Please edit the highlighted portion.

Packet Filter Firewall (iptables) Project

Student Name: Jeremy Escobar

Email: jgescoba@asu.edu

Submission Date: May 23, 2024

Class Name and Term: CSE548 Summer 2024, Session B

I. Project Overview

The goal of the project is to implement two virtual machines: a Client virtual machine and a Gateway virtual machine that should be able to send packets to each other. Using iptables, we can configure a packet filtering firewall that permits or denies network traffic. Basic networking diagnostic for a pair of VMs (Client and Gateway) on an Ubuntu using the setup of two NatNetworks (10.0.2.0/24 and 10.0.1.0/24): ifconfig, route, ping and traceroute. Next, we setup the network configuration to implement firewall filtering rules to permit or deny network traffic (rc.firewall). Finally, install Apache2 web service to test filtering rules on the Gateway vm machine managing the Client's vm traffic.

II. Network Setup

Optional: Not completed by Jeremy Escobar for this lab.

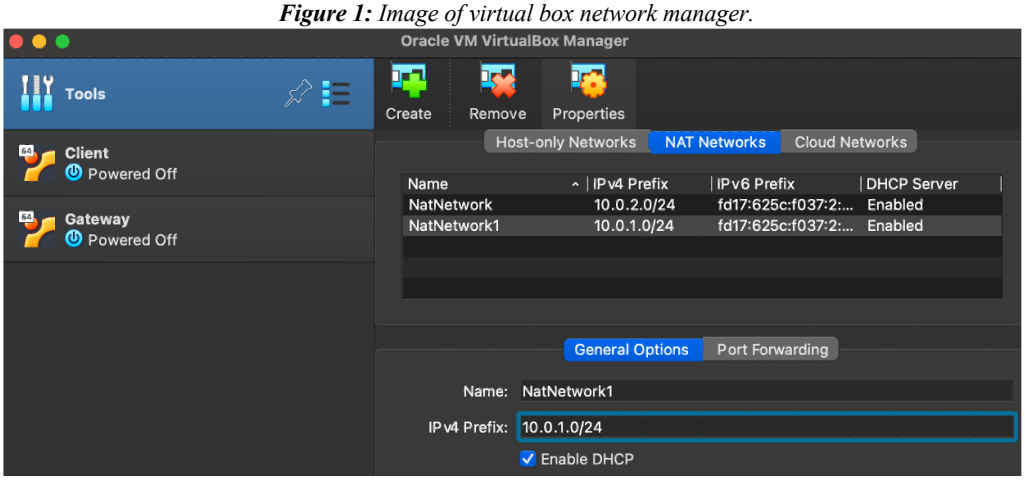
III. Software

In this lab, we've utilized VirtualBox to set up two virtual machines (Client and Gateway). An Ubuntu VDI image, supplied by our instructor, was loaded onto these VMs. To monitor network traffic and check for packet filtering, we installed Wireshark, a packet analysis tool. Additionally, we're employing several network management tools: iptables for establishing routing protocols, Ping for testing connectivity between the client and the gateway server, and Route -n to view the current network routing configurations. Lastly, we used nmap, apache2, index.html, Firefox (loading webpages).

IV. Project Description

Step 1: First step in the process was setting up both VM's (Client and Gateway) according to CS-SYS-00101. Next, I found the settings based on the instructions were different and found that I needed to click "file" the "network manage" to add the two require NatNetworks.

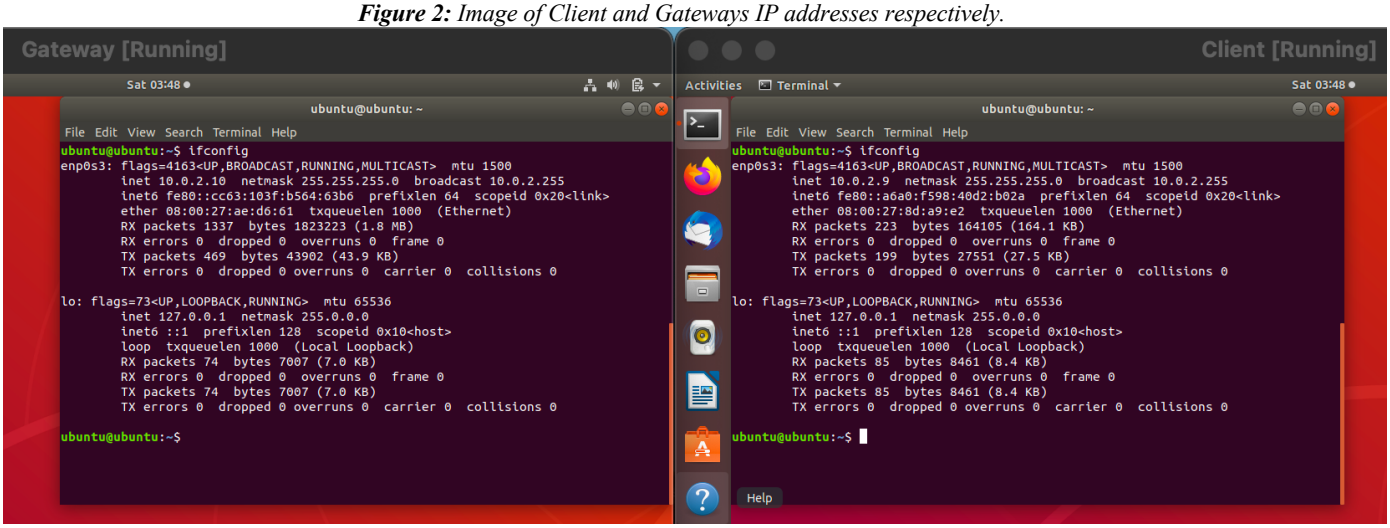
Figure 1: *Image of virtual box network manager.*



Please edit the highlighted portion.

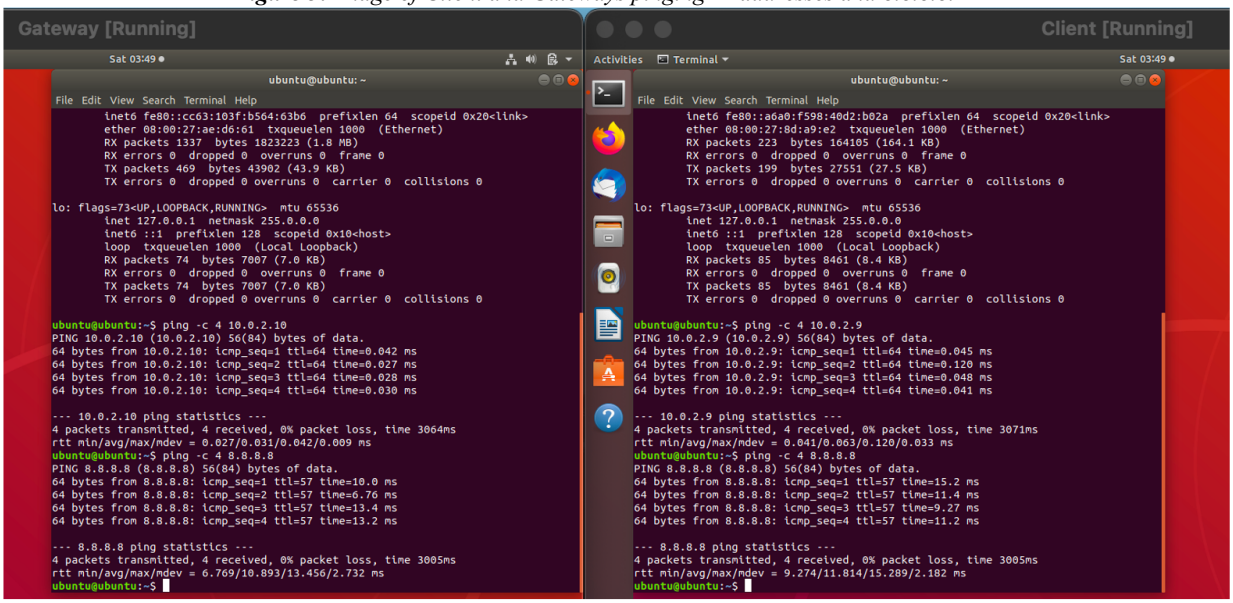
Step 2: Using CS-SYS-00101, I start by verifying the Client and Gateway IP addresses (10.0.2.9 as Client) and (10.0.2.10 as Gateway).

Figure 2: *Image of Client and Gateways IP addresses respectively.*



Step 3: Next tested that both Client and Gateway can ***ping 8.8.8.8*** and their perspective IP addresses successfully.

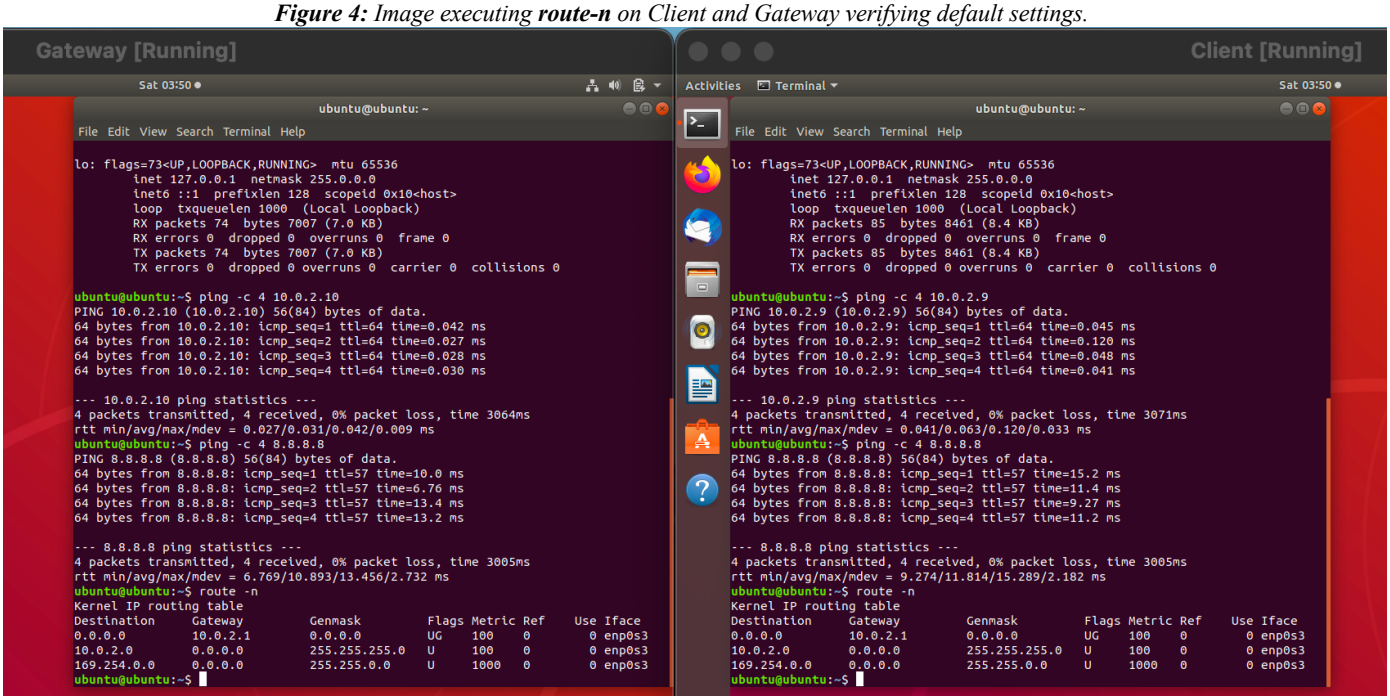
Figure 3: *Image of Client and Gateways pinging IP addresses and 8.8.8.8.*



Please edit the highlighted portion.

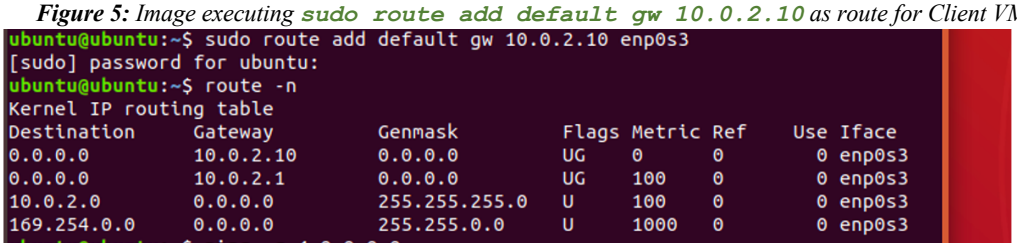
Step 4: Next I checked both routing prior to creating the default gw route.

Figure 4: *Image executing* ***route-n*** *on Client and Gateway verifying default settings.*



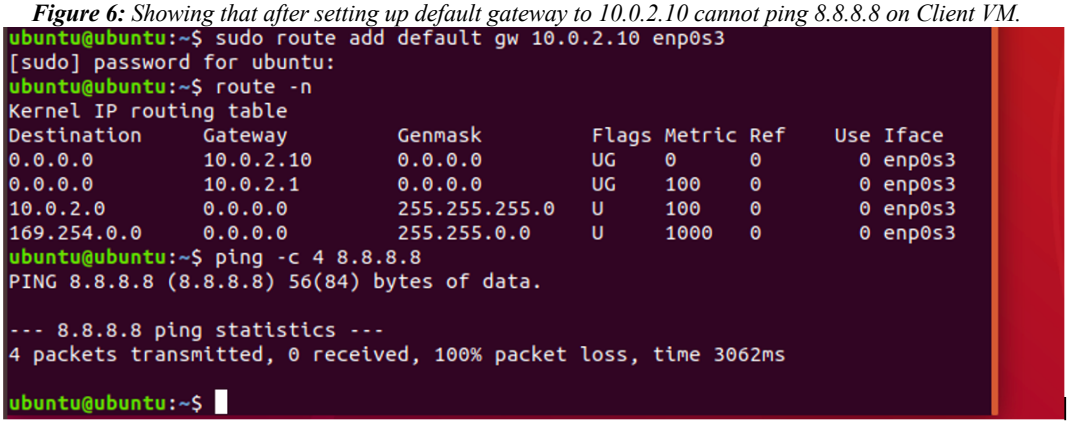
Step 5: Next I will setup the default gateway to point to Gateway VM from Client VM.

Figure 5: *Image executing* `sudo route add default gw 10.0.2.10` *as route for Client VM.*



Step 6: Next, After setting up the default gateway, I checked path with `route -n`, then tried to `ping 8.8.8.8` and failed.

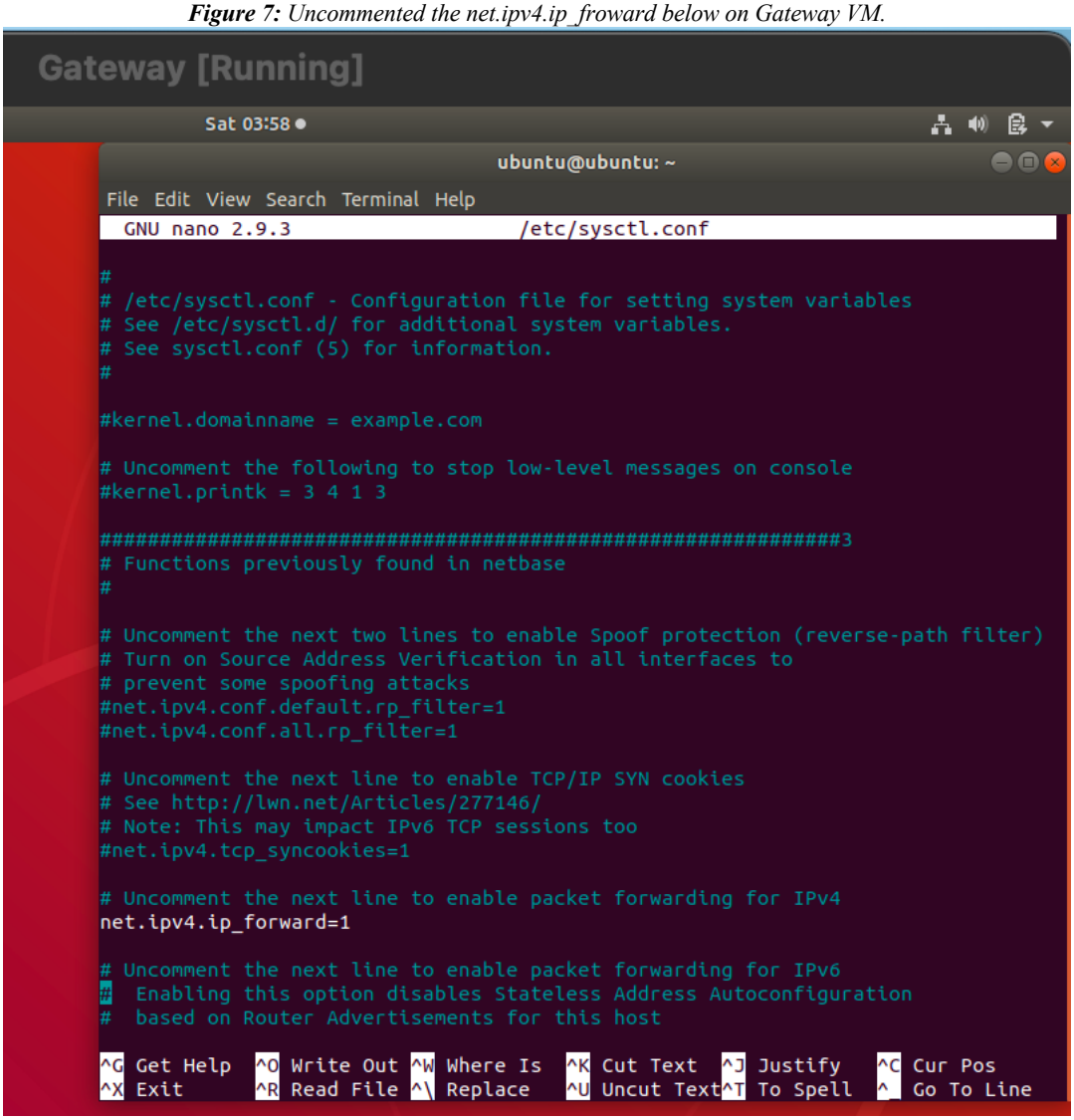
Figure 6: *Showing that after setting up default gateway to 10.0.2.10 cannot ping 8.8.8.8 on Client VM.*



Please edit the highlighted portion.

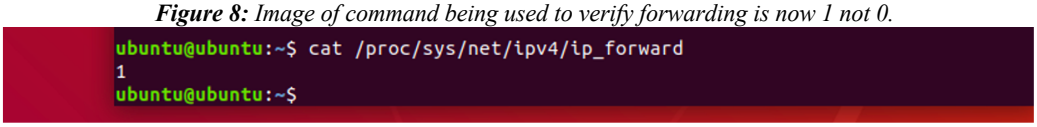
Step 7: Next I `sudo nano /etc/sysctl.conf` the conf file to uncomment net.ipv4.ip_forward=1

Figure 7: *Uncommented the net.ipv4.ip_froward below on Gateway VM.*



Step 8: Next checked to make sure that on Gateway ip_forward was setup with *cat* command.

Figure 8: *Image of command being used to verify forwarding is now 1 not 0.*



Please edit the highlighted portion.

Step 9: Next setup in the gateway the iptables to allow Client vm to access the internet fully again for the time being CS-NET-00002.

Figure 9: *Client is on the right side now able to ping 8.8.8.8 again, and on the left Gateway VM enabling iptables.*

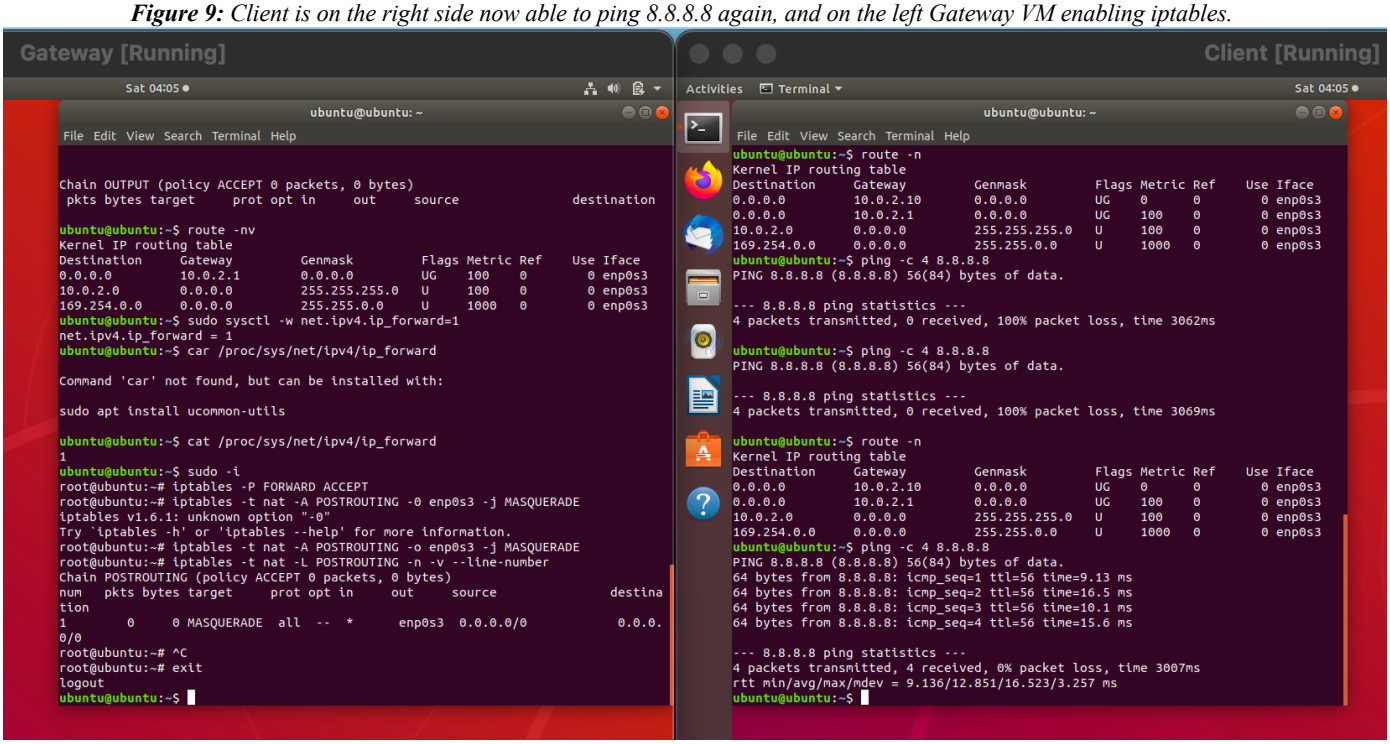
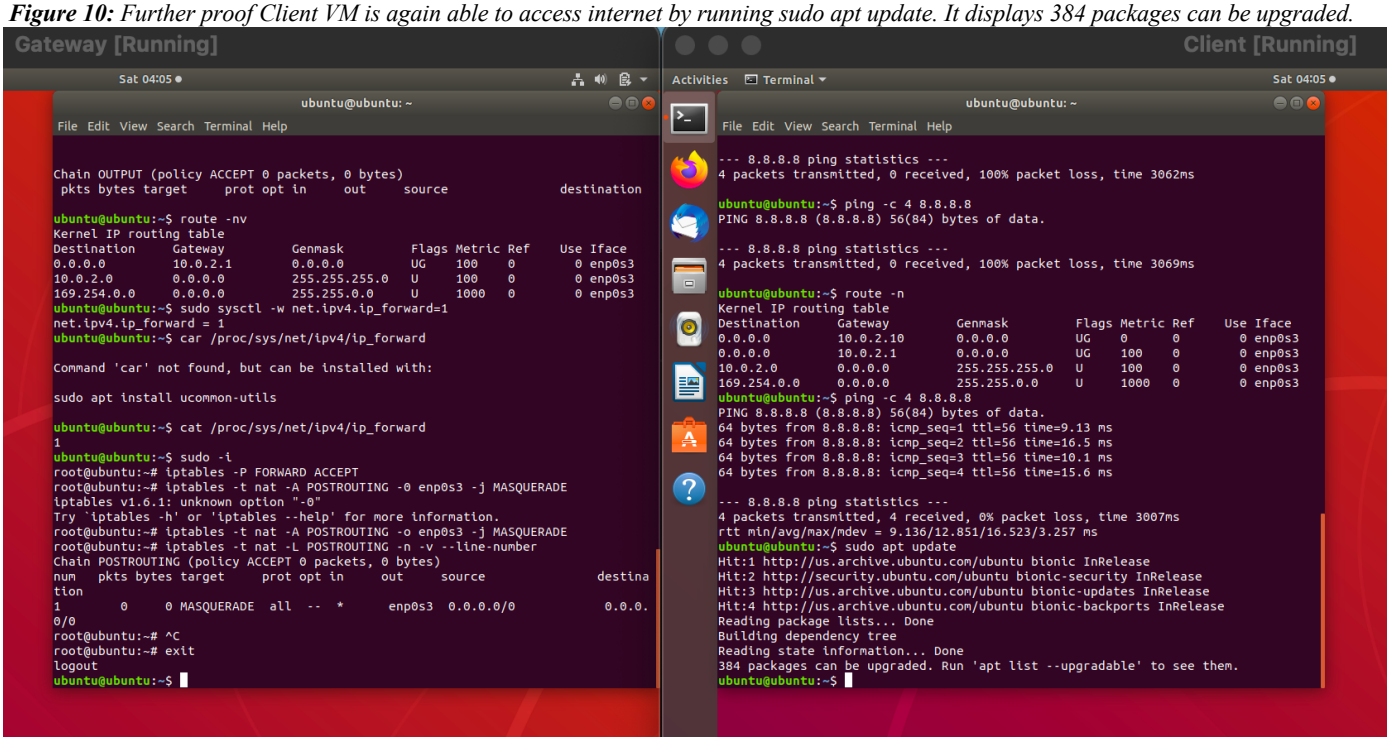


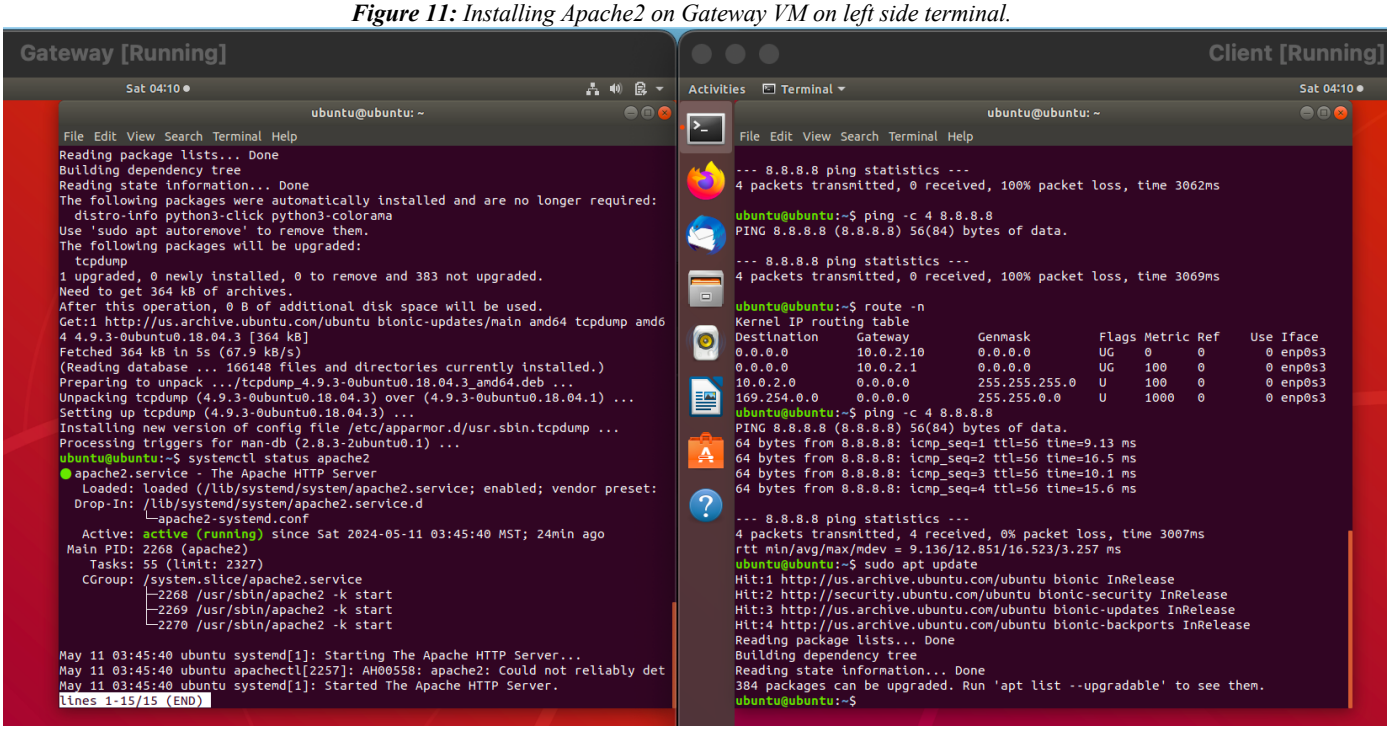
Figure 10: *Further proof Client VM is again able to access internet by running sudo apt update. It displays 384 packages can be upgraded.*



Please edit the highlighted portion.

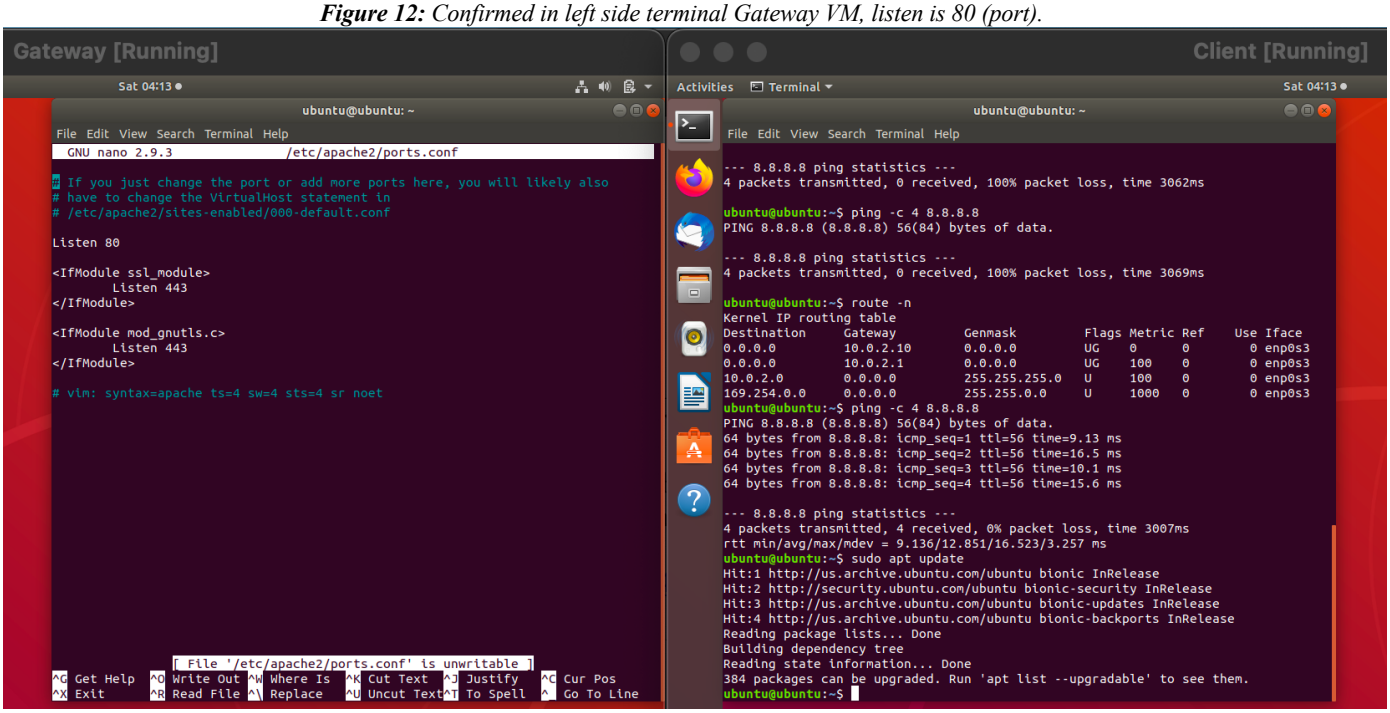
Step 10: Next I installed apache2 on the Gateway Vm for later use.

Figure 11: *Installing Apache2 on Gateway VM on left side terminal.*



Step 11: Then I reviewed the default listen settings for Apache2.

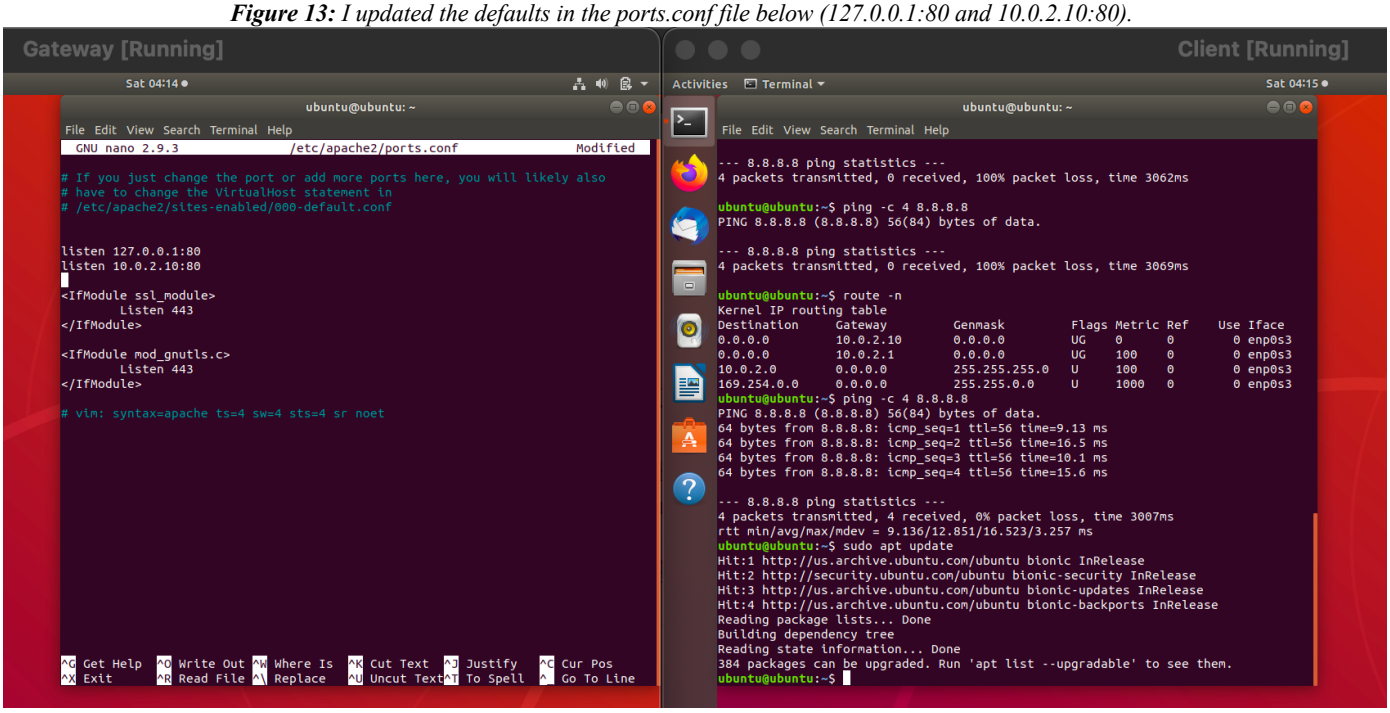
Figure 12: *Confirmed in left side terminal Gateway VM, listen is 80 (port).*



Please edit the highlighted portion.

Step 12: I then updated default port IP addresses to the correct IP addresses on port 80.

Figure 13: *I updated the defaults in the ports.conf file below (127.0.0.1:80 and 10.0.2.10:80).*



Step 13: After doing that I used sudo -i to go in and update the default index.html file for the Apache2 web server and visited the sites from **localhost** on the Gateway VM, then visited **10.0.2.9** on the Client VM side see below:

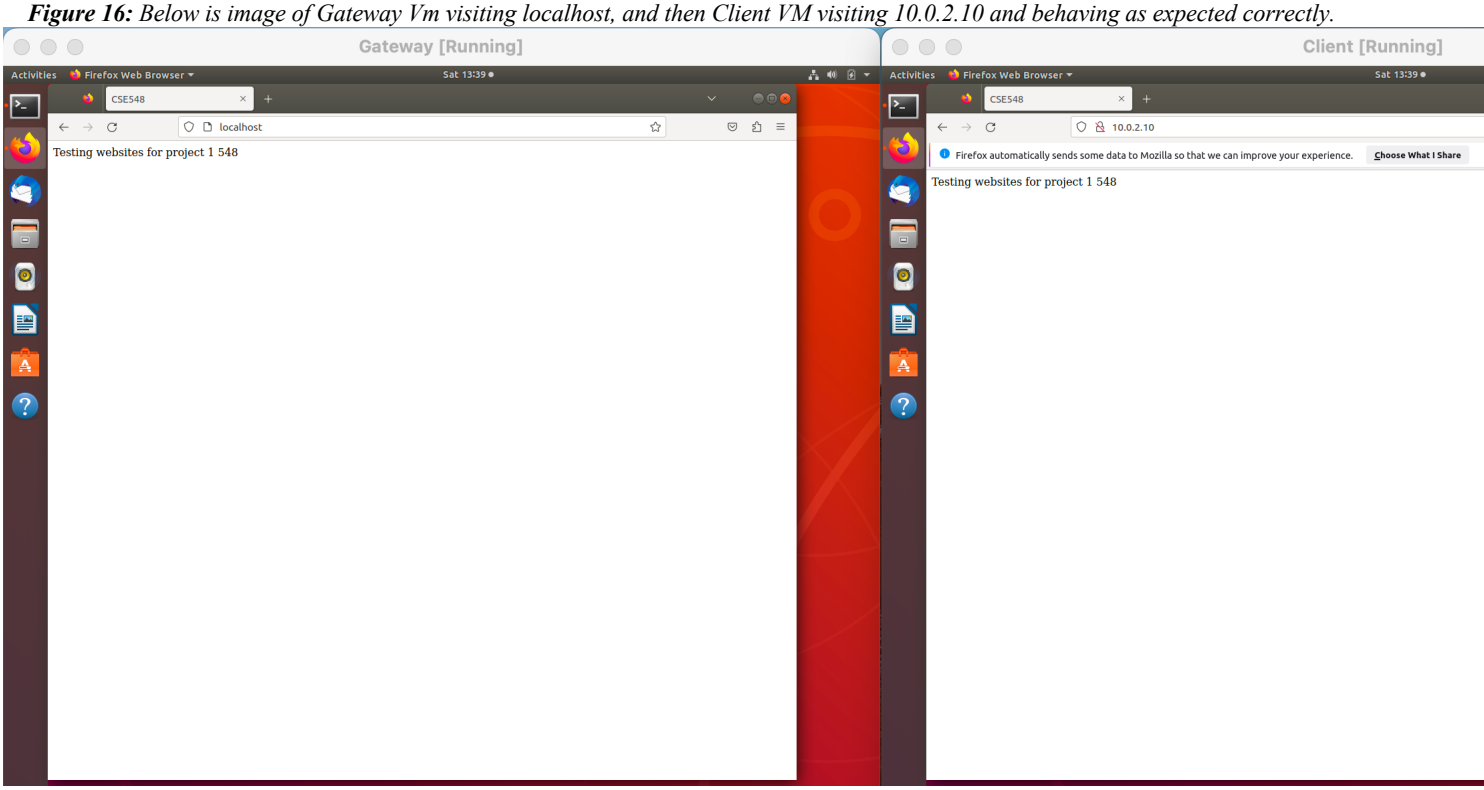
Figure 14: *Below is use sudo -I to go to the default location of the index.html file and create the file..*

```
ubuntu@ubuntu:~$ sudo -i
[sudo] password for ubuntu:
root@ubuntu:~# cd /usr/www/html
-bash: cd: /usr/www/html: No such file or directory
root@ubuntu:~# ls
root@ubuntu:~# cd /var/www/html
root@ubuntu:/var/www/html# ls
index.html
root@ubuntu:/var/www/html# nano index.html
root@ubuntu:/var/www/html#
```

Figure 15: *Below is image of the index.html I updated for use in Gateway and Client Vm's.*

Please edit the highlighted portion.

Figure 16: *Below is image of Gateway Vm visiting localhost, and then Client VM visiting 10.0.2.10 and behaving as expected correctly.*



Step 14: Then I disabled access to the internet to begin setting up for the firewall settings.

Figure 17: *Below enabling DROP on FORWARD, INPUT and OUTPUT.*

```
ubuntu@ubuntu:~$ sudo systemctl restart apache2.service
ubuntu@ubuntu:~$ sudo -i
root@ubuntu:~# iptables -p INPUT DROP
iptables v1.6.1: unknown protocol "input" specified
Try `iptables -h' or 'iptables --help' for more information.
root@ubuntu:~# iptables -P INPUT DROP
root@ubuntu:~# iptables -P OUTPUT DROP
root@ubuntu:~# iptables -P FORWARD DROP
root@ubuntu:~# iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy DROP 2 packets, 656 bytes)
 pkts bytes target     prot opt in     out     source               destination

root@ubuntu:~#
```

Please edit the highlighted portion.

Step 15: After that the Client VM can no longer ping of visit the web see below.

Figure 18: *Client VM unable to visit/ping 8.8.8.8, 10.0.2.10 and web.*

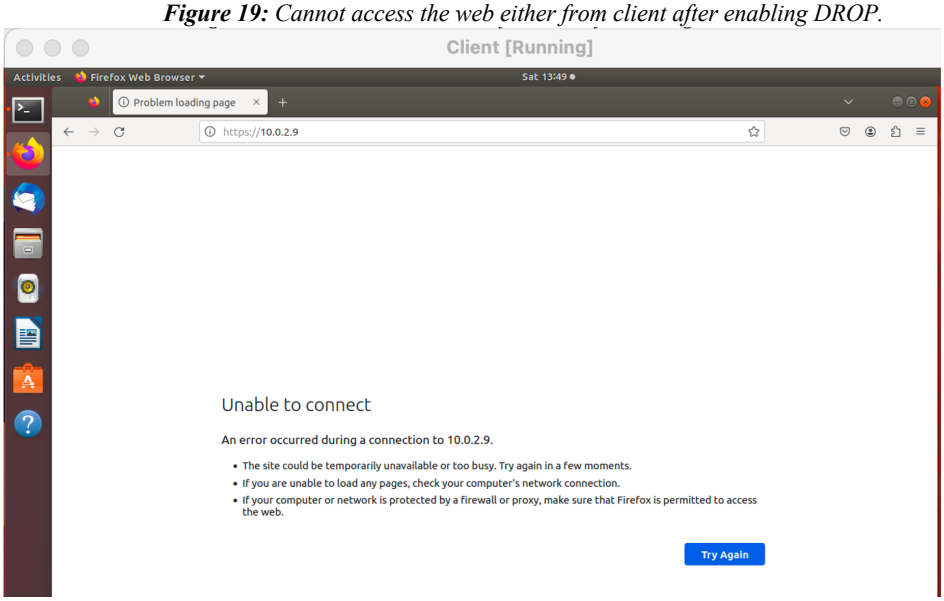
```
ubuntu@ubuntu:~$ ping -c 4 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
^C
--- 10.0.2.10 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ ping -c 3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2039ms

ubuntu@ubuntu:~$
```

Figure 19: *Cannot access the web either from client after enabling DROP.*



Step 16: Next locally I updated the firewall given to us rc.firewall and emailed it to Gateway VM via google so I could download it on the vm machine Gateway.

Figure 20: *Setting up OUTPUT chain rule below.*

```
# Allowed ping from client and server

$IPTABLES -A INPUT -p icmp -j ACCEPT

#####
# 4.3 OUTPUT chain
# Jeremy Rule below
$IPTABLES -A OUTPUT -p TCP --sport 80 -o $Client_NET_IFACE -d $WEB_IP_ADDRESS -j ACCEPT
```

Figure 21: *Setting up the* default FORWARD rules below:

```
#####
# 4.2 FORWARD chain
#


#
# Provide your forwarding rules below
#

# Jeremy Updated example of checking bad tcp packets
$IPTABLES -A FORWARD -p icmp -d $Client_NET_IP -o $Client_NET_IFACE -s 8.8.8.8 -j ACCEPT
$IPTABLES -A FORWARD -p ICMP -s $Client_NET_IP -i $Client_NET_IFACE -d 8.8.8.8 -j ACCEPT
```

Please edit the highlighted portion.

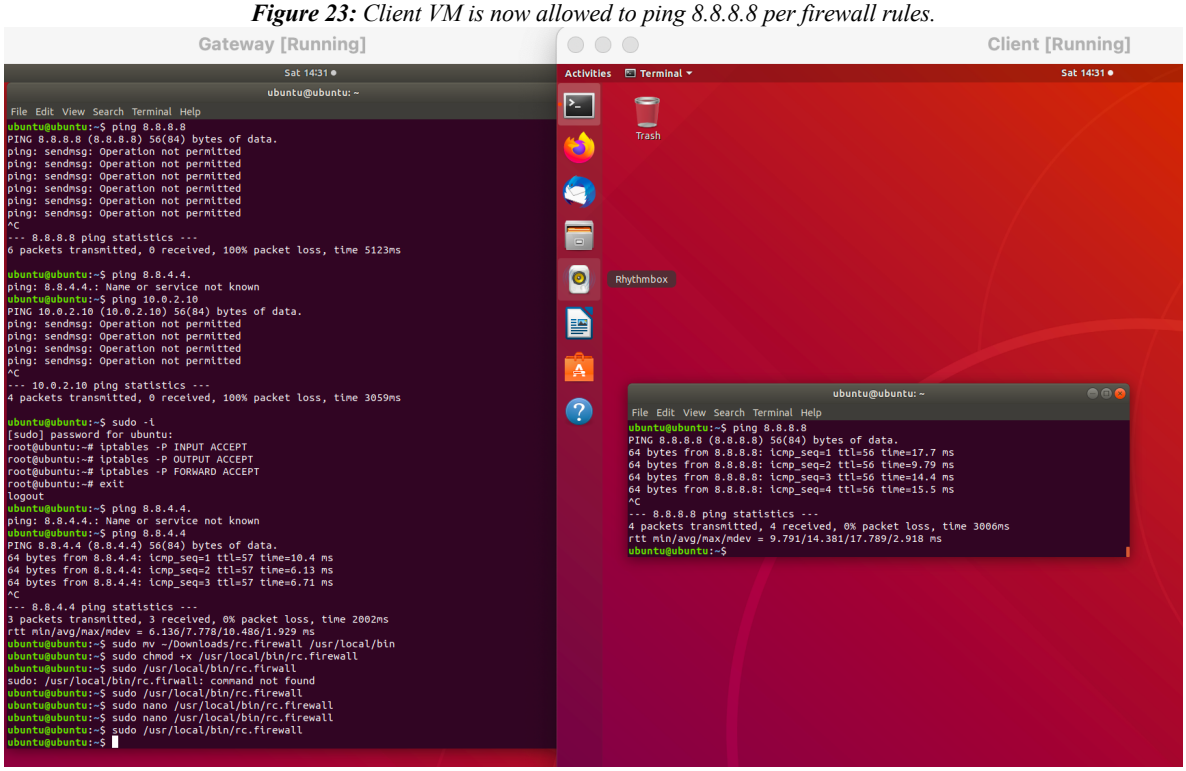
Figure 22: *Setting up the* default POSTROUTING rules below:

```
#####
# 5.2 POSTROUTING chain.
#
#
# Provide your NAT PREROUTING rules (packets go to the internet domain)
# Add your own rule below to only allow ping from client to 8.8.8.8 on internet

# Example: Allow client node to access to all Internet using masquerade
$IPTABLES -t nat -A POSTROUTING -p icmp -o $Internet_IFACE -d 8.8.8.8 -j MASQUERADE
```

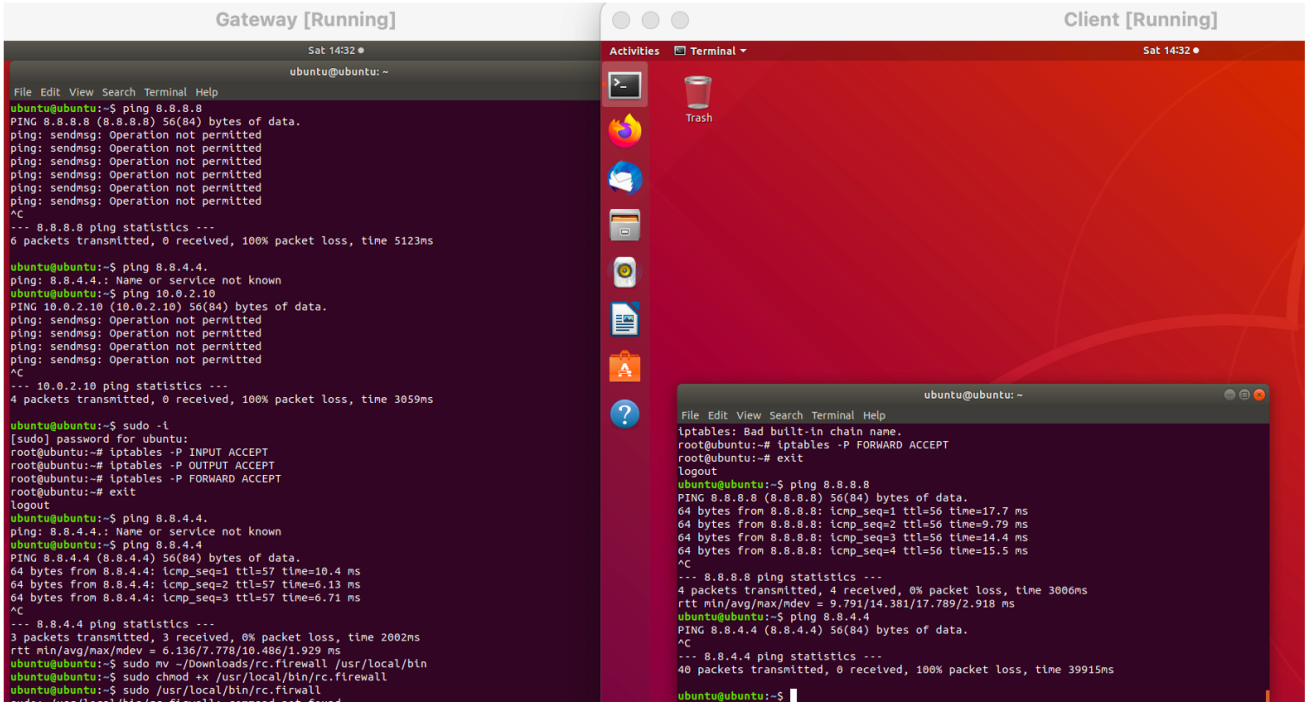
Step 17: Next when trying to ***ping 8.8.8.8*** it worked as expected see below in the Client VM now.

Figure 23: *Client VM is now allowed to ping 8.8.8.8 per firewall rules.*



Please edit the highlighted portion.

Figure 24: *Attempts to ping 8.8.4.4. do not work with 100% packet loss and all other sites fail as well.*



Step 18: Next, I did a UDP and TCP checks below. All worked as expected for both, TCP was good and UDP all ports where open and filtered.

Figure 25: *TCP nmap scan on Gateway VM.*

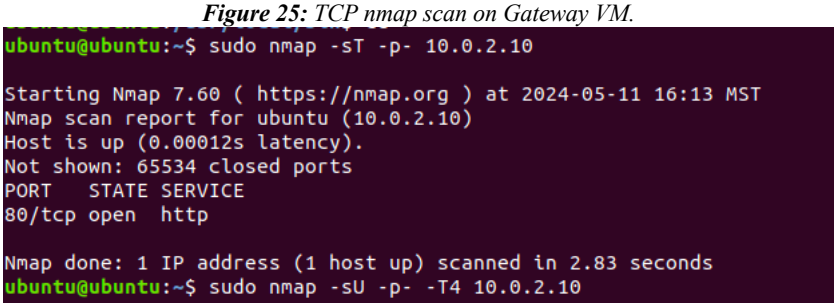
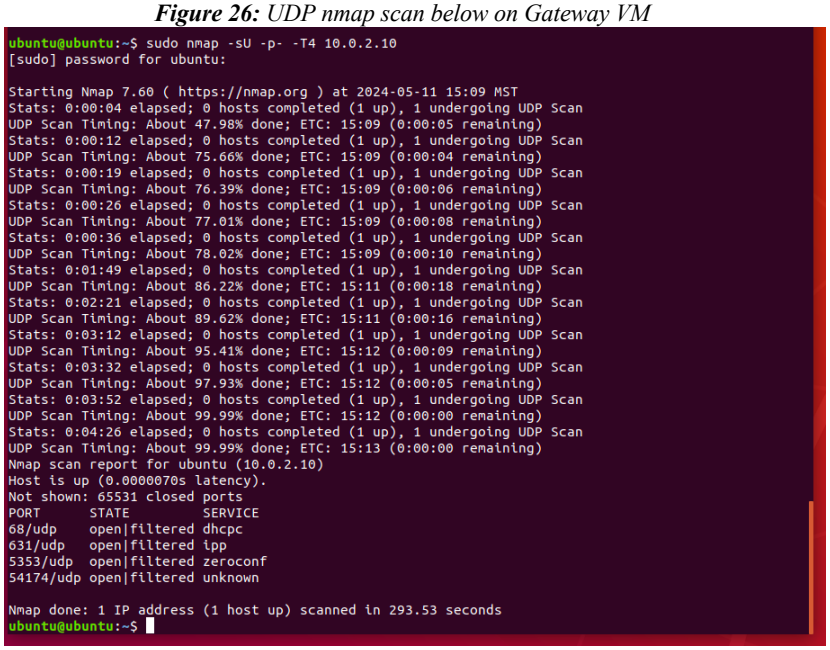


Figure 26: *UDP nmap scan below on Gateway VM*



Please edit the highlighted portion.

V. Conclusion

This lab proved to be quite challenging and difficult, particularly in how it pushed my understanding of iptables. I gained hands-on experience in configuring a gateway to route all traffic from a designated host and in tweaking the iptables rules to observe the flow of packets under the conditions we established. The network tools I discussed in the Software section of my report were instrumental in analyzing the behavior of both the Client and Gateway VMs. Notably, Wireshark was invaluable for scrutinizing the packet exchanges from the client to the gateway. Personally, I encountered few hurdles in setting up the lab. However, I did face challenges with the provided Project1 VDI. I eventually discovered that cloning the Client VM would allow me to create two functional VMs. Another snag I hit involved running the modified rc.firewall script, which initially returned a 'command not found' error.

Appendix B: Attached files

VI. References