NAME : YIKWENMEIN VICTOR MAGHENG

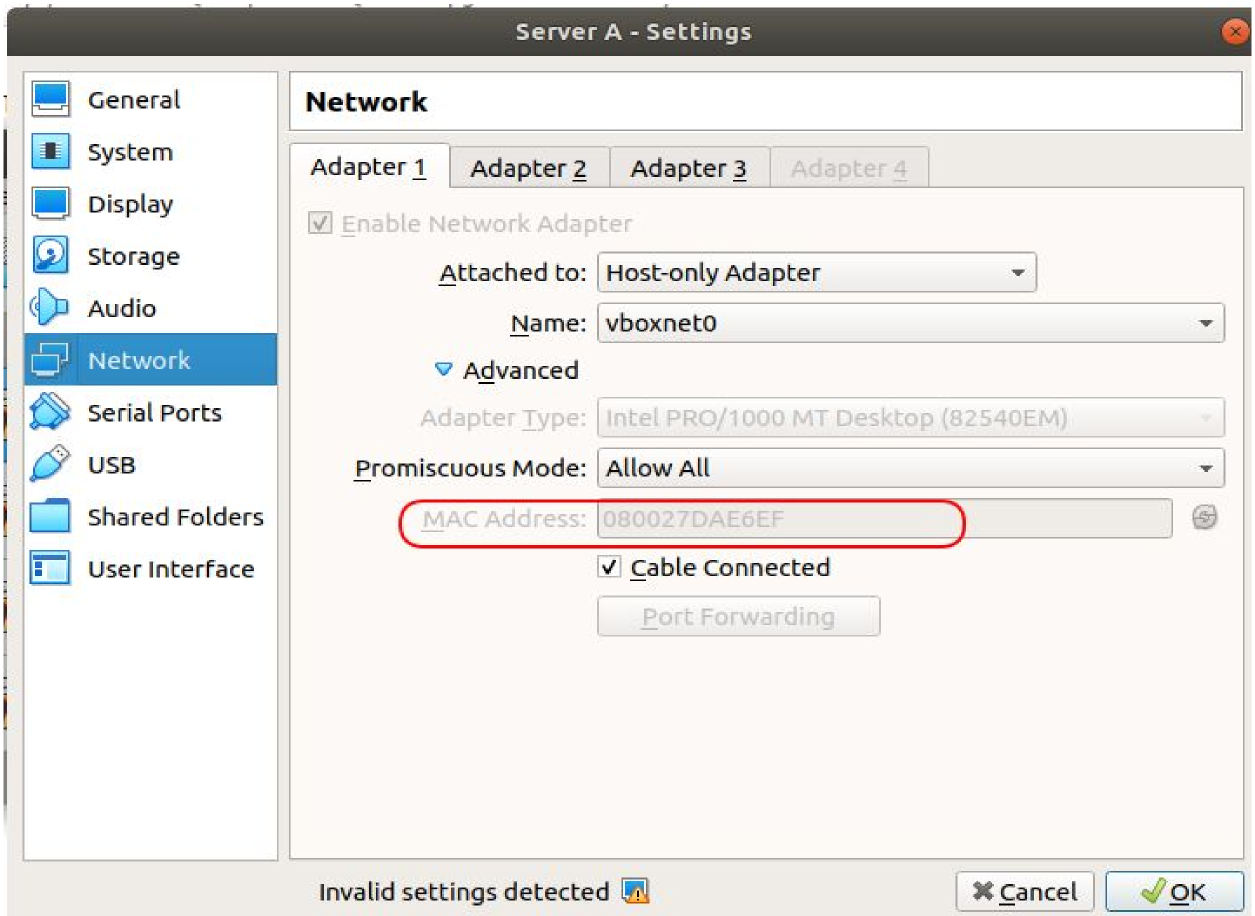
Personal Number: +23767880691

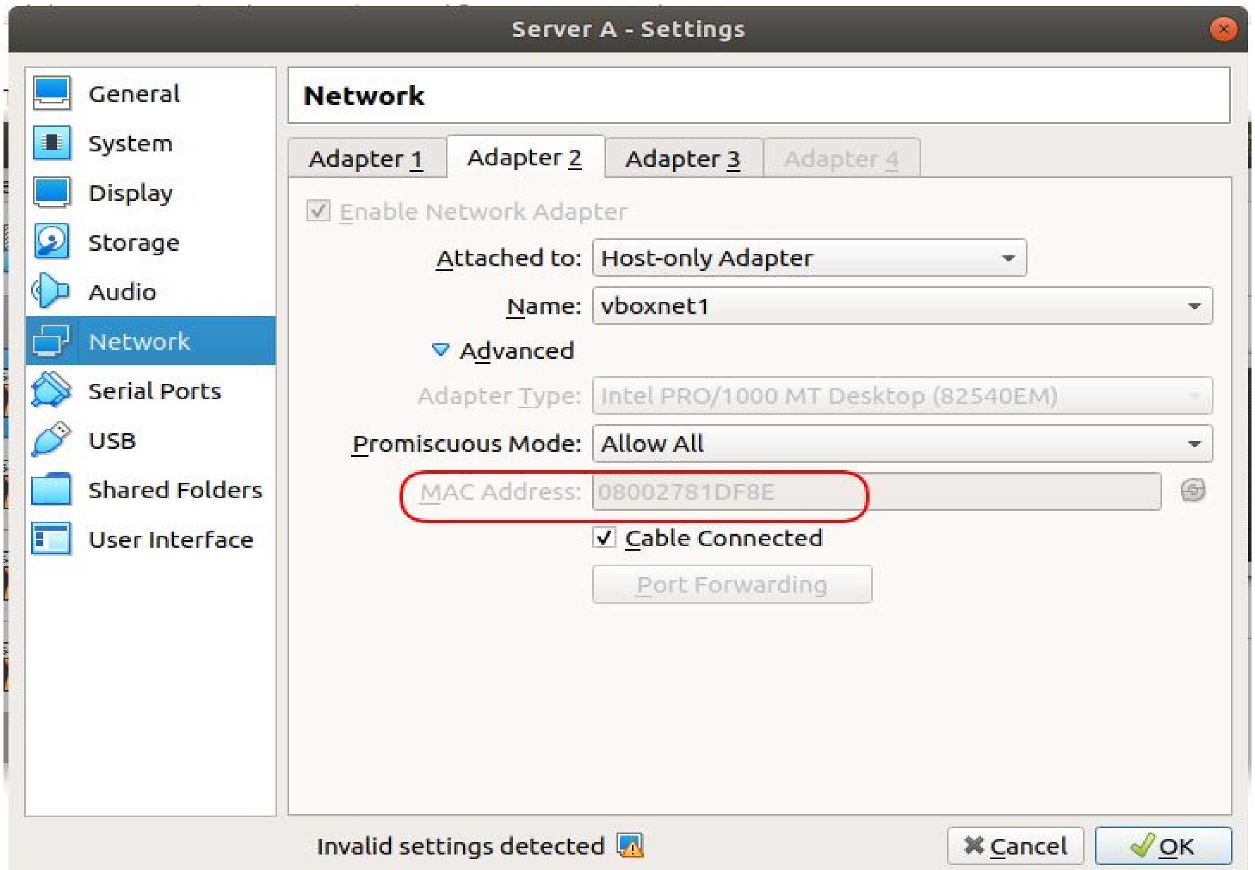
Email address: yikwenmeinvictor1995@gmail.com

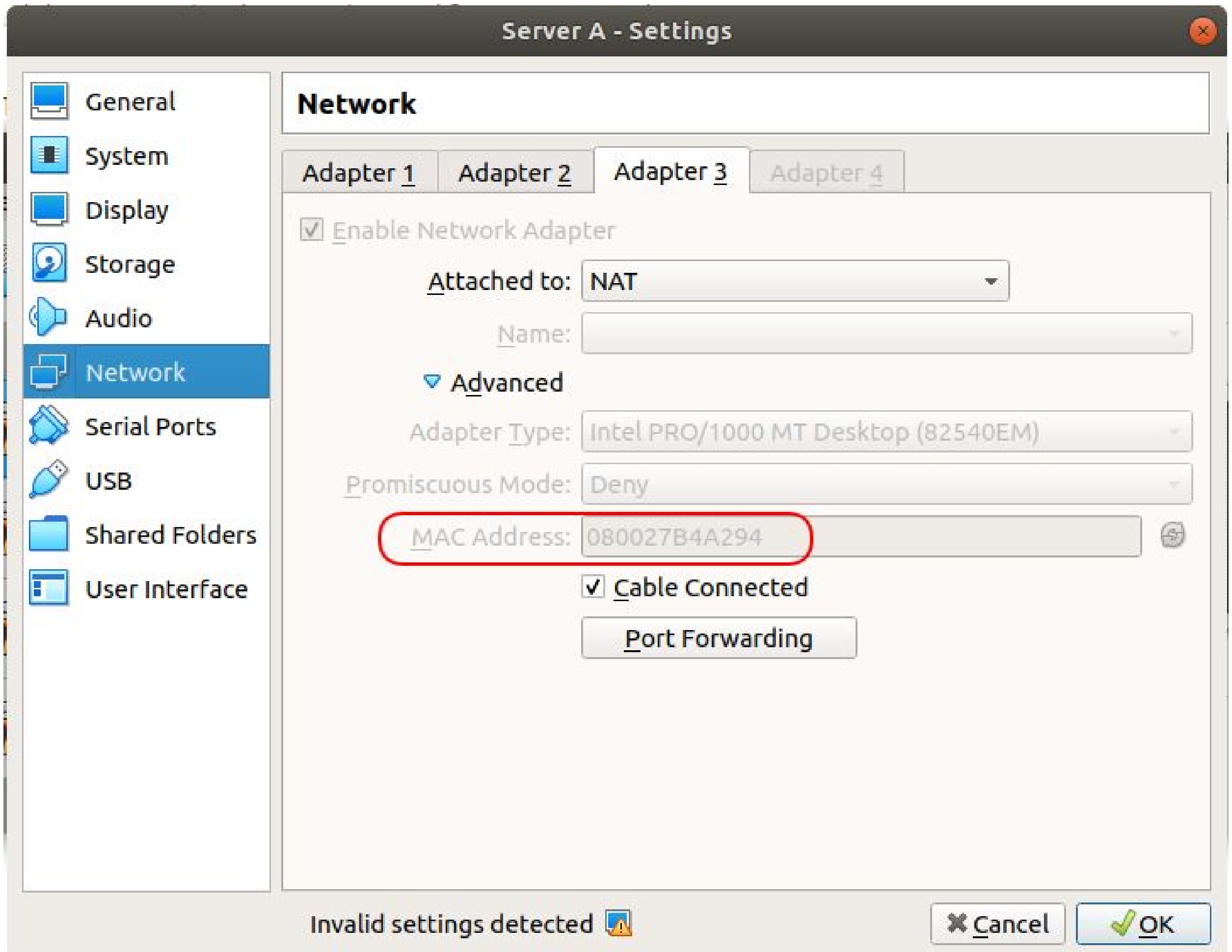
Task 1: MAC addresses

Identify the MAC address of the configured adapters in the Web Server VM and write them down.

Adapter	MAC address
Adapter 1	080027DAE6EF
Adapter 2	08002781DF8E
Adapter 3	080027B4A294



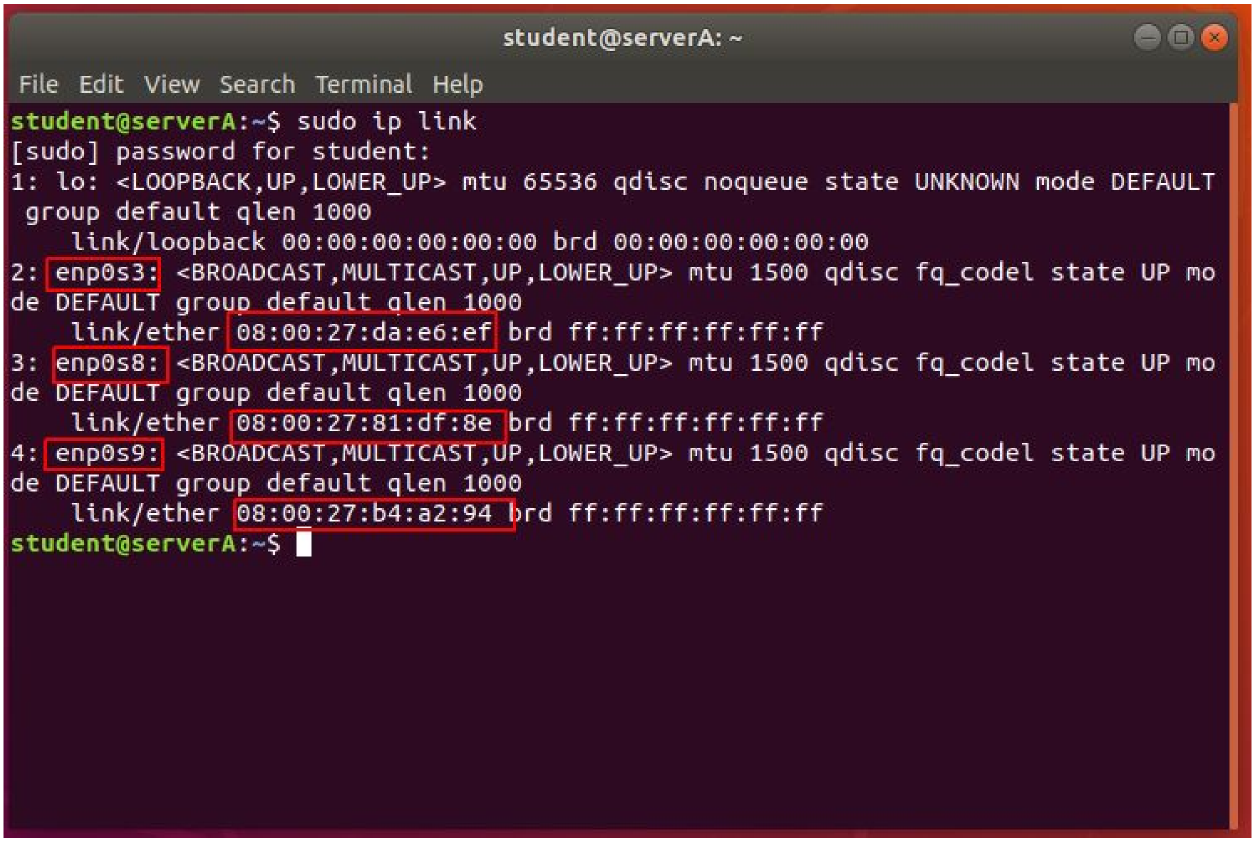

Server A - Settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to: Host-only Adapter
Name: vboxnet1
Advanced
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode: Allow All
MAC Address: 08002781DF8E
Cable Connected
Port Forwarding
Invalid settings detected
Cancel
OK



Task 2: Network interfaces

In the interface list shown by one of the command above, use the MAC numbers from Task 1 to identify which interface is the NAT interface and which ones are the host-only interfaces.

Interface	Interface type
enp0s3	Host-only
enp0s8	Host-only
enp0s9	NAT



Task 3: IP addresses, netmasks and subnet

Note down what IP addresses and netmasks are assigned to which interfaces. Derive the network addresses (the subnets) associated with each address and note them down. Remember from the lecture that the network address can be computed from the following operation (AND is the bitwise AND operation):
(IP address) AND (network mask)

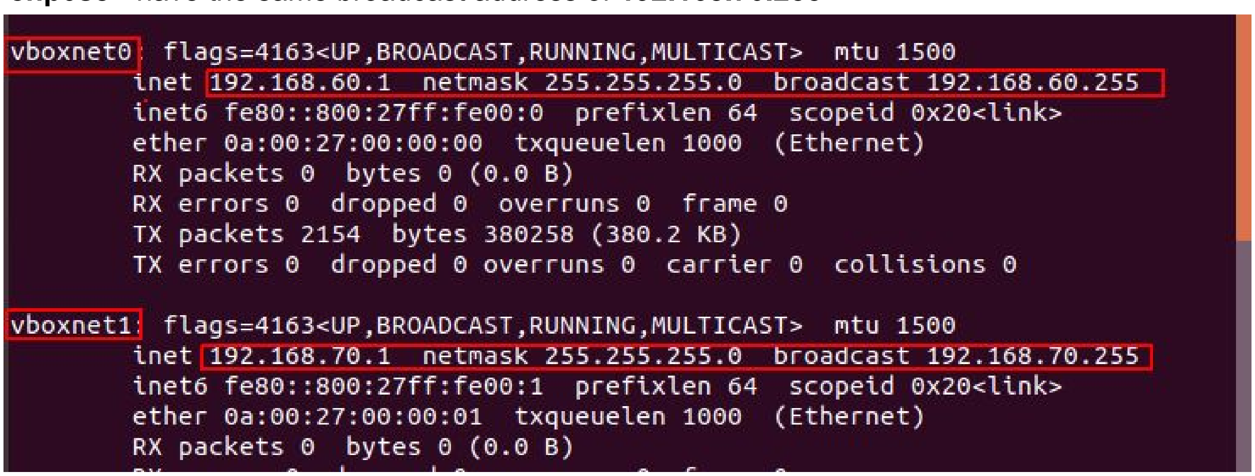
Interface	IP address	Netmask	Subnet = IP address AND Netmask
enp0s3	192.168.60.100/24 1100 0000.1010 1000. 0011 1100.0110 0100	255.255.255.0 1111 1111 . 1111 1111 . 1111 1111 . 0000 0000	192.168.60.0 1100 0000.1010 1000. 0011 1100.0000 0000
enp0s8	192.168.70.5/24 1100 0000.1010 1000. 0100 0110.0000 0101	255.255.255.0 1111 1111 . 1111 1111 . 1111 1111 . 0000 0000	192.168.70.0 1100 0000.1010 1000. 0100 0110.0000 0000

enp0s9	10.0.98.100/24 0000 1010.0000 0000. 0110 0010.0110 0100	255.255.255.0 1111 1111 . 1111 1111 . 1111 1111 . 0000 0000	10.0.98.0 0000 1010.0000 0000. 0110 0010.0000 0000

Task 4: Host-only interfaces

Host-only interface	IP address	Netmask
vboxnet0	192.168.60.1	255.255.255.0
vboxnet1	192.168.70.1	255.255.255.0

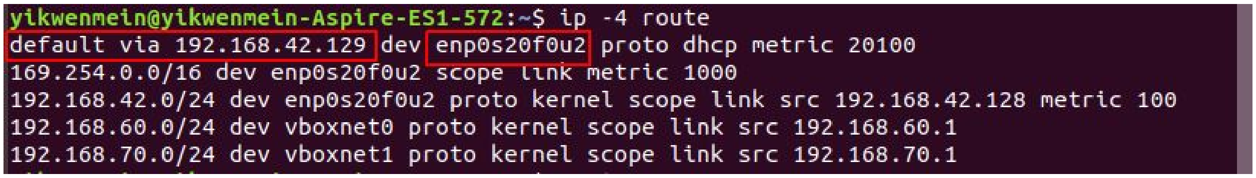
- The host-only interface “**vboxnet0**” on the host is connected to the host-only interface “**enp0s3**” on the guest. This is because the IP address(**192.168.60.1**) of the host interface **vboxnet0** belongs to the subnet(**192.168.60.0**) of the guest interface **enp0s3.** Also, these interfaces “**vboxnet0**” and “**enp0s3**” have the same broadcast address of **192.168.60.255**
- The host-only interface “**vboxnet1**” on the host is connected to the host-only interface “**enp0s8**” on the guest. This is because the IP address(**192.168.70.1**) of the host interface **vboxnet1** belongs to the subnet(192.168.70.0) of the guest interface enp0s8. Also, these interfaces **“vboxnet1**” and “**enp0s8**” have the same broadcast address of **192.168.70.255**



```
vboxnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.1  netmask 255.255.255.0  broadcast 192.168.60.255
        inet6 fe80::800:27ff:fe00:0  prefixlen 64  scopeid 0x20<link>
        ether 0a:00:27:00:00:00  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2154  bytes 380258 (380.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vboxnet1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.70.1  netmask 255.255.255.0  broadcast 192.168.70.255
        inet6 fe80::800:27ff:fe00:1  prefixlen 64  scopeid 0x20<link>
        ether 0a:00:27:00:00:01  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
```

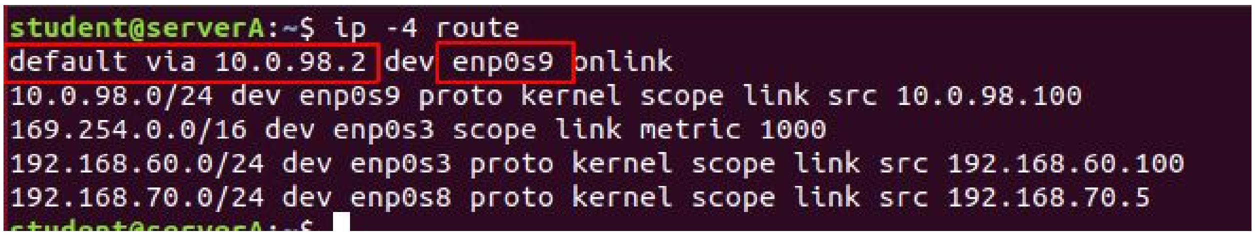
Task 5: Routing tables in the host OS

The default gateway(**192.168.42.129**) of my host OS can be reached through the interface **enp0s20f0u2**



Task 6: Routing tables in the guest OS

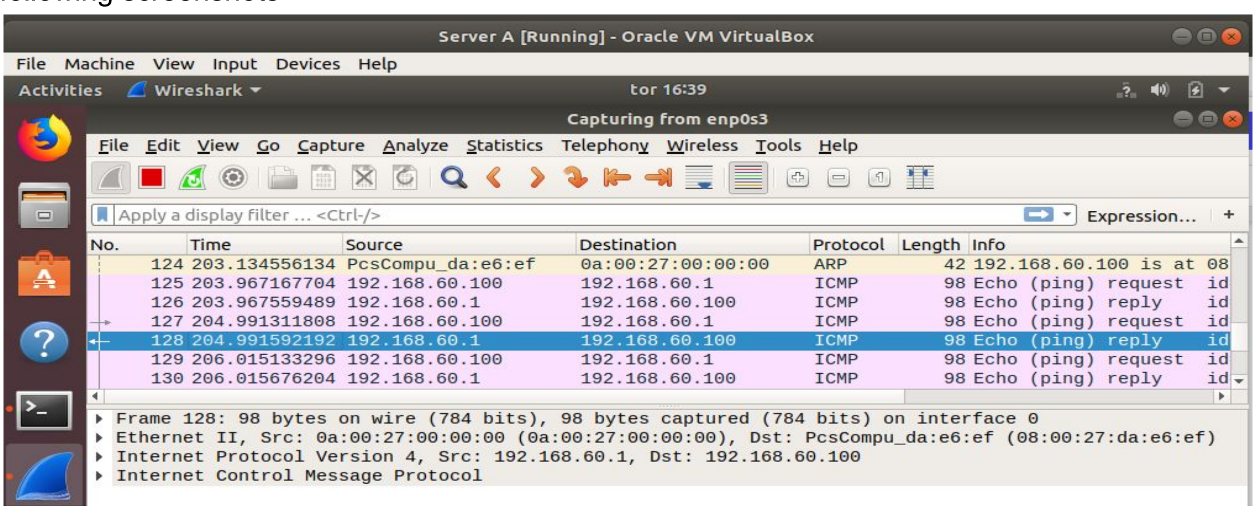
The default gateway(**10.0.98.2**) on the guest OS can be reached through the interface **enp0s9** and this is a NAT interface as can be seen from Task 2

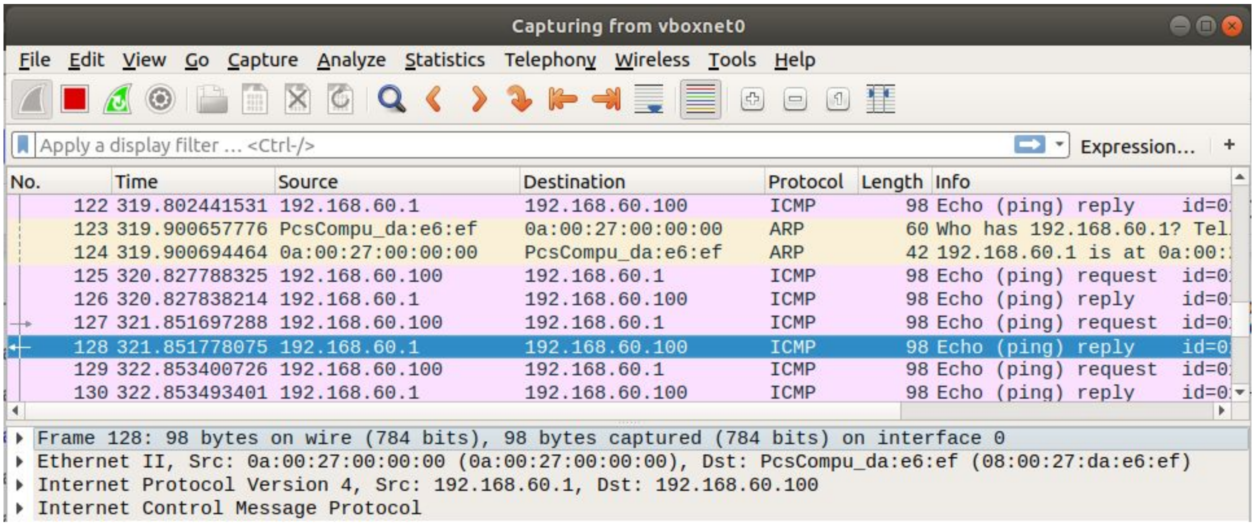


Task 7: Ping the host-based host-only interface

The host-only interface that is assigned IP address 192.168.60.100 by the guest OS is **enp0s3** and it's corresponding interface on the host is **vboxnet0.** So vboxnet0 interface is selected on host's wireshark while the **enp0s3** is selected on guest wireshark.

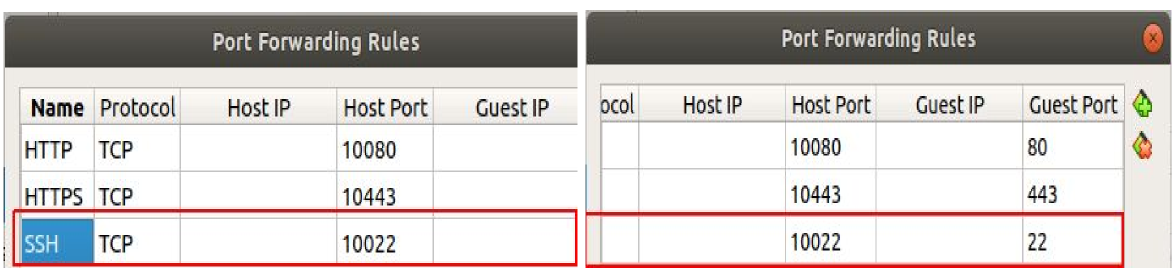
Examination of the ICMP traffic from the two Wireshark instances shows that they are identical as on the following screenshots





Task 8: ssh into VM via localhost

For this task I added a forwarding rule to Adapter3 (the NAT adapter) with the following values as on this screenshot



Afterwards, I opened server A VM, launched a terminal prompt. After the launch,another terminal is opened but now on the host OS and the following command is run on it

ssh -p 10022 student@localhost

The terminal prompts for “student@localhost's password” and after a correct password, the host’s shell prompt displayed **student@serverA** as seen on the following screenshot

```
yikwenmein@yikwenmein-Aspire-ES1-572:~$ ssh -p 10022 student@localhost
student@localhost's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-38-generic x86_64)

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

 * Introducing self-healing high availability clustering for MicroK8s!
   Super simple, hardened and opinionated Kubernetes for production.

     https://microk8s.io/high-availability

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

637 packages can be updated.
444 updates are security updates.

Last login: Sat Nov 10 23:11:06 2018 from 10.0.98.2
student@serverA:~$
```

Task 9: Add forwarding rules for HTTP and HTTPS in VirtualBox

Once again, port forwarding rules for HTTP and HTTPS were added as seen on this screenshot.

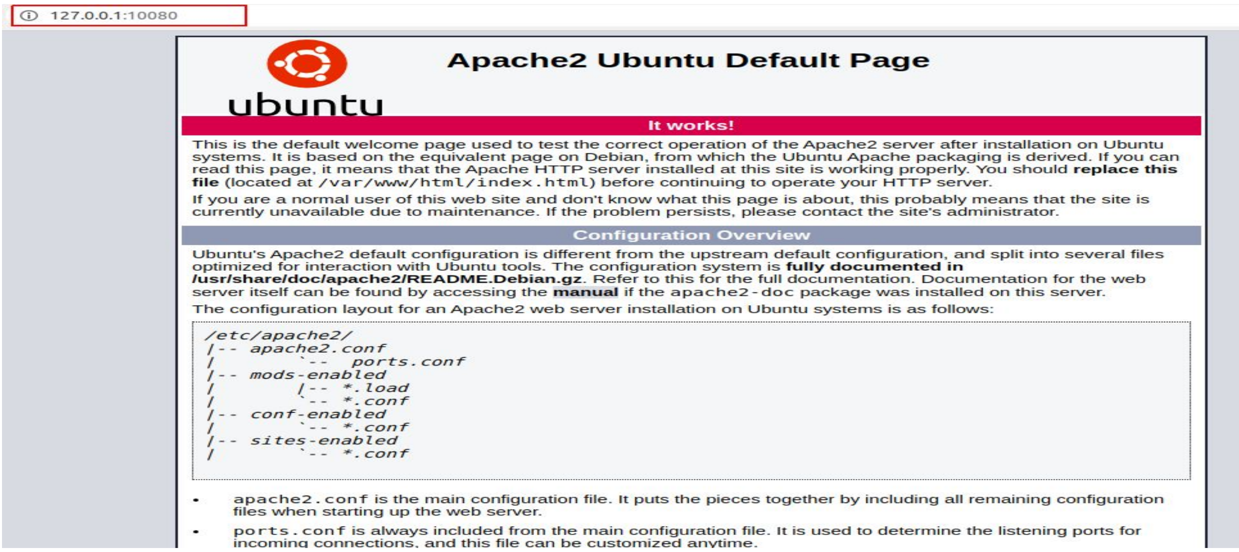
Port Forwarding Rules

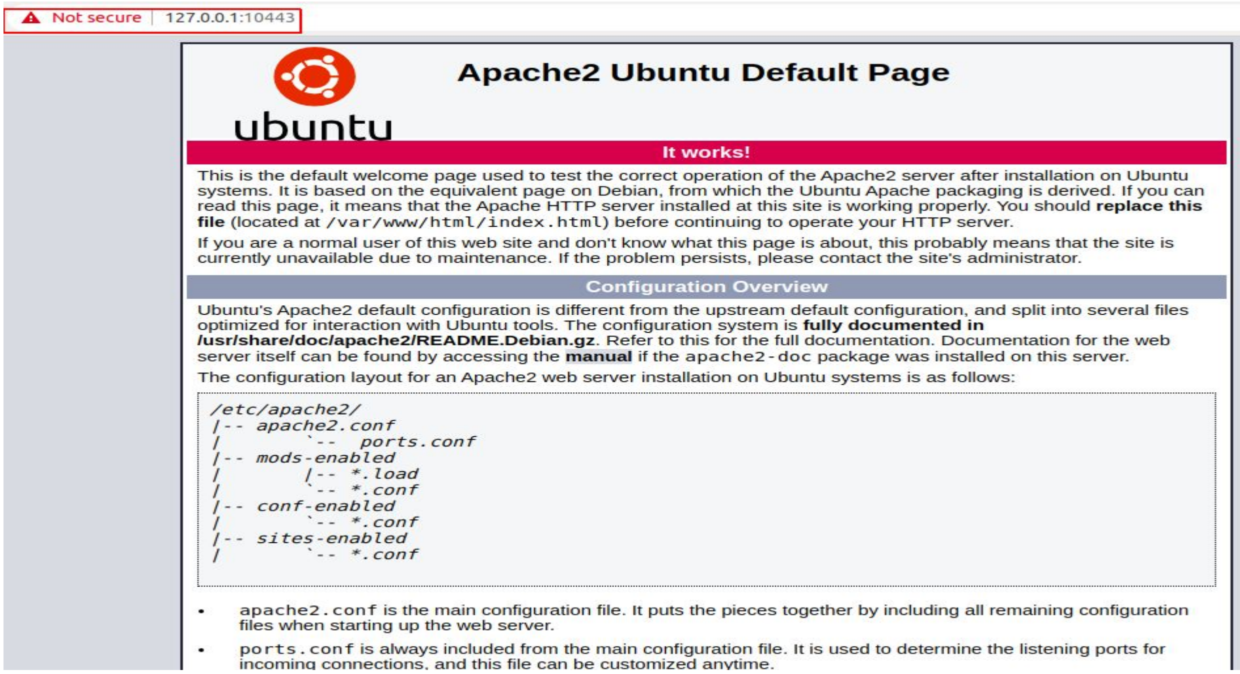
Name	Protocol	Host IP	Host Port
HTTP	TCP		10080
HTTPS	TCP		10443
SSH	TCP		10022

Port Forwarding Rules

Host Port	Guest IP	Guest Port
10080		80
10443		443
10022		22

Host port numbers used are **10080 for HTTP** and **10443 for HTTPS**. After this, a browser is launched on the host system and connections to the apache2 server on the guest are done over HTTP(**http://127.0.0.1:10080/**) and HTTPS(**https://127.0.0.1:10443/**)respectively as shown on the screenshots below





Task 10: Default firewall policy and rules

For this task. These documents [sudo iptables] and [Use of IPTables in a Virtual Machine Environment] which i found online helped me understand iptables, policies and rules in more detail

To get the default policy and rule for the mangle iptable, this command is executed
sudo iptables -t mangle -L
The default policy is ACCEPT but no default rule as shown on this screenshot

```
student@serverA:~$ sudo iptables -t mangle -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

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

To get the default policy and rule for the filter iptable, this command is executed
sudo iptables -t filter -L
The default policy is ACCEPT but no default rule as shown on this screenshot

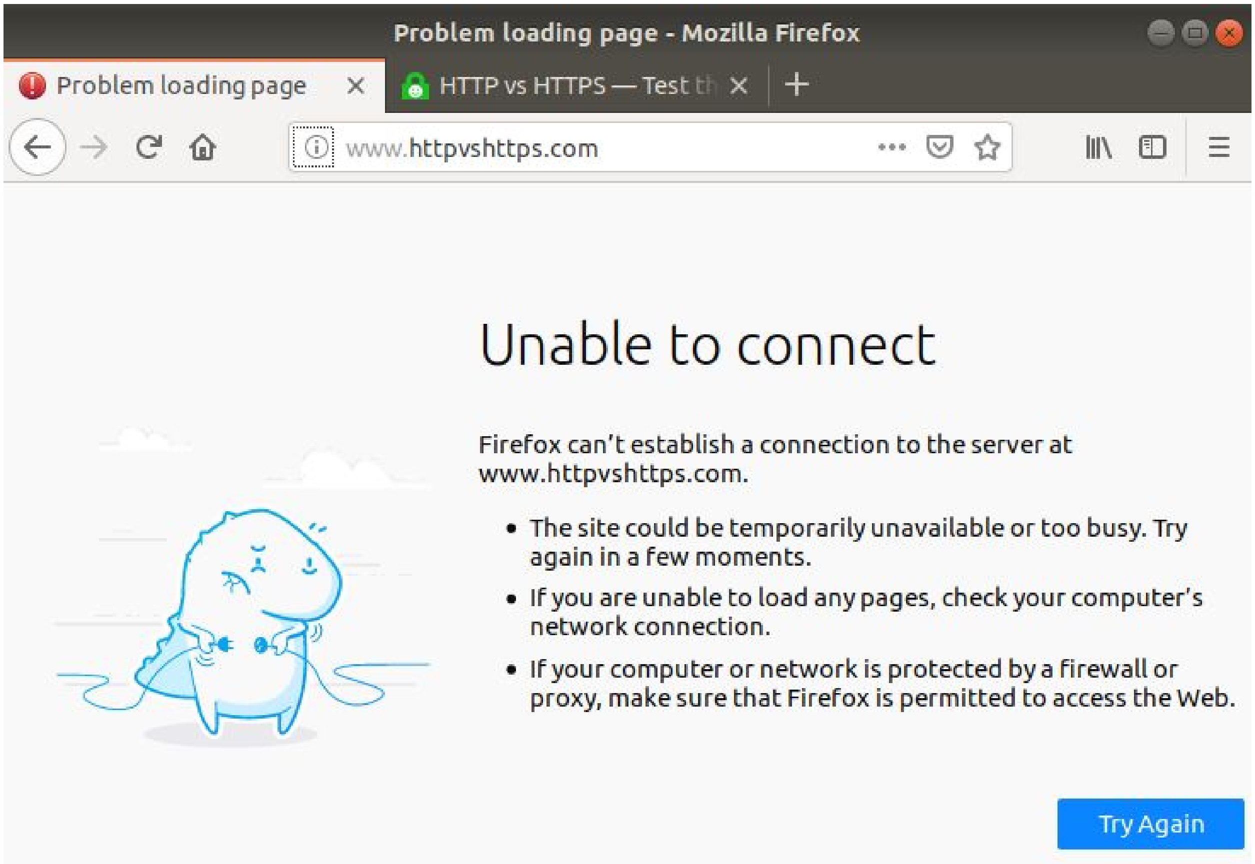
```
student@serverA:~$ sudo iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

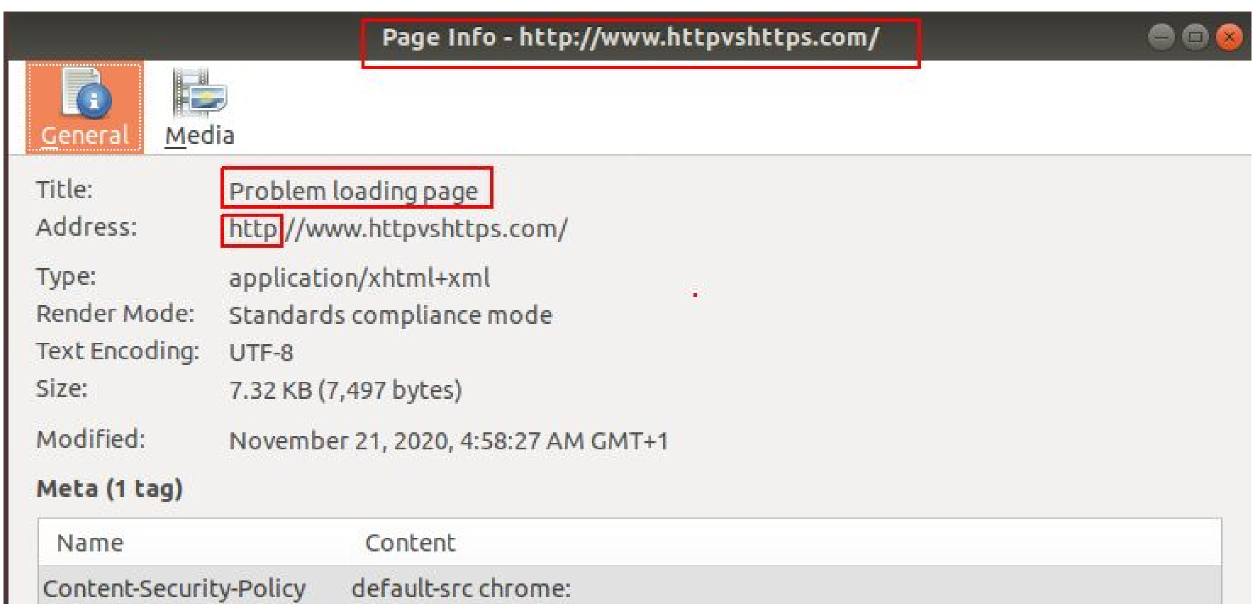
Likewise, to get the default policy and rule for the nat iptable, this command is executed
sudo iptables -t nat -L
The default policy is ACCEPT but no default rule as shown on this screenshot

```
student@serverA:~$ sudo iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

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

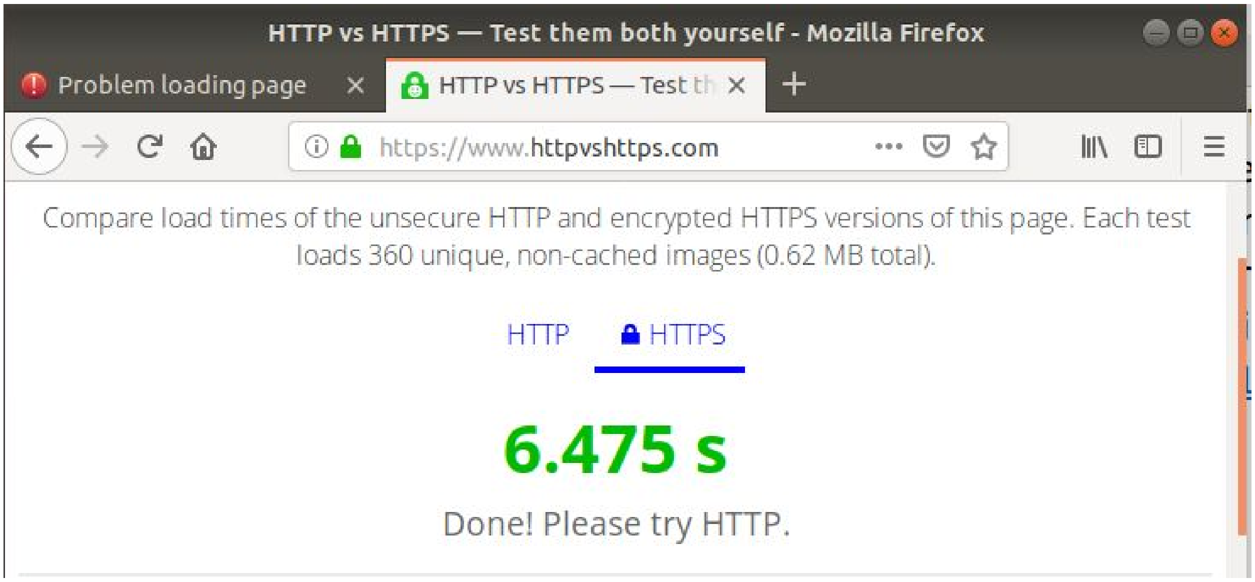
Task 11: Block HTTP-browsing in the guest OS

For this task, I blocked HTTP browsing on the guest by setting up the following reject rule ***sudo iptables -A OUTPUT -p tcp --dport 80 -j REJECT*** in the OUTPUT chain. This rule blocks HTTP(port 80) browsing.

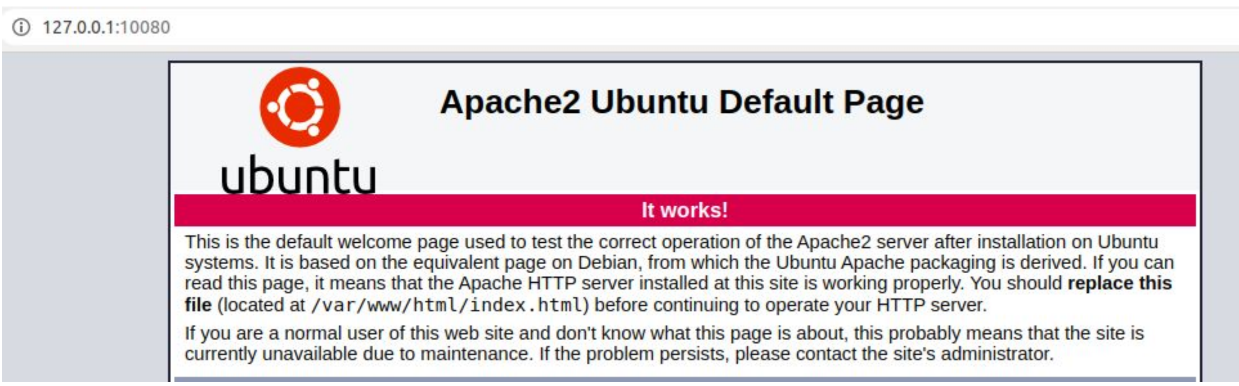




Since other default policies are “ACCEPT”, we are then still able to browse HTTPS and other sites.



Furthermore, HTTP traffic produced by the apache2 server, can still be browsed when browsing from the host.

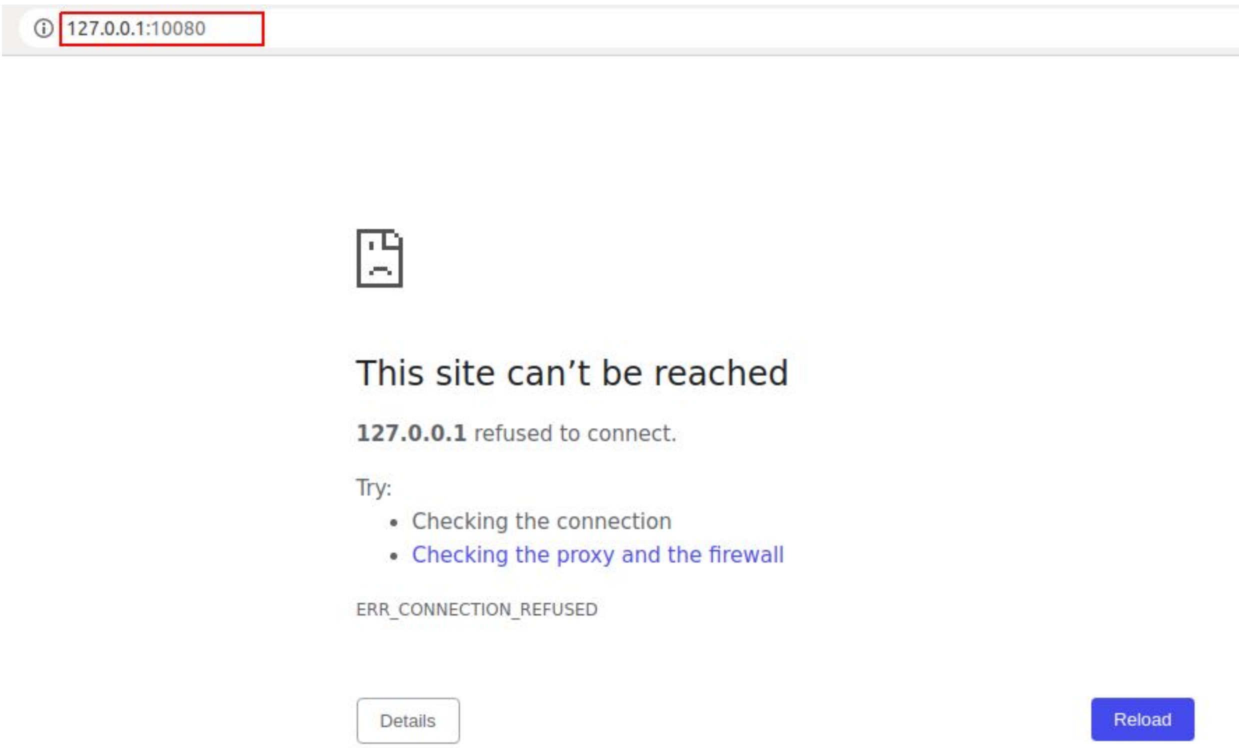


Task 12: Block Apache web server from serving content over HTTP

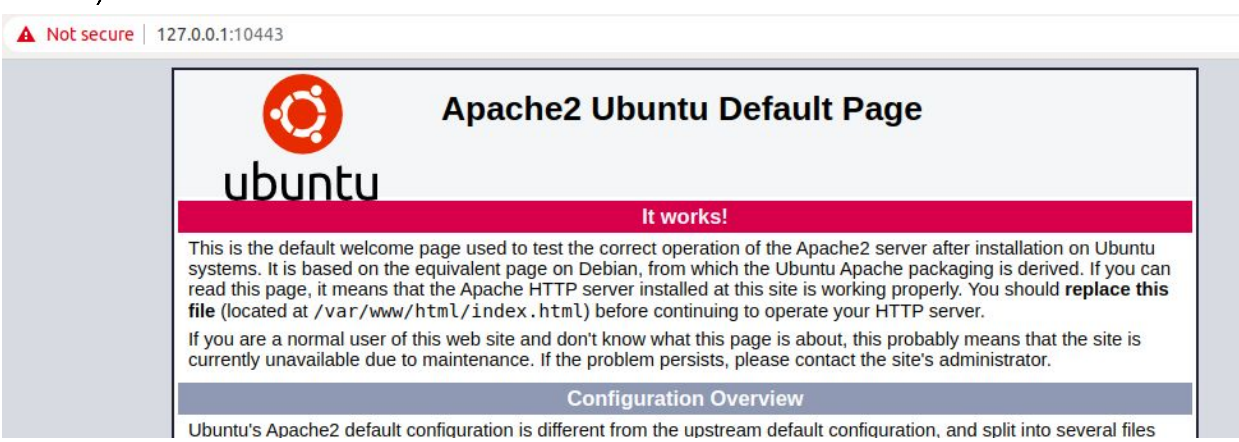
The operation is similar to that of task 11 but performed on the host system. The host is blocked from accessing HTTP content from the apache2 server in the guest OS by adding the following reject rule

sudo iptables -A OUTPUT -p tcp --dport 10080 -j REJECT

in the OUTPUT chain. This rule blocks HTTP(port 10080) browsing.



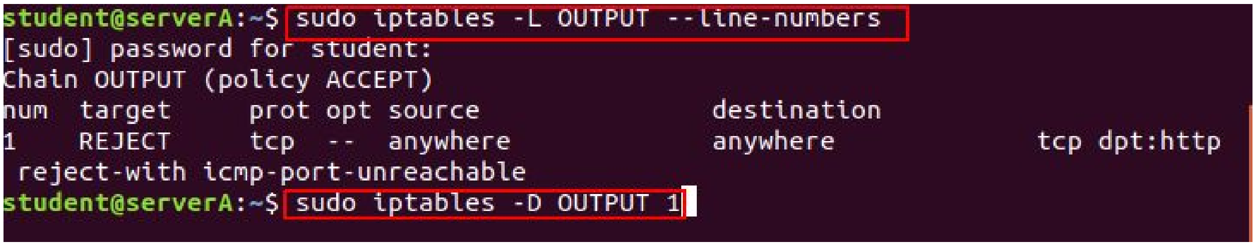
Only HTTP (port 10080) is blocked. So the ability to see content served over HTTPS(port 10443) is still retained



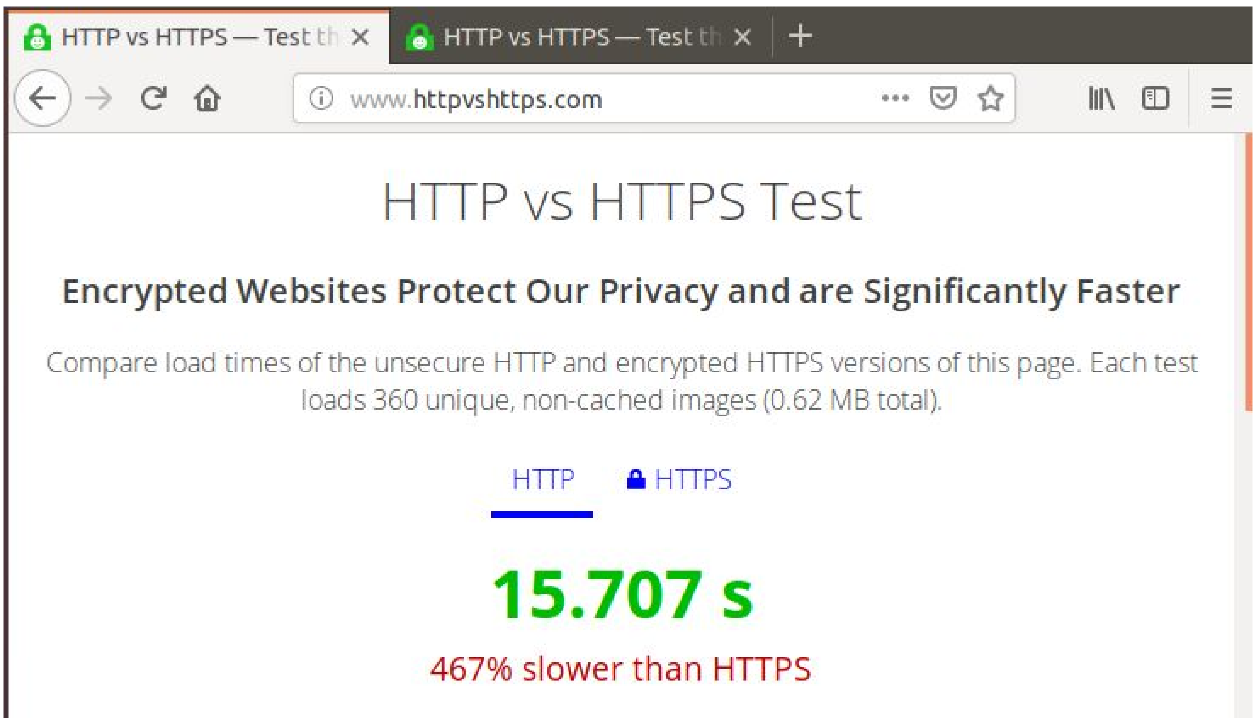
Task 13: Unblock HTTP-browsing in the guest OS

This is quite simple. To unblock HTTP-browsing in the guest OS, we list the various rules in the OUTPUT chain where HTTP(port 80) was previously blocked with this command ***sudo iptables -L OUTPUT --line-numbers*** and then remove the rule with ***sudo iptables***

-D OUTPUT 1 according to the rule number.



With that, we can now access HTTP once again within the guest system and reloading http://www.httpvshttps.com/ proves that



HTTPS browsing was never blocked.

Task 14: Use firewall.sh to configure the firewall

This required that a firewall is configured such that guest OS can view HTTP and HTTPS pages, but apache2 server is blocked from serving HTTP content. So we configure a firewall that blocks the access of the apache2 server(127.0.0.10) specifically over http(port 80) by adding this command

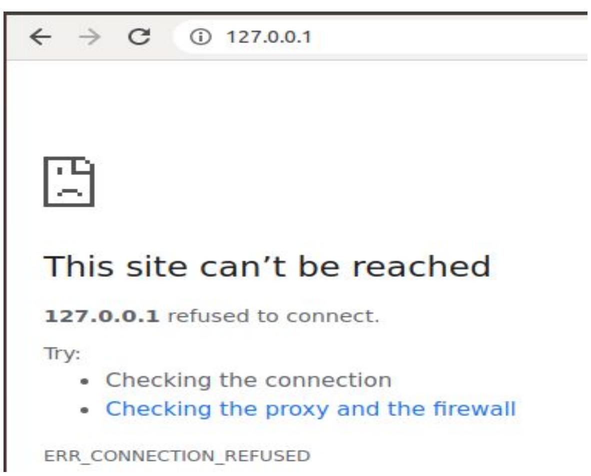
iptables -A OUTPUT -p tcp -d 127.0.0.1 --dport 80 -j REJECT

to the firewall.sh file. Other policies are accept reason why guest OS can view

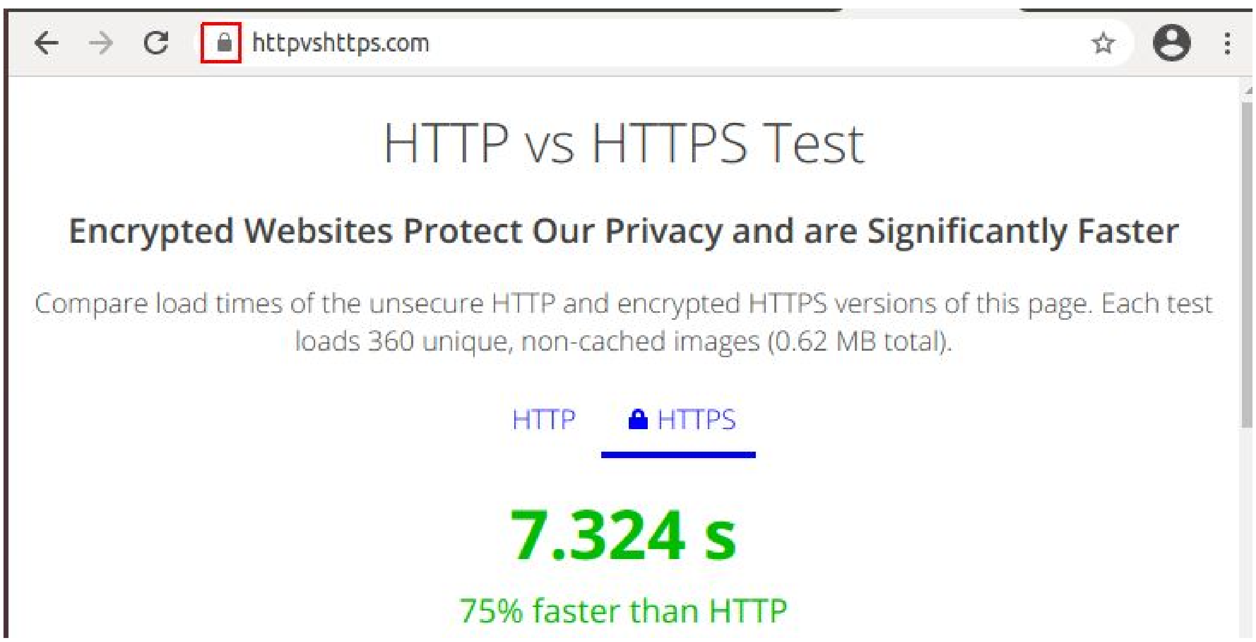
HTTP and HTTPS pages

```
# Flush all chains in FILTER table
$IPT -t filter -F
# Delete any user-defined chains in FILTER table
$IPT -t filter -X
# Flush all chains in NAT table
$IPT -t nat -F
# Delete any user-defined chains in NAT table
$IPT -t nat -X
# Flush all chains in MANGLE table
$IPT -t mangle -F
# Delete any user-defined chains in MANGLE table
$IPT -t mangle -X
# Flush all chains in RAW table
$IPT -t raw -F
# Delete any user-defined chains in RAW table
$IPT -t mangle -X

# Default policy is to send to a dropping chain
$IPT -t filter -P INPUT ACCEPT
$IPT -t filter -P OUTPUT ACCEPT
$IPT -t filter -P FORWARD ACCEPT

iptables -A OUTPUT -p tcp -d 127.0.0.1 --dport 80 -j REJECT
# Create logging chains
#$IPT -t filter -N input_log
#$IPT -t filter -N output_log
#$IPT -t filter -N forward_log

```

Verifying shows that the command has actually blocked just the access of apache2 over http on the guess.



HTTP and HTTPS pages are viewed



Task 15: Change default firewall policy to DROP

Here, the default firewall policy is changed within firewall.sh to drop as follows;

$IPT -t filter -P INPUT DROP

$IPT -t filter -P OUTPUT DROP

$IPT -t filter -P FORWARD DROP

With these changes, I expect that after executing the script firewall.sh, the guest will drop all input,output and forward connections or packets(in general that the guest will not be able to browse the internet)

After executing firewall.sh and executing ***ping 74.125.68.105*** and ***ping 127.0.0.1***, all communications were denied proving my expectations right as shown below

```
student@serverA:~$ ping 74.125.68.105
PING 74.125.68.105 (74.125.68.105) 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
^C
--- 74.125.68.105 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8190ms

student@serverA:~$ ping 127.0.0.1
PING 127.0.0.1 (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
^C
--- 127.0.0.1 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6133ms
```

Task 16: Logging DROPPED packets

For this task, I uncommented the various sections of firewall.sh as stated in the instructions, saved the changes and executed the script. Within a new terminal, I ran "***sudo tail -f /var/log/kern.log***". This is to provide a logging of the communication. On a different terminal still on the guest, I ping 127.0.0.1 and observed the logging of the ping 127.0.0.1 command.

I think these lines

$IPT -t filter -P INPUT DROP

$TIP -t filter -P OUTPUT DROP

on the script is blocking my ping.

Task 17: Enable traffic from loopback interface

For this task, I added firewall rules that will enable traffic to and from the loopback interface. The added rules are;

iptables -A OUTPUT -o lo -j ACCEPT

iptables -A INPUT -i lo -j ACCEPT

Indeed, these rules re-enabled traffic to and from the loopback interface as can be seen on this screenshot of a new ping of 127.0.0.1

```
student@serverA:~$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.036 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.111 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.089 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.102 ms
^C
--- 127.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5083ms
rtt min/avg/max/mdev = 0.036/0.075/0.111/0.030 ms
```

Task 18: Allow Server A to ping the other interfaces

The task here is to add firewall rules that allow ping traffic initiated from Server A(allow outgoing ICMP Echo Request and incoming ICMP Echo Reply messages).In other words,we are allowing ping from inside to outside. To complete the task, the following firewall rules were added to the “firewall.sh” file.

iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

Indeed, these rules allowed ping from inside to outside as can be seen on this screenshot of ping 74.125.68.105

```
student@serverA:~$ ping 74.125.68.105
PING 74.125.68.105 (74.125.68.105) 56(84) bytes of data.
64 bytes from 74.125.68.105: icmp_seq=1 ttl=63 time=454 ms
64 bytes from 74.125.68.105: icmp_seq=2 ttl=63 time=472 ms
64 bytes from 74.125.68.105: icmp_seq=3 ttl=63 time=494 ms
64 bytes from 74.125.68.105: icmp_seq=4 ttl=63 time=516 ms
64 bytes from 74.125.68.105: icmp_seq=5 ttl=63 time=523 ms
64 bytes from 74.125.68.105: icmp_seq=6 ttl=63 time=440 ms
^C
--- 74.125.68.105 ping statistics ---
7 packets transmitted, 6 received, 14% packet loss, time 6011ms
rtt min/avg/max/mdev = 440.141/483.615/523.534/30.580 ms
```

Task 19: Allow Server A to ping all hosts

For this task, firewall rules are added to allow Server A to lookup names with help of the DNS server configured in "/etc/network/interfaces". To accomplish that, I used the following rules

iptables -A OUTPUT -d 10.0.98.3 -p udp --dport 53 -j ACCEPT
iptables -A INPUT -s 10.0.98.3 -p udp --sport 53 -j ACCEPT
iptables -A OUTPUT -d 10.0.98.3 -p tcp --dport 53 -j ACCEPT
iptables -A INPUT -s 10.0.98.3 -p tcp --sport 53 -j ACCEPT

With the above rules, using DNS names for example,
"ping www.google.com" is now possible

```
student@serverA:~$ ping www.google.com
PING www.google.com (142.250.27.105) 56(84) bytes of data.
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=1 ttl=63 time=147 ms
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=2 ttl=63 time=137 ms
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=3 ttl=63 time=141 ms
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=4 ttl=63 time=142 ms
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=5 ttl=63 time=153 ms
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=6 ttl=63 time=193 ms
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=7 ttl=63 time=165 ms
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=8 ttl=63 time=155 ms
64 bytes from 142.250.27.105 (142.250.27.105): icmp_seq=9 ttl=63 time=189 ms
^C
--- www.google.com ping statistics ---
10 packets transmitted, 9 received, 10% packet loss, time 9019ms
rtt min/avg/max/mdev = 137.748/158.682/193.778/19.525 ms
```

Lastly, I Modify the ICMP rules to allow sending IMCP Echo Request to any server and receiving the corresponding ICMP Echo Replies with the follwing rules to add the ones on Task 18

iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

Task 20: Enable stateful firewall

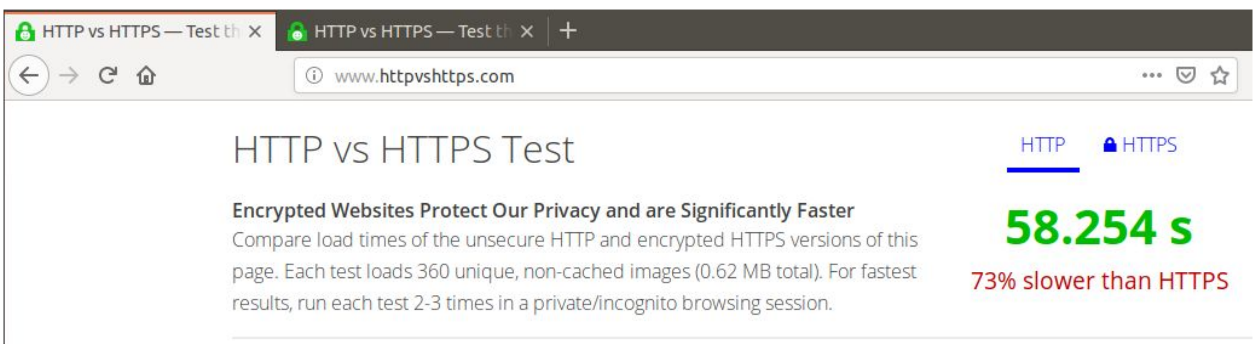
The task here is to establish outgoing TCP connections over HTTP and HTTPS from guest system to the outside world while at the same time blocking incoming connections from the outside world.The accomplish this, I added the following rules within “firewall.sh”

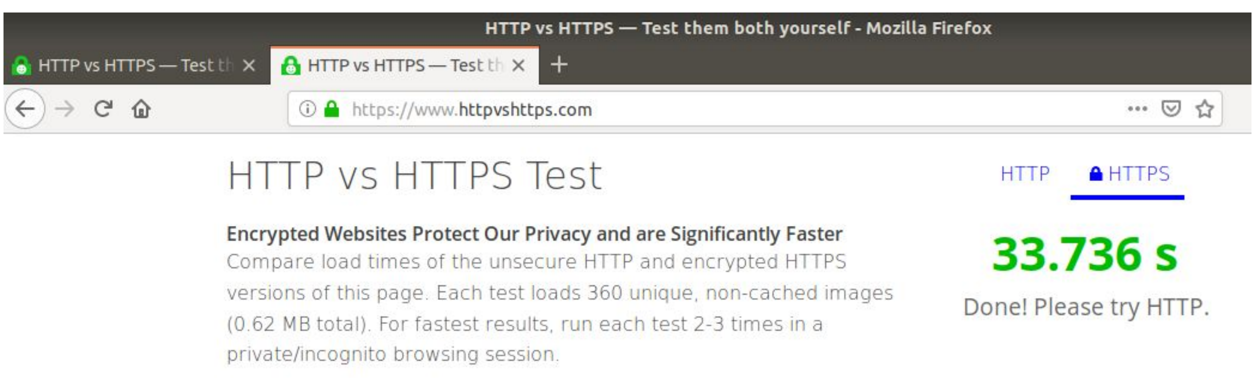
iptables -t filter -A OUTPUT -p tcp -j ACCEPT
iptables -t filter -A INPUT -p tcp -m conntrack --ctstate ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -p tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -t filter -A INPUT -p tcp --sport 80 -m conntrack --ctstate ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -p tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -t filter -A INPUT -p tcp --sport 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT

With the above rules, was able to once more browse HTTP and HTTPS within the guest

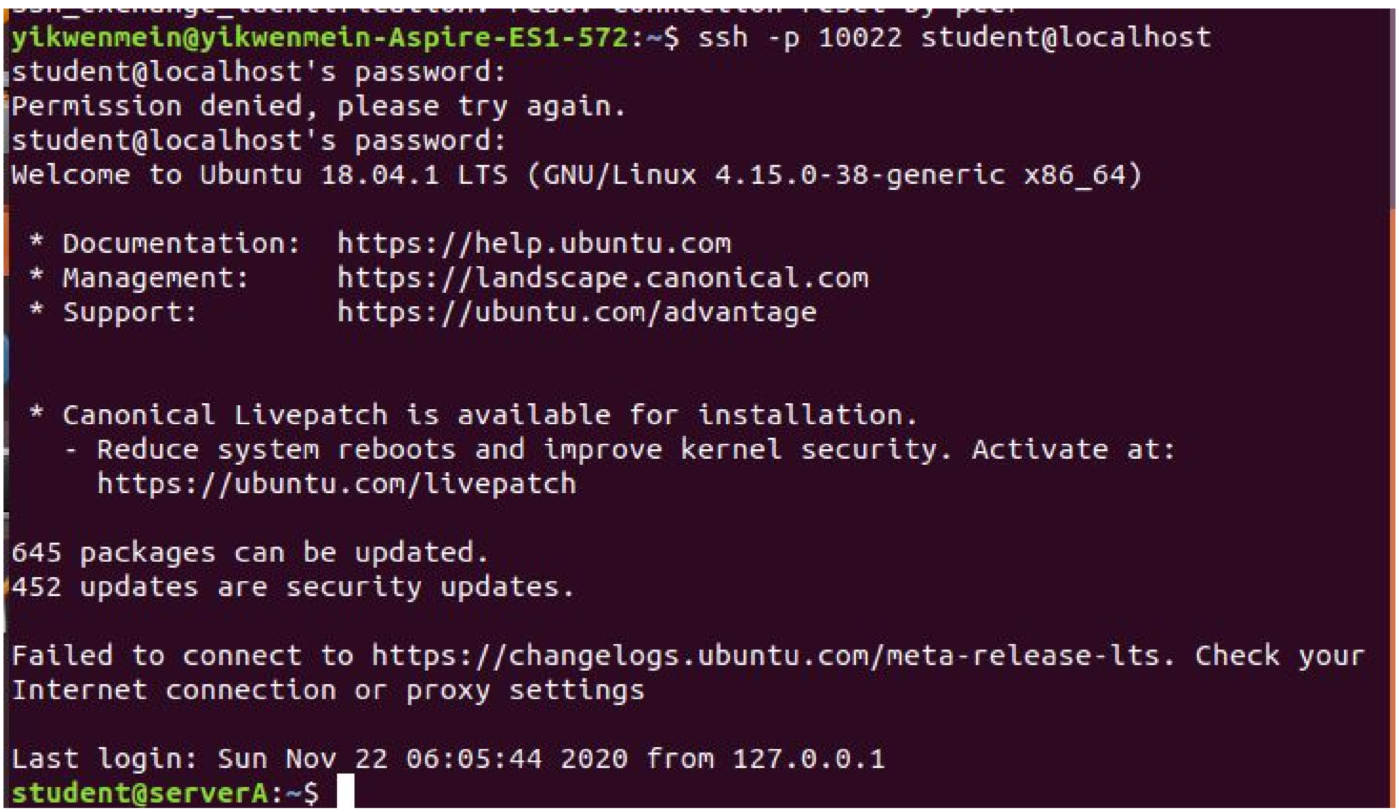




Task 21: Enable SSH and HTTPS content from apache2 server for web browser on host

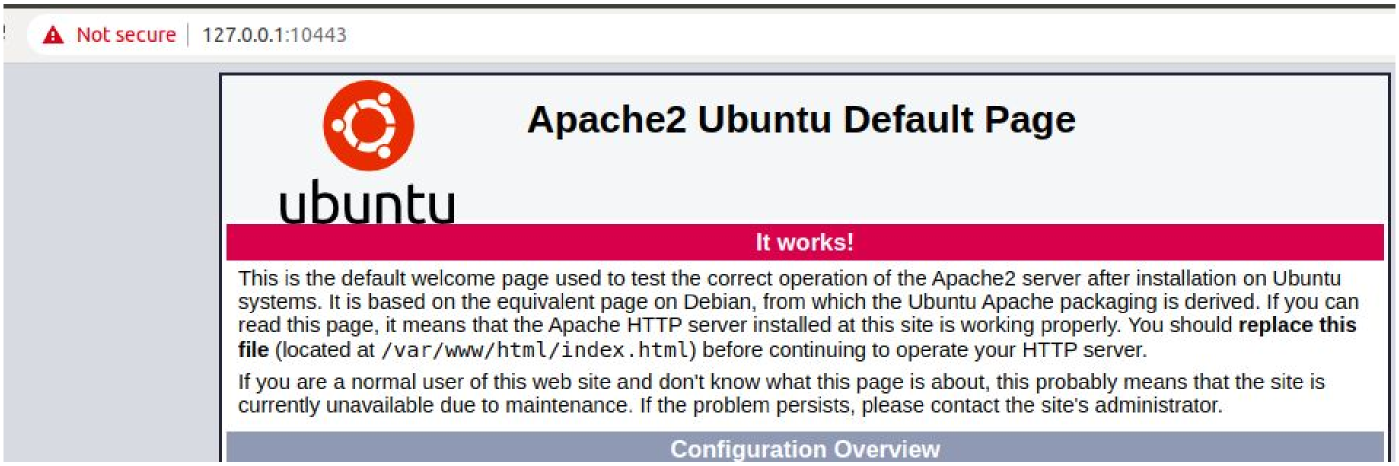
For this task, the following iptables rules were added to “firewalls.sh” to enable a user to ssh into Server A from the host

iptables -t filter -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT



Lastly, the following iptable rules were also added to enable a web browser on the host to view content served by the apache2 server over HTTPS

iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT

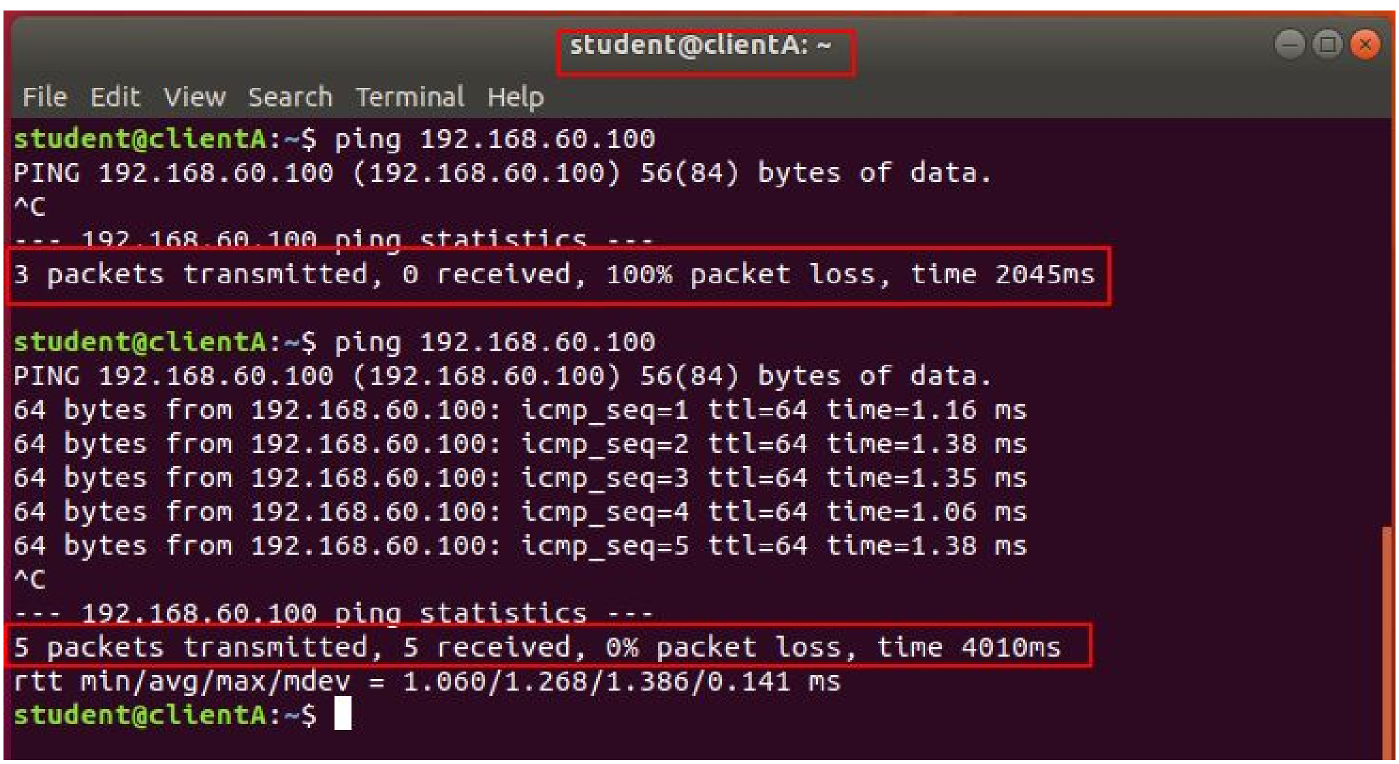


Task 22: Ping Server A from Client A

For this task, I added the following firewall rules to “firewall.sh” within server A and then pinging server A from client A was successful.

iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

The following screenshots shows the results of pinging server A from client A before and after adding the above rules

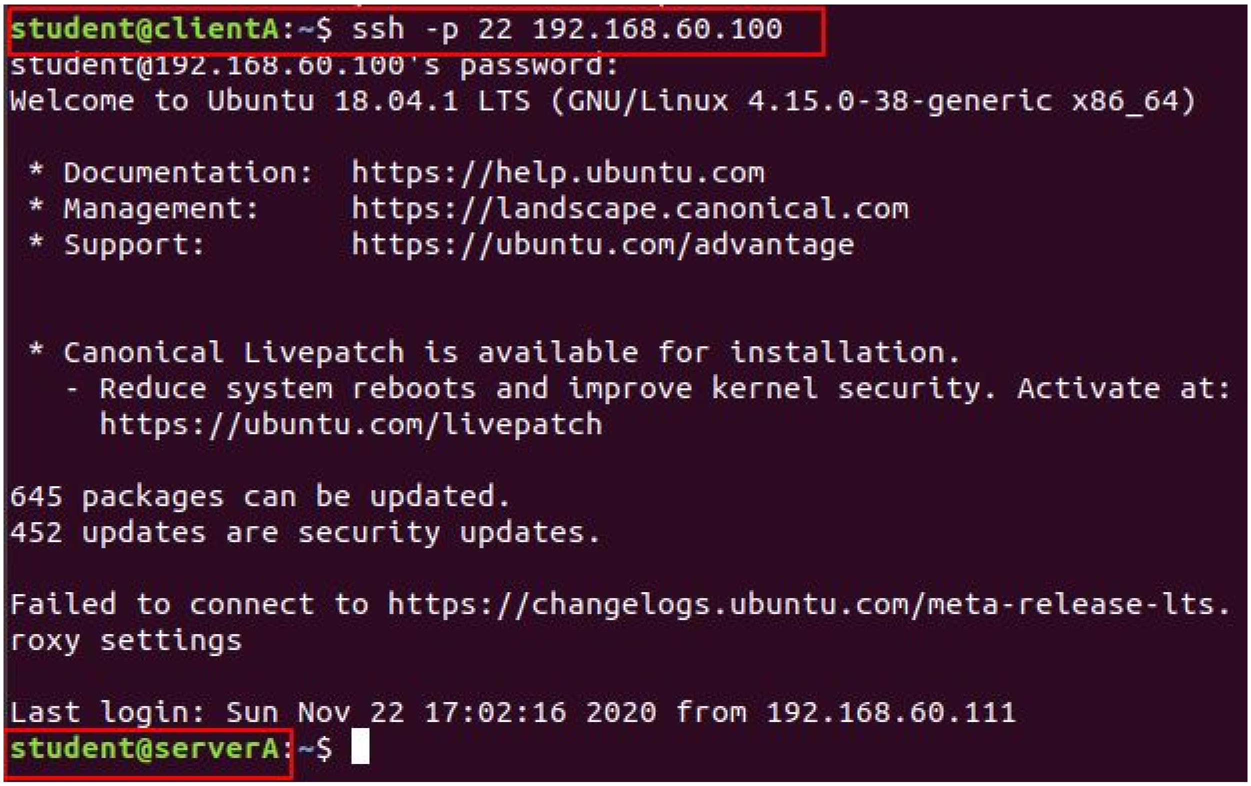


Task 23: SSH from Client A to Server A

For this task, I added the following firewall rules to “firewall.sh” within server A to enable ssh from client A to server A

iptables -A OUTPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT

SSH from client A to server B is as seen below



```
student@clientA:~$ ssh -p 22 192.168.60.100
student@192.168.60.100's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-38-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

645 packages can be updated.
452 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts.
roxy settings

Last login: Sun Nov 22 17:02:16 2020 from 192.168.60.111
student@serverA:~$
```

Task 24: Add gateway and DNS server to Client A

For this task, i used the following command ***sudo gedit /etc/network/interfaces*** within client A and edited /etc/network/interfaces by adding gateway “192.168.60.100”. “/etc/resolv.conf” already has 10.0.98.3 is listed as DNS server(nameserver 10.0.98.3)

Task 25: Enable IP forwarding on Server A

As instructed in the lab manual, I ran the following commands within server A terminal

sudo sysctl -w net.ipv4.ip_forward=1
sudo sysctl -p

The first command turns on IP forwarding and the second command applies the change to the running kernel

Task 26: Change iptables to forward packets

For this task, the following rules were added to “firewall.sh” of server A to enable packet forwarding

$IPT -t filter -A FORWARD -i $HIF -j ACCEPT

$IPT -t filter -A FORWARD -i $NIF -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

Task 27: Enable SNAT on Server A

As instructed in the lab manual, I added the following rule within server A “firewall.sh”

$IPT -t nat -A POSTROUTING -j SNAT -o $NIF --to $NIP

With the addition of this rule, we can access the Internet from client A