Lab2

Report:

The **Lab2** illustrates using Scapy to Sniff and Spoof Packets in the attached Sniffing_Spoofing.pdf file – Packet Sniffing and Spoofing Lab.

Task 1.1 Sniffing Packets

Task 1.1 (A)

Capture only ICMP packet

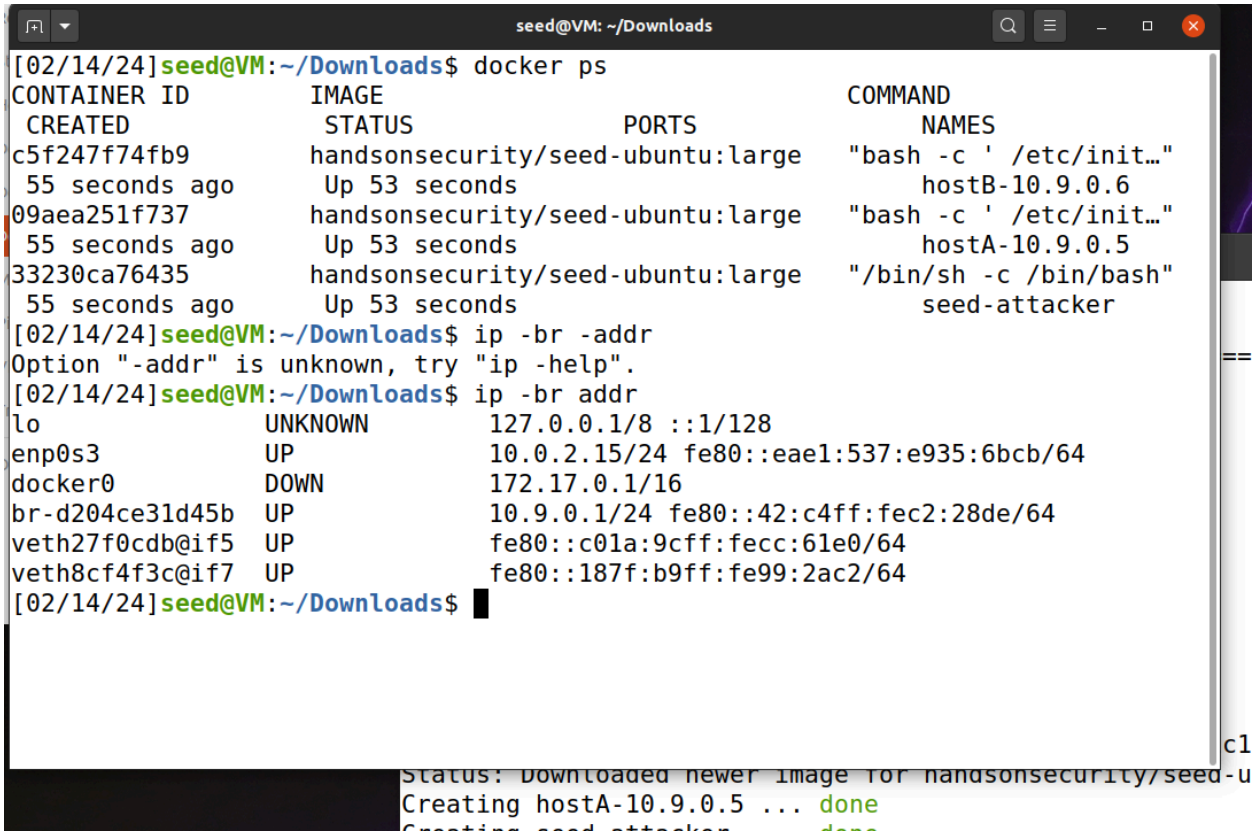

Figure:1- Identifying the name of Network Interface

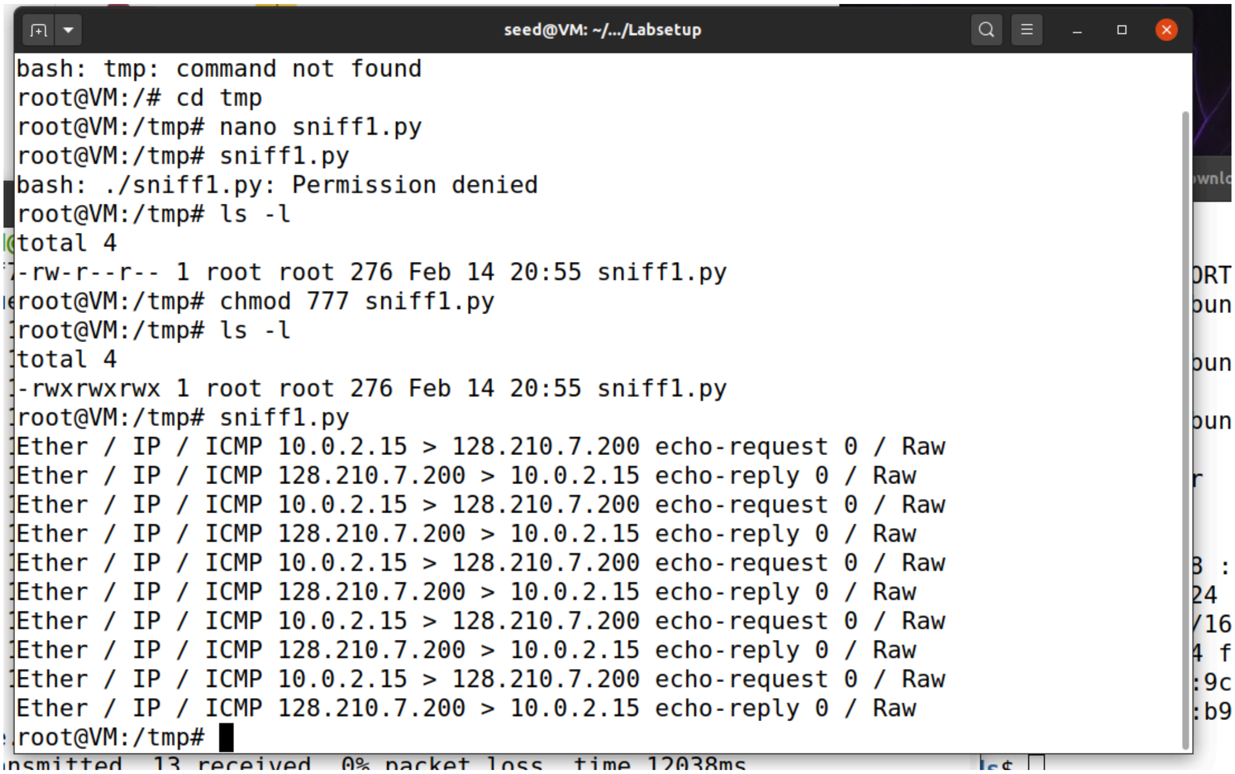

Figure:2- Sniffing ICMP Packets

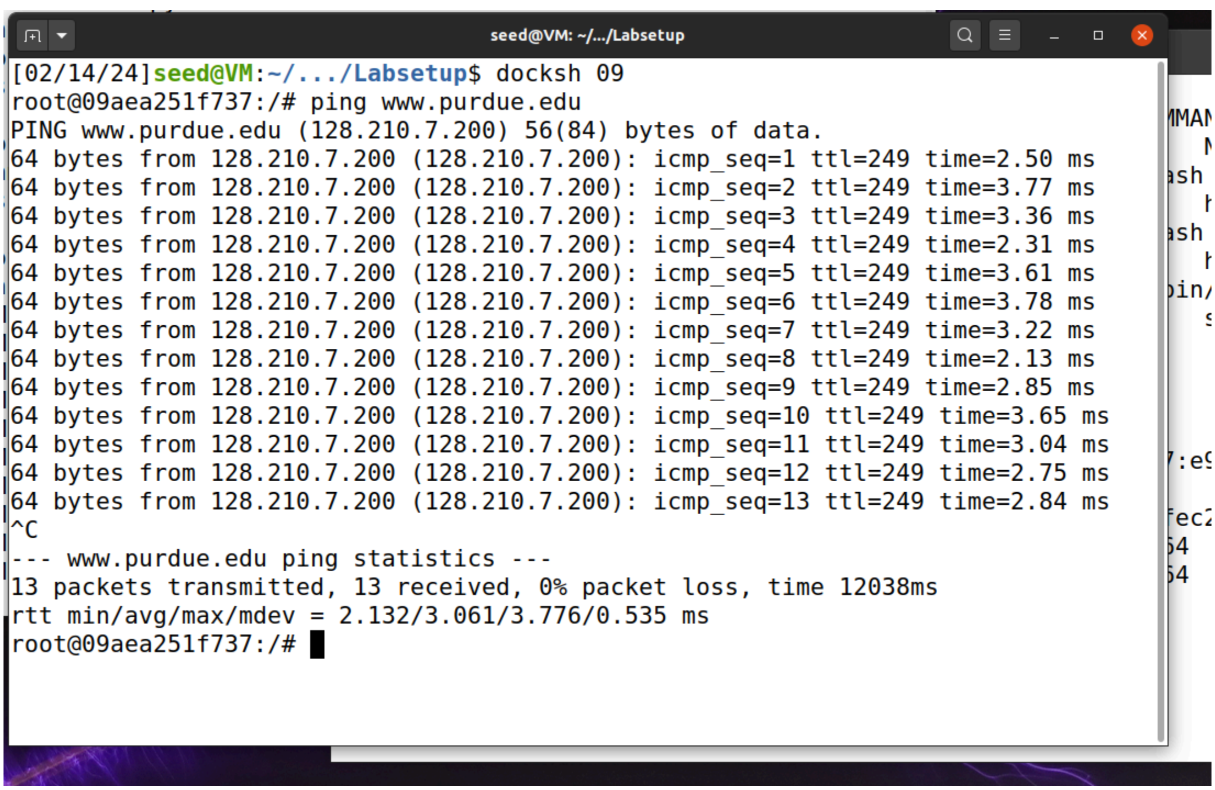

Figure:3- Pinging website from a Host Container for capturing ICMP Packets

Task 1.1 (B)

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

```
seed@VM: ~/.../Labsetup
  GNU nano 4.8                         sniff1.py
#!/usr/bin/python3

from scapy.all import *

#pkt = sniff(iface="enp0s3",filter="icmp", count=10)
pkt=sniff(iface="br-d204ce31d45b", filter="src host 10.9.0.5 and port 23", coun>
#pkt=sniff(iface="br-d204ce31d45b", filter="dst net 128.210.0.0/16", count=10)
pkt.summary()
```

Figure:4- Using scapy to sniff TCP Packets

```
seed@VM: ~

root@09aea251f737:/# exit
exit
[02/14/24]seed@VM:~/.../Labsetup$ docksh 09
root@09aea251f737:/# telnet 10.9.0.1
Trying 10.9.0.1...
Connected to 10.9.0.1.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

Figure:5- Accessing telnet

```
root@VM:/tmp# sniff1.py
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet S
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet A
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet PA / Raw
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet A
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet PA / Raw
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet A
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet PA / Raw
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet A
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet PA / Raw
Ether / IP / TCP 10.9.0.5:56622 > 10.9.0.1:telnet A
```

Figure:6- TCP Packets captured

Task 1.1 (C)

Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.210.0.0/16; you should not pick the subnet that your VM is attached to.

Figure:7- Capturing from a particular subnet

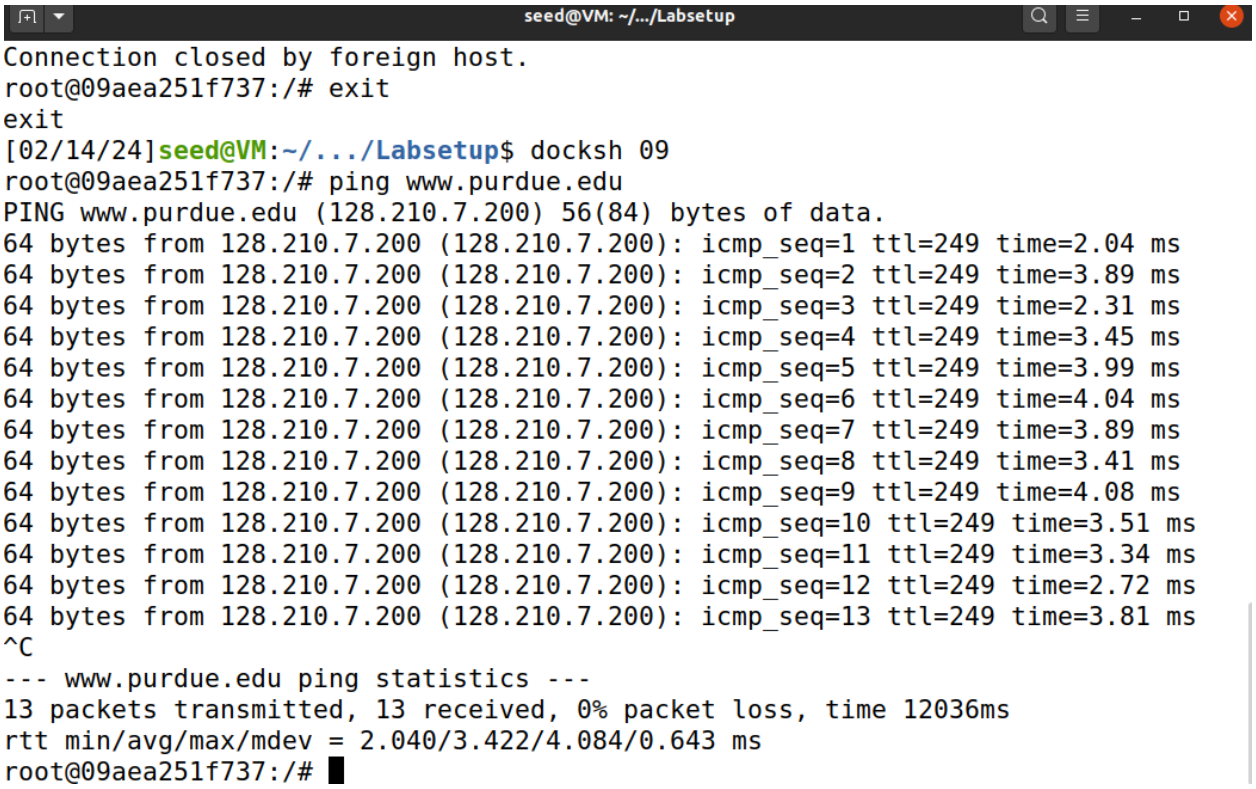

Figure:8- Pinging Website from Host Container

```
root@VM:/tmp# sniff1.py
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 10.9.0.5 > 128.210.7.200 echo-request 0 / Raw
root@VM:/tmp# nano sniff1.py
root@VM:/tmp#
```

Figure:9- ICMP Packets captured from a particular subnet

Task 1.2 Spoofing ICMP packets

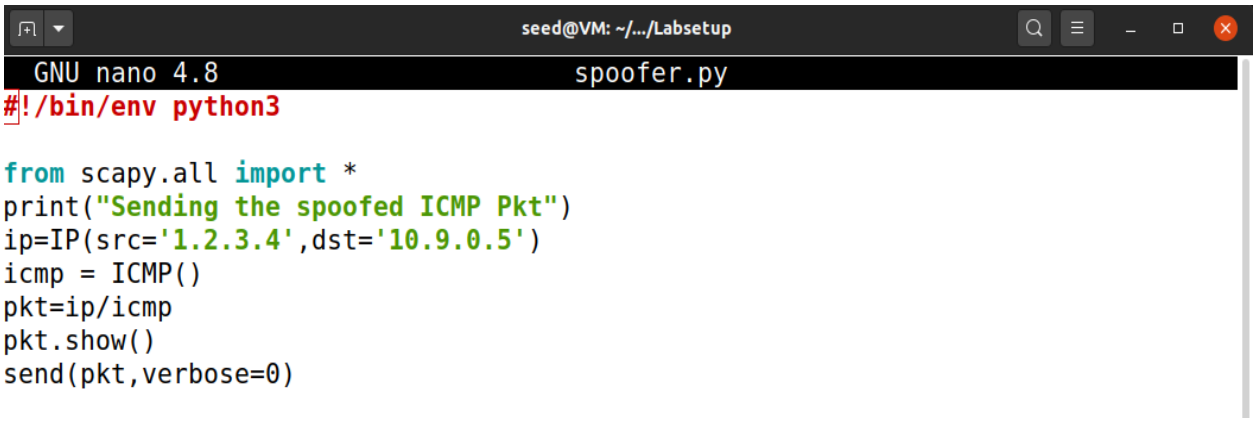

Figure:10- Using scapy to spoof ICMP Packets

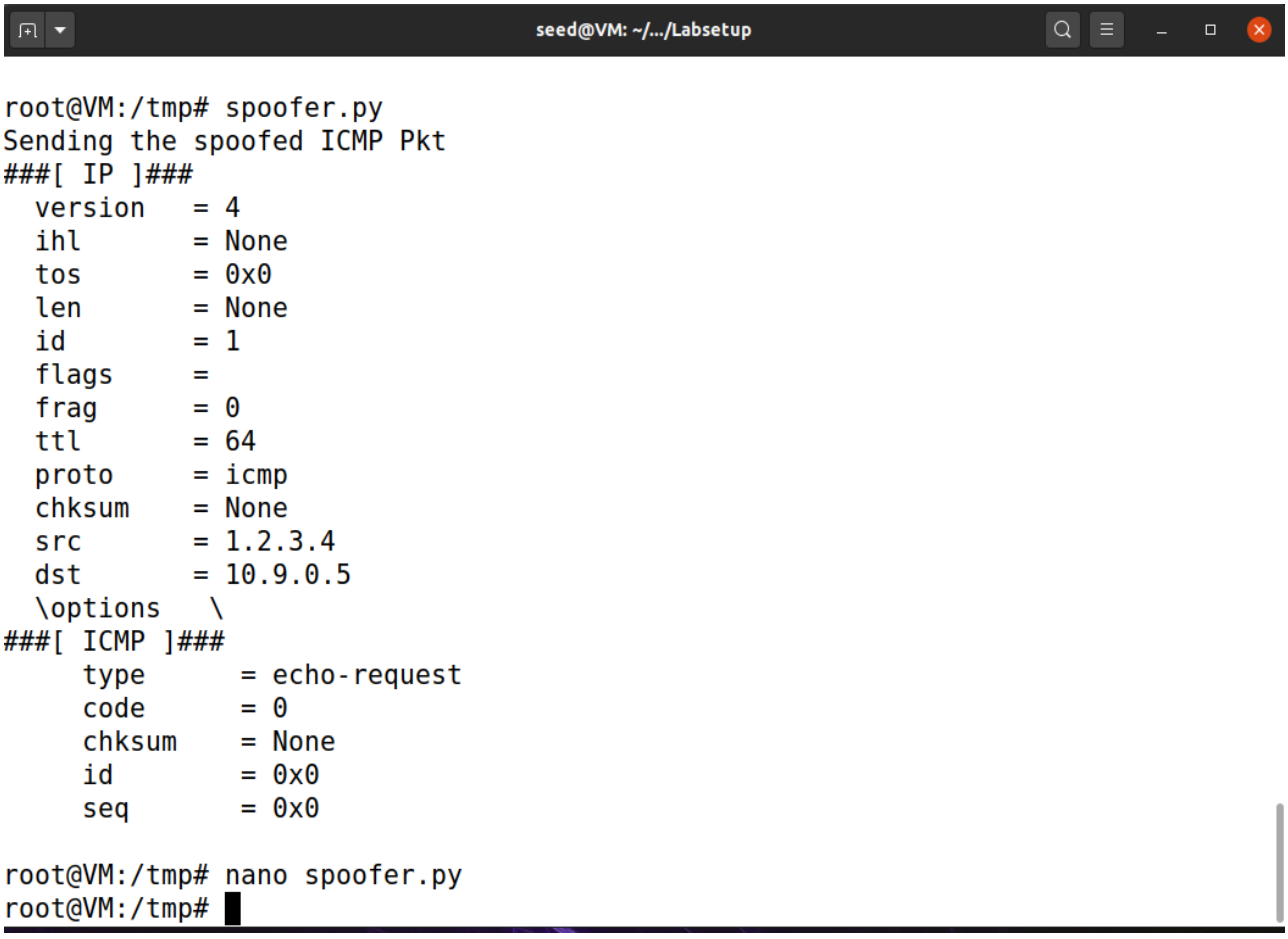

Figure:11- Spoofed ICMP Packets

```
[02/14/24]seed@VM:~/.../Labsetup$ docksh 09
root@09aea251f737:/# tcpdump -n -i eth0 "icmp"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
22:16:16.167505 IP 1.2.3.4 > 10.9.0.5: ICMP echo request, id 0, seq 0, length 8
22:16:16.167578 IP 10.9.0.5 > 1.2.3.4: ICMP echo reply, id 0, seq 0, length 8
```

Figure:12- Using TCP dump in Host container

Task 1.3 Traceroute. Select 8.8.8.8 as the target.

```
  GNU nano 4.8                          tracer.py                          Modified
#!/bin/env python3
import sys
from scapy.all import *

#ip=IP(dst='8.8.8.8')
#icmp = ICMP()
#n = sys.argv[1]
#for i in range(1,int(n)):
#       ip.ttl=int(i)
#       h=sr1(ip/icmp)
#       print("Router: {}".format(h.src))


b = ICMP()
a = IP()
a.dst = '8.8.8.8'

TTL = 4
a.ttl = TTL
h = sr1(a/b, timeout=2, verbose=0)
if h is None:
        print("Router: *** (hops = {})".format(TTL))
else:
        print("Router: {} (hops = {})".format(h.src, TTL))



^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

Figure:12- Using scapy to traceroute

```
root@VM:/# nano tracer.py
root@VM:/# tracer.py
Router: 149.164.180.90 (hops = 4)
root@VM:/#
```

Figure:13- Displaying number of hops and the router

Task 1.4 Sniffing and then Spoofing.

Ping the following 3 IP addresses from the user container (HostA or HostB container).

· **Ping 1.2.3.4 a non-existing host on the Internet**

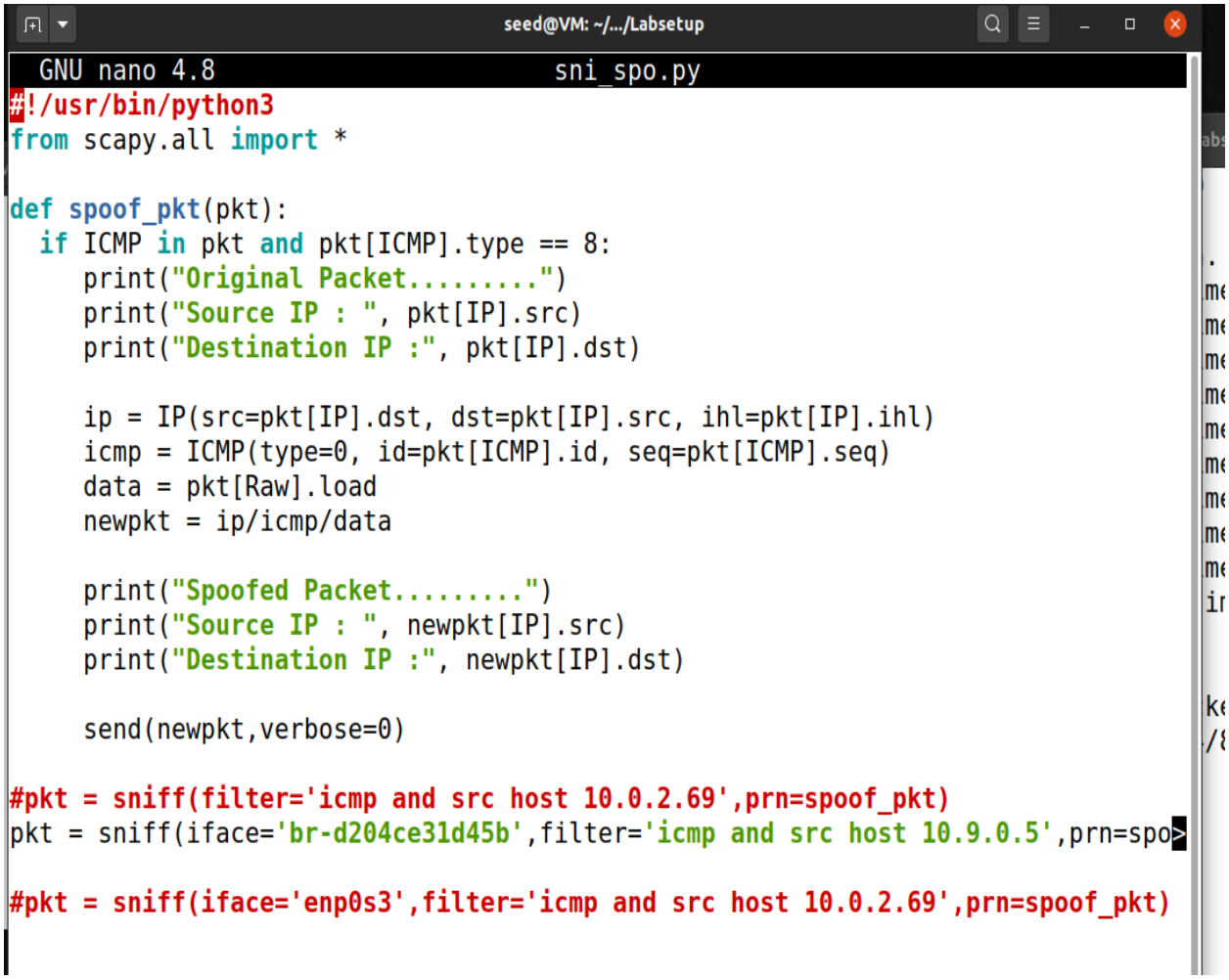

Figure:14- The script to sniff and spoof while pinging the IP Address

```
[02/15/24]seed@VM:~/.../Labsetup$ docksh 09
root@09aea251f737:/# ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=43.7 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=37.2 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=26.5 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=27.9 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=25.1 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=44.5 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=24.6 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=18.8 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=41.0 ms
64 bytes from 1.2.3.4: icmp_seq=10 ttl=64 time=23.9 ms
^C
--- 1.2.3.4 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9045ms
rtt min/avg/max/mdev = 18.806/31.297/44.464/8.866 ms
root@09aea251f737:/#
```

Figure:15- Pinging 1.2.3.4 from a Host container

```
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
```

Figure:16- Spoofed Packets from 1.2.3.4

- **Ping 10.9.0.99 a non-existing host on the LAN**

```
root@09aea251f737:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.5 icmp_seq=1 Destination Host Unreachable
From 10.9.0.5 icmp_seq=2 Destination Host Unreachable
From 10.9.0.5 icmp_seq=3 Destination Host Unreachable
From 10.9.0.5 icmp_seq=4 Destination Host Unreachable
From 10.9.0.5 icmp_seq=5 Destination Host Unreachable
From 10.9.0.5 icmp_seq=6 Destination Host Unreachable
From 10.9.0.5 icmp_seq=7 Destination Host Unreachable
From 10.9.0.5 icmp_seq=8 Destination Host Unreachable
From 10.9.0.5 icmp_seq=9 Destination Host Unreachable
^C
--- 10.9.0.99 ping statistics ---
11 packets transmitted, 0 received, +9 errors, 100% packet loss, time 10237ms
pipe 4
root@09aea251f737:/#
```

Figure:17- Pinging 10.9.0.99 from a Host container

```
root@VM:/tmp# sni_spo.py
```

Figure:18- Spoofed Packets from 10.9.0.99, which is empty

- **Ping 8.8.8.8 an existing host on the Internet**

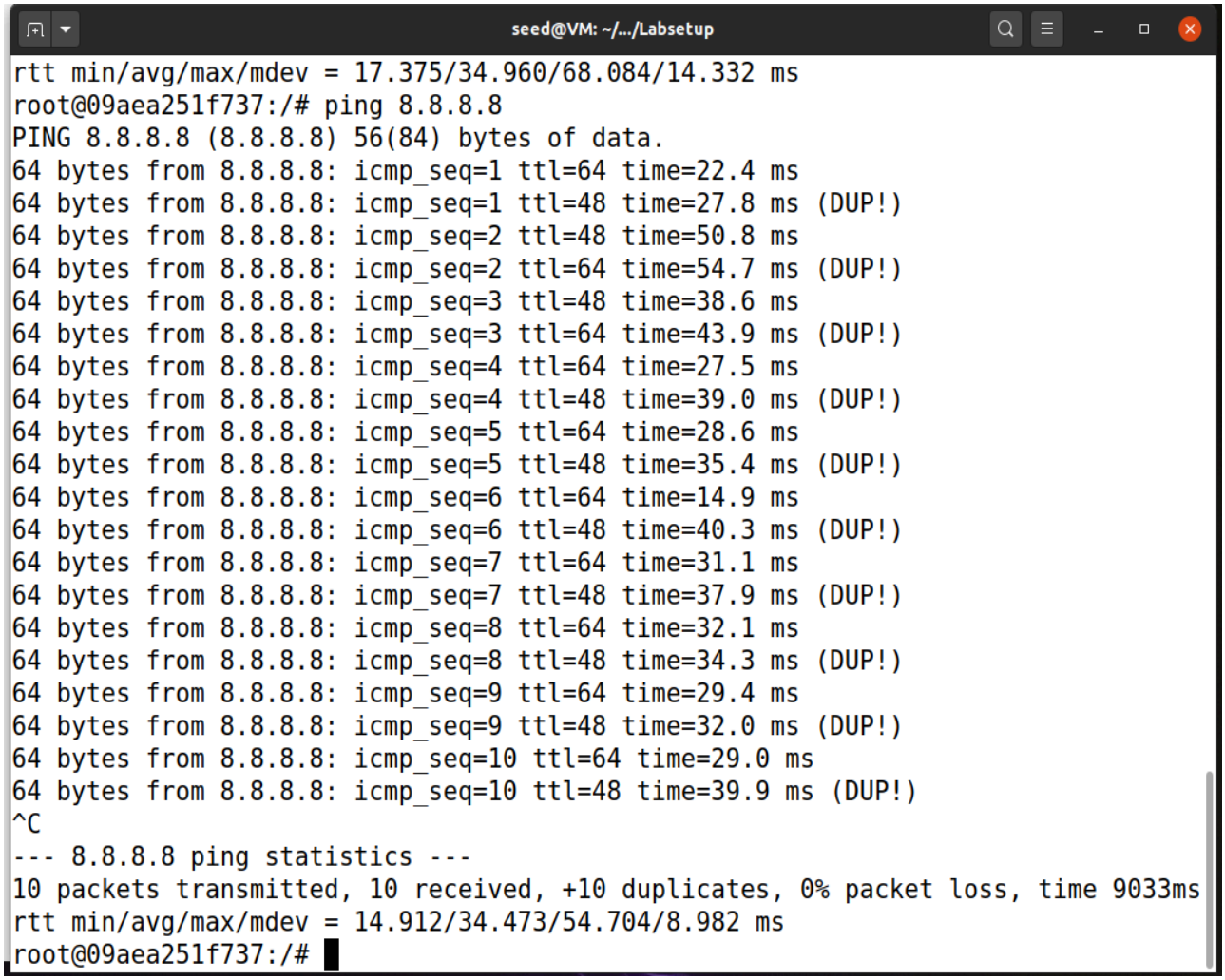

Figure:19- Pinging 8.8.8.8 from a Host container

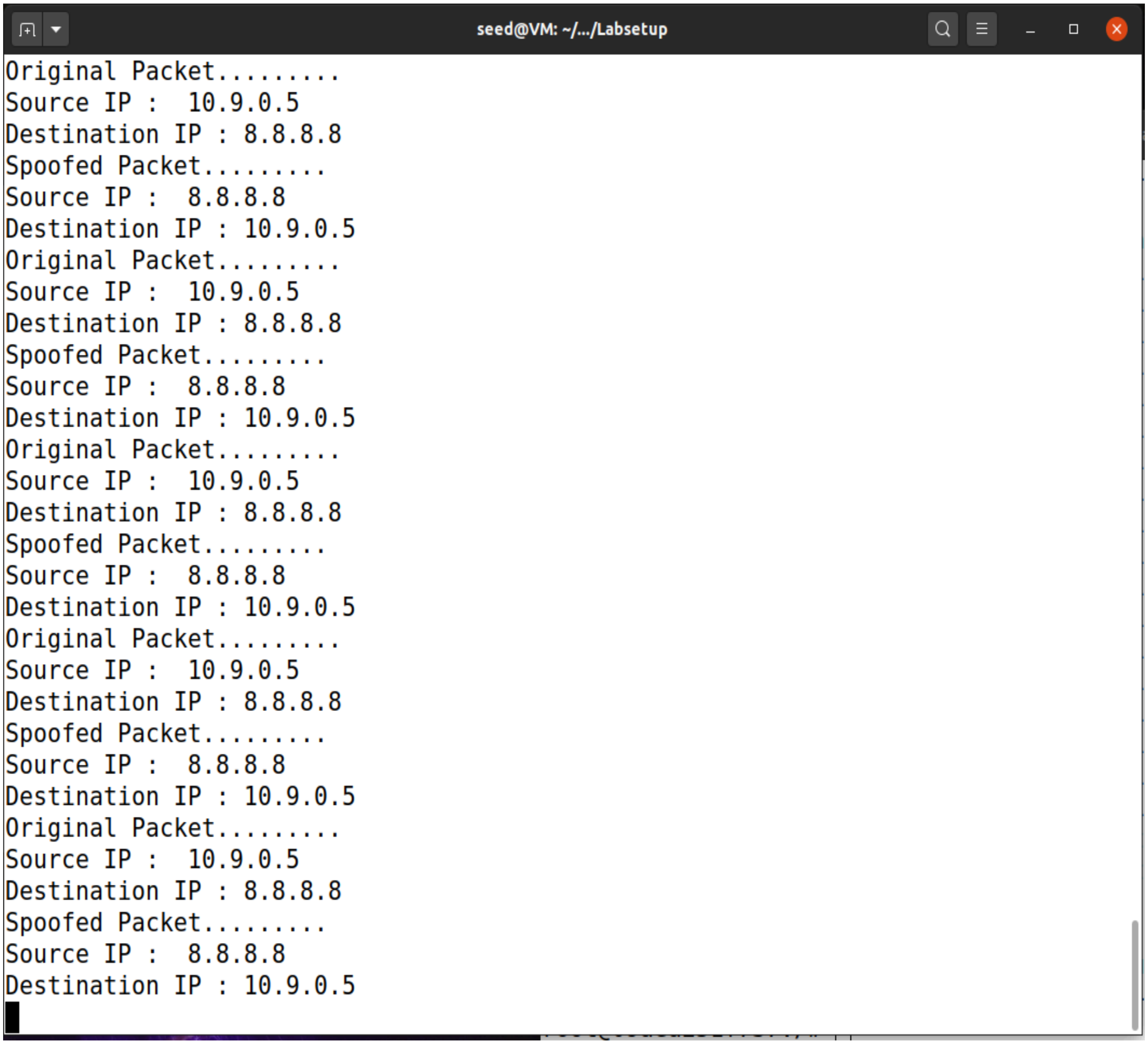

Figure:20- Spoofed Packets from 8.8.8.8
