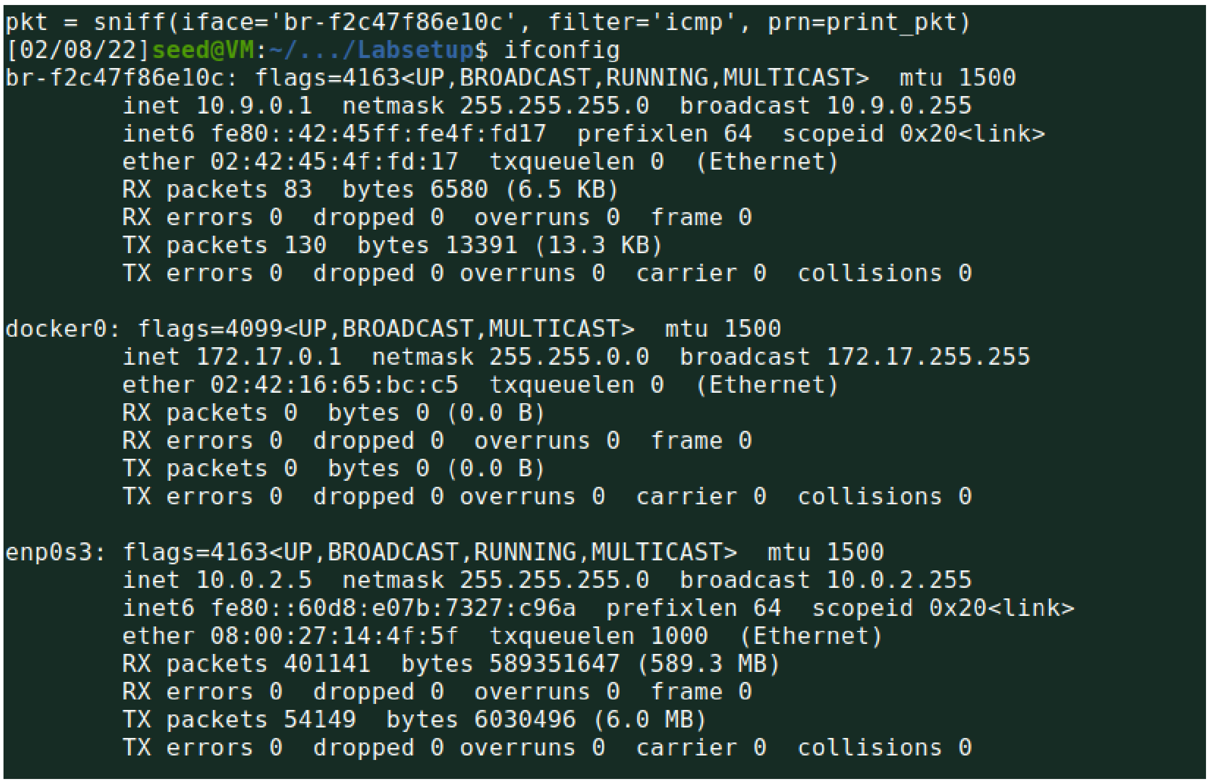
Sniffing/Spoofing Lab 1

Internet Security

Name: Gaurav Upadhyay

Email: gsupadhy@syr.edu

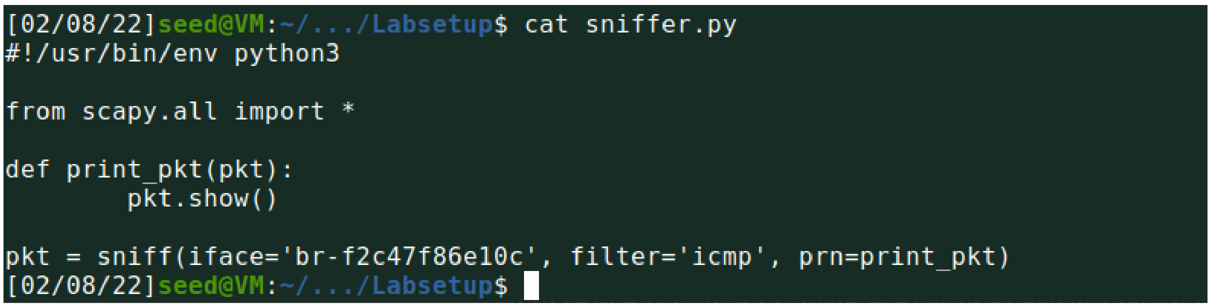
Task 1.1: Sniffing Packets



```
pkt = sniff(iface='br-f2c47f86e10c', filter='icmp', prn=print_pkt)
[02/08/22]seed@VM:~/.../Labsetup$ ifconfig
br-f2c47f86e10c: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:45ff:fe4f:fd17  prefixlen 64  scopeid 0x20<link>
        ether 02:42:45:4f:fd:17  txqueuelen 0  (Ethernet)
        RX packets 83  bytes 6580 (6.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 130  bytes 13391 (13.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:16:65:bc:c5  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.5  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::60d8:e07b:7327:c96a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:14:4f:5f  txqueuelen 1000  (Ethernet)
        RX packets 401141  bytes 589351647 (589.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 54149  bytes 6030496 (6.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

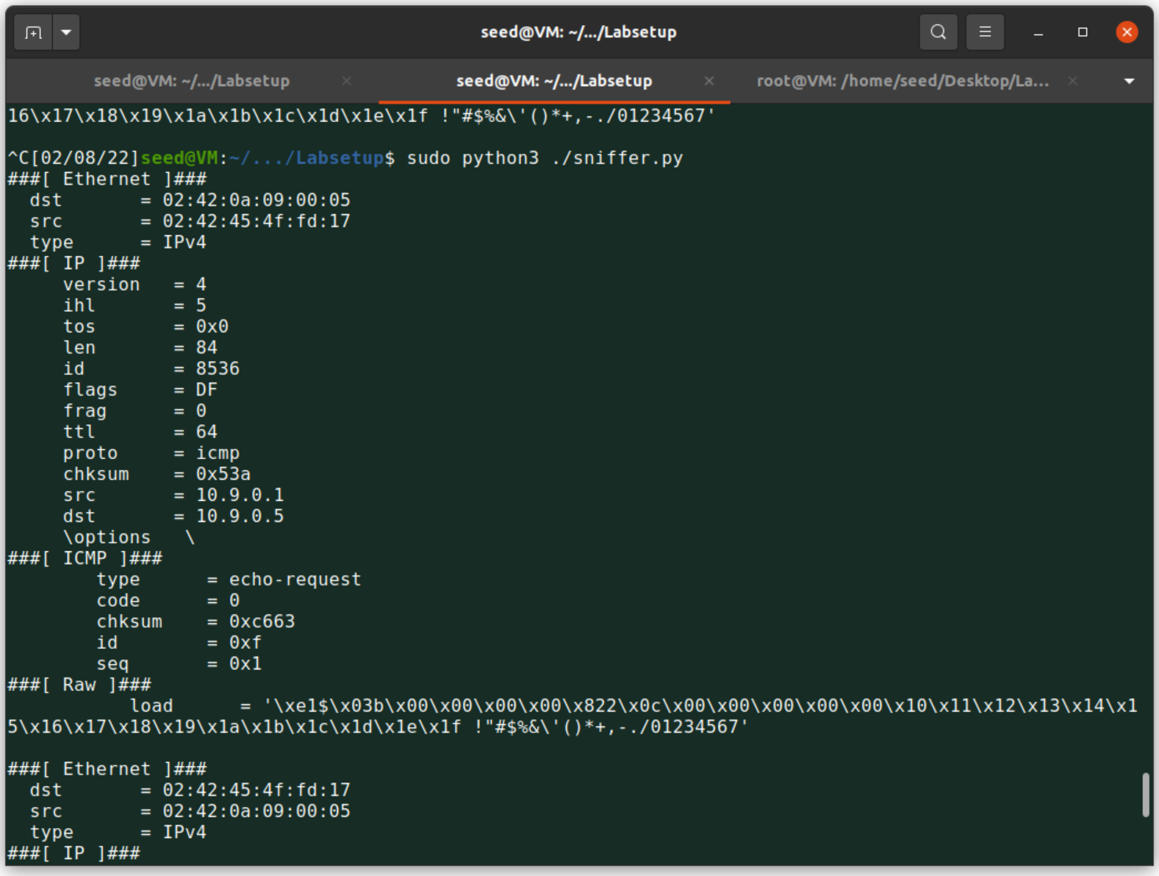
We find the interface name by using the iconfig command.

Task 1.1 A:



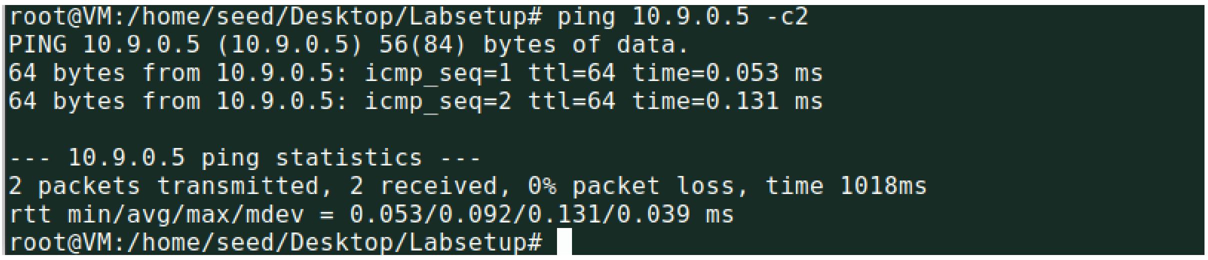
```
[02/08/22]seed@VM:~/.../Labsetup$ cat sniffer.py
#!/usr/bin/env python3

from scapy.all import *

def print_pkt(pkt):
        pkt.show()

pkt = sniff(iface='br-f2c47f86e10c', filter='icmp', prn=print_pkt)
[02/08/22]seed@VM:~/.../Labsetup$ 
```

The code is as follows with the interface name given and filter set to icmp.



```
16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

^C[02/08/22]seed@VM:~/.../Labsetup$ sudo python3 ./sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:45:4f:fd:17
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 8536
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x53a
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xc663
        id        = 0xf
        seq       = 0x1
###[ Raw ]###
           load      = '\xe1$\x03b\x00\x00\x00\x00\x822\x0c\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x1
5\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 02:42:45:4f:fd:17
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
```

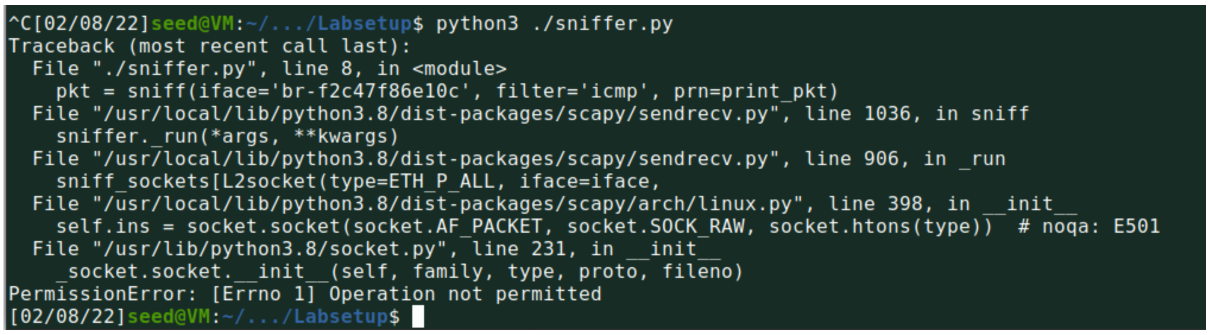
We execute the Scapy code that captures the ICMP packets and displays it. This we do using root privileges first.



```
root@VM:/home/seed/Desktop/Labsetup# ping 10.9.0.5 -c2
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.053 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.131 ms

--- 10.9.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1018ms
rtt min/avg/max/mdev = 0.053/0.092/0.131/0.039 ms
root@VM:/home/seed/Desktop/Labsetup# 
```

Using a separate terminal, we use the ping command to generate ICMP packets which displays the content of the packet such as Ethernet headers, IP headers, ICMP headers, payload, etc.

Now we try to run the same code without the root privileges:

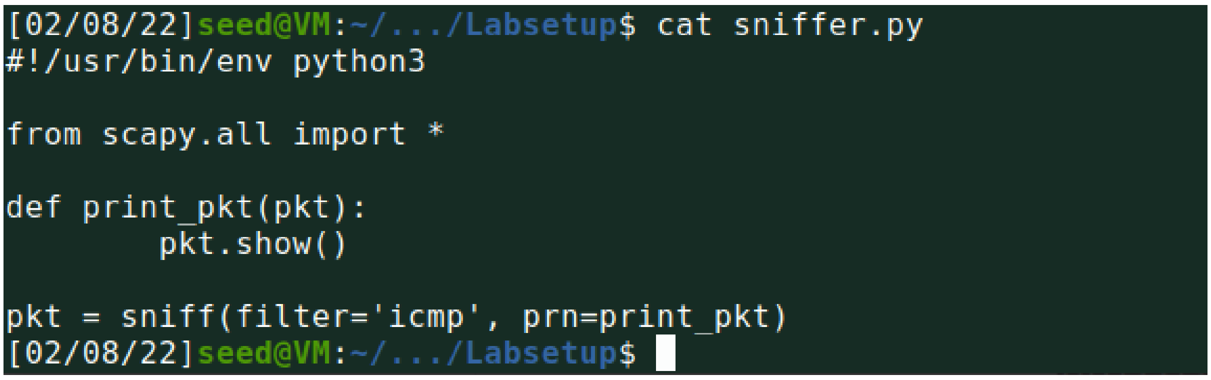


```
^C[02/08/22]seed@VM:~/.../Labsetup$ python3 ./sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 8, in <module>
    pkt = sniff(iface='br-f2c47f86e10c', filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    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
[02/08/22]seed@VM:~/.../Labsetup$ 
```

We can see that it shows an error as this operation is not permitted as sniff function calls for raw socket initialization. This enables the promiscuous mode which requires root privilege which isn’t there. Thus, the error message is displayed.

Task 1.1 B:

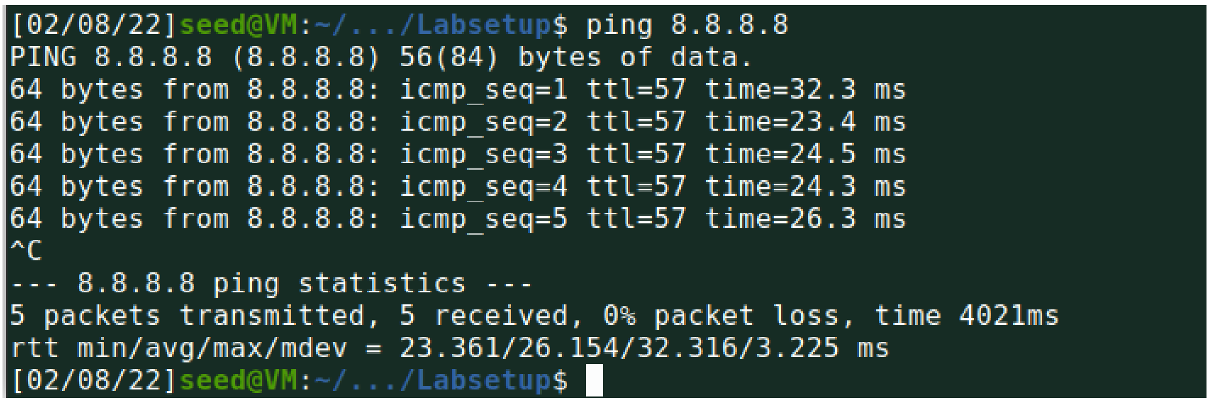
Capture only ICMP packets:



```
[02/08/22]seed@VM:~/.../Labsetup$ cat sniffer.py
#!/usr/bin/env python3

from scapy.all import *

def print_pkt(pkt):
        pkt.show()

pkt = sniff(filter='icmp', prn=print_pkt)
[02/08/22]seed@VM:~/.../Labsetup$ 
```

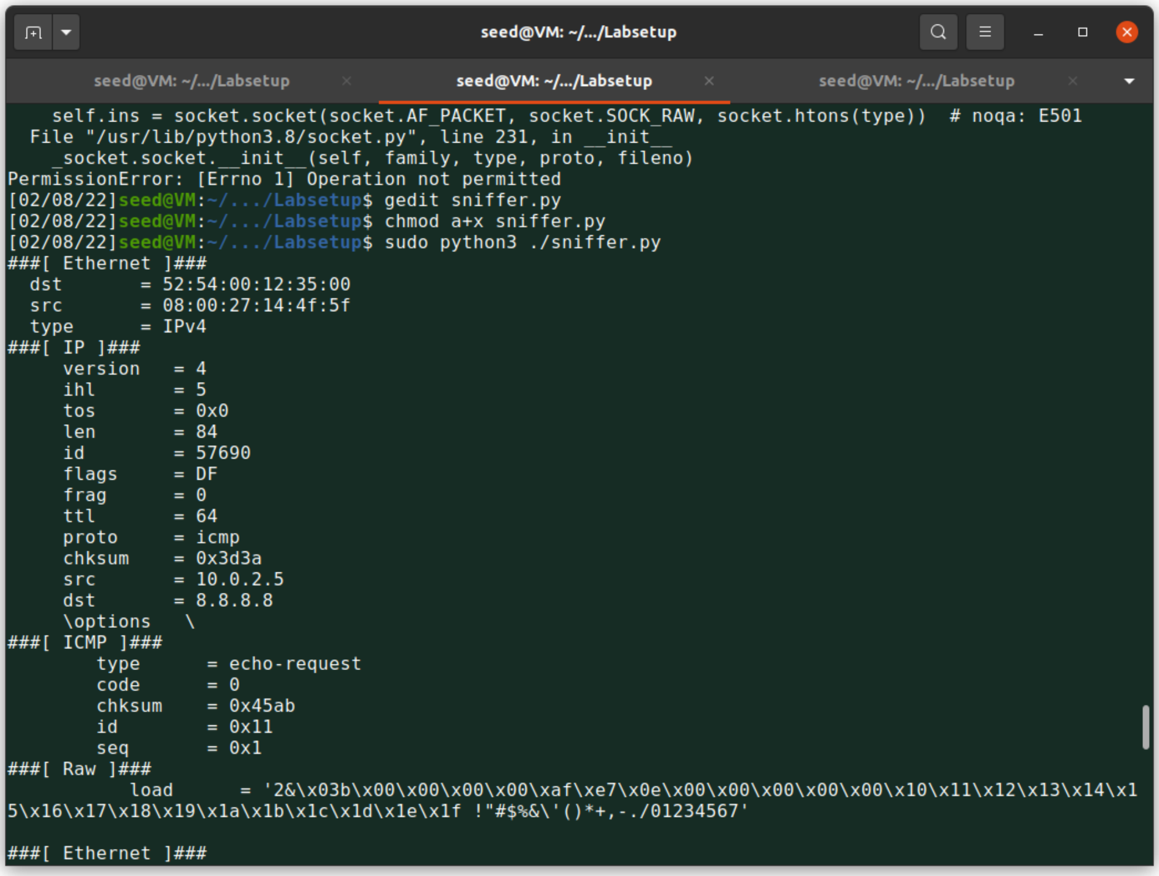
The following code shows Scapy code for ICMP filter only.

We run the above code and we ping from another terminal using the address 8.8.8.8.



```
[02/08/22]seed@VM:~/.../Labsetup$ 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=57 time=32.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=57 time=23.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=57 time=24.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=57 time=24.3 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=57 time=26.3 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4021ms
rtt min/avg/max/mdev = 23.361/26.154/32.316/3.225 ms
[02/08/22]seed@VM:~/.../Labsetup$ 
```

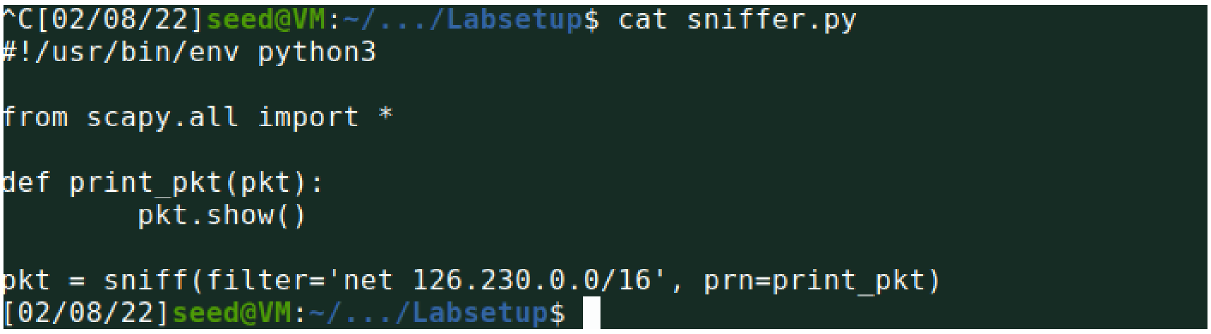
We can see that our program sniffs the packets when we run the program on the network and displays various information with the packets.



```
    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
[02/08/22]seed@VM:~/.../Labsetup$ gedit sniffer.py
[02/08/22]seed@VM:~/.../Labsetup$ chmod a+x sniffer.py
[02/08/22]seed@VM:~/.../Labsetup$ sudo python3 ./sniffer.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:14:4f:5f
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 57690
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x3d3a
     src       = 10.0.2.5
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x45ab
        id        = 0x11
        seq       = 0x1
###[ Raw ]###
           load      = '2&\x03b\x00\x00\x00\x00\xaf\xe7\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x1
5\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
```

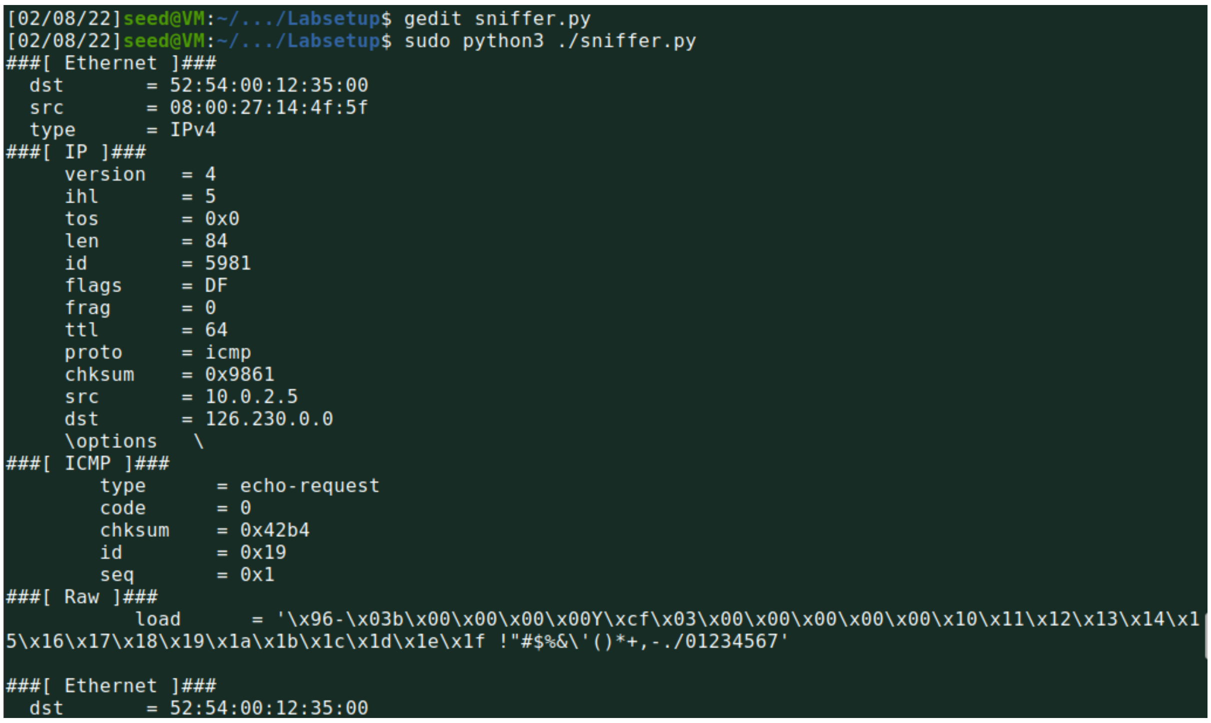
Capture any TCP packet that comes from a particular IP and with a destination port number 23:

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:

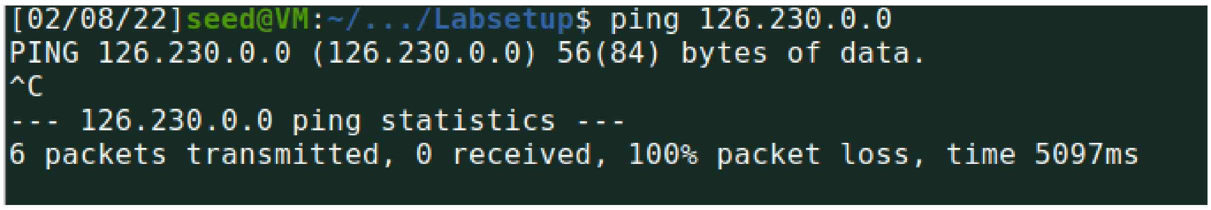


```
^C[02/08/22]seed@VM:~/.../Labsetup$ cat sniffer.py
#!/usr/bin/env python3

from scapy.all import *

def print_pkt(pkt):
        pkt.show()

pkt = sniff(filter='net 126.230.0.0/16', prn=print_pkt)
[02/08/22]seed@VM:~/.../Labsetup$ 
```

The following code shows the Scapy code for the expected filter given.



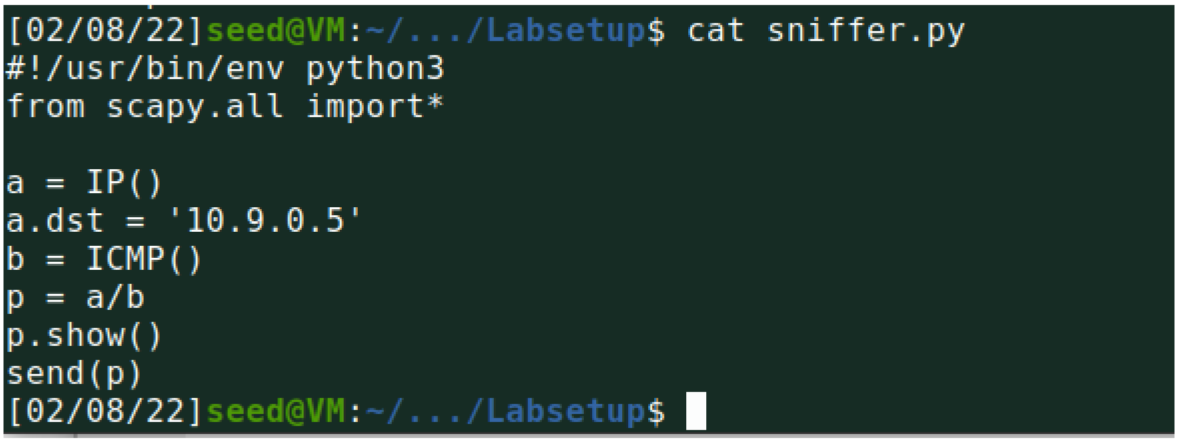
```
[02/08/22]seed@VM:~/.../Labsetup$ gedit sniffer.py
[02/08/22]seed@VM:~/.../Labsetup$ sudo python3 ./sniffer.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:14:4f:5f
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 5981
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x9861
     src       = 10.0.2.5
     dst       = 126.230.0.0
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x42b4
        id        = 0x19
        seq       = 0x1
###[ Raw ]###
           load      = '\x96-\x03b\x00\x00\x00\x00Y\xcf\x03\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x1
5\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
```



```
[02/08/22]seed@VM:~/.../Labsetup$ ping 126.230.0.0
PING 126.230.0.0 (126.230.0.0) 56(84) bytes of data.
^C
--- 126.230.0.0 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5097ms
```

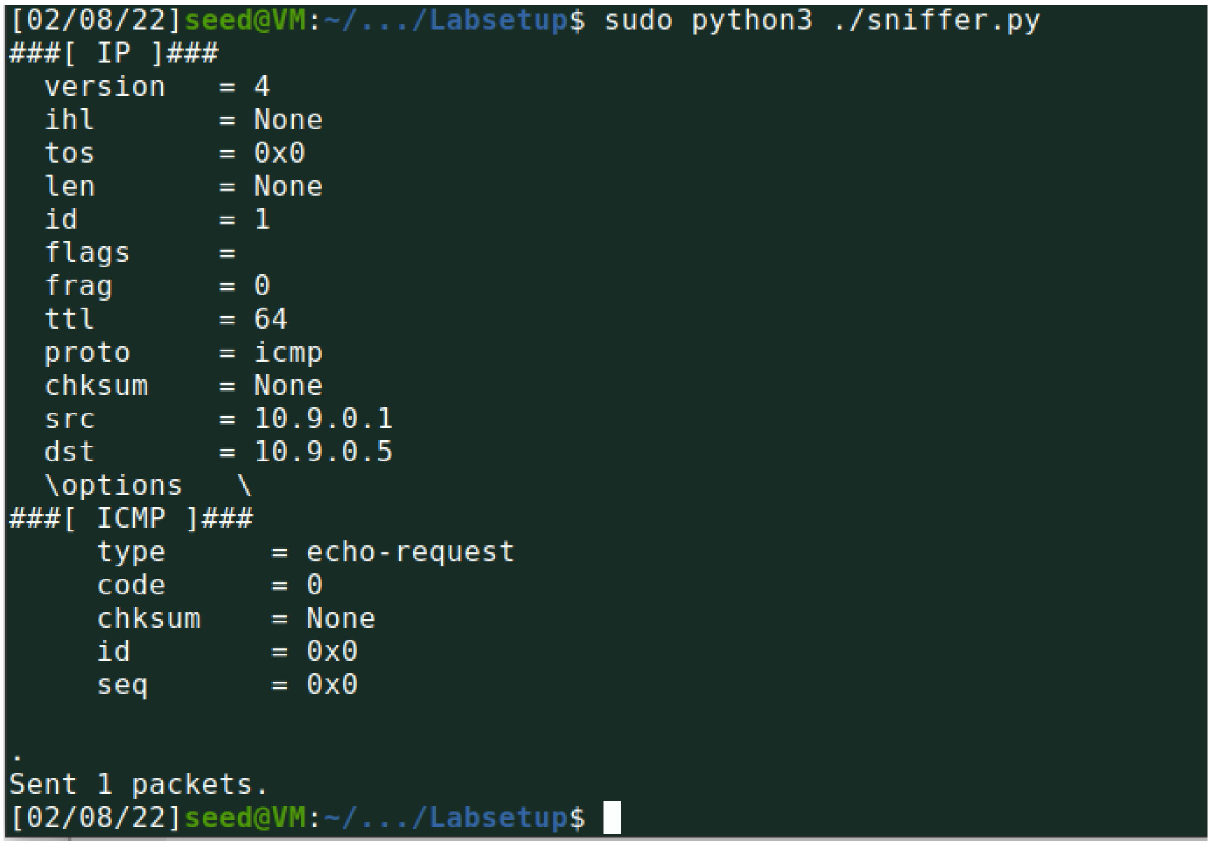
When we run the program and ping through the subnet, we can see that it is only capturing through that filtered subnet.

Task 1.2: Spoofing ICMP Packets



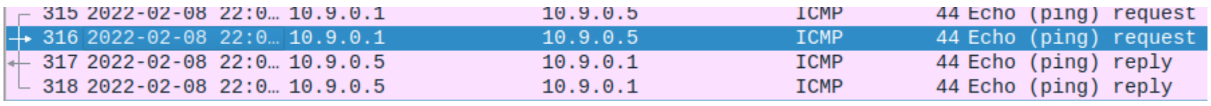
```
[02/08/22]seed@VM:~/.../Labsetup$ cat sniffer.py
#!/usr/bin/env python3
from scapy.all import*

a = IP()
a.dst = '10.9.0.5'
b = ICMP()
p = a/b
p.show()
send(p)
[02/08/22]seed@VM:~/.../Labsetup$
```

Given above is the following code with the necessary changes in the destination IP address.



```
[02/08/22]seed@VM:~/.../Labsetup$ sudo python3 ./sniffer.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = None
  src       = 10.9.0.1
  dst       = 10.9.0.5
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
     id        = 0x0
     seq       = 0x0

.
Sent 1 packets.
[02/08/22]seed@VM:~/.../Labsetup$
```

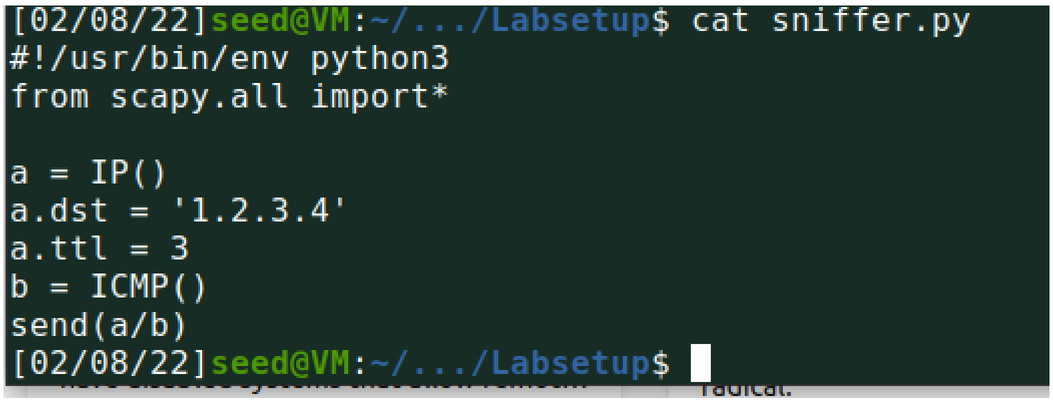
We can see that the packet was sent successfully. We can confirm this using Wireshark application.



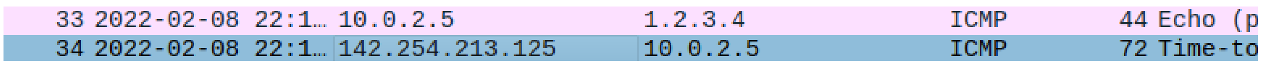
```
315 2022-02-08 22:0… 10.9.0.1    10.9.0.5    ICMP    44 Echo (ping) request
316 2022-02-08 22:0… 10.9.0.1    10.9.0.5    ICMP    44 Echo (ping) request
317 2022-02-08 22:0… 10.9.0.5    10.9.0.1    ICMP    44 Echo (ping) reply
318 2022-02-08 22:0… 10.9.0.5    10.9.0.1    ICMP    44 Echo (ping) reply
```

This validates that we can spoof any IP address.

Task 1.3 Traceroute:



```
[02/08/22]seed@VM:~/.../Labsetup$ cat sniffer.py
#!/usr/bin/env python3
from scapy.all import*

a = IP()
a.dst = '1.2.3.4'
a.ttl = 3
b = ICMP()
send(a/b)
[02/08/22]seed@VM:~/.../Labsetup$
```



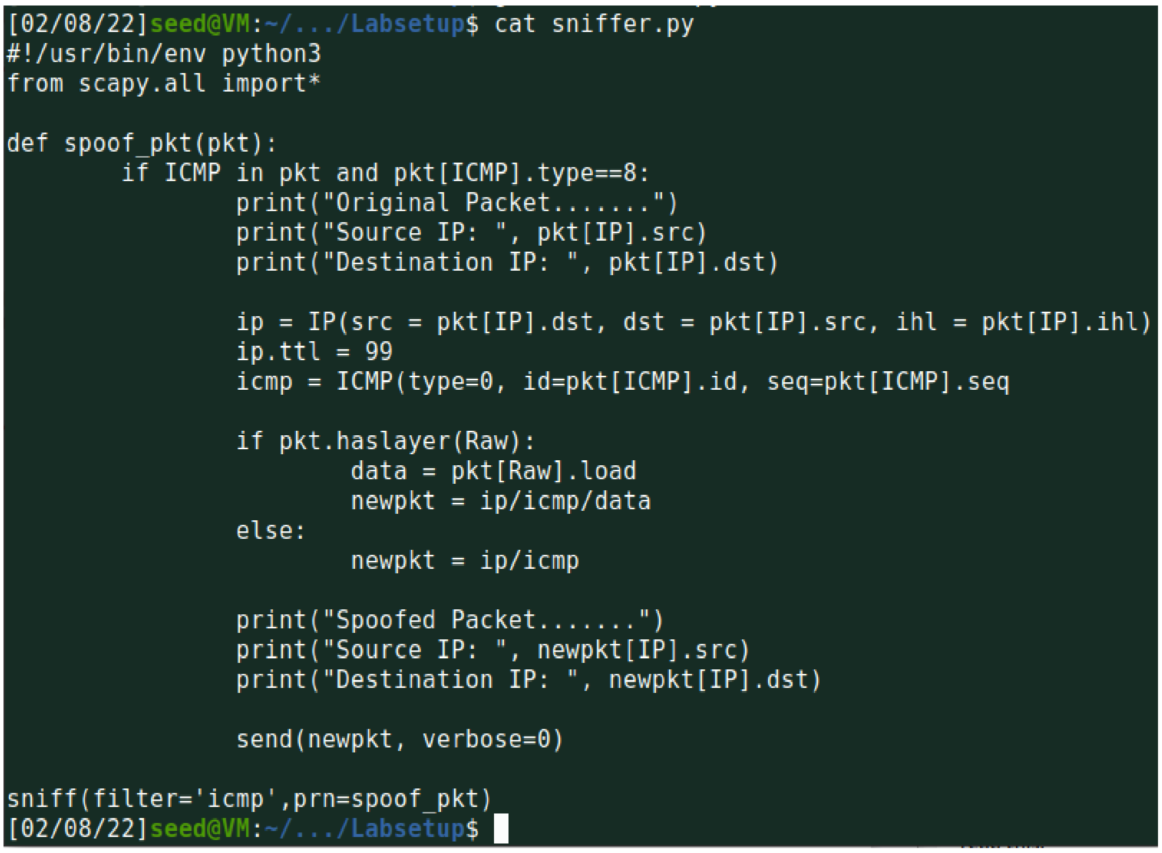
```
33 2022-02-08 22:1… 10.0.2.5         1.2.3.4     ICMP    44 Echo (p
34 2022-02-08 22:1… 142.254.213.125  10.0.2.5    ICMP    72 Time-to
```

When we send a packet with time-to-live field set to 3 as shown, it will drop the packets if it exceeds the ttl and give the IP address of first router and continues so we can reach our destination.

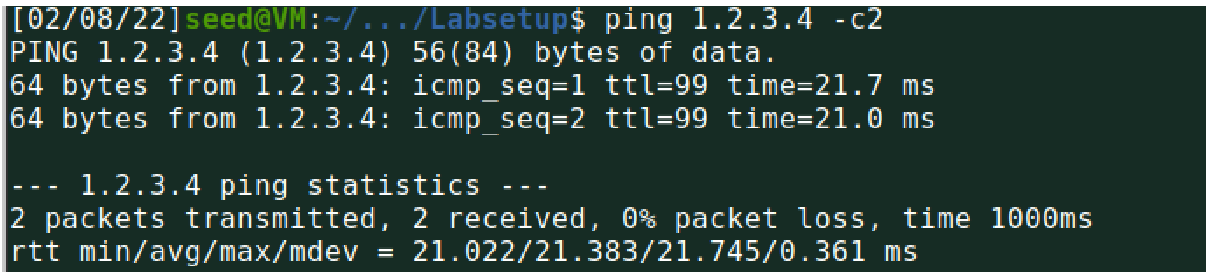
This is done to avoid overuse of the network and drop unwanted data.

Task 1.4: Sniffing and-then Spoofing:

Code:

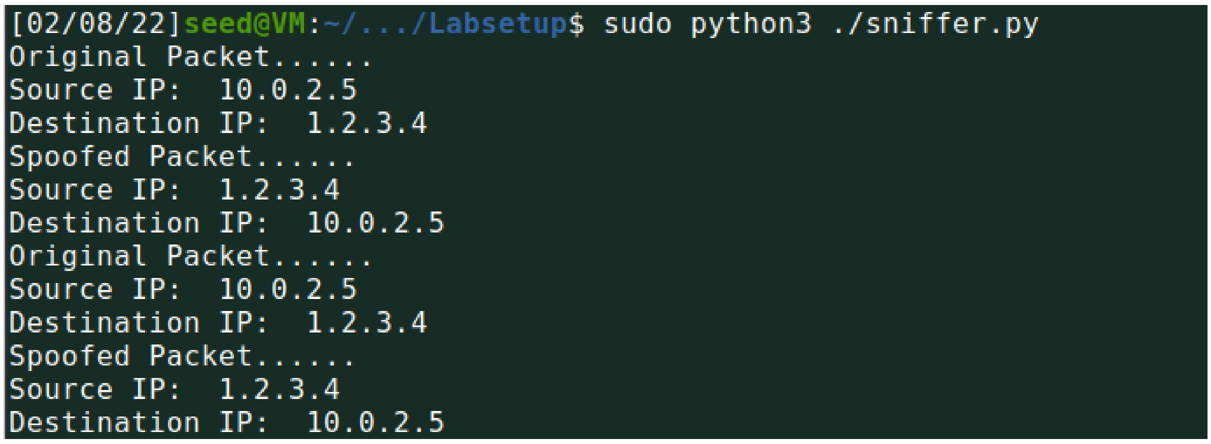


```
[02/08/22]seed@VM:~/.../Labsetup$ cat sniffer.py
#!/usr/bin/env python3
from scapy.all import*

def spoof_pkt(pkt):
        if ICMP in pkt and pkt[ICMP].type==8:
                print("Original Packet.......")
                print("Source IP: ", pkt[IP].src)
                print("Destination IP: ", pkt[IP].dst)

                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

                print("Spoofed Packet.......")
                print("Source IP: ", newpkt[IP].src)
                print("Destination IP: ", newpkt[IP].dst)

                send(newpkt, verbose=0)

sniff(filter='icmp',prn=spoof_pkt)
[02/08/22]seed@VM:~/.../Labsetup$
```



```
[02/08/22]seed@VM:~/.../Labsetup$ ping 1.2.3.4 -c2
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=21.7 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=99 time=21.0 ms

--- 1.2.3.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 21.022/21.383/21.745/0.361 ms
```

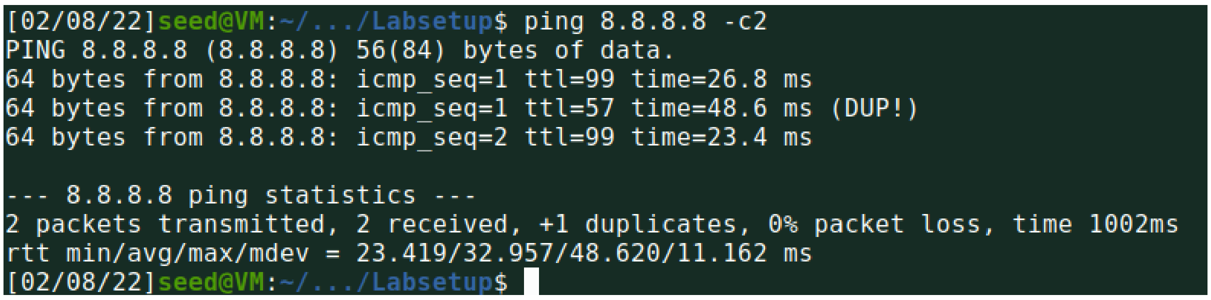
We write the code and ping 1.2.3.4. and run the sniffing and spoofing program where ICMP echo request is sent.



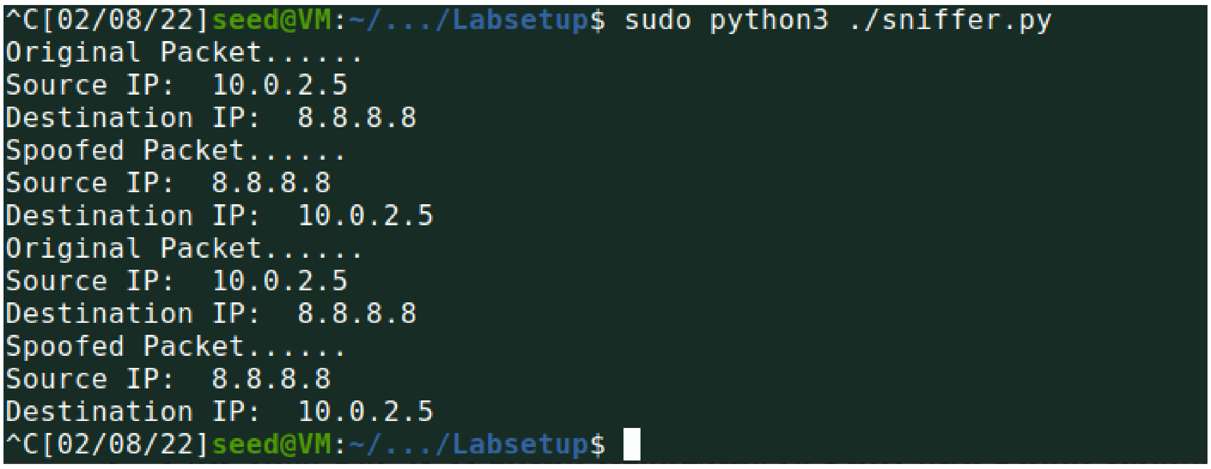
```
[02/08/22]seed@VM:~/.../Labsetup$ sudo python3 ./sniffer.py
Original Packet......
Source IP:  10.0.2.5
Destination IP:  1.2.3.4
Spoofed Packet......
Source IP:  1.2.3.4
Destination IP:  10.0.2.5
Original Packet......
Source IP:  10.0.2.5
Destination IP:  1.2.3.4
Spoofed Packet......
Source IP:  1.2.3.4
Destination IP:  10.0.2.5
```

We can see the output here.

Now we send the ping using 8.8.8.8 an existing host on the Internet:



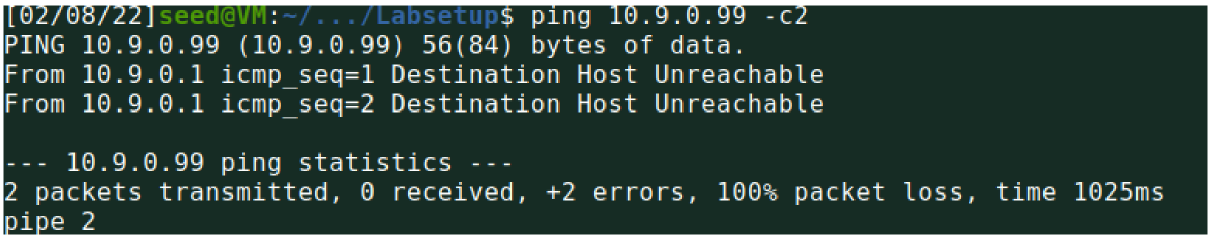
```
[02/08/22]seed@VM:~/.../Labsetup$ ping 8.8.8.8 -c2
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=99 time=26.8 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=48.6 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=99 time=23.4 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, +1 duplicates, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 23.419/32.957/48.620/11.162 ms
[02/08/22]seed@VM:~/.../Labsetup$ 
```



```
^C[02/08/22]seed@VM:~/.../Labsetup$ sudo python3 ./sniffer.py
Original Packet......
Source IP:  10.0.2.5
Destination IP:  8.8.8.8
Spoofed Packet......
Source IP:  8.8.8.8
Destination IP:  10.0.2.5
Original Packet......
Source IP:  10.0.2.5
Destination IP:  8.8.8.8
Spoofed Packet......
Source IP:  8.8.8.8
Destination IP:  10.0.2.5
^C[02/08/22]seed@VM:~/.../Labsetup$ 
```

We can see that the response to the ping is sent as the output.

However when we try to run the program using ping a non-existing host on the LAN, the packets are unable to reach as the destination host is unreachable.



```
[02/08/22]seed@VM:~/.../Labsetup$ ping 10.9.0.99 -c2
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.1 icmp_seq=1 Destination Host Unreachable
From 10.9.0.1 icmp_seq=2 Destination Host Unreachable

--- 10.9.0.99 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1025ms
pipe 2
```