



PES UNIVERSITY

Department of Computer Science & Engineering

Computer Network Security

UE23CS343AB6

Assignment 3 Submission

Name of the Student	Pranav Hemanth
SRN	PES1UG23CS433
Section	G
Department	CSE
Campus	RR

Computer Network Security

UE23CS343AB6

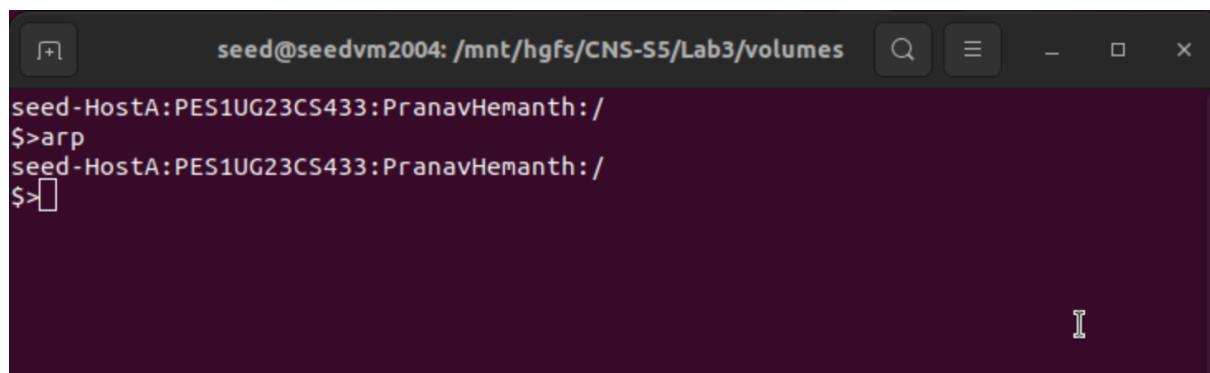
Task 1: Using ARP request

On host M, construct an ARP request packet and send it to host A. Check whether M's MAC address is mapped to B's IP address in A's ARP cache. We can do this in two ways:

Step1:

- 1) Without passing parameters to Ether()

Host A:



```
seed@seedvm2004: /mnt/hgfs/CNS-S5/Lab3/volumes
seed-HostA:PES1UG23CS433:PranavHemanth: /
$>arp
seed-HostA:PES1UG23CS433:PranavHemanth: /
$>
```

Host B:



```
seed@seedvm2004: ~
seed-HostB:PES1UG23CS433:PranavHemanth: /
$>arp
seed-HostB:PES1UG23CS433:PranavHemanth: /
$>
```

So as we see above, both hosts return an empty arp table. This is because there are no entries in either yet

Step2:

- 2) Run tcpdump on both the hosts A and B

```
tcpdump -i eth0 -n
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>[ ]
```

Host B:

```
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>[ ]
```

As we see above, currently there are no packets being sent to eth0. Hence we see no packets captured

Step3:

- 3) Finally, execute the below command on the attacker machine M

```
python3 task1A.py
```

Attacker M:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task1A.py
###[ Ethernet ]###
dst      = da:1d:23:10:a3:ac
src      = d6:a7:6b:5c:b4:f8
type    = ARP
###[ ARP ]###
hwtype   = Ethernet (10Mb)
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = 02:42:0a:09:00:05
pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:01:05.937197 ARP, Request who-has 10.9.0.5 tell 10.9.0.105, length 28
17:01:05.937244 ARP, Reply 10.9.0.5 is-at da:1d:23:10:a3:ac, length 28
17:01:05.985247 ARP, Request who-has 10.9.0.5 (02:42:0a:09:00:05) tell 10.9.0.6,
length 28
17:01:05.985270 ARP, Reply 10.9.0.5 is-at da:1d:23:10:a3:ac, length 28
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>■
```

Host B:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:01:05.937196 ARP, Request who-has 10.9.0.5 tell 10.9.0.105, length 28
17:01:05.985274 ARP, Reply 10.9.0.5 is-at da:1d:23:10:a3:ac, length 28
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>
```

Explanation:

Machine M uses Scapy to craft and send a forged ARP packet, pretending to be B (by spoofing B's IP), so that A updates its ARP table incorrectly. An Ethernet frame is built with Ether(), an ARP packet is spoofed, and sendp() transmits it. Meanwhile, tcpdump is run on A and B to capture traffic on interface eth0.

Step4:

- 4) run arp to check the entries

Attacker M: (Just out of curiosity)

```
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>arp
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
Address          HWtype  HWaddress          Flags Mask      Iface
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:69  C          eth0
M-10.9.0.105.net-10.9.0 ether   d6:a7:6b:5c:b4:f8  C          eth0
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>
```

Host B:

```
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>arp
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>
```

Explanation:

Displaying the ARP table output in all three machines. We notice that in the ARP table of A, both machines M and B have the same hardware address.

Step5:

- 5) Now delete the ARP Cache entries

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
Address          HWtype  HWaddress          Flags Mask      Iface  
B-10.9.0.6.net-10.9.0.0  ether    02:42:0a:09:00:69  C          eth0  
M-10.9.0.105.net-10.9.0   ether    d6:a7:6b:5c:b4:f8  C          eth0  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp -d 10.9.0.6  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp -d 10.9.0.105  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>
```

Step6:

- 6) Ether() with parameters

Screenshot of ARP cache output on both the hosts before the attack

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>
```

Host B:

```
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>■
```

Screenshot of ARP cache output on both the hosts and attacker after the attack

Attacker M:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task11A.py
###[ Ethernet ]###
dst      = 4e:de:c1:a5:6f:41
src      = 02:ca:f2:e1:f7:a0
type     = ARP
###[ ARP ]###
hwtype   = Ethernet (10Mb)
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc   = 02:ca:f2:e1:f7:a0
psrc    = 10.9.0.6
hwdst   = 4e:de:c1:a5:6f:41
pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>█
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:40:55.209365 ARP, Request who-has 10.9.0.5 (4e:de:c1:a5:6f:41) tell 10.
length 28
17:40:55.209403 ARP, Reply 10.9.0.5 is-at 4e:de:c1:a5:6f:41, length 28
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
Address          HWtype  HWaddress          Flags Mask
B-10.9.0.6.net-10.9.0.0  ether   02:ca:f2:e1:f7:a0  C
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>█
```

Host B:

```
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:40:55.209364 ARP, Request who-has 10.9.0.5 (4e:de:c1:a5:6f:41) tell 10.9.0.6,
  length 28
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>arp
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>
```

7) Now delete the ARP Cache entries

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
Address          HWtype  HWaddress          Flags Mask
B-10.9.0.6.net-10.9.0.0  ether   02:ca:f2:e1:f7:a0  C
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp -d 10.9.0.6
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>
```

Explanation:

Machine M crafts and sends a spoofed ARP packet as B to directly update A's ARP table. Unlike the earlier broadcast, this is a targeted attack using Ether() and ARP() with sendp(), so no broadcast is sent and M doesn't appear in the ARP table.

Answer the following questions:

1. What does the 'op' in the screenshot of the attacker machine signify? What is its default value?

Answer: OP is an abbreviation for operation code. It denotes if it's an ARP request or response. The default value is 1, which is ARP Request.

2. What was the difference between the ARP cache results after the attack in the above 2 approaches? Why did you observe this difference?

Answer: In method 1, the forged ARP packet is broadcast, so all hosts on the network receive it and their ARP caches are affected.

In method 2, the Ethernet destination MAC is set to the victim's address, so only the victim's ARP table is poisoned, making the attack targeted and stealthier.

Task 2: Using ARP Reply

OBJECTIVE: In this task you need to attack A using ARP reply packet.

Try the attack under the following two scenarios, and report the results of your attack:

Scenario 1: B's IP is already in A's cache.

Attacker M:

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task11A.py
###[ Ethernet ]###
dst      = 4e:de:c1:a5:6f:41
src      = 02:ca:f2:e1:f7:a0
type     = ARP
###[ ARP ]###
    hwtype   = Ethernet (10Mb)
    ptype    = IPv4
    hwlen    = None
    plen     = None
    op       = who-has
    hwsrc   = 02:ca:f2:e1:f7:a0
    psrc    = 10.9.0.6
    hwdst   = 4e:de:c1:a5:6f:41
    pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
Address          HWtype  HWaddress          Flags Mask
B-10.9.0.6.net-10.9.0.0  ether   02:ca:f2:e1:f7:a0  C
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>
```

Screenshot showing showing the cache entries of A after the attack

Attacker M:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task1B.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type    = ARP
###[ ARP ]###
    hwtype   = Ethernet (10Mb)
    ptype    = IPv4
    hwlen    = None
    plen     = None
    op       = is-at
    hwsrc   = 02:42:0a:09:00:69
    psrc    = 10.9.0.6
    hwdst   = 02:42:0a:09:00:05
    pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:48:59.093653 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
Address          HWtype  HWaddress          Flags Mask
B-10.9.0.6.net-10.9.0.0  ether   02:ca:f2:e1:f7:a0  C
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>
```

Explanation:

The attacker poisons A's ARP table by sending a fake ARP reply that maps B's IP to a false MAC. Since ARP replies are always accepted, A updates its table, leading to ARP cache poisoning.

Scenario 2: B's IP is not in A's cache.

Now delete the ARP Cache entries

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
Address          HWtype  HWaddress          Flags Mask  
B-10.9.0.6.net-10.9.0.0  ether   02:ca:f2:e1:f7:a0  C  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp -d 10.9.0.6  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>
```

Screenshot showing showing the cache entries of A after the attack

Screenshot of the packets captured using tcpdump

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>tcpdump -i eth0 -n  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes  
17:53:10.169309 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28  
^C  
1 packet captured  
1 packet received by filter  
0 packets dropped by kernel  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>■
```

Screenshot of the attacker terminal as well

Attacker M:

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task1B.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = Ethernet (10Mb)
ptype    = IPv4
hwlen    = None
plen     = None
op       = is-at
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = 02:42:0a:09:00:05
pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>
```

Explanation:

In this case, A's ARP table remains unchanged because it has never communicated with B. As a result, when a reply from B arrives, A does not add it to its ARP table.

Answer the following question:

1. What does op=2 mean?

Answer:

Task 3: Using ARP Gratuitous Message:

OBJECTIVE: On host M, construct an ARP gratuitous packet, and use it to map B's IP address to M's MAC address. Please launch the attack under the same two scenarios as those described in Task 1.B.

Scenario 1: B's IP is already in A's cache.

Screenshot showing A's cache (before attack):

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>
```

Screenshot showing A's cache after inserting B's IP:

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
Address          HWtype  HWaddress          Flags Mask  
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:69  C  
M-10.9.0.105.net-10.9.0  ether   f6:2a:c5:db:74:d3  C  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>
```

Attacker M:

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes  
$>python3 task1A.py  
###[ Ethernet ]###  
dst      = 4e:de:c1:a5:6f:41  
src      = f6:2a:c5:db:74:d3  
type     = ARP  
###[ ARP ]###  
hwtype   = Ethernet (10Mb)  
ptype    = IPv4  
hwlen    = None  
plen     = None  
op       = who-has  
hwsrc   = 02:42:0a:09:00:69  
psrc    = 10.9.0.6  
hwdst   = 02:42:0a:09:00:05  
pdst    = 10.9.0.5  
  
.Sent 1 packets.  
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes  
$>
```

Screenshot of the packets captured using tcpdump

Screenshot showing A's cache after the attack

Screenshots of terminals B and M after the attack

Host A:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>tcpdump -i eth0 -n  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes  
18:01:41.033192 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28  
^C  
1 packet captured  
1 packet received by filter  
0 packets dropped by kernel  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
Address          HWtype  HWaddress          Flags Mask  
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:69  C  
M-10.9.0.105.net-10.9.0  ether   f6:2a:c5:db:74:d3  C  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>
```

Host B:

```
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>tcpdump -i eth0 -n  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes  
18:01:41.033190 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28  
^C  
1 packet captured  
1 packet received by filter  
0 packets dropped by kernel  
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>
```

Attacker M:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-attacker-M: PES1UG23CS433: PranavHemanth: /volumes
$> python3 task1C.py
###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = Ethernet (10Mb)
ptype    = IPv4
hwlen    = None
plen     = None
op       = is-at
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = ff:ff:ff:ff:ff:ff
pdst    = 10.9.0.6

.
Sent 1 packets.
seed-attacker-M: PES1UG23CS433: PranavHemanth: /volumes
$>
```

Explanation:

Gratuitous ARP packets are broadcast by new devices on a network to announce their MAC address to all other hosts. This technique can be used in an attempt to poison ARP tables, but whether the tables actually update depends on the operating system.

Scenario 2: B's IP is not in A's cache.

Screenshot showing that the cache entries are deleted for HostA and HostB

Host A:

```
seed-HostA: PES1UG23CS433: PranavHemanth: /
$> arp
Address          HWtype  HWaddress          Flags Mask
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:69  C
M-10.9.0.105.net-10.9.0  ether   f6:2a:c5:db:74:d3  C
seed-HostA: PES1UG23CS433: PranavHemanth: /
$> arp -d 10.9.0.6
seed-HostA: PES1UG23CS433: PranavHemanth: /
$> arp -d 10.9.0.105
seed-HostA: PES1UG23CS433: PranavHemanth: /
$> arp
seed-HostA: PES1UG23CS433: PranavHemanth: /
$>
```

Host B:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>[]
```

After Attack:

Screenshot of the packets captured using tcpdump

Screenshot showing A's cache after the attack

Screenshots of terminals B and M after the attack

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>tcpdump -i eth0 -n  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes  
18:06:57.661517 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28  
^C  
1 packet captured  
1 packet received by filter  
0 packets dropped by kernel  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>
```

Host B:

```
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>tcpdump -i eth0 -n  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes  
18:06:57.661516 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28  
^C  
1 packet captured  
1 packet received by filter  
0 packets dropped by kernel  
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>[]
```

Attacker M:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task1C.py
###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
    hwtype   = Ethernet (10Mb)
    ptype    = IPv4
    hwlen    = None
    plen     = None
    op       = is-at
    hwsrc   = 02:42:0a:09:00:69
    psrc    = 10.9.0.6
    hwdst   = ff:ff:ff:ff:ff:ff
    pdst    = 10.9.0.6

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>
```

Explanation:

In this case, A's ARP table does not get updated with any new entries. This happens because A has never interacted with B, so when it receives a gratuitous ARP message from B, it ignores it and doesn't record it in its table.

Answer the following question:

1. Why does VM B's ARP cache remain unchanged in this approach even though the packet was broadcasted on the network?

Answer: Host B's ARP cache remains the same since a system does not overwrite the ARP entry for its own IP address and simply ignores such packets. In contrast, other machines like Host A can be deceived into updating their cache with the false information.

Task 4: MITM Attack on Telnet using ARP Cache Poisoning

OBJECTIVE: Hosts A and B are communicating using Telnet, and Host M wants to intercept their communication, so it can make changes to the data sent between A and B.

Step 1 - Launch the ARP cache poisoning attack

First, Host M conducts an ARP cache poisoning attack on both A and B. After this step, packets sent between A and B will all be sent to M.

Answer the following question:

What will be the mappings in A and B's cache after the cache poisoning attack by M?

Answer:

After the ARP cache poisoning attack by M:

- Host A's ARP cache will map B's IP address to M's MAC address instead of B's real MAC.
- Host B's ARP cache will map A's IP address to M's MAC address instead of A's real MAC.

This ensures that all traffic between A and B is redirected through M.

Screenshot of A and B's caches (before attack):

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostA:PES1UG23CS433:PranavHemanth:/
```

Host B:

```
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>arp  
seed-HostB:PES1UG23CS433:PranavHemanth:/
```

ARP cache poisoning attack from Task 1:

Attacker M:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task11A.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = 02:42:0a:09:00:05
pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
Address          HWtype  HWaddress          Flags Mask           Iface
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:69  C             eth0
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>■
```

Execute the below code to map A's IP address to M's MAC address in B's ARP Cache:

Screenshot of B's cache and M's terminal:

Attacker M:

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task2.py
.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>■
```

Host B:

```
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>arp
Address          HWtype  HWaddress          Flags Mask           Iface
A-10.9.0.5.net-10.9.0.0  ether   02:42:0a:09:00:69  C             eth0
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>■
```

Step 2 - Testing

Screenshot of A's terminal

Screenshot of the Wireshark packet capture

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

Attacker M:

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task11A.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc    = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst    = 02:42:0a:09:00:05
pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task2.py
.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>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=9 ttl=64 time=0.428 ms
64 bytes from 10.9.0.6: icmp_seq=10 ttl=64 time=0.126 ms
64 bytes from 10.9.0.6: icmp_seq=11 ttl=64 time=0.090 ms
64 bytes from 10.9.0.6: icmp_seq=12 ttl=64 time=0.136 ms
64 bytes from 10.9.0.6: icmp_seq=13 ttl=64 time=0.110 ms
64 bytes from 10.9.0.6: icmp_seq=14 ttl=64 time=0.068 ms
64 bytes from 10.9.0.6: icmp_seq=15 ttl=64 time=0.089 ms
64 bytes from 10.9.0.6: icmp_seq=16 ttl=64 time=0.083 ms
64 bytes from 10.9.0.6: icmp_seq=17 ttl=64 time=0.075 ms
^C
--- 10.9.0.6 ping statistics ---
17 packets transmitted, 9 received, 47.0588% packet loss, time 16366ms
rtt min/avg/max/mdev = 0.068/0.133/0.428/0.106 ms
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>
```

Wireshark:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

No.	Time	Source	Destination	Protocol	Length	Info
1	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:05	ARP	42	42 who has 10.9.0.5 Tell 10.9.0.6
2	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:06	ARP	42	42 who has 10.9.0.5 Tell 10.9.0.6 (duplicate use of 10.9.0.5 de...
3	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:06	ARP	42	42 who has 10.9.0.5 Tell 10.9.0.6 (duplicate use of 10.9.0.5 de...
4	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:06	ARP	42	42 who has 10.9.0.5 is at 02:42:0a:09:00:06 (duplicate use of 10.9.0.5 de...
5	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (no respons...
6	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=2/256, ttl=64 (no respons...
7	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=3/256, ttl=64 (no respons...
8	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=4/256, ttl=64 (no respons...
9	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=5/256, ttl=64 (no respons...
10	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:05	ARP	42	42 who has 10.9.0.5 Tell 10.9.0.5
11	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=6/256, ttl=64 (no respons...
12	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:05	ARP	42	42 who has 10.9.0.5 Tell 10.9.0.5
13	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=7/256, ttl=64 (no respons...
14	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:05	ARP	42	42 who has 10.9.0.5 Tell 10.9.0.5
15	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=8/256, ttl=64 (no respons...
16	2025-08-27 03:4...	02:42:0a:09:00:05	Broadcast	ARP	42	42 who has 10.9.0.5 Tell 10.9.0.5
17	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:05	ARP	42	42 who has 10.9.0.6 is at 02:42:0a:09:00:05
18	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=9/256, ttl=64 (reply in ...
19	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply id=0x0001, seq=9/256, ttl=64 (request i...
20	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=10/256, ttl=64 (reply in ...
21	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply id=0x0001, seq=10/256, ttl=64 (request i...
22	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=11/256, ttl=64 (reply in ...
23	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply id=0x0001, seq=11/256, ttl=64 (request i...
24	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=12/256, ttl=64 (reply in ...
25	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply id=0x0001, seq=12/256, ttl=64 (request i...
26	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x0001, seq=13/256, ttl=64 (reply in ...
27	2025-08-27 03:4...	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) reply id=0x0001, seq=13/256, ttl=64 (request i...
28	2025-08-27 03:4...	02:42:0a:09:00:05	02:42:0a:09:00:05	ARP	42	42 who has 10.9.0.5 Tell 10.9.0.6 (duplicate use of 10.9.0.5 de...

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-ef9d1f472261, id 0
 Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
 > Address Resolution Protocol (request)

```

0000 02 42 0a 09 00 05 02 42 0a 09 00 06 -B- ... B- :1...
0010 08 00 06 04 09 03 02 42 0a 09 00 09 0a 09 00 06 -B- ... B- :1...
0020 02 42 0a 09 00 05 0a 09 00 05
  
```

Explanation:

Since A's ARP table has been poisoned, B's IP is mapped to M's MAC address. With IP forwarding disabled on M, the packets are dropped. After repeated transmission failures, A eventually broadcasts a query asking who owns IP 10.9.0.6, updates its ARP table with the correct entry, and is then able to ping B successfully.

Answer the following question:

1. What do you observe? Explain

Answer: Since A's ARP table has been poisoned, B's IP is mapped to M's MAC address. With IP forwarding disabled on M, the packets are dropped. After repeated transmission failures, A eventually broadcasts a query asking who owns IP 10.9.0.6, updates its ARP table with the correct entry, and is then able to ping B successfully.

Step 3 - Turn on IP Forwarding

Now we turn on the IP forwarding on Host M so that it will forward the packets between A and B.

Screenshot of A's terminal

Screenshot of the Wireshark packet capture

Attacker M:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task1A.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc    = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst    = 02:42:0a:09:00:05
pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task2.py
.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>■
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>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=0.363 ms
From 10.9.0.105 icmp_seq=2 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.200 ms
From 10.9.0.105 icmp_seq=3 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.282 ms
From 10.9.0.105 icmp_seq=4 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=4 ttl=63 time=0.358 ms
From 10.9.0.105 icmp_seq=5 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=5 ttl=63 time=0.324 ms
From 10.9.0.105 icmp_seq=6 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=6 ttl=63 time=0.124 ms
64 bytes from 10.9.0.6: icmp_seq=7 ttl=63 time=0.257 ms
From 10.9.0.105 icmp_seq=8 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=8 ttl=63 time=0.272 ms
^C
--- 10.9.0.6 ping statistics ---
8 packets transmitted, 8 received, +6 errors, 0% packet loss, time 7152ms
rtt min/avg/max/mdev = 0.124/0.272/0.363/0.075 ms
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>■
```

Wireshark:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

No.	Time	Source	Destination	Protocol	Length	Info
1	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) Request id=0x0002, seq=1/256, ttl=64 (no respons..
2	2025-08-27 04:00:10.9.0.5	10.9.0.6		ARP	64	42.19.9.0.6 is at 02:42:0a:09:00:69
3	2025-08-27 04:00:10.9.0.5	10.9.0.6		ARP	64	42.19.9.0.6 is at 02:42:0a:09:00:69
4	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) request id=0x0002, seq=1/256, ttl=63 (reply in 5)
5	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) reply id=0x0002, seq=1/256, ttl=64 (request in..
6	2025-08-27 04:00:10.9.0.105	10.9.0.6		ICMP	126	Redirect (Redirect for host)
7	2025-08-27 04:00:10.9.0.105	Broadcast		ARP	64	42 Who has 10.9.0.6? Tell 10.9.0.105
8	2025-08-27 04:00:10.9.0.105	10.9.0.6		ARP	64	42.19.9.0.6 is at 02:42:0a:09:00:69
9	2025-08-27 04:00:10.9.0.6	10.9.0.5		ICMP	98	Echo (ping) reply id=0x0002, seq=1/256, ttl=63
10	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) request id=0x0002, seq=2/252, ttl=64 (no respons..
11	2025-08-27 04:00:10.9.0.105	10.9.0.5		ICMP	126	Redirect (Redirect for host)
12	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) request id=0x0002, seq=2/252, ttl=63 (reply in 1..
13	2025-08-27 04:00:10.9.0.6	10.9.0.5		ICMP	98	Echo (ping) reply id=0x0002, seq=2/252, ttl=64 (request in..
14	2025-08-27 04:00:10.9.0.105	10.9.0.5		ICMP	126	Redirect (Redirect for host)
15	2025-08-27 04:00:10.9.0.6	10.9.0.5		ICMP	98	Echo (ping) reply id=0x0002, seq=2/252, ttl=63
16	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) request id=0x0002, seq=3/768, ttl=64 (no respons..
17	2025-08-27 04:00:10.9.0.105	10.9.0.5		ICMP	126	Redirect (Redirect for host)
18	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) request id=0x0002, seq=3/768, ttl=63 (reply in 1..
19	2025-08-27 04:00:10.9.0.6	10.9.0.5		ICMP	98	Echo (ping) reply id=0x0002, seq=3/768, ttl=64 (request in..
20	2025-08-27 04:00:10.9.0.105	10.9.0.6		ICMP	126	Redirect (Redirect for host)
21	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) reply id=0x0002, seq=4/1824, ttl=63
22	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) request id=0x0002, seq=4/1824, ttl=64 (no respons..
23	2025-08-27 04:00:10.9.0.105	10.9.0.5		ICMP	126	Redirect (Redirect for host)
24	2025-08-27 04:00:10.9.0.6	10.9.0.5		ICMP	98	Echo (ping) request id=0x0002, seq=4/1824, ttl=63 (reply in ..
25	2025-08-27 04:00:10.9.0.6	10.9.0.5		ICMP	98	Echo (ping) reply id=0x0002, seq=4/1824, ttl=64 (request in..
26	2025-08-27 04:00:10.9.0.105	10.9.0.5		ICMP	126	Redirect (Redirect for host)
27	2025-08-27 04:00:10.9.0.5	10.9.0.5		ICMP	98	Echo (ping) reply id=0x0002, seq=5/1280, ttl=63
28	2025-08-27 04:00:10.9.0.5	10.9.0.6		ICMP	98	Echo (ping) request id=0x0002, seq=5/1280, ttl=64 (no respons..

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-ef0d1472261, id 0
 > Ethernet II, Src: 02:42:0a:09:00:69 (02:42:0a:09:00:69), Dst: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
 > Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
 > Internet Control Message Protocol

```

0000  02 42 0a 09 00 69 02 42 00 09 00 05 08 00 45 00 |B-----E
0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |T @ @ .....h
0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....!#%
0030  00 00 05 1b 03 00 00 00 00 00 10 12 13 14 25 .....!#%
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 .....!#%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &(')*,- ./012345

```

Explanation:

With A's ARP table poisoned, B's IP is associated with M's MAC address. Since IP forwarding is enabled on M, the packets are routed through M. This behavior is visible in both the Wireshark capture and A's terminal output, where the redirect entries confirm the packet forwarding.

Answer the following question:

1. Compare the results between the above two steps.

Answer:

In step 2, when IP forwarding on M is disabled, A's packets destined for B are sent to M but get dropped, breaking communication between A and B. After multiple failed attempts, A realizes the mismatch, broadcasts an ARP request, and restores the correct mapping in its ARP table, allowing communication with B to resume.

In step 3, with IP forwarding enabled on M, the poisoned ARP entry persists, and packets from A are redirected through M instead of being dropped. Communication between A and B continues but is now intercepted by M. The presence of redirect entries in Wireshark and A's terminal confirms that M is actively relaying the traffic.

Step 4 - Launch the MITM Attack

We are ready to make changes to the Telnet data between A and B.

OBJECTIVE: From the previous steps, you are able to redirect the TCP packets to Host M, but instead of forwarding them, you should replace them with a spoofed packet. write a sniff-and-spoof program to accomplish this goal. In particular, we would like to do the following:

Reminder - Make sure to execute Step 1 to update the ARP tables and open Wireshark on the given interface (br-****).

Setup before telnet:

Attacker M:

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task11A.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = 02:42:0a:09:00:05
pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task2.py
.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>arp
Address          HWtype  HWaddress          Flags Mask           Iface
M-10.9.0.105.net-10.9.0  ether   02:42:0a:09:00:69  C               eth0
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:69  C               eth0
seed-HostA:PES1UG23CS433:PranavHemanth:/
```

Host B:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>arp  
Address          HWtype  HWaddress      Flags Mask    Iface  
M-10.9.0.105.net-10.9.0  ether   02:42:0a:09:00:69  C        eth0  
A-10.9.0.5.net-10.9.0.0  ether   02:42:0a:09:00:69  C        eth0  
seed-HostB:PES1UG23CS433:PranavHemanth:/  
$>
```

Please type something on Host A's Telnet window, and see the packets captured on Wireshark, take a screenshot of the same.

Attacker M:

```
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes  
$>python3 mitm.py  
LAUNCHING MITM ATTACK.....  
*** b's', length: 1  
.Sent 1 packets.  
.Sent 1 packets.  
.Sent 1 packets.  
*** b's', length: 1  
.Sent 1 packets.  
.Sent 1 packets.  
.Sent 1 packets.  
*** b's', length: 1  
.Sent 1 packets.  
.Sent 1 packets.  
.Sent 1 packets.  
*** b's', length: 1  
.Sent 1 packets.  
.Sent 1 packets.  
.Sent 1 packets.
```

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

```
Sent 1 packets.  
.Sent 1 packets.  
*** b'f', length: 1  
.Sent 1 packets.  
.Sent 1 packets.  
*** b'nj', length: 2  
.Sent 1 packets.  
.Sent 1 packets.  
*** b'vn', length: 2  
.Sent 1 packets.  
.Sent 1 packets.  
*** b'j', length: 1  
.Sent 1 packets.  
.Sent 1 packets.  
*** b'n', length: 1  
.Sent 1 packets.  
.Sent 1 packets.  
.Sent 1 packets.  
█
```

Host A:

```
seed-HostA:PES1UG23CS433:PranavHemanth:/  
$>telnet 10.9.0.6  
Trying 10.9.0.6...  
Connected to 10.9.0.6.  
Escape character is '^]'.  
Ubuntu 20.04.1 LTS  
c1781ae499dd login: seed  
Password:  
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-139-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 Aug 26 17:12:09 UTC 2025 from A-10.9.0.5.net-10.9.0.0 on pts/5  
seed@c1781ae499dd:~$ ZZZZZZZZZZZZZZ█
```

Wireshark:

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

No.	Time	Source	Destination	Protocol	Length	Info
1	2025-08-27 04:08	10.9.0.5	10.9.0.4	TCP	74	[SYN] Seq=1627871696 Win=64240 Len=8 NS=1400 SACK
2	2025-08-27 04:08	10.9.0.5	10.9.0.6	TCP	74	[TCP Out-Of-Order] 35894 - 23 [SYN] Seq=1627871699 Win=64248
3	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	74	23 - 35894 [SYN, ACK] Seq=530249157 Ack=1627871697 Win=65160
4	2025-08-27 04:08	10.9.0.6	10.9.0.6	ICMP	102	Redirect (Redirect for host)
5	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	74	[TCP Out-Of-Order] 23 - 35894 [SYN, ACK] Seq=530249157 Ack=10
6	2025-08-27 04:08	10.9.0.6	10.9.0.6	TCP	66	35894 - 23 [ACK] Seq=1627871697 Ack=530249158 Win=64256 Len=8
7	2025-08-27 04:08	10.9.0.6	10.9.0.6	TCP	66	[TCP Dup ACK 6#1] 35894 - 23 [ACK] Seq=1627871697 Ack=530249158
8	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	59	Telnet Data ...
9	2025-08-27 04:08	10.9.0.5	10.9.0.6	TCP	80	[TCP Retransmission] 35894 - 23 [PSH, ACK] Seq=1627871697 Ack=
10	2025-08-27 04:08	10.9.0.8	10.9.0.6	TCP	66	23 - 35894 [ACK] Seq=530249158 Ack=1627871721 Win=65152 Len=8
11	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	66	[TCP Dup ACK 10#1] 23 - 35894 [ACK] Seq=530249158 Ack=1027871
12	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	78	Telnet Data ...
13	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	78	[TCP Retransmission] 23 - 35894 [PSH, ACK] Seq=530249158 Ack=
14	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	66	35894 - 23 [ACK] Seq=1627871721 Ack=5302491797 Win=64256 Len=8
15	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	60	[TCP Dup ACK 14#1] 35894 - 23 [ACK] Seq=1627871721 Ack=5302491797
16	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	81	Telnet Data ...
17	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	81	[TCP Retransmission] 23 - 35894 [PSH, ACK] Seq=5302491798 Ack=
18	2025-08-27 04:08	10.9.0.5	10.9.0.6	TCP	66	35894 - 23 [ACK] Seq=1627871721 Ack=530249185 Win=64256 Len=8
19	2025-08-27 04:08	10.9.0.5	10.9.0.6	TCP	66	[TCP Dup ACK 10#1] 35894 - 23 [ACK] Seq=1627871721 Ack=530249185
20	2025-08-27 04:08	10.9.0.6	10.9.0.6	TELNET	78	Telnet Data ...
21	2025-08-27 04:08	10.9.0.6	10.9.0.6	TCP	78	[TCP Retransmission] 35894 - 23 [PSH, ACK] Seq=1627871721 Ack=
22	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	84	Telnet Data ...
23	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	78	[TCP Retransmission] 23 - 35894 [PSH, ACK] Seq=530249185 Ack=
24	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	189	Telnet Data ...
25	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	180	[TCP Retransmission] 35894 - 23 [PSH, ACK] Seq=1627871733 Ack=
26	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	69	Telnet Data ...
27	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	69	[TCP Retransmission] 23 - 35894 [PSH, ACK] Seq=530249203 Ack=
28	2025-08-27 04:08	10.9.0.5	10.9.0.6	TELNET	69	Telnet Data ...

No.	Time	Source	Destination	Protocol	Length	Info
98	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TELNET	67	Telnet Data ...	
99	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TCP	67	[TCP Keep-Alive] 35894 -> [PSH, ACK] Seq=1027871782 Ack=530...	
100	2025-08-27 04:0... 10.9.0.5	10.9.0.5	TCP	66	[TCP Dup ACK 16081] 35894 -> [ACK] Seq=530249265 Ack=1027871783 Win=65152 Len=0	
101	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TELNET	68	Telnet Data ...	
102	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TELNET	68	Telnet Data ...	
103	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TCP	68	[TCP Retransmission] 35894 -> [PSH, ACK] Seq=1027871783 Ack=530249265	
104	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TCP	66	23 -> [ACK] Seq=530249265 Ack=1027871785 Win=65152 Len=0	
105	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TCP	66	[TCP Dup ACK 10414] 23 -> [ACK] Seq=530249265 Ack=102787...	
106	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TELNET	68	Telnet Data ...	
107	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TCP	66	[TCP Dup ACK 10414] 35894 -> [PSH, ACK] Seq=530249265 Ack=1027871785	
108	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TCP	66	35894 -> [ACK] Seq=1027871785 Ack=530249265 Win=65128 Len=8	
109	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TCP	66	[TCP Dup ACK 10801] 35894 -> [ACK] Seq=1027871785 Ack=53024...	
110	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TELNET	478	Telnet Data ...	
111	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TCP	478	[TCP Retransmission] 23 -> [PSH, ACK] Seq=530249267 Ack=53024...	
112	2025-08-27 04:0... 10.9.0.6	10.9.0.6	TCP	66	35894 -> [ACK] Seq=1027871785 Ack=530249679 Win=64128 Len=0	
113	2025-08-27 04:0... 10.9.0.6	10.9.0.6	TCP	66	[TCP Dup ACK 11241] 35894 -> [ACK] Seq=1027871785 Ack=53024...	
114	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TELNET	146	Telnet Data ...	
115	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TCP	146	[TCP Retransmission] 23 -> [PSH, ACK] Seq=530249579 Ack=53024...	
116	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TCP	66	35894 -> [ACK] Seq=1027871785 Ack=530249579 Win=64128 Len=0	
117	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TCP	66	[TCP Dup ACK 11601] 35894 -> [ACK] Seq=1027871785 Ack=53024...	
118	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TELNET	87	Telnet Data ...	
119	2025-08-27 04:0... 10.9.0.6	10.9.0.5	TCP	73	[TCP Retransmission] 23 -> [PSH, ACK] Seq=530249579 Ack=53024...	
120	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TCP	66	35894 -> [ACK] Seq=1027871785 Ack=530249780 Win=64128 Len=8	
121	2025-08-27 04:0... 10.9.0.5	10.9.0.6	TCP	66	[TCP Dup ACK 12081] 35894 -> [ACK] Seq=1027871785 Ack=53024...	
122	2025-08-27 04:0... 02:42:08:09:00:69	02:42:08:09:00:69	ARP	42	Who has 10.9.0.5 Tell 10.9.0.6	
123	2025-08-27 04:0... 02:42:08:09:00:65	02:42:08:09:00:65	ARP	42	10.9.0.5 is at 02:42:08:09:00:65	
124	2025-08-27 04:0... 02:42:08:09:00:69	02:42:08:09:00:69	ARP	42	10.9.0.5 is at 02:42:08:09:00:69 (duplicate use of 10.9.0.5 de...	
125	2025-08-27 04:0... 02:42:08:09:00:69	02:42:08:09:00:69	ARP	42	10.9.0.5 is at 02:42:08:09:00:69 (duplicate use of 10.9.0.5 de...	

No.	Time	Source	Destination	Protocol	Length	Info
180	2025-08-27 04:08	10.9.0.5	10.9.0.6	TELNET	67	Telnet Data ...
181	2025-08-27 04:08	10.9.0.5	10.9.0.6	TELNET	67	[TCP Keep-Alive] 35894 - 23 [PSH, ACK] Seq=1027871793 Ack=530...
182	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	67	Telnet Data ...
183	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	67	[TCP Keep-Alive] 23 - 35894 [PSH, ACK] Seq=530249789 Ack=1027...
184	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	66	35894 - 23 [ACK] Seq=1027871794 Ack=530249789 Win=64128 Len=0...
185	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	67	Telnet Data ...
186	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	66	[TCP Keep-Alive] 35894 - 23 [ACK] Seq=1027871794 Ack=53024978...
187	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	67	[TCP Keep-Alive] 35894 - 23 [PSH, ACK] Seq=1027871794 Ack=530...
188	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	67	Telnet Data ...
189	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	67	[TCP Keep-Alive] 23 - 35894 [PSH, ACK] Seq=530249789 Ack=1027...
190	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	68	Telnet Data ...
191	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	68	[TCP Retransmission] 35894 - 23 [PSH, ACK] Seq=1027871795 Ack=...
192	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	68	Telnet Data ...
193	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	68	[TCP Retransmission] 35894 - 23 [PSH, ACK] Seq=530249790 Ack=...
194	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	68	Telnet Data ...
195	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	68	[TCP Retransmission] 35894 - 23 [PSH, ACK] Seq=1027871797 Ack=...
196	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	68	Telnet Data ...
197	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	68	[TCP Retransmission] 23 - 35894 [PSH, ACK] Seq=530249792 Ack=...
198	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	68	Telnet Data ...
199	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	68	[TCP Keep-Alive] 35894 - 23 [PSH, ACK] Seq=1027871799 Ack=530...
200	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	67	Telnet Data ...
201	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	67	[TCP Keep-Alive] 23 - 35894 [PSH, ACK] Seq=530249794 Ack=1027...
202	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	67	Telnet Data ...
203	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	67	[TCP Keep-Alive] 35894 - 23 [PSH, ACK] Seq=1027871806 Ack=530...
204	2025-08-27 04:08	10.9.0.6	10.9.0.5	TELNET	67	[TCP Keep-Alive] 35894 - 23 [PSH, ACK] Seq=530249795 Ack=1027...
205	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	66	35894 - 23 [ACK] Seq=1027871801 Ack=530249796 Win=64128 Len=0...
206	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	66	[TCP Keep-Alive] 35894 - 23 [ACK] Seq=1027871801 Ack=530249796 Win=64128 Len=0...
207	2025-08-27 04:08	10.9.0.6	10.9.0.5	TCP	66	[TCP Keep-Alive ACK] 35894 - 23 [ACK] Seq=1027871801 Ack=5302...

Explanation:

This demonstrates a successful MITM attack. First, the ARP caches of both A and B are poisoned. Next, IP forwarding is enabled, allowing a Telnet connection to be established between A and B. After the connection is set up, we refresh the ARP caches by executing Task 11A and Task 2. Following this, when M initiates the MITM

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

attack, all characters typed are altered to “Z,” and everything entered on A’s terminal becomes visible in M’s output.

Task 5: MITM Attack on Netcat using ARP Cache Poisoning

Take a screenshot of the caches of A and B to show they have been poisoned

Take a screenshot of the established netcat connection

Take a screenshot of the terminals on A and B showing that the attack worked

Host A:

The screenshot shows three terminal windows for Host A. The first window shows the output of the command \$arp, listing two entries: M-10.9.0.105.net-10.9.0 and B-10.9.0.6.net-10.9.0.0. The second window shows the command \$nc 10.9.0.6 9090, followed by the name Pranav. The third window is empty.

```
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed-HostA:PES1UG23CS433:PranavHemanth:/
$arp
Address      HWtype  HWaddress          Flags Mask   Iface
M-10.9.0.105.net-10.9.0  ether   02:42:0a:09:00:69  C      eth0
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:06  C      eth0
seed-HostA:PES1UG23CS433:PranavHemanth:/
$>nc 10.9.0.6 9090
Pranav
```

Host B:

The screenshot shows three terminal windows for Host B. The first window shows the output of the command \$arp, listing two entries: M-10.9.0.105.net-10.9.0 and A-10.9.0.5.net-10.9.0.0. The second window shows the command \$nc -l 9090, followed by the string AAAAAAA. The third window is empty.

```
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed-HostB:PES1UG23CS433:PranavHemanth:/
$arp
Address      HWtype  HWaddress          Flags Mask   Iface
M-10.9.0.105.net-10.9.0  ether   02:42:0a:09:00:69  C      eth0
A-10.9.0.5.net-10.9.0.0  ether   02:42:0a:09:00:05  C      eth0
seed-HostB:PES1UG23CS433:PranavHemanth:/
$>nc -l 9090
AAAAAAA
```

Aug -Dec 2025 Assignment SUBMISSION_UE23CS343AB6

Attacker M:

```
seed@VM: ~
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task11A.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hwlen    = None
plen     = None
op       = who-has
hwsrc   = 02:42:0a:09:00:69
psrc    = 10.9.0.6
hwdst   = 02:42:0a:09:00:05
pdst    = 10.9.0.5

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task2.py
.

Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task11A.py
###[ Ethernet ]###

.
Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 task2.py
.

Sent 1 packets.
seed-attacker-M:PES1UG23CS433:PranavHemanth:/volumes
$>python3 mitm1.py
LAUNCHING MITM ATTACK.....
*** b'Pranav\n', length: 7
.

Sent 1 packets.
.

Sent 1 packets.
```

Explanation:

This demonstrates a successful MITM attack on Netcat. First, the ARP caches of both A and B are poisoned. Then, IP forwarding is enabled to establish a Netcat connection between A and B. Once the connection is active, IP forwarding remains on. After refreshing the ARP caches by executing Task 11A and Task 2, M launches the MITM attack. When 6 characters, “Pranav,” are typed on A’s terminal and sent, the message is altered to “AAAAAA,” confirming the success of the attack.