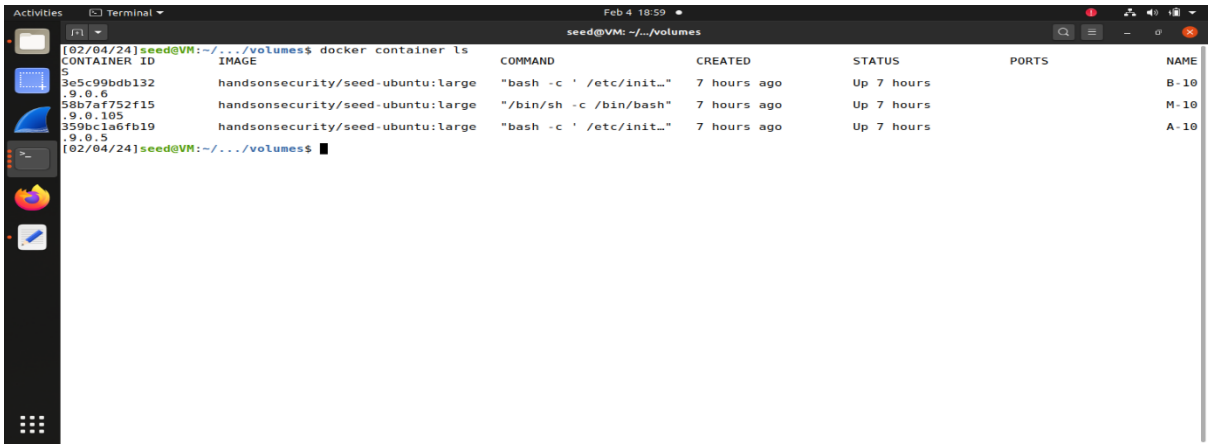
CSC 5290 : Cyber Security Practices
Lab2
Winter 2023
Student name: Kanchan Chopde

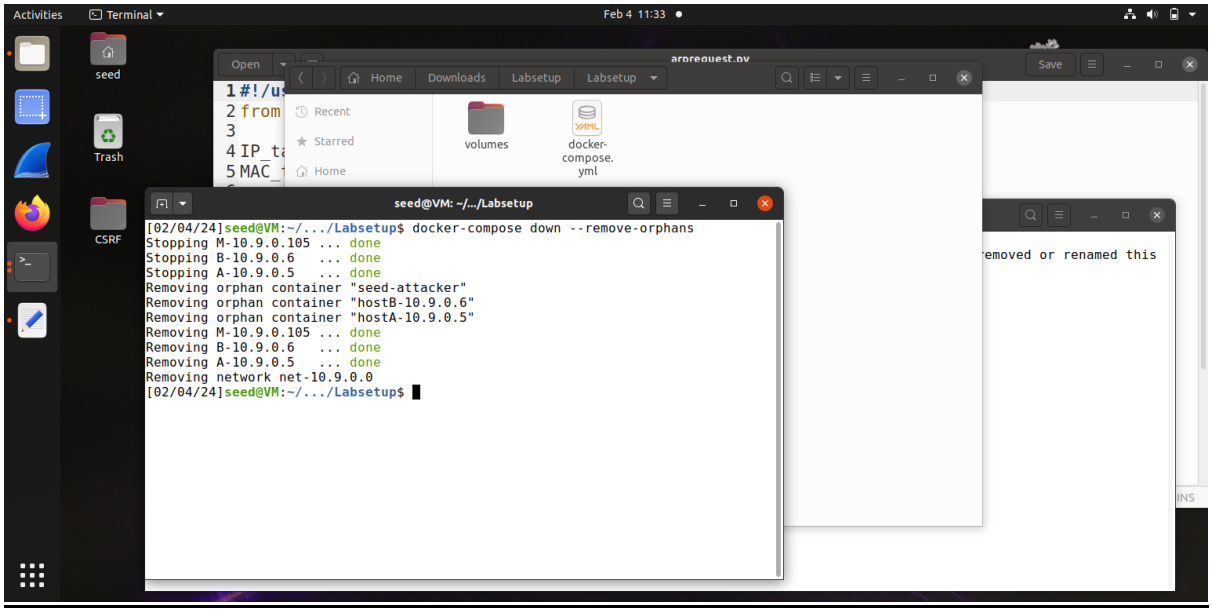
ARP Cache Poisoning Lab Attack

Following screenshot shows information on docker containers with their ids:
A-10.9.0.5 35…19
B-10.9.0.6 58…15

M-10.9.0.105 3e…32



Following 2 screenshots shows docker containers creating and running.
Also interesting observation : need to remove orphan containers used in previous labs for that used command: docker -compose down –remove-orphans



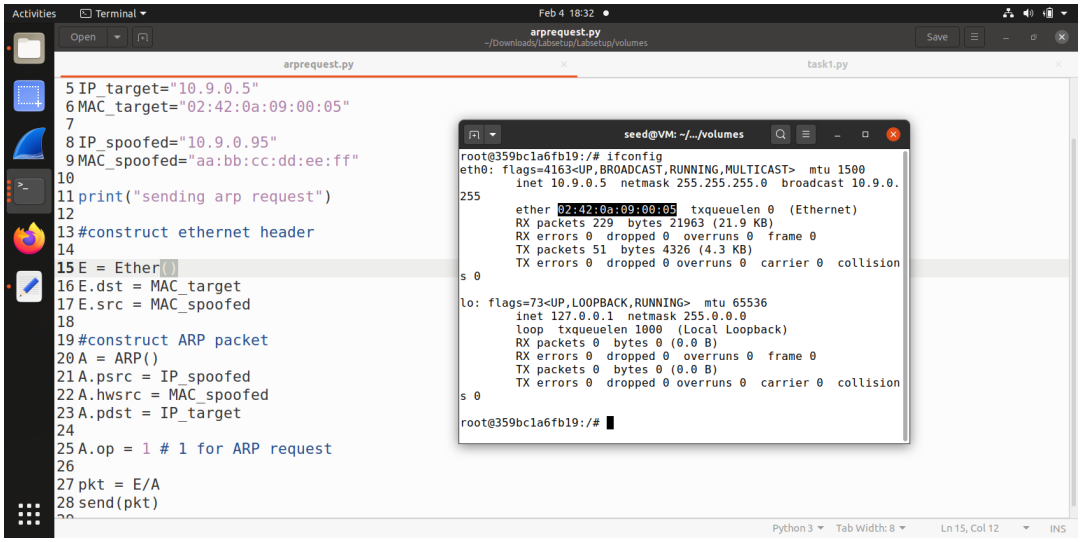
Task 1: ARP Cache Poisoning

Task 1.A (using ARP request)

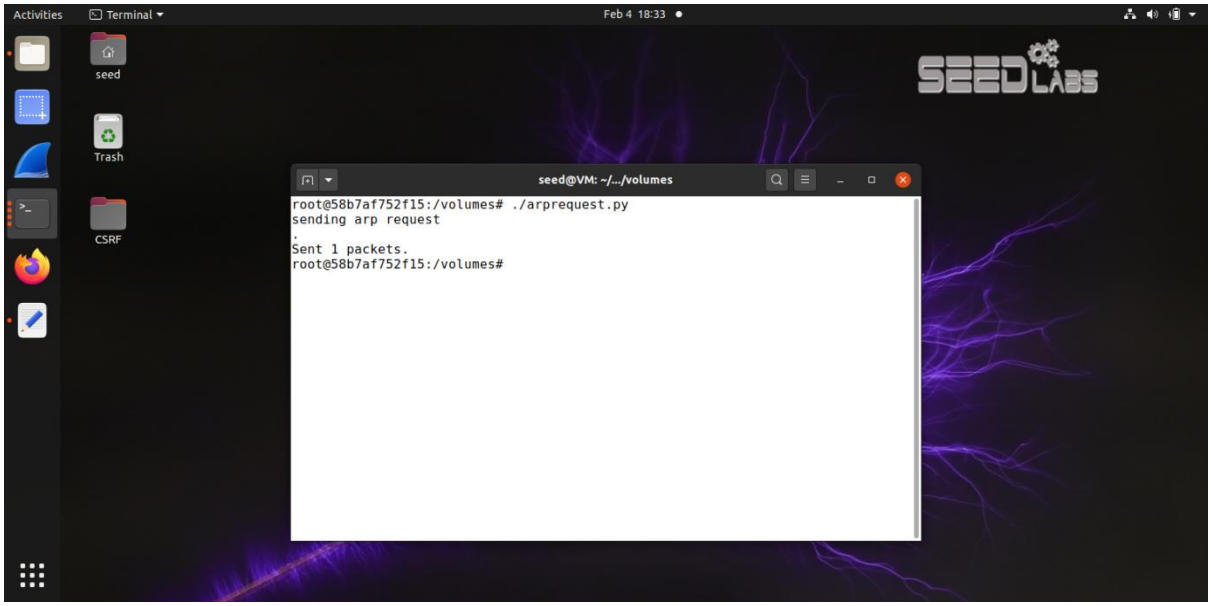
Following screenshots shows sending arp spoofed packet program.
Here we have added IP_target as HostA's IP address, MAC_target as HostA's MAC Address.We get it from running ifconfig command as shown in screenshot.

We construct ethernet object and assign its destination as HostA's MAC address and ethernet source as the spoofed MAC address i.e. aa:bb:cc:dd:ee (fake mac address)

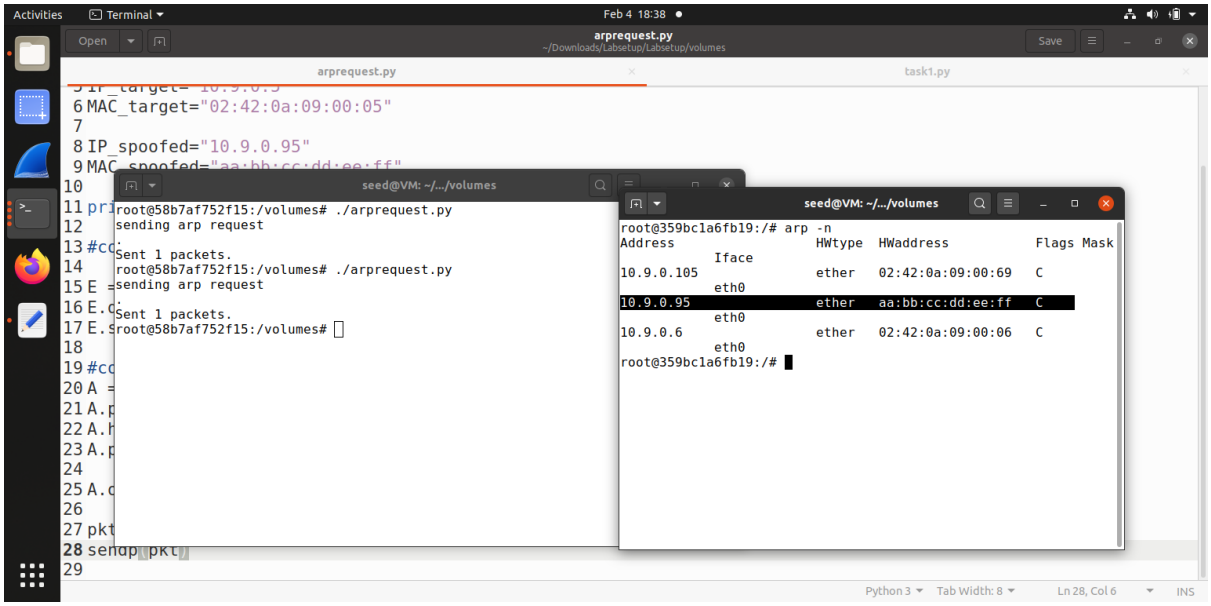
We construct ARP object and assign source ip as spoofed IP address, its mac address as spoofed mac address and its IP filed as target Ip i.e. HostA's ip to which want to send request.



Now we execute arprequest.py program from HostM's container as seen in below screenshot.



After running python script ,we check on HostA's container using arp -n command.
we can clearly see Address 10.9.0.95 i.e. fake IP address and HWaddress i.e. fake MAC address which we executed from HostM.

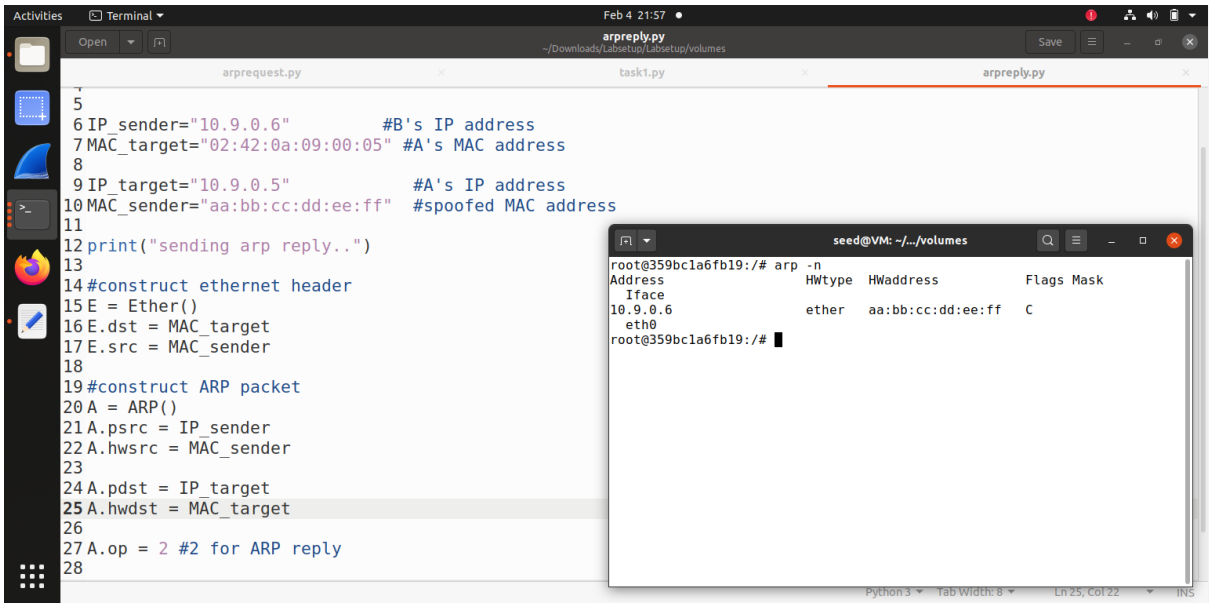


Task 1.B (using ARP reply)

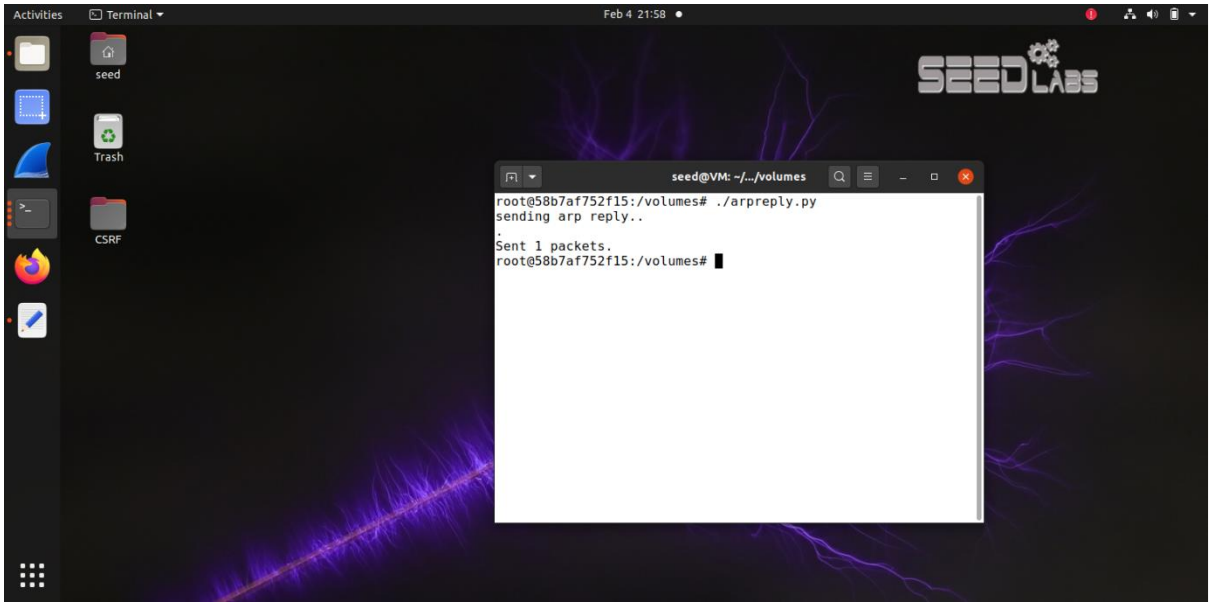
For ARP reply:
a) Scenario 1: B's IP is already in A's cache.

We write program arpreply.py:

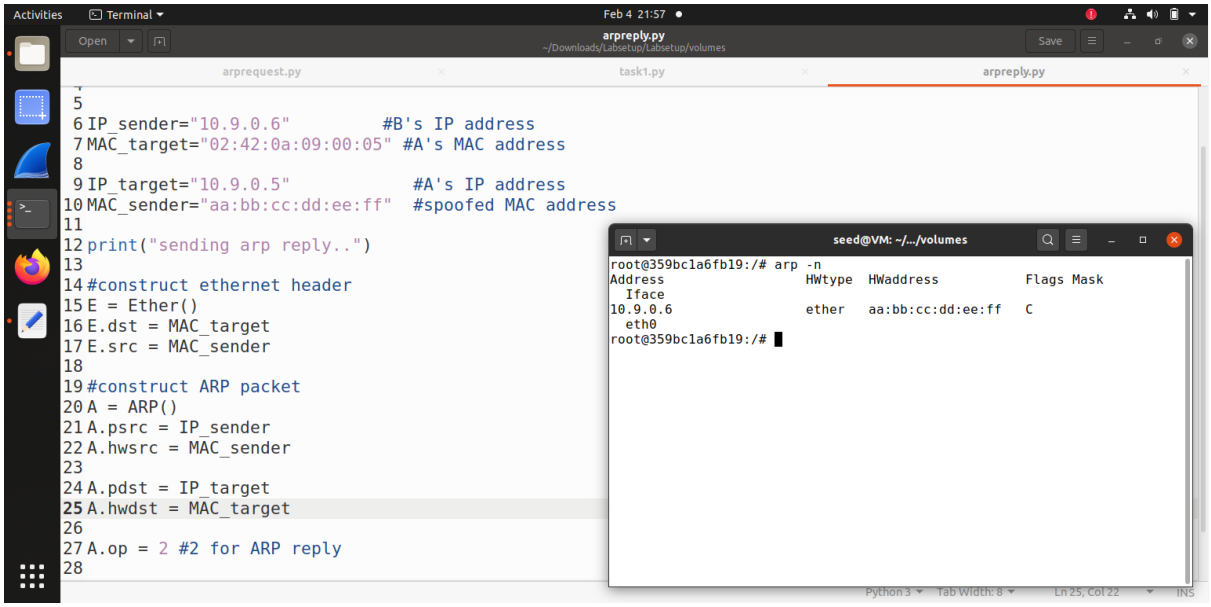
Here we make changes to add IP sender as Host B IP and MAC target as A. Also we give Target IP as A and Spoofed MAC addresss: aa:bb:cc:dd:ee:ff. We add operation code as 2 depicting ARP reply.



We run the program on HOST M.

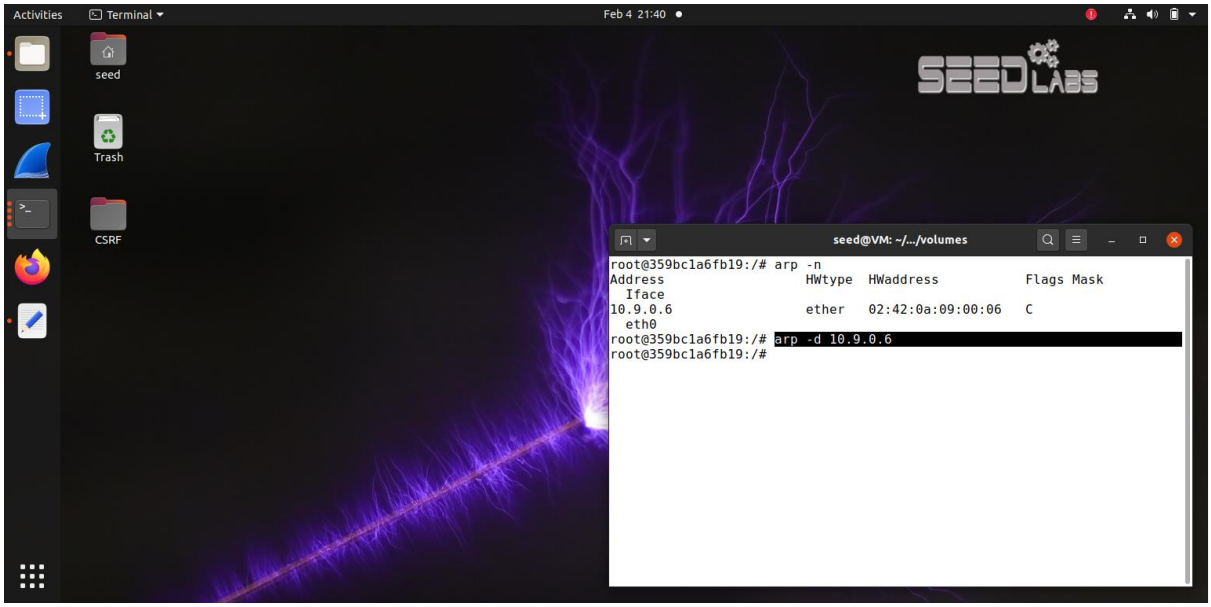


We check at HOST A with command arp -n
As host B was already present in cache , it updated with spoofed MAC address.

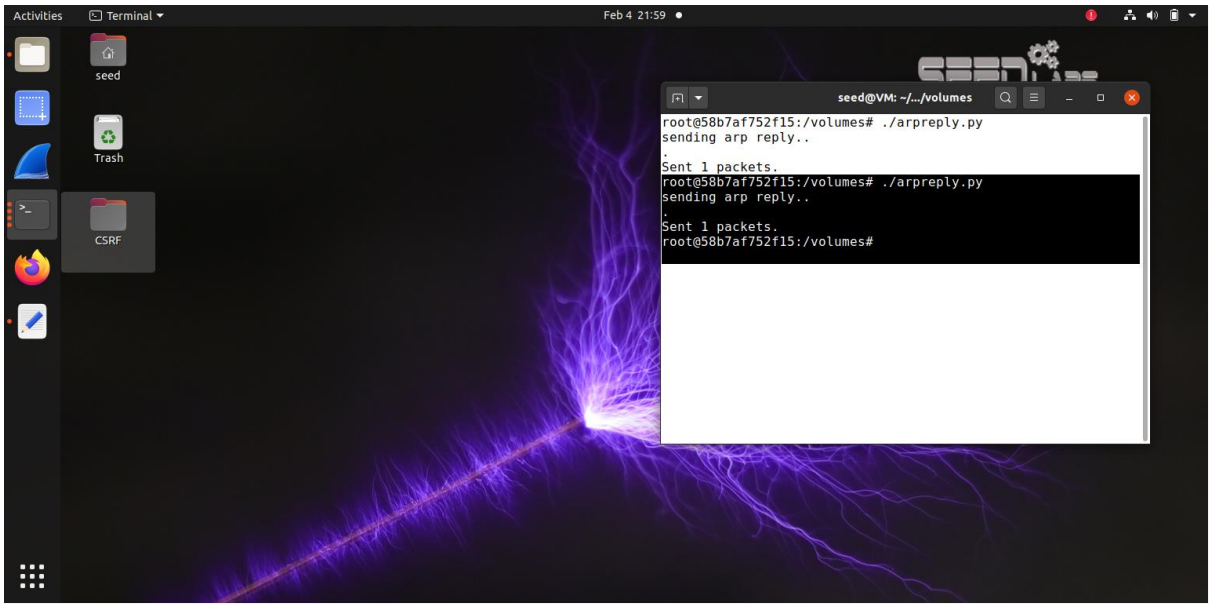


b) Scenario 2: B's IP is not in A's cache. You can use the command "`arp -d a.b.c.d`" to remove the ARP cache entry for the IP address `a.b.c.d`.

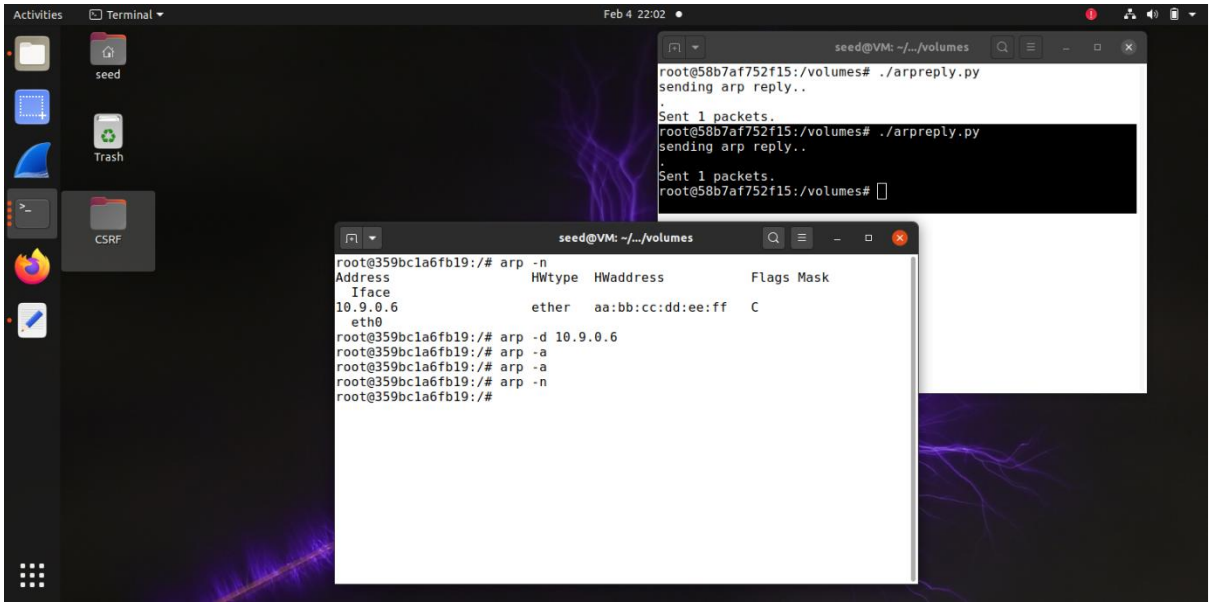
Now using command arp -d 10.9.0.6 we remove B's entry from cache.



We again run the arpreply.py program from HOST M.



There exists no communication between two hosts. Hence arp table is not updated.



Task 1.C (using ARP gratuitous message)

Scenario 1: B's IP is already in A's cache.

Write program gratitiousreply.py:

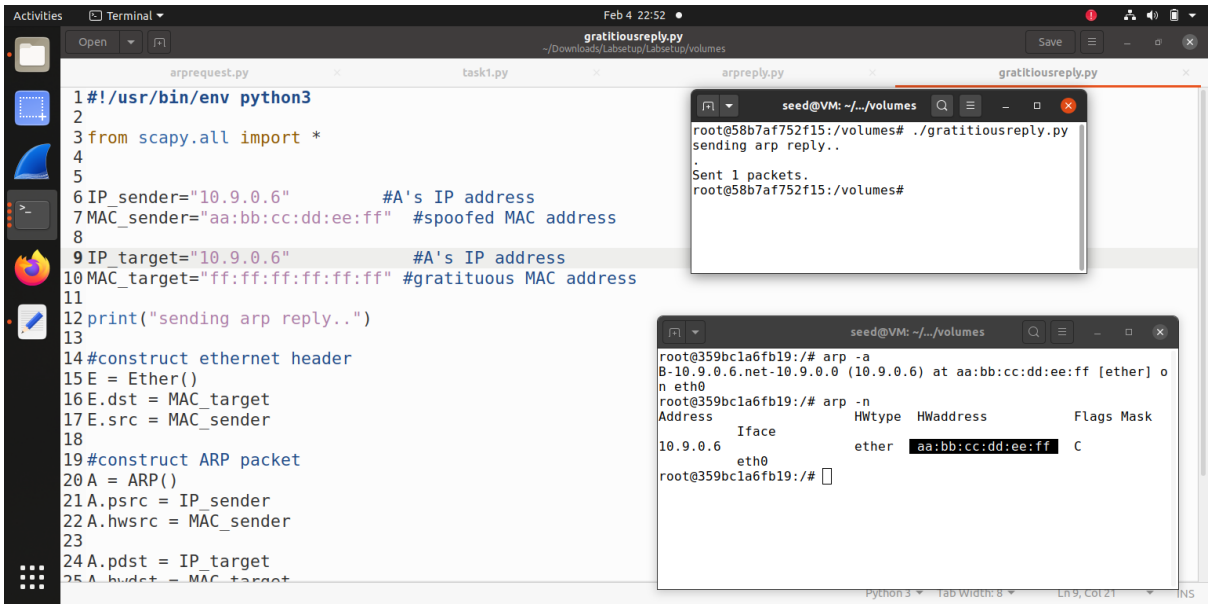
It has same sender and target IP i.e. A's IP address

MAC for sender is spoofed address: aa:bb:cc:dd:ee:ff

MAC for destination is gratitious address: ff:ff:ff:ff:ff:ff

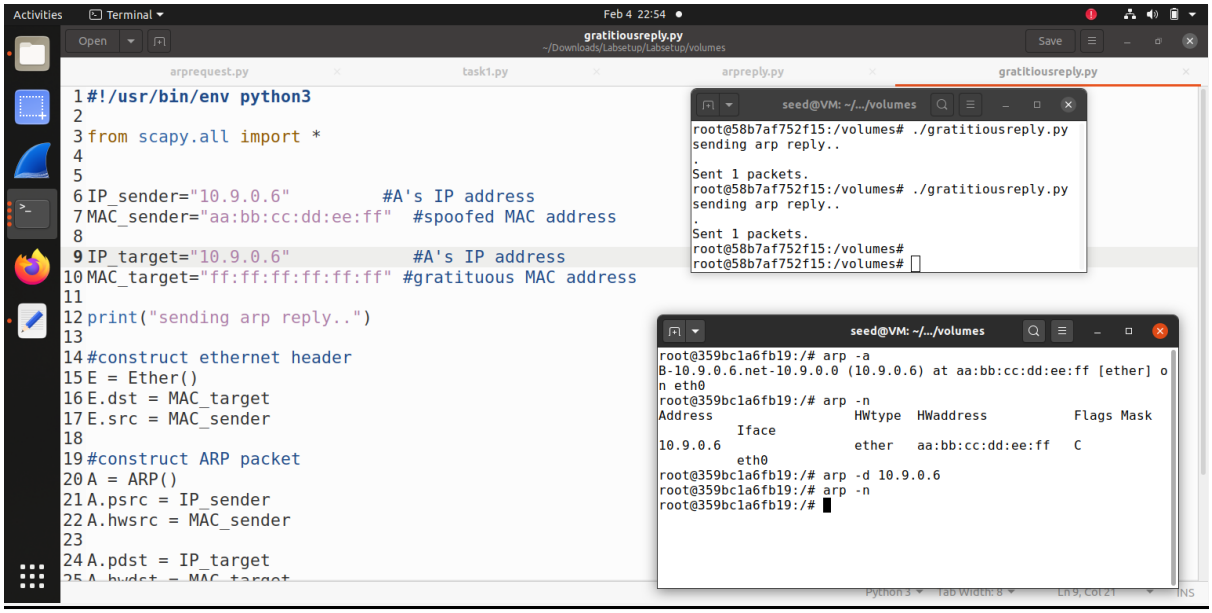
Set operation code to 2 i.e. reply

We run script from HOST M and look at HOST A – we can see spoofed MAC address is updated at HOST A.



Scenario 2: B's IP is not in A's cache.

No reply is observed at HOST A because communication with HOST B does no exists. Arp -n commands gives no table. AS we removed entry of host B 10.9.0.6 using arp -d command.Also there is no communication with host B hence arp table is not updated.



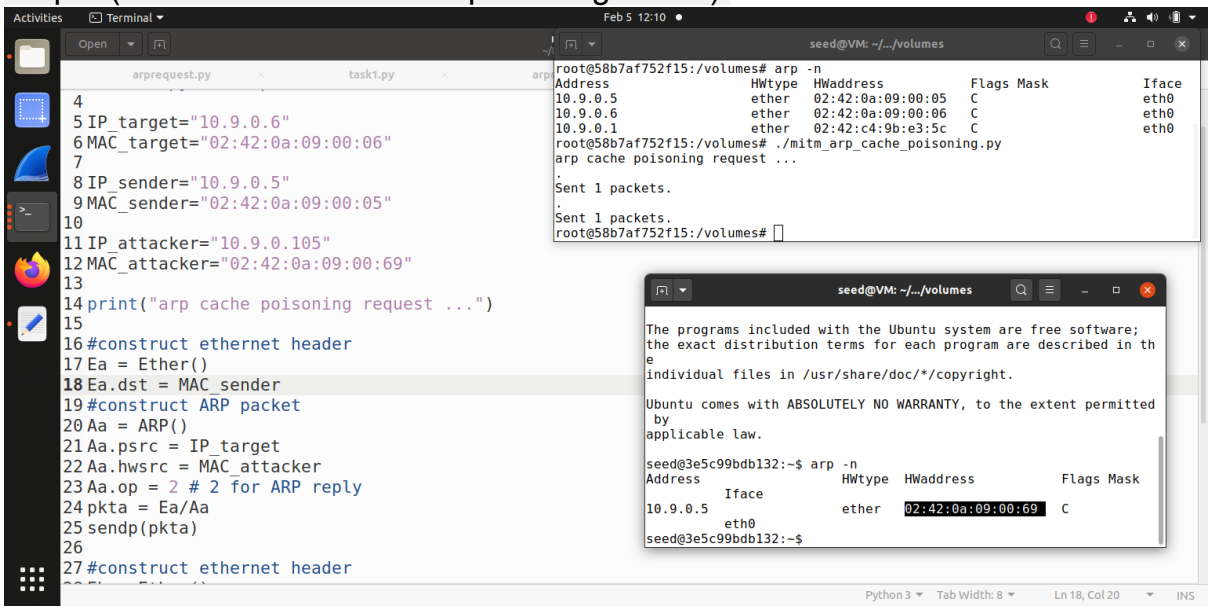
Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Below is program named mitm_arp_cache_poisoning.py.
Here we set sender IP and MAC address as HOST A, target IP and MAC address as HOST B and attacker IP and MAC address as HOST M.

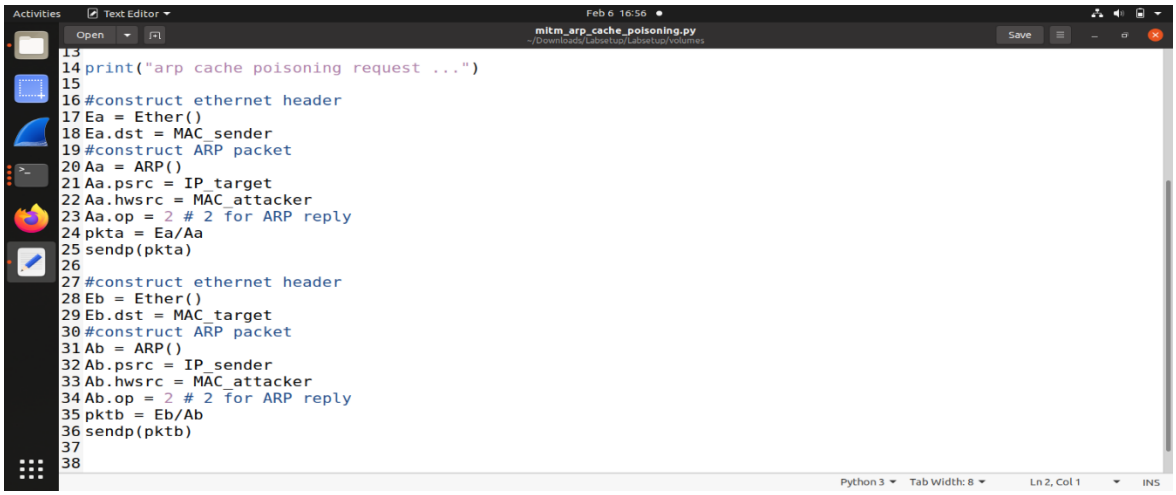
For communication reply we set the Ethernet destination as MAC address of HOST A, for ARP we add source IP as Host B and mac source as HOST M's mac address.

After running the program from HOST A to HOST B, we can see MAC address getting updated as attackers MAC for Host A. From Host A we do telnet 10.9.0.6 and entre login details.

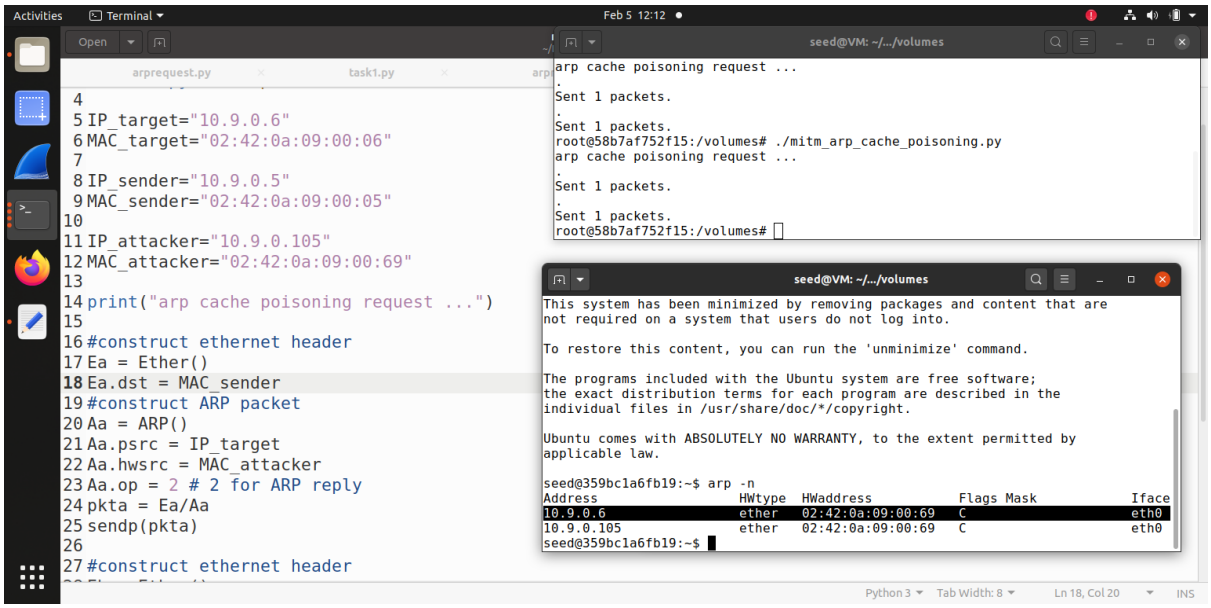
Step 1 (Launch the ARP cache poisoning attack).



Similarly, we construct another ethernet object and assign its destination as HOST B mac address, for ARP object we give HOST A's IP to src and mac source as Attacker ip i.e. HOST M.

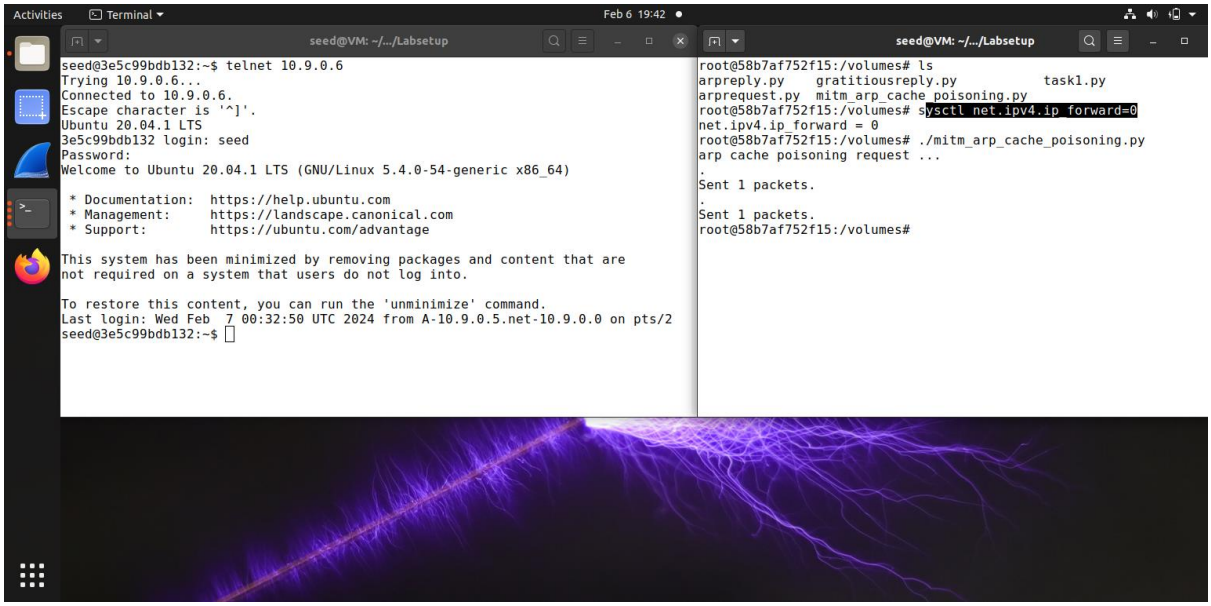


Output can be seen after doing telnet from Host B to A.
We can see host B's MAC address is updated to Host M's MAC address.



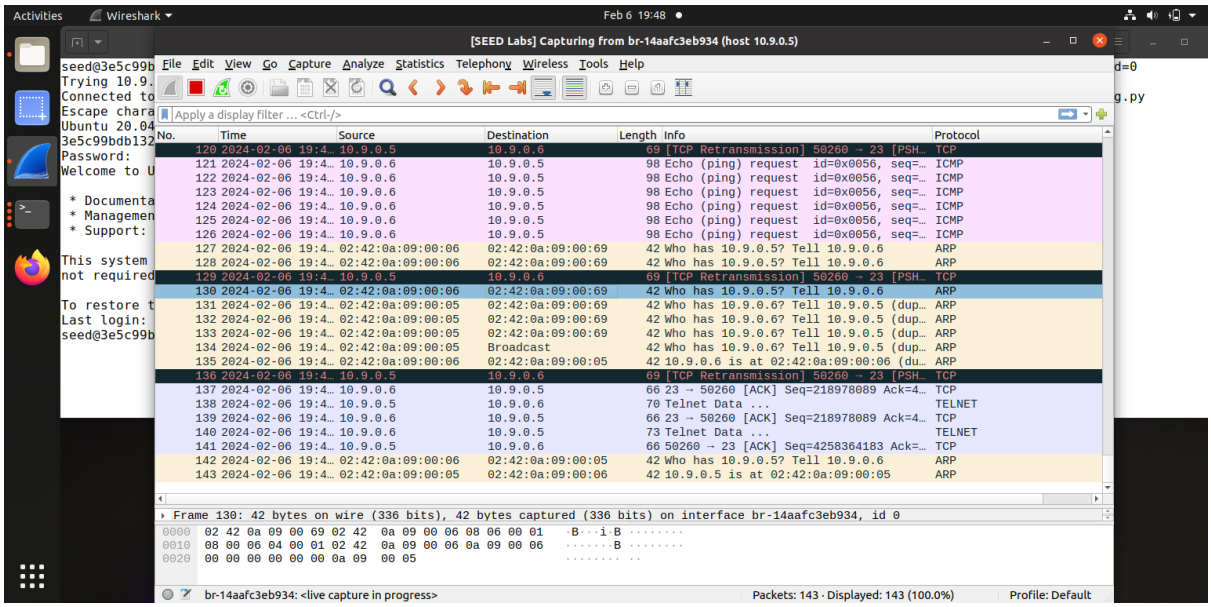
STEP 2: (TESTING)

In following screenshot you can see ip forwarding is disabled i.e. set to 0. And we make arp cache poisoning attack from HOST M.



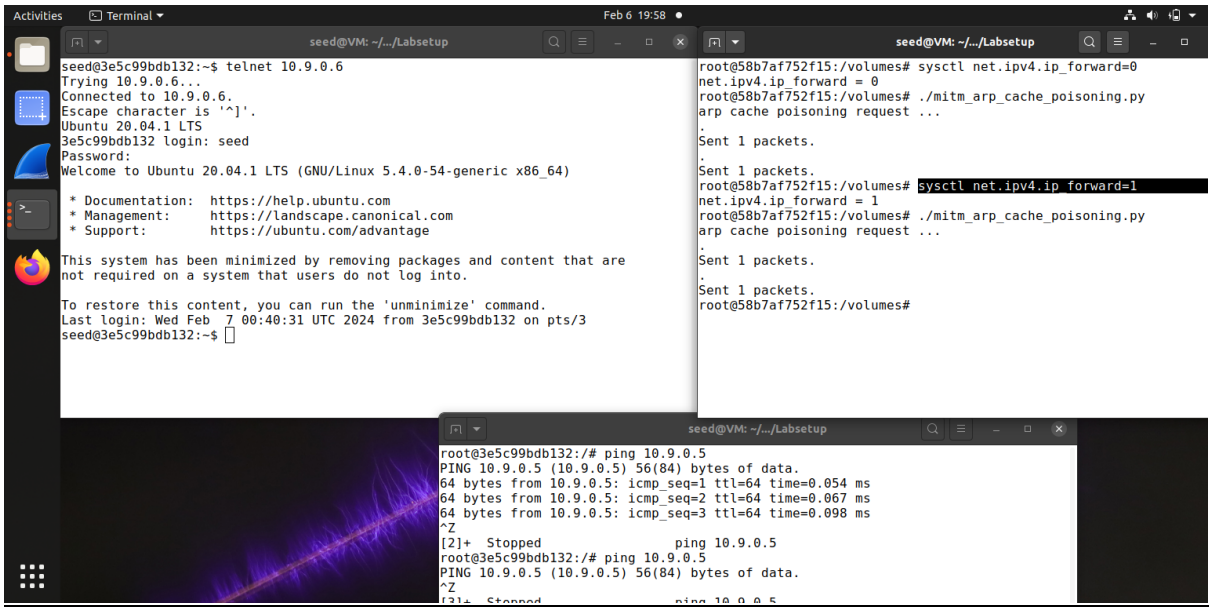
Below are the results:

We can see that HOST M is intercepting the traffic from HOST A to B .

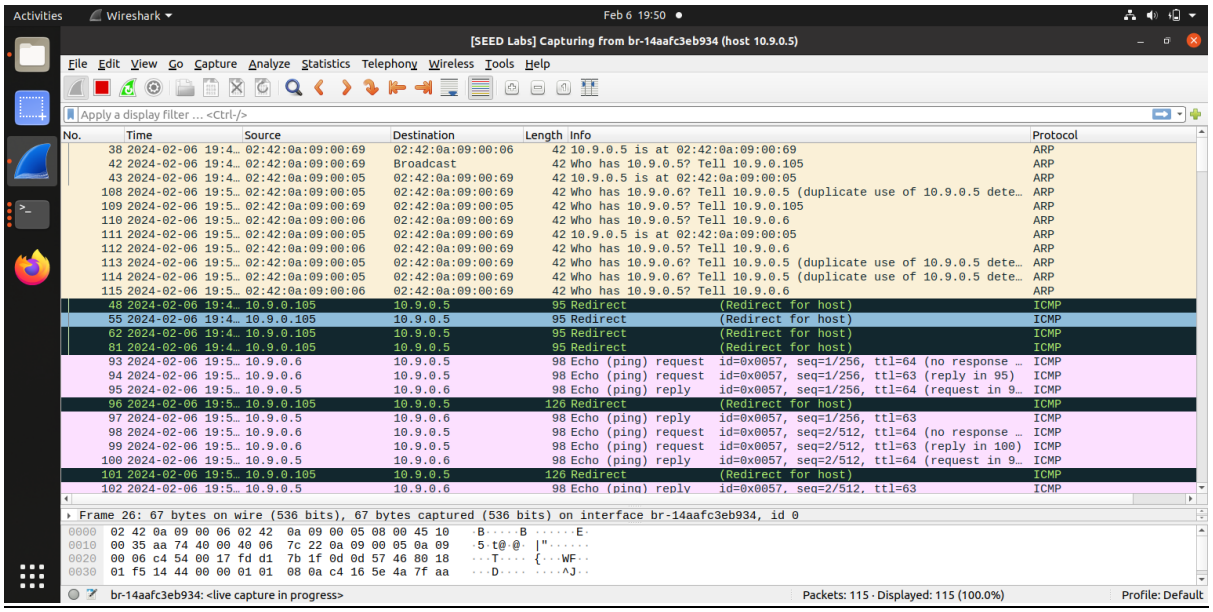


Step 3 (Turn on IP forwarding)

Now we enable the ip forwarding by setting it to 1.Run the program again.



Below are the results after enabling IP forwarding.
We can see that HOST M acts as router and routes packet between Host A and Host B.

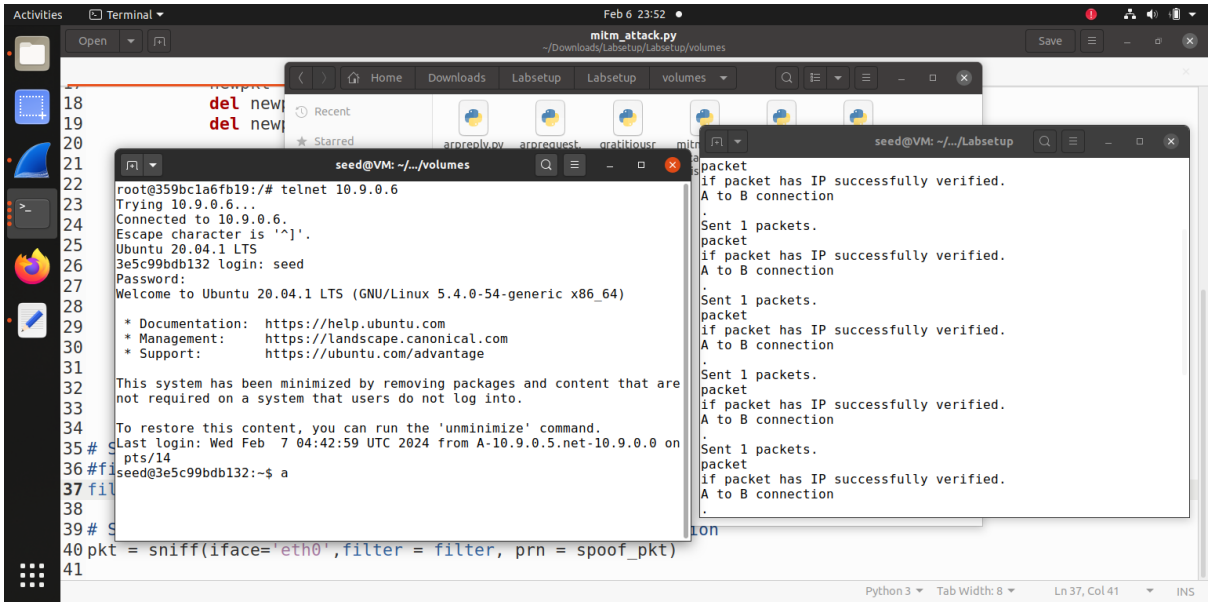


Step 4 (Launch the MITM attack).

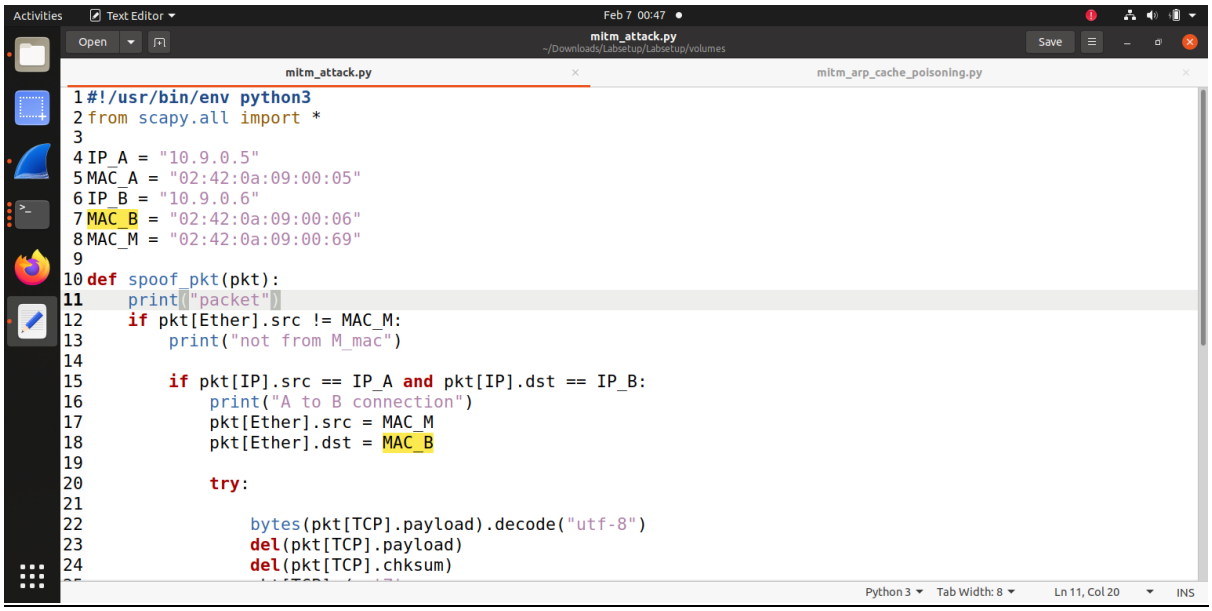
We first keep the IP forwarding on, so we can successfully create a Telnet connection between A to B. Once the connection is established, we turn off the IP forwarding using the following command. Please type something on A's Telnet window, and report your observation:
sysctl net.ipv4.ip_forward=0

Here we can see that after typing letter "a" on HOST A the program got executed and letter a is visible.

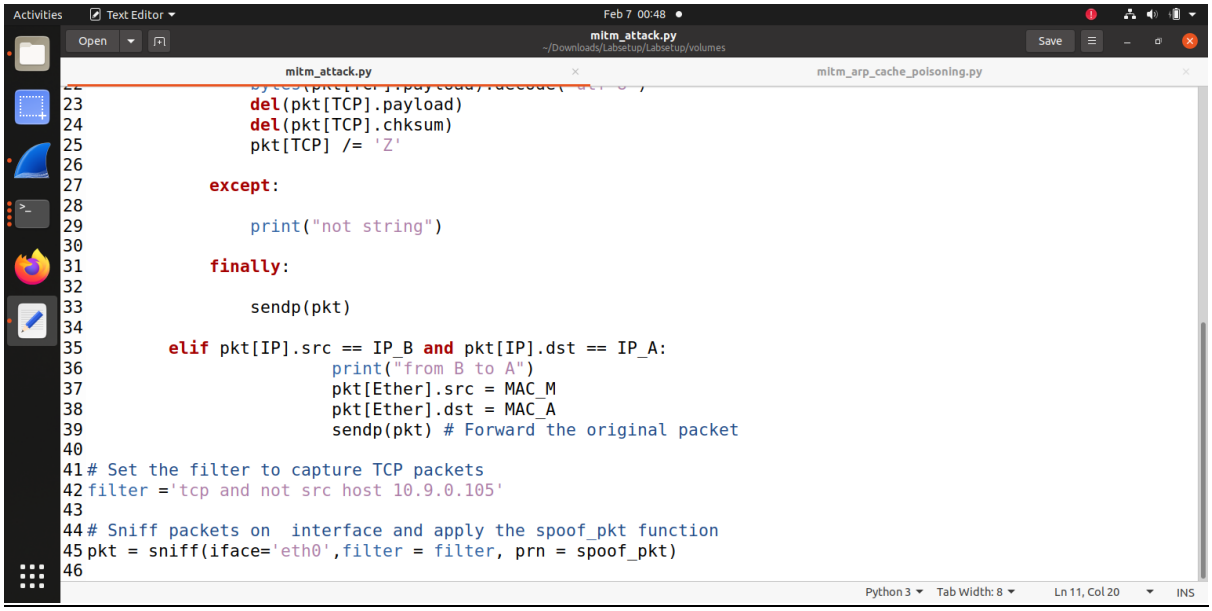
> If IP forwarded is not disabled after Telnet connection is made (i.e. input is sent in after login), the terminal works as usual and input is not replaced.



- In this program we initialize IP and MAC of all hosts A,B,M.
- We define spoof_pkt function : check if it is not coming from source MAC_M and then we check if source is HOST A and destination is B.
- If so, we set the ethernet source and destination as mac of m and mac of B respectively.
- We add try ,except and finally block.
  where we delete the payload and checksum and set pkt tp display character 'Z'.
- Finally we send the packet created by updating character.

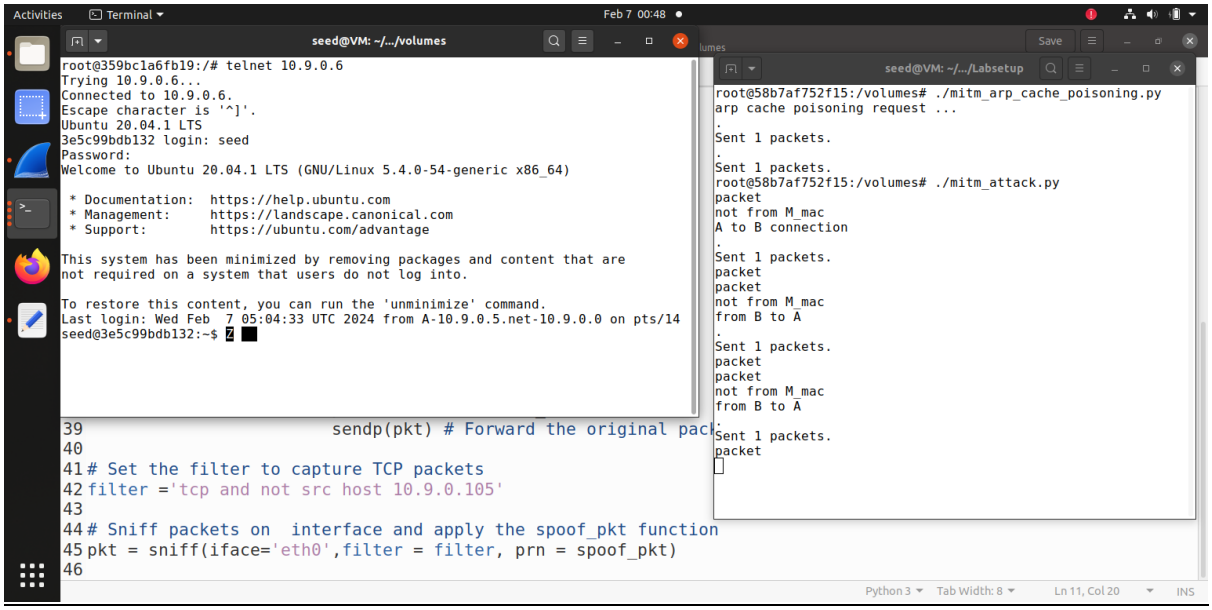


Here we check if src is B and destination is A , we print message, set ethernet src as MAC_M and dst to MAC_A and send the packet.



If IP forwarding is not enabled before Telnet connection is made, ARP cache is restored to the actual MAC addresses before the connection can be initiated.

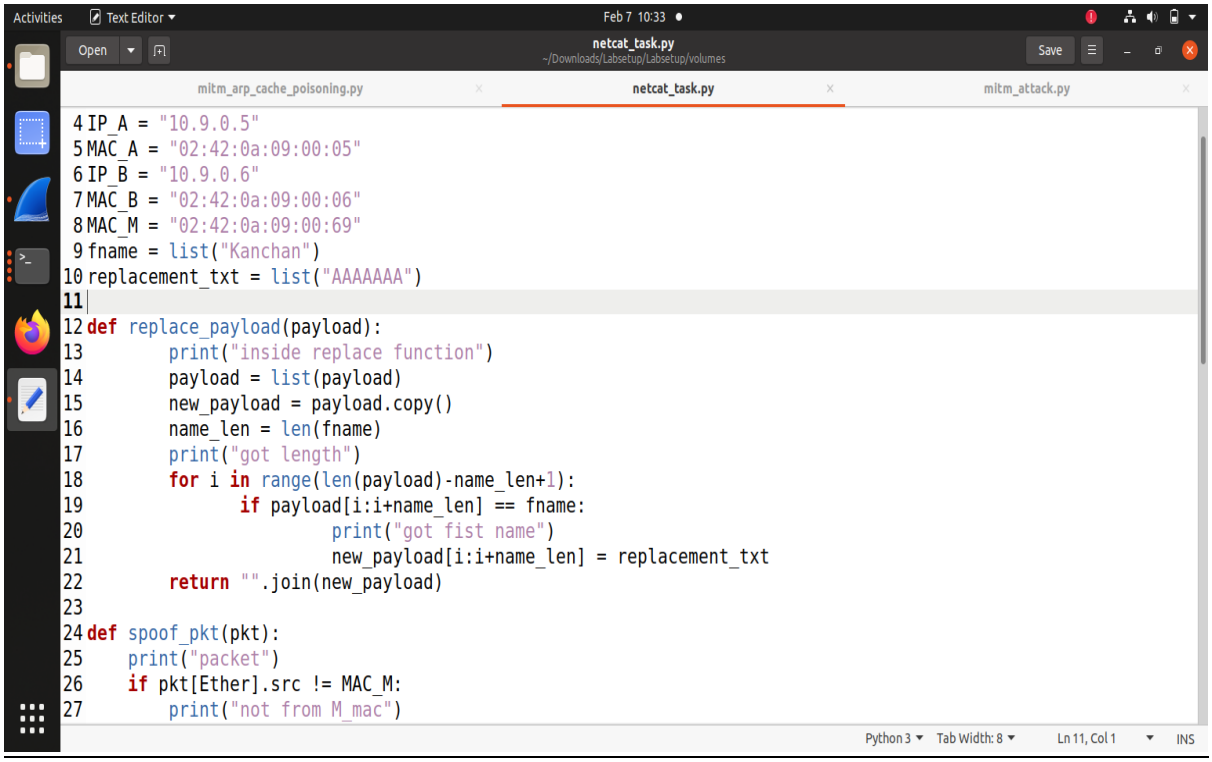
Only if IP forwarding is enabled before Telnet connection and then disabled before other input is made, will the characters input in command prompt be replaced with our character of choice, Z.



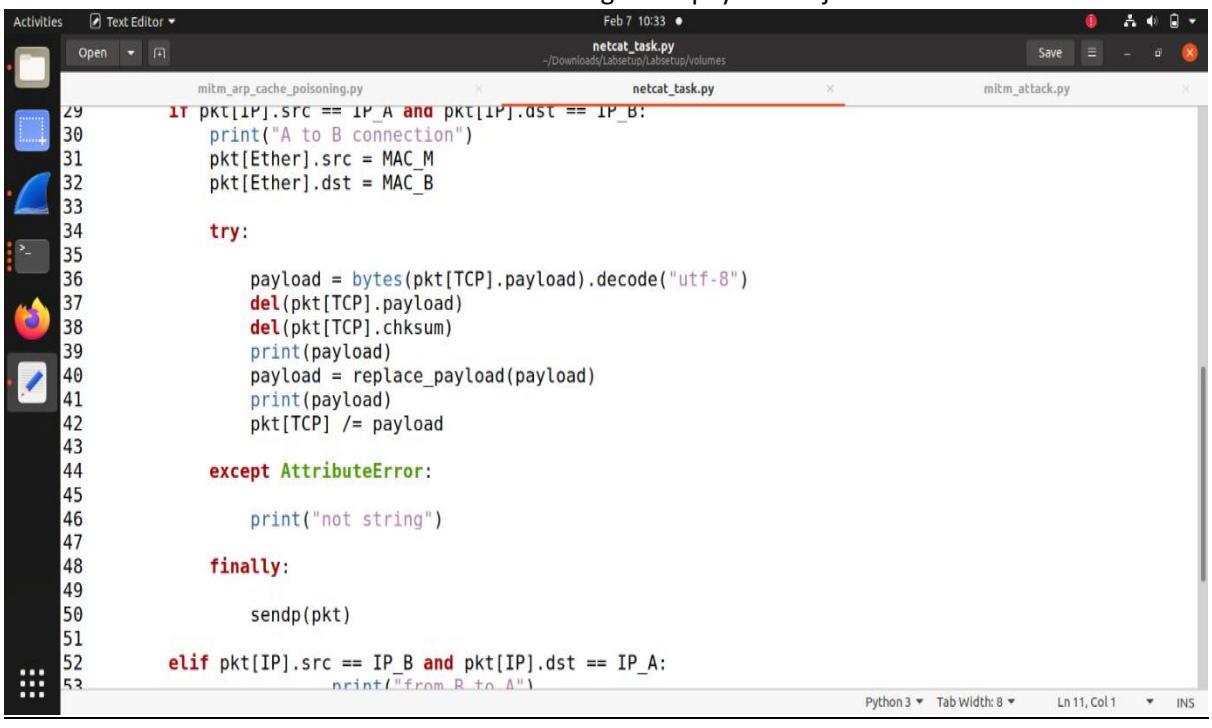
Task 3: MITM Attack on Netcat using ARP Cache Poisoning

Code changes made to previous code: file name netcat_task.py

Added changes to same code above:
fname and replacement has firstname and replacement text.
defined function replace_payload() to which we pass thepayload obtained we use list() to convert payload into list object and we copy payload to new_payload object, we also take length of firstname and iterate over each character of fname until its length and replace it with our replacement text i.e.(“AAAAAAA”) in my case.



Here we make use of function we created and assign it to payload object .



Added few print statements to check if it replaces the payload or not.To run the attack first listen to server using netcat -l 9090, from host A use command : netcat 10.9.0.6 9090.from host M, run first the mitm_Arp_cache_poisoning_attack.py used in previous steps. Then run newly added code netcat_task.py