SDN-Based DoS Attacks and Mitigation Project

Student Name: Jeremy Escobar

Email: jgescoba@asu.edu

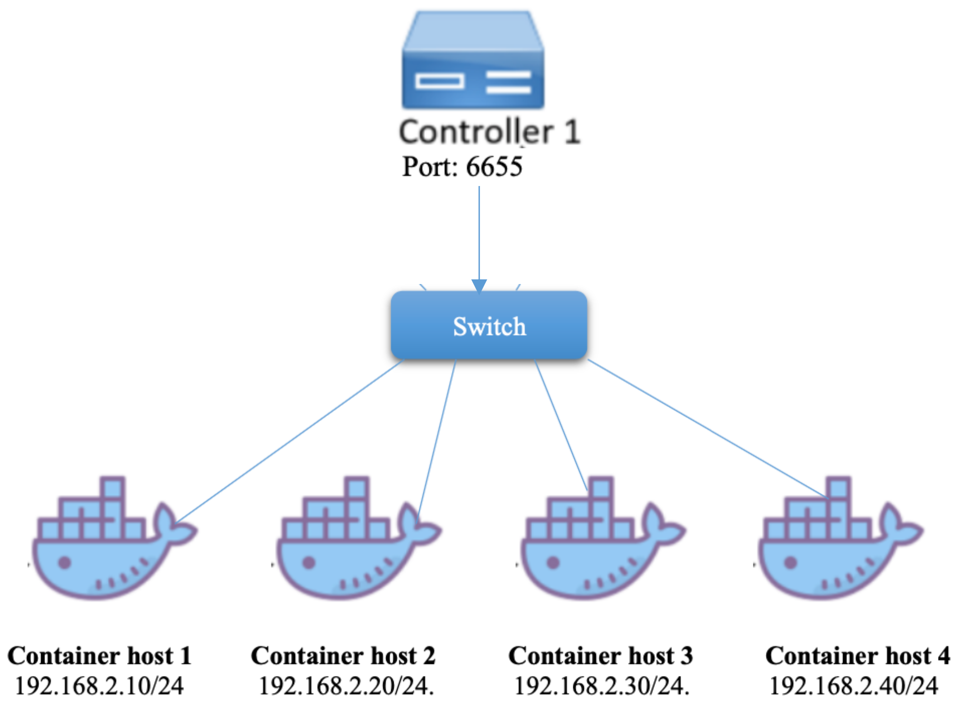
Submission Date: May 31, 2024

Class Name and Term: CSE548 Summer 2024, Session C

I. Project Overview

This project, I aim to explore and mitigate Denial-of-Service (DoS) attacks within a Software-Defined Networking (SDN) environment. The main goal is to understand how DoS attacks impact network resources and services, and then develop strategies to counter these attacks. By setting up an SDN environment with Mininet, POX controller, and Open vSwitch, I will simulate a DoS attack to observe its effects. I will then implement a mitigation technique, such as traffic filtering and rate limiting, using OpenFlow flow rules to prevent and manage the attack. This project enhances my practical skills in network security by engaging with real-world scenarios. Ultimately, the goal is to build a reliable defense against a DoS attack, ensuring network stability and security while deepening my understanding of SDN and network security concepts.

II. Network Setup



III. Software

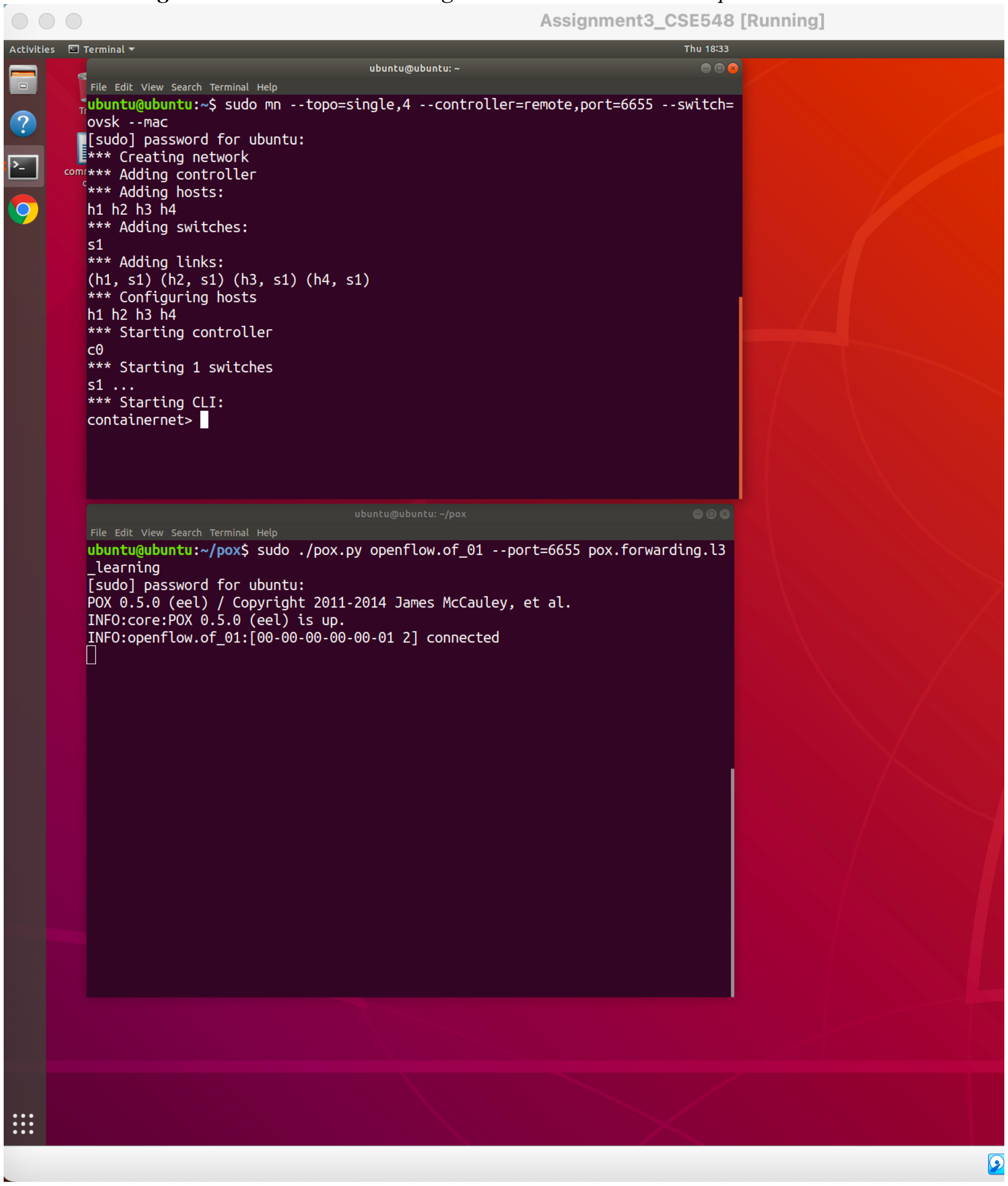
To create this project, I used a virtual machine running Ubuntu 18 vdi, along with Mininet for network simulation, the POX controller for SDN management, and Open vSwitch (OVS) for traffic control. Python was utilized for scripting the DoS attack mitigation, with VS Code serving as my primary code editor. Additionally, I employed tools like hping3 for generating attack traffic and wget for downloading necessary packages.

IV. Project Description

After installing all necessary dependencies and setting up the VM and all necessary tools I first start the project by creating the mininet switch and the pox controller.

Step 1: Created the basis of the DoS Attack by setting up the POX controller and mininet switches. Mininet command is ***sudo mn - -topo=single,4 - -controller=remote,port=6655 - -switch=ovsk - -mac***. Next, I ran the POX command ***sudo ./pox.py openflow.of_01 - -port=6655 pox.forwarding.l3learning.***

Figure 1: *two terminals running the mininet command and pox commands.*

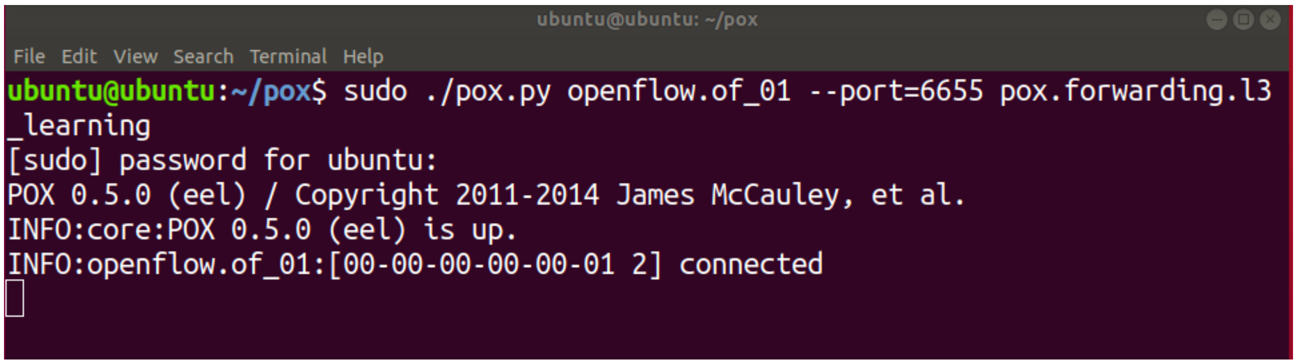

Assignment3_CSE548 [Running]
Activities
Terminal
Thu 18:33
ubuntu@ubuntu: ~
File Edit View Search Terminal Help

```
ubuntu@ubuntu:~$ sudo mn --topo=single,4 --controller=remote,port=6655 --switch=
ovsk --mac
[sudo] password for ubuntu:
*** Creating network
*** Adding controller
*** 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
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet>
```

ubuntu@ubuntu: ~/pox
File Edit View Search Terminal Help

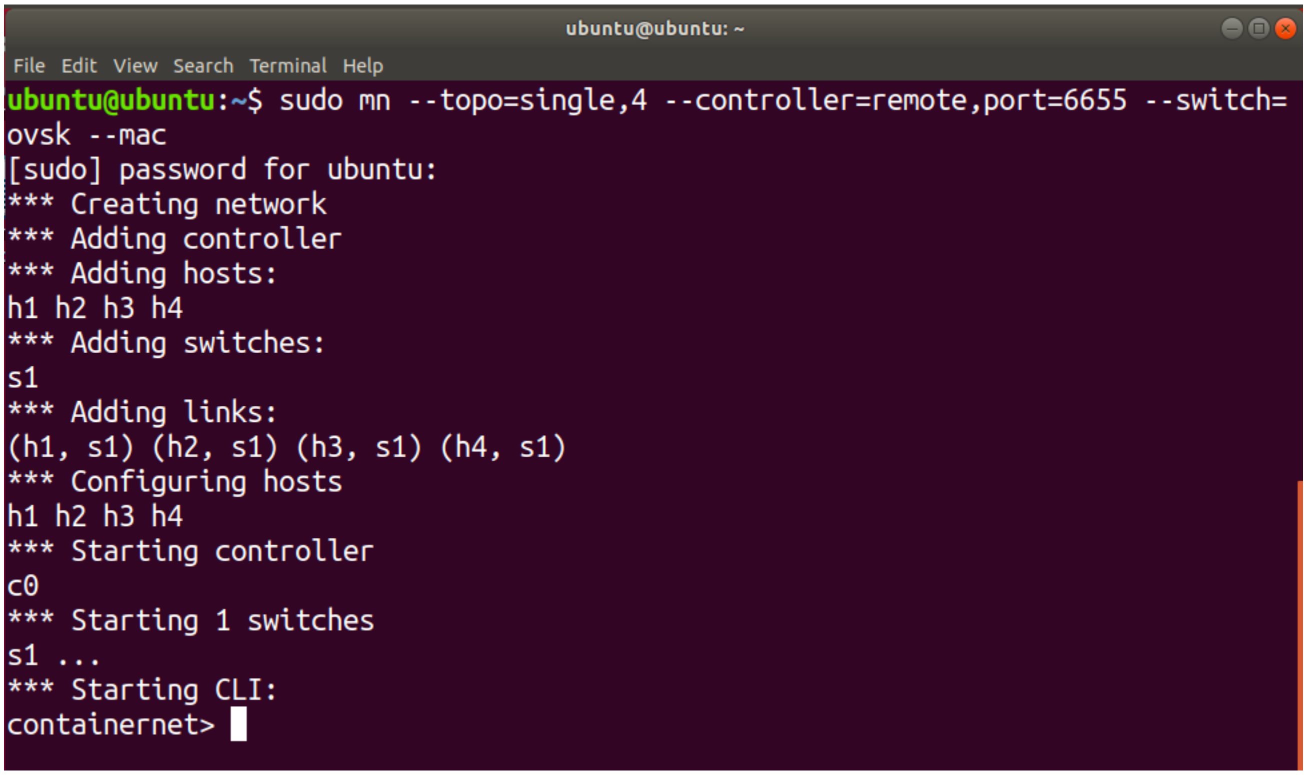
```
ubuntu@ubuntu:~/pox$ sudo ./pox.py openflow.of_01 --port=6655 pox.forwarding.l3
_learning
[sudo] password for ubuntu:
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

Figure 2: *Close up of the pox command*



```
ubuntu@ubuntu:~/pox$ sudo ./pox.py openflow.of_01 --port=6655 pox.forwarding.l3
_learning
[sudo] password for ubuntu:
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

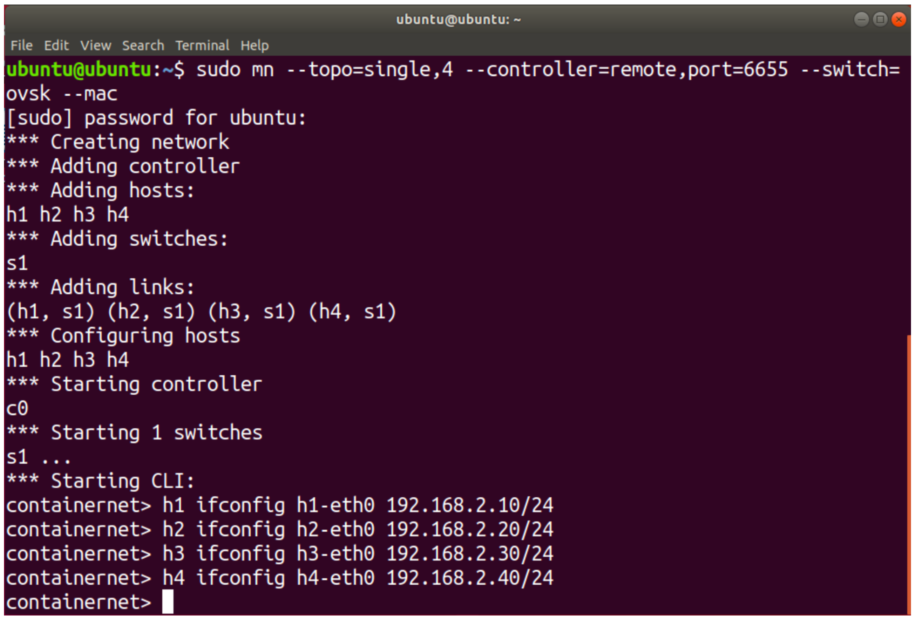
Figure 3: *Close up of the Mininet command*



```
ubuntu@ubuntu:~$ sudo mn --topo=single,4 --controller=remote,port=6655 --switch=
ovsk --mac
[sudo] password for ubuntu:
*** Creating network
*** Adding controller
*** 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
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet>
```

Step 2: I setup each docker interfaces with the correct IP addresses. As demonstrated below in figure 4.

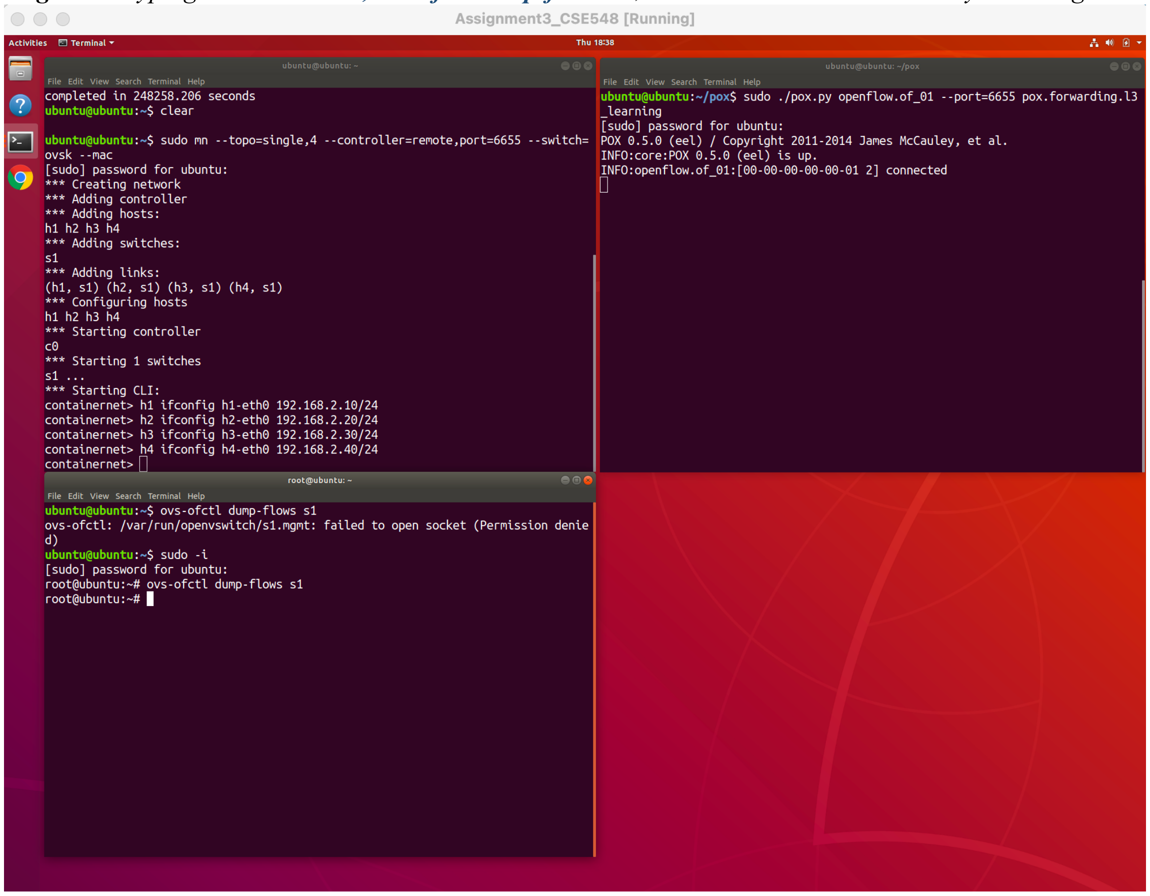
Figure 4: *Typing into the Containernet each interface command for all four hosts h1, h2, h3 and h4.*



```
ubuntu@ubuntu:~$ sudo mn --topo=single,4 --controller=remote,port=6655 --switch=
ovsk --mac
[sudo] password for ubuntu:
*** Creating network
*** Adding controller
*** 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
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet> h1 ifconfig h1-eth0 192.168.2.10/24
containernet> h2 ifconfig h2-eth0 192.168.2.20/24
containernet> h3 ifconfig h3-eth0 192.168.2.30/24
containernet> h4 ifconfig h4-eth0 192.168.2.40/24
containernet>
```

Step 3: Next, I opened up a third terminal to verify no dump details existed as seen below in figure 5.

Figure 5: *Typing into terminal 3, **ovs-ofctl dump-flows s1**, but since not attack occurred yet nothing shows.*



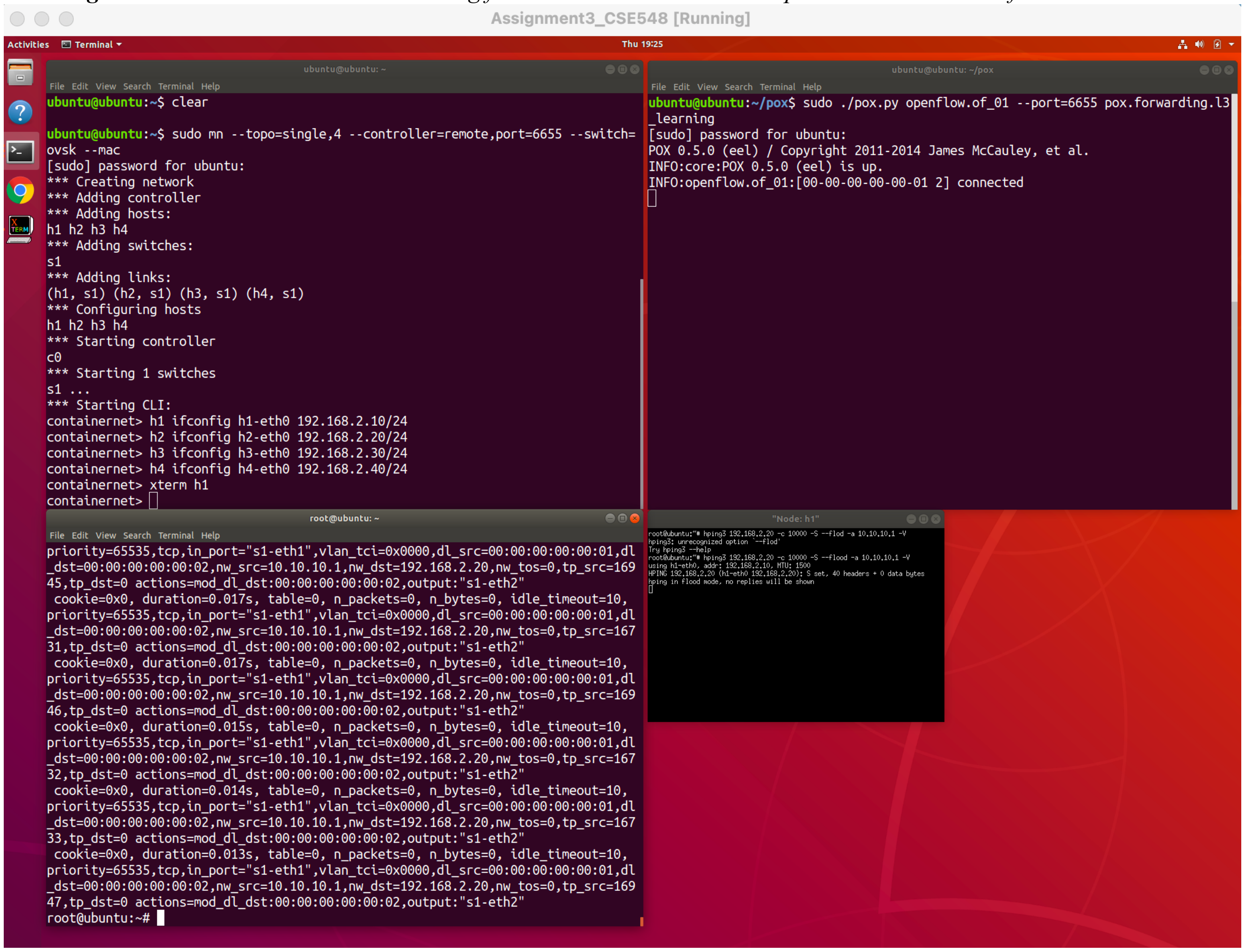
```
completed in 248258.206 seconds
ubuntu@ubuntu:~$ clear

ubuntu@ubuntu:~$ sudo mn --topo=single,4 --controller=remote,port=6655 --switch=
ovsk --mac
[sudo] password for ubuntu:
*** Creating network
*** Adding controller
*** 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
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet> h1 ifconfig h1-eth0 192.168.2.10/24
containernet> h2 ifconfig h2-eth0 192.168.2.20/24
containernet> h3 ifconfig h3-eth0 192.168.2.30/24
containernet> h4 ifconfig h4-eth0 192.168.2.40/24
containernet>
```

```
ubuntu@ubuntu:~/pox$ sudo ./pox.py openflow.of_01 --port=6655 pox.forwarding.l3
_learning
[sudo] password for ubuntu:
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

```
ubuntu@ubuntu:~$ ovs-ofctl dump-flows s1
ovs-ofctl: /var/run/openvswitch/s1.mgmt: failed to open socket (Permission denie
d)
ubuntu@ubuntu:~$ sudo -i
[sudo] password for ubuntu:
root@ubuntu:~# ovs-ofctl dump-flows s1
root@ubuntu:~#
```

Step 4 Next, in Containernet, I typed ***xterm h1***, to load container h1 to use for commands. Next inside the new terminal that popped up which is h1, I typed the DoS attack command using hping3, hping3 192.168.2.20 -c 10000 -S - -flood -a 10.10.10.1 -V. I executed the attack then ran the dump command (***ovs-ofctl dump-flows s1)*** as mentioned earlier and you can see all the packets flooding now below in the terminal on the bottom left.

Figure 6: *Below shows the attack coming from h1 terminal and the dump data in the bottom left terminal window.*



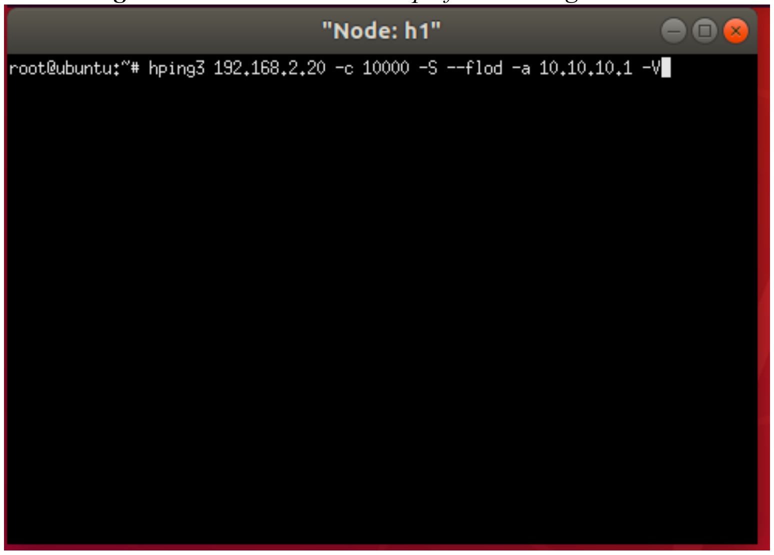
```
ubuntu@ubuntu:~$ clear

ubuntu@ubuntu:~$ sudo mn --topo=single,4 --controller=remote,port=6655 --switch=
ovsk --mac
[sudo] password for ubuntu:
*** Creating network
*** Adding controller
*** 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
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet> h1 ifconfig h1-eth0 192.168.2.10/24
containernet> h2 ifconfig h2-eth0 192.168.2.20/24
containernet> h3 ifconfig h3-eth0 192.168.2.30/24
containernet> h4 ifconfig h4-eth0 192.168.2.40/24
containernet> xterm h1
containernet>
```

```
ubuntu@ubuntu:~/pox$ sudo ./pox.py openflow.of_01 --port=6655 pox.forwarding.l3
_learning
[sudo] password for ubuntu:
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

```
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=169
45,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.017s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=167
31,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.017s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=169
46,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.015s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=167
32,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.014s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=167
33,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.013s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=169
47,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
root@ubuntu:~#
```

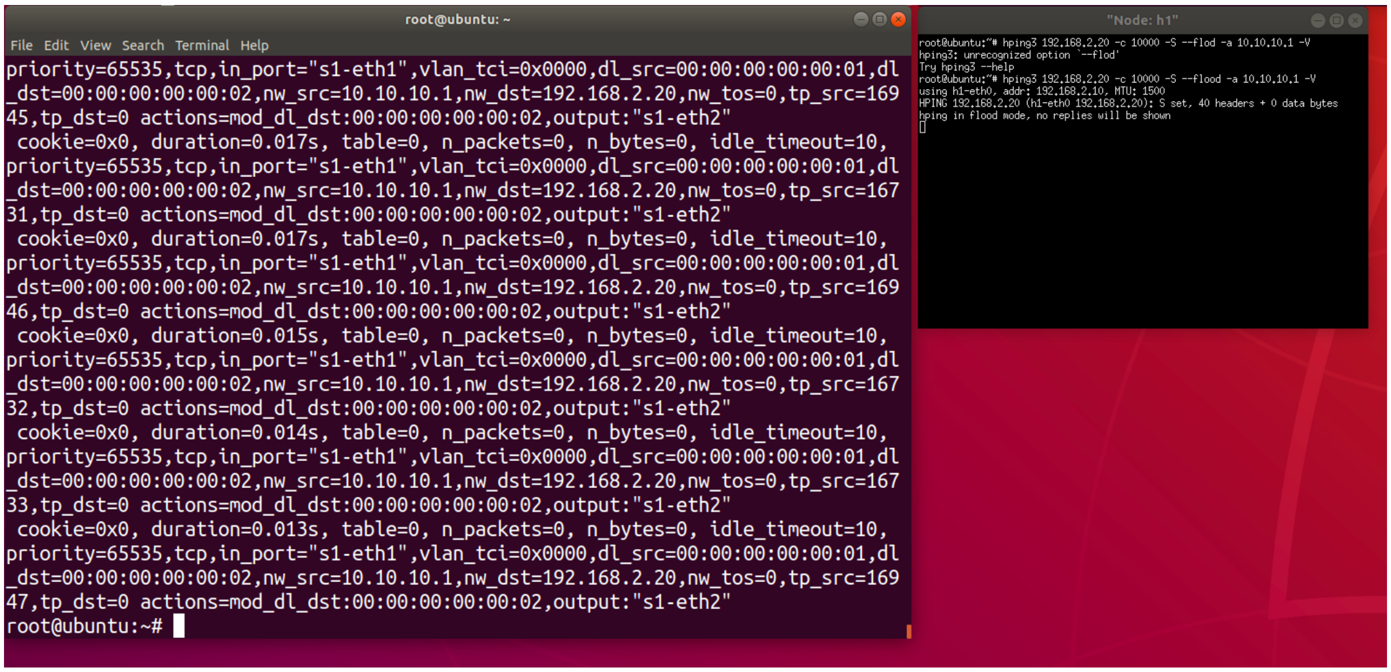
```
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flod -a 10.10.10.1 -V
hping3: unrecognized option `--flod'
Try hping3 --help
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flood -a 10.10.10.1 -V
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 7: *Below shows closeup of h1 running attack DoS.*



```
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flod -a 10.10.10.1 -V
```

Figure 8: *Below shows the attack coming from h1 terminal and the dump data in the bottom left terminal window.*

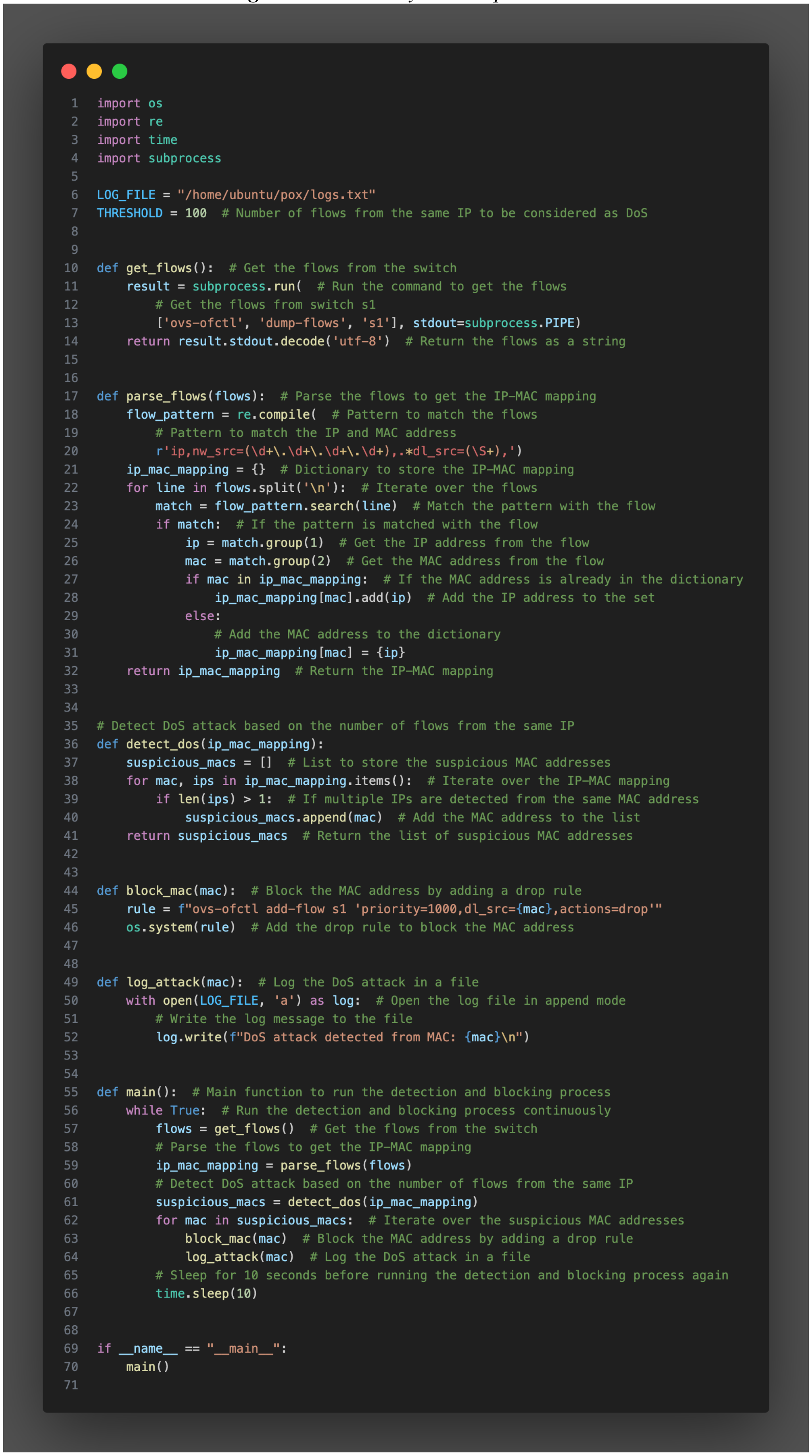


```
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=169
45,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.017s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=167
31,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.017s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=169
46,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.015s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=167
32,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.014s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=167
33,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.013s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=169
47,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
root@ubuntu:~#
```

```
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flod -a 10.10.10.1 -V
hping3: unrecognized option `--flod'
Try hping3 --help
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flood -a 10.10.10.1 -V
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Step 5: Next, to mitigate the DoS attack, The Python script that detects when a single MAC address sends packets with multiple source IPs, indicating a DoS attack. Upon detection, the script updates the switch's flow rules to block the suspicious MAC address, effectively stopping the attack. This automated process ensures the network remains secure and operational. Additionally, the script logs detected attacks for further analysis. By automating detection and mitigation, my solution maintains network integrity without constant manual intervention, enhancing its resilience against DoS attacks.

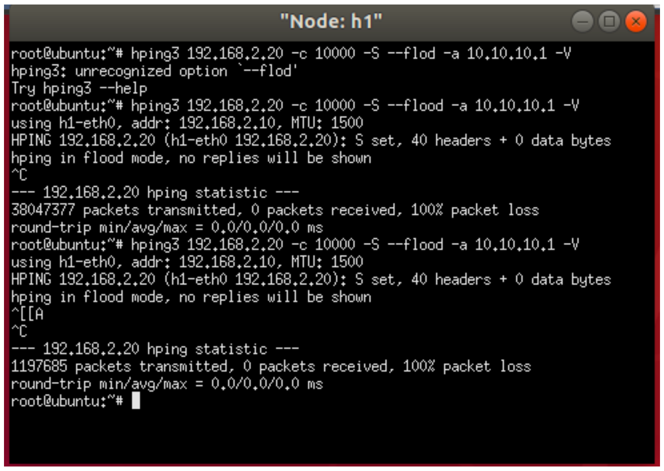
Figure 9: *Below is Python script with notes*



```python
import os
import re
import time
import subprocess

LOG_FILE = "/home/ubuntu/pox/logs.txt"
THRESHOLD = 100  # Number of flows from the same IP to be considered as DoS


def get_flows():  # Get the flows from the switch
    result = subprocess.run(  # Run the command to get the flows
        # Get the flows from switch s1
        ['ovs-ofctl', 'dump-flows', 's1'], stdout=subprocess.PIPE)
    return result.stdout.decode('utf-8')  # Return the flows as a string


def parse_flows(flows):  # Parse the flows to get the IP-MAC mapping
    flow_pattern = re.compile(  # Pattern to match the flows
        # Pattern to match the IP and MAC address
        r'ip,nw_src=(\d+\.\d+\.\d+\.\d+),.*dl_src=(\S+),')
    ip_mac_mapping = {}  # Dictionary to store the IP-MAC mapping
    for line in flows.split('\n'):  # Iterate over the flows
        match = flow_pattern.search(line)  # Match the pattern with the flow
        if match:  # If the pattern is matched with the flow
            ip = match.group(1)  # Get the IP address from the flow
            mac = match.group(2)  # Get the MAC address from the flow
            if mac in ip_mac_mapping:  # If the MAC address is already in the dictionary
                ip_mac_mapping[mac].add(ip)  # Add the IP address to the set
            else:
                # Add the MAC address to the dictionary
                ip_mac_mapping[mac] = {ip}
    return ip_mac_mapping  # Return the IP-MAC mapping


# Detect DoS attack based on the number of flows from the same IP
def detect_dos(ip_mac_mapping):
    suspicious_macs = []  # List to store the suspicious MAC addresses
    for mac, ips in ip_mac_mapping.items():  # Iterate over the IP-MAC mapping
        if len(ips) > 1:  # If multiple IPs are detected from the same MAC address
            suspicious_macs.append(mac)  # Add the MAC address to the list
    return suspicious_macs  # Return the list of suspicious MAC addresses


def block_mac(mac):  # Block the MAC address by adding a drop rule
    rule = f"ovs-ofctl add-flow s1 'priority=1000,dl_src={mac},actions=drop'"
    os.system(rule)  # Add the drop rule to block the MAC address


def log_attack(mac):  # Log the DoS attack in a file
    with open(LOG_FILE, 'a') as log:  # Open the log file in append mode
        # Write the log message to the file
        log.write(f"DoS attack detected from MAC: {mac}\n")


def main():  # Main function to run the detection and blocking process
    while True:  # Run the detection and blocking process continuously
        flows = get_flows()  # Get the flows from the switch
        # Parse the flows to get the IP-MAC mapping
        ip_mac_mapping = parse_flows(flows)
        # Detect DoS attack based on the number of flows from the same IP
        suspicious_macs = detect_dos(ip_mac_mapping)
        for mac in suspicious_macs:  # Iterate over the suspicious MAC addresses
            block_mac(mac)  # Block the MAC address by adding a drop rule
            log_attack(mac)  # Log the DoS attack in a file
        # Sleep for 10 seconds before running the detection and blocking process again
        time.sleep(10)


if __name__ == "__main__":
    main()
```

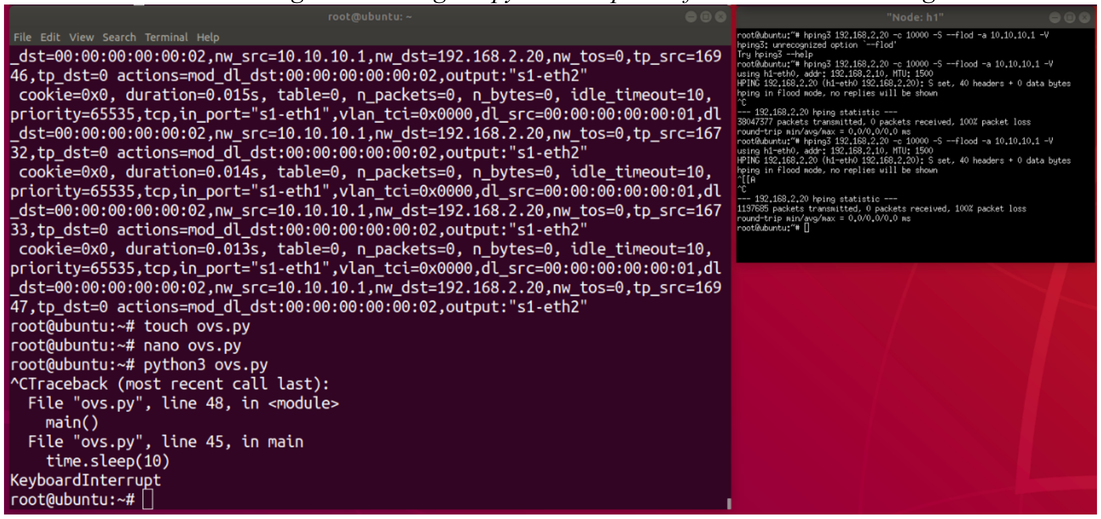
Step 6: After implementing the Python script and running it when I create an attack it now shows 100% packet loss and when I check the dump, it returns zero data now. Therefore, mitigating the attack successfully. As seen in the two screen shots below.

Figure 10: *Below is h1 terminal close up showing the 100% packet loss with python script running.*



```
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flod -a 10.10.10.1 -V
hping3: unrecognized option `--flod'
Try hping3 --help
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flood -a 10.10.10.1 -V
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.2.20 hping statistic ---
38047377 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flood -a 10.10.10.1 -V
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^[[A
^C
--- 192.168.2.20 hping statistic ---
1197685 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@ubuntu:~# 
```

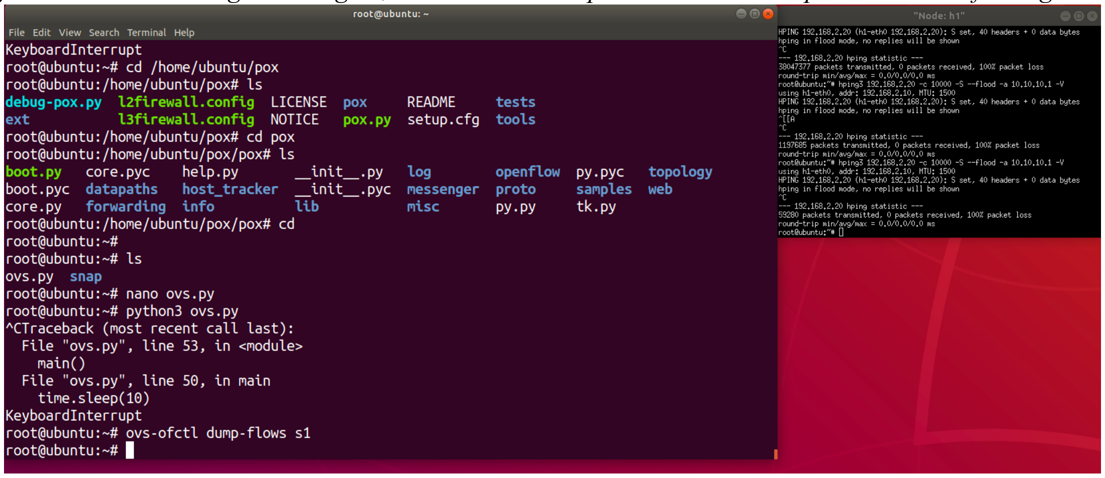
Figure 11: *Below is me running and creating the python script in left terminal and running attack with script on.*



```
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=169
46,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.015s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=167
32,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.014s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=167
33,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
 cookie=0x0, duration=0.013s, table=0, n_packets=0, n_bytes=0, idle_timeout=10,
priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl
_dst=00:00:00:00:00:02,nw_src=10.10.10.1,nw_dst=192.168.2.20,nw_tos=0,tp_src=169
47,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
root@ubuntu:~# touch ovs.py
root@ubuntu:~# nano ovs.py
root@ubuntu:~# python3 ovs.py
^CTraceback (most recent call last):
  File "ovs.py", line 48, in <module>
    main()
  File "ovs.py", line 45, in main
    time.sleep(10)
KeyboardInterrupt
root@ubuntu:~# 
```

```
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flod -a 10.10.10.1 -V
hping3: unrecognized option `--flod'
Try hping3 --help
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flood -a 10.10.10.1 -V
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.2.20 hping statistic ---
38047377 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flood -a 10.10.10.1 -V
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^[[A
^C
--- 192.168.2.20 hping statistic ---
1197685 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@ubuntu:~# 
```

Figure 12: *Below is image showing h1, attack with 100% packet loss and dump command confirming no traffic.*



```
KeyboardInterrupt
root@ubuntu:~# cd /home/ubuntu/pox
root@ubuntu:/home/ubuntu/pox# ls
debug-pox.py  l2firewall.config  LICENSE  pox     README     tests
ext           l3firewall.config  NOTICE   pox.py  setup.cfg  tools
root@ubuntu:/home/ubuntu/pox# cd pox
root@ubuntu:/home/ubuntu/pox/pox# ls
boot.py   core.pyc    help.py       __init__.py   log        openflow  py.pyc   topology
boot.pyc  datapaths   host_tracker  __init__.pyc  messenger  proto     samples  web
core.py   forwarding  info          lib           misc       py.py     tk.py
root@ubuntu:/home/ubuntu/pox/pox# cd
root@ubuntu:~#
root@ubuntu:~# ls
ovs.py  snap
root@ubuntu:~# nano ovs.py
root@ubuntu:~# python3 ovs.py
^CTraceback (most recent call last):
  File "ovs.py", line 53, in <module>
    main()
  File "ovs.py", line 50, in main
    time.sleep(10)
KeyboardInterrupt
root@ubuntu:~# ovs-ofctl dump-flows s1
root@ubuntu:~# 
```

```
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.2.20 hping statistic ---
38047377 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flood -a 10.10.10.1 -V
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^[[A
^C
--- 192.168.2.20 hping statistic ---
1197685 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@ubuntu:~# hping3 192.168.2.20 -c 10000 -S --flood -a 10.10.10.1 -V
using h1-eth0, addr: 192.168.2.10, MTU: 1500
HPING 192.168.2.20 (h1-eth0 192.168.2.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.2.20 hping statistic ---
59280 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@ubuntu:~# 
```

V. Conclusion

Through this project, I have gained a better understanding of how a Denial-of-Service (DoS) attack affect Software-Defined Networking (SDN) environments and the importance of implementing security measures. By setting up and configuring tools like Mininet, POX controller, and Open vSwitch, I simulated DoS attacks to observe their impact on the network. Developing a Python script to automatically detect and mitigate these attacks enhanced my practical skills in network security. This hands-on experience has underscored the significance of proactive network monitoring and automated defenses in maintaining network stability and security. Overall, this project has provided valuable insights into SDN and fortified my knowledge of advanced network security strategies.

VI. Appendix B: Attached files

References