

ARP Cache poisoning

SEED LAB



[Date]

[Company name]

[Company address]

Contents

[Environment Setup 2](#_Toc135406605)

[Task 1 4](#_Toc135406606)

[Task 1A 6](#_Toc135406607)

[Task 1B 7](#_Toc135406608)

[Scenario 1 7](#_Toc135406609)

[Scenario 2 7](#_Toc135406610)

[Task 1C 8](#_Toc135406611)

[Scenario 1 8](#_Toc135406612)

[Scenario 2 9](#_Toc135406613)

[Task 2 10](#_Toc135406614)

[Step 1 11](#_Toc135406615)

[Step 2 13](#_Toc135406616)

[Step 3 14](#_Toc135406617)

[Step 4 16](#_Toc135406618)

# Environment Setup

Setting Dockers.



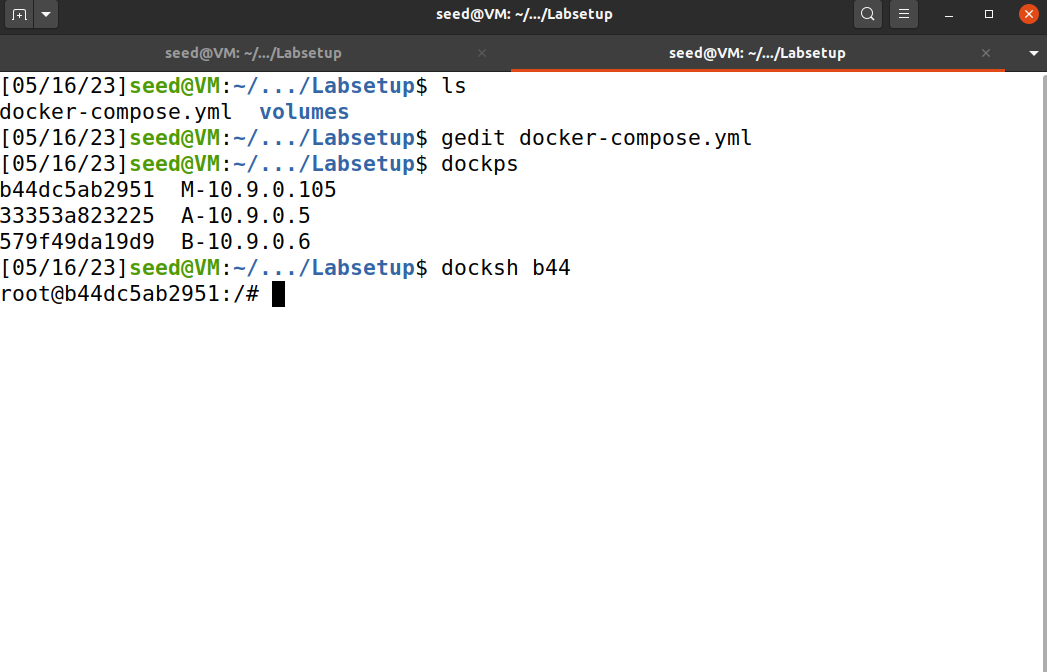
Attaching Host A Docker.

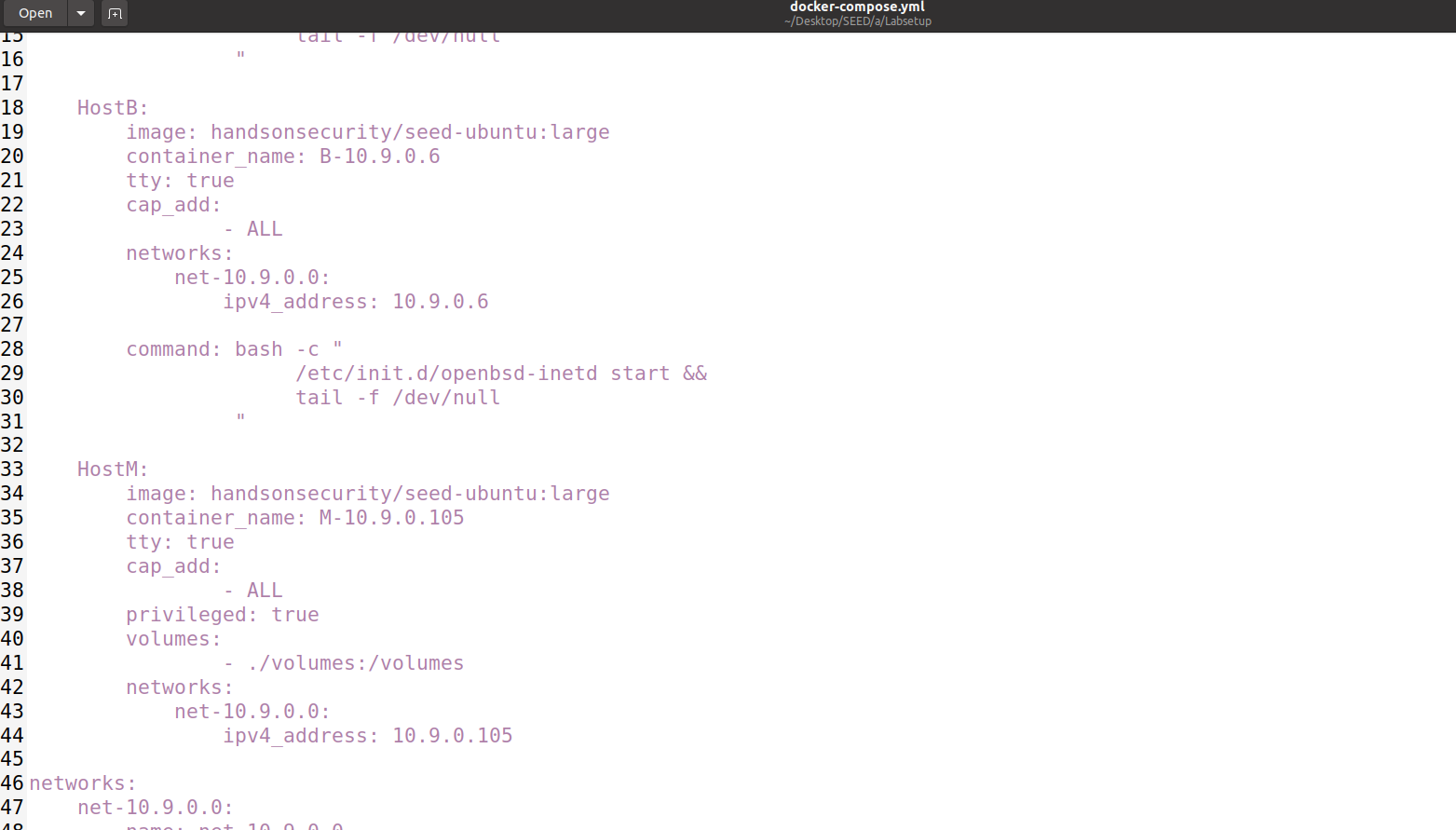


Attaching Host B Docker.



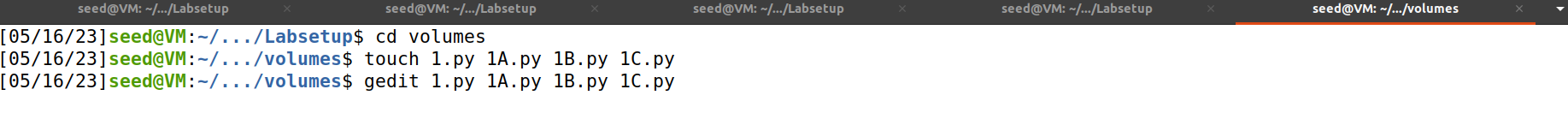
Attaching Attacker’s docker.



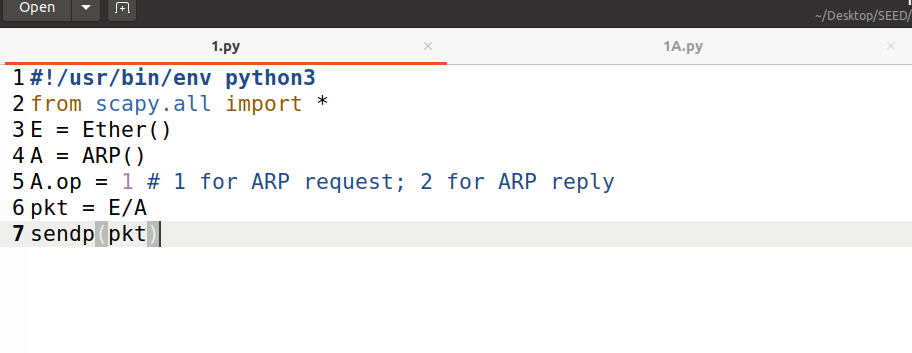
As visible in the screenshot I have identified the Host B and A as a normal user and Host M as the attacker due to the presence of ***privileged: true*** on line 39 of **docker-compose.yml**.

# Task 1

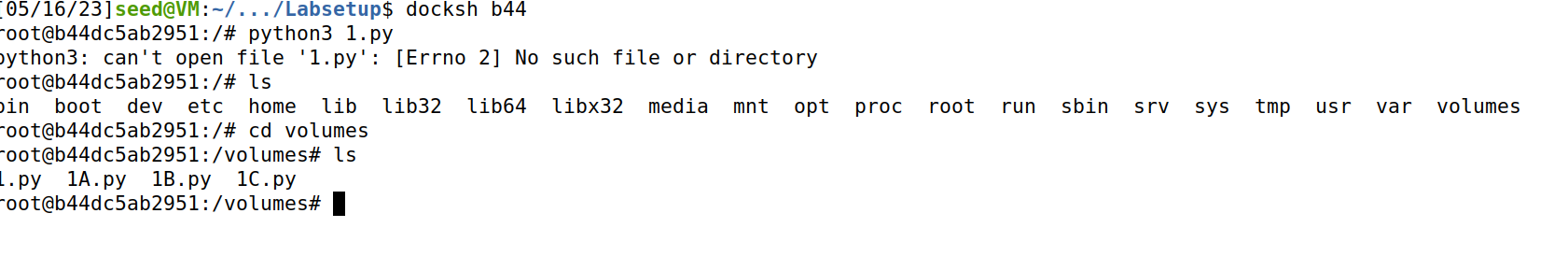
Creating files in volumes to be accessible by the attacker for attack purpose.



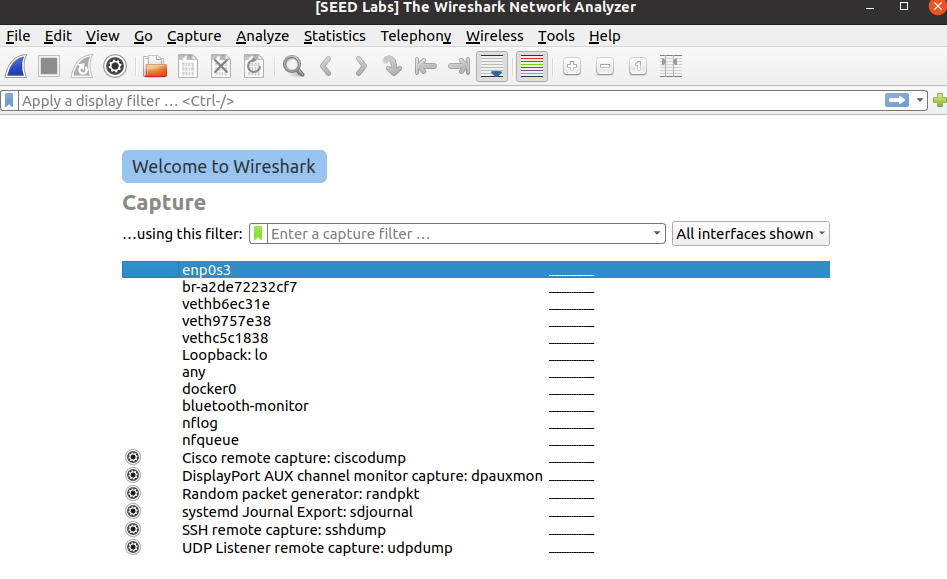
Using the Skeleton Code provided in Manual and testing it.



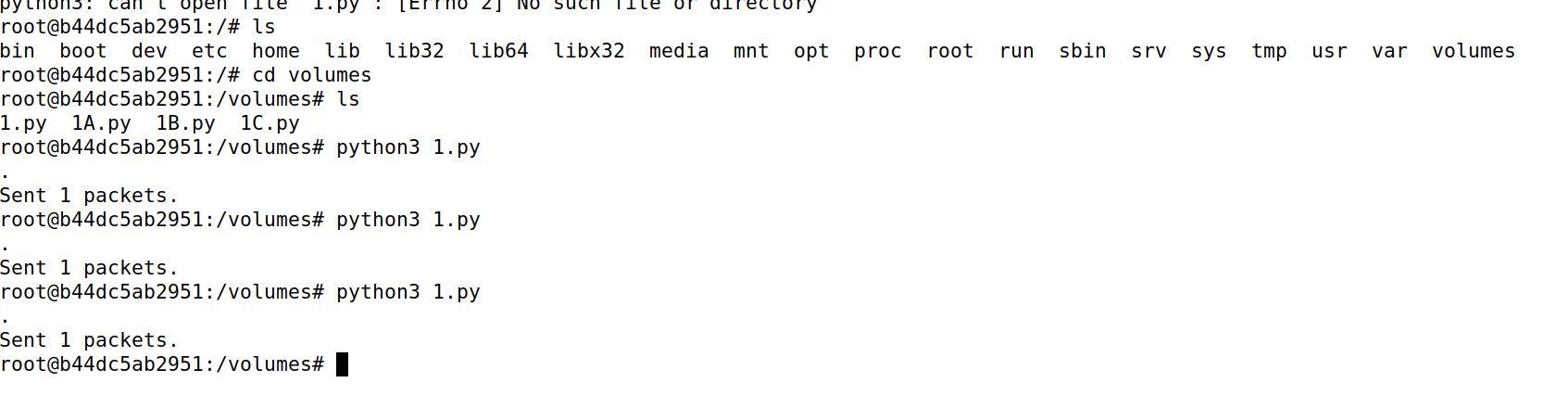
In attacker’s docker the ***volumes***is visible and when entered the scripts are too.



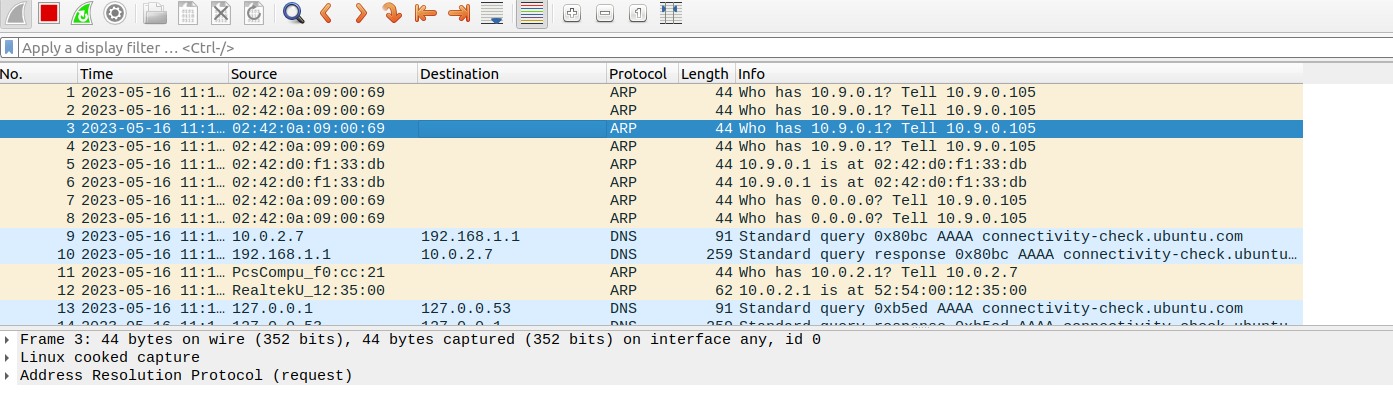
Using wireshark to check movement on network.



Now when launching the skeleton code it shows that the code works and a packet has been sent.

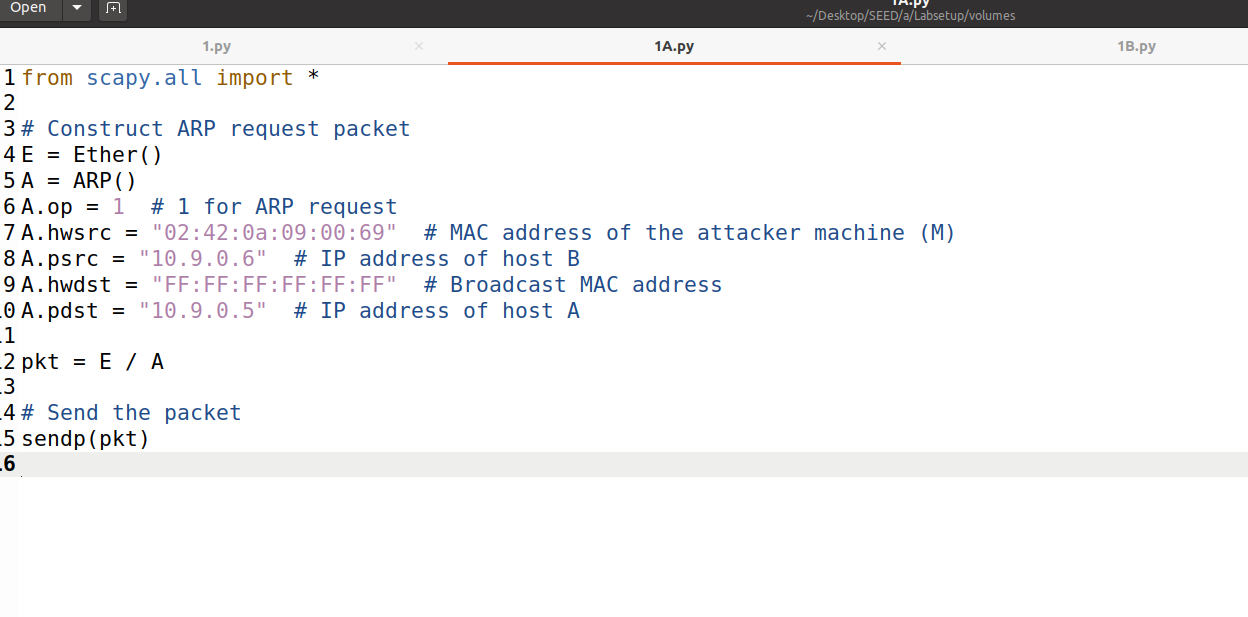


ARP Packets were detected on Wireshark.

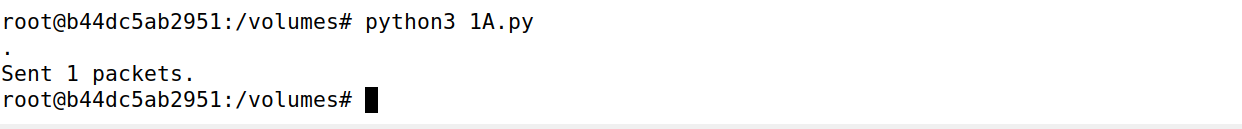


## Task 1A

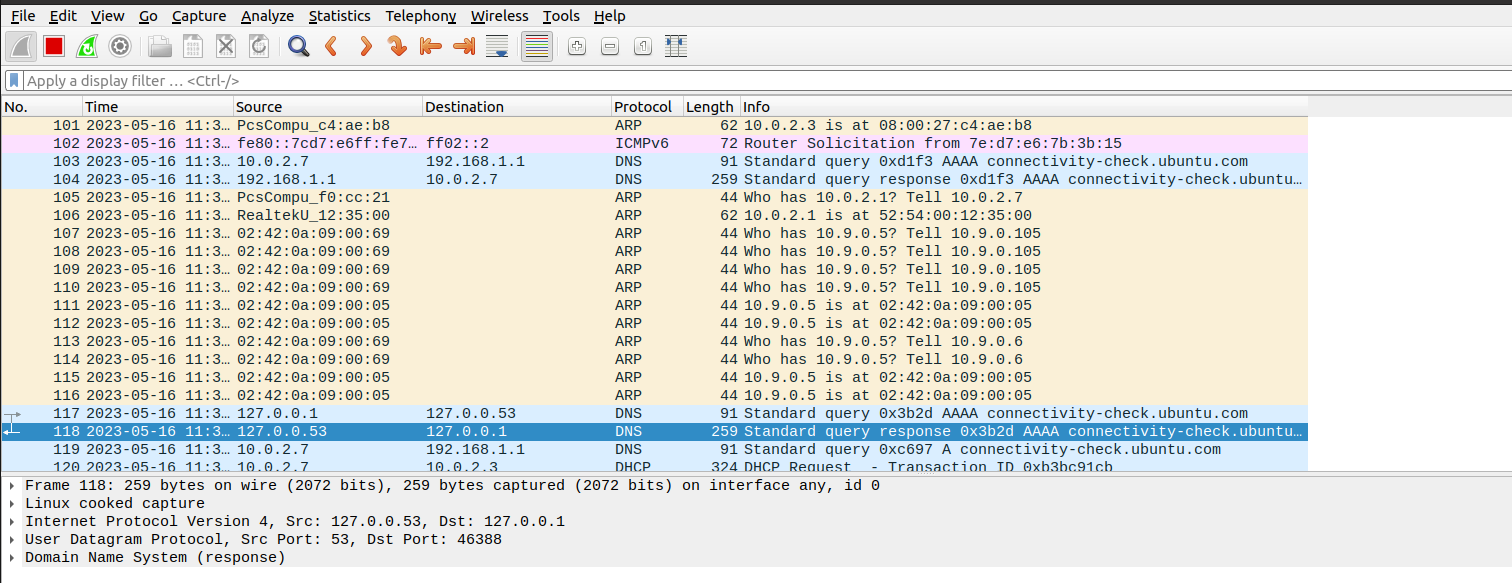
Wrote this script for the task.



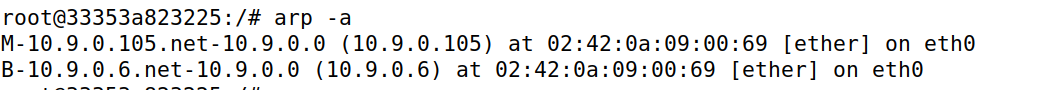
Launching Attack.



Wireshark showed this response.



Checking on Host A if the attack was successful by checking ARP Cache. It proves the success of the attack as the Attacker’s MAC Address has changed to the MAC Address of the Host B.



## Task 1B

Wrote this script for the task.

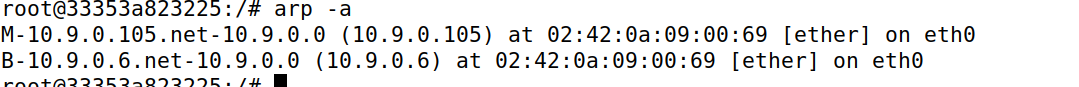


Launching the Attack after which Wireshark shows the ARP Packets.



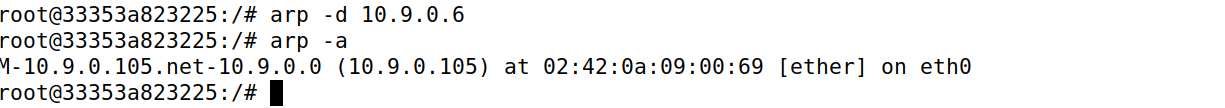
### Scenario 1

When the IP of Host B is in Host A’s ARP Cache the attack seems to be successful but the previous attack was a success so if it overwritten or the previous result we need to move to a better solution.



### Scenario 2

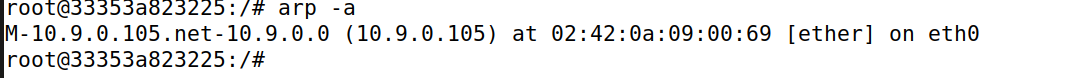
Removing Host B’s IP from ARP Cache of Host A.



Now performing the Attack.

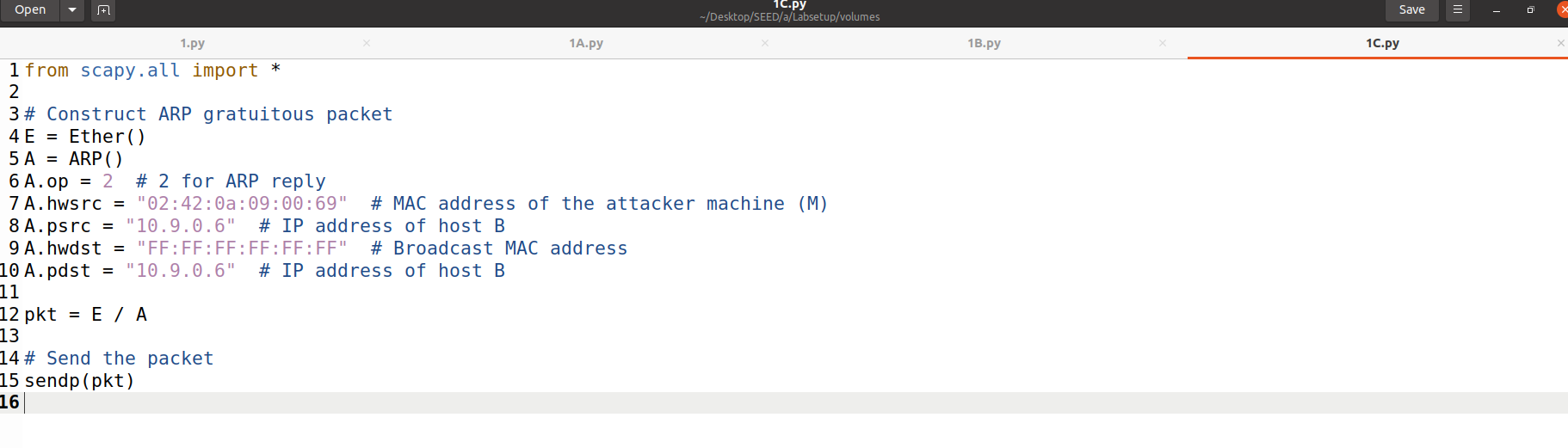


The results suggest that the attack was not successful. From which we can conclude that either scenario 1 was successful or both were not successful.



## Task 1C

Wrote this script for the task.



### Scenario 1

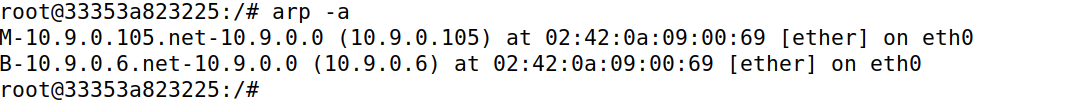
Filling Host A’s cache by using the script for Task 1A.



Launching the Attack



Now the results tell that the attack is successful.



While the Wireshark tells the reply received with attack being successful by broadcasting our desired MAC Address.

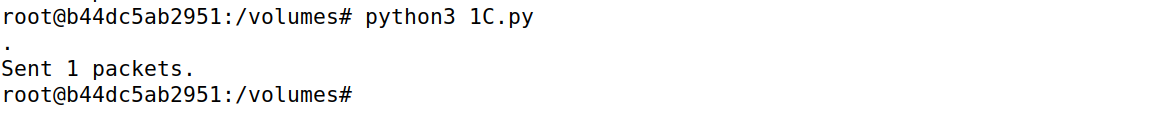


### Scenario 2

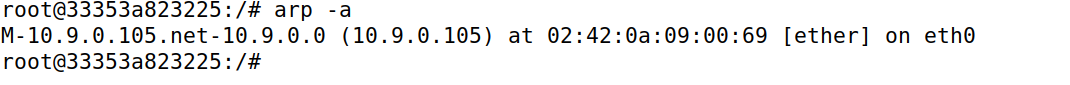
Removing Host B’s IP from ARP Cache of Host A.



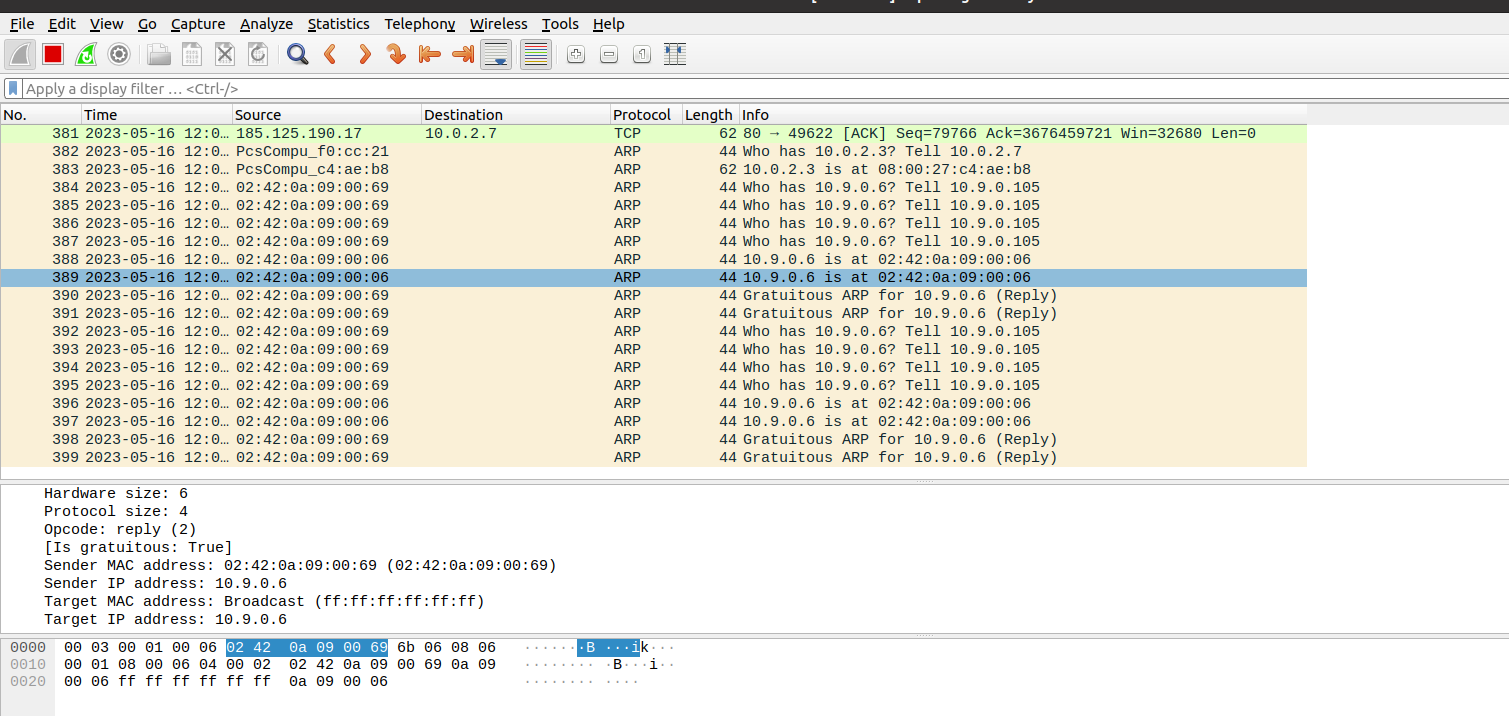
Launching the Attack now.



Checking Host A’s cache which doesn’t display Host B’s IP.

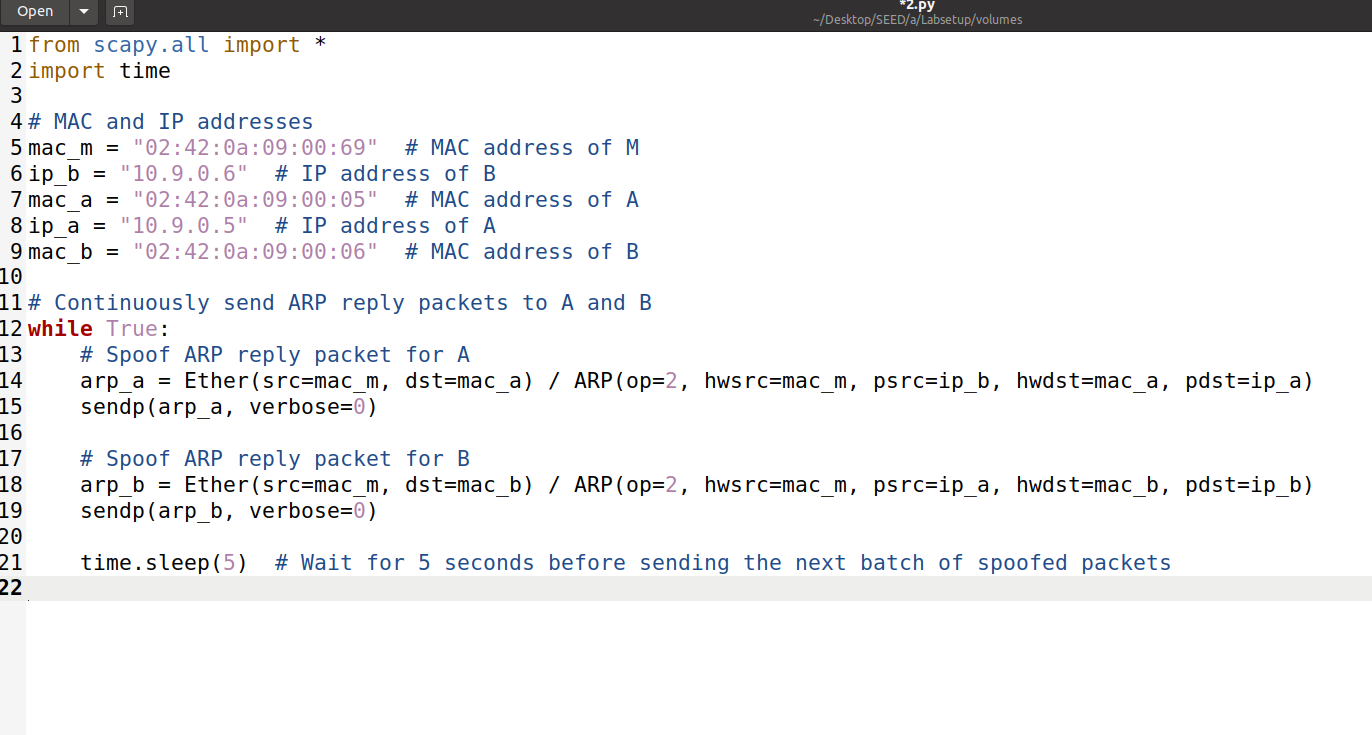


But Wireshark says that the attack is successful given the Broadcasting MAC is our desired MAC which makes our attack as successful even if it is not in the cache.



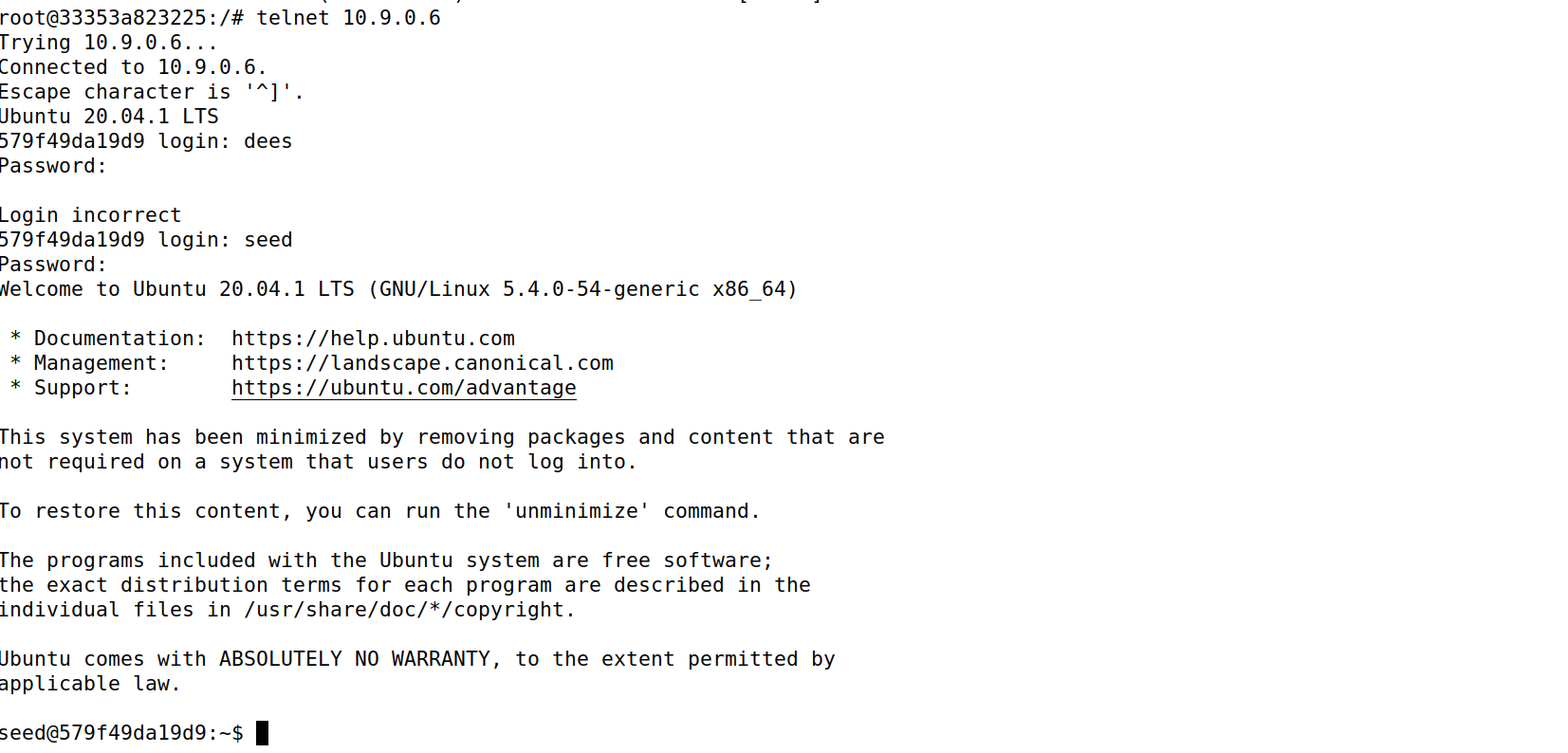
# Task 2

Creating the Packet to intercept telnet connection between Host A and Host B.

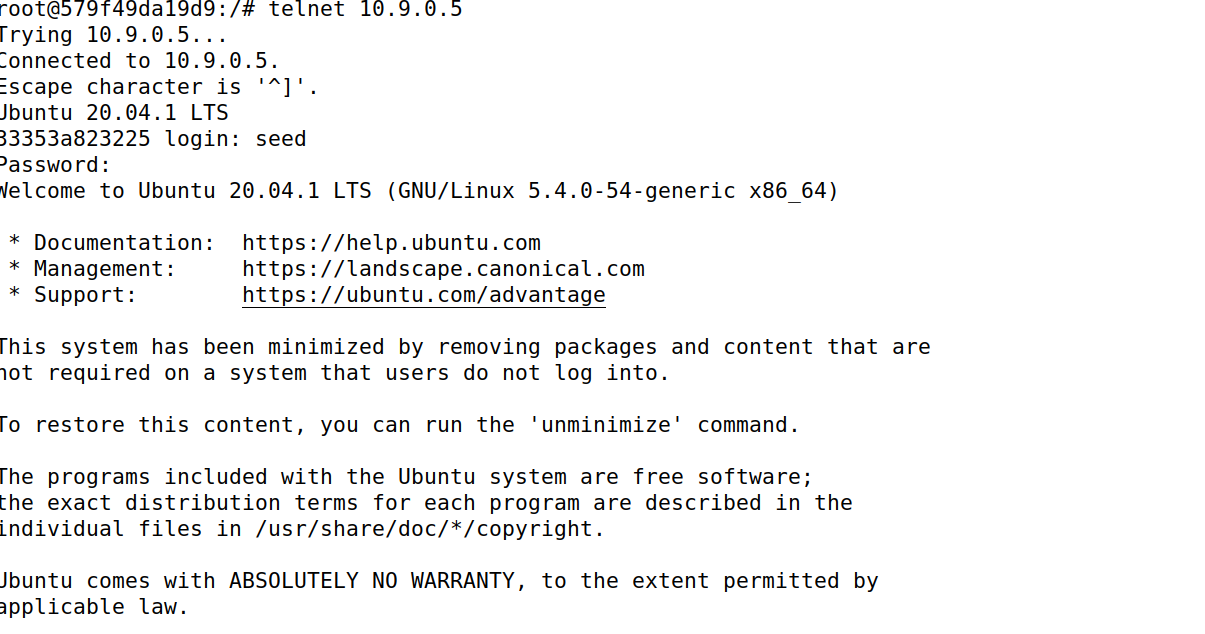


## Step 1

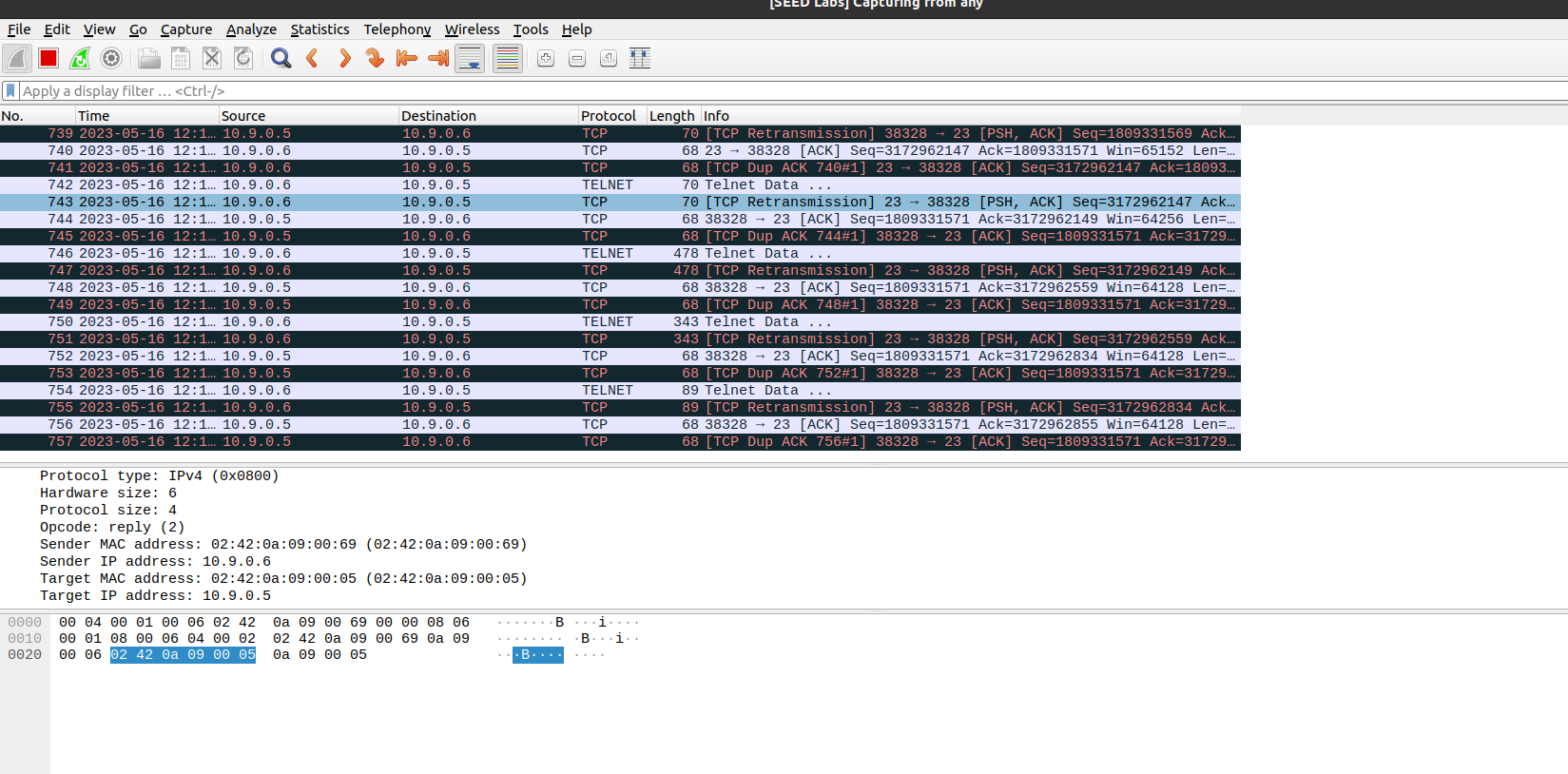
Establishing the telnet connection from Host A to Host B.



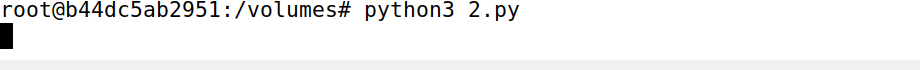
Establishing Telnet from Host B to Host A.



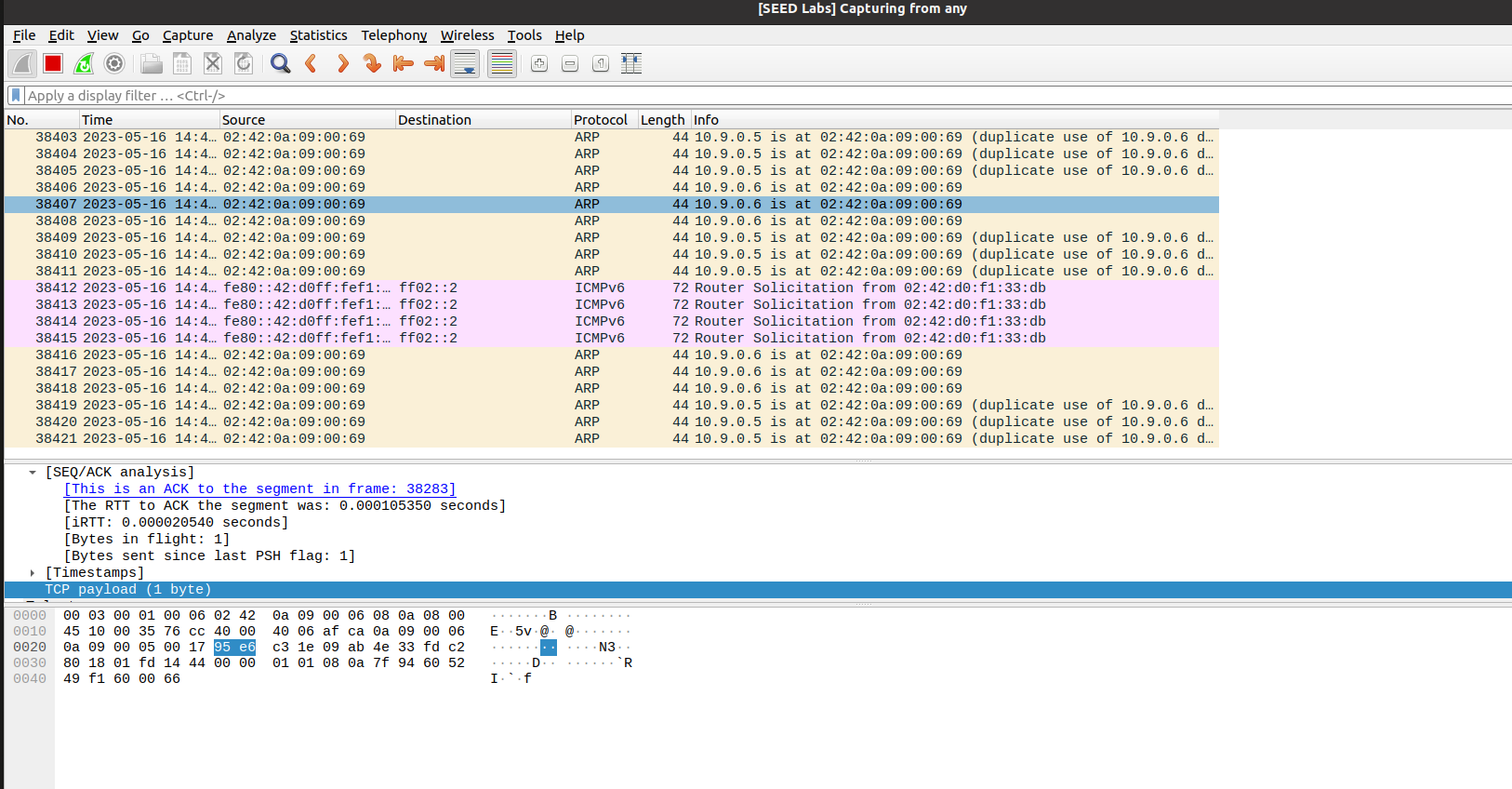
Which was recorded by the Wireshark.



Now launching MITM after initiating ping from Host B to A on telnet.

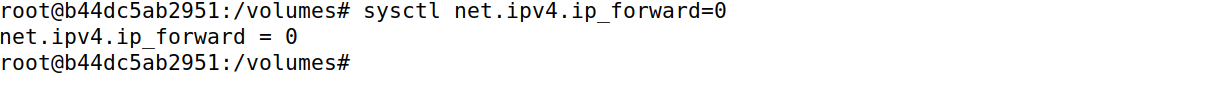


Wireshark shows the ARP Packets duplicate use of M Attacker by Host A and Host B.

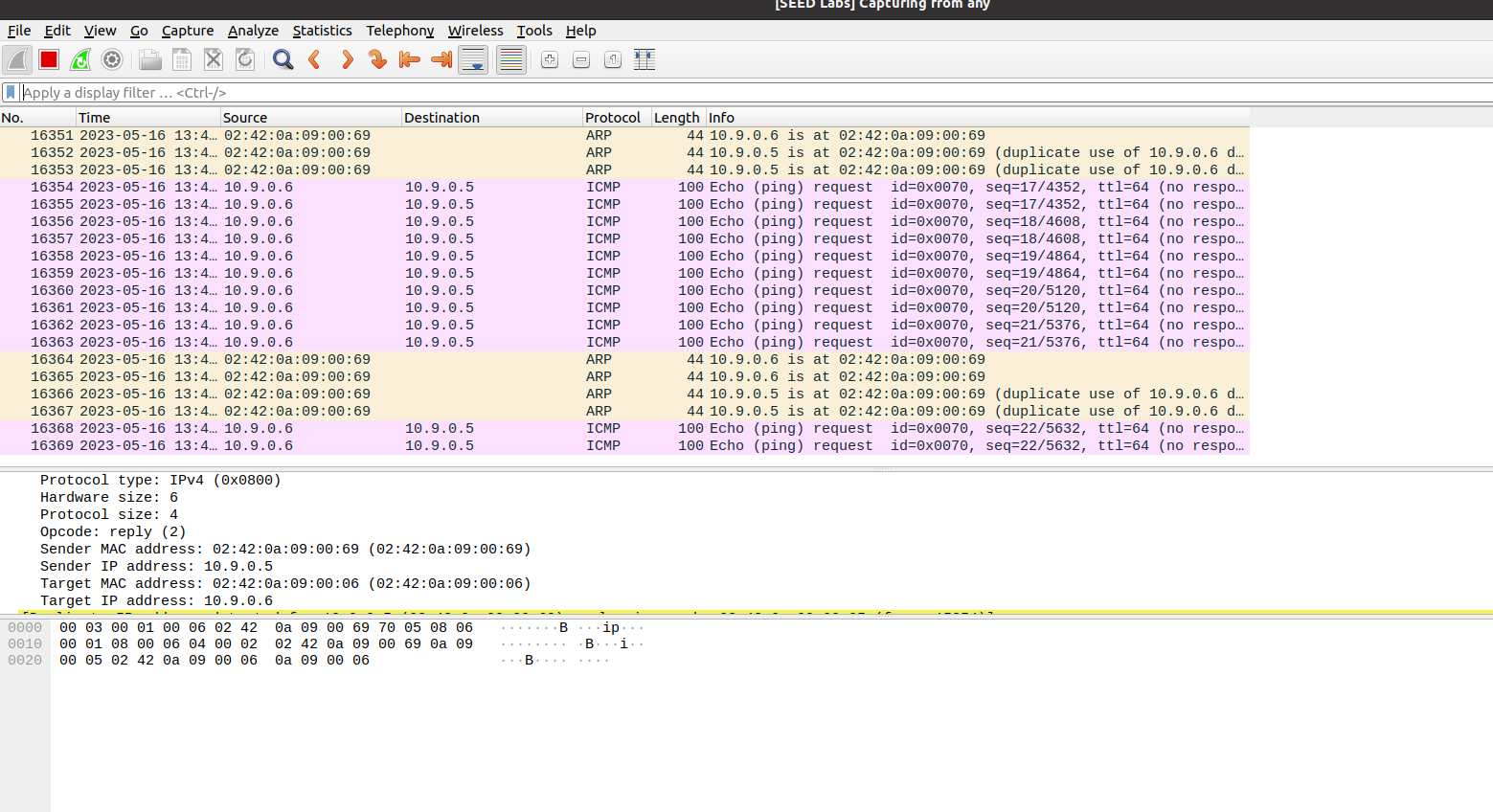


## Step 2

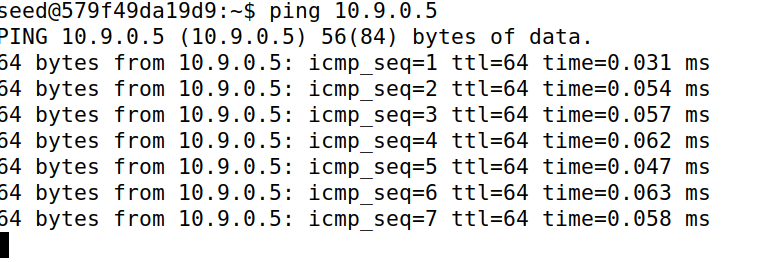
Turning off IP forwarding on Host M which is the attacker.



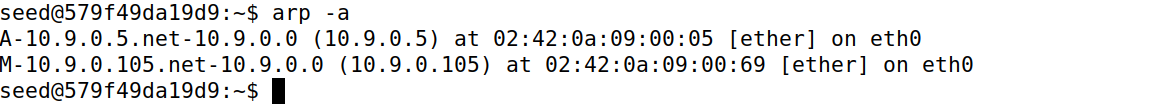
With IP forwarding off no redirection has been observed although the conversation is active between the hosts when pinging but the ICMP ping requests and ARP packets are observable with duplicate use for M’s Mac Address on Host A and B.



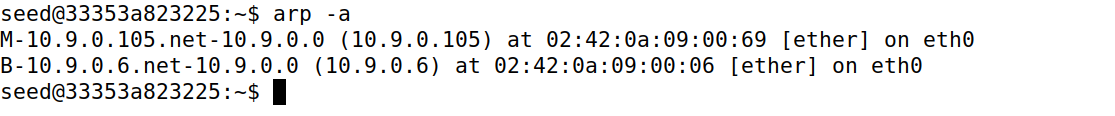
Another observation being no further pings during the attack.



The Host B’s cache shows Host A’s IP Address.



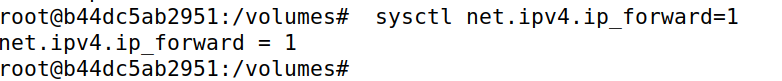
The Host A’s cache shows Host B’s IP Addres.



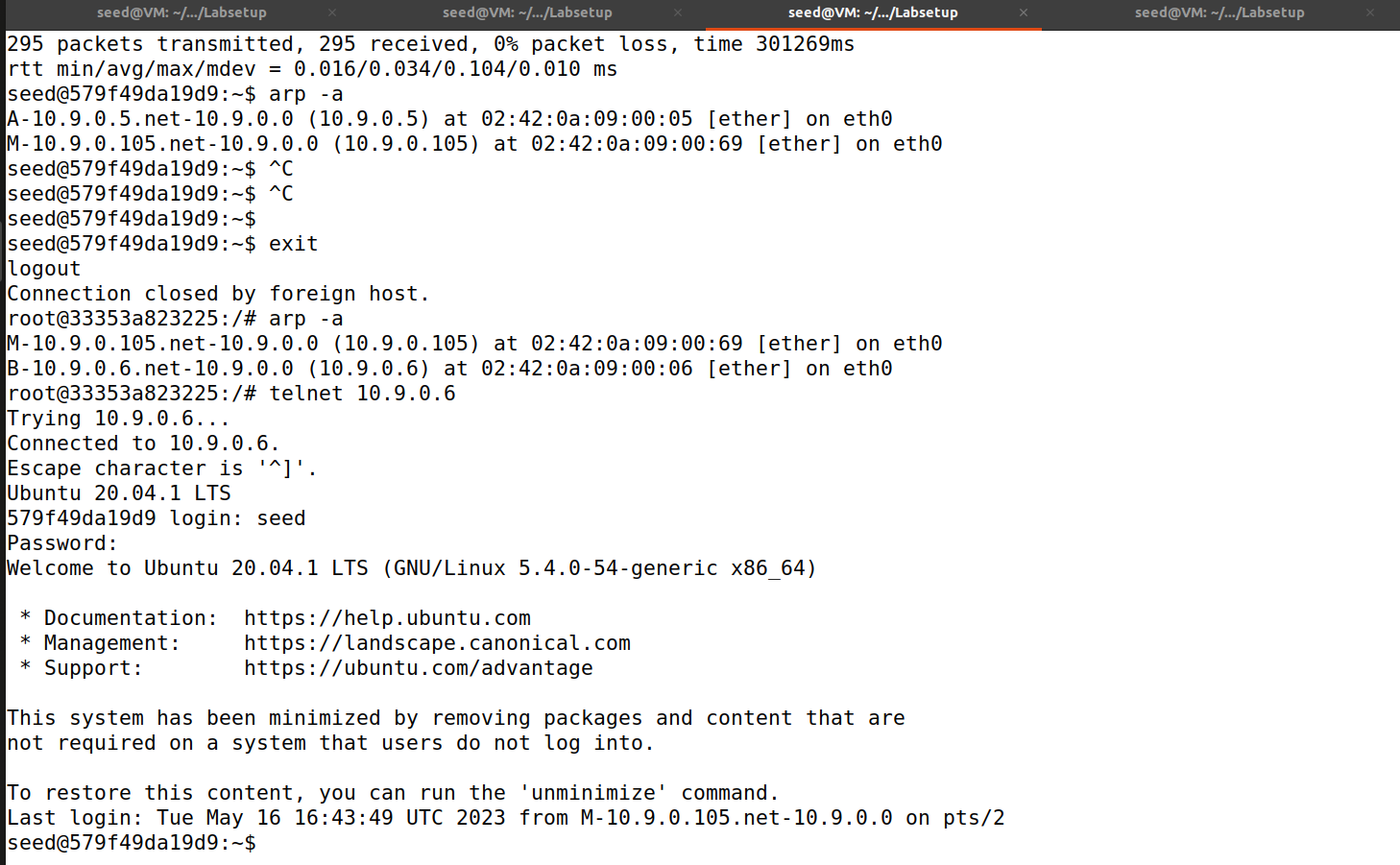
This proves the attack was successful as the communication was shown between Host A and Host B but only while observing the cache but. Redirection is not visible in Wireshark due to current configurations but duplicate use of MAC Address of Attacker M.

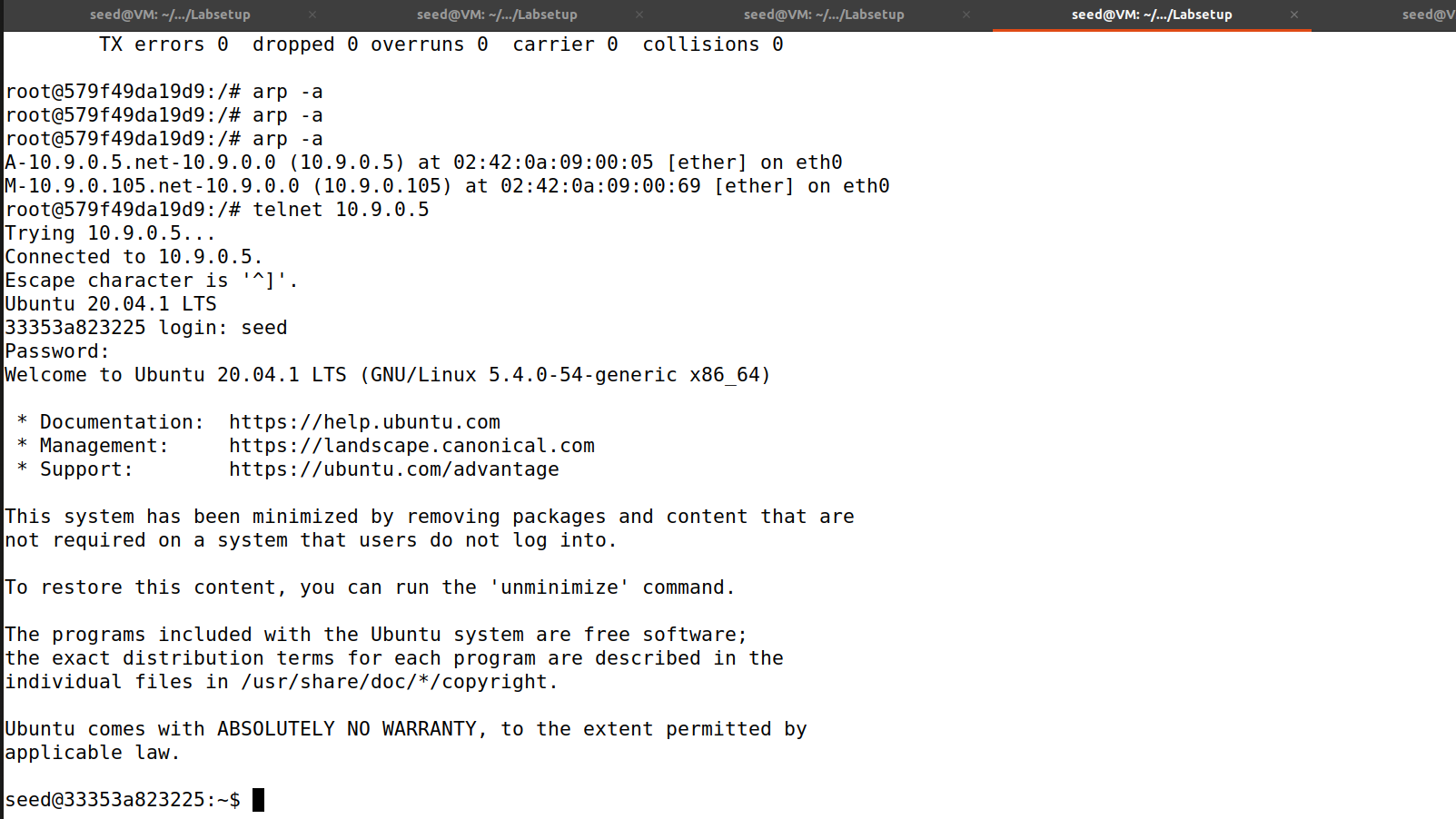
## Step 3

Enabling the IP forwarding on Host M which is the attacker.

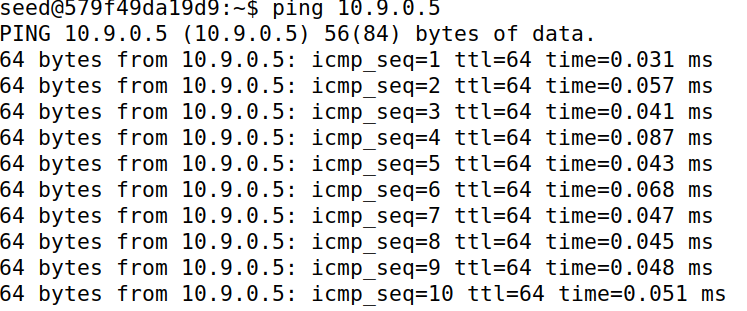


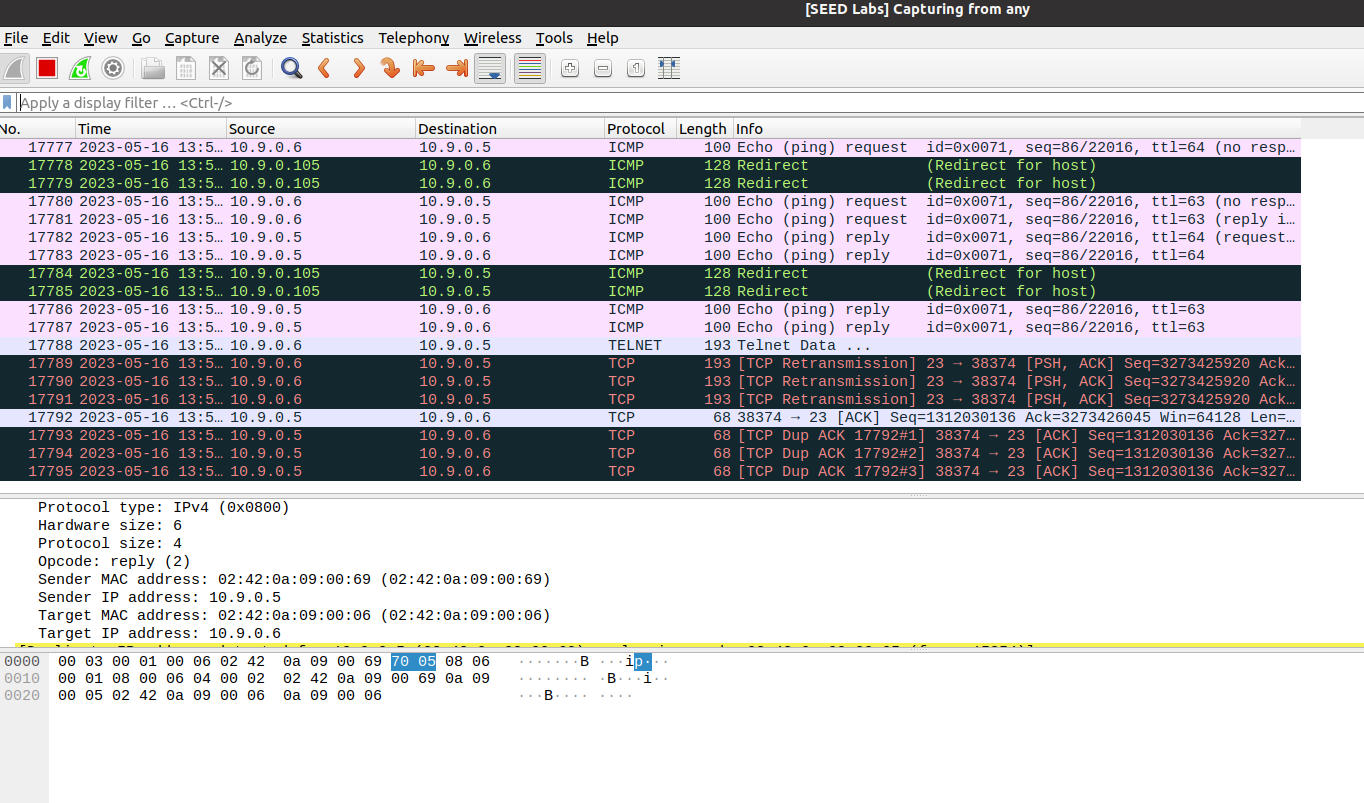
This time connecting both Hosts A and B through telnet and pinging the other.



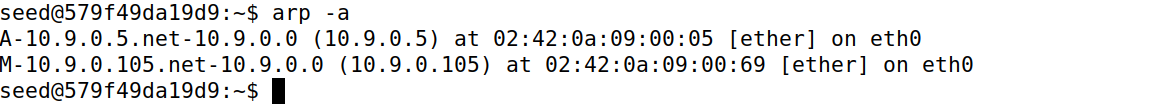


Pinging from Telnet Host B to A.

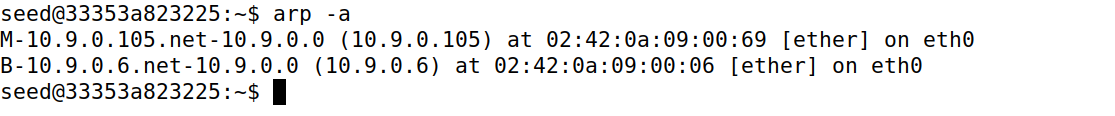


Launched Attack as a precaution from Attacker’s Machine and this time Wireshark displays ICMP redirect spoof packets.  


The Host B’s cache shows Host A’s IP Address.



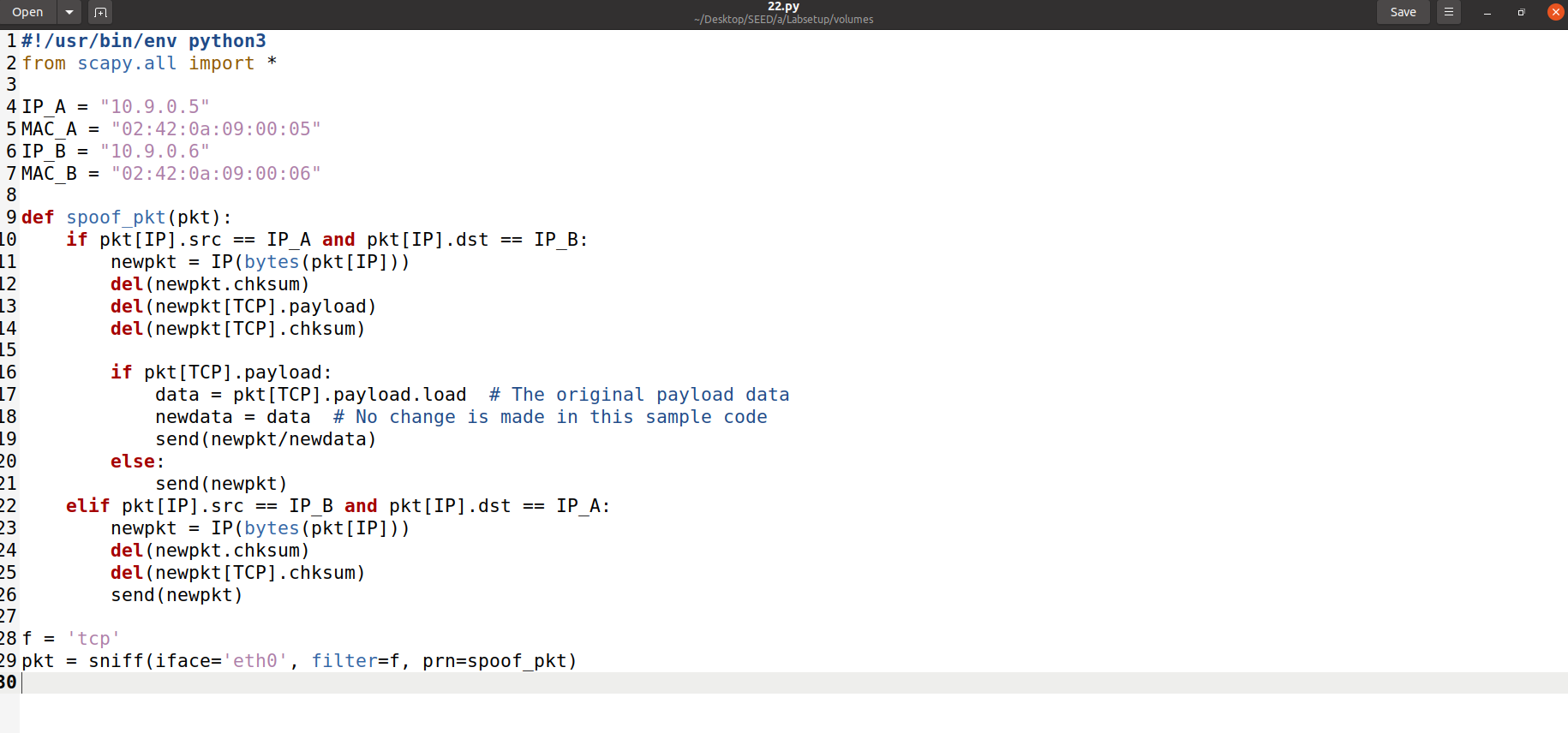
The Host A’s cache shows Host B’s IP Address.



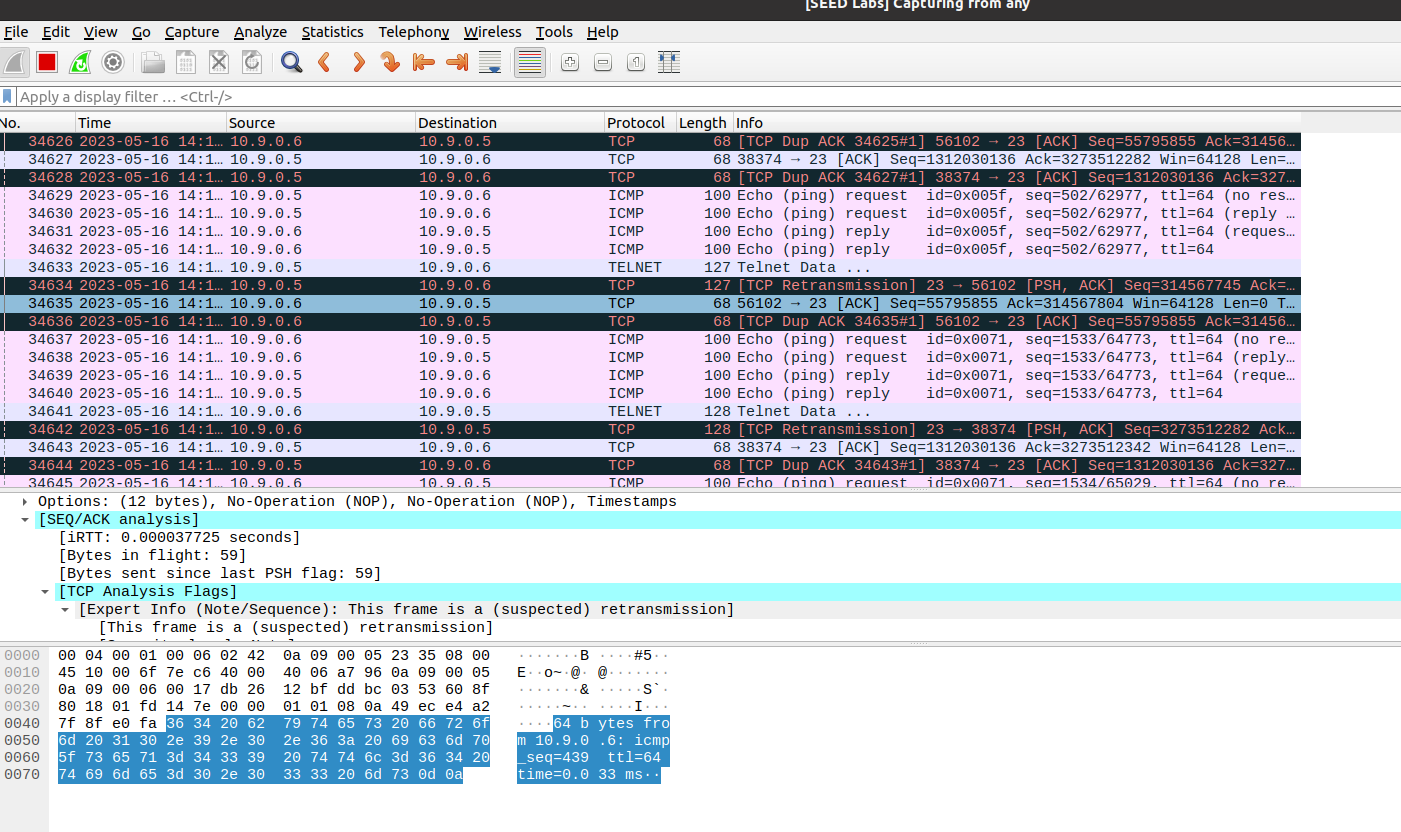
This proves the attack was successful as the communication was shown between Host A and Host B.

## Step 4

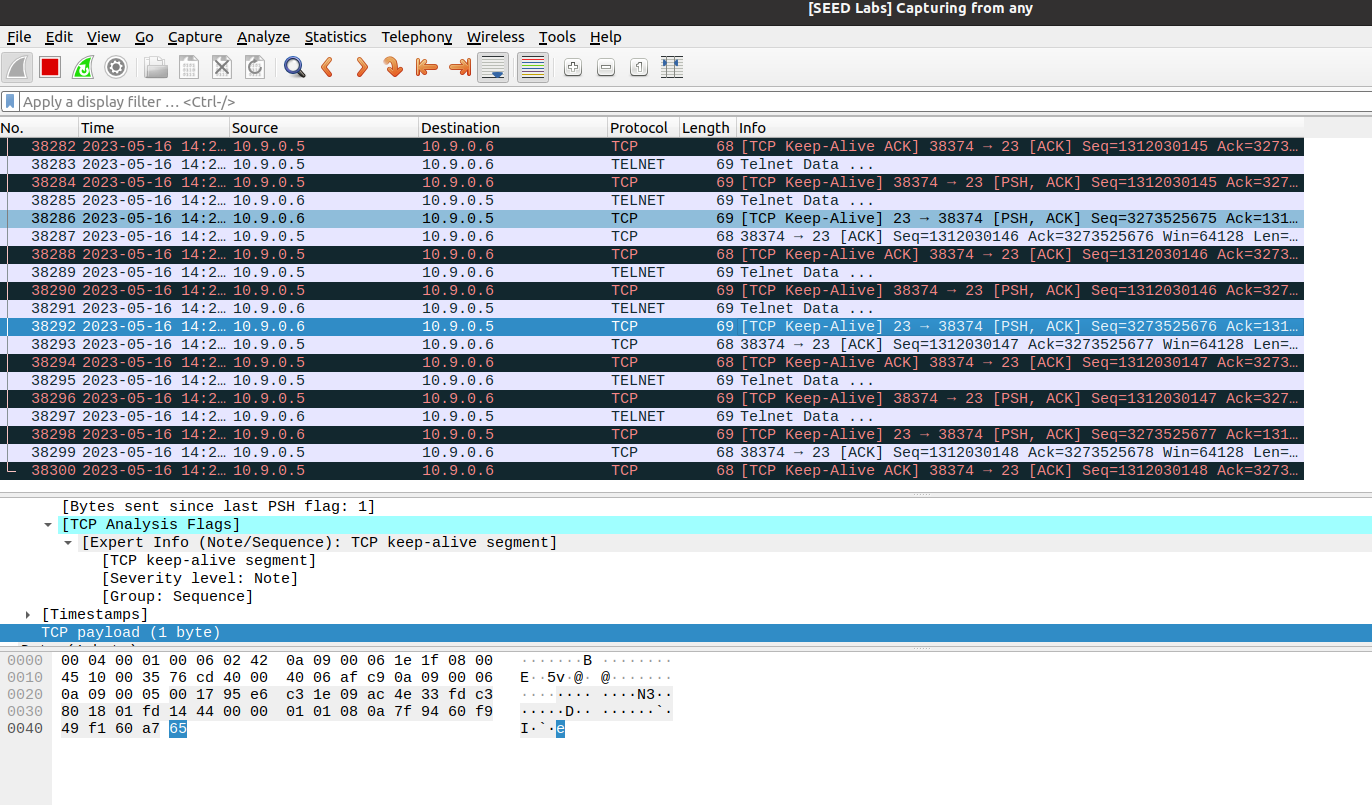
Using this script.



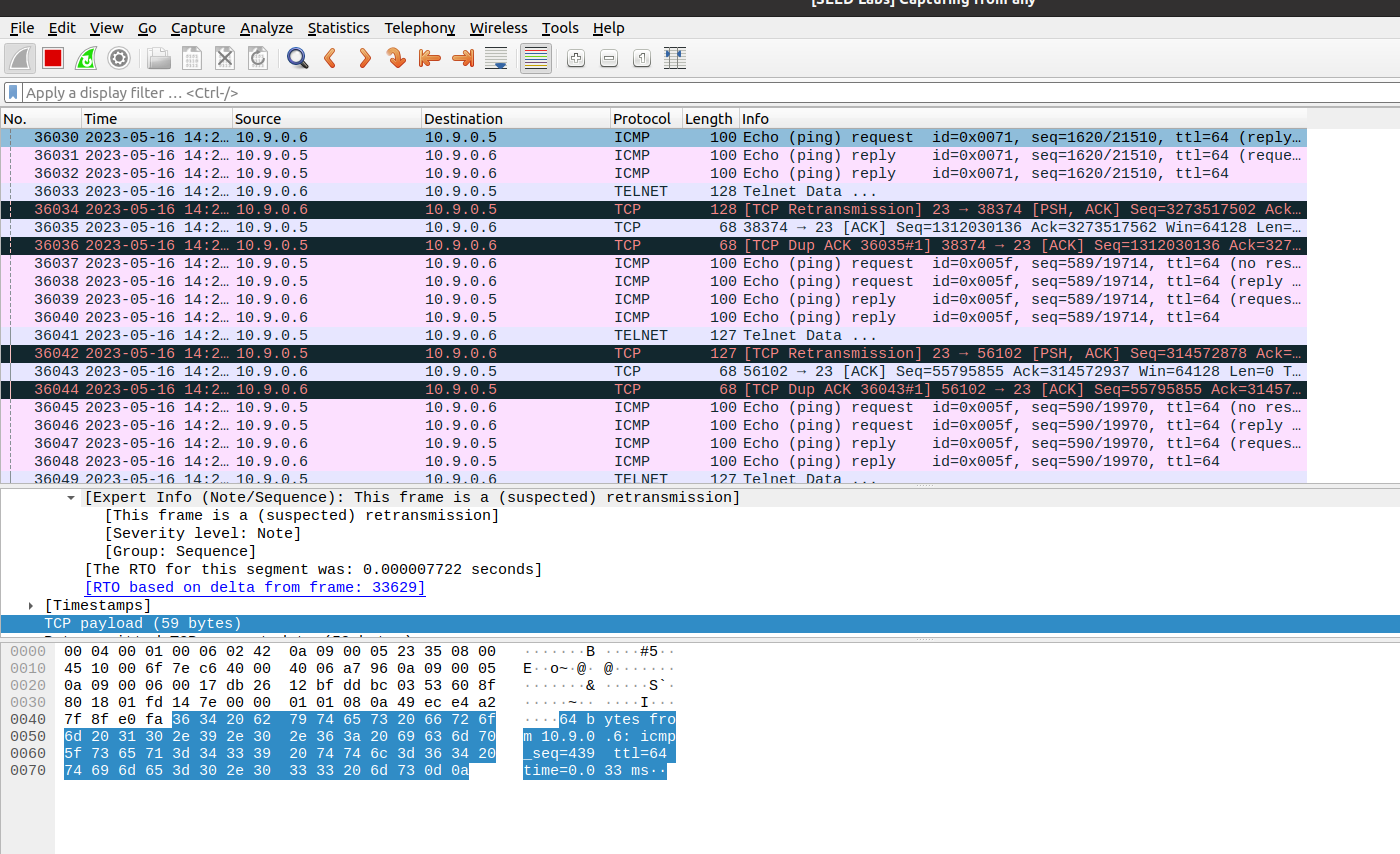
When pinging both hosts via telnet to the other the Wireshark displays this.



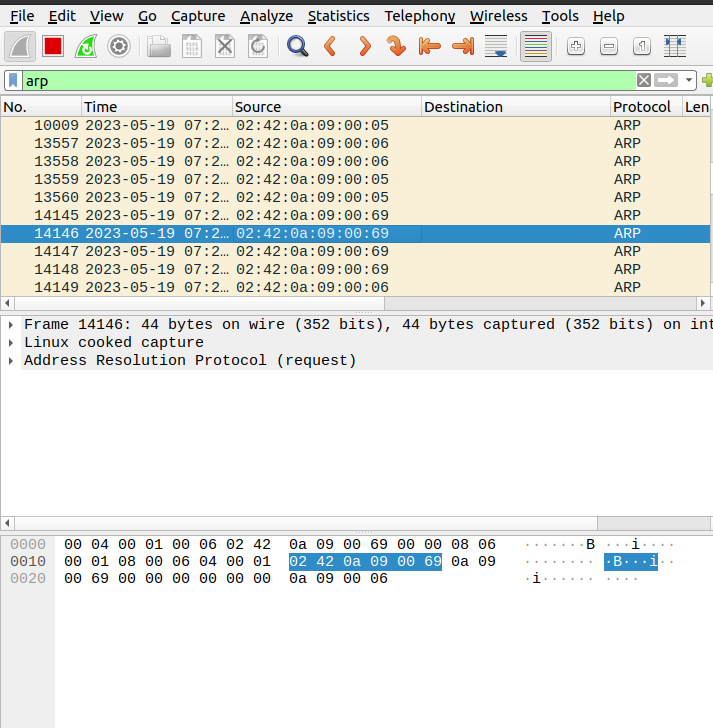
And when not pinging just typing characters this is the result.



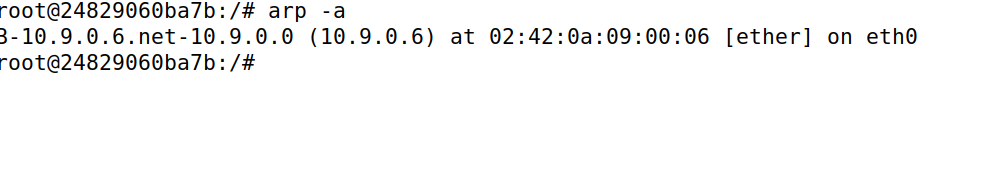
When I launch the attack this turns to this given in our attacks pinging process and typing in telnet were active.



And results surprisingly caught the attack working upon restarting the whole step and filtering ARP packets.



As evidence I have checked the cache of Host A which consists of Host B’s IP and MAC Address.



Then I checked the cache of Host B and found out the MAC and IP of Host A and the Attacker which is Host M.

