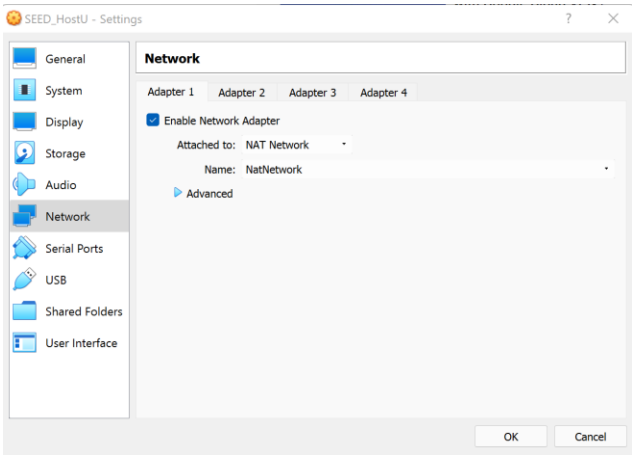
Lab3 VPN Tunneling

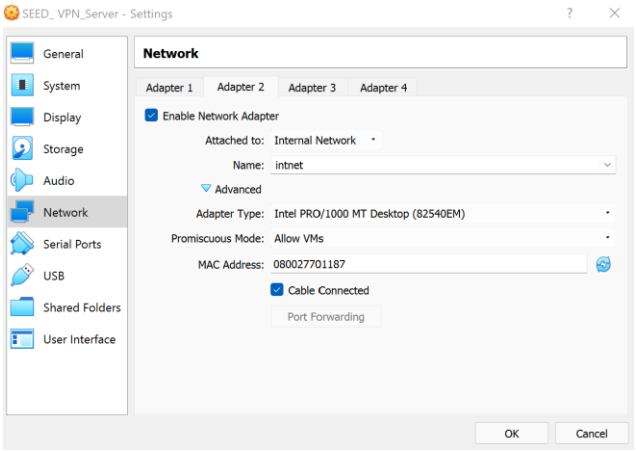
I. Task 1: Lab set up

a. Set up three virtual machines (SEEDLab image)

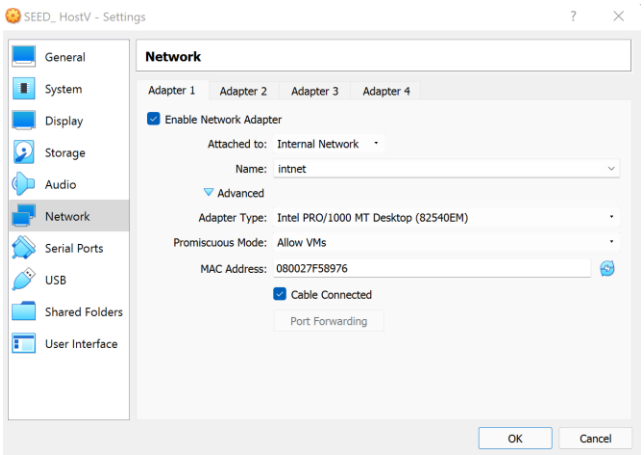
b. Host U: One adapter (NAT)

c. 

d. VPN Server: Two adapters (NAT and internal network with allow VMs)

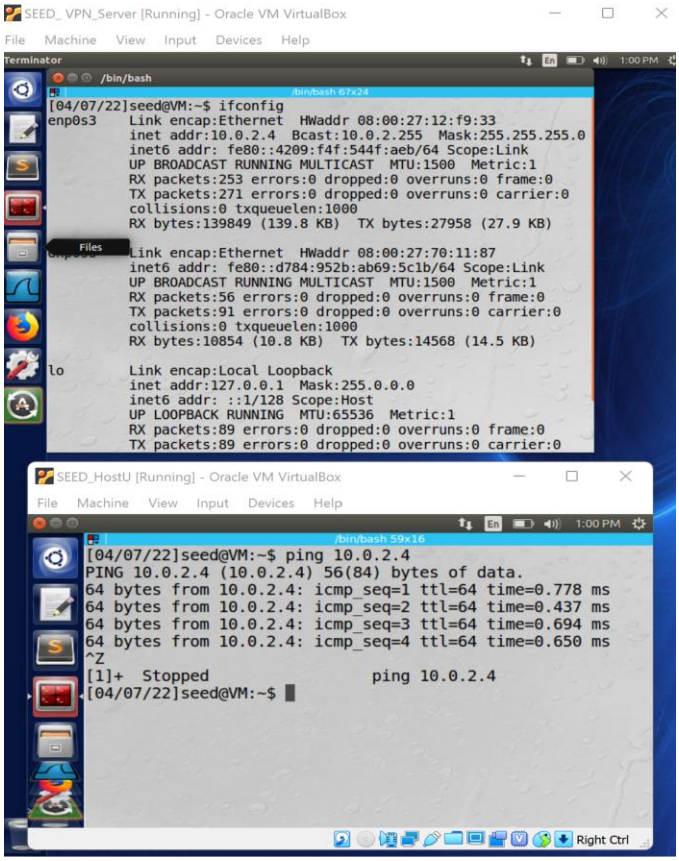
e. 

f. Host V: One adapter (internal network)

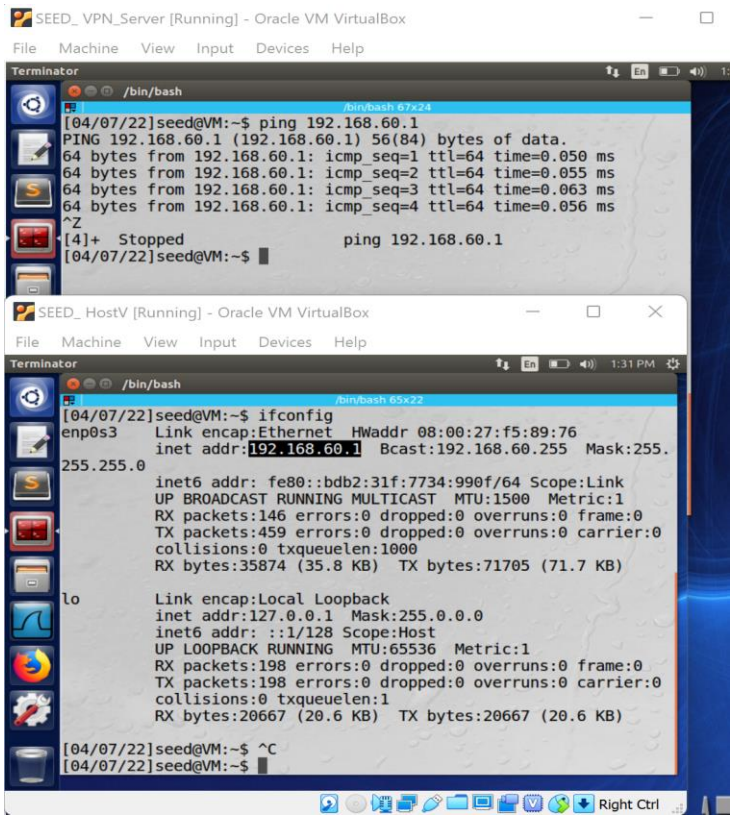
g. 

h. Testing:

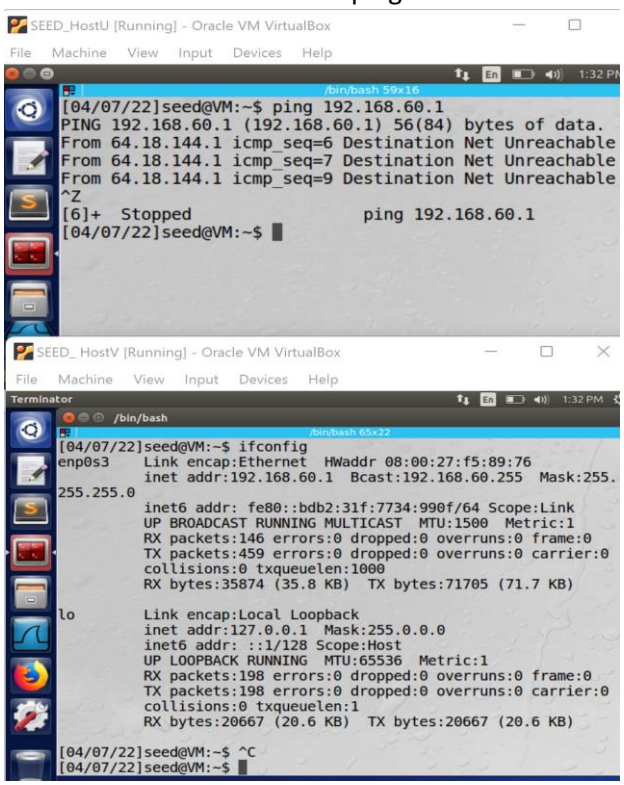
   i. Host U pinging VPN Server

ii. 

iii. VPN server pinging host V

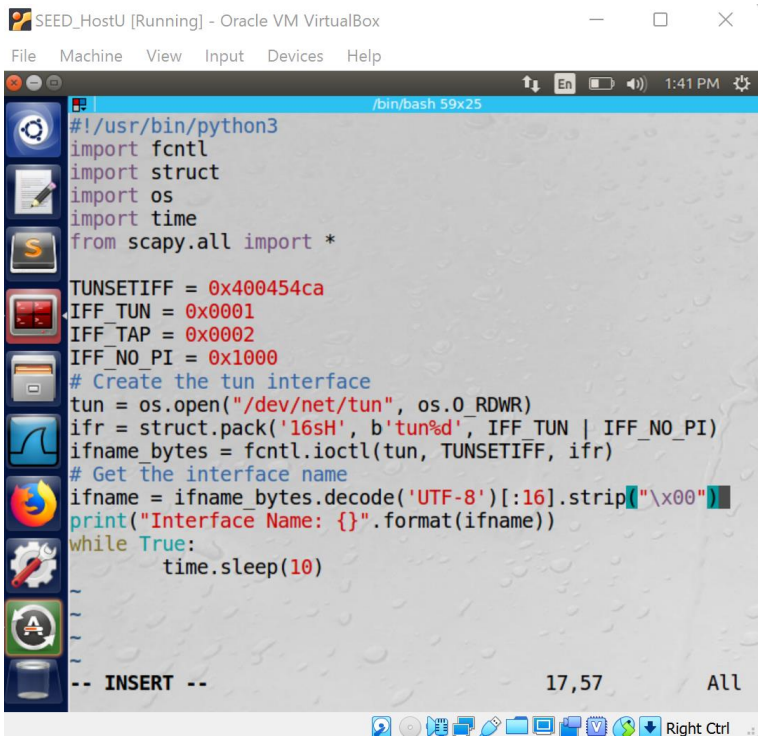
iv. 

v. Host U should not be able to ping Host V

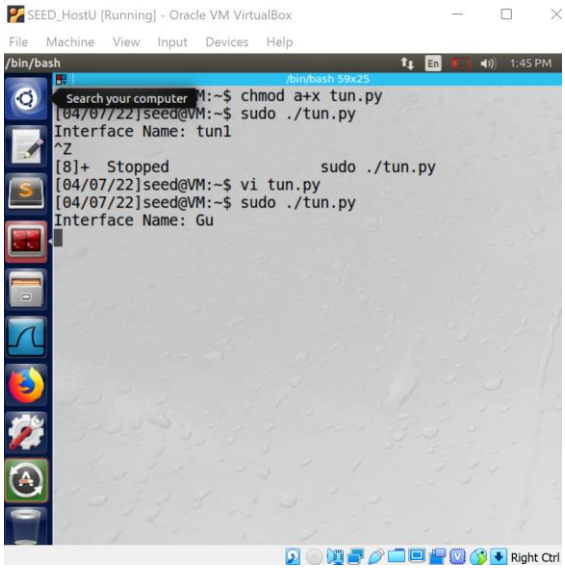
vi. 

II. Task 2 Create and Configure TUN Interface

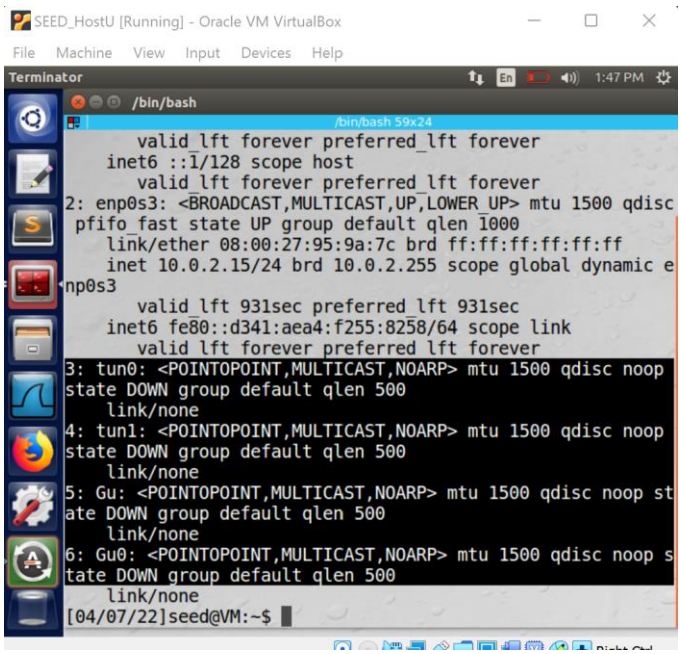
a. Write tun.py from code provided in lab manual.

b. 

III. Task 2.a Name the interface: change interface name to my last name.

a. 

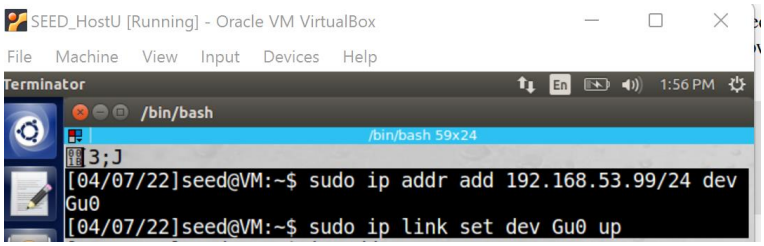
b. Print out all the interfaces on Host U

c. 

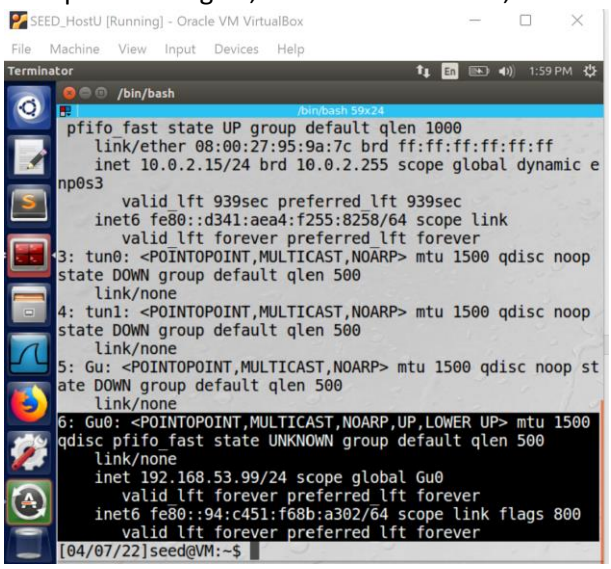
d. Observations: we can see tun0, tun1, Gu, Gu0 are the interfaces on this machine. Every time the script is ran; a new interface will be created.

IV. Task 2.b Set up the TUN interface

a. Run the commands to assign the IP and bring the interface up.

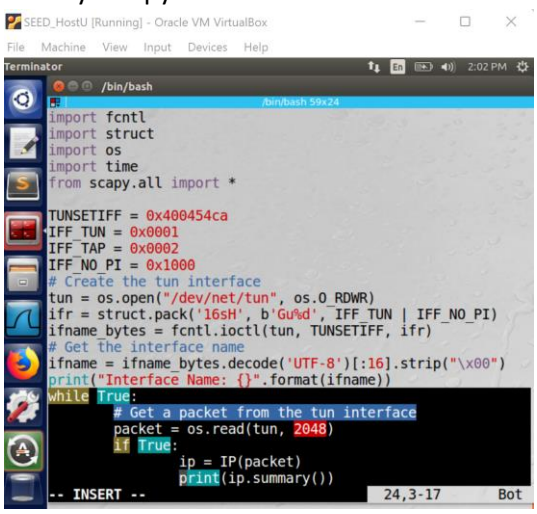
b. 

c. Run ip address again, under Gu0 interface, IP address is successfully assigned.

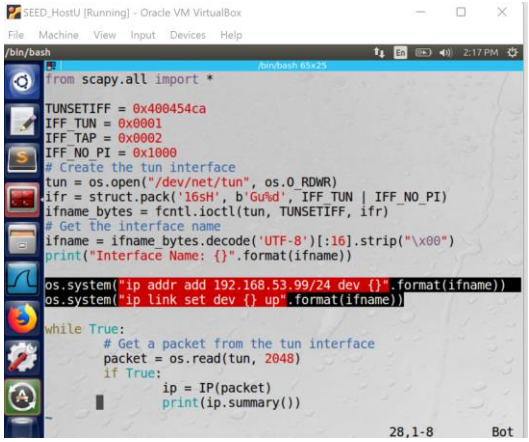
d. 

V. Task 2.c Read from the RUN Interface

a. Modify tun.py.

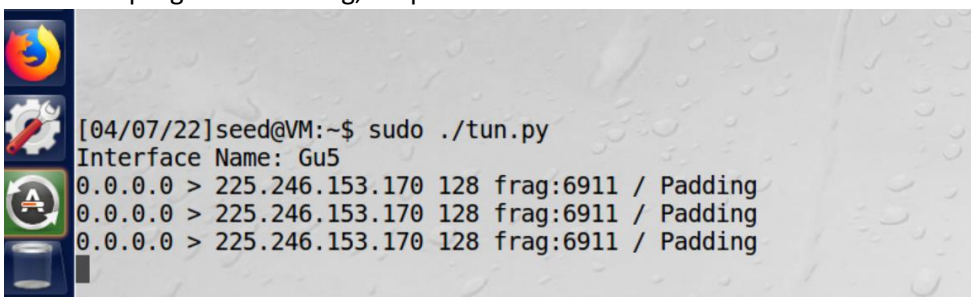
b. 

c. Also added these lines to configure automatically.

d. 

e. Run tun.py and

i. When the program is running, outputs:

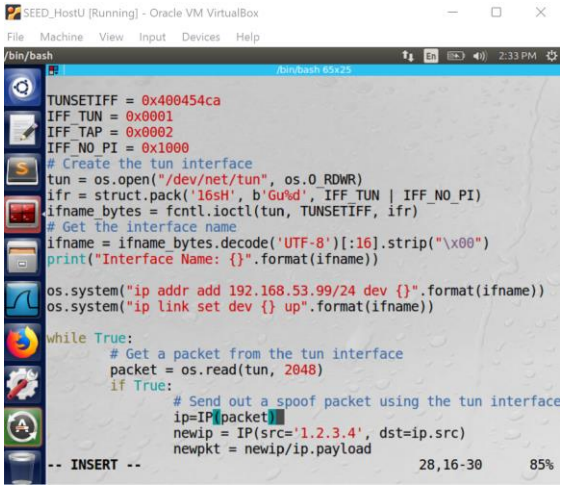
ii. 

iii. Ping 192.168.53.1. The program outputs nothing and ping will output destination net unreachable.

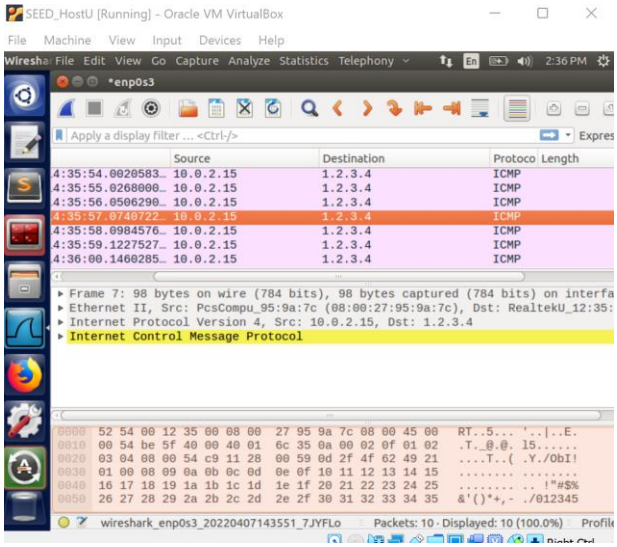
iv. Ping 192.168.60.1. The program outputs nothing and ping will output destination net unreachable.

VI. Task 2.d Write to the TUN interface.

a. Modify tun.py.

b. 

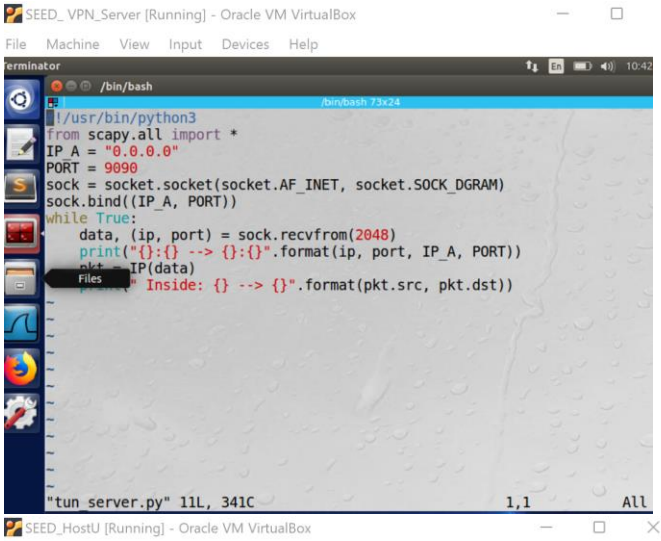
c. Run the code and use Wireshark to capture traffic meanwhile.

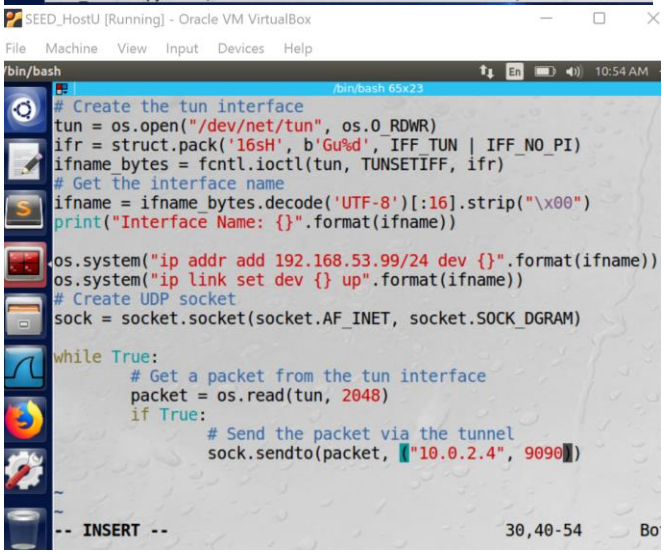
d. 

e. ICMP packets from Host U's IP address to 1.2.3.4 (arbitrary IP in the code) can be spotted. The spoofing of packets was successful.

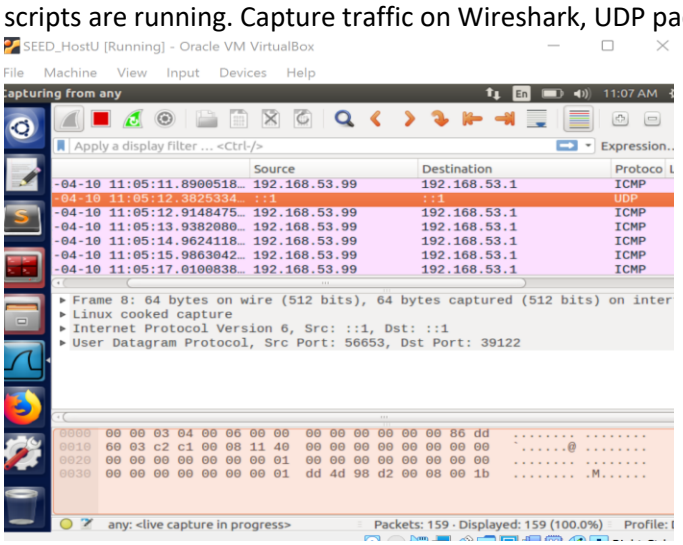
VII. Task 3 Send the IP Packet to VPN Server Through a Tunnel

a. On VPN server, create tun_server.py and modify tun.py to tun_client.py on host U.

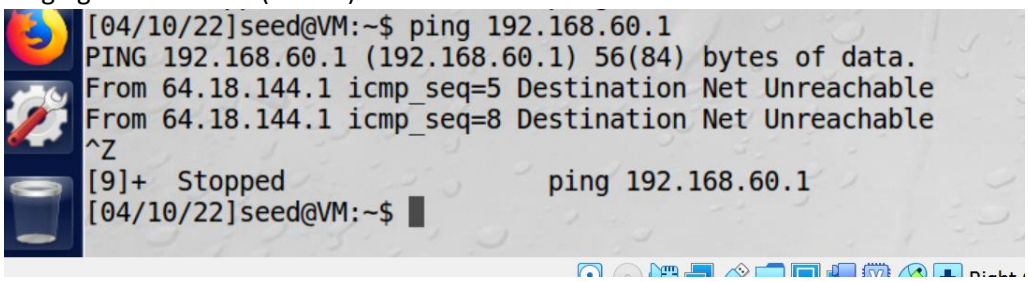
b. 

c. 

d. Run tun_server.py and then run tun_client.py on Host U. Ping 192.168.60.2 while the scripts are running. Capture traffic on Wireshark, UDP packets are observed.

e. 

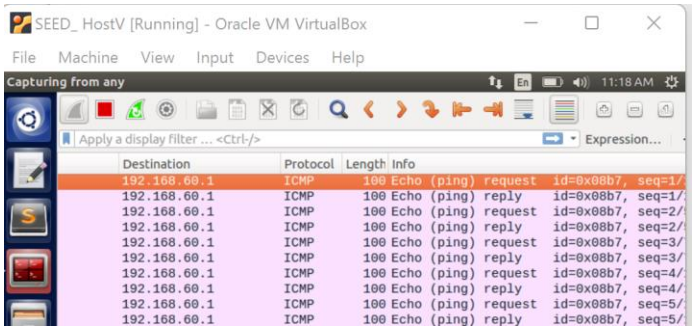
f. Pinging 192.168.60.1 (Host V). It would show a destination net unreachable error.



g. Now use this command to add an entry to the routing table.

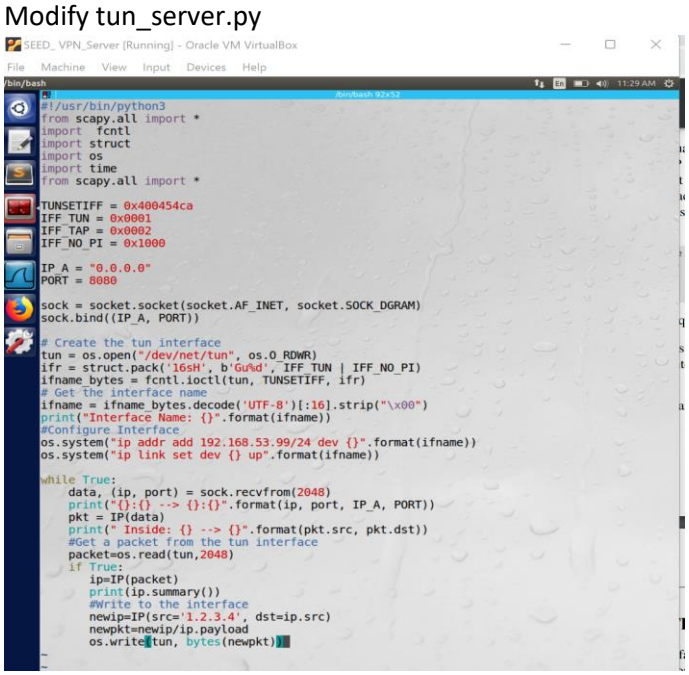
h. 

i. After this entry is added, ping 192.168.60.1, and capture traffic on Wireshark. ICMP packets are spotted that are sent to 192.168.60.1

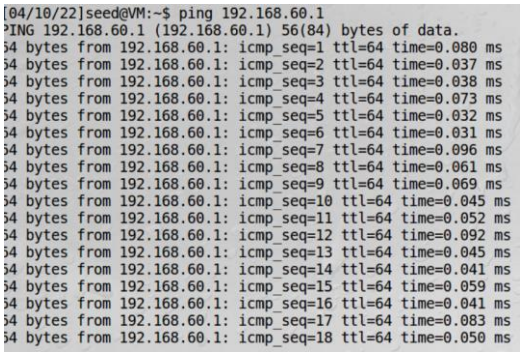
j. 

VIII. Task 4 Set Up the VPN Server

a. Modify tun_server.py

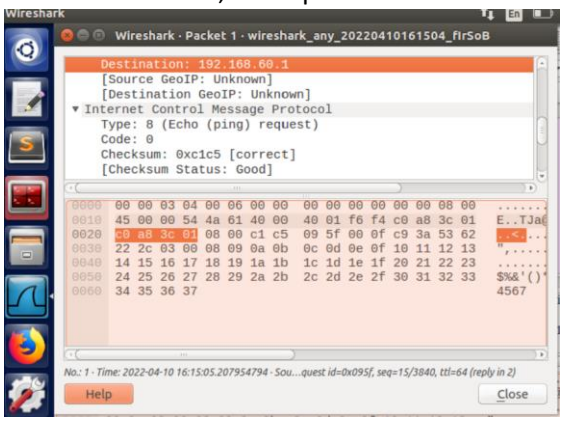
b. 

c. Enable IP forwarding and run the script. And ping host V (192.168.60.1) from host u.

d. 

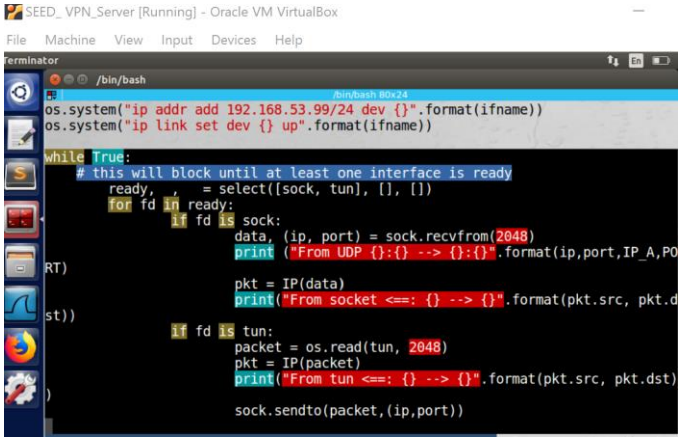
```
[04/10/22]seed@VM:~$ ping 192.168.60.1
PING 192.168.60.1 (192.168.60.1) 56(84) bytes of data.
64 bytes from 192.168.60.1: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.038 ms
64 bytes from 192.168.60.1: icmp_seq=4 ttl=64 time=0.073 ms
64 bytes from 192.168.60.1: icmp_seq=5 ttl=64 time=0.032 ms
64 bytes from 192.168.60.1: icmp_seq=6 ttl=64 time=0.031 ms
64 bytes from 192.168.60.1: icmp_seq=7 ttl=64 time=0.096 ms
64 bytes from 192.168.60.1: icmp_seq=8 ttl=64 time=0.061 ms
64 bytes from 192.168.60.1: icmp_seq=9 ttl=64 time=0.069 ms
64 bytes from 192.168.60.1: icmp_seq=10 ttl=64 time=0.045 ms
64 bytes from 192.168.60.1: icmp_seq=11 ttl=64 time=0.052 ms
64 bytes from 192.168.60.1: icmp_seq=12 ttl=64 time=0.092 ms
64 bytes from 192.168.60.1: icmp_seq=13 ttl=64 time=0.045 ms
64 bytes from 192.168.60.1: icmp_seq=14 ttl=64 time=0.041 ms
64 bytes from 192.168.60.1: icmp_seq=15 ttl=64 time=0.059 ms
64 bytes from 192.168.60.1: icmp_seq=16 ttl=64 time=0.041 ms
64 bytes from 192.168.60.1: icmp_seq=17 ttl=64 time=0.083 ms
64 bytes from 192.168.60.1: icmp_seq=18 ttl=64 time=0.050 ms
```

e. And on Wireshark, ICMP packets have reached host V.

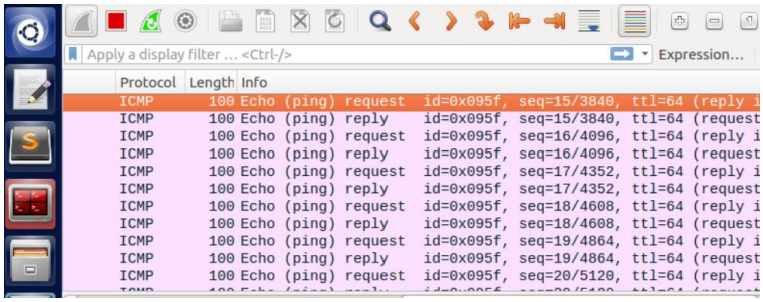
f. 

IX. Task 5 Handling Traffic in Both Directions

a. Modify tun_server.py and tun_client.py with the below code in the while loop.

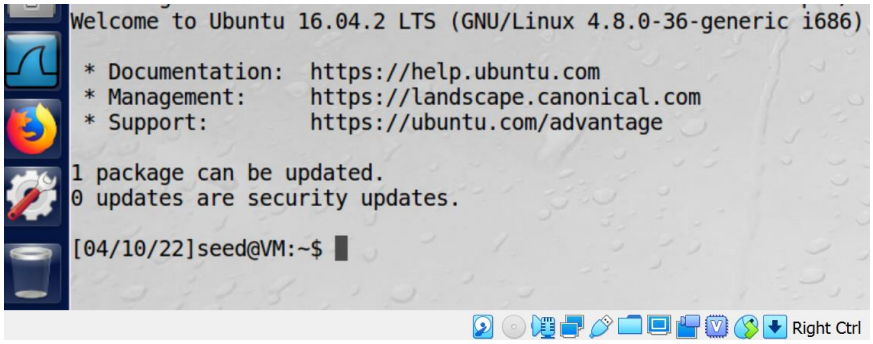
b. 

c. And while running scripts, ping host V from host U, ICMP request that goes to Host V and ICMP reply that goes back to host U are observed.

d. 

X. Task 6 Tunnel-Breaking Experiment

a. On host U telnet host V and break the connection by stopping tun_server.py. Our terminal will freeze and does not show anything we type, since the TCP connection is lost when we stop tun_server.py

b. 

c. Then we restart the tunnel. The terminal can show input, and when typing exit, the connection can be closed.

d.