

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.

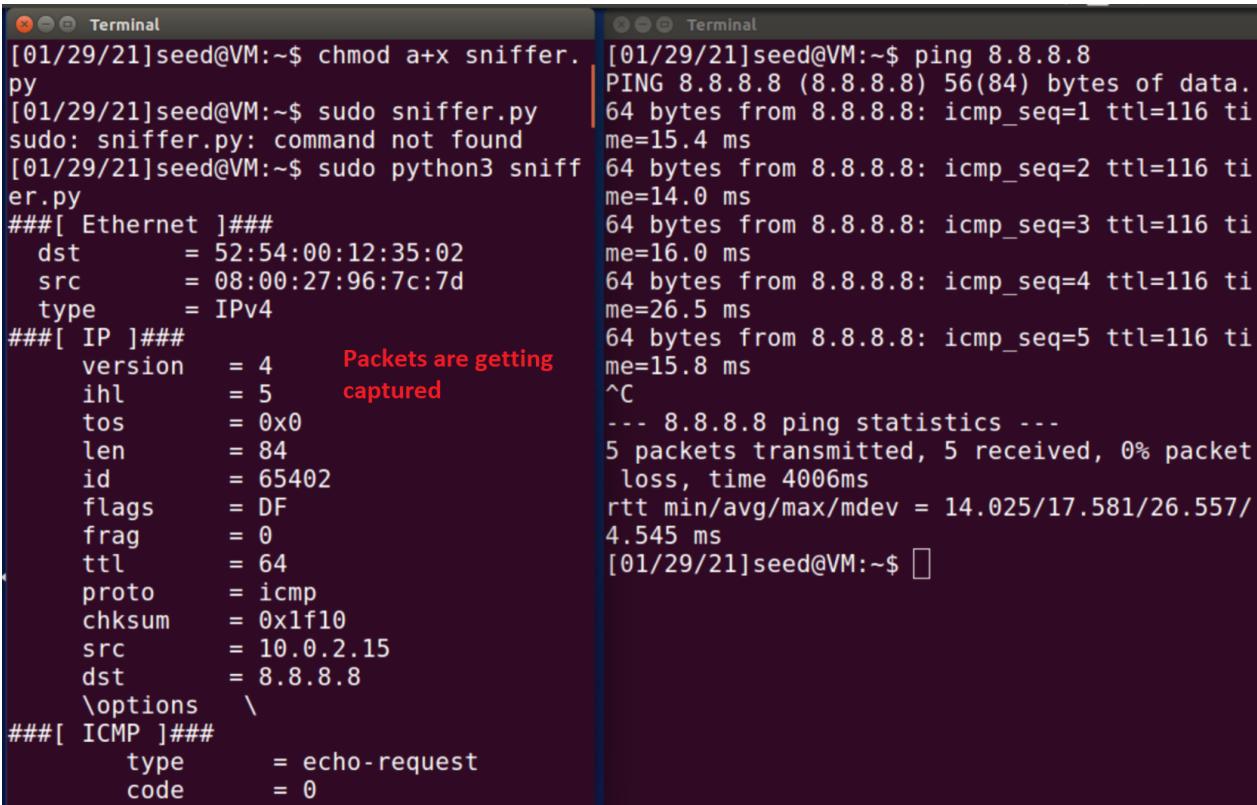
Q1.1 Task 1.1: Sniffing Packets - Task 1.1A

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



The image shows two terminal windows side-by-side. The left terminal window shows the command-line steps to run the sniffer.py script. The right terminal window shows the actual output of the script, which includes details about captured ICMP packets and a ping statistics summary.

Terminal 1 (Left):

```
[01/29/21]seed@VM:~$ chmod a+x sniffer.py
[01/29/21]seed@VM:~$ sudo sniffer.py
sudo: sniffer.py: command not found
[01/29/21]seed@VM:~$ sudo python3 sniffer.py
###[ Ethernet ]###
dst      = 52:54:00:12:35:02
src      = 08:00:27:96:7c:7d
type     = IPv4
###[ IP ]###
version   = 4      Packets are getting captured
ihl       = 5
tos       = 0x0
len       = 84
id        = 65402
flags     = DF
frag      = 0
ttl       = 64
proto     = icmp
chksum    = 0x1f10
src       = 10.0.2.15
dst       = 8.8.8.8
\options  \
###[ ICMP ]###
type      = echo-request
code     = 0
```

Terminal 2 (Right):

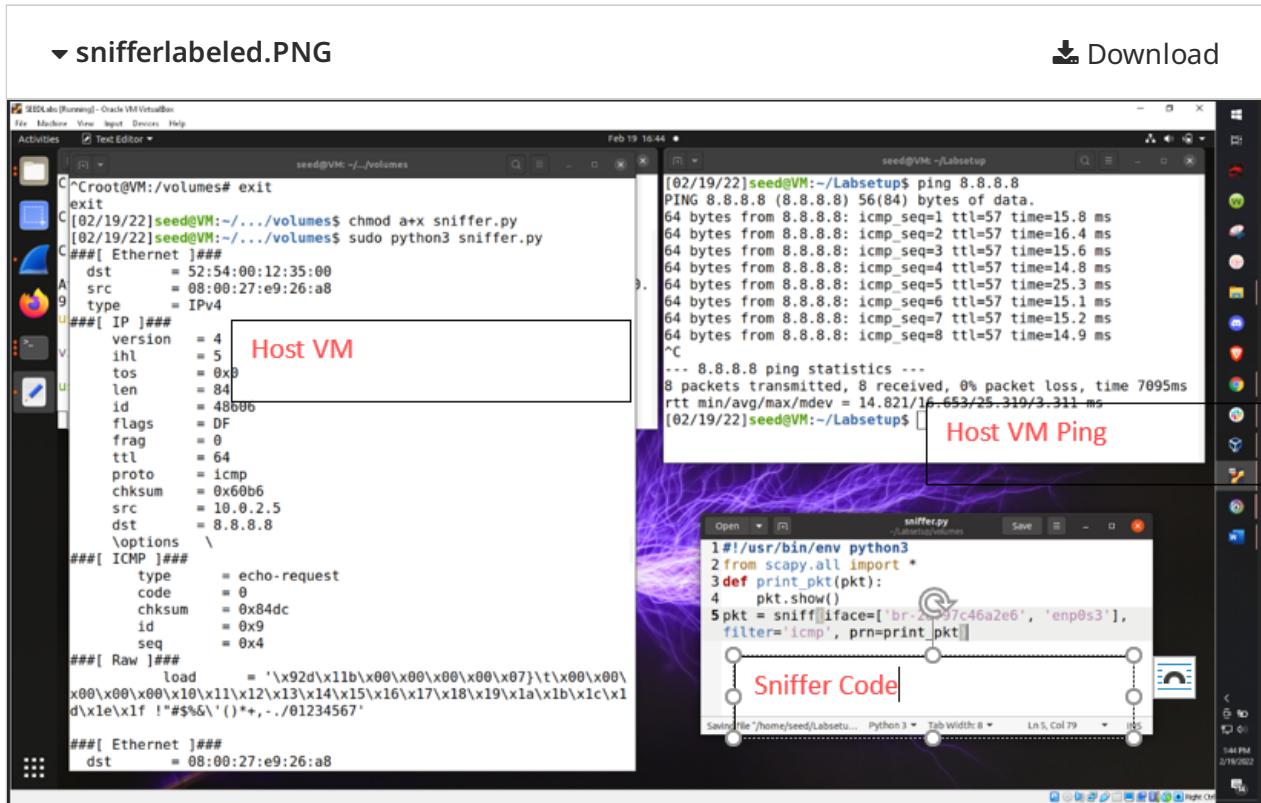
```
[01/29/21]seed@VM:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=15.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=14.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=16.0 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=26.5 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=15.8 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 14.025/17.581/26.557/4.545 ms
[01/29/21]seed@VM:~$ 
```

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)



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

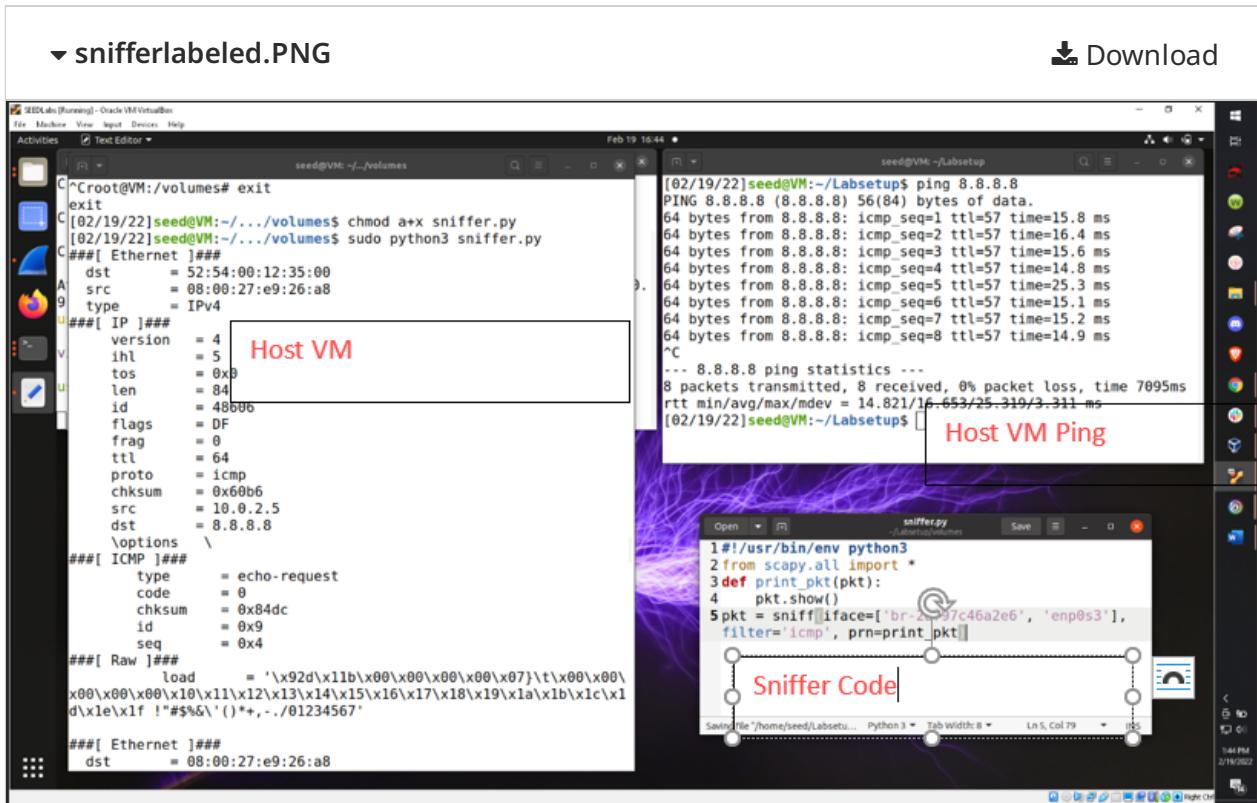
Capture only the ICMP packet.

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).



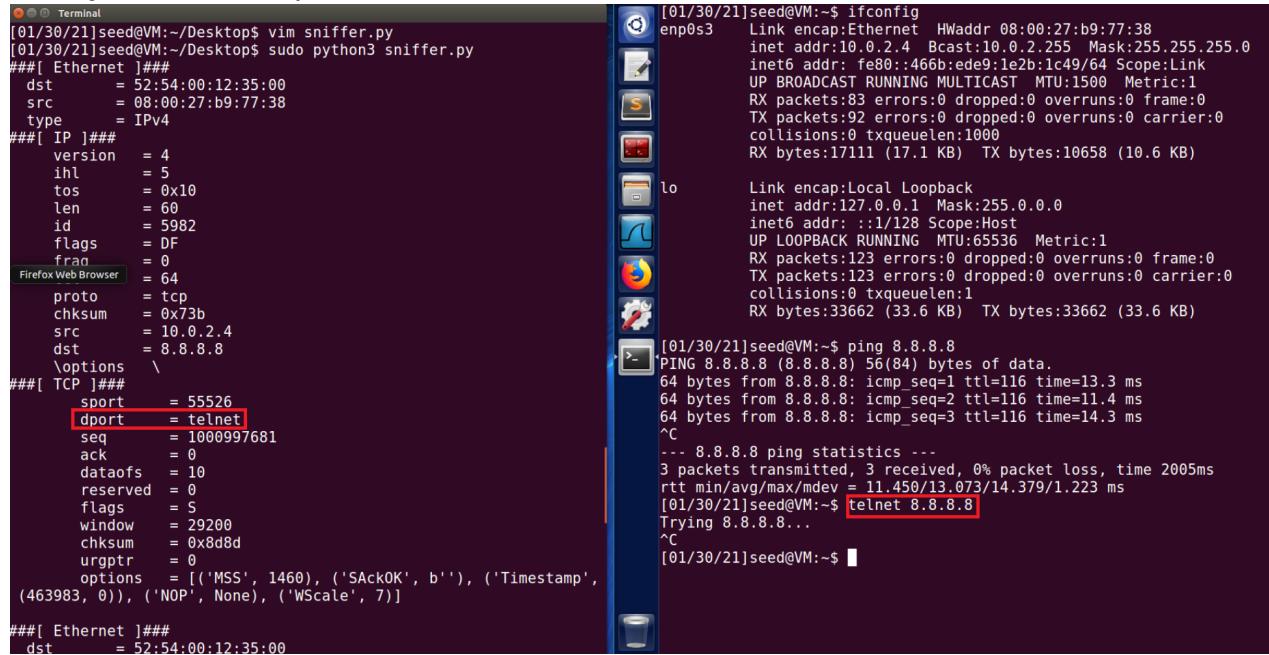
Q1.3 Task 1.1B - Any TCP

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



The screenshot shows a terminal window with several tabs open. The current tab displays the output of the 'ifconfig' command, showing two interfaces: 'enp0s3' (Ethernet) and 'lo' (Loopback). The 'enp0s3' interface has an IP of 10.0.2.4 and is connected to a host with IP 127.0.0.1 via port 80. The 'lo' interface has an IP of 127.0.0.1. Below this, the output of the 'ping' command to 8.8.8.8 is shown. A red box highlights the 'dport' value in the TCP dump output, which is set to 'telnet'. Another red box highlights the 'telnet 8.8.8.8' command being typed in the terminal.

```
[01/30/21]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:b9:77:38
            inet  addr: 10.0.2.4    Bcast: 10.0.2.255  Mask: 255.255.255.0
            inet6 addr: fe80::466b:ede9:le2b:1c49/64 Scope:Link
                      UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
                      RX packets:83 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:1000
                      RX bytes:17111 (17.1 KB)  TX bytes:10658 (10.6 KB)

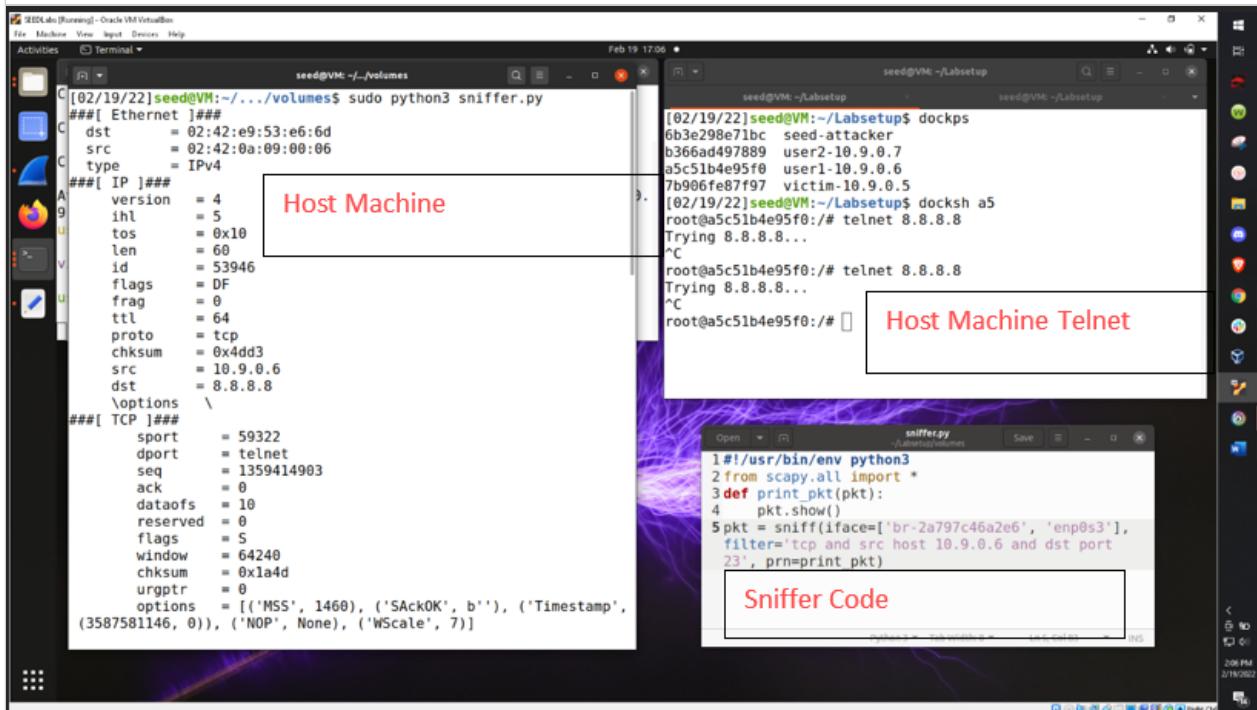
lo          Link encap:Local Loopback
            inet  addr: 127.0.0.1    Mask: 255.0.0.0
            inet6 addr: ::1/128 Scope:Host
                      UP LOOPBACK RUNNING MTU:65536 Metric:1
                      RX packets:123 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:123 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:1
                      RX bytes:33662 (33.6 KB)  TX bytes:33662 (33.6 KB)

[01/30/21]seed@VM:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=13.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=11.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=14.3 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 11.450/13.073/14.379/1.223 ms
[01/30/21]seed@VM:~$ telnet 8.8.8.8
Trying 8.8.8.8...
^C
[01/30/21]seed@VM:~$
```

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

▼ snifferTCP labeled.PNG

 Download



Host Machine

```
[02/19/22]seed@VM:~/.../volumes$ sudo python3 sniffer.py
###[ Ethernet ]##
dst      = 02:42:e9:53:e6:6d
src      = 02:42:0a:09:00:06
type     = IPv4
###[ IP ]##
version  = 4
ihl      = 5
tos      = 0x10
len      = 60
id       = 53946
flags    = DF
frag     = 0
ttl      = 64
proto    = tcp
chksum   = 0x4dd3
src      = 10.9.0.6
dst      = 8.8.8.8
options   \
###[ TCP ]##
sport    = 59322
dport    = telnet
seq      = 1359414903
ack      = 0
dataofs  = 10
reserved = 0
flags    = S
window   = 64240
checksum = 0x1a4d
urgptr   = 0
options   = [('MSS', 1460), ('SACKOK', b''), ('Timestamp', (3587581146, 0)), ('NOP', None), ('WScale', 7)]
```

Host Machine Telnet

```
[02/19/22]seed@VM:~/Labsetup$ dockps
6b3e298e71bc  seed-attacker
b366ad497889  user2-10.9.0.7
a5c51b4e95f0  user1-10.9.0.6
7b906fe87f97  victim-10.9.0.5
[02/19/22]seed@VM:~/Labsetup$ docksh a5
root@a5c51b4e95f0:/# telnet 8.8.8.8
Trying 8.8.8.8...
^C
root@a5c51b4e95f0:/# telnet 8.8.8.8...
Trying 8.8.8.8...
^C
root@a5c51b4e95f0:/#
```

Sniffer Code

```
#!/usr/bin/env python
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface=["br-2a797c46a2e6", 'enp0s3'],
            filter='tcp and src host 10.9.0.6 and dst port 23',
            prn=print_pkt)
```

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.

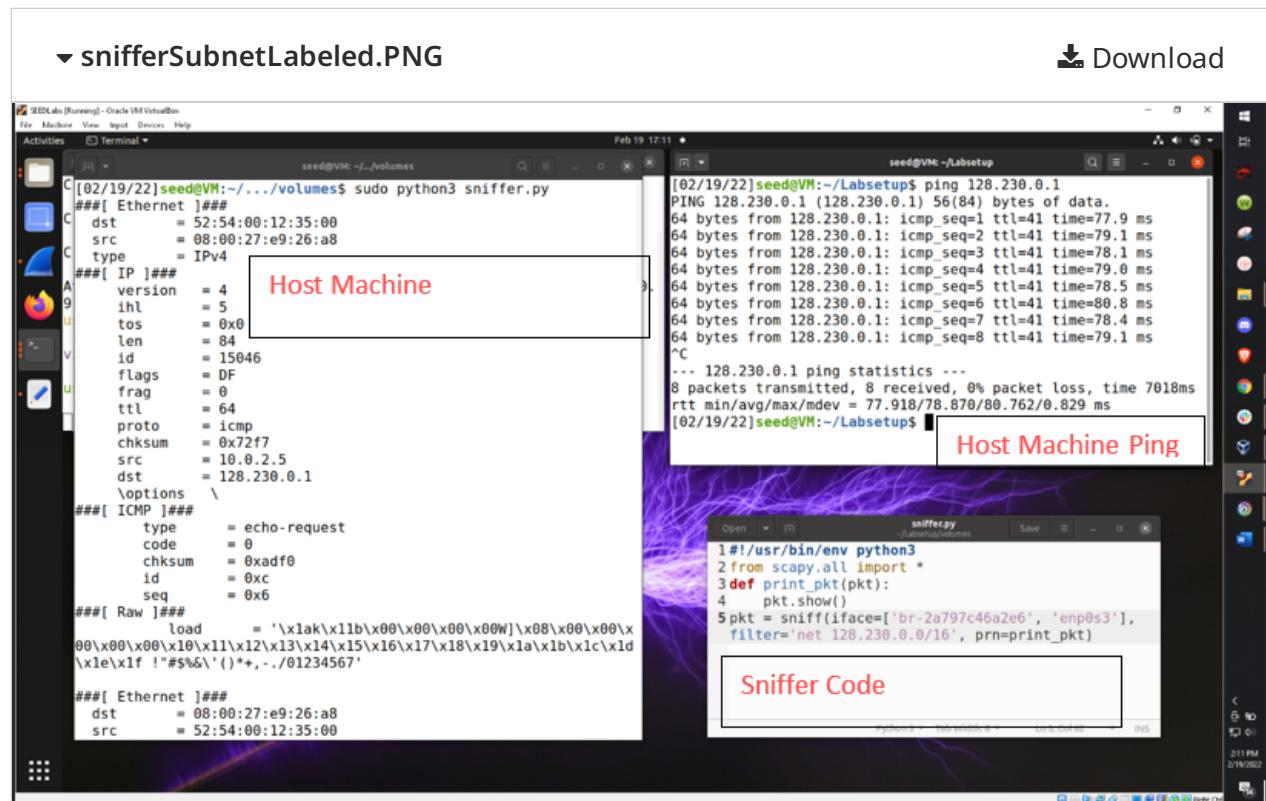
Expected output

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

The screenshot shows a terminal window with two panes. The left pane displays the output of the Python script 'sniffer.py'. A red box highlights the source IP address 'src = 128.230.61.170' in one of the captured packets. The right pane shows the results of pinging '8.8.8.8' and '128.230.1.2'. A red arrow points from the highlighted source IP in the left pane to the 'ping 8.8.8.8' command in the right pane, indicating that the captured packet is being sent to the public internet.

```
^C[01/30/21]seed@VM:~/Desktop$ vim sniffer.py
[01/30/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
dst      = 08:00:27:b9:77:38
src      = 52:54:00:12:35:00
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 56
id       = 35
flags    =
frag    =
ttl     = 54
proto   = icmp
chksum  = 0xbade
src     = 128.230.61.170
dst     = 10.0.2.4
options \
###[ ICMP ]###
type     = dest-unreach
code    = host-unreachable
checksum = 0xd69a
reserved = 0
length   = 0
nexthopmtu= 0
###[ IP in ICMP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 25511
flags    = DF
frag    =
ttl     = 64
[01/30/21]seed@VM:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=13.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=11.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=14.3 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 0 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 11.450/13.073/14.379/1.223 ms
[01/30/21]seed@VM:~$ telnet 8.8.8.8
Trying 8.8.8.8...
^C
[01/30/21]seed@VM:~$ ping 128.230.1.2
PING 128.230.1.2 (128.230.1.2) 56(84) bytes of data.
From 128.230.61.170 icmp_seq=1 Destination Host Unreachable
From 128.230.61.170 icmp_seq=2 Destination Host Unreachable
From 128.230.61.170 icmp_seq=3 Destination Host Unreachable
^C
--- 128.230.1.2 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, tim
e 5083ms
pipe 3
[01/30/21]seed@VM:~$
```

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



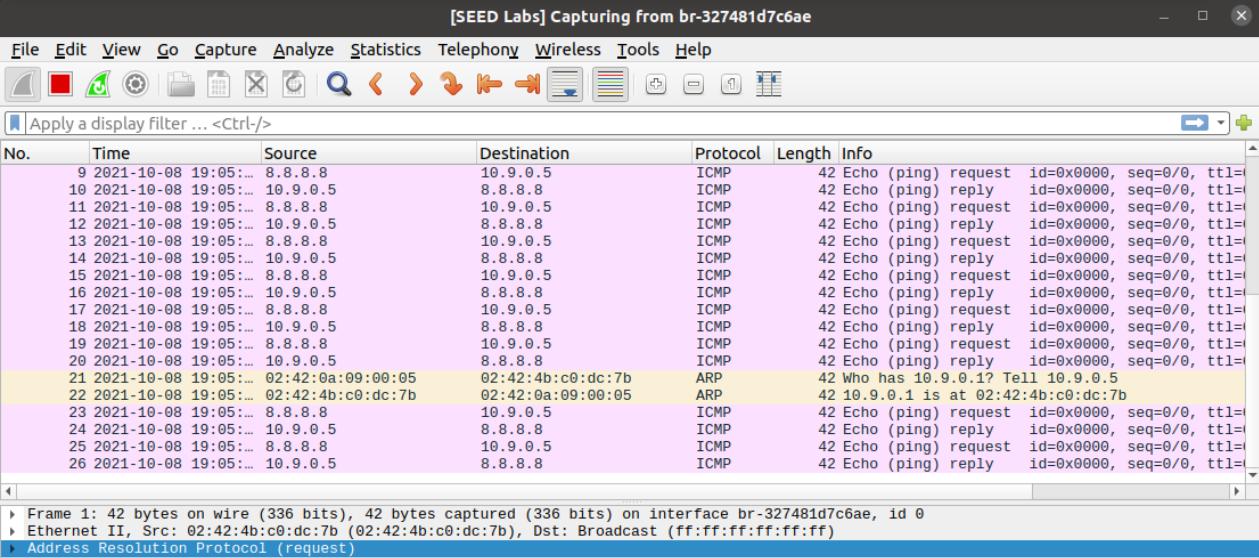
Q2 Task 1.2: Spoofing ICMP Packets

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



The screenshot shows a Wireshark capture window titled "[SEED Labs] Capturing from br-327481d7c6ae". The interface is set to "br-327481d7c6ae". The packet list shows 26 ICMP packets. The first 25 are echo requests (ping) from source 8.8.8.8 to destination 10.9.0.5. The 26th packet is an ARP request from source 02:42:4b:c0:dc:7b to destination broadcast (ff:ff:ff:ff:ff:ff), asking for the MAC address of 10.9.0.1. The details and bytes panes below the list pane show the structure of these packets.

No.	Time	Source	Destination	Protocol	Length	Info
9	2021-10-08 19:05:...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
10	2021-10-08 19:05:...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=1
11	2021-10-08 19:05:...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
12	2021-10-08 19:05:...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=1
13	2021-10-08 19:05:...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
14	2021-10-08 19:05:...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=1
15	2021-10-08 19:05:...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
16	2021-10-08 19:05:...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=1
17	2021-10-08 19:05:...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
18	2021-10-08 19:05:...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=1
19	2021-10-08 19:05:...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
20	2021-10-08 19:05:...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=1
21	2021-10-08 19:05: 02:42:0a:09:00:05		02:42:4b:c0:dc:7b	ARP	42	Who has 10.9.0.1? Tell 10.9.0.5
22	2021-10-08 19:05: 02:42:4b:c0:dc:7b		02:42:0a:09:00:05	ARP	42	10.9.0.1 is at 02:42:4b:c0:dc:7b
23	2021-10-08 19:05:...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
24	2021-10-08 19:05:...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=1
25	2021-10-08 19:05:...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
26	2021-10-08 19:05:...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=1

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

▼ spoofery.PNG [Download](#)

The screenshot shows a code editor window with the file name 'spoof.py' at the top. The code is a simple Scapy script:

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

Below the code editor, there are tabs for 'Python 3', 'Tab Width: 8', 'Ln 5, Col 18', and 'INS'.

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)

▼ spoofrlabeled.PNG [Download](#)

The screenshot displays a Linux desktop environment with several windows open:

- A terminal window titled "Host Machine" showing the command `sudo tcpdump -w /tmp/packets -v icmp` and its output.
- A terminal window titled "Victim Container" showing the command `docks` and its output.
- A terminal window titled "Host Machine, run Spoof" showing the command `sudo python3 spoof.py` and its output.
- A terminal window titled "Setting Up Docker" showing the command `dcup` and its output.
- A Wireshark window titled "Captured Spoofed Packets (Request and Reply)" showing two captured ICMP packets: an echo request from 8.8.8.8 to 10.9.0.5 and an echo reply from 10.9.0.5 to 8.8.8.8.
- A code editor window titled "spoof.py" showing the same Scapy script as in Q2.1.

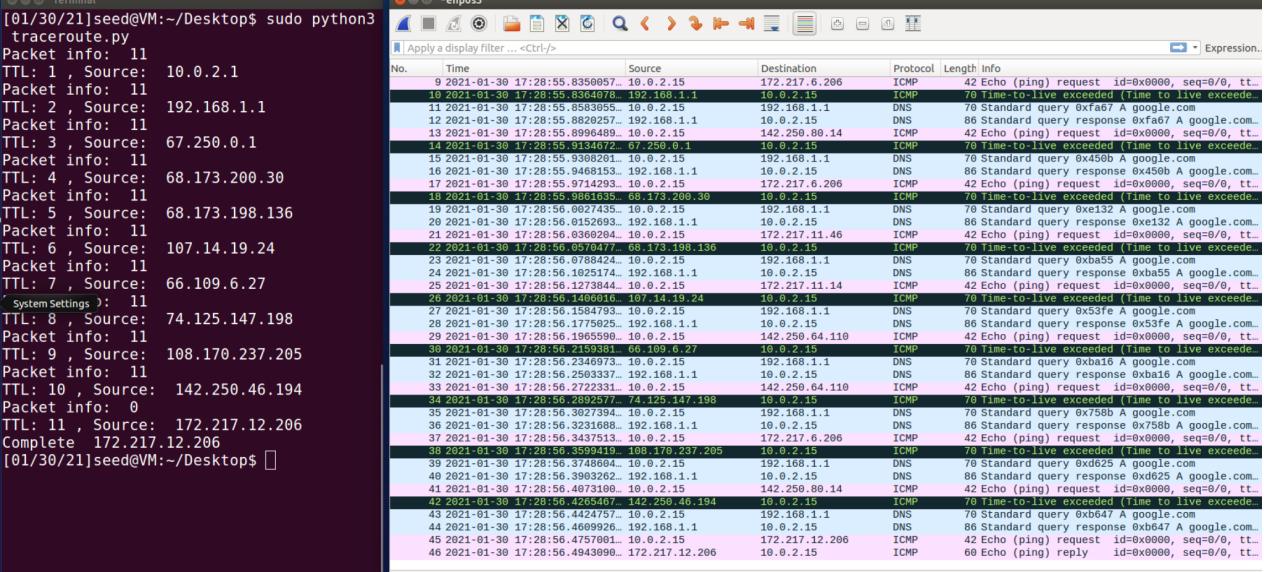
Q3 Task 1.3: Traceroute

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

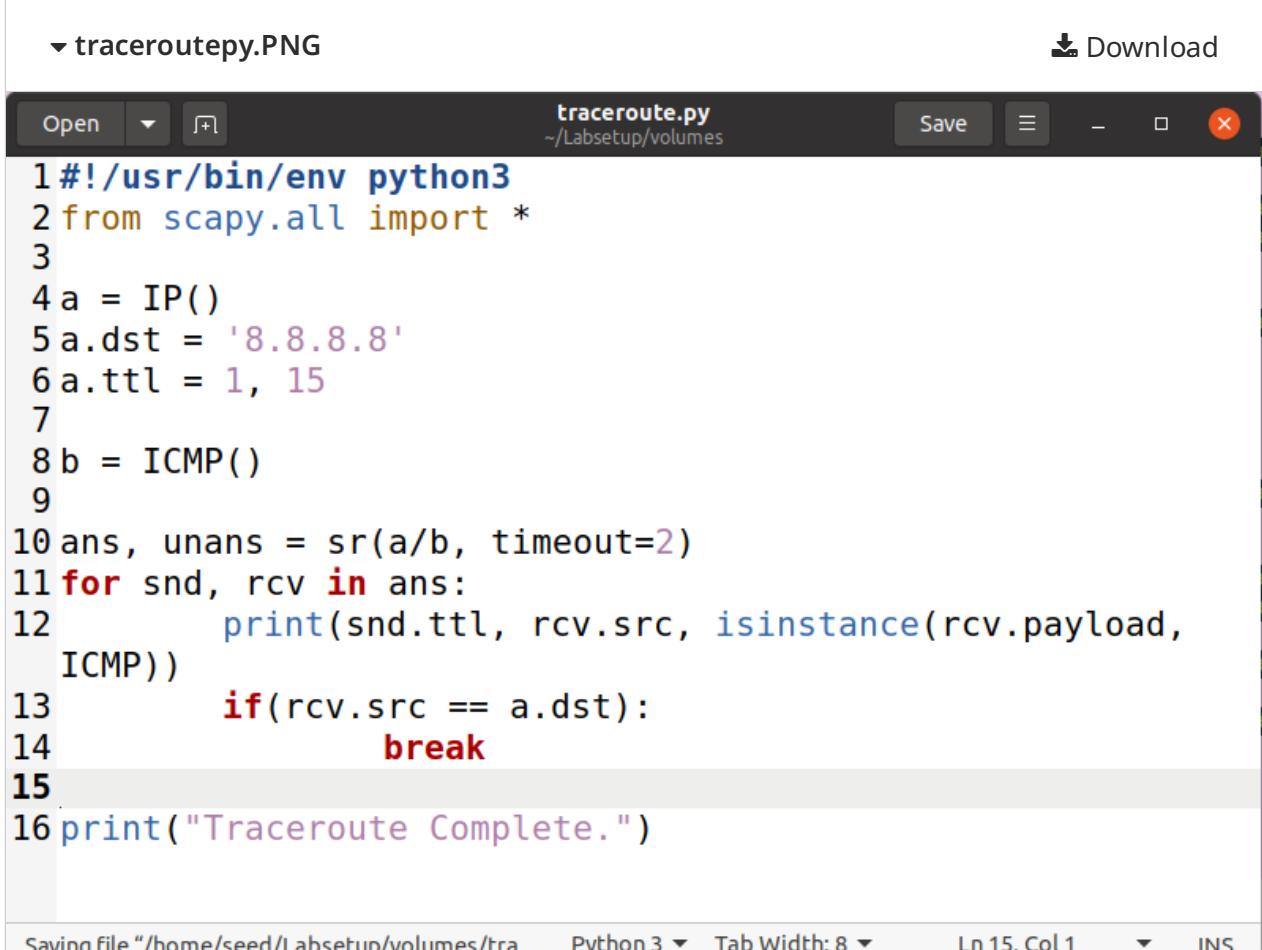


The screenshot shows a terminal window and a Wireshark capture window. The terminal window displays the execution of the Python script 'traceroute.py' to perform a traceroute to the IP address 8.8.8.8. The Wireshark window shows the network traffic, including ICMP echo requests and responses, and DNS queries, illustrating the path taken by the traceroute packets through various routers and the final destination.

```
[01/30/21]seed@VM:~/Desktop$ sudo python3 traceroute.py
Packet info: 11
TTL: 1 , Source: 10.0.2.1
Packet info: 11
TTL: 2 , Source: 192.168.1.1
Packet info: 11
TTL: 3 , Source: 67.250.0.1
Packet info: 11
TTL: 4 , Source: 68.173.200.30
Packet info: 11
TTL: 5 , Source: 68.173.198.136
Packet info: 11
TTL: 6 , Source: 107.14.19.24
Packet info: 11
TTL: 7 , Source: 66.109.6.27
System Settings : 11
TTL: 8 , Source: 74.125.147.198
Packet info: 11
TTL: 9 , Source: 108.170.237.205
Packet info: 11
TTL: 10 , Source: 142.250.46.194
Packet info: 0
TTL: 11 , Source: 172.217.12.206
Complete 172.217.12.206
[01/30/21]seed@VM:~/Desktop$
```

Q3.1 Task 1.3 - Submit your scapy code

15 Points



traceroutepy.PNG

Download

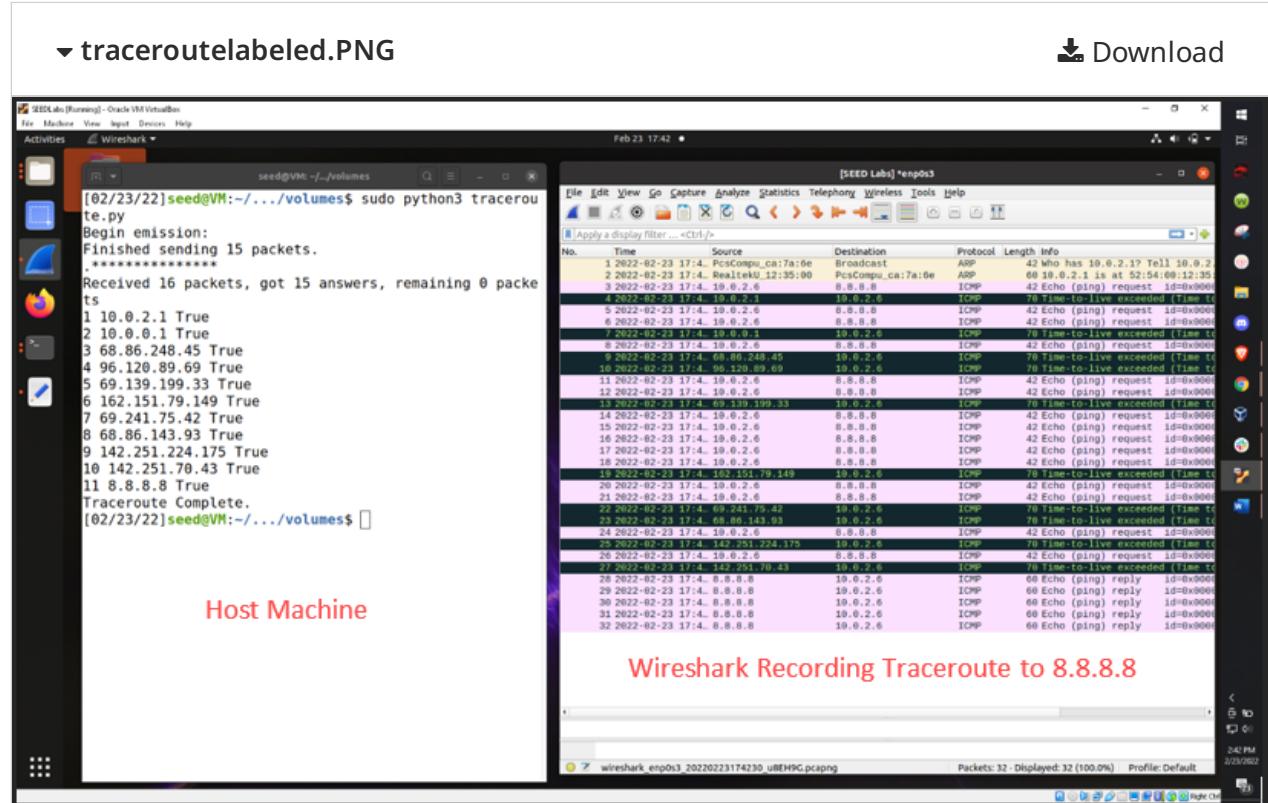
traceroute.py
~/Labsetup/volumes

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

Saving file "/home/seed/Labsetup/volumes/traceroute.py" 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**)



Q4 Task 1.4: Sniffing and-then Spoofing

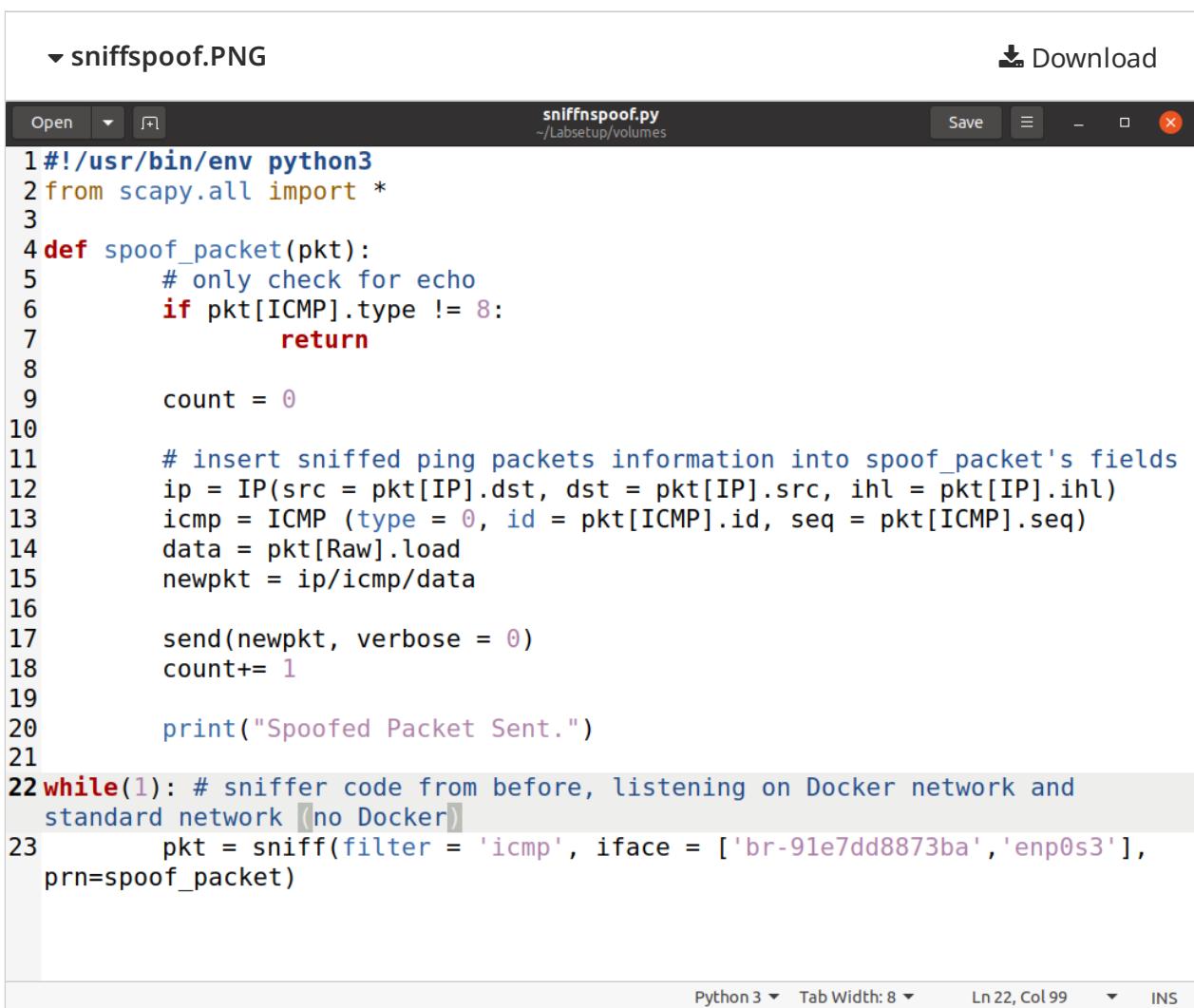
40 Points

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



The screenshot shows a code editor window with the following details:

- Title Bar:** The title bar displays "sniffnspoof.PNG" and "sniffnspoof.py ~/Labsetup/volumes".
- Toolbar:** Standard file operations like Open, Save, and Download are visible.
- Code Area:** The main area contains a Python script named "sniffnspoof.py". The code uses the Scapy library to perform the following steps:
 - Sniffs ICMP packets (line 22).
 - For each packet, it checks if the type is not 8 (echo request). If true, it returns (line 6).
 - If false, it initializes variables: count (line 9), ip (line 11), icmp (line 12), and data (line 13).
 - It creates a new packet (newpkt) by combining the IP and ICMP layers (line 14).
 - It sends the spoofed packet (line 17) and increments the count (line 18).
 - It prints a success message ("Spoofed Packet Sent.") (line 20).
 - It loops back to sniff more packets (line 22).
- Status Bar:** The bottom status bar shows "Python 3" and "Tab Width: 8".

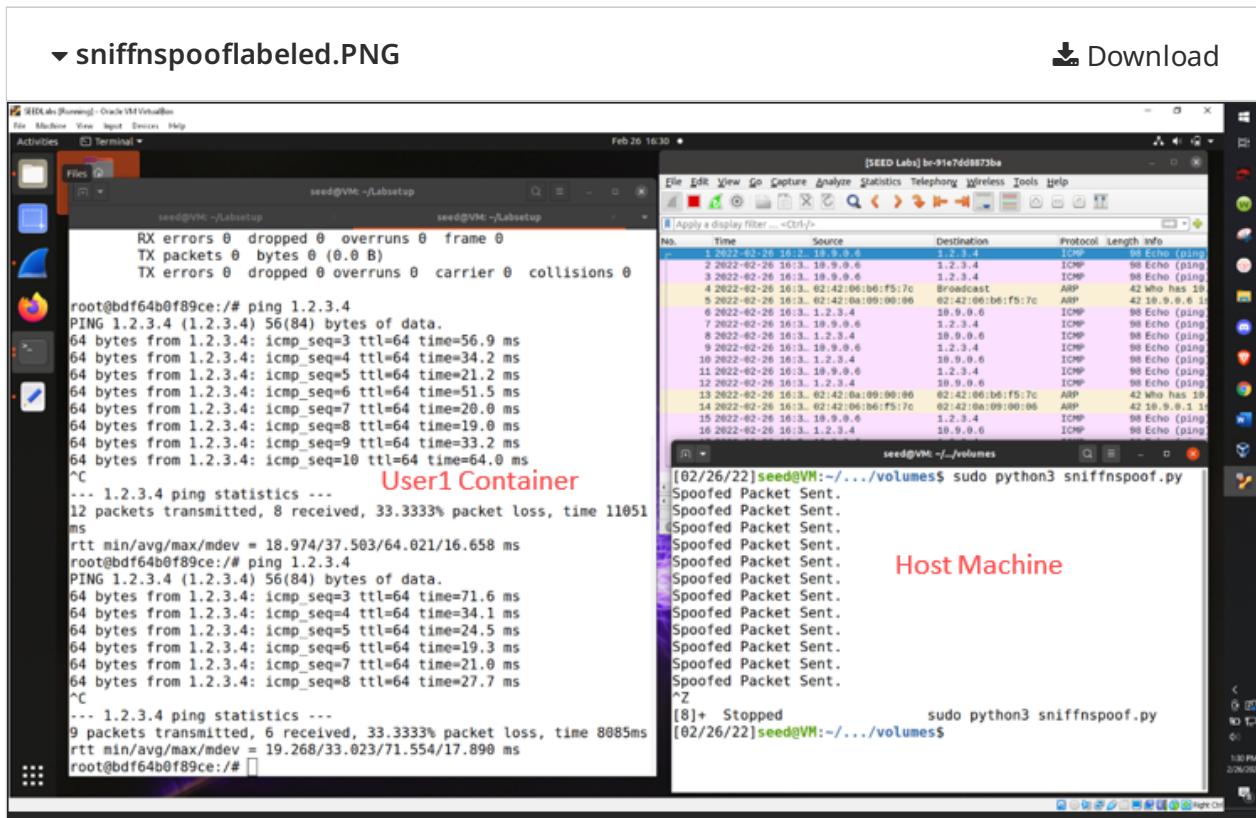
Q4.2 Task 1.4 - Ping 1.2.3.4 and show screenshots of the output from your program and with the ping from the terminal

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



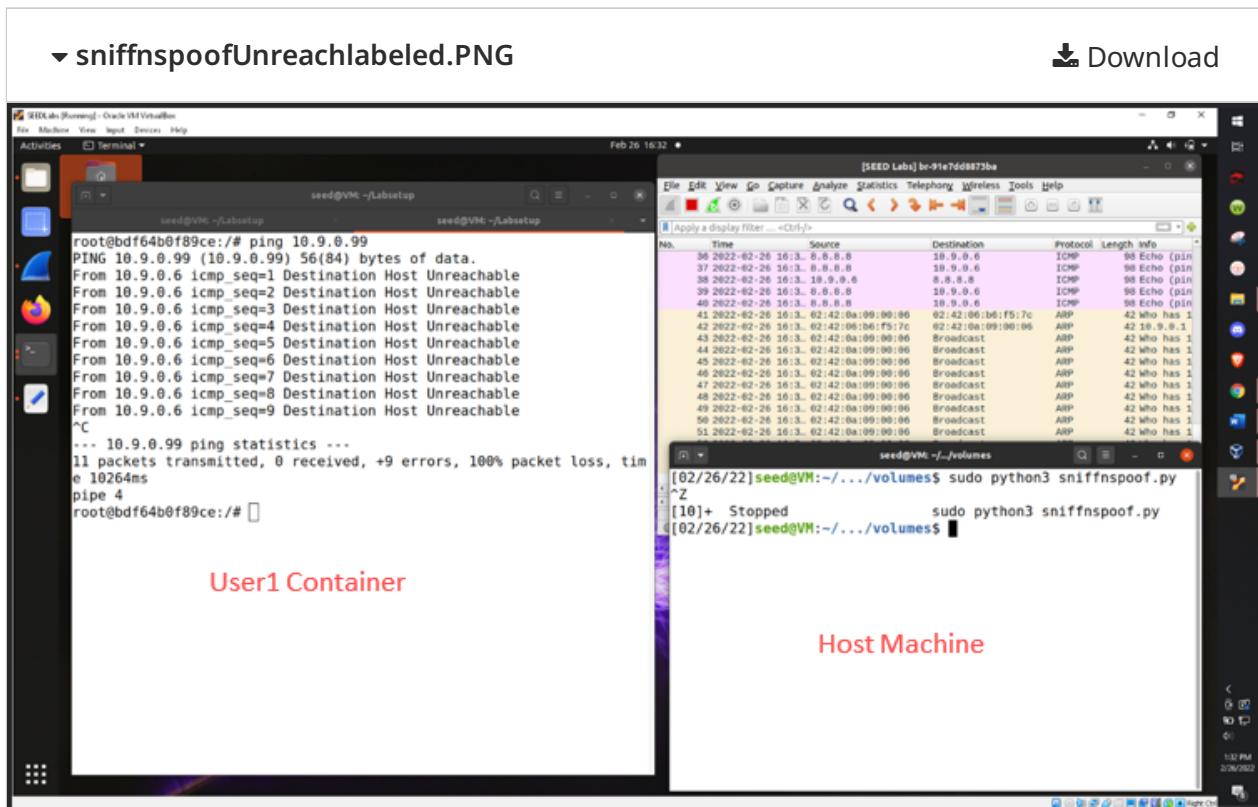
Q4.3 Task 1.4 - Ping 10.9.0.99 and show screenshots of the output from your program and with the ping from the terminal. If this does not work, please explain why.

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



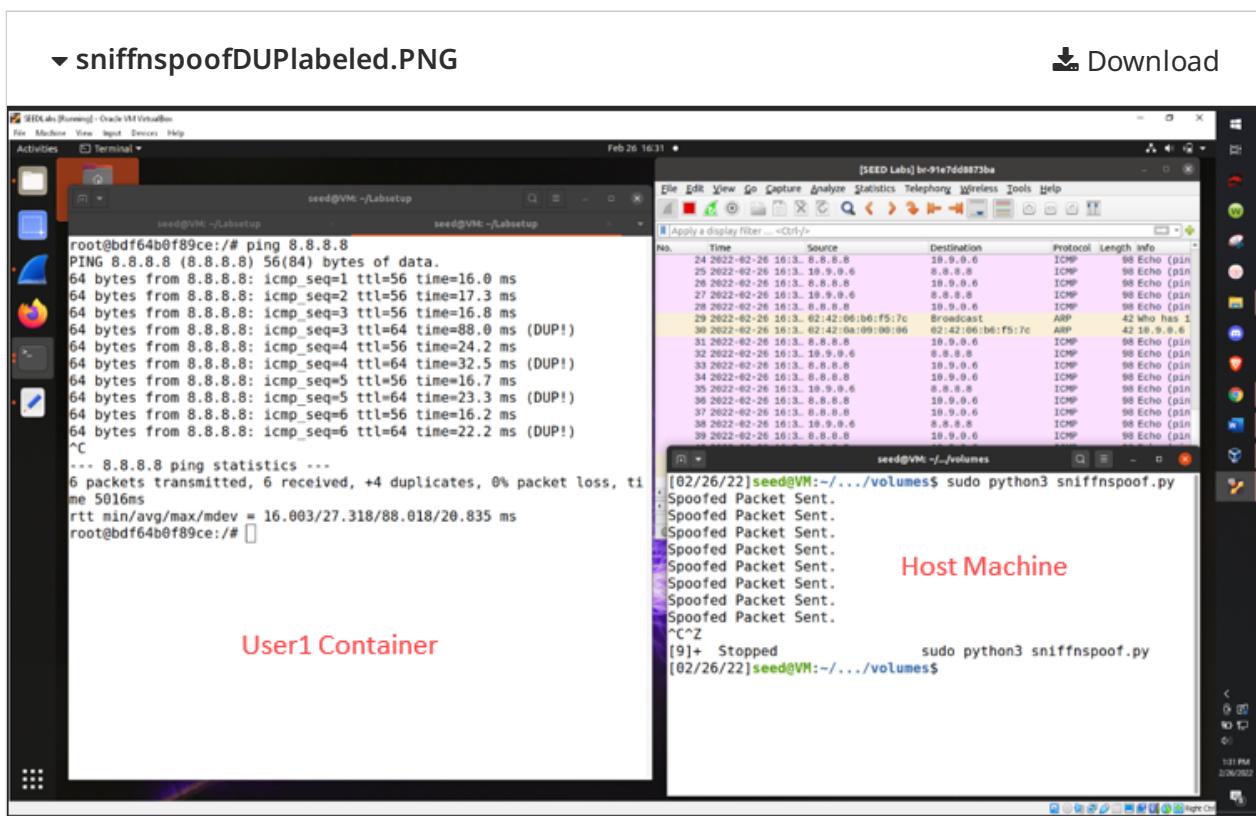
Q4.4 Task 1.4 - Ping 8.8.8.8 and show screenshots of the output from your program and with the ping from the terminal

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



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 successfully.

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.