

Lab2

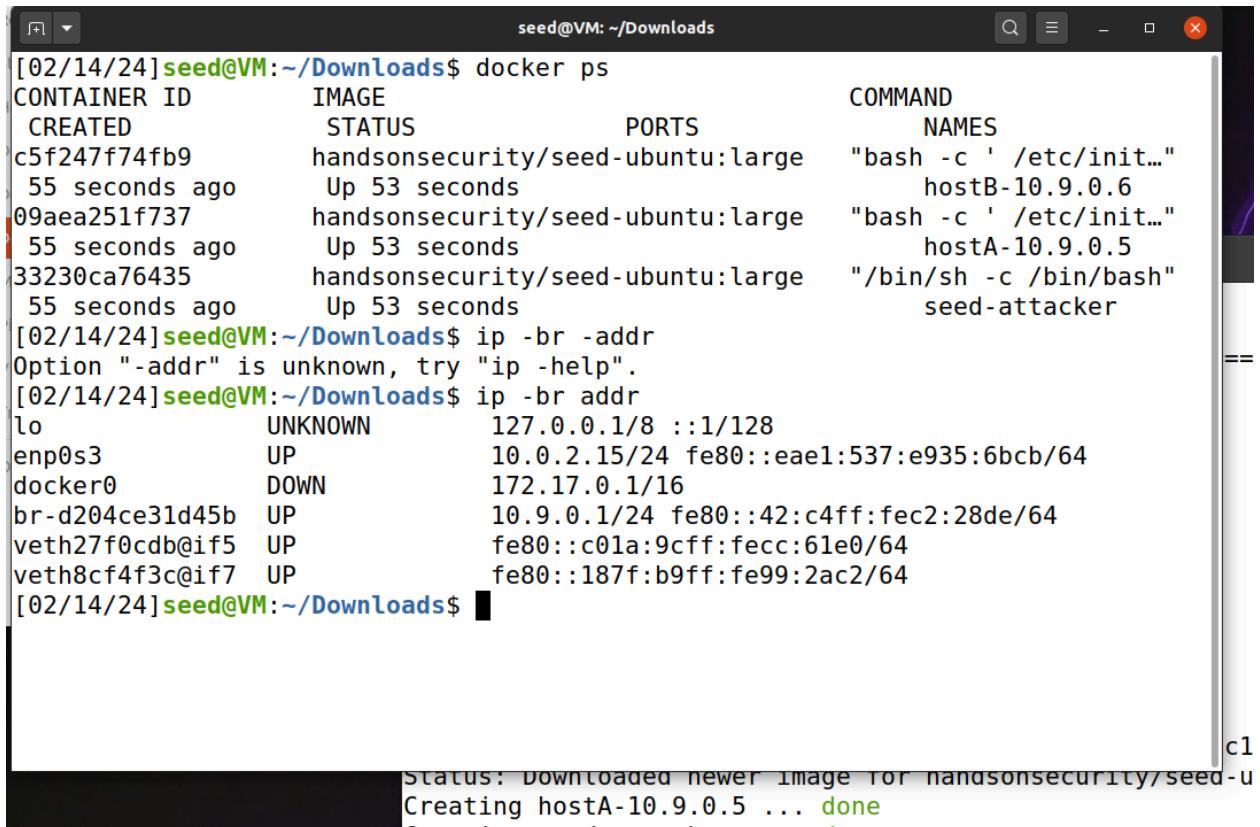
Report:

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



The screenshot shows a terminal window titled "seed@VM: ~/Downloads\$". It displays the output of several commands:

- `docker ps` lists three Docker containers:

CONTAINER ID	IMAGE	COMMAND	NAMES
c5f247f74fb9	handsonsecurity/seed-ubuntu:large	"bash -c '/etc/init..."	hostB-10.9.0.6
09aea251f737	handsonsecurity/seed-ubuntu:large	"bash -c '/etc/init..."	hostA-10.9.0.5
33230ca76435	handsonsecurity/seed-ubuntu:large	"/bin/sh -c /bin/bash"	seed-attacker

- `ip -br -addr` and `ip -br addr` show the list of network interfaces:

INTERFACE	STATE	IP ADDRESS
lo	UNKNOWN	127.0.0.1/8 ::1/128
enp0s3	UP	10.0.2.15/24 fe80::eae1:537:e935:6bcb/64
docker0	DOWN	172.17.0.1/16
br-d204ce31d45b	UP	10.9.0.1/24 fe80::42:c4ff:fec2:28de/64
veth27f0cdb@if5	UP	fe80::c01a:9cff:fecc:61e0/64
veth8cf4f3c@if7	UP	fe80::187f:b9ff:fe99:2ac2/64

- `ip -br` shows the bridge interfaces:

INTERFACE	STATE	IP ADDRESS
br-d204ce31d45b	UP	10.9.0.1/24 fe80::42:c4ff:fec2:28de/64
veth27f0cdb@if5	UP	fe80::c01a:9cff:fecc:61e0/64
veth8cf4f3c@if7	UP	fe80::187f:b9ff:fe99:2ac2/64

- `Creating hostA-10.9.0.5 ... done` indicates the creation of a host interface.

Figure:1- Identifying the name of Network Interface

```
seed@VM: ~/.../Labsetup
bash: tmp: command not found
root@VM:/# cd tmp
root@VM:/tmp# nano sniff1.py
root@VM:/tmp# sniff1.py
bash: ./sniff1.py: Permission denied
root@VM:/tmp# ls -l
total 4
-rw-r--r-- 1 root root 276 Feb 14 20:55 sniff1.py
root@VM:/tmp# chmod 777 sniff1.py
root@VM:/tmp# ls -l
total 4
-rwxrwxrwx 1 root root 276 Feb 14 20:55 sniff1.py
root@VM:/tmp# sniff1.py
Ether / IP / ICMP 10.0.2.15 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 128.210.7.200 > 10.0.2.15 echo-reply 0 / Raw
Ether / IP / ICMP 10.0.2.15 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 128.210.7.200 > 10.0.2.15 echo-reply 0 / Raw
Ether / IP / ICMP 10.0.2.15 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 128.210.7.200 > 10.0.2.15 echo-reply 0 / Raw
Ether / IP / ICMP 10.0.2.15 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 128.210.7.200 > 10.0.2.15 echo-reply 0 / Raw
Ether / IP / ICMP 10.0.2.15 > 128.210.7.200 echo-request 0 / Raw
Ether / IP / ICMP 128.210.7.200 > 10.0.2.15 echo-reply 0 / Raw
root@VM:/tmp#
```

Figure:2- Sniffing ICMP Packets

```
[02/14/24]seed@VM:~/.../Labsetup$ docksh 09
root@09aea251f737:/# ping www.purdue.edu
PING www.purdue.edu (128.210.7.200) 56(84) bytes of data.
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=1 ttl=249 time=2.50 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=2 ttl=249 time=3.77 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=3 ttl=249 time=3.36 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=4 ttl=249 time=2.31 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=5 ttl=249 time=3.61 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=6 ttl=249 time=3.78 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=7 ttl=249 time=3.22 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=8 ttl=249 time=2.13 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=9 ttl=249 time=2.85 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=10 ttl=249 time=3.65 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=11 ttl=249 time=3.04 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=12 ttl=249 time=2.75 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=13 ttl=249 time=2.84 ms
^C
--- www.purdue.edu ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12038ms
rtt min/avg/max/mdev = 2.132/3.061/3.776/0.535 ms
root@09aea251f737:/#
```

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.



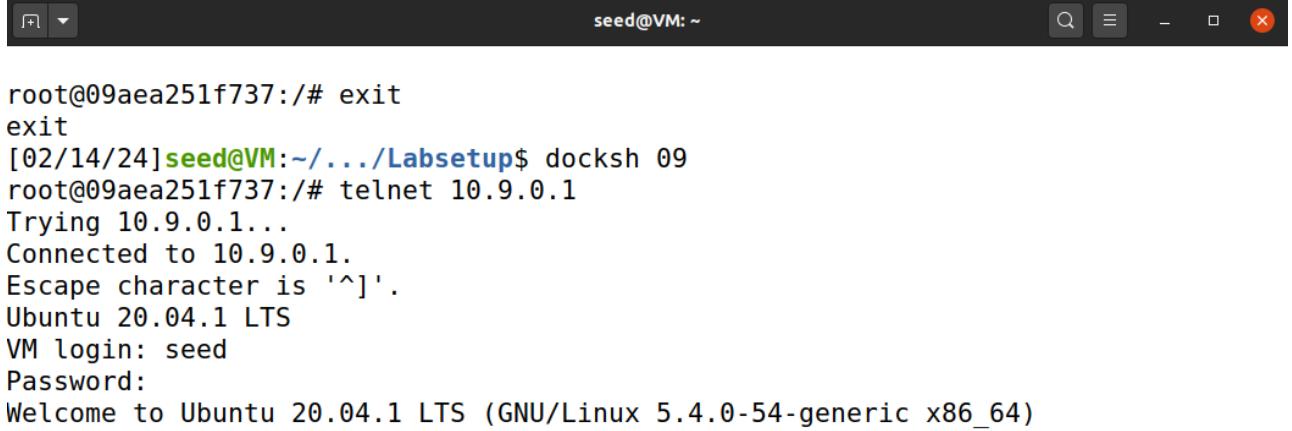
The screenshot shows a terminal window titled "seed@VM: ~.../Labsetup". The file being edited is "sniff1.py". The code uses the scapy library to sniff TCP packets on interface "br-d204ce31d45b" with a filter for source host 10.9.0.5 and port 23, or destination port 23. It also includes a commented-out line for destination network 128.210.0.0/16. The code ends with a call to pkt.summary().

```
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()
```

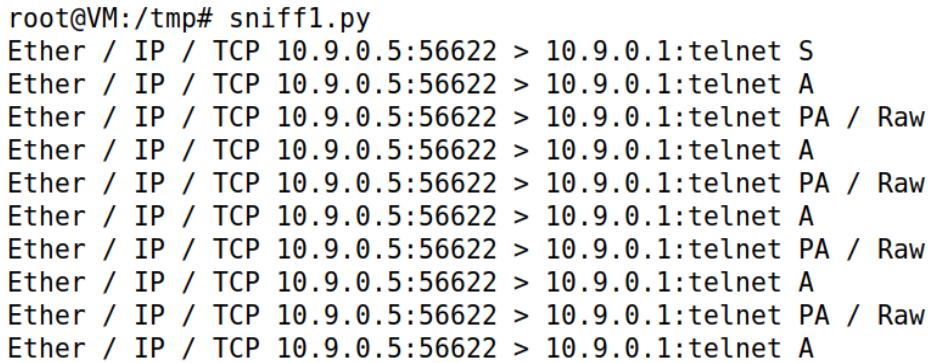
Figure:4- Using scapy to sniff TCP Packets



The screenshot shows a terminal window titled "seed@VM: ~". It displays a telnet session to an Ubuntu 20.04.1 LTS VM. The user logs in as root, providing the password "seed". The session ends with a welcome message for the Ubuntu 20.04.1 LTS distribution.

```
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



The screenshot shows a terminal window titled "root@VM:/tmp#". The command "sniff1.py" is run, and the output lists several captured TCP packets. The packets show traffic between the local machine (IP 10.9.0.5) and the Ubuntu VM (IP 10.9.0.1), specifically telnet connections on port 23.

```
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.

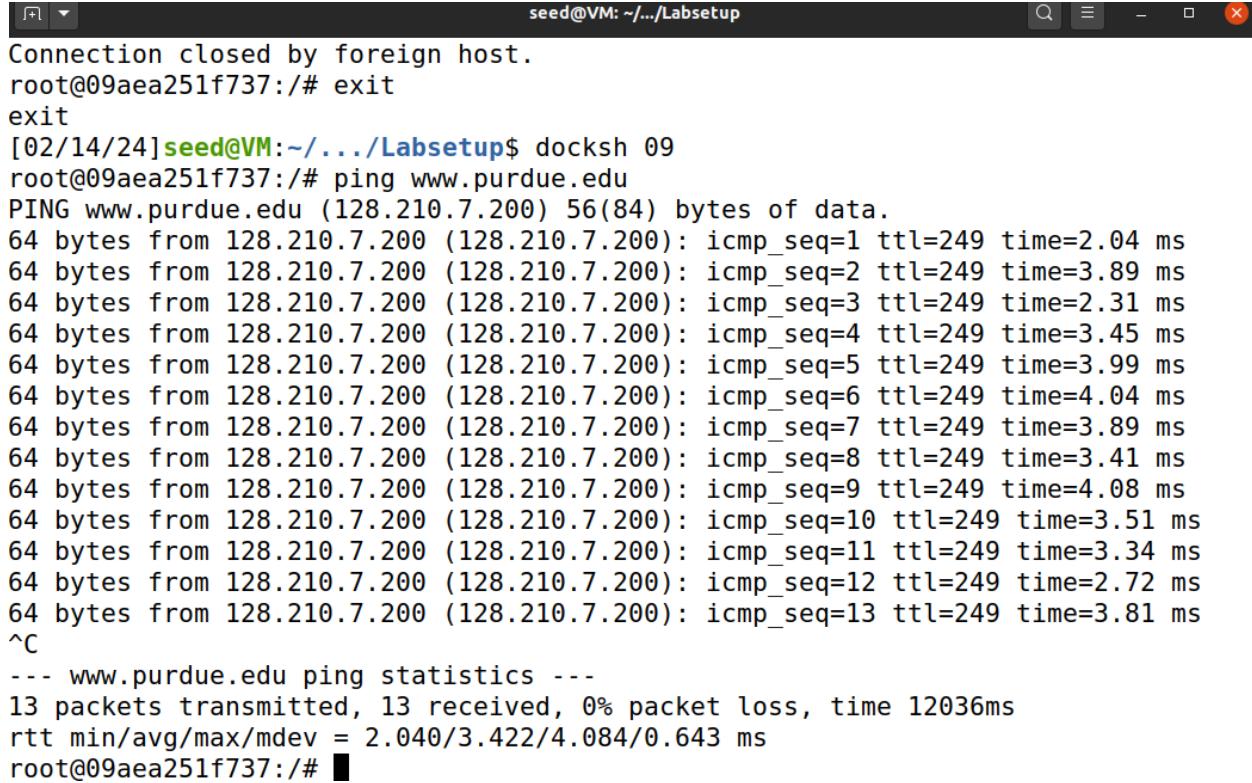


```
seed@VM: ~/.../Labsetup
GNU nano 4.8          sniff1.py
#!/usr/bin/python3

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", cou>
pkt=sniff(iface="br-d204ce31d45b", filter="dst net 128.210.0.0/16", count=10)
pkt.summary()
```

Figure:7- Capturing from a particular subnet



```
seed@VM: ~/.../Labsetup
Connection closed by foreign host.
root@09aea251f737:/# exit
exit
[02/14/24]seed@VM:~/.../Labsetup$ docksh 09
root@09aea251f737:/# ping www.purdue.edu
PING www.purdue.edu (128.210.7.200) 56(84) bytes of data.
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=1 ttl=249 time=2.04 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=2 ttl=249 time=3.89 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=3 ttl=249 time=2.31 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=4 ttl=249 time=3.45 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=5 ttl=249 time=3.99 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=6 ttl=249 time=4.04 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=7 ttl=249 time=3.89 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=8 ttl=249 time=3.41 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=9 ttl=249 time=4.08 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=10 ttl=249 time=3.51 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=11 ttl=249 time=3.34 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=12 ttl=249 time=2.72 ms
64 bytes from 128.210.7.200 (128.210.7.200): icmp_seq=13 ttl=249 time=3.81 ms
^C
--- www.purdue.edu ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12036ms
rtt min/avg/max/mdev = 2.040/3.422/4.084/0.643 ms
root@09aea251f737:/# █
```

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
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

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

from scapy.all import *
print("Sending the spoofed ICMP Pkt")
ip=IP(src='1.2.3.4',dst='10.9.0.5')
icmp = ICMP()
pkt=ip/icmp
pkt.show()
send(pkt,verbose=0)
```

Figure:10- Using scapy to spoof ICMP Packets

```
root@VM:/tmp# spoofer.py
Sending the spoofed ICMP Pkt
###[ IP ]###
version    = 4
ihl        = None
tos        = 0x0
len        = None
id         = 1
flags      =
frag       = 0
ttl        = 64
proto      = icmp
chksum     = None
src        = 1.2.3.4
dst        = 10.9.0.5
\options   \
###[ ICMP ]###
type       = echo-request
code       = 0
chksum     = None
id         = 0x0
seq        = 0x0

root@VM:/tmp# nano spoofer.py
root@VM:/tmp#
```

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.

The screenshot shows a terminal window with the title bar "GNU nano 4.8" and the file name "tracer.py". The code in the editor is a Python script for performing a traceroute. It imports sys and scapy.all, defines a function to print routers based on TTL, creates IP and ICMP objects, sets the destination to '8.8.8.8', and performs the traceroute with a TTL of 4. The terminal also shows the nano editor's command-line interface at the bottom.

```
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

The screenshot shows a terminal window with the command "root@VM:~# nano tracer.py" followed by "root@VM:~# tracer.py". The output shows the traceroute results: "Router: 149.164.180.90 (hops = 4)".

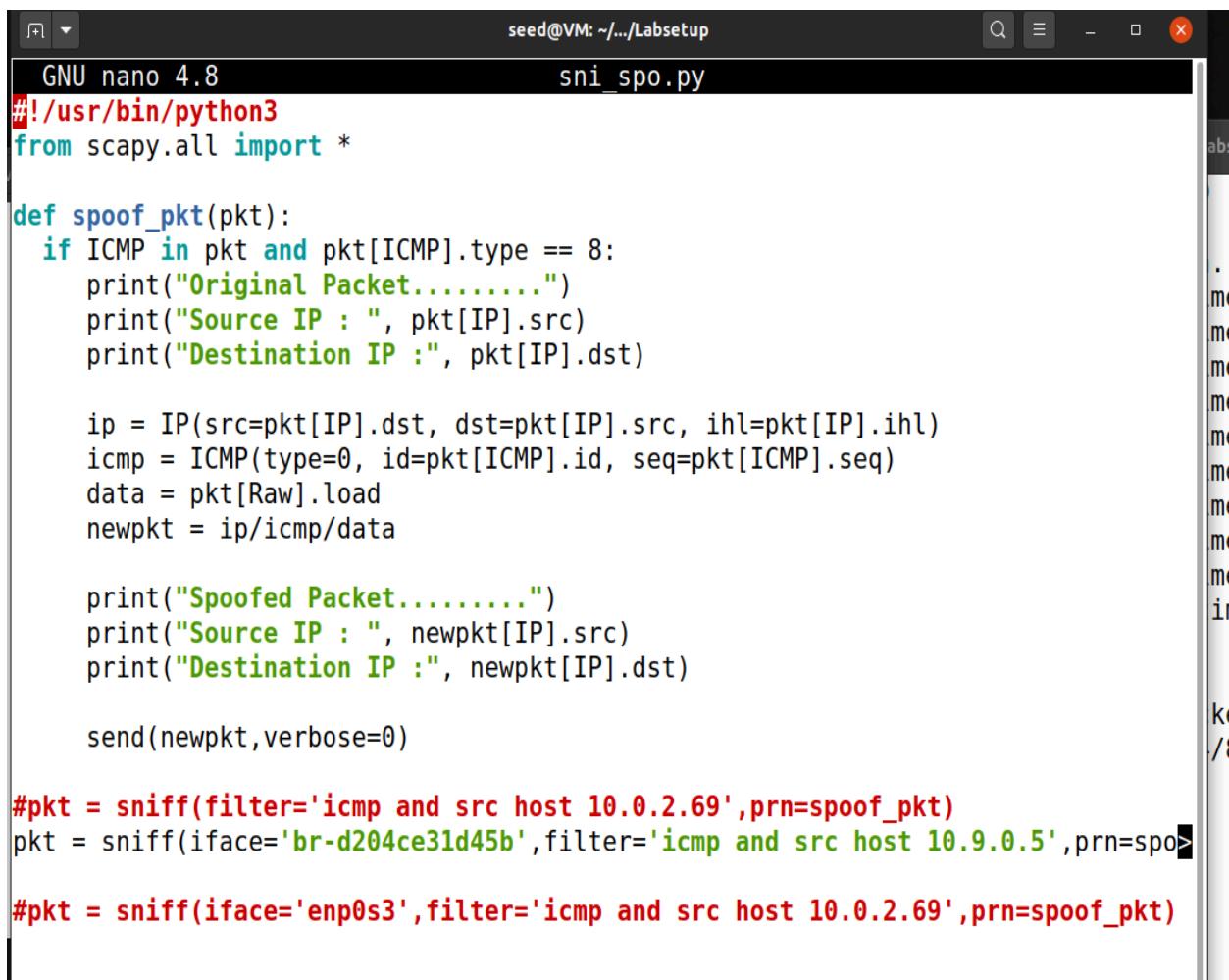
```
root@VM:~# nano tracer.py
root@VM:~# tracer.py
Router: 149.164.180.90 (hops = 4)
root@VM:~#
```

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



```
seed@VM: ~/.../Labsetup
GNU nano 4.8          sni_spo.py
#!/usr/bin/python3
from scapy.all import *

def spoof_pkt(pkt):
    if ICMP in pkt and pkt[ICMP].type == 8:
        print("Original Packet.....")
        print("Source IP : ", pkt[IP].src)
        print("Destination IP : ", pkt[IP].dst)

        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
        icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        newpkt = ip/icmp/data

        print("Spoofed Packet.....")
        print("Source IP : ", newpkt[IP].src)
        print("Destination IP : ", newpkt[IP].dst)

        send(newpkt, verbose=0)

#pkt = sniff(filter='icmp and src host 10.0.2.69',prn=spoof_pkt)
pkt = sniff(iface='br-d204ce31d45b',filter='icmp and src host 10.9.0.5',prn=spo>
#pkt = sniff(iface='enp0s3',filter='icmp and src host 10.0.2.69',prn=spoof_pkt)
```

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

```
[+] seed@VM: ~/.../Labsetup - X  
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

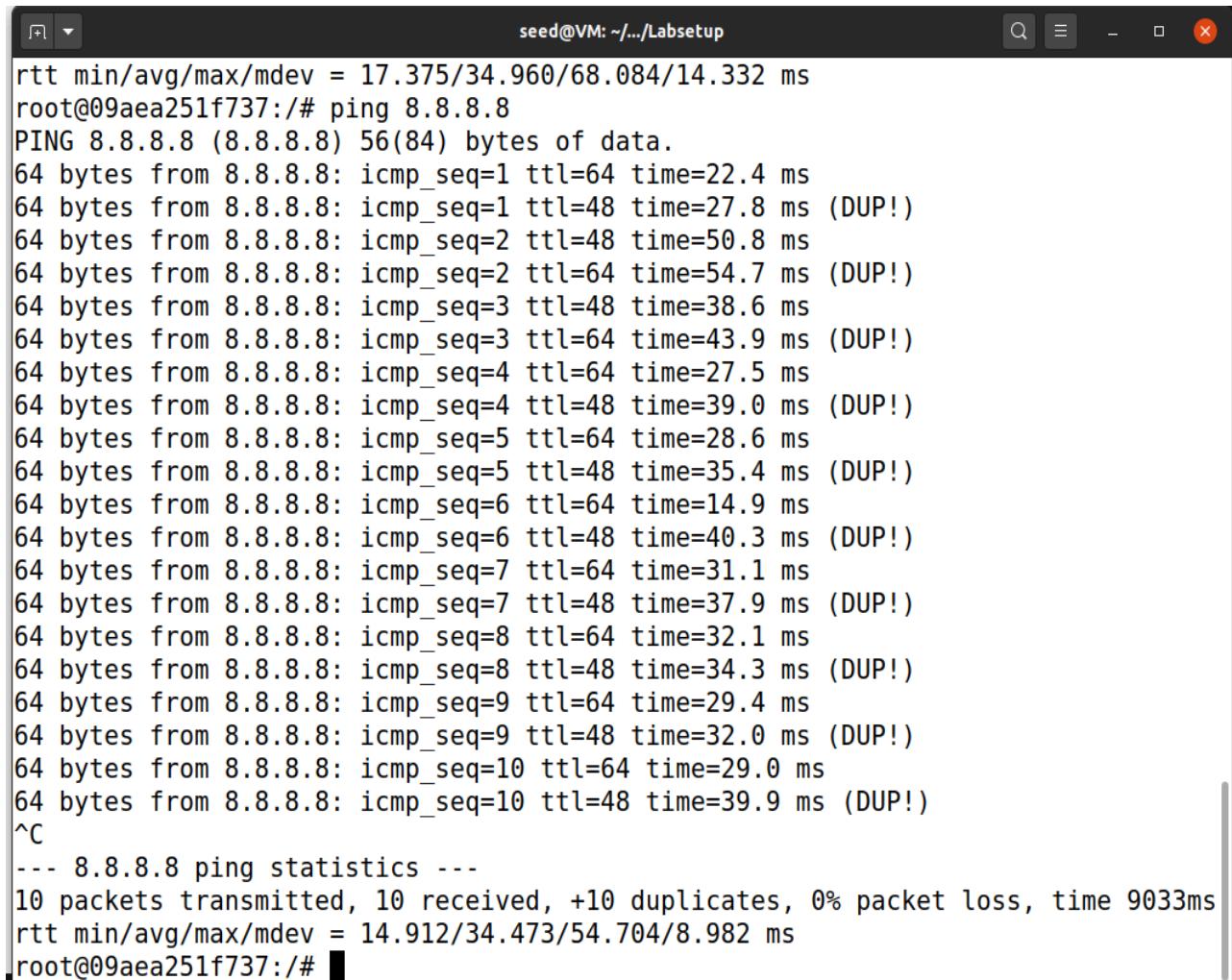
```
root@09aea251f737:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.5 icmp_seq=1 Destination Host Unreachable
From 10.9.0.5 icmp_seq=2 Destination Host Unreachable
From 10.9.0.5 icmp_seq=3 Destination Host Unreachable
From 10.9.0.5 icmp_seq=4 Destination Host Unreachable
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**



```
seed@VM: ~/.../Labsetup
rtt min/avg/max/mdev = 17.375/34.960/68.084/14.332 ms
root@09aea251f737:/# 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=64 time=22.4 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=48 time=27.8 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=48 time=50.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=54.7 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=48 time=38.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=43.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=27.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=48 time=39.0 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=28.6 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=48 time=35.4 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=14.9 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=48 time=40.3 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=31.1 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=48 time=37.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=8 ttl=64 time=32.1 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=48 time=34.3 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=9 ttl=64 time=29.4 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=48 time=32.0 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=10 ttl=64 time=29.0 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=48 time=39.9 ms (DUP!)
^C
--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, +10 duplicates, 0% packet loss, time 9033ms
rtt min/avg/max/mdev = 14.912/34.473/54.704/8.982 ms
root@09aea251f737:/#
```

Figure:19- Pinging 8.8.8.8 from a Host container

```
seed@VM: ~/.../Labsetup
Original Packet.....
Source IP : 10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.....
Source IP : 8.8.8.8
Destination IP : 10.9.0.5
Original Packet.....
Source IP : 10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.....
Source IP : 8.8.8.8
Destination IP : 10.9.0.5
Original Packet.....
Source IP : 10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.....
Source IP : 8.8.8.8
Destination IP : 10.9.0.5
Original Packet.....
Source IP : 10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.....
Source IP : 8.8.8.8
Destination IP : 10.9.0.5
Original Packet.....
Source IP : 10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.....
Source IP : 8.8.8.8
Destination IP : 10.9.0.5
Original Packet.....
Source IP : 10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.....
Source IP : 8.8.8.8
Destination IP : 10.9.0.5
```

Figure:20- Spoofed Packets from 8.8.8.8
