PACKET FILTER FIREWALL (IPTABLES)

Student Name: Hemendu Roy

Email: hroy6@asu.edu

Submission Date: 02-04-2022

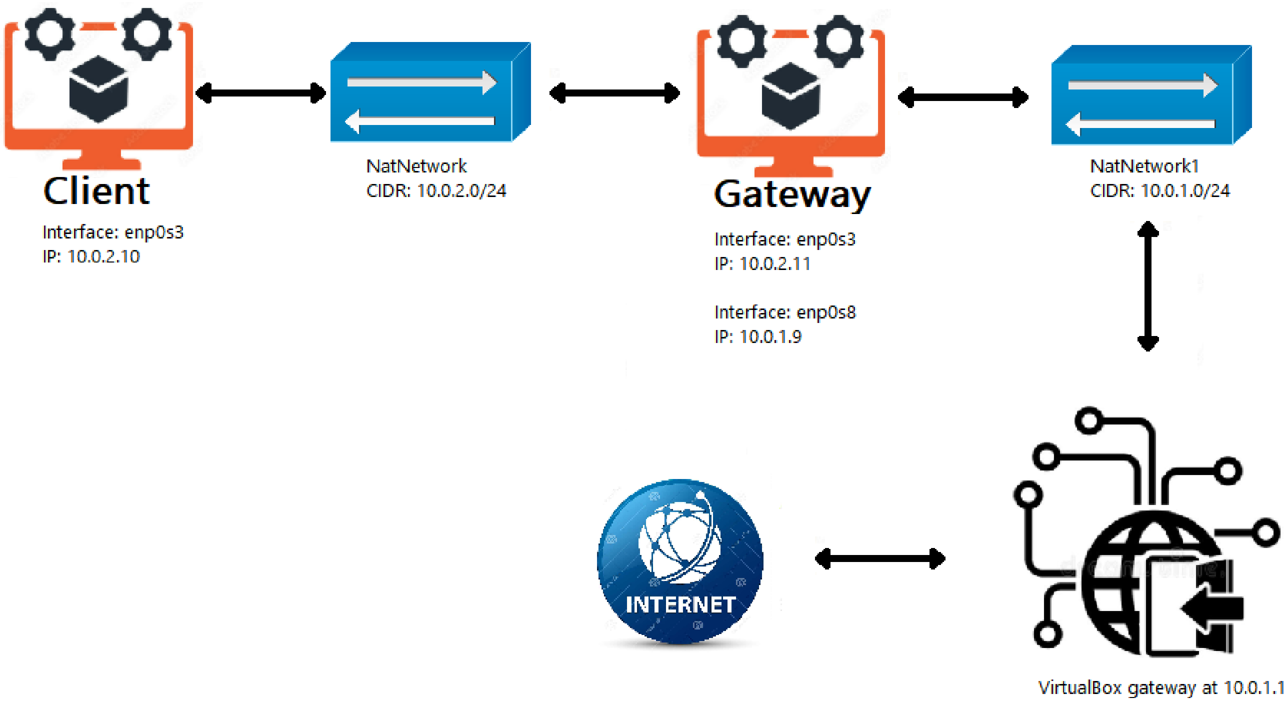
Class Name and Term: CSE548 Summer 2021

I. PROJECT OVERVIEW

In this project, we're exploring how iptables can be used to setup a packet filter firewall. The environment consists of two virtual machines running Ubuntu 20.04 LTS. One a client machine, and the other, acting as gateway machine. The two machines are connected by an internal network. Additionally, the gateway machine is connected to a second network with internet access. A firewall has been setup on the gateway machine to only allow certain traffic from the client and drop the rest, essentially emulating a real-world environment with security constraints.

II. NETWORK SETUP

Network Topology and Configurations



In this setup, we have 2 NAT Networks connecting the Client and Gateway VMs. The first one, is an internal network between the 2 VMs. It is akin to a private network in an organization.

Interface Details:
Name: enp0s3
Client IP: 10.0.2.10

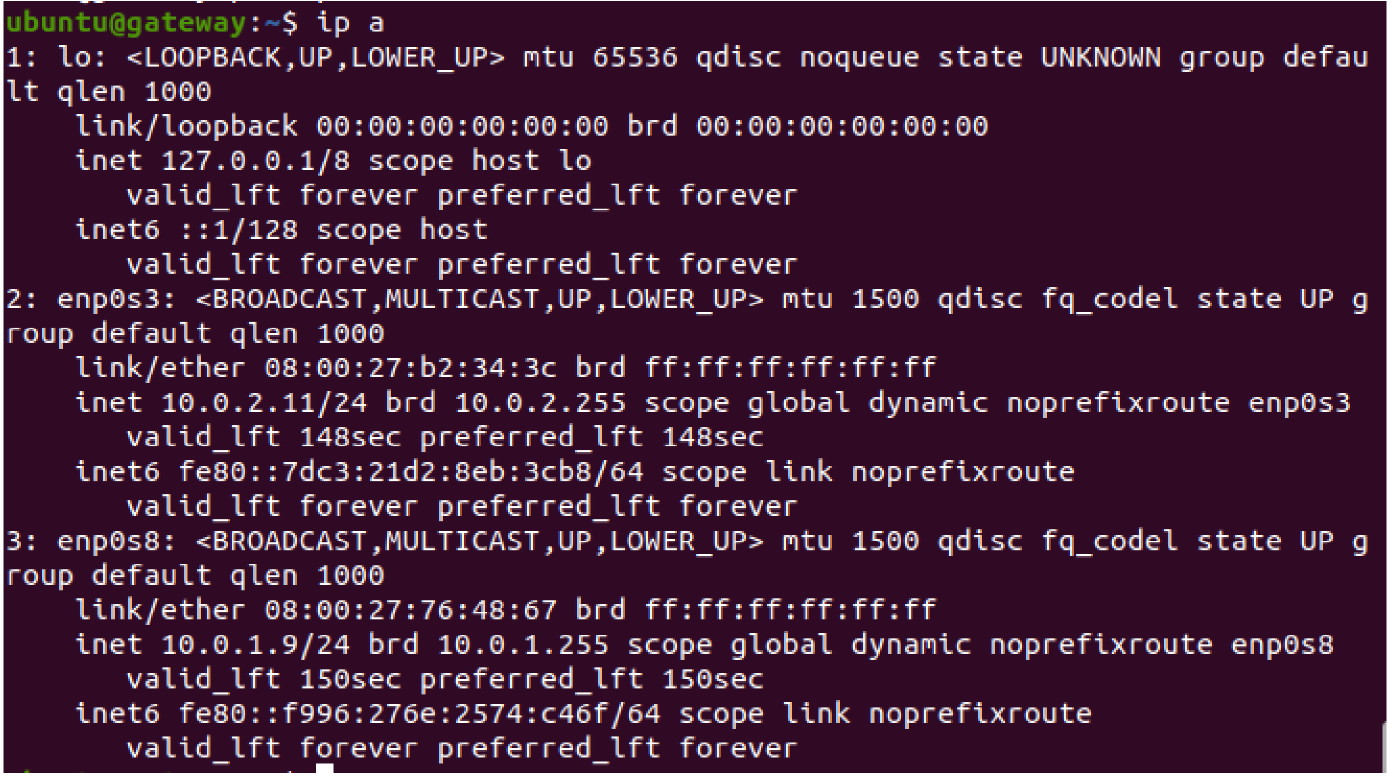
Gateway IP: 10.0.2.11
The second network, is connected to the internet and only to the Gateway VM. Our goal is to allow the client to access the internet exclusively through the Gateway VM using this interface.

Interface Details:
Name: enp0s8
Gateway IP: 10.0.1.1 (VirtualBox assigned)

Initial Reachability

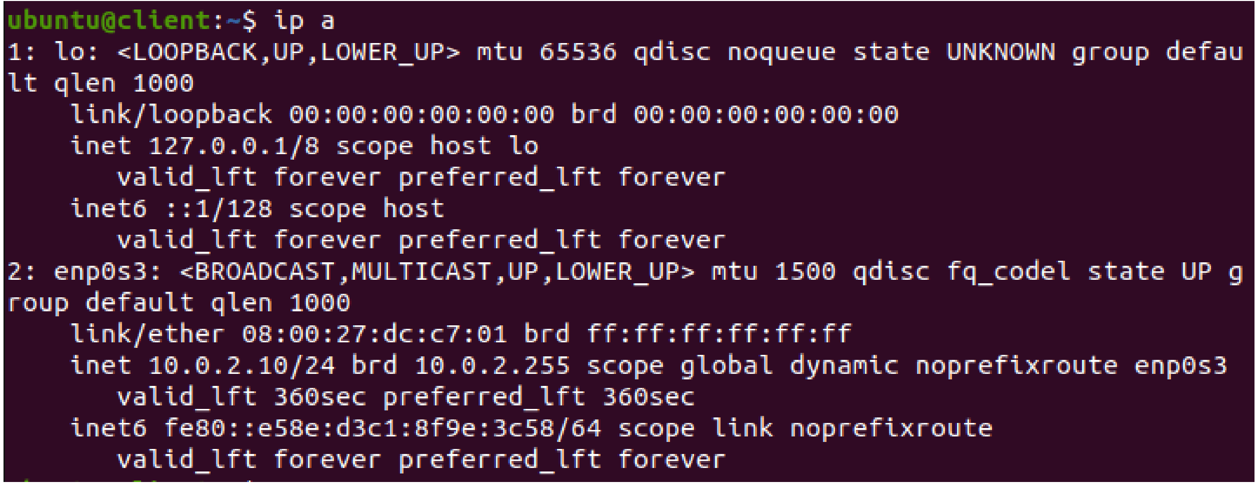
	Gateway public network	Gateway private network	Client private network
IP Address	10.0.1.9	10.0.2.11	10.0.2.10
Via	NatNetwork1	NatNetwork	NatNetwork
Destination	Internet	Client VM	Gateway VM

Gateway config



```
ubuntu@gateway:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:b2:34:3c brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.11/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 148sec preferred_lft 148sec
    inet6 fe80::7dc3:21d2:8eb:3cb8/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:76:48:67 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.9/24 brd 10.0.1.255 scope global dynamic noprefixroute enp0s8
       valid_lft 150sec preferred_lft 150sec
    inet6 fe80::f996:276e:2574:c46f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Client config



```
ubuntu@client:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:dc:c7:01 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.10/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 360sec preferred_lft 360sec
    inet6 fe80::e58e:d3c1:8f9e:3c58/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

III. Software

iptables – [1] Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

tcpdump – [2] tcpdump is used to capture network traffic on a network interface that match a set of Boolean expressions

wireshark – [3] Wireshark is the world's foremost and widely-used network protocol analyzer. It has been used to analyze network traffic from the client to the gateway to verify firewall functionalities.

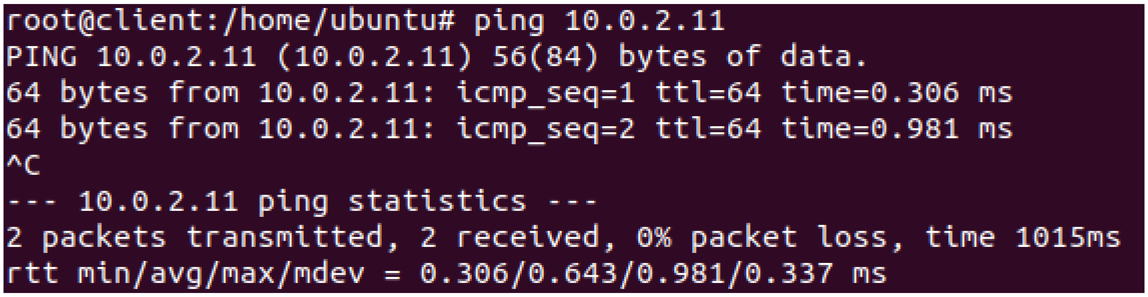
Apache web server – [4] The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. It has been used to set up a demo web page on the gateway VM to test client reachability.

IV. Project Description

In this section, detailed descriptions of the project tasks will be illustrated.

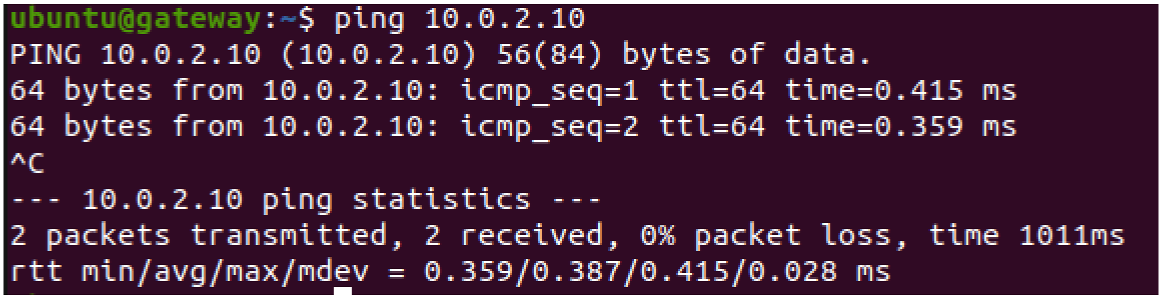
Before setting up firewall rules, we check the connectivity of the two VMs using the *ping* command.

On the client machine, execute *ping <gateway ip>*



```
root@client:/home/ubuntu# ping 10.0.2.11
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data.
64 bytes from 10.0.2.11: icmp_seq=1 ttl=64 time=0.306 ms
64 bytes from 10.0.2.11: icmp_seq=2 ttl=64 time=0.981 ms
^C
--- 10.0.2.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1015ms
rtt min/avg/max/mdev = 0.306/0.643/0.981/0.337 ms
```

Similarly on the gateway machine, execute *ping <client ip>*

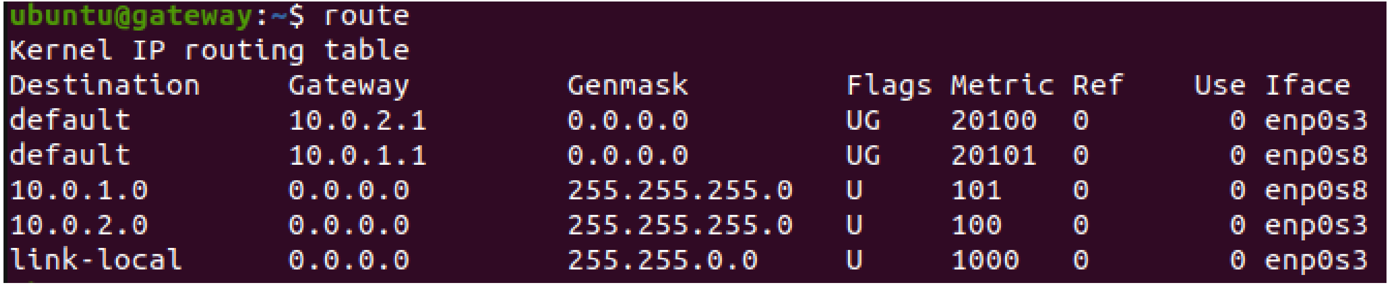


```
ubuntu@gateway:~$ ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=0.415 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=64 time=0.359 ms
^C
--- 10.0.2.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.359/0.387/0.415/0.028 ms
```

Now, we set the default gateway on the client machine to the IP address of the gateway machine on the internal network.
On the client machine, run *sudo route add default gw <gateway ip> <gateway network interface name>*
Now, the client is configured to forward all traffic to the gateway VM.

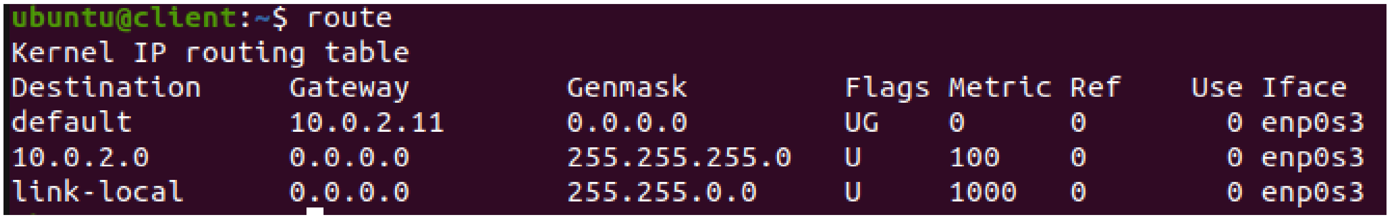
We can run the *route* command on both machines to check these configurations.

Gateway



```
ubuntu@gateway:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.2.1        0.0.0.0         UG    20100  0        0 enp0s3
default         10.0.1.1        0.0.0.0         UG    20101  0        0 enp0s8
10.0.1.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
```

Client

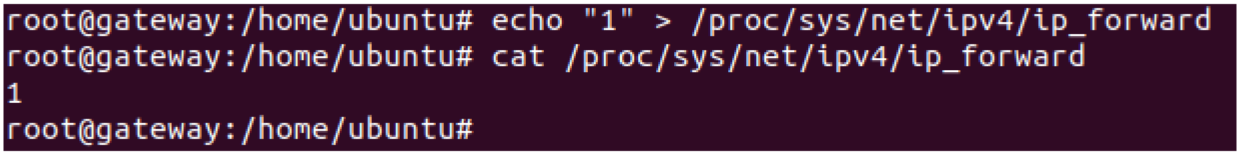


```
ubuntu@client:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.2.11       0.0.0.0         UG    0      0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
```

Now that our client and gateway machines have their connectivity verified, we can proceed with setting up the firewall rules.

First, we enable packet forwarding on the gateway machine by running the following commands.

To enable - *sudo echo "1" > /proc/sys/net/ipv4/ip_forward*
To verify the change – *cat /proc/sys/net/ipv4/ip_forward*



```
root@gateway:/home/ubuntu# echo "1" > /proc/sys/net/ipv4/ip_forward
root@gateway:/home/ubuntu# cat /proc/sys/net/ipv4/ip_forward
1
root@gateway:/home/ubuntu#
```

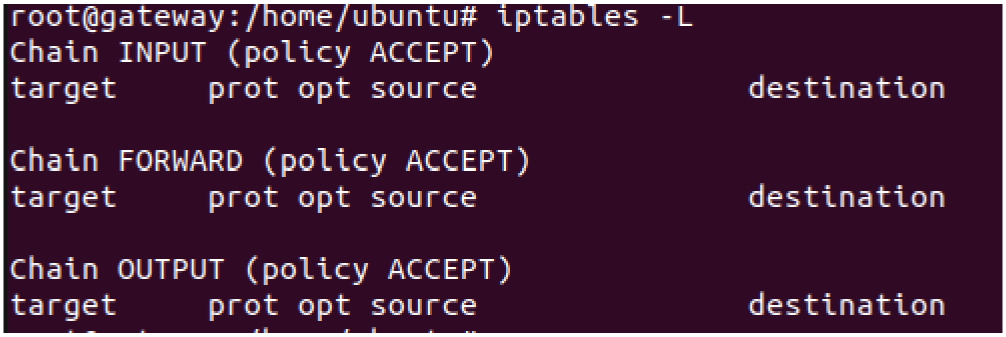
Next, we flush and delete all existing iptables chains from the machine. This step is only required if these rules have been previously set or modified.

iptables -X
iptables -F

At this stage, if we set the firewall policy to blacklist as shown below, we can verify packet flow between the two VMs without any restrictions

Commands:
sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -P FORWARD ACCEPT

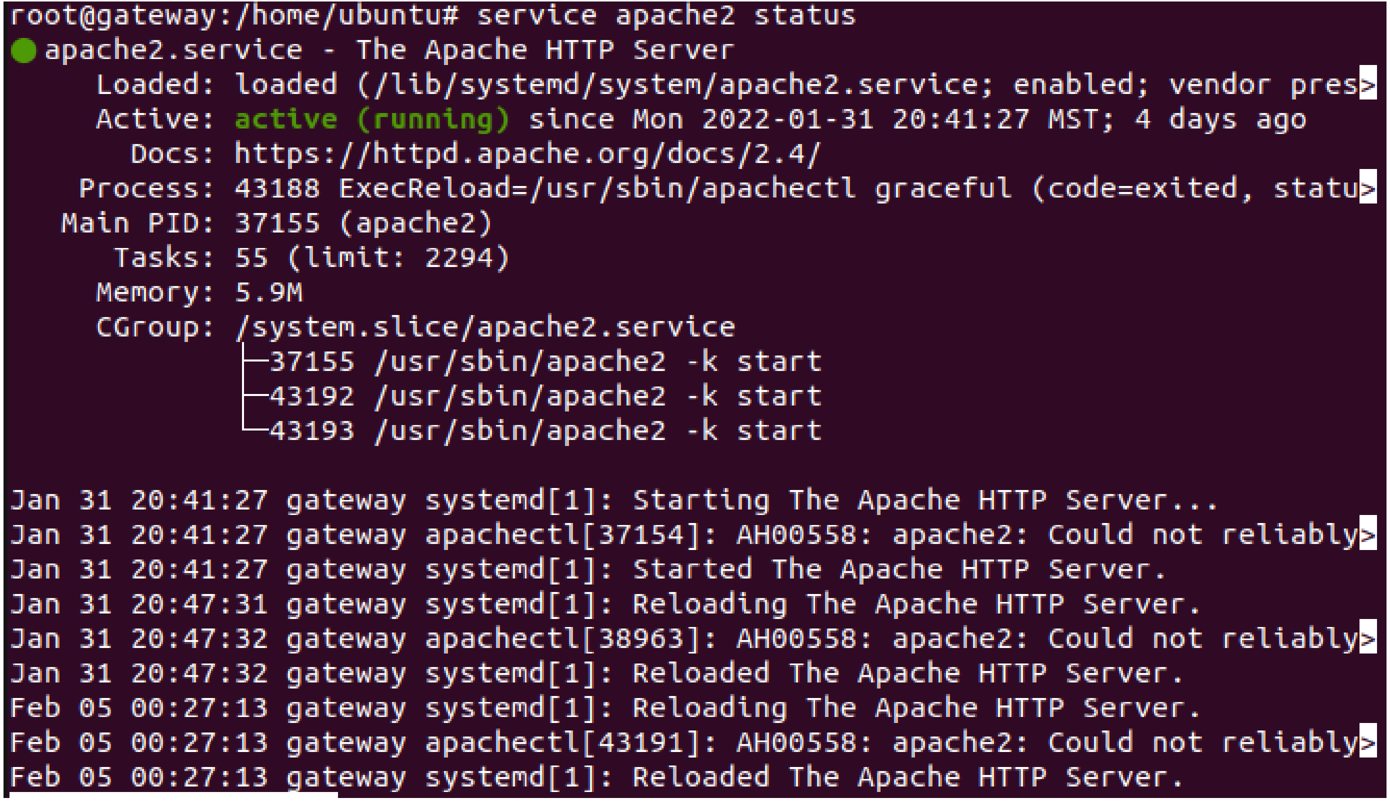
Now, the policies look as shown below:



```
root@gateway:/home/ubuntu# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

At this stage, we have completed all tests and are ready to begin testing our packet filter firewall.
To simulate a real-world scenario, we will set up an apache web server on the gateway machine. We will then attempt to whitelist only this website while blocking other, unwanted traffic from the client.

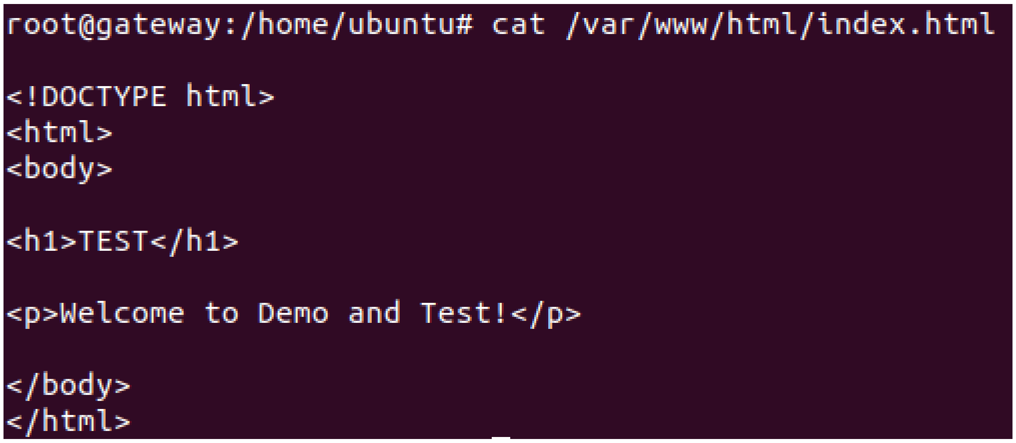
To setup the webpage, we first verify the status of the apache web service



```
root@gateway:/home/ubuntu# service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor pres>
     Active: active (running) since Mon 2022-01-31 20:41:27 MST; 4 days ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 43188 ExecReload=/usr/sbin/apachectl graceful (code=exited, statu>
   Main PID: 37155 (apache2)
      Tasks: 55 (limit: 2294)
     Memory: 5.9M
     CGroup: /system.slice/apache2.service
             ├─37155 /usr/sbin/apache2 -k start
             ├─43192 /usr/sbin/apache2 -k start
             └─43193 /usr/sbin/apache2 -k start

Jan 31 20:41:27 gateway systemd[1]: Starting The Apache HTTP Server...
Jan 31 20:41:27 gateway apachectl[37154]: AH00558: apache2: Could not reliably>
Jan 31 20:41:27 gateway systemd[1]: Started The Apache HTTP Server.
Jan 31 20:47:31 gateway systemd[1]: Reloading The Apache HTTP Server.
Jan 31 20:47:32 gateway apachectl[38963]: AH00558: apache2: Could not reliably>
Jan 31 20:47:32 gateway systemd[1]: Reloaded The Apache HTTP Server.
Feb 05 00:27:13 gateway systemd[1]: Reloading The Apache HTTP Server.
Feb 05 00:27:13 gateway apachectl[43191]: AH00558: apache2: Could not reliably>
Feb 05 00:27:13 gateway systemd[1]: Reloaded The Apache HTTP Server.
```

If the service is not installed, we must install it using apt.

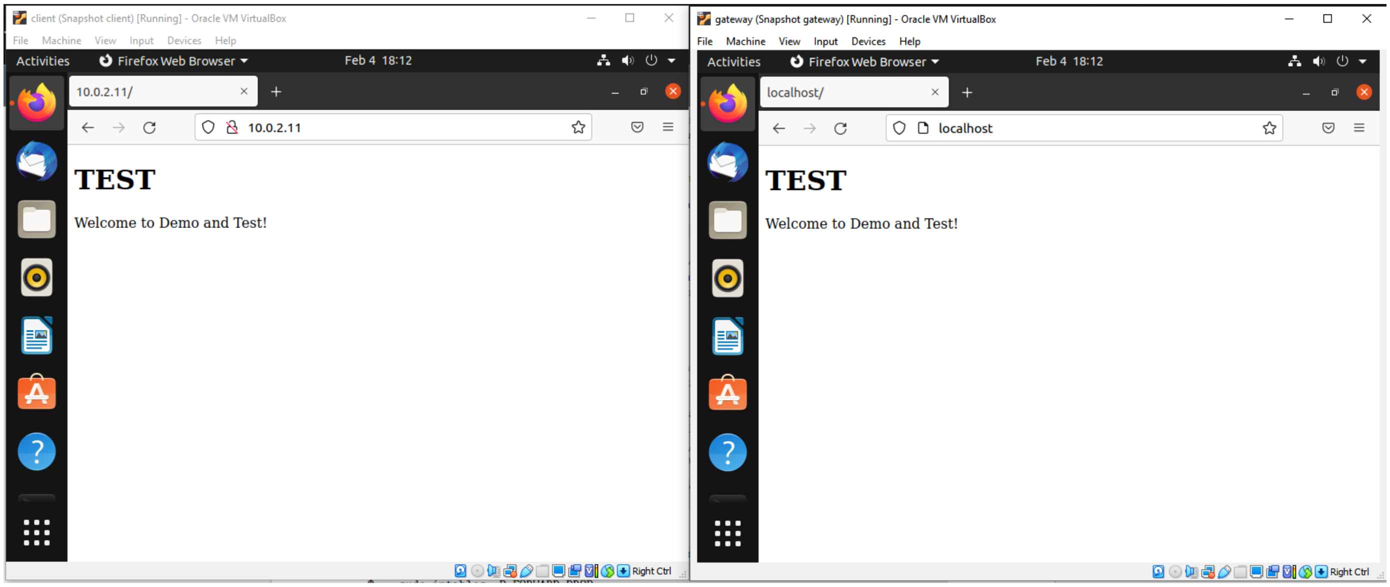
We edit the file in */var/www/html/index.html* to setup our demo page. The new file contents are



```
root@gateway:/home/ubuntu# cat /var/www/html/index.html

<!DOCTYPE html>
<html>
<body>

<h1>TEST</h1>

<p>Welcome to Demo and Test!</p>

</body>
</html>
```

Once, the webserver is up and running, we can verify it by accessing localhost from the browser on the gateway machine.
On the client machine, we can access it by browsing the IP of the gateway machine.

For example,


client (Snapshot client) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Feb 4 18:12
10.0.2.11/
10.0.2.11
Right Ctrl
gateway (Snapshot gateway) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Feb 4 18:12
localhost/
localhost
Right Ctrl

TEST

Welcome to Demo and Test!

TEST

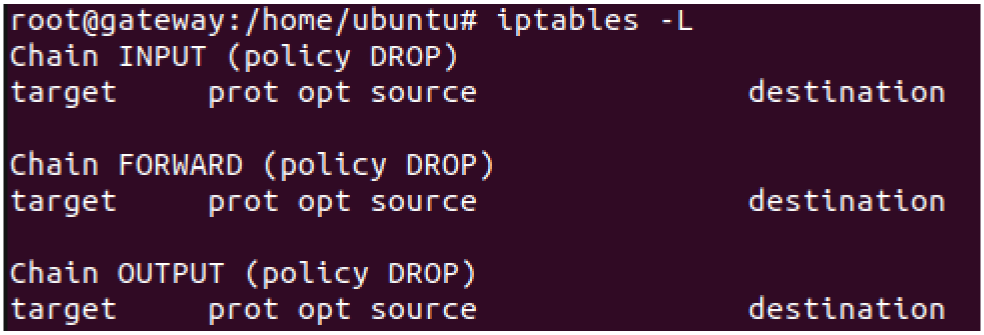
Welcome to Demo and Test!

Now, we proceed with developing our firewall script.

Task 2.1

Setting the whitelist policy-
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT DROP
sudo iptables -P FORWARD DROP

The -P option indicates that it is the default policy for the specified chain. Our iptables policies now look as shown below



```
root@gateway:/home/ubuntu# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
```

Task 2.2

Allowing the client to access the http webpage running on the gateway machine

There are a few ways to achieve this.

In the first method, we only open access to the webpage running on 127.0.0.1 by supplying the destination and source ip addresses.

Commands
iptables -A INPUT -p tcp -s 127.0.0.1 -j ACCEPT
iptables -A OUTPUT -p tcp -d 127.0.0.1 -j ACCEPT

The second method is to allow all tcp traffic from the client and simply open the tcp port or port 80 entirely

iptables -A INPUT -p tcp –dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp –dport 80 -j ACCEPT

The third method is to simply open all tcp traffic to the gateway

iptables -A INPUT -p tcp -j ACCEPT
iptables -A OUTPUT -p tcp -j ACCEPT

This way, all ports except 80 will be closed.

The right method depends on our requirements and security constraints. In all cases however, the client will be able to access the apache webpage as shown above.

Task 2.3 and 2.4

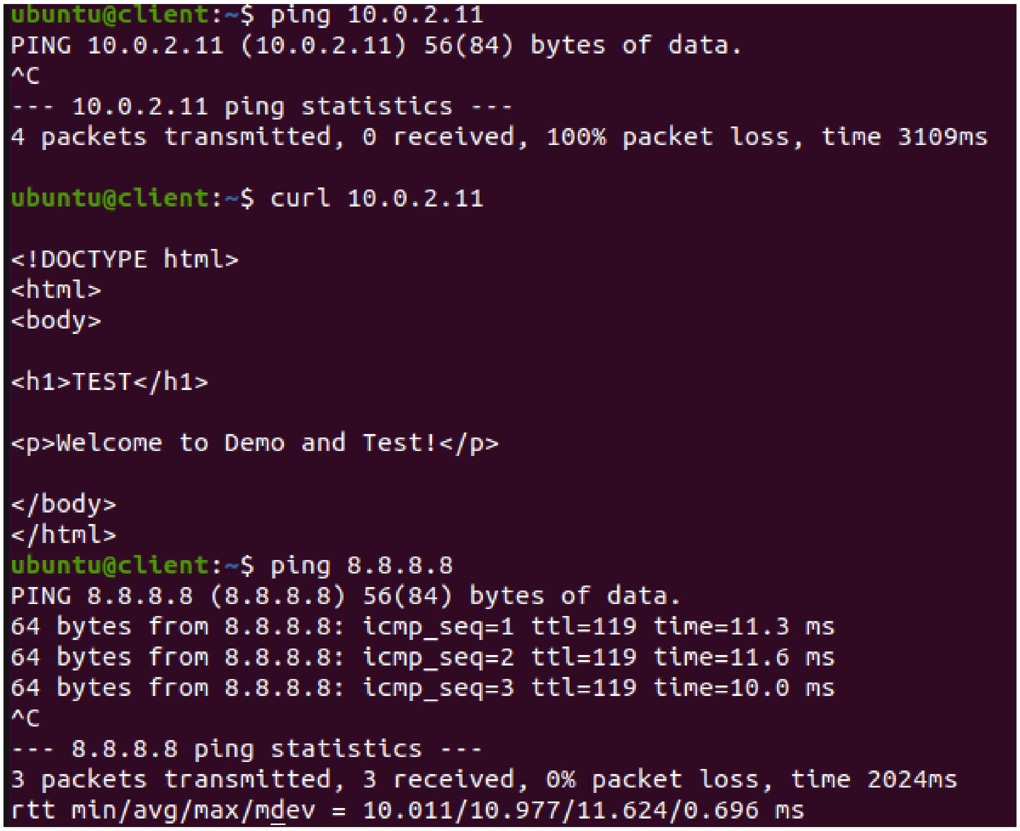
To stop the client from pinging the gateway IP or any other IP except 8.8.8.8, we issue the following commands

iptables -A INPUT -s 10.0.2.10 -d 8.8.8.8 -p icmp --icmp-type 8 -j ACCEPT
iptables -A FORWARD -s 10.0.2.10 -d 8.8.8.8 -p icmp -j ACCEPT

Since the default policy for all chains is DROP, any client traffic that does not meet the above constraints will be blocked by the gateway.

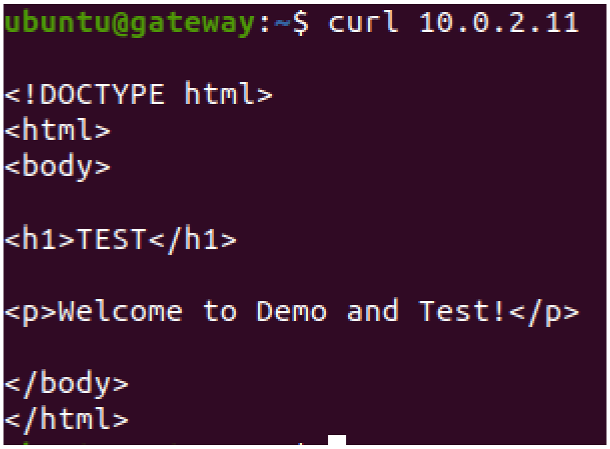
Lab Assessments

1. The client
   - Cannot ping the gateway IP
   - Can access the demo webpage
   - Can ping 8.8.8.8



```
ubuntu@client:~$ ping 10.0.2.11
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data.
^C
--- 10.0.2.11 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3109ms

ubuntu@client:~$ curl 10.0.2.11

<!DOCTYPE html>
<html>
<body>

<h1>TEST</h1>

<p>Welcome to Demo and Test!</p>

</body>
</html>
ubuntu@client:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=119 time=11.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=119 time=11.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=119 time=10.0 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2024ms
rtt min/avg/max/mdev = 10.011/10.977/11.624/0.696 ms
```

2. The gateway/server vm should
   - Set up http service to its own IP
   - Enable POSTROUTING to allow the client to access 8.8.8.8

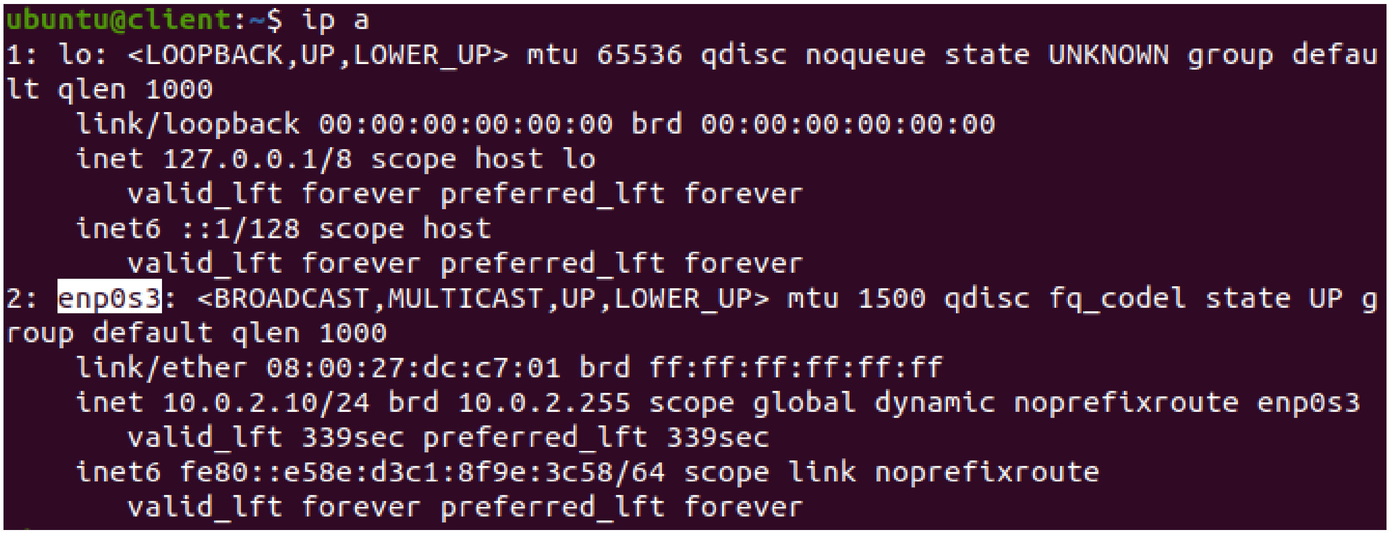


```
ubuntu@gateway:~$ curl 10.0.2.11

<!DOCTYPE html>
<html>
<body>

<h1>TEST</h1>

<p>Welcome to Demo and Test!</p>

</body>
</html>
```

To enable change the source IP addresses, we enable POSTROUTING by issuing the following command

iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE

Here, ***enp0s8*** is the interface that corresponds to the internet network connected to the gateway.



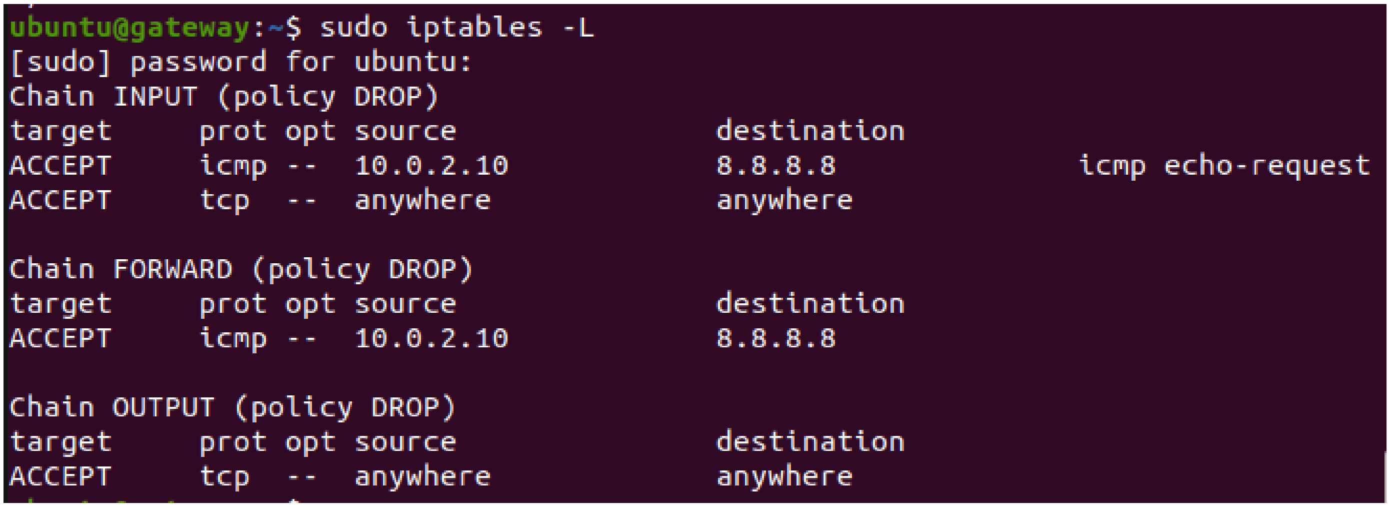
```
ubuntu@client:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:dc:c7:01 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.10/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 339sec preferred_lft 339sec
    inet6 fe80::e58e:d3c1:8f9e:3c58/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Also, since internet access must only be from the public network, we must instruct iptables to forward packets from the interface of the private network to that of the public network by issuing the following commands

iptables -A FORWARD -i enp0s3 -o enp0s8 -j ACCEPT
iptables -A FORWARD -i enp0s8 -o enp0s3 -m state --state ESTABLISHED,RELATED -j ACCEPT

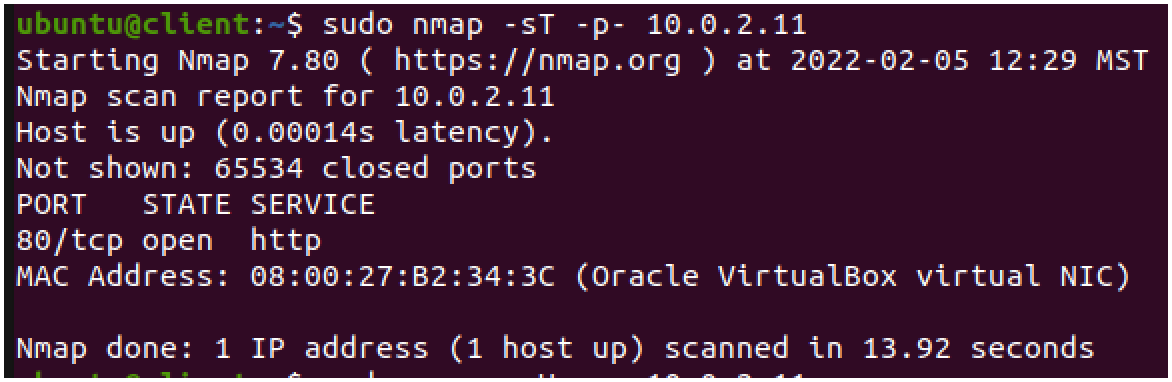
3. Additional Requirements

- The default policy for INPUT, OUTPUT and FORWARD chains must be DROP

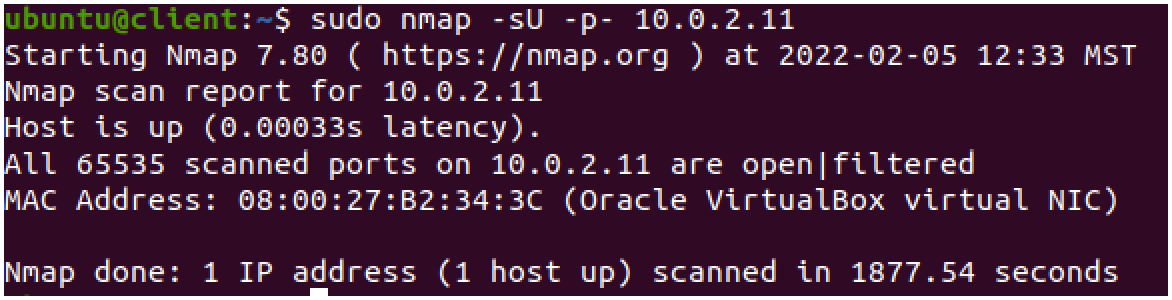


```
ubuntu@gateway:~$ sudo iptables -L
[sudo] password for ubuntu:
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  10.0.2.10            8.8.8.8              icmp echo-request
ACCEPT     tcp  --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  10.0.2.10            8.8.8.8

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere
```

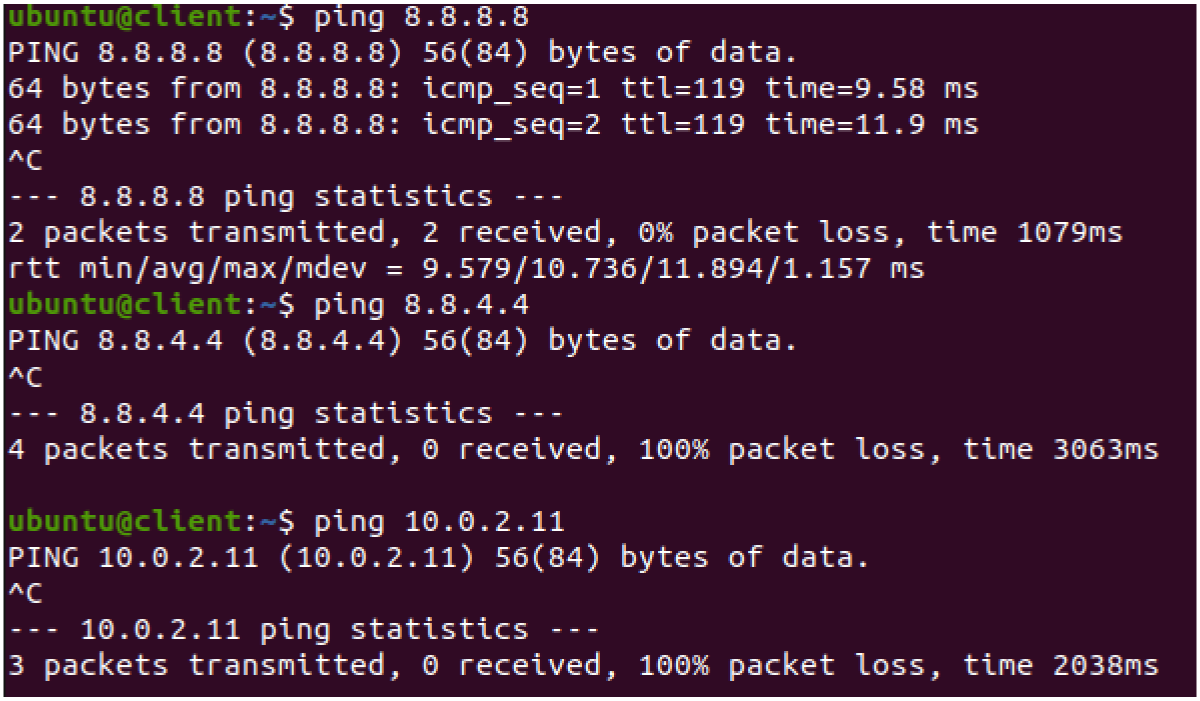
Client VM screenshots:



```
ubuntu@client:~$ sudo nmap -sT -p- 10.0.2.11
Starting Nmap 7.80 ( https://nmap.org ) at 2022-02-05 12:29 MST
Nmap scan report for 10.0.2.11
Host is up (0.00014s latency).
Not shown: 65534 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:B2:34:3C (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.92 seconds
```

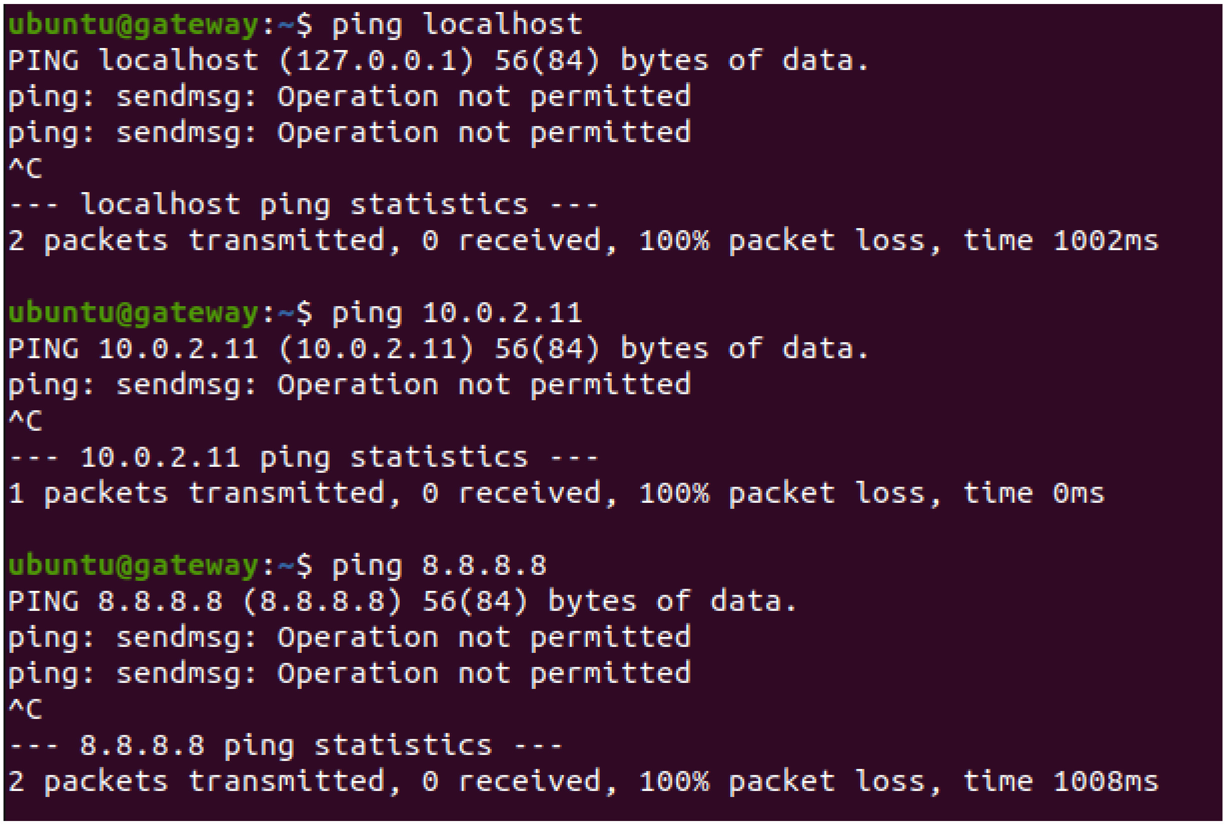


```
ubuntu@client:~$ sudo nmap -sU -p- 10.0.2.11
Starting Nmap 7.80 ( https://nmap.org ) at 2022-02-05 12:33 MST
Nmap scan report for 10.0.2.11
Host is up (0.00033s latency).
All 65535 scanned ports on 10.0.2.11 are open|filtered
MAC Address: 08:00:27:B2:34:3C (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1877.54 seconds
```



```
ubuntu@client:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=119 time=9.58 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=119 time=11.9 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1079ms
rtt min/avg/max/mdev = 9.579/10.736/11.894/1.157 ms
ubuntu@client:~$ ping 8.8.4.4
PING 8.8.4.4 (8.8.4.4) 56(84) bytes of data.
^C
--- 8.8.4.4 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3063ms

ubuntu@client:~$ ping 10.0.2.11
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data.
^C
--- 10.0.2.11 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2038ms
```

Gateway/server VM screenshots:



```
ubuntu@gateway:~$ ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- localhost ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1002ms

ubuntu@gateway:~$ ping 10.0.2.11
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
^C
--- 10.0.2.11 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

ubuntu@gateway:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1008ms
```

V. Conclusion

I learnt about how iptables works. I learnt what input chains such as INPUT, OUTPUT, FORWARD and POSTROUTING do and how to set their rules and policies.

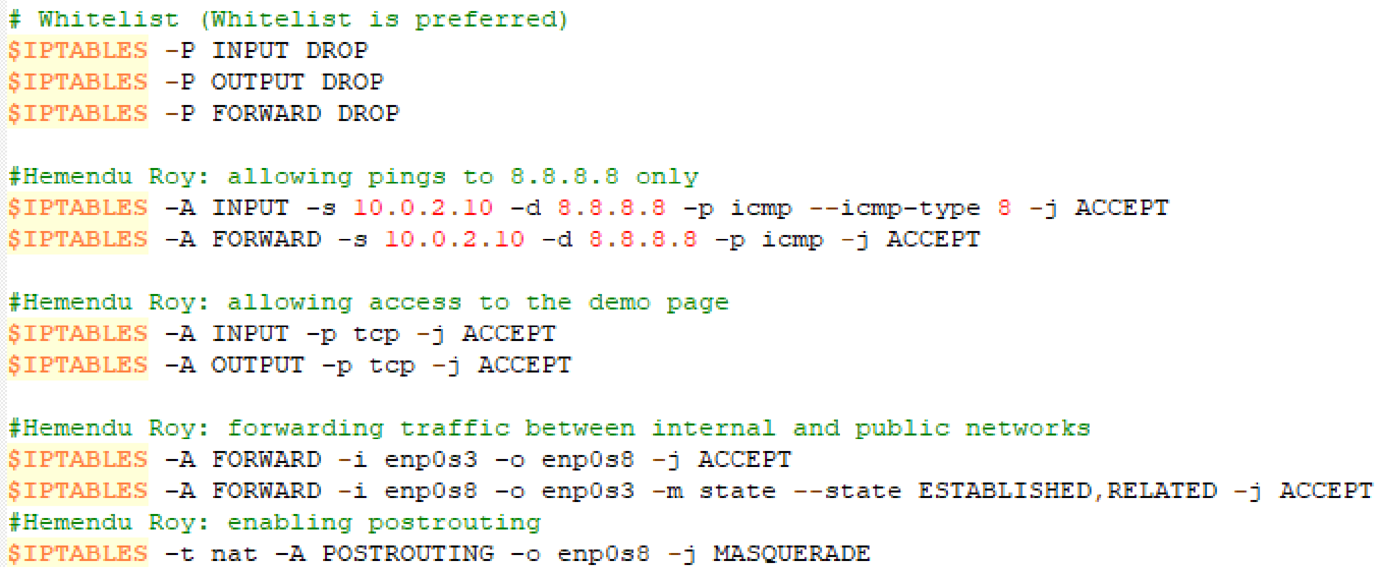
I also learnt that VirtualBox sets up an internet gateway on every NatNetwork adapter by default at the IP 10.0.x.1/24.
I learnt how to set up a basic apache webpage and restrict access to it as required.

Tips-

Playing around iptables policies and rules often breaks VMs and one cannot access the internet. It is a good idea to install all necessary tools first, before disabling internet access to both VMs and creating a snapshot. Incase of misconfigurations, it is easy to revert to an older snapshot.

VI. Appendix B: Attached files

All the iptables commands discussed above have been condensed into a single shell script to make it easier to manage firewall rules and setup packet filtering quicker.



```
# Whitelist (Whitelist is preferred)
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#Hemendu Roy: allowing pings to 8.8.8.8 only
$IPTABLES -A INPUT -s 10.0.2.10 -d 8.8.8.8 -p icmp --icmp-type 8 -j ACCEPT
$IPTABLES -A FORWARD -s 10.0.2.10 -d 8.8.8.8 -p icmp -j ACCEPT

#Hemendu Roy: allowing access to the demo page
$IPTABLES -A INPUT -p tcp -j ACCEPT
$IPTABLES -A OUTPUT -p tcp -j ACCEPT

#Hemendu Roy: forwarding traffic between internal and public networks
$IPTABLES -A FORWARD -i enp0s3 -o enp0s8 -j ACCEPT
$IPTABLES -A FORWARD -i enp0s8 -o enp0s3 -m state --state ESTABLISHED,RELATED -j ACCEPT
#Hemendu Roy: enabling postrouting
$IPTABLES -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE
```

The complete script can be downloaded from -
https://github.com/hemenduroy/Network_Security/blob/main/Packet%20Filter%20Firewall/rc.sh

VII. References

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

[1] iptables Linux man page - https://linux.die.net/man/8/iptables
[2] tcpdump Linux man page - https://linux.die.net/man/8/tcpdump
[3] wireshark - https://www.wireshark.org/
[4] Apache HTTP Server Project - https://httpd.apache.org/