Packet filter firewall project report

Student Name: Tiriac Ioana-Raluca

Email: itiriac@asu.edu

Submission Date: 24.10.2021

Class Name and Term: CSE548 Fall B 2021

I. Project Overview

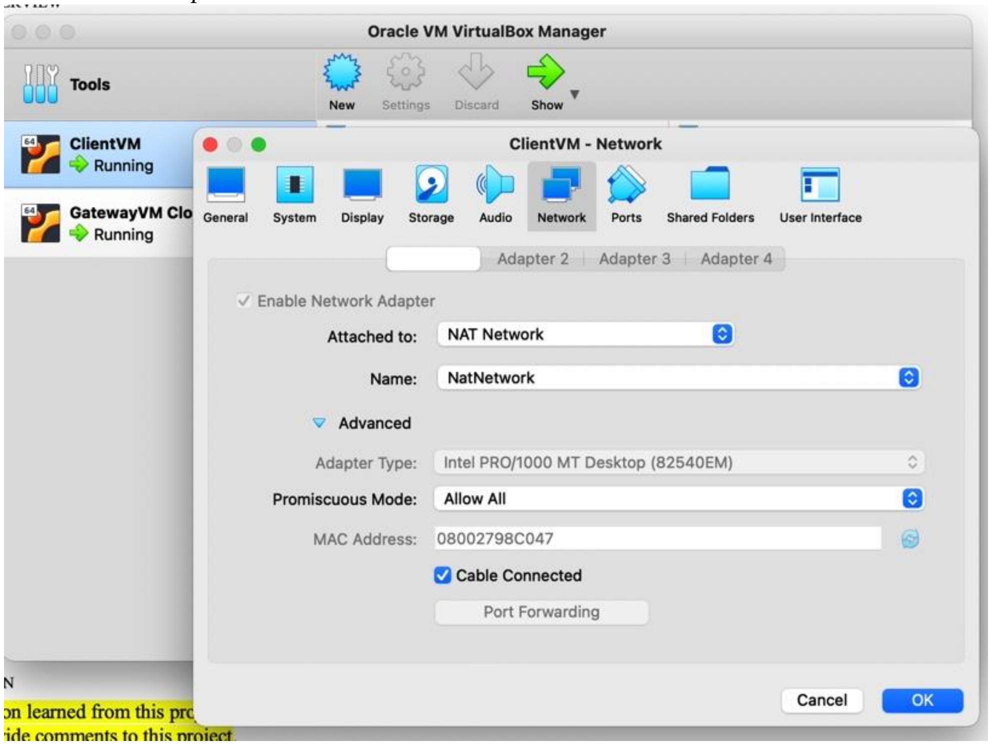
This project implements a stateless packet filter firewall by setting up an environment based on two Linux virtual machines. One of the virtual machines fulfills its role as a gateway accessing external networks through one interface, the other one is a client VM accessing external networks through the gateway. The gateway VM also has an Apache web server running on it and the goal is to setup a whitelist policy firewall on it, that will allow the client VM only certain actions like:

- Access web server from client vm through http and IP of gateway VM
- Not being able to ping gateway's IP
- Being able to ping Google's DNS 8.8.8.8
- Gateway should not be able to ping client VM, 8.8.8.8 nor localhost

II. Network Setup

As described in the background labs, the following are the configurations for the two virtual machines. They were created using the provided *project1.vdi* image.

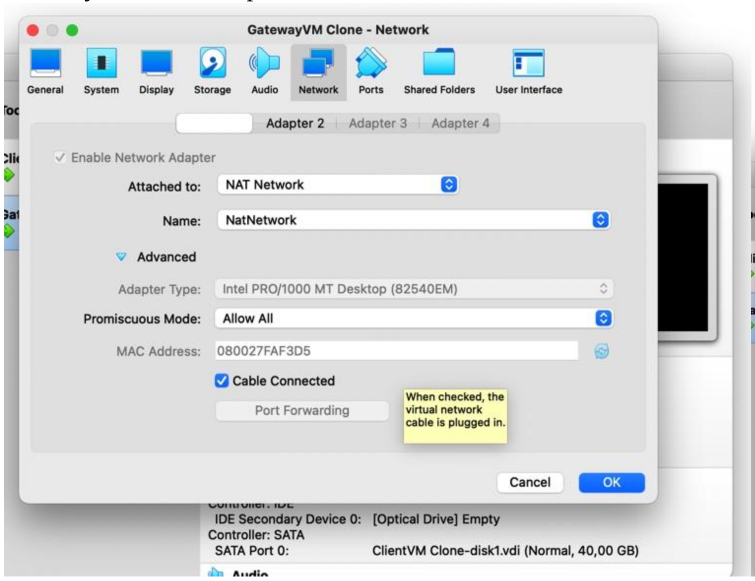
VM Network Setup



Enable Network Adapter: ✓
Attached to: NAT Network
Name: NatNetwork
Advanced
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode: Allow All
MAC Address: 08002798C047
Cable Connected: ✓
Port Forwarding

For this NatNetwork with CIDR 10.0.2.0/24 was created and assigned as Adaptor Network to the ClientVM.

Gateway Network Setup



GatewayVM Clone - Network

Adapter 2 | Adapter 3 | Adapter 4

✓ Enable Network Adapter

Attached to: NAT Network

Name: NatNetwork

Advanced

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow All

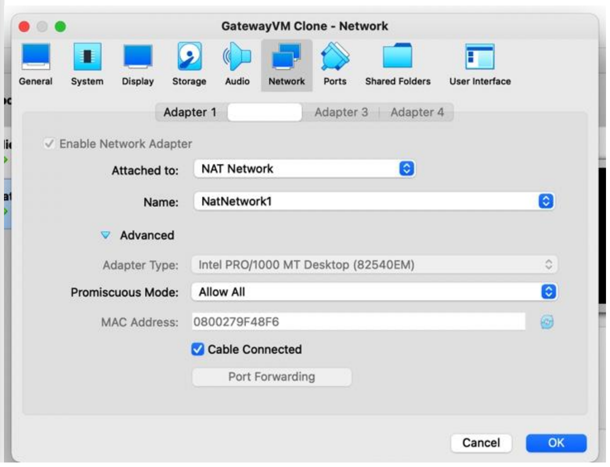
MAC Address: 080027FAF3D5

✓ Cable Connected

Port Forwarding

When checked, the virtual network cable is plugged in.

IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
SATA Port 0: ClientVM Clone-disk1.vdi (Normal, 40,00 GB)



GatewayVM Clone - Network

Adapter 1 | Adapter 3 | Adapter 4

✓ Enable Network Adapter

Attached to: NAT Network

Name: NatNetwork1

Advanced

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow All

MAC Address: 0800279F48F6

✓ Cable Connected

Port Forwarding

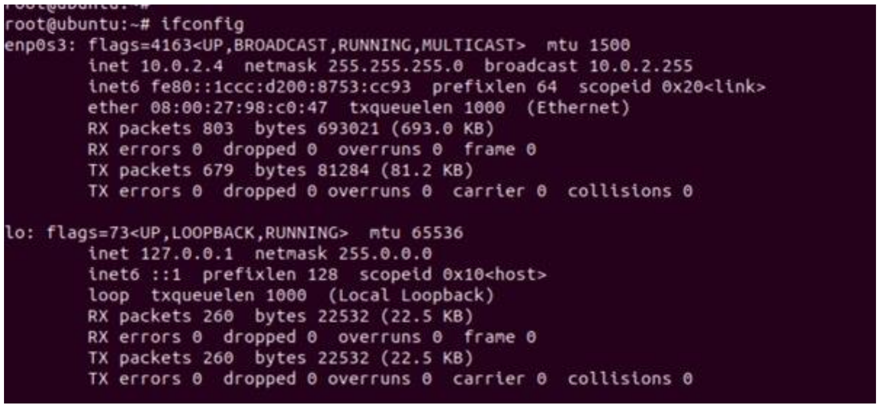
For the gateway VM 2 Nat Networks were assigned as network adapters, NatNetwork with CIDR 10.0.2.0/24 and NatNetwork1 with CIDR 10.0.1.0/24.

Gateway VM was created by cloning it from the ClientVM to clone the image they were both using (project1.vdi) and avoid issues with having both VMs running from the same image at the same time.

III. Software

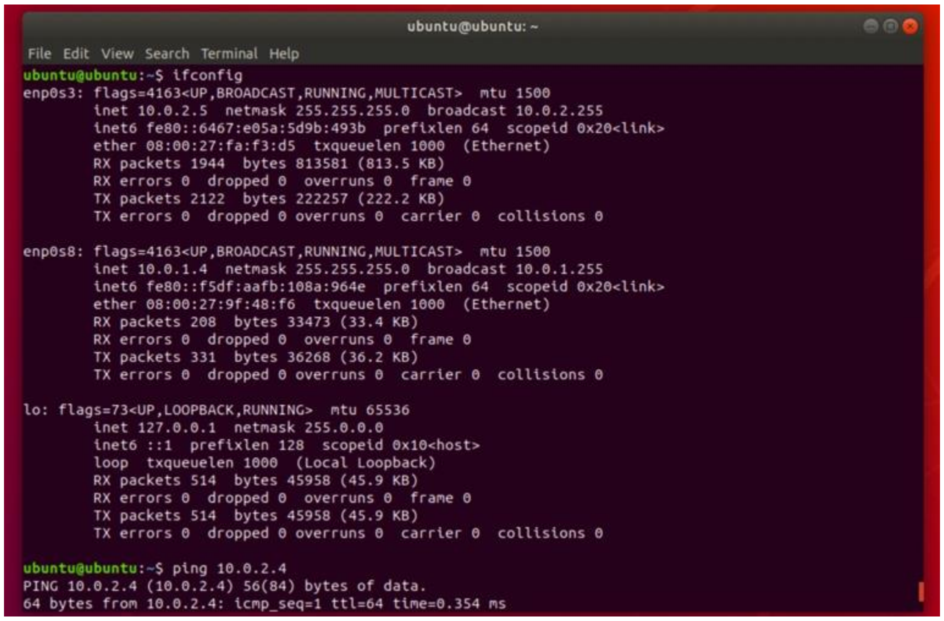
First VirtualBox was installed to Mac OS to be able to run the two virtual machines. After the two virtual machines were created, the following commands were issued on both of the machines to download some network management and diagnostic tools:

```
Sudo apt update
Sudo apt install net-tools
Sudo apt install traceroute
Sudo apt install wireshark
Sudo apt install ifmetric
Sudo apt install gnome-system-tools
Sudo apt install nmap
```



```
root@ubuntu:~# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.4  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::1ccc:d200:8753:cc93  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:98:c0:47  txqueuelen 1000  (Ethernet)
        RX packets 803  bytes 693021 (693.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 679  bytes 81284 (81.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 260  bytes 22532 (22.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 260  bytes 22532 (22.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Running ifconfig on the client VM above reveals it's IP address 10.0.2.4 and the local network interface it's connected to enp0s3.

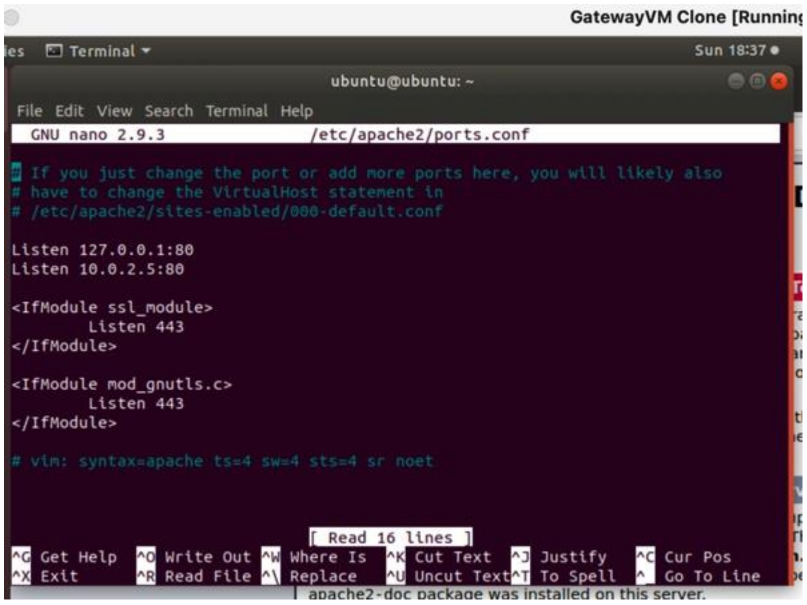


```
ubuntu@ubuntu:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.5  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::6467:e05a:5d9b:493b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:fa:f3:d5  txqueuelen 1000  (Ethernet)
        RX packets 1944  bytes 813581 (813.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2122  bytes 222257 (222.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.1.4  netmask 255.255.255.0  broadcast 10.0.1.255
        inet6 fe80::f5df:aafb:108a:964e  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:9f:48:f6  txqueuelen 1000  (Ethernet)
        RX packets 208  bytes 33473 (33.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 331  bytes 36268 (36.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 514  bytes 45958 (45.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 514  bytes 45958 (45.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@ubuntu:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.354 ms
```

Running ifconfig on the Gateway VM reveals the 2 network interfaces : local enp0s3 and global enp0s8. The gateway's IP is 10.0.2.5.

Apache 2 installation on gateway VM

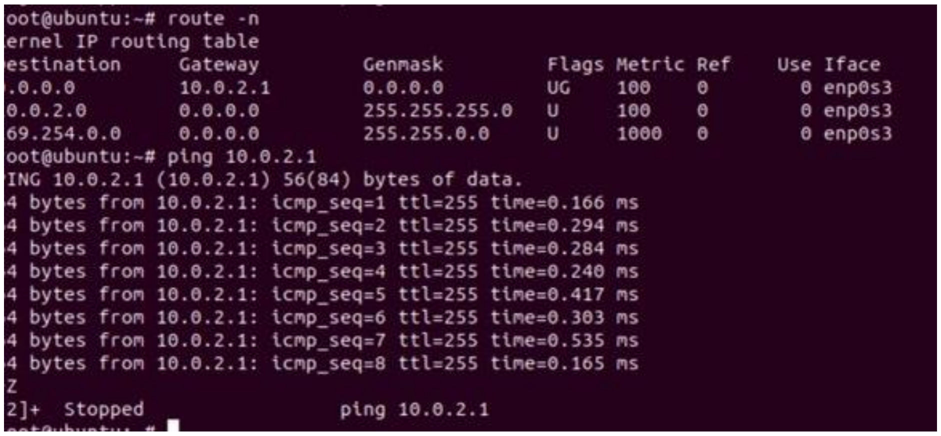
Apache2 web server was installed on the gateway VM by using the following command:
apt install apache2
The initial greeting was modified by going *to /var/www/html/index.html* and modifying the div containing the greeting.
To be able to access the web service from the client VM the list of IP's and ports the web server listens to has been modified to include the gateway's IP. Localhost 127.0.0.1:80 remained in the list to access Apache from the gateway itself through localhost or 127.0.0.1.



```
  GNU nano 2.9.3                 /etc/apache2/ports.conf

# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 127.0.0.1:80
Listen 10.0.2.5:80

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

                              [ Read 16 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

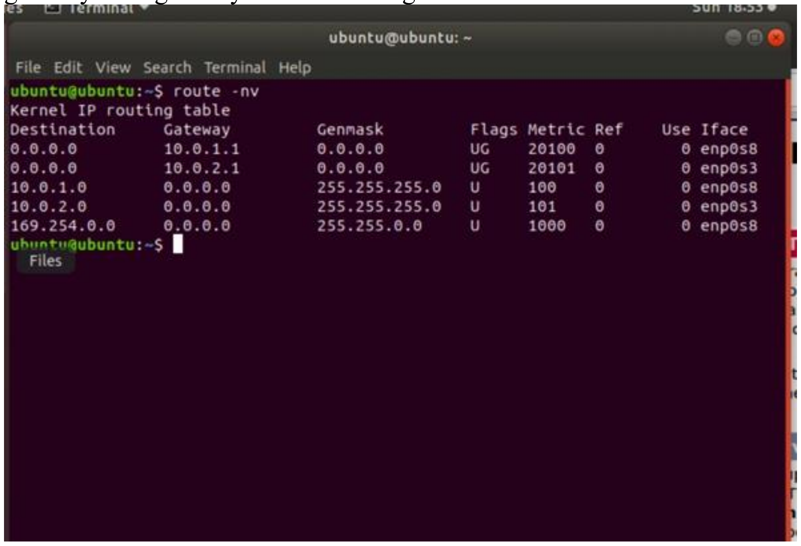
IV. Project Description

By using *route – n* command , the initial configuration of the client's routing tables can be seen below :



```
oot@ubuntu:~# route -n
ernel IP routing table
estination     Gateway         Genmask         Flags Metric Ref    Use Iface
.0.0.0         10.0.2.1        0.0.0.0         UG    100    0        0 enp0s3
0.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
69.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
oot@ubuntu:~# ping 10.0.2.1
ING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
4 bytes from 10.0.2.1: icmp_seq=1 ttl=255 time=0.166 ms
4 bytes from 10.0.2.1: icmp_seq=2 ttl=255 time=0.294 ms
4 bytes from 10.0.2.1: icmp_seq=3 ttl=255 time=0.284 ms
4 bytes from 10.0.2.1: icmp_seq=4 ttl=255 time=0.240 ms
4 bytes from 10.0.2.1: icmp_seq=5 ttl=255 time=0.417 ms
4 bytes from 10.0.2.1: icmp_seq=6 ttl=255 time=0.303 ms
4 bytes from 10.0.2.1: icmp_seq=7 ttl=255 time=0.535 ms
4 bytes from 10.0.2.1: icmp_seq=8 ttl=255 time=0.165 ms
Z
2]+  Stopped                 ping 10.0.2.1
```

It's visible the client VM was able to ping the local network and the gateway VM's ip 10.0.2.5 was not listed as a default gateway. The gateway's initial routing table was:

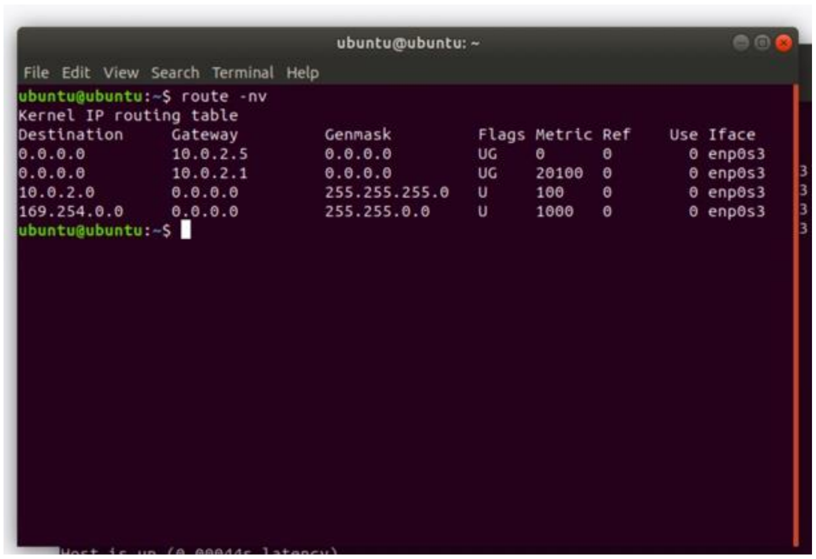


```
ubuntu@ubuntu:~$ route -nv
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.1.1        0.0.0.0         UG    20100  0        0 enp0s8
0.0.0.0         10.0.2.1        0.0.0.0         UG    20101  0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s8
10.0.2.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s8
ubuntu@ubuntu:~$
```

After adding the gateway's IP to the client VM's routing table as a default gateway through command :

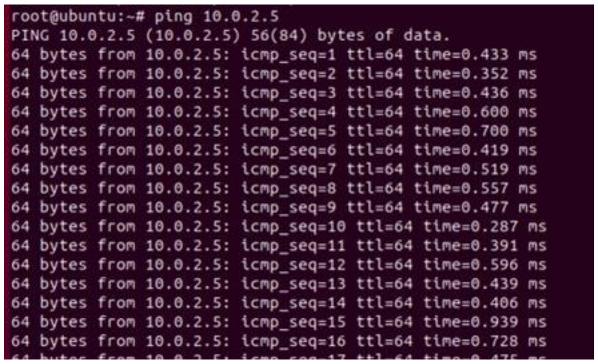
route add default gw 10.0.2.5 enp0s3

the routing table changed like this:



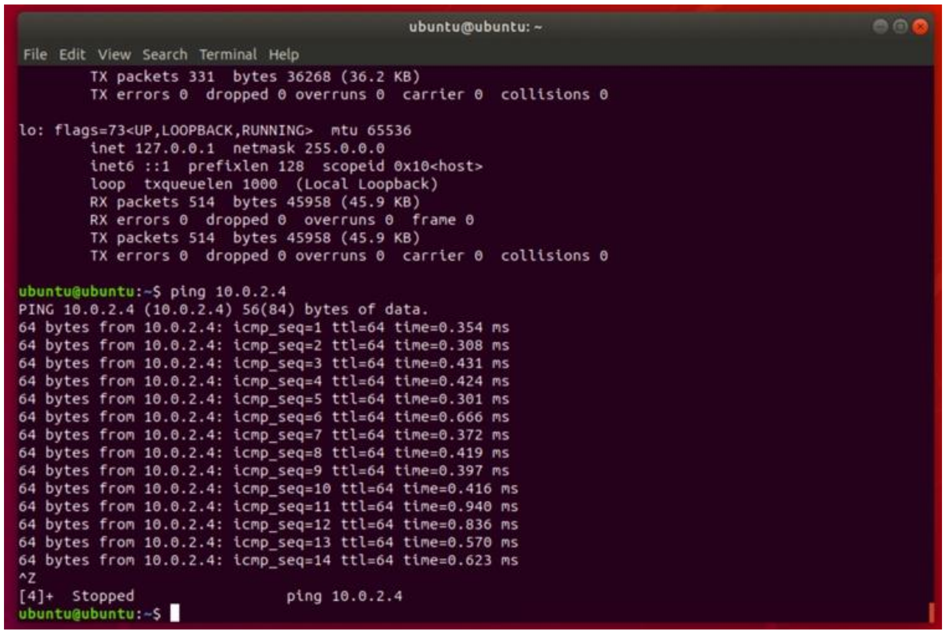
```
ubuntu@ubuntu:~$ route -nv
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.5        0.0.0.0         UG    0      0        0 enp0s3
0.0.0.0         10.0.2.1        0.0.0.0         UG    20100  0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
ubuntu@ubuntu:~$
```

Task 1.1. network connectivity was solved, the client VM could successfully ping the gateway and external network and the gateway VM could successfully ping the client and the external network. Before that the gateway's firewall was checked with iptables -L to find out it had a blacklist policy.



```
root@ubuntu:~# ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.433 ms
64 bytes from 10.0.2.5: icmp_seq=2 ttl=64 time=0.352 ms
64 bytes from 10.0.2.5: icmp_seq=3 ttl=64 time=0.436 ms
64 bytes from 10.0.2.5: icmp_seq=4 ttl=64 time=0.600 ms
64 bytes from 10.0.2.5: icmp_seq=5 ttl=64 time=0.700 ms
64 bytes from 10.0.2.5: icmp_seq=6 ttl=64 time=0.419 ms
64 bytes from 10.0.2.5: icmp_seq=7 ttl=64 time=0.519 ms
64 bytes from 10.0.2.5: icmp_seq=8 ttl=64 time=0.557 ms
64 bytes from 10.0.2.5: icmp_seq=9 ttl=64 time=0.477 ms
64 bytes from 10.0.2.5: icmp_seq=10 ttl=64 time=0.287 ms
64 bytes from 10.0.2.5: icmp_seq=11 ttl=64 time=0.391 ms
64 bytes from 10.0.2.5: icmp_seq=12 ttl=64 time=0.596 ms
64 bytes from 10.0.2.5: icmp_seq=13 ttl=64 time=0.439 ms
64 bytes from 10.0.2.5: icmp_seq=14 ttl=64 time=0.406 ms
64 bytes from 10.0.2.5: icmp_seq=15 ttl=64 time=0.939 ms
64 bytes from 10.0.2.5: icmp_seq=16 ttl=64 time=0.728 ms
```

pinging the gateway worked

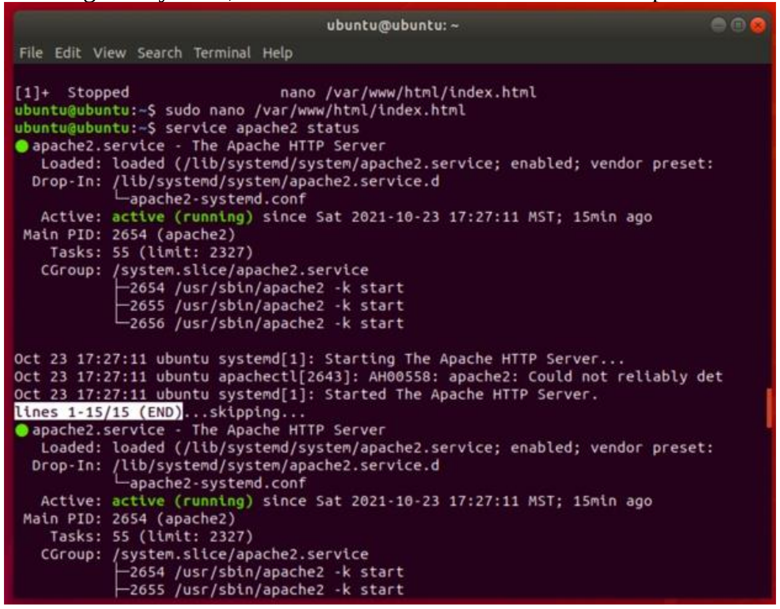


```
        TX packets 331  bytes 36268 (36.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 514  bytes 45958 (45.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 514  bytes 45958 (45.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@ubuntu:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.354 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.308 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.431 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.424 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.301 ms
64 bytes from 10.0.2.4: icmp_seq=6 ttl=64 time=0.666 ms
64 bytes from 10.0.2.4: icmp_seq=7 ttl=64 time=0.372 ms
64 bytes from 10.0.2.4: icmp_seq=8 ttl=64 time=0.419 ms
64 bytes from 10.0.2.4: icmp_seq=9 ttl=64 time=0.397 ms
64 bytes from 10.0.2.4: icmp_seq=10 ttl=64 time=0.416 ms
64 bytes from 10.0.2.4: icmp_seq=11 ttl=64 time=0.940 ms
64 bytes from 10.0.2.4: icmp_seq=12 ttl=64 time=0.836 ms
64 bytes from 10.0.2.4: icmp_seq=13 ttl=64 time=0.570 ms
64 bytes from 10.0.2.4: icmp_seq=14 ttl=64 time=0.623 ms
^Z
[4]+  Stopped                 ping 10.0.2.4
ubuntu@ubuntu:~$
```

pinging the client vm worked

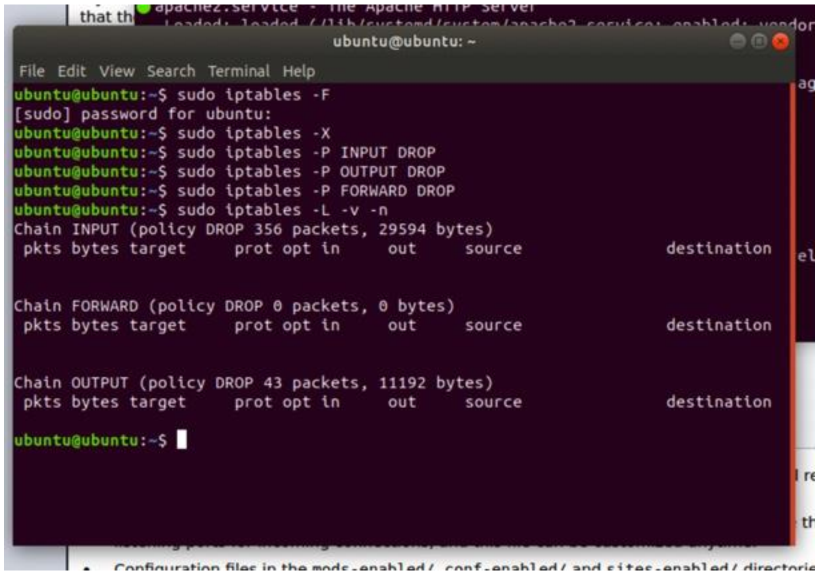
Task 1.2. Test installed software – apache2 and services

On the gateway VM , the web service was started : Service apache2 status



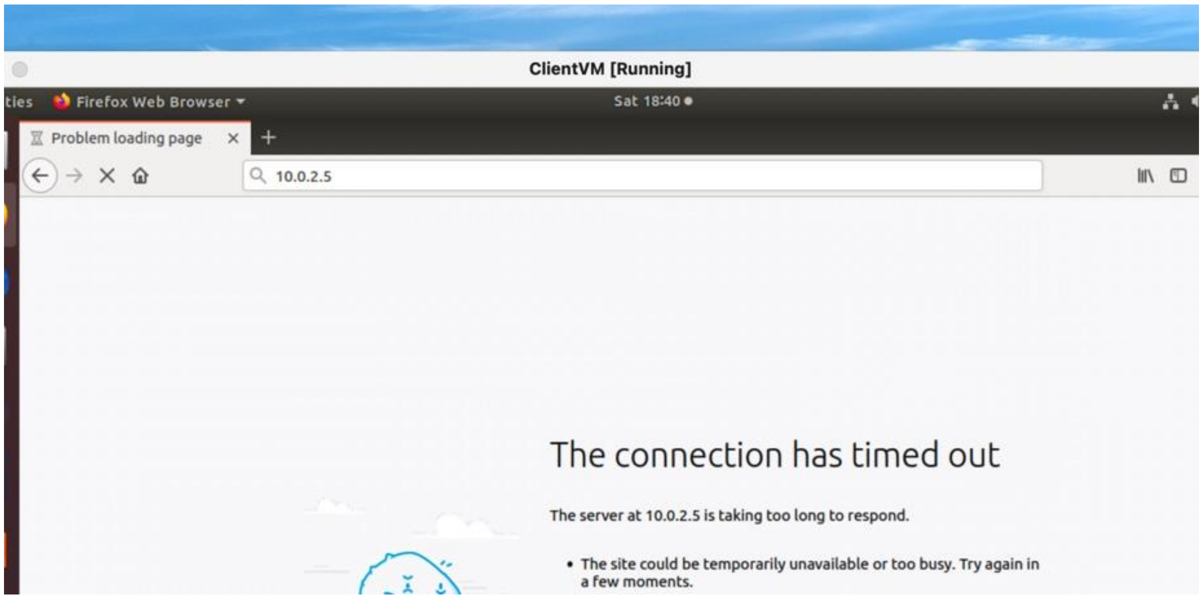
```
[1]+  Stopped                 nano /var/www/html/index.html
ubuntu@ubuntu:~$ sudo nano /var/www/html/index.html
ubuntu@ubuntu:~$ service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Sat 2021-10-23 17:27:11 MST; 15min ago
 Main PID: 2654 (apache2)
    Tasks: 55 (limit: 2327)
   CGroup: /system.slice/apache2.service
           ├─2654 /usr/sbin/apache2 -k start
           ├─2655 /usr/sbin/apache2 -k start
           └─2656 /usr/sbin/apache2 -k start

Oct 23 17:27:11 ubuntu systemd[1]: Starting The Apache HTTP Server...
Oct 23 17:27:11 ubuntu apachectl[2643]: AH00558: apache2: Could not reliably det
Oct 23 17:27:11 ubuntu systemd[1]: Started The Apache HTTP Server.
lines 1-15/15 (END)...skipping...
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Sat 2021-10-23 17:27:11 MST; 15min ago
 Main PID: 2654 (apache2)
    Tasks: 55 (limit: 2327)
   CGroup: /system.slice/apache2.service
           ├─2654 /usr/sbin/apache2 -k start
           ├─2655 /usr/sbin/apache2 -k start
```

Task 1.3. Next the firewall policy on the gateway was changed from blacklist to whitelist by using the following commands. Later it was learned the rc.sh firewall script also did that every time it was run.



```
ubuntu@ubuntu:~$ sudo iptables -F
[sudo] password for ubuntu:
ubuntu@ubuntu:~$ sudo iptables -X
ubuntu@ubuntu:~$ sudo iptables -P INPUT DROP
ubuntu@ubuntu:~$ sudo iptables -P OUTPUT DROP
ubuntu@ubuntu:~$ sudo iptables -P FORWARD DROP
ubuntu@ubuntu:~$ sudo iptables -L -v -n
Chain INPUT (policy DROP 356 packets, 29594 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 43 packets, 11192 bytes)
 pkts bytes target     prot opt in     out     source               destination

ubuntu@ubuntu:~$
```

After changing the firewall policy, the web server was not accessible from the client VM anymore:



The connection has timed out

The server at 10.0.2.5 is taking too long to respond.

- The site could be temporarily unavailable or too busy. Try again in a few moments.

Next Step: Changing and running firewall script rc.sh on gateway VM

The provided rc.sh template script was changed with *chmod +x rc.sh* to make it an executable file and ran with *./rc.sh.* It was modified to allow the following:

- Access web server from client vm through http and IP of gateway VM
- Not being able to ping gateway’s IP from client VM
- Being able to ping Google’s DNS 8.8.8.8
- Gateway should not be able to ping client VM, 8.8.8.8 nor localhost

The following describes a part of the firewall script, containing all the rules for the FORWARD, INPUT, OUTPUT and POSTROUTING chain rules that enforce the above:

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

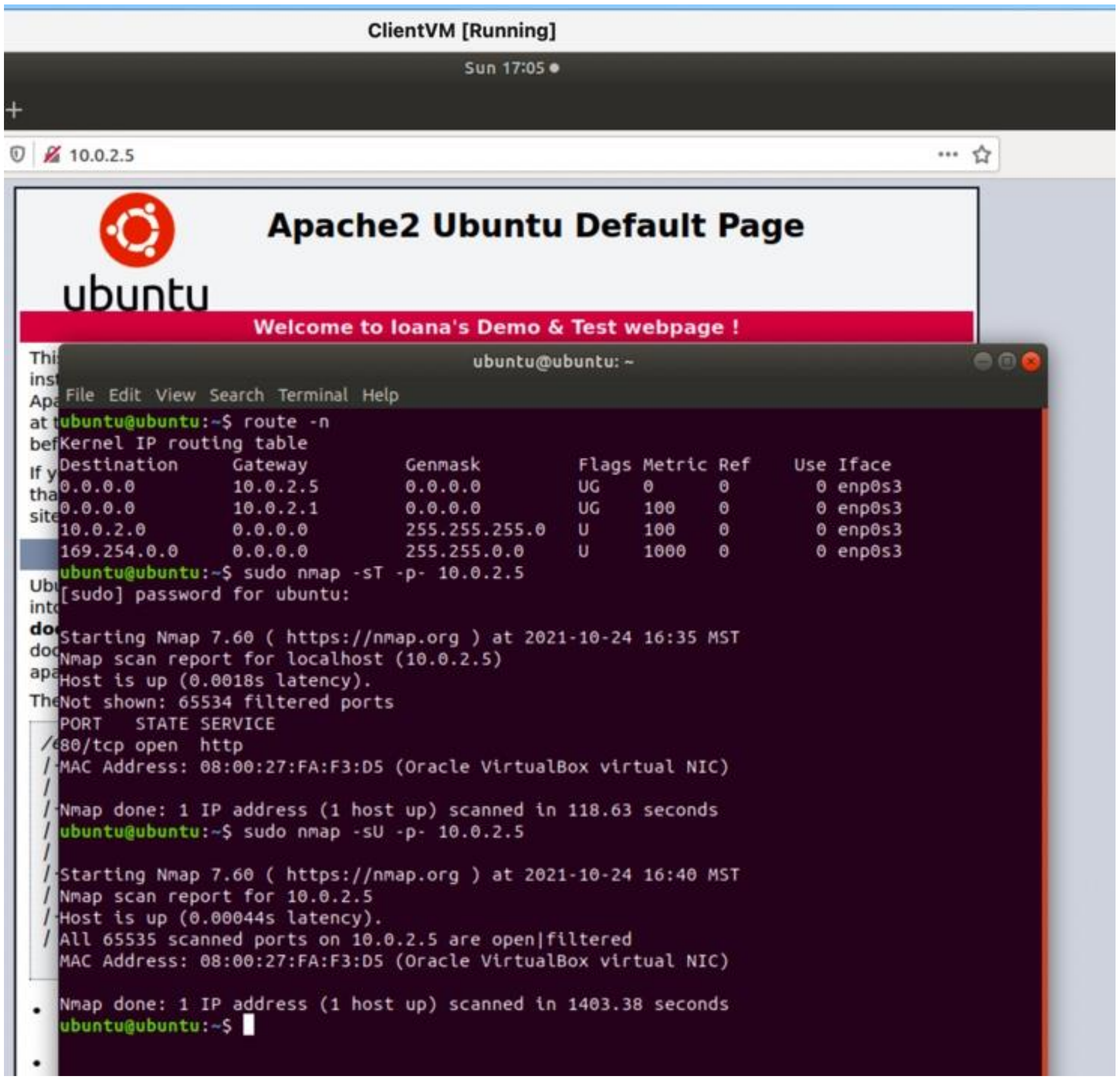
```
#

#
# Provide your forwarding rules below
#

# example of checking bad tcp packets
#$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

# Allow ping from client network to google DNS, and from google DNS back
$IPTABLES -A FORWARD -p icmp -o $Client_NET_IFACE -s 8.8.8.8 -j ACCEPT
$IPTABLES -A FORWARD -p icmp -s 10.0.2.4 -i $Client_NET_IFACE -d 8.8.8.8 -j ACCEPT

# Allow return packets from established connections
$IPTABLES -A FORWARD -p TCP -o $Client_NET_IFACE -i $Internet_IFACE -m state --state
ESTABLISHED,RELATED -j ACCEPT

#Allow HTTP/HTTPS internet traffic for client to update OS packages

$IPTABLES -A FORWARD -p TCP -i $Client_NET_IFACE -o $Internet_IFACE --dport 80 -j
ACCEPT
$IPTABLES -A FORWARD -p TCP -i $Client_NET_IFACE -o $Internet_IFACE --dport 443 -j
ACCEPT


#####
# 4.3 INPUT chain
#

#
# Provide your input rules below to allow web traffic from client
#
#$IPTABLES -A INPUT -p TCP --dport 80 -i $Client_NET_IFACE -s 10.0.2.5 -j ACCEPT

# Allow loopback traffic - 127.0.0.1 or localhost

$IPTABLES -A INPUT -p TCP -i lo -j ACCEPT
$IPTABLES -A INPUT -p TCP -i lo -s 127.0.0.1 -j ACCEPT

# Allow syn and return packets from established connections
$IPTABLES -A INPUT -p TCP --syn -j ACCEPT
$IPTABLES -A INPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT


# Allow client and loopback to connect to web server on Gateway

$IPTABLES -A INPUT -p TCP --dport 80 -i $Client_NET_IFACE -j ACCEPT
$IPTABLES -A INPUT -p TCP --dport 80 -i $LO_IFACE -d $WEB_IP_ADDRESS -j ACCEPT

# Allow traffic to HTTP/HTTPS for Gateway - for OS updates

$IPTABLES -A INPUT -p TCP --sport 80 -i $Internet_IFACE -j ACCEPT
$IPTABLES -A INPUT -p TCP --sport 443 -i $Internet_IFACE -j ACCEPT

#
# Example of checking bad TCP packets we don't want.
#

#$IPTABLES -A INPUT -p tcp -j bad_tcp_packets
```

```
# ACCEPT ping from client VM's IP to DNS 8.8.8.8

$IPTABLES -A INPUT -p icmp -s 10.0.2.4 -d 8.8.8.8 -j ACCEPT

#REJECT ping to localhost
$IPTABLES -A INPUT -p icmp -i lo -d 127.0.0.1 -j REJECT


#####
# 4.3 OUTPUT chain
#

#
# Provide your output rules below to allow web traffic (port 80) go back to client
#

# example Allowed ping message back to client
$IPTABLES -A OUTPUT -p icmp -s 8.8.8.8 -j ACCEPT

# Allow loopback on output chain
$IPTABLES -A OUTPUT -o $LO_IFACE -j ACCEPT
$IPTABLES -A OUTPUT -o $LO_IFACE -s 127.0.0.1 -j ACCEPT

# Allow syn and return packets from established connections
$IPTABLES -A OUTPUT -p TCP --syn -j ACCEPT
$IPTABLES -A OUTPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT

#Allow TCP traffic back to client
# Allow gateway server response to reach client and loopback

$IPTABLES -A OUTPUT -p TCP --sport 80 -o $Client_NET_IFACE -j ACCEPT
$IPTABLES -A OUTPUT -p TCP --sport 80 -o $LO_IFACE -j ACCEPT

#Allow traffic to HTTP/HTTPS for Gateway - to update OS

$IPTABLES -A OUTPUT -p TCP --dport 80 -o $Internet_IFACE -j ACCEPT
$IPTABLES -A OUTPUT -p TCP --dport 443 -o $Internet_IFACE -j ACCEPT
######################################################################
#                                                                    #
# 5. NAT setup                                                       #
#                                                                    #
######################################################################

#####
# 5.1  PREROUTING chain. (No used in this lab)
#
#
# Provide your NAT PREROUTING rules (packets come into your private domain)
#

#
# Example of enable http to internal web server behind the firewall (port forwarding)
#

# web
$IPTABLES -t nat -A PREROUTING -p tcp -d $NAT_WEB_IP_ADDRESS --dport 80 -j DNAT --to
$WEB_IP_ADDRESS
```

```
#####
# 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

# Only allow client node to ping google DNS using masquerade

$IPTABLES -t nat -A POSTROUTING -p icmp -o $Internet_IFACE -d 8.8.8.8 -j MASQUERADE

#Allow Internet Traffic to HTTP/HTTPS for client to be able to update OS
$IPTABLES -t nat -A POSTROUTING -p tcp -o $Internet_IFACE --dport 80 -j MASQUERADE
$IPTABLES -t nat -A POSTROUTING -p tcp -o $Internet_IFACE --dport 443 -j MASQUERADE
```

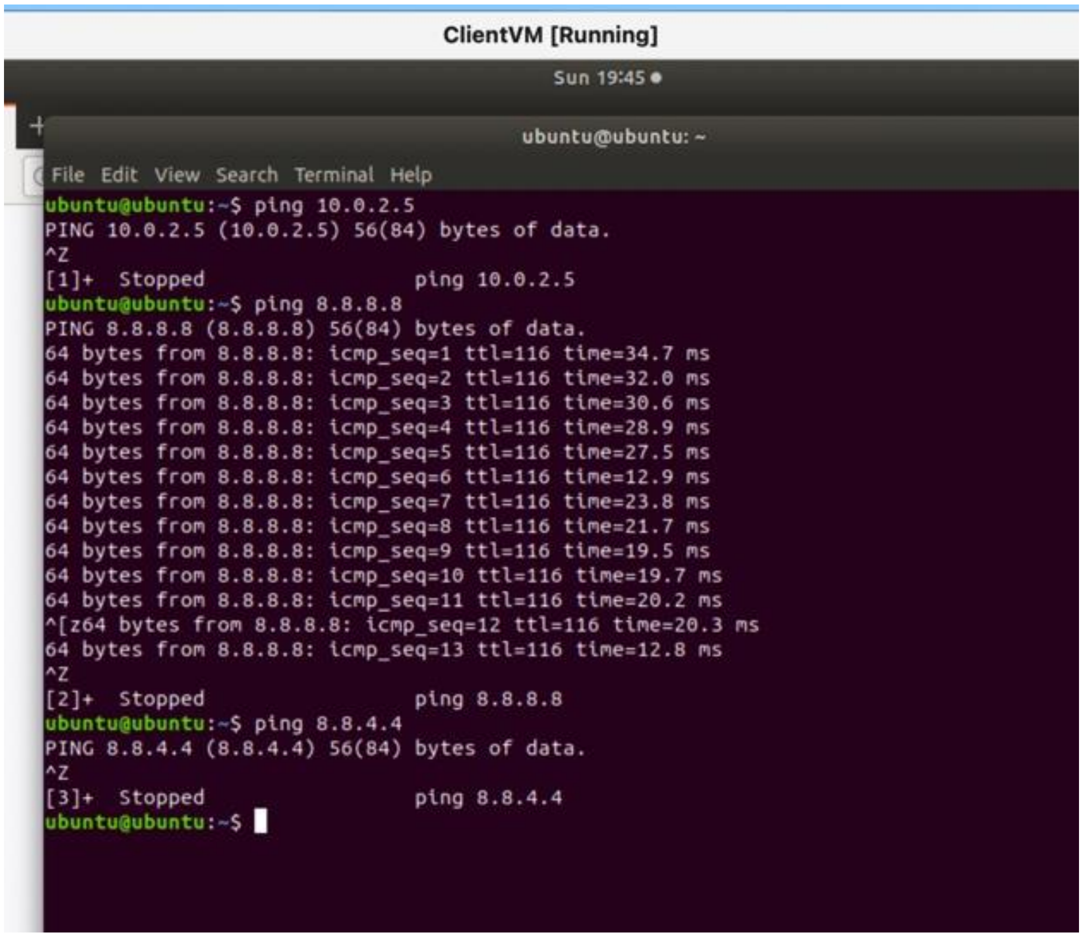
After the firewall script is run with ./rc.sh the following are enforced:

The client vm can access the demo webpage by accessing gateway's IP and stats for *sudo nmap* for gateway IP through TCP and UDP are also provided:


ClientVM [Running]
Sun 17:05
10.0.2.5
Apache2 Ubuntu Default Page
ubuntu
Welcome to Ioana's Demo & Test webpage !
ubuntu@ubuntu: ~
File Edit View Search Terminal Help

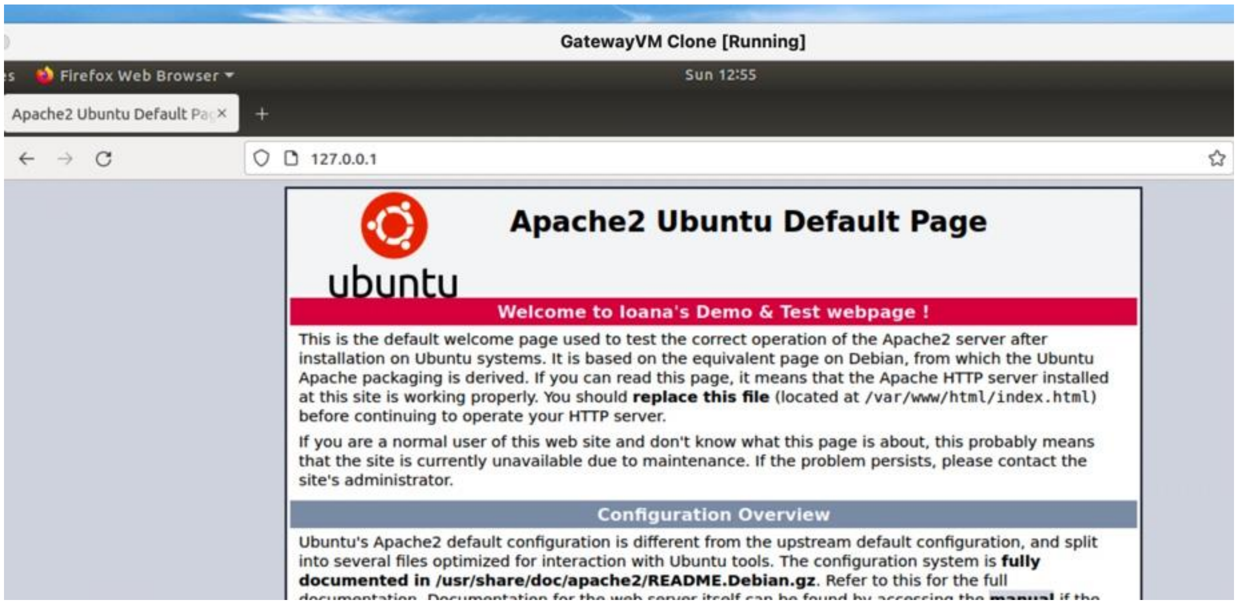
```
ubuntu@ubuntu:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.5        0.0.0.0         UG    0      0        0 enp0s3
0.0.0.0         10.0.2.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
ubuntu@ubuntu:~$ sudo nmap -sT -p- 10.0.2.5
[sudo] password for ubuntu:

Starting Nmap 7.60 ( https://nmap.org ) at 2021-10-24 16:35 MST
Nmap scan report for localhost (10.0.2.5)
Host is up (0.0018s latency).
Not shown: 65534 filtered ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:FA:F3:D5 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 118.63 seconds
ubuntu@ubuntu:~$ sudo nmap -sU -p- 10.0.2.5

Starting Nmap 7.60 ( https://nmap.org ) at 2021-10-24 16:40 MST
Nmap scan report for 10.0.2.5
Host is up (0.00044s latency).
All 65535 scanned ports on 10.0.2.5 are open|filtered
MAC Address: 08:00:27:FA:F3:D5 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1403.38 seconds
ubuntu@ubuntu:~$
```

The client cannot ping the gateway IP or any other IP, except for 8.8.8.8:



```
ubuntu@ubuntu:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
^Z
[1]+  Stopped                 ping 10.0.2.5
ubuntu@ubuntu:~$ 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=116 time=34.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=32.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=30.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=28.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=27.5 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=116 time=12.9 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=116 time=23.8 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=116 time=21.7 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=116 time=19.5 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=116 time=19.7 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=116 time=20.2 ms
^[z64 bytes from 8.8.8.8: icmp_seq=12 ttl=116 time=20.3 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=116 time=12.8 ms
^Z
[2]+  Stopped                 ping 8.8.8.8
ubuntu@ubuntu:~$ ping 8.8.4.4
PING 8.8.4.4 (8.8.4.4) 56(84) bytes of data.
^Z
[3]+  Stopped                 ping 8.8.4.4
ubuntu@ubuntu:~$ 
```

The gateway has setup the webpage to its own IP as it is visible from the screenshots taken that access it through 10.0.2.5 on Client VM and as it was described in ports.conf. The gateway can access its own web service through localhost or 127.0.0.1 thanks to the firewall rules that allow loopback traffic (view above in firewall rules):



Apache2 Ubuntu Default Page

Welcome to Ioana's Demo & Test webpage !

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

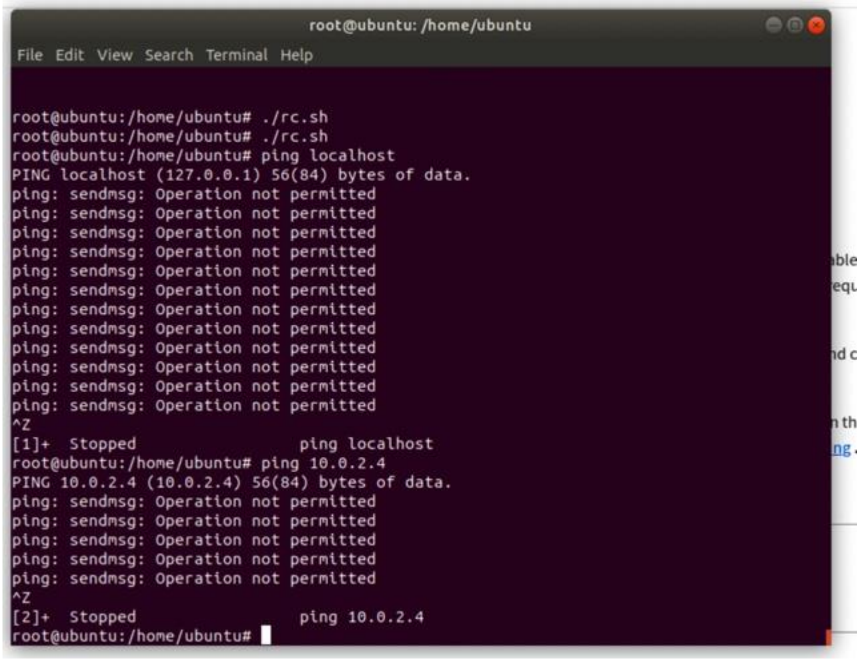
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full

POSTROUTING has also been enabled as is visible here in the firewall script, this allows client vm to ping 8.8.8.8 and changes their source IP address to that of the gateway.

```
#####
# 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

# Tiriac: Only allow client node to ping google DNS using masquerade

$IPTABLES -t nat -A POSTROUTING -p icmp -o $Internet_IFACE -d 8.8.8.8 -j MASQUERADE

# Tiriac: Allow Internet Traffic to HTTP/HTTPS for client to be able to update OS
$IPTABLES -t nat -A POSTROUTING -p tcp -o $Internet_IFACE --dport 80 -j MASQUERADE
$IPTABLES -t nat -A POSTROUTING -p tcp -o $Internet_IFACE --dport 443 -j MASQUERADE
```

Further, the following are not possible on the gateway VM:

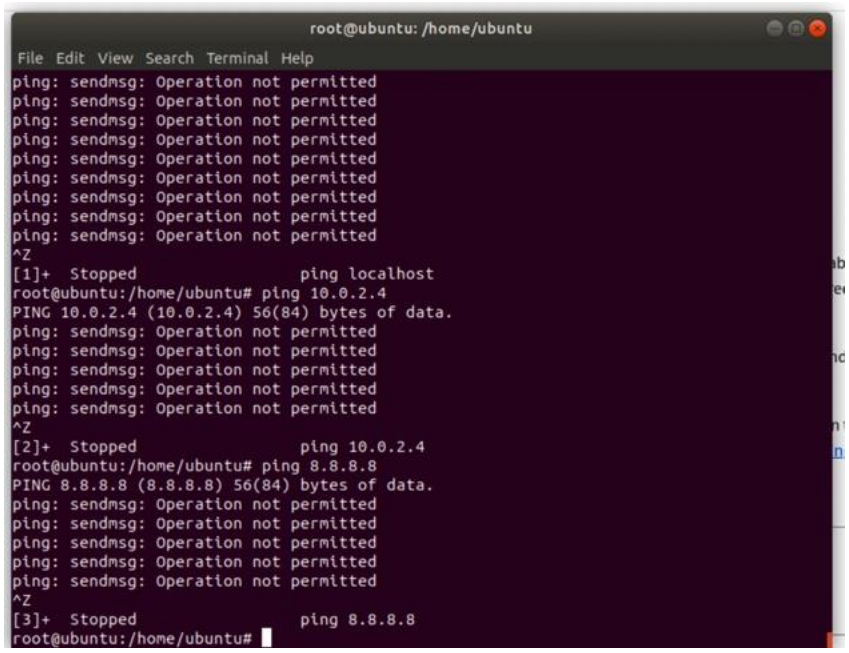
- **Ping localhost and ping client VM (it's IP is 10.0.2.4)**



```
root@ubuntu:/home/ubuntu# ./rc.sh
root@ubuntu:/home/ubuntu# ./rc.sh
root@ubuntu:/home/ubuntu# ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^Z
[1]+  Stopped                 ping localhost
root@ubuntu:/home/ubuntu# ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^Z
[2]+  Stopped                 ping 10.0.2.4
root@ubuntu:/home/ubuntu# 
```

root@ubuntu: /home/ubuntu
File Edit View Search Terminal Help

- **Ping 8.8.8.8 is also not possible**


root@ubuntu: /home/ubuntu
File Edit View Search Terminal Help

```
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^Z
[1]+  Stopped                 ping localhost
root@ubuntu:/home/ubuntu# ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^Z
[2]+  Stopped                 ping 10.0.2.4
root@ubuntu:/home/ubuntu# 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
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^Z
[3]+  Stopped                 ping 8.8.8.8
root@ubuntu:/home/ubuntu# 
```

V. CONCLUSION

- This project has helped me recap basic linux commands and helped me learn use networking tools like netstat, ifconfig and route.
- It has helped me understand how to configure a firewall script on a gateway and how to debug it in case of need.
- I've understood how to use iptables and chain rules to setup a firewall policy.

VI. APPENDIX B: ATTACHED FILES

The modified firewall script rc.sh has been attached to the .zip archive and contains all the added rules that enforce this lab's assessments and their corresponding comments.
Screenshots have been taken for the web server's configuration files: ports.conf and index.html.

VII. REFERENCES

[1] https://linux.die.net/man/8/iptables
[2] https://www.redhat.com/sysadmin/netstat
[3] https://goinbigdata.com/demystifying-ifconfig-and-network-interfaces-in-linux/