Student Name: Seon Hee Ju  
Email: seonheej@asu.edu  
Submission Date: 6th Febrary, 2022  
Class Name and Term: CSE548 Spring 2022

Project 1 Packet Filter Firewall

# Project Overview

In this project, I had hands-on experiences of building a firewall in a gateway. Using two Linux systems in virtual machines, I set up network environments with one client and a gateway/web server and made a shell script of firewall rules. In the firewall shell script, I set filtering policies for the INPUT, FORWARD and OUTPUT chains and configured NAT PREROUTING rules.

# Network Setup

In this lab, I explored a multi-hop network where a network gateway connects to two different networks that one is with the client and the other one is the internet. The diagram below shows the network map of this lab.

Figure 1 : Network map

The client’s IP address is 10.0.2.15 and it is connected to the gateway 10.0.2.7 with the 10.0.2.0 network. The gateway can connect to the internet through the 10.0.1.0 network. The gateway server also plays as a web server and its IP address is 127.0.0.1

The two figures below show the initial interface status.

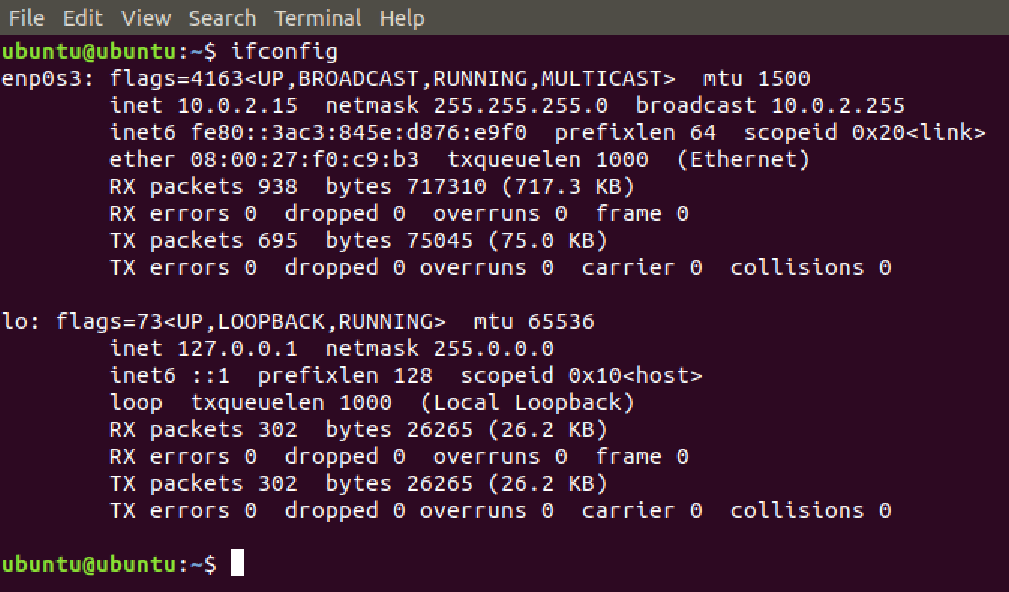


Figure 2 : Client’s initial ifconfig interface

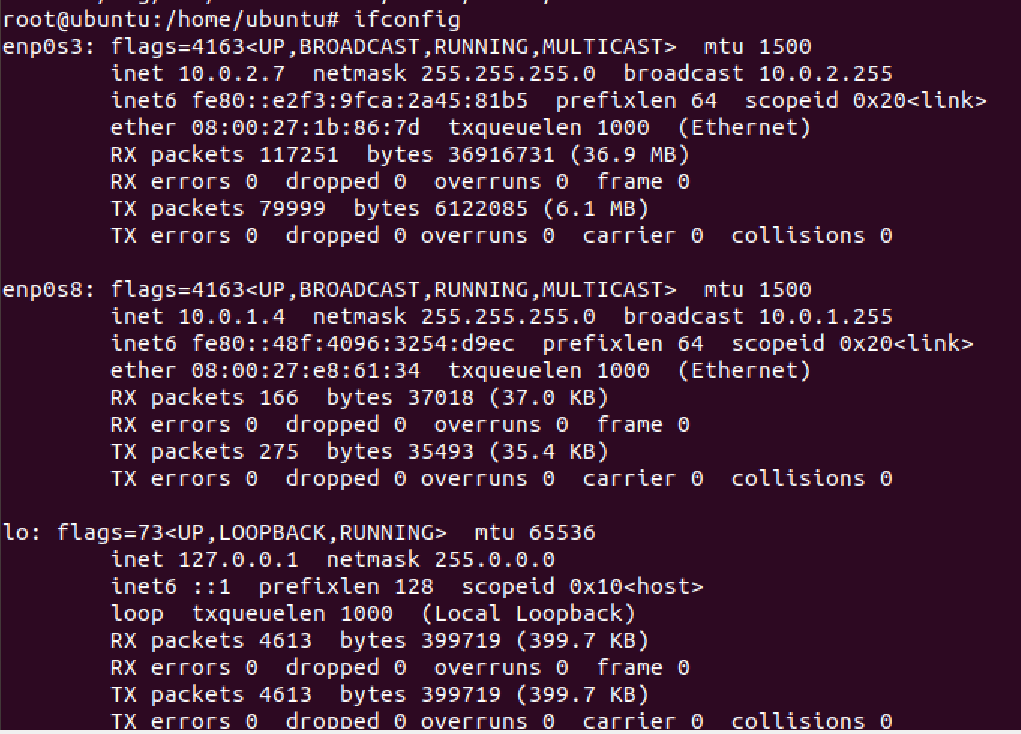


Figure 3 : server’s initial interface

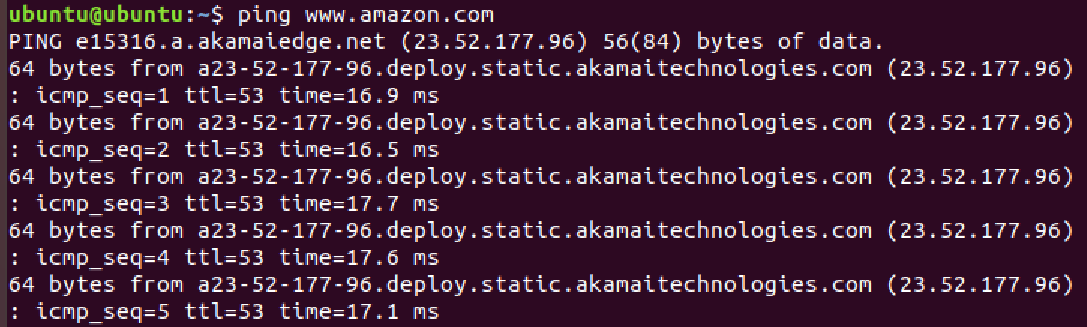
And I tested the client and the server’s initial reachability. Both the client and the server could connect to the site outside.

Figure 4 : Ping to amazon from the client

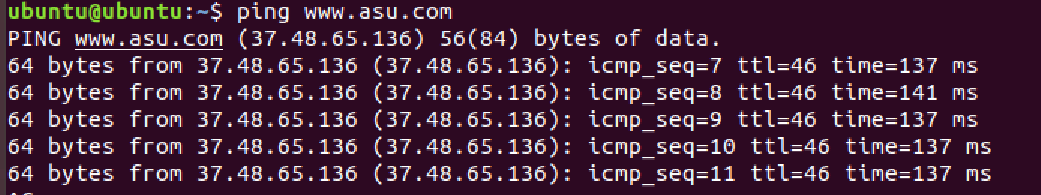


Figure 5 : Ping to asu from the server

With the command “route –n”, I checked the networking setup of the client and the server. Initially, both had a 10.0.2.1 gateway.

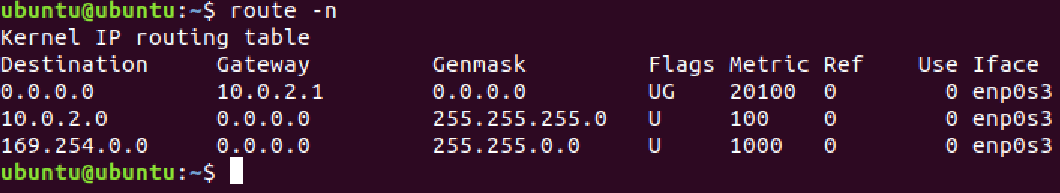


Figure 6 : Client’s initial routing table

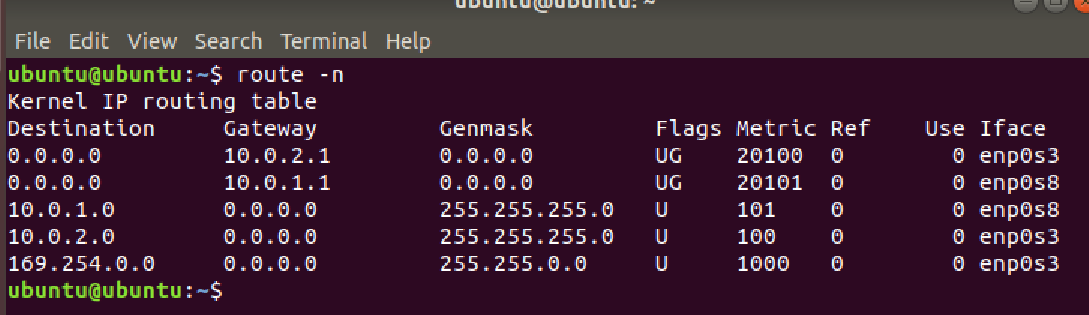
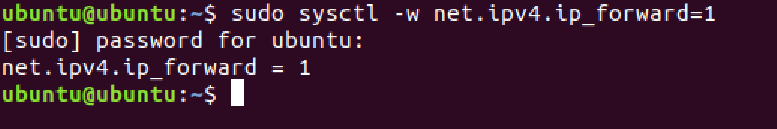
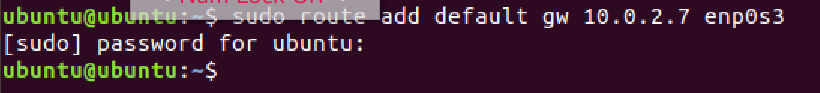


Figure 7 : Server’s initial routing table

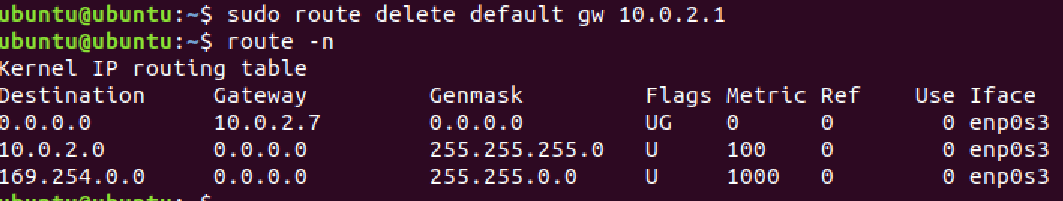
Then I tried to set up a value 1 in the ip forward file in the directory /proc/sys/net/ipv4/ to enable packet forwarding on the Gateway/Server VM. But an error occurred, so I used the command sysctl to enable the IP forwarding.



And I add the gateway 10.0.2.7 to the client’s default gateway setting.



To make the client only connected with the gateway 10.0.2.7, I delete the initial default gateway from the client’s gateway setting. Now, the client would communicate with other sites through the 10.0.2.7 gateway.



# Software

These are the software and network services were used.

- VirtualBox, Ubuntu

- Various network tools (ping, traceroute, tcpdump)

- iptables, NAT firewall

- Apache2 web server

# Project Description

A. Apache 2 Setup

To determine if Apach2 is installed on Ubuntu, I issued the command “apache2 –v”, and found out that it was installed. I decided to use all the configuration files as they are, but created a new index.html file. I deleted the default index.html in /var/www/html directory, and put the new index.html in it. I checked the file permission mode of the file and it was 777.

B. Firewall Setup

Before modifying the firewall rules, I changed the Internet\_IP, Client\_NET\_IP in the shell script to reflect my network environment. And IPTABLES="/sbin/iptables" caused an error, so I replaced the part with IPTABLES="iptables".

I set the rules to allow http traffic from the client network to the server network and ping traffic from the client to 8.8.8.8.

$IPTABLES -A FORWARD -p TCP -i $Client\_NET\_IFACE -o $Internet\_IFACE -j ACCEPT

$IPTABLES -A FORWARD -p icmp -o $Client\_NET\_IFACE -s 8.8.8.8 -j ACCEPT

$IPTABLES -A FORWARD -p icmp -s 10.0.2.15 -i $Client\_NET\_IFACE -d 8.8.8.8 -j ACCEPT

These are the INPUT chain rules for allowing http(80), https(443) traffic from the client and the web server.

And in the requirement, the gateway is supposed to be unable to ping to the localhost but it should see the index.html of the localhost. So I only allowed tcp and udp traffic to the lo interface.

$IPTABLES -A INPUT -p TCP --dport 80 -i $Client\_NET\_IFACE -s 10.0.2.15 -j ACCEPT

$IPTABLES -A INPUT -p TCP --dport 443 -i $Client\_NET\_IFACE -s 10.0.2.15 -j ACCEPT

$IPTABLES -A INPUT -p TCP --dport 80 -i $LO\_IFACE -d $WEB\_IP\_ADDRESS -j ACCEPT

$IPTABLES -A INPUT -p TCP --dport 443 -i $LO\_IFACE -d $WEB\_IP\_ADDRESS -j ACCEPT

$IPTABLES -A INPUT -p tcp -i lo -j ACCEPT

$IPTABLES -A INPUT -p udp -i lo -j ACCEPT

And I allow the client to get ping traffic only from the 8.8.8.8 google server.

$IPTABLES -A INPUT -p icmp -j DROP

$IPTABLES -A INPUT -p ICMP -i $Internet\_IFACE -s 8.8.8.8 -j ACCEPT

Similarly, I set the output chain rules.

$IPTABLES -A OUTPUT -p ICMP --icmp-type echo-reply -j ACCEPT

$IPTABLES -A OUTPUT -p ICMP --icmp-type echo-request -j ACCEPT

$IPTABLES -A OUTPUT -p TCP --sport 80 -o $Client\_NET\_IFACE -d 10.0.2.15 -j ACCEPT

$IPTABLES -A OUTPUT -p TCP --sport 443 -o $Client\_NET\_IFACE -d 10.0.2.15 -j ACCEPT

$IPTABLES -A OUTPUT -p TCP --dport 80 -o $Internet\_IFACE -j ACCEPT

$IPTABLES -A OUTPUT -p TCP --dport 443 -o $Internet\_IFACE -j ACCEPT

$IPTABLES -A OUTPUT -p tcp -o $LO\_IFACE -j ACCEPT

$IPTABLES -A OUTPUT -p udp -o $LO\_IFACE -j ACCEPT

$IPTABLES -A OUTPUT -p TCP --sport 80 -o $LO\_IFACE -j ACCEPT

I enabled POSTROUTING to allow the client to access the outside network (8.8.8.8) and change their source IP addresses.

# Add your own rule below to only allow ping from client to 8.8.8.8 on internet

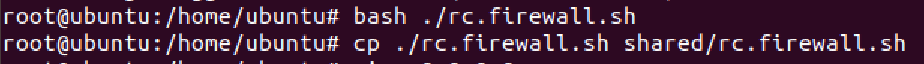
$IPTABLES -t nat -A POSTROUTING -p icmp -o $Internet\_IFACE -d 8.8.8.8 -j MASQUERADE

# Example: Allow client node to access to all Internet using masquerade

$IPTABLES -t nat -A POSTROUTING -o $Internet\_IFACE -d 8.8.8.8 -j MASQUERADE

C. Results

I executed the shell script I created with the command “bash ./rc.firewall.sh”.



The figure below shows the results of ‘nmap’ of before the rules were established / after the rules were established.



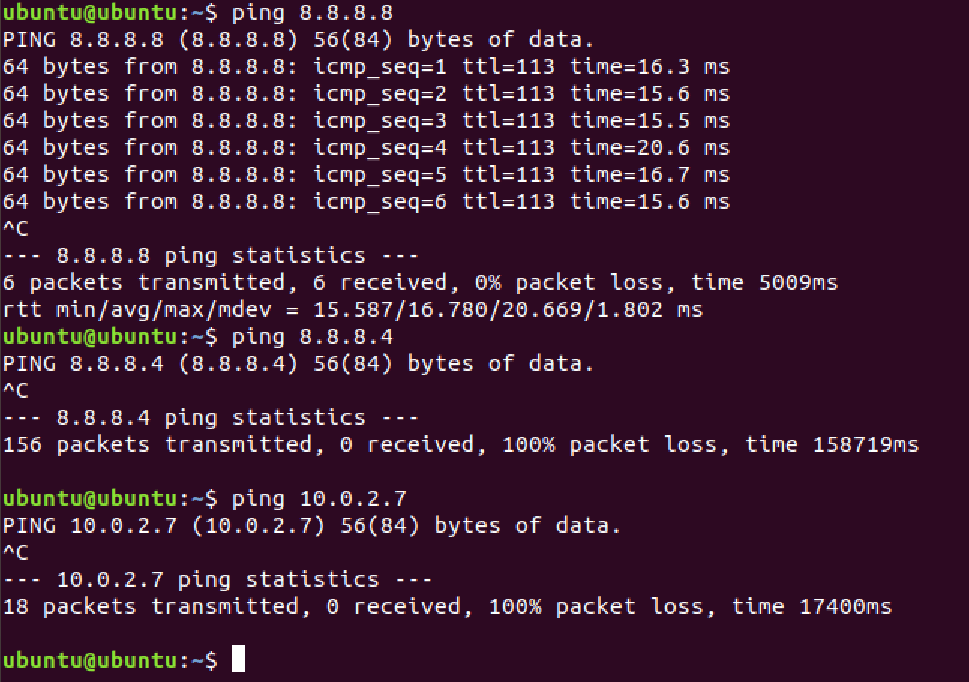
(Client image) After the rules established

(Client image) Before the rules established

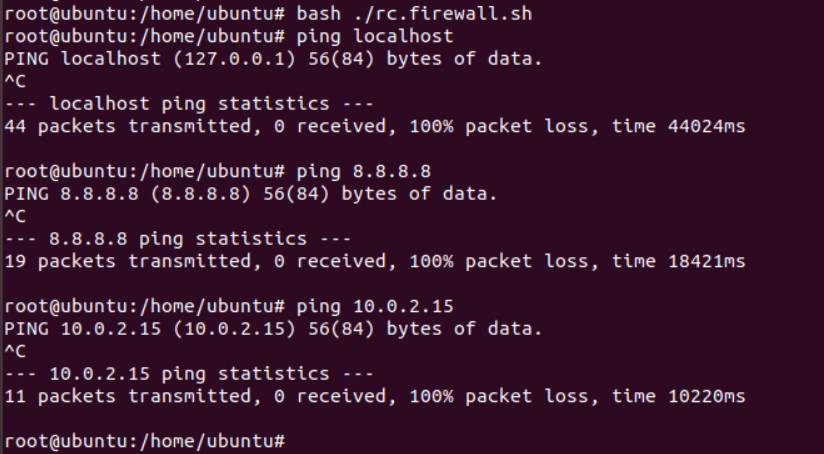
The “nmap” did not return any result after the firewall rules were established in the gateway.

And the client could ping 8.8.8.8 but could not ping the gateway or any other site.

The figure next shows that the ping traffic packets between the client and the 8.8.8.8 server were transferred to each other. But the ping traffic packets from the client to the 8.8.8.4 and the 10.0.2.7(the gateway) could not reach the destination. So the requirement of this lab was fulfilled.



And the server was supposed to be unable to ping any site. I tried to ping the localhost, 8.8.8.8 and 10.0.2.15(the client). All the attempts failed as expected. It means that the firewall policy of the ICMP protocol was successful.



Another requirement of this lab is setting up http(webpage) service to its own IP address. This is the code of my index.html file.

<html>

<head>Welocom to this site</title>

</head>

<body>

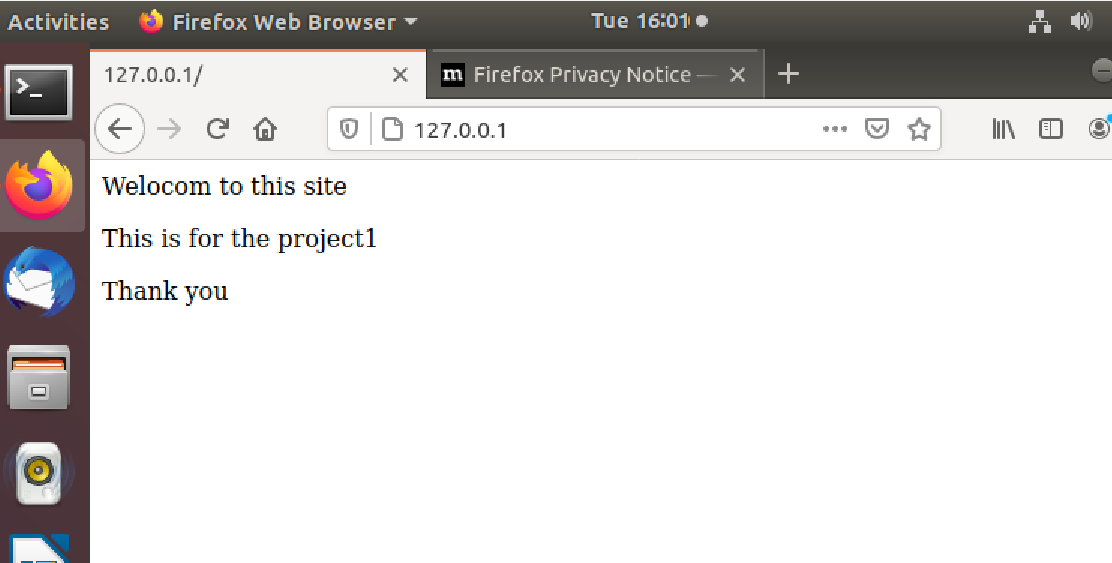
<p>This is for the project1</p>

<p>Thank you</p>

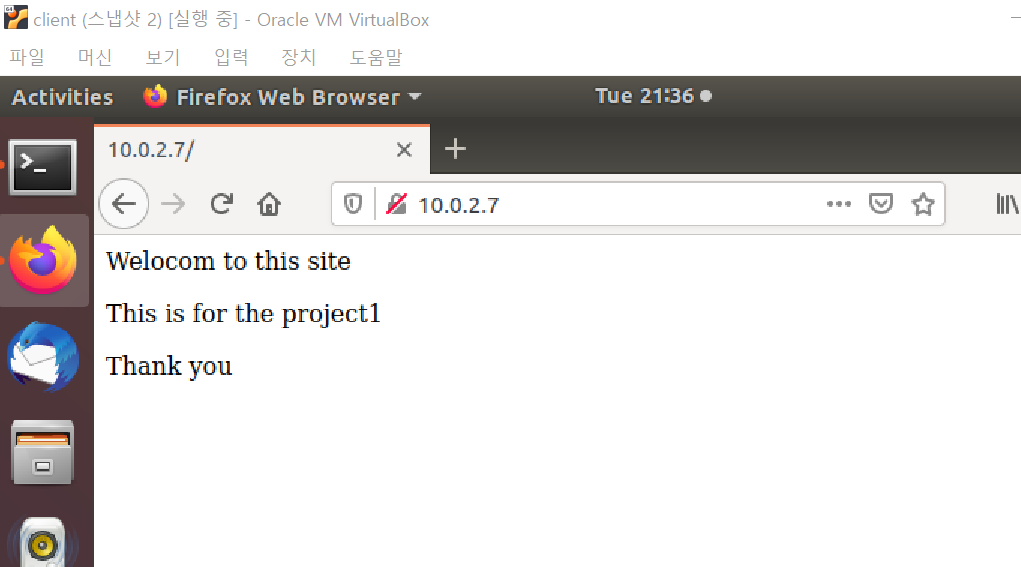
</body>

</html>

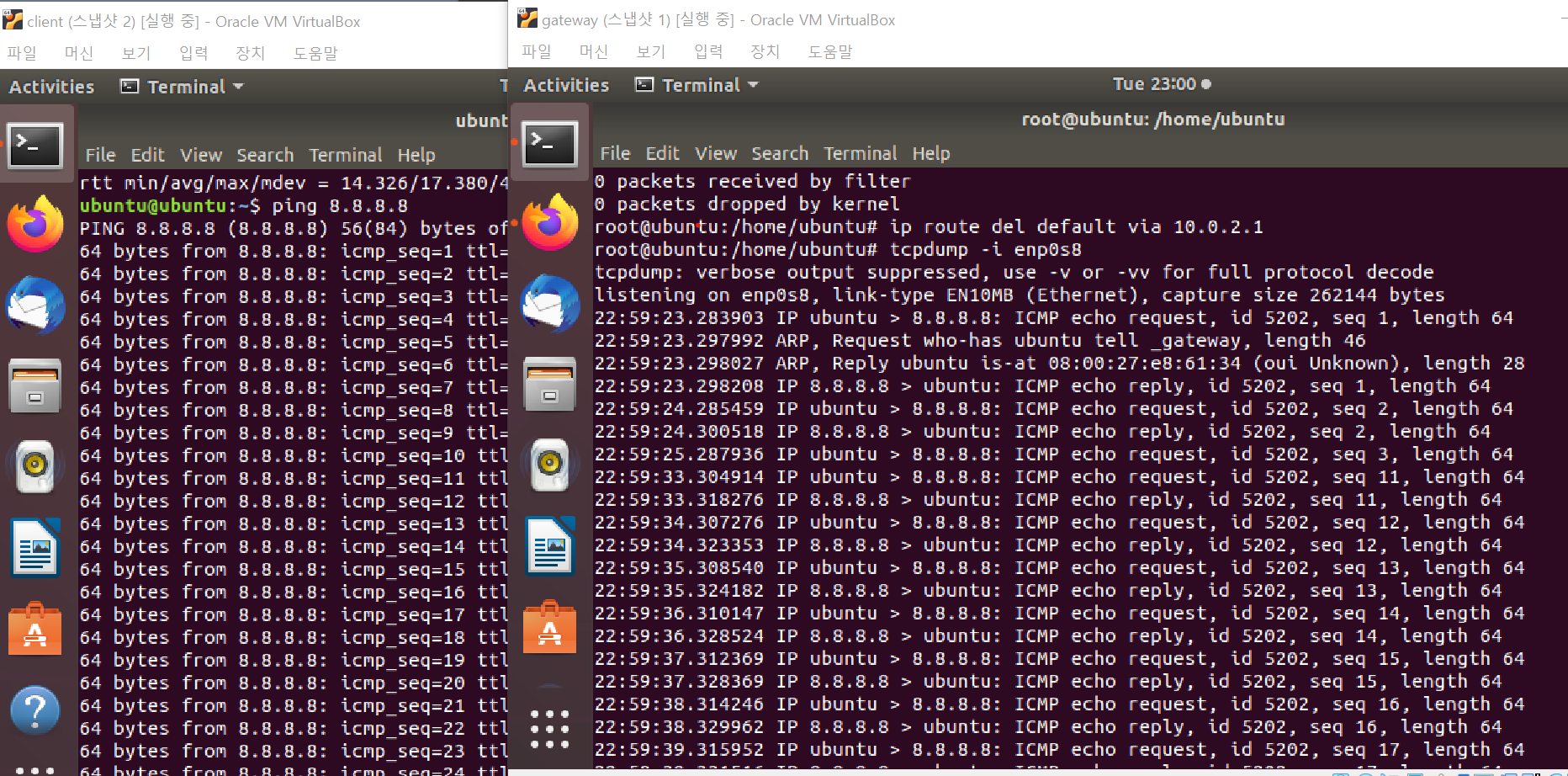
I could access the demo webpage from the gateway server. I tried with the gateway’s IP address 10.0.2.7, and it worked, too.



The client could also access the demo webpage on Gateway/Server VM by access the IP address of Gateway/Server VM in browser.



I enabled POSTROUTING to allow the client to access the outside network(8.8.8.8) and change their source IP addresses. The left figure shows that the client is pinging the 8.8.8.8 server and the right figure shows the results of “tcpdump –i enp0s8”. And from the image, you can see that the client’s IP address(10.0.2.15) was substituted by “ubuntu”, the gateway’s hostname.



# Conclusion

Describe lesson learned from this project, e.g., any interesting discovers, tips, and tricks. Provide a self-assessment about your project and provide comments to this project.

# Appendix B: Attached files

These are the github links of the configuration and developed source files I created in this lab. Except for these two below, I used the default configuration files that the system provided.

rc.firewall.sh : <https://github.com/SeonHeeJu/project1_adv/blob/main/rc.firewall.sh>

index.html : <https://github.com/SeonHeeJu/project1_adv/blob/main/index.html>

# References

Reference is optional, but nice to have to allow others to read your report with additional linked source for validation and learning.

1. Wireshark, available at <https://www.wireshark.org/>, accessed by 8/31/2018.
2. Postel, Jon. "RFC 791: Internet protocol." (1981).