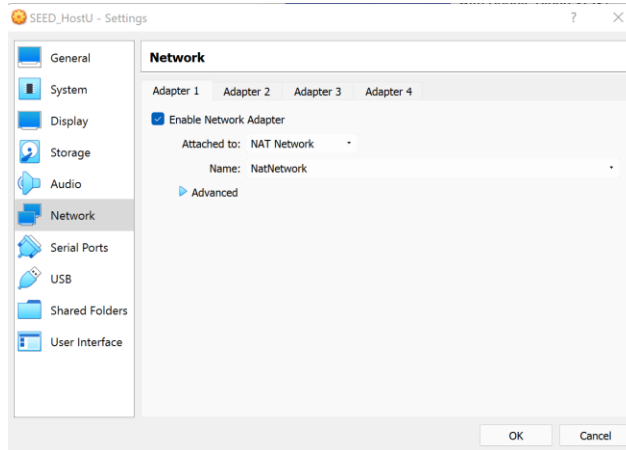
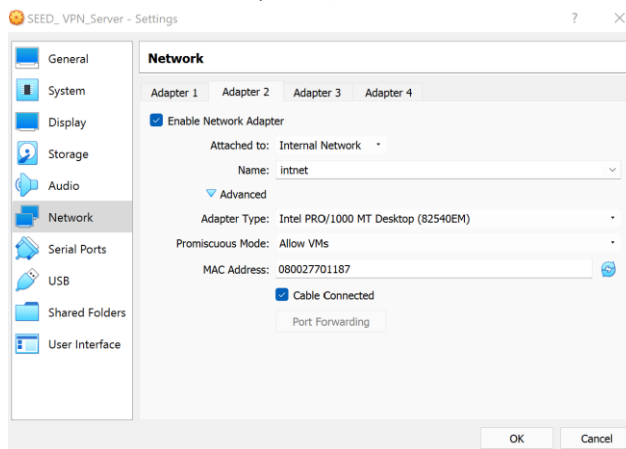


## Lab3 VPN Tunneling

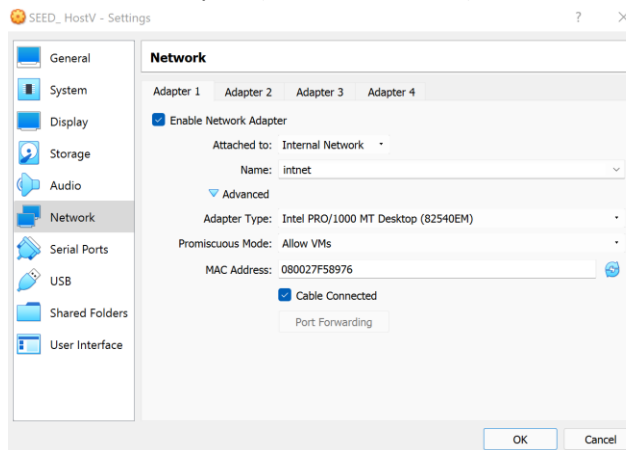
- I. Task 1: Lab set up
  - a. Set up three virtual machines (SEEDLab image)
  - b. Host U: One adapter (NAT)



- 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 ping the VPN Server

```
SEED_VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminator
/bin/bash
[04/07/22]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:12:f9:33
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.0
          inet6 addr: fe80::4209:f4f:544f:aeb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:253 errors:0 dropped:0 overruns:0 frame:0
          TX packets:271 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:139849 (139.8 KB)  TX bytes:27958 (27.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:89 errors:0 dropped:0 overruns:0 frame:0
          TX packets:89 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10854 (10.8 KB)  TX bytes:14568 (14.5 KB)

SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminator
/bin/bash 59x16
[04/07/22]seed@VM:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.778 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.437 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.694 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.650 ms
^Z
[1]+  Stopped                  ping 10.0.2.4
[04/07/22]seed@VM:~$
```

- ii.
- iii. VPN server pinging host V

```
SEED_VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminator
/bin/bash
[04/07/22]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=0.050 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from 192.168.60.1: icmp_seq=4 ttl=64 time=0.056 ms
^Z
[4]+  Stopped                  ping 192.168.60.1
[04/07/22]seed@VM:~$

SEED_HostV [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

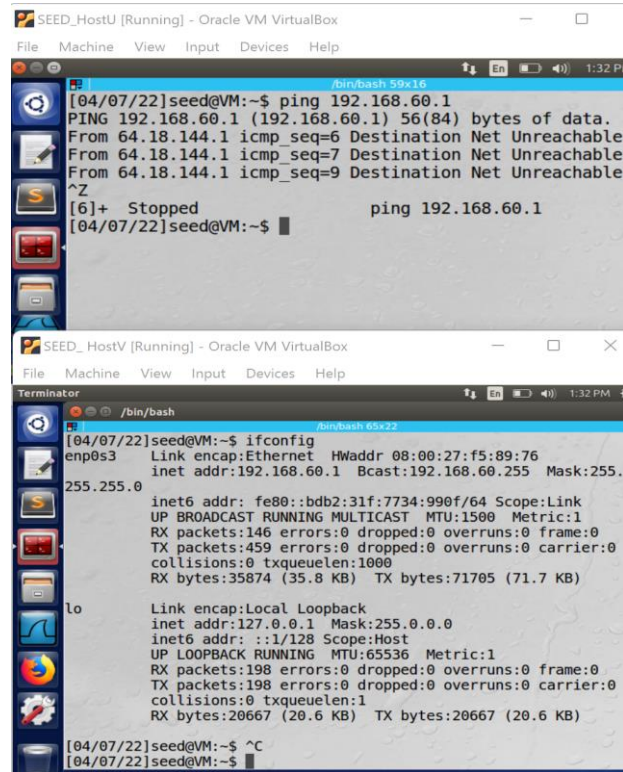
Terminator
/bin/bash
[04/07/22]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:f5:89:76
          inet addr:192.168.60.1  Bcast:192.168.60.255  Mask:255.
          255.255.0
          inet6 addr: fe80::bdb2:31f:7734:990f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:146 errors:0 dropped:0 overruns:0 frame:0
          TX packets:459 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:35874 (35.8 KB)  TX bytes:71705 (71.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:198 errors:0 dropped:0 overruns:0 frame:0
          TX packets:198 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:20667 (20.6 KB)  TX bytes:20667 (20.6 KB)

[04/07/22]seed@VM:~$ ^C
[04/07/22]seed@VM:~$
```

- iv.

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



```
[04/07/22]seed@VM:~$ ping 192.168.60.1
PING 192.168.60.1 (192.168.60.1) 56(84) bytes of data.
From 64.18.144.1 icmp_seq=6 Destination Net Unreachable
From 64.18.144.1 icmp_seq=7 Destination Net Unreachable
From 64.18.144.1 icmp_seq=9 Destination Net Unreachable
^Z
[6]+  Stopped                  ping 192.168.60.1
[04/07/22]seed@VM:~$

[04/07/22]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:f5:89:76
            inet addr:192.168.60.1  Bcast:192.168.60.255  Mask:255.
            255.255.0
            inet6 addr: fe80::bdb2:31f:7734:990f/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:146 errors:0 dropped:0 overruns:0 frame:0
            TX packets:459 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:35874 (35.8 KB)  TX bytes:71705 (71.7 KB)

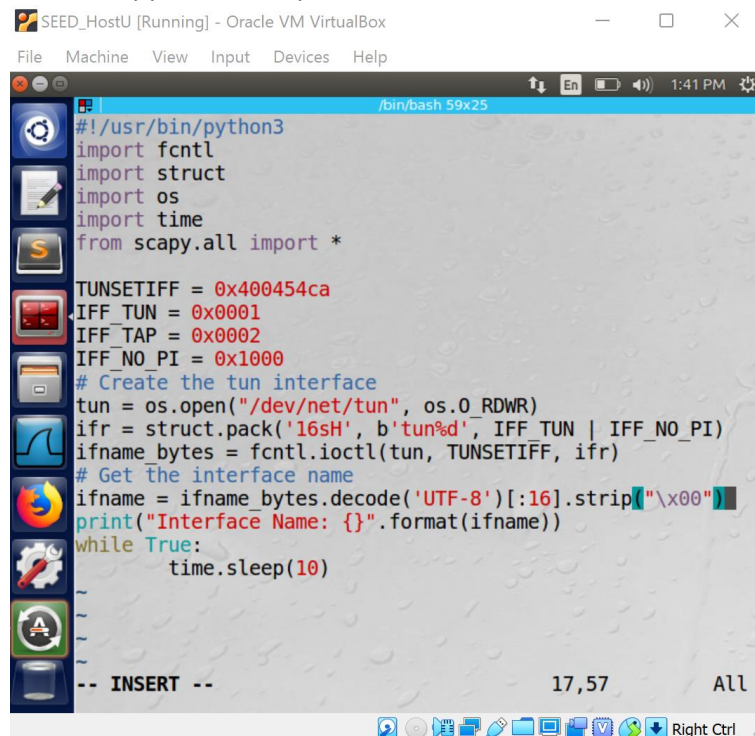
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:198 errors:0 dropped:0 overruns:0 frame:0
            TX packets:198 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:20667 (20.6 KB)  TX bytes:20667 (20.6 KB)

[04/07/22]seed@VM:~$ ^C
[04/07/22]seed@VM:~$
```

- vi.

## II. Task 2 Create and Configure TUN Interface

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



```
#!/usr/bin/python3
import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

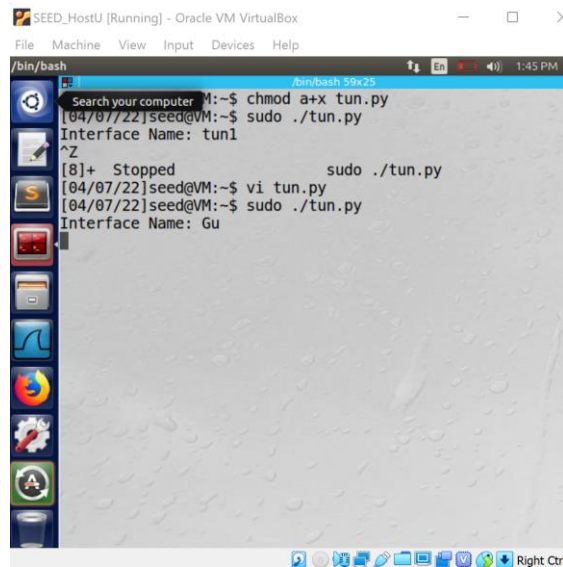
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

while True:
    time.sleep(10)

-- INSERT --
```

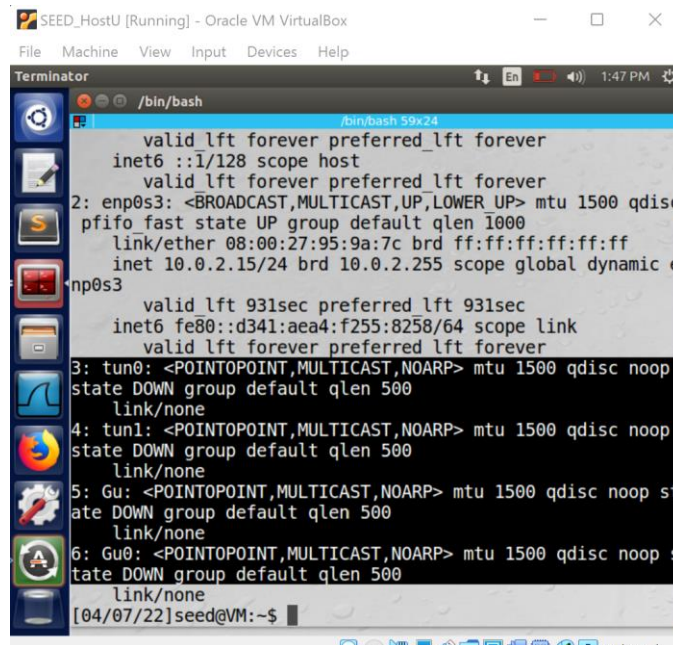
- b.

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



```
SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
Search your computer M:~$ chmod a+x tun.py
[04/07/22]seed@VM:~$ sudo ./tun.py
Interface Name: tun1
^Z
[8]+ Stopped sudo ./tun.py
[04/07/22]seed@VM:~$ vi tun.py
[04/07/22]seed@VM:~$ sudo ./tun.py
Interface Name: Gu
```

- a.
- b. Print out all the interfaces on Host U

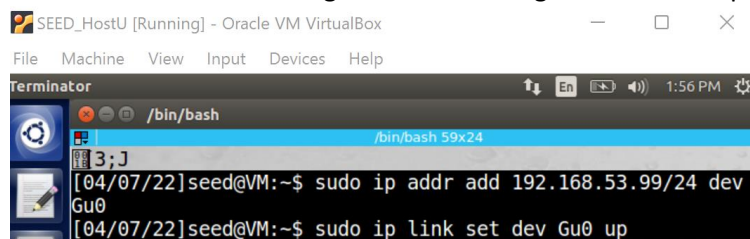


```
SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
valid lft forever preferred_lft forever
inet6 ::1/128 scope host
valid lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc
pfifo fast state UP group default qlen 1000
link/ether 08:00:27:95:9a:7c brd ff:ff:ff:ff:ff:ff
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic e
np0s3
valid lft 931sec preferred_lft 931sec
inet6 fe80::d341:aea4:f255:8258/64 scope link
valid lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop
state DOWN group default qlen 500
link/none
4: tun1: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop
state DOWN group default qlen 500
link/none
5: Gu: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop st
ate DOWN group default qlen 500
link/none
6: Gu0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop s
tate DOWN group default qlen 500
link/none
[04/07/22]seed@VM:~$
```

- 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.



```
SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
[3;J
[04/07/22]seed@VM:~$ sudo ip addr add 192.168.53.99/24 dev
Gu0
[04/07/22]seed@VM:~$ sudo ip link set dev Gu0 up
```

- b.



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

```

SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminator
/bin/bash

/bin/bash 59x24
p0: pfifo fast state UP group default qlen 1000
    link/ether 08:00:27:95:9a:7c brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic e
np0s3
    valid lft 939sec preferred lft 939sec
    inet6 fe80::d341:aea4:f255:8258/64 scope link
    valid lft forever preferred lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop
    state DOWN group default qlen 500
    link/none
4: tun1: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop
    state DOWN group default qlen 500
    link/none
5: Gu: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop st
    ate DOWN group default qlen 500
    link/none
6: Gu0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER UP> mtu 1500
    qdisc pfifo fast state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global Gu0
    valid lft forever preferred lft forever
    inet6 fe80::94:c451:f68b:a302/64 scope link flags 800
    valid lft forever preferred lft forever
[04/07/22]seed@VM:~$

```

d.

## V. Task 2.c Read from the RUN Interface

- a. Modify tun.py.

```

SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminator
/bin/bash

/bin/bash 59x24
import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Gu0d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if True:
        ip = IP(packet)
        print(ip.summary())

-- INSERT --
24,3-17 Bot

```

b.

- c. Also added these lines to configure automatically.

```

SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminator
/bin/bash

/bin/bash 65x25
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Gu0d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.99/24 dev {} {}".format(ifname))
os.system("ip link set dev {} up {}".format(ifname))

while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if True:
        ip = IP(packet)
        print(ip.summary())

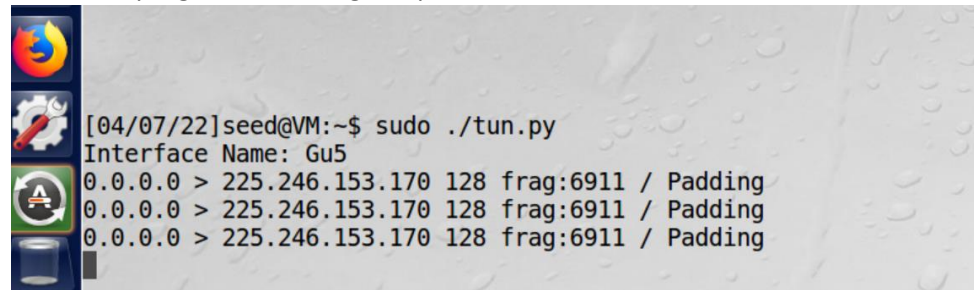
28,1-8 Bot

```

d.

- e. Run tun.py and

- i. When the program is running, outputs:

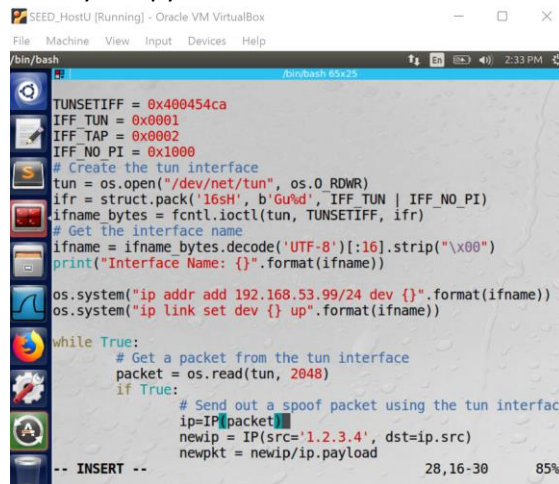


```
[04/07/22]seed@VM:~$ sudo ./tun.py
Interface Name: Gu5
0.0.0.0 > 225.246.153.170 128 frag:6911 / Padding
0.0.0.0 > 225.246.153.170 128 frag:6911 / Padding
0.0.0.0 > 225.246.153.170 128 frag:6911 / Padding
```

- 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.



```
TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

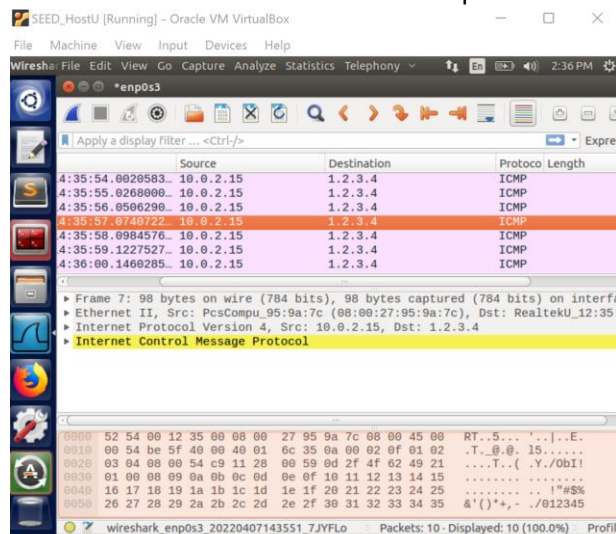
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Gu5d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if True:
        # Send out a spoof packet using the tun interface
        ip=IP(packet)
        newip = IP(src='1.2.3.4', dst=ip.src)
        newpkt = newip/ip.payload
        -- INSERT --
```

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

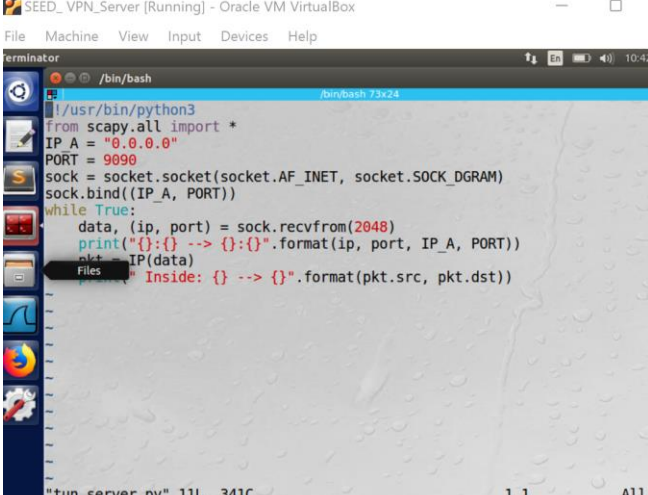


Time	Source	Destination	Protocol	Length
4:35:54.0020583	10.0.2.15	1.2.3.4	ICMP	
4:35:55.0268000	10.0.2.15	1.2.3.4	ICMP	
4:35:56.0506290	10.0.2.15	1.2.3.4	ICMP	
4:35:57.0747228	10.0.2.15	1.2.3.4	ICMP	
4:35:58.0984576	10.0.2.15	1.2.3.4	ICMP	
4:35:59.1227527	10.0.2.15	1.2.3.4	ICMP	
4:36:00.1460285	10.0.2.15	1.2.3.4	ICMP	

- 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.

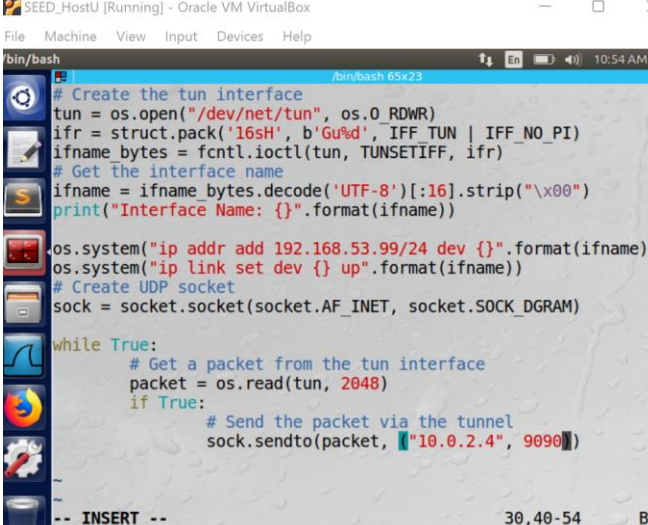
## VII. Task 3 Send the IP Packet to VPN Server Through a Tunnel

- a. On VPN server, create tun\_server.py and modify tun.py to tun\_client.py on host U.



```
SEED_VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
terminator
/bin/bash
~/bin/bash 73x24
~/usr/bin/python3
from scapy.all import *
IP_A = "0.0.0.0"
PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
while True:
    data, (ip, port) = sock.recvfrom(2048)
    print("{}: {} --> {}".format(ip, port, IP_A, PORT))
    IP(data)
    print("Inside: {} --> {}".format(pkt.src, pkt.dst))
```

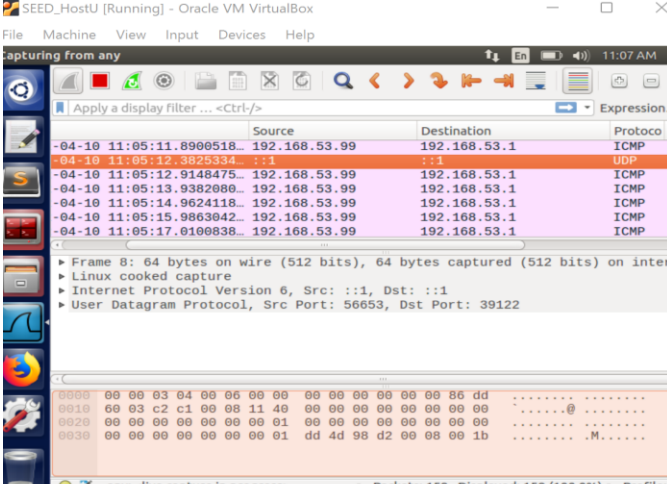
- b.



```
SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
~/bin/bash 65x23
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Gu%d', IFF_TUN | IFF_NO_PI)
ifname bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if True:
        # Send the packet via the tunnel
        sock.sendto(packet, ("10.0.2.4", 9090))
```

- 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.

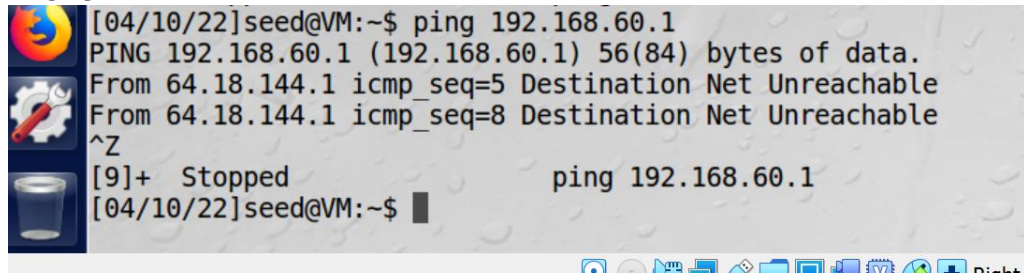


```
SEED_HostU [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Capturing from any
Apply a display filter ... <Ctrl-/>
Expression...
-04-10 11:05:11.8900518... 192.168.53.99 192.168.53.1 ICMP
-04-10 11:05:12.8602635... 192.168.53.99 192.168.53.1 ICMP
-04-10 11:05:12.9148475... 192.168.53.99 192.168.53.1 ICMP
-04-10 11:05:13.9382080... 192.168.53.99 192.168.53.1 ICMP
-04-10 11:05:14.9624118... 192.168.53.99 192.168.53.1 ICMP
-04-10 11:05:15.9863042... 192.168.53.99 192.168.53.1 ICMP
-04-10 11:05:17.0100838... 192.168.53.99 192.168.53.1 ICMP
Frame 8: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on inter
Linux cooked capture
Internet Protocol Version 6, Src: ::1, Dst: ::1
User Datagram Protocol, Src Port: 56653, Dst Port: 39122
```

- e.



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



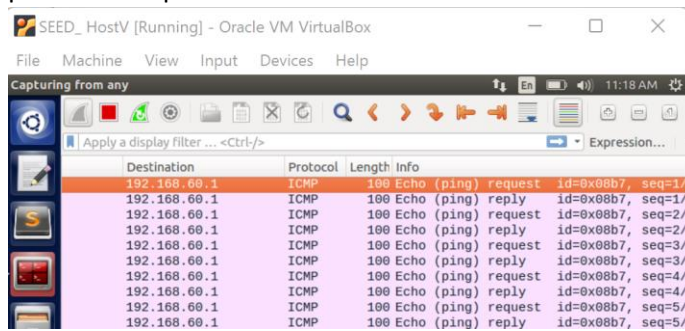
```
[04/10/22]seed@VM:~$ ping 192.168.60.1
PING 192.168.60.1 (192.168.60.1) 56(84) bytes of data.
From 64.18.144.1 icmp_seq=5 Destination Net Unreachable
From 64.18.144.1 icmp_seq=8 Destination Net Unreachable
^Z
[9]+  Stopped                  ping 192.168.60.1
[04/10/22]seed@VM:~$
```

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



```
Terminator
/bin/bash
[04/10/22]seed@VM:~$ sudo ip route add 192.168.60.1 dev Gu1 via 192.168.53.99
```

- h. 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

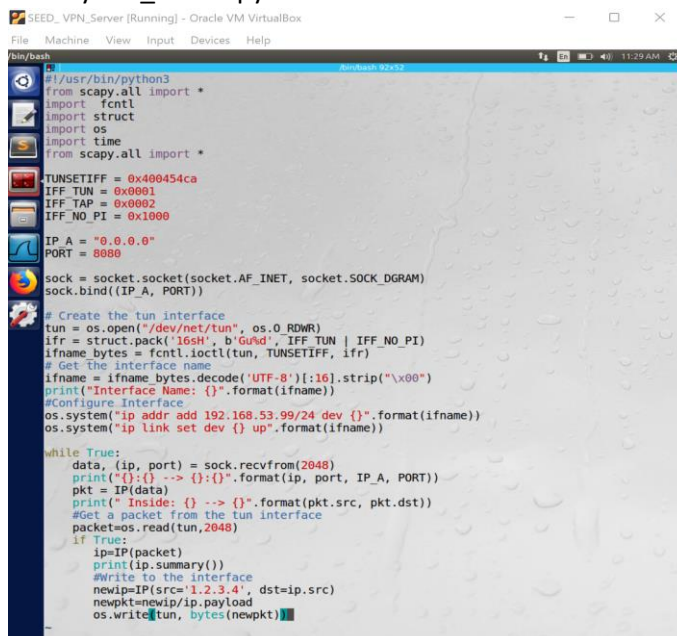


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) request id=0x08b7, seq=1/
2	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) reply id=0x08b7, seq=1/
3	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) request id=0x08b7, seq=2/
4	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) reply id=0x08b7, seq=2/
5	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) request id=0x08b7, seq=3/
6	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) reply id=0x08b7, seq=3/
7	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) request id=0x08b7, seq=4/
8	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) reply id=0x08b7, seq=4/
9	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) request id=0x08b7, seq=5/
10	0.000000	192.168.60.1	192.168.60.1	ICMP	100	Echo (ping) reply id=0x08b7, seq=5/

- j.

## VIII. Task 4 Set Up the VPN Server

- a. Modify tun\_server.py



```
SEED_VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
#!/usr/bin/python3
from scapy.all import *
import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

IP_A = "0.0.0.0"
PORT = 8080

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Gu1d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
# Configure interface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

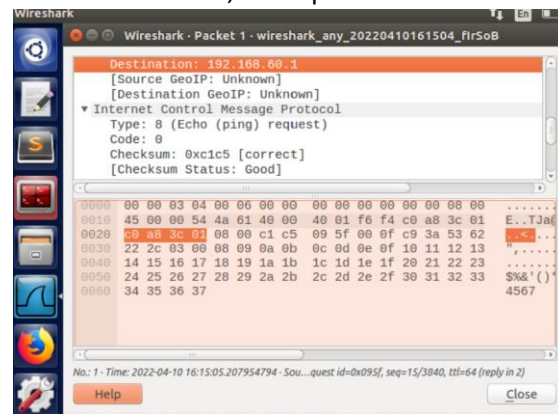
while True:
    data, (ip, port) = sock.recvfrom(2048)
    print("{}:({}) -> {}:({})".format(ip, port, IP_A, PORT))
    pkt = IP(data)
    print("Inside: {} -> {}".format(pkt.src, pkt.dst))
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if True:
        ip = IP(packet)
        print(ip.summary())
        # Write to the interface
        newip = IP(src='1.2.3.4', dst=ip.src)
        newpkt = newip/ip.payload
        os.write(tun, bytes(newpkt))
```

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



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

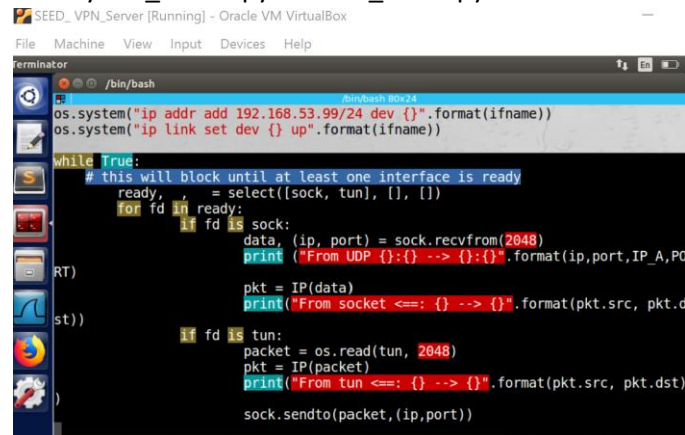
- d.
- e. And on Wireshark, ICMP packets have reached host V.



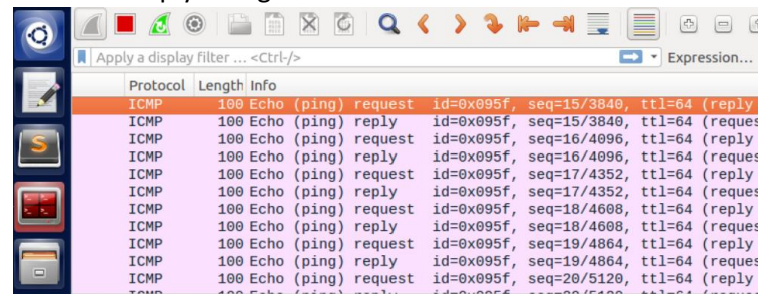
f.

## 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.



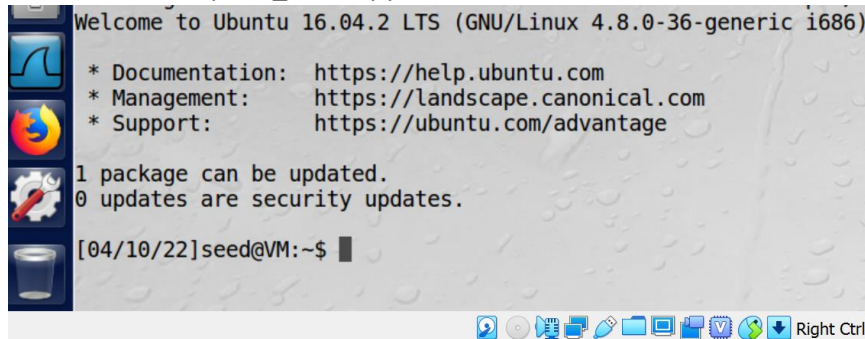
- b.
- c. And while running scripts, ping host V from host U, ICMP request that goes to Host V and ICMP reply that goes back to host U are observed.



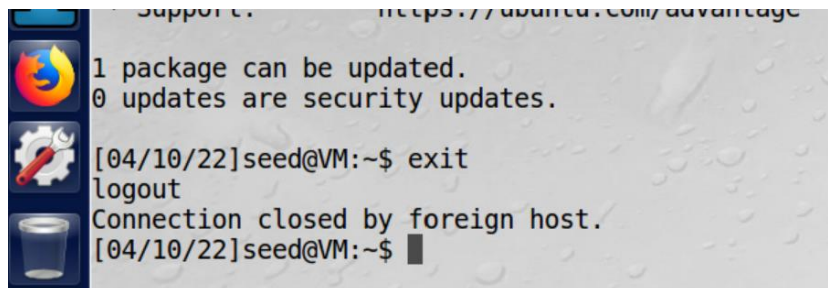
d.

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



d.