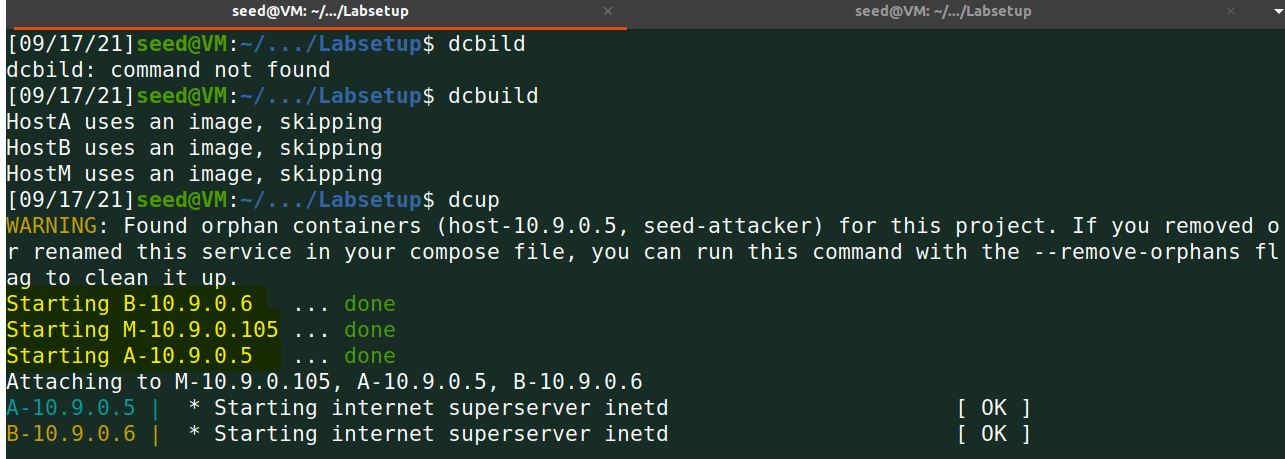
Lab-3 Report: ARP Cache Poisoning Attack

Shrikant Kale - 700727822

1. **Lab Environment Setup**

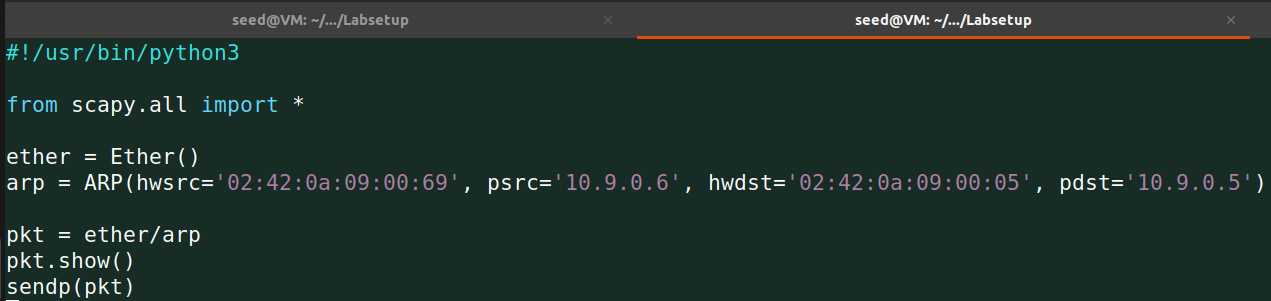
* Created Lab environment using docker containers.
* To execute docker container, I have used docker-compose commands.
  + dcbuild (docker-compose build) – to build the docker container image
  + dcup (docker-compose up) – to run the docker container
  + dcdown (docker-compose down) – to shut down the container

****

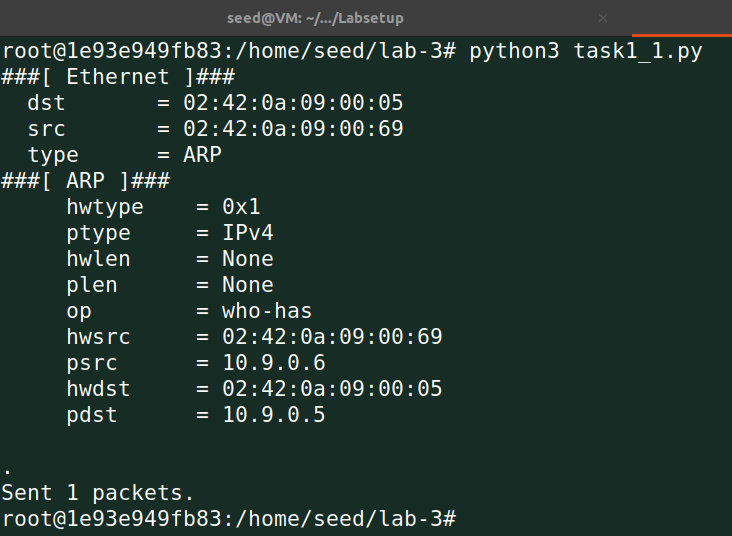
1. **Task 1: ARP Cache poisoning**

**Task 1A) Using ARP request:**

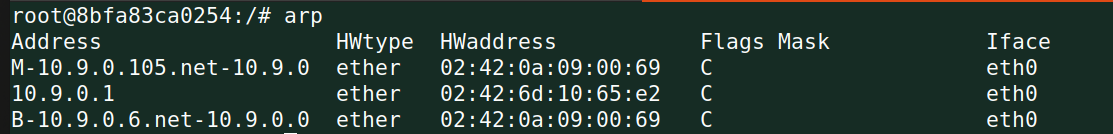
The following is the code to perform ARP Cache poisoning using spoofed ARP request to A:

****

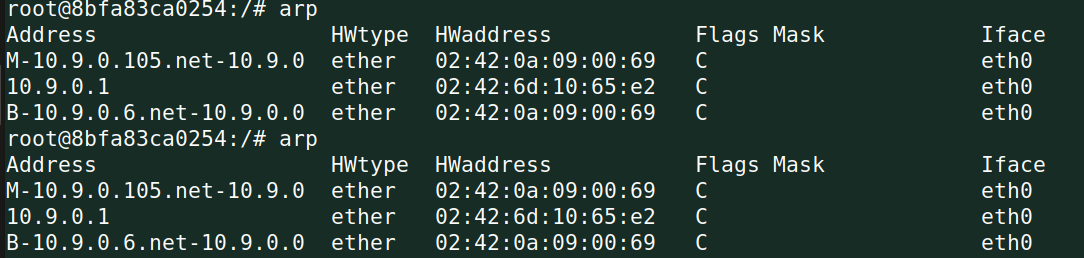
In the above code, we create an ARP packet with source address as B’s IP and M’s MAC and destination as A’s IP and MAC address. We run the above code and see the packet sent out as:



The op who-has field indicates that it is an ARP request. The following show the ARP for B:



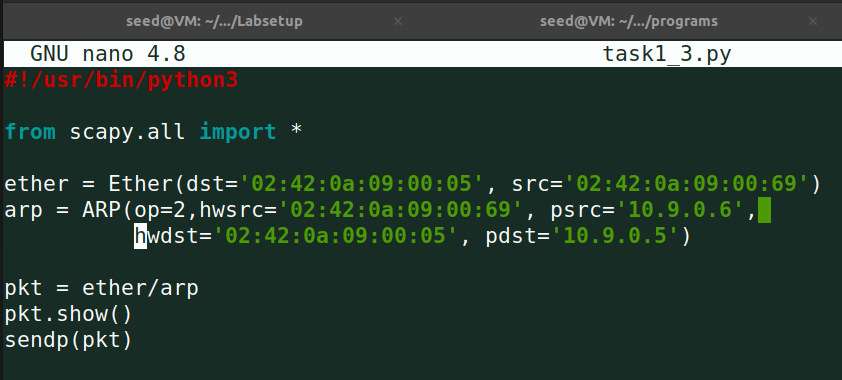
Normally, the ARP requests are broadcasted. However, we wanted to poison only A’s ARP Cache, so we create a message and send it to A. We see that we are successful in our attack.



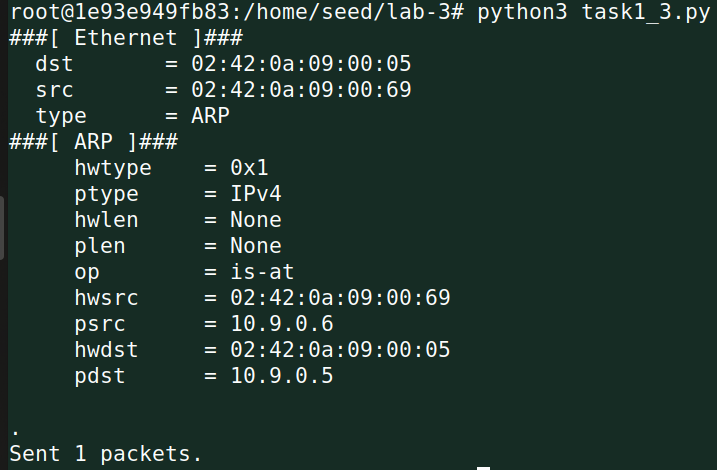
The above code no more results in storing Attacker’s entry in the ARP Cache of A.

**Task 1B) using ARP reply:**

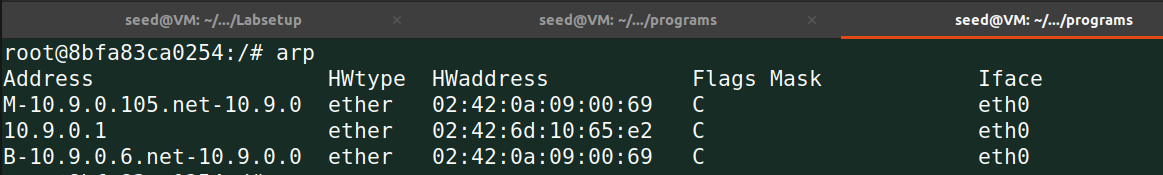
The following is the code to perform ARP Cache poisoning using spoofed ARP reply to A:



The only change here is that the OP field is set to 2 i.e. ARP reply. Rest of the code is same. On executing the program, we see the following packet is sent out:

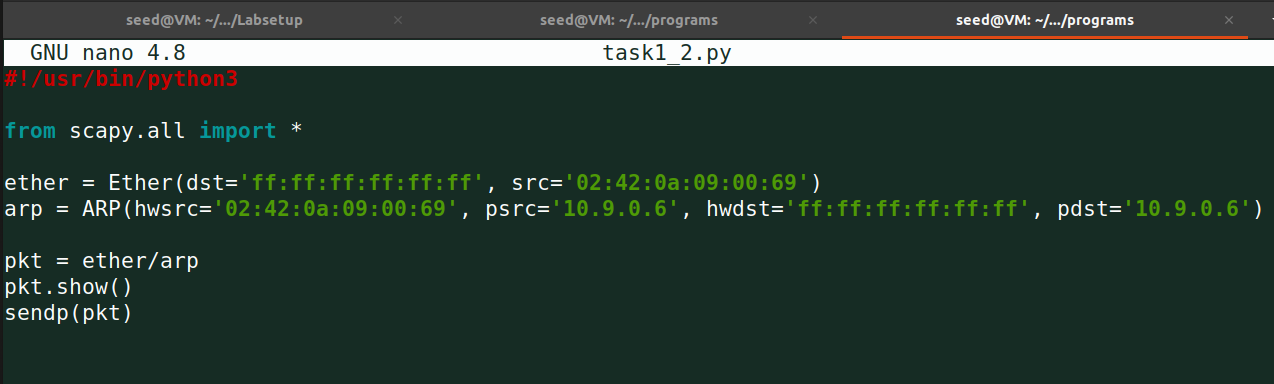


The is-at string in op indicates that it is an ARP reply. The following is the ARP Cache entries.

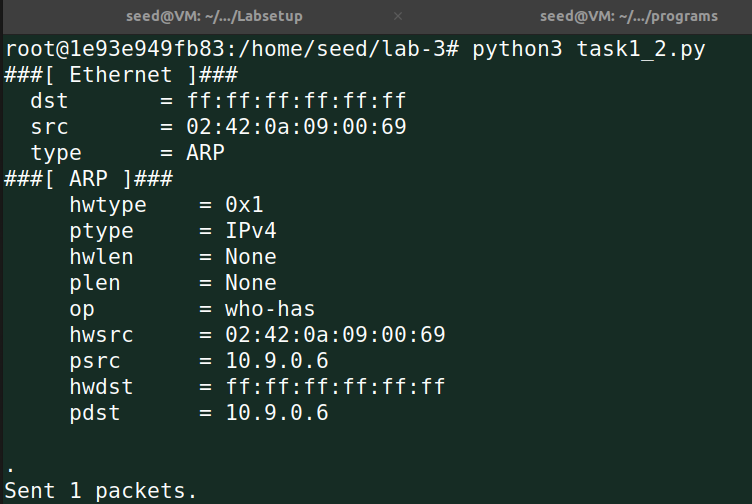
****

**Task 1C) using ARP gratuitous message:**

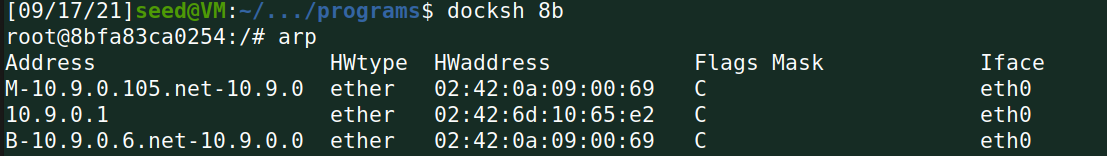
We spoof an ARP gratuitous message with B’s IP address using the following program:



On running the above program, we see that the desired packet is sent out:

****

The following shows the ARP cache after running the program:

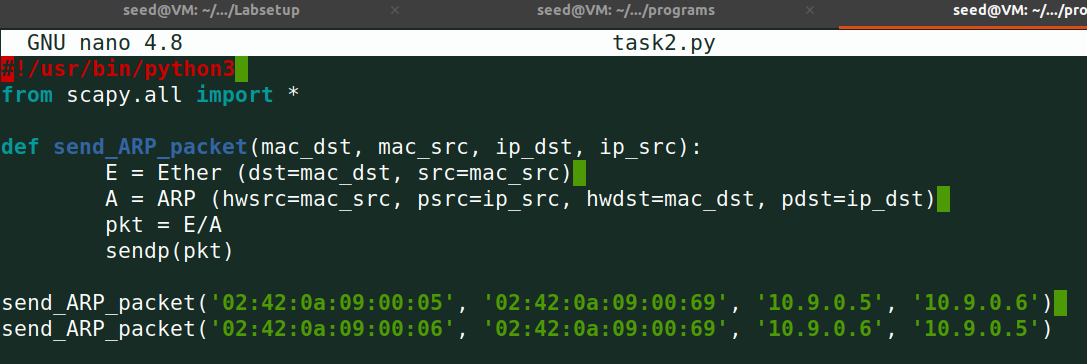
****

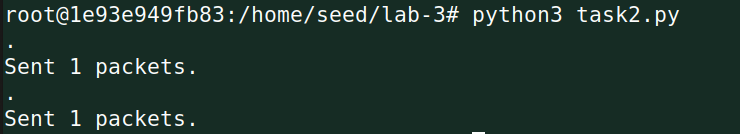
In the above output, we see that only A’s ARP Cache changes and even though B received the packet

1. **Task 2: MITM Attack on Telnet using ARP Cache Poisoning**

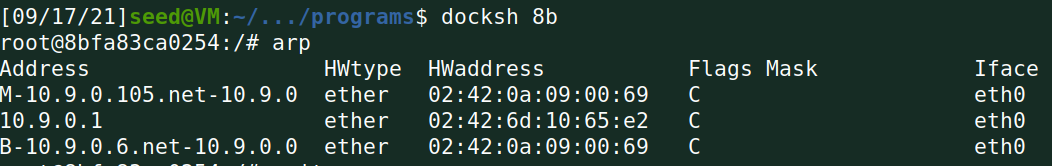
**Step 1 (Launch the ARP cache poisoning attack):**

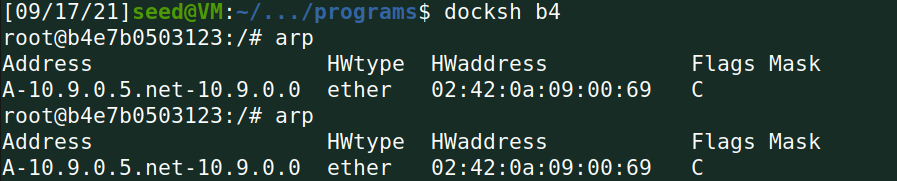
The following provides the code to perform ARP Cache Poisoning on A and B, such that in A’s ARP cache, B’s IP address maps to M’s MAC address, and in B’s ARP cache, A’s IP address also maps to M’s MAC address:



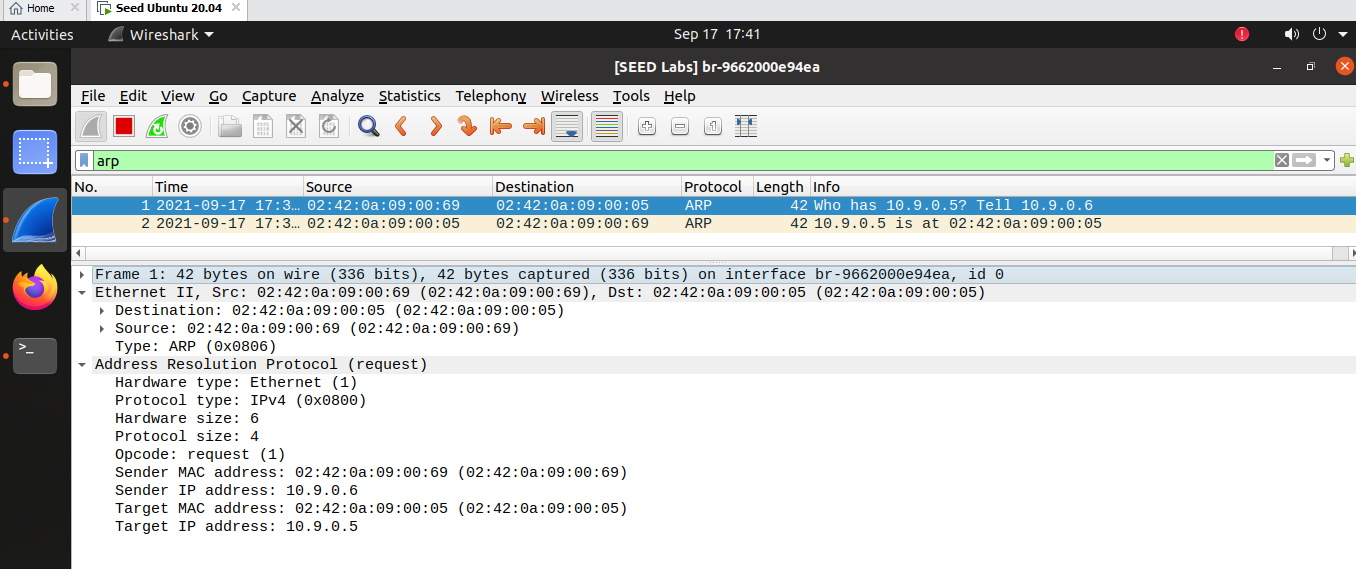


The above code uses the ARP request method to perform ARP Cache Poisoning. The ARP Cache before and after running the code on A and B, respectively, is as follows:



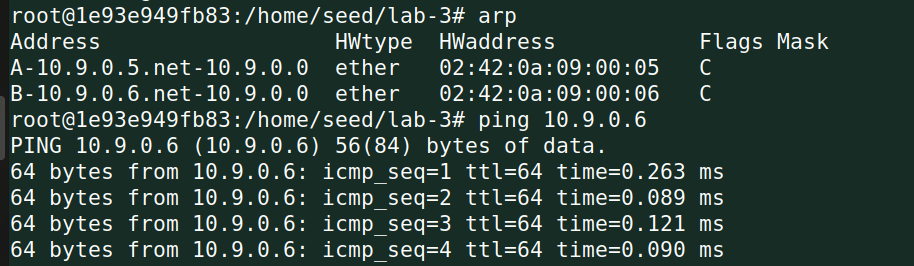


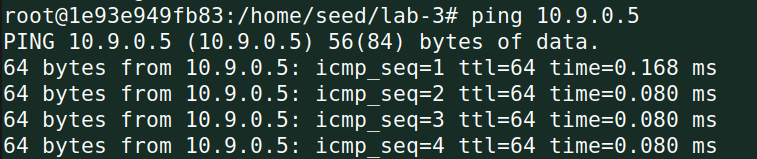
The Wireshark capture show that the ARP request and replies are generated as follows:



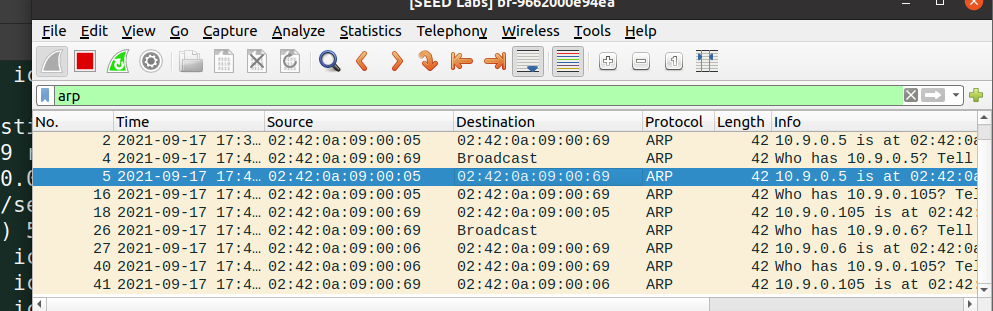
**Step 2 (Testing):**

After performing the ARP Cache poisoning, we ping from A to B and see the following results:



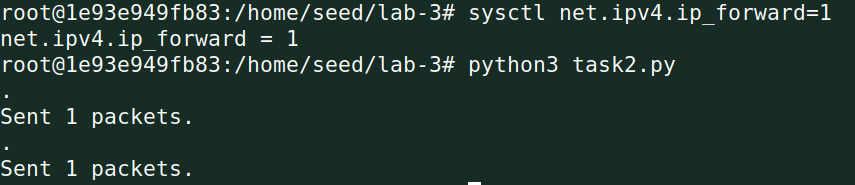


We see that packets are transmitted and are received. The Wireshark capture is as follows:

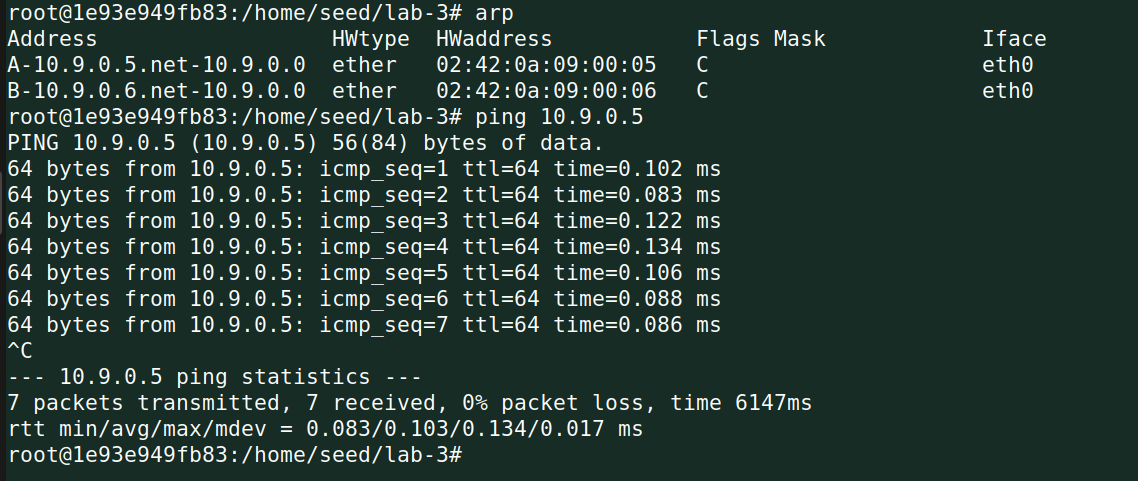


**Step 3 (Turn on IP forwarding):**

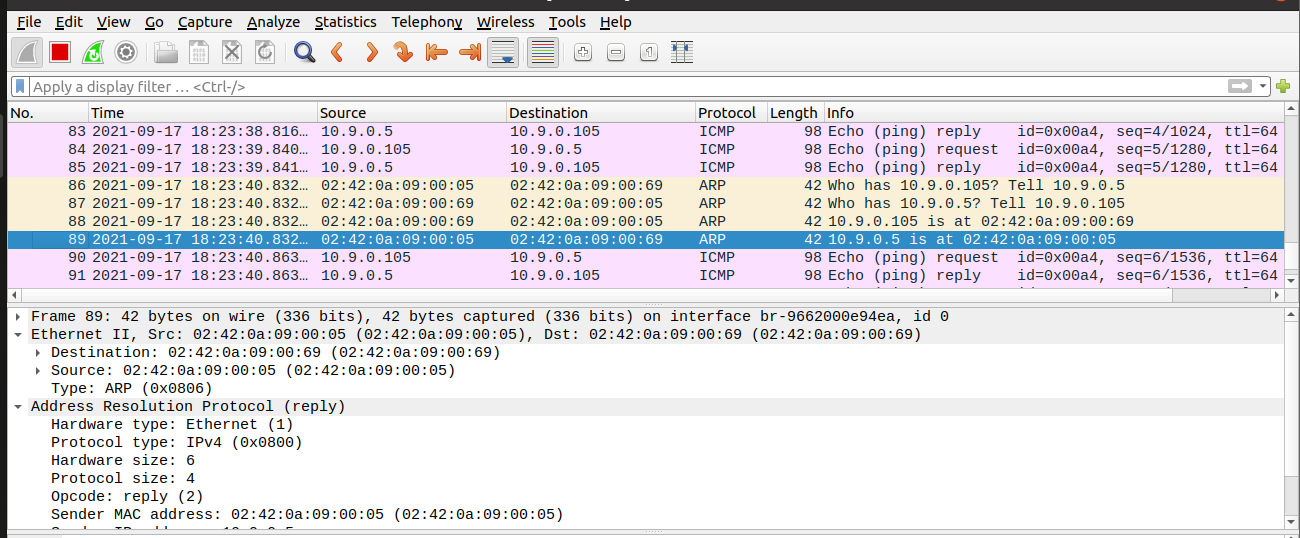
We turn on IP forwarding and perform the attack again:



We ping B from A and see that our ping is successful:



The following show the Wireshark capture of the ping:

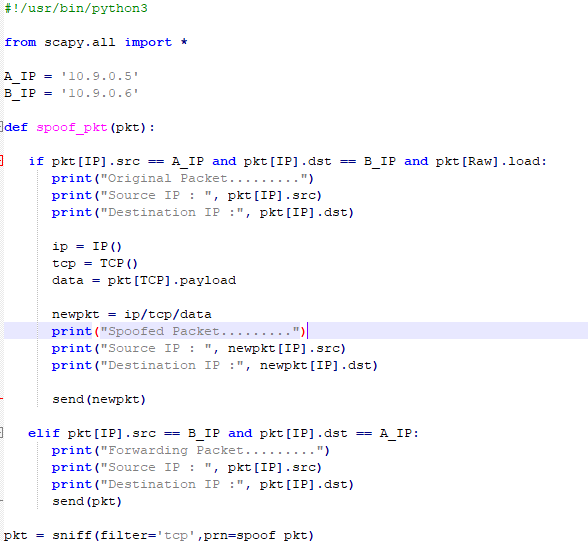


The above shows that the ping request from A to B causes an ICMP redirect message from M to A.

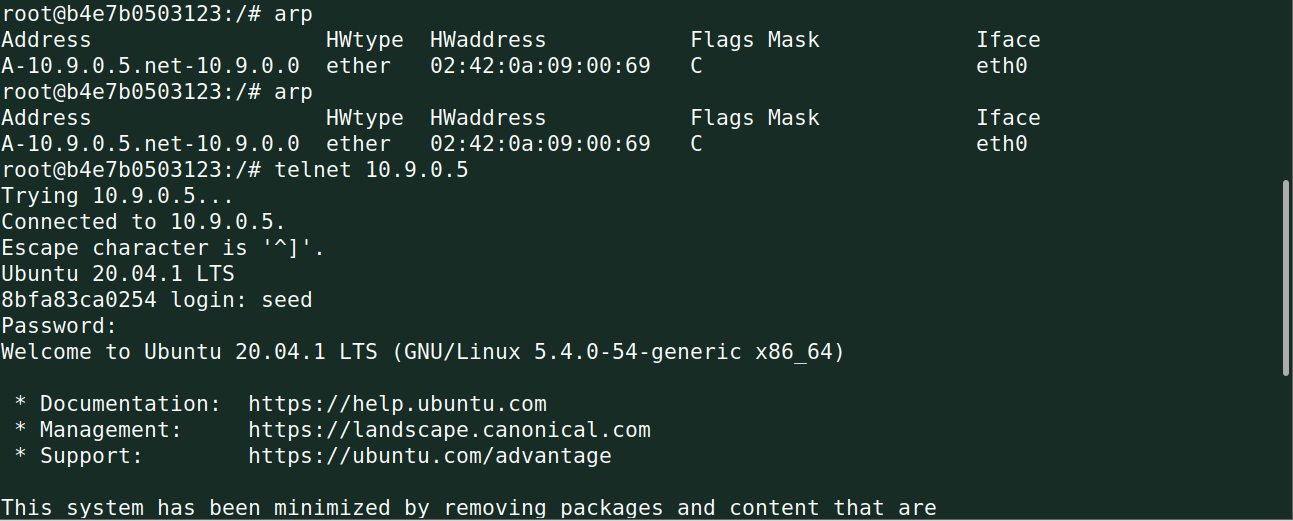
**Step 4 (Launch the MITM attack).**

We first perform the ARP cache poisoning using the same code as before – in Task 2.1. 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 so that we can manipulate the packet. In order to change the contents of the packet, we use the sniffing and spoofing approach, and the above is the code for the same.

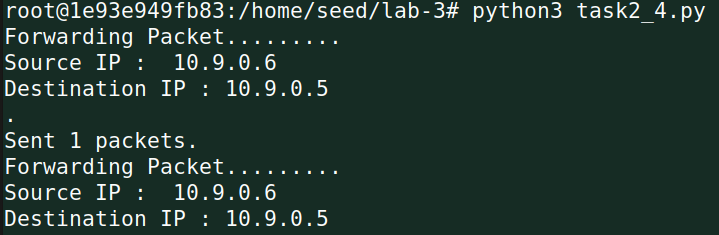
The following provides the code to launch an MITM attack after ARP Cache Poisoning on Telnet session:



The following shows the output on Machine B telnetting to Machine A:



We ping B from A and see that our ping is successful:



The following show the Wireshark capture of the ping:

