Building and Testing a Firewall using iptables

Alaa

March 7, 2020

1 Preparation

Local network lab has been built to implement the Firewall policies with the `iptables`; the following figure demonstrates the lab environment.

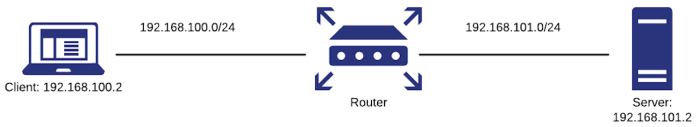
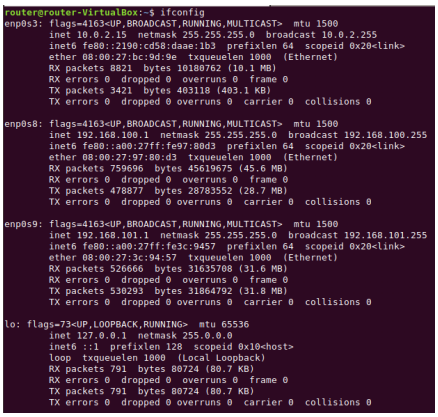


Figure 1: Lab Configuration

The network configurations in each machine (Router, Server and Client) is represented in the forthcoming figures.



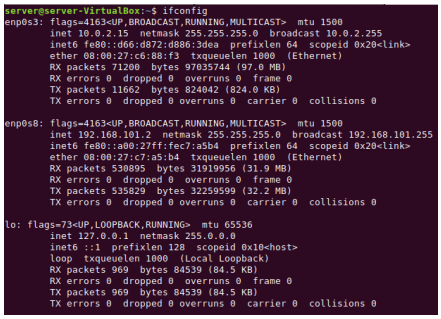
```
router@router-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::2190:cd58:daae:1b3  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:bc:9d:9e  txqueuelen 1000  (Ethernet)
        RX packets 8821  bytes 10180762 (10.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3421  bytes 403118 (403.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.1  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::a00:27ff:fe97:80d3  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:97:80:d3  txqueuelen 1000  (Ethernet)
        RX packets 759696  bytes 45619675 (45.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 478877  bytes 28783552 (28.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s9: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.101.1  netmask 255.255.255.0  broadcast 192.168.101.255
        inet6 fe80::a00:27ff:fe3c:9457  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:3c:94:57  txqueuelen 1000  (Ethernet)
        RX packets 526666  bytes 31635700 (31.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 530293  bytes 31864792 (31.8 MB)
        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 791  bytes 80724 (80.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 791  bytes 80724 (80.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 2: `ifconfig` for Router

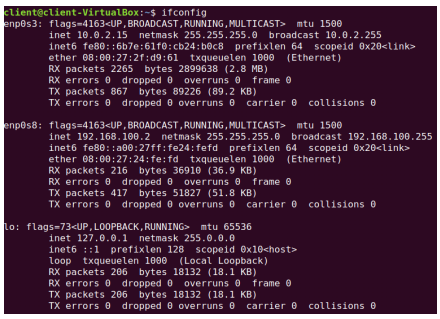


```
server@server-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::d66:d872:d886:3dea  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c6:88:f3  txqueuelen 1000  (Ethernet)
        RX packets 71200  bytes 97035744 (97.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11662  bytes 824042 (824.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.101.2  netmask 255.255.255.0  broadcast 192.168.101.255
        inet6 fe80::a00:27ff:fec7:a5b4  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c7:a5:b4  txqueuelen 1000  (Ethernet)
        RX packets 530895  bytes 31919956 (31.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 535829  bytes 32259599 (32.2 MB)
        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 969  bytes 84539 (84.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 969  bytes 84539 (84.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3: `ifconfig` for Server



```
client@client-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::6b7e:61f0:cb24:b0c8  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:2f:d9:61  txqueuelen 1000  (Ethernet)
        RX packets 2265  bytes 2899638 (2.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 867  bytes 89226 (89.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.2  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::a00:27ff:fe24:fefd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:24:fe:fd  txqueuelen 1000  (Ethernet)
        RX packets 216  bytes 36910 (36.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 417  bytes 51827 (51.8 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 206  bytes 18132 (18.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 206  bytes 18132 (18.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4: `ifconfig` for Client

2 Default Permit - host firewalling

The below bash script tries to prevent the access to **port:80** on the server. However, the other services on the server are still accessable.

Listing 1: **Part2.sh**

```
#!/bin/bash
iptablesPATH="/sbin/iptables"
iptablesSavePATH="/sbin/iptables-save"
# flush all chains
$iptablesPATH -F
# block Access to Port 80
$iptablesPATH -A INPUT -p tcp --dport 80 -j REJECT
# Save Settings
$iptablesSavePATH
```

Before applying the policy , the client was able to access the **http service** on the server.

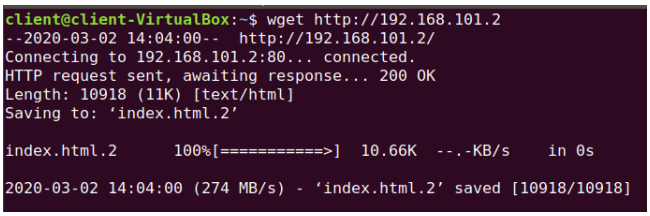


Figure 5: Before the policy: Accessing **port:80**

The `INPUT` chain has been used since we want to implement the firewall rule on the server side. After running the script, the rule has been added to the `iptables's INPUT` chain as follows:

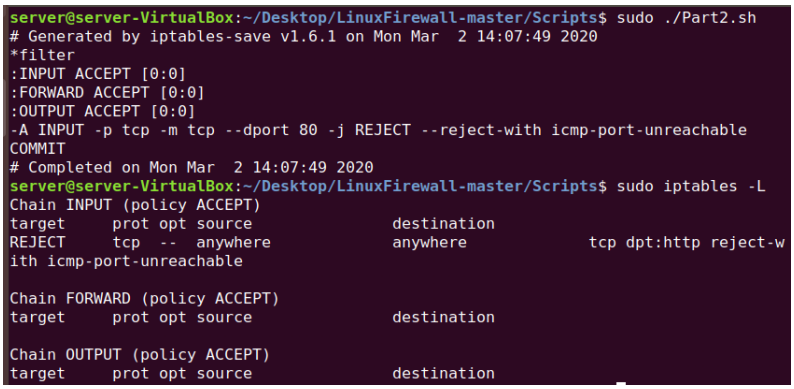


Figure 6: Running `Part2.sh` script

We can see that the client is not able to access the **http service** as below, but the **ssh service** is still accessible.

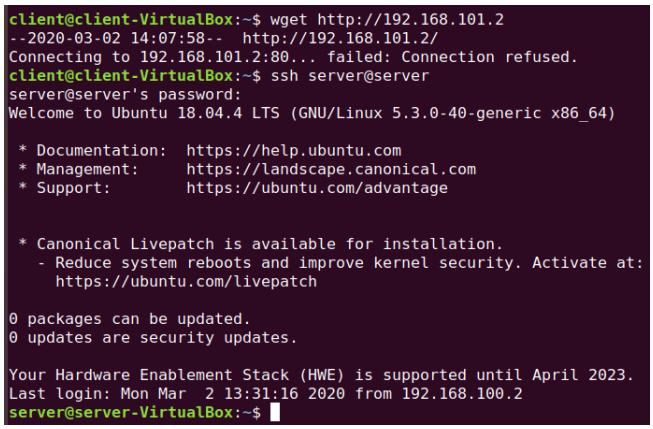


Figure 7: Testing `Part2.sh` script

3 Default Deny - host firewalling

For this part, the below script is being used to block the access for all services on the server, except for **ssh service**.

Listing 2: **Part3.sh**

```
#!/bin/bash
iptablesPATH="/sbin/iptables"
iptablesSavePATH="/sbin/iptables-save"
# flush all chains
$iptablesPATH -F
# Default Deny
$iptablesPATH -P INPUT DROP
# Exception to Default Deny
$iptablesPATH -A INPUT -p tcp --dport 22 -j ACCEPT # SSH
# Accept the packets that are related to the connectins established by the server
$iptablesPATH -A INPUT --match state --state ESTABLISHED,RELATED --jump ACCEPT
# Save Settings
$iptablesSavePATH
```

After applying the rules on the server, the client is not able to access any service except **ssh**.For instance, if the client tries to connect to **port:1234** (as shown in the below figure) the attempt to connect to this port will time out.

```
server@server-VirtualBox:~/Desktop/LinuxFirewall-master/Scripts$ sudo ./Part3.sh
# Generated by iptables-save v1.6.1 on Mon Mar  2 16:29:25 2020
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Mon Mar  2 16:29:25 2020
server@server-VirtualBox:~/Desktop/LinuxFirewall-master/Scripts$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTAB
LISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

Figure 8: Running `Part3.sh` script

```
client@client-VirtualBox:~$ wget http://192.168.101.2
--2020-03-02 16:31:15--  http://192.168.101.2/
Connecting to 192.168.101.2:80... failed: Connection timed out.
Retrying.

--2020-03-02 16:33:27--  (try: 2)  http://192.168.101.2/
Connecting to 192.168.101.2:80... ^Z
[1]+  Stopped                 wget http://192.168.101.2
client@client-VirtualBox:~$ telnet 192.168.101.2 1234
Trying 192.168.101.2...
telnet: Unable to connect to remote host: Connection timed out
client@client-VirtualBox:~$ ssh server@server
server@server's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-40-generic x86_64
)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate
 at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 20
23.
Last login: Mon Mar  2 16:32:38 2020 from 192.168.100.2
server@server-VirtualBox:~$
```

Figure 9: Testing **port:1234** & **port:80**

4 Router firewalling

For this task, the `FORWARD` chain has been used since the script will be running on the router machine. The script for the first part of this task is as follows:

Listing 3: **Part4-2.sh**

```
#!/bin/bash
iptablesPATH="/sbin/iptables"
iptablesSavePATH="/sbin/iptables-save"
ServerIP=192.168.101.2
# flush all chains
$iptablesPATH -F
# Block outgoing connection to 192.168.101.2 on port 80
$iptablesPATH -A FORWARD -d $ServerIP -p tcp --dport 80 -j DROP
# Save Settings
$iptablesSavePATH
```

For testing the correctness of the rules, the same steps have been followed.

```
router@router-VirtualBox:~/Desktop/LinuxFirewall-master/Scripts$ sudo ./Part4-2.sh
# Generated by iptables-save v1.6.1 on Mon Mar  2 17:29:55 2020
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A FORWARD -d 192.168.101.2/32 -p tcp -m tcp --dport 80 -j DROP
COMMIT
# Completed on Mon Mar  2 17:29:55 2020
router@router-VirtualBox:~/Desktop/LinuxFirewall-master/Scripts$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             server               tcp dpt:http

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

Figure 10: Running `Part4-2.sh` script

```
client@client-VirtualBox:~$ wget http://192.168.101.2
--2020-03-02 17:30:12--  http://192.168.101.2/
Connecting to 192.168.101.2:80... failed: Connection timed out.
Retrying.

--2020-03-02 17:32:24--  (try: 2)  http://192.168.101.2/
Connecting to 192.168.101.2:80... ^Z
[3]+  Stopped                 wget http://192.168.101.2
client@client-VirtualBox:~$ ssh server@server
server@server's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-40-generic x86_64
)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate
at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 20
23.
Last login: Mon Mar  2 16:40:10 2020 from 192.168.100.2
server@server-VirtualBox:~$
```

Figure 11: Testing `Part4-2.sh` script

The second part of this task achieved using the follow script:

Listing 4: **Part4-3.sh**

```
#!/bin/bash
iptablesPATH="/sbin/iptables"
iptablesSavePATH="/sbin/iptables-save"
ServerIP="192.168.101.2"
Client_Net_Interface="enp0s8"
Server_Net_Interface="enp0s9"
# flush all chains
$iptablesPATH -F
# Accept connection on 22 port (SSH)
$iptablesPATH -A FORWARD -i $Client_Net_Interface -o $Server_Net_Interface -d $ServerIP -p tcp --dport 22 -j
    ACCEPT
# Accept the packets that are related to the connectins established by the server
$iptablesPATH -A FORWARD -i $Client_Net_Interface -d $ServerIP --match state --state ESTABLISHED,RELATED --jump
    ACCEPT
# Default Deny to incoming connection (from client interface) to 192.168.101.2 on all services
$iptablesPATH -A FORWARD -i $Client_Net_Interface -o $Server_Net_Interface -d $ServerIP -j DROP
# Save Settings
$iptablesSavePATH
```

Similarly, same testing approach has been applied here

```
router@router-VirtualBox:~/Desktop/LinuxFirewall-master/Scripts$ sudo ./Part4-3.sh
# Generated by iptables-save v1.6.1 on Fri Mar  6 15:35:37 2020
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A FORWARD -d 192.168.101.2/32 -i enp0s8 -o enp0s9 -p tcp -m tcp --dport 22 -j ACCEPT
-A FORWARD -d 192.168.101.2/32 -i enp0s8 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -d 192.168.101.2/32 -i enp0s8 -o enp0s9 -j DROP
COMMIT
# Completed on Fri Mar  6 15:35:37 2020
router@router-VirtualBox:~/Desktop/LinuxFirewall-master/Scripts$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             server               tcp dpt:ssh
ACCEPT     all  --  anywhere             server               state RELATED,ESTABLISHED
DROP       all  --  anywhere             server

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
router@router-VirtualBox:~/Desktop/LinuxFirewall-master/Scripts$
```

Figure 12: Running `Part4-3.sh` script

```
client@client-VirtualBox:~$ wget http://192.168.101.2
--2020-03-06 16:15:42--  http://192.168.101.2/
Connecting to 192.168.101.2:80... failed: Connection timed out.
Retrying.

--2020-03-06 16:17:53--  (try: 2)  http://192.168.101.2/
Connecting to 192.168.101.2:80... ^Z
[9]+  Stopped                 wget http://192.168.101.2
client@client-VirtualBox:~$ telnet 192.168.101.2 4321
Trying 192.168.101.2...
telnet: Unable to connect to remote host: Connection timed out
client@client-VirtualBox:~$ ssh server@server
server@server's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-40-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at
     https://ubuntu.com/livepatch

15 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Fri Mar  6 15:36:22 2020 from 192.168.100.2
server@server-VirtualBox:~$ exit
logout
Connection to server closed.
client@client-VirtualBox:~$
```

Figure 13: Testing **port:4321** & **port:80**

5 BCP38 Ingress and Egress Control

The script for this task is as follows:

Listing 5: **Part5.sh**

```
#!/bin/bash
iptablesPATH="/sbin/iptables"
iptablesSavePATH="/sbin/iptables-save"
Client_Net_Interface="enp0s8"
Server_Net_Interface="enp0s9"
# flush all chains
$iptablesPATH -F
# Drop all FORWARD packet
$iptablesPATH -P FORWARD DROP
# Exception to the policy , allow trafic from Server_Net_Interface => Client_Net_Interface
$iptablesPATH -A FORWARD -i $Server_Net_Interface -o $Client_Net_Interface -j ACCEPT
# Exception to the policy , allow trafic from Client_Net_Interface => Server_Net_Interface
$iptablesPATH -A FORWARD -i $Client_Net_Interface -o $Server_Net_Interface -j ACCEPT
# Save Settings
$iptablesSavePATH
```

To test the effectiveness of the script, `hping3`[1] tool has been used. In particular, to check if the server can spoof its ip and if it's protected from receiving traffic originated from fake or private IPs.

For example, the below command was being used to spoof the server ip, pretended to be the router (192.169.100.1) using the `-a` flag to spoof the IP and the `-S` flag to send the `SYN` packet to the client (192.168.100.2)

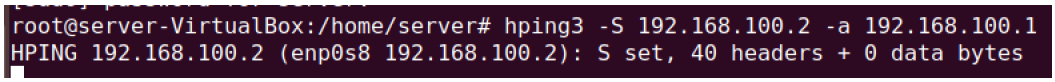


Figure 14: Test 1 : Spoofing IP

After running the script the command failed and the server is not able to send packets with incorrect IP. In more detail, if we try to spoof the server IP with another IP as the follow figure the command is still failing and the client does not receive the `SYN` packet from the spoof IP since the router policy drop that packet:

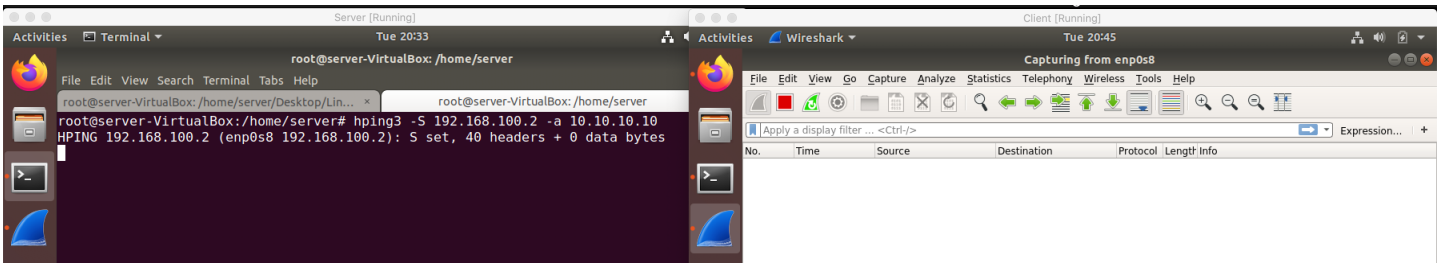


Figure 15: Test 2 : Spoofing IP

However, same command will be working if the server send `SYN` packet with its source IP , we can see that the client receive that packet in `Wireshark` as follow:

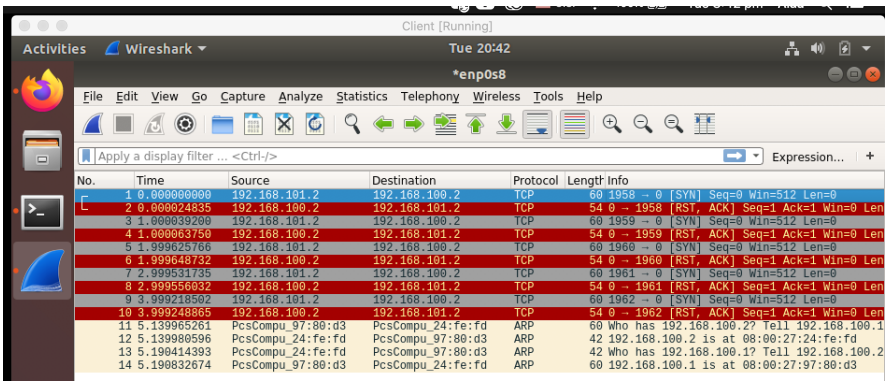


Figure 16: Sending `SYN` packet from Server (correct IP)

[1] https://linux.die.net/man/8/hping3

To test the second part of this task, the same approach has been followed, except that we use the `hping3` on the client-side by crafting a packet with Source IP outside the range of the client and server network and the packet capturing process was on the server-side. All the crafted packets with the source IP outside the range have been dropped by the router policy. The forthcoming figures explain the test process.

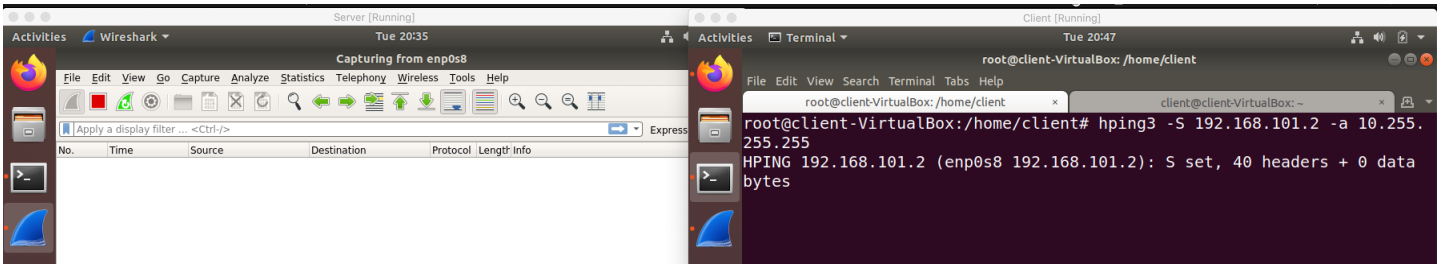


Figure 17: Sending SYN packet from Client with fake IP:10.255.255.255.255

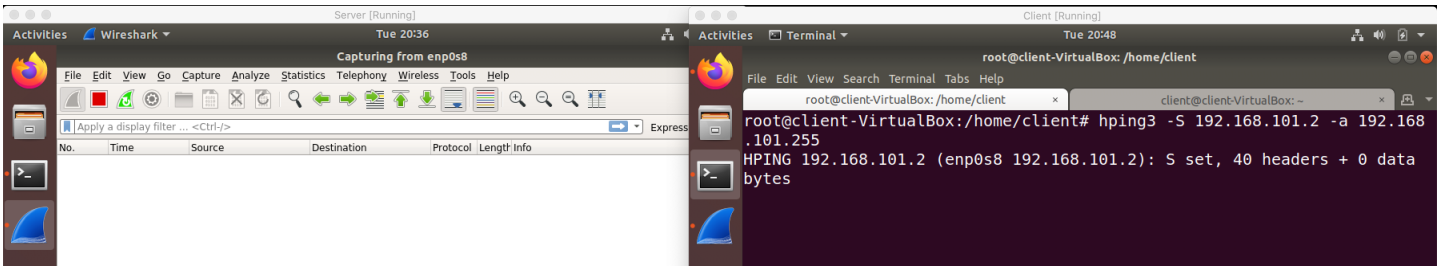


Figure 18: Sending SYN packet from Client with fake IP:192.168.101.255

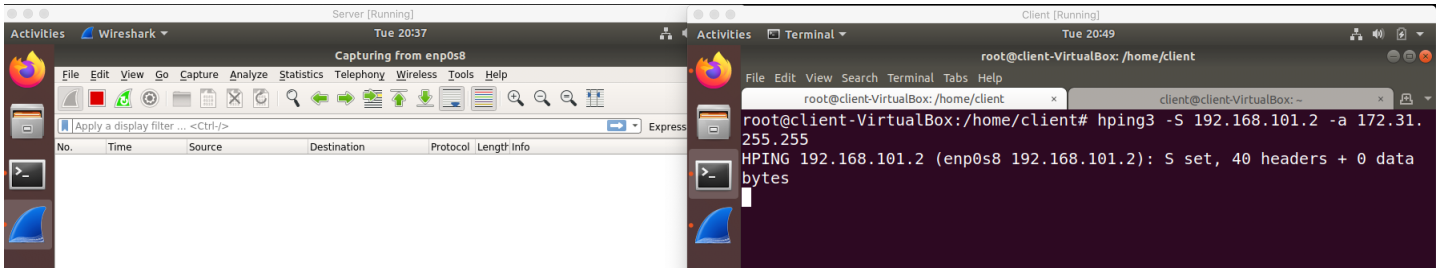


Figure 19: Sending SYN packet from Client with fake IP:172.31.255.255

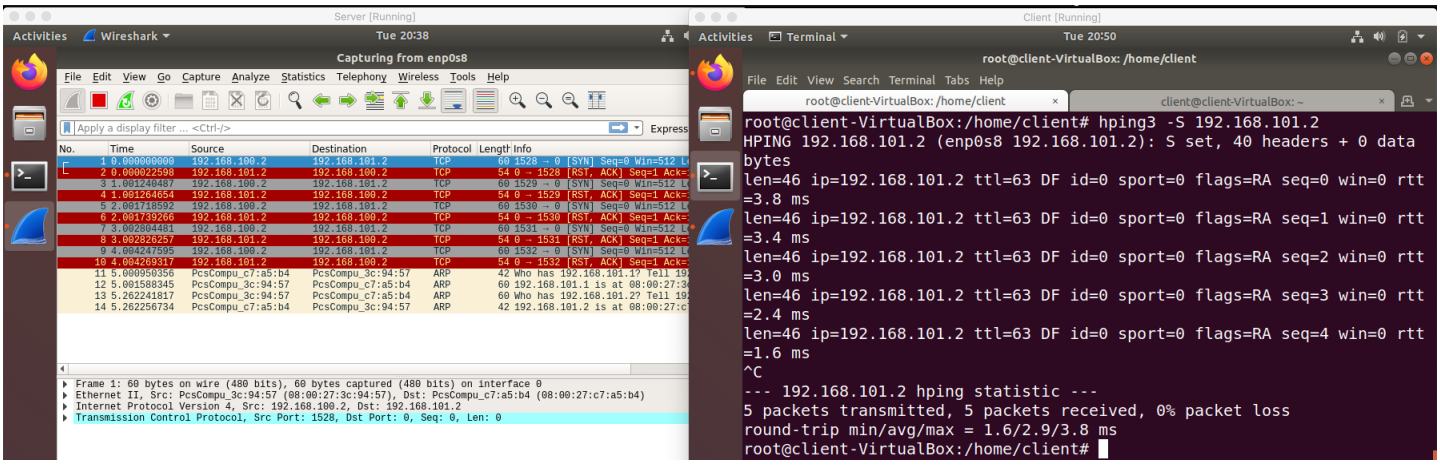


Figure 20: Sending SYN packet from Client (IP within the network range)

6 Logging

To achieve this task, the below two scripts were used for logging the dropped packets. Furthermore, a new chain (LOGGING) was created to log and drop the packet. In addition, the `hashlimit` has been used to reduce the logs records by using the `hashlimit-mode` flag.

Listing 6: **Part6-2.sh**

```
#!/bin/bash
iptablesPATH="/sbin/iptables"
iptablesSavePATH="/sbin/iptables-save"
ServerIP="192.168.101.2"
Client_Net_Interface="enp0s8"
Server_Net_Interface="enp0s9"
# flush all chains
$iptablesPATH -F
# Create a new chain called LOGGING
$iptablesPATH -N LOGGING
$iptablesPATH -A LOGGING -j LOG -m hashlimit --hashlimit 1/sec --hashlimit-mode dstip,dstport,srcip,srcport --
    hashlimit-name hosts --log-prefix "LOGDROPPED: Connection to 80 dropped" --log-level 6
$iptablesPATH -A LOGGING -j DROP
 # Block incoming connection to 192.168.101.2 on port 80
$iptablesPATH -A FORWARD -i $Client_Net_Interface -o $Server_Net_Interface -d $ServerIP -p tcp --dport 80 -j
    LOGGING
# Save Settings
$iptablesSavePATH
```

Listing 7: **Part6-2.sh**

```
#!/bin/bash
iptablesPATH="/sbin/iptables"
iptablesSavePATH="/sbin/iptables-save"
ServerIP="192.168.101.2"
Client_Net_Interface="enp0s8"
Server_Net_Interface="enp0s9"
# flush all chains
$iptablesPATH -F
# Create a new chain called LOGGING2
$iptablesPATH -N LOGGING2
$iptablesPATH -A LOGGING2 -j LOG -m hashlimit --hashlimit 1/sec --hashlimit-mode dstip,dstport,srcip,srcport --
    hashlimit-name hosts --log-prefix "LOGDROPPED2: dropped!!" --log-level 6
$iptablesPATH -A LOGGING2 -j DROP
# Accept the packets that are related to the connectins established by the server
$iptablesPATH -A FORWARD -i $Client_Net_Interface -d $ServerIP --match state --state ESTABLISHED,RELATED --jump
    ACCEPT
# Accept connection on 22 port (SSH)
$iptablesPATH -A FORWARD -i $Client_Net_Interface -o $Server_Net_Interface -d $ServerIP -p tcp --dport 22 -j
    ACCEPT
# Default Deny to outgoing connection to 192.168.101.2 on all services
$iptablesPATH -A FORWARD -i $Client_Net_Interface -o $Server_Net_Interface -d $ServerIP -j LOGGING2
# Save Settings
$iptablesSavePATH
```

The following figures show the logs records for each script

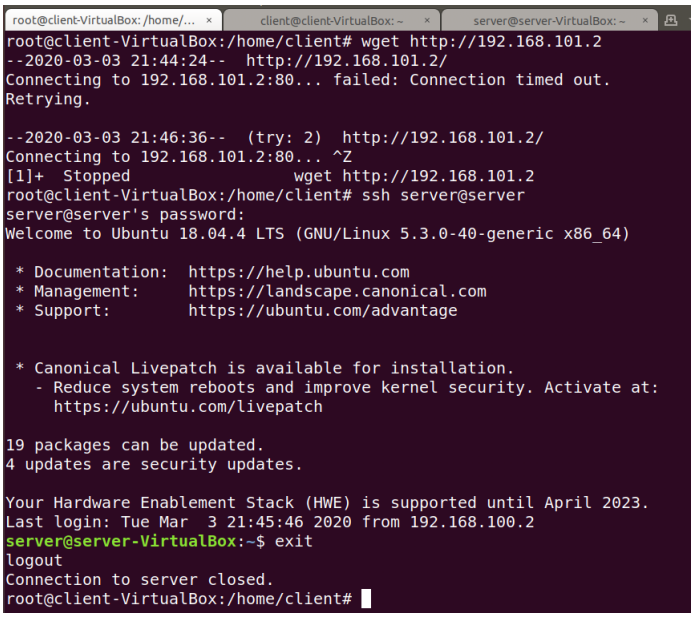


Figure 21: Generating Dropped Packets

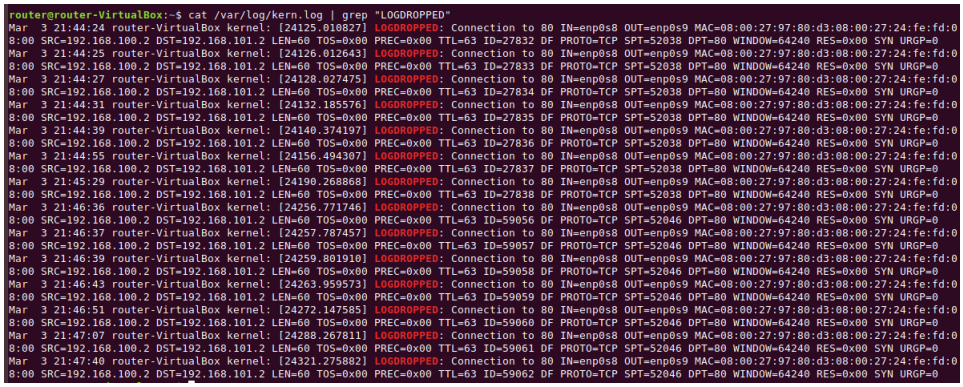


Figure 22: Logs for `Part6-2.sh`

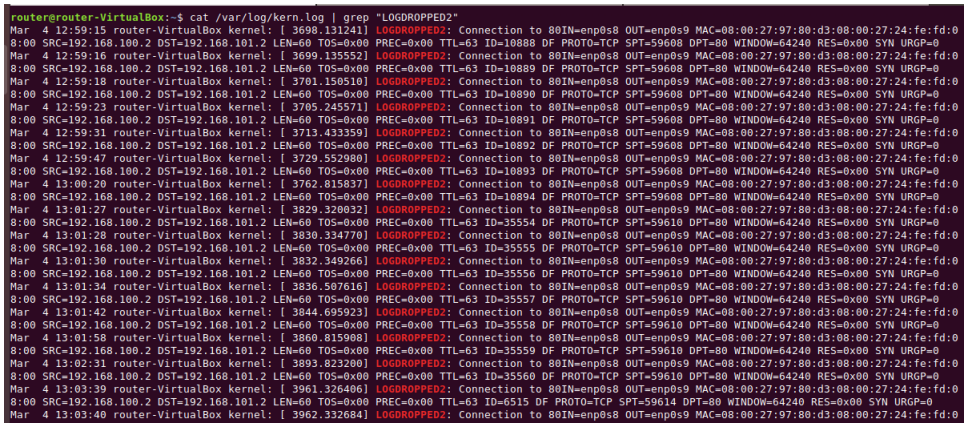


Figure 23: Logs for `Part6-3.sh`