LAB – 1 Networking and Firewalls

Version 2

By
Nasirali Kovvuru
nako17@student.bth.se
P No 960521-3231

Changes in this Version according to comments:

Comments:

Task-1:

- 10.0.2.0/24 is not an IP address.

Changes done: Yes, that is not IP address, it is Subnet Mask and I have made corresponding changes in the task.

Task-2:

- Not a “Randomly assigned Ip by the Virtualbox”. It is configured to get that. Check the configuration file.

Changes done: Yes, I have checked to Configuration file and made changes in the task.

Task-5:

- “193.11.185.57” or “193.11.185.75”?
- What is the gateway that your host is able to reach to?

Changes done: after referring to screenshot, I found my mistake and changed it to 193.11.185.75. The host gateway is also changes after referring to screenshot, the reason for my mistake will be parallax error while writing the report.

Task-6:

- “gateway” doesn’t assign ip addresses. Correct your statement.

Changes done: from the above comment I found wrong in my statement and corrected.

Task 18, 23:

- The problem with that implementation is, if the IP address is changed, then the rules won’t apply anymore. Think from ports perspective.

Changes done: yes sir, I did a mistake and corresponding changes are done in those tasks by thinking on ports perspective.

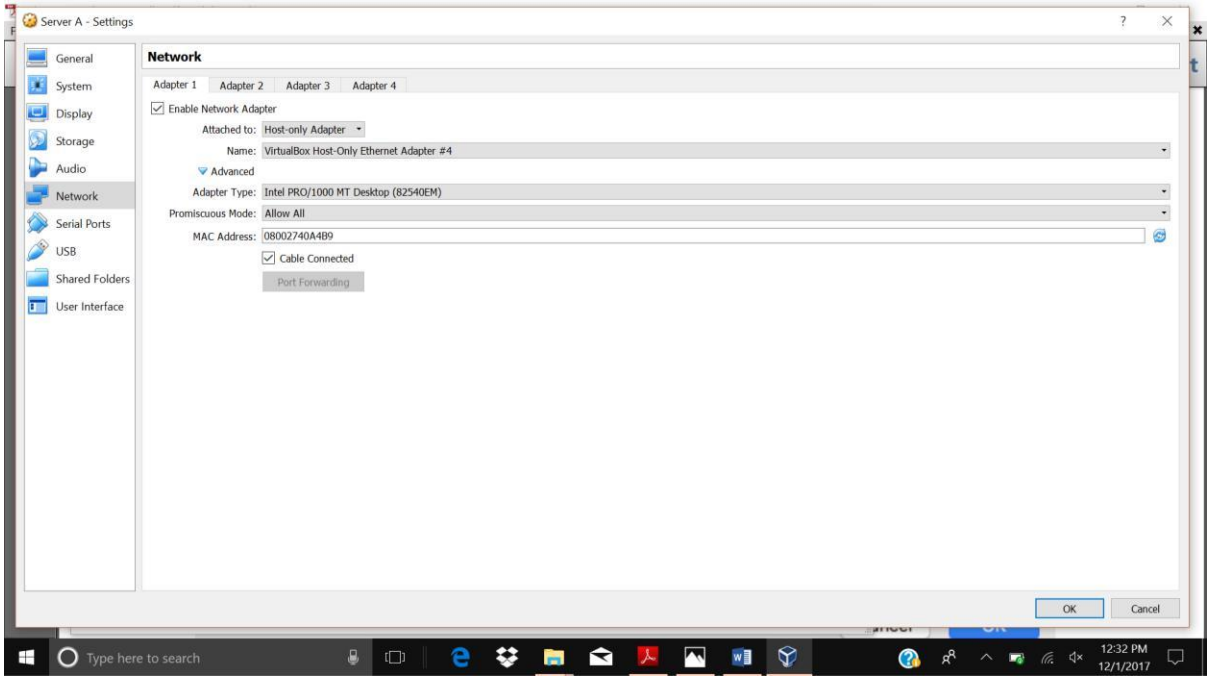
<u>Task 1:</u> To know IP and Mac addresses of each Network card.

Server A

Adapter 1 → ip:192.168.60.1 → Mac:08002740A4B9

Adapter 2 → Subnet:10.0.2.0/24 → Mac:080027868510

Adapter 3 → ip:192.168.70.1 → Mac:080027B666E4



<u>Task2:</u> To know the name of the interface and type of the interface by using Mac addresses from Task1.

Adapter 1 → Host only → ip:192.168.60.1 → interface: enpos3 → Assigned IP by the Virtualbox:192.168.60.100

Adapter 2 → NAT → Subnet:10.0.2.0/24 → interface: enpos9 →supports DHCP:10.0.98.100

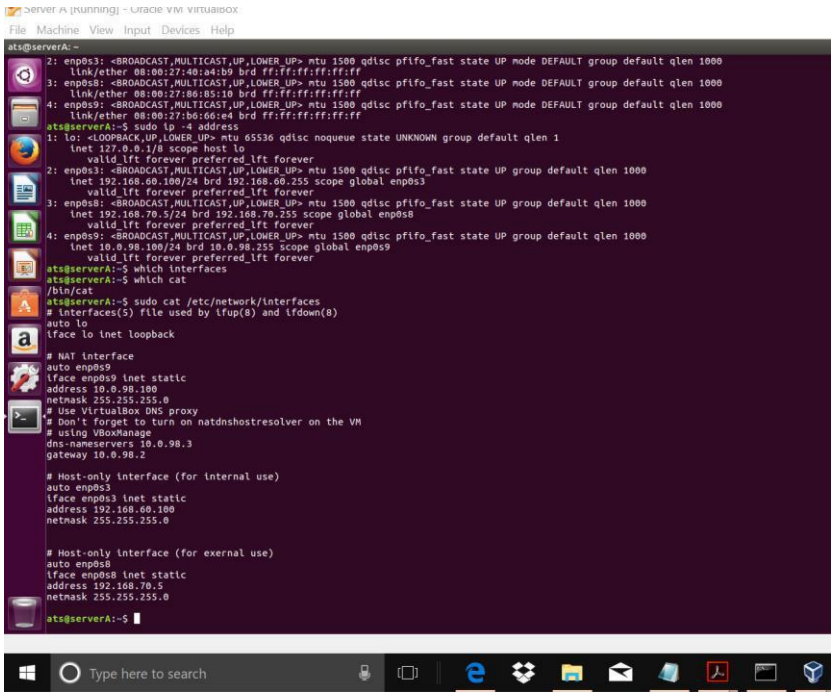
Adapter 3 → Host only → ip:192.168.70.1 → interface: enpos8 →Assigned Ip by the Virtualbox:192.168.70.5

<u>Task3:</u>

#sudo cat /etc/network/interfaces

- ➔ #NAT interface; enpos9; ip:10.0.98.100 255.255.255.0
- ➔ dns-nameservers 10.0.98.3; **gateway:10.0.98.2**

➔ NAT interface allows your virtual machine to communicate to outer world, over the NAT interface have their source ip address replaced with gateway
➔ DNS name server (10.0.98.3)
➔ When virtual box receives DNS requests on this address from the virtual machine, it passes them on to DNS proxy which will convert that IP to corresponding IP address by seeing the Table.
➔ When packets are sent to Gateway 10.0.98.2, here virtual box uses its own **NAT engine** to do network address translation. And asks the host to forward the packets to the destination



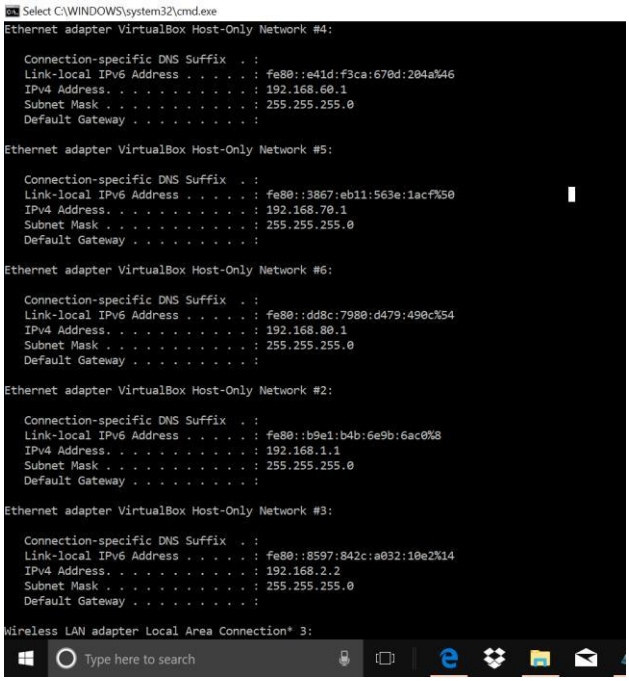
Task4:

➔ By seeing the result from command ($sudo ifconfig -a), we can see that there are different ip's which are not configured at host-only network
➔ In my case, I have configured host only networks and NAT networks with 192.168.60.1(255.255.255.0), 192.168.70.1(255.255.255.0), 10.0.2.0/24(255.255.255.0) respectively but I observed that all the addresses are varied after switching on the virtual machine.
➔ From this result, by moving to internal virtual guest O.S settings, I came to conclusion that, our **virtual box is assigning different ip addresses to different guests to make it possible to use the same card(adapter) by different guests**.
➔ Very important observation is that, **Mac address assignment is constant, irrespective of dynamic IP address assignment,** so

from the above observation I practically came to conclusion that **observing Mac address instead host-only network IP address is preferable**.

➔ And, **from my base machine command prompt I found that Host-only network IP addresses are same as I configured at first**.

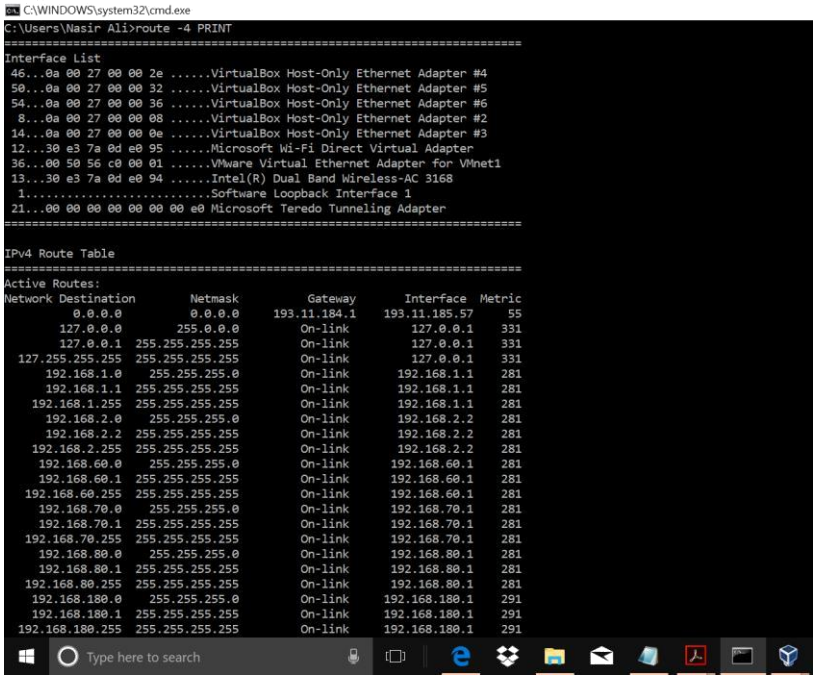
➔ Output:
enpos3 is interface card is connected to **host only adapter#4**
enpos9 interface card is connected to **host only adapter#5**
enpos3 interface card is connected to **NAT Adapter**



<u>TASK 5:</u> Routing table in the host Operating system

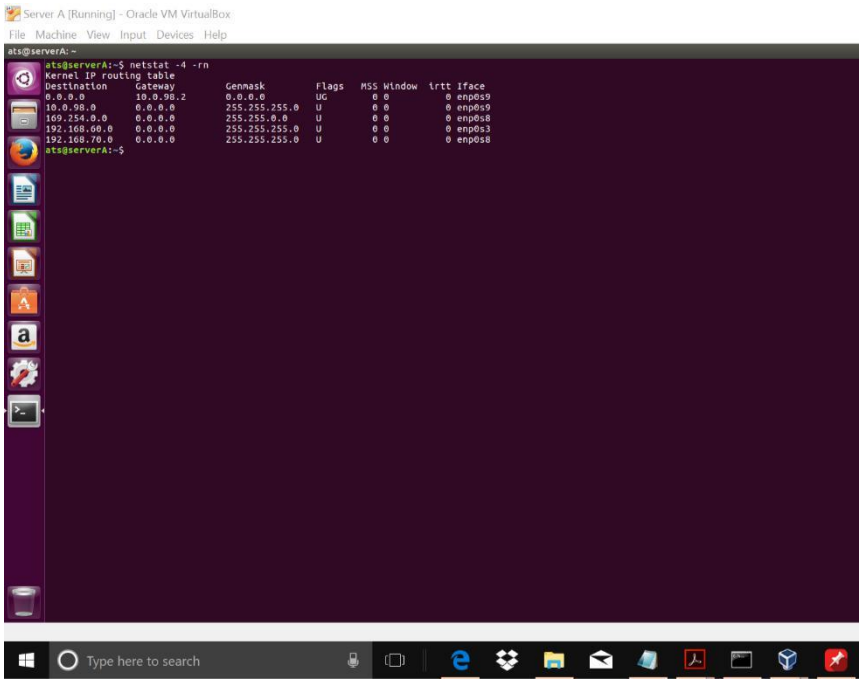
➔ in my case, I can reach the default gateway through Wifi LAN Adapter from the Host os(Windows) is through its ip address 193.11.184.1, and the interface is 193.11.185.75

➔ In the output from command (#route -4 PRINT), I got it as **193.11.184.1**.



TASK 6: Routing table in the guest Operating system

➔ From the command ($route -n), I found my default gateway (DNS proxy) at 10.0.98.2 to reach that adapter through the interface **enpos9**.



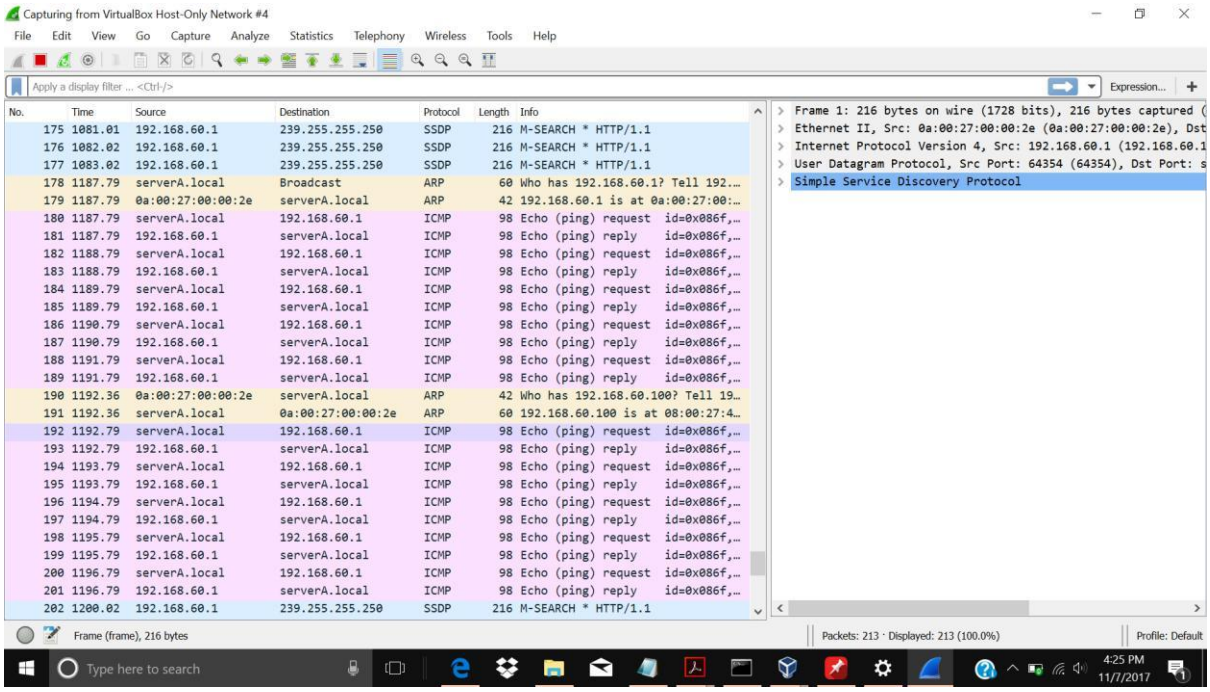
➔ From the above result I can conclude my exploration that, NAT server acts like a **switch to all the virtual machines** to communicate, but whereas to send the packets to outer world it should make use of **Proxy server as a geteway**

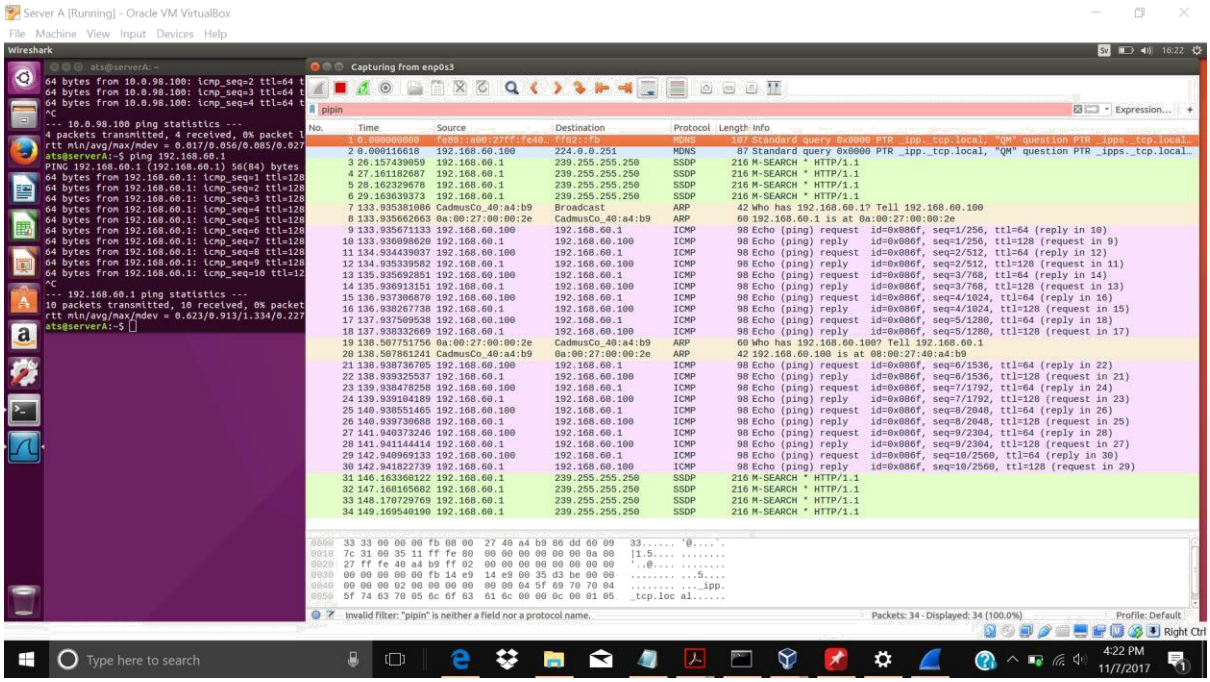
➔ We can conclude from the above screenshot that through host-only interface the guest OS can reach the default gateway for host OS.

➔ Here, **we can reach to gateway via NAT interface only, because where our DNS server is configured.**

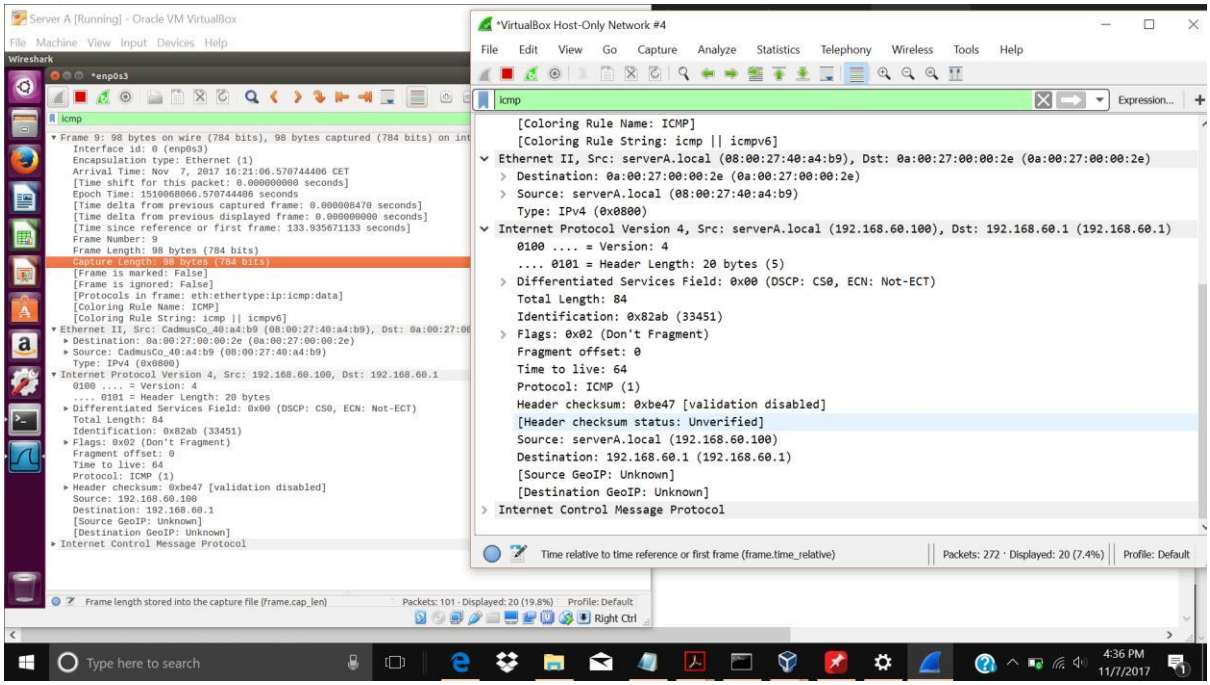
➔ Wireshark Traffic:

Packets are captured at both guest and host PC's for host-only interface 192.168.60.100.

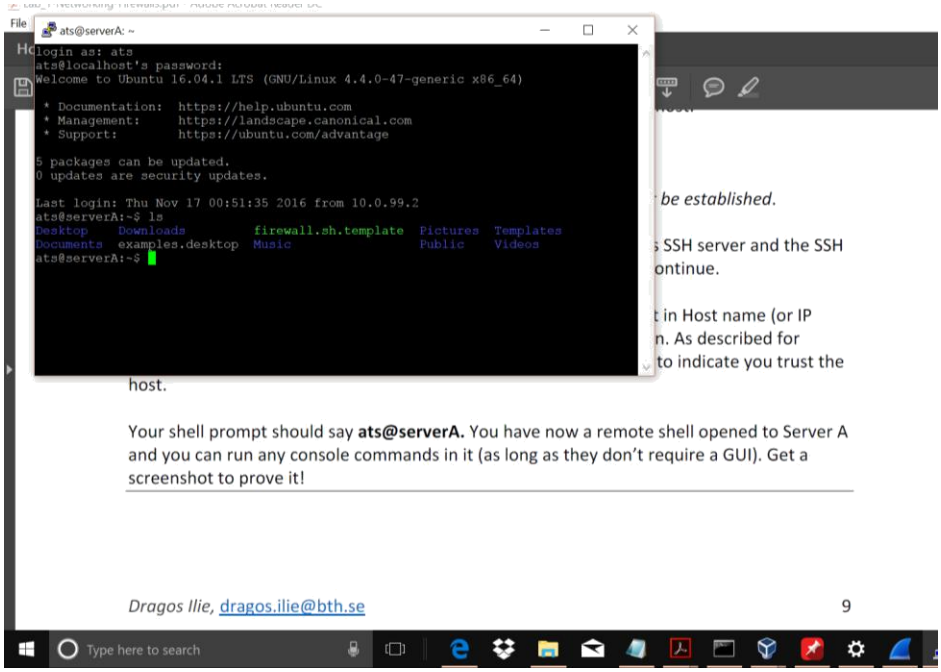




TASK 7: we can see that the Wireshark ICMP traffic is same at both ends.

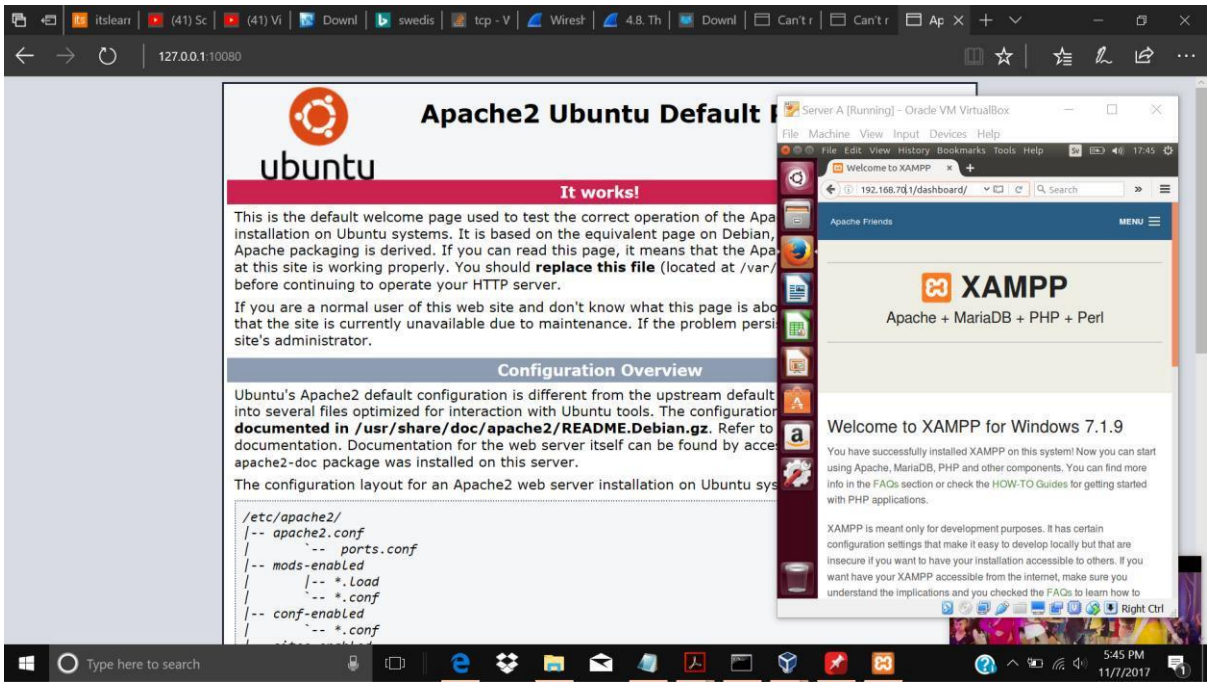


Task 8: SSH configuration



Task9: add forwarding rules for http and https in virtual box.

For http → 10080, and for https → 10443, these port numbers are used



Task 10: Default firewall policy and rules.

$iptables tool is used to find the default policies, the following screenshot shows them. By default all the filters and chains are in accept list.

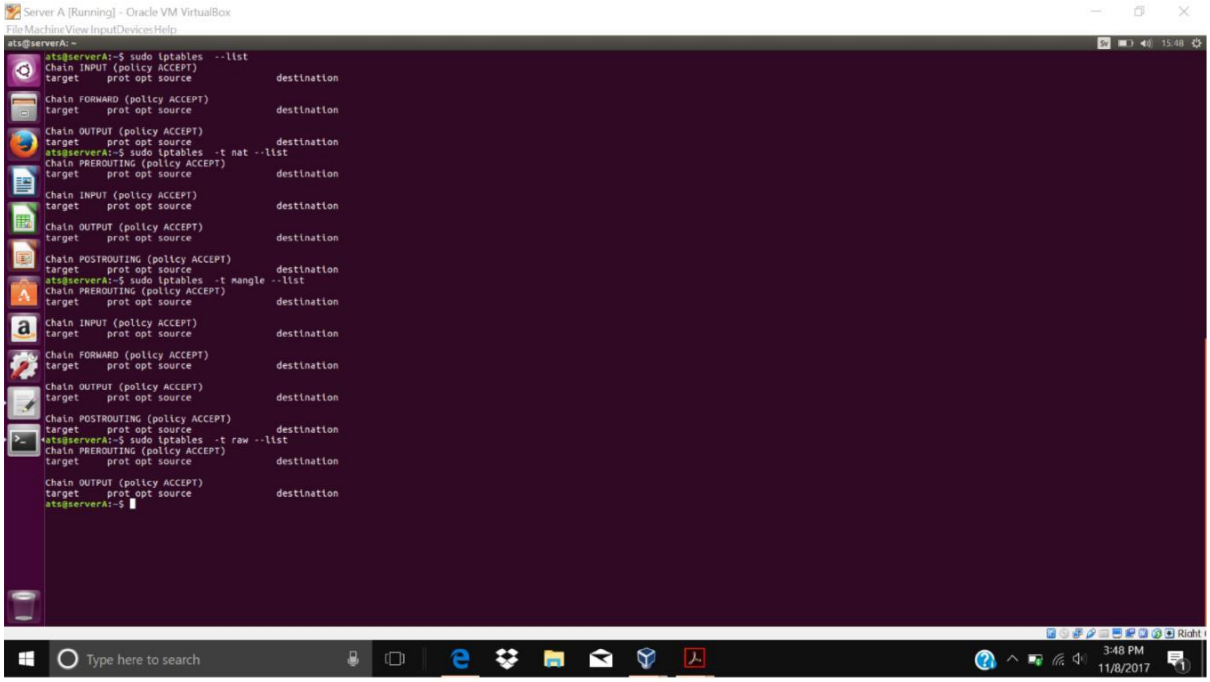
The following commands helps in listing the default firewall rules:

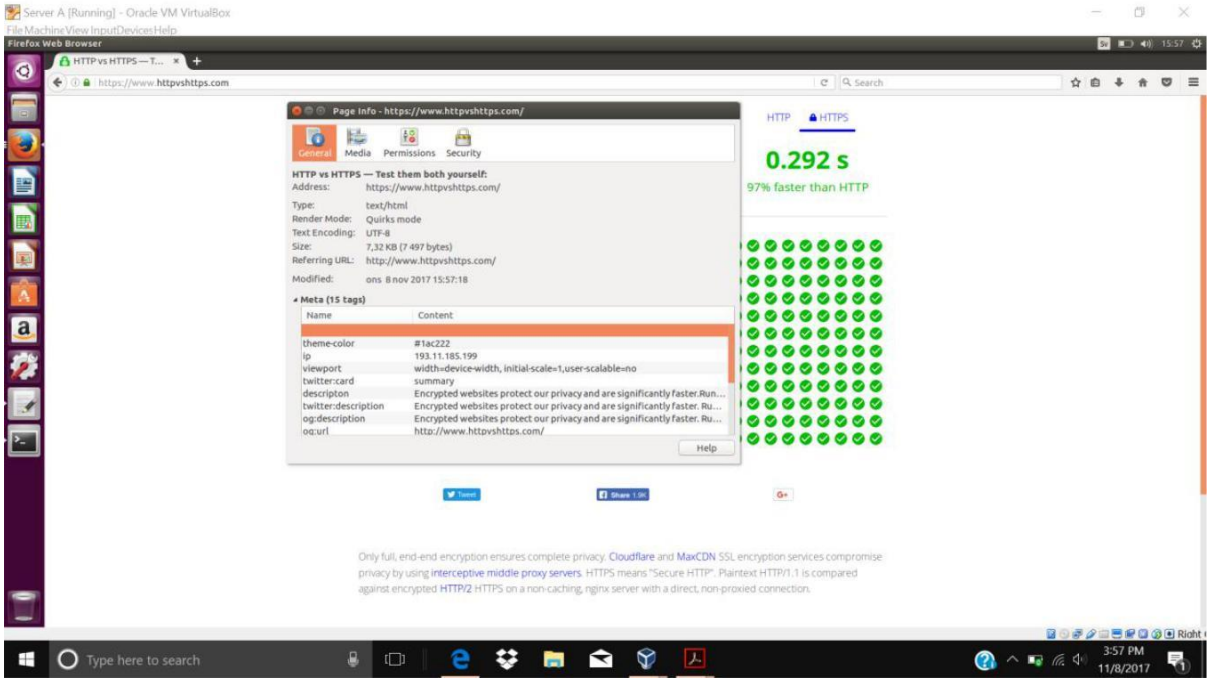
$sudo iptables - -list → by default it shows filter table policies.

$sudo iptables -t nat - -list → this command shows nat table policies.

$sudo iptables -t mangle - -list → this command shows mangle table policies.

$sudo iptables -t raw - -list → this command shows raw table policies.



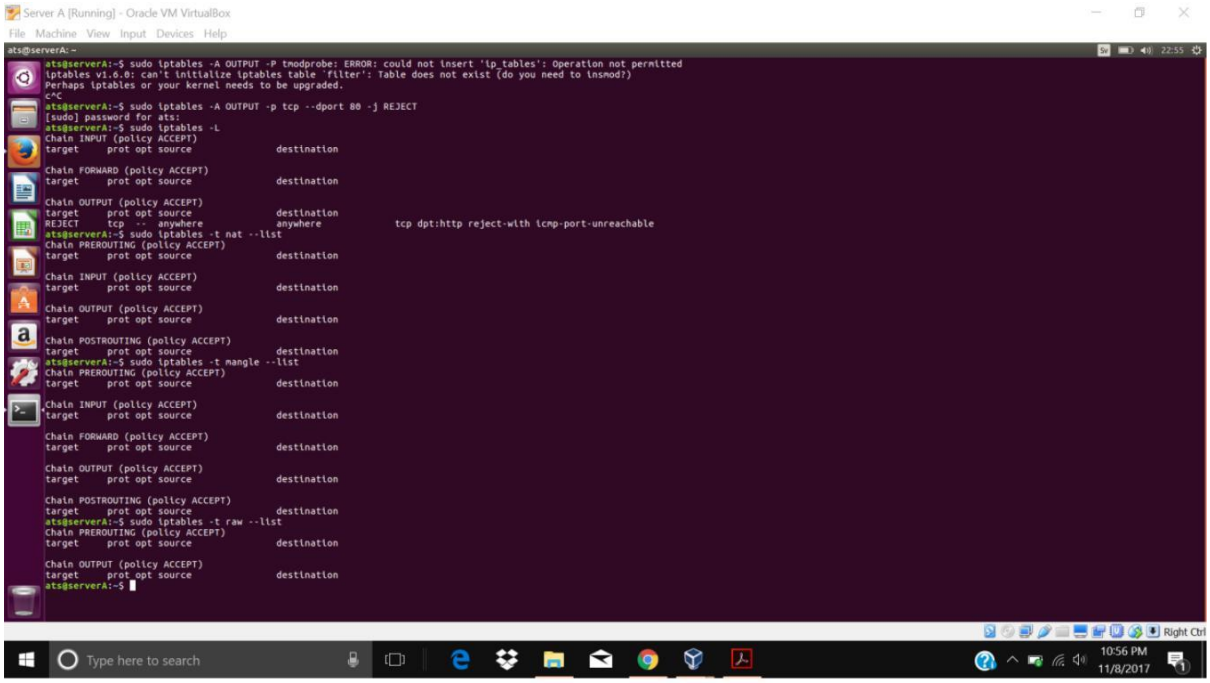


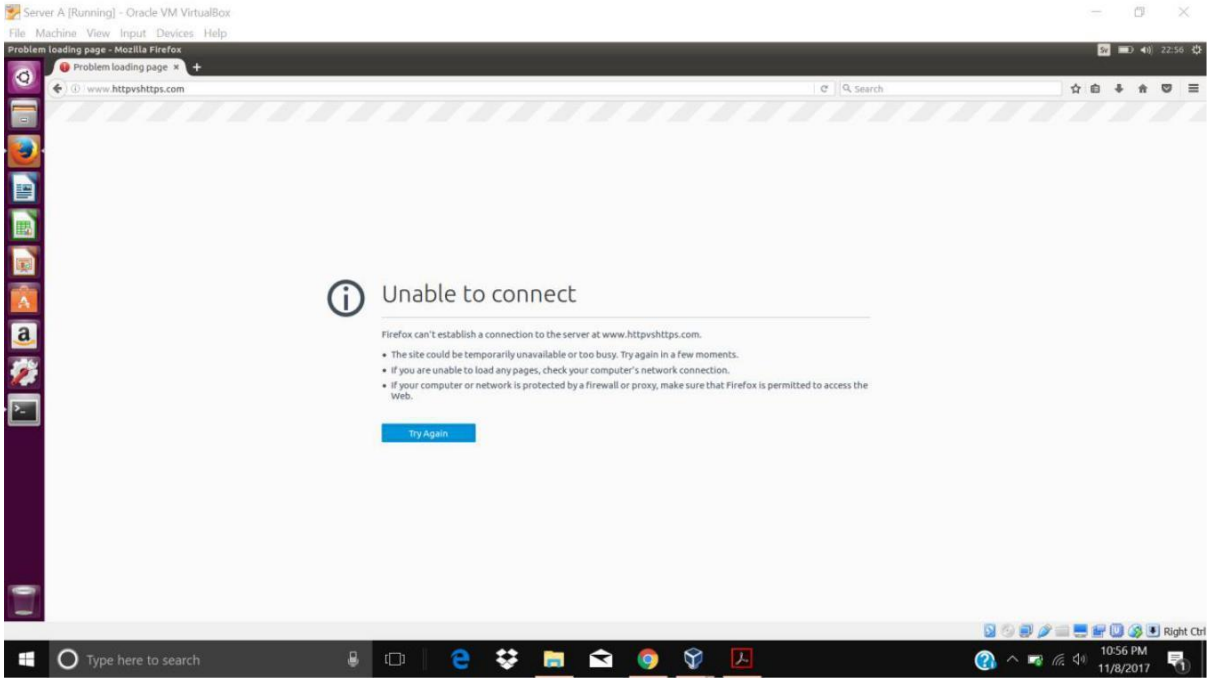
<u>Task 11:</u> blocking http traffic in the guest system.

$sudo iptables -A INPUT -p tcp –dport 80 -j REJECT

➔ This command rejects all the http browsing.

➔ **Note**: the above command only affects the filter table, output chain; the reason behind is, it is the default table for iptables.





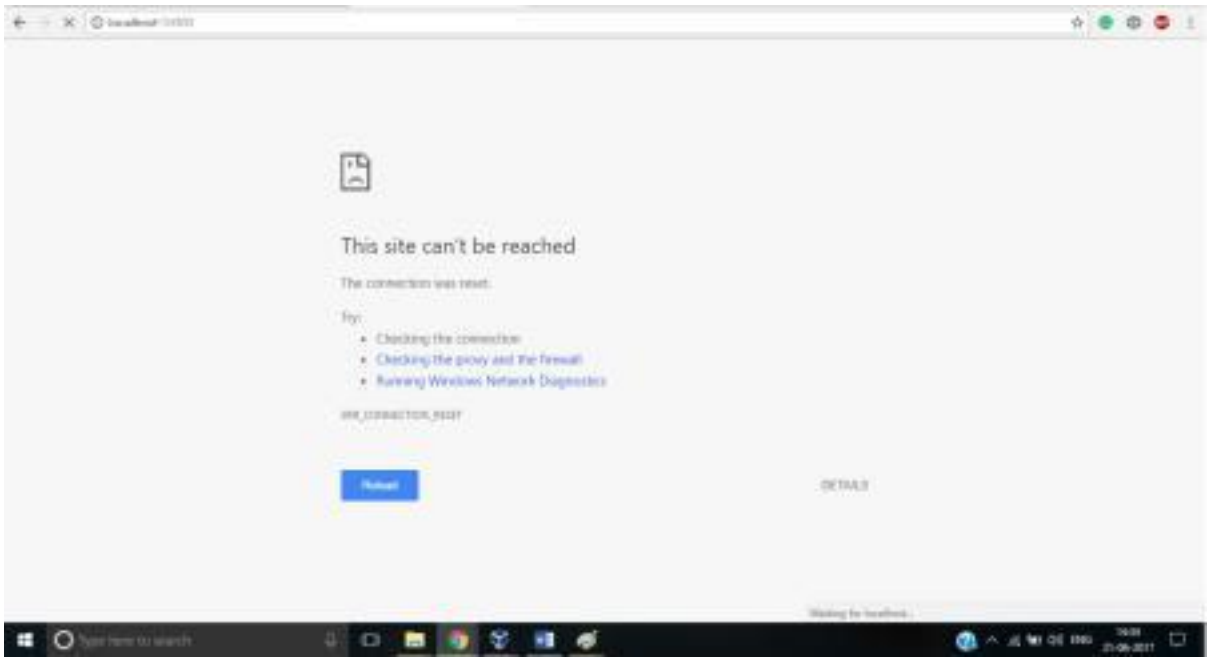
TASK 12: Block Apache web server from serving content over http

➔ I have achieved the above task by rejecting the output traffic from firewall. Or in other way we can reject the traffic going away from

loopback interface as the apache server resides there. as shown in the following screenshot.

➔ $sudo iptables -A OUTPUT -p tcp –-dport 80 -j Reject

➔ (or) $sudo iptables -A OUTPUT -p tcp -s 127.0.0.1/24 –dport 80 -j Reject.



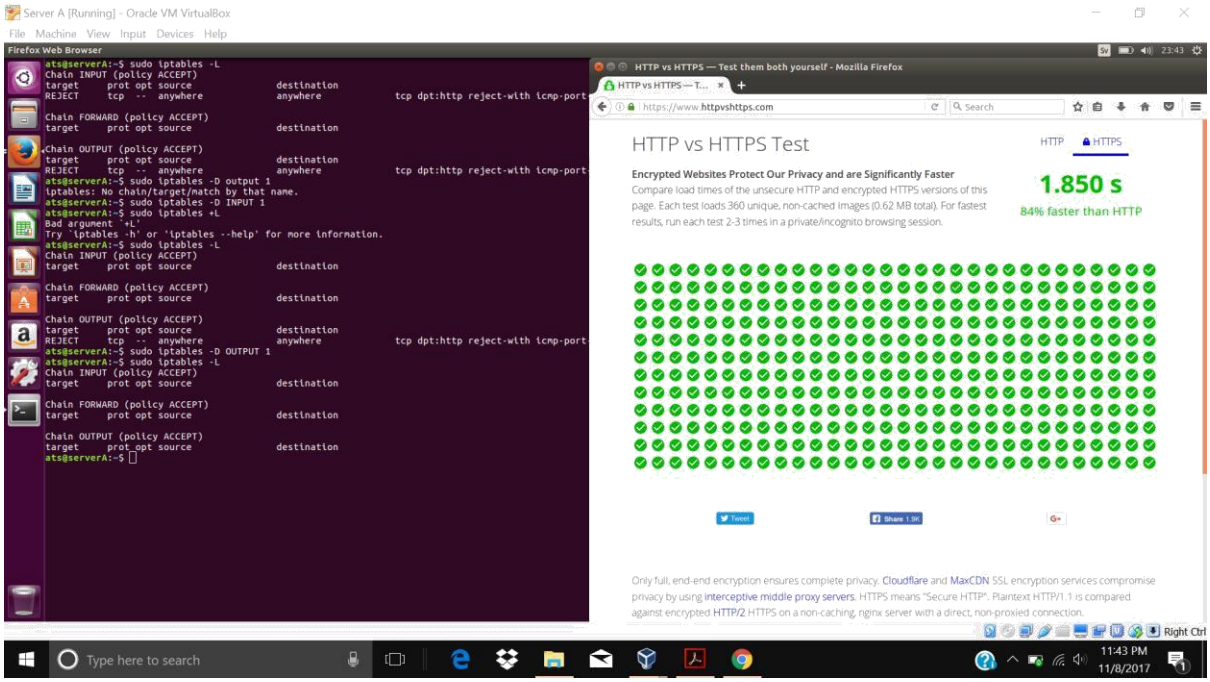
Task 13: **Unblock HTTP browsing in the guest O.S**

➔ Undo tasks 11 and 12

➔ The following commands helps

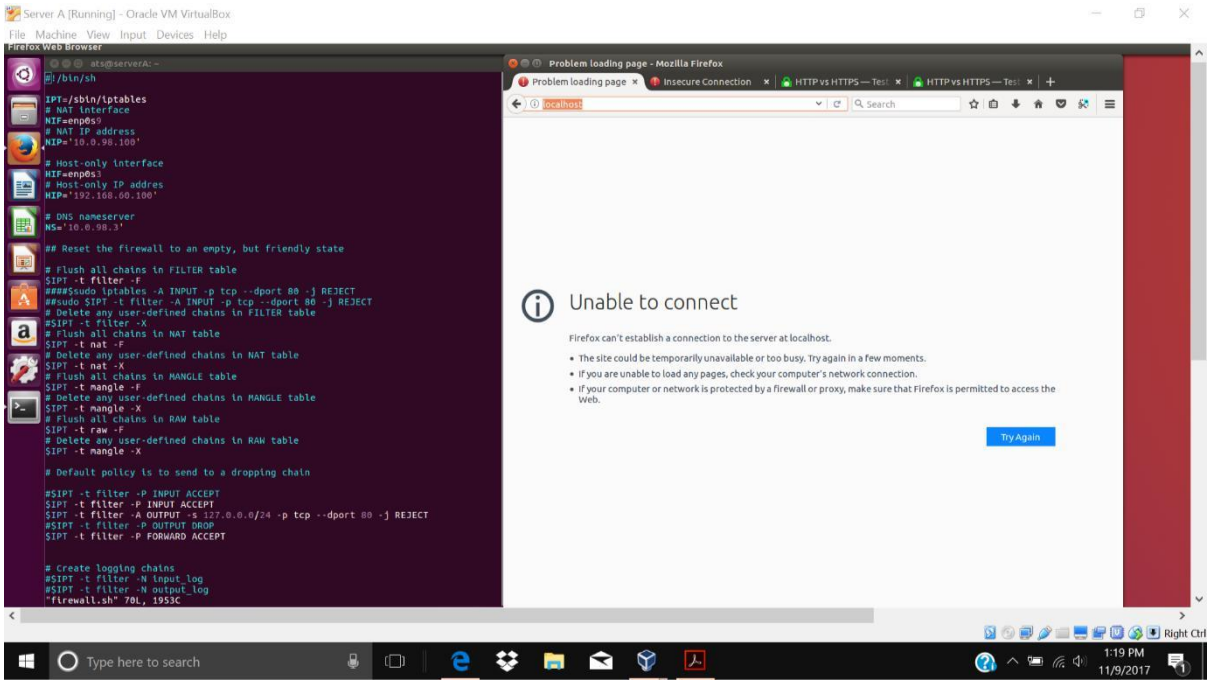
➔ $sudo iptables -D OUTPUT1 → -D to Delete

➔ $sudo iptables -D INPUT1



TASK 14: use firewall.sh to configure the firewall

- ➔ Modify the script to such that guest OS can view http and https services but apache2 service should be blocked.
- ➔ The following screenshot shows such behaviour
- ➔ $ iptables -A OUTPUT -p tcp -s 127.0.0.1/24 –dport 80 -j Reject.



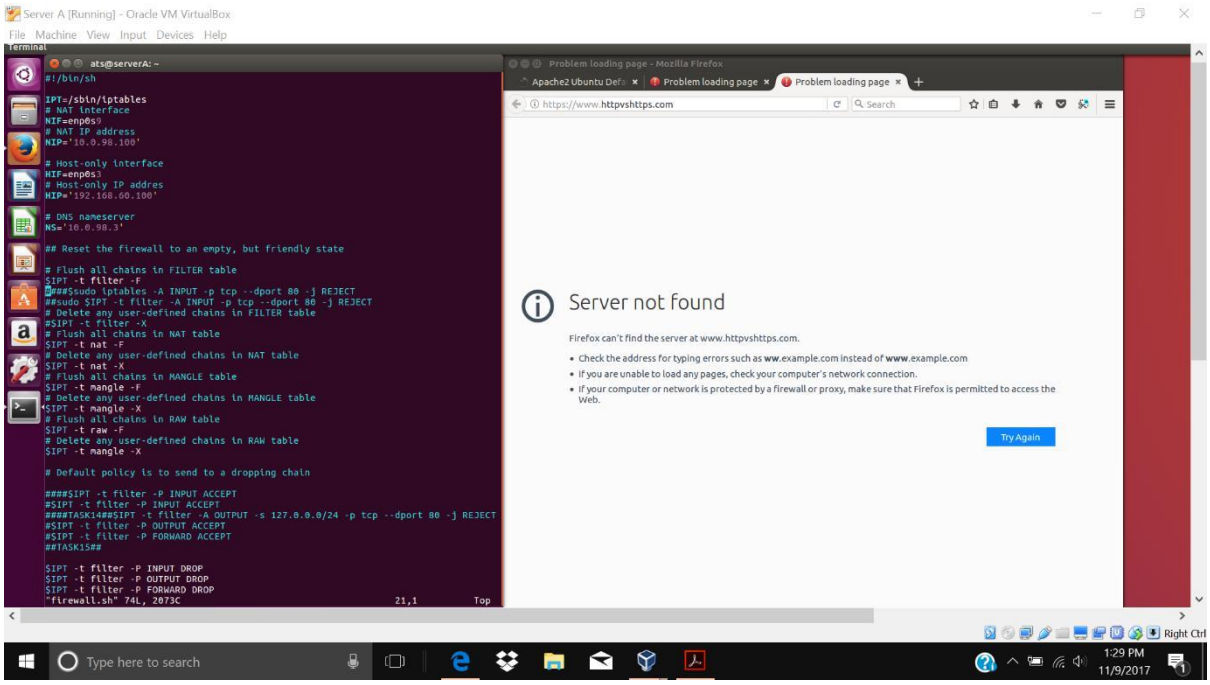
TASK 15: change default firewall policy to

drop →$iptables -t filter -P INPUT DROP

→$iptables -t filter -P OUTPUT

DROP →$iptables -t filter -P

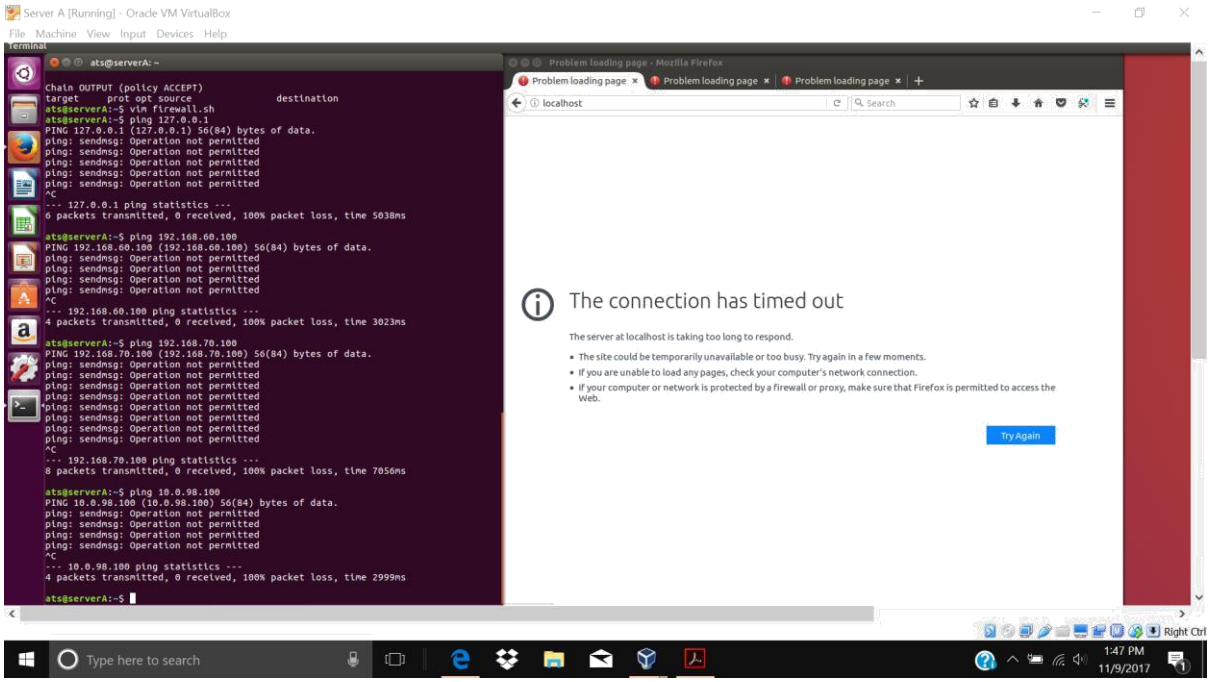
FORWARD DROP



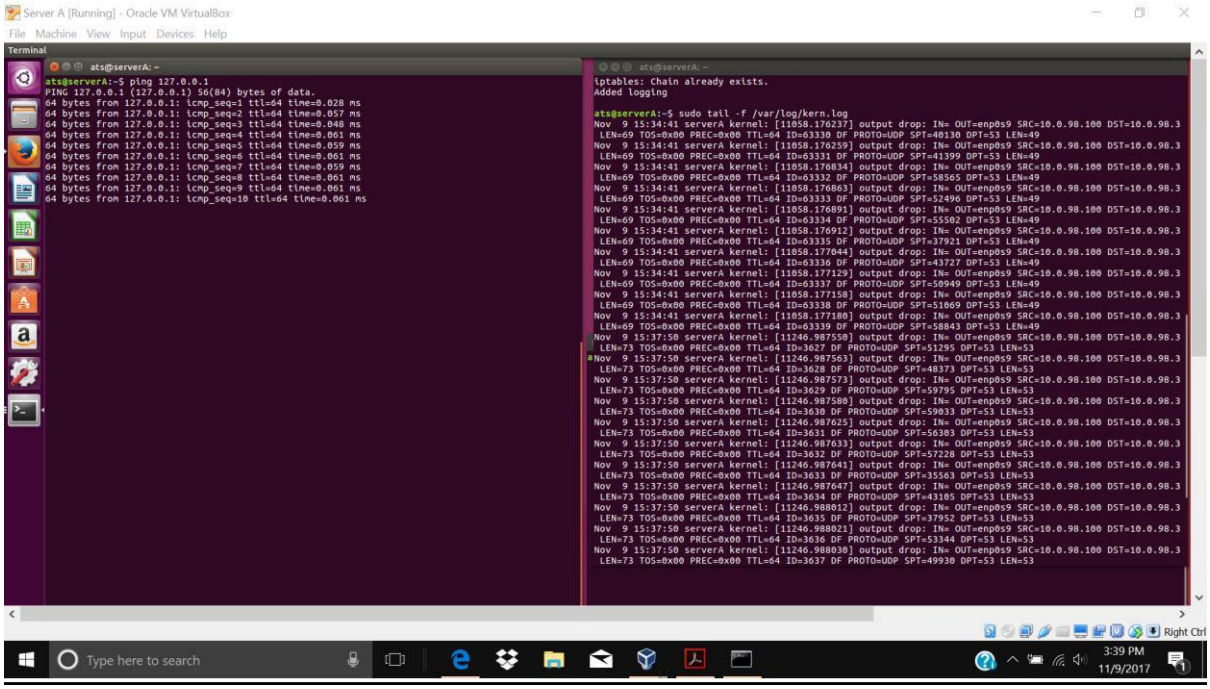
Result → localhost not loaded, http, https are not responding.

Execute the script to install the new policy to ping outside world from the virtual machine, try to ping the loopback interface.

➔ Following rules make the above requirement fulfil.
➔ $ iptables -I OUTPUT -p icmp -j ACCEPT
➔ $iptables -I INPUT -p icmp -j ACCEPT



➔ **Live logs from the kernel is shown below.**



TASK 16: Logging Dropped packets

➔ By executing the command $sudo tail -f var/log/kern.log
➔ And by adding commands: echo $input_log, echo $output_log, echo $forward_log
➔ We can see the kernel log
➔ In the output we can find that out ping messages are blocking
➔ **What do you think is blocking your ping?**

➔ In my case my I am getting some error message as in description screenshot (output drop) and in my case source and destination addresses are in between NAT's DHCP and its DNS nameserver.

➔ The Possible reason behind it, will be; we already know from previous tasks that NAT interface is something out from the virtual machine and it stays on virtualbox hypervisor and the script which tells to DROP all the output packets except ICMP but here other management messages are going in a particular interval gap; so they are blocking by the firewall.

➔ So that packets got dropped.

TASK 17: Enable Traffic from Loopback interface

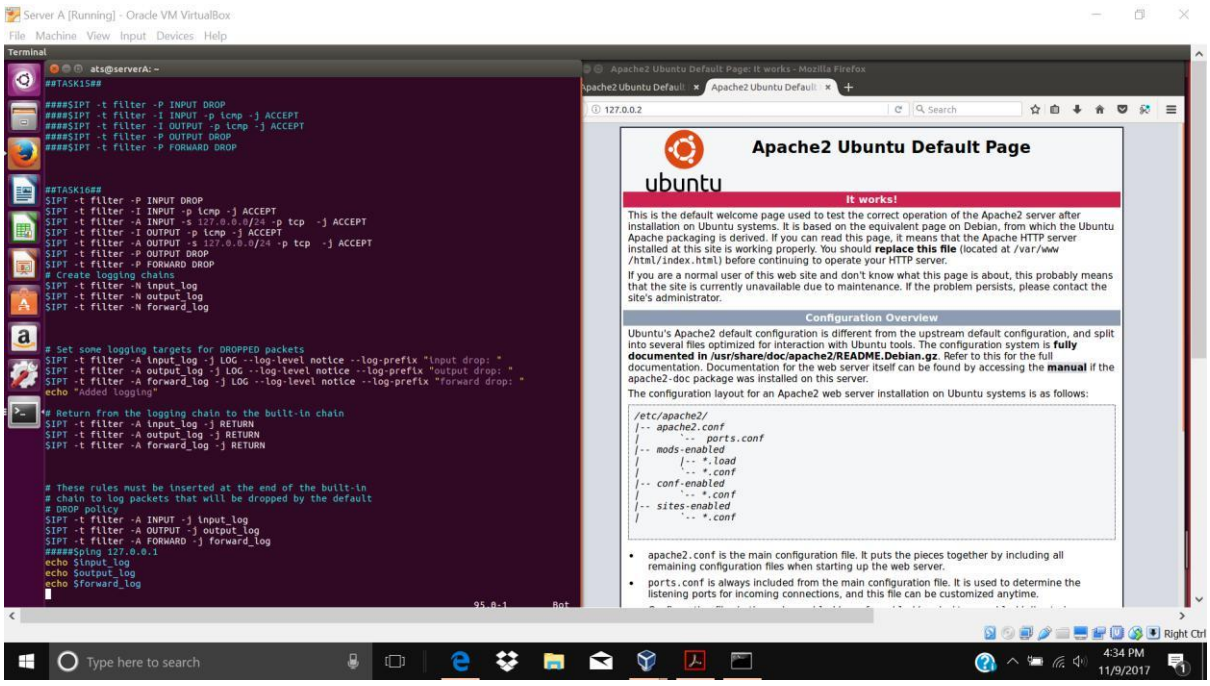
➔ $ iptables -I OUTPUT -s 127.0.0.1/24 -p tcp -j ACCEPT

➔ $iptables -I INPUT -s 127.0.0.1/24 -p tcp -j ACCEPT

➔ The above commands are used to enable only TCP taffic but if you want to enable all types of traffic then following command helps.

➔ $ iptables -I OUTPUT -s 127.0.0.1/24 -j ACCEPT

➔ $iptables -I INPUT -s 127.0.0.1/24 -j ACCEPT



TASK 18: Allow server A to ping the other interfaces.

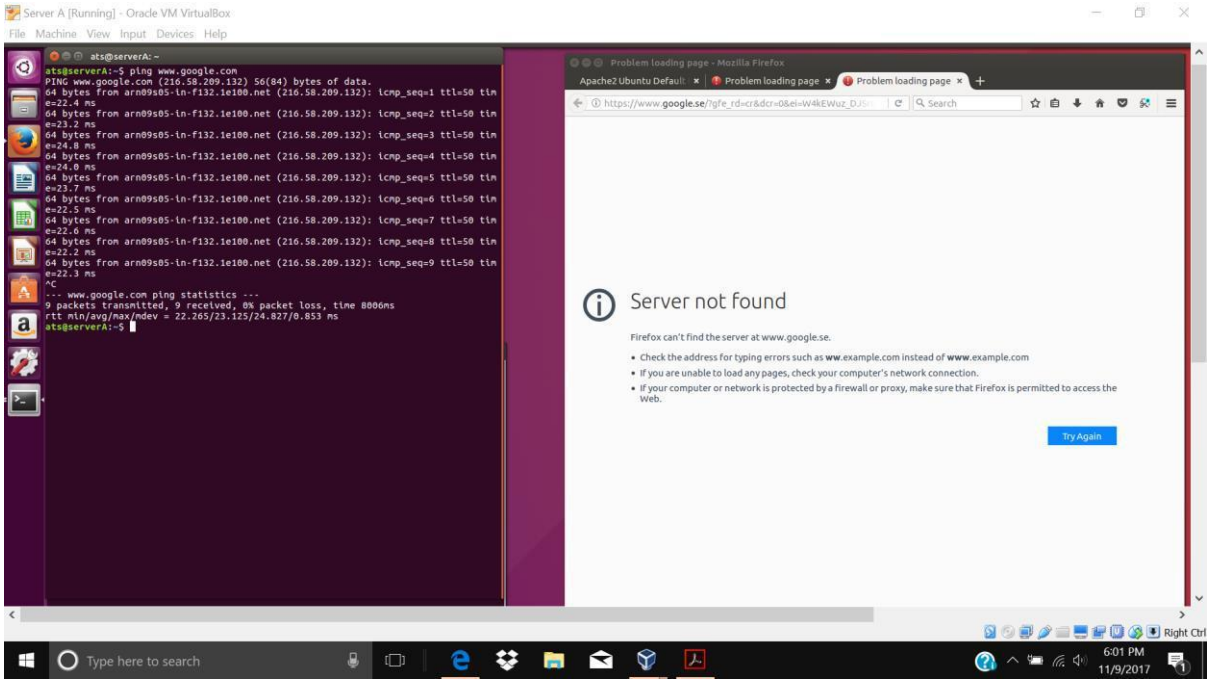
➔ $ iptables -I OUTPUT -p icmp -j ACCEPT

➔ $iptables -I INPUT -p icmp -j ACCEPT

☐ **To ping with www.google.se we should accept DNS name server**

➔ We can go to DNS name server only via NAT engine, so it is essential to allow both to communicate with us such we have the service.
➔ Following commands helps in that work

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

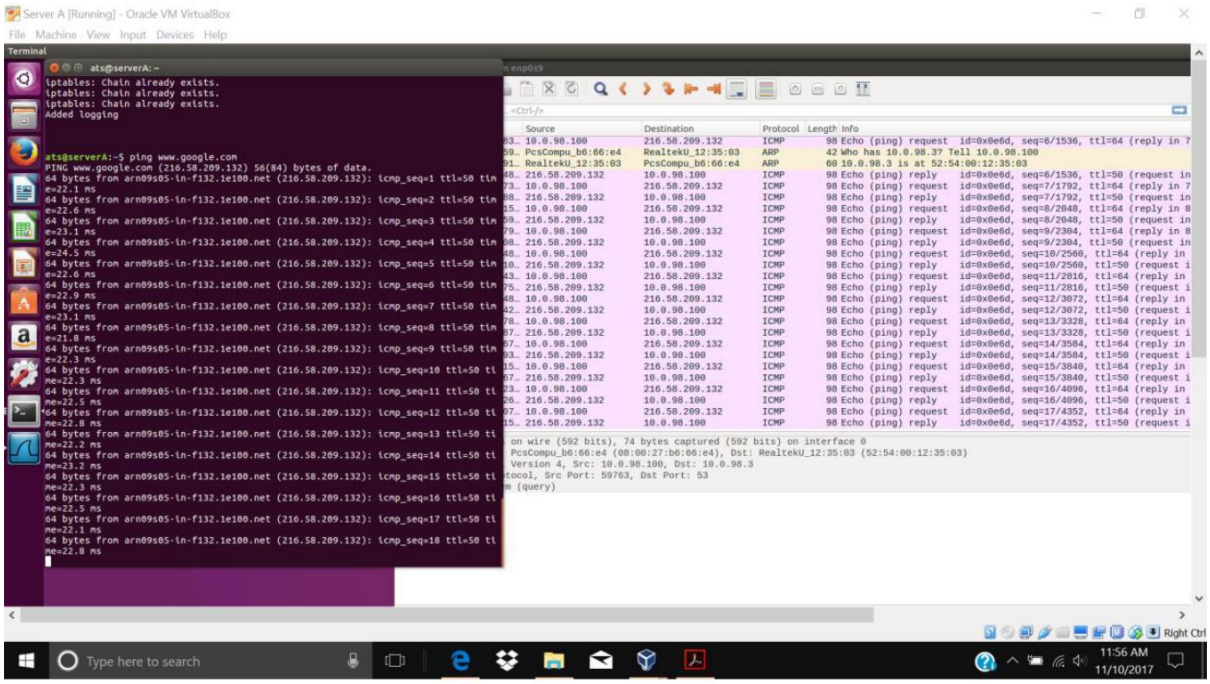


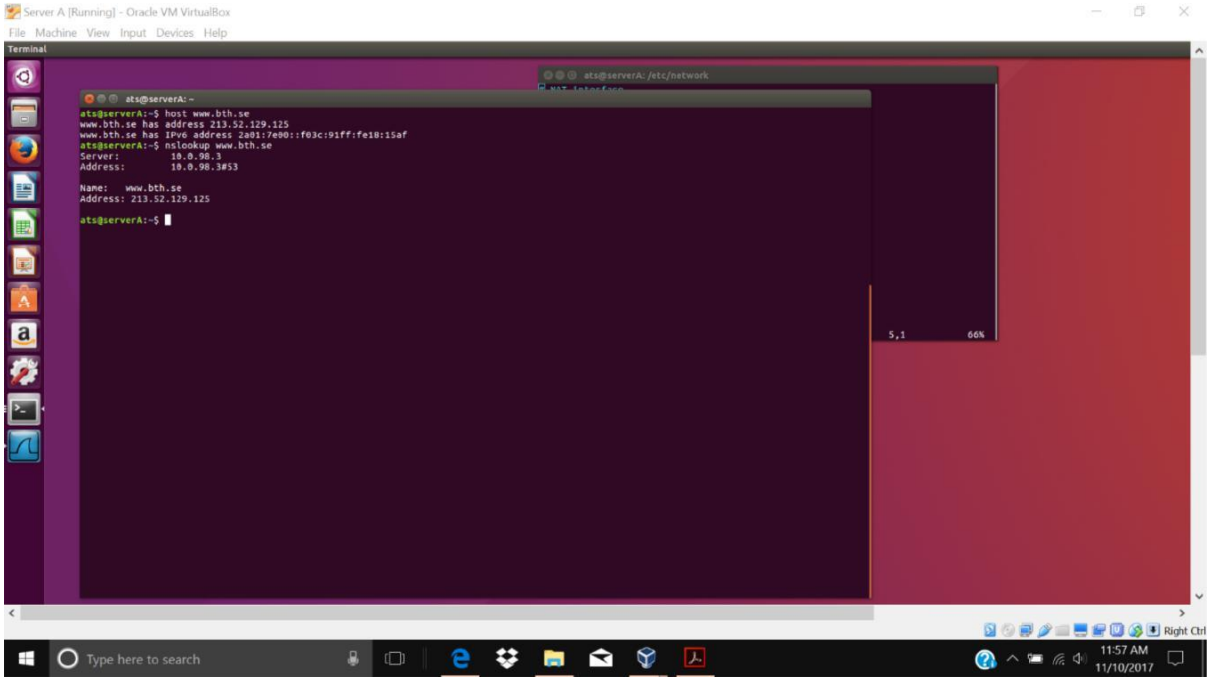
<u>TASK 19:</u> Allow server A to ping all hosts

This task is bit like before task, but to accomplish this task I have used a debugging and sniffing tool called wireshark to know where my packets got dropped and to add appropriate rules.

To allow the server to ping all hosts. By adding the following rules to the firewall.sh script and executing it, we are allowing the firewall to accept the outgoing ICMP traffic to any server and corresponding ICMP replies.
$IPT –A OUTPUT –p udp –m conntrack --ctstate \NEW,ESTABLISHED – j ACCEPT

$IPT –A INPUT –p udp –m conntrack --ctstate ESTABLISHED,RELATED –j ACCEPT





TASK 20: Stateful firewall

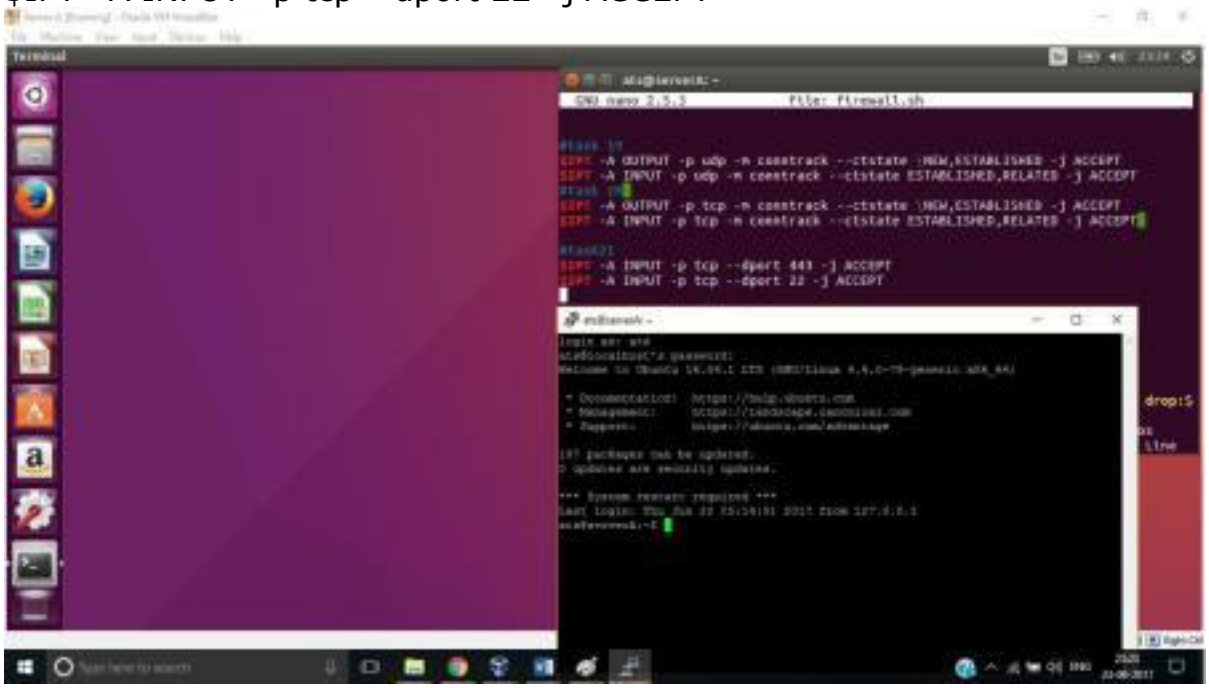
➔ To make the state full firewall, following two commands are used

➔ $iptables -t filter -A INPUT -p tcp -m Conntrack –-cstate ESTABLISHED,RELATED -j ACCEPT

- ➔ It is for filtering the input traffic which is already in established or establishing modes
- ➔ $iptables -t filter -I OUTPUT -p tcp -m conntrack –-cstate NEW,ESTABLISHED -j ACCEPT
- ➔ This command will filter, all the other requests going out except NEW and ESTABLISHED modes of operation.
- ➔ To disable SSH we should comment the command that is for accepting all the TCP traffic.
- ➔ #iptables -t filter -I OUTPUT -p tcp -j ACCEPT
- ➔ # iptables -t filter -I INPUT -p tcp -j ACCEPT

<u>TASK 21:</u> Enable SSH and HTTPS contents from apache2 server for web browser on host (base machine).

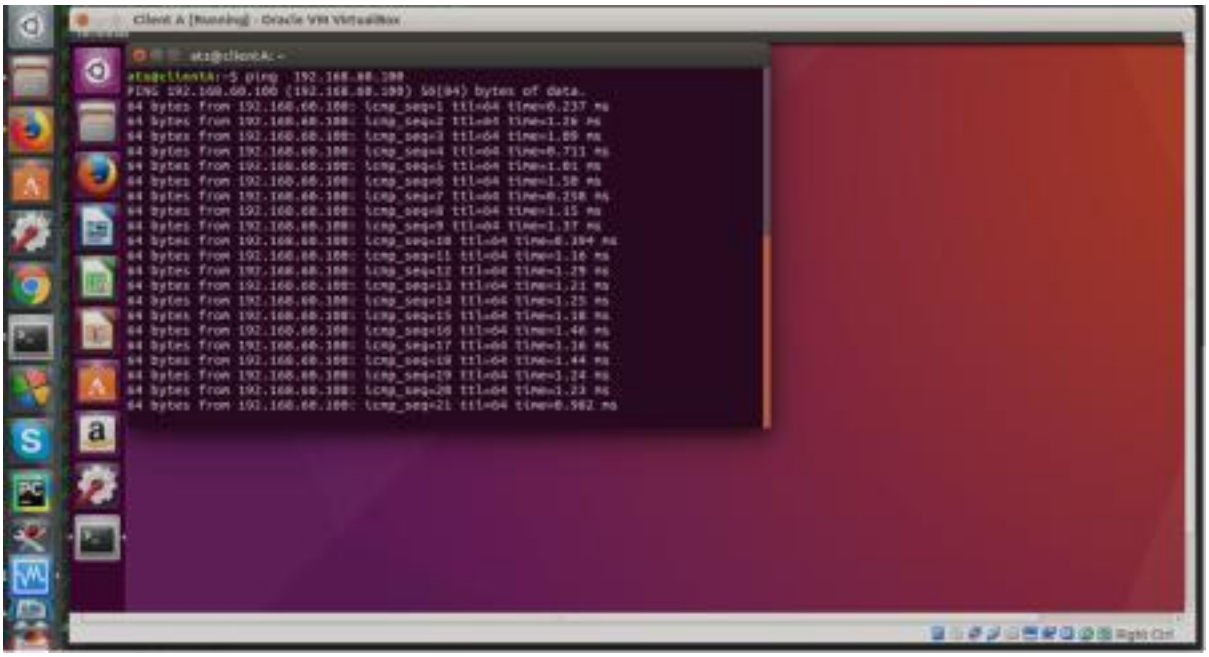
To Enable SSH and HTTPS content from apache2 server for web browser on HOST, add the following commands.
$IPT –A INPUT –p tcp --dport 443 –j ACCEPT
$IPT –A INPUT –p tcp --dport 22 –j ACCEPT



Task 22:
To add the firewall rules to ping server A from client A. add the following rules in firewall.sh

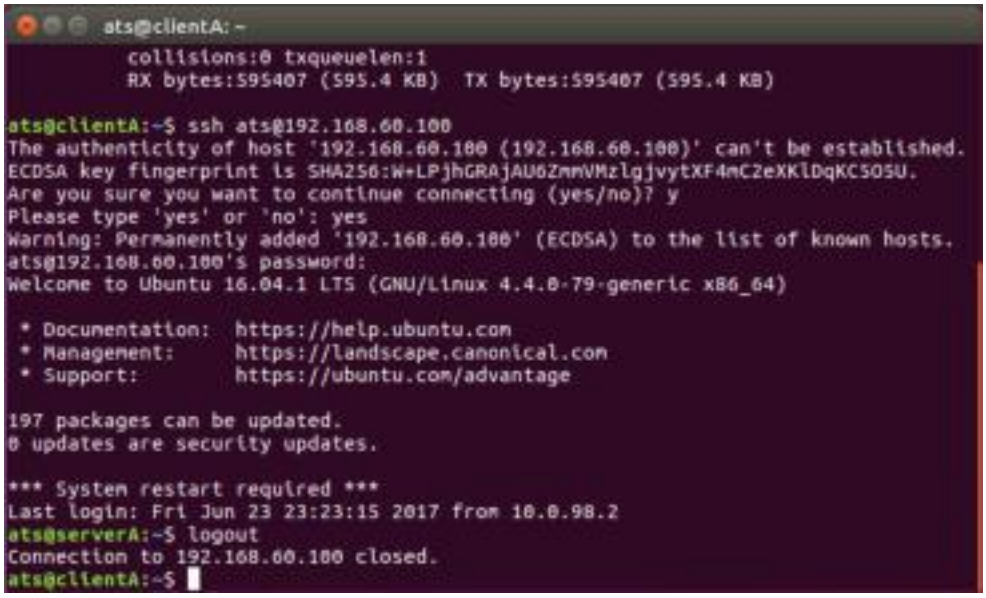
$IPT -t filter -A INPUT -p icmp -s 192.168.60.111 -j ACCEPT
$IPT -t filter -A OUTPUT -p icmp -s 192.168.60.111 -j ACCEPT

(or)
$iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
$iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT


Client A [Running] - Oracle VM VirtualBox
atc@clientA: ~
atc@clientA:~$ ping 192.168.60.100
PING 192.168.60.100 (192.168.60.100) 56(84) bytes of data.
64 bytes from 192.168.60.100: icmp_seq=1 ttl=64 time=0.237 ms
64 bytes from 192.168.60.100: icmp_seq=2 ttl=64 time=1.26 ms
64 bytes from 192.168.60.100: icmp_seq=3 ttl=64 time=1.89 ms
64 bytes from 192.168.60.100: icmp_seq=4 ttl=64 time=0.711 ms
64 bytes from 192.168.60.100: icmp_seq=5 ttl=64 time=1.01 ms
64 bytes from 192.168.60.100: icmp_seq=6 ttl=64 time=1.58 ms
64 bytes from 192.168.60.100: icmp_seq=7 ttl=64 time=0.258 ms
64 bytes from 192.168.60.100: icmp_seq=8 ttl=64 time=1.15 ms
64 bytes from 192.168.60.100: icmp_seq=9 ttl=64 time=1.37 ms
64 bytes from 192.168.60.100: icmp_seq=10 ttl=64 time=0.184 ms
64 bytes from 192.168.60.100: icmp_seq=11 ttl=64 time=1.16 ms
64 bytes from 192.168.60.100: icmp_seq=12 ttl=64 time=1.29 ms
64 bytes from 192.168.60.100: icmp_seq=13 ttl=64 time=1.21 ms
64 bytes from 192.168.60.100: icmp_seq=14 ttl=64 time=1.25 ms
64 bytes from 192.168.60.100: icmp_seq=15 ttl=64 time=1.18 ms
64 bytes from 192.168.60.100: icmp_seq=16 ttl=64 time=1.46 ms
64 bytes from 192.168.60.100: icmp_seq=17 ttl=64 time=1.16 ms
64 bytes from 192.168.60.100: icmp_seq=18 ttl=64 time=1.44 ms
64 bytes from 192.168.60.100: icmp_seq=19 ttl=64 time=1.24 ms
64 bytes from 192.168.60.100: icmp_seq=20 ttl=64 time=1.23 ms
64 bytes from 192.168.60.100: icmp_seq=21 ttl=64 time=0.502 ms
Right Ctrl

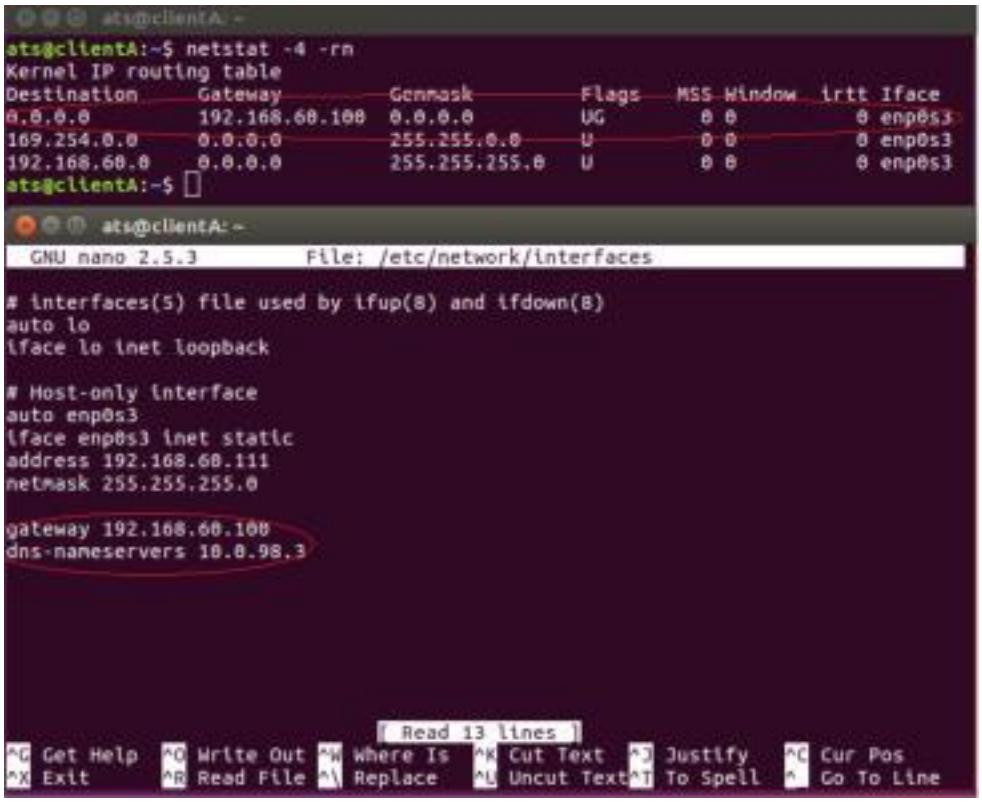
Task 23:
To fix the firewall rules such that we can SSH from Client A to Server A. Add the following rules to the firewall.

sudo iptables -I INPUT -s 192.168.60.111 -p tcp -m tcp --dport 10022 -j ACCEPT



For ssh I used below command.
Sudo ssh -p 10022 ats@192.168.60.100

Task 24:
In this task and the gateway and dns-nameserver in **/etc/network/interface** file in client A, so that we can able to add gateway and dns-servername to client A.



Task 25:
To execute following command in the terminal of Server A so that IP forwarding is enabled on the Server A. This will forward the packets from enp0s3 to enp0s9

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

Task 26:
To change the iptables rules to forward packets. add the following rules to forward packets from enp0s3 to enp0s9.

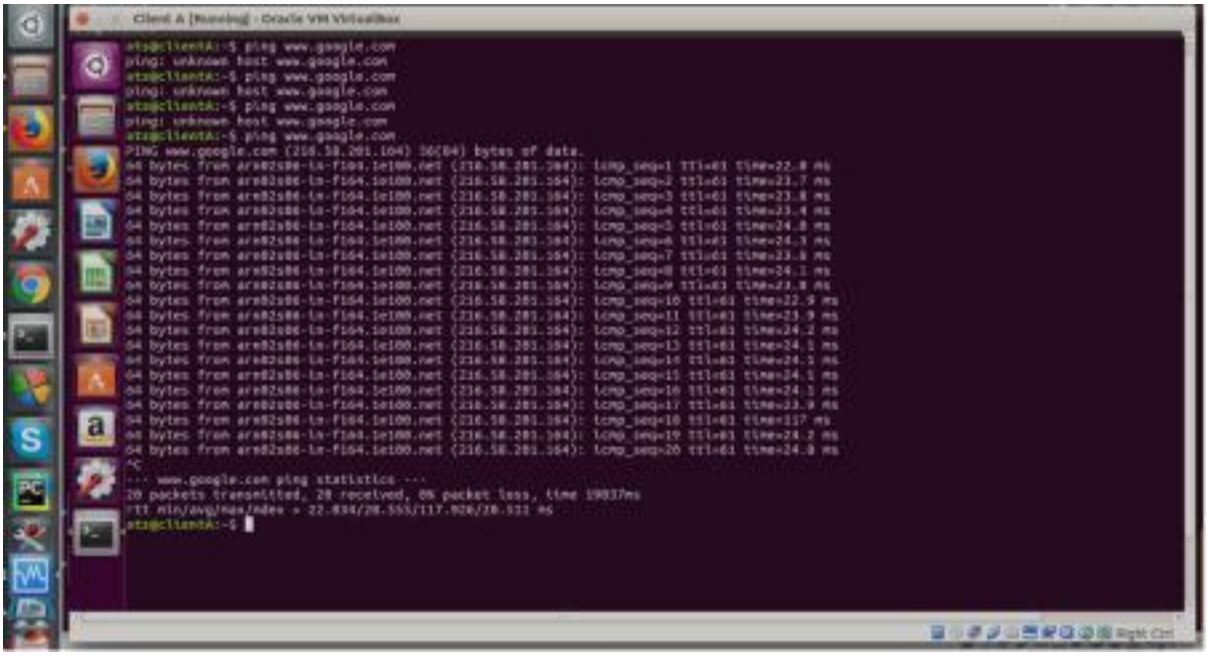
$IPT -t filter -A FORWARD -i $HIF -j ACCEPT
$IPT -t filter -A FORWARD -i $NIF -m conntrack --ctstate ESTABLISHED, RELATED -j ACCEPT

After changing these rules the packets are forwarded to the NAT interface. But here the problem is that Client A uses private address (192.168.60.111) and all routers will have a basic default rule to drop packets coming from the private addreses. So we need to tell Server A to use NAT (more specifically Source NAT - SNAT). In order to do this we need to enable the SNAT on Server A.

Task 27:

To fix the problem outlined above you need to tell Server A to do SNAT on the NAT interface. You must add the following iptables rule.

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



SCRIPT:

```
#!/bin/sh

IPT=/sbin/iptables
# NAT interface
NIF=enp0s9
# NAT IP address
NIP='10.0.98.100'

# Host-only interface
HIF=enp0s3
# Host-only IP addres
HIP='192.168.60.100'

# DNS nameserver
NS='10.0.98.3'
#########33NS='8.8.8.8'

## Reset the firewall to an empty, but friendly state

# Flush all chains in FILTER
table $IPT -t filter -F
####$sudo iptables -A INPUT -p tcp --dport 80 -j REJECT
##sudo $IPT -t filter -A INPUT -p tcp --dport 80 -j REJECT
# Delete any user-defined chains in FILTER table
#$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 INPUT ACCEPT

####TASK14##$IPT -t filter -A OUTPUT -s 127.0.0.0/24 -p tcp --dport
80 -j REJECT

#$IPT -t filter -P OUTPUT ACCEPT

#$IPT -t filter -P FORWARD ACCEPT

##TASK15##


####$IPT -t filter -P INPUT DROP  ##1

####$IPT -t filter -I INPUT -p icmp -j ACCEPT  ##2

####$IPT -t filter -I OUTPUT -p icmp -j ACCEPT ##2
```

####$IPT -t filter -P OUTPUT DROP ##1

####$IPT -t filter -P FORWARD DROP ##1

##TASKS17-20#####

$IPT -t filter -P INPUT DROP

$IPT -t filter -I INPUT -p icmp -j ACCEPT ##1

$IPT -t filter -A INPUT -s 127.0.0.0/24 -p tcp -j ACCEPT ##2

##$IPT -t filter -A INPUT -s 10.0.98.3 -p udp --dport 53 -j ACCEPT

#$IPT -t filter -A INPUT -s 10.0.98.2 -p udp --dport 53 -j ACCEPT

#$IPT -t filter -A INPUT -s 10.0.98.100 -p udp --dport 53 -j ACCEPT

#$IPT -t filter -A INPUT -s 10.0.98.3 -p tcp --dport 53 -j ACCEPT

#$IPT -t filter -A INPUT -s 10.0.98.2 -p tcp --dport 53 -j ACCEPT

#$IPT -t filter -A INPUT -s 10.0.98.100 -p tcp --dport 53 -j

ACCEPT $IPT -t filter -A INPUT -s 10.0.98.100 -j ACCEPT ##3

$IPT -t filter -A INPUT -s 10.0.98.3 -j ACCEPT ##3

#$IPT -t filter -A INPUT -p tcp -j ACCEPT ##4##20

$IPT -t filter -A INPUT -p tcp -m conntrack --ctstate
ESTABLISHED,RELATED -j ACCEPT ##20

###IPADDRHOST=192.168.0.1

###UNPRIVPORTS="1025:65535"

$IPT -t filter -A INPUT -p tcp --dport 22 -j ACCEPT ##21

###$IPT -A INPUT -i $HIP -p tcp !--syn --sport 443 -d $IPADDRHOST --
match multiport --dports $UNPRIVPORTS -j ACCEPT ##21

$IPT -t filter -A INPUT -p tcp --dport 443 -j ACCEPT ##21

$IPT -t filter -A INPUT -p icmp -s 192.168.60.111 -j ACCEPT
$IPT -t filter -A OUTPUT -p icmp -s 192.168.60.111 -j ACCEPT##22

sudo iptables -I INPUT -s 192.168.60.111 -p tcp -m tcp --dport 10022 -j ACCEPT##23

##5

###

$IPT -t filter -P OUTPUT DROP

$IPT -t filter -A OUTPUT -s 127.0.0.0/24 -p tcp -j ACCEPT ##2

$IPT -t filter -A OUTPUT -p icmp -j ACCEPT ##1

#$IPT -t filter -A OUTPUT -s 10.0.98.3 -p udp --dport 53 -j ACCEPT

#$IPT -t filter -A OUTPUT -s 10.0.98.2 -p udp --dport 53 -j ACCEPT

#$IPT -t filter -A OUTPUT -s 10.0.98.100 -p udp --dport 53 -j ACCEPT

#$IPT -t filter -A OUTPUT -s 10.0.98.3 -p tcp --dport 53 -j ACCEPT

#$IPT -t filter -A OUTPUT -s 10.0.98.2 -p tcp --dport 53 -j ACCEPT

#$IPT -t filter -A OUTPUT -s 10.0.98.100 -p tcp --dport 53 -j ACCEPT

##$IPT -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
##$IPT -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

$IPT -t filter -A OUTPUT -s 10.0.98.100 -j ACCEPT ##3 $IPT -t filter

-A OUTPUT -s 10.0.98.3 -j ACCEPT ##3

#$IPT -t filter -I OUTPUT -p tcp -j ACCEPT ##4##20

#task 20

$IPT -t filter -I OUTPUT -p tcp -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

```
###$IPT -t filter -I OUTPUT -p tcp  --dport 22  -j ACCEPT ##21
```

###$IPT -A OUTPUT -o $HIP -p tcp -s $IPADDRHOST -m multiport --sports $UNPRIVPORTS --dport 443 -j ACCEPT ##21

$IPT -t filter -P FORWARD DROP

$IPT -t filter -A FORWARD -i $HIF -j ACCEPT
$IPT -t filter -A FORWARD -i $NIF -m conntrack --ctstate ESTABLISHED, RELATED -j ACCEPT ###26

$IPT -t nat -A POSTROUTING -j SNAT -o $NIF --to $NIP ##27

Create logging chains

$IPT -t filter -N input_log

$IPT -t filter -N output_log

$IPT -t filter -N forward_log

Set some logging targets for DROPPED packets

$IPT -t filter -A input_log -j LOG --log-level notice --log-prefix "input drop: "

$IPT -t filter -A output_log -j LOG --log-level notice --log-prefix "output drop: "

$IPT -t filter -A forward_log -j LOG --log-level notice --log-prefix "forward drop: "

echo "Added logging"

Return from the logging chain to the built-in

chain $IPT -t filter -A input_log -j RETURN

```
$IPT -t filter -A output_log -j RETURN
$IPT -t filter -A forward_log -j RETURN



# These rules must be inserted at the end of the built-in
# chain to log packets that will be dropped by the default
# DROP policy
$IPT -t filter -A INPUT -j input_log
$IPT -t filter -A OUTPUT -j output_log
$IPT -t filter -A FORWARD -j forward_log
#####$ping 127.0.0.1
echo $input_log
echo $output_log
echo $forward_log
```