Packet Sniffing and Spoofing Lab

Task 1.1A

The program will sniff for ICMP (Internet Control Message Protocol) packets and print information about the packets to the terminal.

```
#!/usr/bin/python3
from scapy.all import *

def print_pkt(pkt):
        pkt.show()

#sniff for ICMP packets and print the packet info on the terminal
pkt = sniff(filter = 'icmp', prn = print_pkt)
```

Running the program with the root privilege:

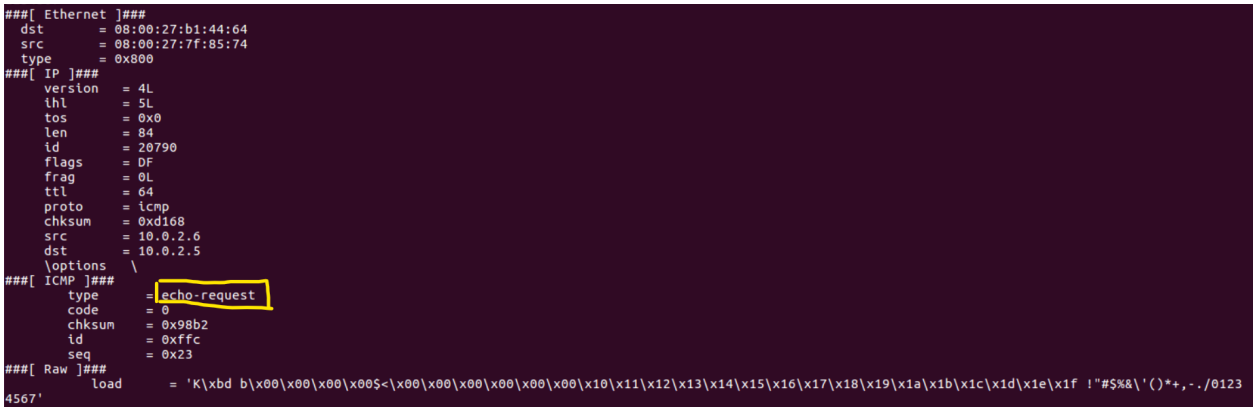
First, I make it executable:

```
ubuntu@attacker:~/sniffing_and_spoofing$ ls -l
total 4
-rw-rw-r-- 1 ubuntu ubuntu 189 Mar  3 07:45 sniffer.py
ubuntu@attacker:~/sniffing_and_spoofing$ chmod a+x sniffer.py
ubuntu@attacker:~/sniffing_and_spoofing$
ubuntu@attacker:~/sniffing_and_spoofing$ ls -l
total 4
-rwxrwxr-x 1 ubuntu ubuntu 189 Mar  3 07:45 sniffer.py
```

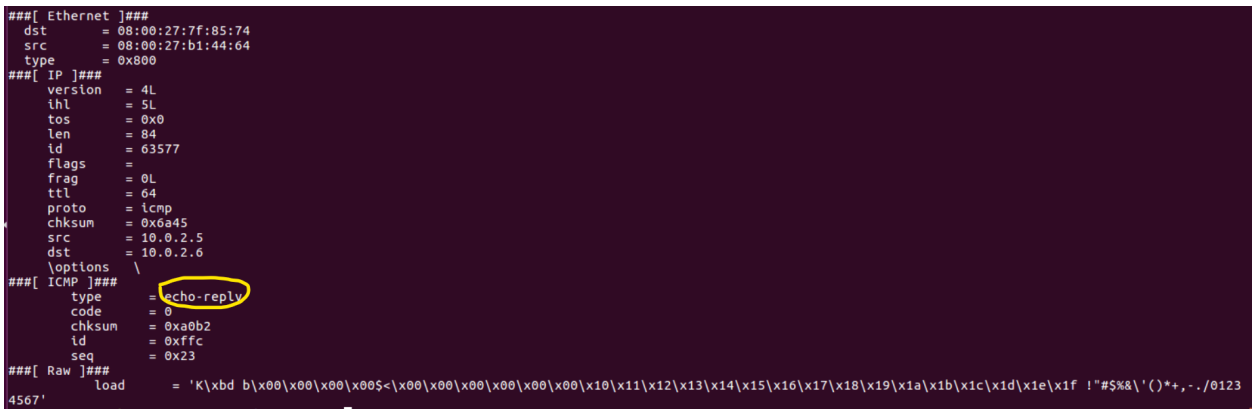
Then, I run it with root privileges.
When I first run it, nothing happens. This is because there are no ICMP packets being sent on my network. Opening up another terminal and pinging a website will cause ICMP packets to be sent.
I ping the server 10.0.2.5 and the sniffer begins to capture the ICMP packets:

```
ubuntu@attacker:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.785 ms
64 bytes from 10.0.2.5: icmp_seq=2 ttl=64 time=0.446 ms
64 bytes from 10.0.2.5: icmp_seq=3 ttl=64 time=0.636 ms
64 bytes from 10.0.2.5: icmp_seq=4 ttl=64 time=0.464 ms
64 bytes from 10.0.2.5: icmp_seq=5 ttl=64 time=0.597 ms
64 bytes from 10.0.2.5: icmp_seq=6 ttl=64 time=0.430 ms
```

The echo request packets are what is being sent by ping, and then 10.0.2.5 (server) is sending back an echo-reply:



```
###[ Ethernet ]###
  dst       = 08:00:27:b1:44:64
  src       = 08:00:27:7f:85:74
  type      = 0x800
###[ IP ]###
     version   = 4L
     ihl       = 5L
     tos       = 0x0
     len       = 84
     id        = 20790
     flags     = DF
     frag      = 0L
     ttl       = 64
     proto     = icmp
     chksum    = 0xd168
     src       = 10.0.2.6
     dst       = 10.0.2.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x98b2
        id        = 0xffc
        seq       = 0x23
###[ Raw ]###
           load      = 'K\xbd b\x00\x00\x00\x00$<\x00\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./0123
4567'
```



```
###[ Ethernet ]###
  dst       = 08:00:27:7f:85:74
  src       = 08:00:27:b1:44:64
  type      = 0x800
###[ IP ]###
     version   = 4L
     ihl       = 5L
     tos       = 0x0
     len       = 84
     id        = 63577
     flags     =
     frag      = 0L
     ttl       = 64
     proto     = icmp
     chksum    = 0x6a45
     src       = 10.0.2.5
     dst       = 10.0.2.6
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0xa0b2
        id        = 0xffc
        seq       = 0x23
###[ Raw ]###
           load      = 'K\xbd b\x00\x00\x00\x00$<\x00\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./0123
4567'
```

Running the program without the root privilege:

```
ubuntu@attacker:~/sniffing_and_spoofing$ python sniffer.py
WARNING: No route found for IPv6 destination :: (no default route?)
Traceback (most recent call last):
  File "sniffer.py", line 8, in <module>
    pkt = sniff(filter = 'icmp', prn = print_pkt)
  File "/usr/lib/python2.7/dist-packages/scapy/sendrecv.py", line 561, in sniff
    s = L2socket(type=ETH_P_ALL, *arg, **karg)
  File "/usr/lib/python2.7/dist-packages/scapy/arch/linux.py", line 451, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))
  File "/usr/lib/python2.7/socket.py", line 191, in __init__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] Operation not permitted
```

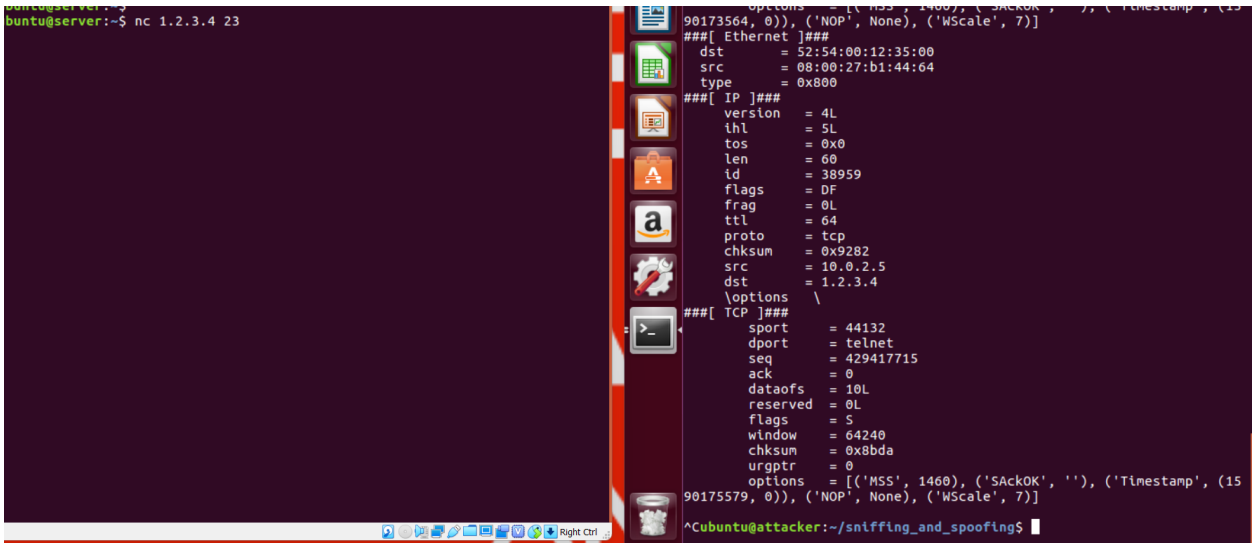
As soon as I run sniffer.py I get a permission error.

Task 1.1B

- Capture only the ICMP packet : accomplished in Task 1.1A
- Capture any TCP packet that comes from a particular IP and with a destination port number 23.
  I edit the sniffer.py program to use 'tcp and src host 10.0.2.5 and dst port 23' to filter for only tcp packets coming from host 10.0.2.5 and heading to any IP's port 23:

```
#!/usr/bin/python3
from scapy.all import *

def print_pkt(pkt):
        pkt.show()

#sniff for ICMP packets and print the packet info on the terminal
pkt = sniff(filter = 'tcp and src host 10.0.2.5 and dst port 23', prn = print_pkt)
```

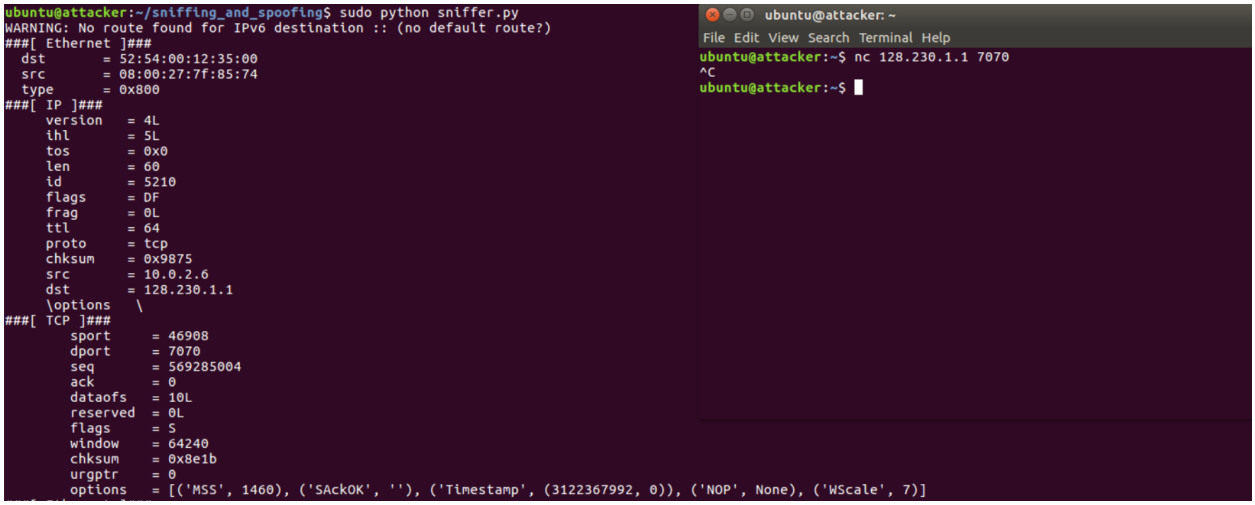
I can use the nc (netcat) command to open a TCP connection from the 10.0.2.5 (server) virtual machine :



- Capture packets comes from or to go 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.

```
#!/usr/bin/python3
from scapy.all import *

def print_pkt(pkt):
        pkt.show()

#sniff for ICMP packets and print the packet info on the terminal
pkt = sniff(filter = 'net 192.168.0.0/16', prn = print_pkt)
```

Sending a TCP packet with the nc command to an IP that is a part of the subnet:



Task 1.2: Spoofing ICMP Packets

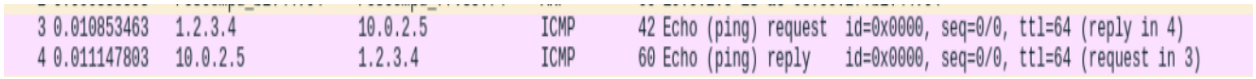
Below shown is a python program that uses Scapy to create a spoofed ICMP echo request packet with an arbitrary source IP address and send it to another virtual machine on my network.

The arbitrary source IP address is 1.2.3.4 and the destination address is10.0.2.5 (IP address of Server VM). Here is the program:

```
#!/usr/bin/python3
from scapy.all import *

ip = IP()
ip.dst = '10.0.2.5'
ip.src = '1.2.3.4'

icmp = ICMP()

packet = ip/icmp

send(packet)
```

Running the spoofing.py program with root privilege. It sent out one packet.

```
ubuntu@attacker:~/sniffing_and_spoofing$ sudo python spoofing.py
WARNING: No route found for IPv6 destination :: (no default route?)
.
Sent 1 packets.
```



The first ICMP packet was the echo request that the spoofing.py program sent. The second ICMP packet captured was the reply sent back from 10.0.2.5 (the Server VM).

The source for the request is 1.2.3.4 and the destination for the reply is also 1.2.3.4. This means that the spoofing.py program successfully spoofed an ICMP packet and assigned it an arbitrary source IP address.

Task 1.3: Traceroute

GIven below is the program for trace.py:

The maximum number of hops is 255. This means that if the packet fails to reach its destination by the time its TTL has been incremented all the way to 255, the program will stop.

```
#!/usr/bin/python3
from scapy.all import *
import sys

argumentList = sys.argv

if len(argumentList) != 2:
        print("Usage: sudo python trace.py [hostname]")
        exit()

MAX_TTL = 255
dstHostname = argumentList[1]
dstIP = socket.gethostbyname(argumentList[1])

print("trace.py to " + dstHostname +" (" + dstIP + ") 255 hops max")

ip = IP()
ip.dst = dstIP
ip.ttl = 1

icmp = ICMP()

while ip.ttl <=MAX_TTL:
        #Sending the packet
        reply = sr1(ip/icmp, verbose=0, timeout=2)

        #if no echo reply
        if reply == None:
                print(str(ip.ttl) + "\t* * *")
                ip.ttl += 1
                continue

        print(str(ip.ttl) + "\t" + reply.src)

        if(reply.src == dstIP):
                break
```

trace.py program attempts to go to www.google.com and it does so successfully after 16 stops.

```
ubuntu@attacker:~/sniffing_and_spoofing$ sudo python trace.py www.google.com
WARNING: No route found for IPv6 destination :: (no default route?)
trace.py to www.google.com (142.251.40.196) 255 hops max
1       10.0.2.1
2       10.18.0.2
3       10.254.8.44
4       128.122.1.4
5       192.168.184.228
6       128.122.254.114
7       192.168.184.221
8       192.76.177.202
9       10.254.255.27
10      10.254.255.63
11      199.109.105.5
12      199.109.107.166
13      72.14.202.166
14      108.170.248.97
15      216.239.40.187
16      142.251.40.196
```

After 10 hops, the program is no longer receiving a reply. This is because the packets that the program is sending are reaching a certain router and then being dropped due to a firewall before a reply can be sent.

```
ubuntu@attacker:~/sniffing_and_spoofing$ sudo python trace.py 1.2.3.4WARNING: No route found for IPv6 destination :: (no default route?)
trace.py to 1.2.3.4 (1.2.3.4) 255 hops max
1       10.0.2.1
2       10.18.0.2
3       10.254.6.44
4       128.122.1.4
5       192.168.184.228
6       128.122.254.114
7       192.168.184.221
8       192.76.177.202
9       128.122.254.109
10      128.122.254.72
11      * * *
12      * * *
13      * * *
14      * * *
```

The TTL value is decreased every hop the packet makes. If the TTL value runs out before reaching the destination, a ICMP packet with a ‘Time Exceeded’ error type will be sent back as shown in the wireshark screenshot below.

4	0.015668712	10.0.2.1	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
5	0.063366700	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=2 (no response found!)
6	0.065810061	10.18.0.2	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
7	0.116093432	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=3 (no response found!)
8	0.118579568	10.254.6.44	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
9	0.175984739	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=4 (no response found!)
10	0.178230960	128.122.1.4	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
11	0.248579906	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=5 (no response found!)
12	0.251039397	192.168.184.228	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
13	0.303785994	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=6 (no response found!)
14	0.308149408	128.122.254.114	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
15	0.360658181	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=7 (no response found!)
16	0.363420730	192.168.184.221	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
17	0.435536317	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=8 (no response found!)
18	0.438847723	192.76.177.202	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
19	0.492140344	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=9 (no response found!)
20	0.496833299	128.122.254.109	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
21	0.547717073	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=10 (no response found!)
22	0.550532396	128.122.254.72	10.0.2.6	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
23	0.596297335	10.0.2.6	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=11 (no response found!)

Task 1.4: Sniffing and-then Spoofing

The Server VM uses the ping command to ping any website, as long as the Attacker machine is running the sniff and spoof python program, a spoofed ICMP echo reply will be sent back to the Server VM, regardless of whether the website they are pinging is alive.

```
#!/usr/bin/python3
from scapy.all import *

def spoof_pkt(packet):
        if packet[ICMP].type != 8:
                return

        ip  = IP(src = packet[IP].dst, dst = packet[IP].src, ihl = packet[IP].ihl)
        icmp = ICMP(type = 0, id = packet[ICMP].id, seq = packet[ICMP].seq)
        data = packet[Raw].load
        newpacket = ip/icmp/data

        send(newpkt, verbose = 0)

        print("Spoofed packet sent\n")

while(1):
        pkt = sniff(filter = 'icmp', prn = spoof_pkt)
```

Pinging an IP address that isn't alive (1.2.3.4) on the Server machine, we get no replies as seen below, when the sniff_and_spoof.py program isn't running.

```
ubuntu@server:~$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
```

Now, after running the sniff_and_spoof.py program on the attacker machine and pinging 1.2.3.4 on the server machine gives the following result.

```
ubuntu@attacker:~/sniffing_and_spoofing$ sudo python sniff_and_spoof.
py
WARNING: No route found for IPv6 destination :: (no default route?)
Spoofed packet sent

Spoofed packet sent

Spoofed packet sent

Spoofed packet sent

Spoofed packet sent

Spoofed packet sent

Spoofed packet sent

Spoofed packet sent

```

```
ubuntu@server:~$ 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=206 ttl=64 time=57.7 ms
64 bytes from 1.2.3.4: icmp_seq=207 ttl=64 time=25.1 ms
64 bytes from 1.2.3.4: icmp_seq=208 ttl=64 time=20.1 ms
64 bytes from 1.2.3.4: icmp_seq=209 ttl=64 time=15.7 ms
64 bytes from 1.2.3.4: icmp_seq=210 ttl=64 time=18.2 ms
64 bytes from 1.2.3.4: icmp_seq=211 ttl=64 time=16.8 ms
64 bytes from 1.2.3.4: icmp_seq=212 ttl=64 time=17.1 ms
64 bytes from 1.2.3.4: icmp_seq=213 ttl=64 time=19.3 ms
^C
--- 1.2.3.4 ping statistics ---
213 packets transmitted, 8 received, 96% packet loss, time 217931ms
rtt min/avg/max/mdev = 15.757/23.795/57.725/13.105 ms
```

As seen above, echo replies keep coming on the Server machine. It says the replies are from IP 1.2.3.4, but this isn't true because that is a dead IP address as shown before. The replies are actually coming from the sniff_and_spoof.py program that is running on the Attacker machine. The program is working.