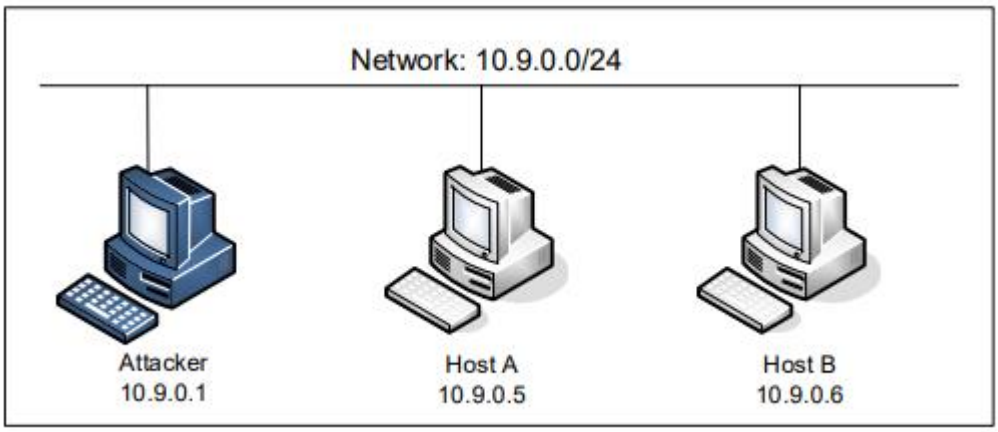
# COMPUTER NETWORK SECURITY LABORATORY

| NAME : PREM SAGAR J S | SRN : PES1UG20CS825 | SEC : H |
|---|---|---|

## Sniffing and Spoofing Lab

## Lab Task Set-1: Using Tools to Sniff and Spoof Packets using Scapy

Lab Environment Setup




## Task 1.1 : Sniffing Packets

## Task 1.1 A : Sniff IP packets using Scapy

The given program ,for each captured packet, the callback function print pkt() will be invoked; this function will print out some of the information about the packet.

First I'm going to ping 8.8.8.8 which is google's DNS Server on Host A machine then i'm gonna use the sniffer code to sniff those packets.

**On the Host A terminal running the command:**
**# ping 8.8.8.8**

```
seed@VM: ~/.../Labsetup
Seed-HostA:PES1UG20CS825:Prem Sagar:/
$>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=111 time=177 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=111 time=88.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=111 time=111 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=111 time=129 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=111 time=150 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=111 time=53.7 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=111 time=57.4 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=111 time=50.6 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=111 time=79.7 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=111 time=45.6 ms
^C
--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9050ms
```

I have Replaced the interface in the code with attacker system's interface.

Now I'm Running the program with the root privilege.

**On the Attacker terminal running the command:**
**# python3 Task1.1A.py**

```
seed@VM: ~/.../Labsetup
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>python3 Task1.1A.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 02:42:97:de:da:53
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 60686
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x337d
     src       = 10.9.0.5
     dst       = 8.8.8.8
     \options   \
```

Above screenshot shows that we can indeed capture packets, packets are captured by sniffer code While Host A making requests to the Machine 8.8.8.8

Explain on which VM you ran this command and why?

-> I'm running these commands on the seed-attacker VM, because as an attacker I have to sniff the packets from hosts present in the same network, it doesn't make any sense if run this commands on the victims system.

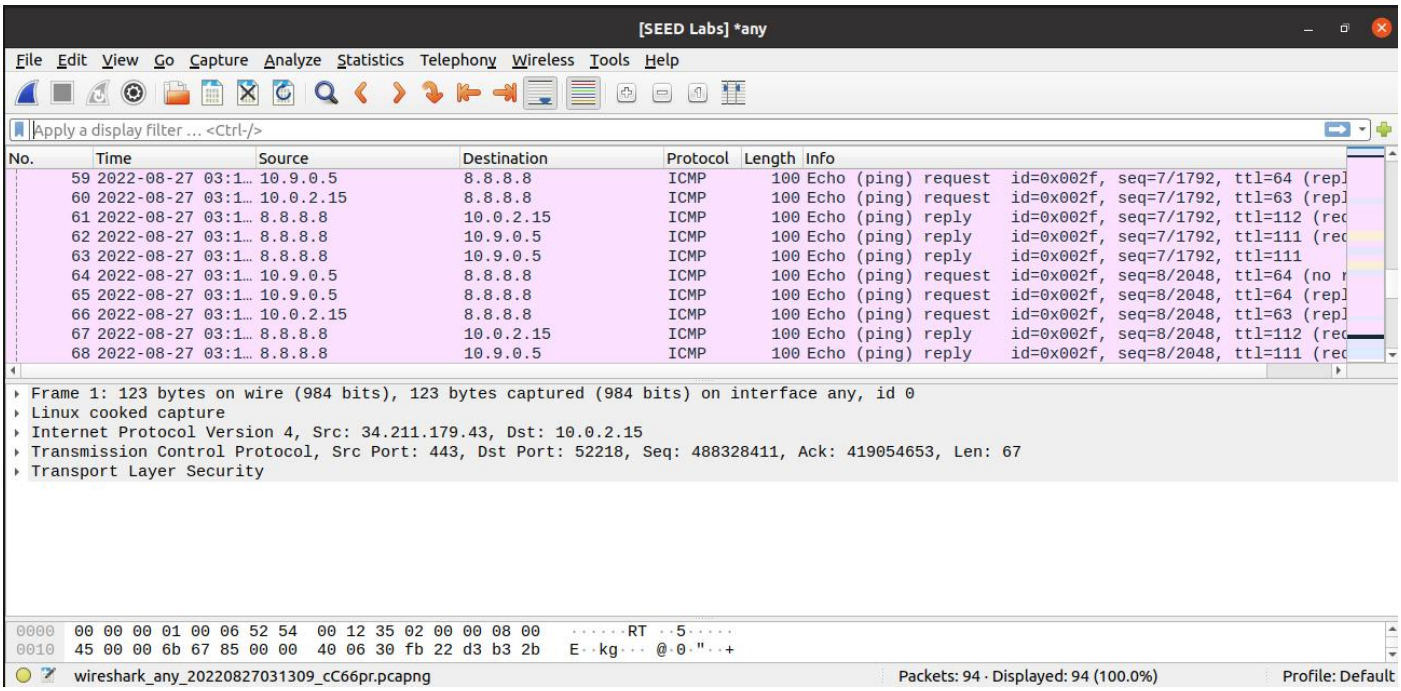
Running the same command without the root privileges and to see if it works

```
seed@VM: ~/.../Labsetup
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>su seed
seed@VM:/volumes/Code$ python3 Task1.1A.py
SNIFFING PACKETS...
Traceback (most recent call last):
  File "Task1.1A.py", line 6, in <module>
    pkt = sniff(iface = "br-0cf5ff514f8b",prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 906, in _run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))  # noqa: E501
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
seed@VM:/volumes/Code$
```

Above Screenshot shows that I'm not able run the commands without root privileges as promiscuous mode cannot be turned on the user mode,

User needs root privilege to turn on promiscuous mode.

**Wireshark screenshot of capturing same packets as the sniffer script**



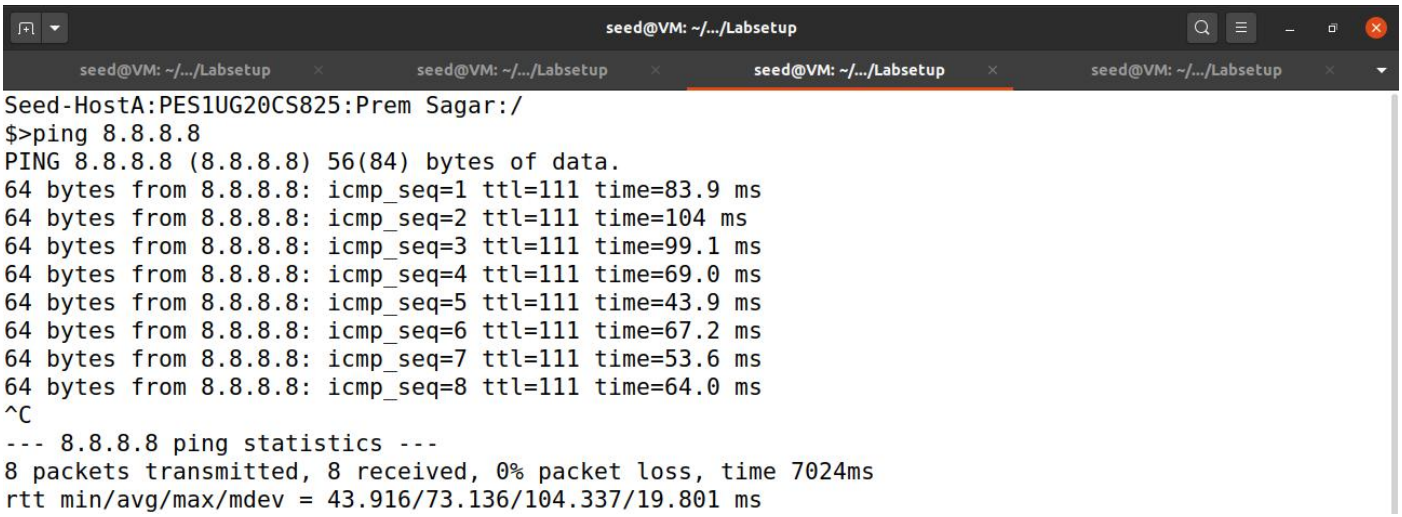

# Task 1.1 B : Capturing ICMP, TCP packet and Subnet

when we sniff packets, we are only interested in certain types of packets.

**Capture only the ICMP packet**

Sniffer program sniffs when some machine on the same network sends ping requests, the packets get captured by the sniffer.

In this program we specifically capturing ICMP Packets.
From the **host A** machine's terminal ping a random IP address(8.8.8.8)



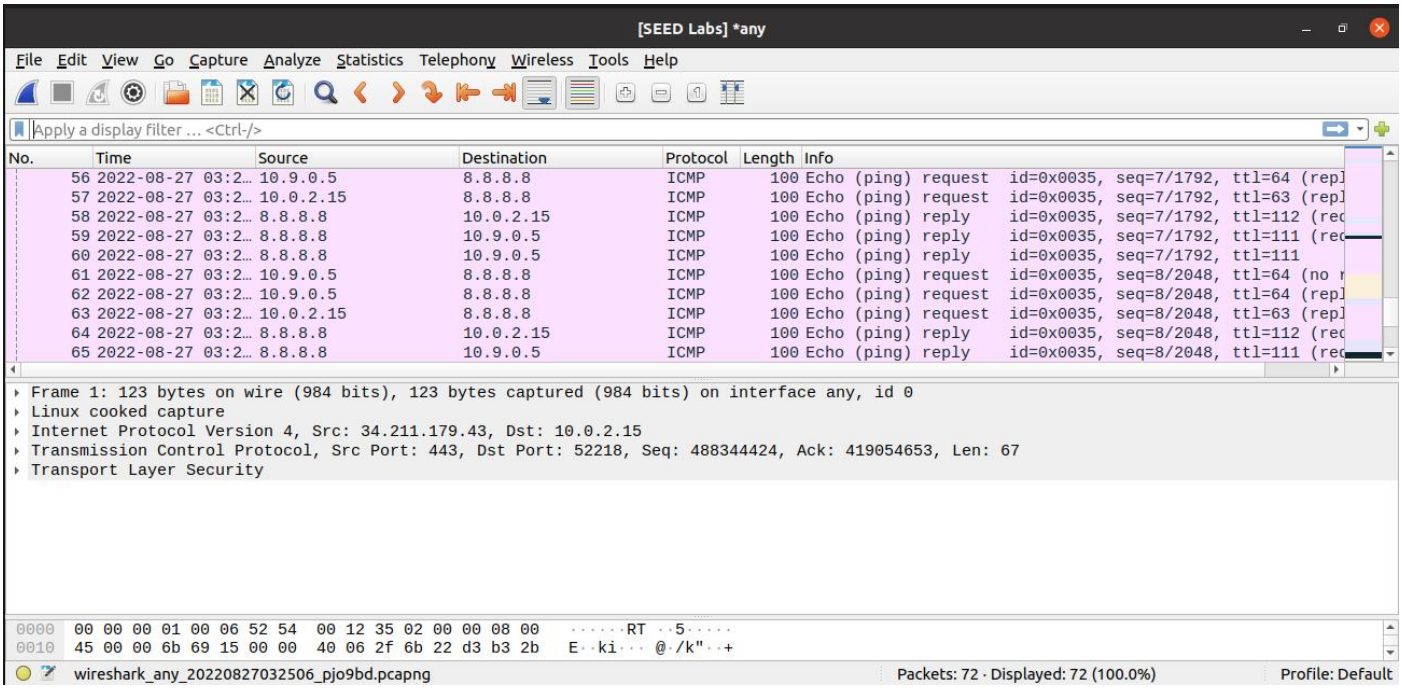
I have Replaced the interface in the code with attacker system's interface.

Now I'm Running the program with the root privilege in the Attacker's Machine.

```
seed@VM: ~/.../Labsetup
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>python3 Task1.1B-ICMP.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:97:de:da:53
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0xa
     len       = 84
     id        = 26909
     flags     =
     frag      = 0
     ttl       = 111
     proto     = icmp
     chksum    = 0xc864
     src       = 8.8.8.8
     dst       = 10.9.0.5
     \options   \
```

As you can see in the above screen shot that we able capture only ICMP packets and the filter is indeed working fine and packets has been captured By the Attacker which are been sent out by Host A while pinging 8.8.8.8.

**Wireshark screenshot of capturing same packets as the sniffer script :**




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

**On the Host A terminal running the command::**
**# telnet 10.9.0.1**

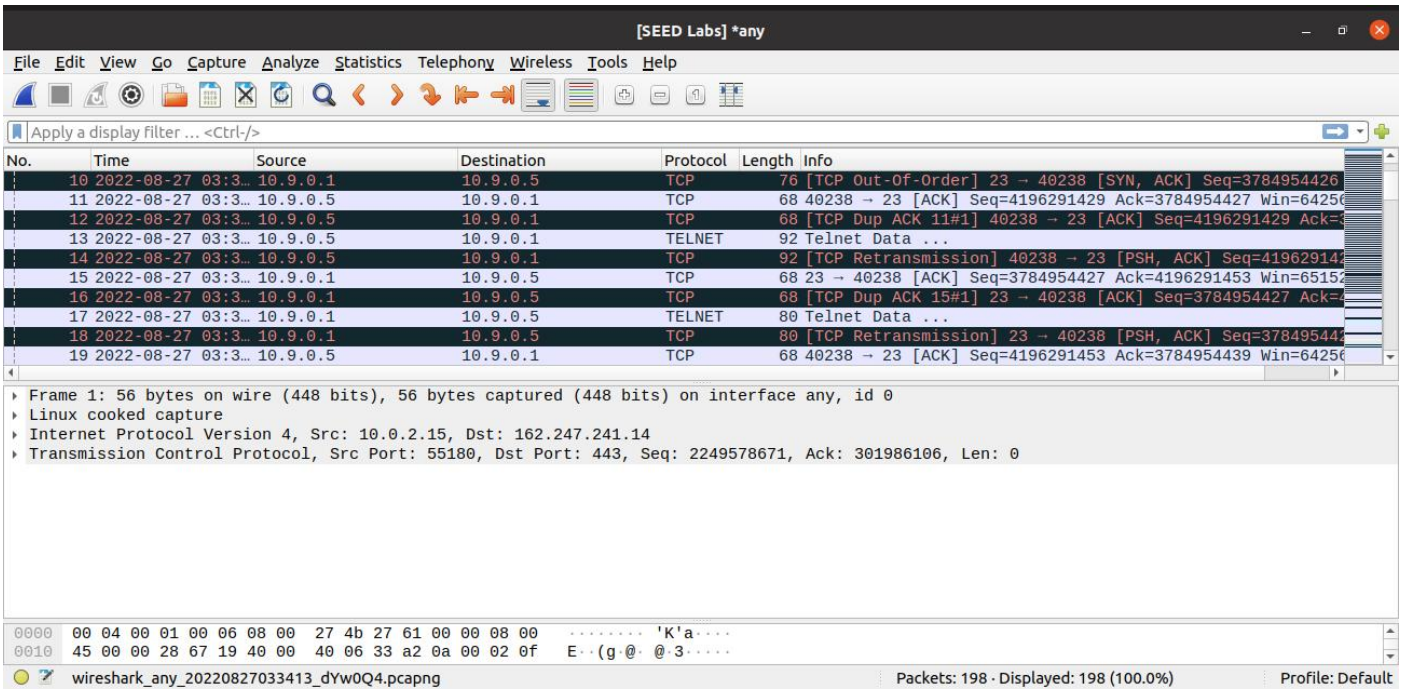
```
seed@VM: ~
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~   seed@VM: ~/.../Labsetup
Seed-HostA:PES1UG20CS825:Prem Sagar:/
$>telnet 10.9.0.1
Trying 10.9.0.1...
Connected to 10.9.0.1.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

0 updates can be installed immediately.
0 of these updates are security updates.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
```

**On the Attacker terminal running the command:**

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~   seed@VM: ~/.../Labsetup
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>python3 Task1.1B-TCP.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 02:42:97:de:da:53
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 40258
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x8952
     src       = 10.9.0.5
     dst       = 10.9.0.1
     \options   \
```

The program will capture the TCP packets being sent from the specified IP address on the port 23, As the Host A is trying the telnet protocol to remote login to the machine with IP 10.9.0.1 Sending out TCP packets which are already been captured by our sniffer code as shown in the above screenshot .

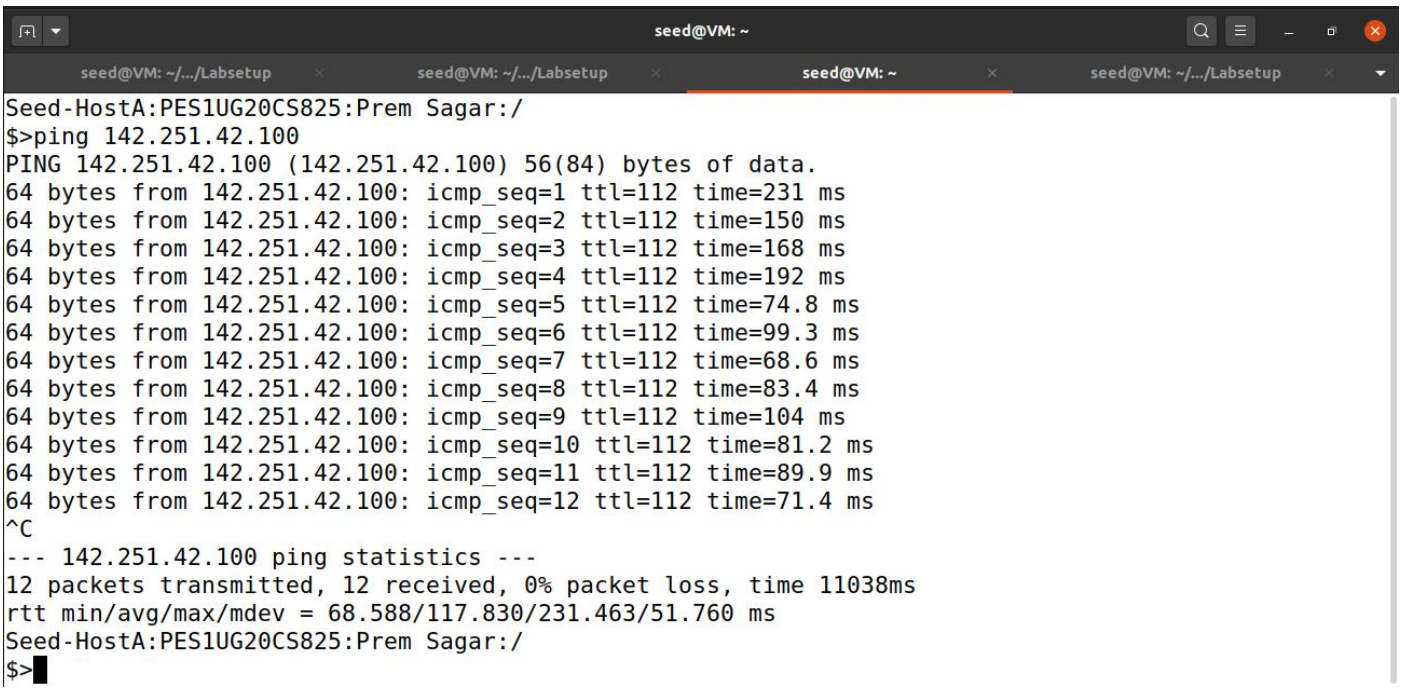
**Wireshark screenshot of capturing packets same as the Sniffer script:**




**Capture packets that come from or go to a particular subnet**

**On the Host A terminal running the command::**

**Pinging to the google's subnet IP 142.251.42.100 on the Host A machine :**

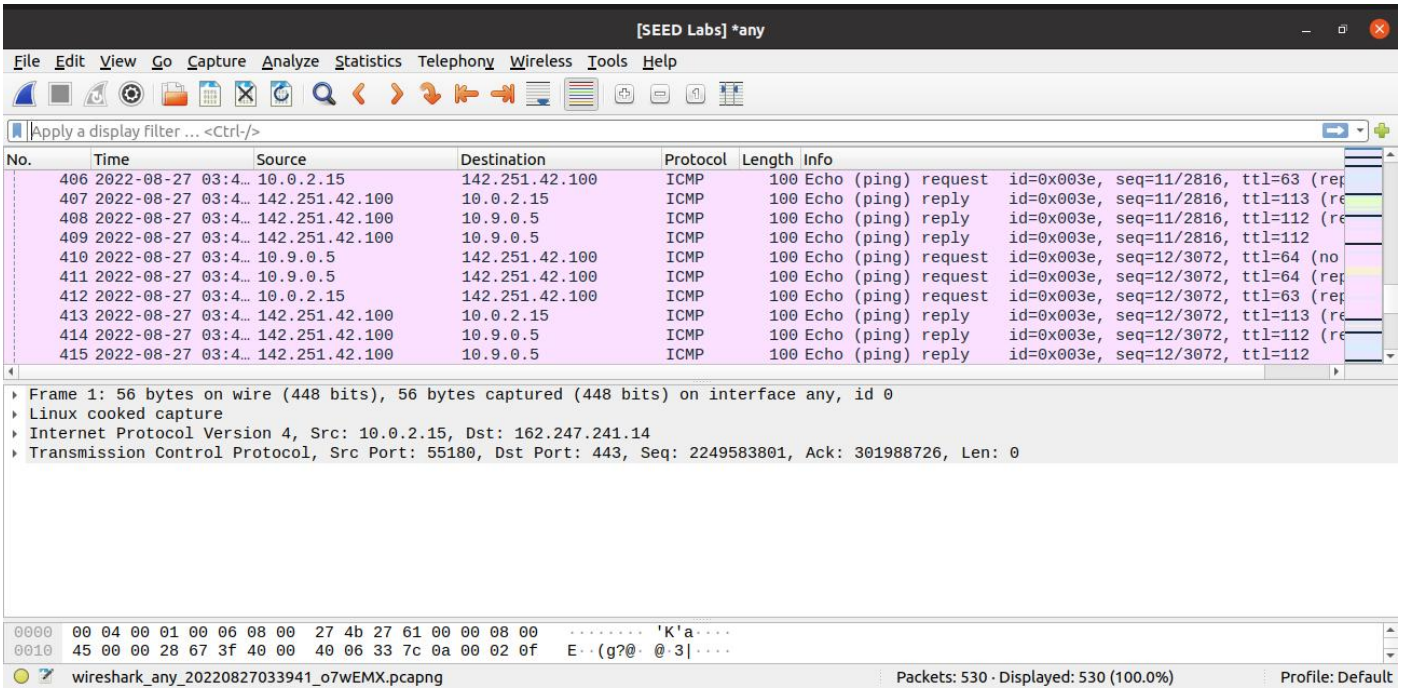


**On the Attacker terminal running the command:**

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~   seed@VM: ~/.../Labsetup
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>python3 Task1.1B-Subnet.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:97:de:da:53
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x18
     len       = 84
     id        = 27442
     flags     =
     frag      = 0
     ttl       = 112
     proto     = icmp
     chksum    = 0x1bf2
     src       = 142.251.42.100
     dst       = 10.9.0.5
     \options   \
```

I have replace the attackers interface in this code.
We are sending ICMP packets to 142.251.42.100 from the Host A machine, the sniffer program capturing the packets sent out from 142.251.42.100, As you can see in the above screen sniffer successfully captured the ICMP packets sent out from the 142.251.42.100.

**Wireshark screenshot of capturing packets same as the sniffer script :**



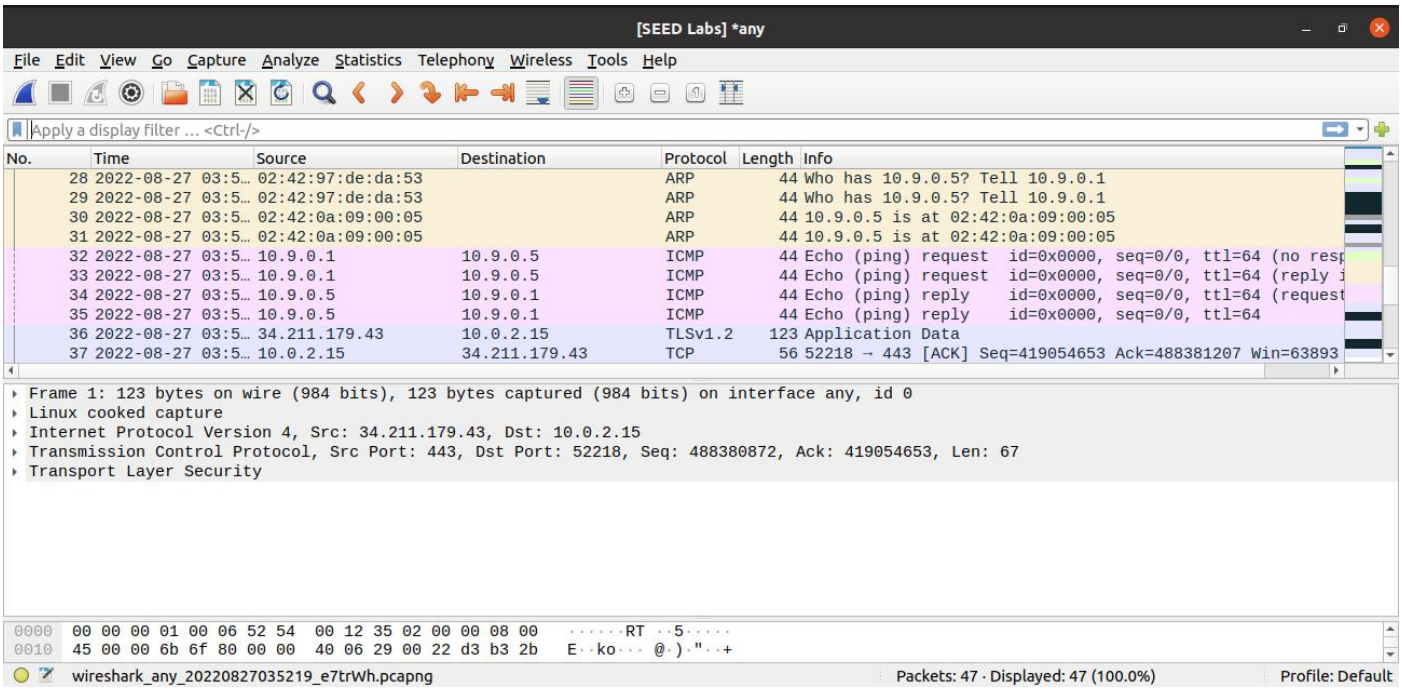

# Task 1.2 : Spoofing

**On the Attacker terminal run the command:**

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~   seed@VM: ~/.../Labsetup
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>python3 Task1.2A.py
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       = 10.9.0.1
  dst       = 10.9.0.5
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
```

The Sniffer code will spoof **ICMP echo request packets** and send them to another VM on the same network.
We are using Wireshark to observe whether our request will be accepted by the receiver. If it is Accepted.

**Wireshark screenshot of spoofed packets ：**



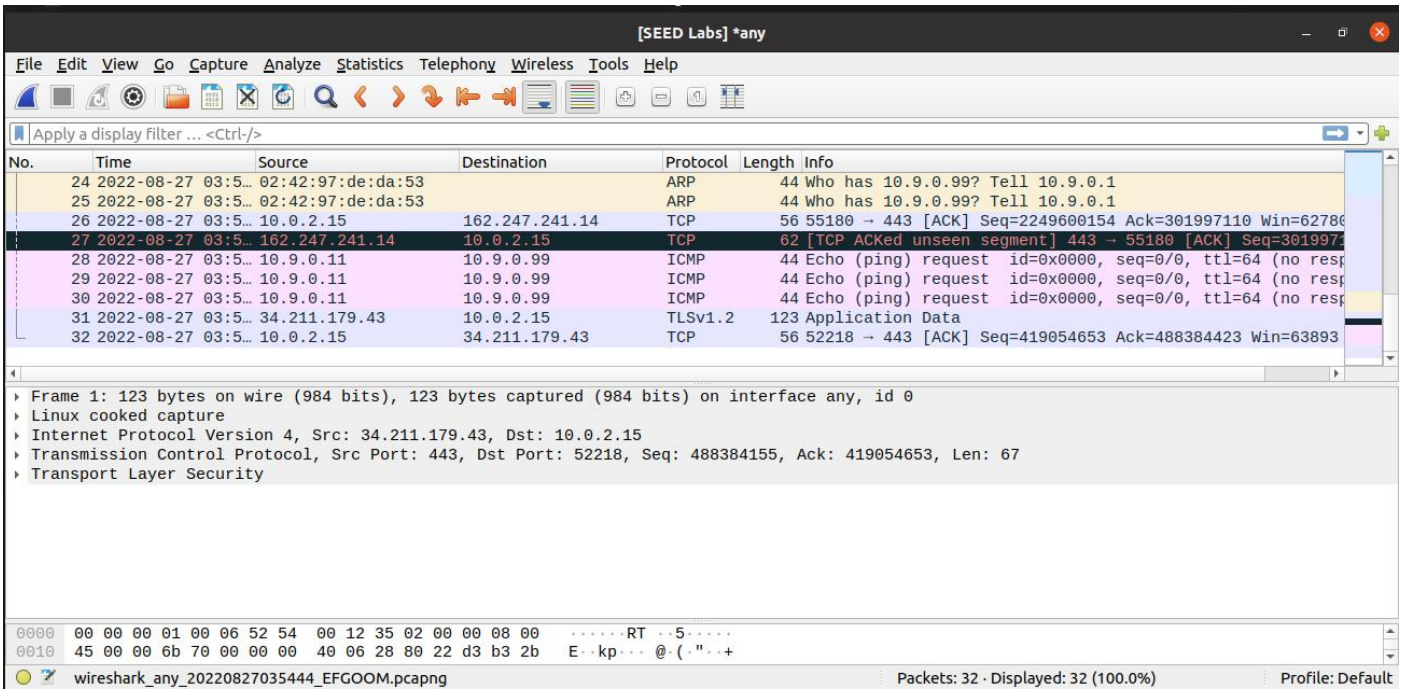

Request As been accepted by the receiver echo reply packet has been be sent to the spoofed IP address.
The spoofed request is formed by creating our own packet with
the header specifications.

Spoofing an ICMP echo request packet with an **arbitrary source IP address**.

**On the Attacker terminal run the command:**
**# python Task1.2B.py**

```
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>python3 Task1.2B.py
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       = 10.9.0.11
  dst       = 10.9.0.99
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
```

**Wireshark capture of spoofed packets :**



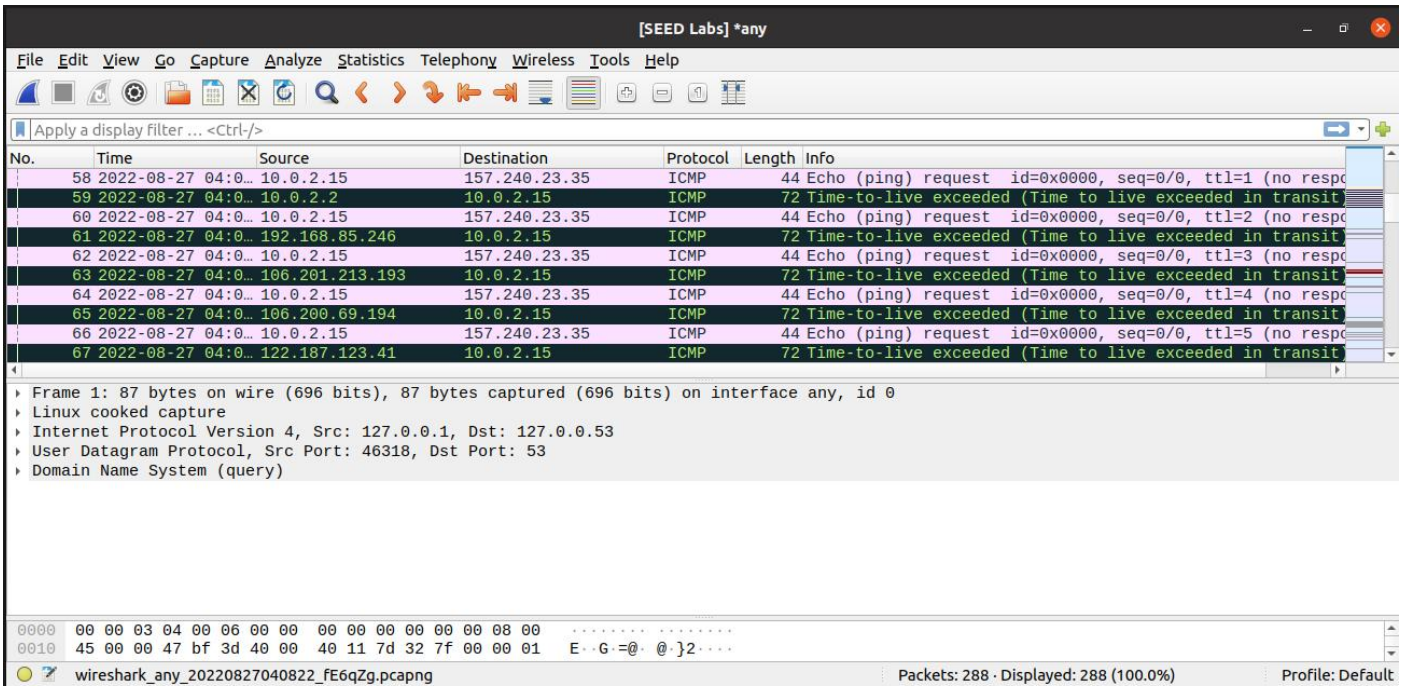

# Task 1.3 : Traceroute

**On the Attacker terminal running the command:**
**# python3 Task1.3.py 157.240.23.35**

**157.240.23.35 is the IP address for facebook.com**

```
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>python3 Task1.3.py 157.240.23.35
Traceroute 157.240.23.35
1 hops away:  10.0.2.2
2 hops away:  192.168.85.246
3 hops away:  106.201.213.193
4 hops away:  106.200.69.194
5 hops away:  122.187.123.41
6 hops away:  182.79.177.69
7 hops away:  157.240.84.206
8 hops away:  129.134.34.153
9 hops away:  157.240.38.121
10 hops away:  157.240.23.35
Done 157.240.23.35
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>
```

**Wireshark Screenshot:**




We created an IP packet with destination address and TTL value and ICMP packet. We sent the packet using function sr1(). This function waits for the reply from the destination. If the ICMP reply type is 0, we receive an echo response from the destination, else we increase the TTL value and resend the packet.

As shown in the above screenshots our code worked successfully and we got distance, in terms of number of routers, between our VM and a selected destination.
the route info with number of Hops and time taken.

# Task 1.4 : Sniffing and-then Spoofing

**On the Attacker terminal running the command:**
**# python3 Task1.4.py**

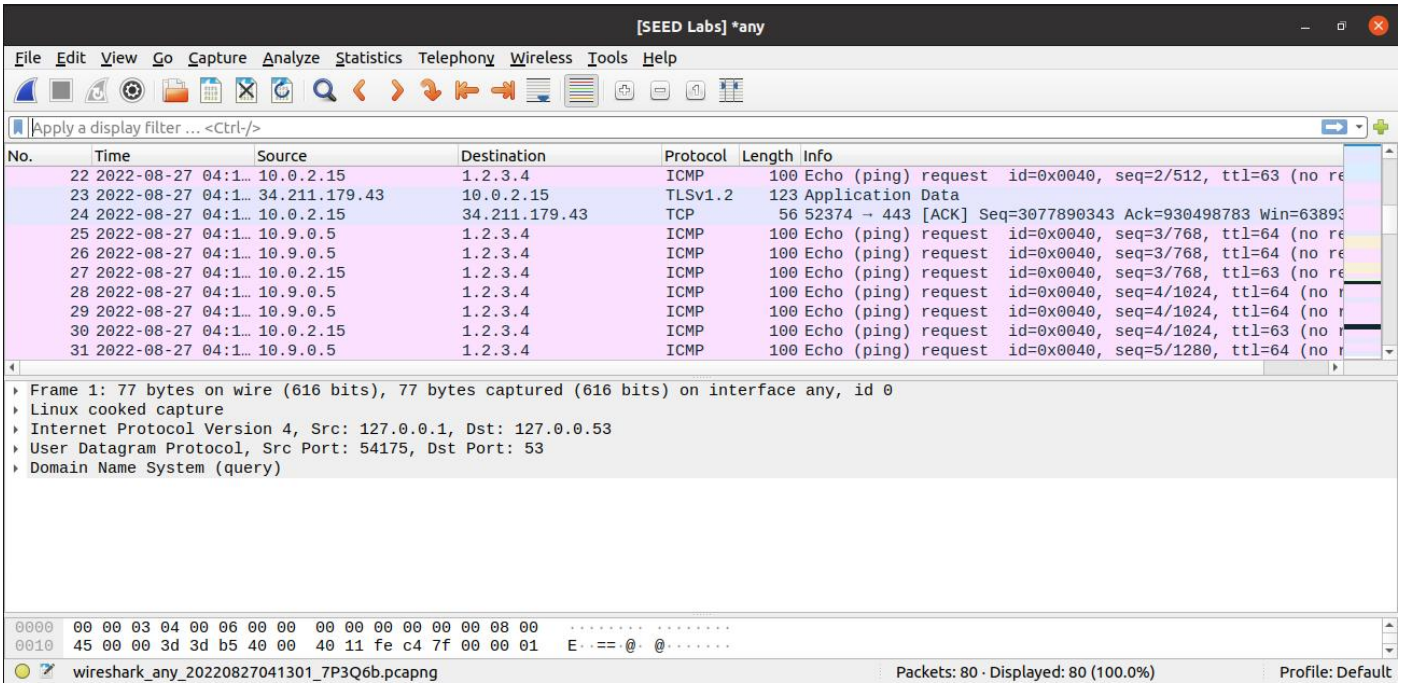
```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~   seed@VM: ~/.../Labsetup
Seed-Attacker:PES1UG20CS825:Prem Sagar:/volumes/Code
$>python3 Task1.4.py
original packet.........
source IP : 10.9.0.5
Destination IP : 1.2.3.4
spoofed packet........
Source IP: 1.2.3.4
Destination IP: 10.9.0.5
original packet.........
source IP : 10.9.0.5
Destination IP : 1.2.3.4
spoofed packet........
Source IP: 1.2.3.4
Destination IP: 10.9.0.5
original packet.........
source IP : 10.9.0.5
Destination IP : 1.2.3.4
spoofed packet........
Source IP: 1.2.3.4
Destination IP: 10.9.0.5
original packet.........
```

From the **host A** machine's terminal ping 1.2.3.4

**On the Host A terminal running the command::**
**# ping 1.2.3.4**

```
seed@VM: ~
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~   seed@VM: ~/.../Labsetup
Seed-HostA:PES1UG20CS825:Prem Sagar:/
$>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=5 ttl=64 time=155 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=81.7 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=36.1 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=28.6 ms

--- 1.2.3.4 ping statistics ---
11 packets transmitted, 4 received, 63.6364% packet loss, time 10151ms
rtt min/avg/max/mdev = 28.559/75.236/154.559/50.103 ms
Seed-HostA:PES1UG20CS825:Prem Sagar:/
$>
```

**Wireshark screenshot :**




The victim machine pinging a non-existing IP address “1.2.3.4”.
As the attacker machine is on the same network, machine sniffs the request packet, created a new echo reply packet with IP and ICMP header and sent 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.

We are retrieving source IP and destination IP from the sniffed packet and create a new IP packet. The new source IP of the spoofed packet is the sniffed packet’s destination IP address and vice versa.

As shown in the above screenshots we are able to sniff and spoof successfully packets.