COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

5th Semester E section

1. Task1.1A

a) The given command was run on the attacker terminal (with root privileges). The attacker is in turn going to sniff the packets in the network and therefore this code is executed on the attacker's end. The code provides header information of the four layers (Ethernet, IP, TCP and Raw data). The output is a continuous stream of all the packets' headers that are being exchanged in the network. One such packet information screenshot is put up below-

OUTPUT SCREENSHOT

```
PES1UG20CS280_ROOT(10.0.2.5) -$python3 Task1.1A.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:94:43:70
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 14891
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xf4e2
     src       = 10.0.2.5
     dst       = 34.107.221.82
     \options   \
###[ TCP ]###
        sport     = 52490
        dport     = http
        seq       = 1561905654
        ack       = 351454
        dataofs   = 5
        reserved  = 0
        flags     = A
        window    = 30016
        chksum    = 0xbdd
        urgptr    = 0
        options   = ''
```

```
###[ Padding ]###
        load      = '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00'

###[ Ethernet ]###
  dst       = 08:00:27:94:43:70
  src       = 52:54:00:12:35:00
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 107
     id        = 20580
     flags     =
     frag      = 0
```

COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

When ping 8.8.8.8 is done on Host A , the client(Host A or Alice in this case) pings the server(8.8.8.8) sending out echo request messages to the server. ICMP packets are sent out from Alice/Host A. So when the attacker sniffs the packets in the network, ICMP packets are also displayed along with the other packets in the network. The client(Alice) keeps requesting the server and so ICMP packets are seen constantly.

OUTPUT SCREENSHOT

```
PES1UG20CS280_ROOT(10.0.2.5) -$python3 Task1.1A.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:c6:fa:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 08:00:27:c6:fa:69
     psrc      = 10.0.2.4
     hwdst     = 00:00:00:00:00:00
     pdst      = 10.0.2.1
###[ Padding ]###
        load      = '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'

###[ Ethernet ]###
  dst       = 08:00:27:c6:fa:69
  src       = 52:54:00:12:35:00
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
```

```
###[ Raw ]###
        load      = 'w:\x08c\\xee\\xfa\x03\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       = 52:54:00:12:35:00
  src       = 08:00:27:c6:fa:69
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 60028
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x3419
     src       = 10.0.2.4
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xae96
        id        = 0xc18
        seq       = 0x14
        unused    = ''
###[ Raw ]###
        load      = 'x:\x08c\x04\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'
```

When the file is executed without the root privileges then the below output is generated. Execution operation is only permitted in root and hence the output is as shown.

COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

OUTPUT SCREENSHOT

```
PES1UG20CS280(10.0.2.5) -$python3 Task1.1A.py
SNIFFING PACKETS...
Traceback (most recent call last):
  File "Task1.1A.py", line 6, in <module>
    pkt = sniff(iface = "enp0s3",prn=print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 1263, in
 sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 1128, in
 _run
    **karg)] = iface
  File "/usr/local/lib/python3.5/dist-packages/scapy/arch/linux.py", line 487, i
n __init__
    socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))
  File "/usr/lib/python3.5/socket.py", line 134, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
PES1UG20CS280(10.0.2.5) -$
```

2. Task 1.1B-ICMP packets

a) Capture only the ICMP packet

```
Terminal
PES1UG20CS280_ROOT(10.0.2.5) -$python3 Task1.1B-ICMP.py
SNIFFING PACKETS...
```

There are no ICMP packets in the network. Therefore, no packet information is displayed on the terminal.

b) ICMP packets on pinging 8.8.8.8 at Host A

```
PES1UG20CS280_ROOT(10.0.2.5) -$python3 Task1.1B-ICMP.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:c6:fa:69
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 40132
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x81d1
     src       = 10.0.2.4
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xe93f
        id        = 0xed4
        seq       = 0x1
        unused    = ''
###[ Raw ]###
           load      = '\\xb6\\x99\x08cN\\xeb\x07\x00\x08\t\n\x0b\x0c\r\x0
```

ICMP packets are sent from Alice to the server (8.8.8.8). These are echo request ICMP messages. When this is done, the attacker is able to see all the ICMP packets that are in the network and the output is as displayed above.

3. Task 1.1B- TCP packets

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

There are no packets being exchanged at port 23 (port used by the Telnet protocol). Therefore, although the device sniffs several other packets in the network, it drops them off because the filter asks for only those packets that have destination port as port 23. Therefore, there is no display of packet information.

```
Terminal
PES1UG20CS280_ROOT(10.0.2.5) -$python3 Task1.1B-TCP.py
SNIFFING PACKETS...
```

b) On using the command telnet 10.9.0.1 at Alice/Host A's terminal

On using telnet, Host A pings the port (port 23) at the server 10.9.0.1 to check if it is open or not. Telnet is a connection-oriented protocol and hence, when it is used, TCP packets are sent out from client to the server. TCP packets are exchanged in the network and are directed to the destination port 23 (which satisfies the filter criteria). Therefore, information about those packets are displayed to us on the screen when the program is executed.

```
PES1UG20CS280_ROOT(10.0.2.5) -$python3 Task1.1B-TCP.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:c6:fa:69
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 5614
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xeb1
     src       = 10.0.2.4
     dst       = 10.9.0.1
     \options   \
###[ TCP ]###
        sport     = 55134
        dport     = telnet
        seq       = 2914030616
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 29200
        chksum    = 0x2495
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (1606134, 0)), ('NOP', None), ('WScale', 7)]
```

COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

Task 1.1B – SUBNET

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

There are no packets being exchanged between systems in the current network and the particular subnet (192.168.254.0/24 in this case). Therefore, no packet information is being displayed on the terminal screen when the program is executed.

```
Terminal
PES1UG20CS280_ROOT(10.0.2.5) -$python3 Task1.1B-Subnet.py
SNIFFING PACKETS...
```

b) When a random IP address is pinged in the chosen subnet

Host A/Alice pinging an IPv4 address in the chosen subnet (192.168.254.1)

```
PES1UG20CS280(10.0.2.4) -$ ping 192.168.254.1
PING 192.168.254.1 (192.168.254.1) 56(84) bytes of data.
64 bytes from 192.168.254.1: icmp_seq=1 ttl=61 time=6.39 ms
64 bytes from 192.168.254.1: icmp_seq=2 ttl=61 time=6.30 ms
64 bytes from 192.168.254.1: icmp_seq=3 ttl=61 time=96.1 ms
64 bytes from 192.168.254.1: icmp_seq=4 ttl=61 time=8.56 ms
64 bytes from 192.168.254.1: icmp_seq=5 ttl=61 time=11.0 ms
64 bytes from 192.168.254.1: icmp_seq=6 ttl=61 time=6.87 ms
64 bytes from 192.168.254.1: icmp_seq=7 ttl=61 time=7.31 ms
64 bytes from 192.168.254.1: icmp_seq=8 ttl=61 time=5.64 ms
64 bytes from 192.168.254.1: icmp_seq=9 ttl=61 time=9.27 ms
^Z
[2]+  Stopped                 ping 192.168.254.1
PES1UG20CS280(10.0.2.4) -$
```

On pinging the subnet, ICMP packets are exchanged between Host A and that particular subnet. The subnet returns a set of Echo-replies back to the host system (Host A) as can be seen above. The program captures these packets and displays it on the screen when the program is executed and hence the output is as shown below.

```
PES1UG20CS280_ROOT(10.0.2.5) -$python3 Task1.1B-Subnet.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 08:00:27:c6:fa:69
  src       = 52:54:00:12:35:00
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 23258
     flags     =
     frag      = 0
     ttl       = 61
     proto     = icmp
     chksum    = 0x5821
     src       = 192.168.254.1
     dst       = 10.0.2.4
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0x420c
        id        = 0xf44
        seq       = 0x1
        unused    = ''
###[ Raw ]###
           load      = '\x00\x9c\x08c\xb1\xac\t\x00\x08\t\n\x0b\x0c\r\x0e\x0f
```

COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

TASK 1.2 → SPOOFING

Task 1.2A – Spoofing a packet from the given IP address to any remote machine that is live

A spoofed ICMP echo request packet is created using Scapy library in the python file. The source IP address is the IP address of a machine on the local network. The spoofed ICMP packet is in turn sent across to the destination IP address (8.8.8.8). The server (8.8.8.8) responds to this by sending an ICMP echo response message. The details of the ICMP packet header can be seen in the screenshot below and the Wireshark capture of request response packets can be seen below the packet detail screenshot.

```
PES1UG20CS280_ROOT(10.0.2.5) -$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       = 10.0.2.5
  dst       = 8.8.8.8
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
     id        = 0x0
     seq       = 0x0
     unused    = ''

PES1UG20CS280_ROOT(10.0.2.5) -$
```

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

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-08-28 09:39:19.0437457…	::1	::1	UDP	64	45709 → 48344 Len=0
2	2022-08-28 09:39:20.3439896…	10.0.2.5	224.0.0.251	MDNS	89	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" quest
3	2022-08-28 09:39:21.3741127…	fe80::7cf3:ab06:1a8…	ff02::fb	MDNS	109	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" quest
4	2022-08-28 09:39:36.1484385…	PcsCompu_94:43:70		ARP	44	Who has 10.0.2.1? Tell 10.0.2.5
5	2022-08-28 09:39:36.1488580…	RealtekU_12:35:00		ARP	62	10.0.2.1 is at 52:54:00:12:35:00
6	2022-08-28 09:39:36.1892379…	10.0.2.5	8.8.8.8	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 7)
7	2022-08-28 09:39:36.2275417…	8.8.8.8	10.0.2.5	ICMP	62	Echo (ping) reply id=0x0000, seq=0/0, ttl=59 (request in 6)
8	2022-08-28 09:39:36.2358293…	::1	::1	UDP	64	45709 → 48344 Len=0

COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

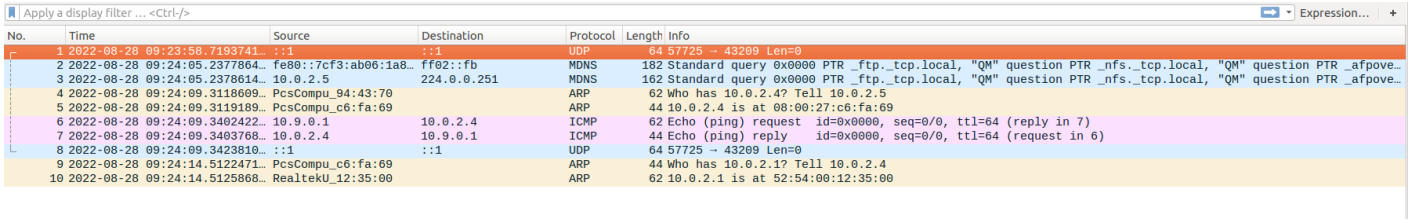
Task 1.2B -- Spoofing an ICMP echo request packet with an arbitrary source IP address.

The given program uses Scapy to create a spoofed ICMP packet to be sent out into the network. An arbitrary source IP address is assigned to the IP source address in the IP header field. The destination IP address is Host A's IP address. The spoofed ICMP packet is sent out into the network by the attacker and all packet header information of that ICMP packet is displayed on the attacker's screen as shown below-

```
PES1UG20CS280_ROOT(10.0.2.5) -$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       = 10.0.2.4
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
     id        = 0x0
     seq       = 0x0
     unused    = ''

PES1UG20CS280_ROOT(10.0.2.5) -$
```

When observed on Wireshark (running on Host A system), we see an Echo request packet reaching Host A from the arbitrary source IP. Host A sends back an Echo response message in turn to this source (the IP may or may not exist in reality). This can be seen in the output screenshot attached below-



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

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-08-28 09:23:58.7193741…	::1	::1	UDP	64	57725 → 43209 Len=0
2	2022-08-28 09:24:05.2377864…	fe80::7cf3:ab06:1a8…	ff02::fb	MDNS	182	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local, "QM" question PTR _afpove…
3	2022-08-28 09:24:05.2378614…	10.0.2.5	224.0.0.251	MDNS	162	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local, "QM" question PTR _afpove…
4	2022-08-28 09:24:09.3118609…	PcsCompu_94:43:70		ARP	62	Who has 10.0.2.4? Tell 10.0.2.5
5	2022-08-28 09:24:09.3119189…	PcsCompu_c6:fa:69		ARP	44	10.0.2.4 is at 08:00:27:c6:fa:69
6	2022-08-28 09:24:09.3402422…	10.9.0.1	10.0.2.4	ICMP	62	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 7)
7	2022-08-28 09:24:09.3403768…	10.0.2.4	10.9.0.1	ICMP	44	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 6)
8	2022-08-28 09:24:09.3423810…	::1	::1	UDP	64	57725 → 43209 Len=0
9	2022-08-28 09:24:14.5122471…	PcsCompu_c6:fa:69		ARP	44	Who has 10.0.2.1? Tell 10.0.2.4
10	2022-08-28 09:24:14.5125868…	RealtekU_12:35:00		ARP	62	10.0.2.1 is at 52:54:00:12:35:00

COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

TASK 1.3 → TRACEROUTE

Task 1.3

When the given code is executed, this is what is displayed on the screen. The source sends an ICMP echo request packet to the server expecting an ICMP response. If the packet gets dropped before it can reach the server, then the router sends out an ICMP message to the host/client stating that the TTL has been exceeded. The code given, then increments the TTL value and resends the echo request packet. Similar response messages are shown on Wireshark and the IP addresses of the router interfaces sending the ICMP TTL exceeded messages are being displayed on the screen.

```
PES1UG20CS280_ROOT(10.0.2.5) -$python Task1.3.py 154.240.23.35
WARNING: No route found for IPv6 destination :: (no default route?)
Traceroute 154.240.23.35
(1, 'hops away:', '10.0.2.1')
(2, 'hops away:', '10.20.200.1')
(3, 'hops away:', '192.168.4.1')
(4, 'hops away:', '192.168.254.1')
```

Every time there is a TTL exceeded message that pops up, the client increments its TTL value and resends the Echo request packet. The resulting ICMP response messages are as shown below

icmp

No.	Time	Source	Destination	Protocol	Length	Info
12	2022-08-26 06:59:37.3301186…	10.0.2.5	154.240.23.35	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (no response found!)
13	2022-08-26 06:59:37.3304265…	10.0.2.1	10.0.2.5	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
15	2022-08-26 06:59:37.4887339…	10.0.2.5	154.240.23.35	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=2 (no response found!)
16	2022-08-26 06:59:37.5021514…	10.20.200.1	10.0.2.5	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
17	2022-08-26 06:59:37.5908246…	10.0.2.5	154.240.23.35	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=3 (no response found!)
18	2022-08-26 06:59:37.6090898…	192.168.4.1	10.0.2.5	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
19	2022-08-26 06:59:37.7042915…	10.0.2.5	154.240.23.35	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=4 (no response found!)
20	2022-08-26 06:59:37.7971830…	192.168.254.1	10.0.2.5	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
21	2022-08-26 06:59:37.8609013…	10.0.2.5	154.240.23.35	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=5 (no response found!)

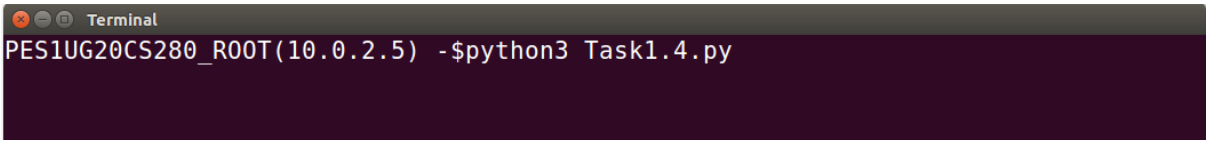
COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

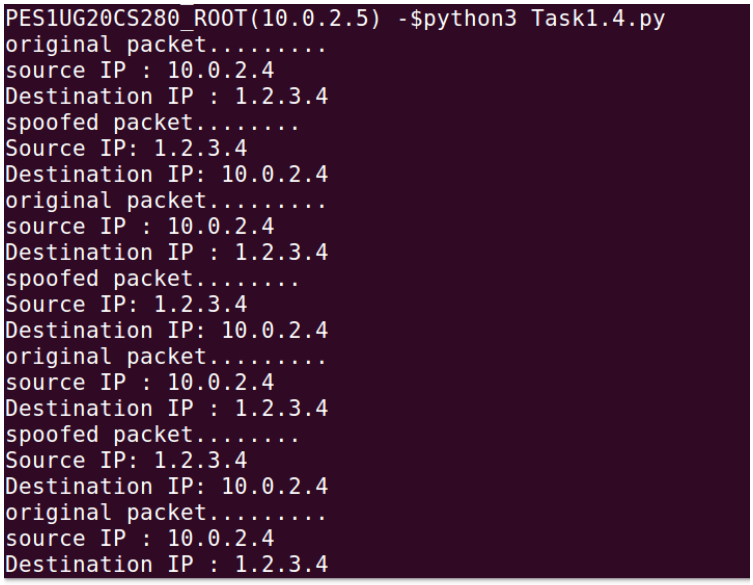
TASK 1.4 → SNIFFING AND SPOOFING

Task 1.4

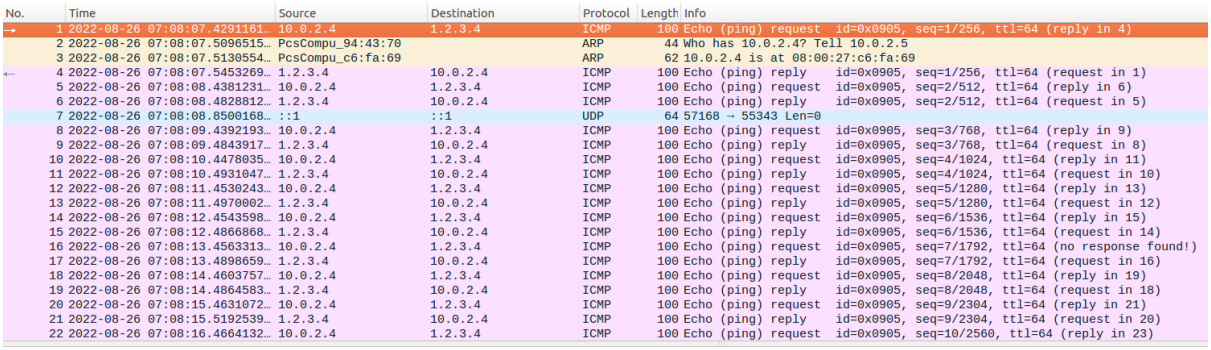
The given program Task1.4 sniffs the network for any ICMP request packets that are directed to an IP address that does not exist (1.2.3.4 in our case). When Host A is not doing anything, there are no such packets in the network and hence no output is displayed when the code is executed.



When Host A pings 1.2.3.4, then this program sniffs those specific ICMP echo request packets and sends out appropriate ICMP echo response packets back to the client. The program displays the original source and destination IPs of the ICMP echo requests and the corresponding source and destination Ips of the spoofed ICMP response packets being sent from the attacker.



When seen on Wireshark, it appears as though 1.2.3.4 is actually responding to the ICMP requests of Host A.



No.	Time	Source	Destination	Protocol	Length	Info
→ 1	2022-08-26 07:08:07.4291161…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=1/256, ttl=64 (reply in 4)
2	2022-08-26 07:08:07.5096515…	PcsCompu_94:43:70		ARP	44	Who has 10.0.2.4? Tell 10.0.2.5
3	2022-08-26 07:08:07.5130554…	PcsCompu_c6:fa:69		ARP	62	10.0.2.4 is at 08:00:27:c6:fa:69
← 4	2022-08-26 07:08:07.5453269…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=1/256, ttl=64 (request in 1)
5	2022-08-26 07:08:08.4381231…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=2/512, ttl=64 (reply in 6)
6	2022-08-26 07:08:08.4828812…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=2/512, ttl=64 (request in 5)
7	2022-08-26 07:08:08.8500168…	::1	::1	UDP	64	57168 → 55343 Len=0
8	2022-08-26 07:08:09.4392193…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=3/768, ttl=64 (reply in 9)
9	2022-08-26 07:08:09.4843917…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=3/768, ttl=64 (request in 8)
10	2022-08-26 07:08:10.4478035…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=4/1024, ttl=64 (reply in 11)
11	2022-08-26 07:08:10.4931047…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=4/1024, ttl=64 (request in 10)
12	2022-08-26 07:08:11.4530243…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=5/1280, ttl=64 (reply in 13)
13	2022-08-26 07:08:11.4970002…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=5/1280, ttl=64 (request in 12)
14	2022-08-26 07:08:12.4543598…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=6/1536, ttl=64 (reply in 15)
15	2022-08-26 07:08:12.4866868…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=6/1536, ttl=64 (request in 14)
16	2022-08-26 07:08:13.4563313…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=7/1792, ttl=64 (no response found!)
17	2022-08-26 07:08:13.4898659…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=7/1792, ttl=64 (request in 16)
18	2022-08-26 07:08:14.4603757…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=8/2048, ttl=64 (reply in 19)
19	2022-08-26 07:08:14.4864583…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=8/2048, ttl=64 (request in 18)
20	2022-08-26 07:08:15.4631072…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=9/2304, ttl=64 (reply in 21)
21	2022-08-26 07:08:15.5192539…	1.2.3.4	10.0.2.4	ICMP	100	Echo (ping) reply id=0x0905, seq=9/2304, ttl=64 (request in 20)
22	2022-08-26 07:08:16.4664132…	10.0.2.4	1.2.3.4	ICMP	100	Echo (ping) request id=0x0905, seq=10/2560, ttl=64 (reply in 23)

COMPUTER NETWORK SECURITY LAB -01

Name: Pavan R Kashyap SRN: PES1UG20CS280

Because Host A is in turn receiving ICMP echo response packets, it displays the corresponding ping results (although 1.2.3.4 does not exist in actuality).

```
PES1UG20CS280(10.0.2.4) -$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=108 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=28.0 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=47.8 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=42.9 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=48.8 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=35.2 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=52.2 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=40.4 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=46.2 ms
64 bytes from 1.2.3.4: icmp_seq=10 ttl=64 time=44.0 ms
64 bytes from 1.2.3.4: icmp_seq=11 ttl=64 time=52.8 ms
```