

Project Report 1 - Packet Filter Firewall

Student Name: Marco Ermini

Email: mermini@asu.edu

Submission Date: 29th May, 2021

Class Name and Term: CSE548 Summer 2021

I. PROJECT OVERVIEW

In this lab we are exploring how a packet filter firewall works by setting up an environment based on two Linux virtual machines – one working as a dual-homes Gateway client, which can access external networks on one interface – and another set up as a Client, which can only access external networks through the gateway. The gateway will be configured with the Linux iptables firewall and will also enable NAT for selected protocols.

The lab will also setup a web server on the Gateway with a test web page. We will test having full control of the network traffic, allowing only specific protocols for specific destinations at will, by modifying specific parameters in the firewall script.

All the files and configurations used for this lab have been uploaded on GitHub; references are provided throughout the text and in the Appendix A at the bottom of the File.

II. NETWORK SETUP

Please find below the set-up of the virtual infrastructure as I have configured it in VirtualBox.

The peculiarity of my infrastructure is that I have not used DHCP and a default gateway for the internal network (10.0.2/24). This is because, even if I force the Client VM to use the Gateway VM as the default gateway, the Ubuntu machine will still detect the default gateway offered by VirtualBox on the same network (which is the DHCP server configured by VirtualBox), and the VM will configure itself to use that default gateway, bypassing the Gateway VM, and would end up using VirtualBox' DHCP server on that subnet as a router to reach the internet.

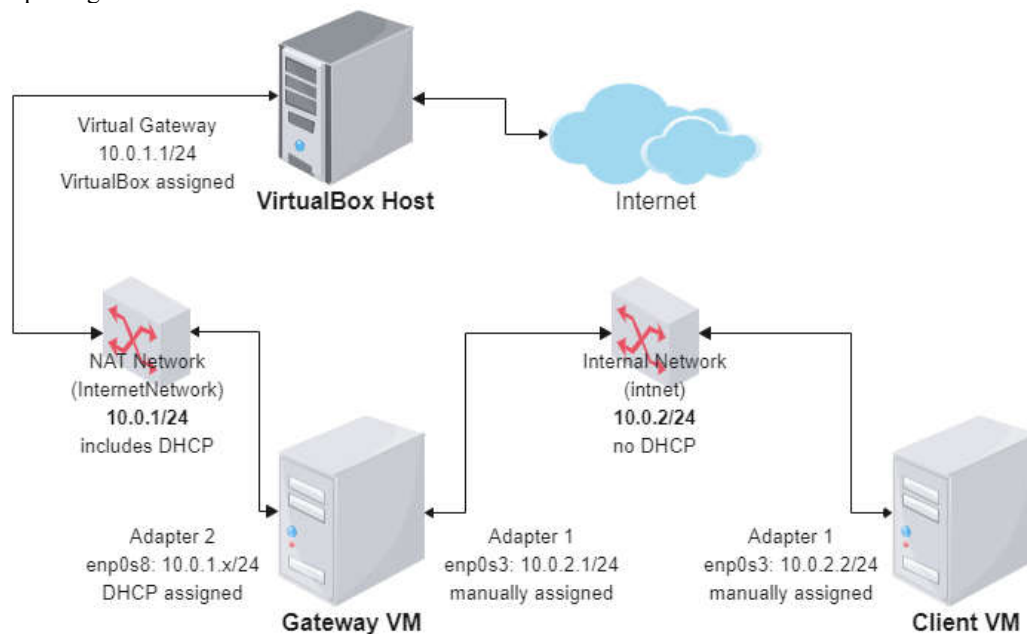


Figure 1 - Network Map of the Lab

Therefore, for the purpose of effectively executing the exercises in this lab and working with routing and iptables firewall rules, I have opted for a different set up, where the Gateway is forced in the path as the only possible gateway for the client. To achieve that, I have disabled VirtualBox's DHCP in the internal network (10.0.2/24) and manually configured the IPs on the internal interfaces of the Gateway VM and the Client VM. I have also configured the Client VM to use the Gateway VM's IP (10.0.2.1) as the default gateway.

By using this set up, the Client VM can only pass through the Gateway VM for external reachability. Note that because of the kind of configuration, there is no reason to mess up with any routing command on the client – **the correct routing is automatic**.

The overall initial configuration looks like the followings:

	Gateway VM – Adapter 1	Gateway VM – Adapter 2	Client VM – Adapter 1
IP address	10.0.2.x/24 (DHCP assigned)	10.0.1.1/24	10.0.2.2/24
Gateway	On interface	Adapter 1	10.0.2.1
Initial Reachability	Internet via VirtualBox host	Client VM IP	Gateway Adapter 2 IP

The way in which Ubuntu's 18.04 LTS versions configure network interfaces is through an application called *NetPlan*. Its configuration scripts, which assume the form of YAML files, are located under /etc/netplan. By default, on that Ubuntu version, NetPlan is configured to use NetworkManager to handle the IP addresses. I have found it was easier to just override the default configuration (present on the file *01-network-manager-all.yaml*) and manually assign the IP address and configure the use of DHCP where required. The following configurations were used (**all the files are available on GitHub**, see Appendix A):

On the gateway:

```
root@ubuntu:/etc/netplan# cat 02-gateway-networks.yaml
```

```
network:
```

```
  renderer: networkd
```

```
  version: 2
```

```
  ethernets:
```

```
    enp0s3:
```

```
      addresses: [10.0.2.1/24]
```

```
      #gateway4: 10.0.2.1
```

```
      dhcp4: no
```

```
      dhcp6: no
```

```
      nameservers:
```

```
        addresses: [208.67.222.222,208.67.220.222]
```

```
    enp0s8:
```

```
      addresses: []
```

```
      #gateway4: 10.0.2.1
```

```
      dhcp4: yes
```

```
      dhcp6: no
```

```
      nameservers:
```

```
        addresses: [208.67.222.222,208.67.220.220]
```

```
root@ubuntu:/etc/netplan#
```

On the client:

```
root@ubuntu:/etc/netplan# cat 02-internal-network.yaml
```

```
network:
```

```
  renderer: networkd
```

```
  version: 2
```

```
  ethernets:
```

```
    enp0s3:
```

```
      addresses: [10.0.2.2/24]
```

```
      gateway4: 10.0.2.1
```

```
      dhcp4: no
```

```
      dhcp6: no
```

```
      nameservers:
```

```
        addresses: [208.67.222.222,208.67.220.220]
```

```
root@ubuntu:/etc/netplan#
```

To enable the NetPlan configuration, it is necessary to issue the command “`sudo netplan apply`”.

On VirtualBox, the two internal networks were configured as it follows: via the Preferences menu, select “Network”.

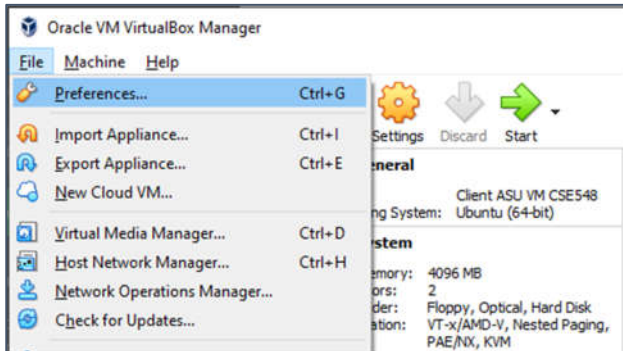


Figure 2 - VirtualBox Preferences

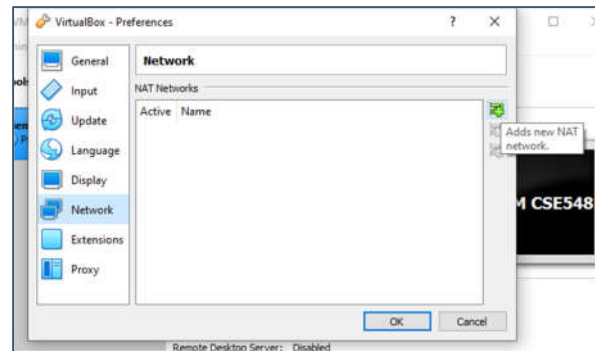


Figure 3 - VirtualBox Networks Configuration

Add a network with the “+” plus button, and configure the NatNetwork as following, then repeat for the InternetNetwork. Please notice that DHCP is disabled for the NatNetwork (the internal).

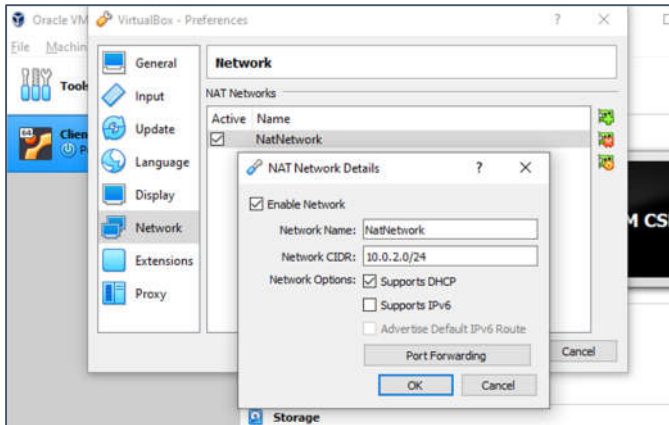


Figure 4 - InternetNetwork configuration

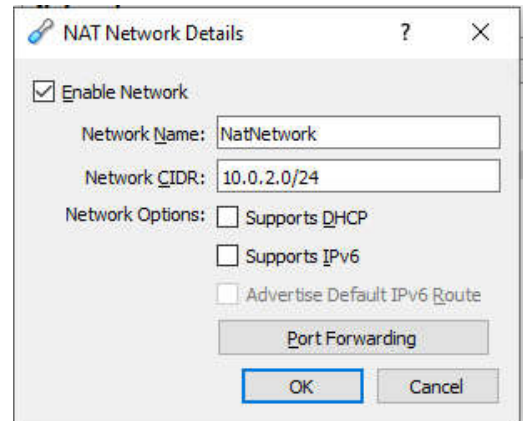


Figure 5 - NatNetwork Configuration

On the VM’s configuration, assign the interfaces on the correct networks. The followings are for the Gateway:

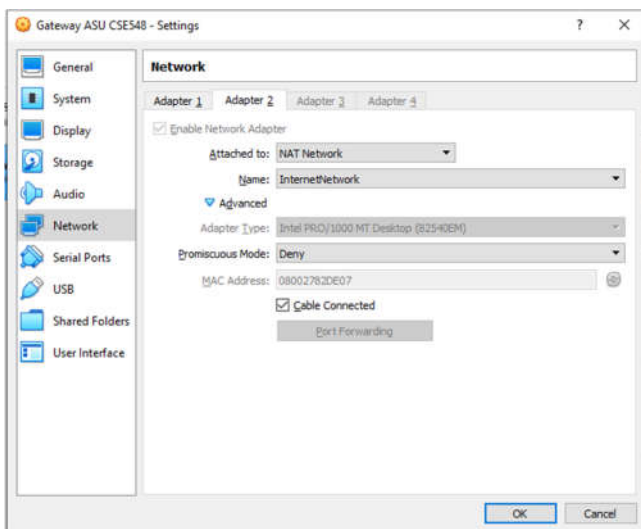


Figure 7 - Gateway’s Adapter 2 set up on VirtualBox

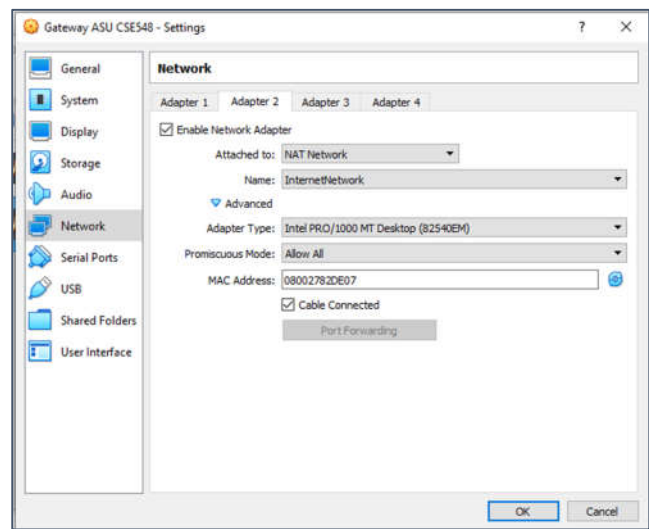


Figure 6 - Gateway’s Adapter 1 set up on VirtualBox

This is for the Client:

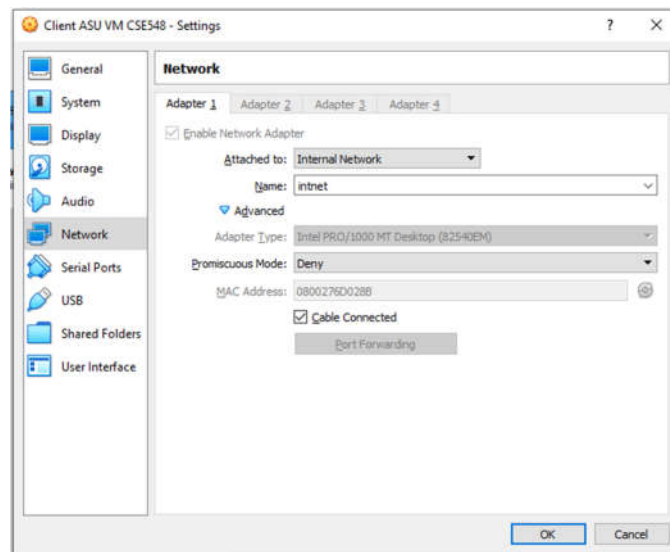


Figure 8 - Client's Adapter Configuration

On both systems, the command “**sudo service networking restart**” must be issued to ensure new network configurations are enabled.

At the end of the day, the following configurations should result when using Linux’s command line:

```
root@ubuntu: ~
File Edit View Search Terminal Help
root@ubuntu:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    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 group default qlen 1000
    link/ether 08:00:27:6d:02:8b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.2/24 brd 10.0.2.255 scope global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::6154:b3a2:188f:2e65/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
root@ubuntu:~#
```

Figure 10 - Client VM's interface as it appears at the Linux console

```
root@ubuntu:~# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.1 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::bb19:9ce2:253b:7920 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:6d:f6:f9 txqueuelen 1000 (Ethernet)
    RX packets 3643 bytes 246276 (246.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6408 bytes 10307763 (10.3 MB)
    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::b93e:c713:21d2:5726 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:82:de:07 txqueuelen 1000 (Ethernet)
    RX packets 13486 bytes 19306185 (19.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7904 bytes 547175 (547.1 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 162 bytes 14545 (14.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 162 bytes 14545 (14.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@ubuntu:~#
```

Figure 9 - Gateway VM's interfaces as they appear at the Linux console

At the very beginning, there is nothing configured on the Gateway to forward traffic:

```
ubuntu@ubuntu:~$ cat /proc/sys/net/ipv4/ip_forward
0
ubuntu@ubuntu:~$ sudo iptables -L -v -n
[sudo] password for ubuntu:
Chain INPUT (policy ACCEPT 3 packets, 744 bytes)
  pkts bytes target     prot opt in     out     source               destination

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

Chain OUTPUT (policy ACCEPT 3 packets, 484 bytes)
  pkts bytes target     prot opt in     out     source               destination
ubuntu@ubuntu:~$
```

Figure 11 - Gateway VM's initial forward routing and iptable rules

Therefore, the Client VM is completely isolated in the InternalNetwork, and can only ping the Gateway's VM interface.

```
ubuntu@ubuntu:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5104ms

ubuntu@ubuntu:~$ nslookup google.com
^C

ubuntu@ubuntu:~$ ping 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=64 time=0.354 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=64 time=0.414 ms
^C
--- 10.0.2.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1044ms
rtt min/avg/max/mdev = 0.354/0.384/0.414/0.030 ms
ubuntu@ubuntu:~$
```

Figure 12 - Client's VM initial network situation

This is the client's initial routing table:

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

Figure 13 - Client's routing table

Conversely, the Gateway has no problems doing name resolution and reaching the internet by its default route:

```
ubuntu@ubuntu:~$ cat /proc/sys/net/ipv4/ip_forward
0
ubuntu@ubuntu:~$ sudo iptables -L -v -n
[sudo] password for ubuntu:
Chain INPUT (policy ACCEPT 3 packets, 744 bytes)
pkts bytes target prot opt in out source destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 3 packets, 484 bytes)
pkts bytes target prot opt in out source destination
ubuntu@ubuntu:~$ ping cisco.com
PING cisco.com (72.163.4.185) 56(84) bytes of data.
64 bytes from cisco.com (72.163.4.185): icmp_seq=1 ttl=239 time=147 ms
64 bytes from cisco.com (72.163.4.185): icmp_seq=2 ttl=239 time=145 ms
64 bytes from cisco.com (72.163.4.185): icmp_seq=3 ttl=239 time=148 ms
^C
--- cisco.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 145.935/146.994/148.008/0.955 ms
ubuntu@ubuntu:~$
```

Figure 14 - Initial Gateway internet connection

This is gateways' initial routing table.

```
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 100 0 0 enp0s8
10.0.1.0 0.0.0.0 255.255.255.0 U 0 0 0 enp0s8
10.0.1.1 0.0.0.0 255.255.255.255 UH 100 0 0 enp0s8
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 enp0s3
ubuntu@ubuntu:~$
```

Figure 15 - Gateway's routing table

III. SOFTWARE

For this first lab, the following software has been used:

- Various network tools (specifically, tcpdump, ping, traceroute)
- Linux's built in packet forwarding router
- Linux's built-in packet inspection and NAT firewall, iptables
- Apache web server

IV. PROJECT DESCRIPTION

The following tasks have been executed to obtain the required results for the lab, and they can be repeated step-by-step by pasting the commands listed below, provided that the initial configuration, as described in the "Network Setup" chapter, is reproduced exactly.

A. Initial setup and connectivity testing

Please set up the VMs as described in the "Network Setup" page. The result must be so that the gateway can reach the internet, but the client cannot.

B. Check the network setup on the Gateway/Server VM

From a terminal window on the gateway, please run the commands **in bold** to demonstrate internet reachability (please note that ping would run indefinitely, and you have to interrupt it with CTRL+C); the part *in italic* is the result that should be obtained:

```
ubuntu@ubuntu:~$ ping cisco.com
PING cisco.com (72.163.4.185) 56(84) bytes of data.
64 bytes from cisco.com (72.163.4.185): icmp_seq=1 ttl=239 time=147 ms
64 bytes from cisco.com (72.163.4.185): icmp_seq=2 ttl=239 time=156 ms
^C
--- cisco.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 144.485/149.644/156.627/5.141 ms
ubuntu@ubuntu:~$ dig google.com

; <<>> DiG 9.11.3-1ubuntu1.15-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16376
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.      INA

;; ANSWER SECTION:
google.com.      108 INA 216.58.206.78

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sat May 29 15:24:17 MST 2021
;; MSG SIZE rcvd: 55

ubuntu@ubuntu:~$
```

Please note that response times and latency will vary.

```
ubuntu@ubuntu:~$ ping cisco.com
PING cisco.com (72.163.4.185) 56(84) bytes of data.
64 bytes from cisco.com (72.163.4.185): icmp_seq=1 ttl=239 time=147 ms
64 bytes from cisco.com (72.163.4.185): icmp_seq=2 ttl=239 time=156 ms
64 bytes from cisco.com (72.163.4.185): icmp_seq=3 ttl=239 time=144 ms
^C
--- cisco.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 144.485/149.644/156.627/5.141 ms
ubuntu@ubuntu:~$ dig google.com

; <<>> DiG 9.11.3-1ubuntu1.15-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16376
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                IN      A

;; ANSWER SECTION:
google.com.                108     IN      A      216.58.206.78

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sat May 29 15:24:17 MST 2021
;; MSG SIZE rcvd: 55

ubuntu@ubuntu:~$
```

Figure 16 - Evidence of Gateway connectivity

After this test, verify that the Gateway has connectivity to the Client, issuing the command ping 10.0.2.2 (again, ping commands need to be manually interrupted with CTRL+C):

```
ubuntu@ubuntu:~$ ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=0.394 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=64 time=0.373 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=64 time=0.433 ms
^C
--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
rtt min/avg/max/mdev = 0.373/0.400/0.433/0.024 ms
ubuntu@ubuntu:~$
```

We have already verified on the Network Setup chapter, that there is no forwarding nor iptables rules enabled on the Gateway right now. No further demonstration is necessary, but at will, further commands can be run:

```
ubuntu@ubuntu:~$ sudo iptables -L -v -n
[sudo] password for ubuntu: 123456
Chain INPUT (policy ACCEPT 256 packets, 24309 bytes)
pkts bytes target prot opt in out source destination

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

Chain OUTPUT (policy ACCEPT 435 packets, 30539 bytes)
pkts bytes target prot opt in out source destination
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  100  0    0 enp0s8
10.0.1.0    0.0.0.0    255.255.255.0 U  0  0    0 enp0s8
10.0.1.1    0.0.0.0    255.255.255.255 UH 100  0    0 enp0s8
10.0.2.0    0.0.0.0    255.255.255.0 U  0  0    0 enp0s3
ubuntu@ubuntu:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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 group default qlen 1000
    link/ether 08:00:27:6d:f6:f9 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.1/24 brd 10.0.2.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::bb19:9ce2:253b:7920/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:82:de:07 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.5/24 brd 10.0.1.255 scope global dynamic enp0s8
        valid_lft 514sec preferred_lft 514sec
    inet6 fe80::b93e:c713:21d2:5726/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
ubuntu@ubuntu:~$

```

C. Enable Packet forwarding on Gateway

To enable Linux kernel's packet forwarding engine, it is necessary to flip a configuration parameter in a system table. This is done by one of the two commands:

```

$ sudo sysctl -w net.ipv4.ip_forward=1
or
$ echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward

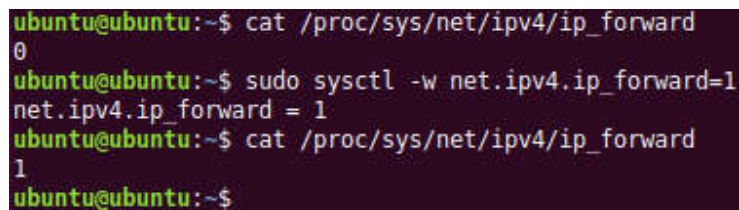
```

Using "cat" to print the value of the parameter will verify it is set correctly.

```

ubuntu@ubuntu:~$ cat /proc/sys/net/ipv4/ip_forward
1

```



```

ubuntu@ubuntu:~$ cat /proc/sys/net/ipv4/ip_forward
0
ubuntu@ubuntu:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
ubuntu@ubuntu:~$ cat /proc/sys/net/ipv4/ip_forward
1
ubuntu@ubuntu:~$

```

Figure 17 - Kernel routing on the Gateway

D. Configure default routes

Because of our set up, there is no need to configure default routes – they are automatic. We may want to test that the Client is able to reach the Gateway.

On the Client, type the followings:

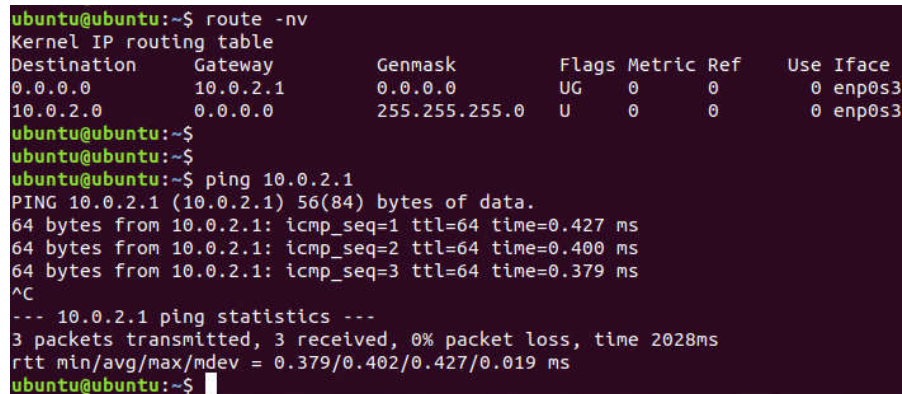
```

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

```



```
ubuntu@ubuntu:~$ ping 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=64 time=0.427 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=64 time=0.400 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=64 time=0.379 ms
^C
--- 10.0.2.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.379/0.402/0.427/0.019 ms
ubuntu@ubuntu:~$
```



```
ubuntu@ubuntu:~$ route -nv
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 10.0.2.1 0.0.0.0 UG 0 0 0 enp0s3
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 enp0s3
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ ping 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=64 time=0.427 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=64 time=0.400 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=64 time=0.379 ms
^C
--- 10.0.2.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.379/0.402/0.427/0.019 ms
ubuntu@ubuntu:~$
```

Figure 18 - Routing on the Client

However, trying to ping anything beyond the Gateway is hopeless at the moment, and it will fail.

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

(after circa four seconds...)

```
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4085ms
```

```
ubuntu@ubuntu:~$
```

This is because the Gateway does not know yet how to process those packets – we will need NAT rules with iptables to accomplish that. Anyway, at the moment, routes are working correctly.

E. Enable network traffic to Internet

To enable NAT rules on the Gateway and allow reachability of the Internet from the Client, type the followings:

```
ubuntu@ubuntu:~$ sudo -i
[sudo] password for ubuntu: 123456
root@ubuntu:~# iptables -P FORWARD ACCEPT
root@ubuntu:~# iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE
root@ubuntu:~# iptables -t nat -L POSTROUTING -n -v --line-number
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination
1 0 0 MASQUERADE all -- * enp0s8 0.0.0.0/0 0.0.0.0/0
root@ubuntu:~#
```

```

ubuntu@ubuntu:~$ sudo -i
root@ubuntu:~# iptables -P FORWARD ACCEPT
root@ubuntu:~# iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE
root@ubuntu:~# iptables -t nat -L POSTROUTING -n -v --line-number
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source         destination
1      0      0 MASQUERADE  all  --  *      enp0s8  0.0.0.0/0      0.0.0.0/0
root@ubuntu:~# █

```

Figure 19 - Lazy NAT configuration on the Gateway

You should now be able to verify on the Client that you can reach certain Internet IP addresses (e.g. 8.8.8.8) and hosts which imply name resolution (e.g. google.com, cisco.com...) via simple “ping” commands.

```

ubuntu@ubuntu:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4085ms

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=114 time=27.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=23.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=23.8 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2064ms
rtt min/avg/max/mdev = 23.493/25.073/27.925/2.028 ms

ubuntu@ubuntu:~$ ping cisco.com
PING cisco.com (72.163.4.185) 56(84) bytes of data.
64 bytes from cisco.com (72.163.4.185): icmp_seq=1 ttl=238 time=144 ms
64 bytes from cisco.com (72.163.4.185): icmp_seq=2 ttl=238 time=144 ms
64 bytes from cisco.com (72.163.4.185): icmp_seq=3 ttl=238 time=145 ms
^C
--- cisco.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 144.398/144.856/145.688/0.666 ms
ubuntu@ubuntu:~$ █

```

Figure 20 - Successfully pinging the Internet from the client

F. Install Software in Linux

Now that the Client can reach the Internet, we can upgrade the software and install further packages. Please run “**sudo apt update**” at the terminal prompt, which will refresh the package list to the latest version:

```

ubuntu@ubuntu:~$ sudo apt update
[sudo] password for ubuntu:
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [48.5 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11 Metadata [60.4 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Metadata [294 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:9 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11 Metadata [290 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-11 Metadata [2,468 B]
Get:11 http://us.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 Metadata [9,284 B]
Fetched 959 kB in 3s (367 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ubuntu:~$ █

```

Figure 21 - apt update command

Please run now “**sudo apt -y dist-upgrade**” to update the already installed packages to the current version (output may vary):

```

Unpacking libpam-modules-bin (1.1.8-3.6ubuntu2.18.04.3) over (1.1.8-3.6ubuntu2.18.04.2) ...
Setting up libpam-modules-bin (1.1.8-3.6ubuntu2.18.04.3) ...
(Reading database ... 168314 files and directories currently installed.)
Preparing to unpack .../libpam-modules_1.1.8-3.6ubuntu2.18.04.3_amd64.deb ...
Unpacking libpam-modules:amd64 (1.1.8-3.6ubuntu2.18.04.3) over (1.1.8-3.6ubuntu2.18.04.2) ...
Setting up libpam-modules:amd64 (1.1.8-3.6ubuntu2.18.04.3) ...
(Reading database ... 168318 files and directories currently installed.)
Preparing to unpack .../libpam-runtime_1.1.8-3.6ubuntu2.18.04.3_all.deb ...
Unpacking libpam-runtime (1.1.8-3.6ubuntu2.18.04.3) over (1.1.8-3.6ubuntu2.18.04.2) ...
Setting up libpam-runtime (1.1.8-3.6ubuntu2.18.04.3) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
ubuntu@ubuntu:~$

```

Figure 22 - apt dist-upgrade

Repeat the same operations on the Gateway, and in addition, install the “tcpdump” package (which will be needed soon) by typing “**sudo apt install tcpdump**”:

```

ubuntu@ubuntu:~$ sudo apt install tcpdump
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tcpdump
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 364 kB of archives.
After this operation, 1,092 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 tcpdump amd64 4.9.3-0ubuntu0.18.04.1 [364 kB]
Fetched 364 kB in 1s (314 kB/s)
Selecting previously unselected package tcpdump.
(Reading database ... 178719 files and directories currently installed.)
Preparing to unpack .../tcpdump_4.9.3-0ubuntu0.18.04.1_amd64.deb ...
Unpacking tcpdump (4.9.3-0ubuntu0.18.04.1) ...
Setting up tcpdump (4.9.3-0ubuntu0.18.04.1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
ubuntu@ubuntu:~$

```

Figure 23 - Installation of tcpdump on the Gateway

G. Install Apache2 on the Gateway

Please issue the command “**sudo apt install apache2**” on the Gateway to install apache2.

```

ubuntu@ubuntu:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  apache2
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 95.1 kB of archives.
After this operation, 536 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 apache2 amd64 2.4.29-1ubuntu4.14 [95.1 kB]
Fetched 95.1 kB in 1s (169 kB/s)
Selecting previously unselected package apache2.
(Reading database ... 178679 files and directories currently installed.)
Preparing to unpack .../apache2_2.4.29-1ubuntu4.14_amd64.deb ...
Unpacking apache2 (2.4.29-1ubuntu4.14) ...
Setting up apache2 (2.4.29-1ubuntu4.14) ...
Processing triggers for systemd (237-3ubuntu10.47) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ufw (0.36-0ubuntu0.18.04.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
ubuntu@ubuntu:~$

```

Figure 24 - Installation of Apache2 on the Gateway

We need now to be sure that the “ufw” application is stopped and disabled (will not reactivate on restart), by issuing the commands: “**sudo service ufw stop**” and “**sudo systemctl disable ufw**”.


```

ubuntu@ubuntu:~$ sudo service ufw stop
ubuntu@ubuntu:~$ sudo systemctl disable ufw
Synchronizing state of ufw.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable ufw
ubuntu@ubuntu:~$ sudo service ufw status
● ufw.service - Uncomplicated firewall
   Loaded: loaded (/lib/systemd/system/ufw.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:ufw(8)

May 29 16:19:48 ubuntu systemd[1]: Stopping Uncomplicated firewall...
May 29 16:19:48 ubuntu ufw-init[10392]: Skip stopping firewall: ufw (not enabled)
May 29 16:19:48 ubuntu systemd[1]: Stopped Uncomplicated firewall.
ubuntu@ubuntu:~$

```

Figure 25 - Disabling of ufw

H. Setup Apache2

The configuration file “/etc/apache2/sites-available/test-and-demo.conf” needs to be created and be equal to the listing below. It can be accomplished by using the “sudo” command with the favorite Linux editor, such as vim or nano.

If preferred, it can be downloaded from this link on GitHub: <https://github.com/markoer73/CSE-548/blob/main/Project%20-%20Packet%20Filter%20Firewall/test-and-demo.conf>

```

<VirtualHost *:80>
    ServerName test-and-demo.com
    ServerAlias *.test-and-demo.com

    ServerAdmin mermini@asu.edu
    DocumentRoot /var/www/test/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

```

The next configuration file to change is “/etc/apache2/ports.conf”, which needs to be set as the followings:

```

Listen 127.0.0.1:80
Listen 10.0.2.1:80

```

If preferred, the file ports.conf can be downloaded from here: <https://github.com/markoer73/CSE-548/blob/main/Project%20-%20Packet%20Filter%20Firewall/ports.conf>

While still on the Gateway, please add now the hostname “www.test-and-demo.com” on the /etc/hosts file by typing:

```

ubuntu@ubuntu:~$ sudo -i
[sudo] password for ubuntu: 123456
root@ubuntu:~# echo "127.0.0.1 www.test-and-demo.com" >> /etc/hosts
root@ubuntu:~# cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1 www.test-and-demo.com
root@ubuntu:~#

```


After this is accomplished, please enable the new Apache2 VirtualHost and restart Apache2 with the following commands:

```
ubuntu@ubuntu:~$ sudo a2ensite test-and-demo
ubuntu@ubuntu:~$ sudo systemctl restart apache2.service
```

I. Enable Reachability of Apache2 from the Gateway

At this point, on your Gateway you should be able to open a Firefox browser tab, point it to “www.test-and-demo.com” and see the default Apache2 Debian page appear.

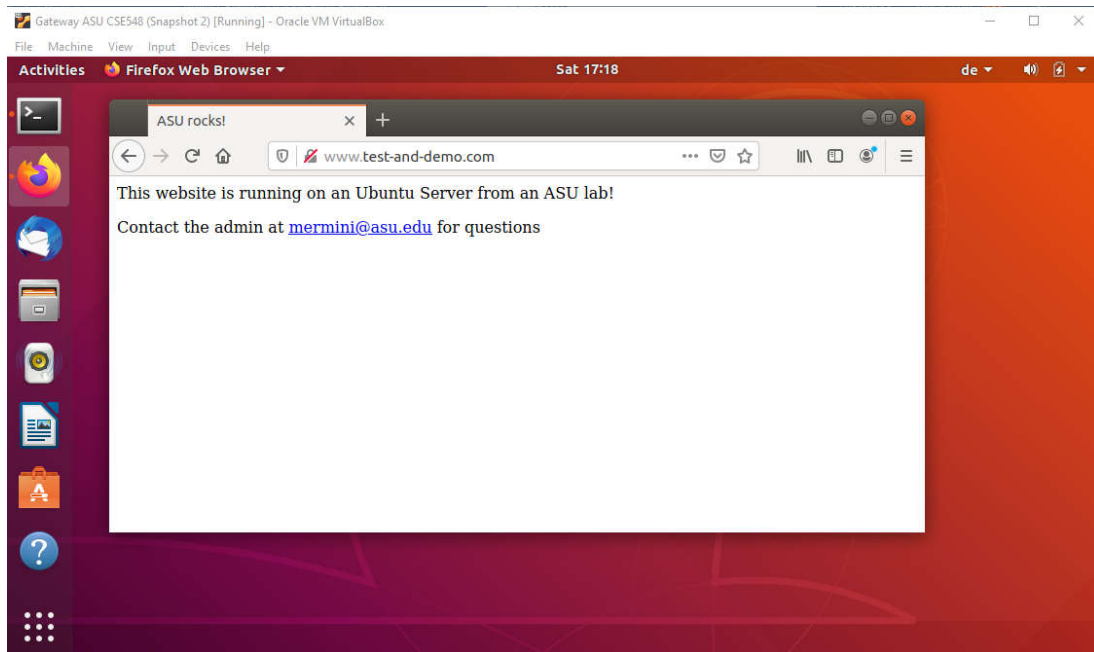


Figure 28 - Firefox browser testing out web site on the Gateway

J. Enable Reachability of Apache2 from the Client

To enable reachability from the Client, we only need to perform similar commands affecting `/etc/hosts`.

```
ubuntu@ubuntu:~$ sudo -i
[sudo] password for ubuntu: 123456
root@ubuntu:~# echo "10.0.2.1 www.test-and-demo.com" >> /etc/hosts
root@ubuntu:~# cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.0.2.1 www.test-and-demo.com
root@ubuntu:~#
```

```

root@ubuntu:~# cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.0.2.1 www.test-and-demo.com
root@ubuntu:~#

```

Figure 29 - /etc/hosts file on the Client

It will be then possible also from the Client to visualize the testing web page on the Gateway web server, by just opening Firefox and pointing it towards “www.test-and-demo.com”.

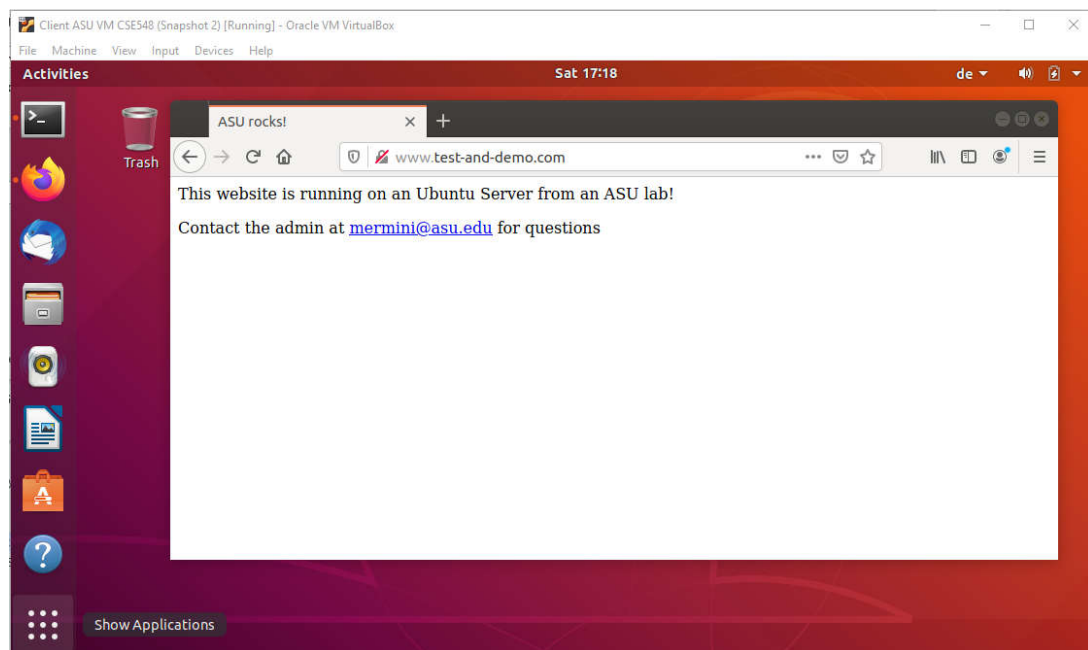


Figure 30 - Firefox browser testing out web site from the Client

K. Set up of the Packet filter firewall

Setting up the stateless packet filter packet firewall requires running a script, which is provided here in the Appendix A or in GitHub at the address: <https://github.com/markoer73/CSE-548/blob/main/Project%201%20-%20Packet%20Filter%20Firewall/rc.sh>

We need to use the file rc.sh as the “root” user, therefore, if we are not it, please run “**sudo -i**”. After that, the file needs to be made executable with the command “**chmod +x rc.sh**”, or in alternative it can be run in this way: “**./rc.sh**”. In any case, it should run displaying verbose output of iptables rules being set, but with no errors.

```

+ /sbin/iptables -A INPUT -p ICMP --icmp-type echo-request -i lo -d 127.0.0.1 -j ACCEPT
+ /sbin/iptables -A INPUT -p ICMP --icmp-type echo-request -i lo -d 127.0.0.1 -j ACCEPT
+ /sbin/iptables -A INPUT -p ICMP --icmp-type echo-reply -i lo -d 127.0.0.1 -j ACCEPT
+ /sbin/iptables -A INPUT -p UDP -i enp0s8 --sport 53 -s 208.67.222.222 -j ACCEPT
+ /sbin/iptables -A INPUT -p UDP -i enp0s8 --sport 53 -s 208.67.220.220 -j ACCEPT
+ /sbin/iptables -A INPUT -p UDP --dport 53 -d 208.67.222.222 -j ACCEPT
+ /sbin/iptables -A INPUT -p UDP --dport 53 -d 208.67.220.220 -j ACCEPT
+ /sbin/iptables -A INPUT -j LOG --log-prefix DROPPED-INGRESS-
+ /sbin/iptables -A OUTPUT -o lo -j ACCEPT
+ /sbin/iptables -A OUTPUT -o lo -s 127.0.0.0/8 -j ACCEPT
+ /sbin/iptables -A OUTPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
+ /sbin/iptables -A OUTPUT -p UDP --sport 53 -d 127.0.0.53 -j ACCEPT
+ /sbin/iptables -A OUTPUT -p UDP --sport 53 -d 208.67.222.222 -j ACCEPT
+ /sbin/iptables -A OUTPUT -p UDP --sport 53 -d 208.67.220.220 -j ACCEPT
+ /sbin/iptables -A OUTPUT -p UDP --dport 53 -d 208.67.222.222 -j ACCEPT
+ /sbin/iptables -A OUTPUT -p UDP --dport 53 -d 208.67.220.220 -j ACCEPT
+ /sbin/iptables -A OUTPUT -p TCP --sport 80 -o enp0s3 -j ACCEPT
+ /sbin/iptables -A OUTPUT -p TCP --sport 80 -o lo -j ACCEPT
+ /sbin/iptables -A OUTPUT -p ICMP --icmp-type echo-reply -j ACCEPT
+ /sbin/iptables -A OUTPUT -p ICMP --icmp-type echo-request -j ACCEPT
+ '[' Y == Y ']'
+ /sbin/iptables -A OUTPUT -p TCP --dport 80 -o enp0s8 -j ACCEPT
+ /sbin/iptables -A OUTPUT -p TCP --dport 443 -o enp0s8 -j ACCEPT
+ /sbin/iptables -A OUTPUT -j LOG --log-prefix DROPPED-EGRESS-
+ /sbin/iptables -t nat -A POSTROUTING -p icmp -o enp0s8 -d 8.8.8.8 -j MASQUERADE
+ /sbin/iptables -t nat -A POSTROUTING -p icmp -o enp0s8 -d 208.67.222.222 -j MASQUERADE
+ /sbin/iptables -t nat -A POSTROUTING -p icmp -o enp0s8 -d 208.67.220.220 -j MASQUERADE
+ /sbin/iptables -t nat -A POSTROUTING -p udp -o enp0s8 -d 208.67.222.222 -j MASQUERADE
+ /sbin/iptables -t nat -A POSTROUTING -p udp -o enp0s8 -d 208.67.220.220 -j MASQUERADE
+ '[' Y == Y ']'
+ /sbin/iptables -t nat -A POSTROUTING -p tcp -o enp0s8 --dport 80 -j MASQUERADE
+ /sbin/iptables -t nat -A POSTROUTING -p tcp -o enp0s8 --dport 443 -j MASQUERADE
+ /sbin/iptables -t nat -A POSTROUTING -j LOG --log-prefix NO-MASQUERADE-MATCH-
root@ubuntu:~#

```

Figure 31 - One run of the rc.sh script

After the script is run, the Gateway will be configured to filter certain ports, blocking others, and propagating others via NAT.

This is the default configuration set up in the script:

Source	Destination	Protocol	Rule	Comment
Client	Gateway	TCP/80	Allowed	Reach the Web Server on the Gateway
Client	Internet	TCP/80	Allowed	Reach Web for updates
Client	Internet	TCP/443	Allowed	Reach Web for updates
Client	DNS servers	ICMP	Allowed	Ping DNS servers
Client	DNS servers	DNS	Allowed	Allow DNS resolution
Client	Gateway	ICMP	Allowed	Ping the Gateway
Client	8.8.8.8	ICMP	Allowed	Ping Google DNS
Client	*	*	Deny	Deny the rest
Gateway	Loopback	*	Allowed	Allow loopback
Gateway	Client	ICMP	Allowed	Ping the Client
Gateway	Internet	TCP/80	Allowed	Reach Web for updates
Gateway	Internet	TCP/443	Allowed	Reach Web for updates
Gateway	DNS servers	ICMP	Allowed	Ping DNS servers
Gateway	DNS servers	DNS	Allowed	Allow DNS resolution
Gateway	8.8.8.8	ICMP	Allowed	Ping Google DNS
Gateway	*	*	Deny	Deny the rest

The scripts also set up NAT for the allowed protocols for the Client.

As we can see from the screenshot, the client can ping the DNS servers, but although it can resolve the name cisco.com via DNS, it cannot ping it. It can anyway browse the web, while also being still able to reach the web server on the Gateway.

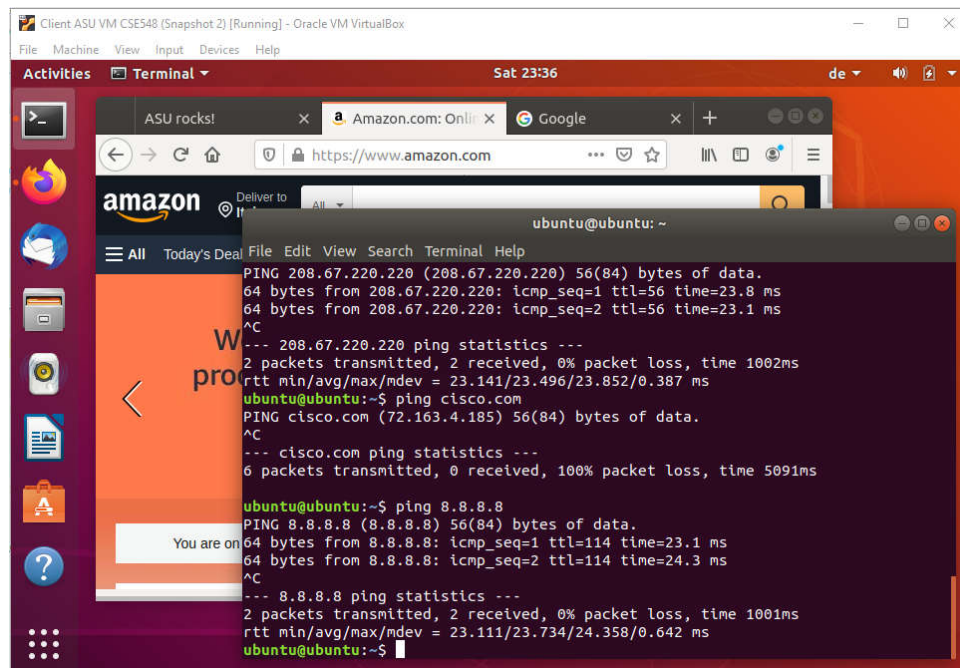


Figure 32 - Testing connectivity from the Client

L. Sniffing traffic on the Gateway

One interesting aspect of this configuration is that the Gateway is in the position to “sniff” all the traffic of the Client.

Please run the command “`sudo tcpdump -i enp0s3`” on a Terminal on the Gateway, and then move to the Client and execute an allowed network command – for instance, ping 8.8.8.8.

```

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=114 time=29.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=23.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=29.4 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 23.644/27.554/29.600/2.769 ms
ubuntu@ubuntu:~$

```

Figure 33 - Pinging Google DNS from the Client

All the traffic generated is captured in the tcpdump session on the Gateway, as visible.


```

root@ubuntu:~# tcpdump -i enp0s3
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, Link-type EN10MB (Ethernet), capture size 262144 bytes
23:39:46.131261 IP 10.0.2.2.34533 > resolver1.opendns.com.domain: 32219+ A? endpoint.prod.us-east-1.forester.a2z.com. (58)
23:39:46.132280 IP 10.0.2.2.42072 > ec2-34-195-123-154.compute-1.amazonaws.com.https: Flags [P.], seq 412749081:412749208, ack 25162849,
win 62780, length 127
23:39:46.132280 IP 10.0.2.2.42072 > ec2-34-195-123-154.compute-1.amazonaws.com.https: Flags [P.], seq 127:537, ack 1, win 62780, length
410
23:39:46.134526 IP ec2-34-195-123-154.compute-1.amazonaws.com.https > 10.0.2.2.42072: Flags [P.], ack 537, win 32768, length 0
23:39:46.134581 IP 10.0.2.2.46235 > resolver1.opendns.com.domain: 27829+ AAAA? endpoint.prod.us-east-1.forester.a2z.com. (58)
23:39:46.156885 IP resolver1.opendns.com.domain > 10.0.2.2.34533: 32219 8/0/0 A 54.81.131.78, A 3.214.160.144, A 3.215.241.85, A 3.218.1
79.45, A 35.153.96.52, A 52.5.47.81, A 52.55.38.43, A 52.207.6.131 (200)
23:39:46.157234 IP resolver1.opendns.com.domain > 10.0.2.2.46235: 27829 0/1/0 (156)
23:39:46.252497 IP ec2-34-195-123-154.compute-1.amazonaws.com.https > 10.0.2.2.42072: Flags [P.], seq 1:195, ack 537, win 32768, length
194
23:39:48.903998 IP 10.0.2.2 > dns.google: ICMP echo request, id 2787, seq 1, length 64
23:39:48.932760 IP dns.google > 10.0.2.2: ICMP echo reply, id 2787, seq 1, length 64
23:39:49.905928 IP 10.0.2.2 > dns.google: ICMP echo request, id 2787, seq 2, length 64
23:39:49.929061 IP dns.google > 10.0.2.2: ICMP echo reply, id 2787, seq 2, length 64
23:39:50.907310 IP 10.0.2.2 > dns.google: ICMP echo request, id 2787, seq 3, length 64
23:39:50.936318 IP dns.google > 10.0.2.2: ICMP echo reply, id 2787, seq 3, length 64
23:39:56.133198 IP 10.0.2.2.42072 > ec2-34-195-123-154.compute-1.amazonaws.com.https: Flags [P.], seq 537:669, ack 195, win 62780, leng
th 132
23:39:56.133198 IP 10.0.2.2.42072 > ec2-34-195-123-154.compute-1.amazonaws.com.https: Flags [P.], seq 669:2019, ack 195, win 62780, leng
th 1350
23:39:56.133909 IP ec2-34-195-123-154.compute-1.amazonaws.com.https > 10.0.2.2.42072: Flags [P.], ack 2019, win 32768, length 0
23:39:56.134002 IP 10.0.2.2.42072 > ec2-34-195-123-154.compute-1.amazonaws.com.https: Flags [P.], seq 2019:3545, ack 195, win 62780, len
gth 1526
23:39:56.135013 IP ec2-34-195-123-154.compute-1.amazonaws.com.https > 10.0.2.2.42072: Flags [P.], ack 3545, win 32768, length 0
23:39:56.258050 IP ec2-34-195-123-154.compute-1.amazonaws.com.https > 10.0.2.2.42072: Flags [P.], seq 195:389, ack 3545, win 32768, leng
th 194
23:39:56.258655 IP 10.0.2.2.42072 > ec2-34-195-123-154.compute-1.amazonaws.com.https: Flags [P.], ack 389, win 62780, length 0
^C
21 packets captured
22 packets received by filter
1 packet dropped by kernel

```

Figure 34 - tcpdump run from the Gateway

Using a software with a graphical interface such as Wireshark may provide a more rewarding user experience.

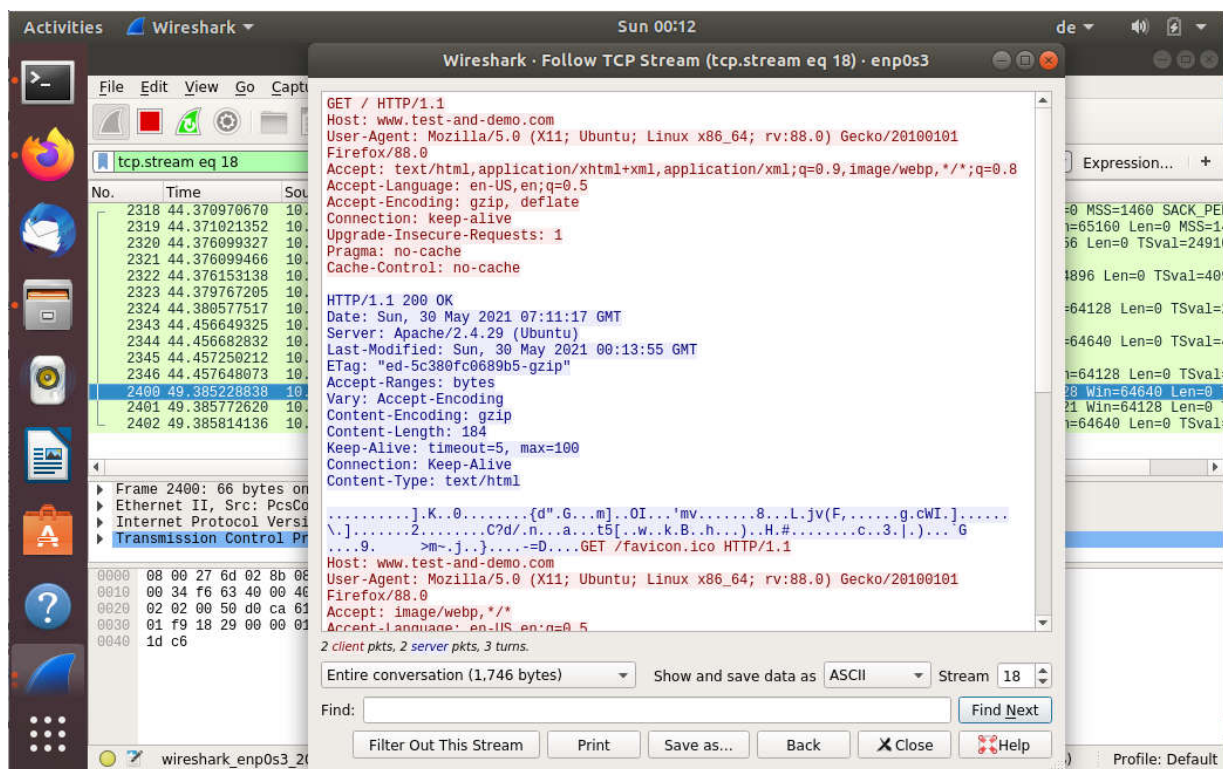


Figure 35 - Running Wireshark on the Gateway

M. Changing Allowed Protocols on the Gateway

On the Gateway we have the possibility to decide which protocols are allowed through for the clients, and what is NATted and what not.

A simple way to test it is to edit the rc.sh firewall script and change two values: at the lines 45 and 48, change the values from “Y” to any other value (e.g. “N”).

Change from:

```
# Allow web access to the Client (for Ubuntu updates)
export Client_Allowed_Web="Y"
```

```
# Allow web access to the Gateway (for Ubuntu updates)
export GW_Allowed_Web="Y"
```

To:

```
# Allow web access to the Client (for Ubuntu updates)
export Client_Allowed_Web="N"
```

```
# Allow web access to the Gateway (for Ubuntu updates)
export GW_Allowed_Web="N"
```

After the rc.sh file is saved and it is run again, the client and the gateway will lose the ability to browse the web, while all the other services – name resolution, ping, etc. – will keep working as usual (including the test web page hosted on Apache2 on the gateway).

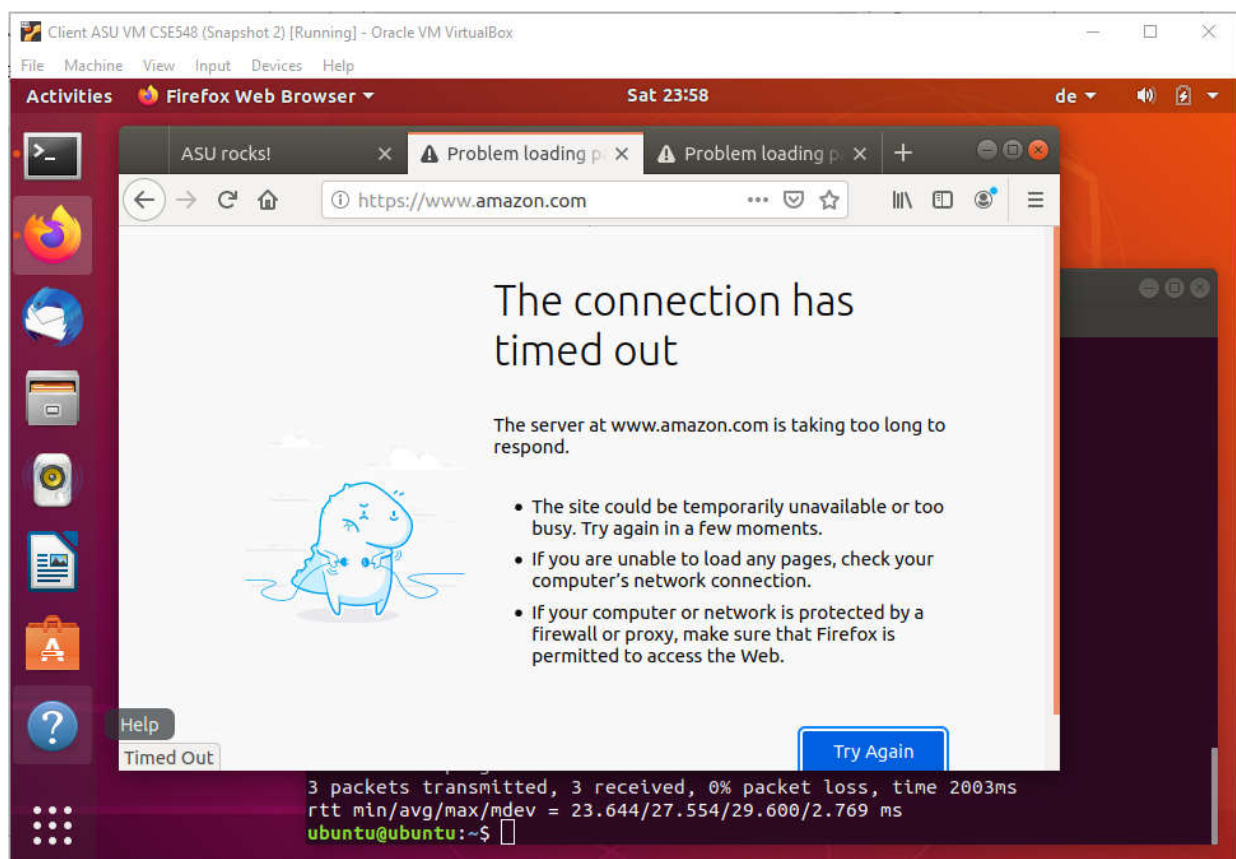


Figure 36 - Internet web sites timing out on the Client

It is worth noticing that the web sites will timeout after circa one minute from when the page is being requested. This is because on the iptables rules on the Gateway we have not specified a “reject” or sent a TCP Reset back to the client – we have simply dropped the packet and ignored it. The browser and the client operative system do not know if the Gateway is not allowing the traffic, or simply the Internet is very slow, or the destination web site is unreachable for any other reason. If desired, protocols can be explicitly rejected by setting the appropriate configuration on iptables.

V. CONCLUSION

The following are the most interesting tricks I have learned:

1. By choosing to run the internal network without DHCP and assigning manually the addressed on the internal network, we have maximized the control and we are operating like we are placed in a dual-homed host architecture. It would be possible to enable the IPS system with Snort, for instance.
2. NetPlan and systemd don't like each other's. If you typed the wrong IP for the DNS server in NetPlan, after you correct it, it won't be refreshed. You need to reboot your Linux box to fix this issue!
3. You can put logging rules at the end of every iptables chains to record a log when the pattern matching is not happening. In this way you can troubleshoot your iptables mistakes (this is what allowed me to figure out I was still using the wrong IP address for the DNS servers).

One of these rules can be like the following:

```
$IPTABLES -A INPUT -j LOG --log-prefix DROPPED-INGRESS-
```

This rule should be placed as the last of the INPUT chain. While troubleshooting, it is possible to open the /var/log/messages file with “**tail -f /var/log/syslog**”:

```
root@ubuntu:~# tail -f /var/log/syslog
May 30 00:52:33 ubuntu kernel: [15214.599148] SKIPPED-FORWARD-IN=enp0s3 OUT=enp0s8 MAC=08:00:27:6d:f6:f9:08:00:27:6d:02:8b:08:00 SRC=10.0.2.2 DST=208.67.220.220 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=60360 DF PROTO=TCP SPT=52588 DPT=53 WINDOW=64240 RES=0x00 SYN URG=0
May 30 00:52:34 ubuntu kernel: [15215.601052] SKIPPED-FORWARD-IN=enp0s3 OUT=enp0s8 MAC=08:00:27:6d:f6:f9:08:00:27:6d:02:8b:08:00 SRC=10.0.2.2 DST=208.67.220.220 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=60361 DF PROTO=TCP SPT=52588 DPT=53 WINDOW=64240 RES=0x00 SYN URG=0
May 30 00:52:35 ubuntu gnome-software[1930]: no app for changed ubuntu-dock@ubuntu.com
May 30 00:52:35 ubuntu gnome-software[1930]: no app for changed ubuntu-appindicators@ubuntu.com
May 30 00:52:35 ubuntu gvfsd-metadata[4434]: g_udev_device_has_property: assertion 'G_UDEV_IS_DEVICE (device)' failed
May 30 00:52:35 ubuntu gvfsd-metadata[4434]: g_udev_device_has_property: assertion 'G_UDEV_IS_DEVICE (device)' failed
May 30 00:52:35 ubuntu gnome-shell[1536]: [AppIndicatorSupport-DEBUG] Registering StatusNotifierItem :1.70/org/ayatana/NotificationItem/software.update.available
May 30 00:52:35 ubuntu gnome-shell[1536]: [AppIndicatorSupport-DEBUG] Registering StatusNotifierItem :1.70/org/ayatana/NotificationItem/livepatch
May 30 00:52:36 ubuntu kernel: [15217.617201] SKIPPED-FORWARD-IN=enp0s3 OUT=enp0s8 MAC=08:00:27:6d:f6:f9:08:00:27:6d:02:8b:08:00 SRC=10.0.2.2 DST=208.67.220.220 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=60362 DF PROTO=TCP SPT=52588 DPT=53 WINDOW=64240 RES=0x00 SYN URG=0
May 30 00:52:40 ubuntu kernel: [15221.680968] SKIPPED-FORWARD-IN=enp0s3 OUT=enp0s8 MAC=08:00:27:6d:f6:f9:08:00:27:6d:02:8b:08:00 SRC=10.0.2.2 DST=208.67.220.220 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=60363 DF PROTO=TCP SPT=52588 DPT=53 WINDOW=64240 RES=0x00 SYN URG=0
May 30 00:52:44 ubuntu kernel: [15225.849063] SKIPPED-FORWARD-IN=enp0s3 OUT=enp0s8 MAC=08:00:27:6d:f6:f9:08:00:27:6d:02:8b:08:00 SRC=10.0.2.2 DST=208.67.220.220 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=50494 DF PROTO=TCP SPT=52590 DPT=53 WINDOW=64240 RES=0x00 SYN URG=0
May 30 00:52:45 ubuntu kernel: [15226.865086] SKIPPED-FORWARD-IN=enp0s3 OUT=enp0s8 MAC=08:00:27:6d:f6:f9:08:00:27:6d:02:8b:08:00 SRC=10.0.2.2 DST=208.67.220.220 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=50495 DF PROTO=TCP SPT=52590 DPT=53 WINDOW=64240 RES=0x00 SYN URG=0
May 30 00:52:47 ubuntu kernel: [15228.880652] SKIPPED-FORWARD-IN=enp0s3 OUT=enp0s8 MAC=08:00:27:6d:f6:f9:08:00:27:6d:02:8b:08:00 SRC=10.0.2.2 DST=208.67.220.220 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=50496 DF PROTO=TCP SPT=52590 DPT=53 WINDOW=64240 RES=0x00 SYN URG=0
May 30 00:52:51 ubuntu kernel: [15232.944488] DROPPED-INGRESS-IN=enp0s8 OUT= MAC=08:00:27:82:de:07:08:00:27:02:45:bc:08:00 SRC=10.0.1.3 DST=10.0.1.5 LEN=576 TOS=0x00 PREC=0x00 TTL=255 ID=104 PROTO=UDP SPT=67 DPT=68 LEN=556
May 30 00:52:51 ubuntu systemd-networkd[333]: enp0s8: Configured
May 30 00:52:51 ubuntu kernel: [15232.944488] SKIPPED-FORWARD-IN=enp0s3 OUT=enp0s8 MAC=08:00:27:6d:f6:f9:08:00:27:6d:02:8b:08:00 SRC=10.0.2.2 DST=208.67.220.220 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=50497 DF PROTO=TCP SPT=52590 DPT=53 WINDOW=64240 RES=0x00 SYN URG=0
```

Figure 37 - Example of monitoring for rule drops via SYSLOG.

VI. APPENDIX A: FILES FOR THE LAB

Please find the list of files created for this lab and mentioned throughout this document, plus their GitHub link for download.

rc.sh	https://github.com/markoer73/CSE-548/blob/main/Project%201%20-%20Packet%20Filter%20Firewall/rc.sh
index.html	https://github.com/markoer73/CSE-548/blob/main/Project%201%20-%20Packet%20Filter%20Firewall/index.html
test-and-demo.conf	https://github.com/markoer73/CSE-548/blob/main/Project%201%20-%20Packet%20Filter%20Firewall/test-and-demo.conf
ports.conf	https://github.com/markoer73/CSE-548/blob/main/Project%201%20-%20Packet%20Filter%20Firewall/ports.conf
02-gateway-networks.yaml	https://github.com/markoer73/CSE-548/blob/main/Project%201%20-%20Packet%20Filter%20Firewall/02-gateway-networks.yaml
02-internal-network.yaml	https://github.com/markoer73/CSE-548/blob/main/Project%201%20-%20Packet%20Filter%20Firewall/02-internal-network.yaml

VII. REFERENCES

- Linux NAT Tutorial: https://www.karlrupp.net/en/computer/nat_tutorial
- Ubuntu “Basic Iptables HOWTO”: <https://help.ubuntu.com/community/IptablesHowTo>
- “Iptables Tutorial: Ultimate Guide to Linux Firewall”: <https://phoenixnap.com/kb/iptables-tutorial-linux-firewall>

VIII. TABLE OF FIGURES

Figure 1 - Network Map of the Lab.....	1
Figure 2 - VirtualBox Preferences.....	3
Figure 3 - VirtualBox Networks Configuration.....	3
Figure 4 - InternetNetwork configuration.....	3
Figure 5 - NatNetwork Configuration	3
Figure 6 - Gateway’s Adapter 1 set up on VirtualBox	3
Figure 7 - Gateway’s Adapter 2 set up on VirtualBox	3
Figure 8 - Client's Adapter Configuration	4
Figure 9 - Gateway VM's interfaces as they appear at the Linux console	4
Figure 10 - Client VM's interface as it appears at the Linux console	4
Figure 11 - Gateway VM's initial forward routing and iptable rules	4
Figure 12 - Client's VM initial network situation.....	5
Figure 13 - Client's routing table	5
Figure 14 - Initial Gateway internet connection	5
Figure 15 - Gateway's routing table.....	5
Figure 16 - Evidence of Gateway connectivity.....	7
Figure 17 - Kernel routing on the Gateway	8
Figure 18 - Routing on the Client.....	9
Figure 19 - Lazy NAT configuration on the Gateway.....	10
Figure 20 - Successfully ping the Internet from the client.....	10
Figure 21 - apt update command	10
Figure 22 - apt dist-upgrade.....	11
Figure 23 - Installation of tcpdump on the Gateway	11
Figure 24 - Installation of Apache2 on the Gateway	11
Figure 25 - Disabling of ufw	12
Figure 26 - Adding the host name to /etc/hosts	13
Figure 27 - the index.html of our test web site	13
Figure 28 - Firefox browser testing out web site on the Gateway	14
Figure 29 - /etc/hosts file on the Client.....	15
Figure 30 - Firefox browser testing out web site from the Client.....	15
Figure 31 - One run of the rc.sh script.....	16
Figure 32 - Testing connectivity from the Client.....	17
Figure 33 - Pinging Google DNS from the Client	17
Figure 34 - tcpdump run from the Gateway.....	18
Figure 35 - Running Wireshark on the Gateway	18
Figure 36 - Internet web sites timing out on the Client	19
Figure 37 - Example of monitoring for rule drops via SYSLOG.	20