

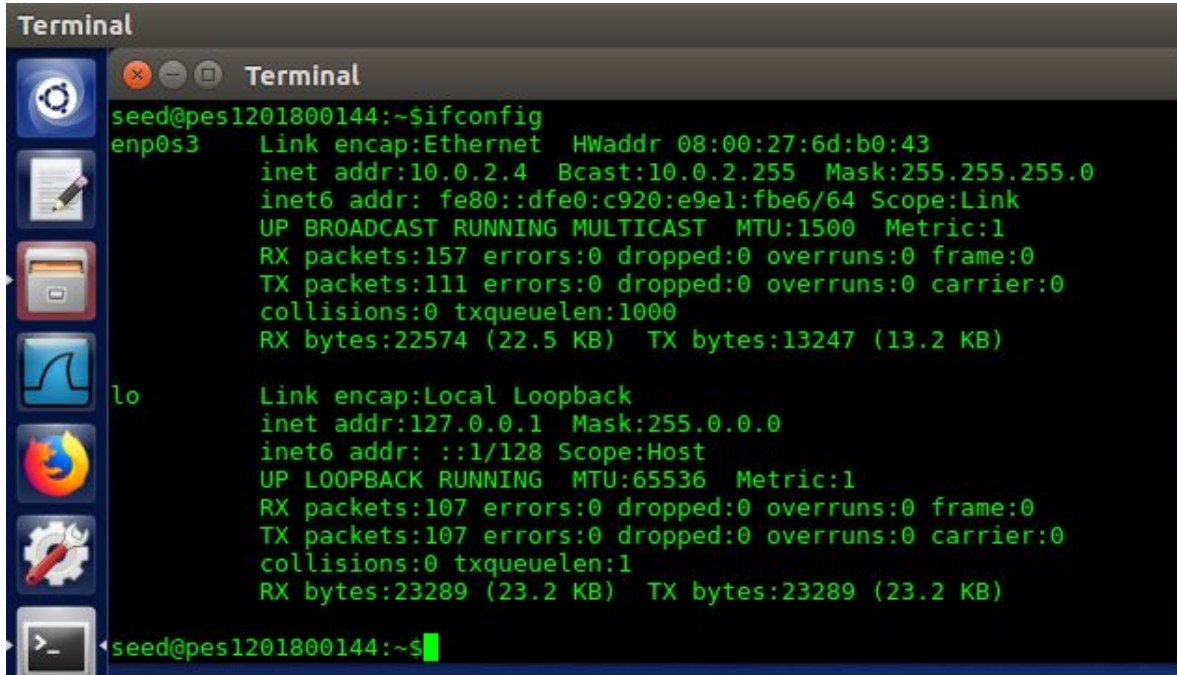
**COMPUTER NETWORKS SECURITY**  
**LABORATORY**  
**WEEK 1**  
**BY: RAJDEEP SENGUPTA**  
**SRN: PES1201800144**  
**SECTION: C**

**NOTE: Please find my SRN 'PES1201800144' in all the screenshots attached below. Also find these screenshots in the attached folder in the 'attacker machine screenshots' and 'victim machine screenshots' directories.**

## MY CONFIGURATIONS:

VM1 (Attacker)

IP Address: 10.0.2.4 WITH GREEN TERMINAL TEXT



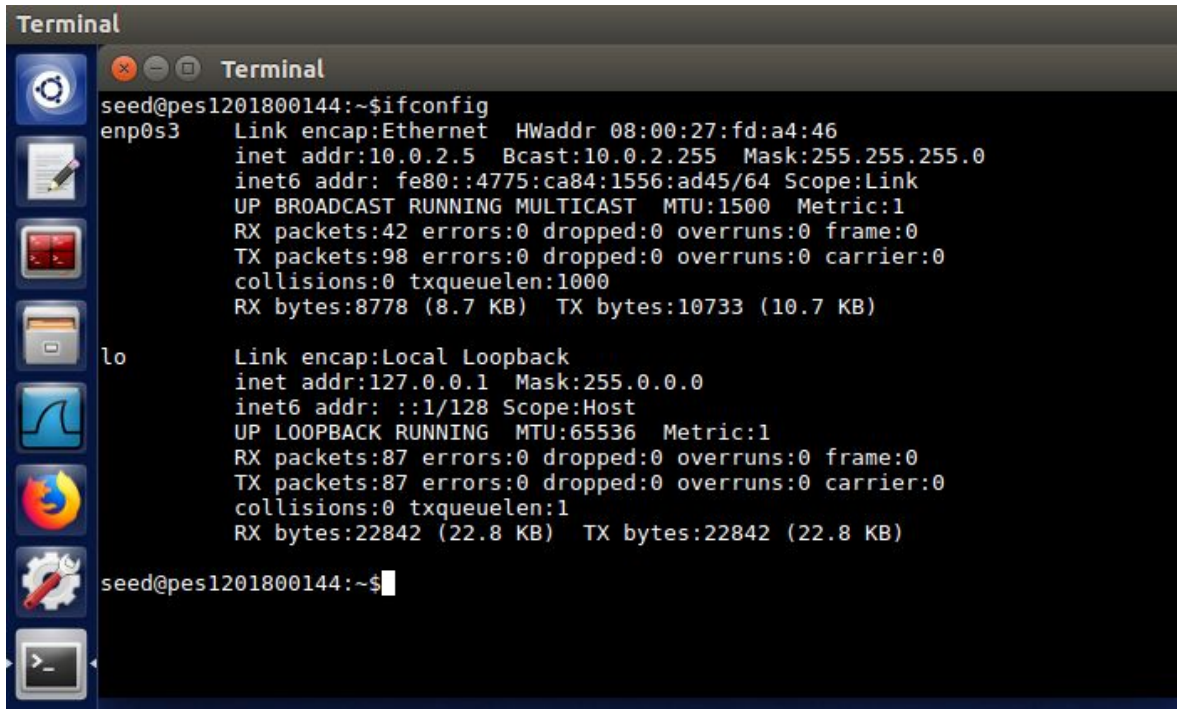
```
Terminal
seed@pes1201800144:~$ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:6d:b0:43
            inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::dfe0:c920:e9e1:fbe6/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:157 errors:0 dropped:0 overruns:0 frame:0
            TX packets:111 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:22574 (22.5 KB)  TX bytes:13247 (13.2 KB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:107 errors:0 dropped:0 overruns:0 frame:0
            TX packets:107 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:23289 (23.2 KB)  TX bytes:23289 (23.2 KB)

seed@pes1201800144:~$
```

VM2 (Victim)

IP Address: 10.0.2.5 WITH WHITE TERMINAL TEXT



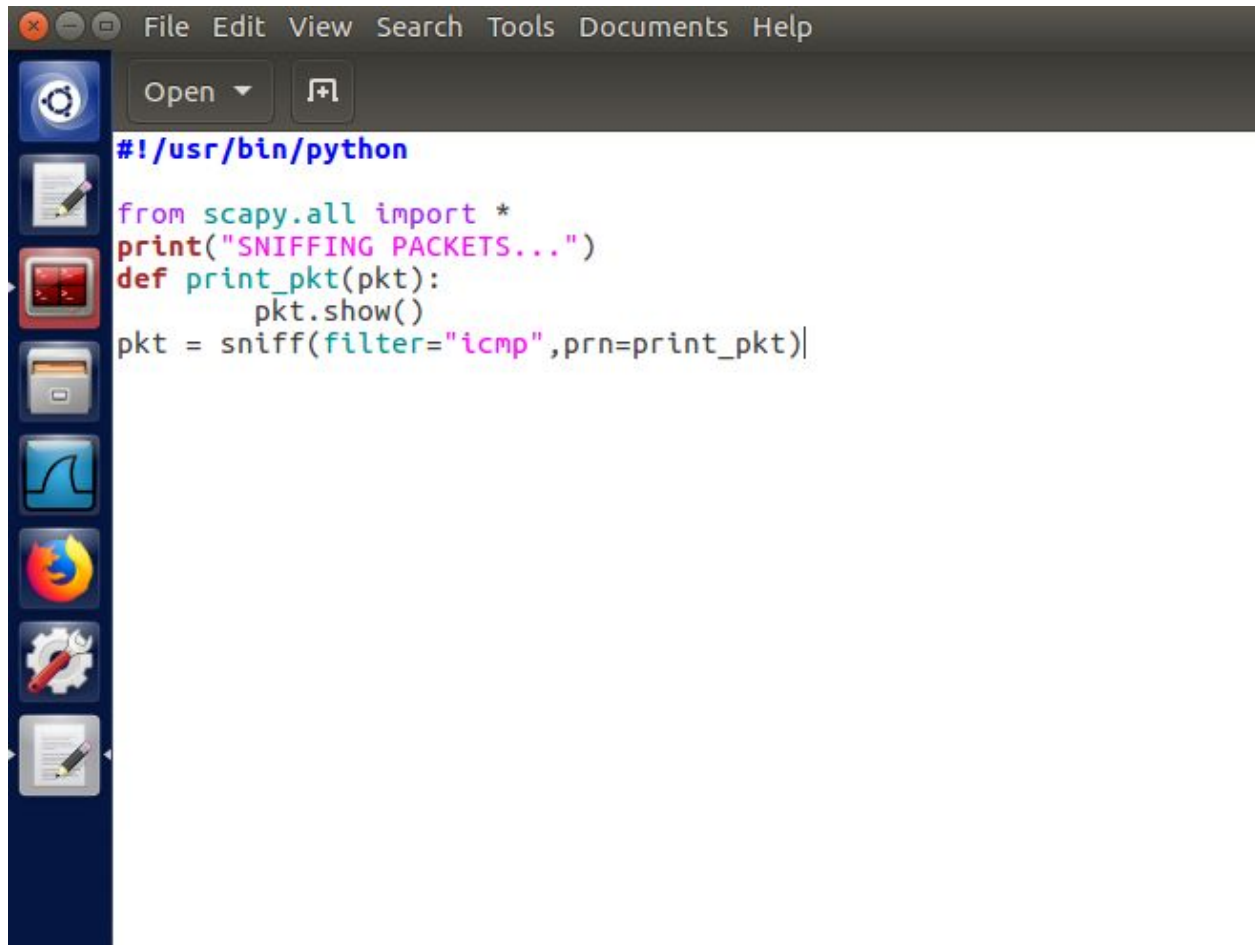
```
Terminal
seed@pes1201800144:~$ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:fd:a4:46
            inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::4775:ca84:1556:ad45/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:42 errors:0 dropped:0 overruns:0 frame:0
            TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:8778 (8.7 KB)  TX bytes:10733 (10.7 KB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:87 errors:0 dropped:0 overruns:0 frame:0
            TX packets:87 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:22842 (22.8 KB)  TX bytes:22842 (22.8 KB)

seed@pes1201800144:~$
```

# TASKS

## TASK 2.1.1

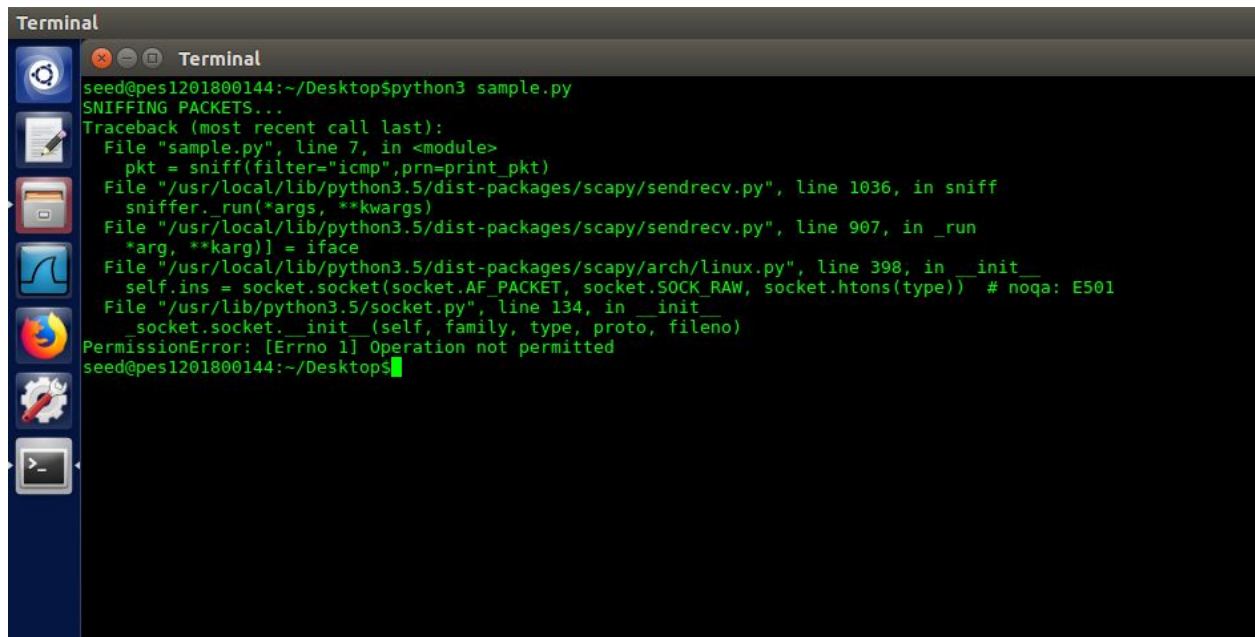
A screenshot of a Linux terminal window. The window has a dark gray title bar with standard window controls (close, minimize, maximize) and a menu bar with options: File, Edit, View, Search, Tools, Documents, and Help. Below the menu bar is a toolbar with an 'Open' button and a file icon. The main area of the terminal displays Python code for sniffing ICMP packets. The code is as follows:

```
#!/usr/bin/python

from scapy.all import *
print("SNIFFING PACKETS...")
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter="icmp",prn=print_pkt)|
```

The code is color-coded: the shebang is blue, imports are green, strings are pink, and function names/variables are blue. The left sidebar of the terminal shows a vertical stack of application icons, including a terminal, a text editor, a file manager, a web browser, and a system settings icon.

Figure: Code to capture ICMP packets

A terminal window titled "Terminal" is shown. The user has executed the command `python3 sample.py`. The output shows the script attempting to sniff packets, but it fails with a `PermissionError: [Errno 1] Operation not permitted`. The error traceback indicates the failure occurs in the `sniff` function of the `scapy` library, specifically when trying to create a raw socket.

```
seed@pes1201800144:~/Desktop$python3 sample.py
SNIFFING PACKETS...
Traceback (most recent call last):
  File "sample.py", line 7, in <module>
    pkt = sniff(filter="icmp",prn=print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 907, in _run
    *arg, **karg)) = iface
  File "/usr/local/lib/python3.5/dist-packages/scapy/arch/linux.py", line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type)) # noqa: E501
  File "/usr/lib/python3.5/socket.py", line 134, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
seed@pes1201800144:~/Desktop$
```

Figure: Running the code without sudo

## TASK 2.1.2.1

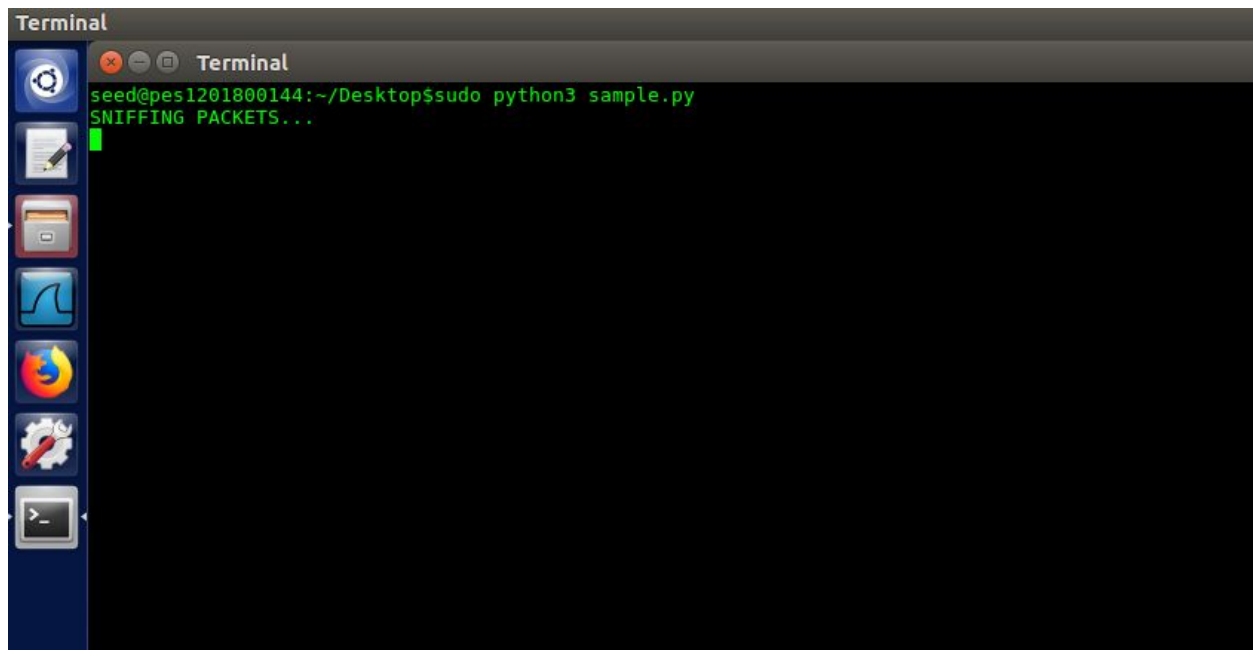


Figure: Running with sudo (root permission)

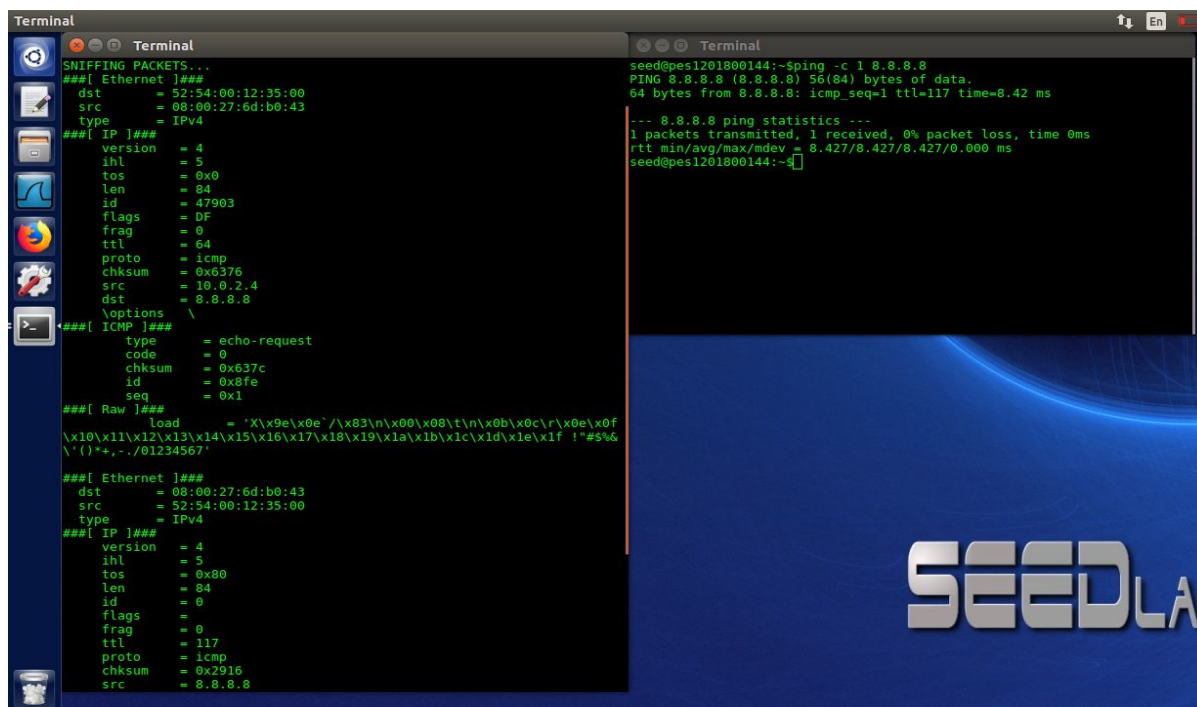
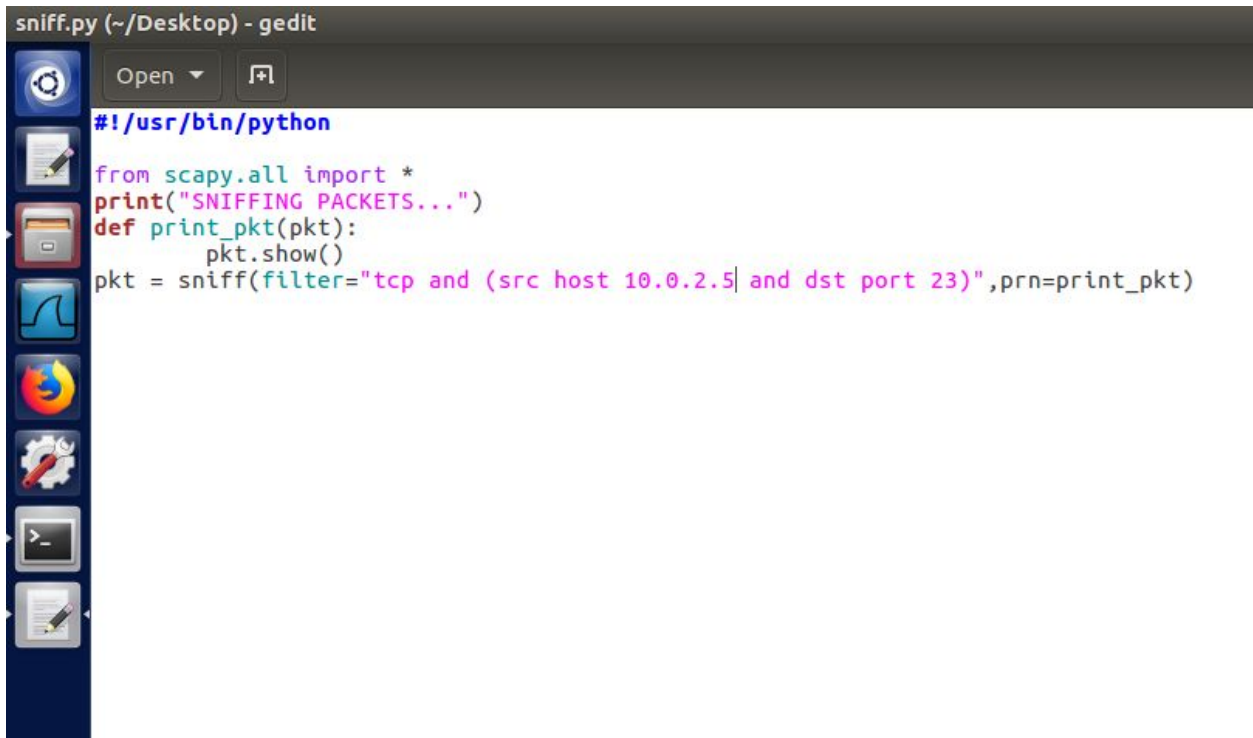


Figure: ICMP packets captured by sniffer program

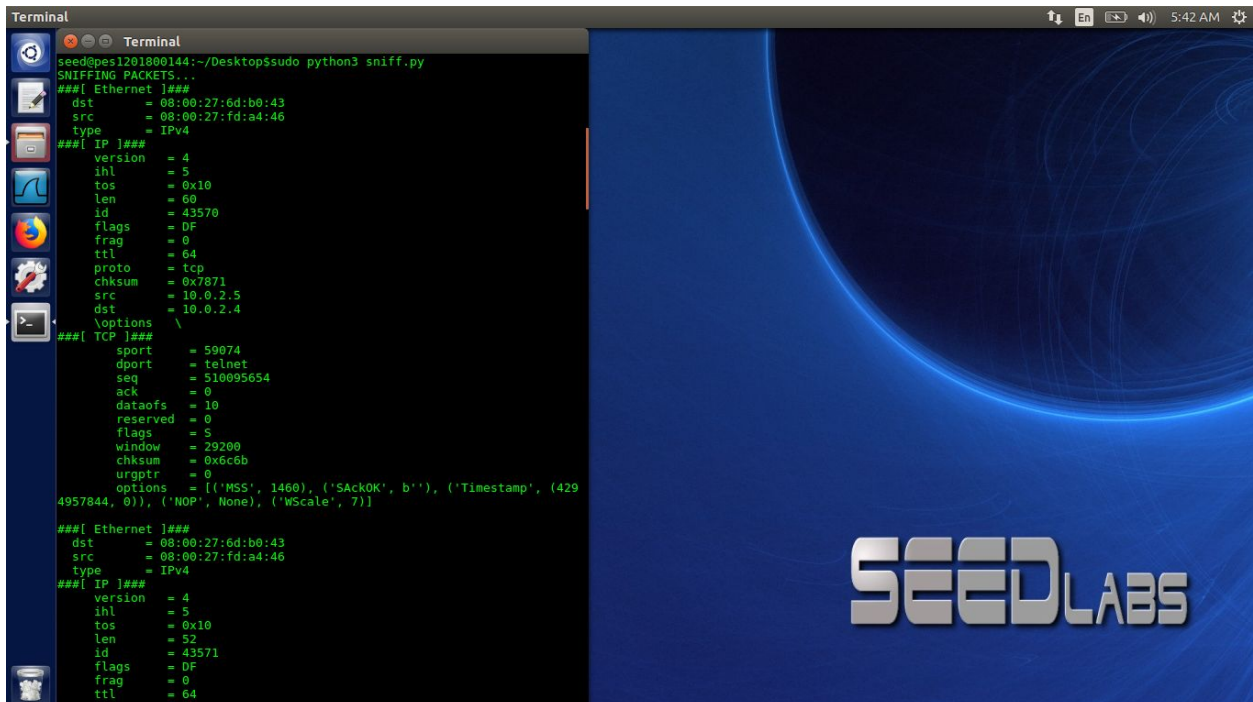
## TASK 2.1.2.2

A screenshot of a gedit text editor window titled "sniff.py (~/Desktop) - gedit". The editor shows a Python script for sniffing network packets. The script starts with a shebang line, imports scapy, prints a message, defines a callback function, and then uses the sniff function with a specific filter. The left sidebar shows various application icons.

```
#!/usr/bin/python

from scapy.all import *
print("SNIFFING PACKETS...")
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter="tcp and (src host 10.0.2.5 and dst port 23)",prn=print_pkt)
```

Figure: TCP sniffer code

A screenshot of a terminal window showing the output of the sniff.py script. The output displays the details of two captured packets, including Ethernet II, Internet Protocol (IPv4), and Transmission Control Protocol (TCP) headers. The background of the terminal window features a blue abstract design and the "SEEDLABS" logo.

```
seed@ps1201800144:~/Desktop$ sudo python3 sniff.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst      = 08:00:27:6d:b0:43
  src      = 08:00:27:fd:a4:46
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x10
  len      = 60
  id       = 43570
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = tcp
  checksum = 0x7871
  src      = 10.0.2.5
  dst      = 10.0.2.4
  \options \
###[ TCP ]###
  sport    = 59074
  dport    = telnet
  seq      = 510095654
  ack      = 0
  dataofs  = 10
  reserved = 0
  flags    = S
  window   = 29200
  checksum = 0x6c6b
  urgptr   = 0
  options  = [('MSS', 1460), ('SACKOK', b''), ('Timestamp', (429
4957844, 0)), ('NOP', None), ('WScale', 7)]
###[ Ethernet ]###
  dst      = 08:00:27:6d:b0:43
  src      = 08:00:27:fd:a4:46
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x10
  len      = 52
  id       = 43571
  flags    = DF
  frag     = 0
  ttl      = 64
```

Figure: TCP packets captured by sniffer program



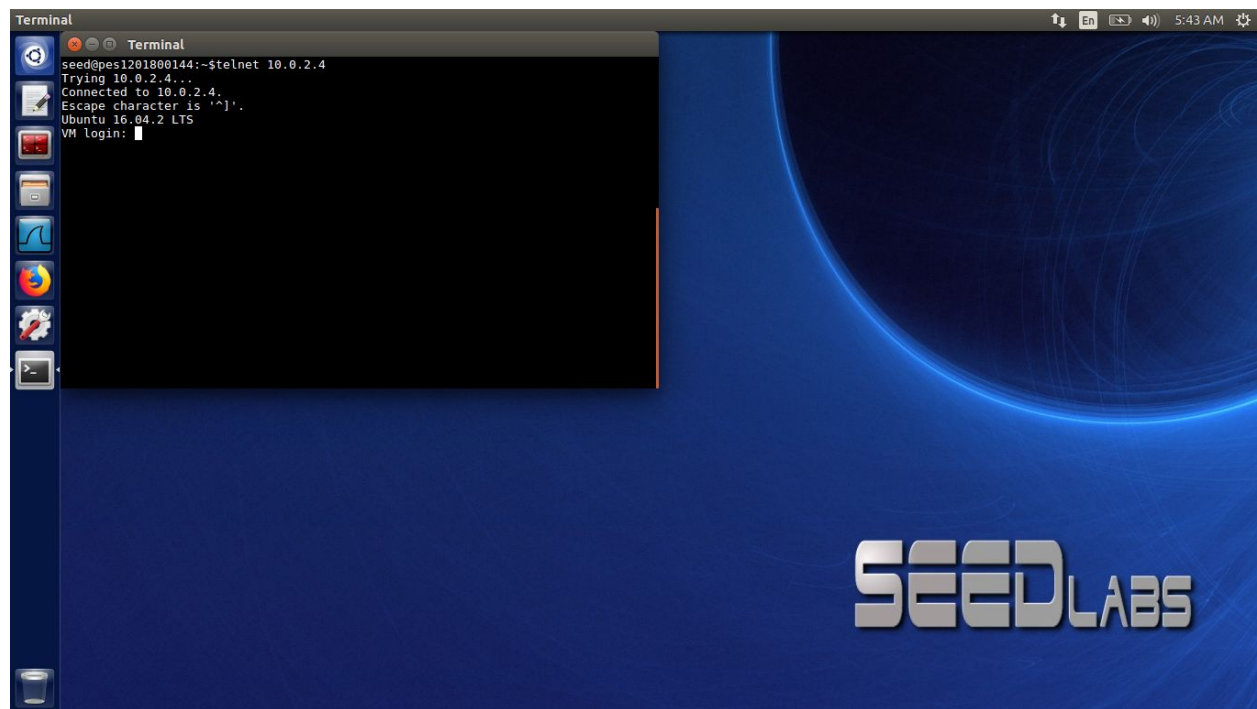
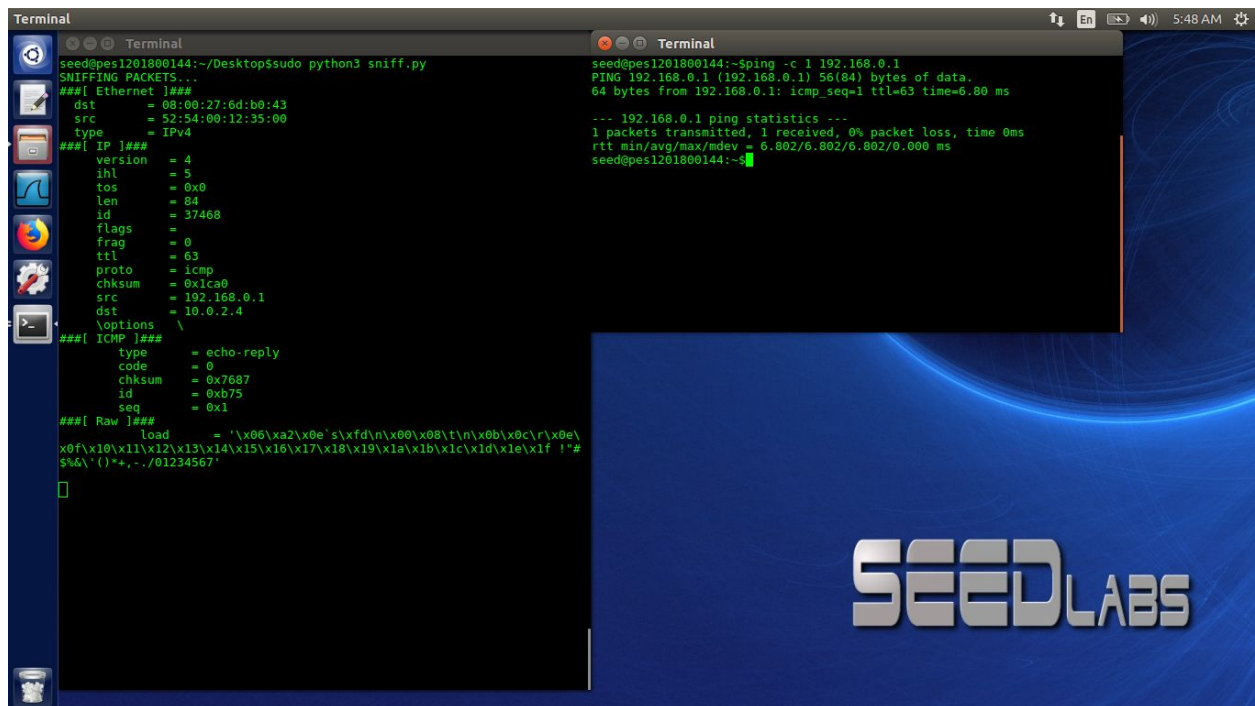


Figure: Telnet connecting from victim machine

## TASK 2.1.2.2.1



The image shows two terminal windows on a Linux desktop. The left window displays the output of a packet sniffing script, showing details for an Ethernet II frame, an IP packet, and an ICMP Echo Reply. The right window shows the output of a ping command to 192.168.0.1, including the raw data and ping statistics.

```
seed@pes1201800144:~/Desktop$ sudo python3 sniff.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst      = 08:00:27:6d:b0:43
  src      = 52:54:00:12:35:00
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 37468
  flags    =
  frag     = 0
  ttl      = 63
  proto    = icmp
  chksum   = 0x1ca0
  src      = 192.168.0.1
  dst      = 10.0.2.4
  \options \
###[ ICMP ]###
  type     = echo-reply
  code     = 0
  chksum   = 0x7687
  id       = 0xb75
  seq      = 0x1
###[ Raw ]###
  load     = '\x06\xa2\xe's\xfd\n\x00\x08\t\n\x0b\x0c\r\x0e\x
x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f'!"#
$%&'()*+,-./01234567'
[]

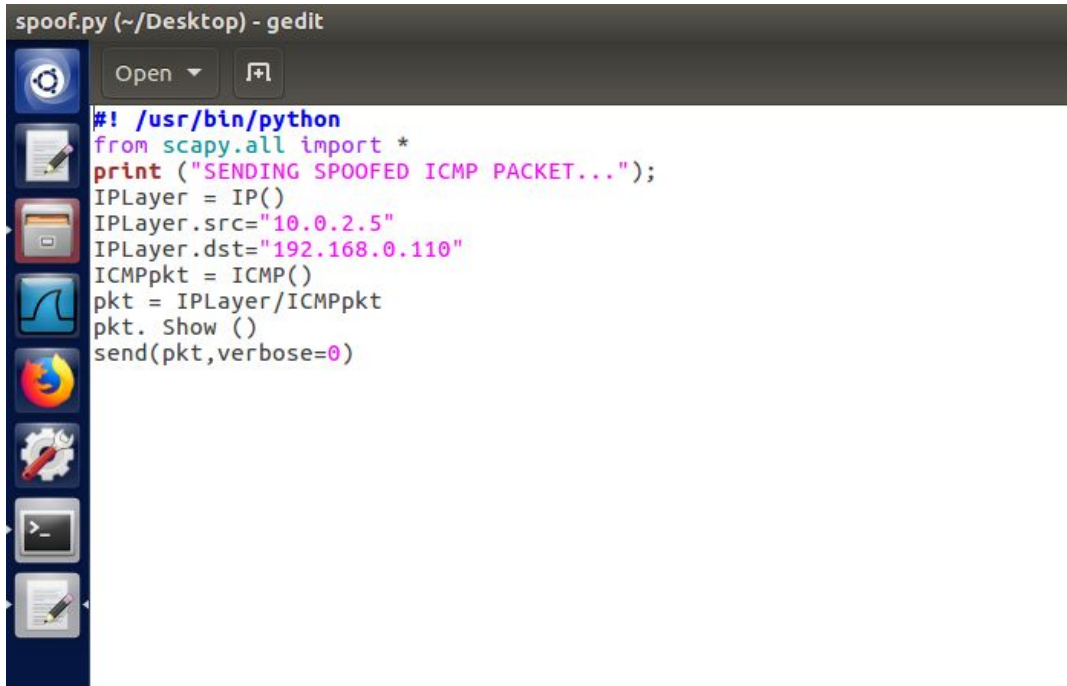
seed@pes1201800144:~$ ping -c 1 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data:
64 bytes from 192.168.0.1: icmp_seq=1 ttl=63 time=6.80 ms

--- 192.168.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 6.802/6.802/6.802/0.000 ms
seed@pes1201800144:~$
```

Figure: Capturing packets coming or going from a particular subnet



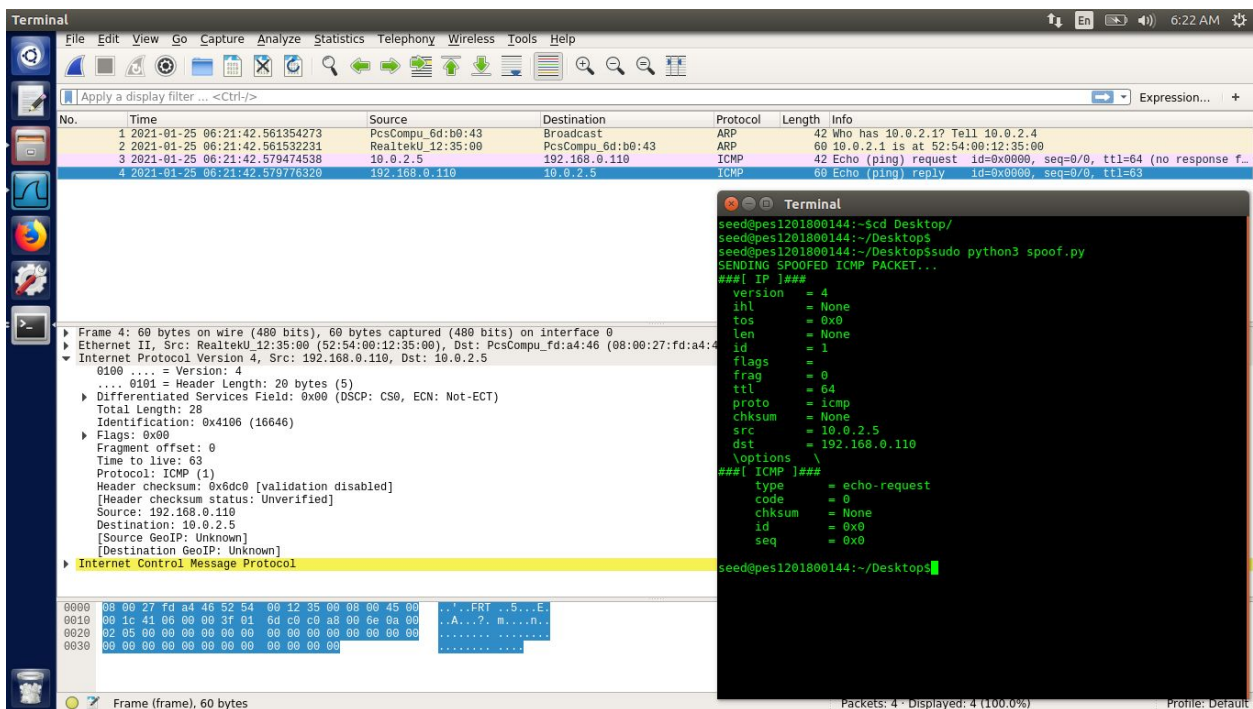
## TASK 2.1.3



```
spooof.py (~/Desktop) - gedit

#!/usr/bin/python
from scapy.all import *
print ("SENDING SPOOFED ICMP PACKET...");
IPLayer = IP()
IPLayer.src="10.0.2.5"
IPLayer.dst="192.168.0.110"
ICMPpkt = ICMP()
pkt = IPLayer/ICMPpkt
pkt.show ()
send(pkt,verbose=0)
```

Figure: Code to spoof ICMP packets



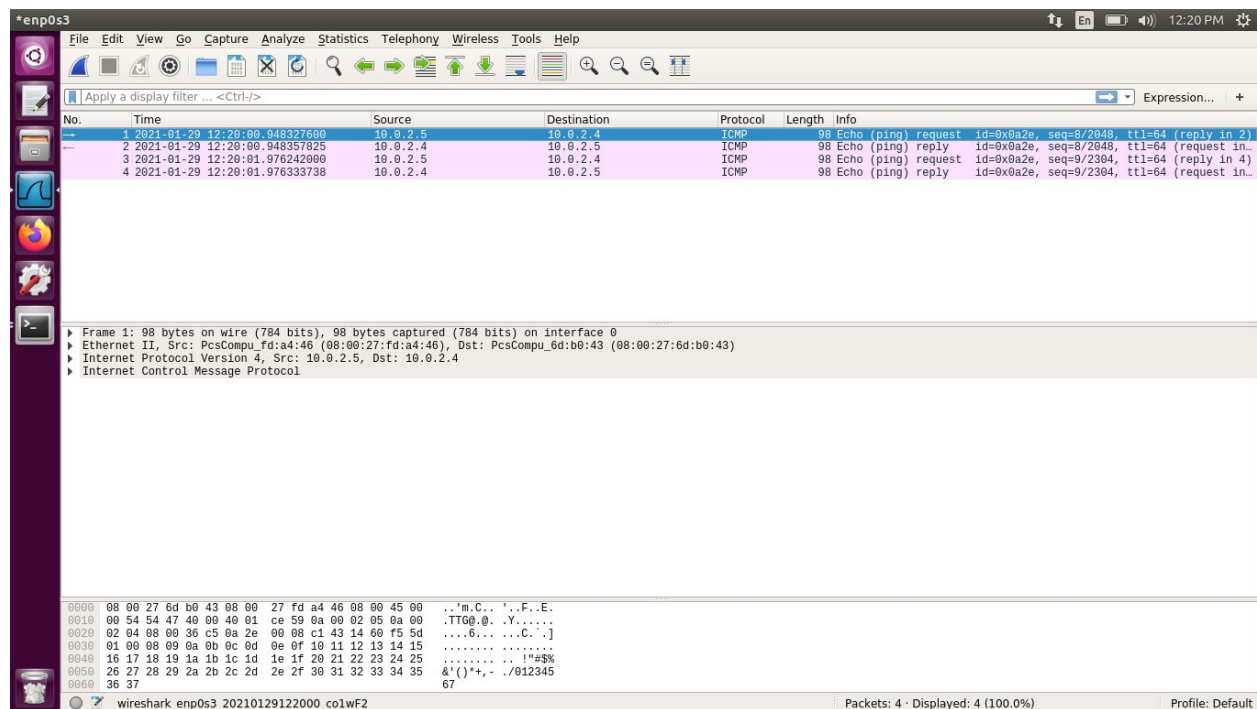
The screenshot shows a Wireshark network packet capture. The packet list on the left shows four packets. Packet 4 is selected, showing details for an Internet Control Message Protocol (ICMP) packet. The details pane shows the following information:

- Internet Protocol Version 4, Src: 192.168.0.110, Dst: 10.0.2.5
- 0100 .... = Version: 4
- .... 0101 = Header Length: 20 bytes (5)
- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 28
- Identification: 0x4106 (16646)
- Flags: 0x00
- Fragment offset: 0
- Time to live: 63
- Protocol: ICMP (1)
- Header checksum: 0x6dc0 [validation disabled]
- [Header checksum status: Unverified]
- Source: 192.168.0.110
- Destination: 10.0.2.5
- [Source GeoIP: Unknown]
- [Destination GeoIP: Unknown]
- Internet Control Message Protocol

The packet bytes pane shows the raw data of the packet, and the packet details pane shows the decoded structure of the packet. A terminal window in the background shows the execution of the script:

```
seed@pes1201800144:~/Desktop$ python3 spooof.py
SENDING SPOOFED ICMP PACKET...
###[ IP ]###
version = 4
ihl = None
tos = 0x0
len = None
id = 1
flags =
frag = 0
ttl = 64
proto = icmp
chksum = None
src = 10.0.2.5
dst = 192.168.0.110
Options \
###[ ICMP ]###
type = echo-request
code = 0
chksum = None
id = 0x0
seq = 0x0
seed@pes1201800144:~/Desktop$
```

Figure: spoofing ICMP packets



Pinging to attacker 10.0.2.4

## TASK 2.1.4

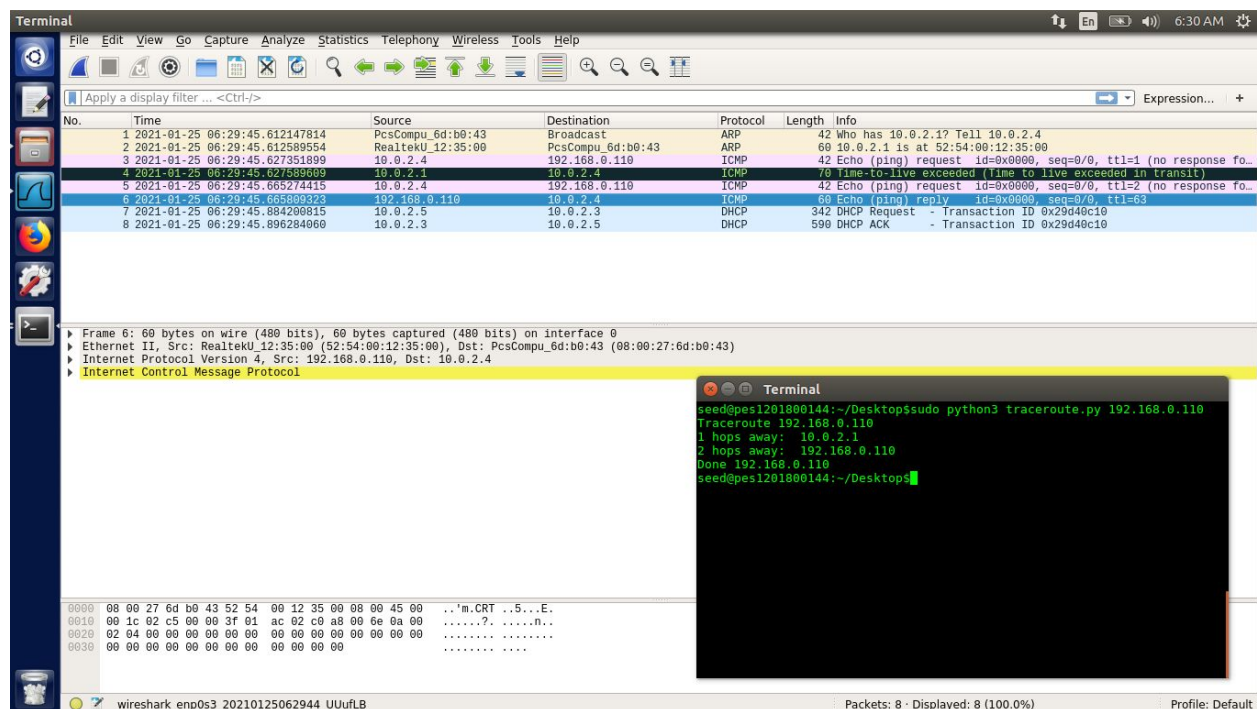


Figure: Capturing packets through wireshark and performing traceroute. TTL=1 and TTL=2 packets can be seen clearly. Also Time-To-Live exceeded response packet can also be seen clearly.

## TASK 2.1.5

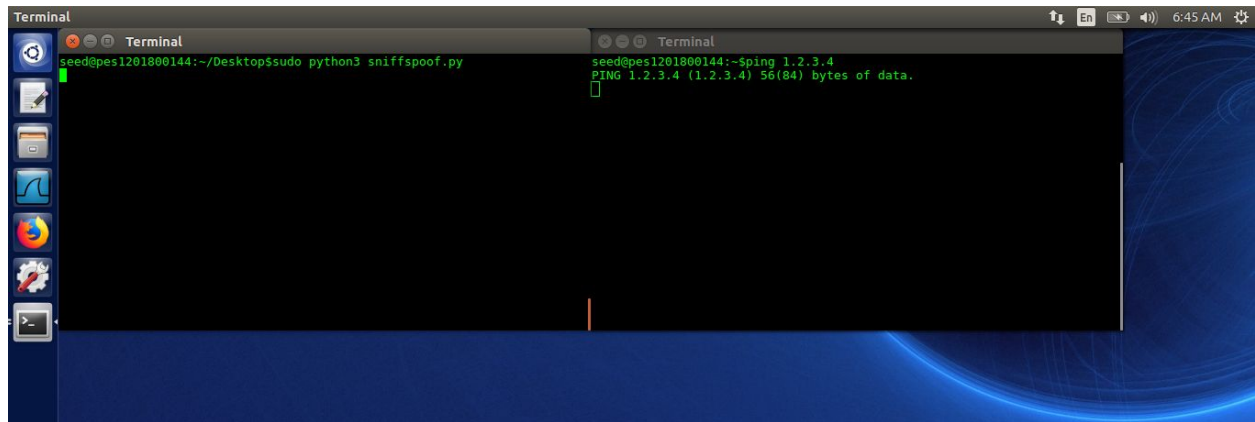


Figure: Spoofing an 1.2.3.4 from same machine(attacker machine)

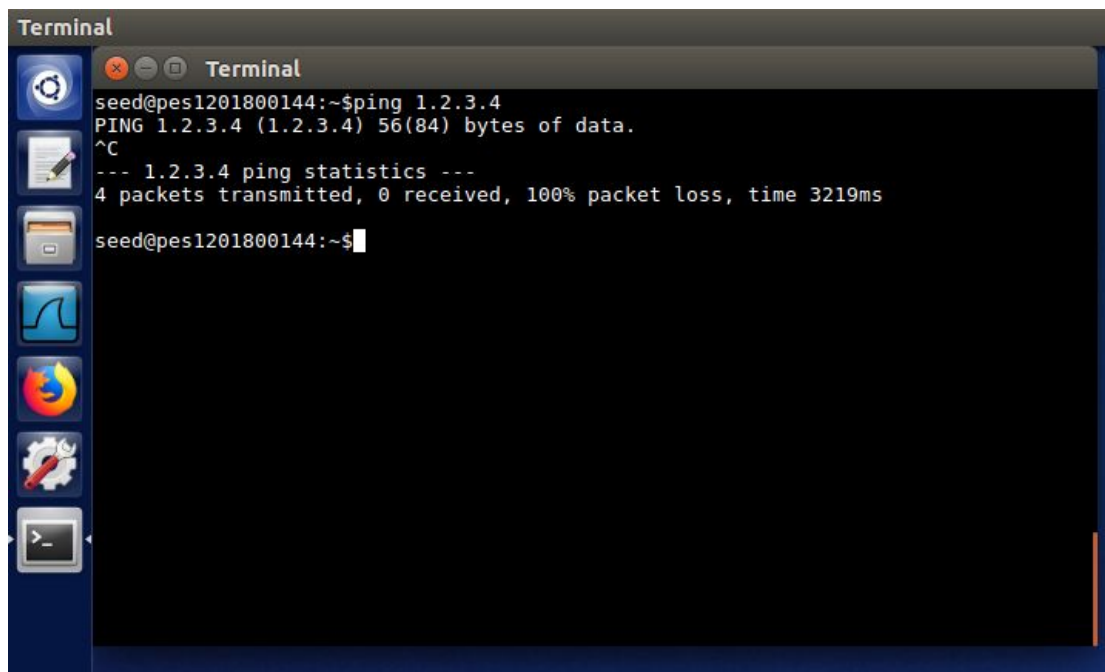
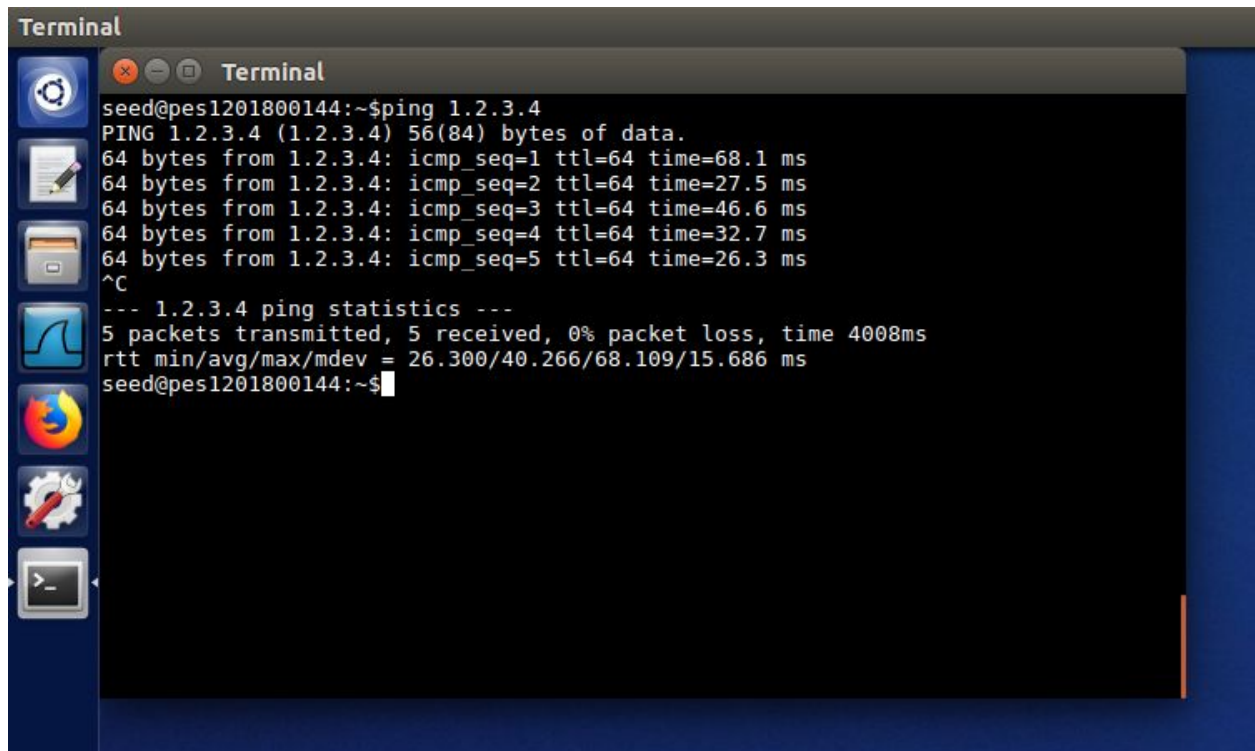


Figure: pinging 1.2.3.4 from victim machine without running spoofing code on attacker machine

A terminal window titled "Terminal" is shown. The prompt is "seed@pes1201800144:~\$". The user has entered "ping 1.2.3.4". The output shows five successful ping responses from 1.2.3.4, each with 64 bytes of data, an icmp\_seq number from 1 to 5, a TTL of 64, and various response times. After the responses, the user pressed Ctrl-C (^C). The terminal then displays "1.2.3.4 ping statistics ---" followed by the summary: "5 packets transmitted, 5 received, 0% packet loss, time 4008ms" and "rtt min/avg/max/mdev = 26.300/40.266/68.109/15.686 ms". The prompt is now "seed@pes1201800144:~\$".

```
Terminal
seed@pes1201800144:~$ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=68.1 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=27.5 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=46.6 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=32.7 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=26.3 ms
^C
--- 1.2.3.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 26.300/40.266/68.109/15.686 ms
seed@pes1201800144:~$
```

Figure: pinging 1.2.3.4 from victim machine while running spoofing code on attacker machine

```
Terminal
seed@pesl201800144:~/Desktop$ sudo python3 sniffsp
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
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
```

Figure: attacker machine terminal while victim pings to 1.2.3.4

**END**