CS 201P Lab Assignment #3
Sandhya Chandramohan UCI ID: 61167891

Packet Sniffing and Spoofing Lab

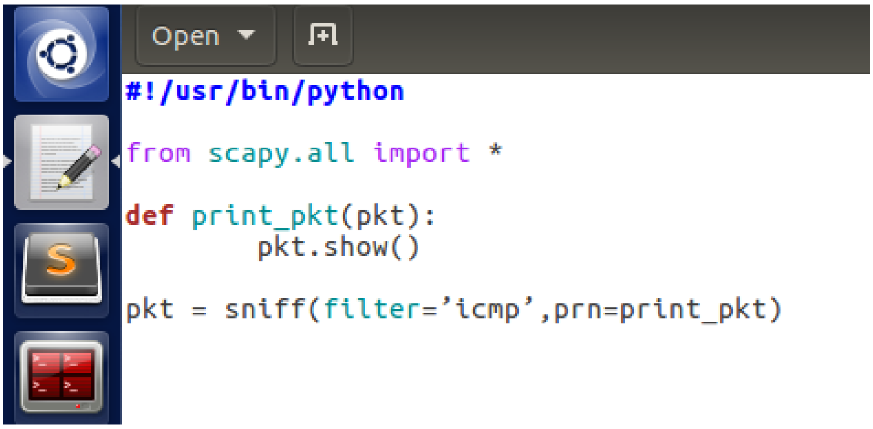
Lab Task Set 1: Using Tools to Sniff and Spoof Packets

Task 1.1: Sniffing Packets

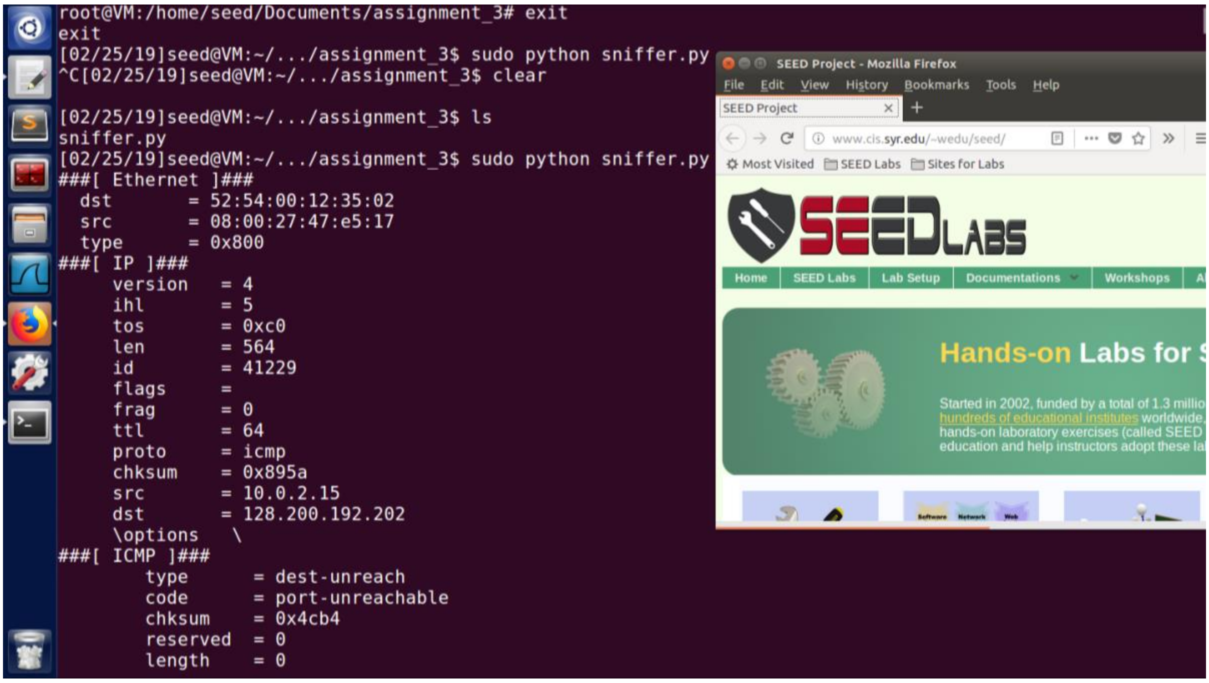
Task 1.1A Sniff Packets

Procedure:

i) The program shown below to sniff packets is run.
ii) First with root privilege and then without root privilege
iii) The difference in output in the two runs is noted



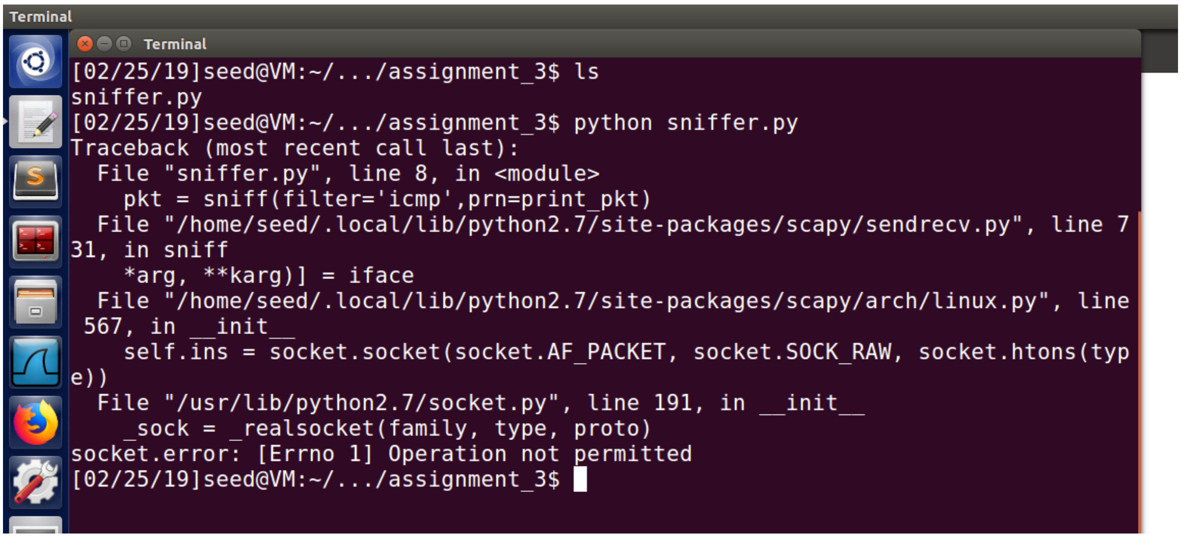
```
#!/usr/bin/python

from scapy.all import *

def print_pkt(pkt):
        pkt.show()

pkt = sniff(filter='icmp',prn=print_pkt)
```



```
root@VM:/home/seed/Documents/assignment_3# exit
exit
[02/25/19]seed@VM:~/.../assignment_3$ sudo python sniffer.py
^C[02/25/19]seed@VM:~/.../assignment_3$ clear

[02/25/19]seed@VM:~/.../assignment_3$ ls
sniffer.py
[02/25/19]seed@VM:~/.../assignment_3$ sudo python sniffer.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:47:e5:17
  type      = 0x800
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0xc0
     len       = 564
     id        = 41229
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x895a
     src       = 10.0.2.15
     dst       = 128.200.192.202
     \options   \
###[ ICMP ]###
        type      = dest-unreach
        code      = port-unreachable
        chksum    = 0x4cb4
        reserved  = 0
        length    = 0
```

SEED Project - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SEED Project
www.cis.syr.edu/~wedu/seed/
Most Visited SEED Labs Sites for Labs
SEEDLABS
Home SEED Labs Lab Setup Documentations Workshops
Hands-on Labs for

Please Turn Over…



```
[02/25/19]seed@VM:~/.../assignment_3$ ls
sniffer.py
[02/25/19]seed@VM:~/.../assignment_3$ python sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 8, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/home/seed/.local/lib/python2.7/site-packages/scapy/sendrecv.py", line 7
31, in sniff
    *arg, **karg)] = iface
  File "/home/seed/.local/lib/python2.7/site-packages/scapy/arch/linux.py", line
 567, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(typ
e))
  File "/usr/lib/python2.7/socket.py", line 191, in __init__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] Operation not permitted
[02/25/19]seed@VM:~/.../assignment_3$ 
```

Observations and Explanation:

i. When the program is run with root privilege and a webpage is requested via a browser, then we observe that the packets are captured successfully

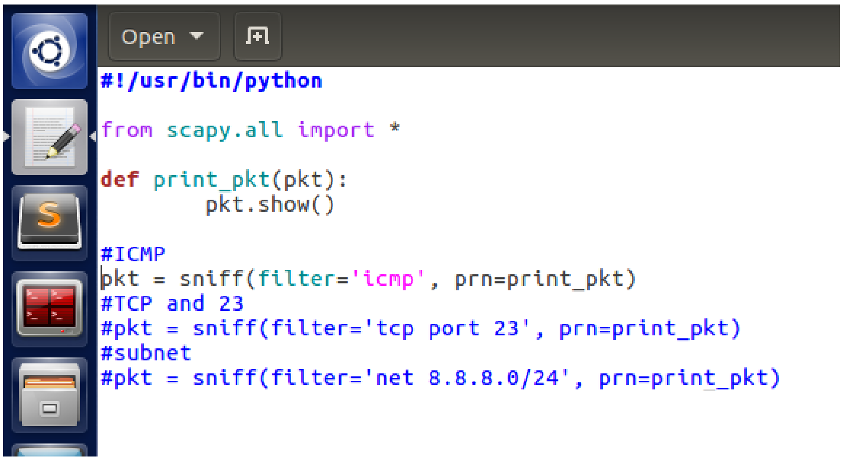
ii. When the program is run without root privilege then, an error is thrown. From the traceback we can see that the sniff operation in line 8 requires the use of the socket.py python2.7 library which requires root privilege for execution.

Please Turn Over…

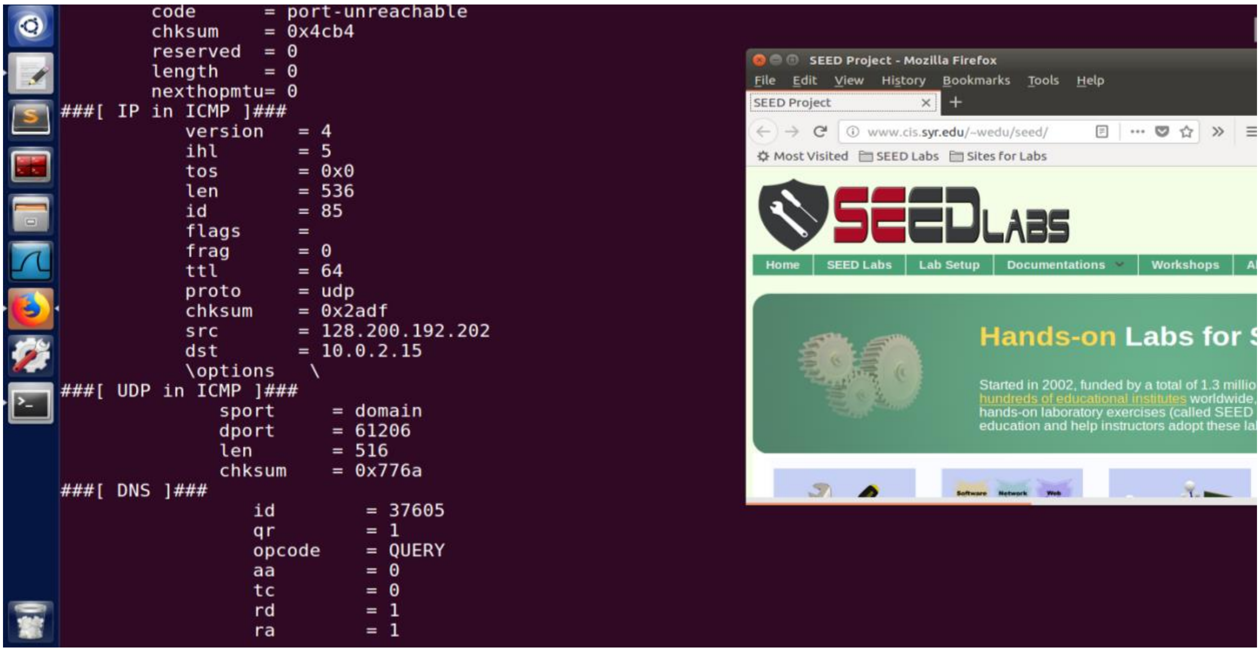
Task 1.1B Sniff Packets with Filters

Procedure:

i. Filters are added to the sniffing to sniff only certain kinds of packets using the BPF (Berkely packet filter) syntax as shown below in the program:
   a. Capture only the ICMP packet



```
#!/usr/bin/python

from scapy.all import *

def print_pkt(pkt):
        pkt.show()

#ICMP
pkt = sniff(filter='icmp', prn=print_pkt)
#TCP and 23
#pkt = sniff(filter='tcp port 23', prn=print_pkt)
#subnet
#pkt = sniff(filter='net 8.8.8.0/24', prn=print_pkt)
```



```
     code      = port-unreachable
     chksum    = 0x4cb4
     reserved  = 0
     length    = 0
     nexthopmtu= 0
###[ IP in ICMP ]###
        version   = 4
        ihl       = 5
        tos       = 0x0
        len       = 536
        id        = 85
        flags     =
        frag      = 0
        ttl       = 64
        proto     = udp
        chksum    = 0x2adf
        src       = 128.200.192.202
        dst       = 10.0.2.15
        \options   \
###[ UDP in ICMP ]###
           sport     = domain
           dport     = 61206
           len       = 516
           chksum    = 0x776a
###[ DNS ]###
              id        = 37605
              qr        = 1
              opcode    = QUERY
              aa        = 0
              tc        = 0
              rd        = 1
              ra        = 1
```

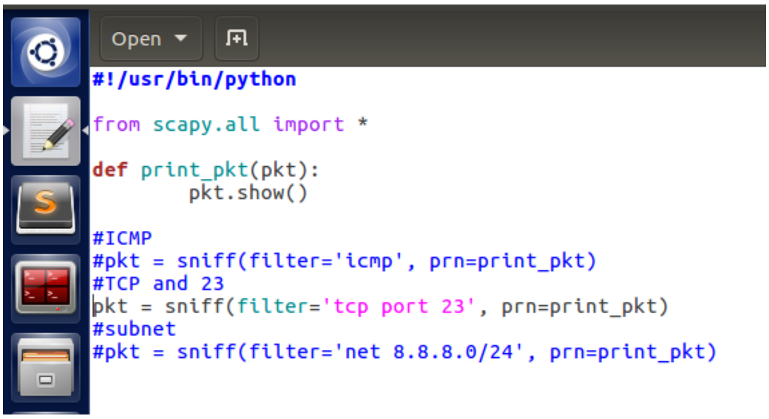
Hands-on Labs for S

Started in 2002, funded by a total of 1.3 millio
hundreds of educational institutes worldwide,
hands-on laboratory exercises (called SEED
education and help instructors adopt these la

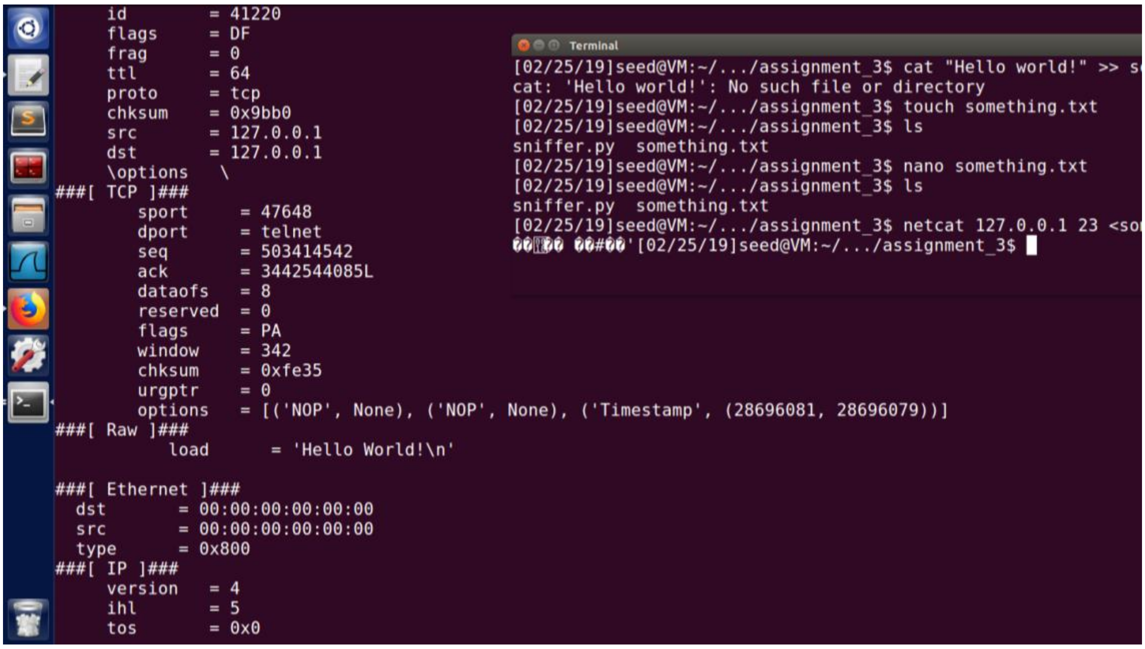
Observations and Explanation:
We note that when the ICMP only filter is used, pinging any web page captures packets.

Please Turn Over…

b. Capture any TCP packet that comes from a particular IP and with a destination port number 23.
A file something.txt with the text “Hello World!” is sent with the following command: *netcat 127.0.0.1 23 < something.txt*



```
#!/usr/bin/python

from scapy.all import *

def print_pkt(pkt):
        pkt.show()

#ICMP
#pkt = sniff(filter='icmp', prn=print_pkt)
#TCP and 23
pkt = sniff(filter='tcp port 23', prn=print_pkt)
#subnet
#pkt = sniff(filter='net 8.8.8.0/24', prn=print_pkt)
```



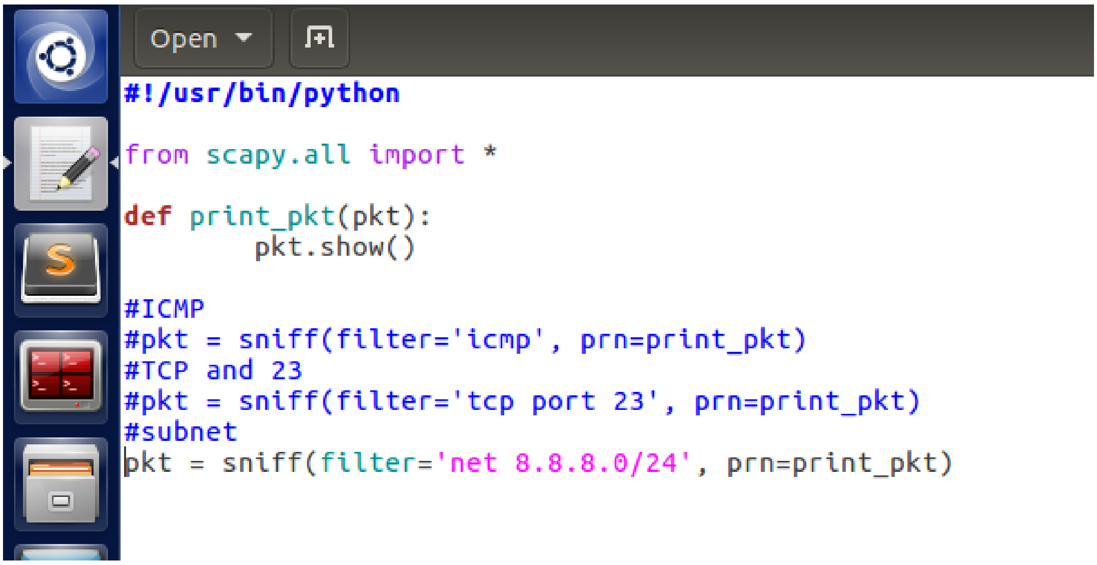
```
     id        = 41220
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x9bb0
     src       = 127.0.0.1
     dst       = 127.0.0.1
     \options   \
###[ TCP ]###
        sport     = 47648
        dport     = telnet
        seq       = 503414542
        ack       = 3442544085L
        dataofs   = 8
        reserved  = 0
        flags     = PA
        window    = 342
        chksum    = 0xfe35
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (28696081, 28696079))]
###[ Raw ]###
           load      = 'Hello World!\n'

###[ Ethernet ]###
  dst       = 00:00:00:00:00:00
  src       = 00:00:00:00:00:00
  type      = 0x800
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
```

```
[02/25/19]seed@VM:~/.../assignment_3$ cat "Hello world!" >> s
cat: 'Hello world!': No such file or directory
[02/25/19]seed@VM:~/.../assignment_3$ touch something.txt
[02/25/19]seed@VM:~/.../assignment_3$ ls
sniffer.py  something.txt
[02/25/19]seed@VM:~/.../assignment_3$ nano something.txt
[02/25/19]seed@VM:~/.../assignment_3$ ls
sniffer.py  something.txt
[02/25/19]seed@VM:~/.../assignment_3$ netcat 127.0.0.1 23 <so
����� ��#��'[02/25/19]seed@VM:~/.../assignment_3$ █
```

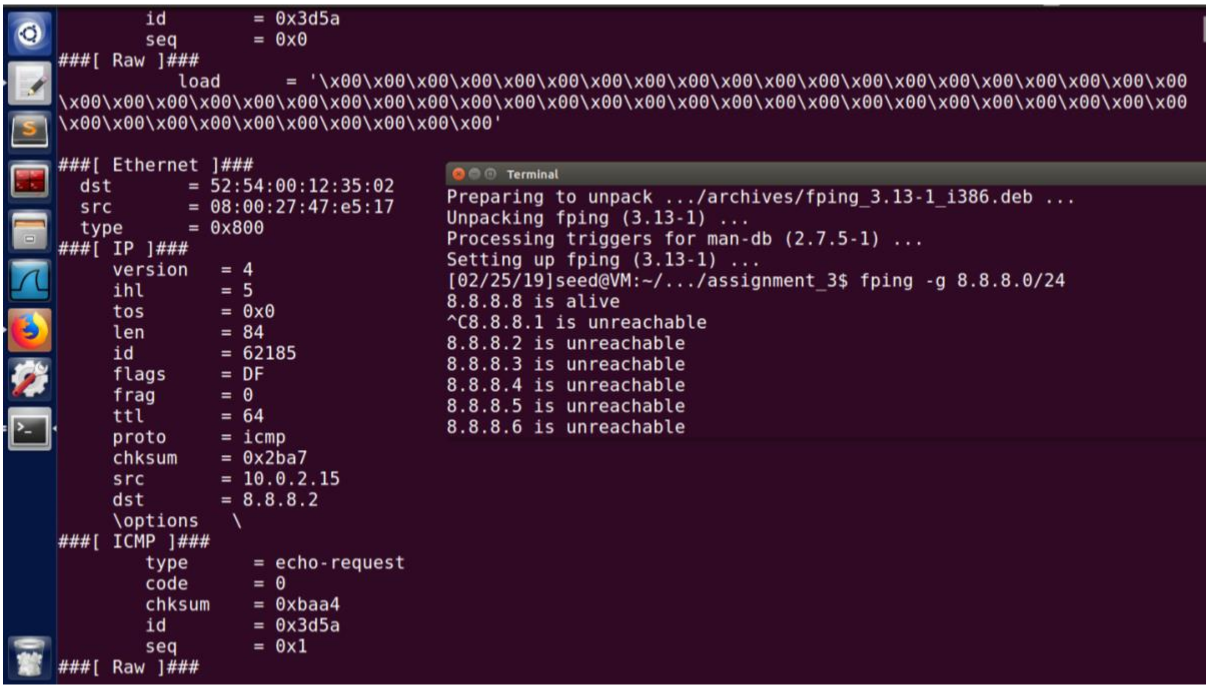
Observations and Explanation:
When the filter for capturing a packet from port 23 is used, no packets are captured. Only when the command *netcat 127.0.0.1 23 < something.txt* is used to send packets from port 23, then the packet is captured. This can be seen from the Raw payload “Hello World!” sent in the packet that is captured.

Please Turn Over…

c. 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.
The subnet 8.8.8.0/24 is used in the filter as shown in the program below.
The following command is used: *fping –g 8.8.8.0/24*



```
#!/usr/bin/python

from scapy.all import *

def print_pkt(pkt):
        pkt.show()

#ICMP
#pkt = sniff(filter='icmp', prn=print_pkt)
#TCP and 23
#pkt = sniff(filter='tcp port 23', prn=print_pkt)
#subnet
pkt = sniff(filter='net 8.8.8.0/24', prn=print_pkt)
```



```
     id        = 0x3d5a
     seq       = 0x0
###[ Raw ]###
        load      = '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'

###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:47:e5:17
  type      = 0x800
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 62185
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x2ba7
     src       = 10.0.2.15
     dst       = 8.8.8.2
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xbaa4
        id        = 0x3d5a
        seq       = 0x1
###[ Raw ]###
```

```
Preparing to unpack .../archives/fping_3.13-1_i386.deb ...
Unpacking fping (3.13-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up fping (3.13-1) ...
[02/25/19]seed@VM:~/.../assignment_3$ fping -g 8.8.8.0/24
8.8.8.8 is alive
^C8.8.8.1 is unreachable
8.8.8.2 is unreachable
8.8.8.3 is unreachable
8.8.8.4 is unreachable
8.8.8.5 is unreachable
8.8.8.6 is unreachable
```

Observations and Explanation:
When the filter for capturing packets to or from a particular subnet is used, then again, no packets are captured. Only when the command *fping –g 8.8.8.0/24* is used to send packets to IPs in the subnet 8.8.8.0/24 then the packets are captured. As can be seen in the destination IP: 8.8.8.2 in the screenshot shown in the image.

Please Turn Over…

Task 1.2 Spoofing ICMP Packets

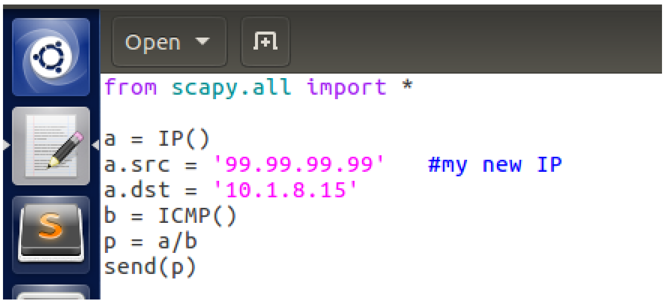
Procedure:

i. We set the source IP address of the packet to some arbitrary value such as 99.99.99.99 using scapy.

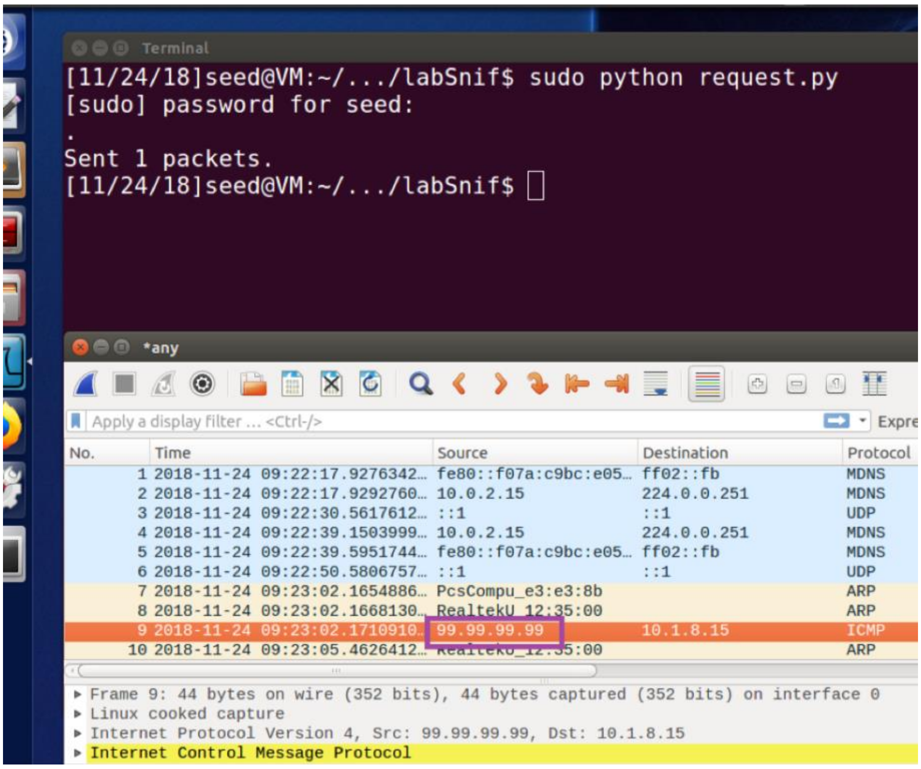
ii. Then, the ICMP echo request packets are spoofed and sent to another VM on the network.

iii. Wireshark is then used to observe whether the request is accepted by the receiver.

iv. If the request is accepted then an echo reply packet will be sent to the spoofed IP address which is observed.



```
from scapy.all import *

a = IP()
a.src = '99.99.99.99'    #my new IP
a.dst = '10.1.8.15'
b = ICMP()
p = a/b
send(p)
```



```
[11/24/18]seed@VM:~/.../labSnif$ sudo python request.py
[sudo] password for seed:
.
Sent 1 packets.
[11/24/18]seed@VM:~/.../labSnif$ 
```

```
No.  Time                           Source                    Destination   Protocol
   1 2018-11-24 09:22:17.9276342… fe80::f07a:c9bc:e05…  ff02::fb      MDNS
   2 2018-11-24 09:22:17.9292760… 10.0.2.15              224.0.0.251   MDNS
   3 2018-11-24 09:22:30.5617612… ::1                    ::1           UDP
   4 2018-11-24 09:22:39.1503999… 10.0.2.15              224.0.0.251   MDNS
   5 2018-11-24 09:22:39.5951744… fe80::f07a:c9bc:e05…  ff02::fb      MDNS
   6 2018-11-24 09:22:50.5806757… ::1                    ::1           UDP
   7 2018-11-24 09:23:02.1654886… PcsCompu_e3:e3:8b                    ARP
   8 2018-11-24 09:23:02.1668130… RealtekU_12:35:00                    ARP
   9 2018-11-24 09:23:02.1710910… 99.99.99.99            10.1.8.15     ICMP
  10 2018-11-24 09:23:05.4626412… RealtekU_12:35:00                    ARP

▸ Frame 9: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 99.99.99.99, Dst: 10.1.8.15
▸ Internet Control Message Protocol
```

Observations and Explanation:

i. The ICMP echo request packet is spoofed with arbitrary source IP 99.99.99.99 and sent.

ii. We observe that the packet is captured in wireshark, by noting the source and destination IP addresses and protocal when the program is run as root.

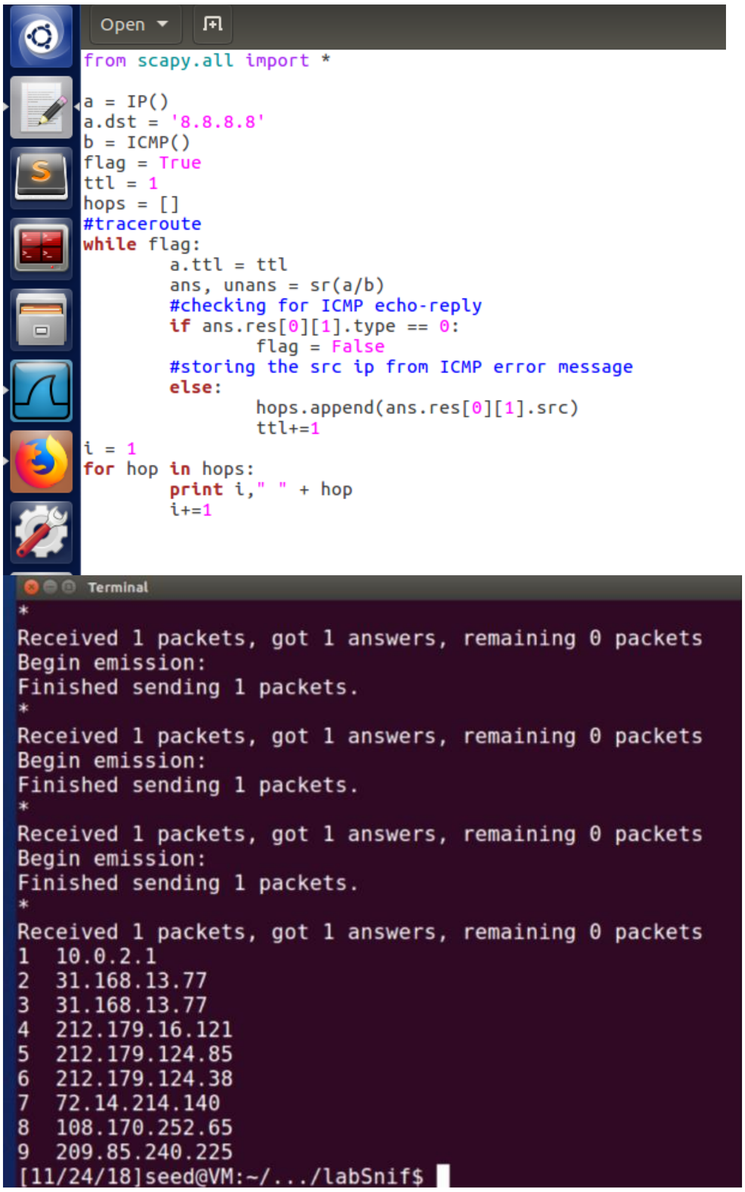
Please Turn Over…

Task 1.3 Traceroute

Procedure:

i. To estimate the distance between the VM and some destination, in terms of the no. of routers between them, the program shown below is used.

ii. A packet with TTL with progressively increasing TTLs starting from 1 is sent to the destination, which fails with an error message that the TTL was exceeded and gives the IP addresses of each successive router. Until the packet finally reaches the destination and returns a success code.



```
from scapy.all import *

a = IP()
a.dst = '8.8.8.8'
b = ICMP()
flag = True
ttl = 1
hops = []
#traceroute
while flag:
        a.ttl = ttl
        ans, unans = sr(a/b)
        #checking for ICMP echo-reply
        if ans.res[0][1].type == 0:
                flag = False
        #storing the src ip from ICMP error message
        else:
                hops.append(ans.res[0][1].src)
                ttl+=1
i = 1
for hop in hops:
        print i," " + hop
        i+=1
```

```
*
Received 1 packets, got 1 answers, remaining 0 packets
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
1  10.0.2.1
2  31.168.13.77
3  31.168.13.77
4  212.179.16.121
5  212.179.124.85
6  212.179.124.38
7  72.14.214.140
8  108.170.252.65
9  209.85.240.225
[11/24/18]seed@VM:~/.../labSnif$
```

Observations and Explanation:

i. We observe that packets are sent in a loop continuously while it keeps failing with an error message due to the TTL being exceeded, giving us the router IP which is stored in an array and outputted at the end

ii. We get the IPs of 9 routers listed at the end on the way to the destination IP 8.8.8.8. The loop breaks when the TTL is big enough for the packet to reach the destination successfully and list the router IPs.

Please Turn Over…

Task 1.4 Sniffing and-then Spoofing

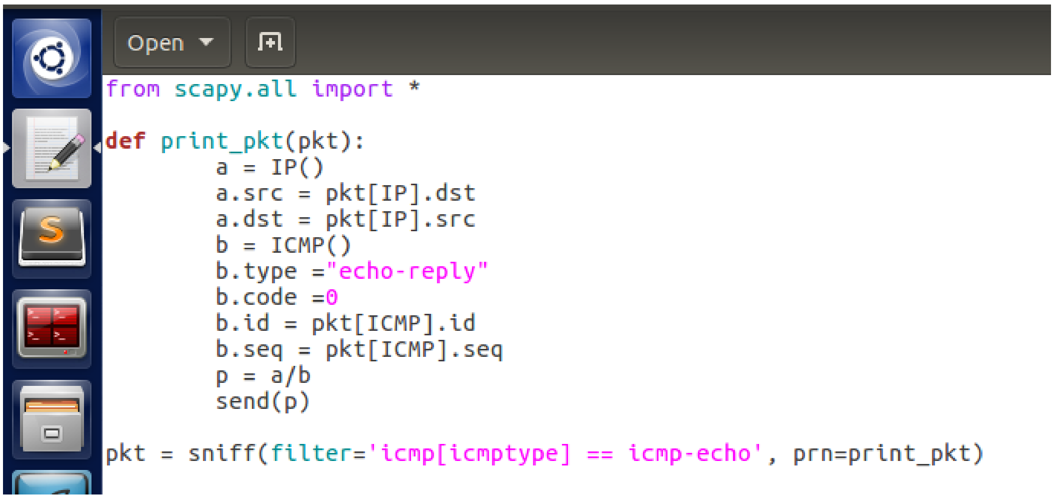
Procedure:

i. We setup two VMs. From the first VM A we ping an IP X, this will generate an ICMP echo request packet. The ping program prints out the response if X is alive and it receives an echo reply.

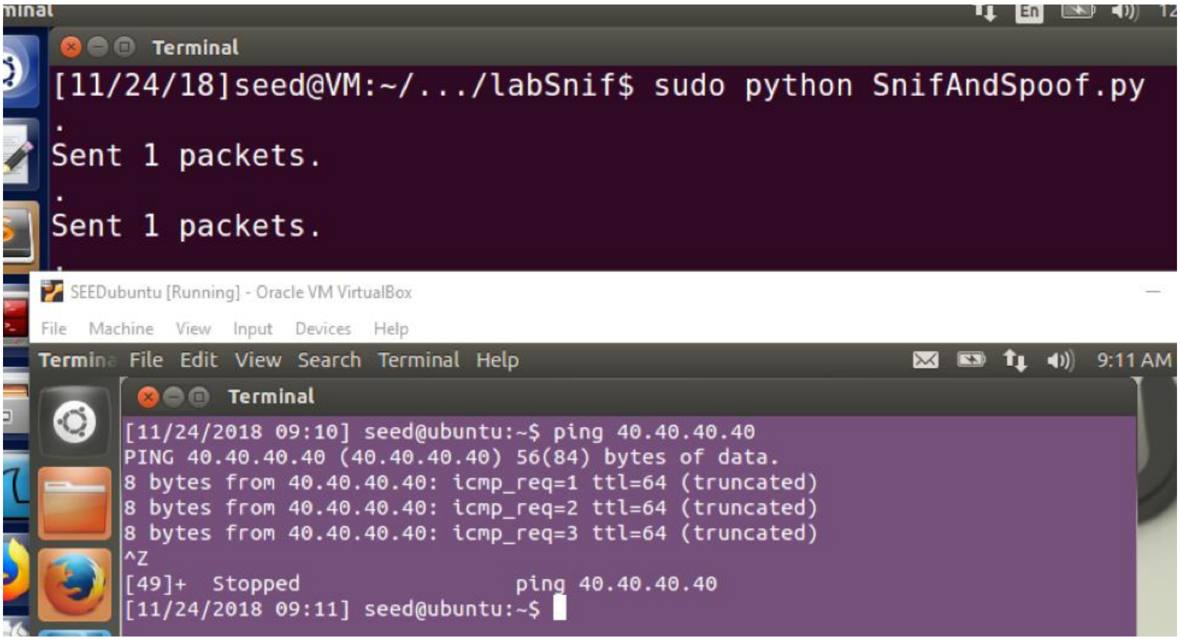
ii. The program shown below runs on the other VM B that is monitoring the LAN via packet sniffing.

iii. Whenever the program notices an ICMP echo request packet, irrespective of the target IP, it sends an echo reply via the packet spoofing technique.

iv. The observations are then noted.



```python
from scapy.all import *

def print_pkt(pkt):
        a = IP()
        a.src = pkt[IP].dst
        a.dst = pkt[IP].src
        b = ICMP()
        b.type ="echo-reply"
        b.code =0
        b.id = pkt[ICMP].id
        b.seq = pkt[ICMP].seq
        p = a/b
        send(p)

pkt = sniff(filter='icmp[icmptype] == icmp-echo', prn=print_pkt)
```


Terminal
SEEDubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal File Edit View Search Terminal Help
9:11 AM
Terminal

```
[11/24/18]seed@VM:~/.../labSnif$ sudo python SnifAndSpoof.py
.
Sent 1 packets.
.
Sent 1 packets.
```

```
[11/24/2018 09:10] seed@ubuntu:~$ ping 40.40.40.40
PING 40.40.40.40 (40.40.40.40) 56(84) bytes of data.
8 bytes from 40.40.40.40: icmp_req=1 ttl=64 (truncated)
8 bytes from 40.40.40.40: icmp_req=2 ttl=64 (truncated)
8 bytes from 40.40.40.40: icmp_req=3 ttl=64 (truncated)
^Z
[49]+  Stopped                 ping 40.40.40.40
[11/24/2018 09:11] seed@ubuntu:~$ 
```

Please Turn Over…

Observations and Explanation:

i. We observe that when a ping is made to a host on the network from VM A, 40.40.40.40, the ICMP request is sniffed by the attacker (on VM B) who spoofs the reply back to the source. And thus, the user (on VM A) ends up receiving the ICMP reply from the attacker (on VM B).

ii. When the ping is made from VM A by user, the attacker on VM B receives the ICMP packet via pcap packet capture that listens to traffic (in the promiscuous mode). The attacker then spoofs an ICMP reply via raw socket by swapping the source IP as the destination and the destination IP as the source. Furthermore, the other fields in the IP header and ICMP header are also spoofed by the attacker. Thus, to the user on VM A, it appears that he/she received a normal reply from the host the original packet was sent to.

Please Turn Over…