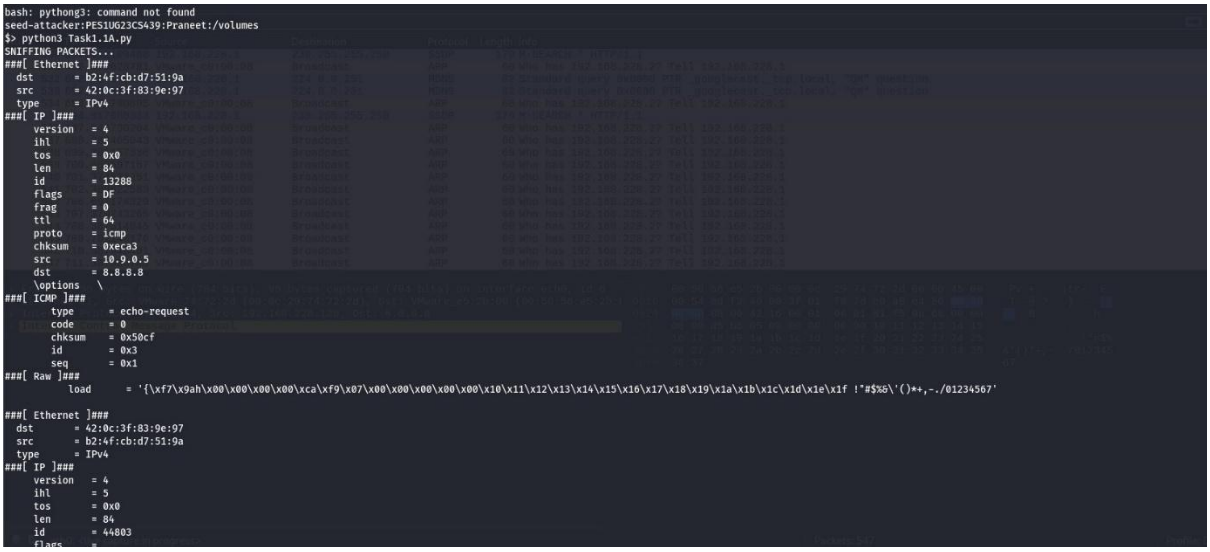
Name: Praneet T.H

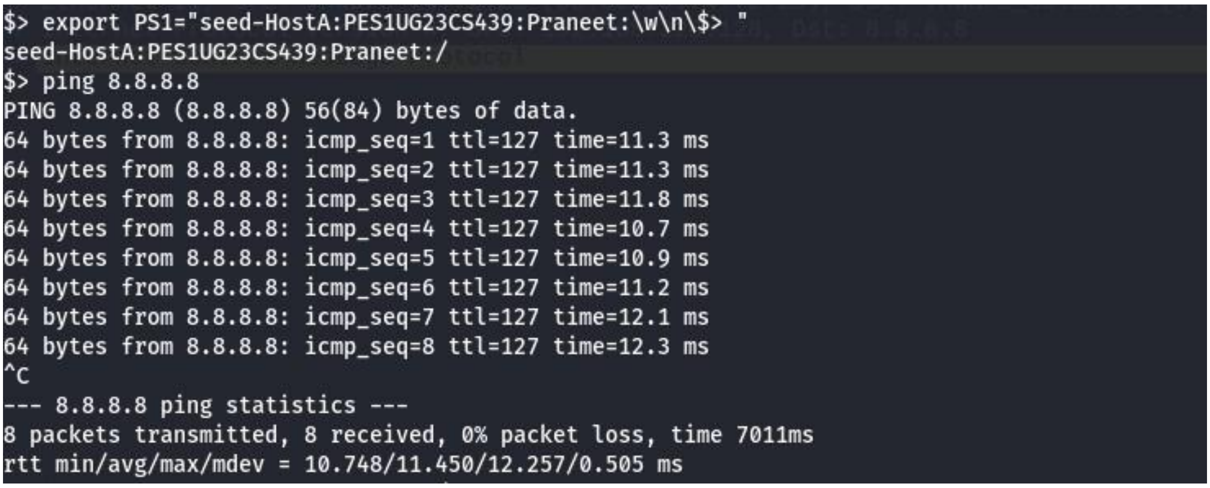
SRN: PES1UG23CS439

Section: H

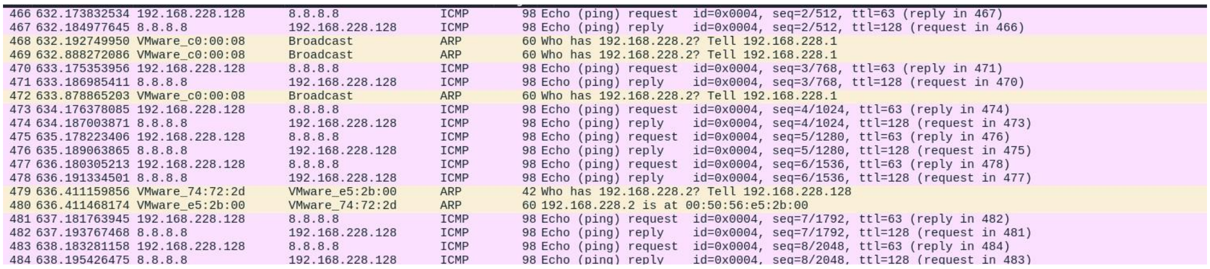
Task 1.1A: Step 1:



```
bash: pythong3: command not found
seed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.1A.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = b2:4f:cb:d7:51:9a
  src       = 42:0c:3f:83:9e:97
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 13288
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xeca3
     src       = 10.9.0.5
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x50cf
        id        = 0x3
        seq       = 0x1
###[ Raw ]###
           load      = '{\xf7\x9ah\x00\x00\x00\x00\xca\xf9\x07\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 42:0c:3f:83:9e:97
  src       = b2:4f:cb:d7:51:9a
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 44803
     flags     =
```



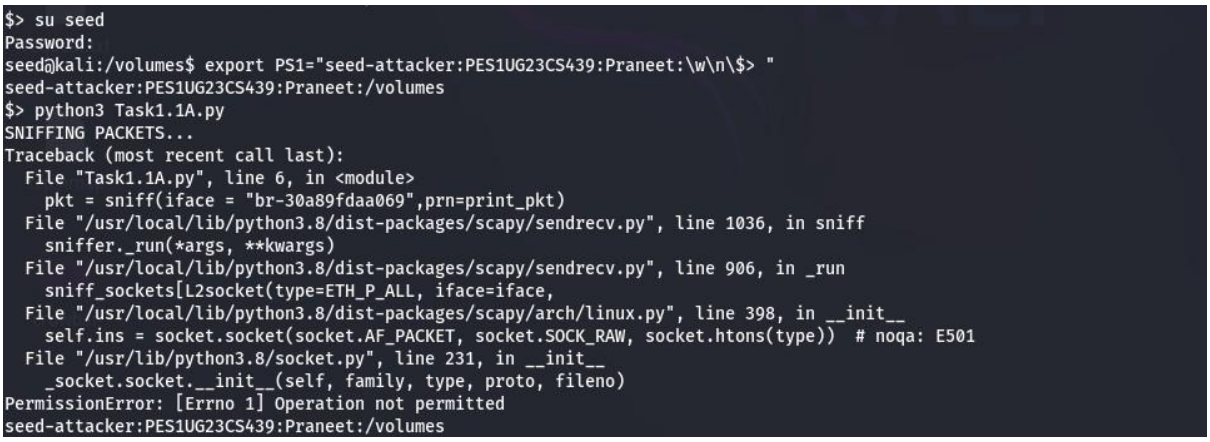
```
$> export PS1="seed-HostA:PES1UG23CS439:Praneet:\w\n\$> "
seed-HostA:PES1UG23CS439:Praneet:/
$> ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=11.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=11.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=11.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=127 time=10.7 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=127 time=10.9 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=127 time=11.2 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=127 time=12.1 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=127 time=12.3 ms
^C
--- 8.8.8.8 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7011ms
rtt min/avg/max/mdev = 10.748/11.450/12.257/0.505 ms
```



```
466 632.173832534 192.168.228.128   8.8.8.8           ICMP     98 Echo (ping) request  id=0x0004, seq=2/512, ttl=63 (reply in 467)
467 632.184977645 8.8.8.8           192.168.228.128   ICMP     98 Echo (ping) reply    id=0x0004, seq=2/512, ttl=128 (request in 466)
468 632.192749950 VMware_c0:00:08   Broadcast         ARP      60 Who has 192.168.228.2? Tell 192.168.228.1
469 632.888272086 VMware_c0:00:08   Broadcast         ARP      60 Who has 192.168.228.2? Tell 192.168.228.1
470 633.175353956 192.168.228.128   8.8.8.8           ICMP     98 Echo (ping) request  id=0x0004, seq=3/768, ttl=63 (reply in 471)
471 633.186985411 8.8.8.8           192.168.228.128   ICMP     98 Echo (ping) reply    id=0x0004, seq=3/768, ttl=128 (request in 470)
472 633.878865203 VMware_c0:00:08   Broadcast         ARP      60 Who has 192.168.228.2? Tell 192.168.228.1
473 634.176378085 192.168.228.128   8.8.8.8           ICMP     98 Echo (ping) request  id=0x0004, seq=4/1024, ttl=63 (reply in 474)
474 634.187003871 8.8.8.8           192.168.228.128   ICMP     98 Echo (ping) reply    id=0x0004, seq=4/1024, ttl=128 (request in 473)
475 635.178223406 192.168.228.128   8.8.8.8           ICMP     98 Echo (ping) request  id=0x0004, seq=5/1280, ttl=63 (reply in 476)
476 635.189063865 8.8.8.8           192.168.228.128   ICMP     98 Echo (ping) reply    id=0x0004, seq=5/1280, ttl=128 (request in 475)
477 636.180305213 192.168.228.128   8.8.8.8           ICMP     98 Echo (ping) request  id=0x0004, seq=6/1536, ttl=63 (reply in 478)
478 636.191334501 8.8.8.8           192.168.228.128   ICMP     98 Echo (ping) reply    id=0x0004, seq=6/1536, ttl=128 (request in 477)
479 636.411159856 VMware_74:72:2d   VMware_e5:2b:00   ARP      42 Who has 192.168.228.2? Tell 192.168.228.128
480 636.411468174 VMware_e5:2b:00   VMware_74:72:2d   ARP      60 192.168.228.2 is at 00:50:56:e5:2b:00
481 637.181763945 192.168.228.128   8.8.8.8           ICMP     98 Echo (ping) request  id=0x0004, seq=7/1792, ttl=63 (reply in 482)
482 637.193767468 8.8.8.8           192.168.228.128   ICMP     98 Echo (ping) reply    id=0x0004, seq=7/1792, ttl=128 (request in 481)
483 638.183281158 192.168.228.128   8.8.8.8           ICMP     98 Echo (ping) request  id=0x0004, seq=8/2048, ttl=63 (reply in 484)
484 638.195426475 8.8.8.8           192.168.228.128   ICMP     98 Echo (ping) reply    id=0x0004, seq=8/2048, ttl=128 (request in 483)
```

Here a python file is run in the attacker machine which has scapy's sniff function in it which is used to sniff for packets being transmitted over the network. In host A a ping to 8.8.8.8 executed, during this the reply and response of the host to 8.8.8.8 is being sniffed by the attacker. The wireshark snip shows the ICMP packets being requested and the reply to it. During the sniffing process the entire details of the packet is shown.

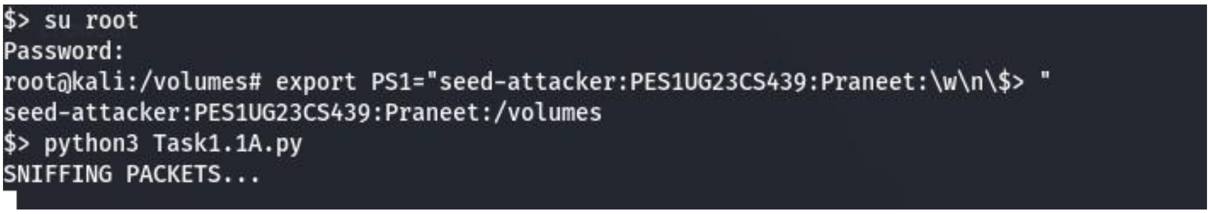
Step 2:



```
$> su seed
Password:
seed@kali:/volumes$ export PS1="seed-attacker:PES1UG23CS439:Praneet:\w\n\$> "
seed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.1A.py
SNIFFING PACKETS...
Traceback (most recent call last):
  File "Task1.1A.py", line 6, in <module>
    pkt = sniff(iface = "br-30a89fdaa069",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-attacker:PES1UG23CS439:Praneet:/volumes
```

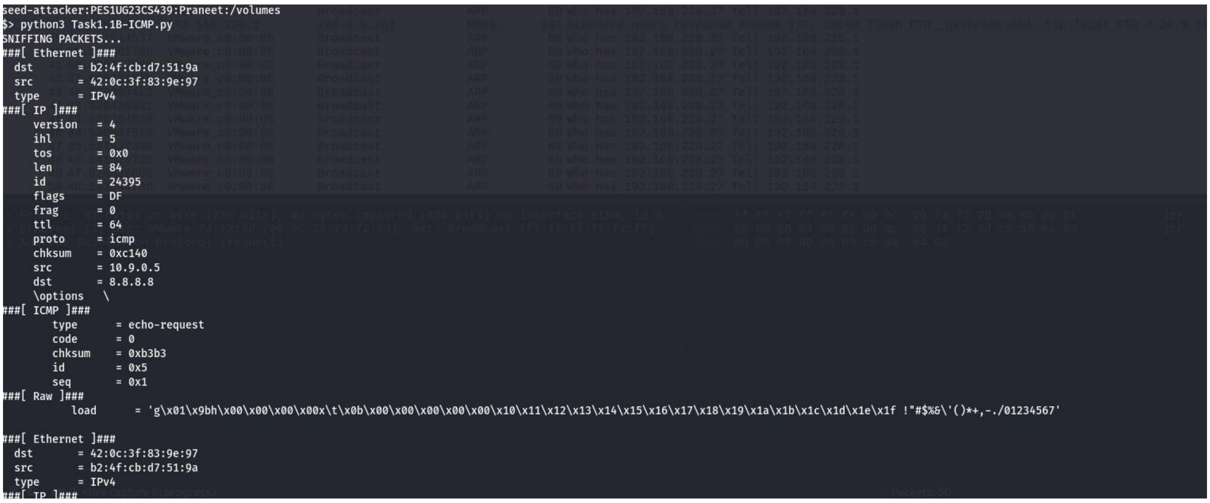
When the program is run as a non-privileged seed user it fails with a *permission denied* error because sending spoofed ICMP packets requires creating raw sockets, which is a privileged operation in Linux. Only the root user or processes can create raw sockets, as they allow crafting arbitrary packets and bypassing the normal network stack. Since seed lacks these privileges, the script cannot execute successfully.

Step 3:

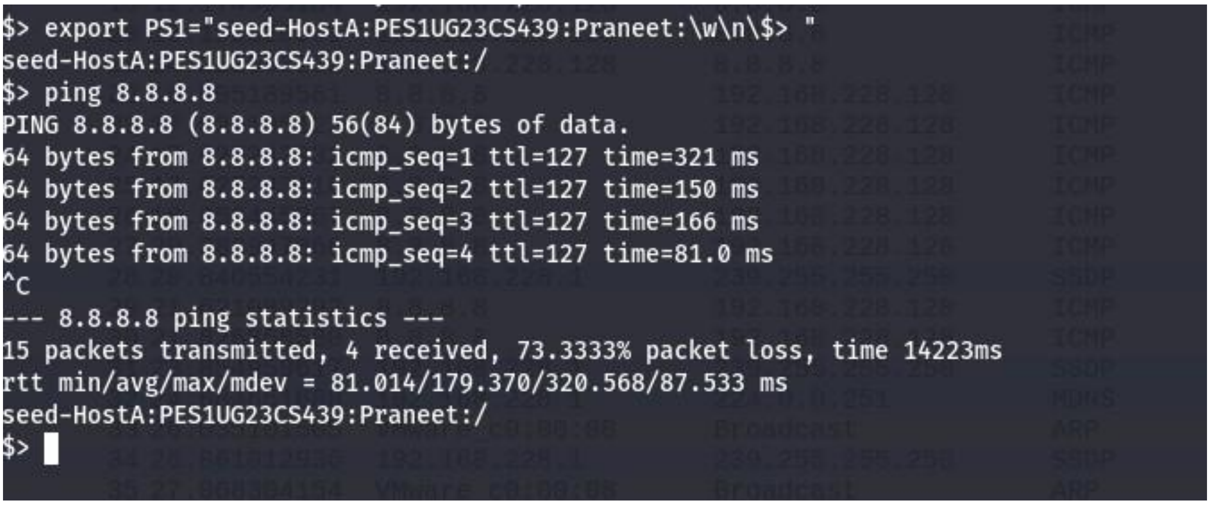


```
$> su root
Password:
root@kali:/volumes# export PS1="seed-attacker:PES1UG23CS439:Praneet:\w\n\$> "
seed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.1A.py
SNIFFING PACKETS...
```

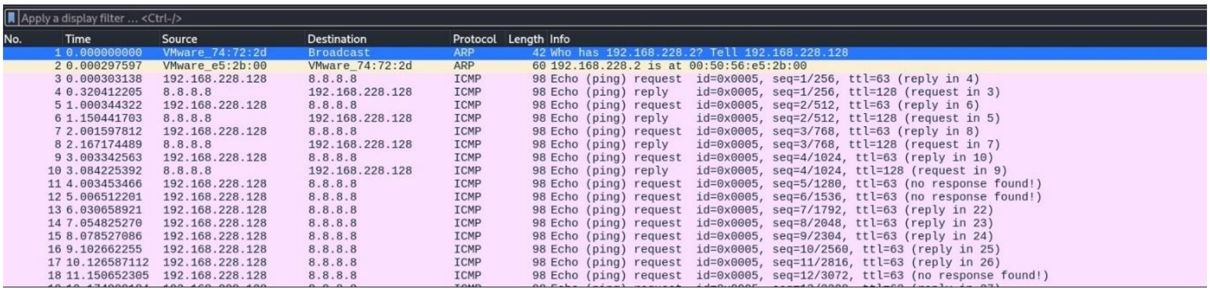
Task 1.1B :Step 1:



```
seed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.1B-ICMP.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = b2:4f:cb:d7:51:9a
  src       = 42:0c:3f:83:9e:97
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 24395
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xc140
     src       = 10.9.0.5
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xb3b3
        id        = 0x5
        seq       = 0x1
###[ Raw ]###
           load      = 'g\x01\x9bh\x00\x00\x00\x00x\t\x0b\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 42:0c:3f:83:9e:97
  src       = b2:4f:cb:d7:51:9a
  type      = IPv4
###[ IP ]###
```



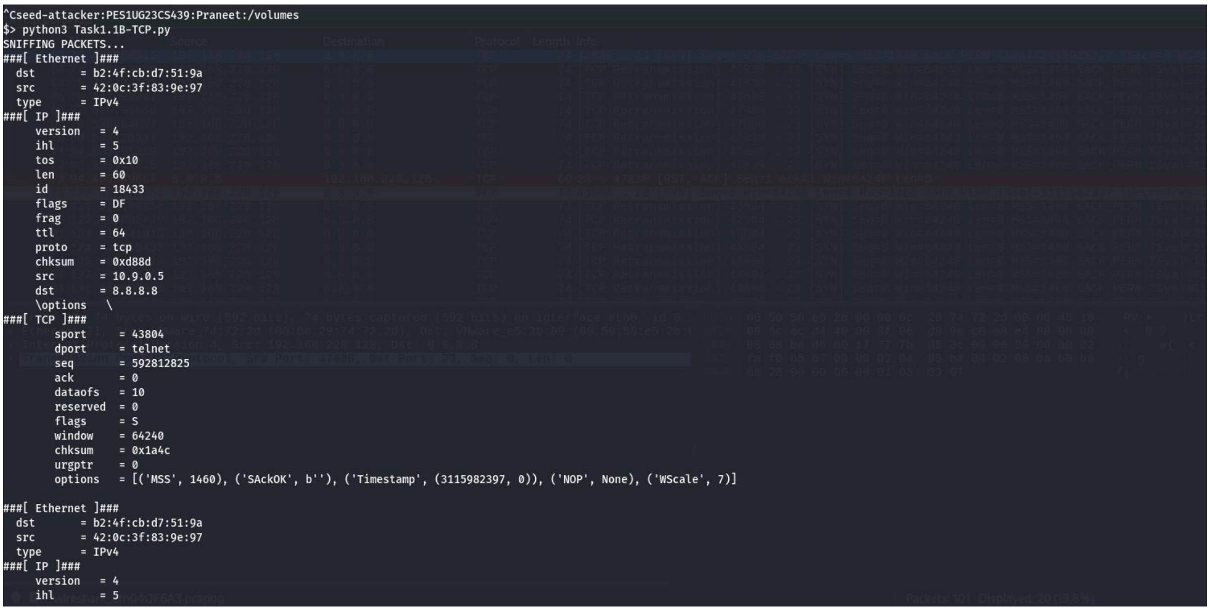
```
$> export PS1="seed-HostA:PES1UG23CS439:Praneet:\w\n\$> "
seed-HostA:PES1UG23CS439:Praneet:/
$> ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=321 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=150 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=166 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=127 time=81.0 ms
^C
--- 8.8.8.8 ping statistics ---
15 packets transmitted, 4 received, 73.3333% packet loss, time 14223ms
rtt min/avg/max/mdev = 81.014/179.370/320.568/87.533 ms
seed-HostA:PES1UG23CS439:Praneet:/
$> 
```



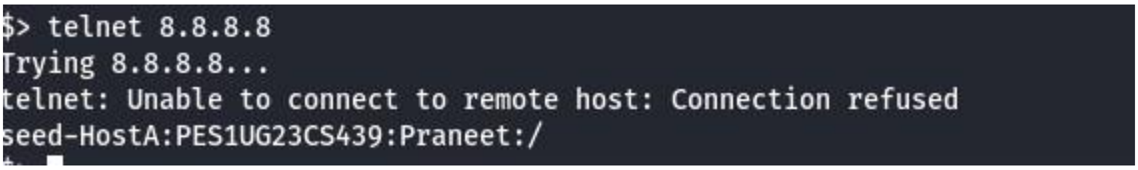
```
Apply a display filter ... <Ctrl-/>
No.  Time          Source             Destination        Protocol  Length Info
  1  0.000000000   VMware_74:72:2d    Broadcast          ARP       42 Who has 192.168.228.2? Tell 192.168.228.128
  2  0.000297597   VMware_e5:2b:00    VMware_74:72:2d    ARP       60 192.168.228.2 is at 00:50:56:e5:2b:00
  3  0.000303138   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=1/256, ttl=63 (reply in 4)
  4  0.320412205   8.8.8.8            192.168.228.128    ICMP      98 Echo (ping) reply    id=0x0005, seq=1/256, ttl=128 (request in 3)
  5  1.000344322   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=2/512, ttl=63 (reply in 6)
  6  1.150441703   8.8.8.8            192.168.228.128    ICMP      98 Echo (ping) reply    id=0x0005, seq=2/512, ttl=128 (request in 5)
  7  2.001597812   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=3/768, ttl=63 (reply in 8)
  8  2.167174489   8.8.8.8            192.168.228.128    ICMP      98 Echo (ping) reply    id=0x0005, seq=3/768, ttl=128 (request in 7)
  9  3.003342563   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=4/1024, ttl=63 (reply in 10)
 10  3.084225392   8.8.8.8            192.168.228.128    ICMP      98 Echo (ping) reply    id=0x0005, seq=4/1024, ttl=128 (request in 9)
 11  4.003453466   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=5/1280, ttl=63 (no response found!)
 12  5.006512201   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=6/1536, ttl=63 (no response found!)
 13  6.030658921   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=7/1792, ttl=63 (reply in 22)
 14  7.054825270   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=8/2048, ttl=63 (reply in 23)
 15  8.078527086   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=9/2304, ttl=63 (reply in 24)
 16  9.102662255   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=10/2560, ttl=63 (reply in 25)
 17 10.126587112   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=11/2816, ttl=63 (reply in 26)
 18 11.150652305   192.168.228.128    8.8.8.8            ICMP      98 Echo (ping) request  id=0x0005, seq=12/3072, ttl=63 (no response found!)
```

In this task the attacker machine’s sniffer script was configured with its active network interface and run while Wireshark was simultaneously capturing traffic on the same interface. The script’s filter was set to capture only ICMP packets, so when Host A sent ping requests to 8.8.8.8 both the script and Wireshark detected and displayed the ICMP echo request packets leaving Host A as well as the ICMP echo reply packets coming back from the destination. This confirmed that the sniffer was functioning correctly which filtered only ICMP traffic and that Wireshark’s capture matched the packets observed by the Scapy sniffer in real time.

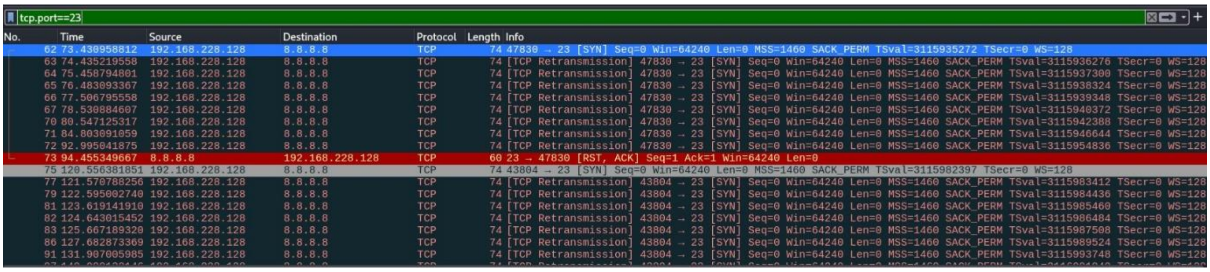
Step 2:



```
^Cseed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.1B-TCP.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = b2:4f:cb:d7:51:9a
  src       = 42:0c:3f:83:9e:97
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 18433
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xd88d
     src       = 10.9.0.5
     dst       = 8.8.8.8
     \options   \
###[ TCP ]###
        sport     = 43804
        dport     = telnet
        seq       = 592812825
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0x1a4c
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (3115982397, 0)), ('NOP', None), ('WScale', 7)]

###[ Ethernet ]###
  dst       = b2:4f:cb:d7:51:9a
  src       = 42:0c:3f:83:9e:97
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
```



```
$> telnet 8.8.8.8
Trying 8.8.8.8...
telnet: Unable to connect to remote host: Connection refused
seed-HostA:PES1UG23CS439:Praneet:/
```

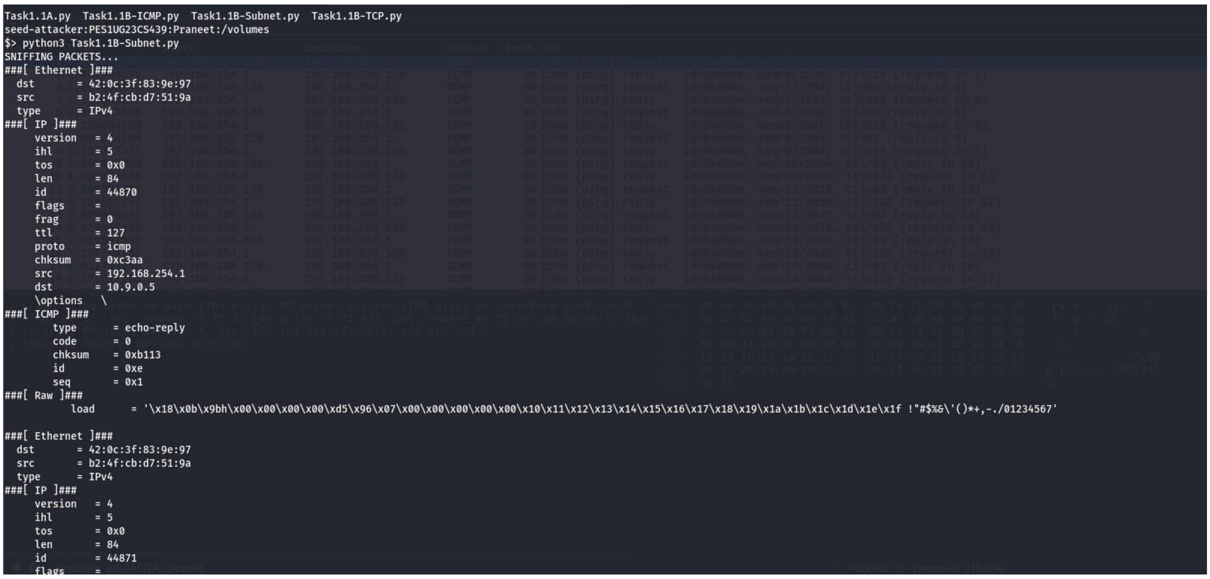


tcp.port==23

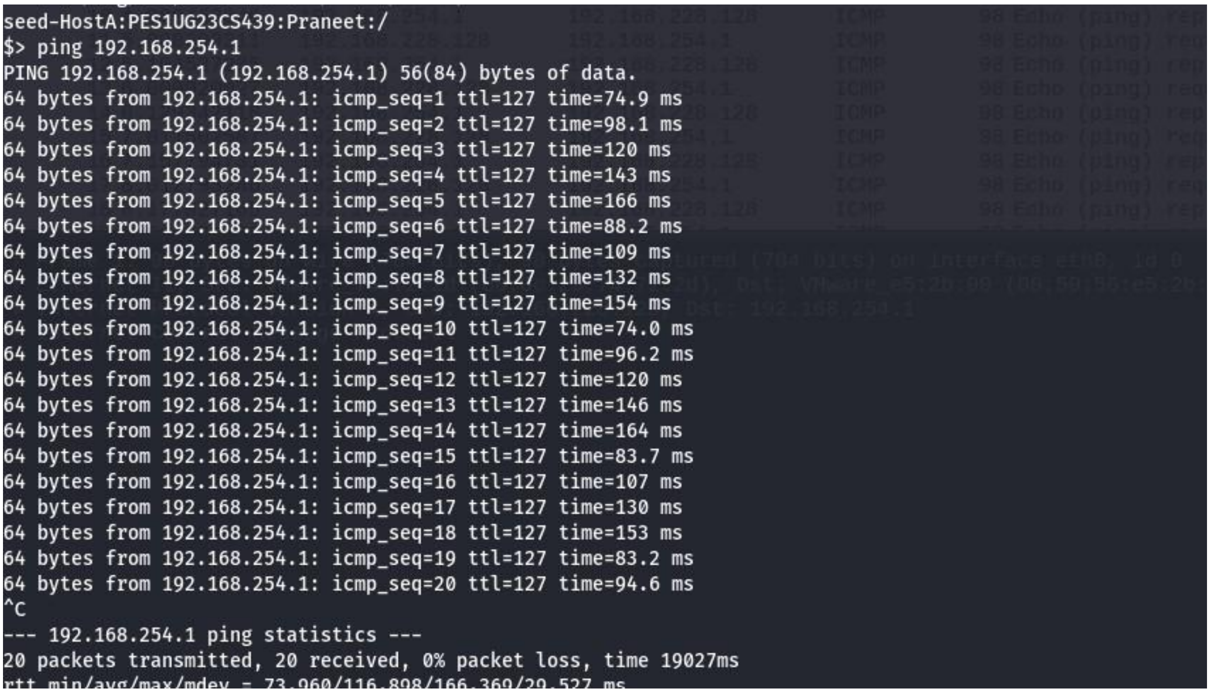
```
No.  Time          Source           Destination      Protocol Length Info
  62 73.430958812  192.168.228.128  8.8.8.8          TCP      74 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115935272 TSecr=0 WS=128
  63 74.435219558  192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115936276 TSecr=0 WS=128
  64 75.458794801  192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115937300 TSecr=0 WS=128
  65 76.483093367  192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115938324 TSecr=0 WS=128
  66 77.506795558  192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115939348 TSecr=0 WS=128
  67 78.530884607  192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115940372 TSecr=0 WS=128
  70 80.547125317  192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115942388 TSecr=0 WS=128
  71 84.803091059  192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115946644 TSecr=0 WS=128
  72 92.995041875  192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 47830 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115954836 TSecr=0 WS=128
  73 94.455349667  8.8.8.8          192.168.228.128  TCP      60 23 → 47830 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
  75 120.556381851 192.168.228.128  8.8.8.8          TCP      74 43804 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115982397 TSecr=0 WS=128
  77 121.570788256 192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 43804 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115983412 TSecr=0 WS=128
  79 122.595002740 192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 43804 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115984436 TSecr=0 WS=128
  81 123.619141910 192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 43804 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115985460 TSecr=0 WS=128
  82 124.643015452 192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 43804 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115986484 TSecr=0 WS=128
  83 125.667189320 192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 43804 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115987508 TSecr=0 WS=128
  86 127.682873369 192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 43804 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115989524 TSecr=0 WS=128
  91 131.907005985 192.168.228.128  8.8.8.8          TCP      74 [TCP Retransmission] 43804 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3115993748 TSecr=0 WS=128
```

In this task the script on the attacker machine was configured with the correct network interface and set to filter only TCP traffic originating from the specific IP address on port 23 (Telnet). With Wireshark capturing on the same interface, the script was run and host A initiated a Telnet connection to 8.8.8.8. Both the Scapy sniffer and Wireshark captured the TCP packets for this session including the three-way handshake (SYN, SYN-ACK, ACK) and subsequent Telnet data packets. This confirmed that the filter was working also isolating only TCP port 23 traffic and that the packets observed by the Python sniffer were consistent with those displayed in Wireshark.

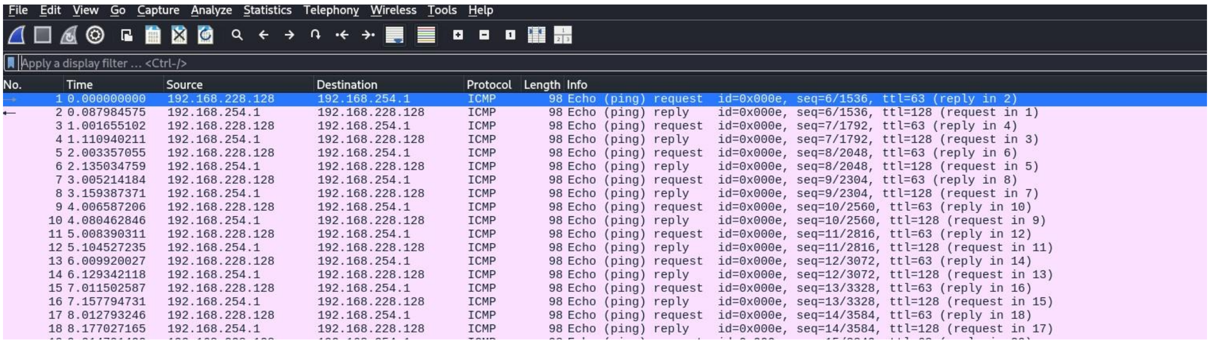
Step 3:



```
Task1.1A.py  Task1.1B-ICMP.py  Task1.1B-Subnet.py  Task1.1B-TCP.py
seed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.1B-Subnet.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 42:0c:3f:83:9e:97
  src       = b2:4f:cb:d7:51:9a
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 44870
     flags     =
     frag      = 0
     ttl       = 127
     proto     = icmp
     chksum    = 0xc3aa
     src       = 192.168.254.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0xb113
        id        = 0xe
        seq       = 0x1
###[ Raw ]###
           load      = '\x18\x0b\x9bh\x00\x00\x00\x00\xd5\x96\x07\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 42:0c:3f:83:9e:97
  src       = b2:4f:cb:d7:51:9a
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 44871
     flags     =
```



```
seed-HostA:PES1UG23CS439:Praneet:/
$> 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=127 time=74.9 ms
64 bytes from 192.168.254.1: icmp_seq=2 ttl=127 time=98.1 ms
64 bytes from 192.168.254.1: icmp_seq=3 ttl=127 time=120 ms
64 bytes from 192.168.254.1: icmp_seq=4 ttl=127 time=143 ms
64 bytes from 192.168.254.1: icmp_seq=5 ttl=127 time=166 ms
64 bytes from 192.168.254.1: icmp_seq=6 ttl=127 time=88.2 ms
64 bytes from 192.168.254.1: icmp_seq=7 ttl=127 time=109 ms
64 bytes from 192.168.254.1: icmp_seq=8 ttl=127 time=132 ms
64 bytes from 192.168.254.1: icmp_seq=9 ttl=127 time=154 ms
64 bytes from 192.168.254.1: icmp_seq=10 ttl=127 time=74.0 ms
64 bytes from 192.168.254.1: icmp_seq=11 ttl=127 time=96.2 ms
64 bytes from 192.168.254.1: icmp_seq=12 ttl=127 time=120 ms
64 bytes from 192.168.254.1: icmp_seq=13 ttl=127 time=146 ms
64 bytes from 192.168.254.1: icmp_seq=14 ttl=127 time=164 ms
64 bytes from 192.168.254.1: icmp_seq=15 ttl=127 time=83.7 ms
64 bytes from 192.168.254.1: icmp_seq=16 ttl=127 time=107 ms
64 bytes from 192.168.254.1: icmp_seq=17 ttl=127 time=130 ms
64 bytes from 192.168.254.1: icmp_seq=18 ttl=127 time=153 ms
64 bytes from 192.168.254.1: icmp_seq=19 ttl=127 time=83.2 ms
64 bytes from 192.168.254.1: icmp_seq=20 ttl=127 time=94.6 ms
^C
--- 192.168.254.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19027ms
rtt min/avg/max/mdev = 73.960/116.898/166.369/29.527 ms
```



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

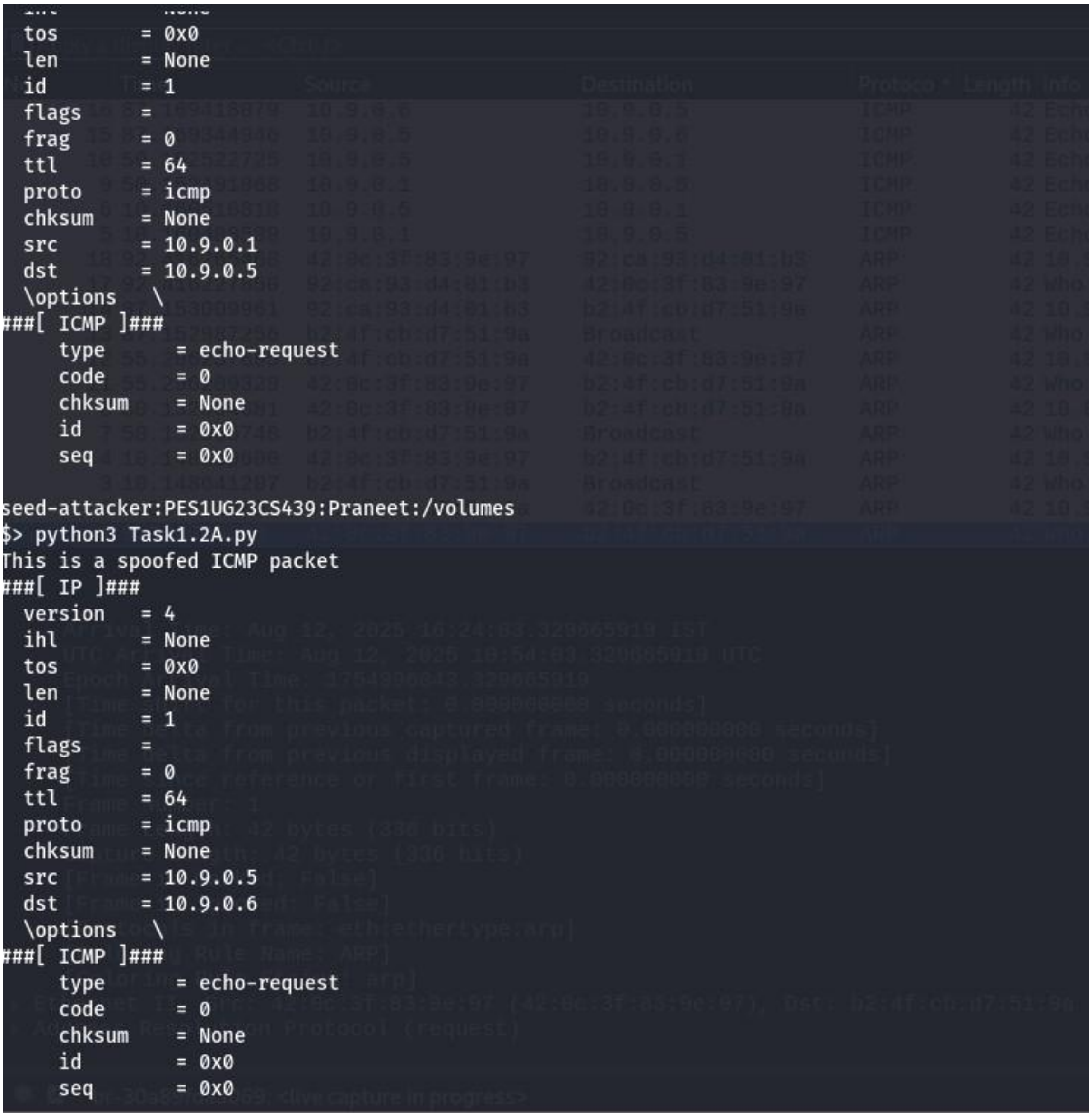
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=6/1536, ttl=63 (reply in 2)
2	0.087984575	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=6/1536, ttl=128 (request in 1)
3	1.001655102	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=7/1792, ttl=63 (reply in 4)
4	1.110940211	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=7/1792, ttl=128 (request in 3)
5	2.003357055	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=8/2048, ttl=63 (reply in 6)
6	2.135034759	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=8/2048, ttl=128 (request in 5)
7	3.005214184	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=9/2304, ttl=63 (reply in 8)
8	3.159387371	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=9/2304, ttl=128 (request in 7)
9	4.006587206	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=10/2560, ttl=63 (reply in 10)
10	4.080462846	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=10/2560, ttl=128 (request in 9)
11	5.008390311	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=11/2816, ttl=63 (reply in 12)
12	5.104527235	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=11/2816, ttl=128 (request in 11)
13	6.009920027	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=12/3072, ttl=63 (reply in 14)
14	6.129342118	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=12/3072, ttl=128 (request in 13)
15	7.011502587	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=13/3328, ttl=63 (reply in 16)
16	7.157794731	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=13/3328, ttl=128 (request in 15)
17	8.012793246	192.168.228.128	192.168.254.1	ICMP	98	Echo (ping) request id=0x000e, seq=14/3584, ttl=63 (reply in 18)
18	8.177027165	192.168.254.1	192.168.228.128	ICMP	98	Echo (ping) reply id=0x000e, seq=14/3584, ttl=128 (request in 17)

In this task the script on the attacker machine was configured with the correct network interface and a filter to capture traffic belonging to a chosen subnet (192.168.254.0/24), which was intentionally different from the subnet used by the VM. With Wireshark on the same interface the script was executed and Host A sent ICMP ping requests to

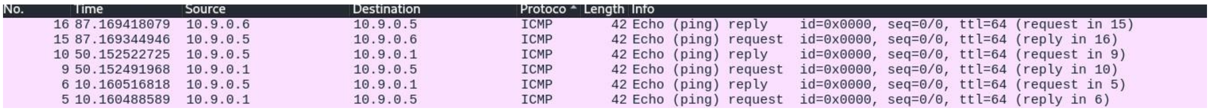
192.168.254.1. The sniffer captured these packets because they matched the specified subnet filter, even though the destination was outside the attacker's active network. Wireshark simultaneously displayed the same ICMP echo request packets (and any replies) confirming that the Python sniffer correctly filtered and displayed only traffic from the targeted subnet.

Task 1.2:

Step 1:



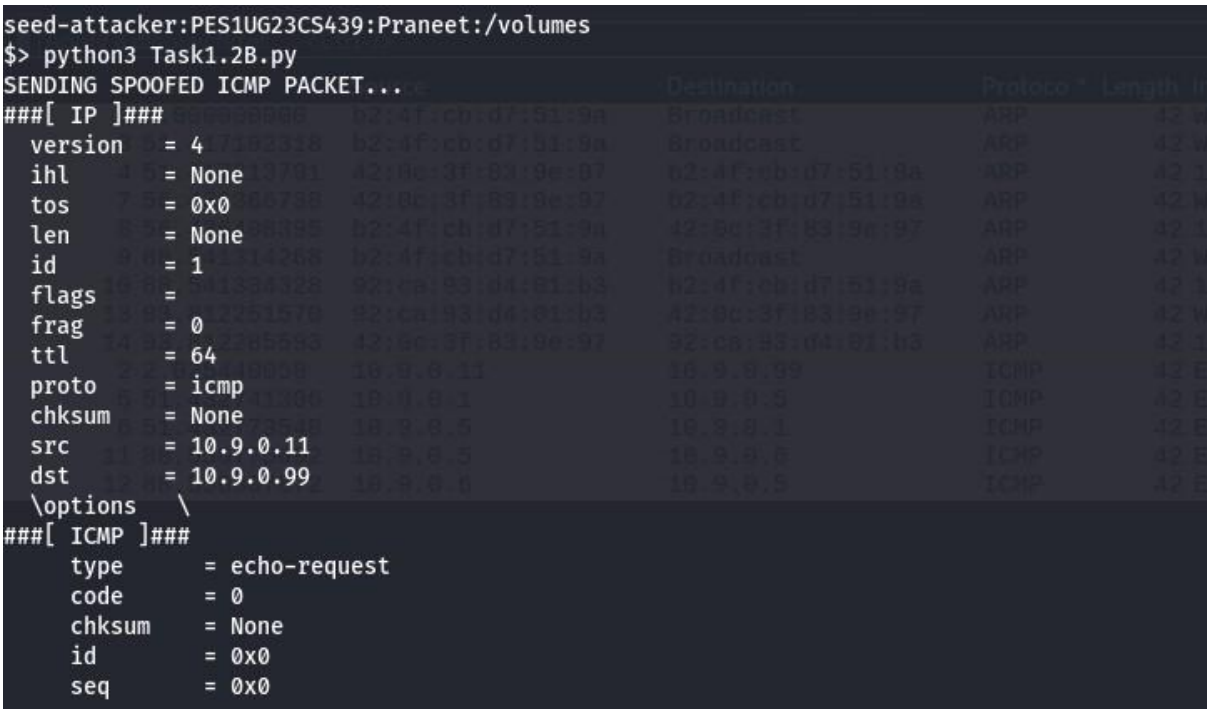
```
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = None
  src       = 10.9.0.1
  dst       = 10.9.0.5
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
     id        = 0x0
     seq       = 0x0

seed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.2A.py
This is a spoofed ICMP packet
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = None
  src       = 10.9.0.5
  dst       = 10.9.0.6
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
     id        = 0x0
     seq       = 0x0
```



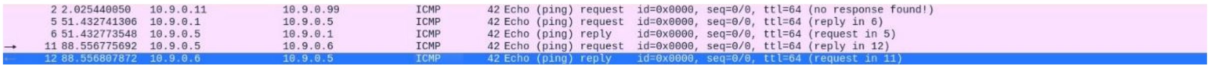
```
No.    Time          Source       Destination   Protoco ^ Length Info
    16 87.169418079  10.9.0.6     10.9.0.5      ICMP      42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 15)
    15 87.169344946  10.9.0.5     10.9.0.6      ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (reply in 16)
    10 50.152522725  10.9.0.5     10.9.0.1      ICMP      42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 9)
     9 50.152491968  10.9.0.1     10.9.0.5      ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (reply in 10)
     6 10.160516818  10.9.0.5     10.9.0.1      ICMP      42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 5)
     5 10.160488589  10.9.0.1     10.9.0.5      ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (reply in 6)
```

In this task the script was run on the attacker machine with Wireshark capturing traffic on the same interface specified in the code. The script used Scapy to craft and send a spoofed ICMP Echo Request packet setting the source IP to a machine within the local network and the destination IP to an active host on the internet. Since the source address was spoofed any kind of ICMP Echo Reply from the destination was sent to the spoofed machine not the attacker. In Wireshark, the crafted ICMP Echo Request appeared exactly as sent by the script showing the falsified source IP and the intended destination. This demonstrated that the attacker could inject forged packets into the network while the replies would never return to them due to the spoofing.

Step 2:



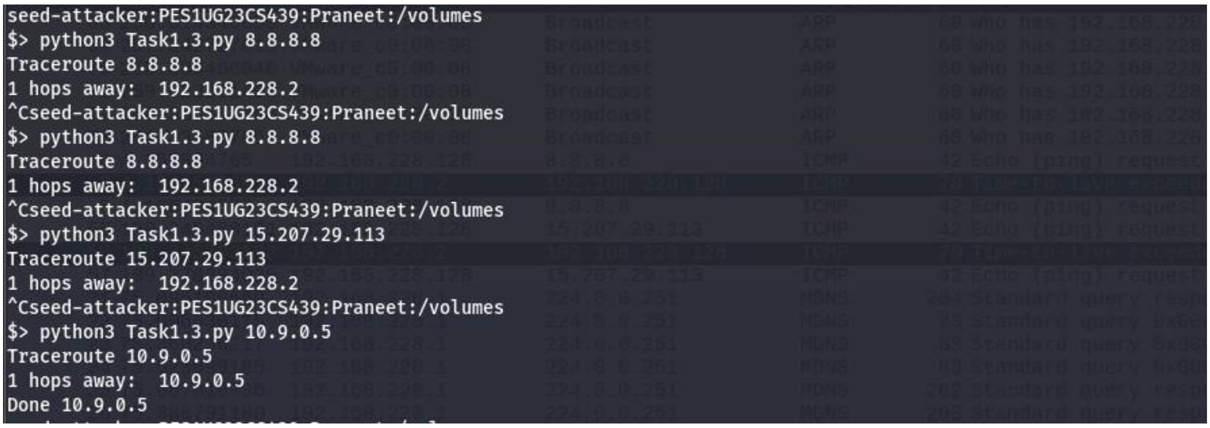
```
seed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.2B.py
SENDING SPOOFED ICMP PACKET...
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = None
  src       = 10.9.0.11
  dst       = 10.9.0.99
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
     id        = 0x0
     seq       = 0x0
```



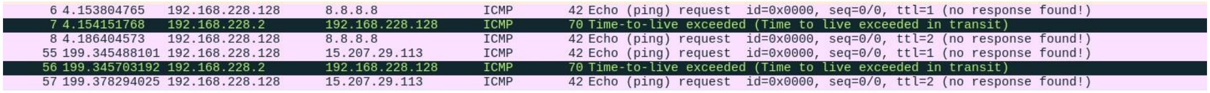
```
    2 2.025440050   10.9.0.11    10.9.0.99    ICMP   42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (no response found!)
    5 51.432741306  10.9.0.1     10.9.0.5     ICMP   42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (reply in 6)
    6 51.432773548  10.9.0.5     10.9.0.1     ICMP   42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 5)
→  11 88.556775692  10.9.0.5     10.9.0.6     ICMP   42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (reply in 12)
←  12 88.556807872  10.9.0.6     10.9.0.5     ICMP   42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 11)
```

In this task the script was executed on the attacker machine while Wireshark captured traffic on the same interface defined in the program. The script generated and sent a spoofed ICMP Echo Request packet with an arbitrary non-existent source IP address and a chosen destination IP. In Wireshark the packet appeared with the fake source address and the real destination confirming that the source field had been successfully falsified. Since the source IP did not correspond to an actual reachable machine no ICMP Echo Reply was observed illustrating how spoofing can disguise the true origin of network traffic.

Task 1.3:



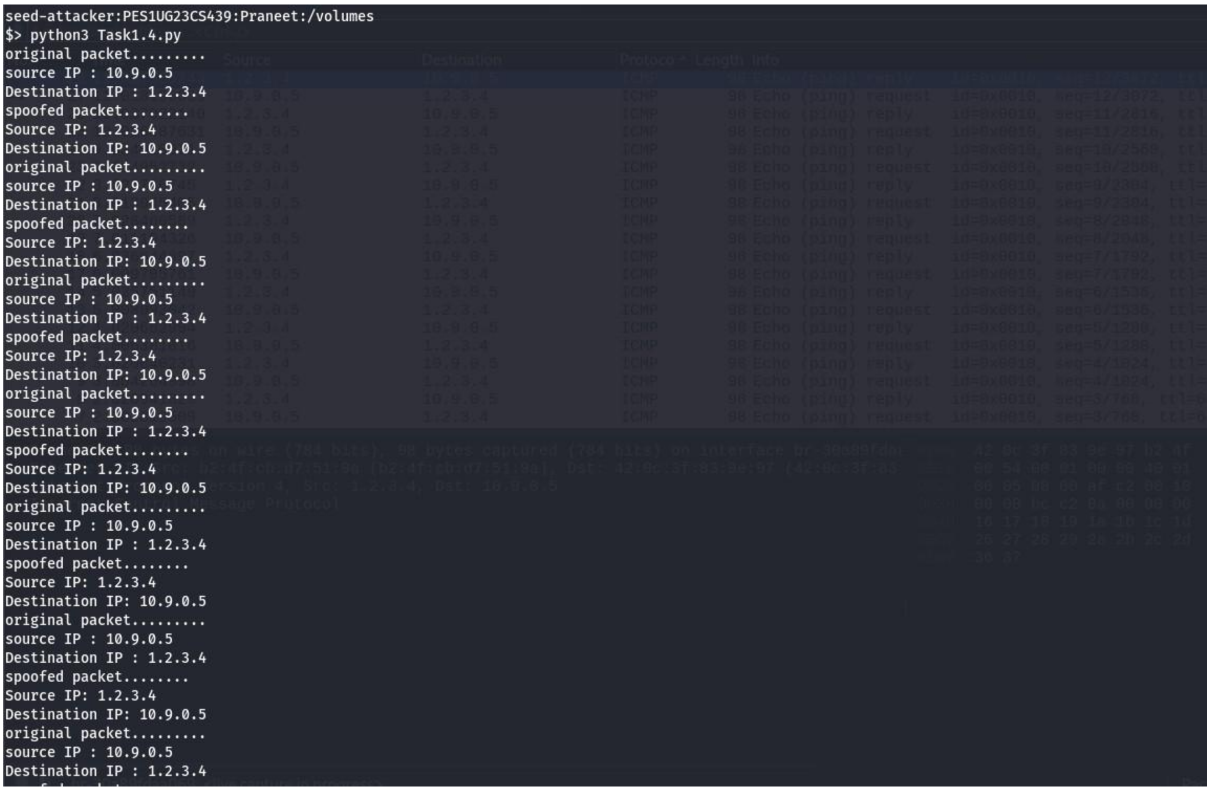
```
seed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.3.py 8.8.8.8
Traceroute 8.8.8.8
1 hops away:  192.168.228.2
^Cseed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.3.py 8.8.8.8
Traceroute 8.8.8.8
1 hops away:  192.168.228.2
^Cseed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.3.py 15.207.29.113
Traceroute 15.207.29.113
1 hops away:  192.168.228.2
^Cseed-attacker:PES1UG23CS439:Praneet:/volumes
$> python3 Task1.3.py 10.9.0.5
Traceroute 10.9.0.5
1 hops away:  10.9.0.5
Done 10.9.0.5
```



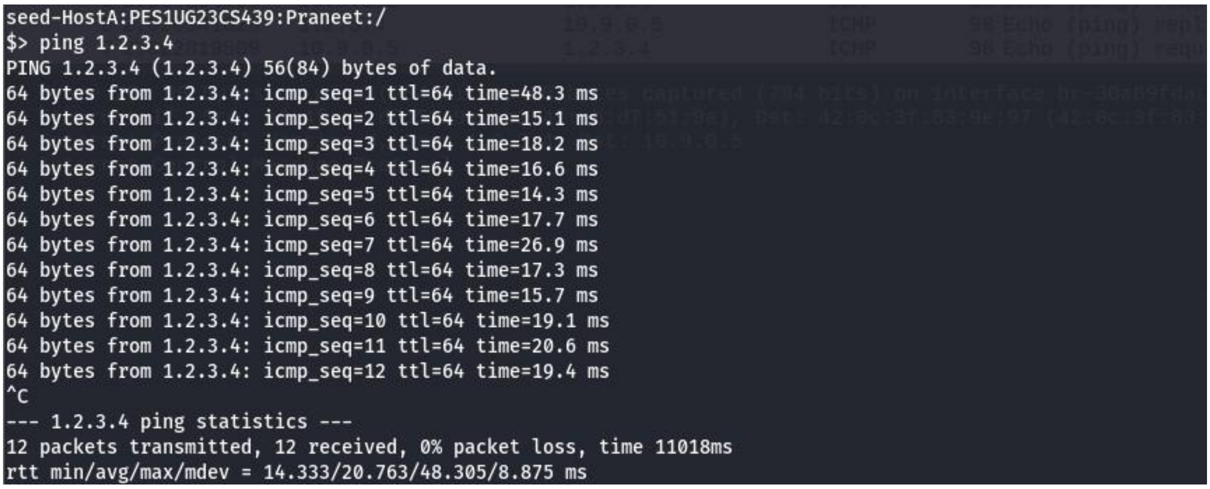
```
    6 4.153804765   192.168.228.128    8.8.8.8           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=1 (no response found!)
    7 4.154151768   192.168.228.2      192.168.228.128   ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)
    8 4.186404573   192.168.228.128    8.8.8.8           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=2 (no response found!)
   55 199.345488101 192.168.228.128    15.207.29.113     ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=1 (no response found!)
   56 199.345703192 192.168.228.2      192.168.228.128   ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)
   57 199.378294025 192.168.228.128    15.207.29.113     ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=2 (no response found!)
```

In this task the script was run on the attacker machine with Wireshark capturing traffic on the same interface specified in the code. The script implemented a basic traceroute using Scapy by sending ICMP Echo Request packets to the target IP address with an incrementally increasing TTL (Time-To-Live) value. Each time a router along the path decremented the TTL to zero it returned an ICMP Time Exceeded message which allows the script to identify that hop. Wireshark displayed the sequence of ICMP Echo Requests leaving the attacker and the corresponding ICMP Time Exceeded responses from intermediate routers, followed by an ICMP Echo Reply from the destination when reached. This confirmed that the program could estimate the hop count and map the route between the attacker and the required destination.

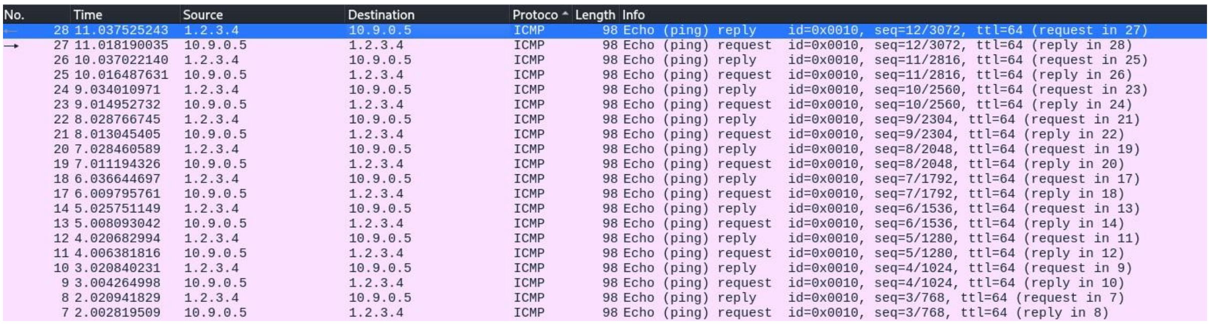
Task 1.4:



```
seed-attacker:PES1UG23CS439:Praneet:/volumes
$> 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.........
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.........
source IP : 10.9.0.5
Destination IP : 1.2.3.4
```



```
seed-HostA:PES1UG23CS439:Praneet:/
$> 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=48.3 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=15.1 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=18.2 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=16.6 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=14.3 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=17.7 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=26.9 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=17.3 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=15.7 ms
64 bytes from 1.2.3.4: icmp_seq=10 ttl=64 time=19.1 ms
64 bytes from 1.2.3.4: icmp_seq=11 ttl=64 time=20.6 ms
64 bytes from 1.2.3.4: icmp_seq=12 ttl=64 time=19.4 ms
^C
--- 1.2.3.4 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11018ms
rtt min/avg/max/mdev = 14.333/20.763/48.305/8.875 ms
```



No.	Time	Source	Destination	Protoco	Length	Info
28	11.037525243	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=12/3072, ttl=64 (request in 27)
27	11.018190035	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=12/3072, ttl=64 (reply in 28)
26	10.037022140	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=11/2816, ttl=64 (request in 25)
25	10.016487631	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=11/2816, ttl=64 (reply in 26)
24	9.034010971	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=10/2560, ttl=64 (request in 23)
23	9.014952732	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=10/2560, ttl=64 (reply in 24)
22	8.028766745	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=9/2304, ttl=64 (request in 21)
21	8.013045405	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=9/2304, ttl=64 (reply in 22)
20	7.028460589	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=8/2048, ttl=64 (request in 19)
19	7.011194326	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=8/2048, ttl=64 (reply in 20)
18	6.036644697	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=7/1792, ttl=64 (request in 17)
17	6.009795761	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=7/1792, ttl=64 (reply in 18)
14	5.025751149	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=6/1536, ttl=64 (request in 13)
13	5.008093042	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=6/1536, ttl=64 (reply in 14)
12	4.020682994	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=5/1280, ttl=64 (request in 11)
11	4.006381816	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=5/1280, ttl=64 (reply in 12)
10	3.020840231	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=4/1024, ttl=64 (request in 9)
9	3.004264998	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=4/1024, ttl=64 (reply in 10)
8	2.020941829	1.2.3.4	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0010, seq=3/768, ttl=64 (request in 7)
7	2.002819509	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x0010, seq=3/768, ttl=64 (reply in 8)

In this task the script was executed on the attacker machine with Wireshark capturing traffic on the same interface configured in the program. The script passively sniffed ICMP Echo Requests from Host A's machine destined for the non-existent IP address 1.2.3.4. When it detected this packet, it crafted a spoofed ICMP Echo Reply with the source IP set to 1.2.3.4 and the destination IP set to Host A’s machine where it copied the identifier, sequence number and payload from the original request. Wireshark displayed the victim’s Echo Request to 1.2.3.4 followed immediately by the attacker’s forged Echo Reply even though no real host existed at that address. On the victim’s terminal the ping appeared successful which mis leaded it into believing that 1.2.3.4 was alive. This demonstrated how packet spoofing can falsify network reachability results.