**ACS 54500 Cryptography and Network Security - Lab 3**

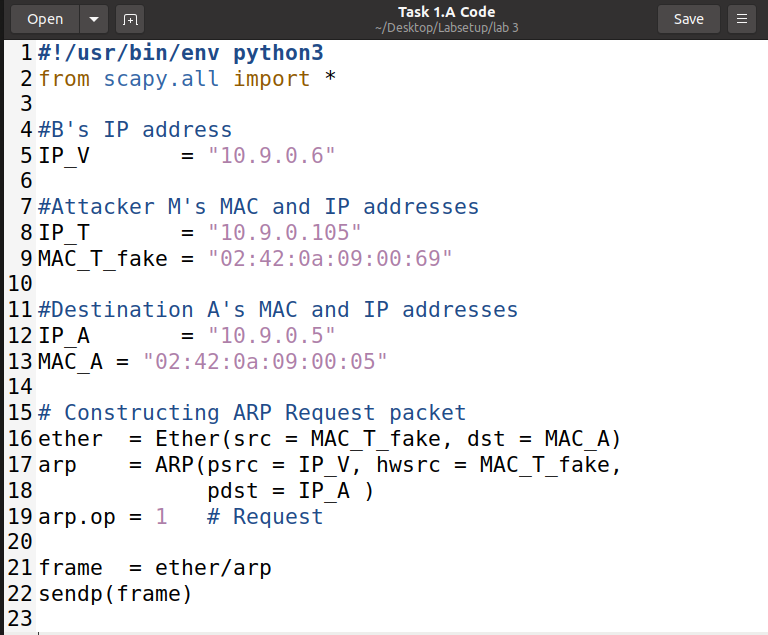
In this lab, we implement ARP cache poisoning attacks.

**Task 1**

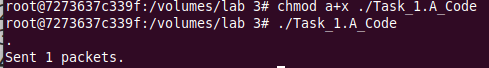
The aim of this task is to use packet spoofing to implement the ARP cache poisoning attack part of a Man-In-The-Middle Attack. There are three sub tasks under this task.

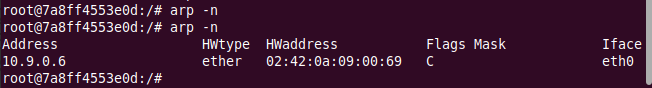
**Task 1.A**

The below code shows how to construct an ARP request packet to map host B’s IP address to the attacker machine M’s MAC address.



Before the attack is sent, A’s ARP cache is seen to be empty. After we run the code and send the request packet, we can see that there is an entry with B’s IP address mapped to M’s MAC address, as shown below:

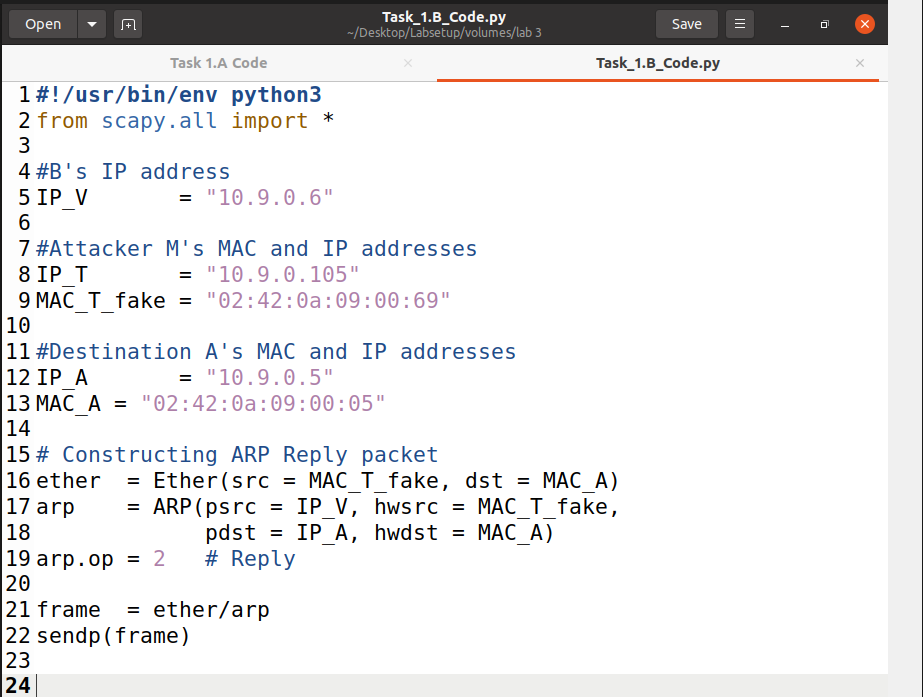




Thus, the attack is successful.

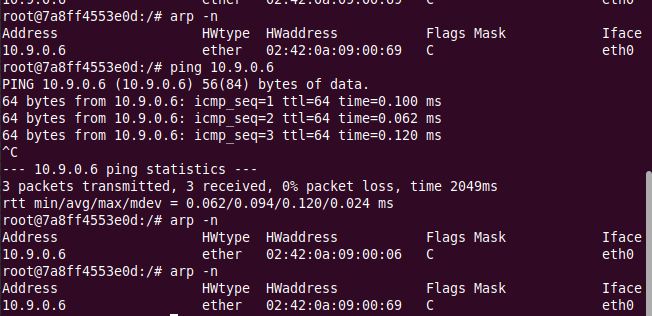
**Task 1.B**

In this sub task, we need to write code to construct an ARP reply packet to map B’s IP address to M’s MAC address. This packet is then sent from M to A. The code is shown below -

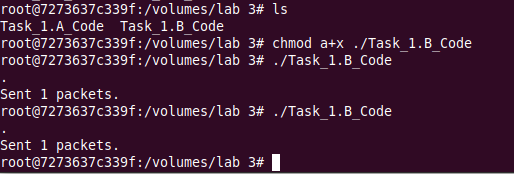


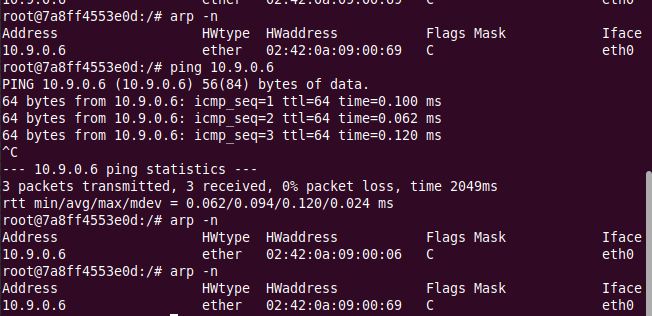
**Scenario – 1**

We first ping B from A to ensure B’s IP entry is already present in A’s ARP cache and then run the above code on M. This is shown as below –



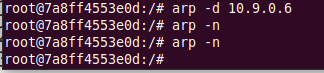
This is how A’s ARP cache looks after running the code. We can see that the previous entry with B’s IP and MAC address has been modified to B’s IP and M’s MAC address, showing a successful attack.





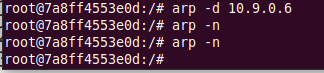
**Scenario – 2**

In this scenario, we need to run the same attack when B’s IP is not already present in A’s ARP cache. To do this, we delete the entry by running ‘arp -d 10.9.0.6’ as shown below and then run the program –



This is how A’s ARP cache looks after running the code. We can see that there is no change in the cache, proving that the attack has failed. Text

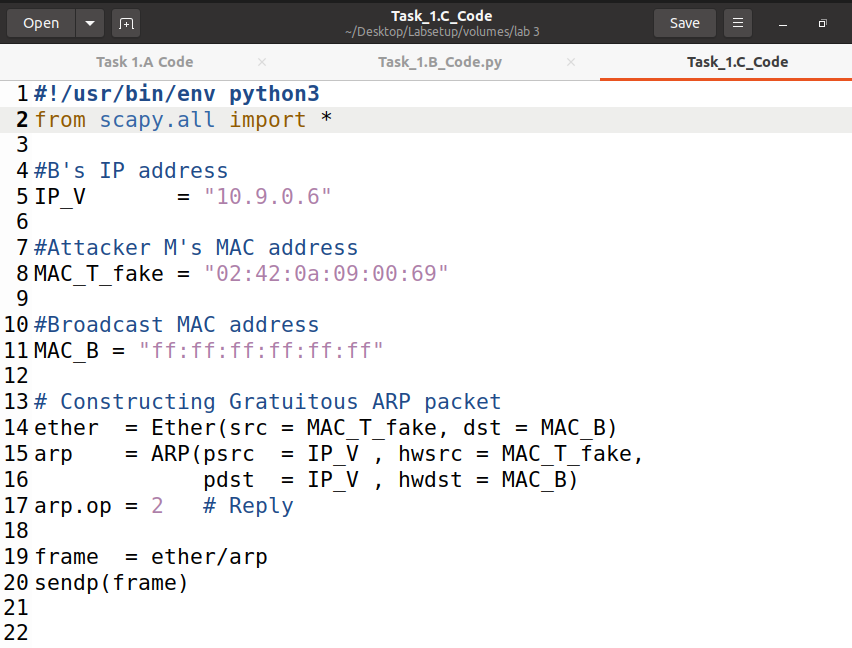
Description automatically generated



**Task 1.3**

The below code shows the code to send an ARP gratuitous attack. Here we ensure two things:

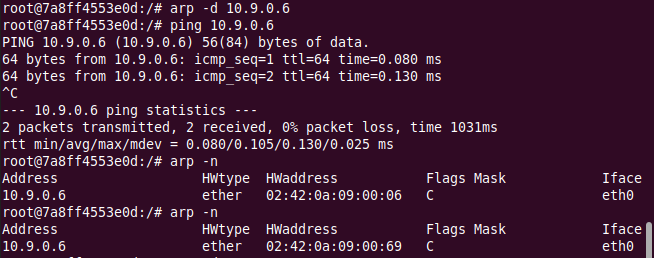
1. The source and destination IP addresses are the same, and they are the IP address of the host issuing the gratuitous ARP, i.e., host B.
2. The destination MAC addresses in both ARP header and Ethernet header are the broadcast MAC address (ff:ff:ff:ff:ff:ff).



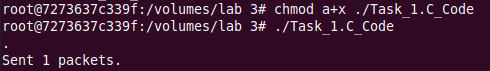
We run the code in the same two scenarios as Task 1.B.

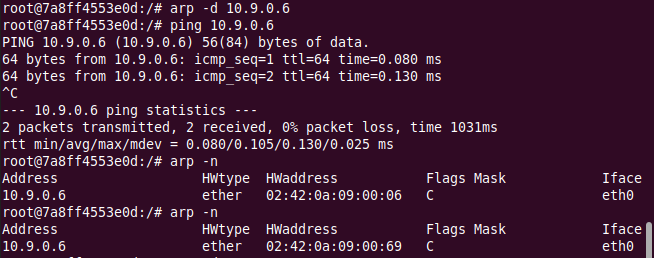
**Scenario – 1**

This is how A’s ARP cache looks like before the attack is run –



After we run the code, this entry is modified to map B’s IP address to M’s MAC address, as shown below –





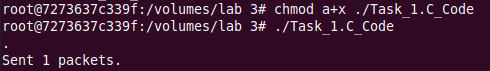
This shows that the attack was successful.

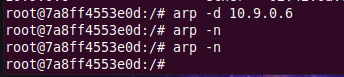
**Scenario – 2**

We run the same attack after deleting B’s entry from A’s ARP cache. Results are as shown below –

Text

Description automatically generated

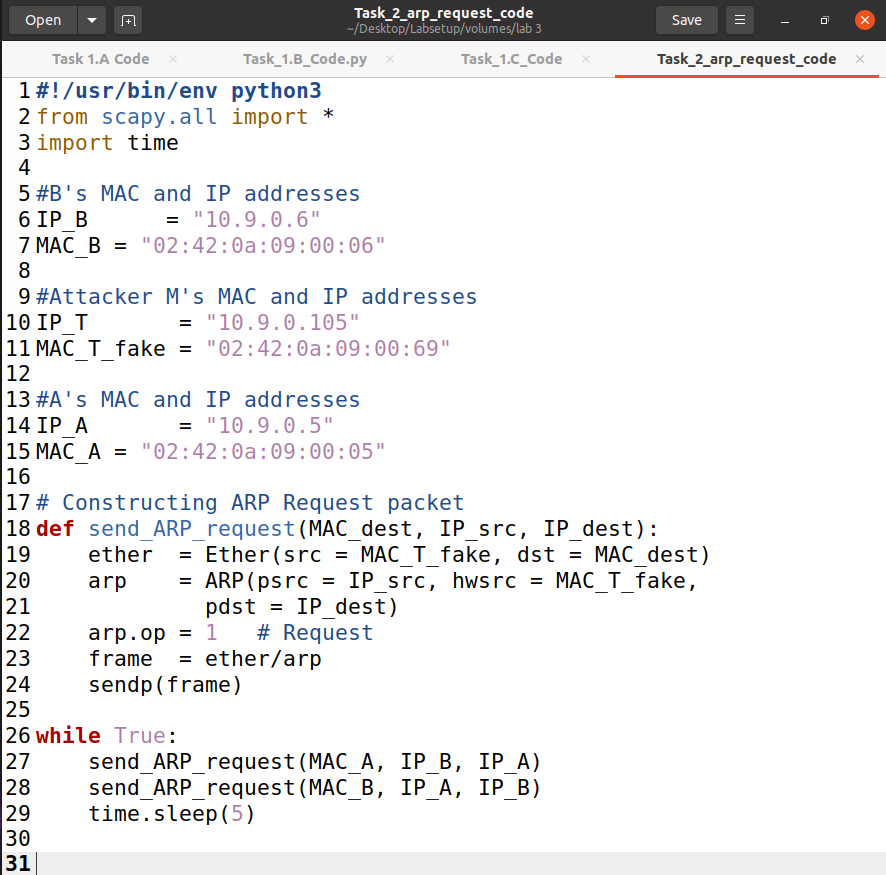


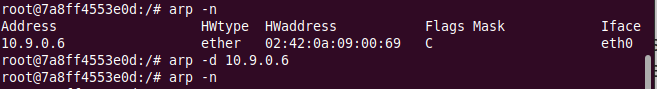


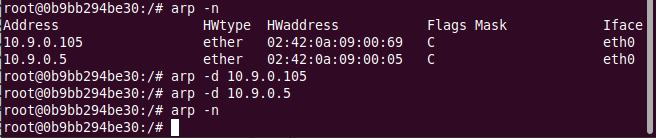
As we can see, there is no change – thus the attack failed.

**Task 2**

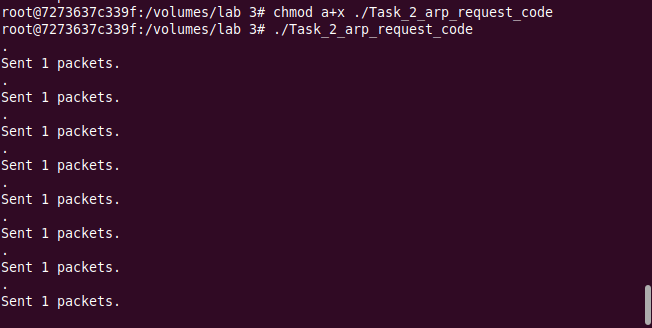
In this task we have to make sure that both B’s IP address and A’s IP address are each mapped to M’s MAC address. This is done by running a code that sends out a request packet with this information to both A and B on a loop every 5 seconds. This code is shown below –



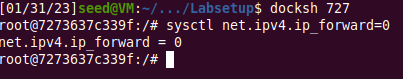




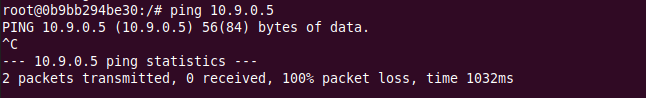
**Step – 1:** We run the above code to start the attack loop as shown below –



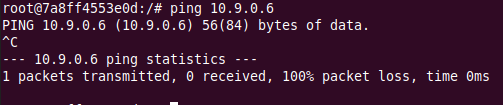
**Step – 2:** To test and see if this works, we ping both hosts A and B from each other and record live capture through Wireshark. Before doing this, we make sure that IPv4 forwarding is turned off on attacker M as shown –



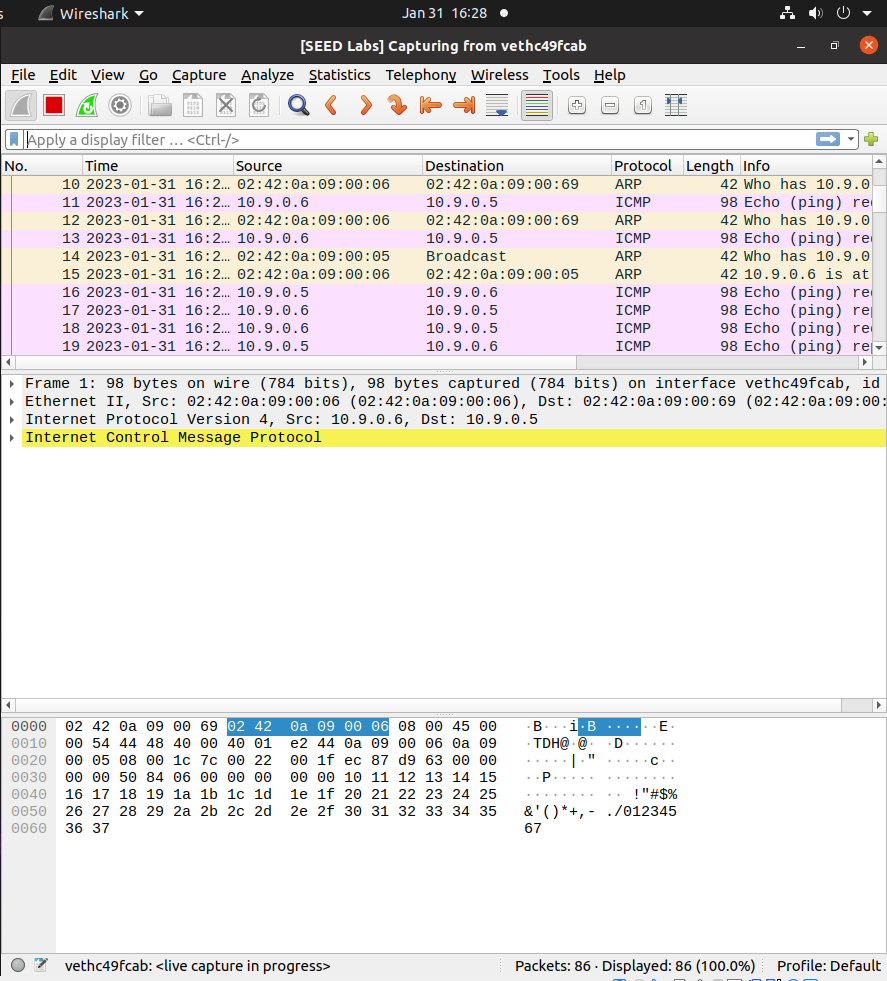
First, we ping A from B and look at the results -



Next, we ping B from A and look at the results -



The below screenshot shows the results of live capture in Wireshark -



**Step – 3:** We now turn on IPv4 forwarding on M to ensure that it will forward the packets between A and B.



We then repeat Step 2.

Text

Description automatically generated

Text

Description automatically generated

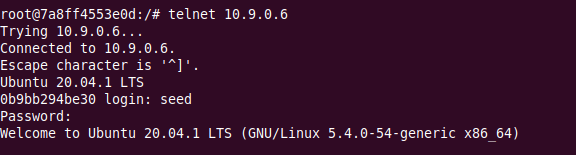
Graphical user interface, text, application

Description automatically generated

We can see that now packets are being redirected.

**Step – 4:** We create a telnet connection between A and B, making sure that IPv4 forwarding is turned on to successfully create this. Assume that A is the Telnet client and B is the Telnet server. We now turn off IPv4 forwarding –

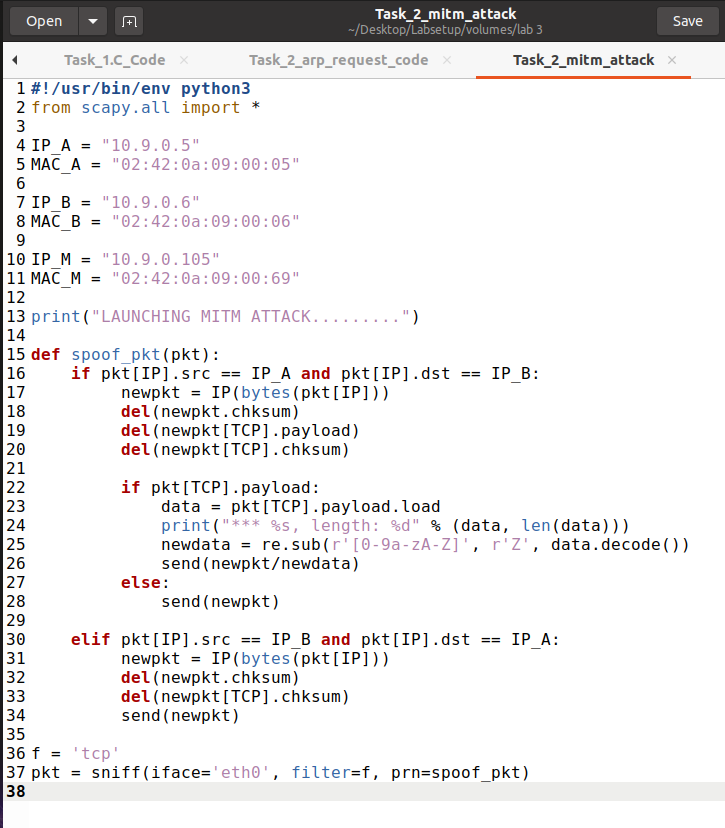




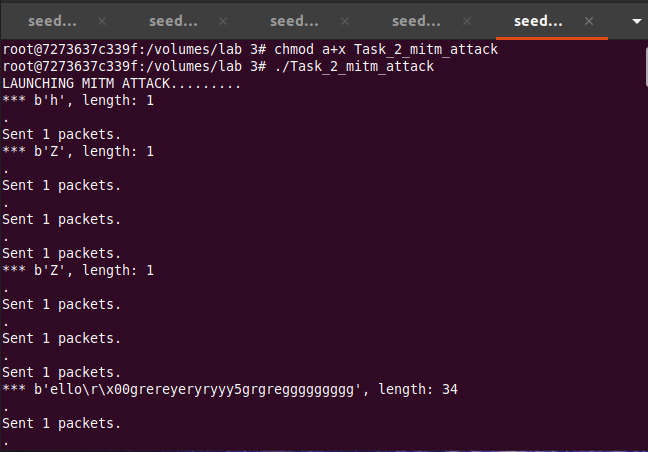
It is observed that any characters typed do not show up on the telnet window. This is because the server has been disconnected.



The following sniff-and-spoof program is now run on attacker M to capture all the TCP packets. This code substitutes all characters entered in the Telnet window with a ‘Z’ using regular expressions (regex).



As shown below, any characters typed will be replaced –

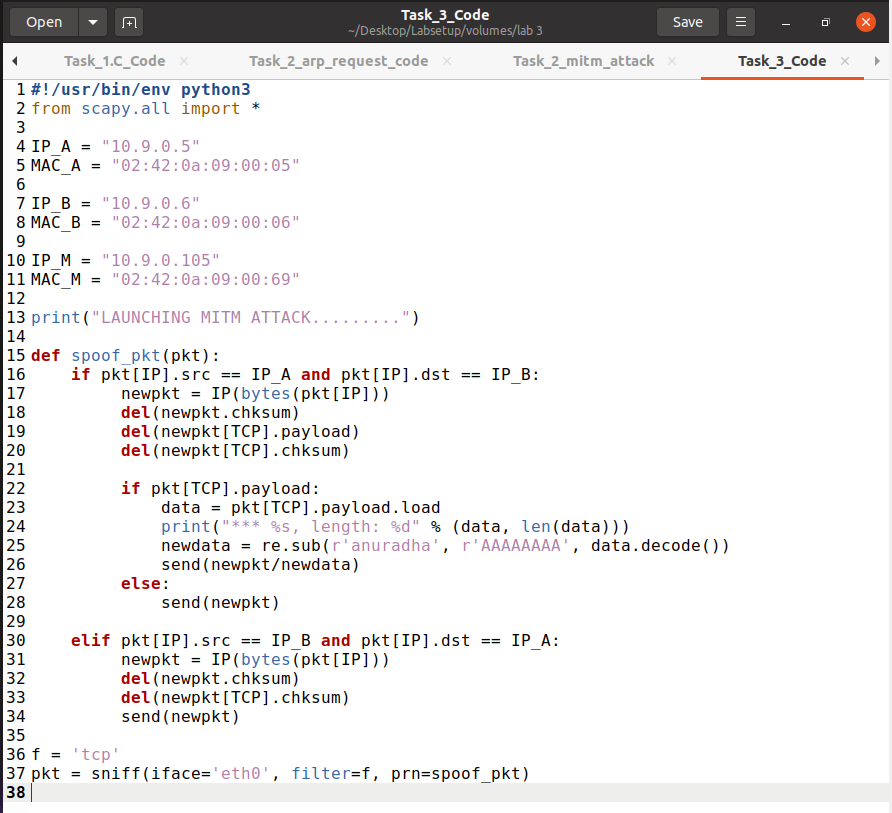




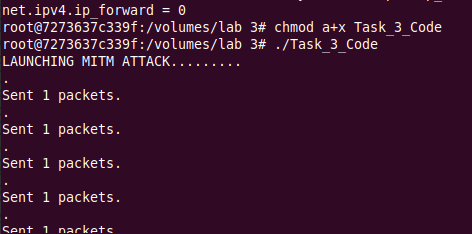
This shows that the MITM attack is running successfully. The attacker M keeps intercepting packets sent from A to B and replaces the contents with desired replacements.

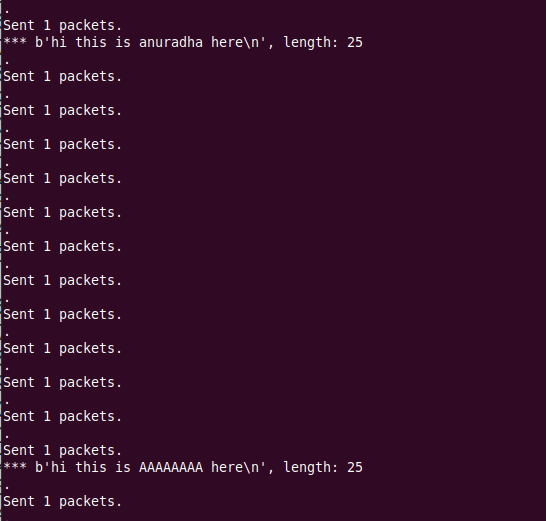
**Task 3**

Here we implement a similar situation as Task 2, except that we use netcat as the mode of communication between A and B instead of using telnet. This task requires us to replace all the letters of our first names with an equal number of A’s. The code is shown below –



We first run this code on machine M.





Now we use netcat to send a message from host A to B. I chose to send the message ‘hi this is anuradha here’.



On host B, we can see that the letters of my name have been replaced with a series of A’s –



Thus, a MITM attack has been successfully implemented on netcat using ARP cache poisoning.