Network Protocol Structures & ICMP Redirect Attack

February 2024

Abdulfatah Abdillahi

Contents

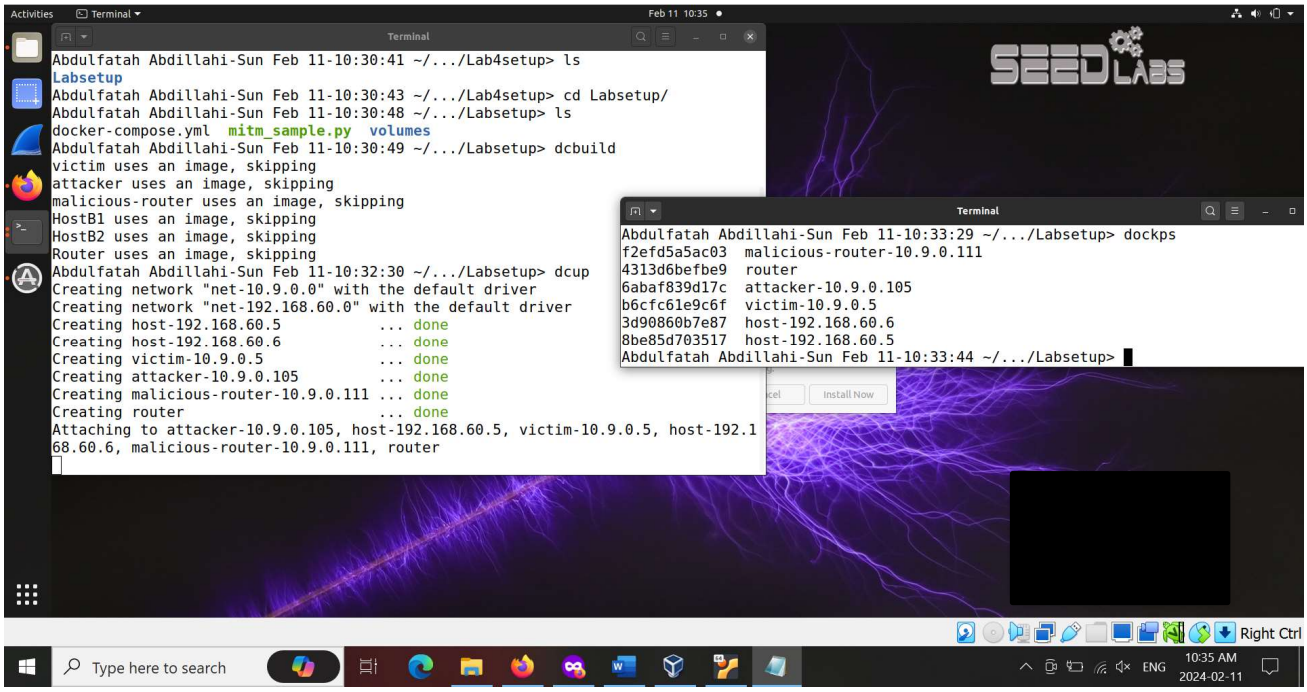
Introduction

In this lab we explored some topics like MITM (Man in The Middle) attacks using ICMP redirect messages as well as learning to decode different encoded text using the Cyberchef tool.

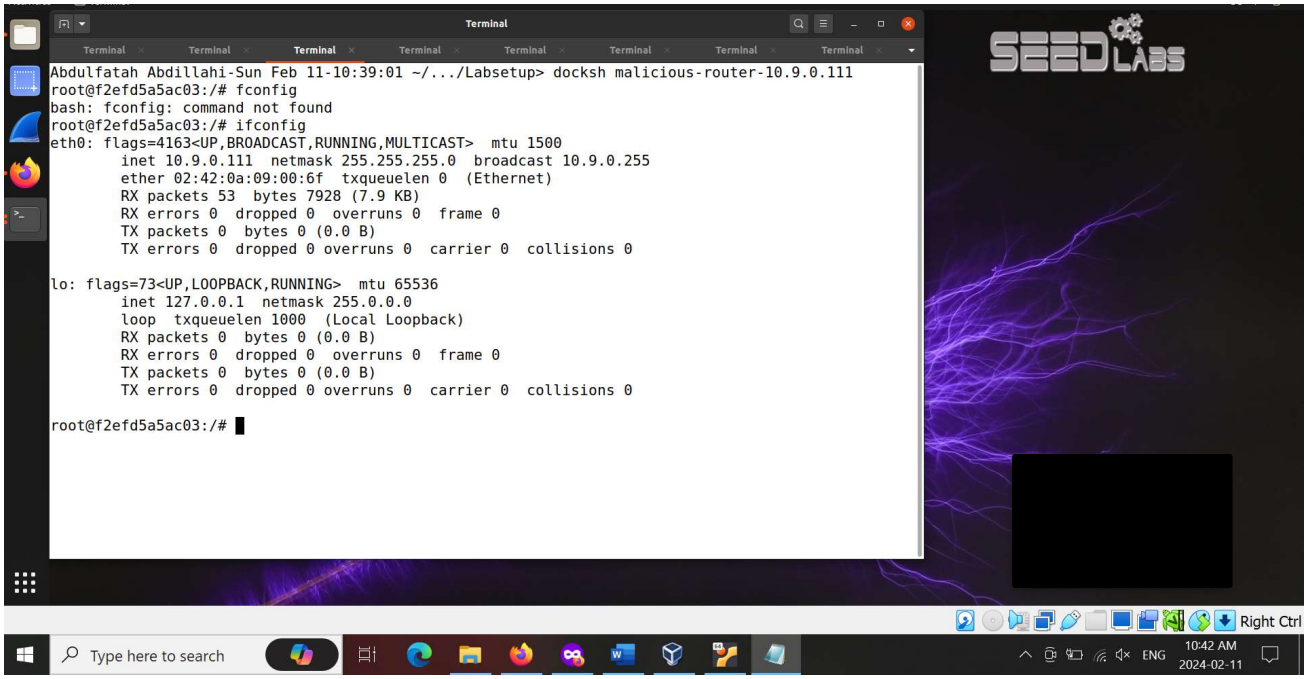
Part 1: SEED Lab ARP Cache Poisoning Attack

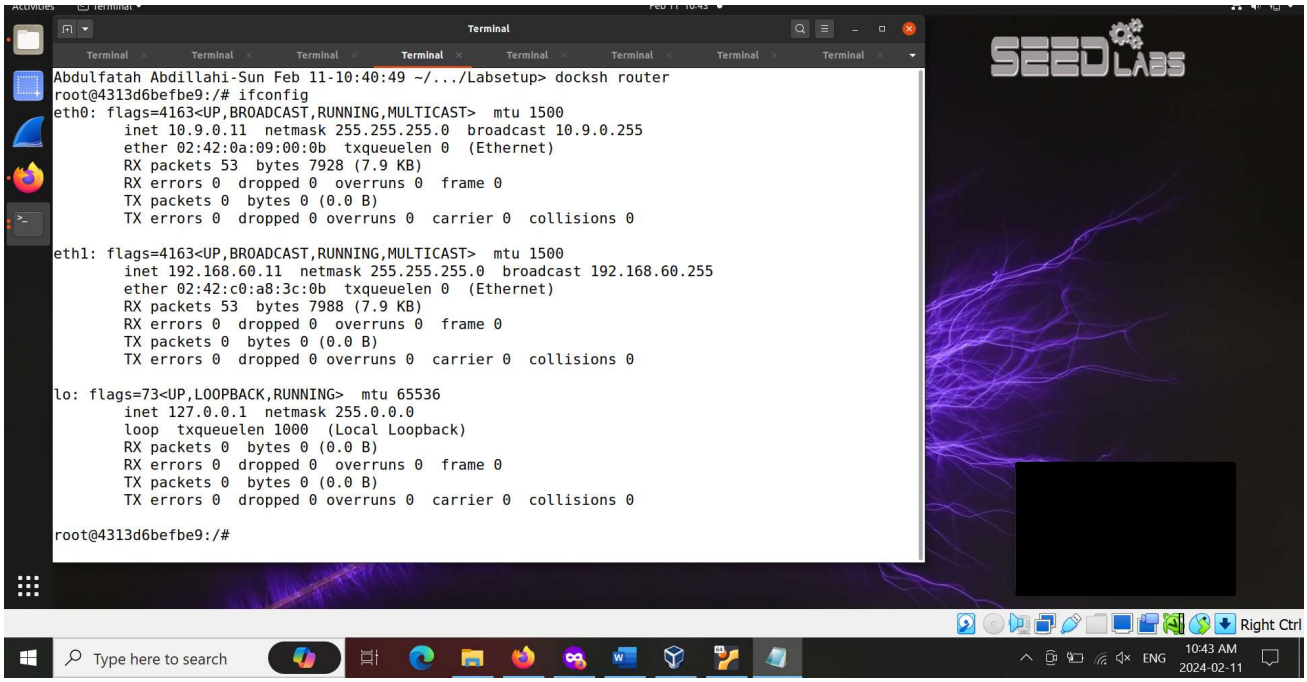
Setting up the Environment

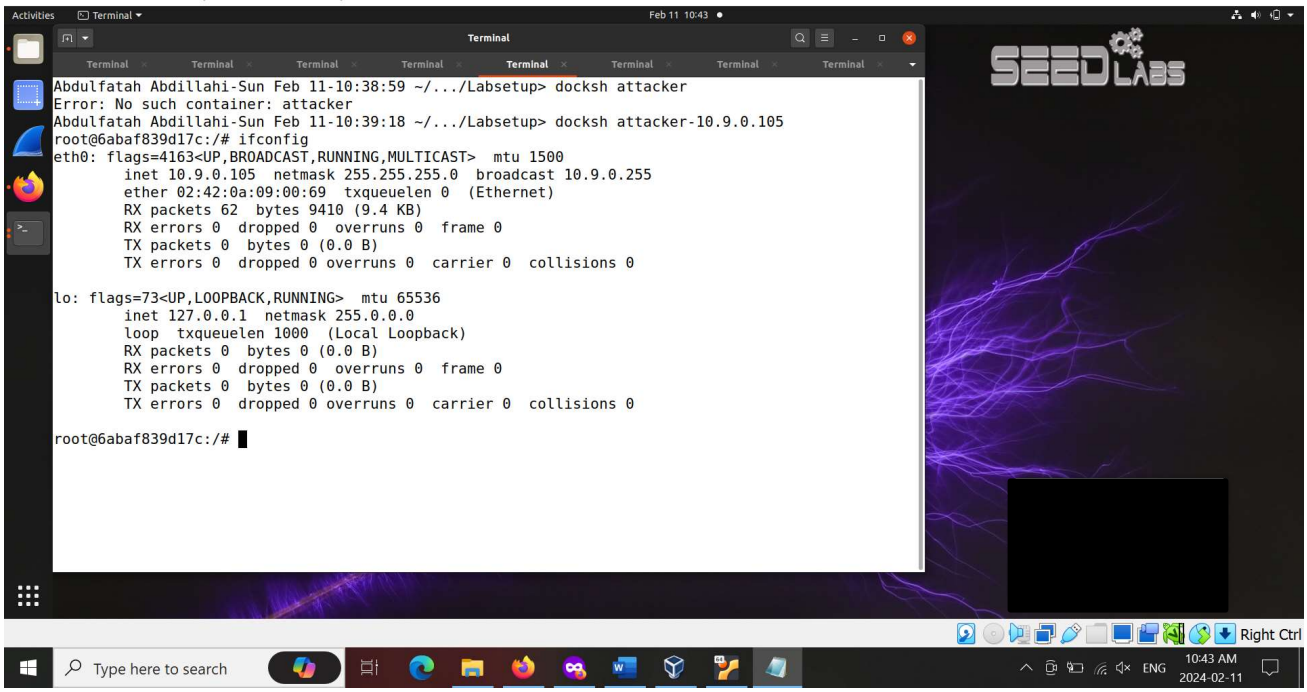
Creating six docker containers

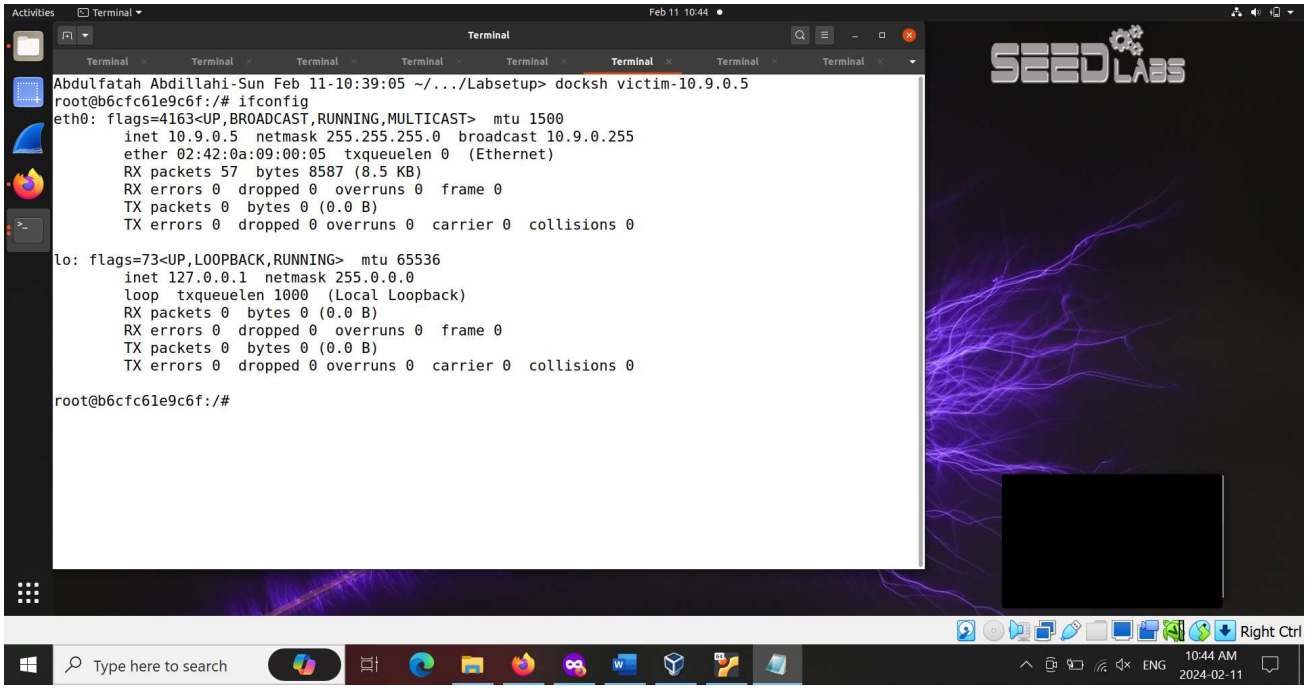


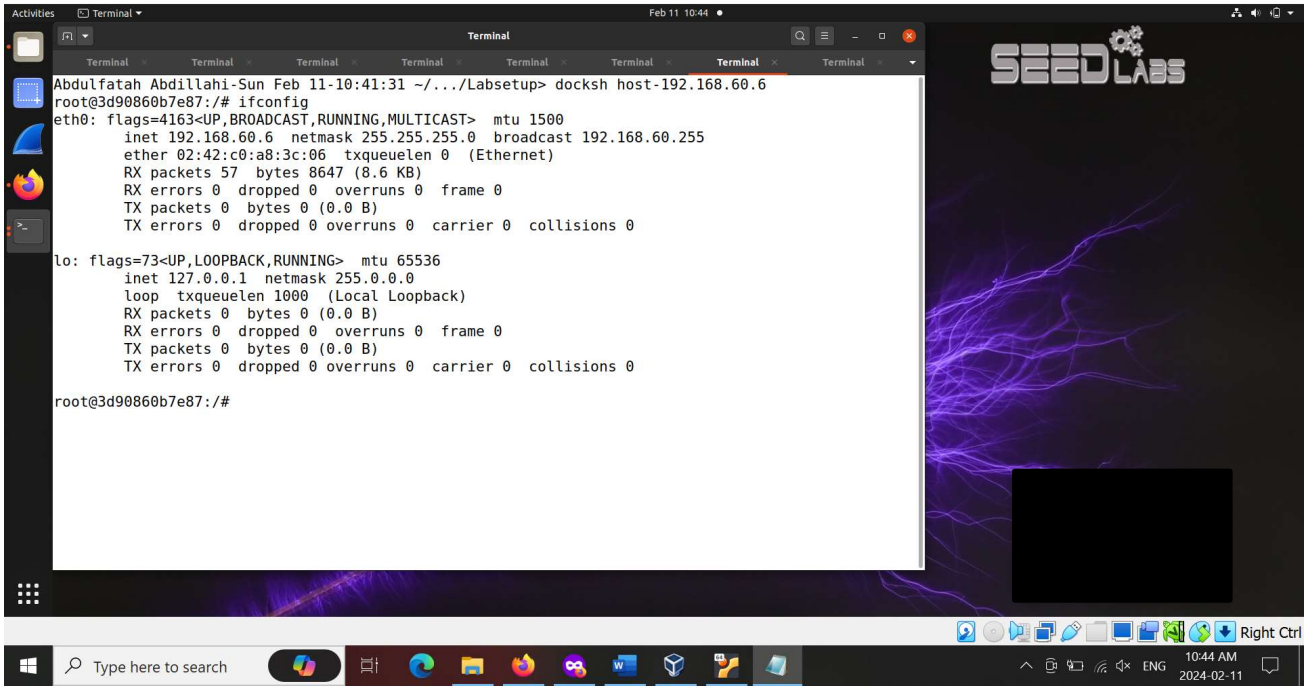
Checking the IP addresses for each of the six containers (malicious-router, router, attacker, victim, host 60.5, and host 60.6, respectively).

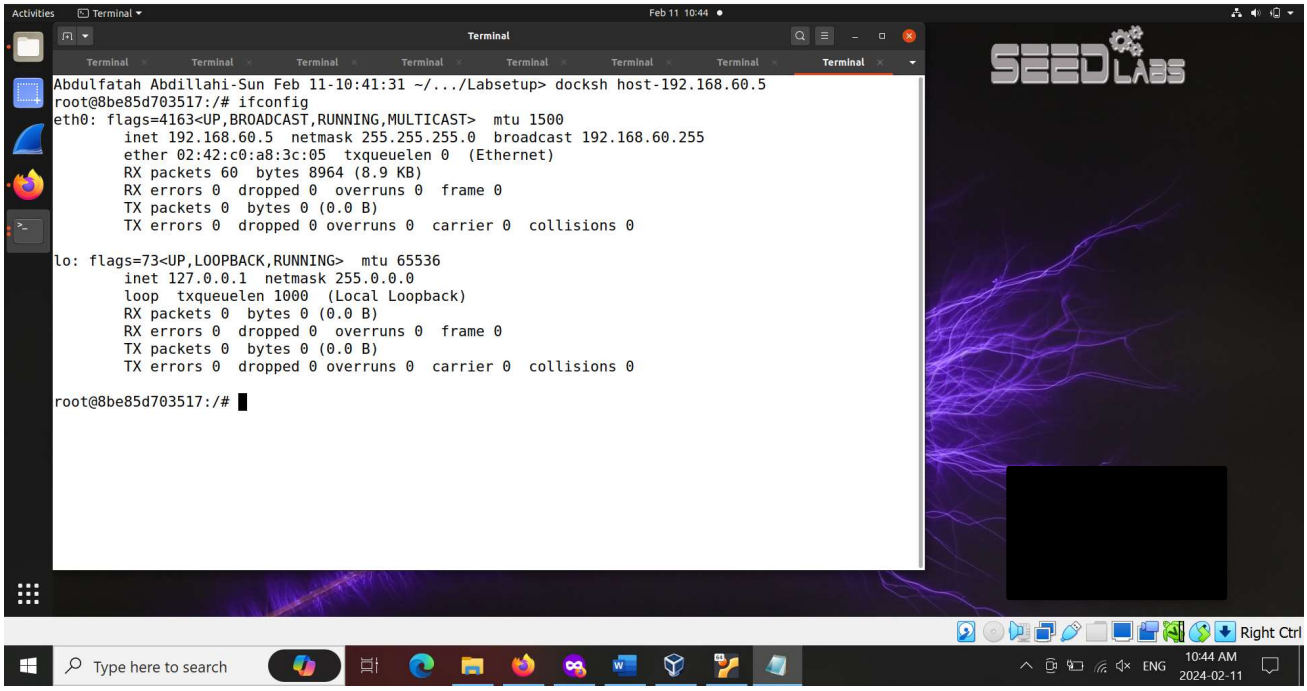

Terminal
Abdulfatah Abdillahi-Sun Feb 11-10:39:01 ~/.../Labsetup> docksh malicious-router-10.9.0.111
root@f2efd5a5ac03:/# fconfig
bash: fconfig: command not found
root@f2efd5a5ac03:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.9.0.111 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:6f txqueuelen 0 (Ethernet)
RX packets 53 bytes 7928 (7.9 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
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
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
root@f2efd5a5ac03:/#
SEED LABS
Right Ctrl
Type here to search
ENG
10:42 AM
2024-02-11


Terminal
Abdulfatah Abdillahi-Sun Feb 11-10:40:49 ~/.../Labsetup> docksh router
root@4313d6befbe9:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.9.0.11 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:0b txqueuelen 0 (Ethernet)
RX packets 53 bytes 7928 (7.9 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
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.60.11 netmask 255.255.255.0 broadcast 192.168.60.255
ether 02:42:c0:a8:3c:0b txqueuelen 0 (Ethernet)
RX packets 53 bytes 7988 (7.9 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
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
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
root@4313d6befbe9:/#
SEED LABS
Right Ctrl
Type here to search
ENG
10:43 AM
2024-02-11


Activities
Terminal
Feb 11 10:43
Terminal
Abdulfatah Abdillahi-Sun Feb 11-10:38:59 ~/.../Labsetup> docksh attacker
Error: No such container: attacker
Abdulfatah Abdillahi-Sun Feb 11-10:39:18 ~/.../Labsetup> docksh attacker-10.9.0.105
root@6abaf839d17c:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.9.0.105 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:69 txqueuelen 0 (Ethernet)
RX packets 62 bytes 9410 (9.4 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
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
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
root@6abaf839d17c:/#
SEED LABS
Right Ctrl
Type here to search
ENG
10:43 AM
2024-02-11

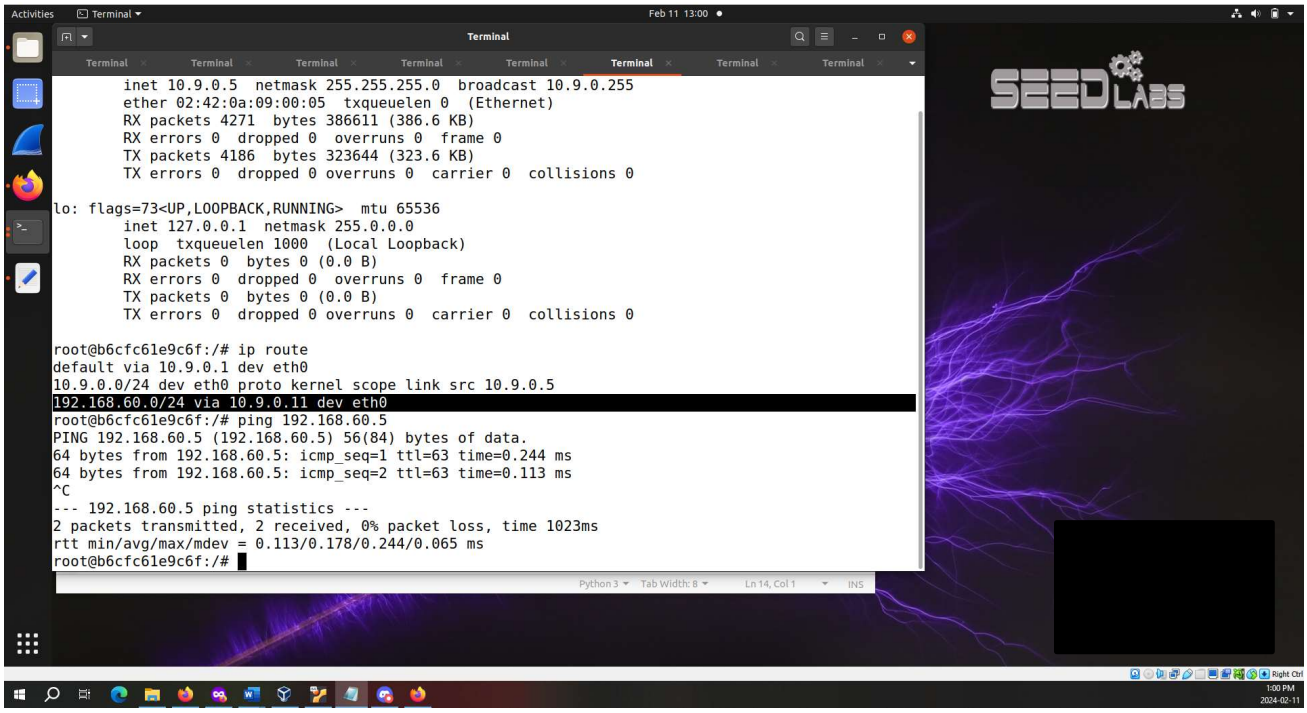

Activities
Terminal
Feb 11 10:44
Terminal
Abdulfatah Abdillahi-Sun Feb 11-10:39:05 ~/.../Labsetup> docksh victim-10.9.0.5
root@b6cfc61e9c6f:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
RX packets 57 bytes 8587 (8.5 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
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
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
root@b6cfc61e9c6f:/#
SEED LABS
Right Ctrl
Type here to search
ENG
10:44 AM
2024-02-11

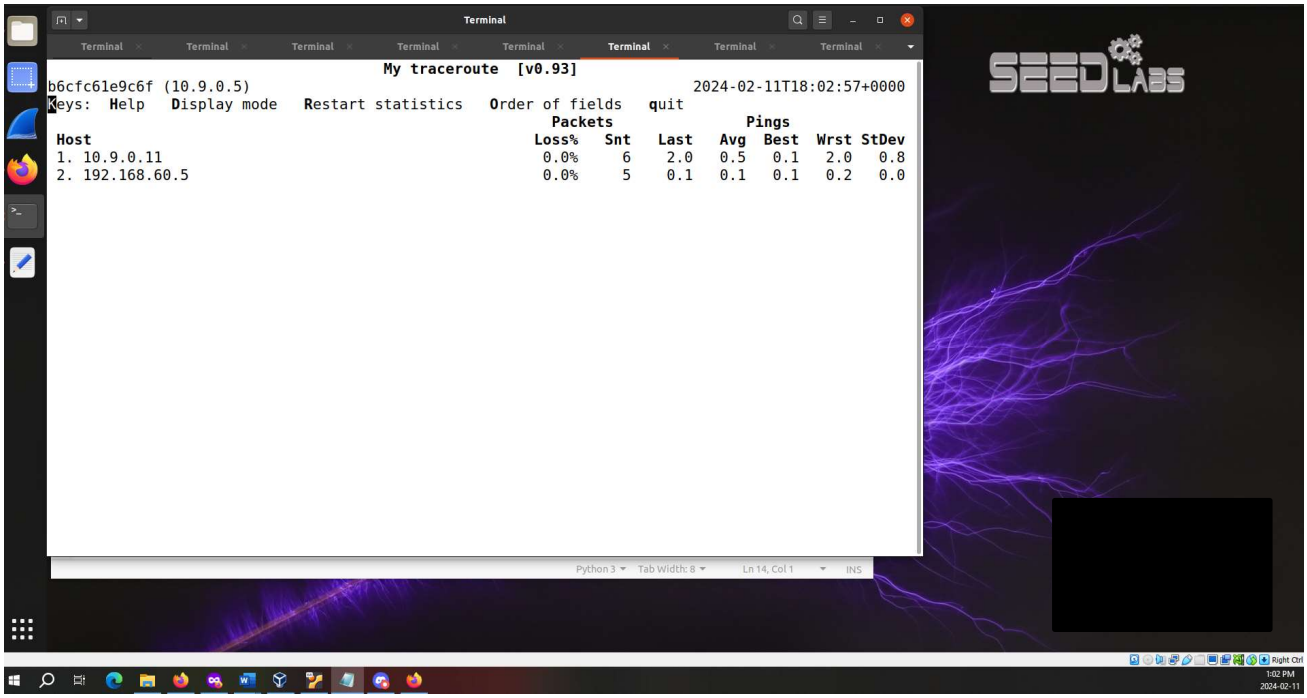




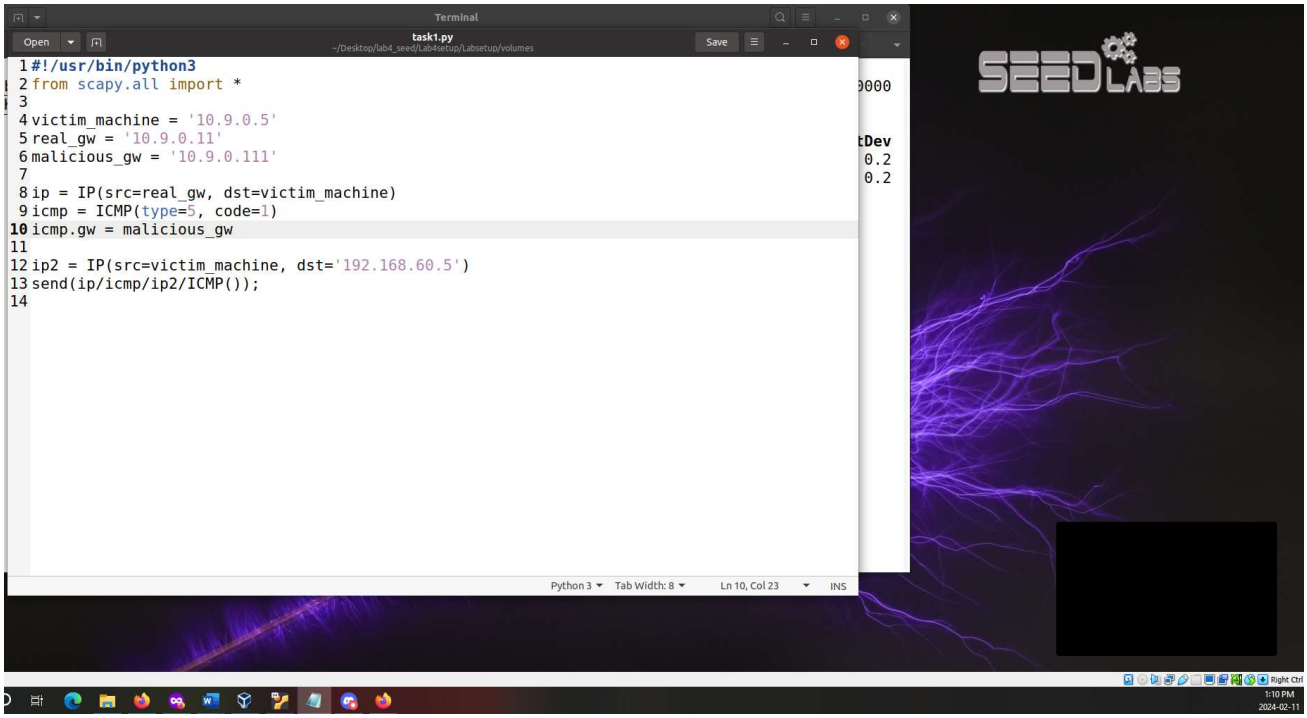
Lab Task 1: Launching ICMP Redirect Attack

Here, before attempting anything I would like to show the default routing of the victim machine. I ran the command ip route. This screenshot shows that the default gateway of the victim is currently set to the real router (192.168.60.11). I have further proven this by checking the traceroute of a packet after pinging an external host (192.168.60.5).

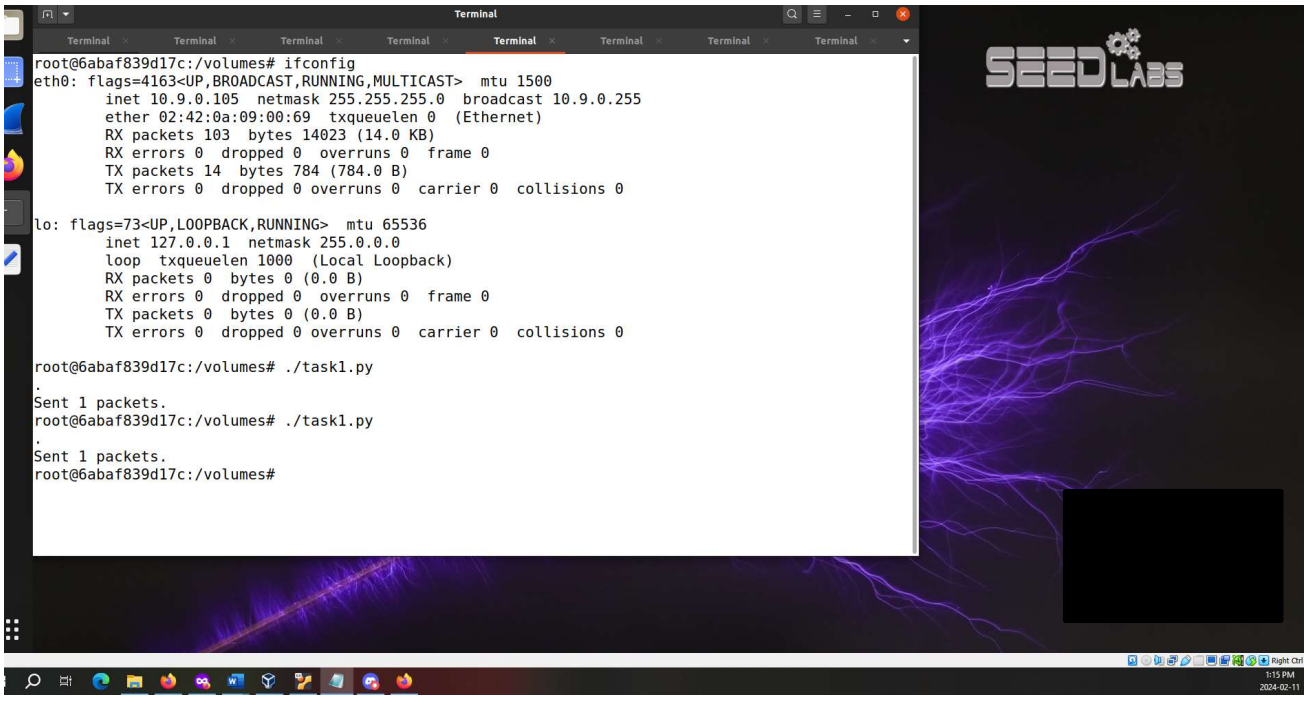


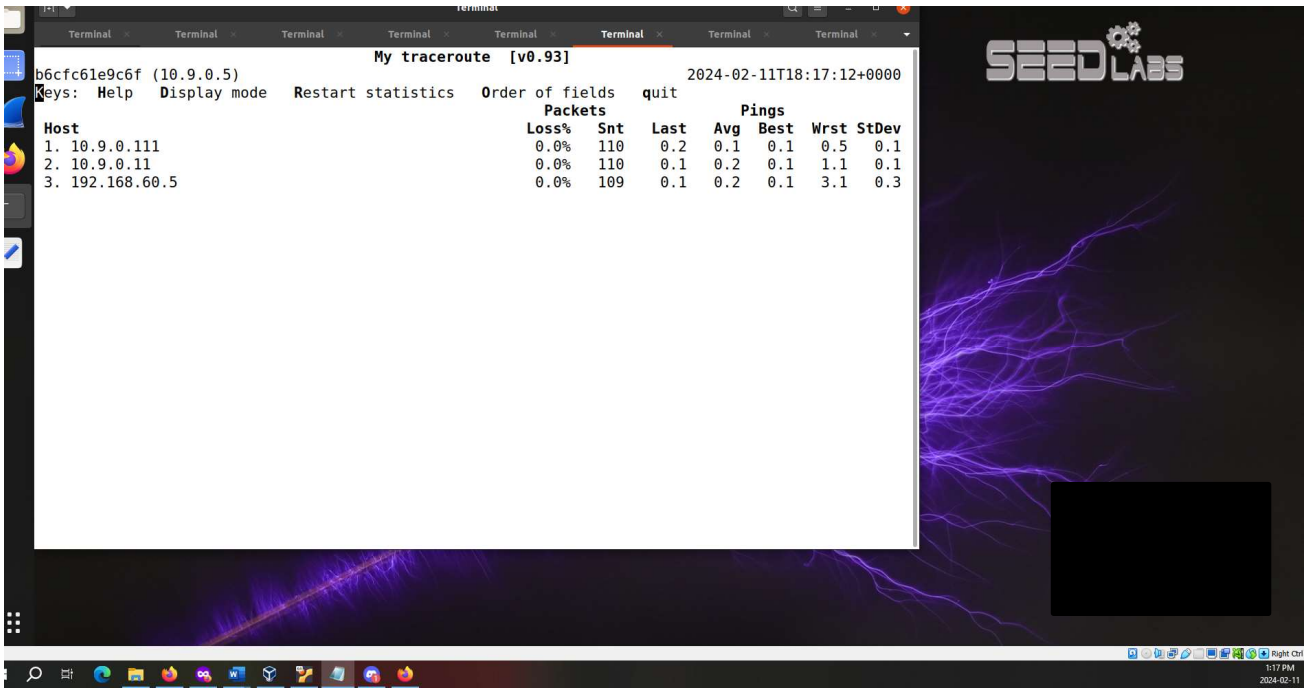


Here, I'm showing the modified code skeleton which I will be using to attack the victim container from the attacker container.



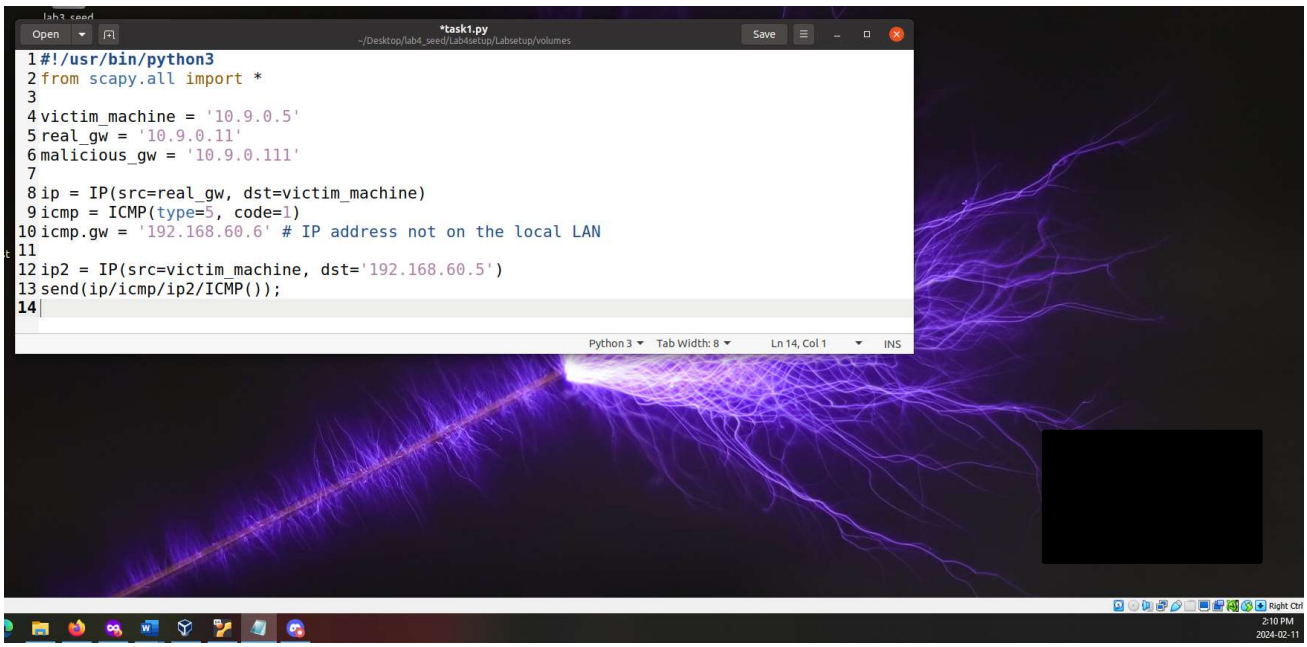
Then I ran the program on the attacker's machine and checked the traceroute on the victims' machine. As you can see this attack was successful.



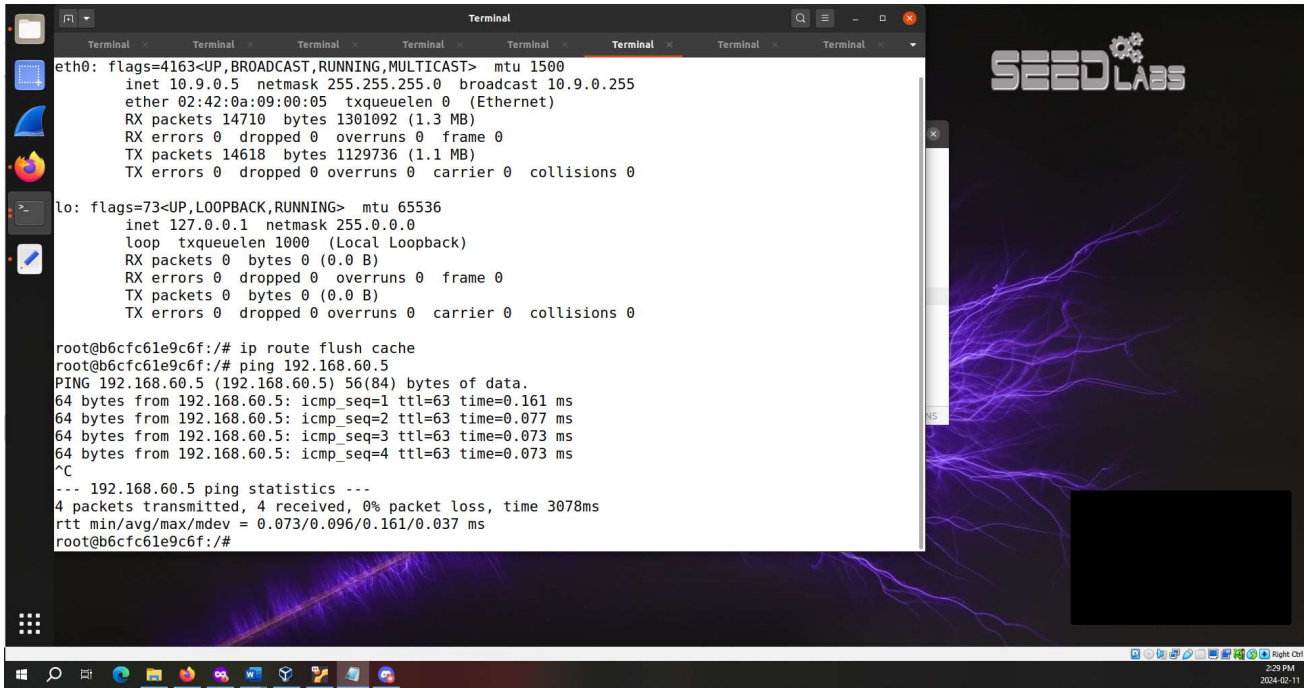


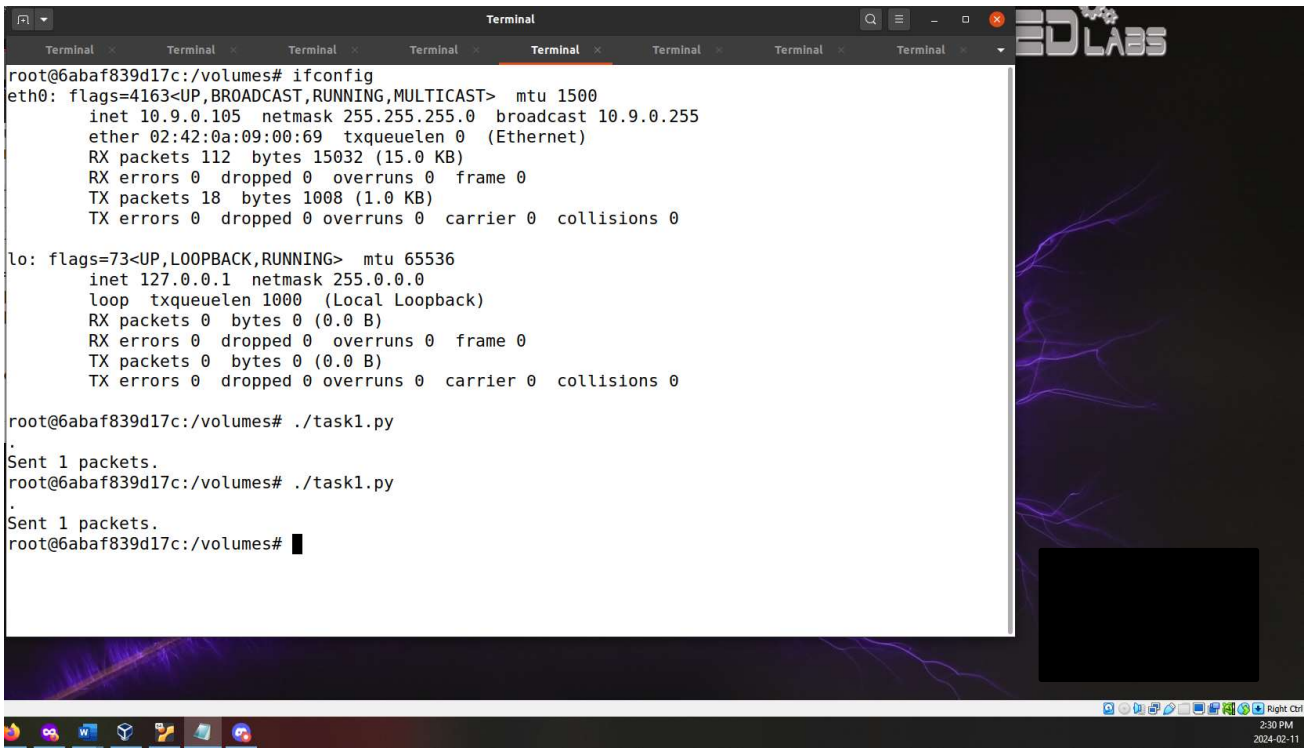
Question1:

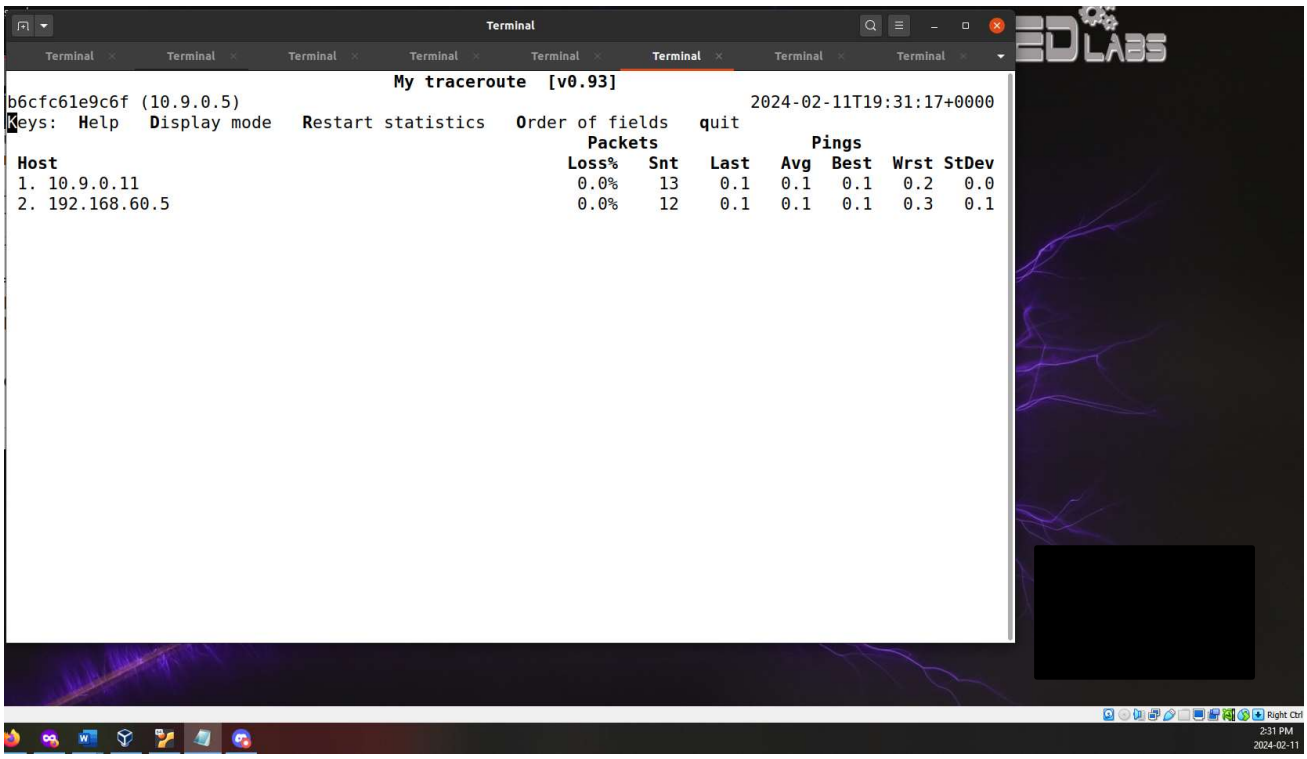
Here, I modified the code skeleton once again but this time it is such that the IP address assigned to *icmp.gw* variable is a computer not on the local LAN.



I then flushed the cache to ensure nothing from the previous activity interferes with the experiment. Then I ran the script on the attacker machine and went to check the traceroute on the victim machine. As you will see this did not work. This is because the victim would not accept a redirect message that suggests a remote gateway for a local destination.

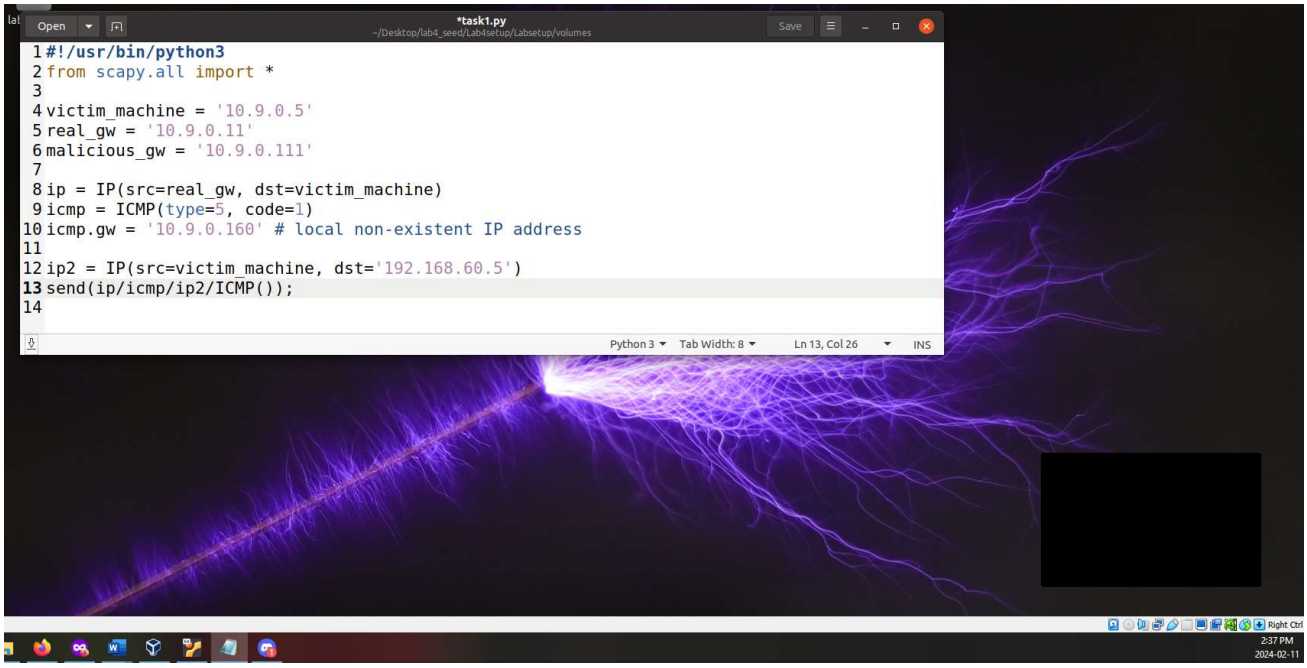

Terminal
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
RX packets 14710 bytes 1301092 (1.3 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 14618 bytes 1129736 (1.1 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
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
root@b6cfc61e9c6f:/# ip route flush cache
root@b6cfc61e9c6f:/# 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=0.161 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.073 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.073 ms
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3078ms
rtt min/avg/max/mdev = 0.073/0.096/0.161/0.037 ms
root@b6cfc61e9c6f:/#
SEED LABS


Terminal
root@6abaf839d17c:/volumes# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.9.0.105 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:69 txqueuelen 0 (Ethernet)
RX packets 112 bytes 15032 (15.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 18 bytes 1008 (1.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
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
root@6abaf839d17c:/volumes# ./task1.py
.
Sent 1 packets.
root@6abaf839d17c:/volumes# ./task1.py
.
Sent 1 packets.
root@6abaf839d17c:/volumes#

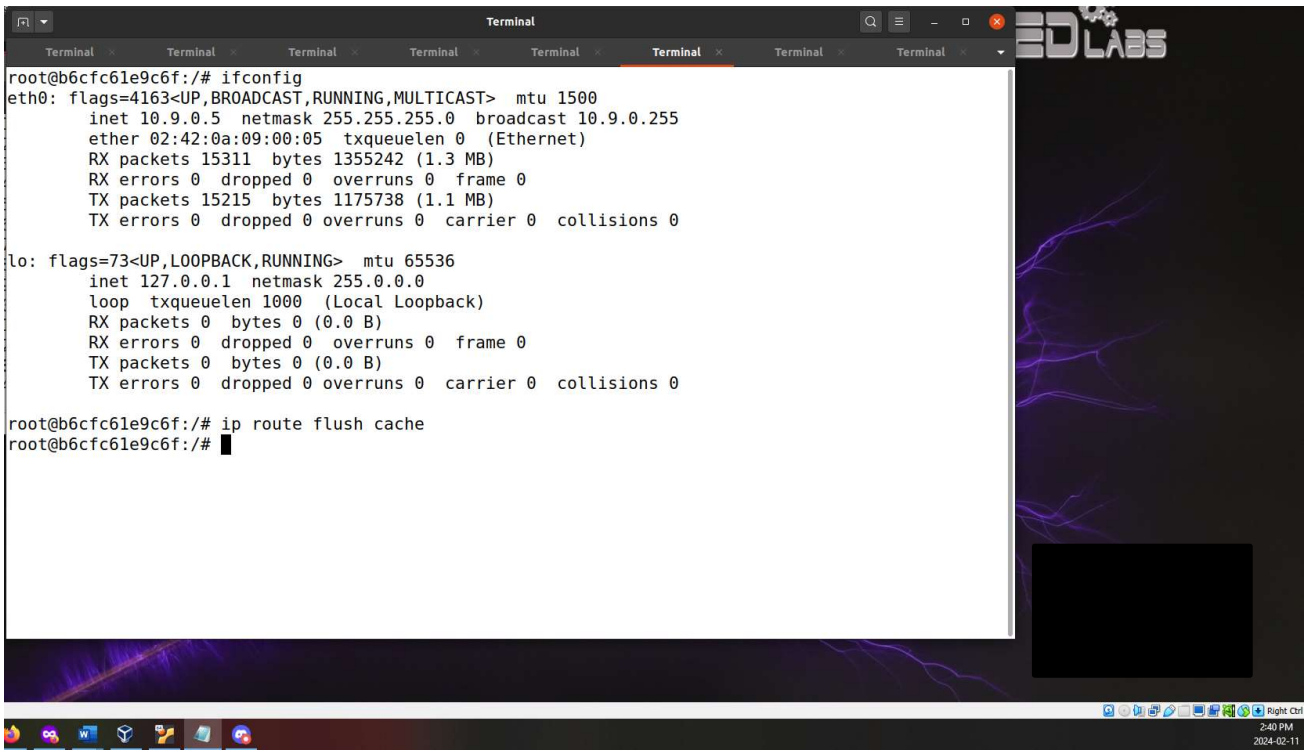


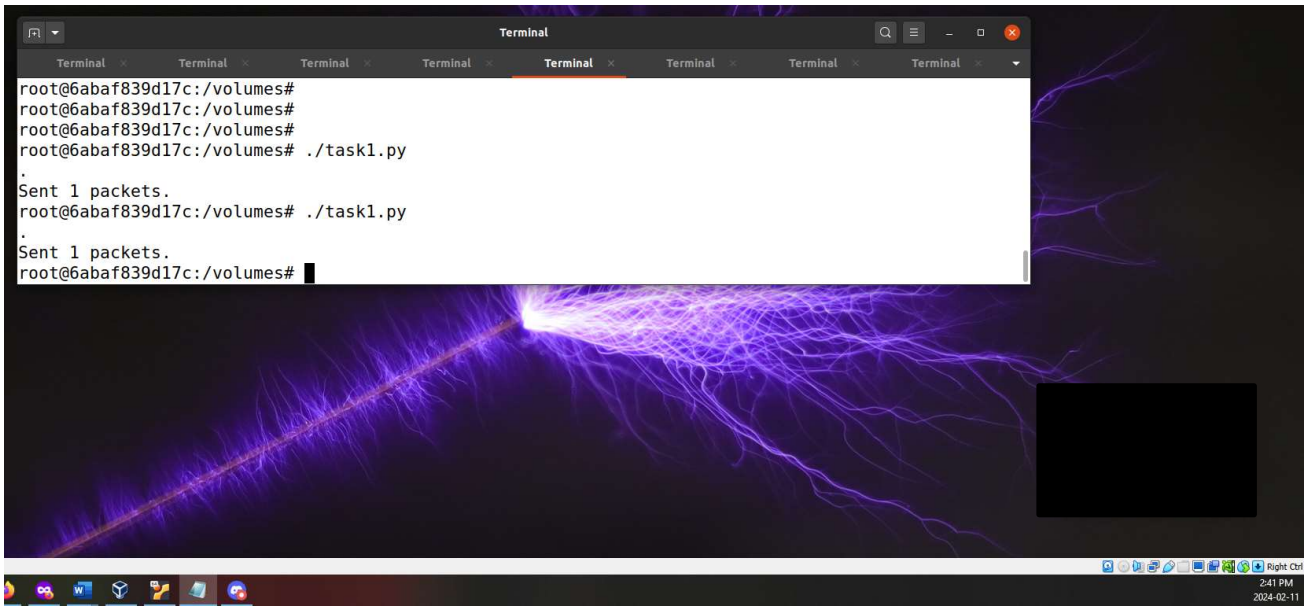
Question 2:

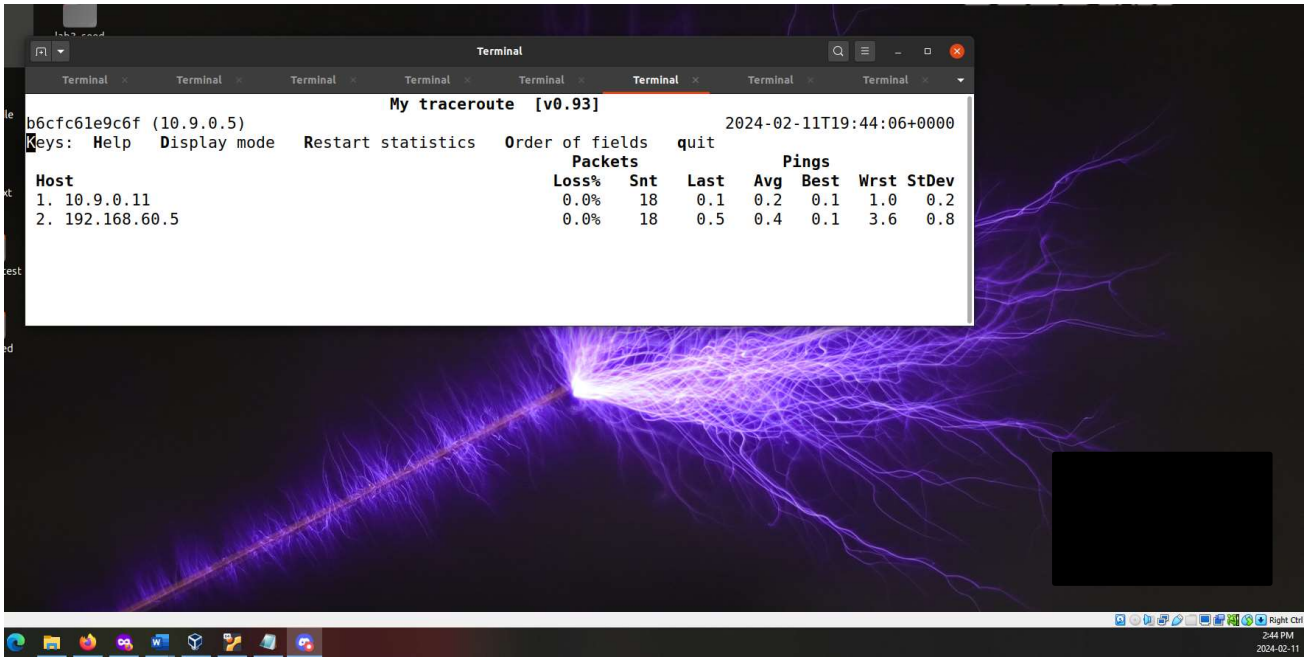
Here, this experiment is similar to the previous one but this time we are modifying the code skeleton such that the IP address assigned to *icmp.gw* variable is set to a non-existing machine on the same network.



Then I flushed the cache to ensure nothing from the previous activity interferes with the experiment. Next, I ran the script on the attacker machine and went to check the traceroute on the victim machine. As you can see this did not work because the victim would not be able to reach the suggested gateway.

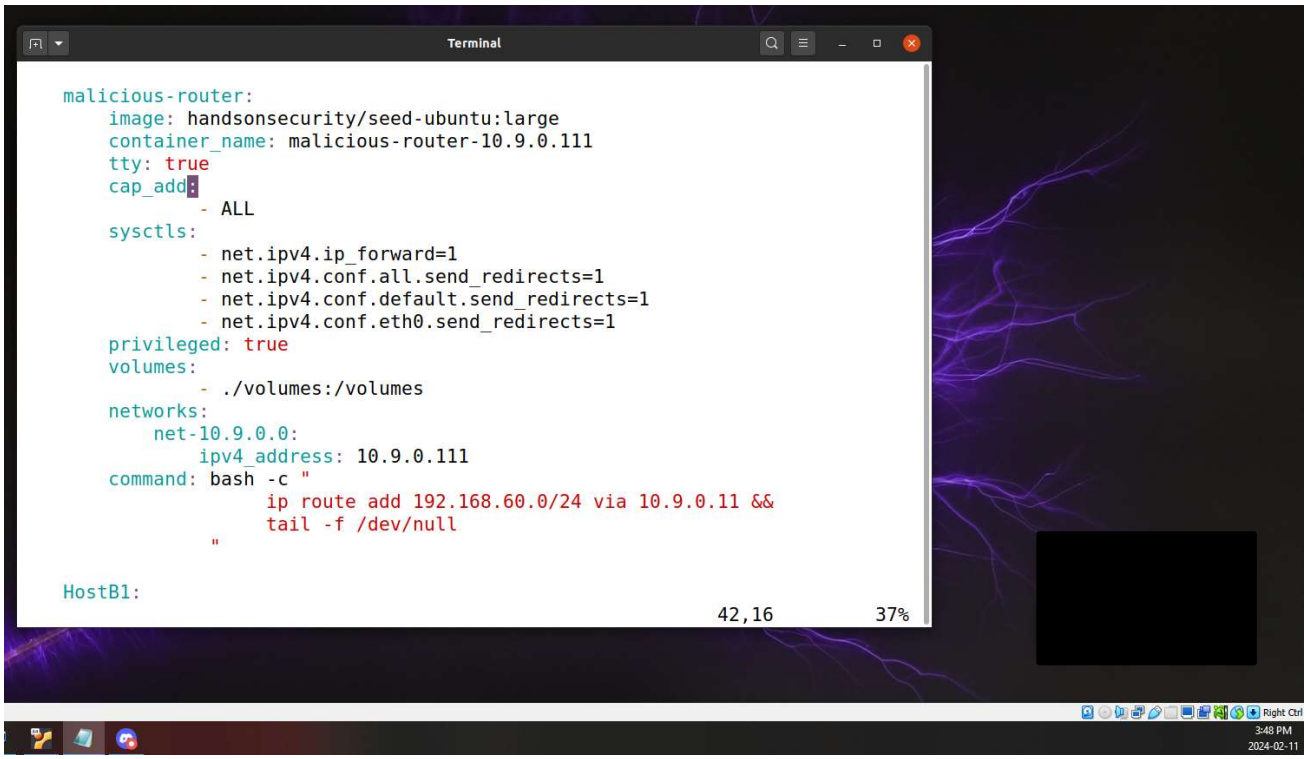

Terminal
root@b6cfc61e9c6f:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
RX packets 15311 bytes 1355242 (1.3 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 15215 bytes 1175738 (1.1 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
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
root@b6cfc61e9c6f:/# ip route flush cache
root@b6cfc61e9c6f:/#
Right Ctrl
2:40 PM
2024-02-11

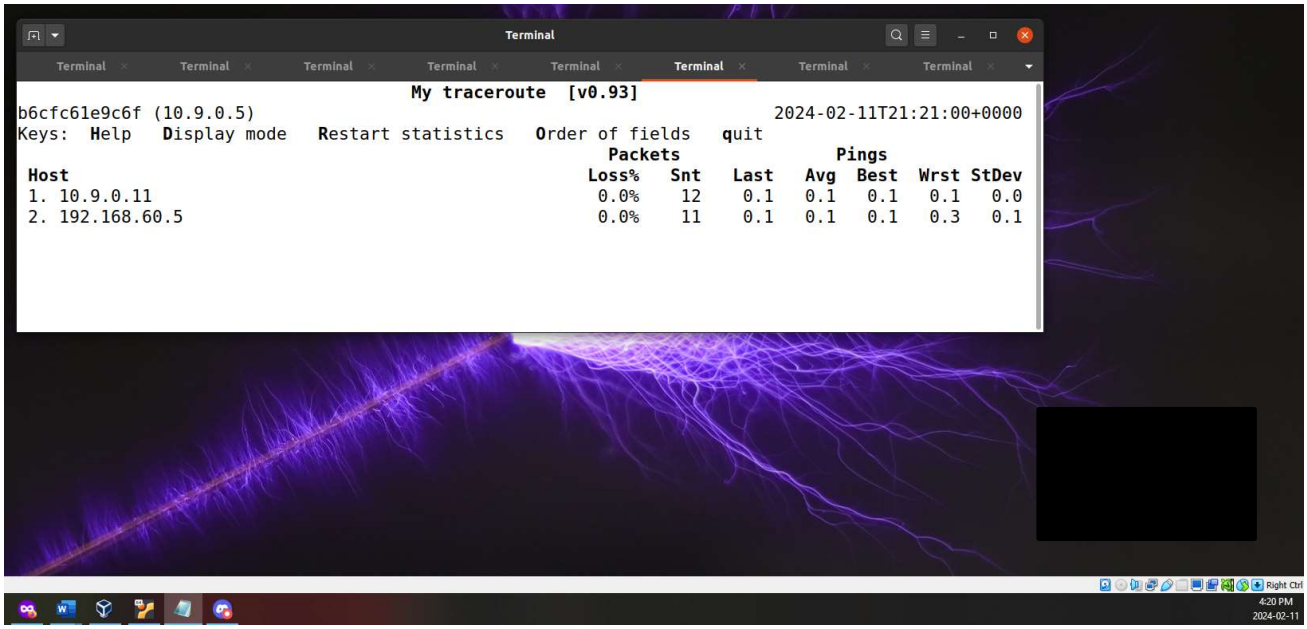

Terminal
root@6abaf839d17c:/volumes#
root@6abaf839d17c:/volumes#
root@6abaf839d17c:/volumes#
root@6abaf839d17c:/volumes# ./task1.py
.
Sent 1 packets.
root@6abaf839d17c:/volumes# ./task1.py
.
Sent 1 packets.
root@6abaf839d17c:/volumes#
Right Ctrl
2:41 PM
2024-02-11



Question3:

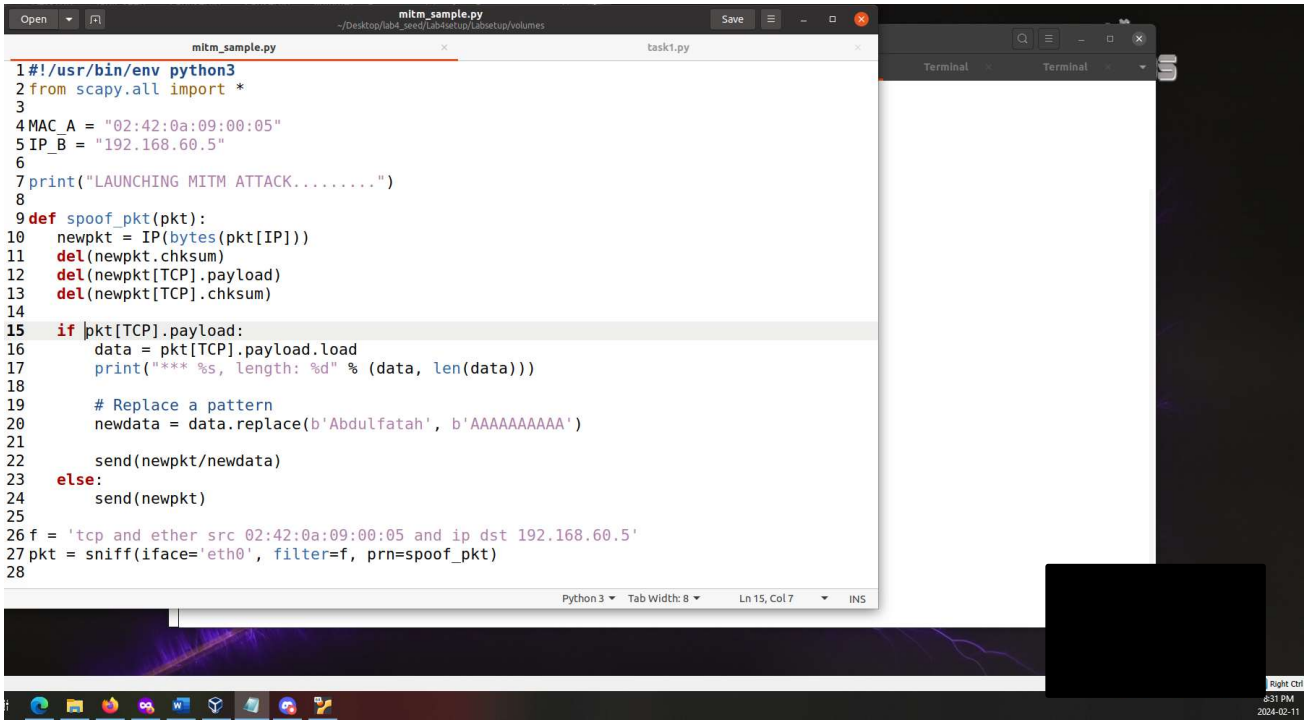
Here, I modified the *sysctl* section of the *docker-compose.yml* file to ensure the malicious router could send ICMP redirect messages. There are 4 lines under that section. The first line enables IP forwarding. The second line allows the system to send ICMP redirect messages. The third line allows the system to accept ICMP redirect messages by default. And the last line configures the interface *eth0* to send ICMP redirect messages. After making those changes, I launched the attack once again and ran the code. This did not work as you can see from the traceroute output.



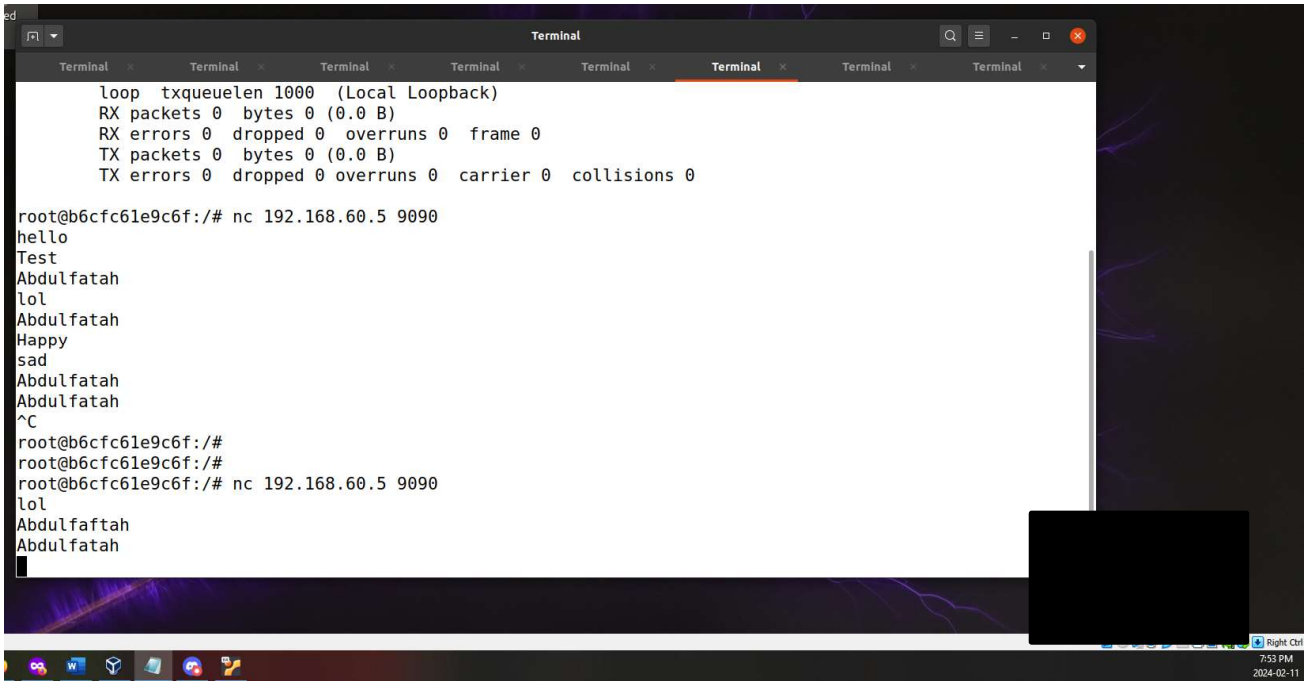


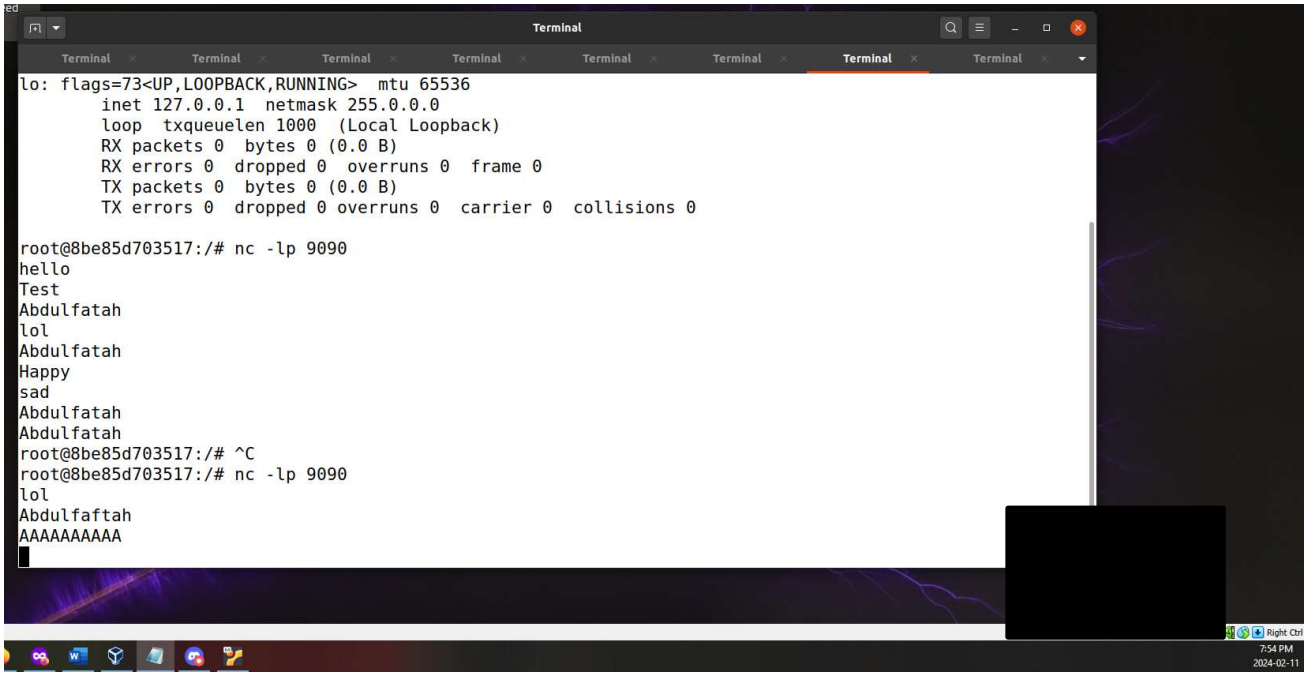
Task 2: Launching the MITM Attack

Here, I'm running the modified version of the code that was provided in the manual.



Here, I created a netcat connection between the victim and one of the host machines. I then disabled the IP forwarding in the malicious-router so that it can intercept and alter the communication. I then ran the above code on the malicious router and tested the netcat communication. As you can see this worked in the screenshots below.





Question 4:

In our case of the MITM program we had to capture the traffic in the direction of victim → host and not the other way around. This is because we have a malicious router embedded within the victim's network as opposed to the hosts network (which is a different network) thus allowing us to have control over the victim's packets which are going through the malicious router first (where the packets are modified before being sent). On the other hand, the packets of the host are not going through a different (malicious) router when communicating back to the victim.

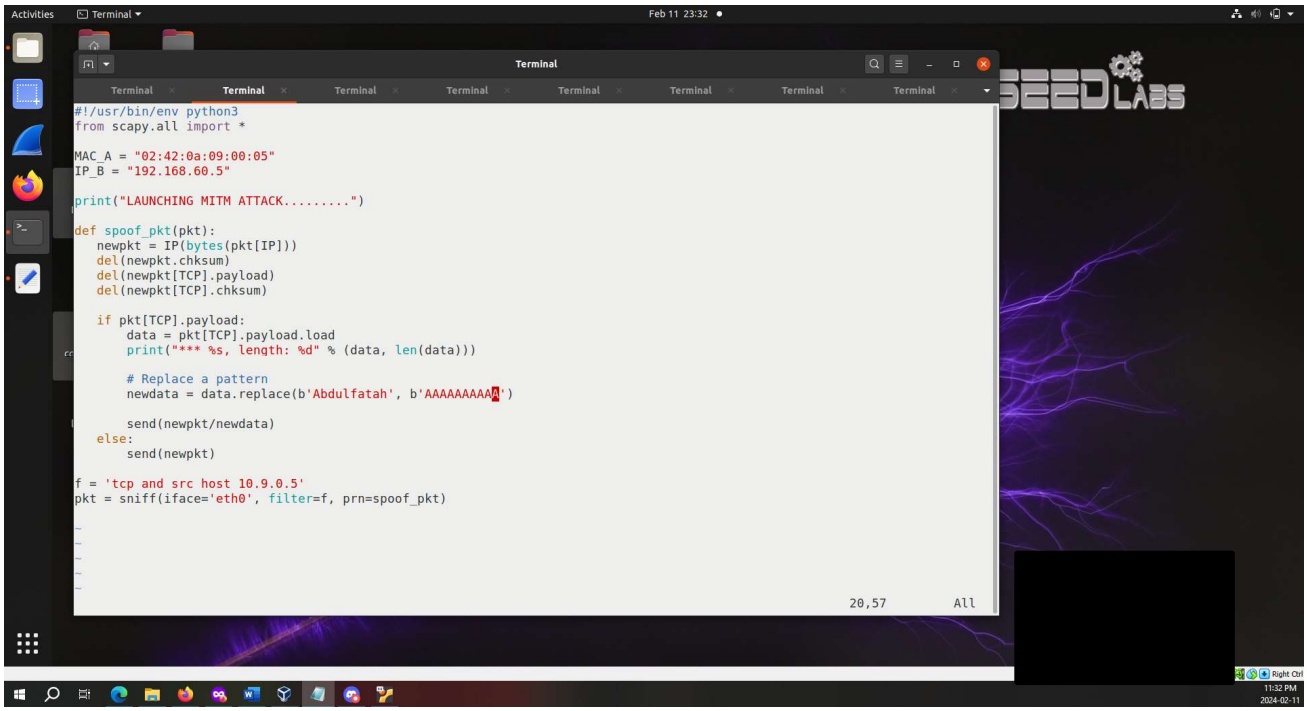
Question 5:

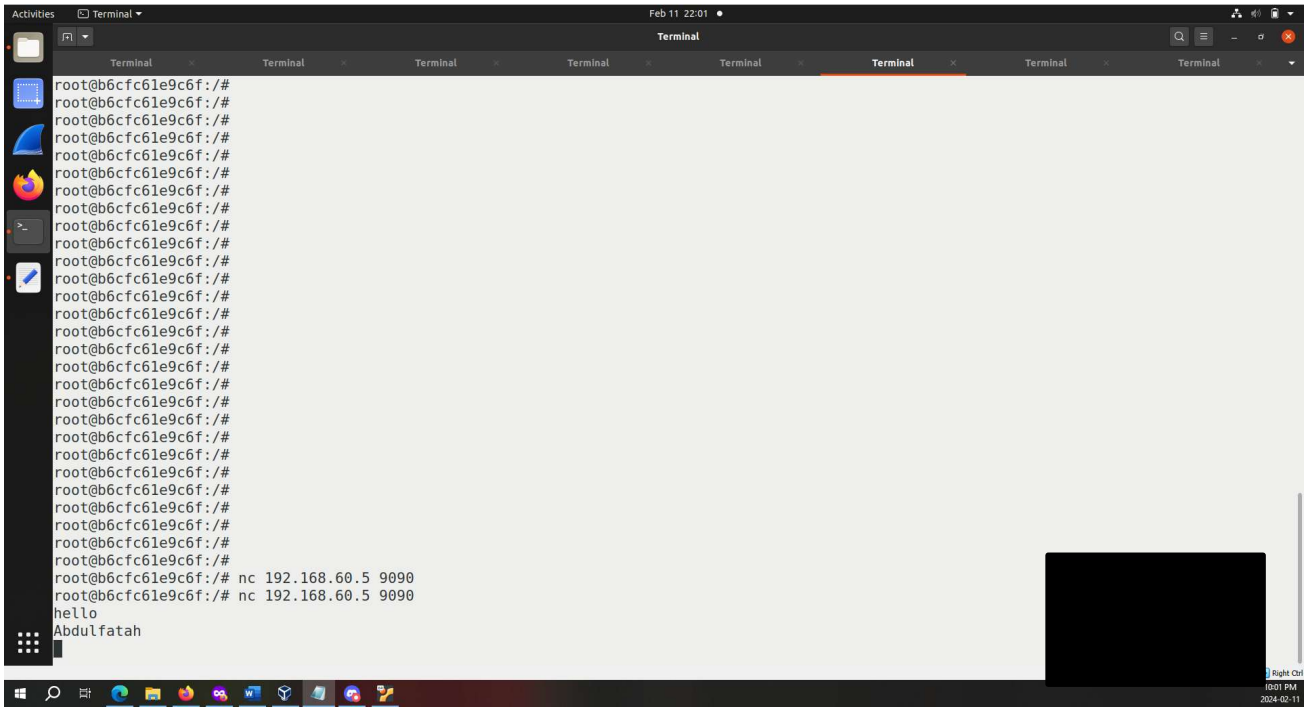
As you can see in the below screenshots both methods work but the first method results in more "TCP retransmission" error (Black colour) as you see in Wireshark. When compared to the second method there is less of that error. Also, when you compare the output of the program in both

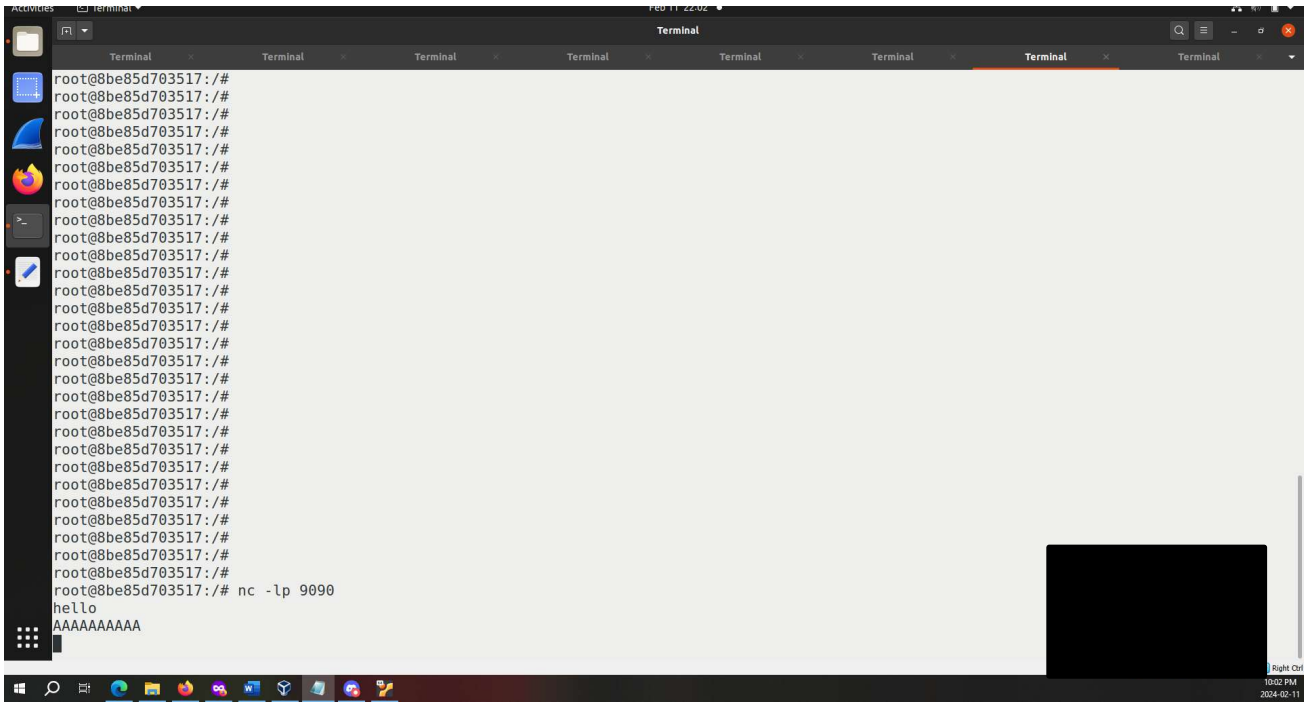
methods (second last screenshots) you will see that they both reflect the same issue as the first method has a lot of retransmission messages in the terminal compared to the second method. Therefore, I conclude that the second method is the correct.

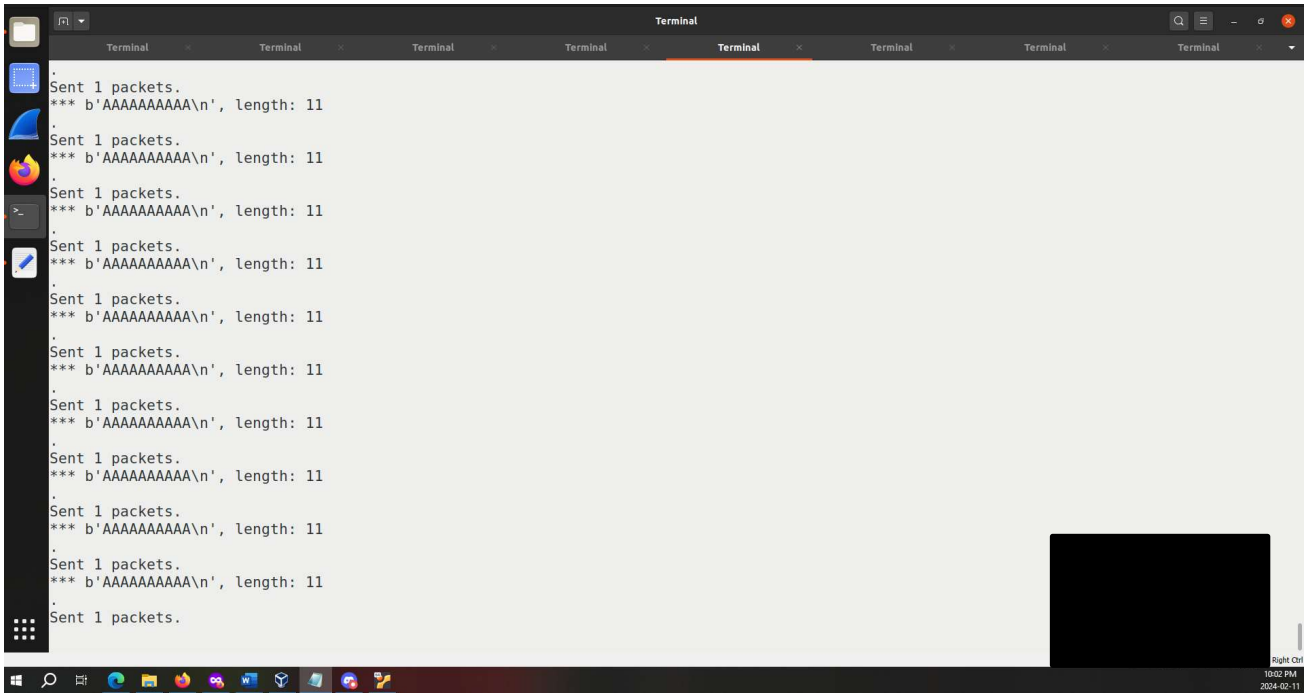
Method 1

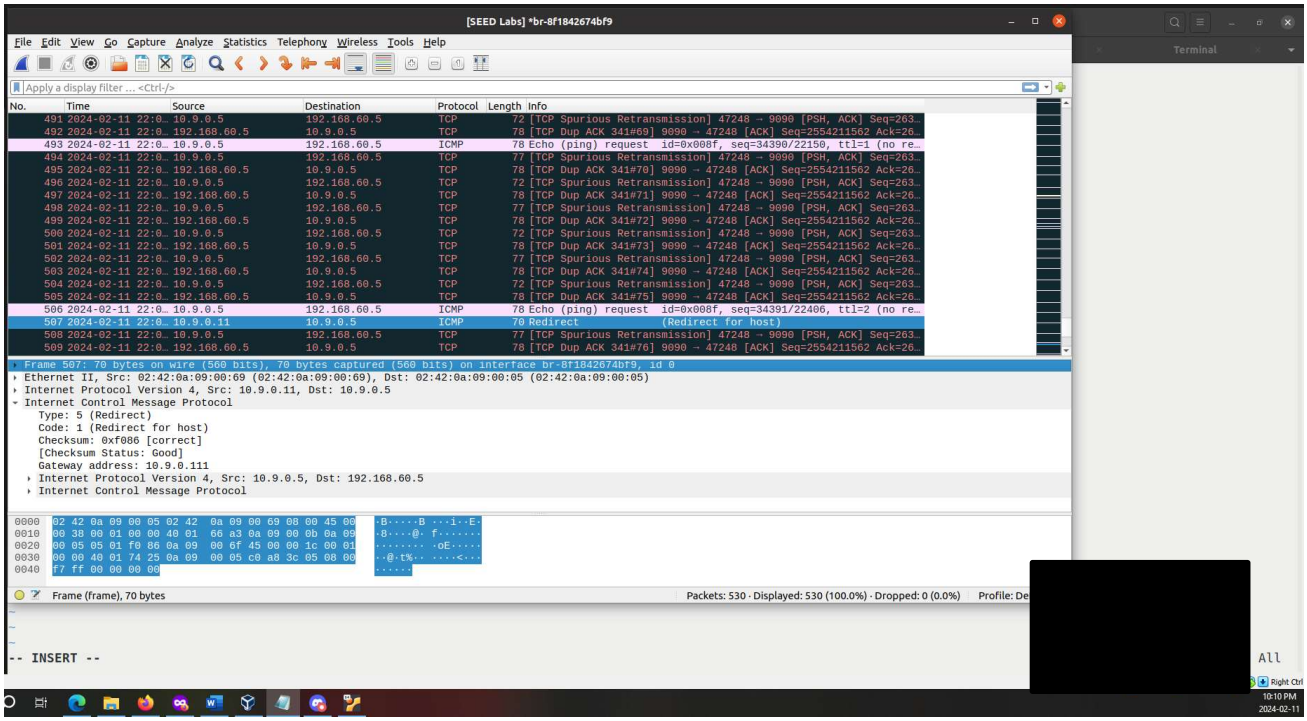
In this method I'm using A's IP address in the filter part of the code and trying the same attack again.





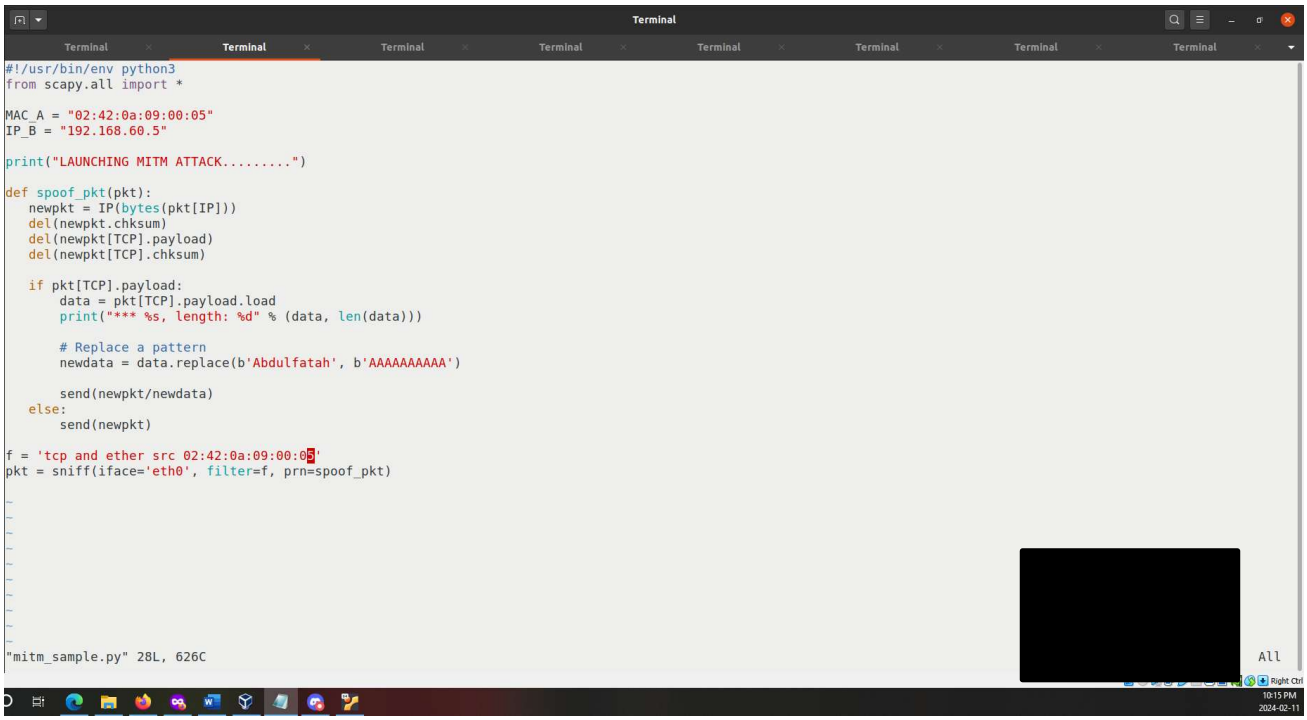

Terminal
root@8be85d703517:/#
root@8be85d703517:/# nc -lp 9090
hello
AAAAAAAAAA

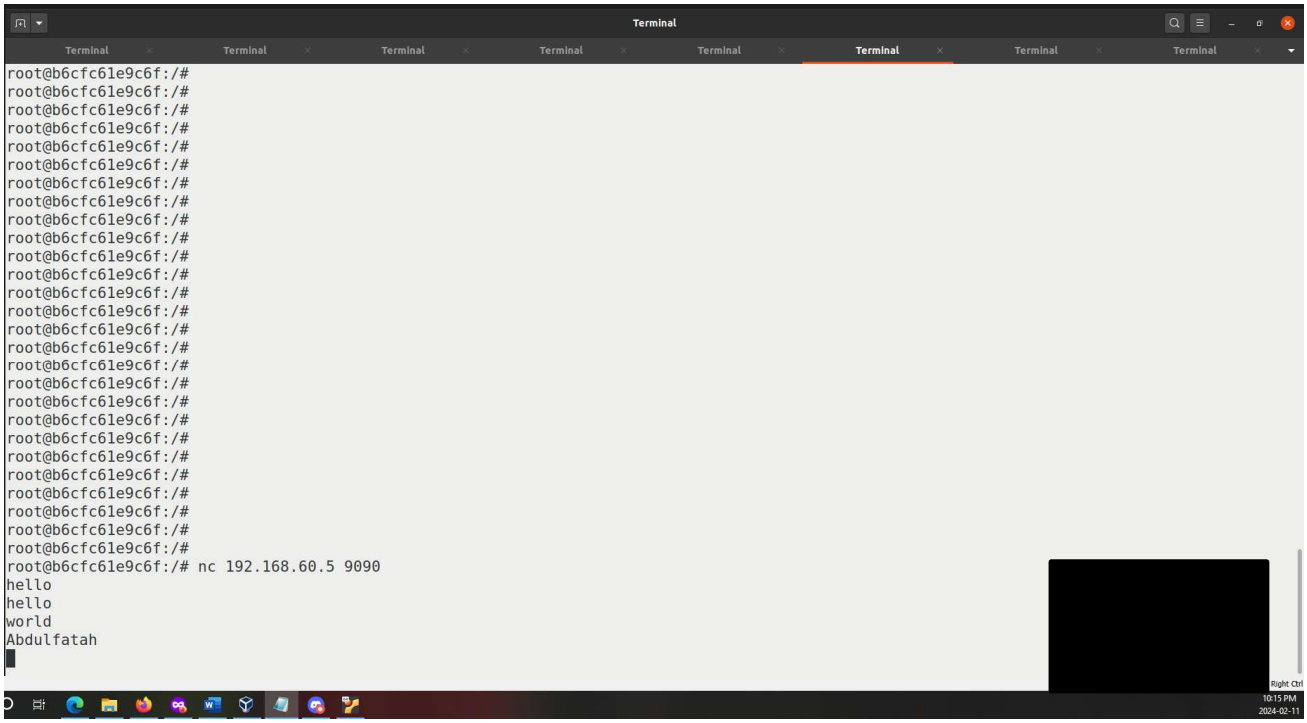

Terminal
Sent 1 packets.
*** b'AAAAAAAAAA\n', length: 11
Sent 1 packets.

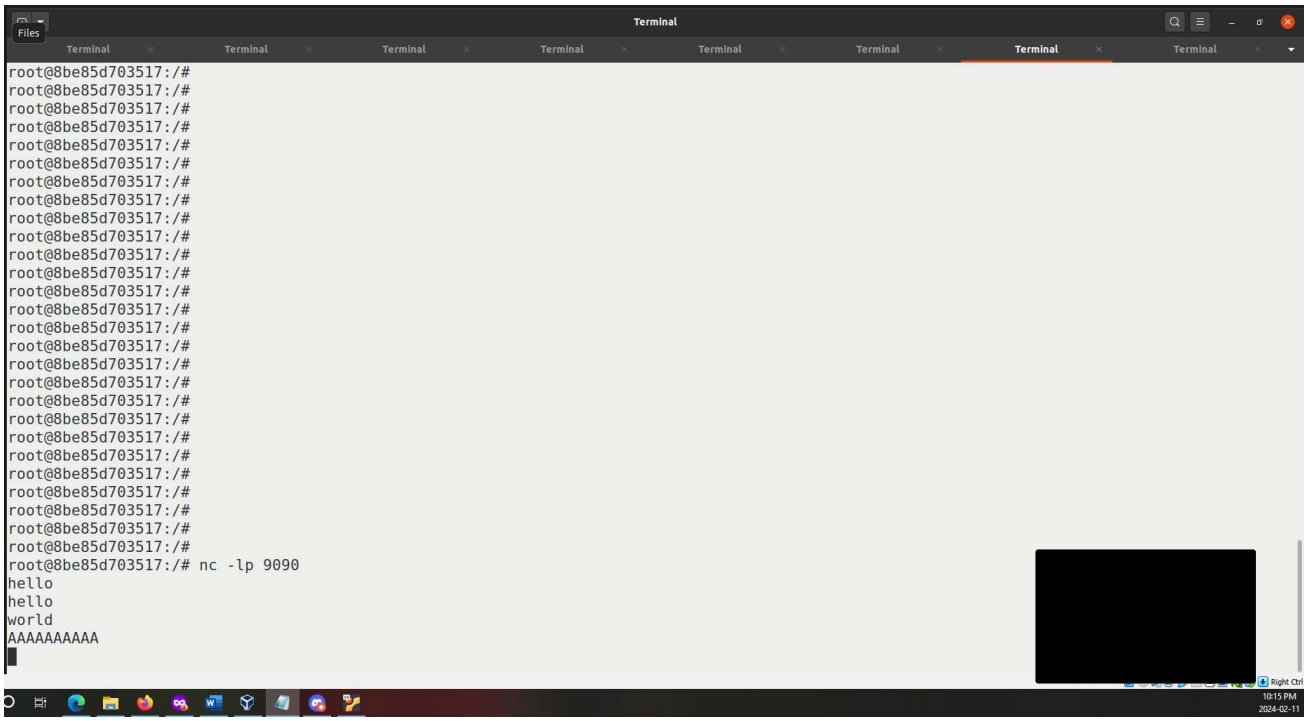


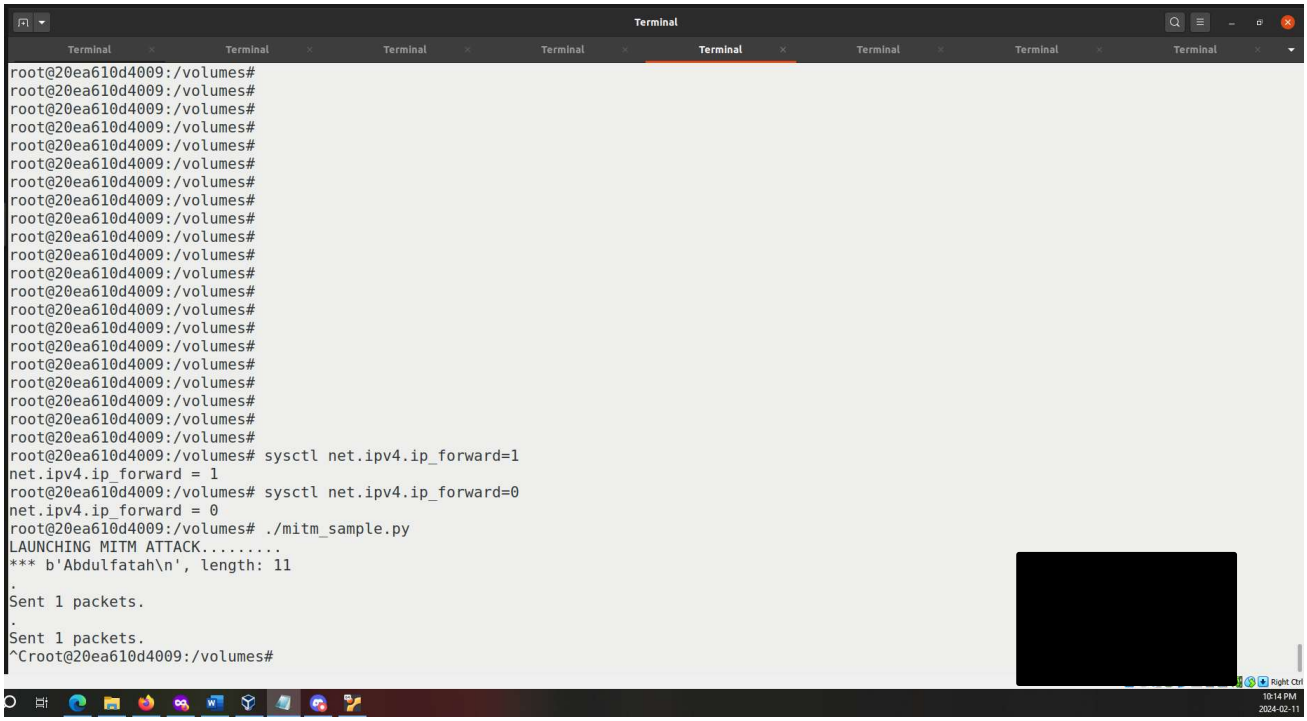
Method 2

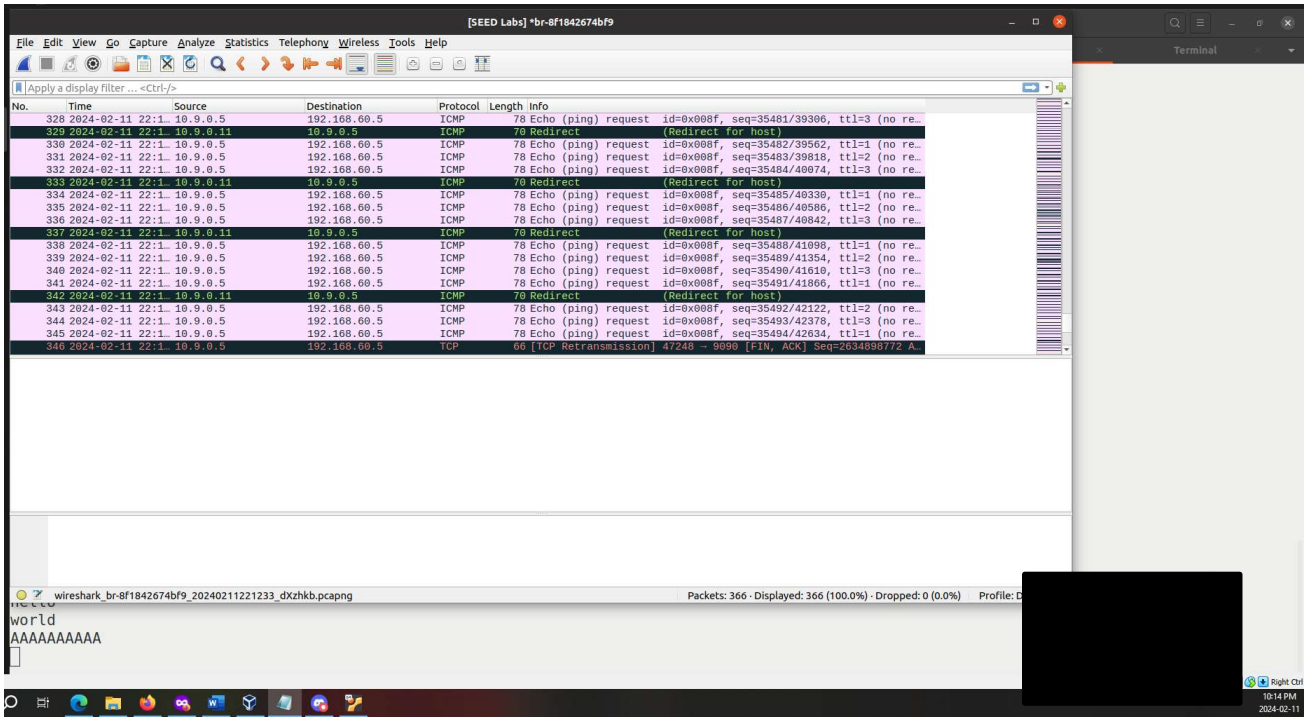
In this method I'm using A's MAC address in the filter part of the code and trying to launch the same attack again.




Terminal
root@b6cfc61e9c6f:/#
root@b6cfc61e9c6f:/# nc 192.168.60.5 9090
hello
hello
world
Abdulfatah


Terminal
Files
root@8be85d703517:/#
root@8be85d703517:/# nc -lp 9090
hello
hello
world
AAAAAAAAAA

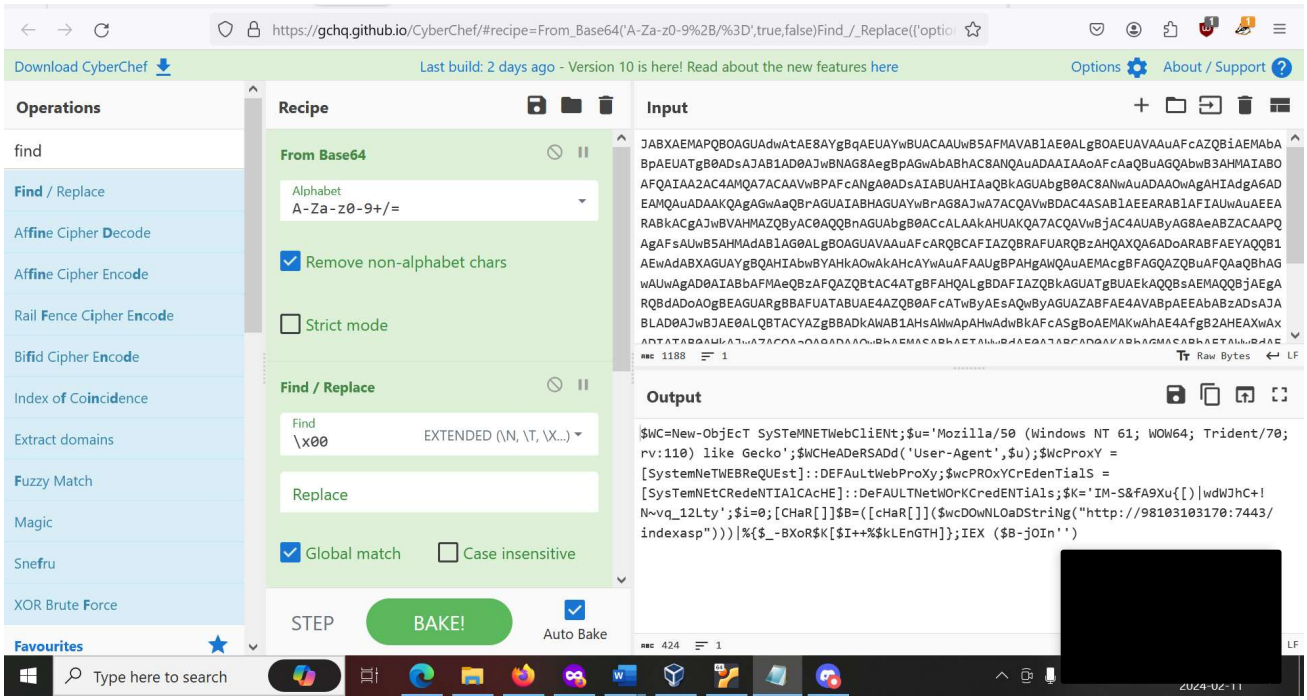




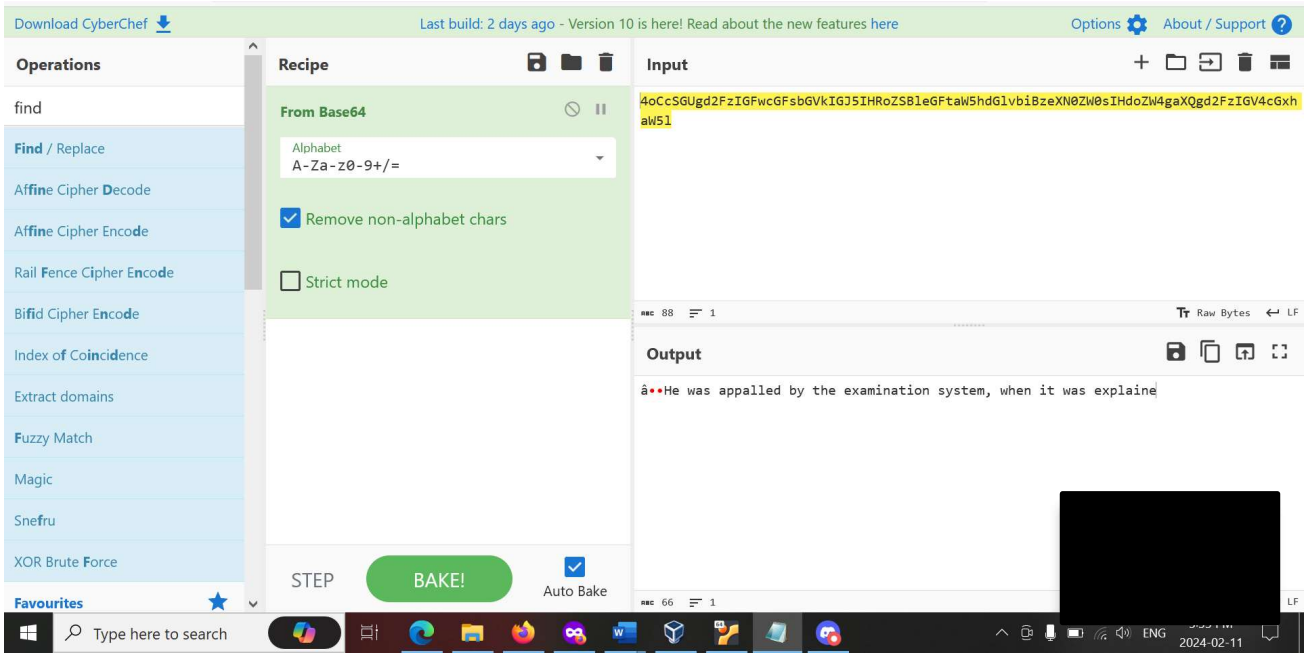
Task 3: Cyberchef Exercises

Excercise1:

Here, I followed the article and attempted to decode the provided Base64 encoded text. I first utilized the *From Base64* operation, but this resulted in having some "NULL" in the output. To fix this I utilized a *Find and* Replace Operation and inputted "\x00" in the Find field which is the python code for Null. This resulted in the correct output as described in the article.

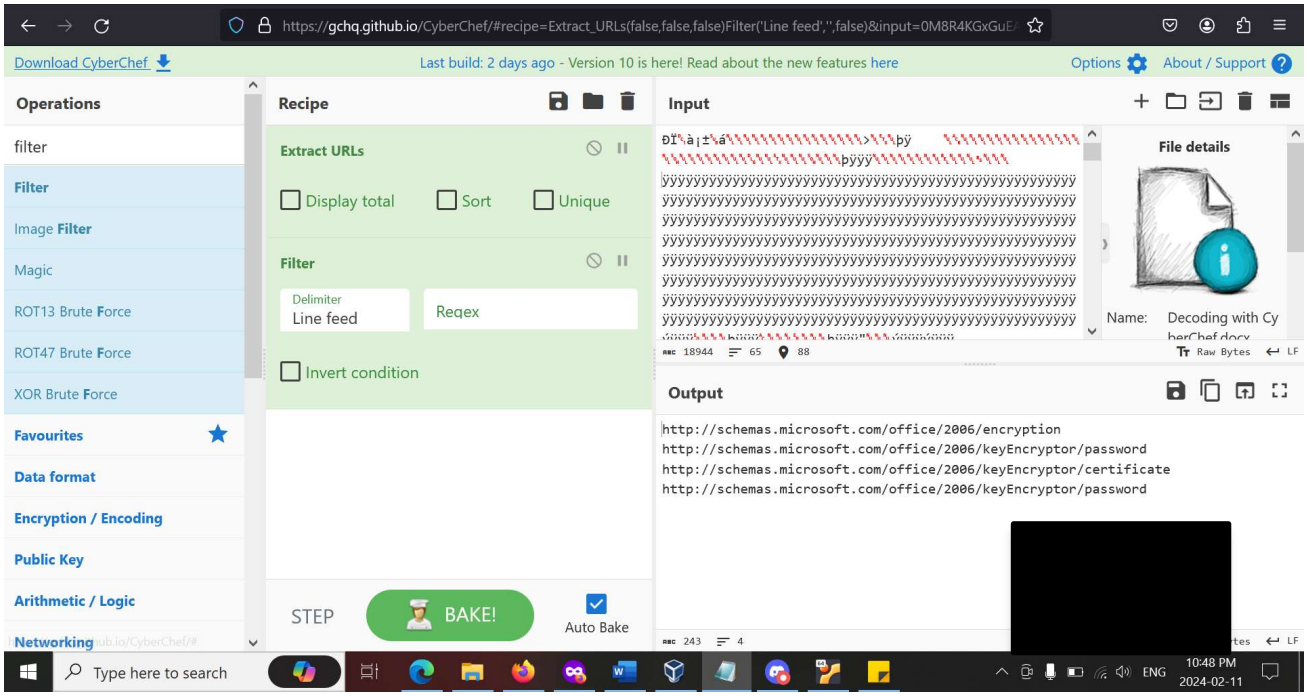


Here, I'm attempting to decode the Base64 encoded text provided in the Blackboard instructions. Similar to the code provided by the article, I applied the the *From Base64* operation which resulted in the message: "He was appalled by the examination system, when it was explaine".



Excercise2:

Here, I uploaded the password-protected file from Blackboard to Cyberchef and extracted the URLs from the file using the *Extract URLs* operation. They do more stuff in the article such as using *Defang URL* operation to help prevent accidentally clicking on any malicious URLs. I couldn't apply those operation in my case as the malicious link did not show up in my Output section.


https://gchq.github.io/CyberChef/#recipe=Extract_URLs(false,false,false)Filter('Line feed','',false)&input=0M8R4KGxGuE
Download CyberChef
Last build: 2 days ago - Version 10 is here! Read about the new features here
Options
About / Support
Operations
filter
Filter
Image Filter
Magic
ROT13 Brute Force
ROT47 Brute Force
XOR Brute Force
Favourites
Data format
Encryption / Encoding
Public Key
Arithmetic / Logic
Networking
Recipe
Extract URLs
Display total
Sort
Unique
Filter
Delimiter
Line feed
Regex
Invert condition
STEP
BAKE!
Auto Bake
Input
File details
Name:
Decoding with CyberChef.docx
Raw Bytes
LF
Output
http://schemas.microsoft.com/office/2006/encryption
http://schemas.microsoft.com/office/2006/keyEncryptor/password
http://schemas.microsoft.com/office/2006/keyEncryptor/certificate
http://schemas.microsoft.com/office/2006/keyEncryptor/password
Type here to search
ENG
10:48 PM
2024-02-11

References

Stamm, E. (2021, May 31). CyberChef – Data Decoding Made Easy. *CSNP.* Retrieved from https://www.csnp.org/post/cyberchef-data-decoding-made-easy

Cybergibbons. (2020, September 19). Understanding binary and data representation with CyberChef. Retrieved from https://cybergibbons.com/reverse-engineering-2/understanding-binary-and-data-representation-with-cyberchef/

Du, W. (n.d.). ICMP redirect attack lab. *seedsecuritylabs.* Retrieved from https://seedsecuritylabs.org/Labs_20.04/Files/ICMP_Redirect/ICMP_Redirect.pdf