Project Report 1 - Packet Filter Firewall

Student Name: Suyash Patel

Email: spate167@asu.edu

Submission Date: 1 February 2022

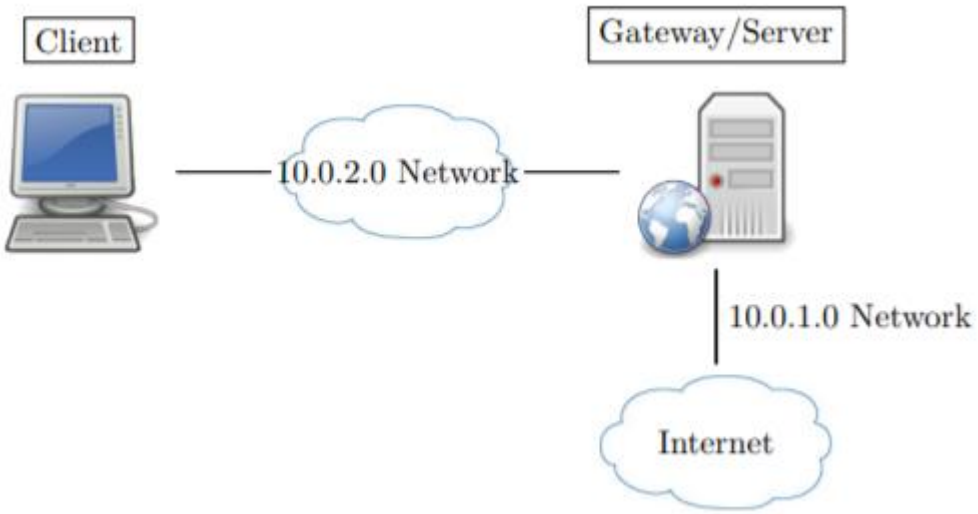
Class Name and Term: CSE548 Spring 2022

I. Project Overview

This project is all about setting up appropriate firewall rules to allow desired network traffic. The lab environment consists of 2 virtual machines: a Client VM and a Gateway VM. Client VM is on an isolated network 10.0.2.0/24 and has an IP of 10.0.2.2. Gateway VM has 2 interfaces – one on the internal network which has an IP of 10.0.2.3 and another that is connected to the internet which has an IP of 10.0.1.5. There are several objectives that need to be accomplished by writing the correct firewall rules. The objectives are as are follows -

1. Client cannot ping the Gateway.
2. Client can access the web server running on the Gateway VM.
3. Client can also ping 8.8.8.8 and all the ping requests need to be routed via the Gateway.
4. Gateway VM can neither ping the Client VM nor can it ping any public IP address.
5. Gateway VM can also not access the Apache web server.
   All the aforementioned objectives have been achieved.

II. Network Setup



Note that in the above setup, the "NAT Network" used to create the internal network has DHCP disabled. The default route on the Client is via 10.0.2.3 (Gateway VM). Both the VM have been assigned static IPs of 10.0.2.2 (Client VM) and 10.0.2.3 (Gateway VM). Netplan command line utility was used to assign static ip addresses. Following are the configuration files used for static ip configuration:

On Gateway VM:

ubuntu@ubuntu:/etc/netplan$ **cat config-gateway.yaml**
network:
  renderer: networkd

```
version: 2
ethernets:
  enp0s3:
    addresses: [10.0.2.3/24]
    dhcp4: no
    dhcp6: no
    nameservers:
      addresses: [208.67.222.222,208.67.220.222]
  enp0s8:
    addresses: []
    dhcp4: yes
    dhcp6: no
    nameservers:
      addresses: [208.67.222.222,208.67.220.220]
```

Note that a static IP address is not assigned on the enp0s8 interface. Since DHCP is enabled on that network, it will automatically get an IP address.

On Client VM:

```
ubuntu@ubuntu:/etc/netplan# cat config-client.yaml
network:
  renderer: networkd
  version: 2
  ethernets:
    enp0s3:
      addresses: [10.0.2.2/24]
      gateway4: 10.0.2.3
      dhcp4: no
      dhcp6: no
```

Client VM has an IP address of 10.0.2.2

Once the config files have been saved, execute “sudo netplan apply” to assign static ips to the interfaces as described above.

Note that on my setup the interface connected to the internet on the Gateway received an IP address of 10.0.1.5. It could be different for different setups since this IP has been assigned via DHCP.

III. Software

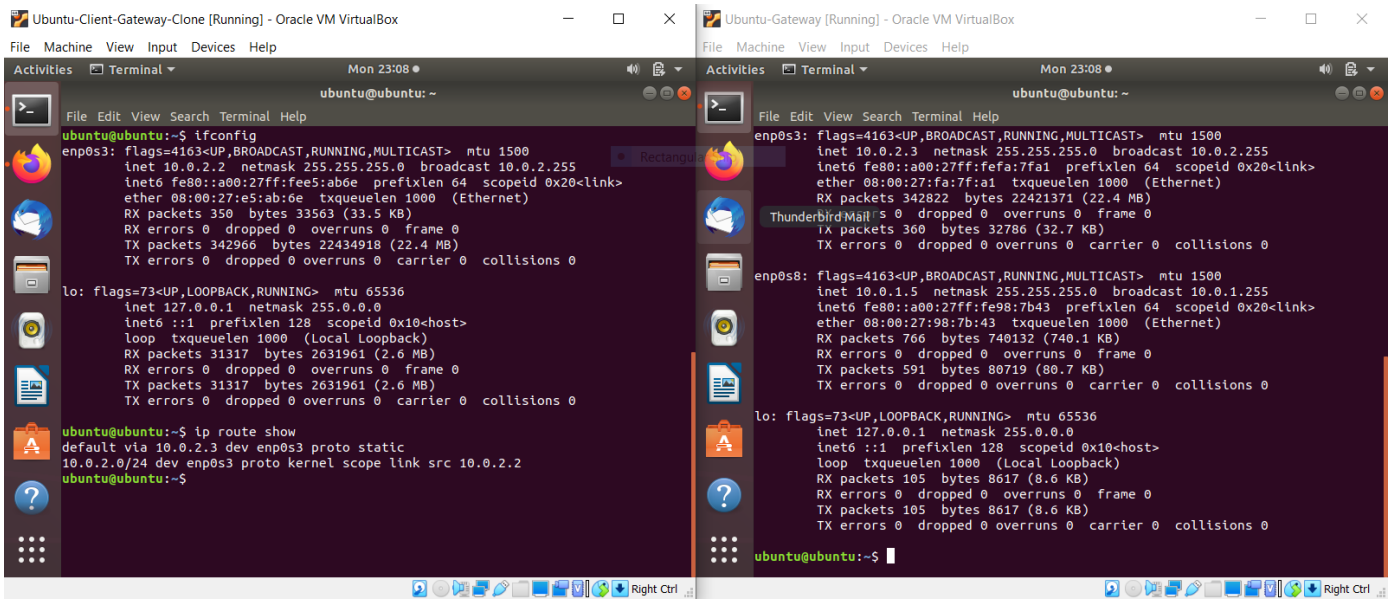
Following software has been used:

1. VirtualBox as a hypervisor
2. Ubuntu VMs
3. Apache web server to setup a basic web server for demonstration purposes.

IV. Project Description

1. Basic Setup – Verifying network configuration

To meet the task requirements mentioned in the lab pdf I have created a shell file called rc.sh which contains all the iptables rules. Before proceeding further please ensure that the network setup is in alignment with the setup described previously. Before executing the rc.sh file, one should be able to ping to the Client VM from the Gateway VM and vice-versa. Gateway VM and Client VM both should be able to access the internet. The Client VM should be able to access the internet via the Gateway VM since the default route on the Client VM is via the Gateway as shown below.



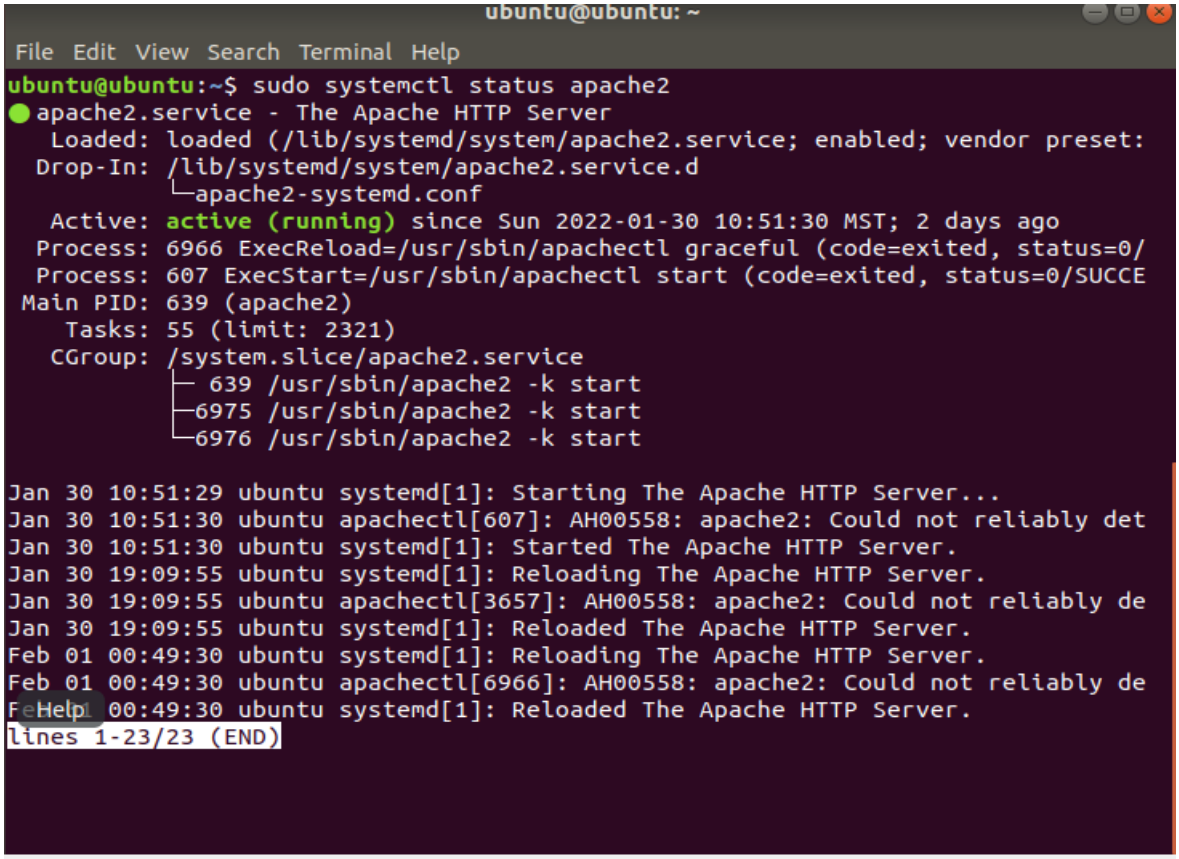
In order to let the packets be forwarded via the Gateway, packet forwarding will need to be enabled on the Gateway if it is not already enabled. To enable packet forwarding execute, **sudo sysctl -w net.ipv4.ip_forward=1.**

2. Setup basic web server

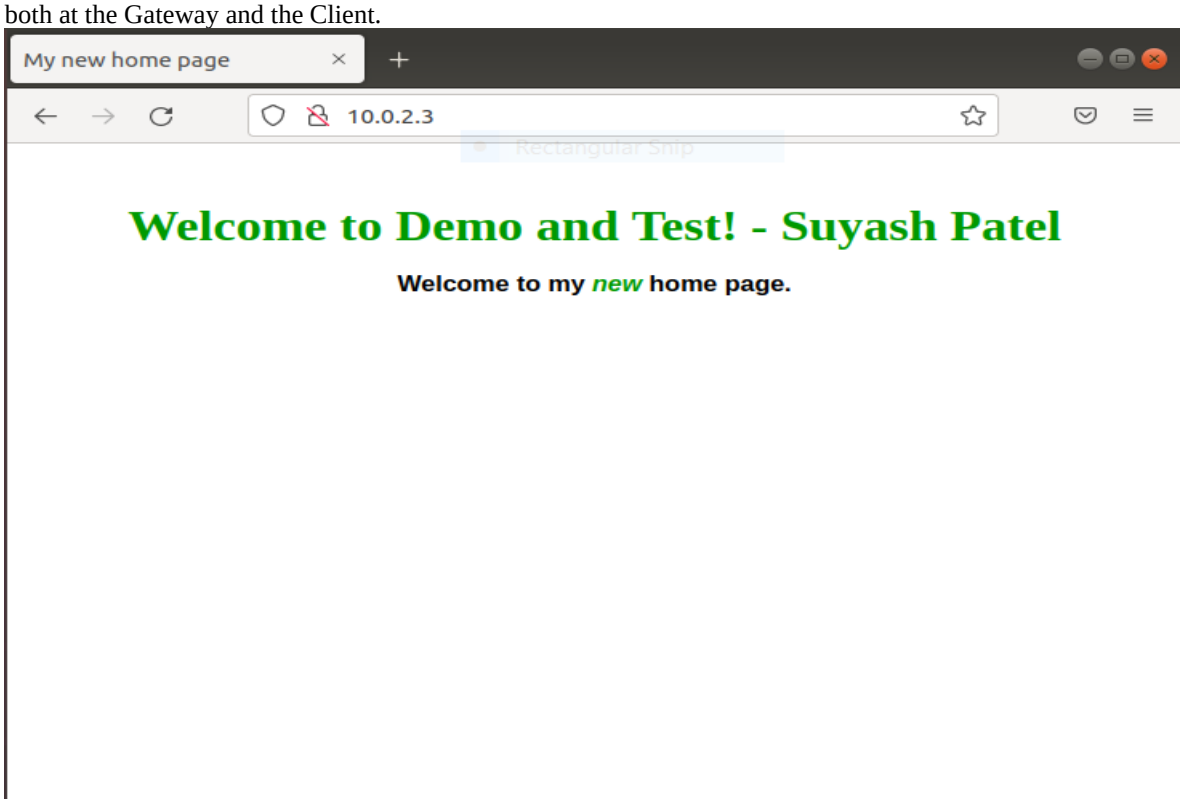
Now before we write the firewall rules, let us setup a basic apache web server as it is a requirement for one of the tasks. To install apache2, run "**sudo apt install apache2**" on the Gateway. There are numerous configurations that can be applied but since we need the web server to serve a html page for demonstration purposes, we stick to just changing the html page the web server serves. Go to /var/www/html on the Gateway and replace the contents of index.html with the following html code.

```
<html>

<head>

<title>My new home page</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

</head>

<body bgcolor="#FFFFFF" text="#000000">

<p align="center"> </p>

<p align="center"><font face="Verdana, Arial, Helvetica, sans-serif"
size="3"><b><font color="#009900" size="6" face="Times New Roman, Times, serif">

Welcome to Demo and Test! - Suyash Patel</font></b></font></p>

<p align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif"
size="3">

Welcome to my <i><font color="#009900">new</font></i> home page.</font></b></p>

</body>

</html>
```

After saving the changes, please run "**sudo systemctl restart apache2**" and then "**sudo systemctl status apache2**" to check the status of the service.

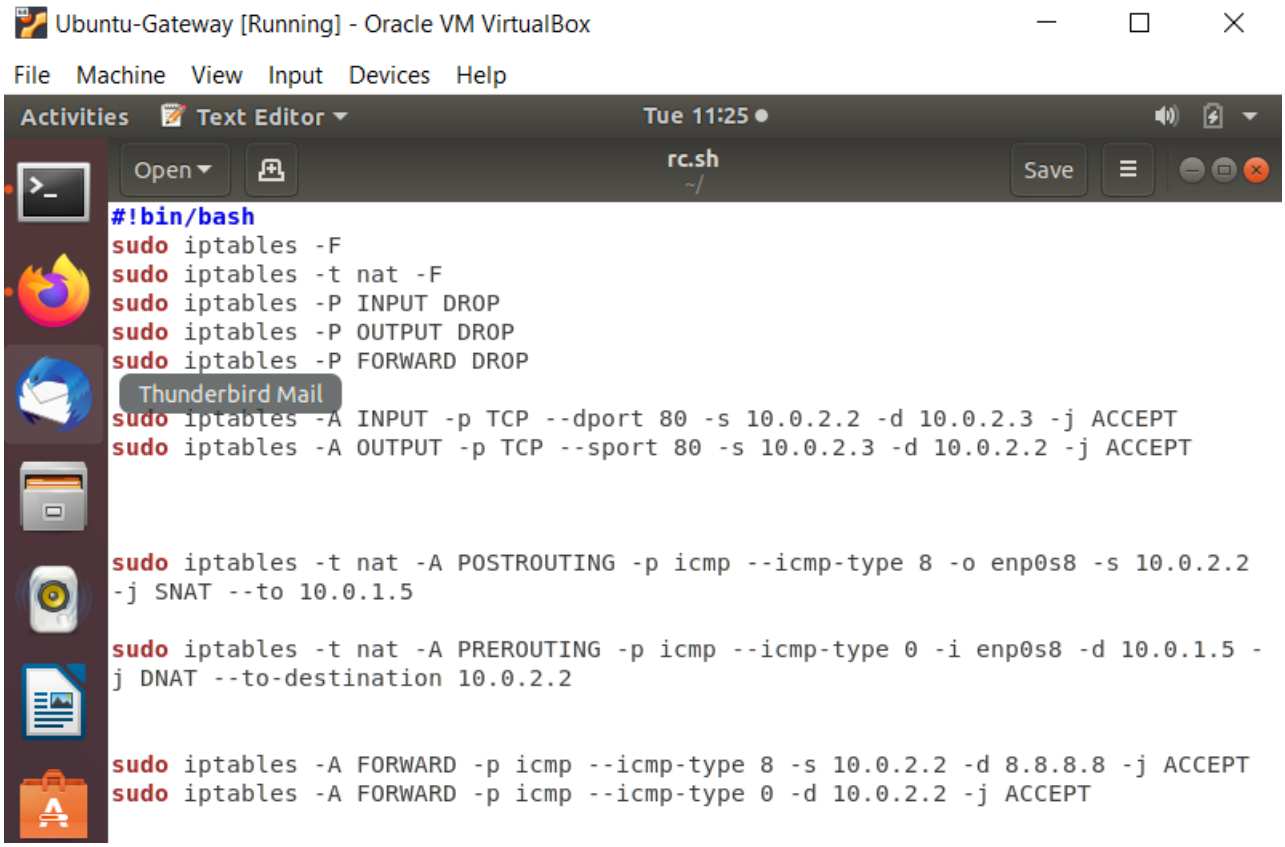


At this point, since no iptables rules have been applied, the following web page should be available on the 10.0.2.3 ip address both at the Gateway and the Client.



3. Setting up Firewall rules

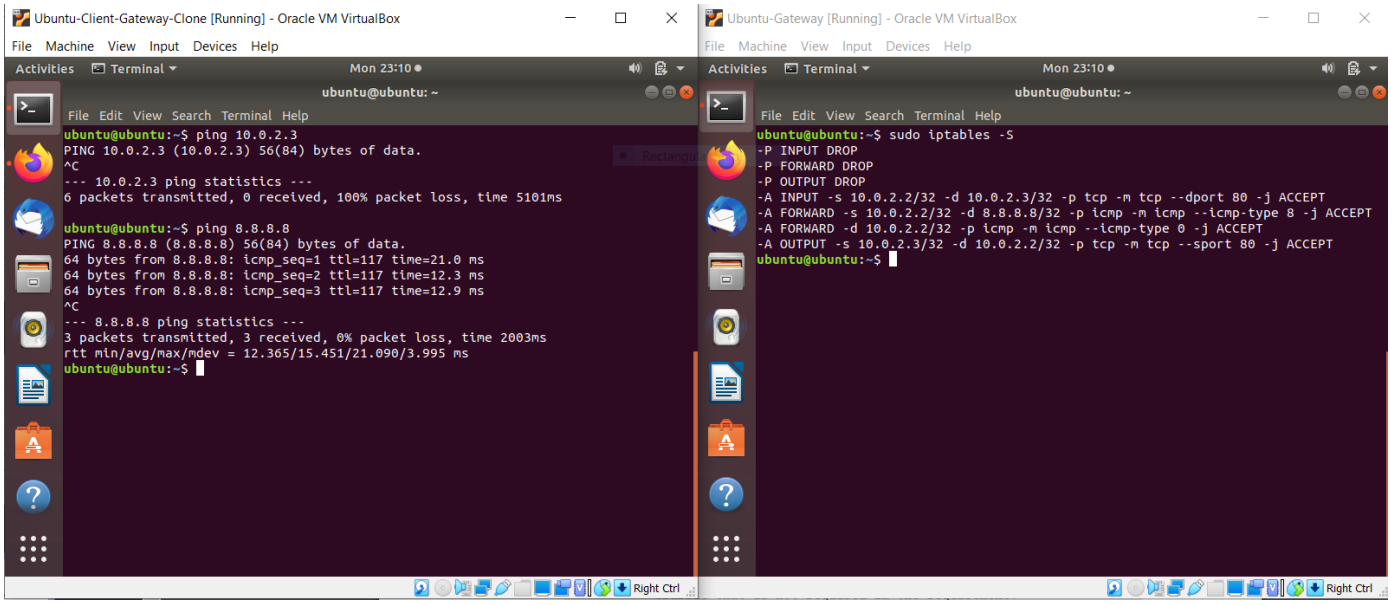
We will write all the rules in a file and save it in a .sh file. One can name the file whatever he/she wants. I have saved all the rules in a file called rc.sh. Following are the contents of the rc.sh file:



Now I will explain each rule.

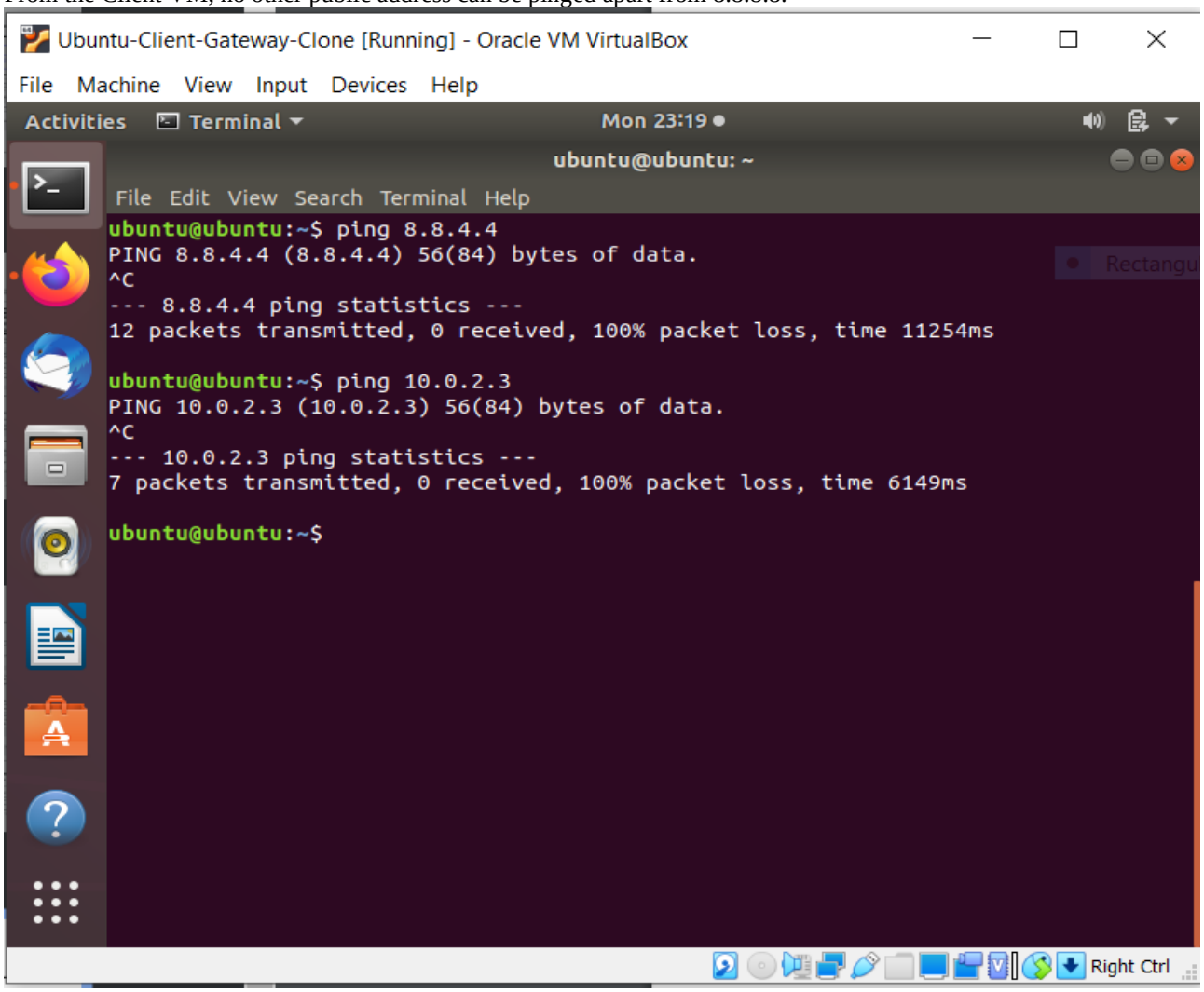
- Lines 1 and 2 flush the pre-existing iptables rules. This is important so that the existing rules don't interfere with the rules that are going to be written. Because these lines are written at the top of the file, every time the file is executed, we start with a blank slate.
- Lines 3 – 5 set the default policy to whitelist i.e all packets that do not satisfy the rules in iptables chains will be dropped. This is done in accordance with the project requirements.
- Lines 6 – 7 are written so that the Client VM can access the web service running on the Gateway VM via the HTTP protocol (port 80). The first rule allows all TCP packets with destination IP 10.0.2.3, source IP 10.0.2.2, and destination port 80 via the INPUT chain. The second rule allows HTTP packets from the web server to reach the Client VM. Note that the Gateway VM wouldn't be able to access this web page since the packets from the Gateway wouldn't match any of the rules and will be dropped since the default rule is DROP.
- Lines 8-11 are to ensure that the Client VM can ping 8.8.8.8 via the Gateway VM and when these ICMP packets exit the Gateway VM, the source IP on these packets needs to be changed as mentioned in the project PDF. Lines 8-9 change the source IP addresses by configuring the PREROUTING and POSTROUTING chains in the NAT table. Once the packets have passed through the PREROUTING/POSTROUTING chains they pass through the FORWARD chain. The last 2 rules help the packets to pass through the FORWARD chain. Note that we only allow ICMP packets from the Client VM that have the destination IP address of 8.8.8.8. All other PING packets are dropped as required by the project objectives.

Please run the rc.sh file (permissions might need to be changed for the file to be executed). Following images show the result of the iptables rules.

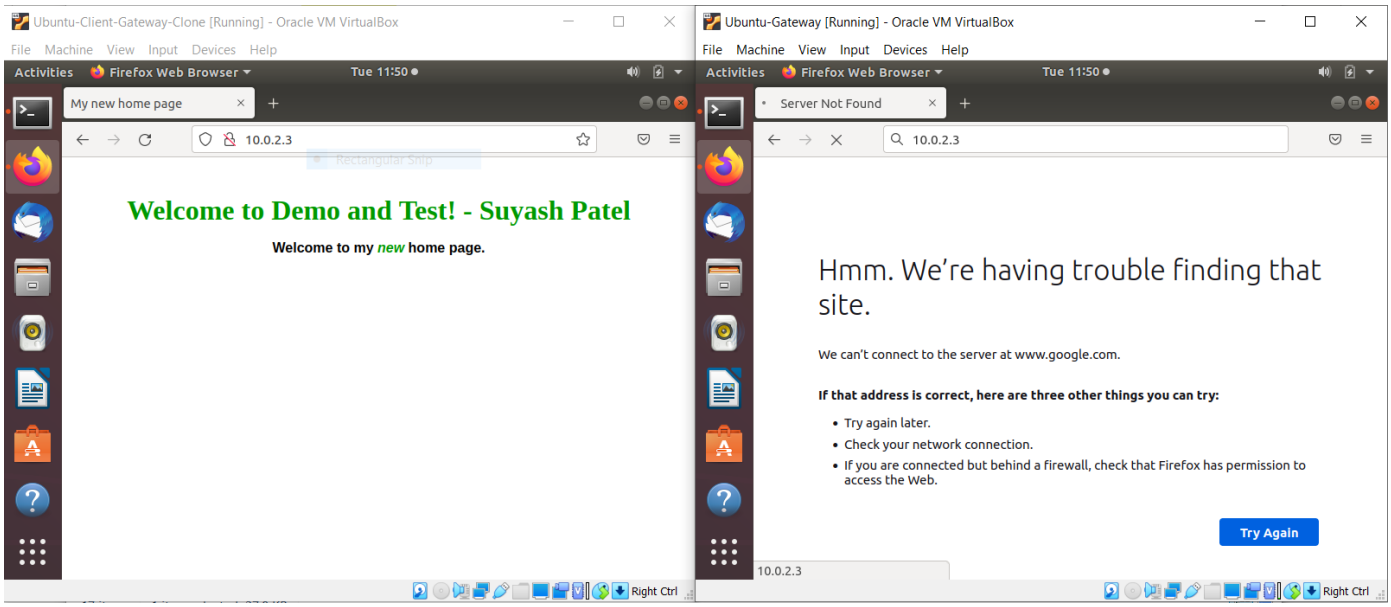


The client can ping 8.8.8.8 but cannot ping 10.0.2.3. The iptables rules are also shown on the gateway (rules that apply to the NAT table aren't shown. To list those rules run "**sudo iptables -t nat -S**").

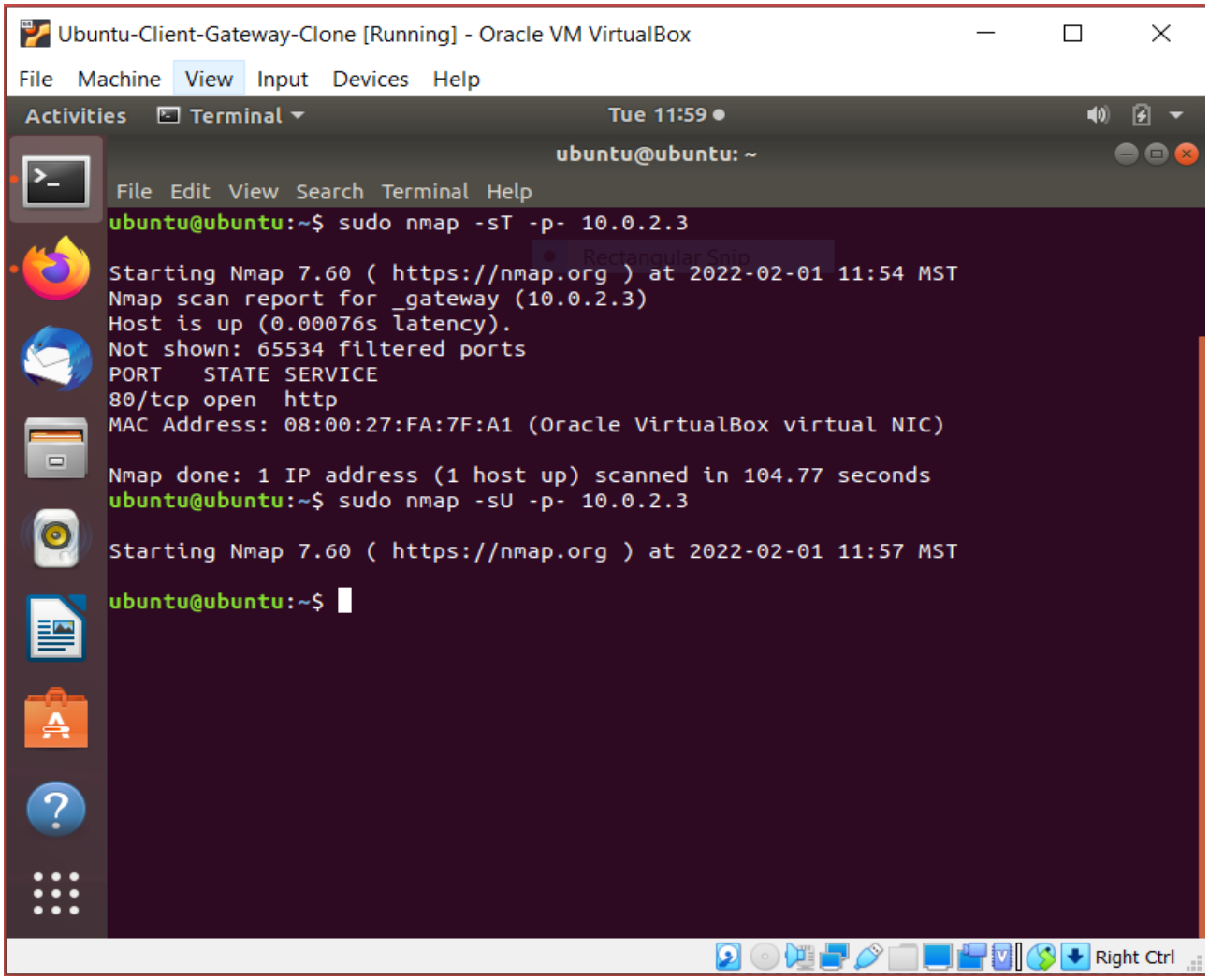
From the Client VM, no other public address can be pinged apart from 8.8.8.8.



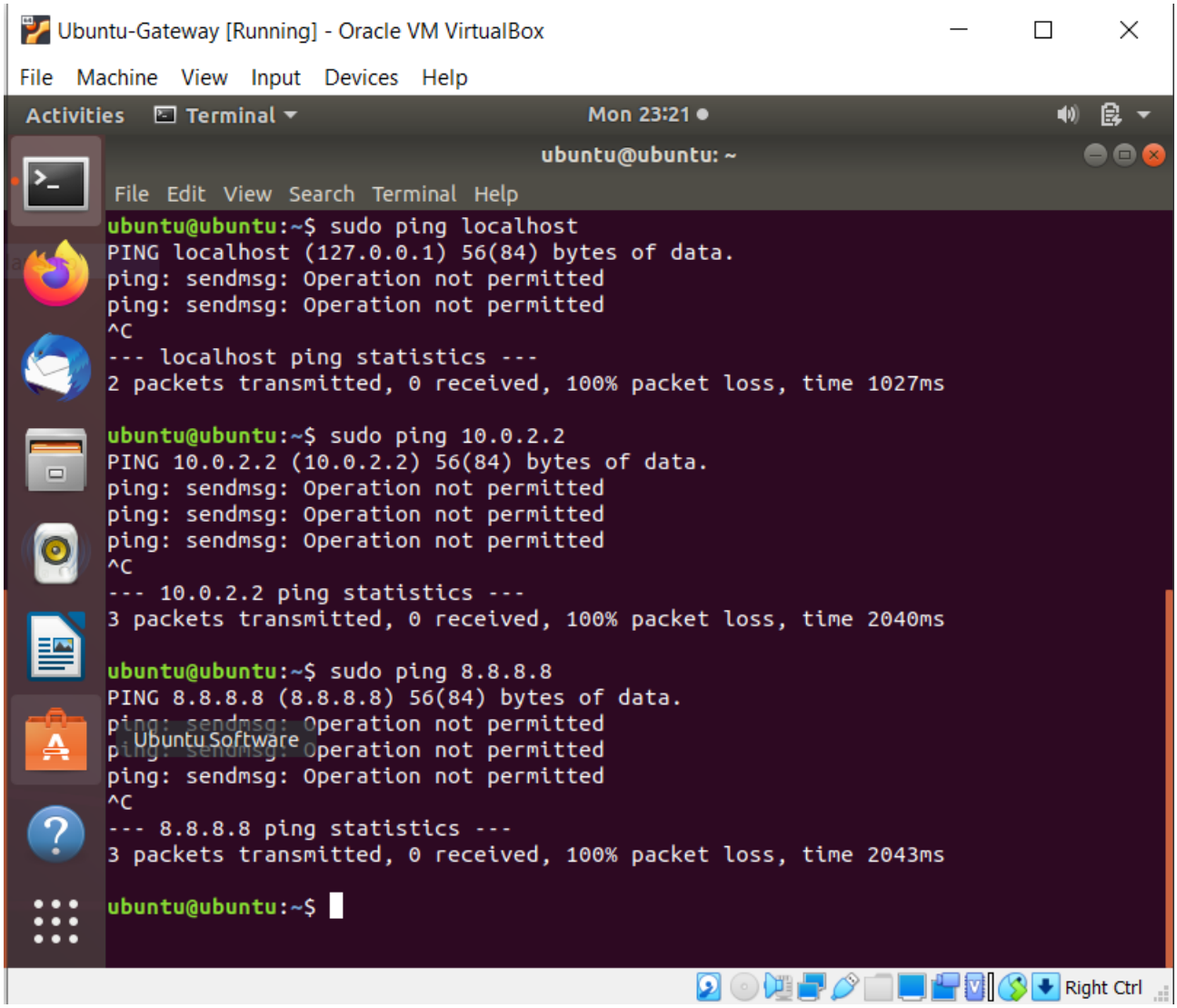
In the following image one can see that the Client VM can access the web page served by the web server on the Gateway VM but the Gateway VM cannot.



Following are the results of some nmap commands that had to be run as part of the project on the Client VM.



The first command return some result and the second command doesn't. This is because the port 80 on the Gateway is open for TCP connections but not for UDP connections. The T in the first command stands for TCP and the U in the second command stands for UDP. Following are the ping results from the Gateway:



V. Conclusion

Following are the tips I would give to anyone trying out this project:

1. It is imperative to disable DHCP on the internal NAT Network (10.0.2.0/24) because if DHCP is enabled, a default route is added on the Client VM automatically that routes all the packets through 10.0.2.1 which is an internal Gateway that NAT network provides. Debugging this can be difficult and therefore, I recommend that one should assign static IP addresses to the interfaces on the internal network and stay away from the 10.0.2.1 IP address. Choose any IP but that.
2. Once one starts writing iptables rules, it is a good idea to write them in a bash file instead of executing the commands directly on the terminal so that if those commands need to be run again, one can just run the shell file instead of running several commands.
3. Also, it is a good idea to add the iptables flush command at the top of the shell file as I have done. This ensures that every time one runs the shell file, one starts from a clean slate. Debugging can become difficult if pre-existing rules start to interfere.
4. Moreover, one can also write another shell file that restores the previous state of the setup. This saves time as one doesn't have to run multiple commands.

VI. Appendix B: Attached files

rc.sh	https://github.com/suyashpatel98/CSE-548-Packet-Filter/blob/master/rc.sh
config-client.yaml	https://github.com/suyashpatel98/CSE-548-Packet-Filter/blob/master/config-client.yaml

config-gateway.yaml	https://github.com/suyashpatel98/CSE-548-Packet-Filter/blob/master/config-gateway.yaml
restore.sh	https://github.com/suyashpatel98/CSE-548-Packet-Filter/blob/master/restore.sh