Intrusion Detection
Group 14
Sniffing-Spoofing
Vlad Smirnov, Raim Aliyev, Mark Musseau

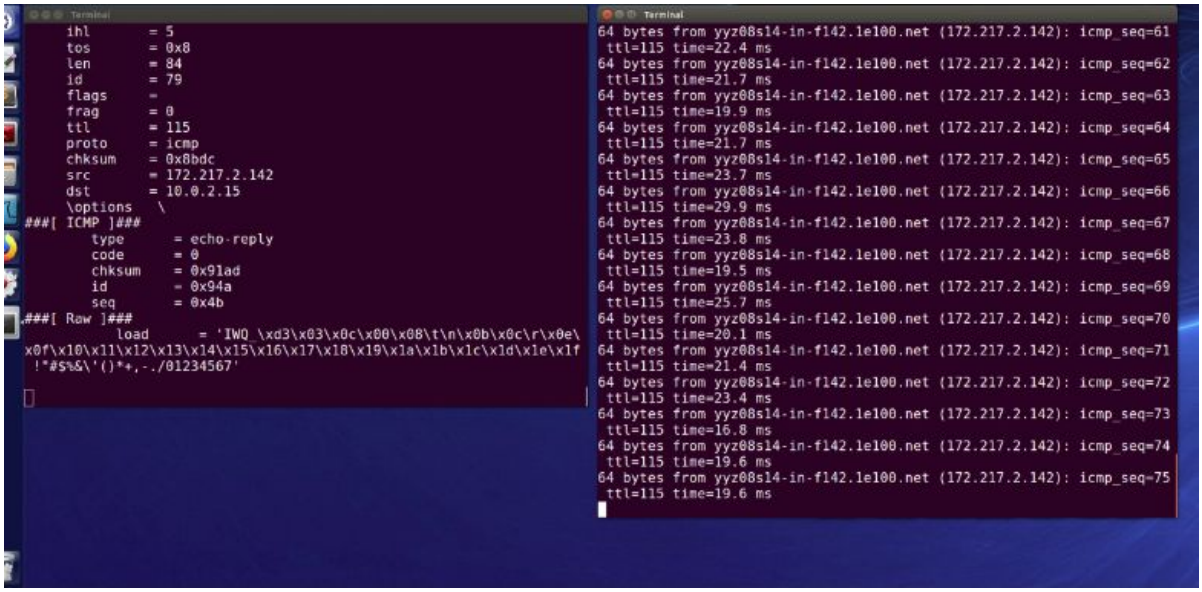
Task 1.1A

Using scapy the task was to make a sniffer that can sniff packets and then we ran it and pinged google and started receiving packets from google.

Task 1.1B

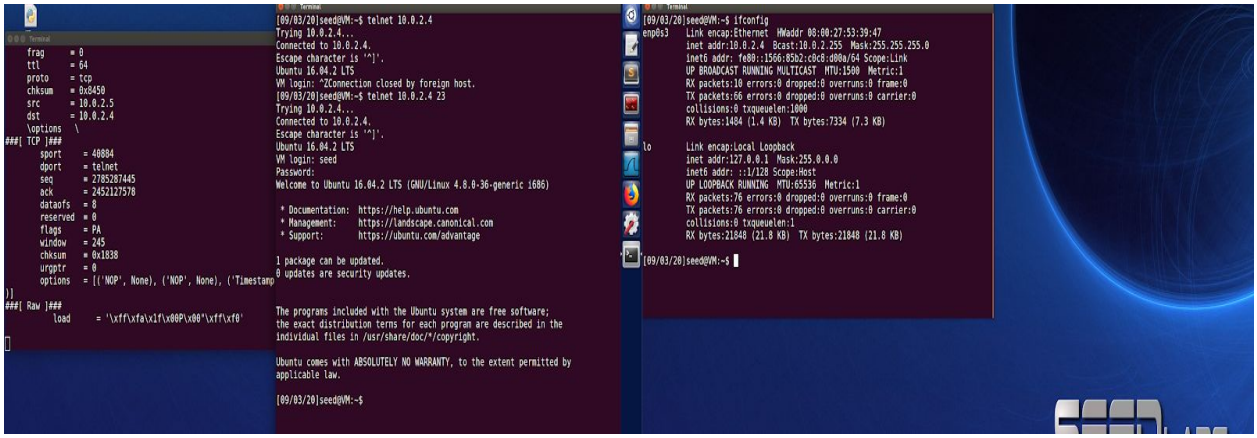
By using scapy, we captured ICMP packets from google by pinging google.com, after running the python script.

```
#!/usr/bin/python3

from scapy.all import *

#a = IP()
#a.show()

def print_pkt(pkt):
        pkt.show()

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

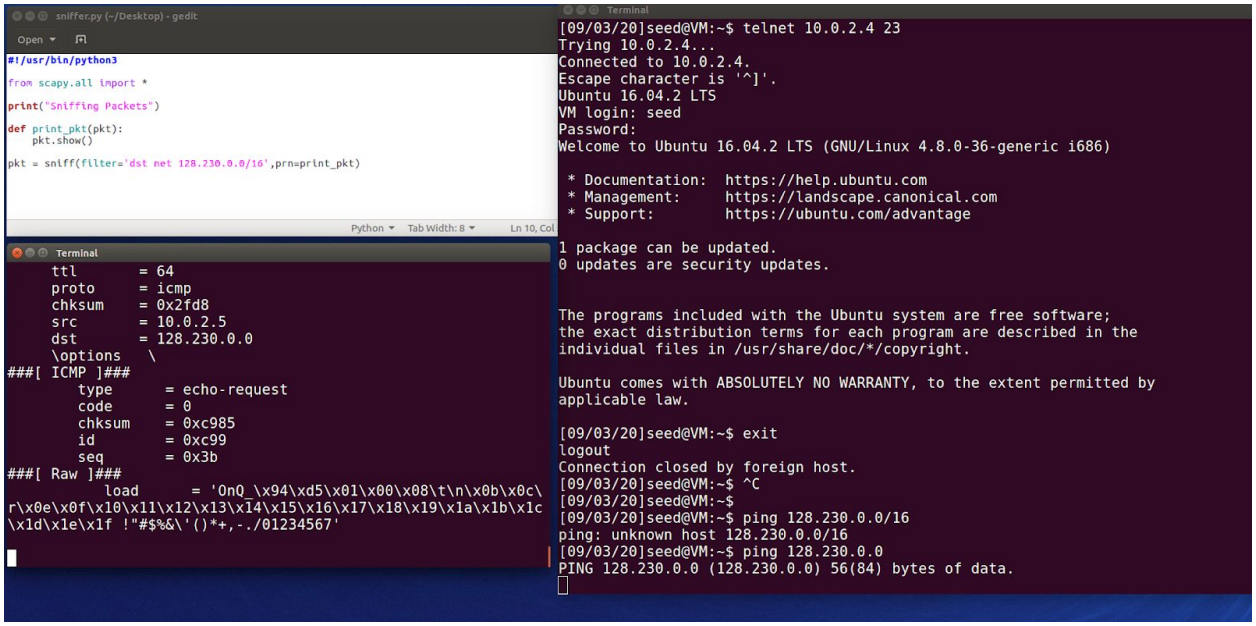


We captured TCP packets by using the telnet command to connect to another VM.

```
#!/usr/bin/python3

from scapy.all import *

#a = IP()
#a.show()

def print_pkt(pkt):
        pkt.show()

pkt = sniff(filter='tcp and host 10.0.2.4 and dst port 23',prn=print_pkt)
```

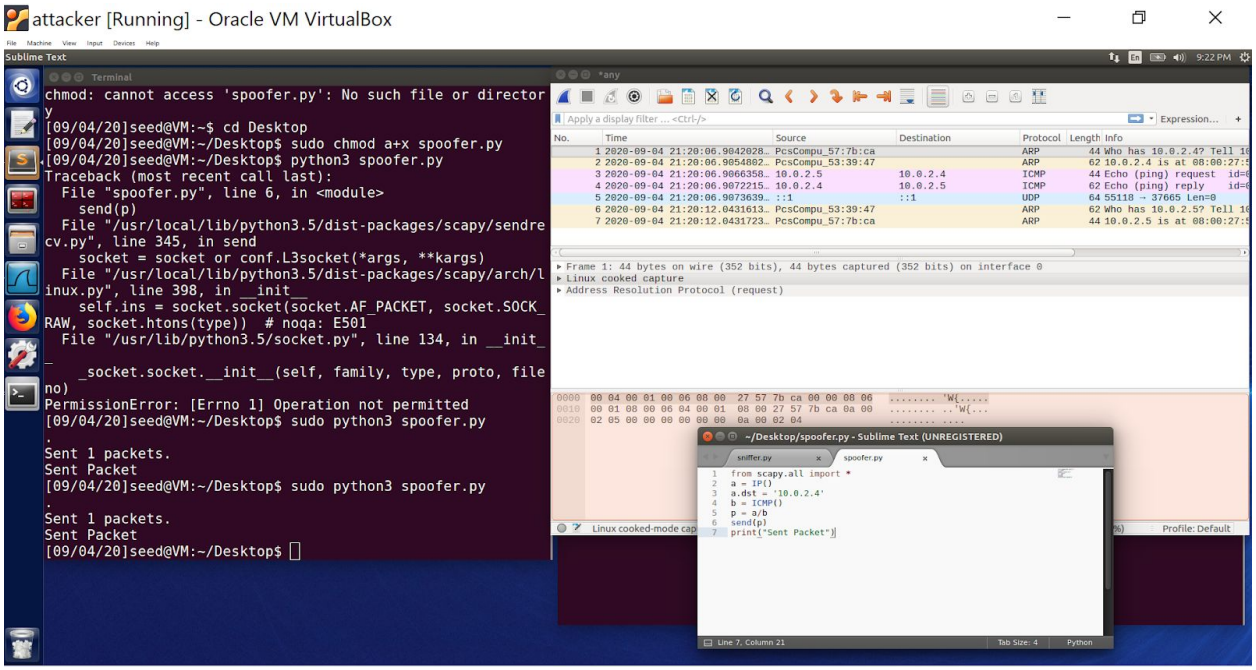


We captured packets from a particular subnet the machine is not attached to we captured from 128.230.0.0/16.

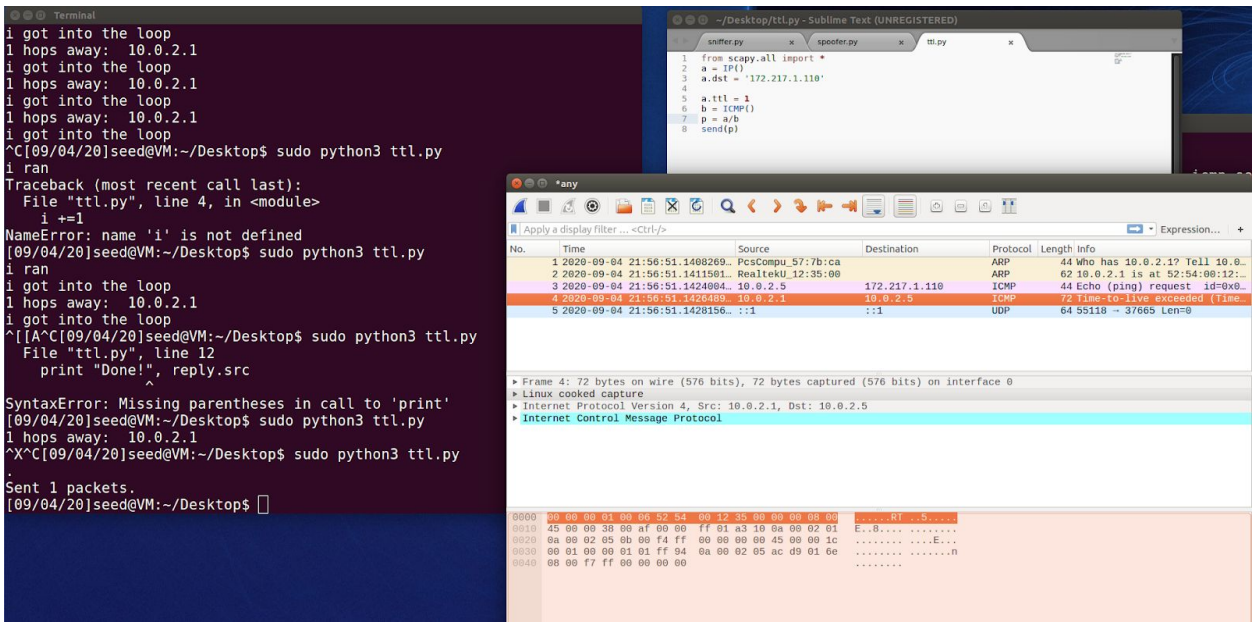


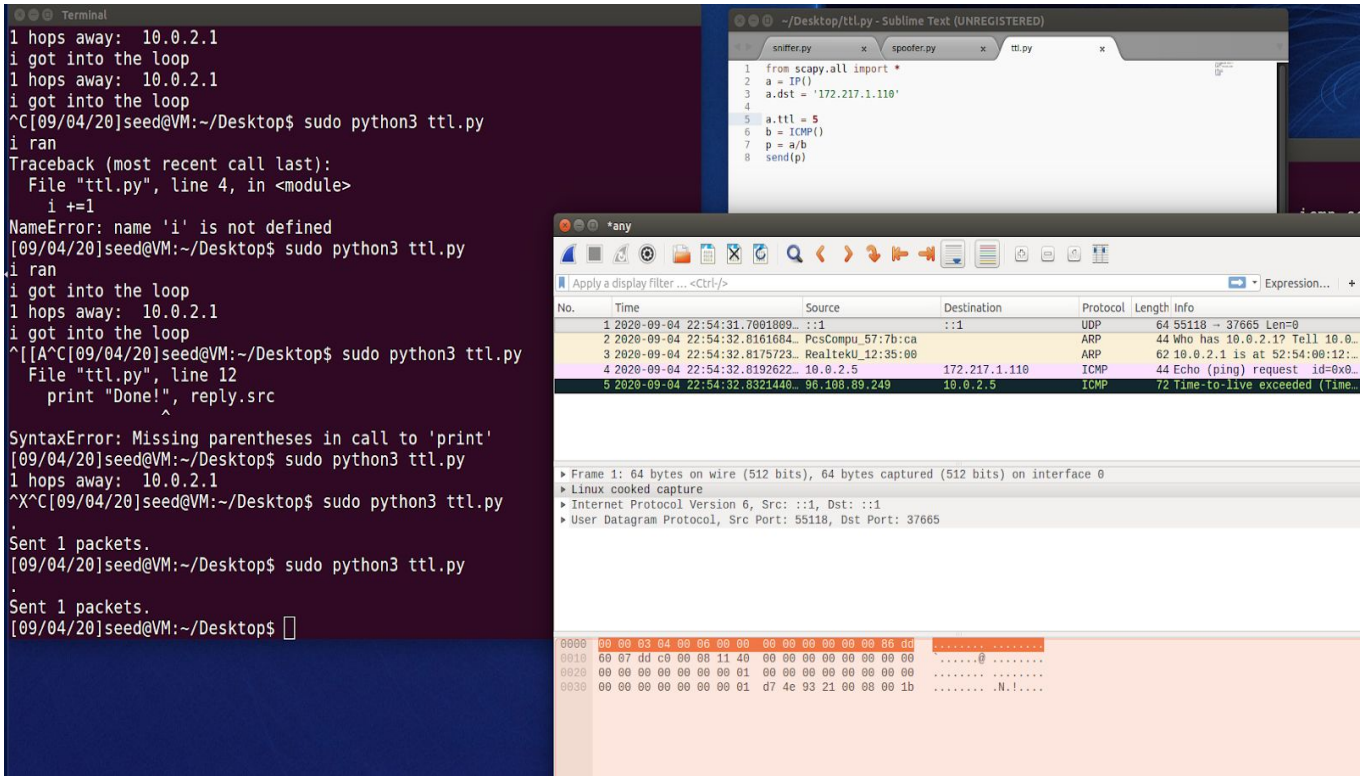
Task 1.2

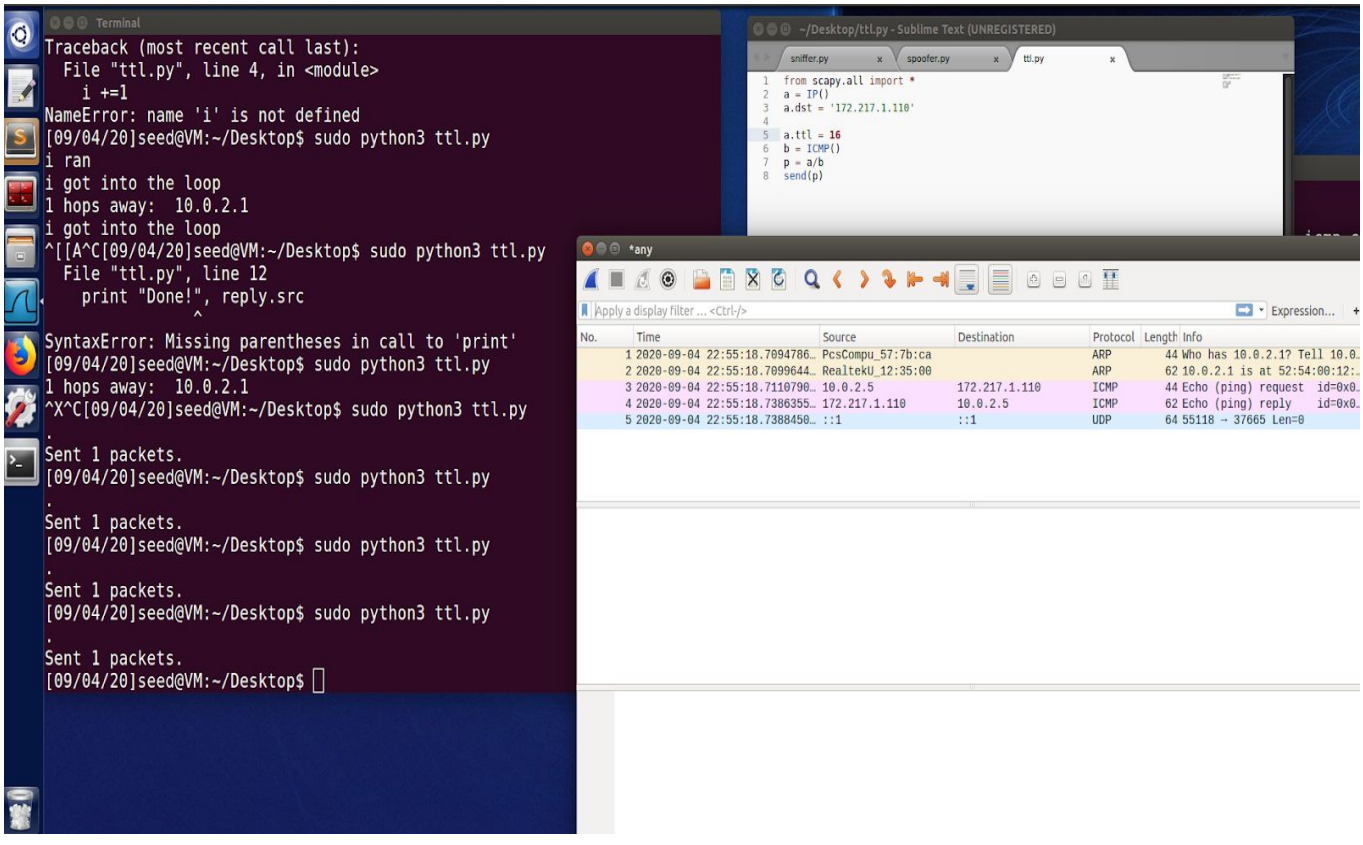
We sent an ICMP packet using scapy to the IP of another VM on our network. Using Wireshark, we can see that the other VM received the request and sent back a reply.



Task 1.3: We found the Time-To-Live from our machine to Google by starting from 1 and incrementing the TTL until we received a response back from Google, which was 16.



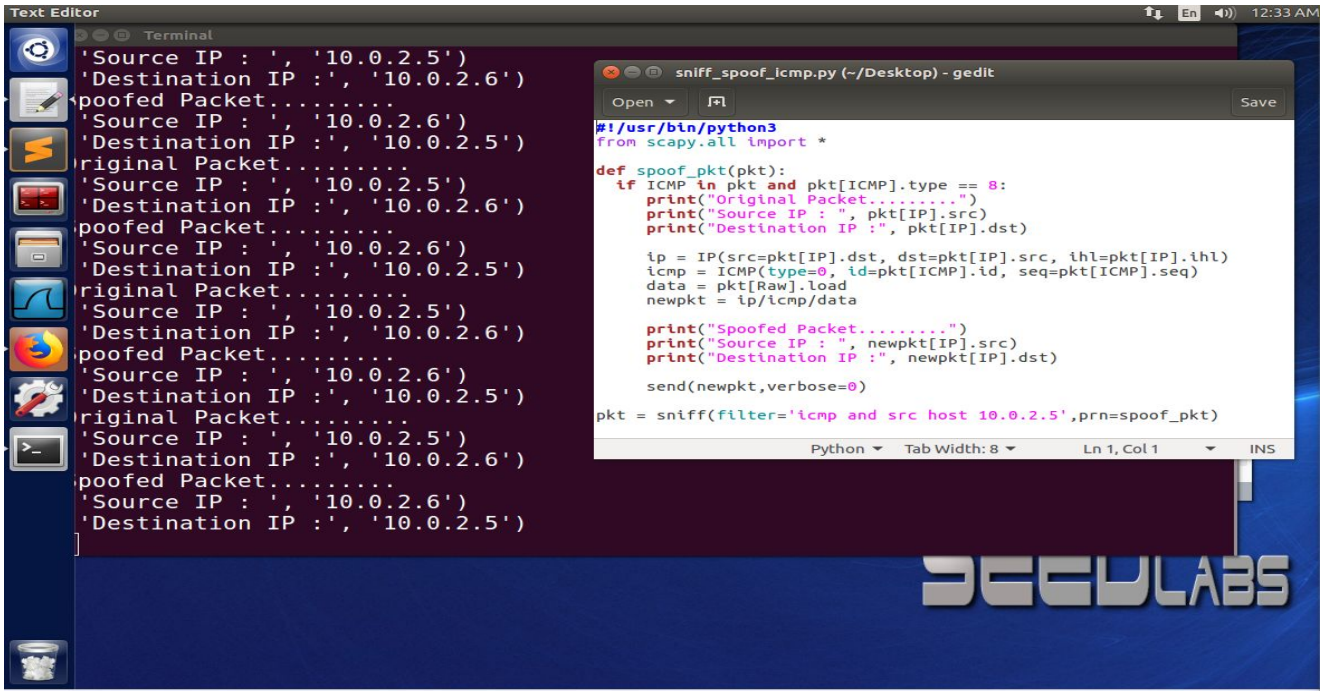

Terminal
1 hops away: 10.0.2.1
i got into the loop
1 hops away: 10.0.2.1
i got into the loop
^C[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
i ran
Traceback (most recent call last):
  File "ttl.py", line 4, in <module>
    i +=1
NameError: name 'i' is not defined
[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
i ran
i got into the loop
1 hops away: 10.0.2.1
i got into the loop
^[[A^C[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
  File "ttl.py", line 12
    print "Done!", reply.src
                ^
SyntaxError: Missing parentheses in call to 'print'
[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
1 hops away: 10.0.2.1
^X^C[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
.
Sent 1 packets.
[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
.
Sent 1 packets.
[09/04/20]seed@VM:~/Desktop$
~/Desktop/ttl.py - Sublime Text (UNREGISTERED)
sniffer.py
spoofer.py
ttl.py
from scapy.all import *
a = IP()
a.dst = '172.217.1.110'
a.ttl = 5
b = ICMP()
p = a/b
send(p)
*any
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
1 2020-09-04 22:54:31.7001809… ::1 ::1 UDP 64 55118 → 37665 Len=0
2 2020-09-04 22:54:32.8161684… PcsCompu_57:7b:ca ARP 44 Who has 10.0.2.1? Tell 10.0…
3 2020-09-04 22:54:32.8175723… RealtekU_12:35:00 ARP 62 10.0.2.1 is at 52:54:00:12:…
4 2020-09-04 22:54:32.8192622… 10.0.2.5 172.217.1.110 ICMP 44 Echo (ping) request id=0x0…
5 2020-09-04 22:54:32.8321440… 96.108.89.249 10.0.2.5 ICMP 72 Time-to-live exceeded (Time…
▸ Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 6, Src: ::1, Dst: ::1
▸ User Datagram Protocol, Src Port: 55118, Dst Port: 37665
0000 00 00 03 04 00 06 00 00 00 00 00 00 00 00 86 dd ....
0010 60 07 dd c0 00 08 11 40 00 00 00 00 00 00 00 00 `......@ ....
0020 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 ....
0030 00 00 00 00 00 00 00 01 d7 4e 93 21 00 08 00 1b .... .N.!....


Terminal
Traceback (most recent call last):
  File "ttl.py", line 4, in <module>
    i +=1
NameError: name 'i' is not defined
[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
i ran
i got into the loop
1 hops away: 10.0.2.1
i got into the loop
^[[A^C[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
  File "ttl.py", line 12
    print "Done!", reply.src
                ^
SyntaxError: Missing parentheses in call to 'print'
[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
1 hops away: 10.0.2.1
^X^C[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
.
Sent 1 packets.
[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
.
Sent 1 packets.
[09/04/20]seed@VM:~/Desktop$ sudo python3 ttl.py
.
Sent 1 packets.
[09/04/20]seed@VM:~/Desktop$
~/Desktop/ttl.py - Sublime Text (UNREGISTERED)
sniffer.py
spoofer.py
ttl.py
from scapy.all import *
a = IP()
a.dst = '172.217.1.110'
a.ttl = 16
b = ICMP()
p = a/b
send(p)
*any
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
1 2020-09-04 22:55:18.7094786… PcsCompu_57:7b:ca ARP 44 Who has 10.0.2.1? Tell 10.0…
2 2020-09-04 22:55:18.7099644… RealtekU_12:35:00 ARP 62 10.0.2.1 is at 52:54:00:12:…
3 2020-09-04 22:55:18.7110790… 10.0.2.5 172.217.1.110 ICMP 44 Echo (ping) request id=0x0…
4 2020-09-04 22:55:18.7386355… 172.217.1.110 10.0.2.5 ICMP 62 Echo (ping) reply id=0x0…
5 2020-09-04 22:55:18.7388450… ::1 ::1 UDP 64 55118 → 37665 Len=0

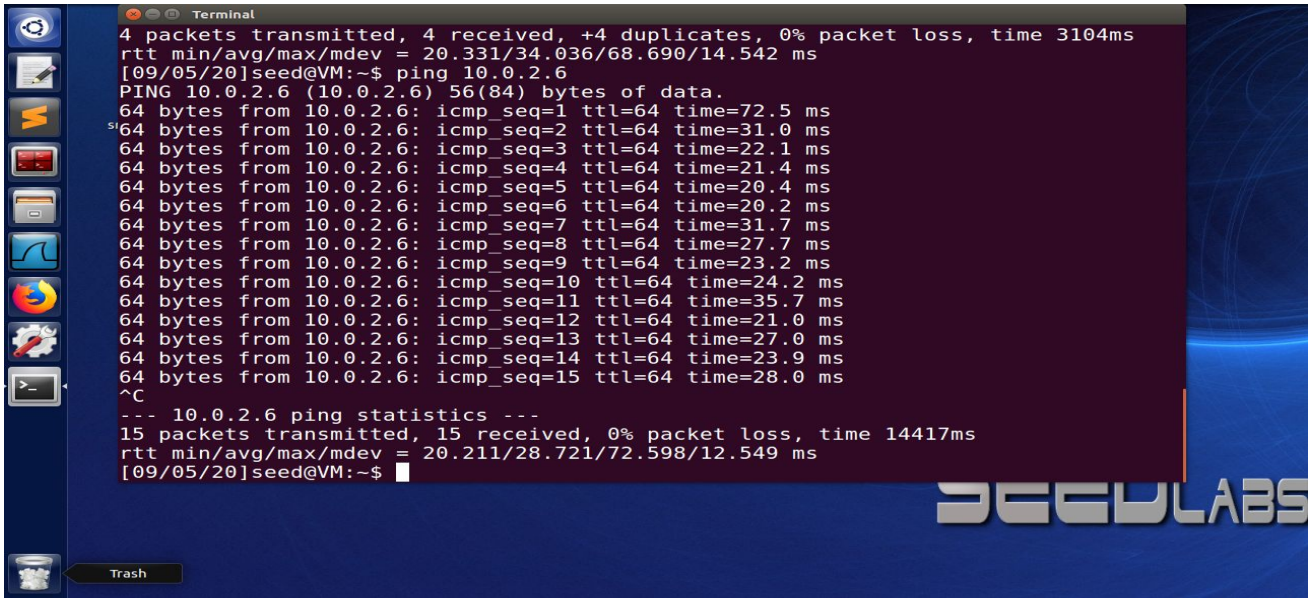
Task 1.4:

We have 3 machines and each with a unique ip address. Machine a would pretend to be machine c and machine b will try to ping machine c. This will work and it does work when machine c is on or off. Machine b would ping machine c but machine a would intercept the ping request by sniffing the network for packets from machine b and then spoof a packet by sending a packet back to b pretending it is c and the reply on b's end looks like its coming from c.

Machine A

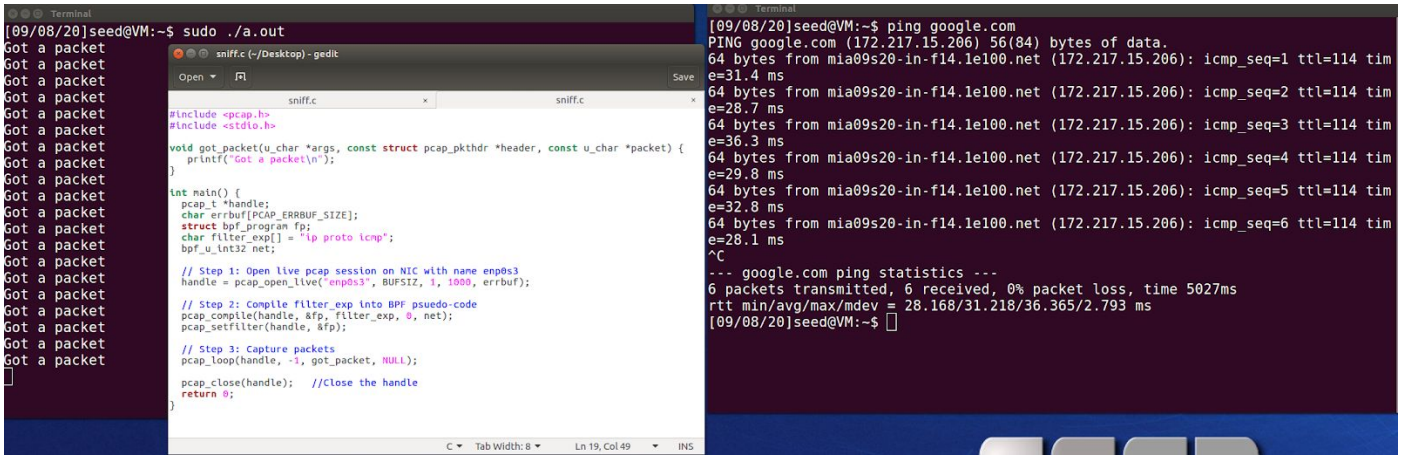


Machine B



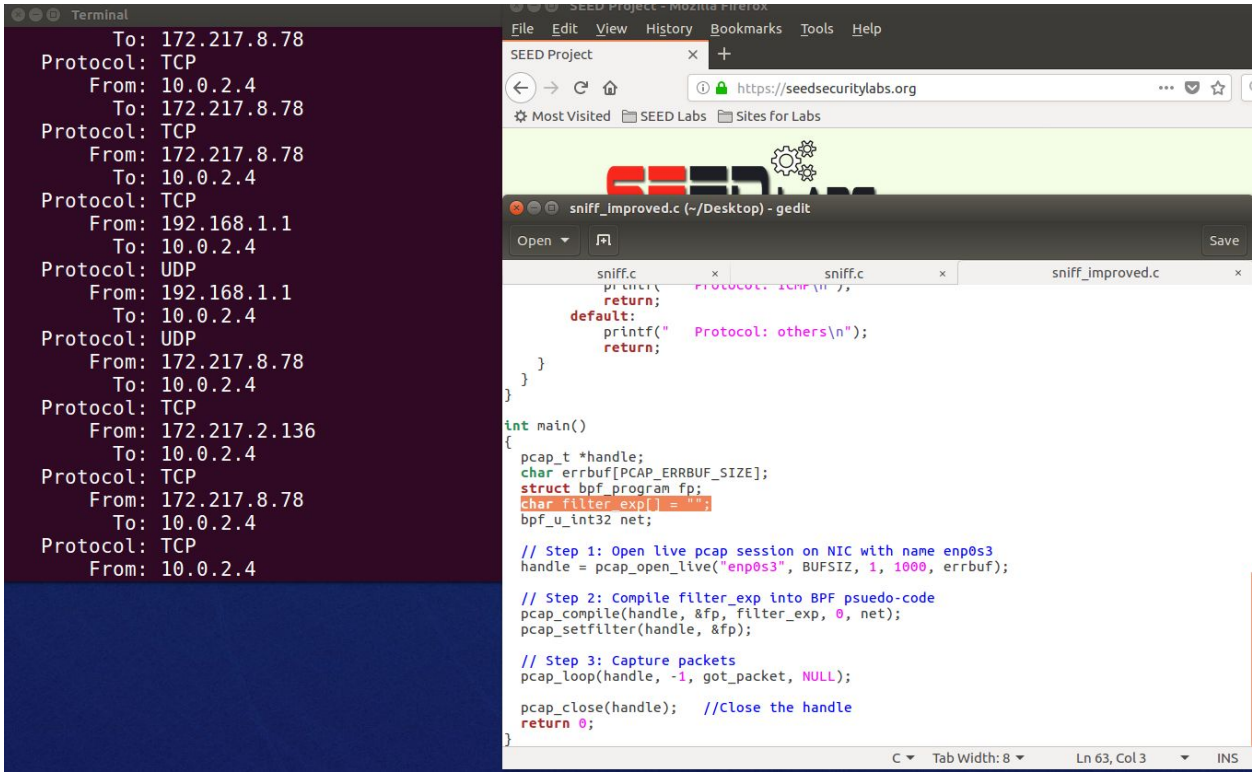
Task 2.1

Using the C code provided, we captured ICMP packets by pinging google.com.



Task 2.1A:

Captured packets with ICMP, UDP and TCP protocols by removing the filter. We found the source and the destination address by browsing the seed labs website.

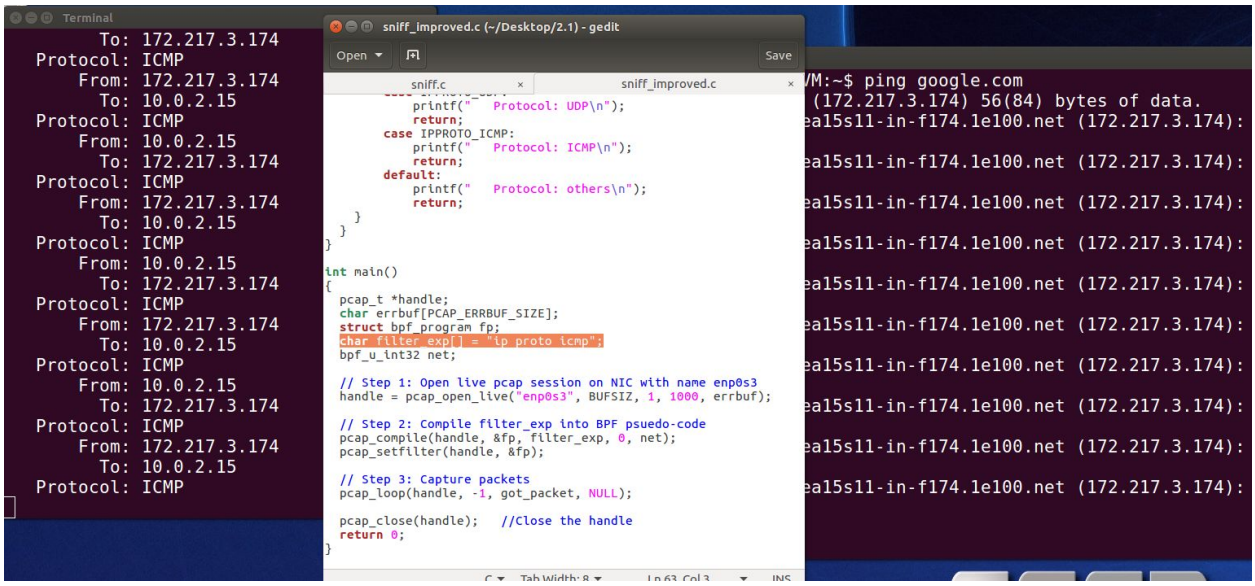


Question 1.
First a sniffer program finds a capture device to capture on and then returns the network number and mask for device then it sniffs then returns the device you're capturing on and compiles filter expression and and the end closes the session.

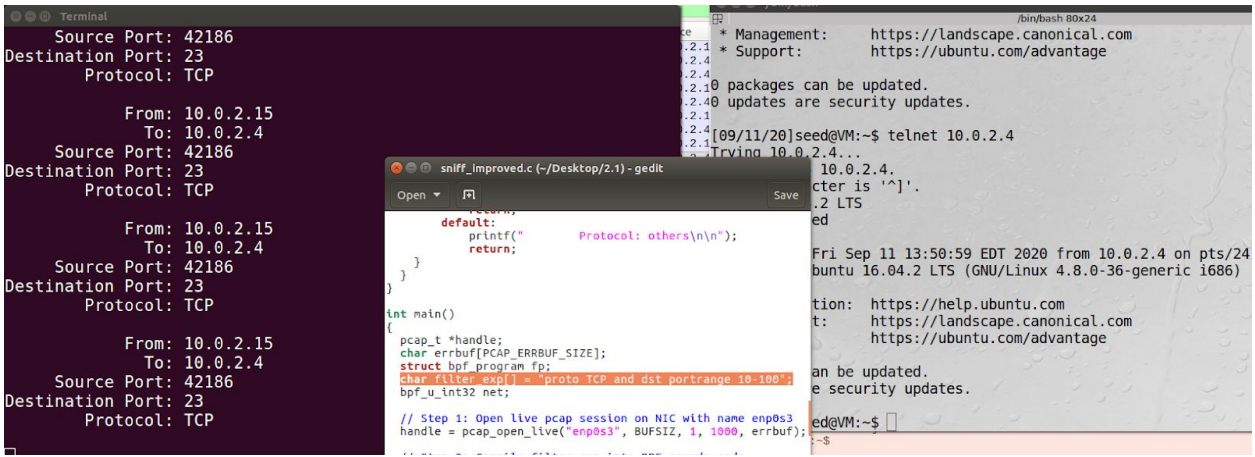
Question 2.
Because to sniff your network device needs to be accessed and only root can do that.
Question 3.
Promiscuous mode allows for network sniffer to get all traffic from a network and not only the traffic only intended to that network controller.

Task 2.1B:

- We set the filter to catch ICMP packets and we did by pinging google.com

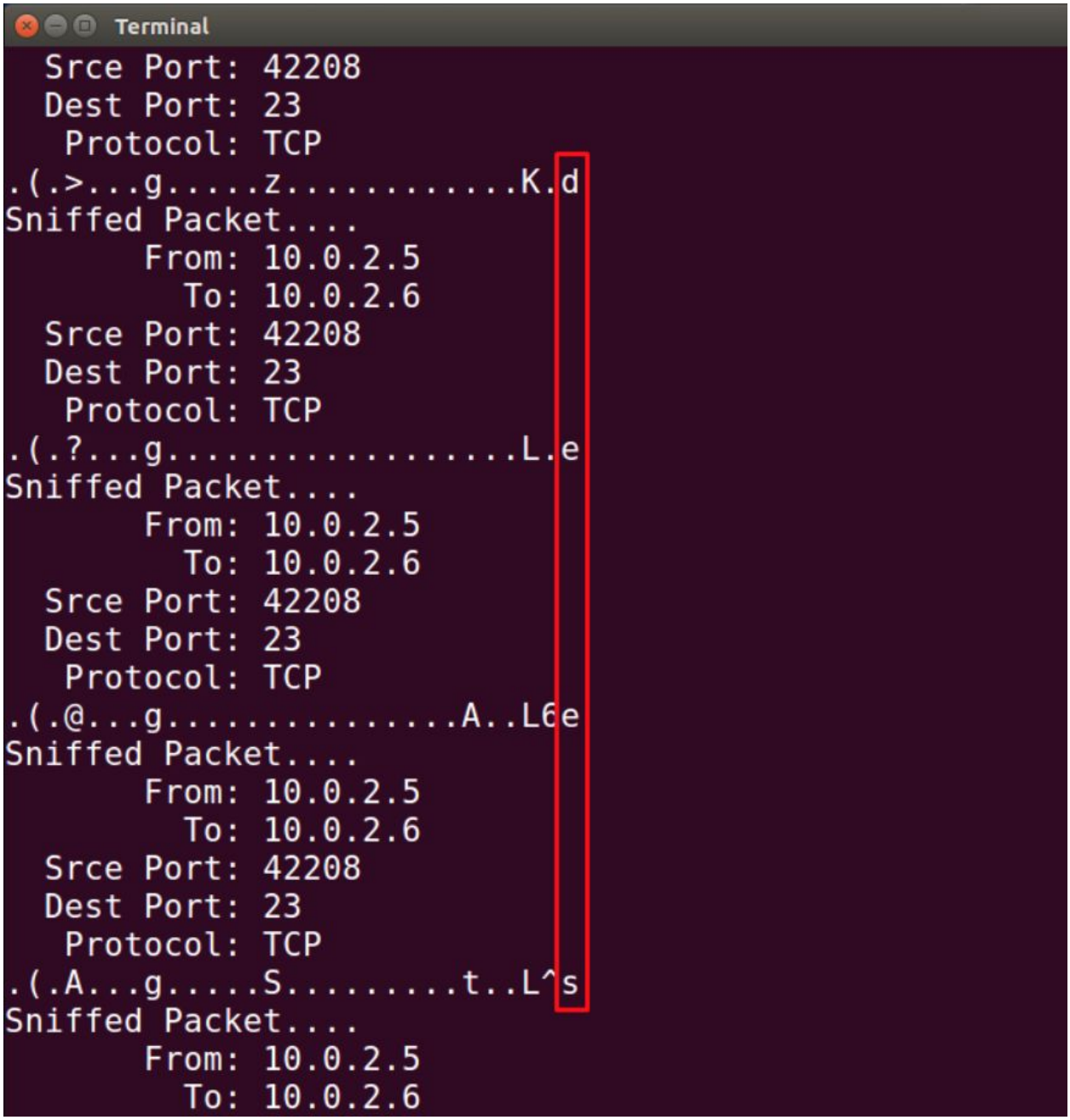


- We set the filter to catch TCP packets and the destination port between 10 and 100. Then we used the telnet command to connect to another VM and captured packets with the destination port between 10 and 100.



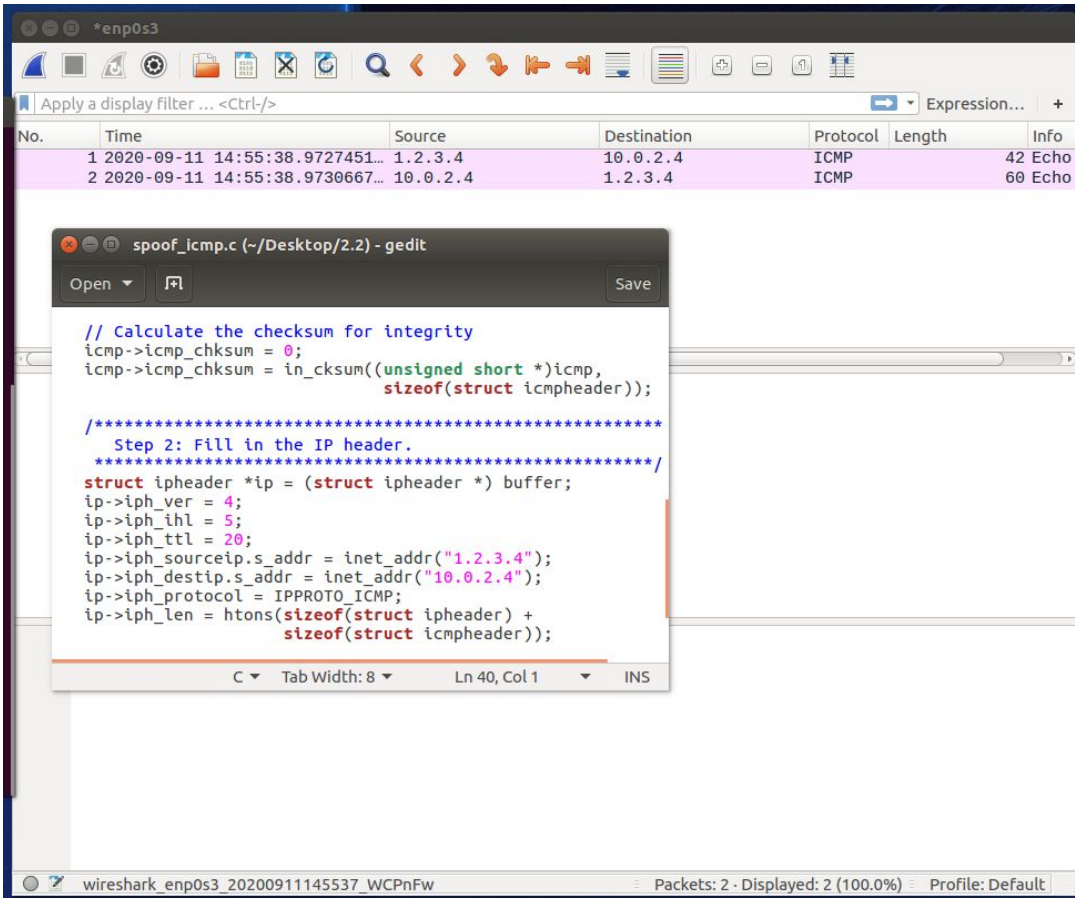
Task 2.1C:

We added the code to our sniffing program to capture data from the packets. We were able to capture the password when we connected to another VM using telnet.



Task 2.2A/2.1B

We sent a spoofed ICMP packet by changing the source IP to 1.2.3.4 and used the IP for the other VM. Using Wireshark, we can see that it successfully sent the packet and received a reply.



Question 4:

The total size of an IPv4 packet is 65535 bytes; you can't make the packet bigger, can't manipulate the header for udp/tcp.

Question 5:
The kernel builds the packet including the checksum for the data. But in raw socket programming you have to calculate the checksum for every data packet sent.

Question 6:
Raw packets need to use protocols which are part of the kernel and the usermode won't be able to access some of those protocols needed in raw packet creation therefore root is needed.