

CSE 406

Computer Security Sessional

Final Report on Project No. 1

ARP Cache Poisoning with Man in the Middle Attack

Submitted by-

Raihan Rasheed

Student ID. 1605062

Lab Group. B1

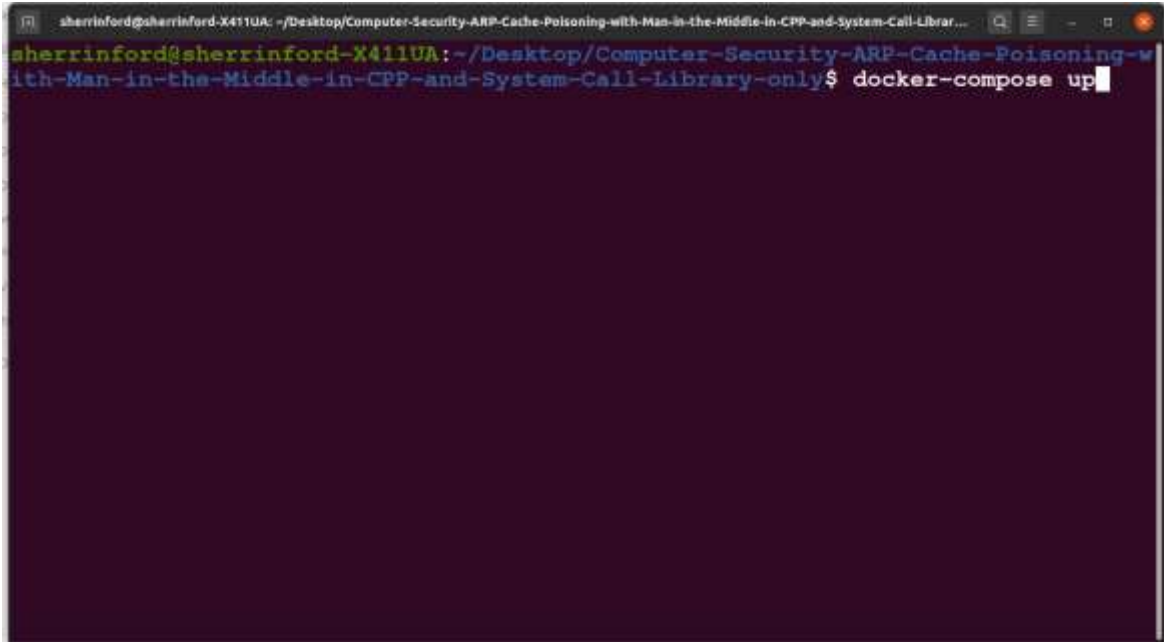
Project Group No. 2

Level 4 Term 1

Department of CSE, BUET

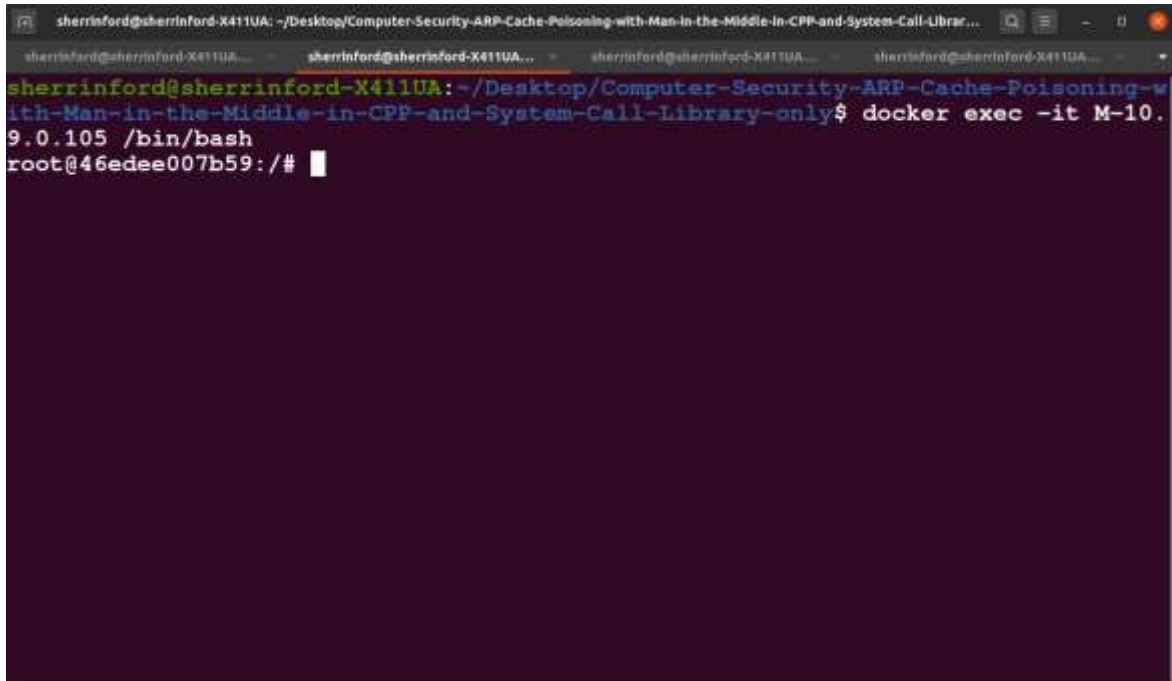
Attack Steps

1. Go to project directory and open a terminal from linux and type following command in the picture.

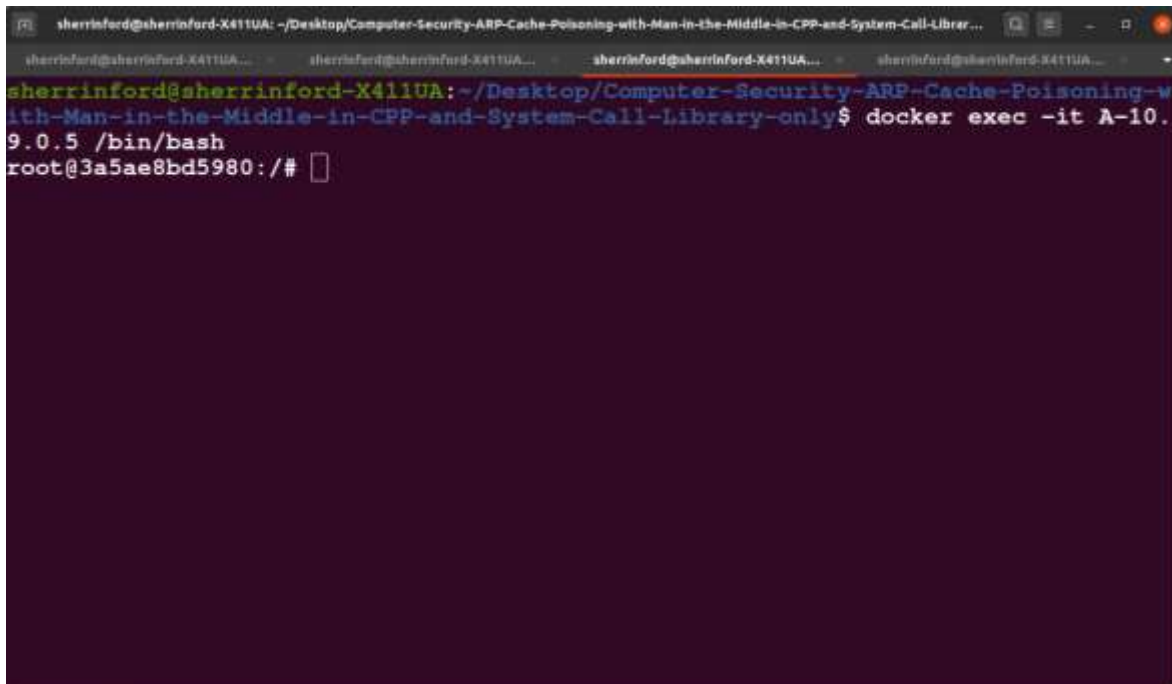
A screenshot of a Linux terminal window. The window title is "sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only". The prompt is "sherrinford@sherrinford-X411UA:~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only\$". The command "docker-compose up" is being typed at the end of the line.

```
sherrinford@sherrinford-X411UA:~/Desktop/Computer-Security-ARP-Cache-Poisoning-w  
ith-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker-compose up
```

2. Open three other tabs to spawn attacker, host-A, host-B containers respectively. Same process will be applicable if we want additional terminals for respective containers.



```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Librar...
sherrinford@sherrinford-X411UA:~/Desktop/Computer-Security-ARP-Cache-Poisoning-w
ith-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it M-10.
9.0.105 /bin/bash
root@46edee007b59:/#
```



```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Librar...
sherrinford@sherrinford-X411UA:~/Desktop/Computer-Security-ARP-Cache-Poisoning-w
ith-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it A-10.
9.0.5 /bin/bash
root@3a5ae8bd5980:/#
```

```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it B-10.9.0.6 /bin/bash
root@de7f1e3ba5a6:/#
```

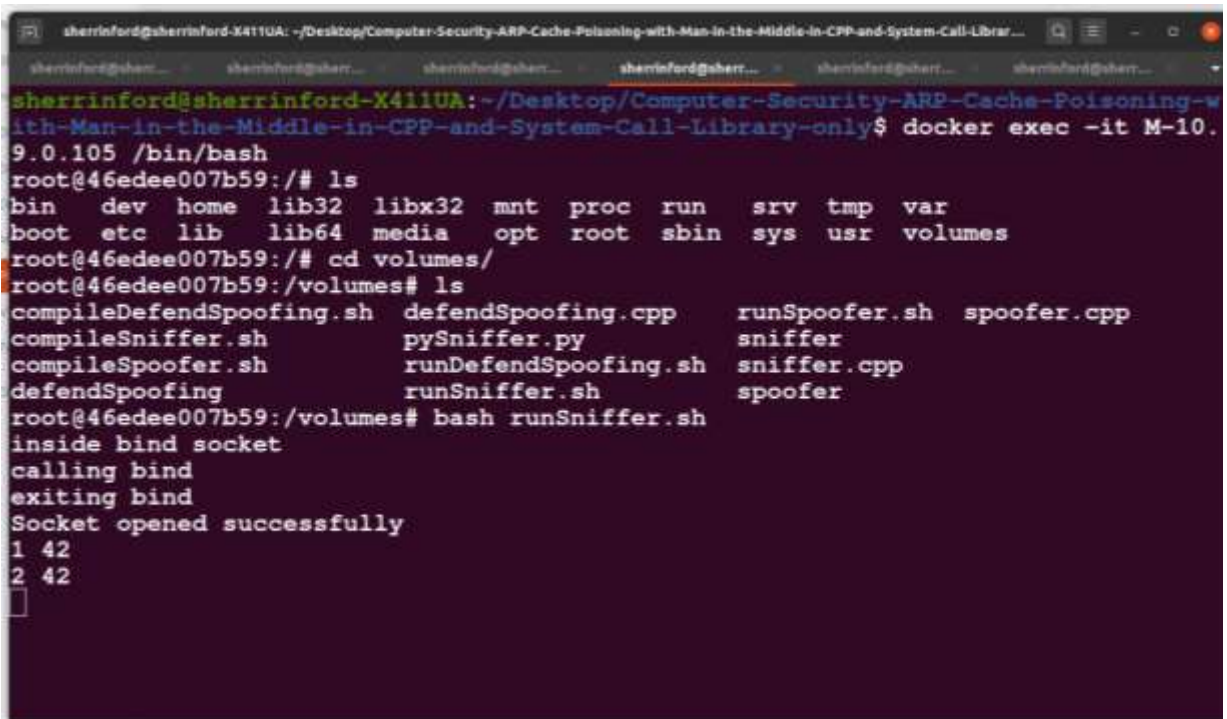
3. From host operating system compile all attack codes inside volumes folder.
Like below-

```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only/volumes$ ls
compileDefendSpoofing.sh  defendSpoofing.cpp  runSpoofer.sh  spoofer.cpp
compileSniffer.sh        pySniffer.py        sniffer
compileSpoofer.sh        runDefendSpoofing.sh  sniffer.cpp
defendSpoofing           runSniffer.sh       spoofer
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only/volumes$ bash compileDefendSpoofing.sh
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only/volumes$ bash compileSniffer.sh
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only/volumes$ bash compileSpoofer.sh
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only/volumes$
```

4. Go to an attacker terminal and run arp spoofing code. That will poison host-A and host-B's ARP cache.

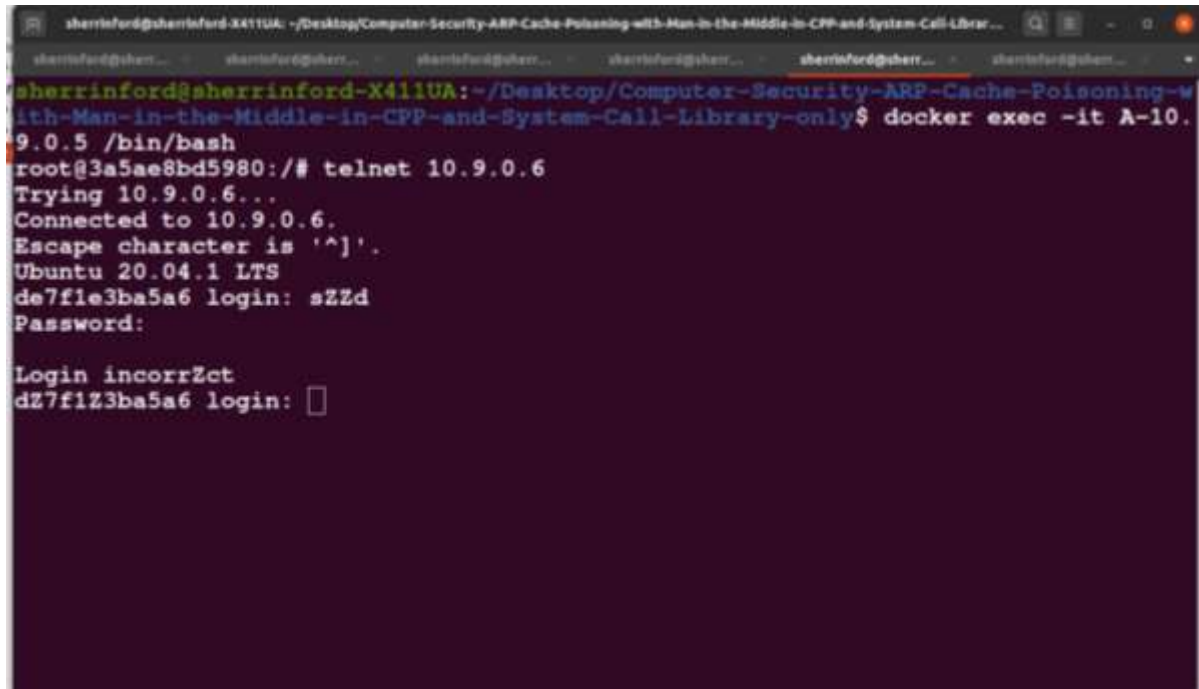
```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it M-10.9.0.105 /bin/bash
root@46edee007b59:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr  volumes
root@46edee007b59:/# cd volumes/
root@46edee007b59:/volumes# ls
compileDefendSpoofing.sh  defendSpoofing.cpp  runSpoofer.sh  spoofer.cpp
compileSniffer.sh        pySniffer.py       sniffer
compileSpoofer.sh        runDefendSpoofing.sh  sniffer.cpp
defendSpoofing           runSniffer.sh      spoofer
root@46edee007b59:/volumes# bash runSpoofer.sh
inside bind socket
calling bind
exiting bind
Socket opened successfully
inside craftARPFrame
ARP frame: 02 42 0a 09 00 05 02 42 0a 09 00 69 08 06 00 01 08 00 06 04 00 02 02
42 0a 09 00 69 0a 09 00 06 02 42 0a 09 00 05 0a 09 00 05
exiting crafting
ARP NO. 0
total sent: 42
sendVictim1: 0
```

5. Open another attacker terminal. At first stop ip forwarding and then we will run sniffing code like below.



```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it M-10.9.0.105 /bin/bash
root@46edee007b59:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr  volumes
root@46edee007b59:/# cd volumes/
root@46edee007b59:/volumes# ls
compileDefendSpoofing.sh  defendSpoofing.cpp  runSpoofer.sh  spoofer.cpp
compileSniffer.sh         pySniffer.py        sniffer
compileSpoofer.sh         runDefendSpoofing.sh  sniffer.cpp
defendSpoofing            runSniffer.sh        spoofer
root@46edee007b59:/volumes# bash runSniffer.sh
inside bind socket
calling bind
exiting bind
Socket opened successfully
1 42
2 42
█
```

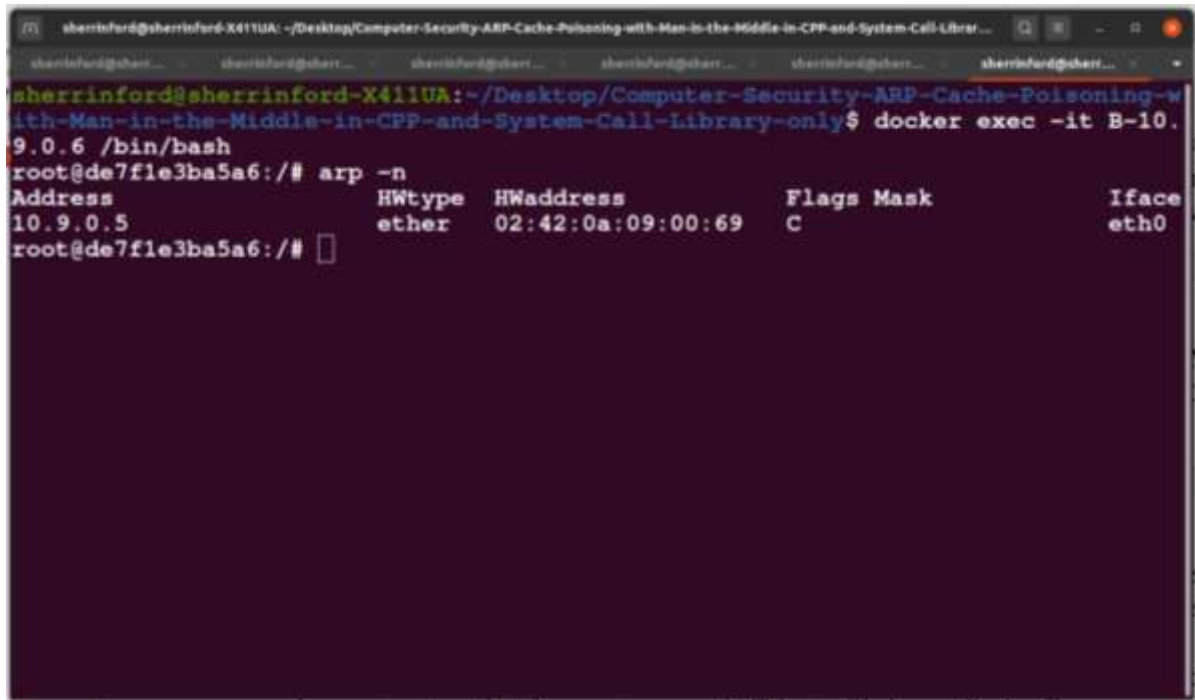
6. To demonstrate MITM in telnet application. We first run telnet command from host-A to host-B's IP address. As we can see when we enter username seed out attack code replace occurrence of letter 'e' with 'Z' hence even though we type seed on our keyboard it shows sZZd on the screen.



```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it A-10.9.0.5 /bin/bash
root@3a5ae8bd5980:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
de7f1e3ba5a6 login: sZZd
Password:

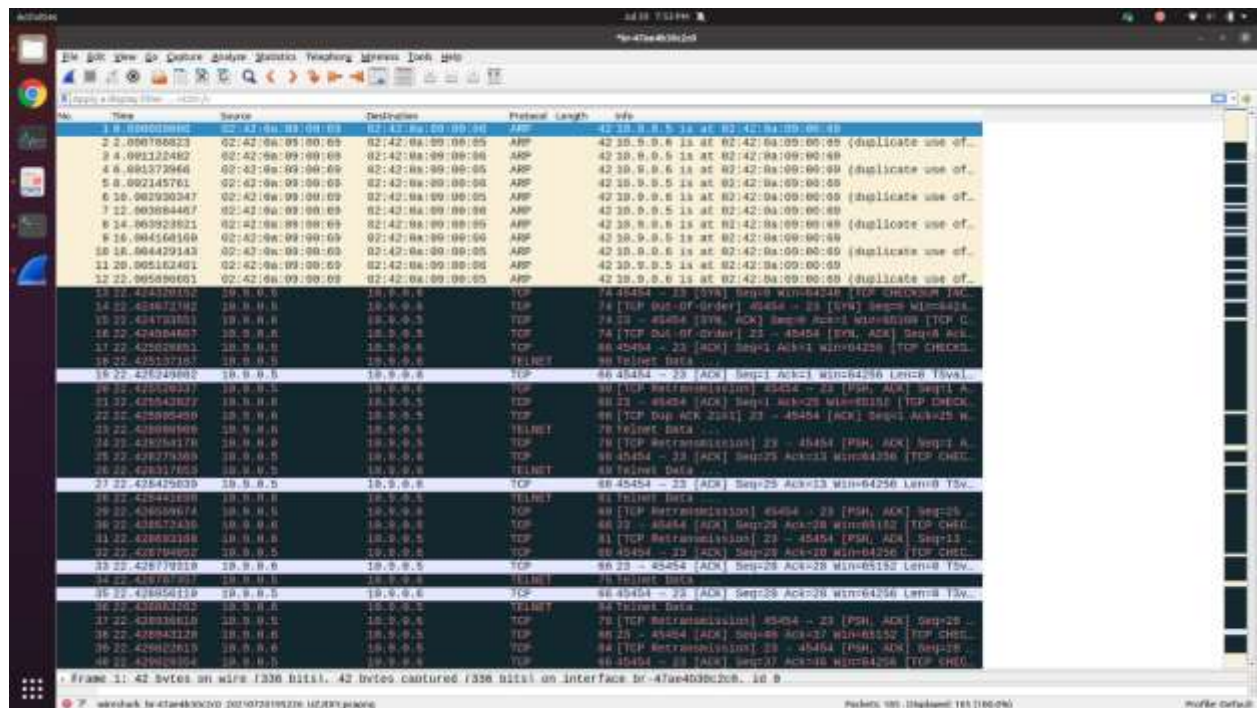
Login incorrZct
dZ7f1Z3ba5a6 login: 
```

7. To verify that indeed we have intercepted the message we can lookup arp table of host-B. It shows host-A's IP address is bind with hardware address of attacker like expected.



```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it B-10.9.0.6 /bin/bash
root@de7f1e3ba5a6:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.5         ether    02:42:0a:09:00:69  C             eth0
root@de7f1e3ba5a6:/#
```


8. We can verify the attack again using Wireshark. We have taken a screenshot of Docker network interface. As we can see from packet 1 to 12 are spoofed ARP reply. We can then see in 13 and 14 same packet is being passed from host-A to attacker machine and then attacker machine to host-B. For every packet sent from host-A to host-B and vice-versa attacker machine at first intercept it and then forward it with some modification. Here attacker just change any occurrence of letter 'e' with 'Z'.

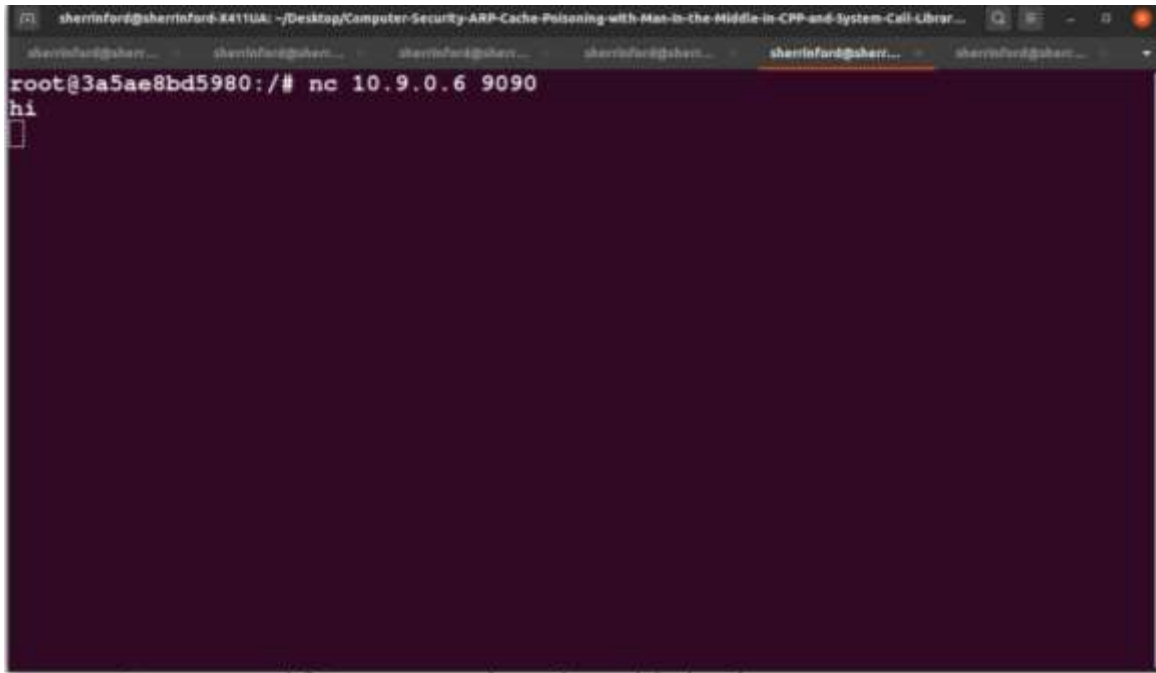


9. Now let's demonstrate netcat attack. We don't need to change or restart anything to demonstrate attack over netcat after we have shown telnet attack. In a host-B terminal run netcat on port 9090 like below-

A terminal window with a dark purple background. The title bar shows the user 'sherrinford' and the path '~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only'. The prompt is 'sherrinford@sherrinford-X411UA:~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only\$'. The command 'docker exec -it B-10.9.0.6 /bin/bash' has been entered. The prompt has changed to 'root@de7f1e3ba5a6:/#'. The command 'nc -lp 9090' has been entered, and a small white cursor is visible on the line.

```
sherrinford@sherrinford-X411UA:~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it B-10.9.0.6 /bin/bash
root@de7f1e3ba5a6:/# nc -lp 9090
```

10. Now in host-A try to connect with port 9090 of host-B and send message "hi". This message will also show up on host-B. This message was transferred unchanged as this message doesn't have any occurrence of 'e' in it. Snapshots will look like below-

A terminal window on host-A with a dark purple background. The prompt is 'root@3a5ae8bd5980:/#'. The command 'nc 10.9.0.6 9090' has been entered. The output 'hi' is displayed on the next line, followed by a cursor on a new line.

```
root@3a5ae8bd5980:/# nc 10.9.0.6 9090
hi

```

A terminal window on host-B with a dark purple background. The prompt is 'root@3a5ae8bd5980:/#'. The command 'nc 10.9.0.6 9090' has been entered. The output 'hi' is displayed on the next line, followed by a cursor on a new line.

```
root@3a5ae8bd5980:/# nc 10.9.0.6 9090
hi

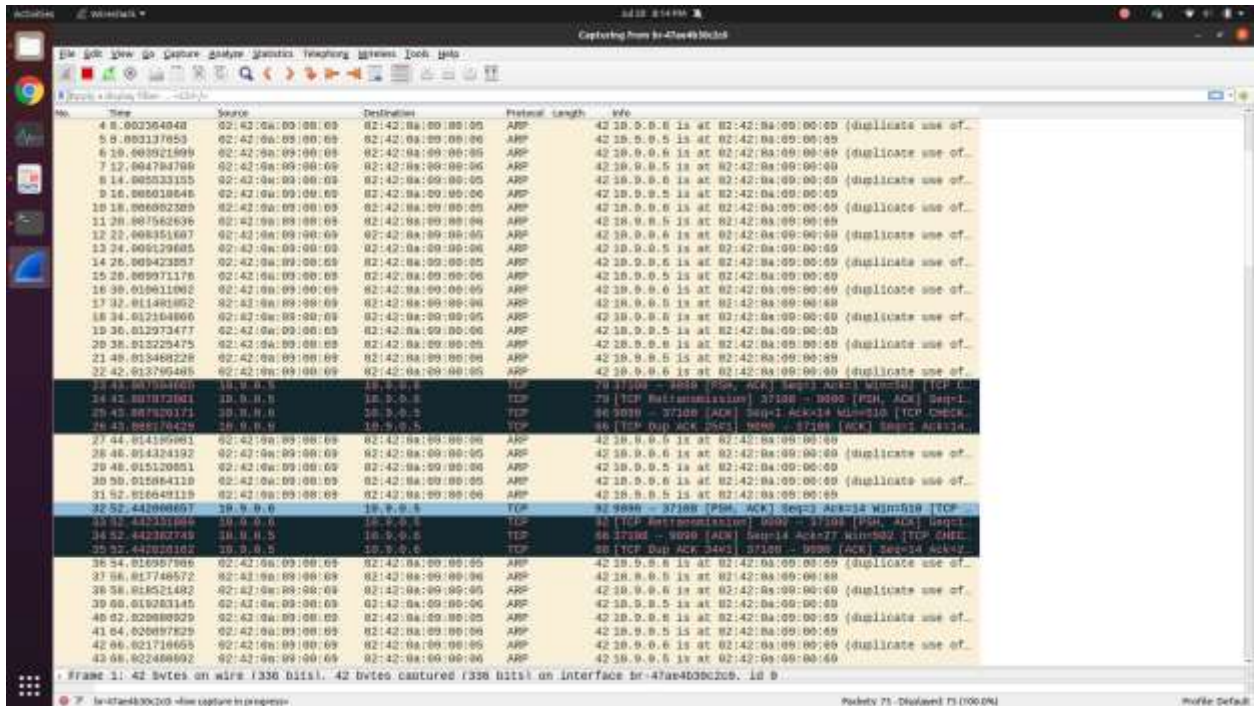
```

11. Now just sending some random message so that we can understand the changed done in MITM attack.

```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-with-Man-in-the-Middle-in-CPP-and-System-Call-Librar...
sherrinford@... sherrinford@... sherrinford@... sherrinford@... sherrinford@... sherrinford@... sherrinford@...
root@3a5ae8bd5980:/# nc 10.9.0.6 9090
hi
h2y
how are you?
I am finZ what about you?
```

```
sherrinford@sherrinford-X411UA: ~/Desktop/Computer-Security-ARP-Cache-Poisoning-w
ith-Man-in-the-Middle-in-CPP-and-System-Call-Library-only$ docker exec -it B-10.
9.0.6 /bin/bash
root@de7f1e3ba5a6:/# nc -lp 9090
hi
hey
how arZ you?
I am fine what about you?
```

12. Wireshark generates same sort of output like it did for telnet.



No.	Time	Source	Destination	Protocol	Length	Info
4	0.003304040	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
5	0.003317053	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
6	0.003321509	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
7	0.003326065	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
8	0.003330621	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
9	0.003335177	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
10	0.003339733	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
11	0.003344289	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
12	0.003348845	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
13	0.003353401	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
14	0.003357957	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
15	0.003362513	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
16	0.003367069	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
17	0.003371625	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
18	0.003376181	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
19	0.003380737	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
20	0.003385293	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
21	0.003389849	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
22	0.003394405	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
23	0.003398961	18.0.0.0	18.0.0.0	TCP	70	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
24	0.003403517	18.0.0.0	18.0.0.0	TCP	70	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
25	0.003408073	18.0.0.0	18.0.0.0	TCP	70	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
26	0.003412629	18.0.0.0	18.0.0.0	TCP	70	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
27	0.003417185	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
28	0.003421741	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
29	0.003426297	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
30	0.003430853	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
31	0.003435409	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
32	0.003440065	18.0.0.0	18.0.0.0	TCP	70	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
33	0.003444621	18.0.0.0	18.0.0.0	TCP	70	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
34	0.003449177	18.0.0.0	18.0.0.0	TCP	70	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
35	0.003453733	18.0.0.0	18.0.0.0	TCP	70	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
36	0.003458289	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
37	0.003462845	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
38	0.003467401	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
39	0.003471957	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
40	0.003476513	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
41	0.003481069	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...
42	0.003485625	02:42:5a:00:00:00	02:42:5a:00:00:00	ARP	42	18.0.0.0.13 at 02:42:5a:00:00:00 (duplicate use of...

Frame 11: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-47a4d09c20b, id 9

13. We have a counter measure tool that can detect if someone is spoofing host-B or not. This tool should be run from host-A like below. As output suggest it catches spoofed ARP reply from attacker immediately and reports back.

[illegible]

14.If we now close the spoofer in attacker that generates false ARP reply then our defense tool stop generating new message informing that no one is currently spoofing host-B.

[illegible]

Discussion

Our attack was completely successful as we can not only sniff frame transferred from host-A to host-B we could also modify any part of the frame if we wanted to. Output shown in step 6, 7, 8 for telnet and output shown in step 11, 12, 13 proves that our attack was successful indeed for netcat application.

We have also written a counter measure code. That will check whether any machine is between the communication of host-A to host-B. It sniffs all ARP packets and check whether anyone is spoofing host-B or not. In steps 14, 15 we have demonstrated its function.