Lab 2 - Packet Sniffing and Spoofing Lab

Graded

Student

Alexander Lotero

Total Points

105 / 100 pts

Question 1

(no title) **10** / 10 pts

1.1 Task 1.1A **10** / 10 pts

✔ **– 0 pts** Correct

Question 2

Task 1.1B **15** / 15 pts

✔ **– 0 pts** Correct

Question 3

Task 1.2 **15** / 15 pts

✔ **+ 15 pts** Correct

Question 4

Task 1.3 **20** / 20 pts

✔ **– 0 pts** Correct

Question 5

Task 1.4 **40** / 40 pts

✔ **– 0 pts** Correct

Question 6

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

✔ **+ 5 pts** Early submission

Q1
10 Points

Lab PDF:
https://seedsecuritylabs.org/Labs_20.04/Files/Sniffing_Spoofing/Sniffing_Spoofing.pdf
Note: For this lab, we will only do up to task 1.4, the scapy portions. 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

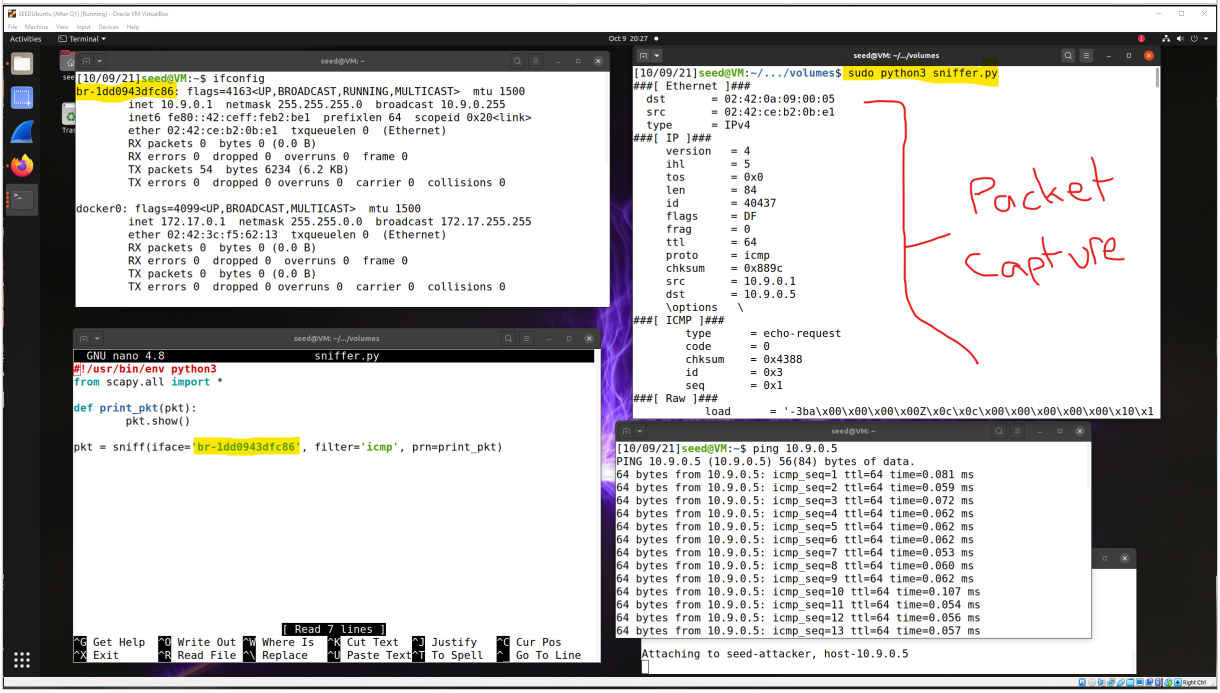
Earliest acceptance date: 14 October (+1% per day, up to +5% bonus for five days early)
Normal due date: 19 October
Latest acceptance date: 29 October (-10% points)

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

Q1.1 Task 1.1A
10 Points

Run the program with root privilege and demonstrate that you can indeed capture packets.
(1) Show a screenshot showing that packets are being captured.

Lab2Task1.1A.png Download



sniffer.py Download

```
1  #!/usr/bin/env python3
2  from scapy.all import *
3
4  def print_pkt(pkt):
5          pkt.show()
6
7  pkt = sniff(filter='net 128.230.0.0/16', prn=print_pkt)
8
```

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

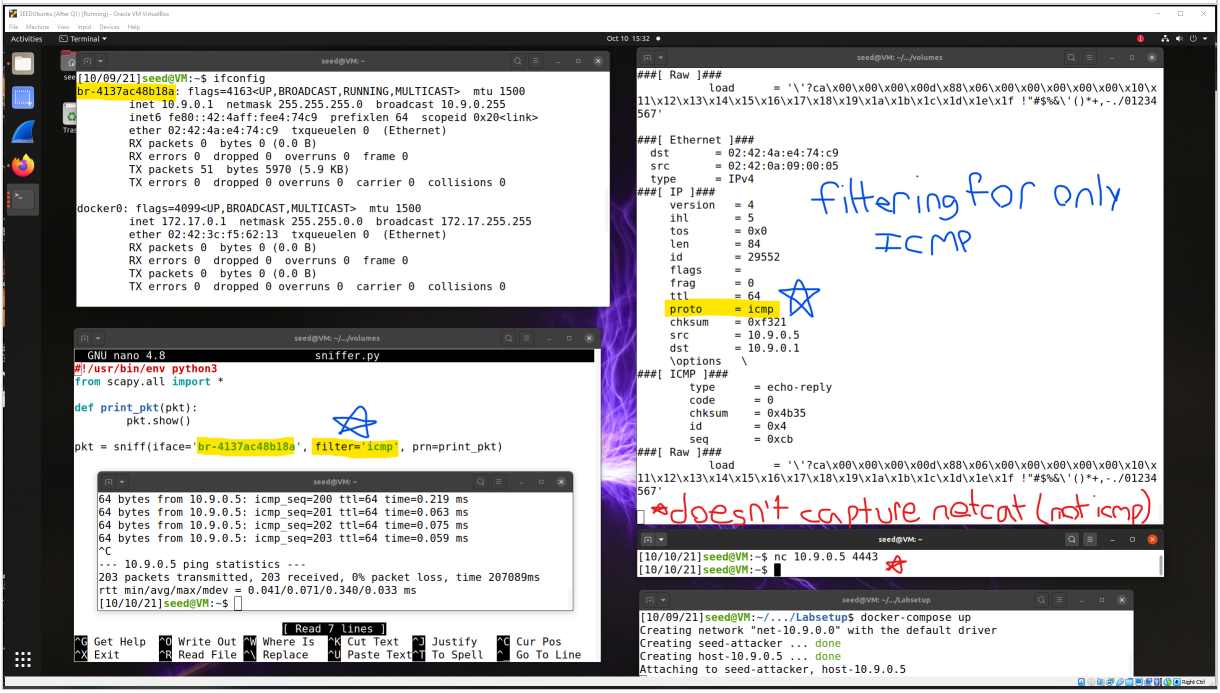
Attempting to run sniffer.py to sniff packets without root privileges results in the following error: "PermissionError: [Errno 1] Operation not permitted." This means that the current user does not have permission to sniff packets. More specifically, when not run with Administrator privileges, the OS denies python/scapy's request to analyze traffic on the socket. This request is permitted when run with Administrator privileges.

Q2 Task 1.1B

15 Points

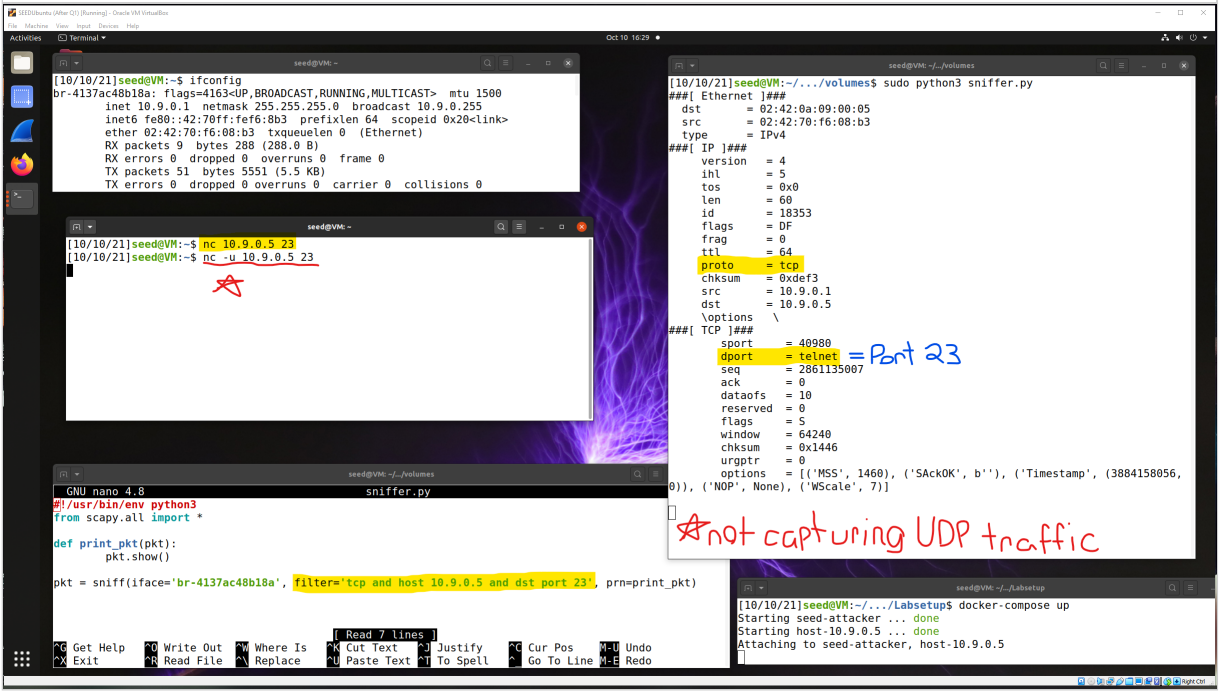
(1) Capture only the ICMP packet. Submit your code and take a screenshot demonstrating that ICMP packets are captured.

Lab2Task1.1B-1.png Download



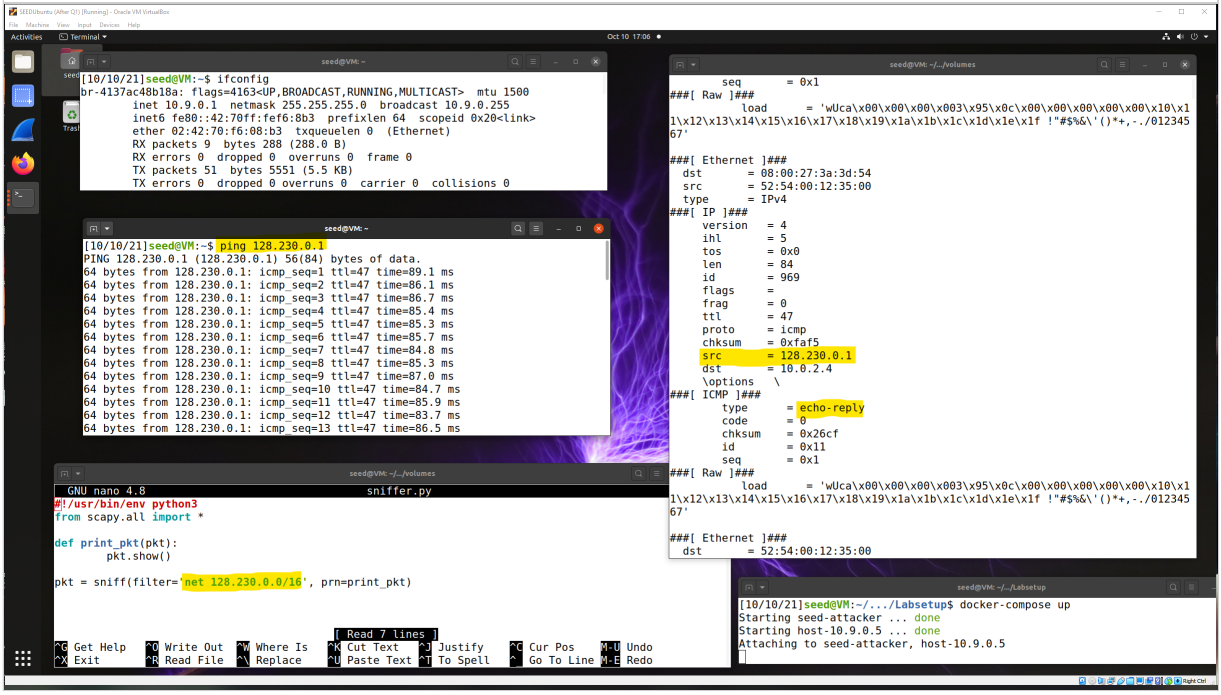
(2) Capture any TCP packet that comes from a particular IP and with a destination port number 23. Submit your code and take a screenshot demonstrating that.

Lab2Task1.1B-2.png Download



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

Lab2Task1.1B-3.png Download



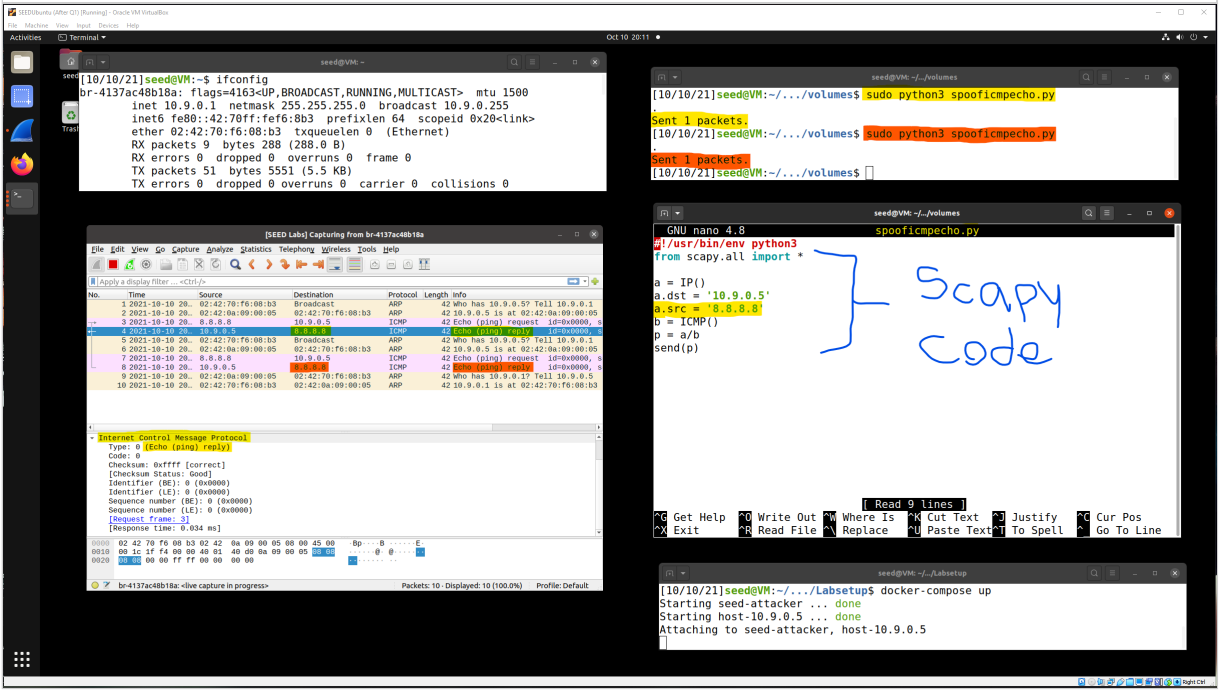
Q3 Task 1.2

15 Points

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

Lab2Task1.2-1.png Download



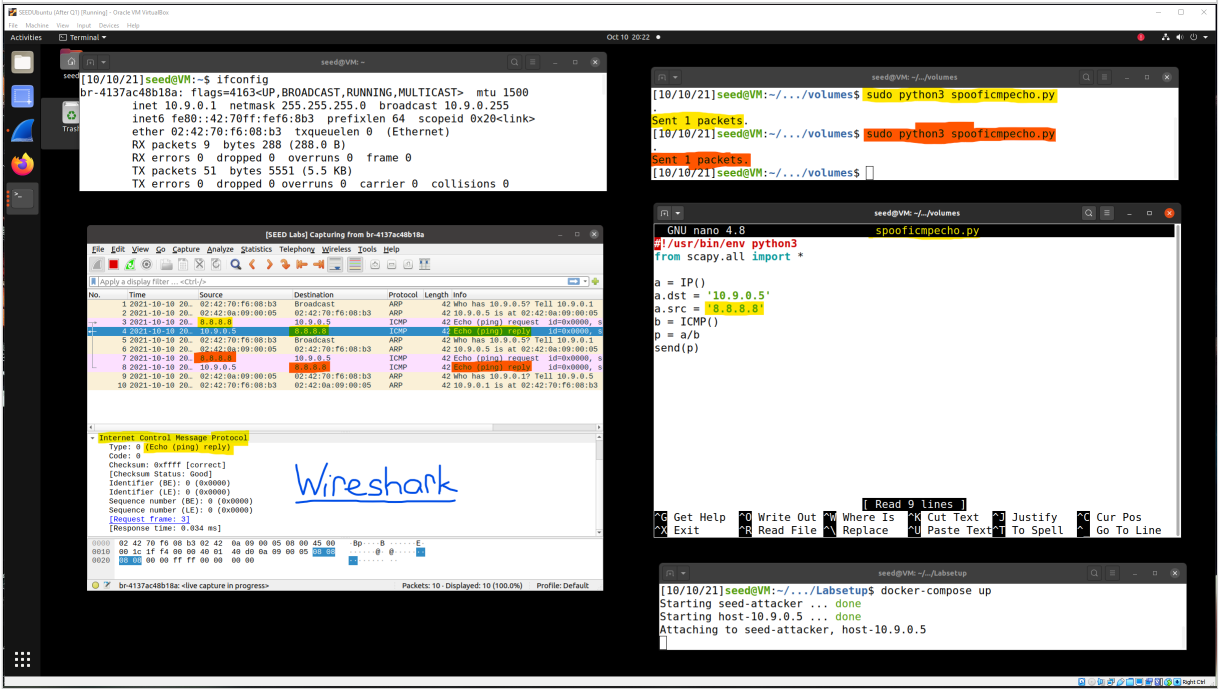
spooficmpecho.py Download

```
1  #!/usr/bin/env python3
2  from scapy.all import *
3
4  a = IP()
5  a.dst = '10.9.0.5'
6  a.src = '8.8.8.8'
7  b = ICMP()
8  p = a/b
9  send(p)
10
```

(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.

Lab2Task1.2-2.png

Download

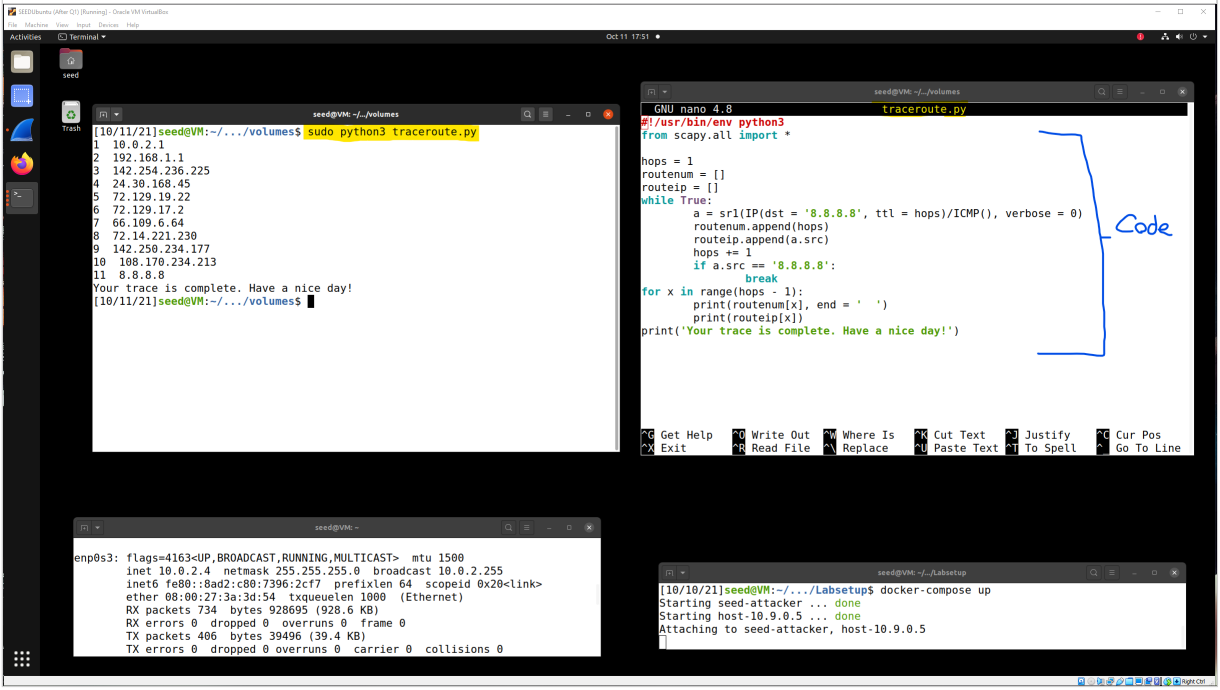


Q4 Task 1.3

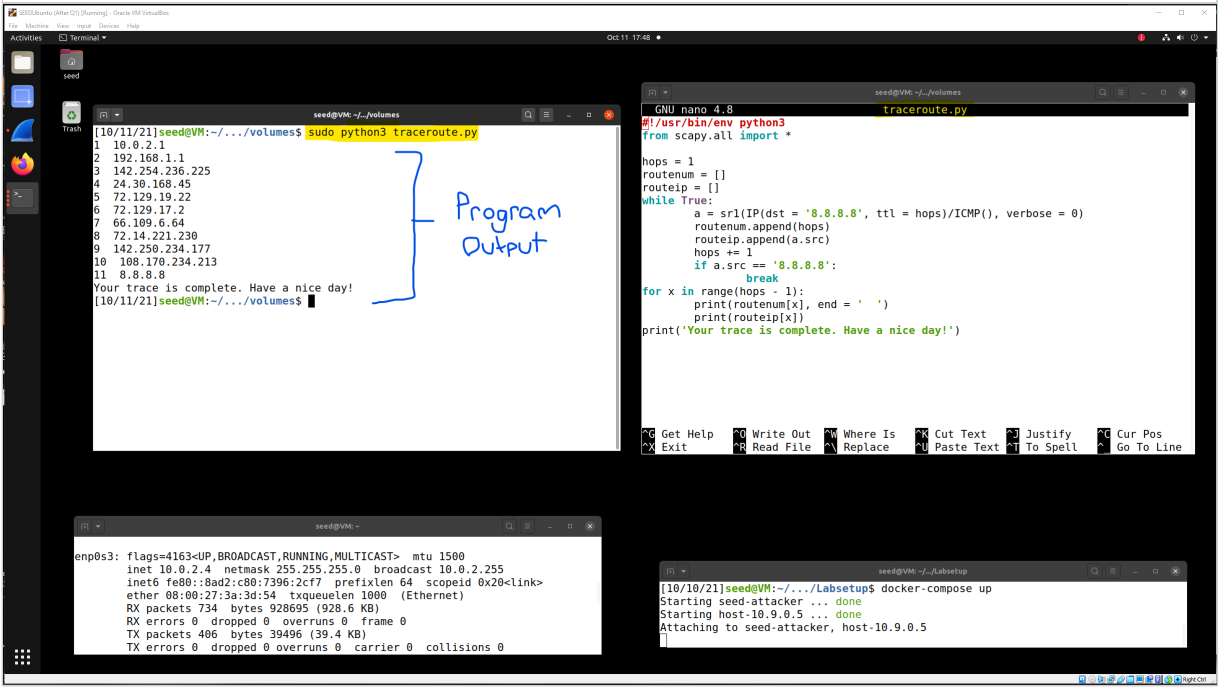
20 Points

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 screenshot of your program's output.

Lab2Task1.3-Code.png Download



Lab2Task1.3-Output.png Download



traceroute.py Download

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

hops = 1
routenum = []
routeip = []
while True:
	a = sr1(IP(dst = '8.8.8.8', ttl = hops)/ICMP(), verbose = 0)
	routenum.append(hops)
	routeip.append(a.src)
	hops += 1
	if a.src == '8.8.8.8':
		break
for x in range(hops - 1):
	print(routenum[x], end = '  ')
	print(routeip[x])
print('Your trace is complete. Have a nice day!')


```

Q5 Task 1.4
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, this IP 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).

---

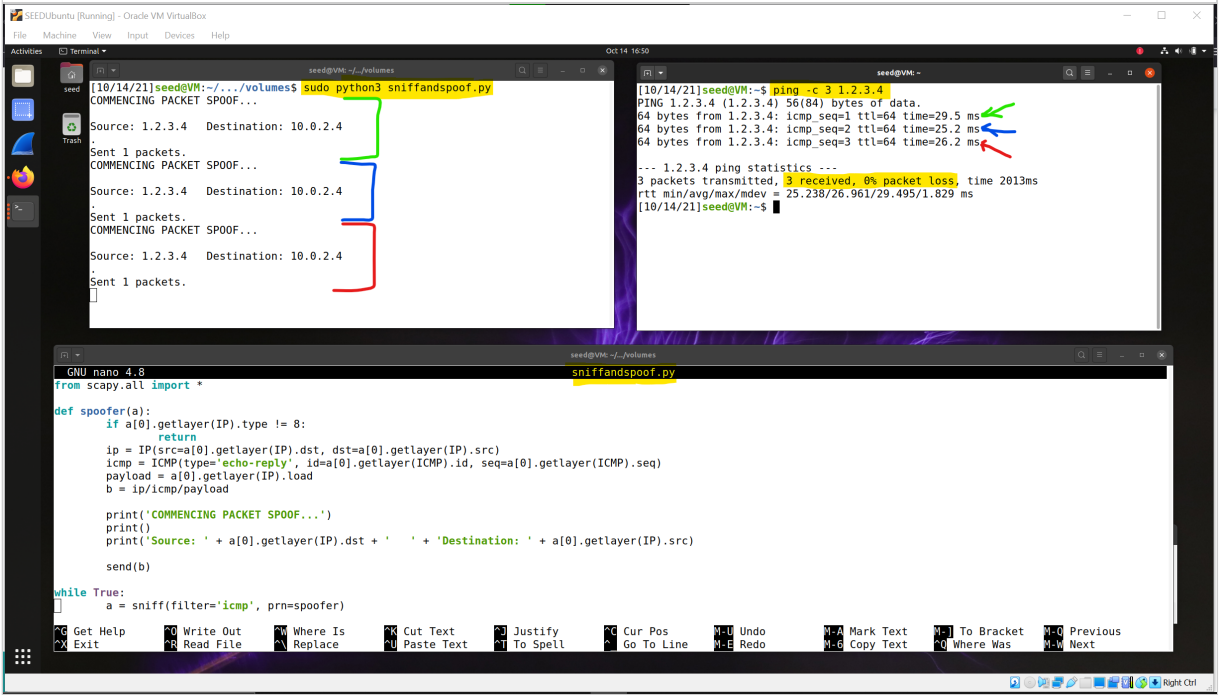
(1) Submit your scapy code for sniffing and then spoofing.

sniffandspoof.py Download

```
1  #!/usr/bin/env python3
2  from scapy.all import *
3
4  def spoofer(a):
5          if a[0].getlayer(IP).type != 8:
6                  return
7          ip = IP(src=a[0].getlayer(IP).dst, dst=a[0].getlayer(IP).src)
8          icmp = ICMP(type='echo-reply', id=a[0].getlayer(ICMP).id,
   seq=a[0].getlayer(ICMP).seq)
9          payload = a[0].getlayer(IP).load
10         b = ip/icmp/payload
11
12         print('COMMENCING PACKET SPOOF...')
13         print()
14         print('Source: ' + a[0].getlayer(IP).dst + '    ' + 'Destination: ' +
   a[0].getlayer(IP).src)
15
16         send(b)
17
18 while True:
19         a = sniff(filter='icmp', prn=spoofer)
20
21
```

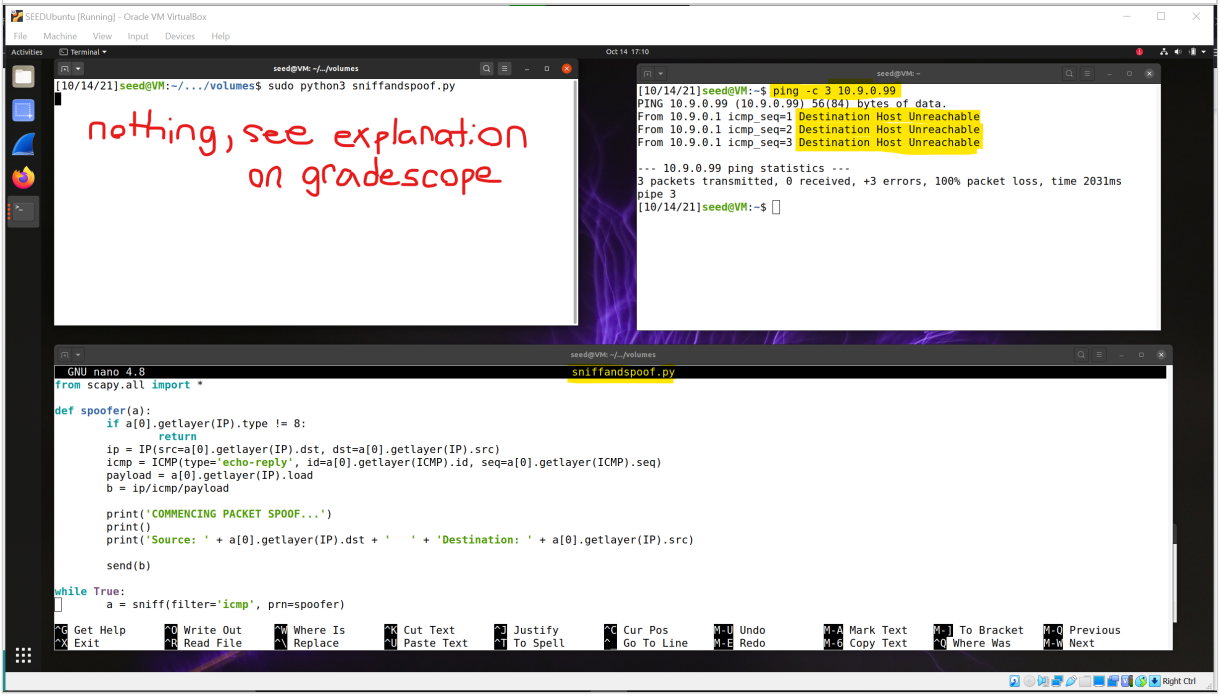
(2) Ping 1.2.3.4 and show screenshots of the output from your program and with ping from the terminal

▾ **Lab2Task1.4-2.png** Download



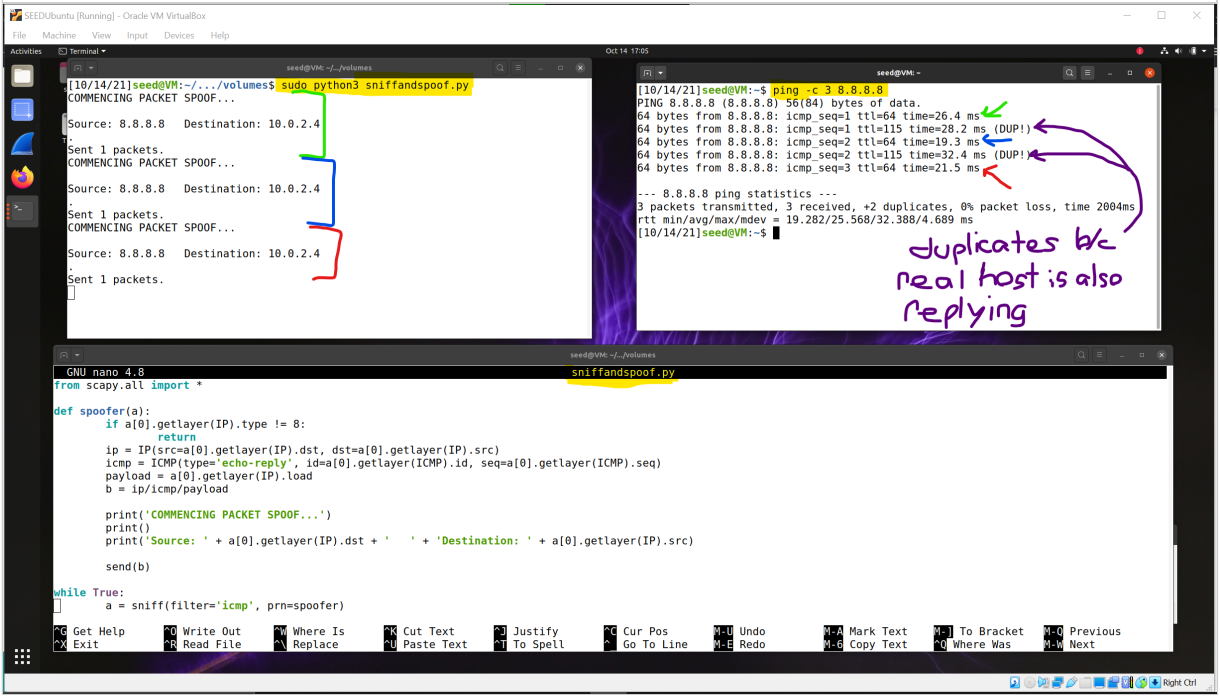
(3) Ping 10.9.0.99 and show screenshots of the output from your program and with ping from the terminal. If this does not work, please explain why.

▾ **Lab2Task1.4-3.png** Download



(4) Ping 8.8.8.8 and show screenshots of the output from your program and with ping from the terminal

Lab2Task1.4-4.png Download



(5) Explain the results for (2), (3), and (4).

Ping 1.2.3.4 is best example of the sniffandspoof.py program in action. The 1.2.3.4 host is not actually live on the internet, so without our program the ping is unable to reach the host and receives 100% packet loss. However, with our program running, the ping receives every icmp echo-reply packet that it is expecting. These packets are spoofed of course, but to the terminal running ping, our program makes it appear as though the 1.2.3.4 host is live and responding (see screenshot).

Ping 10.9.0.99 does not work because while the address is within the LAN, it is not actually an existing host. This result is explained by the ARP protocol. Firstly, our device will check its ARP cache for the 10.9.0.99 device's MAC address. There is no MAC address stored in our ARP cache for this address because we have not communicated with it before. Next, our device will issue an ARP request broadcast across the LAN via the router at 10.9.0.1 looking for the MAC address of the device at 10.9.0.99. No such device exists, so our machine will never resolve that address and therefore our ping will fail with the "Destination Host Unreachable" error (see screenshot).

As for 8.8.8.8, this ping, and our program's sniff and spoof work without a problem. That said, because 8.8.8.8 is an actual live host, we see duplicate reply packets in our ping output. This is because the ping terminal is first receiving a spoofed reply from our program followed by the real reply from the destination host (see screenshot).

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.