Lab 2 - Packet Sniffing and Spoofing Lab

● Graded

Student

Christian Jed Patino

Total Points

105 / 105 pts

Question 1

Lab 2 - Packet Sniffing and Spoofing Lab — **25** / 25 pts

1.1 **Task 1.1: Sniffing Packets - Task 1.1A** — **10** / 10 pts

✔ **+ 10 pts** Correct

1.2 **Task 1.1B - Only ICMP** — **5** / 5 pts

✔ **+ 5 pts** Correct

1.3 **Task 1.1B - Any TCP** — **5** / 5 pts

✔ **– 0 pts** Correct

1.4 **Task 1.1B - Particular Subnet** — **5** / 5 pts

✔ **+ 5 pts** Correct

Question 2

Task 1.2: Spoofing ICMP Packets — **15** / 15 pts

2.1 **Task 1.2 - Submit your scapy code** — **10** / 10 pts

✔ **+ 10 pts** Correct

2.2 **Task 1.2 - Submit a screenshot of Wireshark showing the spoof echo request from 8.8.8.8 and that the VM replied back to it with an echo reply.** — **5** / 5 pts

✔ **+ 5 pts** Correct

Question 3

Task 1.3: Traceroute — **20** / 20 pts

3.1 **Task 1.3 - Submit your scapy code** — **15** / 15 pts

✔ **+ 15 pts** Correct

3.2 **Task 1.3 - Submit screenshot of Wireshark showing proof of traceroute to 8.8.8.8** — **5** / 5 pts

✔ **+ 5 pts** Correct

Question 4

Task 1.4: Sniffing and-then Spoofing **40** / 40 pts

4.1 **Task 1.4 - Submit your scapy code for sniffing and then spoofing.** **8** / 8 pts

✔ **– 0 pts** Correct

4.2 **Task 1.4 - Ping 1.2.3.4 and show screenshots of the output from your program and with the ping from the terminal** **8** / 8 pts

✔ **– 0 pts** Correct

4.3 **Task 1.4 - Ping 10.9.0.99 and show screenshots of the output from your program and with the ping from the terminal. If this does not work, please explain why.** **8** / 8 pts

✔ **– 0 pts** Correct

4.4 **Task 1.4 - Ping 8.8.8.8 and show screenshots of the output from your program and with the ping from the terminal** **8** / 8 pts

✔ **– 0 pts** Correct

4.5 **Task 1.4 - Explain the results for Q4.2 - Q4.4.** **8** / 8 pts

✔ **– 0 pts** Correct

Question 5

Extra Credit - Ping 10.9.0.99 **0** / 5 pts

✔ **+ 0 pts** No Attempt

Question 6

Early/Late Submission Bonus **5** / 0 pts

✔ **+ 5 pts** Early Submission

Q1 Lab 2 - Packet Sniffing and Spoofing Lab
25 Points

Lab PDF:
https://seedsecuritylabs.org/Labs_20.04/Files/Sniffing_Spoofing/Sniffing_Spoofing.pdf
Note: For this lab, please only complete Task 1.1 - Task 1.4 (i.e. the scapy portions) from "3 Lab Task Set 1: Using Scapy to Sniff and Spoof Packets" in the SEED Labs PDF. Stop on page 7.

- Lab setup files: Labsetup.zip - **Please use this Labsetup for this lab.**
- Docker Manual (if more help is needed)

Note: Make sure your environment is setup, especially docker, in section 2.1, before proceeding to the tasks

Normal due date: **3 March**
Earliest acceptance date: 26 February (+1% per day, up to +5% bonus for five days early)
Latest acceptance date: 13 March (-10% points)

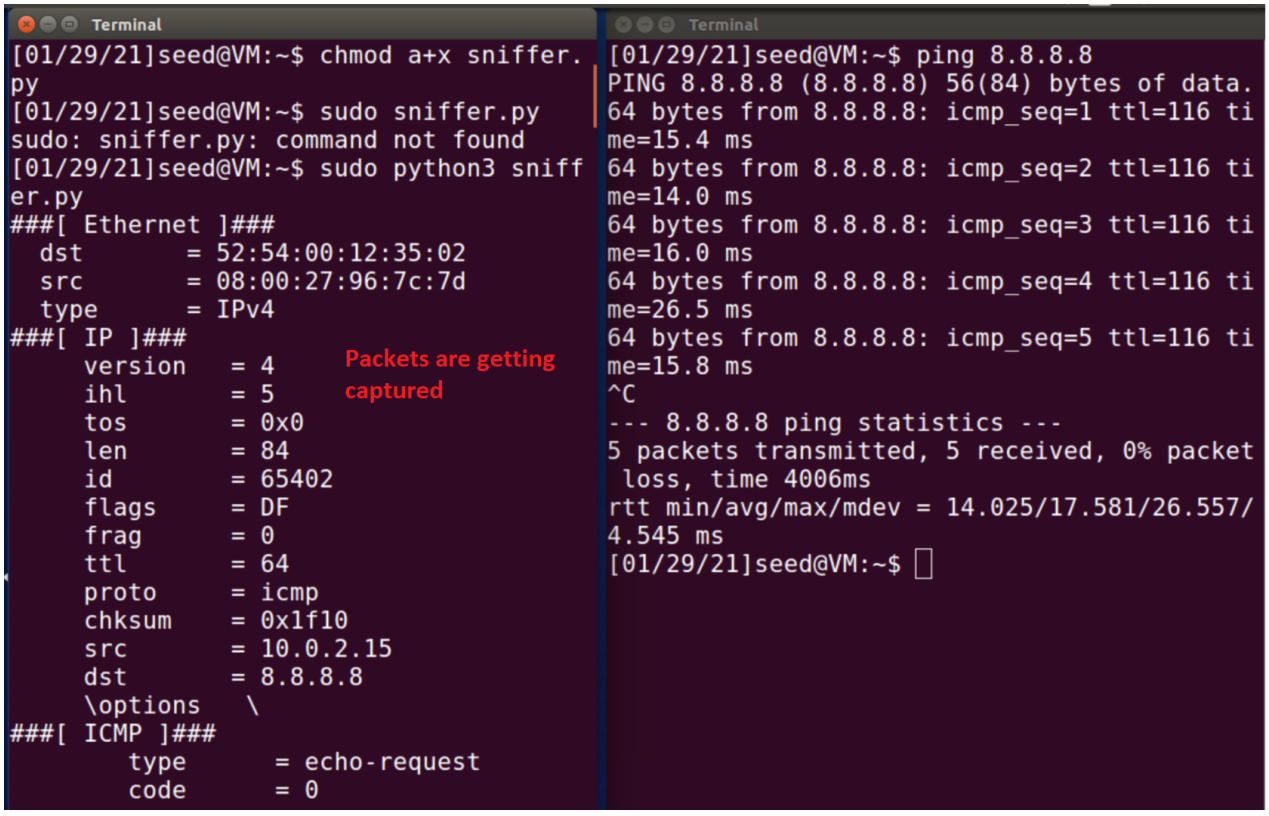
Please follow the instructions from the PDF document, but submit your work based on the instructions below.

Q1.1 Task 1.1: Sniffing Packets - Task 1.1A
10 Points

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface='br-c93733e9f913', filter='icmp', prn=print_pkt)
```

```
// Make the program executable
# chmod a+x sniffer.py
// Run the program with the root privilege
# sniffer.py
// Switch to the "seed" account, and
// run the program without the root privilege
# su seed
$ sniffer.py
```

Expected output



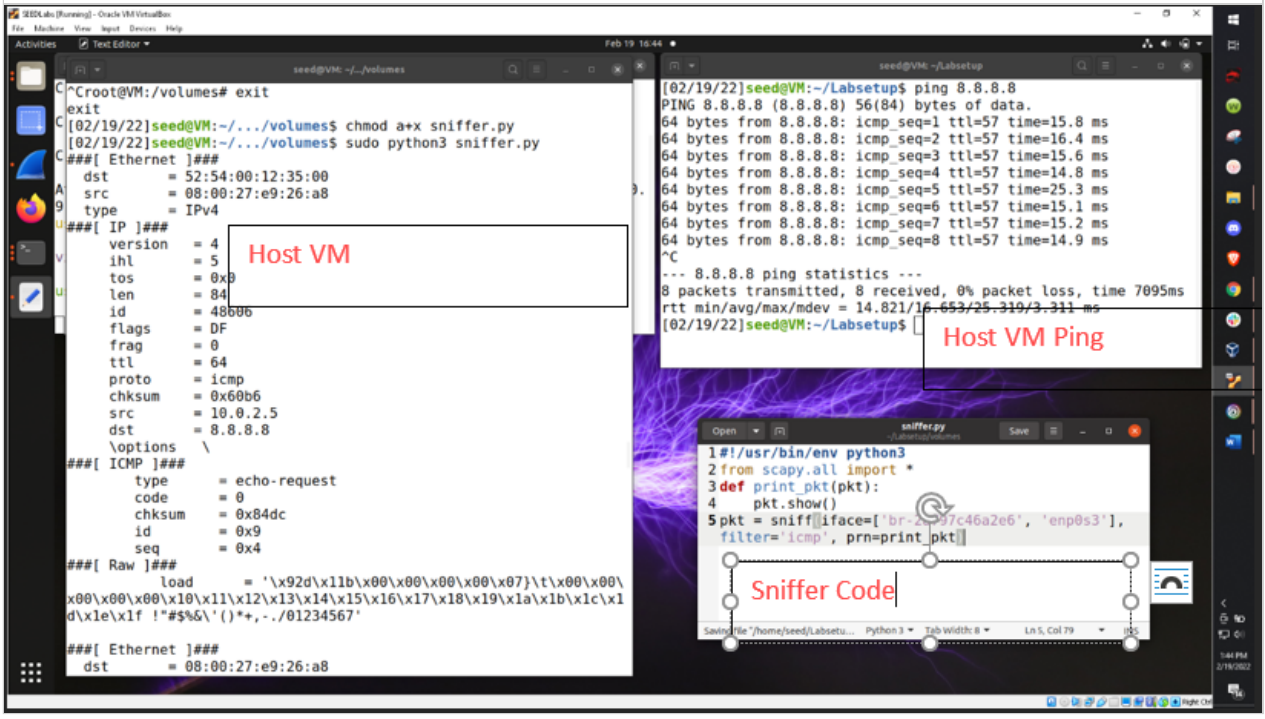
Expected output without root

```
[01/29/21]seed@VM:~$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.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
```

Run the program with the root privilege and demonstrate that you can indeed capture packets.

Show a screenshot showing that packets are being captured. (must screenshot the entire VM along with date and time)

snifferlabeled.PNG Download



Write a sentence to answer: What happens when you don't run with root privileges? Why?

When I try to run without root privileges, the operation is not permitted. This is most likely due to the inability to access certain sockets due to the lowered privileges for the sniffer to work.

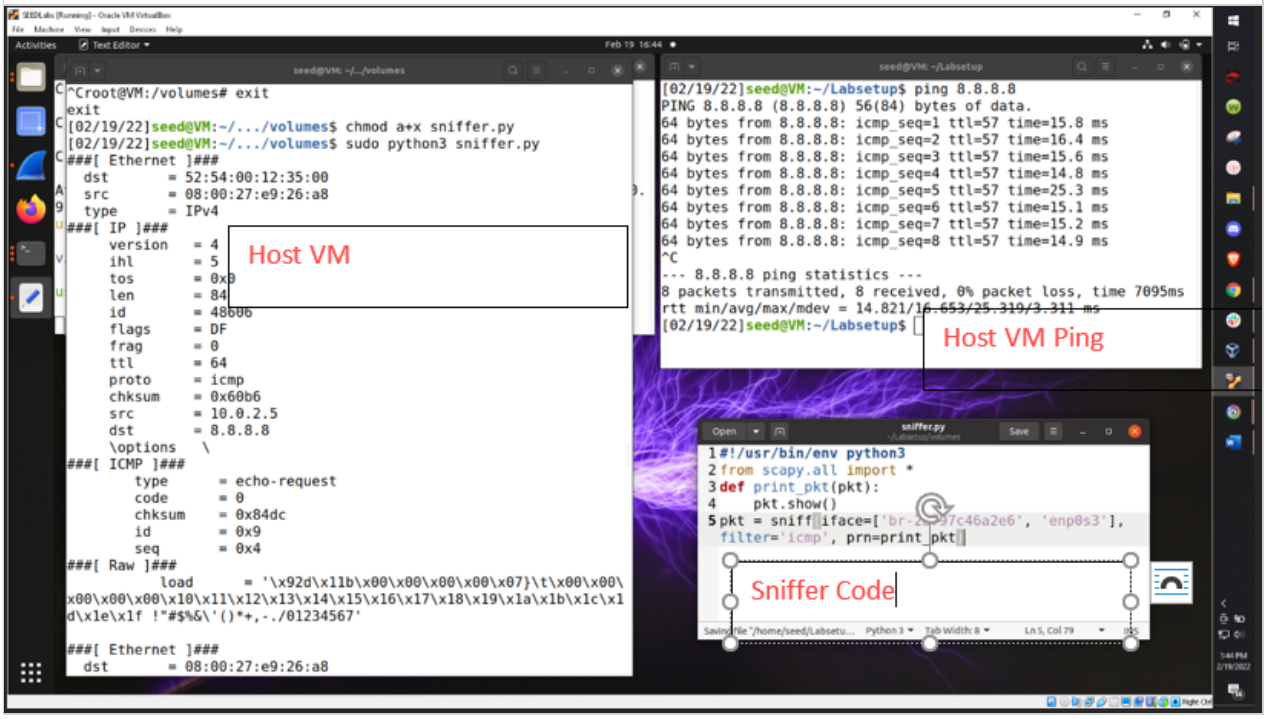
Q1.2 Task 1.1B - Only ICMP
5 Points

Capture only the ICMP packet.

Expected output
Note: you are not required to use the same IP as the screenshot
(See screenshot under sniffing packets)

Submit your code and take a screenshot demonstrating that ICMP packets are captured (must screenshot the entire VM along with date and time).

▾ **snifferlabeled.PNG** Download

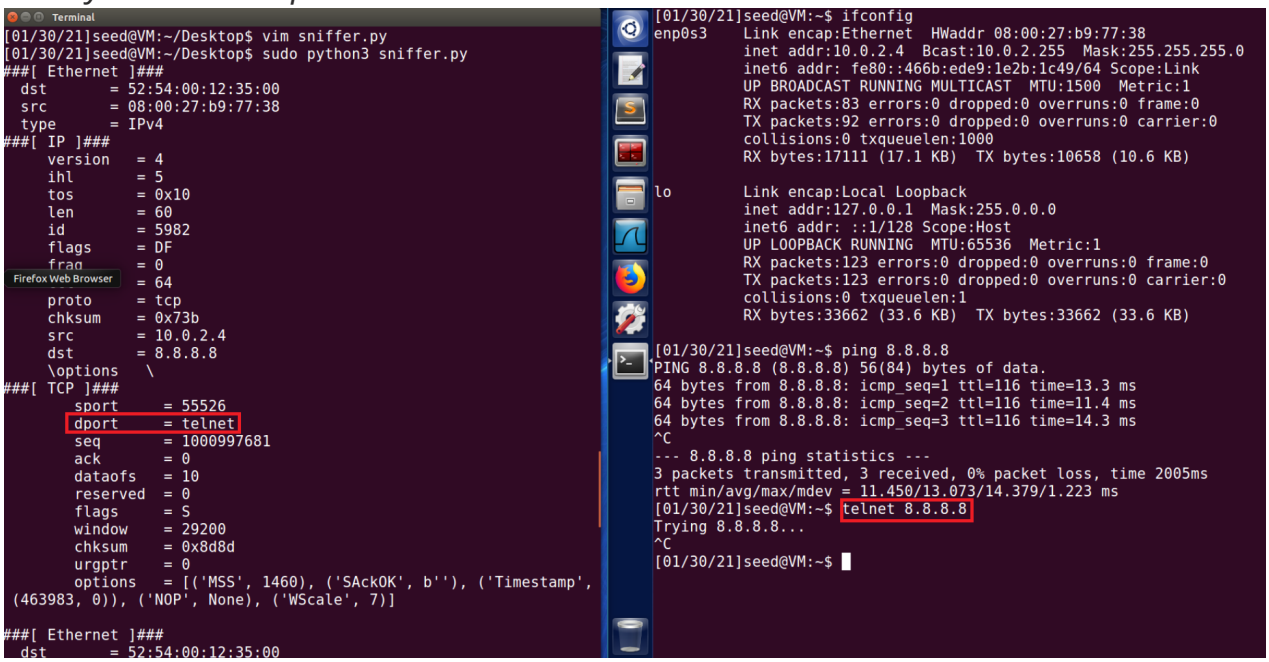


Q1.3 Task 1.1B - Any TCP
5 Points

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

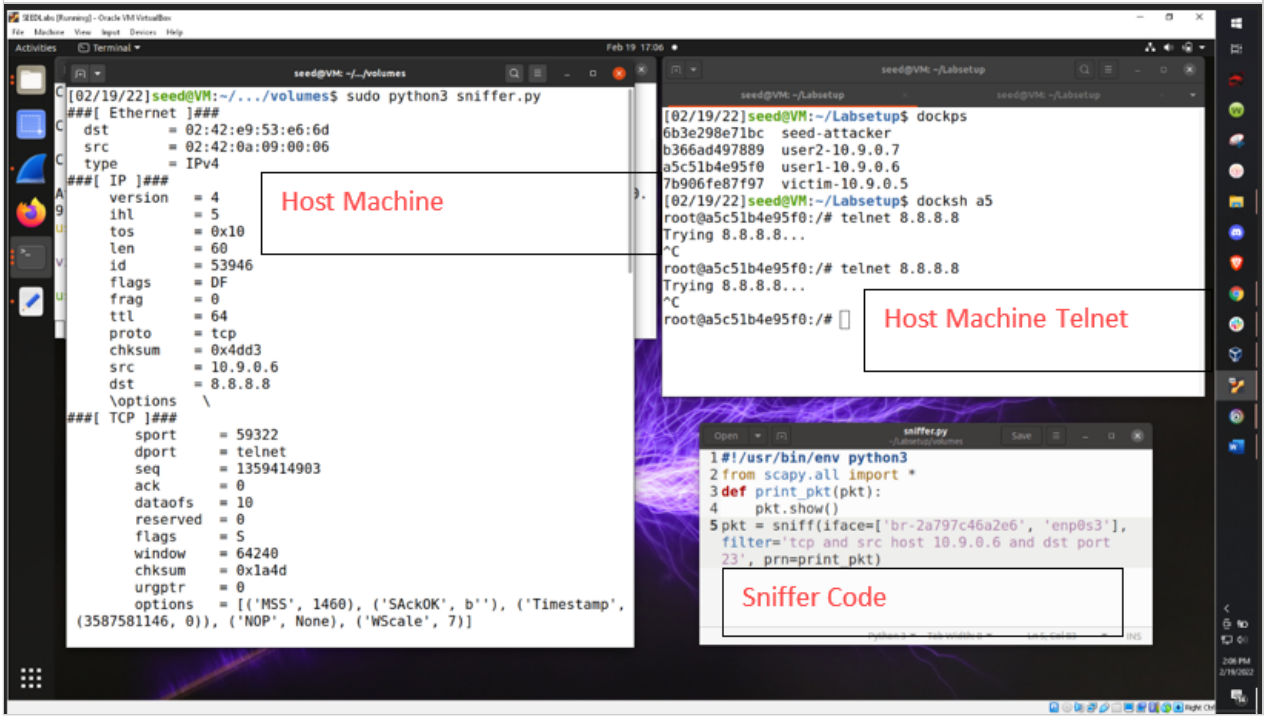
Expected output

Note: you are not required to use the same IP as the screenshot



Submit your code and take a screenshot demonstrating that (must screenshot the entire VM along with date and time).

snifferTCPlabeled.PNG Download

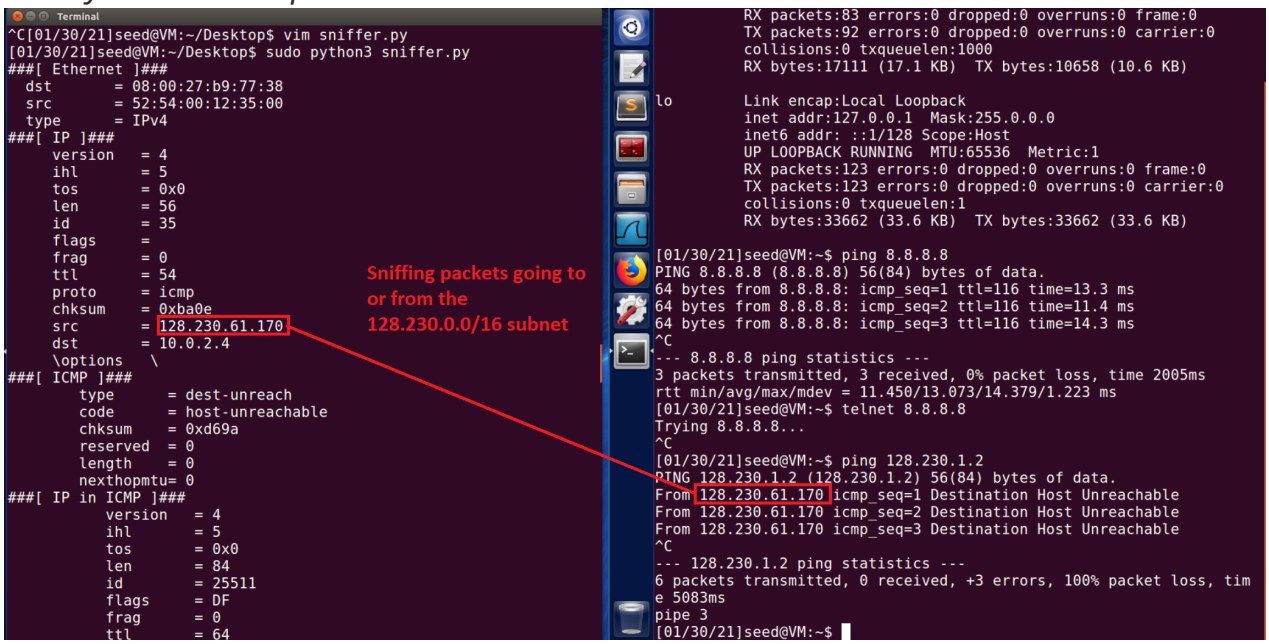


Q1.4 Task 1.1B - Particular Subnet
5 Points

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

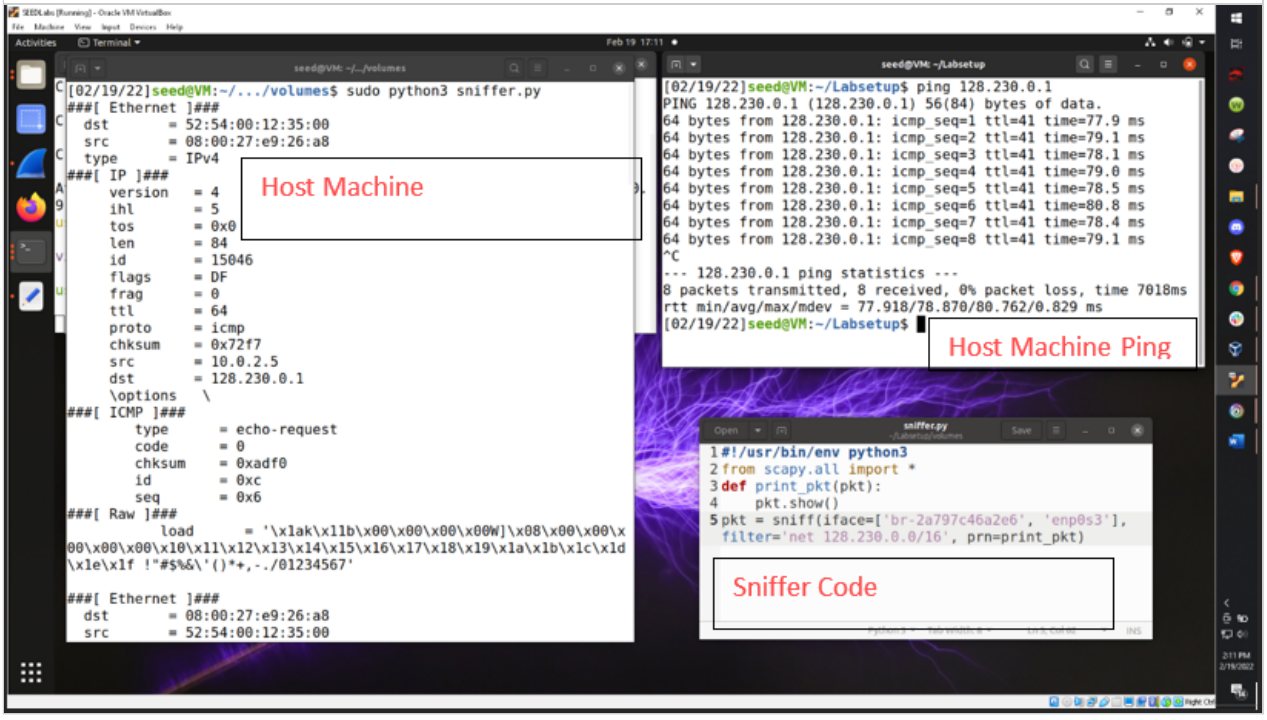
Excepted output

Note: you are not required to use the same IP as the screenshot



Submit your code and take a screenshot demonstrating that (must screenshot the entire VM along with date and time).

▾ snifferSubnetLabeled.PNG Download



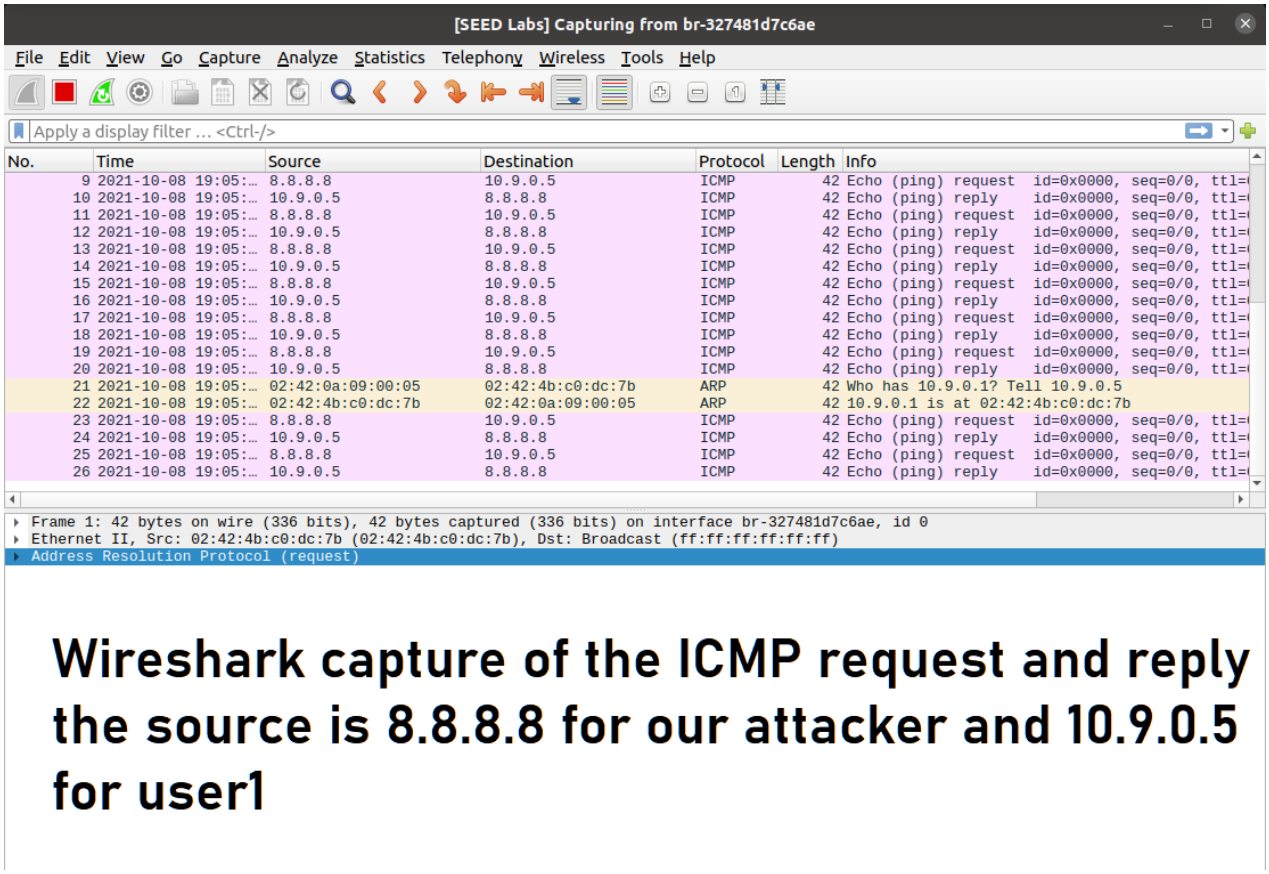
Q2 Task 1.2: Spoofing ICMP Packets

15 Points

```
>>> from scapy.all import *
>>> a = IP()
>>> a.dst = '10.0.2.3'
>>> b = ICMP()
>>> p = a/b
>>> send(p)
.
Sent 1 packets.
```

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.

Expected wireshark output



Wireshark capture of the ICMP request and reply the source is 8.8.8.8 for our attacker and 10.9.0.5 for user1

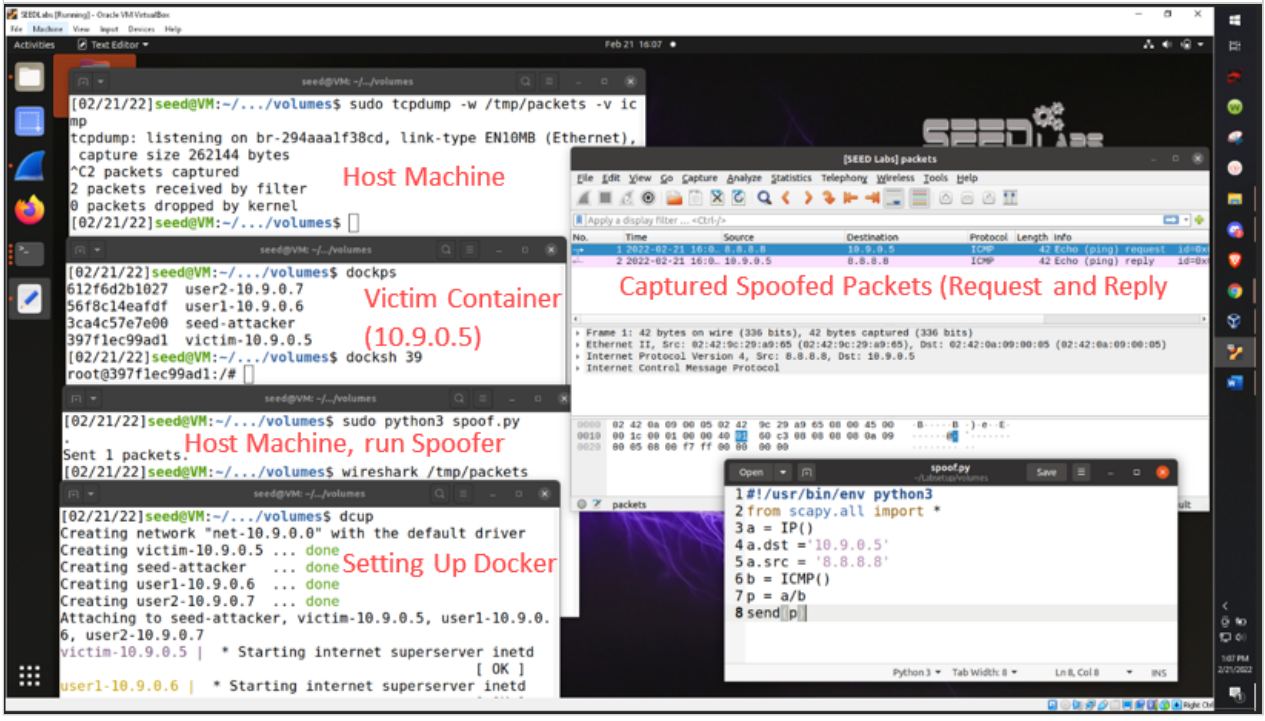
Q2.1 Task 1.2 - Submit your scapy code
10 Points

spooferpy.PNG Download

Q2.2 Task 1.2 - Submit a screenshot of Wireshark showing the spoof echo request from 8.8.8.8 and that the VM replied back to it with an echo reply.
5 Points

(must screenshot the entire VM along with date and time)

spooferlabeled.PNG Download



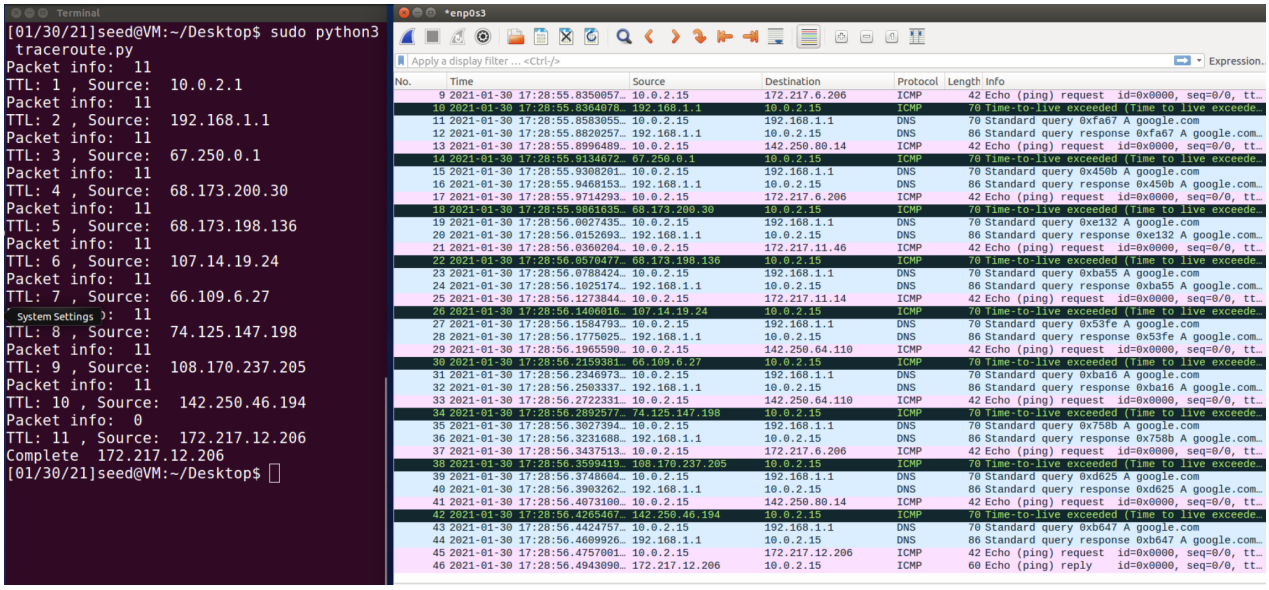
Q3 Task 1.3: Traceroute

20 Points

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 a screenshot of your program's output.

Expected output

Note: you can see we finally got a reply in the end in wireshark



Q3.1 Task 1.3 - Submit your scapy code
15 Points

▾ **traceroutepy.PNG** Download

traceroute.py
~/Labsetup/volumes

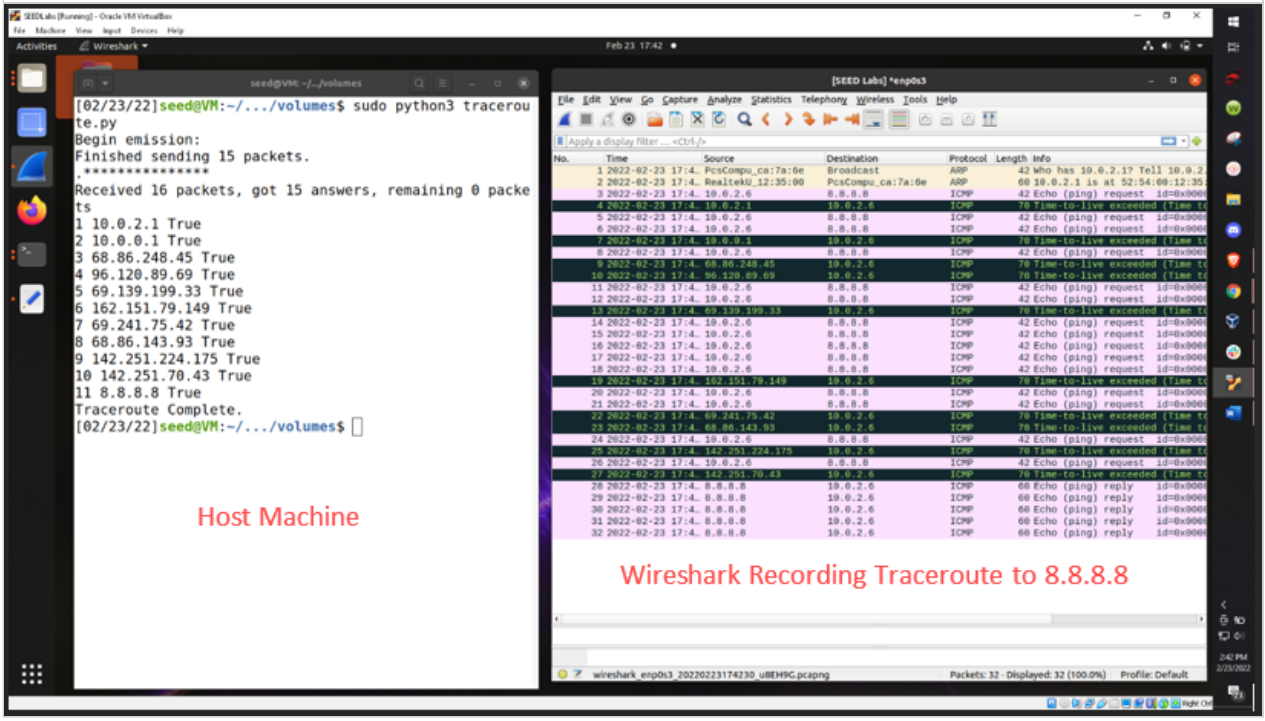
```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 a = IP()
5 a.dst = '8.8.8.8'
6 a.ttl = 1, 15
7
8 b = ICMP()
9
10 ans, unans = sr(a/b, timeout=2)
11 for snd, rcv in ans:
12         print(snd.ttl, rcv.src, isinstance(rcv.payload,
   ICMP))
13         if(rcv.src == a.dst):
14                 break
15
16 print("Traceroute Complete.")
```

Saving file "/home/seed/Labsetup/volumes/tra... Python 3 Tab Width: 8 Ln 15, Col 1 INS

Q3.2 Task 1.3 - Submit screenshot of Wireshark showing proof of traceroute to 8.8.8.8
5 Points

Submit a screenshot of Wireshark **(must screenshot the entire VM along with date and time)**

▾ **traceroutelabeled.PNG** Download



Q4 Task 1.4: Sniffing and-then Spoofing
40 Points

Sniffing and-then Spoofing. You need two machines on the same LAN: the VM and the user container.

You will find that when you ping from the terminal, 10.9.0.99 will have a destination unreachable response. That is the expected result. Your program does not need to work for this IP. (You do not need to force it to work by performing ARP spoofing.) You will, however, need to explain why your program does not work for IP 10.9.0.99 (while it's suppose to work for 1.2.3.4 and 8.8.8.8).

Q4.1 Task 1.4 - Submit your scapy code for sniffing and then spoofing.
8 Points

▾ **sniffspoof.PNG** Download

Open sniffnspoof.py ~/Labsetup/volumes Save

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 def spoof_packet(pkt):
5         # only check for echo
6         if pkt[ICMP].type != 8:
7                 return
8
9         count = 0
10
11        # insert sniffed ping packets information into spoof_packet's fields
12        ip = IP(src = pkt[IP].dst, dst = pkt[IP].src, ihl = pkt[IP].ihl)
13        icmp = ICMP (type = 0, id = pkt[ICMP].id, seq = pkt[ICMP].seq)
14        data = pkt[Raw].load
15        newpkt = ip/icmp/data
16
17        send(newpkt, verbose = 0)
18        count+= 1
19
20        print("Spoofed Packet Sent.")
21
22 while(1): # sniffer code from before, listening on Docker network and
   standard network (no Docker)
23        pkt = sniff(filter = 'icmp', iface = ['br-91e7dd8873ba','enp0s3'],
   prn=spoof_packet)
```

Python 3 Tab Width: 8 Ln 22, Col 99 INS

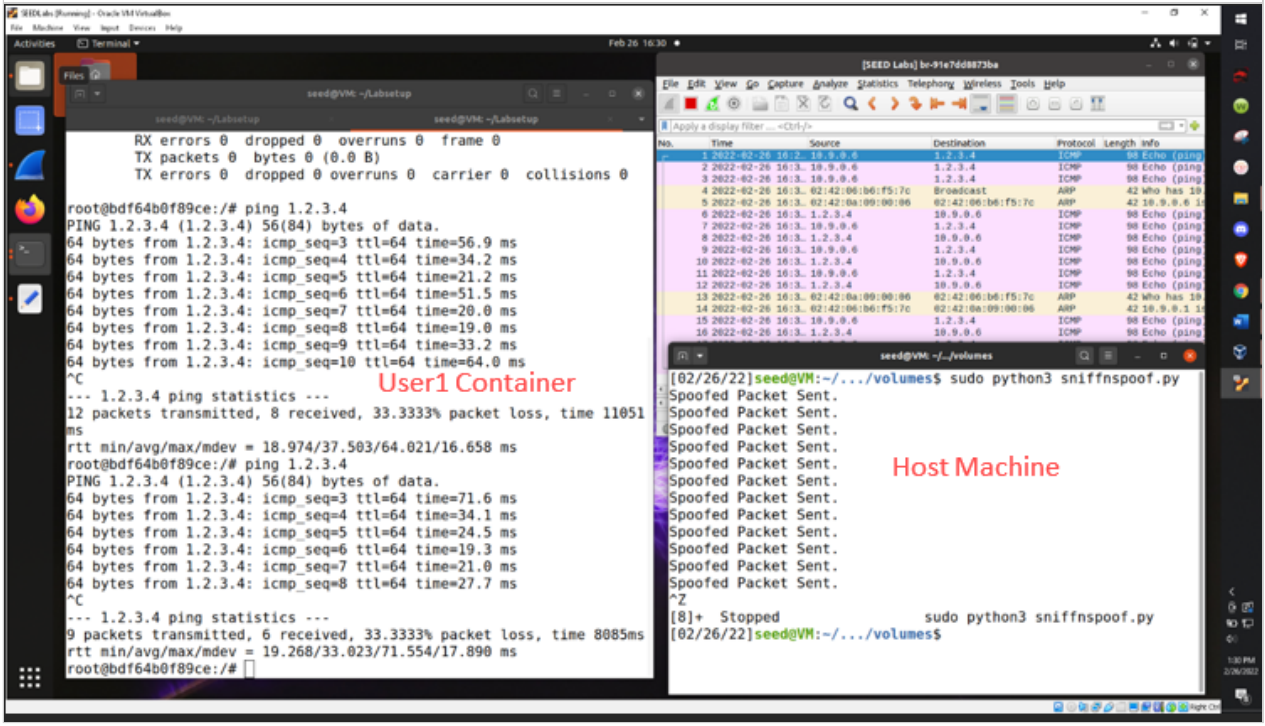
Q4.2 Task 1.4 - Ping 1.2.3.4 and show screenshots of the output from your program and with the ping from the terminal
8 Points

Expected output for 1.2.3.4

```
64 bytes from 1.2.3.4: icmp_seq=29 ttl=64 time=35.4 ms
64 bytes from 1.2.3.4: icmp_seq=30 ttl=64 time=17.0 ms
64 bytes from 1.2.3.4: icmp_seq=31 ttl=64 time=15.9 ms
64 bytes from 1.2.3.4: icmp_seq=32 ttl=64 time=28.0 ms
64 bytes from 1.2.3.4: icmp_seq=33 ttl=64 time=27.0 ms
64 bytes from 1.2.3.4: icmp_seq=34 ttl=64 time=27.5 ms
64 bytes from 1.2.3.4: icmp_seq=35 ttl=64 time=15.9 ms
64 bytes from 1.2.3.4: icmp_seq=36 ttl=64 time=19.2 ms
64 bytes from 1.2.3.4: icmp_seq=37 ttl=64 time=25.0 ms
```

Must screenshot the entire VM along with date and time

▾ sniffnspooflabeled.PNG Download



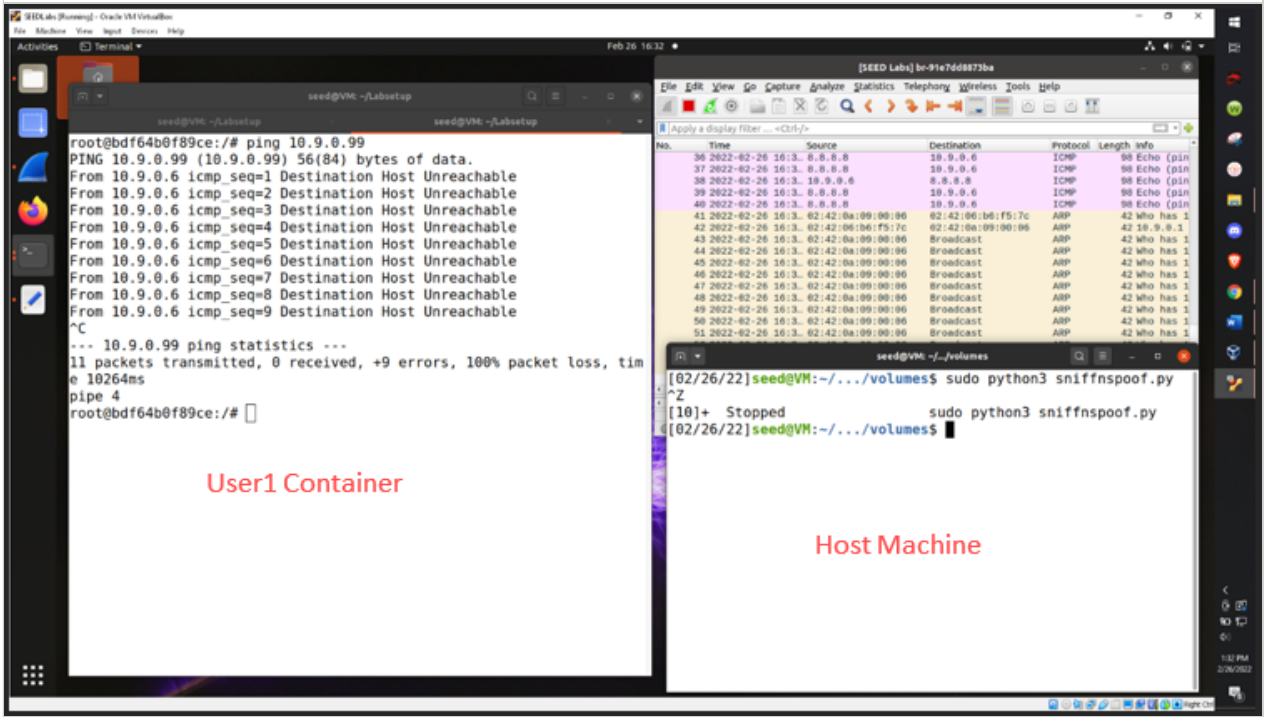
Q4.3 Task 1.4 - Ping 10.9.0.99 and show screenshots of the output from your program and with the ping from the terminal. If this does not work, please explain why.
8 Points

Expected output for 10.9.0.99

```
root@d6d2c918703b:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.7 icmp_seq=1 Destination Host Unreachable
From 10.9.0.7 icmp_seq=2 Destination Host Unreachable
From 10.9.0.7 icmp_seq=3 Destination Host Unreachable
From 10.9.0.7 icmp_seq=4 Destination Host Unreachable
From 10.9.0.7 icmp_seq=5 Destination Host Unreachable
From 10.9.0.7 icmp_seq=6 Destination Host Unreachable
```

Must screenshot the entire VM along with date and time

▾ sniffnspoofUnreachlabeled.PNG Download



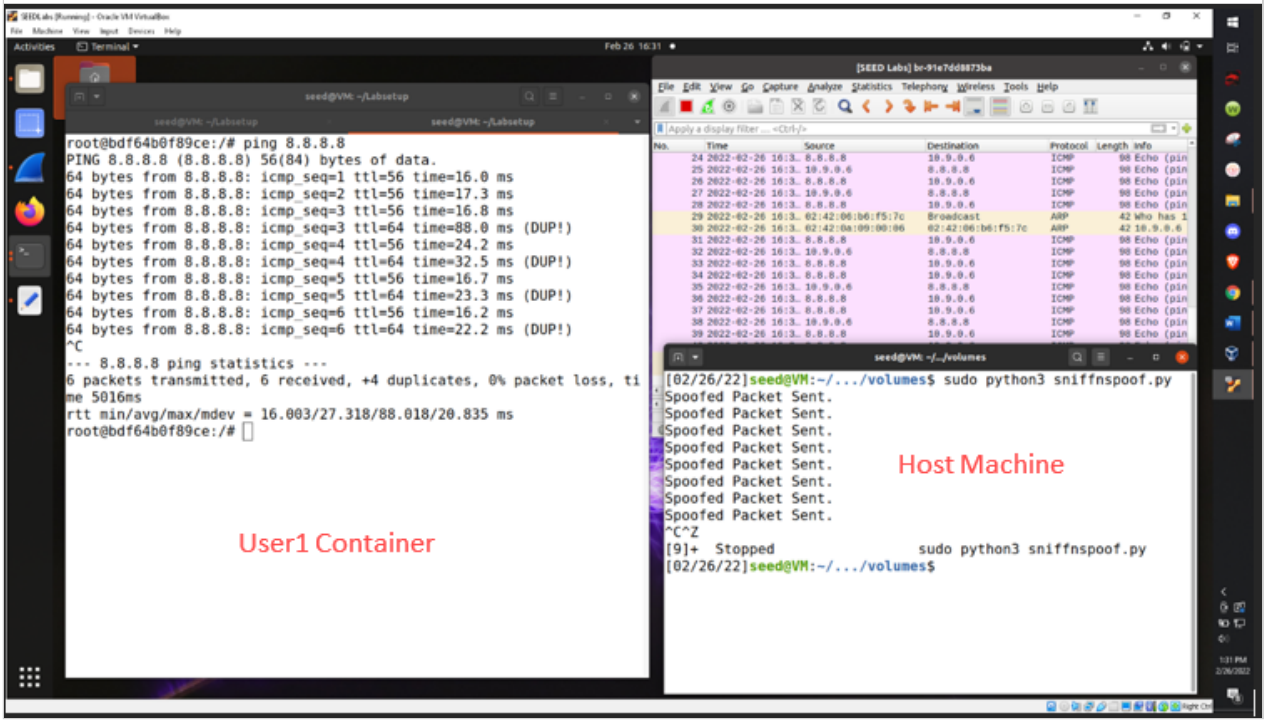
Q4.4 Task 1.4 - Ping 8.8.8.8 and show screenshots of the output from your program and with the ping from the terminal
8 Points

Expected output for 8.8.8.8

```
64 bytes from 8.8.8.8: icmp_seq=5 ttl=56 time=11.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=27.5 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=56 time=13.4 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=28.2 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=56 time=14.8 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=28.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=8 ttl=56 time=15.6 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=64 time=24.5 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=9 ttl=56 time=11.3 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=64 time=22.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=10 ttl=56 time=14.6 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=64 time=24.7 ms (DUP!)
```

Must screenshot the entire VM along with date and time

▾ sniffnspoofDUPlabeled.PNG Download



Q4.5 Task 1.4 - Explain the results for Q4.2 - Q4.4.
8 Points

Q4.2 - 1.2.3.4 is a valid address, working IP, so sniffing and spoofing packets is possible. So, I get pings from 1.2.3.4 succesfully.
Q4.3 - Upon further research and a whois for 10.9.0.99, it seems like it is a non-valid IP address, making it wholly unreachable in any way, which is why the output is unreachable.
Q4.4 - Pinging 8.8.8.8 is successful, even without the spoofed packets, so when we introduce our own craft packets, we end up getting duplicate, just like the output in the terminal denoted as "(DUP!)"

Q5 Extra Credit - Ping 10.9.0.99
5 Points

Optional: Students are encouraged to write a program that successfully allows them to ping 10.9.0.99.

In order for you to accomplish this task, you must first perform ARP cache poisoning (ARP spoofing).

Hint: Please make sure that when you use Scapy's sniff function, you appropriately configure the filter argument.

5% bonus points will be given for the optional question.

Show screenshots of the output from your program and with the ping from the terminal, and Wireshark showing the replies.
Must screenshot the entire VM along with date and time.

No files uploaded

Submit your code

No files uploaded

Provide an explanation of your ARP attack and how it changed the results from Q4.3:

Q6 Early/Late Submission Bonus
0 Points

Bonus points for early or late submission will be added here. You may submit up to five days early for an extra 5% bonus points added to the grade of this assignment, or up to 10% deducted for late submission.

Submissions more than 10 days late are not accepted without a medical or work approved reason.