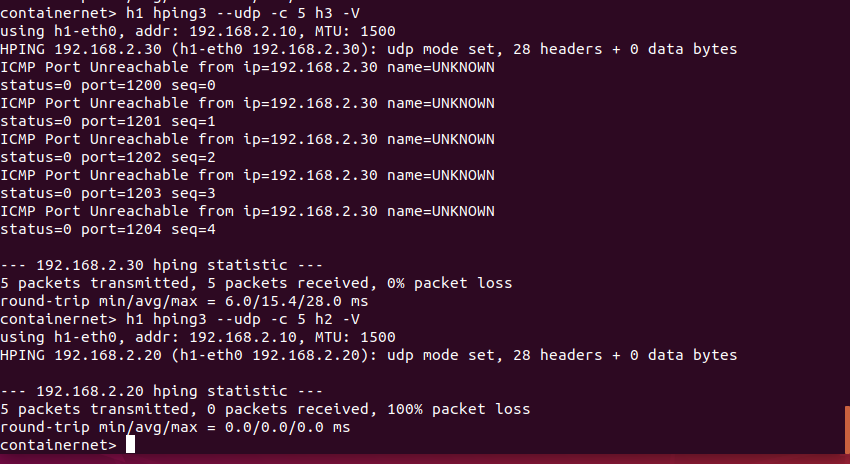
Lab Assessment 8):

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

Here we need to block all UDP traffic from h1 to h2, so it is all ports in the new rule.

New rule is: *5,any,any,192.168.2.10,192.168.2.20,any,any,udp*

Then we test traffic by command: *h1 hping3 --udp -c 5 h2 -V*. The test shows that h1 can go to h3 for UDP traffic but UDP traffic with h2 is blocked.



IV-24 UDP traffic from h1 to h3 and h1 to h2

# Conclusion

## In this experiment, from the installation and setup of a series of software such as mininet, pox and ovs, to the setup of SDN-based openflow firewall rules. I started to build and learn step by step, and initially mastered and understood the protection for computer security in SDN environment. I can set up rules to allow, deny and forward data flow traffic. By setting these rules, we can effectively protect the security of the computer. And in which I also found that when waking up the pox rules, there is often the problem that the rules fail when operating in mininet. At this point I need to restart the rule to take effect. Guess it may be due to a request timeout at some level of the pox call source code. Adding an error report to the pox code at this point can effectively indicate that status.

# Appendix B: Attached files

|  |  |
| --- | --- |
| l2firewall.config | https://github.com/MatthewLLLL/Adv-Network-Security/blob/main/Project%202/l2firewall.config |
| l3firewall.config | https://github.com/MatthewLLLL/Adv-Network-Security/blob/main/Project%202/l3firewall.config |
| L3Firewall.py | https://github.com/MatthewLLLL/Adv-Network-Security/blob/main/Project%202/L3Firewall.py |

# References

1. Hping3 guide book, available at [https://linux.die.net/man/8/hping3/](https://www.wireshark.org/),
2. Pox at https://noxrepo.github.io/pox-doc/html/.
3. Mininet at http://mininet.org/walkthrough/