

Name: HEATHER RALEIGH

NetID:

## Lab 1 - Packet Sniffing and Spoofing Lab

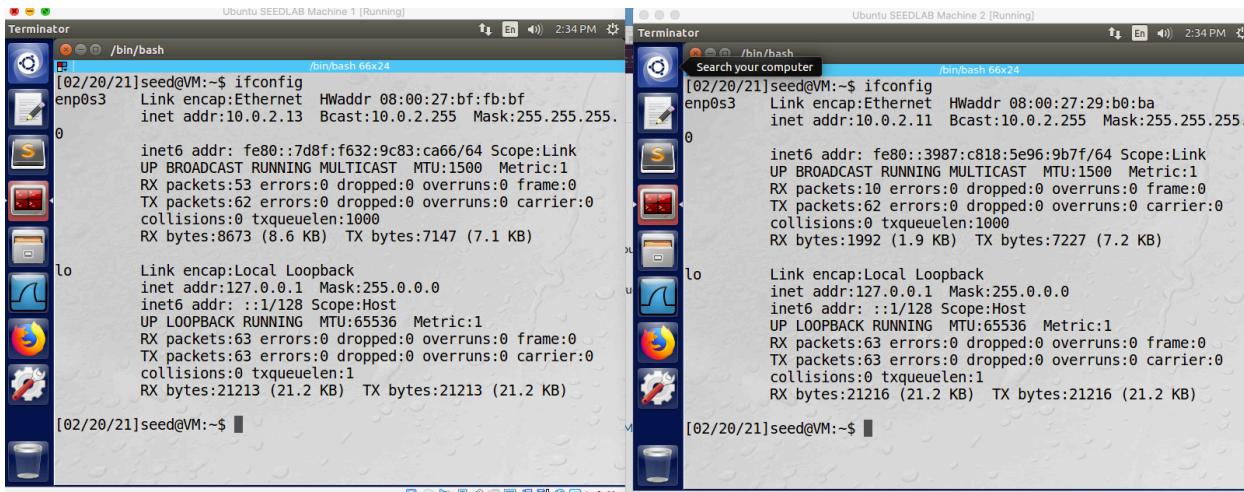
### Submission Guidelines

Submit a Word document with the items shown below.

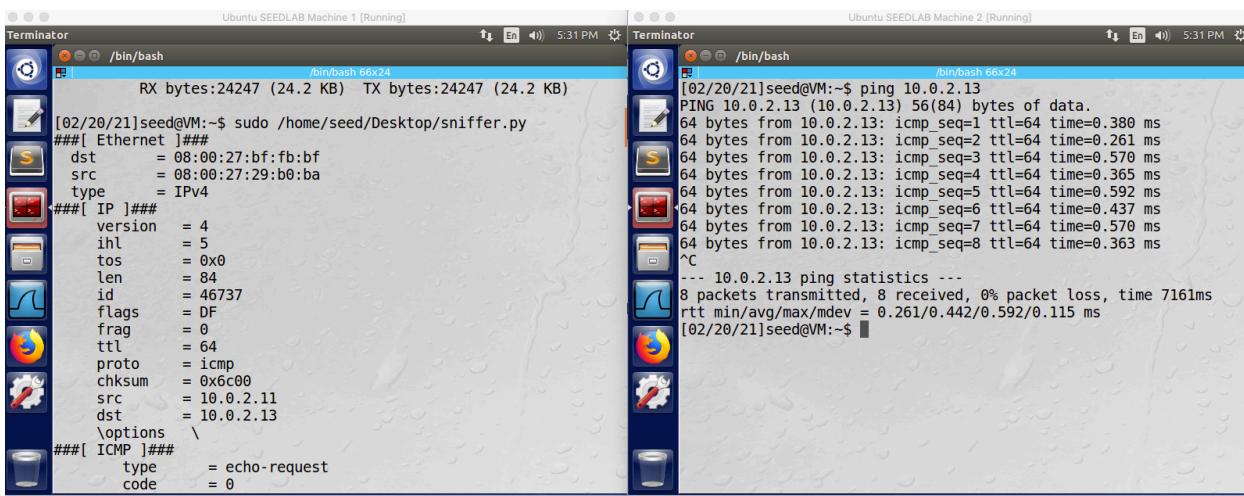
Task 1.1A [5 pts]: Run the program with root privilege and demonstrate that you can indeed capture packets.

(1) Show a screenshot showing that packets are being captured.

First, a screen shot to show that the Lab was setup successfully, with two different MAC addresses so the machines could ping each other:



Ran sniffer.py with root privileges and captured packets:



The image shows two terminal windows side-by-side. Both are running on Ubuntu SEEDLAB Machine 1 [Running]. The left terminal window displays the output of a packet sniffer (likely Wireshark) with several captured ICMP and IP packets. The right terminal window shows the command `ping 10.0.2.13` being run, followed by the ping statistics output.

```

Ubuntu SEEDLAB Machine 1 [Running]
Terminator /bin/bash /bin/bash 66x24
[02/20/21]seed@VM:~$ ping 10.0.2.13
PING 10.0.2.13 (10.0.2.13) 56(84) bytes of data.
64 bytes from 10.0.2.13: icmp seq=1 ttl=64 time=0.380 ms
64 bytes from 10.0.2.13: icmp seq=2 ttl=64 time=0.261 ms
64 bytes from 10.0.2.13: icmp seq=3 ttl=64 time=0.570 ms
64 bytes from 10.0.2.13: icmp seq=4 ttl=64 time=0.592 ms
64 bytes from 10.0.2.13: icmp seq=5 ttl=64 time=0.437 ms
64 bytes from 10.0.2.13: icmp seq=6 ttl=64 time=0.570 ms
64 bytes from 10.0.2.13: icmp seq=7 ttl=64 time=0.365 ms
64 bytes from 10.0.2.13: icmp seq=8 ttl=64 time=0.363 ms
^C
--- 10.0.2.13 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7161ms
rtt min/avg/max/mdev = 0.261/0.442/0.592/0.115 ms
[02/20/21]seed@VM:~$ 

Ubuntu SEEDLAB Machine 1 [Running]
Terminator /bin/bash /bin/bash 66x24
[02/20/21]seed@VM:~$ ping 10.0.2.13
PING 10.0.2.13 (10.0.2.13) 56(84) bytes of data.
64 bytes from 10.0.2.13: icmp seq=1 ttl=64 time=0.380 ms
64 bytes from 10.0.2.13: icmp seq=2 ttl=64 time=0.261 ms
64 bytes from 10.0.2.13: icmp seq=3 ttl=64 time=0.570 ms
64 bytes from 10.0.2.13: icmp seq=4 ttl=64 time=0.365 ms
64 bytes from 10.0.2.13: icmp seq=5 ttl=64 time=0.437 ms
64 bytes from 10.0.2.13: icmp seq=6 ttl=64 time=0.570 ms
64 bytes from 10.0.2.13: icmp seq=7 ttl=64 time=0.363 ms
64 bytes from 10.0.2.13: icmp seq=8 ttl=64 time=0.363 ms
^C
--- 10.0.2.13 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7161ms
rtt min/avg/max/mdev = 0.261/0.442/0.592/0.115 ms
[02/20/21]seed@VM:~$ 

```

```

Ubuntu SEEDLAB Machine 1 [Running]
Terminator /bin/bash /bin/bash 66x24
[02/20/21]seed@VM:~$ ping 10.0.2.13
PING 10.0.2.13 (10.0.2.13) 56(84) bytes of data.
64 bytes from 10.0.2.13: icmp seq=1 ttl=64 time=0.380 ms
64 bytes from 10.0.2.13: icmp seq=2 ttl=64 time=0.261 ms
64 bytes from 10.0.2.13: icmp seq=3 ttl=64 time=0.570 ms
64 bytes from 10.0.2.13: icmp seq=4 ttl=64 time=0.365 ms
64 bytes from 10.0.2.13: icmp seq=5 ttl=64 time=0.437 ms
64 bytes from 10.0.2.13: icmp seq=6 ttl=64 time=0.570 ms
64 bytes from 10.0.2.13: icmp seq=7 ttl=64 time=0.363 ms
64 bytes from 10.0.2.13: icmp seq=8 ttl=64 time=0.363 ms
^C
--- 10.0.2.13 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7161ms
rtt min/avg/max/mdev = 0.261/0.442/0.592/0.115 ms
[02/20/21]seed@VM:~$ 

```

```

Ubuntu SEEDLAB Machine 1 [Running]
Terminator /bin/bash /bin/bash 66x24
[02/20/21]seed@VM:~$ ping 10.0.2.13
PING 10.0.2.13 (10.0.2.13) 56(84) bytes of data.
64 bytes from 10.0.2.13: icmp seq=1 ttl=64 time=0.380 ms
64 bytes from 10.0.2.13: icmp seq=2 ttl=64 time=0.261 ms
64 bytes from 10.0.2.13: icmp seq=3 ttl=64 time=0.570 ms
64 bytes from 10.0.2.13: icmp seq=4 ttl=64 time=0.365 ms
64 bytes from 10.0.2.13: icmp seq=5 ttl=64 time=0.437 ms
64 bytes from 10.0.2.13: icmp seq=6 ttl=64 time=0.570 ms
64 bytes from 10.0.2.13: icmp seq=7 ttl=64 time=0.363 ms
64 bytes from 10.0.2.13: icmp seq=8 ttl=64 time=0.363 ms
^C
--- 10.0.2.13 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7161ms
rtt min/avg/max/mdev = 0.261/0.442/0.592/0.115 ms
[02/20/21]seed@VM:~$ 

```

(2) Write a paragraph to answer: What happens when you don't run with root privileges? Why?

Screen capture of attempt to run sniffer.py without root privileges:

The image shows two terminal windows side-by-side. Both are running on Ubuntu SEEDLAB Machine 1 [Running]. The left terminal window shows the execution of `sniffer.py` without root privileges, resulting in a `PermissionError` due to the lack of permission to bind to port 8888. The right terminal window shows the command `ping 10.0.2.13` being run, followed by the ping statistics output.

```

Ubuntu SEEDLAB Machine 1 [Running]
Terminator /bin/bash /bin/bash 66x24
[02/21/21]seed@VM:~$ /home/seed/Desktop/sniffer.py
SNIFFING PACKETS.....
Traceback (most recent call last):
  File "/home/seed/Desktop/sniffer.py", line 13, in <module>
    pkt = sniff(count=3, filter='net 8.8.', prn=print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    sniff_.run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 997, 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
[02/21/21]seed@VM:~$ 

Ubuntu SEEDLAB Machine 2 [Running]
Terminator /bin/bash /bin/bash 66x24
[02/21/21]seed@VM:~$ ping 10.0.2.13
PING 10.0.2.13 (10.0.2.13) 56(84) bytes of data.
64 bytes from 10.0.2.13: icmp seq=23 ttl=64 time=0.479 ms
64 bytes from 10.0.2.13: icmp seq=24 ttl=64 time=0.392 ms
64 bytes from 10.0.2.13: icmp seq=25 ttl=64 time=0.309 ms
64 bytes from 10.0.2.13: icmp seq=26 ttl=64 time=0.590 ms
64 bytes from 10.0.2.13: icmp seq=27 ttl=64 time=0.644 ms
64 bytes from 10.0.2.13: icmp seq=28 ttl=64 time=0.461 ms
64 bytes from 10.0.2.13: icmp seq=29 ttl=64 time=0.279 ms
64 bytes from 10.0.2.13: icmp seq=30 ttl=64 time=0.442 ms
64 bytes from 10.0.2.13: icmp seq=31 ttl=64 time=0.405 ms
64 bytes from 10.0.2.13: icmp seq=32 ttl=64 time=0.485 ms
64 bytes from 10.0.2.13: icmp seq=33 ttl=64 time=0.491 ms
64 bytes from 10.0.2.13: icmp seq=34 ttl=64 time=0.732 ms
64 bytes from 10.0.2.13: icmp seq=35 ttl=64 time=0.604 ms
64 bytes from 10.0.2.13: icmp seq=36 ttl=64 time=0.297 ms
64 bytes from 10.0.2.13: icmp seq=37 ttl=64 time=0.462 ms
64 bytes from 10.0.2.13: icmp seq=38 ttl=64 time=0.646 ms
64 bytes from 10.0.2.13: icmp seq=39 ttl=64 time=0.407 ms
64 bytes from 10.0.2.13: icmp seq=40 ttl=64 time=0.490 ms
64 bytes from 10.0.2.13: icmp seq=41 ttl=64 time=0.347 ms
^C
--- 10.0.2.13 ping statistics ---
41 packets transmitted, 41 received, 0% packet loss, time 40905ms
rtt min/avg/max/mdev = 0.279/0.497/1.199/0.196 ms
[02/21/21]seed@VM:~$ 

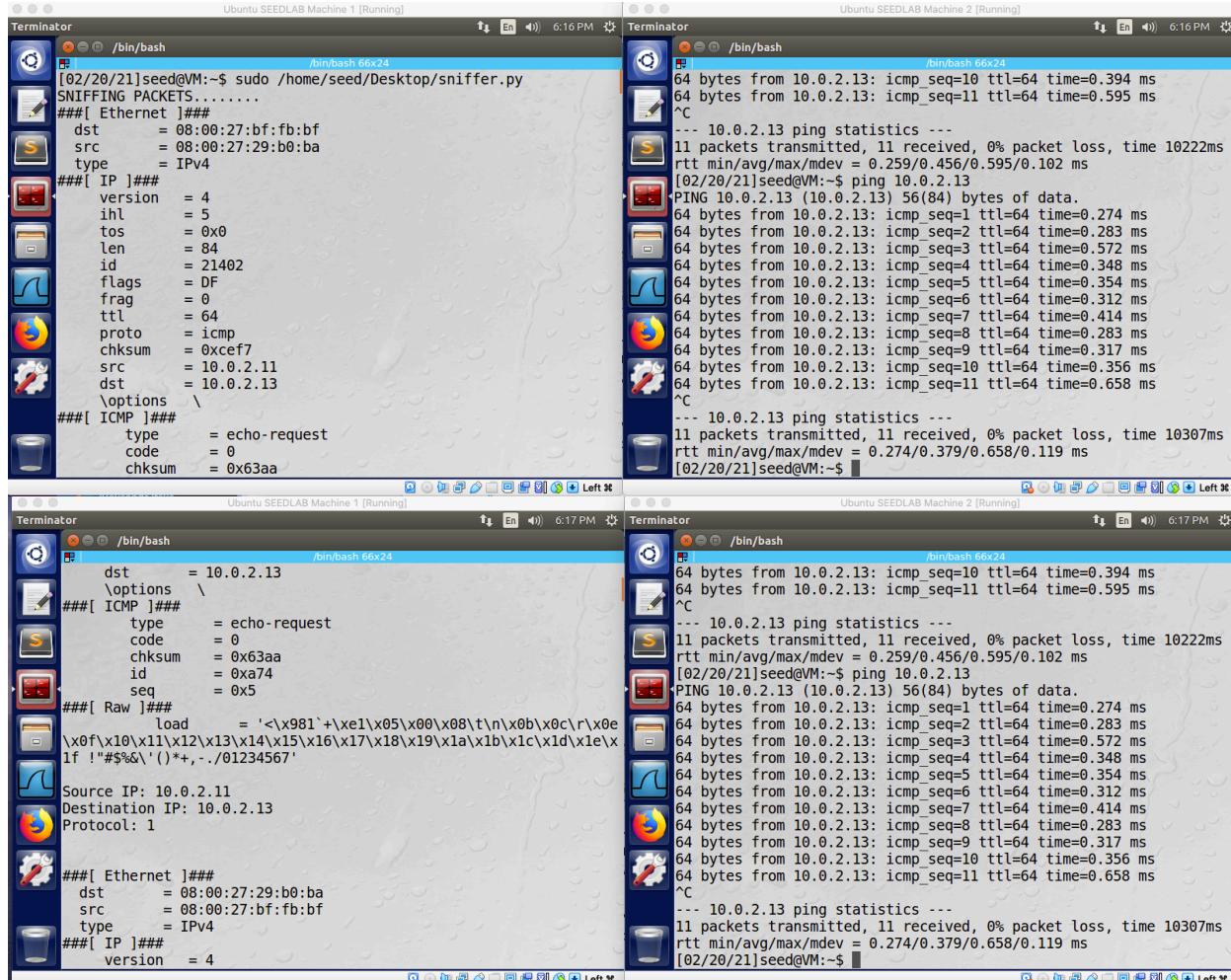
```

Error reply stated that the operation was not permitted. The reason it was not possible to run the program without root privileges is that Linux has a 'least permissions' policy for security reasons. In other words, the default setting is the most restricted.

Task 1.1B [15 pts]:

- (1) Capture only the ICMP packet. Submit your code and take a screenshot demonstrating that ICMP packets are captured.

Screen capture of traffic captured on Machine 1, using sniffer.py program, while a ping was sent from Machine 2 to Machine 1:



Sniffer.py code:

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

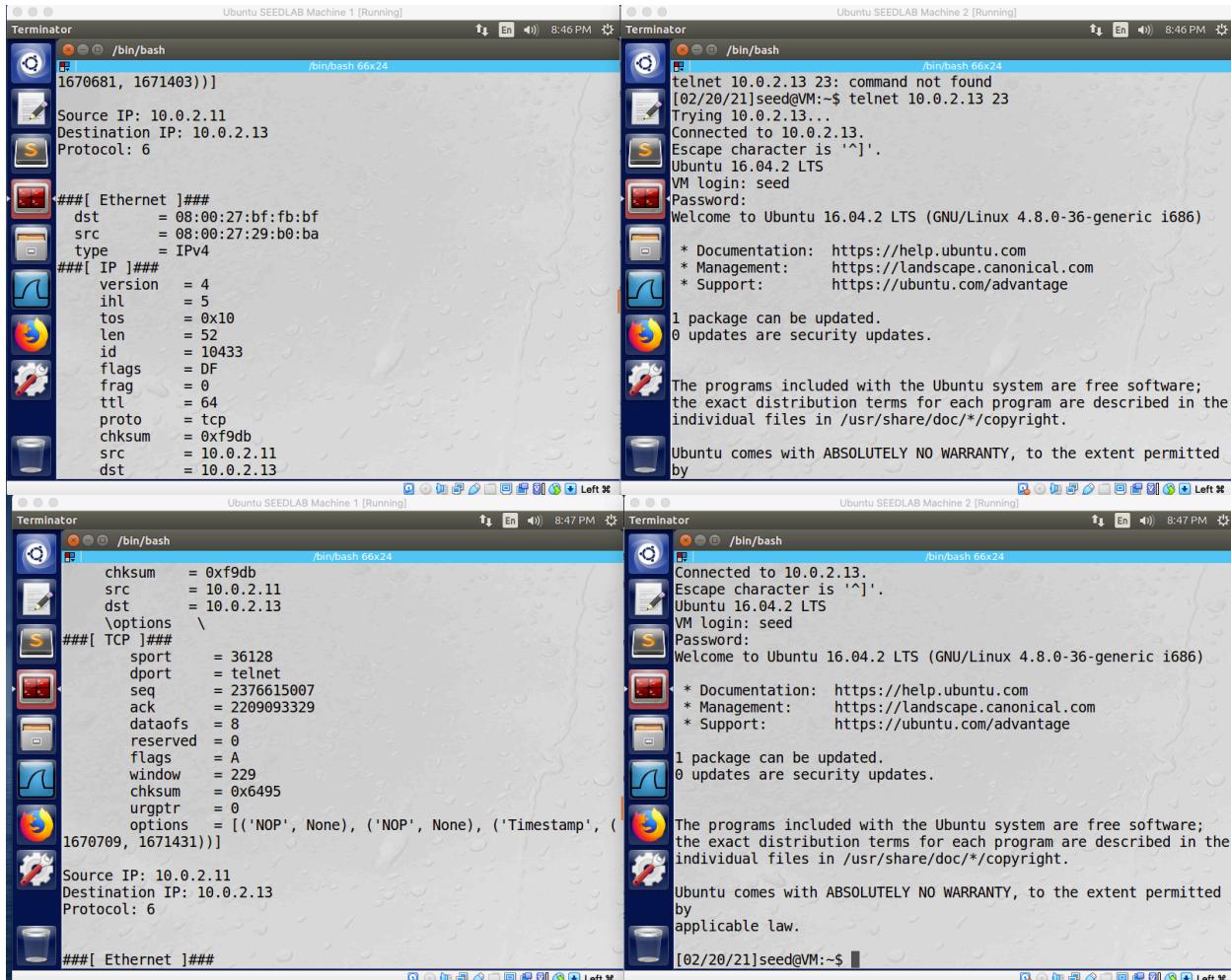
print("SNIFFING PACKETS.....")

def print_pkt(pkt):
    pkt.show()
    print("Source IP:", pkt[IP].src)
    print("Destination IP:", pkt[IP].dst)
    print("Protocol:", pkt[IP].proto)
    print("\n")

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

- (2) Capture any TCP packet that comes from a particular IP and with a destination port number 23. Submit your code and take a screenshot demonstrating that.

Ran sniffer.py on Machine 1, while connected via telnet from Machine 2:



Code:

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

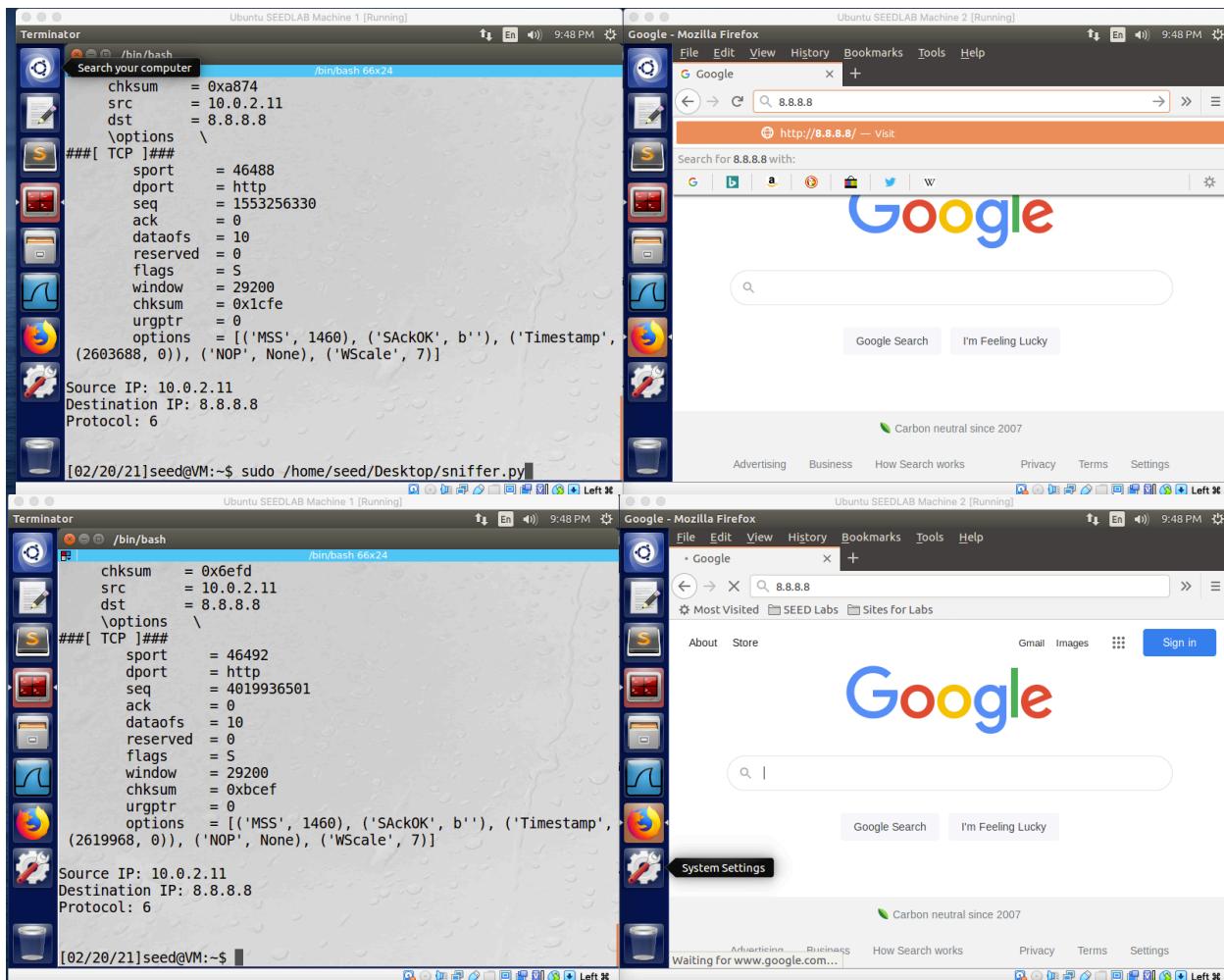
print("SNIFFING PACKETS.....")

def print_pkt(pkt):
    pkt.show()
    print("Source IP:", pkt[IP].src)
    print("Destination IP:", pkt[IP].dst)
    print("Protocol:", pkt[IP].proto)
    print("\n")

pkt = sniff(filter='tcp and src host 10.0.2.11 and dst port 23', prn=print_pkt)
```

(3) Capture packets coming from or to going to a particular subnet. Submit your code and take a screenshot demonstrating the traffic generated.

The IP for [www.google.com](http://www.google.com), 8.8.8.8 was chosen to generate traffic. In the code, 8.8 was the subnet filtered.



Code:

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

print("SNIFFING PACKETS.....")

def print_pkt(pkt):
    pkt.show()
    print("Source IP:", pkt[IP].src)
    print("Destination IP:", pkt[IP].dst)
    print("Protocol:", pkt[IP].proto)
    print("\n")

pkt = sniff(count=3, filter='net 8.8', prn=print_pkt)
```

Task 1.2 [20 pts]: Spoof an ICMP echo request packet with source IP address 8.8.8.8 from the first VM and send to the second VM. Use Wireshark on the second VM to show that it replies back with echo replies.

(1) Submit your scapy code

```
#!/usr/bin/python3
```

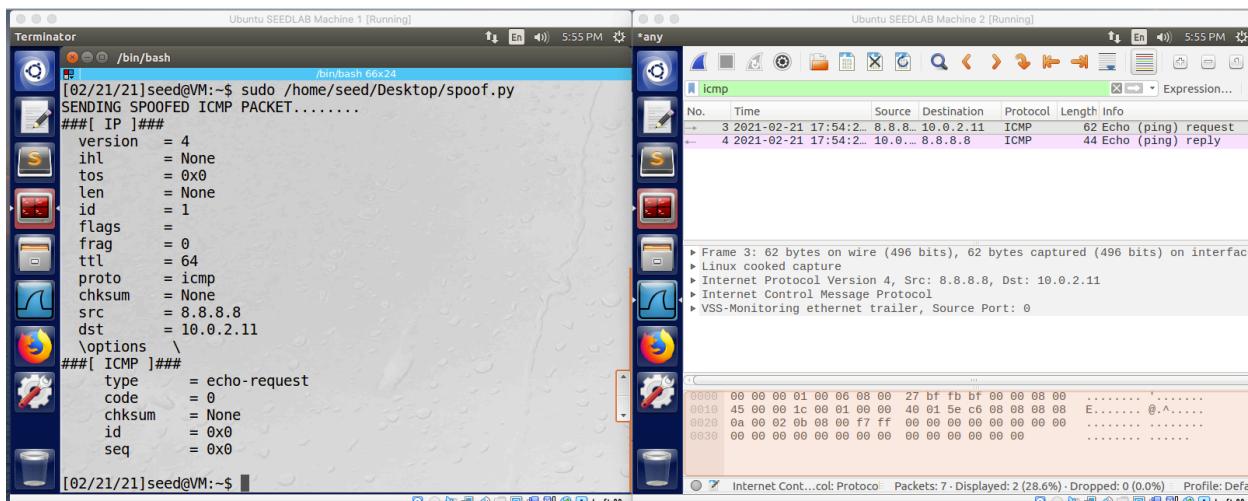
```

from scapy.all import *

print("SENDING SPOOFED ICMP PACKET.....")
ip = IP(src="8.8.8.8", dst="10.0.2.11")
icmp = ICMP()
pkt = ip/icmp
pkt.show()
send(pkt, verbose=0)

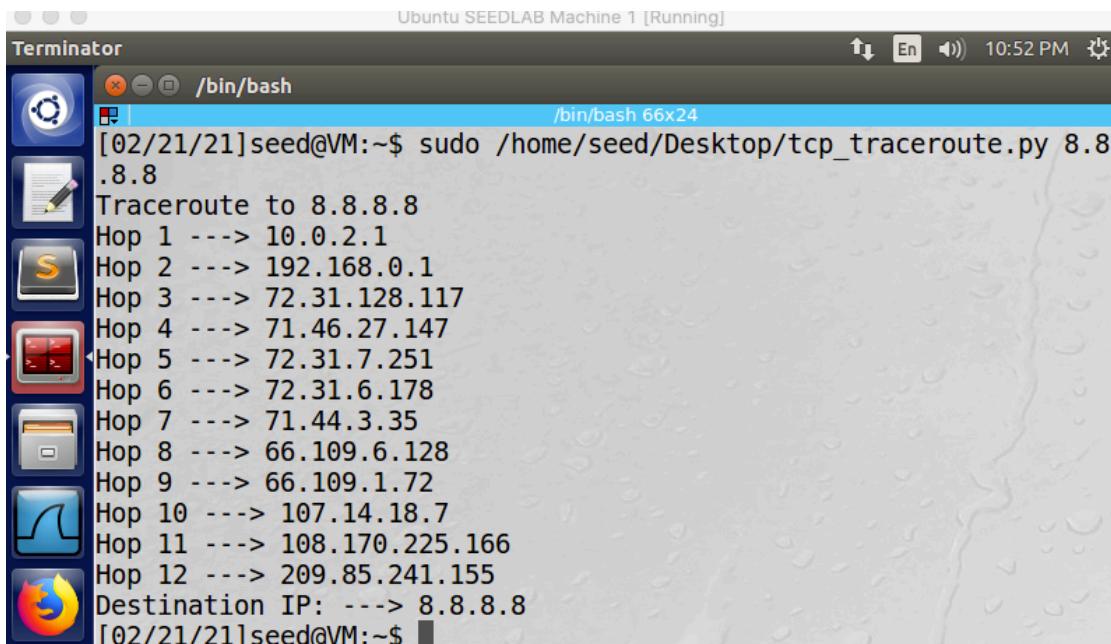
```

(2) Submit a screenshot of Wireshark showing that the VM replies back with echo requests from 8.8.8.8.



Task 1.3 [20 pts]: Implement TCP traceroute using scapy. Do NOT use the built-in scapy traceroute function. Perform a traceroute to 8.8.8.8. Submit your code and take screenshot of your program's output.

Implementation of traceroute to 8.8.8.8:



Code:

```
#!/usr/bin/python3
```

```

import sys
import os
from scapy.all import *

print("Traceroute to 8.8.8.8")

if len(sys.argv) != 2:
    sys.exit('Usage: tcp_traceroute.py 8.8.8.8')

# we start with 1
ttl = 1
while 1:
    p=sr1(IP(dst=sys.argv[1],ttl=ttl)/ICMP(id=os.getpid()),
          verbose=0)
    # if time exceeded due to TTL exceeded
    if p[ICMP].type == 11 and p[ICMP].code == 0:
        print("Hop", ttl, "---->", p.src)
        ttl += 1
    elif p[ICMP].type == 0:
        print("Destination IP: ---->", p.src)
        break

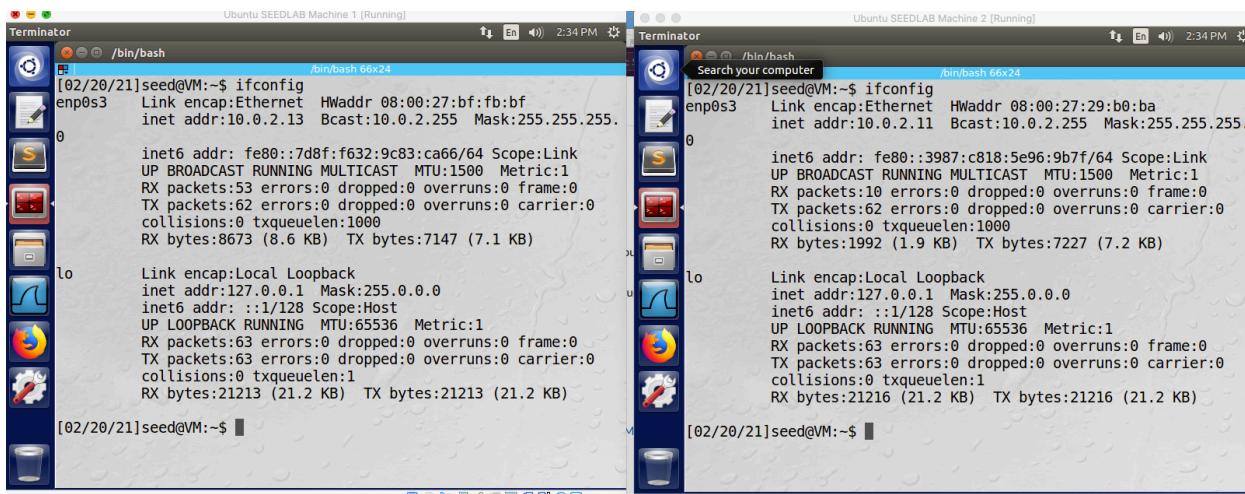
```

Task 1.4 [40 pts]: Sniffing and-then Spoofing. You will use two different VMs to sniff ping packets and then spoof a reply.

Note 1: No ARP spoofing or other Attacker-in-the-Middle (AIM) techniques are required.

Note 2: if you're using VirtualBox, be sure to follow the instructions on Appendix B of the [SEED Lab Setup Instructions](#) to setup the network adaptor correctly.

First, a screen shot to confirm that the network adaptor was set up according to the SEED Lab Instructions, with unique MAC addresses.



(1) Submit your scapy code for sniffing and then spoofing

```
#!/usr/bin/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)

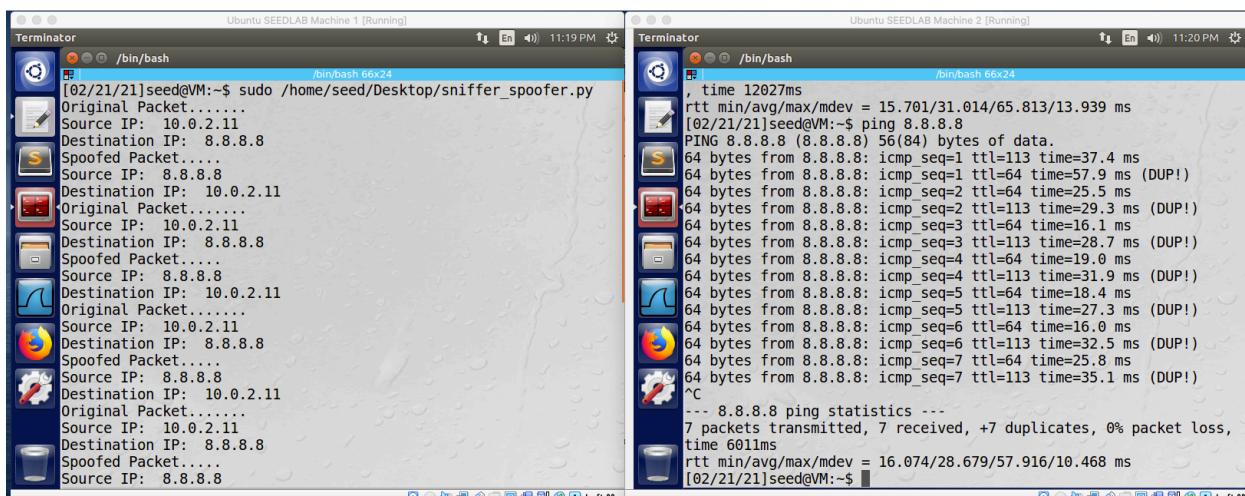
        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
        icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        newpkt = ip/icmp/data

        print("Spoofed Packet.....")
        print("Source IP: ", newpkt[IP].src)
        print("Destination IP: ", newpkt[IP].dst)
        send(newpkt, verbose=0)

pkt = sniff(filter='icmp and src host 10.0.2.11', prn=spoof_pkt)

```

(2) Ping 8.8.8.8 and show screenshots of the output from your program and from ping



(3) Ping 1.2.3.4 and show screenshots of the output from your program and from ping

## Sources:

Du, Wenliang, "Computer & Internet Security: A Hands-on Approach", Second Edition

<https://scapy.readthedocs.io/en/latest/introduction.html>

[https://wiki.sans.blue/Tools/pdfs/ScapyCheatSheet\\_v0.2.pdf](https://wiki.sans.blue/Tools/pdfs/ScapyCheatSheet_v0.2.pdf)

<https://buildmedia.readthedocs.org/media/pdf/scapy/latest/scapy.pdf>