

Q1 Lab 2 - Packet Sniffing and Spoofing Lab

15 Points

Lab PDF: [Packet Sniffing and Spoofing Lab](#)

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: [Packet Sniffing and Spoofing Lab](#) - **Please use this Labsetup for this lab. Make sure to take down the docker containers from Lab 1, and install the ones for Lab 2.** You should use `Labsetup.zip` unless you're on Apple Silicon in which you should use `Labsetup-arm.zip`
- [Docker Manual](#) (if more help is needed)

Note: Only images or TXT files are acceptable upload formats

Normal due date: **5 Mar**

Earliest acceptance date: 29 Feb (+1% per day, up to +5% bonus for five days early)

Latest acceptance date: 13 Mar (-8% points)

Please follow the instructions from the PDF document, but submit your work based on the instructions below. For any code snippets, use the ones from Gradescope as they would have any corrections.

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

0 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
```

Nothing need to be submitted for Task 1.1.

Q1.2 Task 1.1B(a) - Only ICMP

2.5 Points

Using `sniff(filter='__')`, capture only ICMP packets.

Select the option that goes in the blank

```
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter=__ (1) __, prn=print_pkt)
```

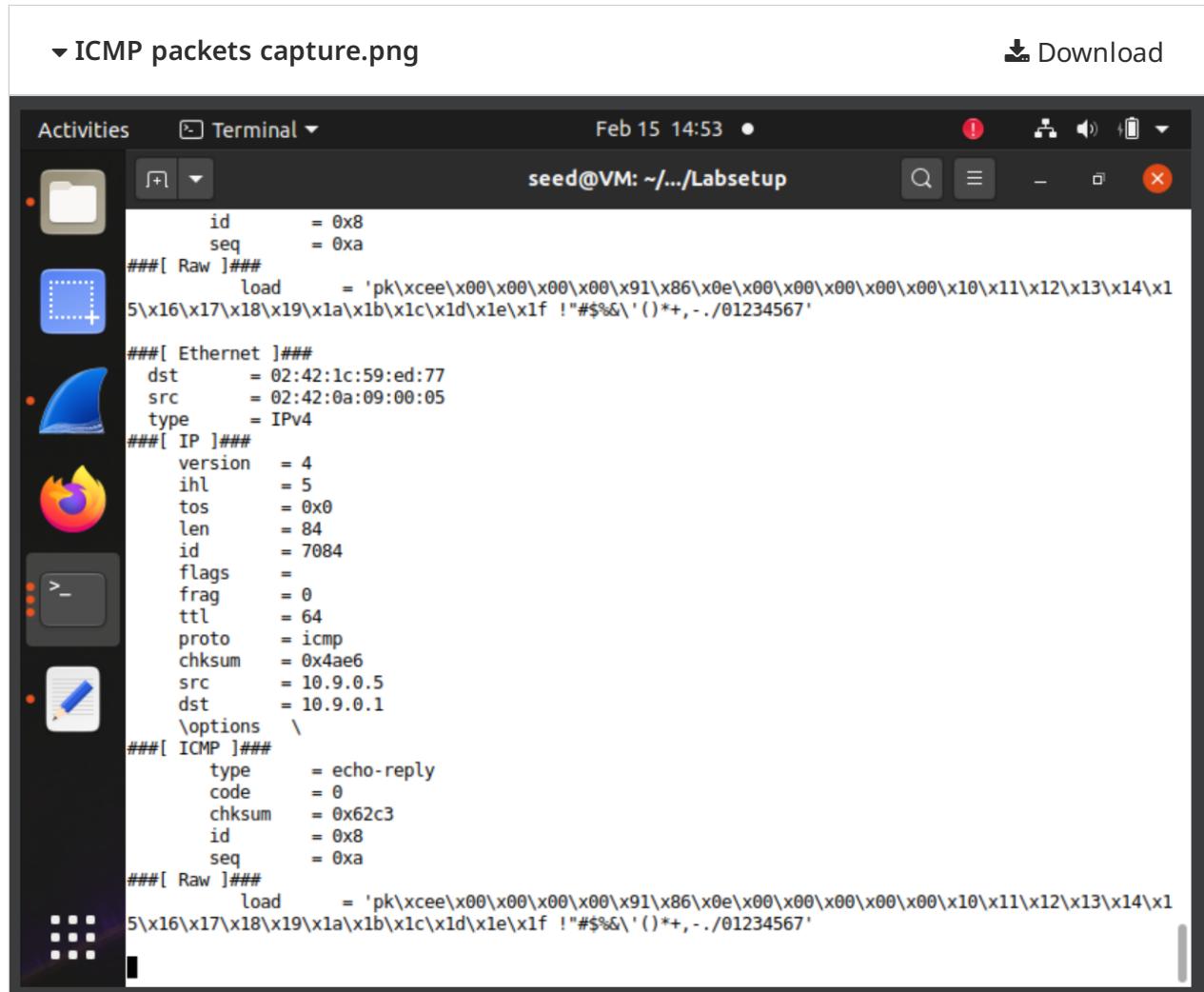
- ☐ ping
- ☒ icmp
- ☐ tcp

Q1.3 Task 1.1B(a) - Only ICMP (Screenshot)

2.5 Points

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

Hint: Be sure to adjust the `iface` correctly when working locally



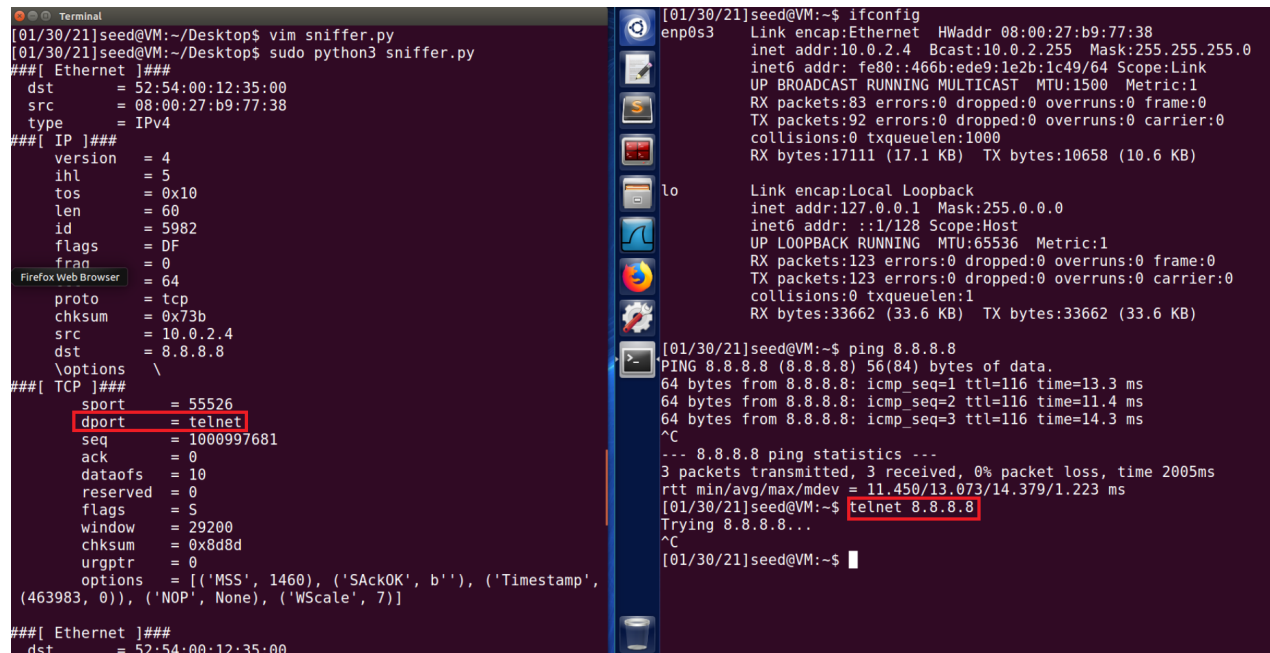
Q1.4 Task 1.1B(b) - Capture specific TCP/port 23

2.5 Points

Capture any TCP packet that comes from a specific IP and with a destination port number 23. Your code must include the following: (1) capture TCP; (2) capture from any specific IP address; and (3) capture to destination port 23.

Hint: Use the attacker machine and write your code for sniffing. Then generate telnet traffic from the attacker machine to test that your program works.

Expected output



The screenshot shows a terminal window with two panes. The left pane displays the output of the 'vim sniffer.py' command, showing the configuration of a packet filter. The right pane shows the output of the 'ifconfig' command, displaying network interface details for 'enp0s3' and 'lo'. Below the 'ifconfig' output, the terminal shows the execution of 'ping 8.8.8.8' and 'telnet 8.8.8.8' commands, with the telnet command being highlighted in red.

```
[01/30/21]seed@VM:~/Desktop$ vim sniffer.py
[01/30/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
dst      = 52:54:00:12:35:00
src      = 08:00:27:b9:77:38
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x10
len      = 60
id       = 5982
flags    = DF
frag     = 0
proto    = tcp
chksum   = 0x73b
src      = 10.0.2.4
dst      = 8.8.8.8
\options
###[ TCP ]###
sport    = 55526
dport    = telnet
seq      = 1000997681
ack      = 0
dataofs  = 10
reserved = 0
flags    = S
window   = 29200
chksum   = 0x8d8d
urgptr   = 0
options  = [('MSS', 1460), ('SACKOK', b''), ('Timestamp',
(463983, 0)), ('NOP', None), ('WScale', 7)]
###[ Ethernet ]###
dst      = 52:54:00:12:35:00
```

```
[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:ede9:1e2b: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...
[01/30/21]seed@VM:~$
```

Select the option that goes in the blank

```
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter= (1) ,prn=print_pkt)
```

- ☐ tcp port 23
- ☐ host 10.9.0.3 and port 23
- ☒ tcp and host 10.9.0.1 and port 23
- ☐ icmp and host 10.9.0.2 and port 23

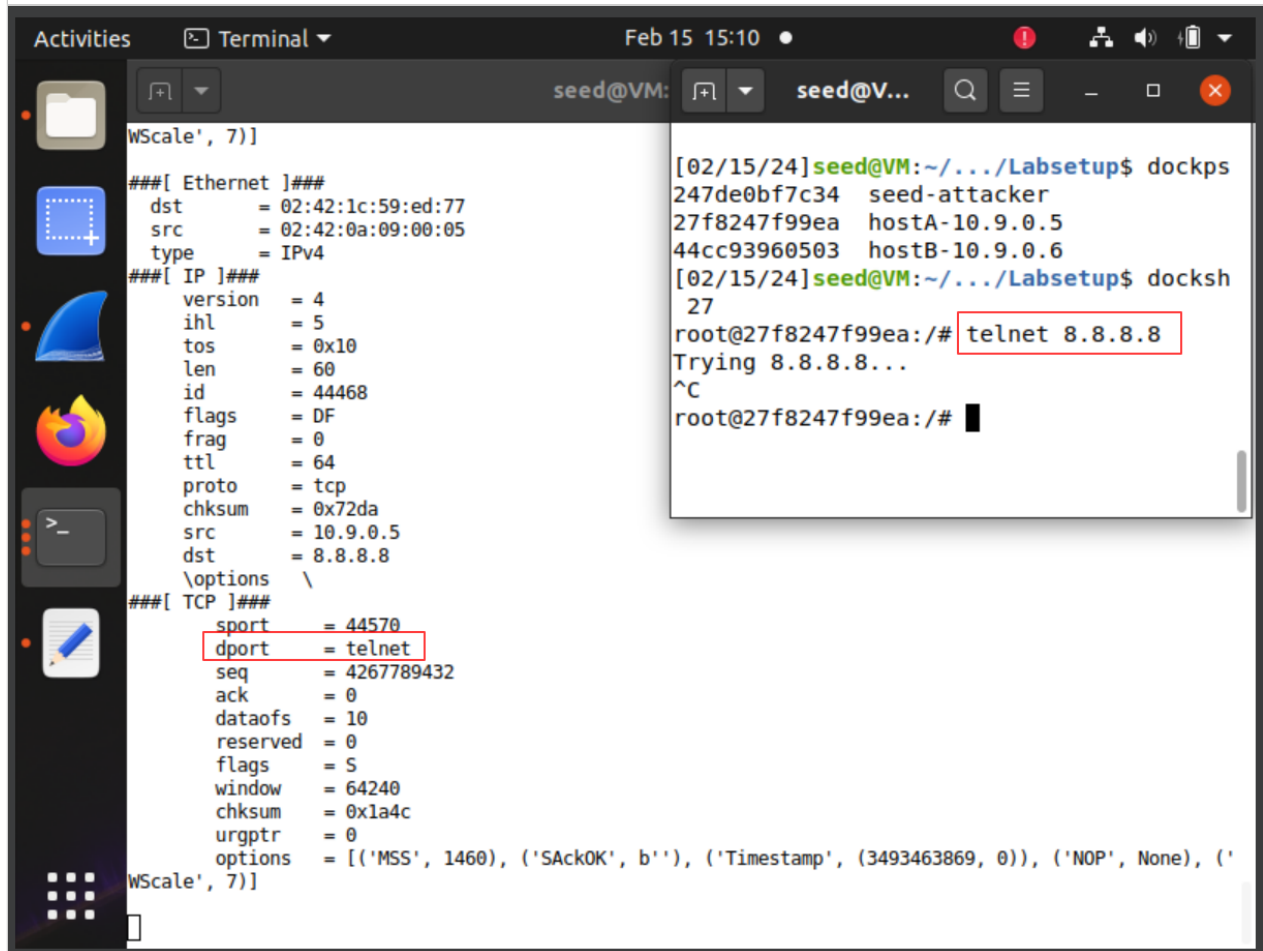
Q1.5 Task 1.1B(b) - Capture specific TCP/port 23 (Screenshot)

2.5 Points

Take a screenshot demonstrating that the TCP/port 23 packet was captured (must screenshot the entire VM along with date and time).

▼ Capture of specific TCP-subnet-port labelled.png

Download



The screenshot shows a Linux desktop environment. The top panel displays the date and time as 'Feb 15 15:10'. The left sidebar contains icons for the Dash, Home, and Applications menus, along with icons for a file manager, a terminal, and a web browser. The main window is a terminal titled 'seed@VM:'. It displays the output of a network capture, showing the details of a TCP packet. The packet is an IPv4 packet with a destination IP of 8.8.8.8 and a destination port of 23 (telnet). The source IP is 10.9.0.5. The packet is captured on the interface 'eth0'. The terminal also shows the output of a 'telnet 8.8.8.8' command, which results in a connection attempt to 8.8.8.8. The terminal window is titled 'seed@V...' and has a search bar and window controls.

```
WScale', 7)]  
  
###[ Ethernet ]###  
dst      = 02:42:1c:59:ed:77  
src      = 02:42:0a:09:00:05  
type     = IPv4  
###[ IP ]###  
version  = 4  
ihl      = 5  
tos      = 0x10  
len      = 60  
id       = 44468  
flags    = DF  
frag     = 0  
ttl      = 64  
proto    = tcp  
chksum   = 0x72da  
src      = 10.9.0.5  
dst      = 8.8.8.8  
\options \  
###[ TCP ]###  
sport    = 44570  
dport    = telnet  
seq      = 4267789432  
ack      = 0  
dataofs  = 10  
reserved = 0  
flags    = S  
window   = 64240  
chksum   = 0x1a4c  
urgptr   = 0  
options  = [('MSS', 1460), ('SACKOK', b''), ('Timestamp', (3493463869, 0)), ('NOP', None), ('  
WScale', 7)]
```

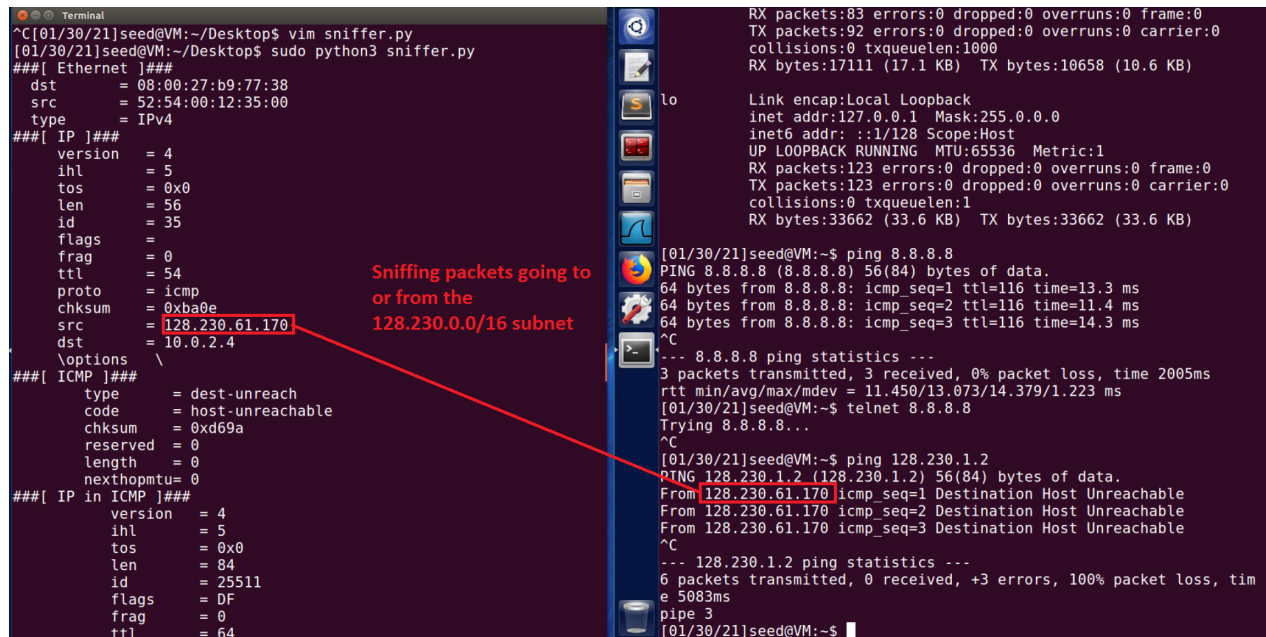
```
[02/15/24]seed@VM:~/.../Labsetup$ dockps  
247de0bf7c34 seed-attacker  
27f8247f99ea hostA-10.9.0.5  
44cc93960503 hostB-10.9.0.6  
[02/15/24]seed@VM:~/.../Labsetup$ docksh  
27  
root@27f8247f99ea:/# telnet 8.8.8.8  
Trying 8.8.8.8...  
^C  
root@27f8247f99ea:/#
```

Q1.6 Task 1.1B(c) - Particular Subnet

2.5 Points

Capture packets coming from or going to a subnet 10.9.0.0/24. Fill in the blank and take a screenshot demonstrating the traffic generated.

Expected output



```
^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     = 0
  ttl      = 54
  proto    = icmp
  chksum   = 0xba0e
  src      = 128.230.61.170
  dst      = 10.0.2.4
  \options \
###[ ICMP ]###
  type     = dest-unreach
  code     = host-unreachable
  chksum   = 0xd69a
  reserved = 0
  length   = 0
  nextthopmtu= 0
###[ IP in ICMP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 25511
  flags    = DF
  frag     = 0
  ttl      = 64

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:~$ 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, time 5083ms
pipe 3
[01/30/21]seed@VM:~$
```

Select the option that goes in the blank

Note: Use subnet 10.9.0.0/24

```
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter=(1), prn=print_pkt)
```

- ☒ net 10.9.0.0/24
- ☐ net 10.9.0.0
- ☐ net 10.9.0.0/26
- ☐ net 128.238.0.0/24

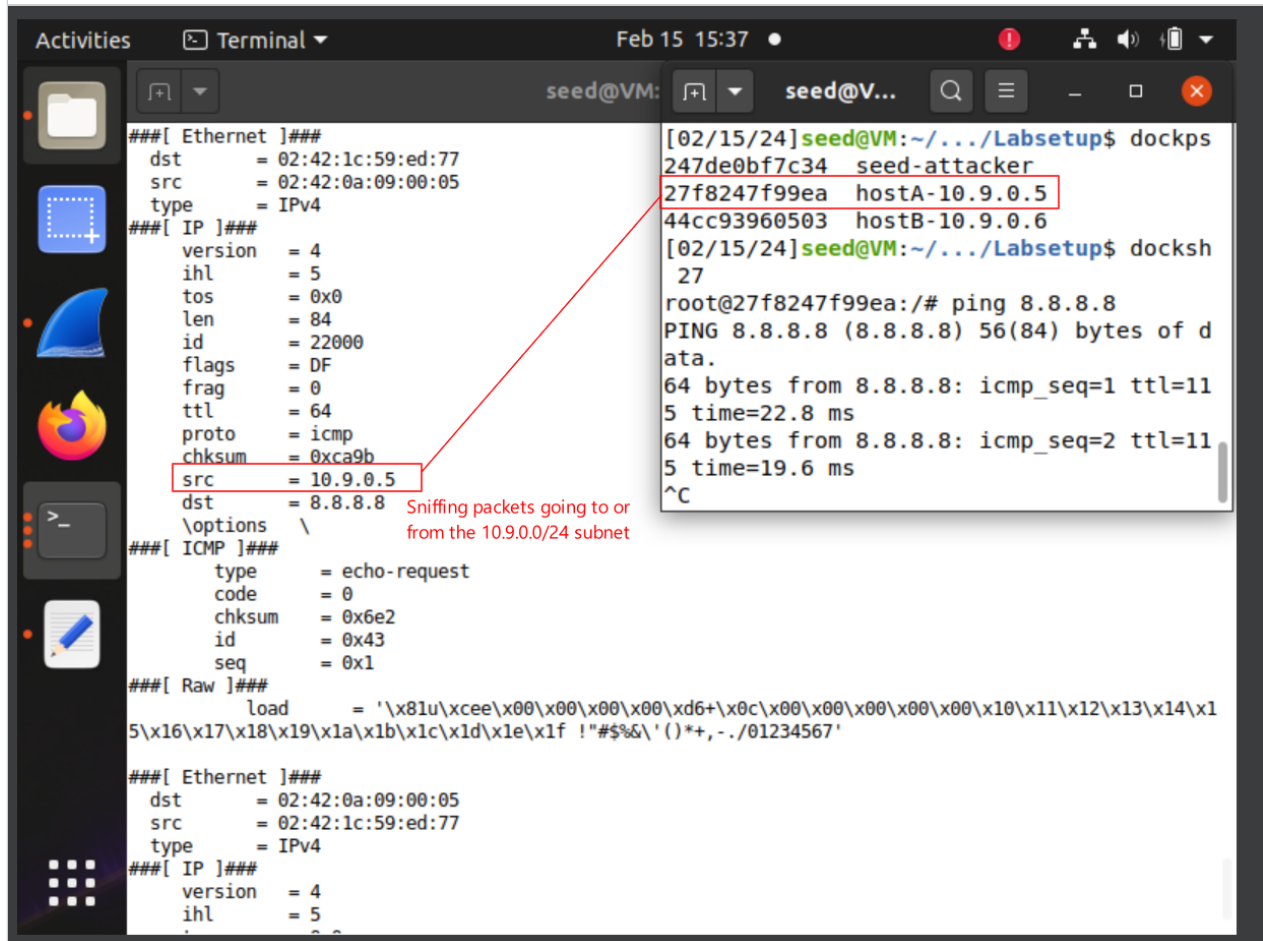
Q1.7 Task 1.1B(c) - Particular Subnet (Screenshot)

2.5 Points

Take a screenshot demonstrating that that packets from the 10.9.0.0/24 subnet were captured (must screenshot the entire VM along with date and time).

▼ Capture of particular subnet labelled.png

Download



```
###[ Ethernet ]###
dst      = 02:42:1c:59:ed:77
src      = 02:42:0a:09:00:05
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 22000
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
chksum   = 0xca9b
src       = 10.9.0.5
dst      = 8.8.8.8
\options \
###[ ICMP ]###
type     = echo-request
code     = 0
chksum   = 0x6e2
id       = 0x43
seq      = 0x1
###[ Raw ]###
load     = '\x81u\xcee\x00\x00\x00\x00\xd6+\x0c\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&'()*+,-./01234567'
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:1c:59:ed:77
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
```

[02/15/24] seed@VM: ~/.../Labsetup\$ dockps
247de0bf7c34 seed-attacker
27f8247f99ea hostA-10.9.0.5
44cc93960503 hostB-10.9.0.6
[02/15/24] seed@VM: ~/.../Labsetup\$ docksh
27
root@27f8247f99ea:/# 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=115 time=22.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=19.6 ms
^C

Sniffing packets going to or from the 10.9.0.0/24 subnet

Q2 Task 1.2: Spoofing ICMP Packets

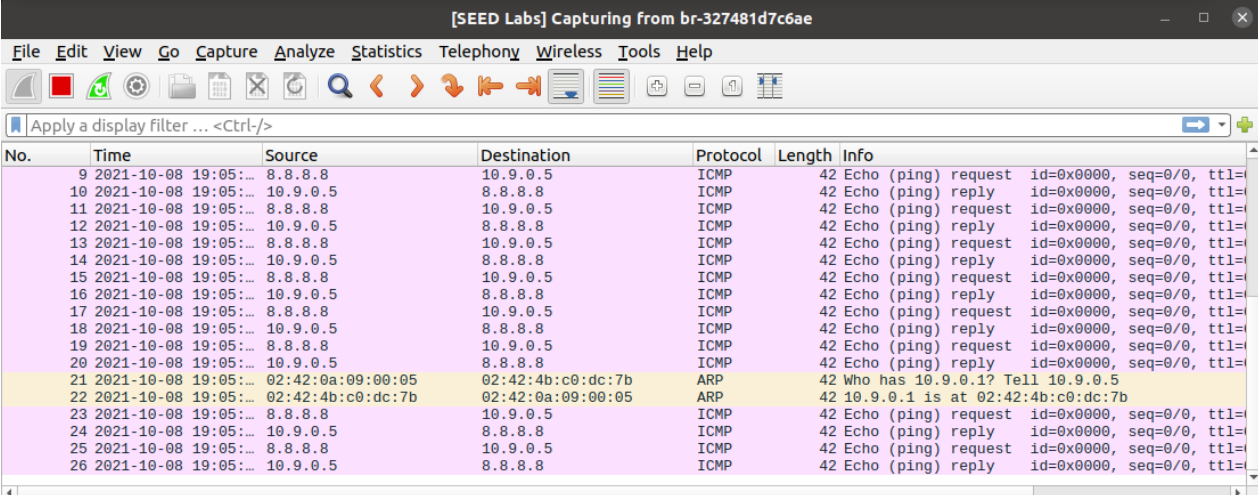
15 Points

```
>>> 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 packet list shows a series of ICMP Echo (ping) requests and replies. The source IP is 8.8.8.8 and the destination is 10.9.0.5. The packet details pane shows the selected packet (No. 21) as an ARP request from 02:42:0a:09:00:05 to 02:42:4b:c0:dc:7b.

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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...
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=...

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-327481d7c6ae, id 0
Ethernet II, Src: 02:42:4b:c0:dc:7b (02:42:4b:c0:dc:7b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)

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 - Scapy code

10 Points

```
from scapy.all import *  
ip = IP(src=(1), dst=(2))  
icmp = ICMP()  
packet = ip/icmp  
packet.show()  
send(packet)
```

a. Based on the code above, what is the IP address for (1)?

8.8.8.8

b. Based on the code above, what is the IP address for (2)?

10.9.0.5

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)

▼ Spoofing ICMP packets.png

Download

ActivitiesWiresharkFeb 15 18:02

[SEED Labs] Capturing from br-02838e20ba31

FileEditViewGoCaptureAnalyzeStatisticsTelephonyWirelessToolsHelp

Apply a display filter ... <Ctrl-/>

Packet listNarrow & WideCase sensitiveDisplay filter

No.	Time	Source	Destination	Protocol	Length	Info
1	2024...	02:42:1c:5...	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2024...	02:42:0a:0...	02:42:1c:5...	ARP	42	10.9.0.5 is at 02:42:0a:09:00:0
3	2024...	8.8.8.8	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000,
4	2024...	10.9.0.5	8.8.8.8	ICMP	42	Echo (ping) reply id=0x0000,
5	2024...	02:42:0a:0...	02:42:1c:5...	ARP	42	Who has 10.9.0.1? Tell 10.9.0.5
6	2024...	02:42:1c:5...	02:42:0a:0...	ARP	42	10.9.0.1 is at 02:42:1c:59:ed:7

Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-02838e20ba31
Interface id: 0 (br-02838e20ba31)
Encapsulation type: Ethernet (1)
Arrival Time: Feb 15, 2024 17:58:00.148435010 EST
Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1708037880.148435010 seconds
Time delta from previous captured frame: 0.022664918 seconds]
Time delta from previous displayed frame: 0.022664918 seconds]
Time since reference or first frame: 0.022698546 seconds]
Frame Number: 3

br-02838e20ba31: <live capture in progress>Packets: 6 · Displayed: 6 (100%)

Q3 Task 1.3: Fully-automated Traceroute

20 Points

Using the skeleton code below, implement ICMP traceroute using scapy. Do NOT use the built-in scapy traceroute function. Perform a traceroute to 8.8.8.8. Show proof using a Wireshark capture 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 on the left and a Wireshark packet capture on the right. The terminal output shows the results of a traceroute to 8.8.8.8, including TTL, source IP, and destination IP for each hop. The Wireshark capture shows the corresponding ICMP and DNS packets for each hop, with the final hop showing a successful echo (ping) reply.

Terminal Output:

```
[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$
```

Wireshark Packet Capture:

No.	Time	Source	Destination	Protocol	Length	Info
9	2021-01-30 17:28:55.8350957	10.0.2.15	172.217.6.206	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
10	2021-01-30 17:28:55.8364978	192.168.1.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
11	2021-01-30 17:28:55.8583055	10.0.2.15	192.168.1.1	DNS	70	Standard query 0xfa67 A google.com
12	2021-01-30 17:28:55.8820257	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0xfa67 A google.com...
13	2021-01-30 17:28:55.8996409	10.0.2.15	142.250.46.14	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
14	2021-01-30 17:28:55.9134972	67.250.0.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
15	2021-01-30 17:28:55.9308201	10.0.2.15	192.168.1.1	DNS	70	Standard query 0x450b A google.com
16	2021-01-30 17:28:55.9468153	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0x450b A google.com...
17	2021-01-30 17:28:55.9714293	10.0.2.15	172.217.6.206	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
18	2021-01-30 17:28:55.9918386	68.173.200.30	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
19	2021-01-30 17:28:56.0027435	10.0.2.15	192.168.1.1	DNS	70	Standard query 0xe132 A google.com
20	2021-01-30 17:28:56.0152693	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0xe132 A google.com...
21	2021-01-30 17:28:56.0360284	10.0.2.15	172.217.11.46	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
22	2021-01-30 17:28:56.0570477	68.173.198.136	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
23	2021-01-30 17:28:56.0768424	10.0.2.15	192.168.1.1	DNS	70	Standard query 0xba55 A google.com
24	2021-01-30 17:28:56.1025174	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0xba55 A google.com...
25	2021-01-30 17:28:56.1273844	10.0.2.15	172.217.11.14	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
26	2021-01-30 17:28:56.1460010	107.14.19.24	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
27	2021-01-30 17:28:56.1584793	10.0.2.15	192.168.1.1	DNS	70	Standard query 0x53fe A google.com
28	2021-01-30 17:28:56.1775025	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0x53fe A google.com...
29	2021-01-30 17:28:56.1965598	10.0.2.15	142.250.64.110	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
30	2021-01-30 17:28:56.2159381	66.109.6.27	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
31	2021-01-30 17:28:56.2346973	10.0.2.15	192.168.1.1	DNS	70	Standard query 0xba16 A google.com
32	2021-01-30 17:28:56.2503337	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0xba16 A google.com...
33	2021-01-30 17:28:56.2722331	10.0.2.15	142.250.64.110	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
34	2021-01-30 17:28:56.2892577	74.125.147.198	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
35	2021-01-30 17:28:56.3027394	10.0.2.15	192.168.1.1	DNS	70	Standard query 0x758b A google.com
36	2021-01-30 17:28:56.3231088	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0x758b A google.com...
37	2021-01-30 17:28:56.3437513	10.0.2.15	172.217.6.206	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
38	2021-01-30 17:28:56.3609115	108.170.237.205	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
39	2021-01-30 17:28:56.3748604	10.0.2.15	192.168.1.1	DNS	70	Standard query 0xd625 A google.com
40	2021-01-30 17:28:56.3903262	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0xd625 A google.com...
41	2021-01-30 17:28:56.4073108	10.0.2.15	142.250.88.14	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
42	2021-01-30 17:28:56.4083419	142.250.46.194	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceede...
43	2021-01-30 17:28:56.4424757	10.0.2.15	192.168.1.1	DNS	70	Standard query 0xb647 A google.com
44	2021-01-30 17:28:56.4609926	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0xb647 A google.com...
45	2021-01-30 17:28:56.4757081	10.0.2.15	172.217.12.206	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt...
46	2021-01-30 17:28:56.4943098	172.217.12.206	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, tt...

Q3.1 Task 1.3 - Scapy code

15 Points

```
from scapy.all import *

ttl = (1)
while True:
    a = IP(dst=(2), ttl=ttl)
    b = ICMP()
    p = a/b
    pkt = sr1(p, verbose = 0)

    if (3) == 0:
        print("Complete ", pkt[IP].src)
        break
    else:
        print("TTL: %d, Source: " %ttl , pkt[IP].src)
        ttl +=1
```

a. Fill in the blank for 1.

1

b. Fill in the blank for 2.

"8.8.8.8"

c. Fill in the blank for 3.

pkt[ICMP].type

Q3.2 Task 1.3 - Write your own trace route program

5 Points

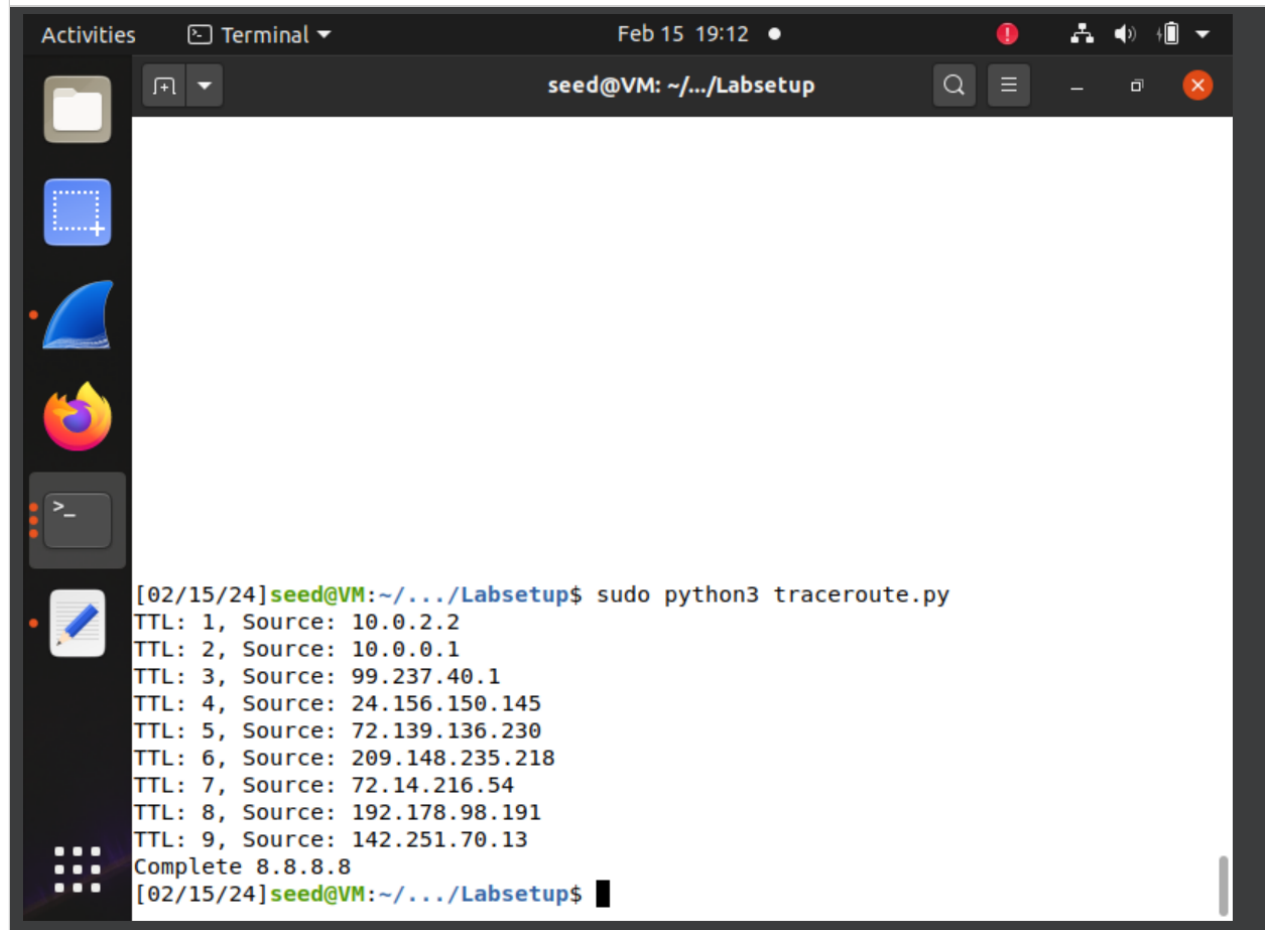
Write your own traceroute program using the skeleton code above. Test your traceroute program by tracerouting 8.8.8.8. Note: You may need to add additional code to handle conditions where the router does not respond. **Your code must fully automate automate the trace route for credit.**

For Q3.2 Task 1.3, submit the following:

- Your traceroute program
- Screenshot of Wireshark showing proof of traceroute to 8.8.8.8
- Your program output
- Screenshot of built-in Ubuntu traceroute to 8.8.8.8 (`traceroute -I 8.8.8.8`)

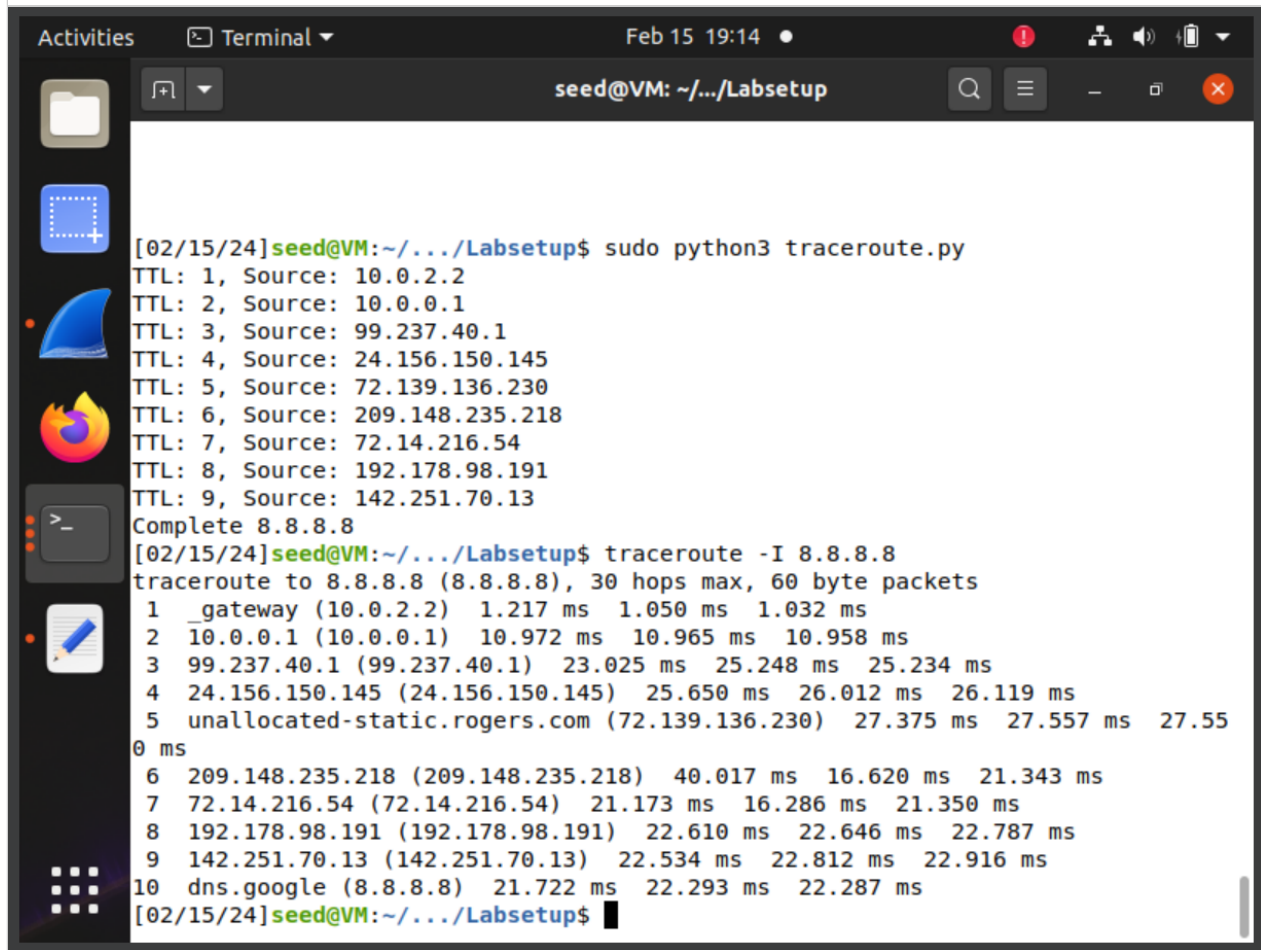
Submit a screenshot of Wireshark (**must screenshot the entire VM along with date and time**)

▼ Traceroute program output.png Download



The screenshot shows a terminal window titled 'seed@VM: ~/.../Labsetup' with a date and time of 'Feb 15 19:12'. The terminal displays the output of a custom traceroute program executed via `sudo python3 traceroute.py`. The output lists 9 hops with their respective TTLs and source IP addresses, ending with 'Complete 8.8.8.8'. The terminal interface includes a sidebar with application icons and a top bar with system status icons.

```
[02/15/24]seed@VM:~/.../Labsetup$ sudo python3 traceroute.py
TTL: 1, Source: 10.0.2.2
TTL: 2, Source: 10.0.0.1
TTL: 3, Source: 99.237.40.1
TTL: 4, Source: 24.156.150.145
TTL: 5, Source: 72.139.136.230
TTL: 6, Source: 209.148.235.218
TTL: 7, Source: 72.14.216.54
TTL: 8, Source: 192.178.98.191
TTL: 9, Source: 142.251.70.13
Complete 8.8.8.8
[02/15/24]seed@VM:~/.../Labsetup$
```



The image shows a terminal window titled "seed@VM: ~/.../Labsetup" with a dark theme. The window displays the output of a traceroute command. The first command is `sudo python3 traceroute.py`, which shows a series of TTLs and source IP addresses. The second command is `traceroute -I 8.8.8.8`, which shows a detailed traceroute to the destination IP 8.8.8.8, including hop numbers, IP addresses, and round-trip times in milliseconds.

```
[02/15/24]seed@VM:~/.../Labsetup$ sudo python3 traceroute.py
TTL: 1, Source: 10.0.2.2
TTL: 2, Source: 10.0.0.1
TTL: 3, Source: 99.237.40.1
TTL: 4, Source: 24.156.150.145
TTL: 5, Source: 72.139.136.230
TTL: 6, Source: 209.148.235.218
TTL: 7, Source: 72.14.216.54
TTL: 8, Source: 192.178.98.191
TTL: 9, Source: 142.251.70.13
Complete 8.8.8.8
[02/15/24]seed@VM:~/.../Labsetup$ traceroute -I 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1 _gateway (10.0.2.2)  1.217 ms  1.050 ms  1.032 ms
 2 10.0.0.1 (10.0.0.1)  10.972 ms  10.965 ms  10.958 ms
 3 99.237.40.1 (99.237.40.1)  23.025 ms  25.248 ms  25.234 ms
 4 24.156.150.145 (24.156.150.145)  25.650 ms  26.012 ms  26.119 ms
 5 unallocated-static.rogers.com (72.139.136.230)  27.375 ms  27.557 ms  27.55
0 ms
 6 209.148.235.218 (209.148.235.218)  40.017 ms  16.620 ms  21.343 ms
 7 72.14.216.54 (72.14.216.54)  21.173 ms  16.286 ms  21.350 ms
 8 192.178.98.191 (192.178.98.191)  22.610 ms  22.646 ms  22.787 ms
 9 142.251.70.13 (142.251.70.13)  22.534 ms  22.812 ms  22.916 ms
10 dns.google (8.8.8.8)  21.722 ms  22.293 ms  22.287 ms
[02/15/24]seed@VM:~/.../Labsetup$
```

Activities Wireshark Feb 15 19:10 [SEED Labs] *enp0s3

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	2024-...	PcsCompu_44...	Broadcast	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
2	2024-...	RealtekU_12...	PcsCompu_44:...	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
3	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
4	2024-...	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
5	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
6	2024-...	10.0.0.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
7	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
8	2024-...	99.237.40.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
9	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
10	2024-...	24.156.150...	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
11	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
12	2024-...	72.139.136...	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
13	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
14	2024-...	209.148.235...	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
15	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
16	2024-...	72.14.216.54	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
17	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
18	2024-...	192.178.98...	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
19	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
20	2024-...	142.251.70...	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live
21	2024-...	10.0.2.15	8.8.8.8	ICMP	42	Echo (ping) request id=0x0000, seq
22	2024-...	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3...

0000 ff ff ff ff ff ff 08 00 27 44 aa b5 08 06 00 01 'D.....

wireshark_enp0s...3_6NG6Th.pcapng Packets: 22 · Displayed: 22 (100.0%) Profile: Default

```
1 from scapy.all import *
2
3 ttl=1
4 while True:
5     ip = IP(dst='8.8.8.8', ttl=ttl)
6     icmp = ICMP()
7     p = ip/icmp
8
9     resp = sr1(p, verbose=0, timeout=5)
10
11     if (resp==None):
12         print("TTL: {0}, Source: ???".format(ttl))
13         ttl+=1
14     elif (resp[ICMP].type==0):
15         print("Complete", resp[IP].src)
16         break
17     else:
18         print("TTL: {0}, Source: {1}".format(ttl, resp[IP].src))
19         ttl+=1
20
```

Q4 Task 1.4: Sniffing and-then Spoofing

50 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 - Sniffing and then spoofing program

20 Points

Using the skeleton code below, implement the sniffing and the spoofing program.

```
from scapy.all import *
def sniff_and_spoof(packet):
    if ICMP in packet:

        ip = IP(src=(1), dst=(2))
        icmp = ICMP(type=(3), id=(4), seq=(5))

        raw_data=(6)
        newpacket=(7)

        send(newpacket, verbose=0)

pkt = sniff(filter=(8), prn=sniff_and_spoof)
```

a. Fill in the blank for 1.

packet[IP].dst

b. Fill in the blank for 2.

packet[IP].src

c. Fill in the blank for 3.

0

d. Fill in the blank for 4.

packet[ICMP].id

e. Fill in the blank for 5.

packet[ICMP].seq

f. Fill in the blank for 6.

```
packet[Raw].load
```

g. Fill in the blank for 7.

```
ip/icmp/raw_data
```

h. [10 pts] Fill in the blank for 8. **You must filter out the packets that are sent out by your program. This question is worth ten points.**

```
"not ether src 02:42:1c:59:ed:77"
```

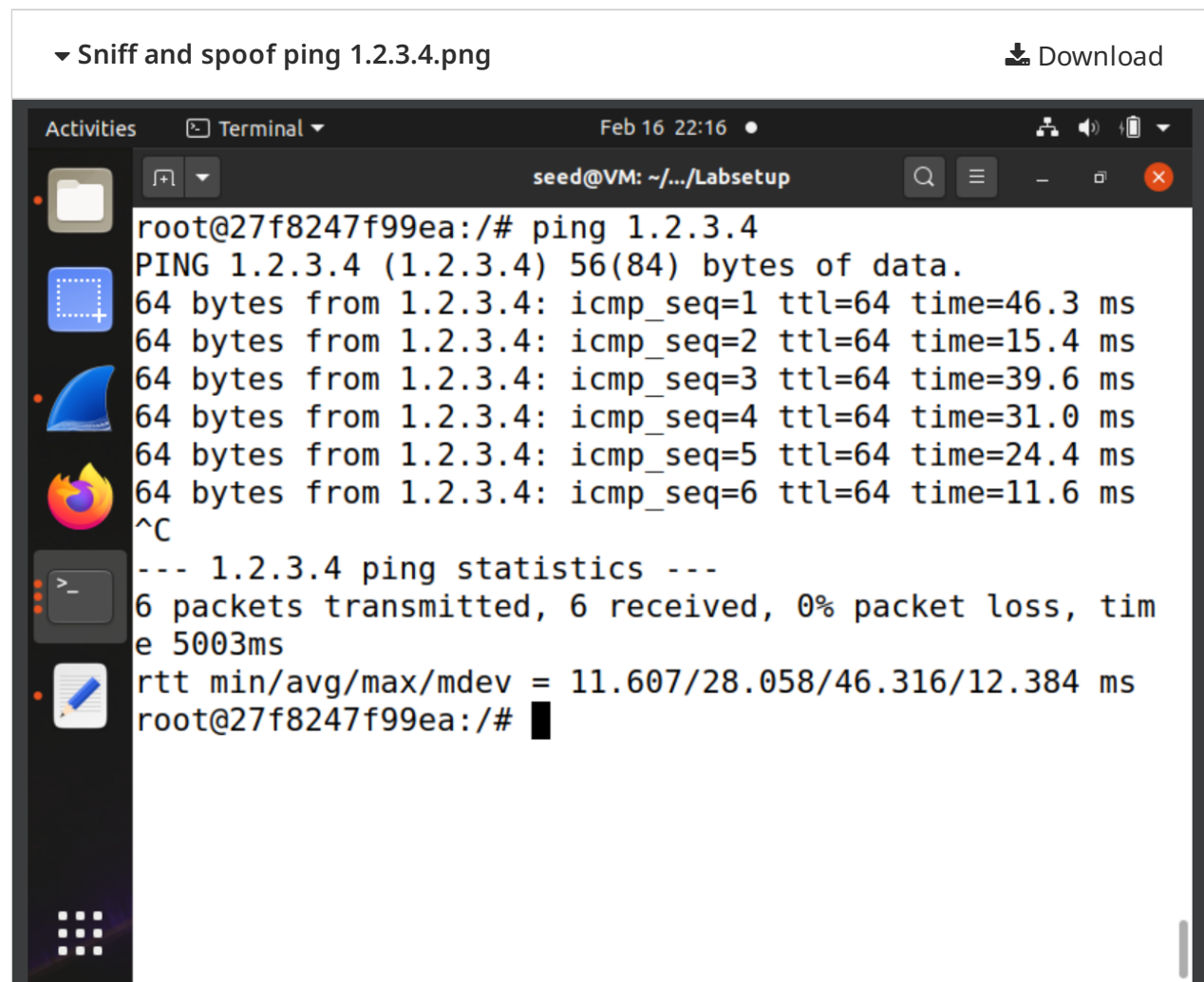
Q4.2 Task 1.4 - Ping 1.2.3.4 and show screenshots of the output from your program and with the ping from the terminal

10 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



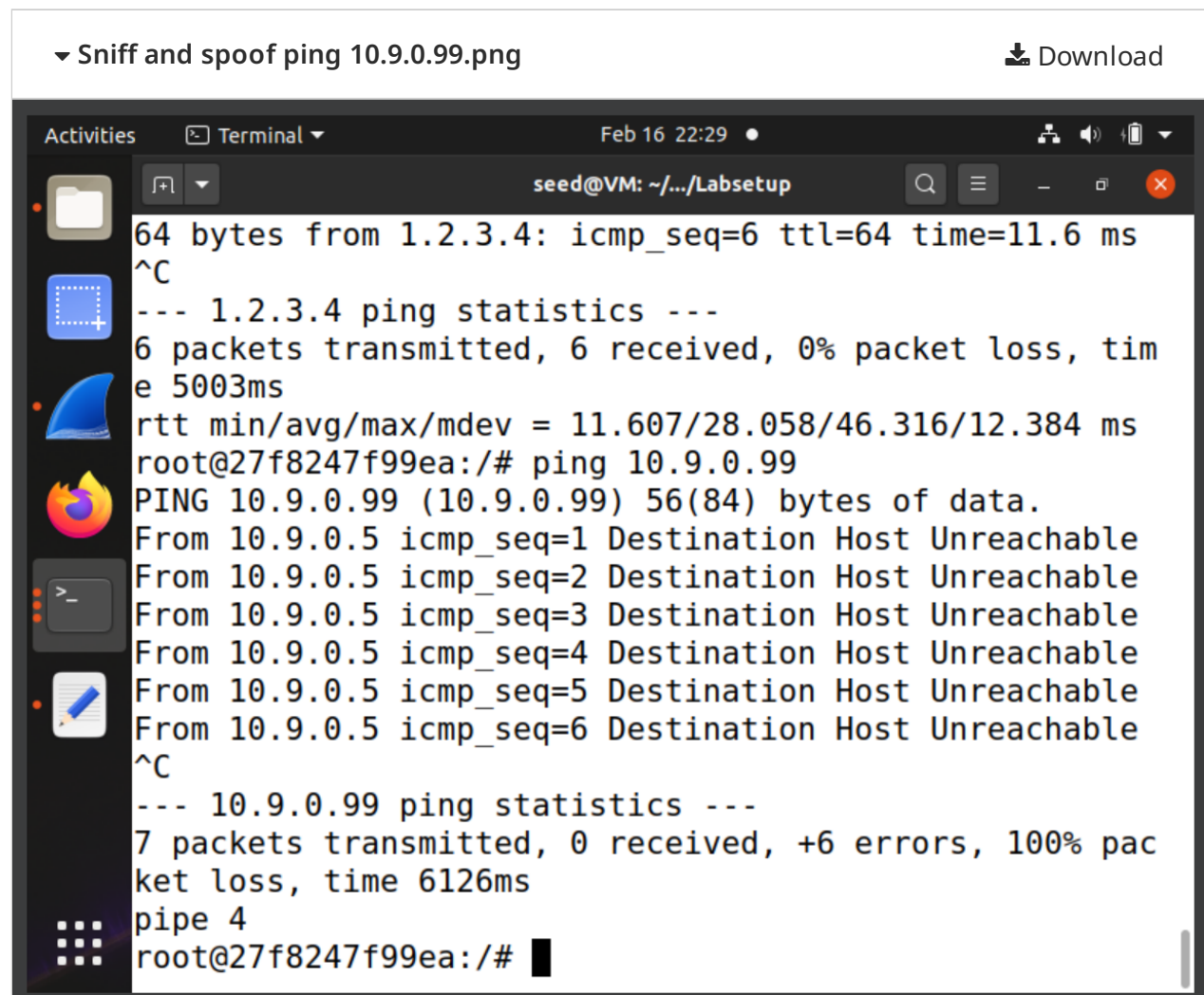
Q4.3 Task 1.4 - Ping 10.9.0.99 and show screenshots of the output from your program and with the ping from the terminal. If this does not work, please explain why.

10 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



Explain why it does not work. Explain what the host is doing and what protocol is failing to cause this error.

When we type in this command, the attacker VM looks at the IP and realizes that it is on the local subnet. So, the attacker VM will do an ARP request to see the MAC address associated with the IP being pinged. Since the IP does not exist on the subnet, the ARP request fails, and the attacker VM does not even send the ICMP ping request. Since an ICMP ping request is not sent, our program fails, since that is what it replies to.

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

10 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

