Assignment 3
ARP Cache Poisoning Attack Lab

Name:Nikhil T M

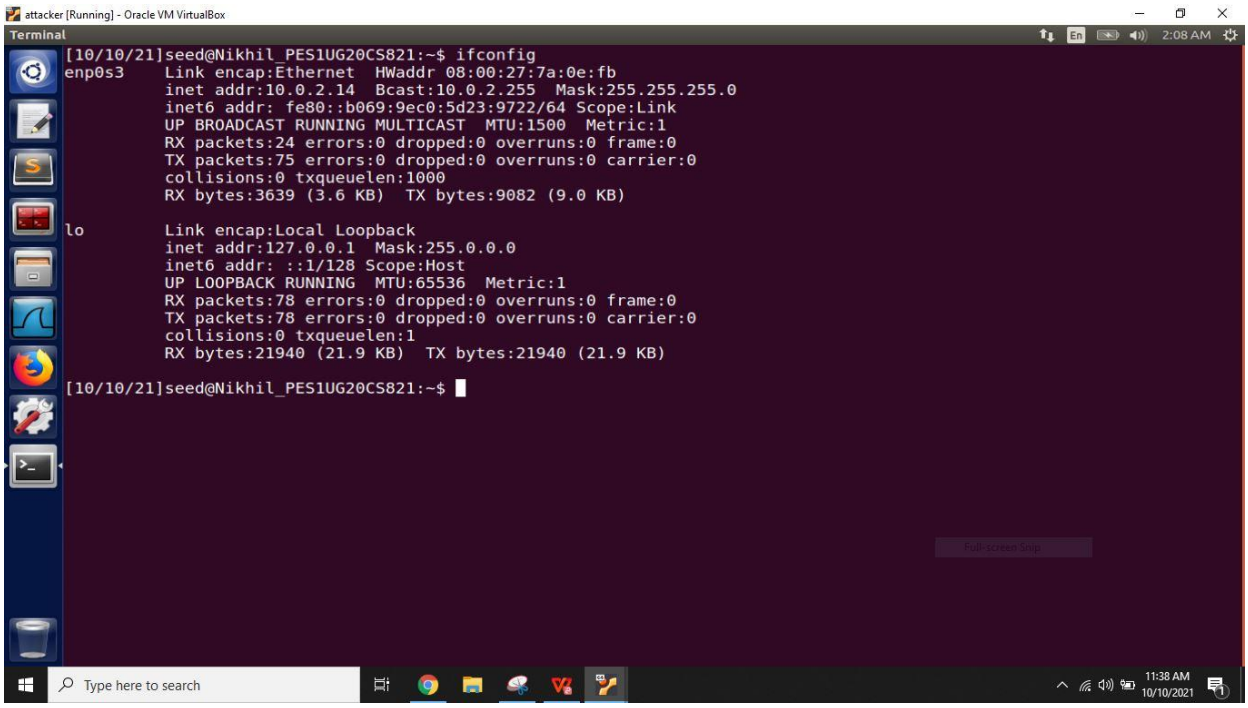
SRN:PES1UG20CS821

Section:F

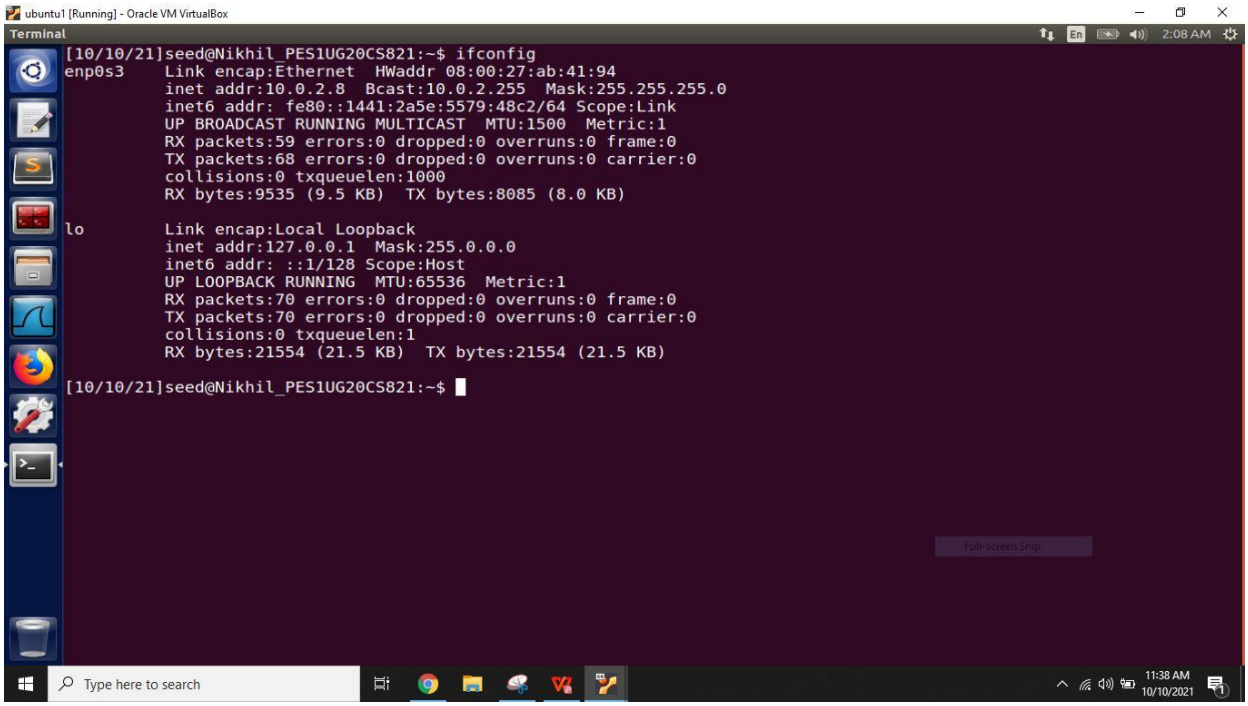
Lab Set Up:

Virtual Machine	IP Address	MAC Address
Attacker	10.0.2.14	08:00:27:7a:0e:fb
VM 1(Ubuntu 1)	10.0.2.8	08:00:27:ab:41:94
VM 2(Ubuntu 2)	10.0.2.9	08:00:27:16:1d:52

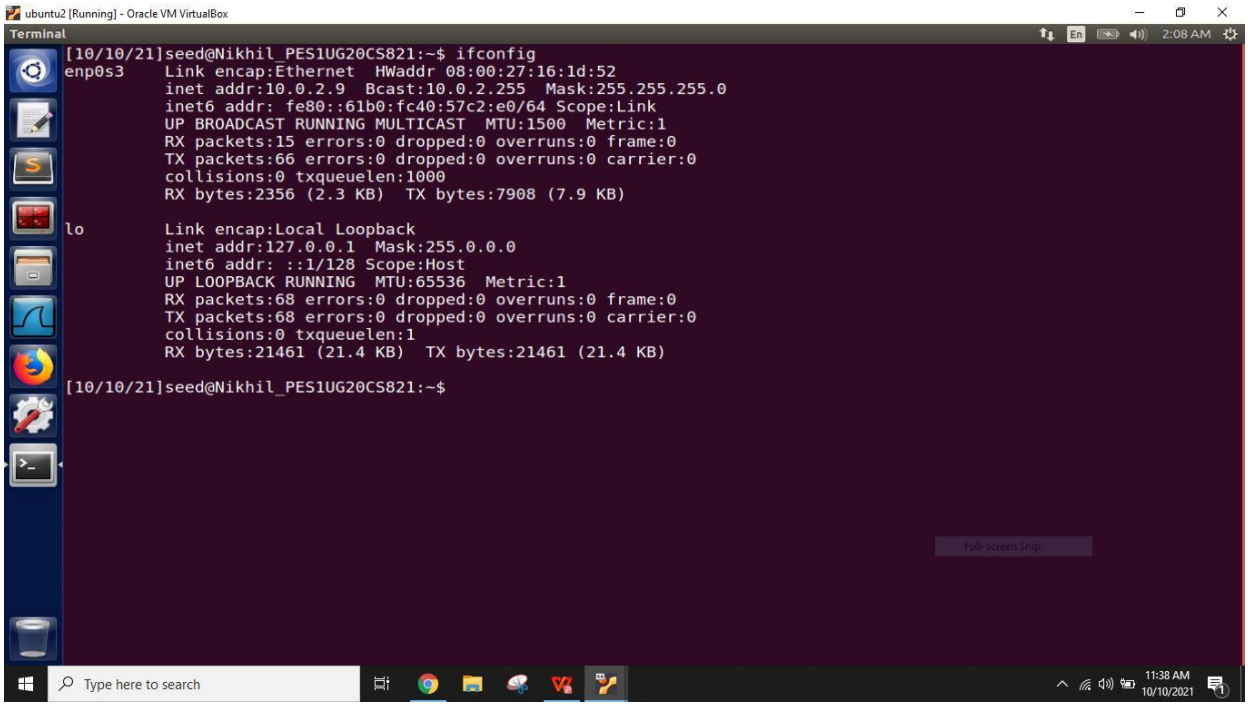
Attacker machine



ubuntu1 machine

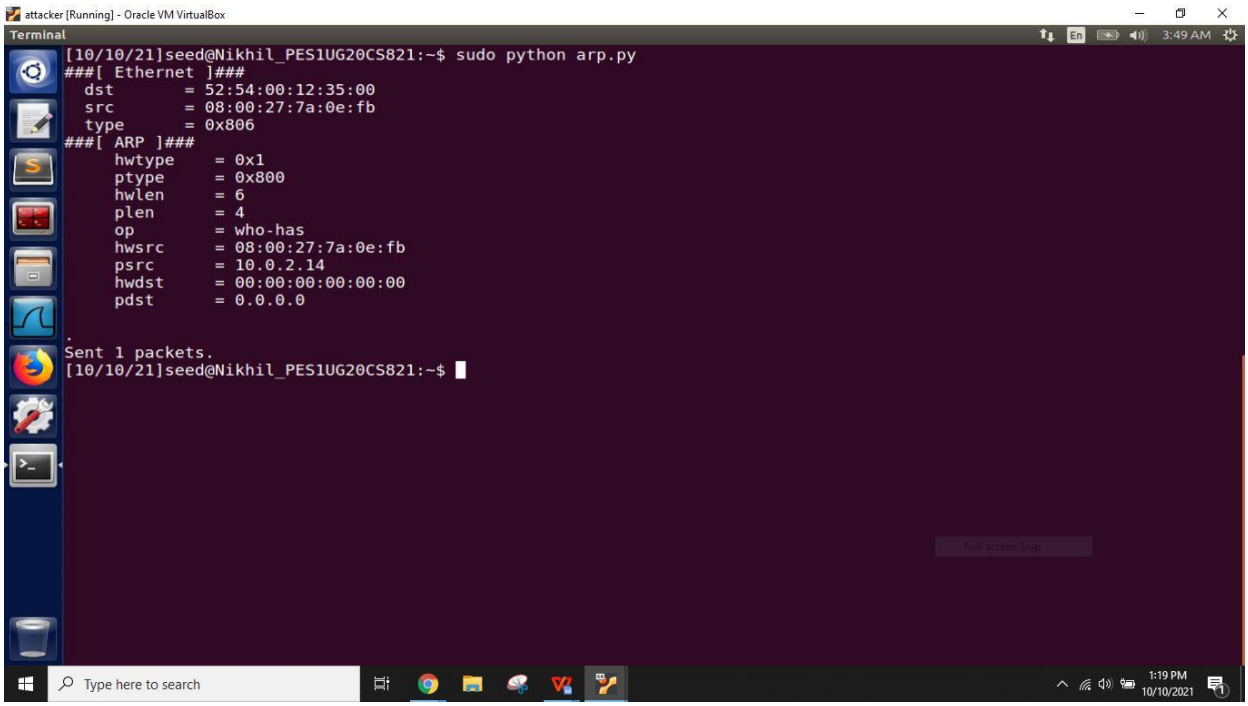


ubuntu2 machine

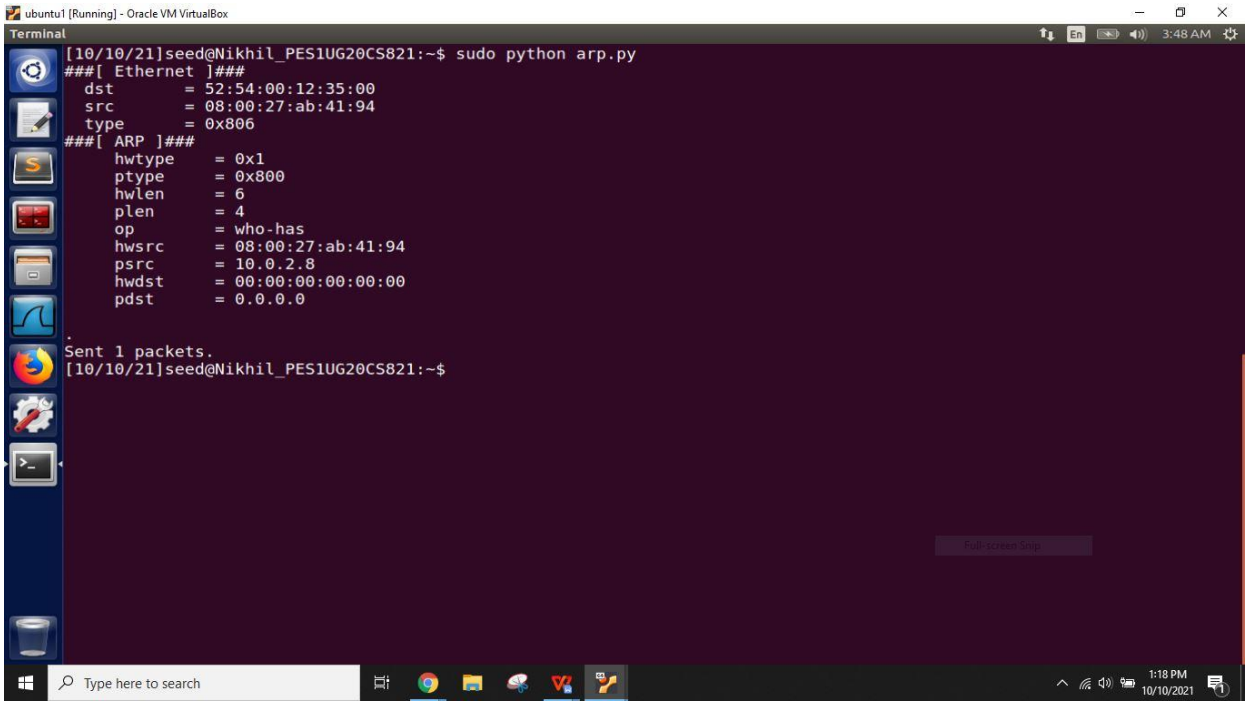


Task 1: ARP Cache Poisoning

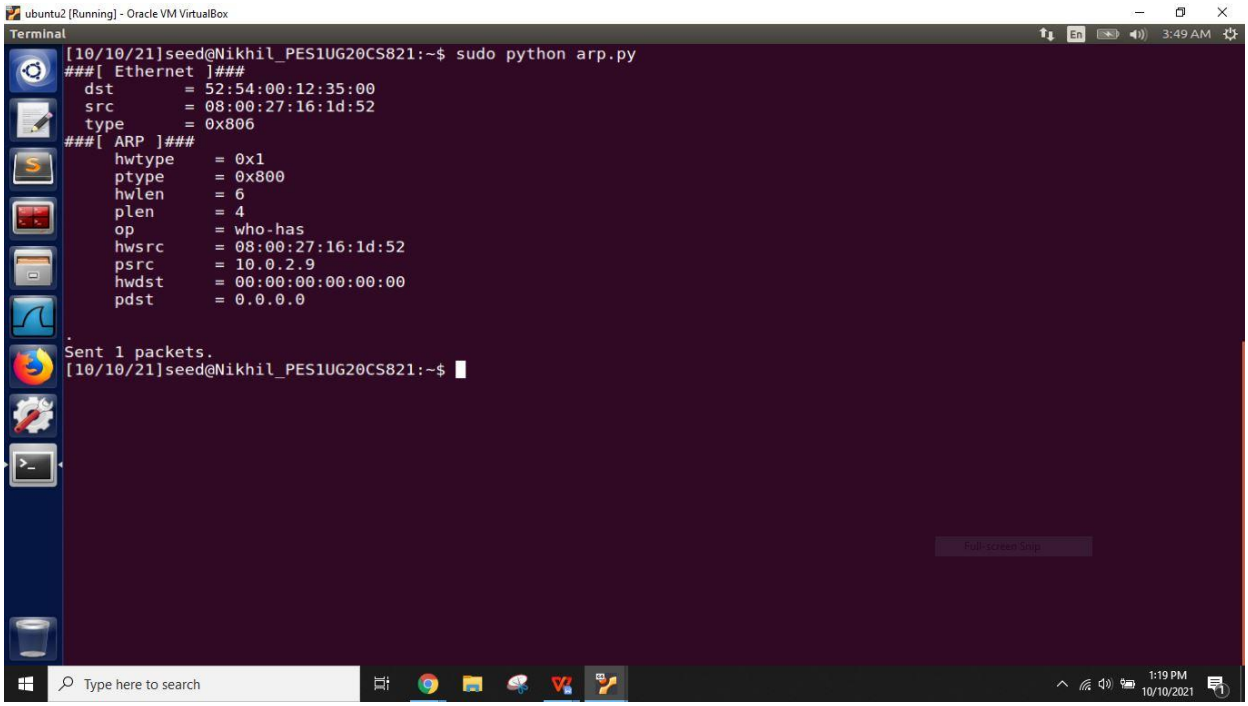
After executing skeleton code to construct ARP packet on Attacker



After executing skeleton code to construct ARP packet on Ubuntu 1



After executing skeleton code to construct ARP packet on Ubuntu 2

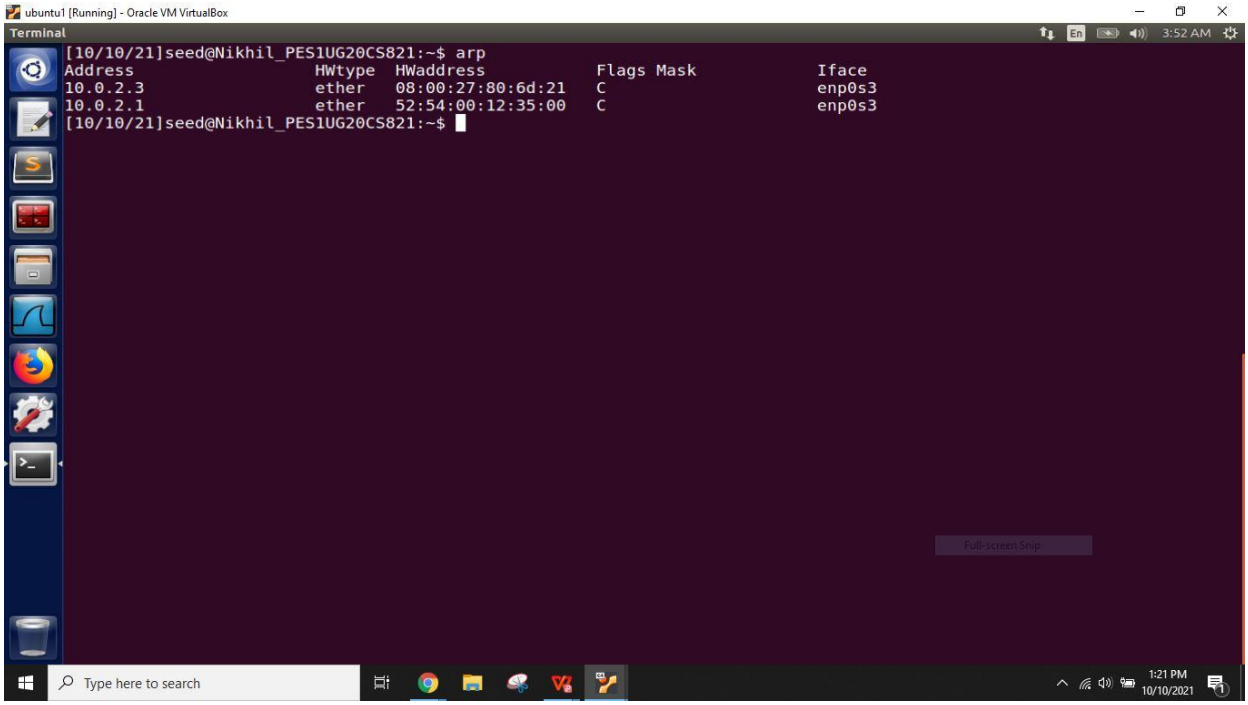


Task 1: ARP Cache Poisoning

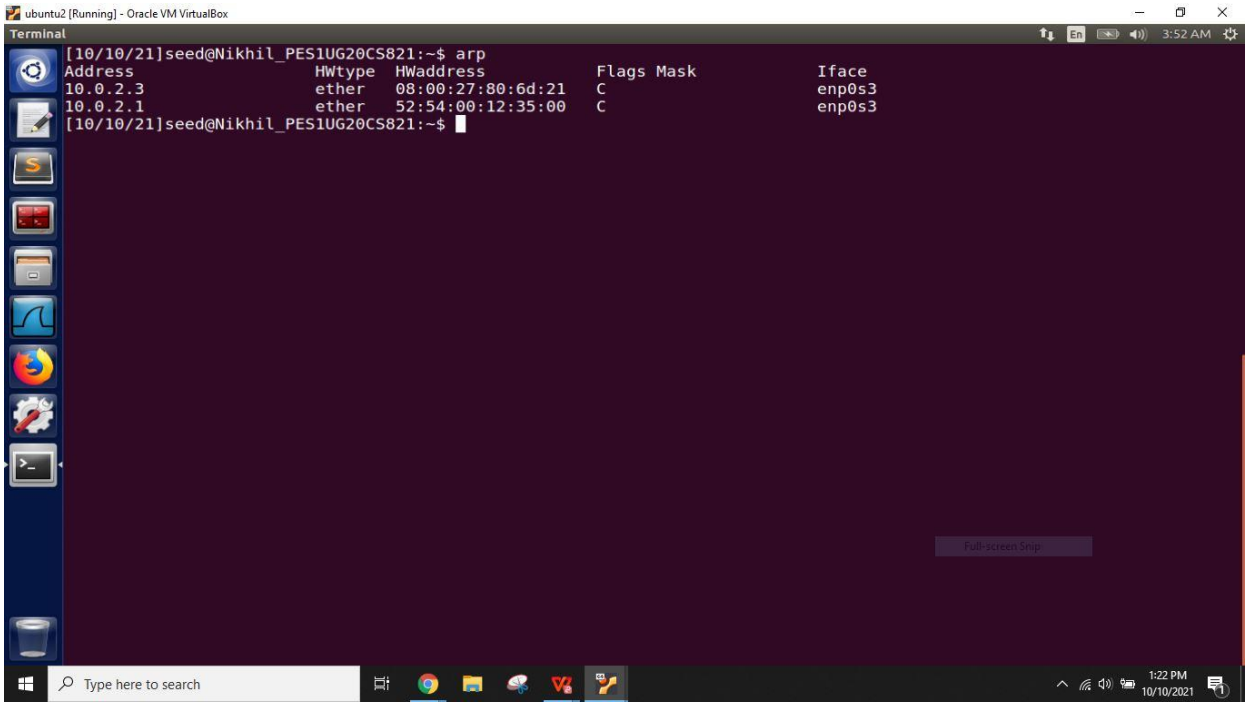
Task1A (using ARP request)

Before attack

ARP table cache in ubuntu1

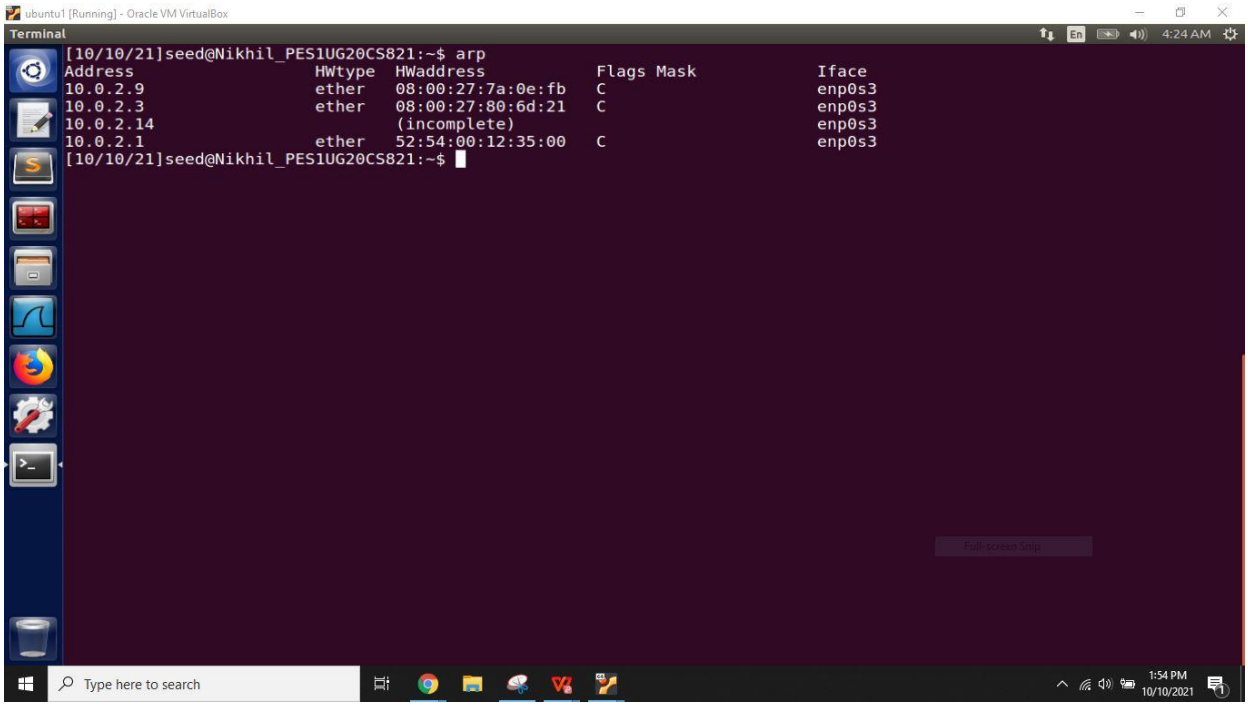


ARP table cache in Ubuntu 2

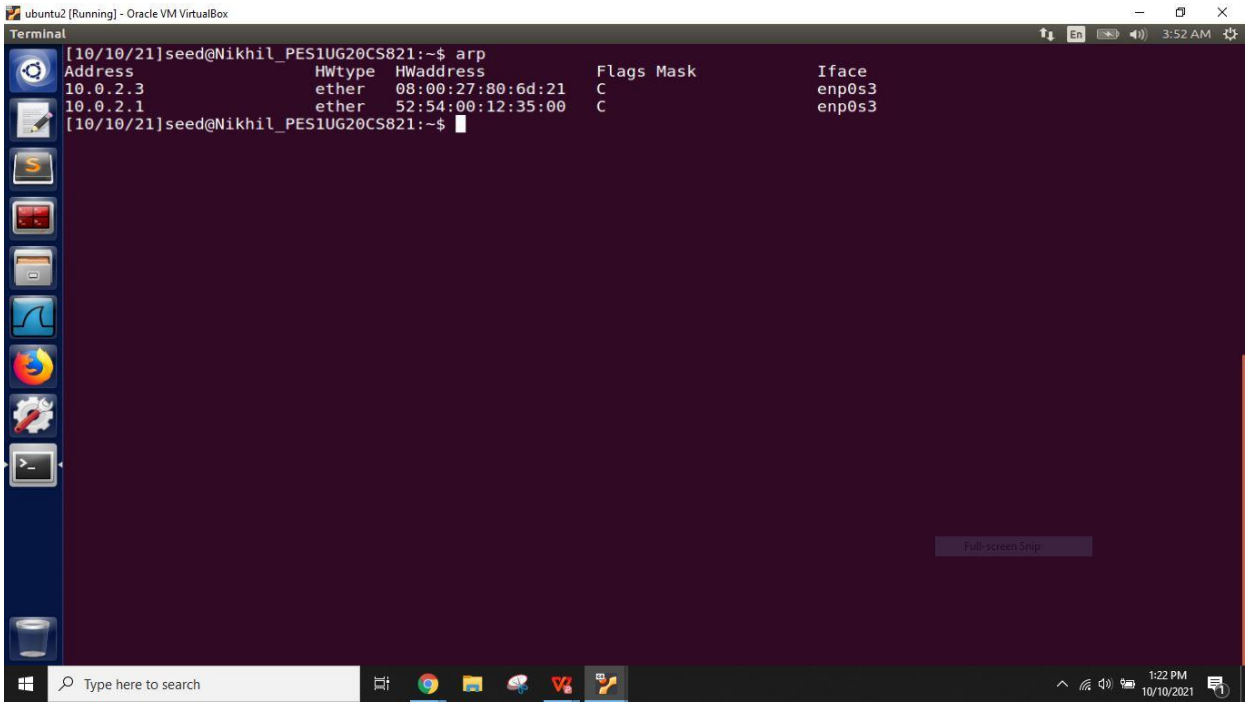


After attack

ARP table cache in ubuntu1



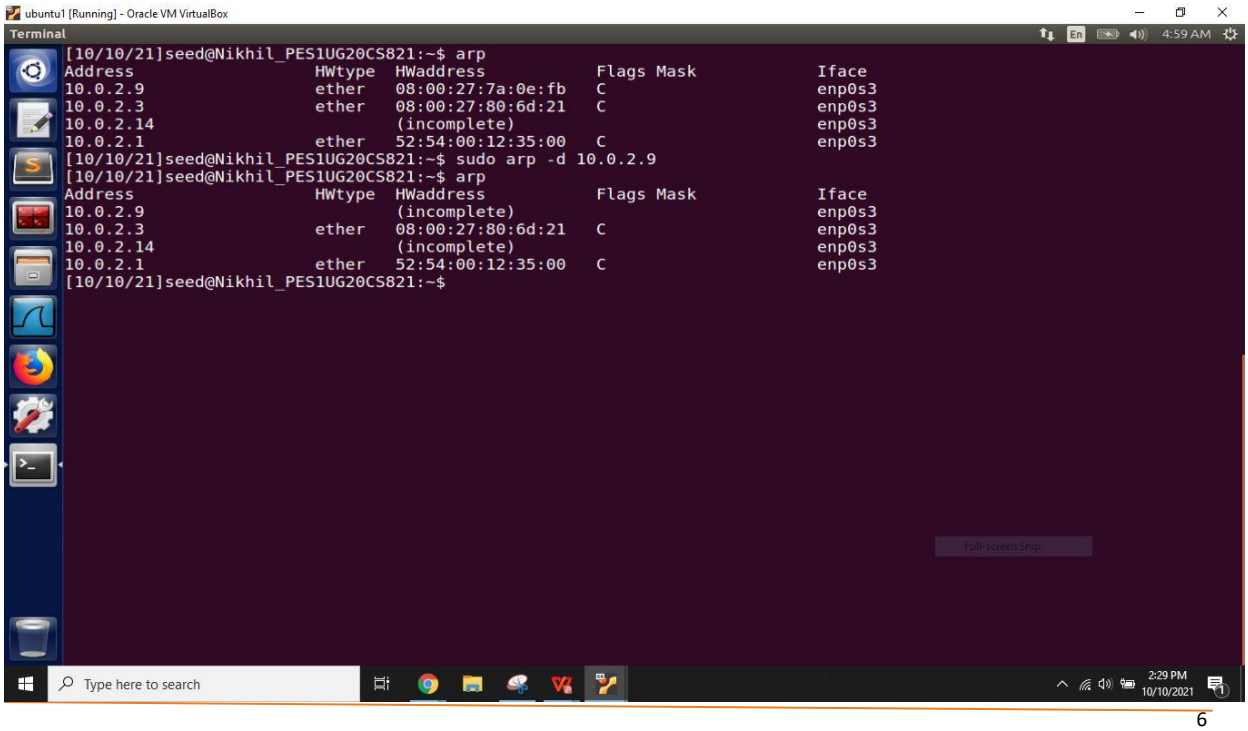
ARP table cache in ubuntu2



Before the attack we can see that there are 2 entries in the Ubuntu 1 and Ubuntu 2 in the ARP cache. after the attack we can see that the new entry adds in the ARP cache in ubuntu1 which have IP address and MAC address of the ubuntu2

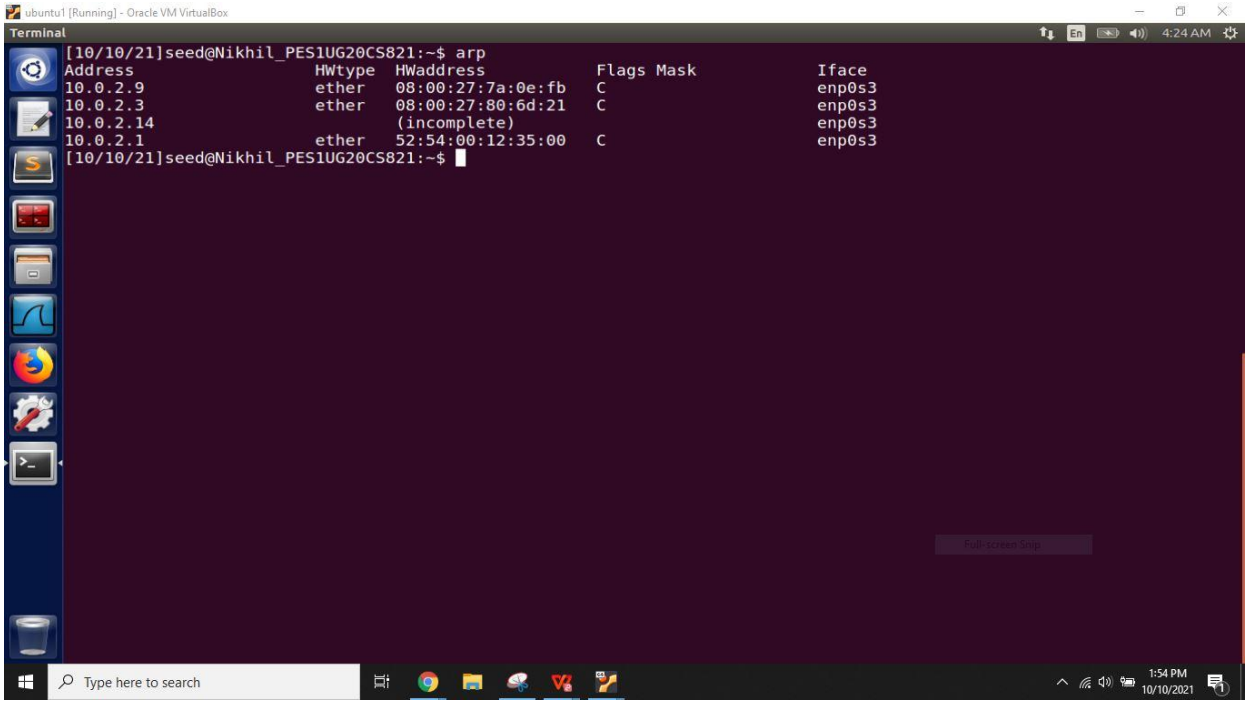
After deleting

sudo arp -d 10.0.2.9

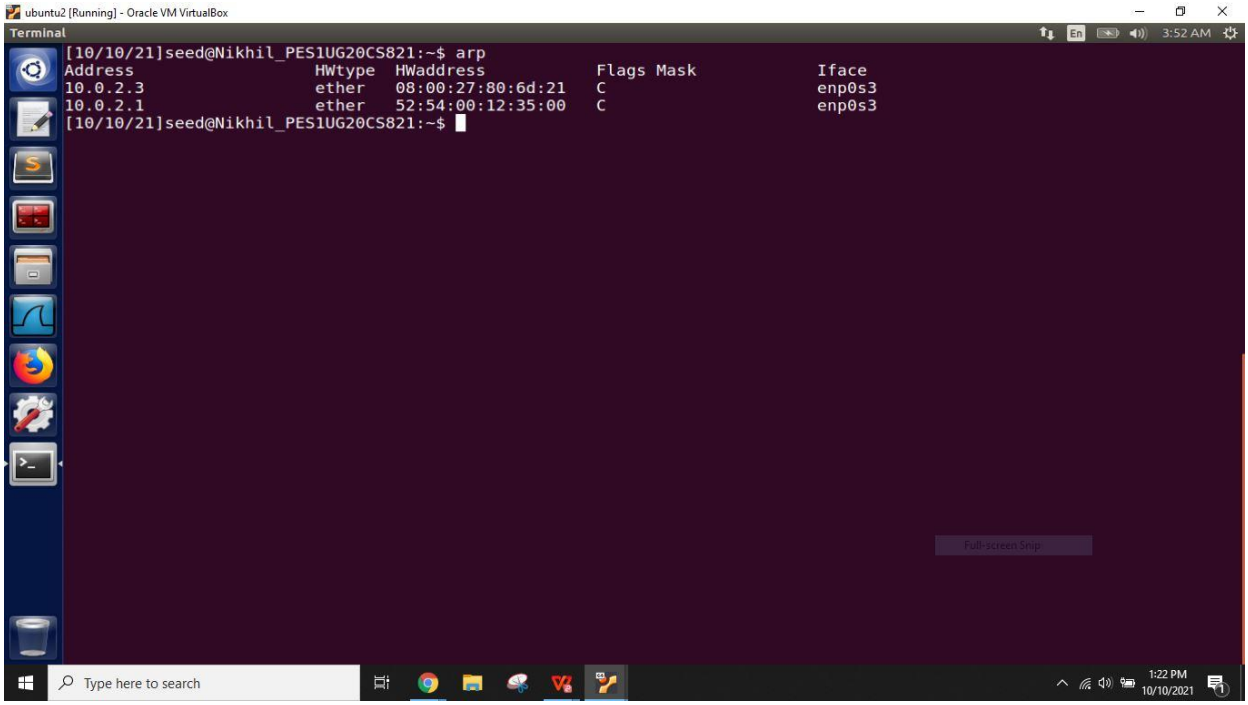


Before attack

ubuntu1

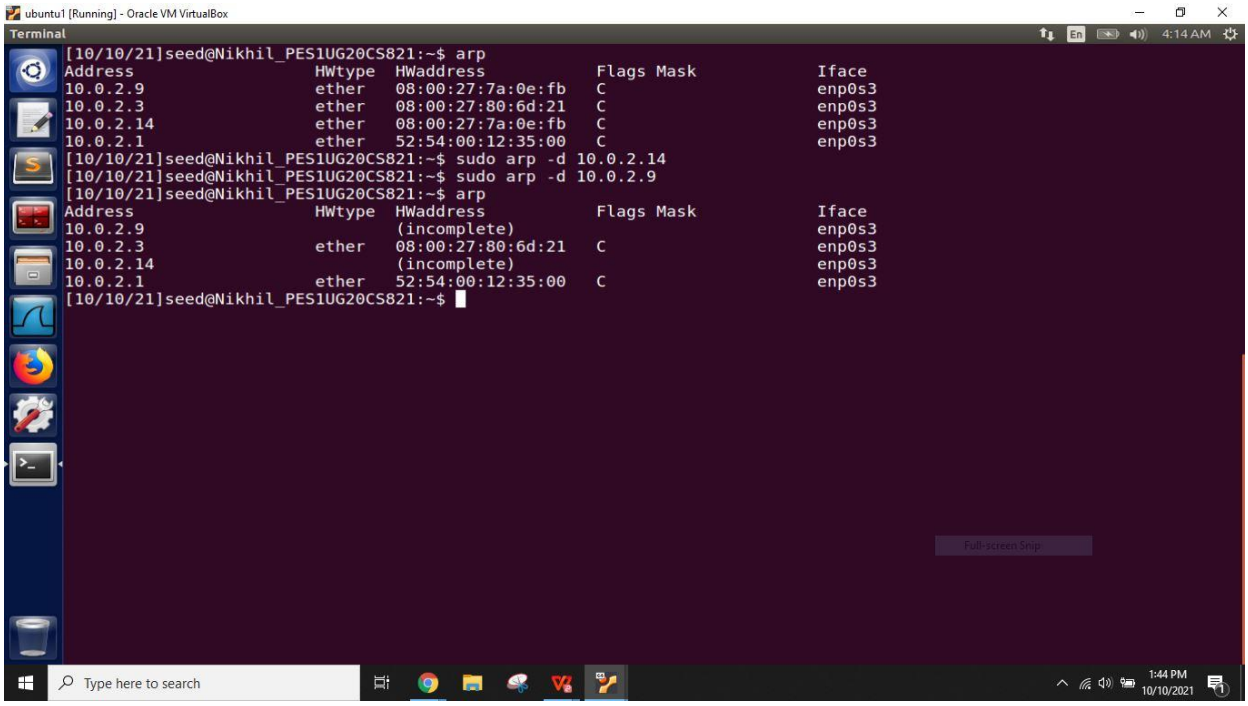


ubuntu2

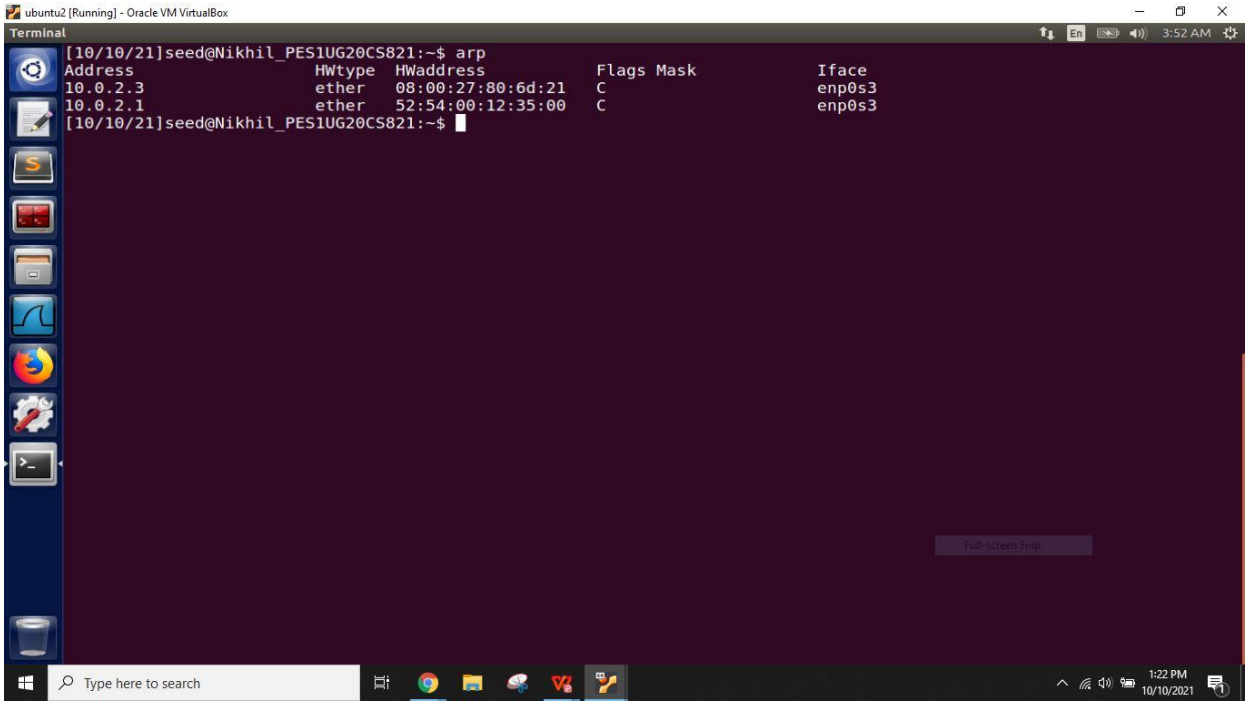


After attack

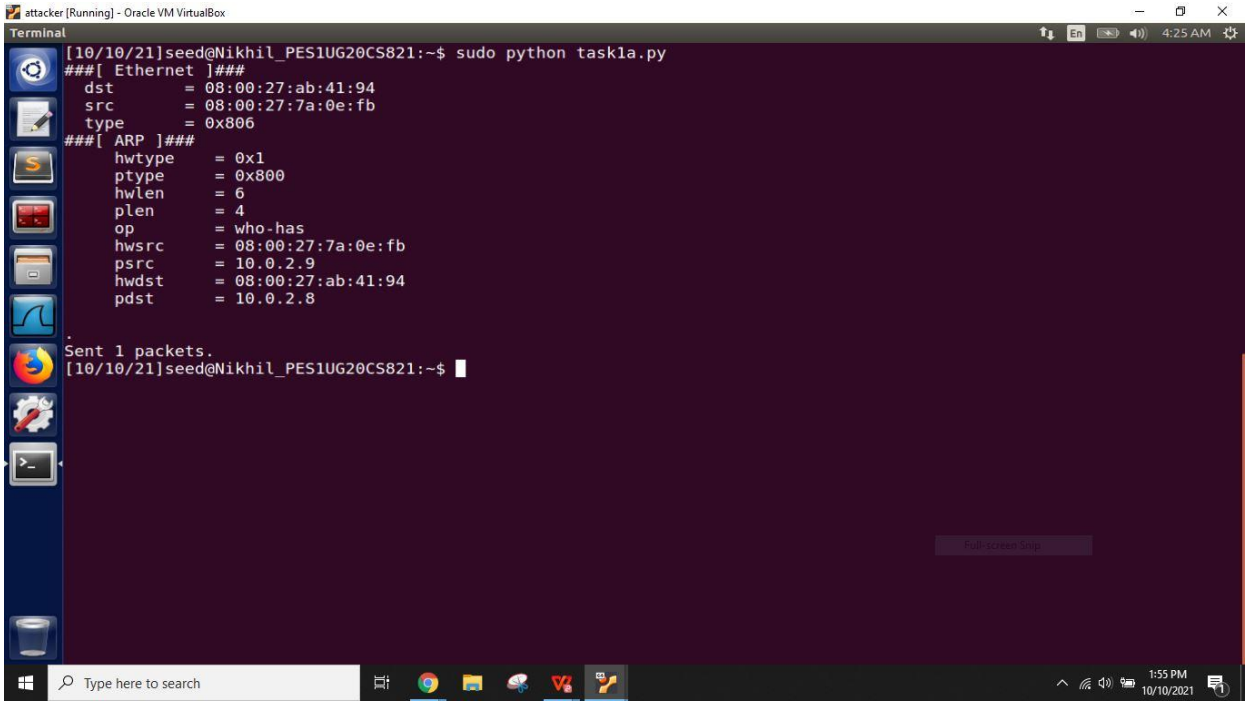
ubuntu1



ubuntu2



Attacker



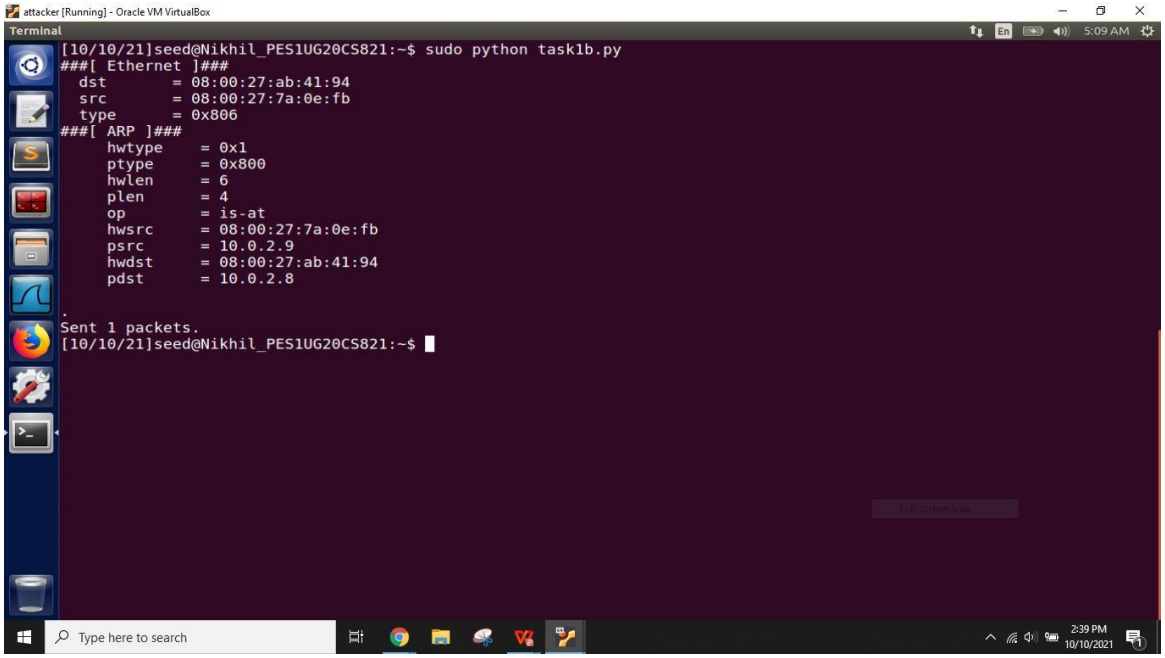
After running the code in the task1a.py on the attacker machine the ARP cache of attacker and ubuntu2 is stored in the ubuntu1 machines arp cache table.

'op' is the length of the logical address in bytes; it specifies the nature of the ARP message. An ARP Request has an assigned value of 1, whereas the ARP reply holds the value of 2.

In the approach 1 ubuntu2 arp cache is stored in the ubuntu1 arp table whereas in approach 2 both the attacker machine and ubuntu2 machine arp cache is stored in the arp table of ubuntu1 also in approach 1 and 2 mac address of ubuntu2 is spoofed with the attacker mac address

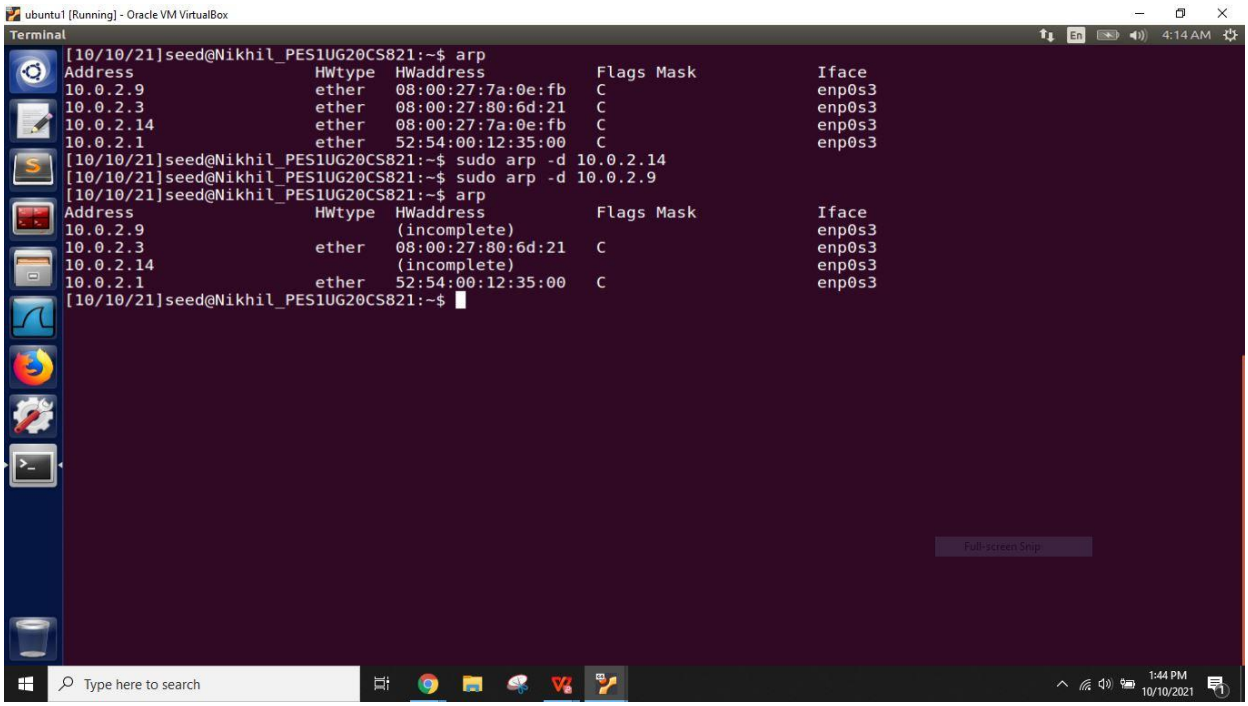
Task 1B (using ARP reply)

attacker

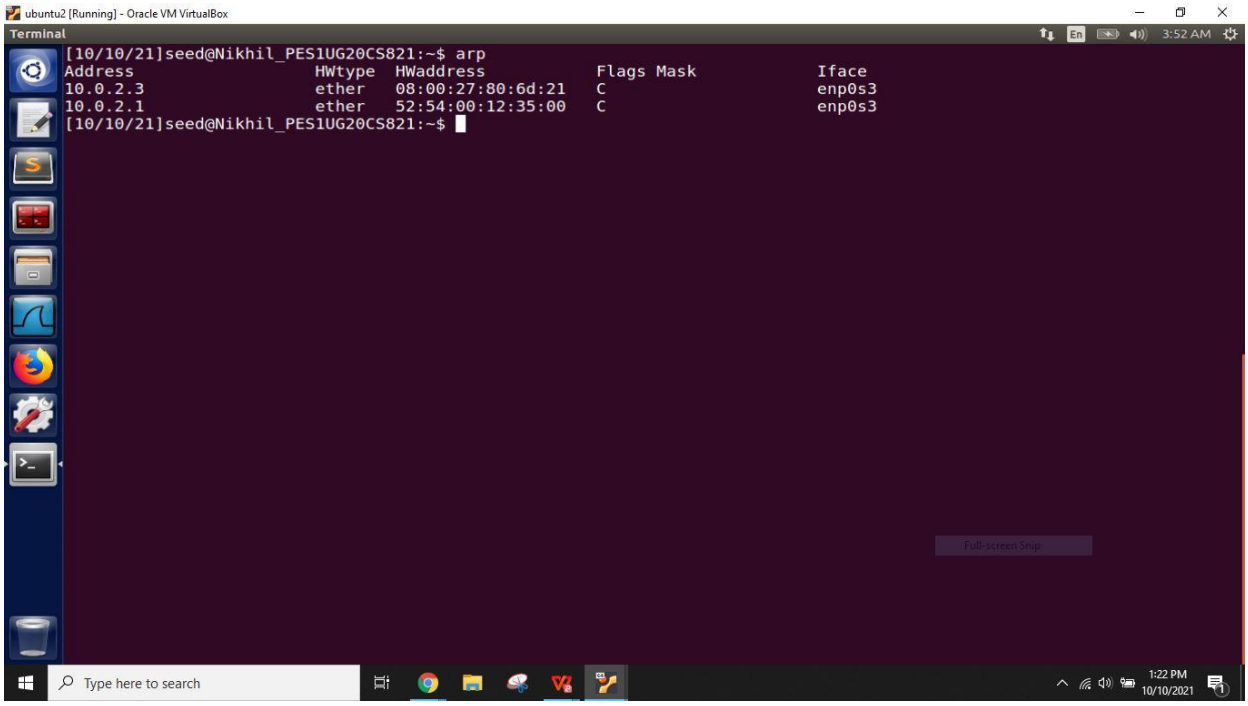


Before attack

ubuntu1

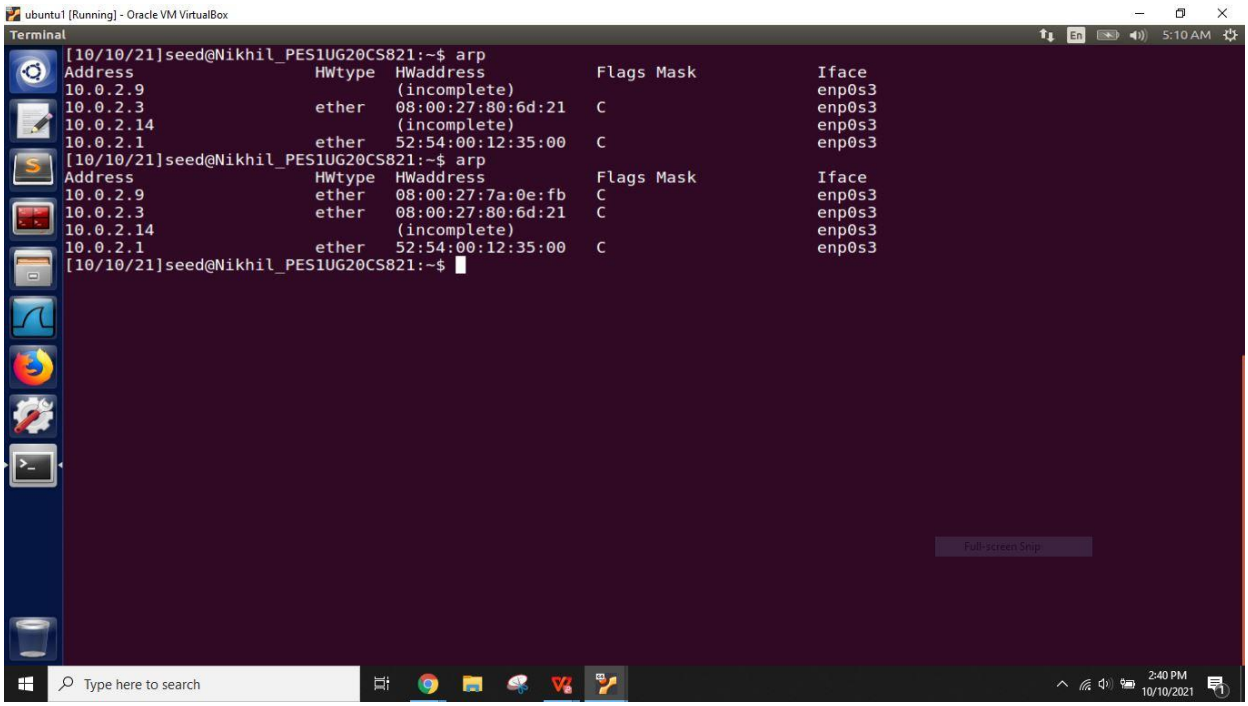


ubuntu2

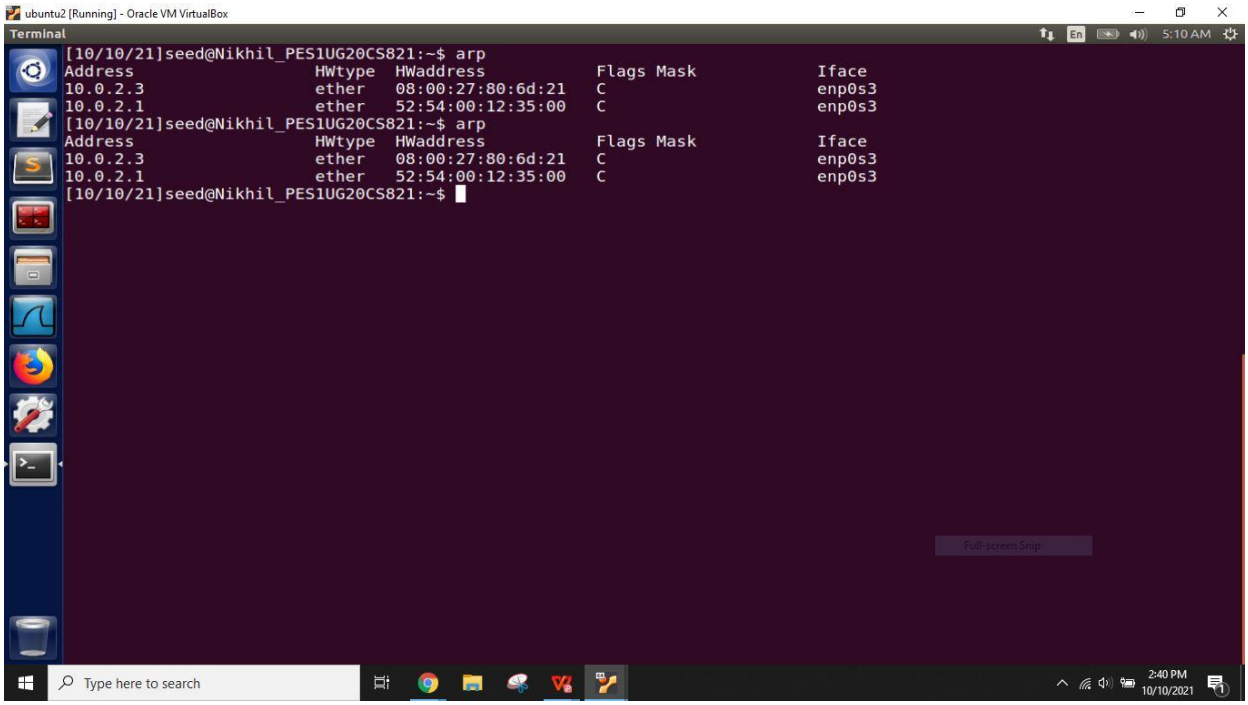


After attack

ubuntu1



ubuntu2

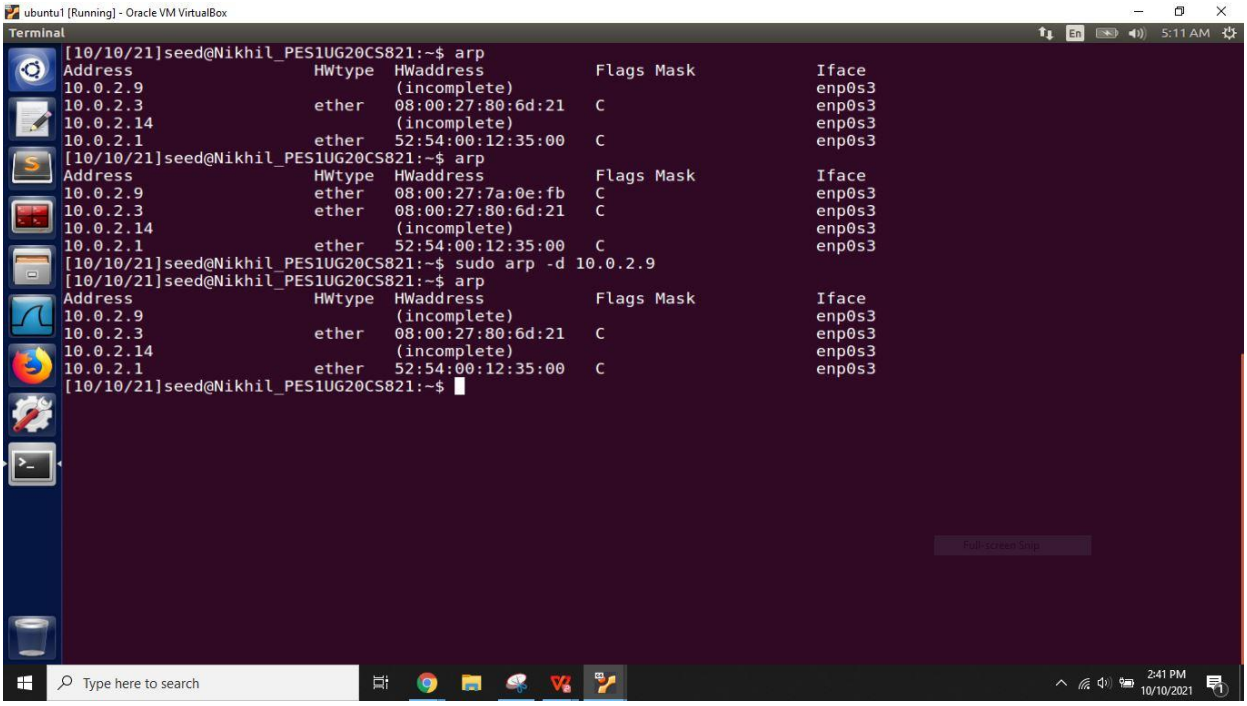


The above task is repeated again but the op code is set to 2.‘op’ is the length of the logical address in bytes; it specifies the nature of the ARP message. The value 2 in the op signify that it is ARP reply.

Deleting the ARP cache

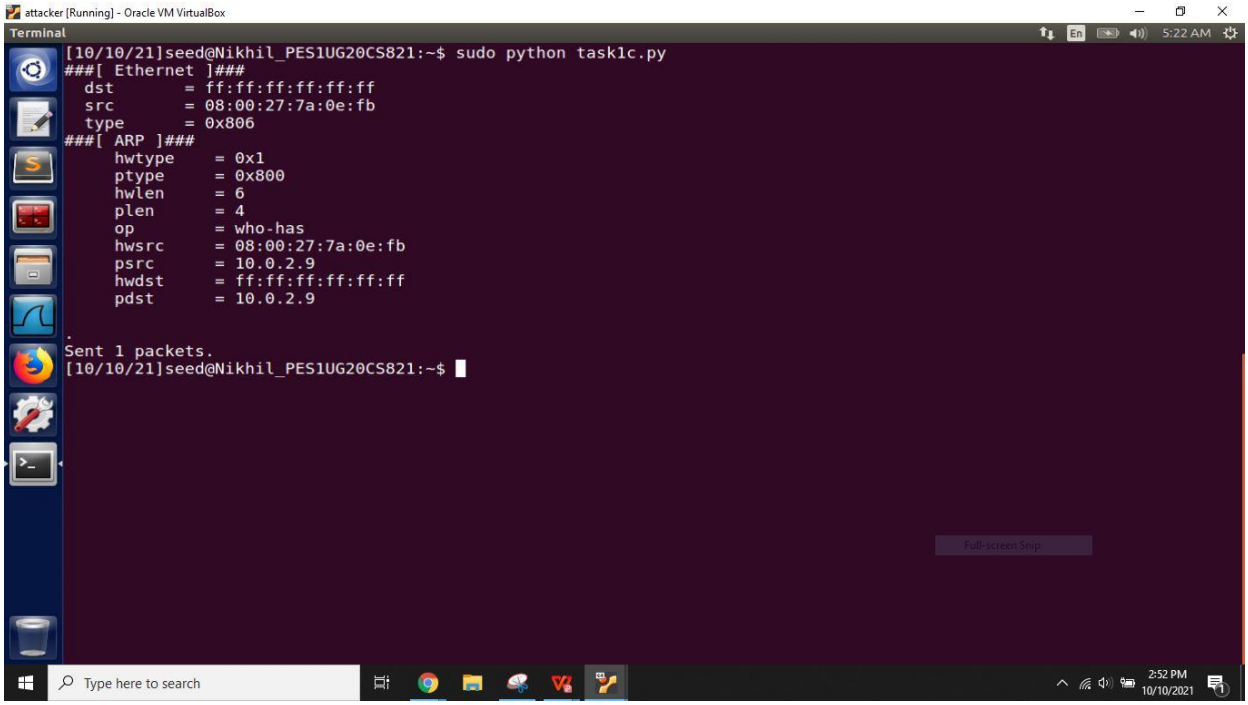
sudo arp -d 10.0.2.9

ubuntu1

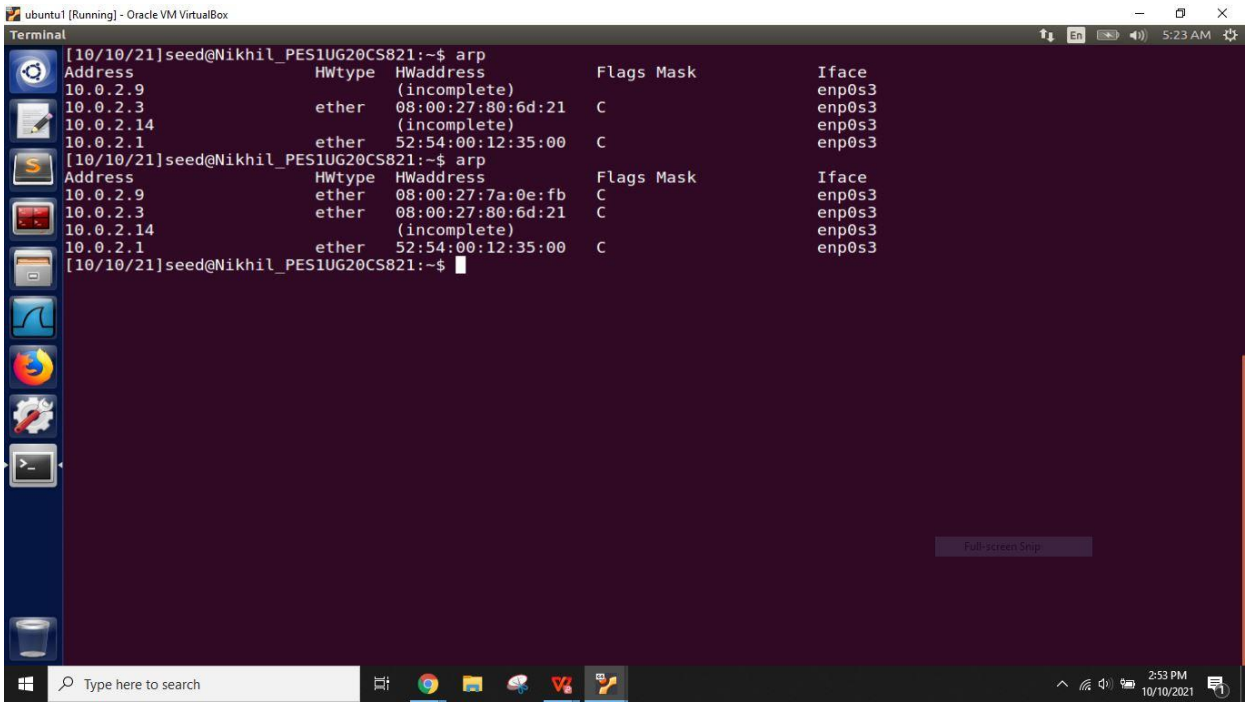


Task 1C (using ARP gratuitous message)

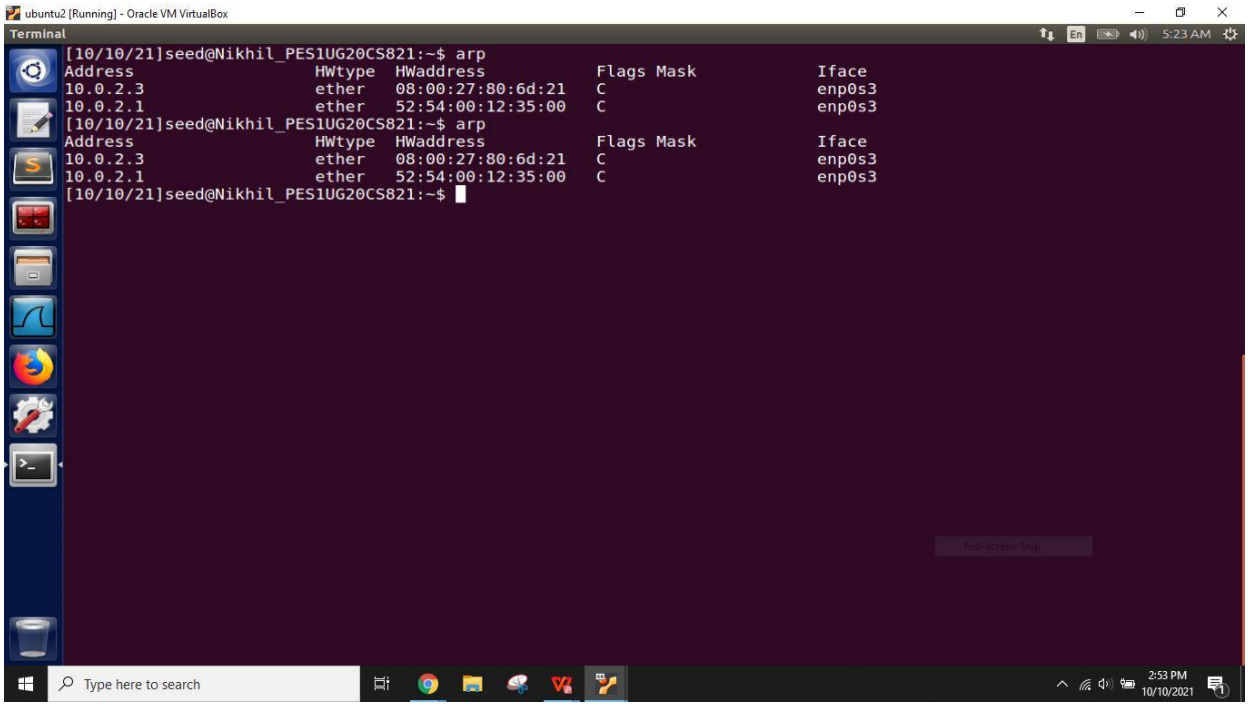
Attacker



ubuntu1 after attack



ubuntu2 after attack

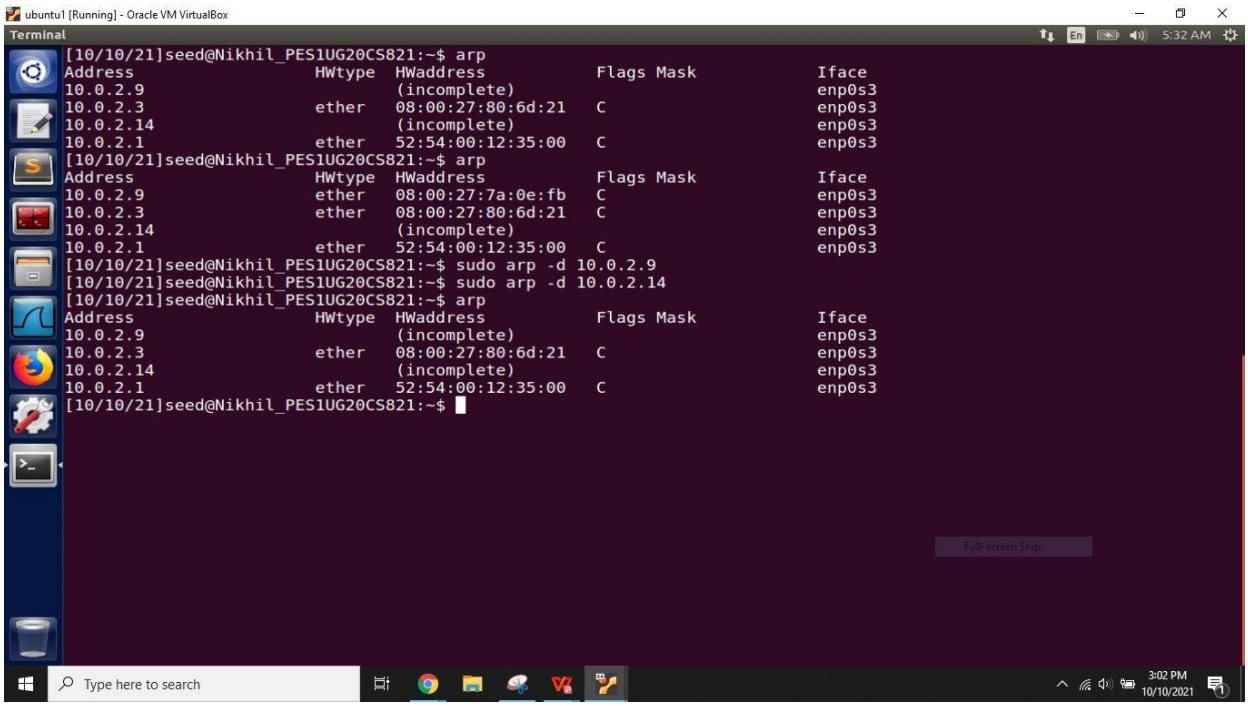


In this approach host machine update outdated information on all the other machine's ARP cache. The destination MAC addresses in both ARP header and Ethernet header are the broadcast MAC address .

Even though the packet is broadcast-ed the arp cache remains unchanged because the attack is done only on the ubuntu1

Yes on ubuntu2 arp cache table we can see for all the 3 approaches the result is same.

After deletion

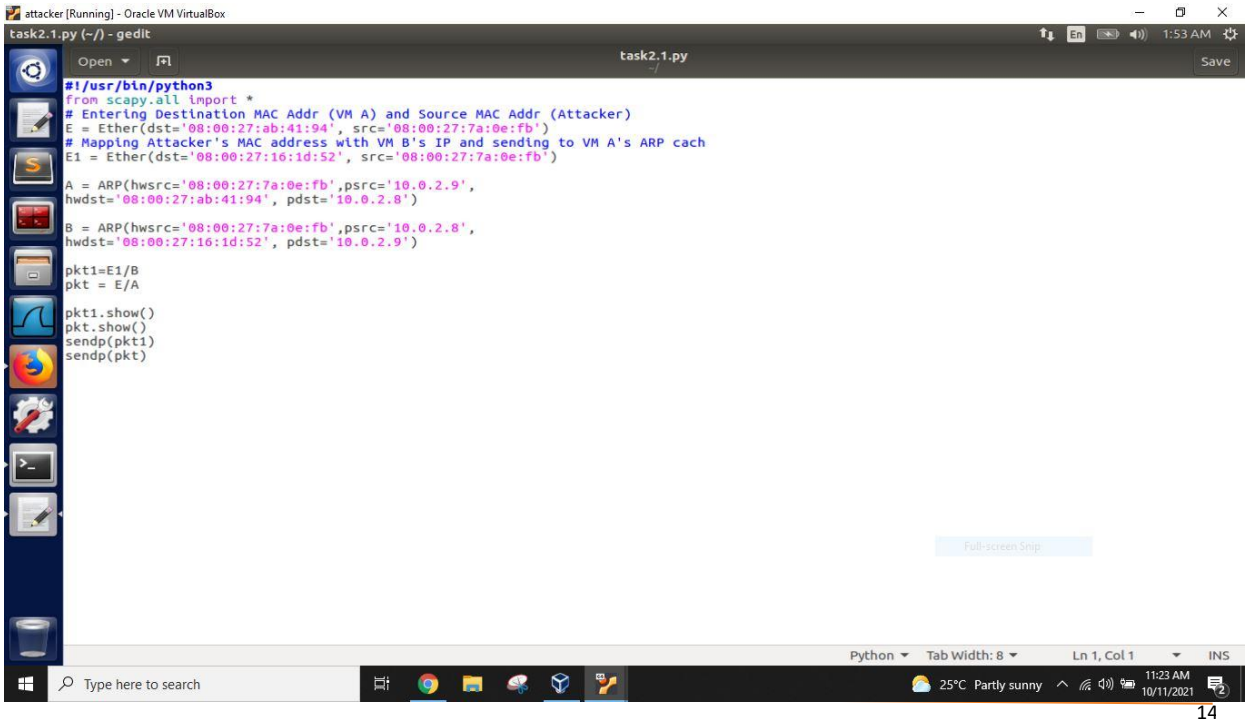


Task 2: MITM Attack on Telnet using ARP Cache Poisoning

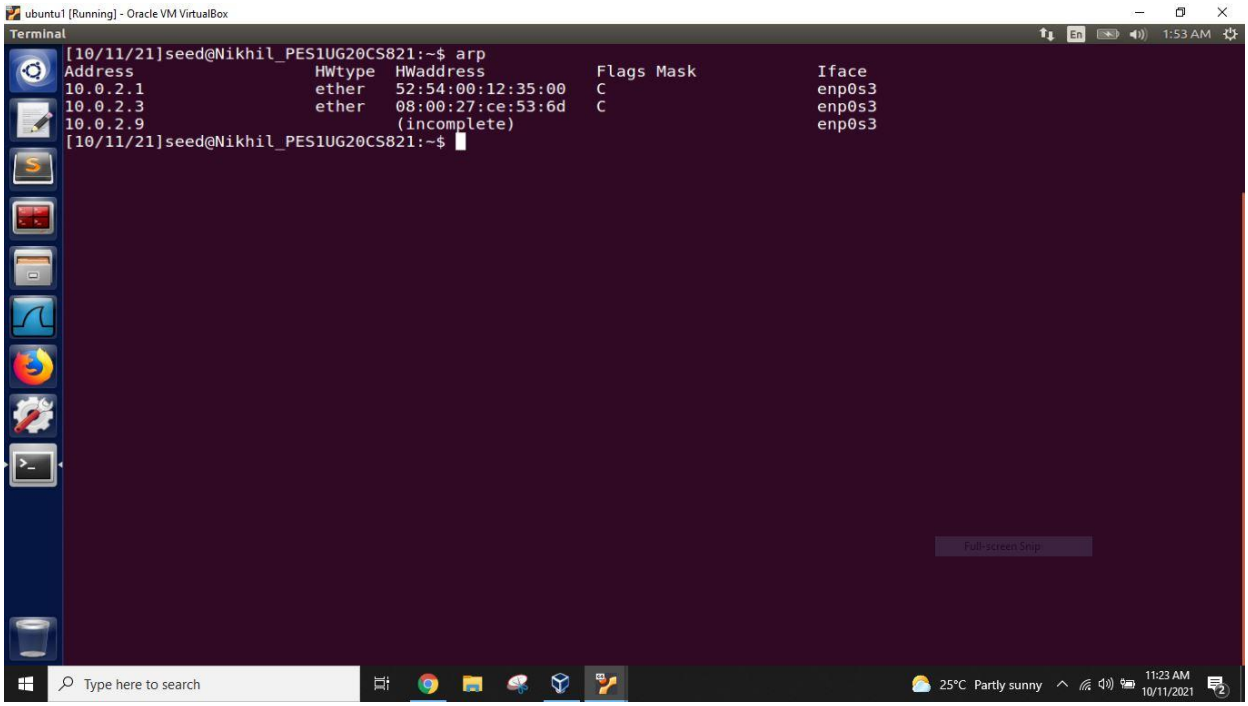
Step 1 (Launch the ARP cache poisoning attack)

Before attack

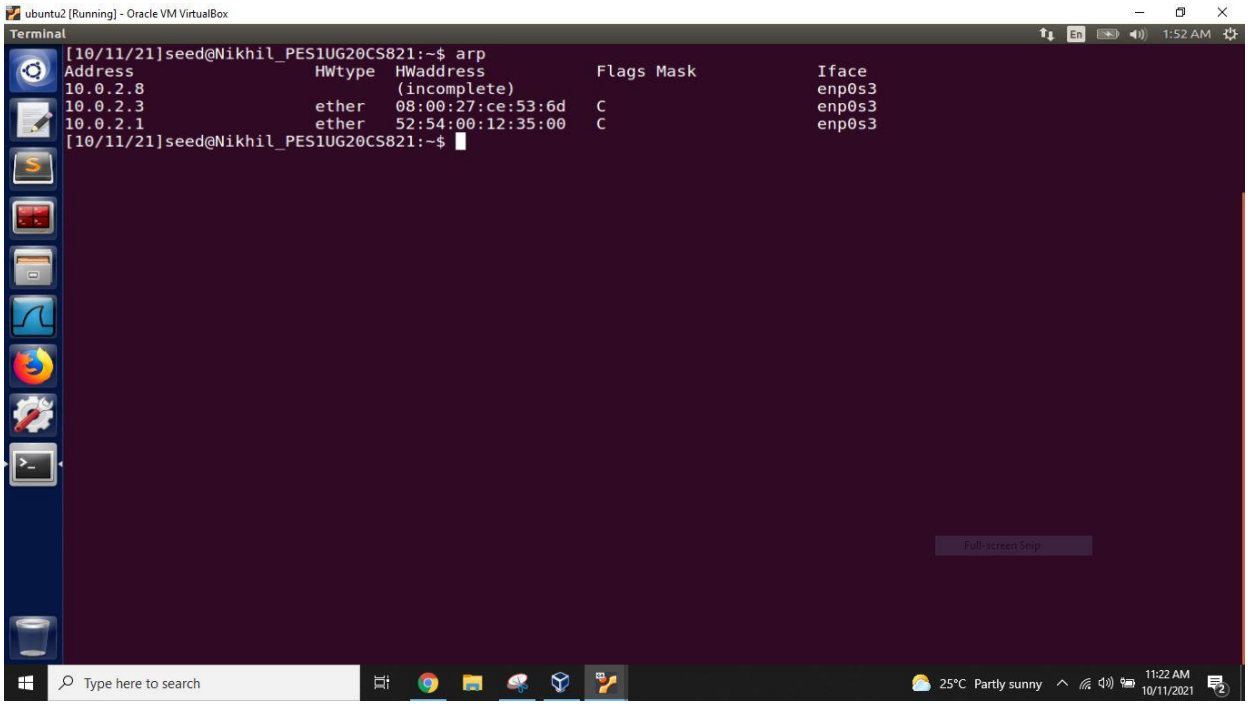
Code in attacker machine



ARP table in ubuntu1

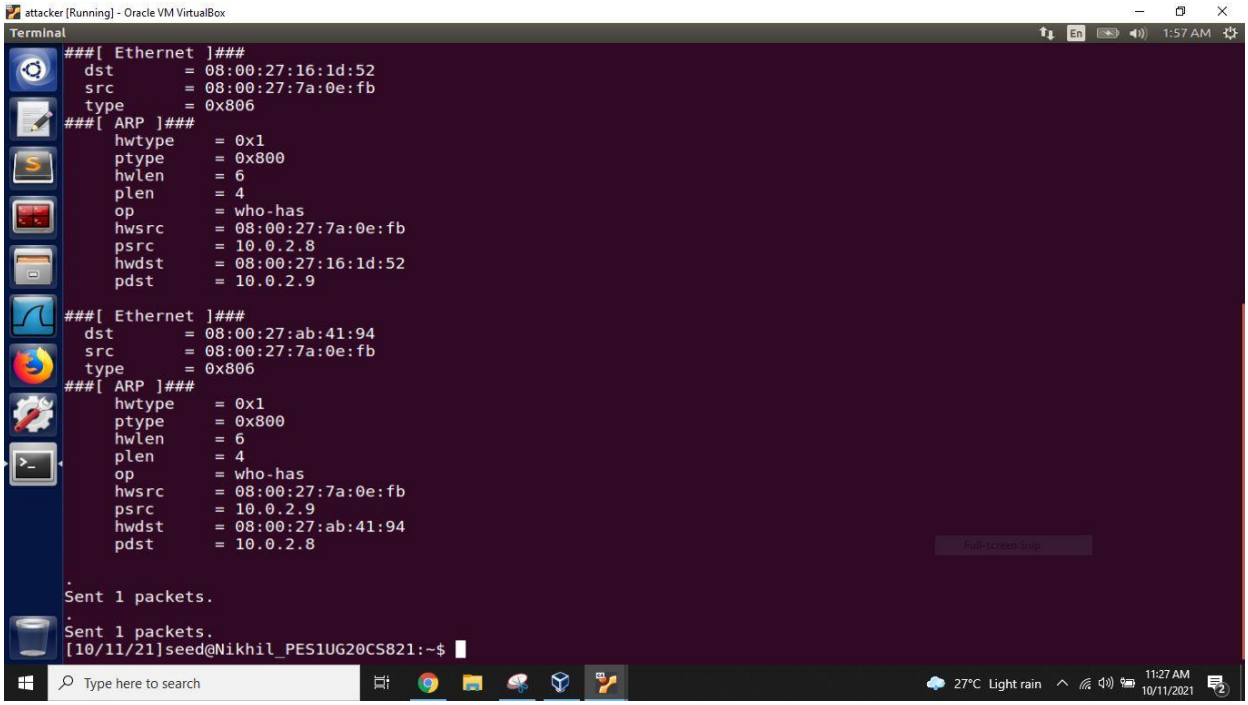


ARP table in ubuntu2

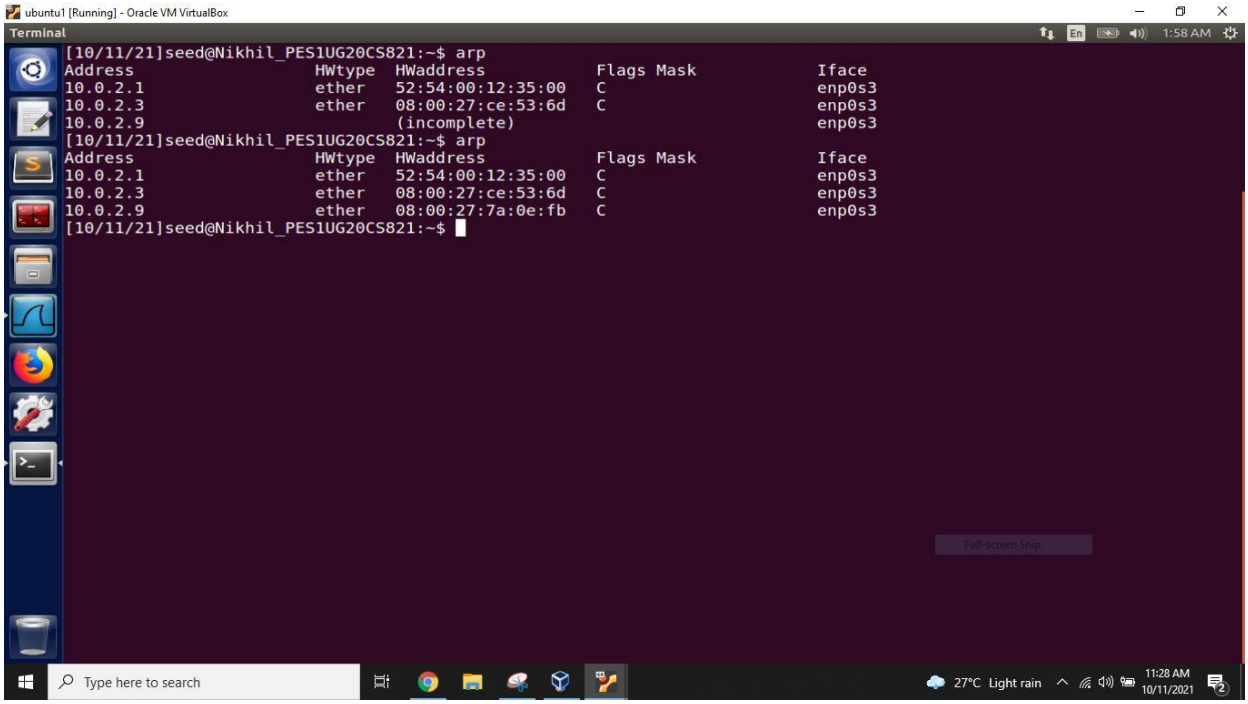


After attack

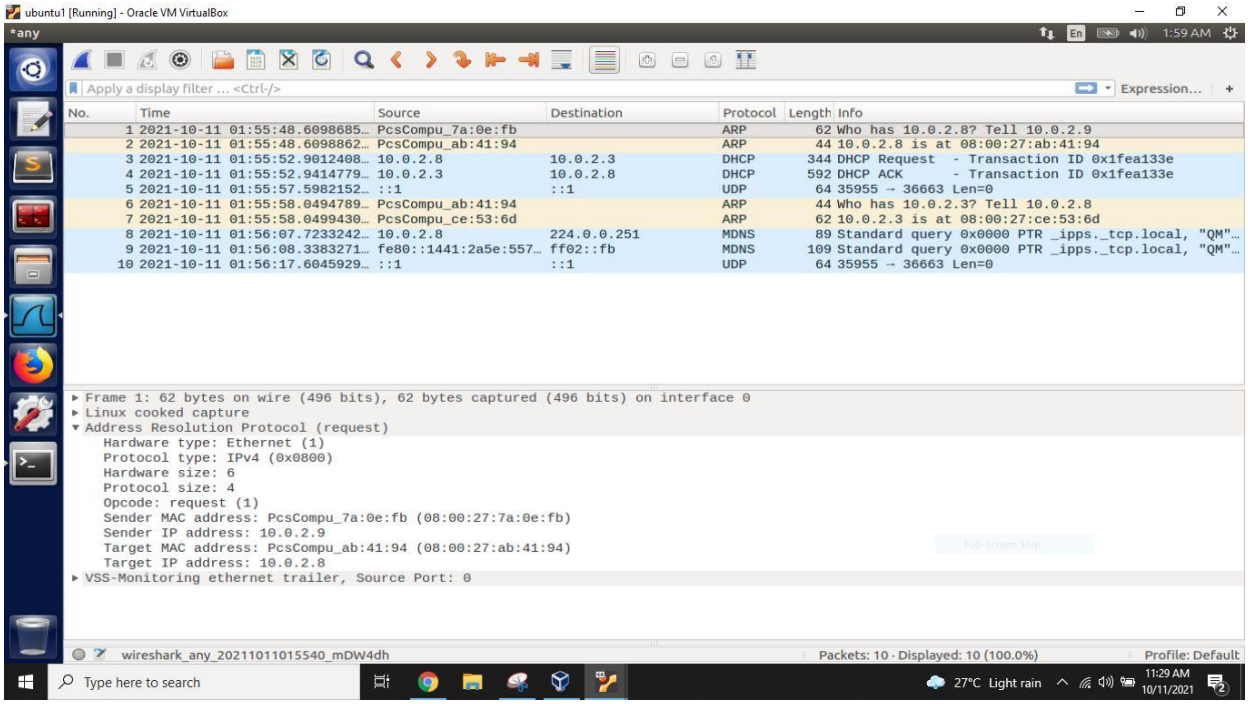
attacker



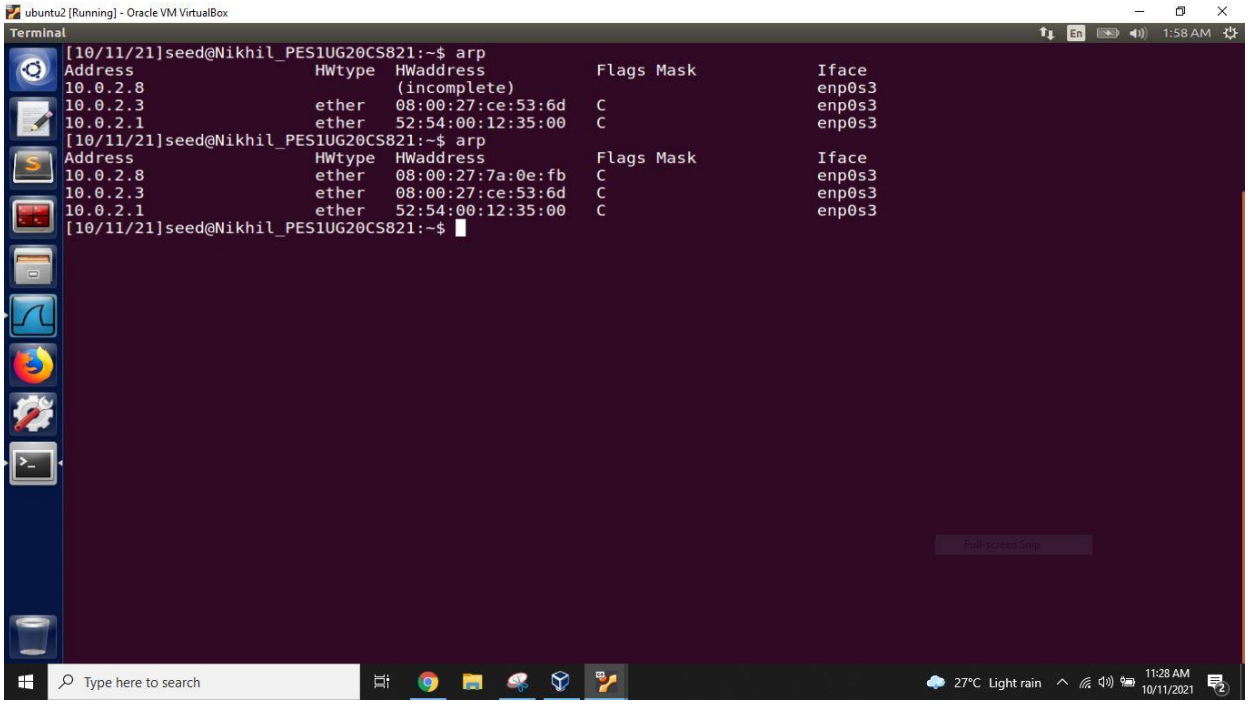
ubuntu1



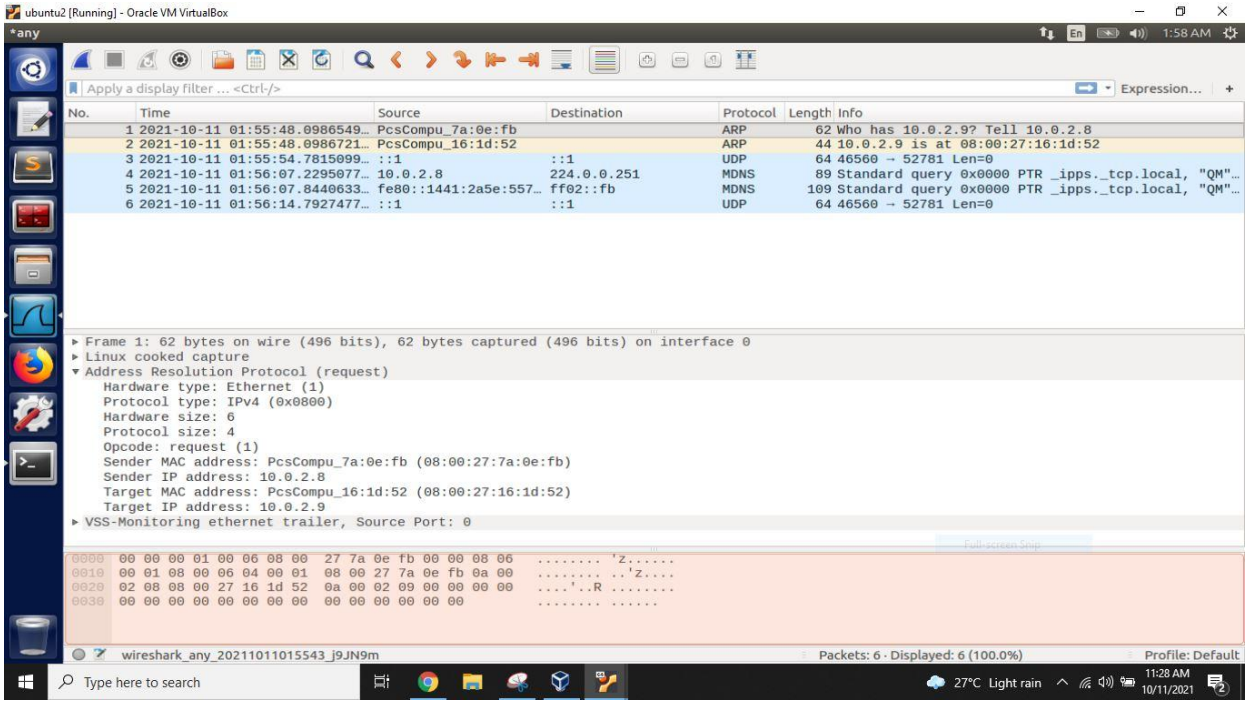
wire shark observation in ubuntu1



ubuntu2



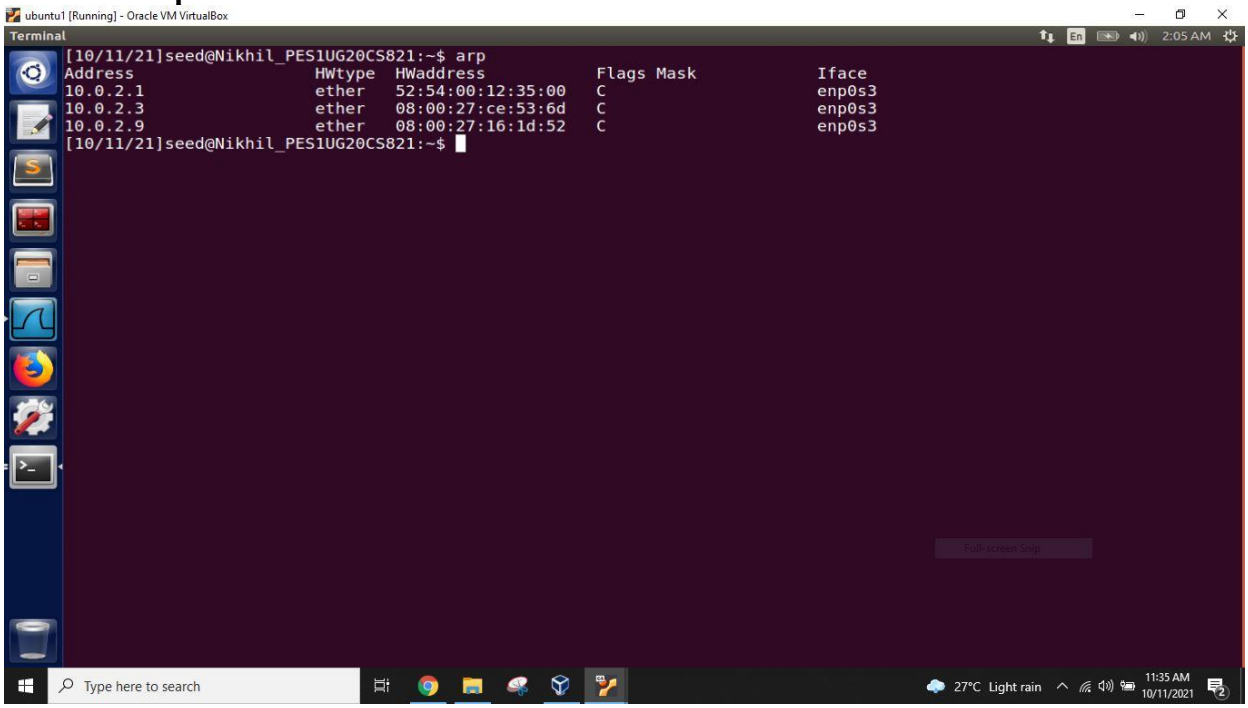
wire shark observation in ubuntu2



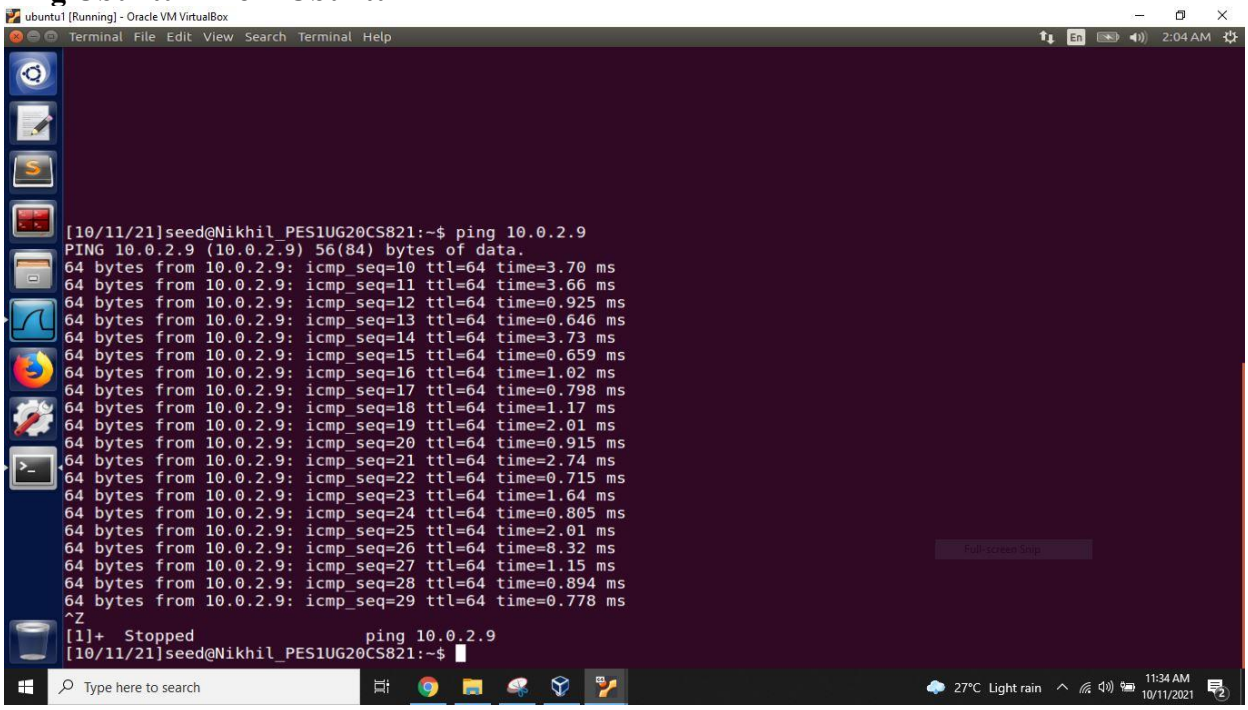
In this step the arp cache poisoning is done and it is observed in the wireshark

Step 2 (Testing)

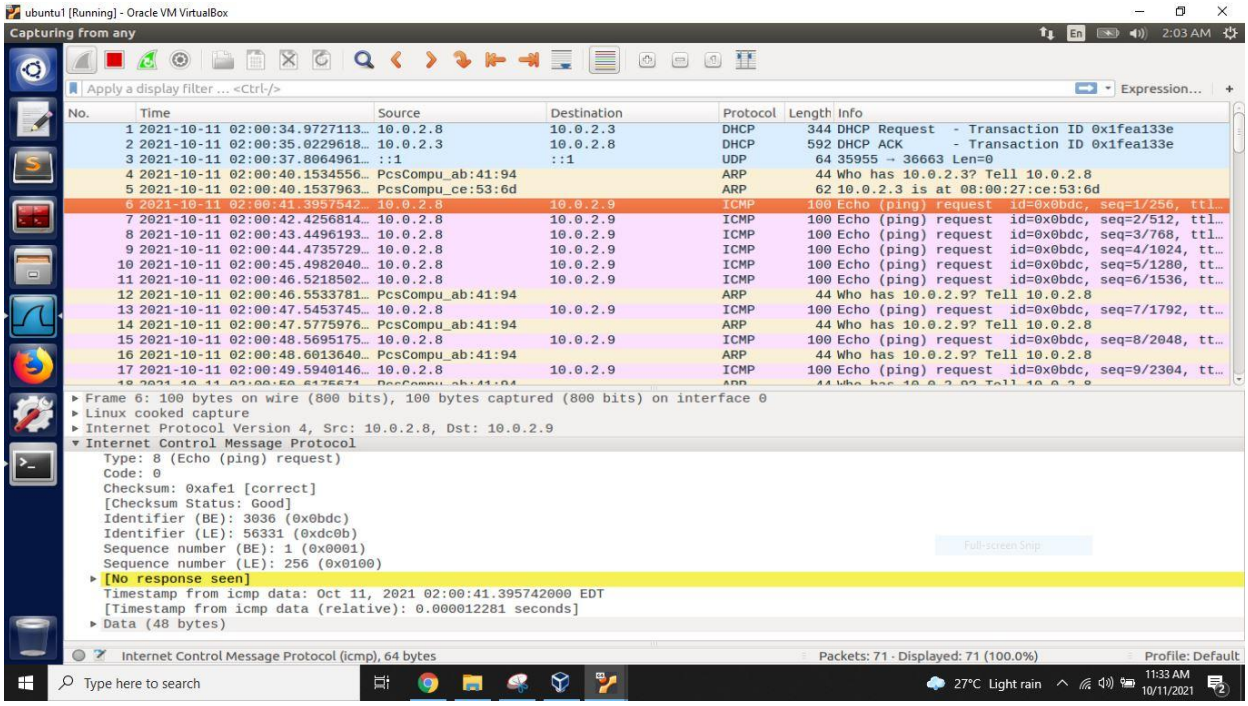
ubuntu1 arp



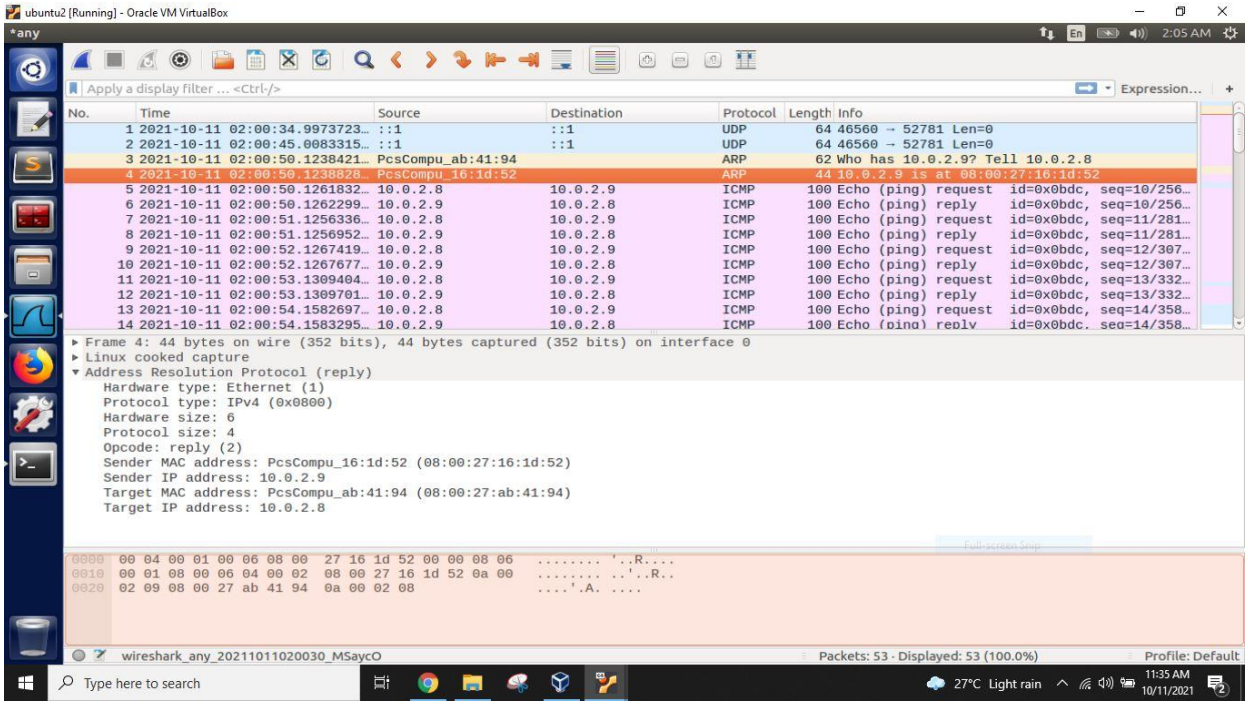
Ping Ubuntu 2 from Ubuntu 1



wire shark observation in ubuntu1



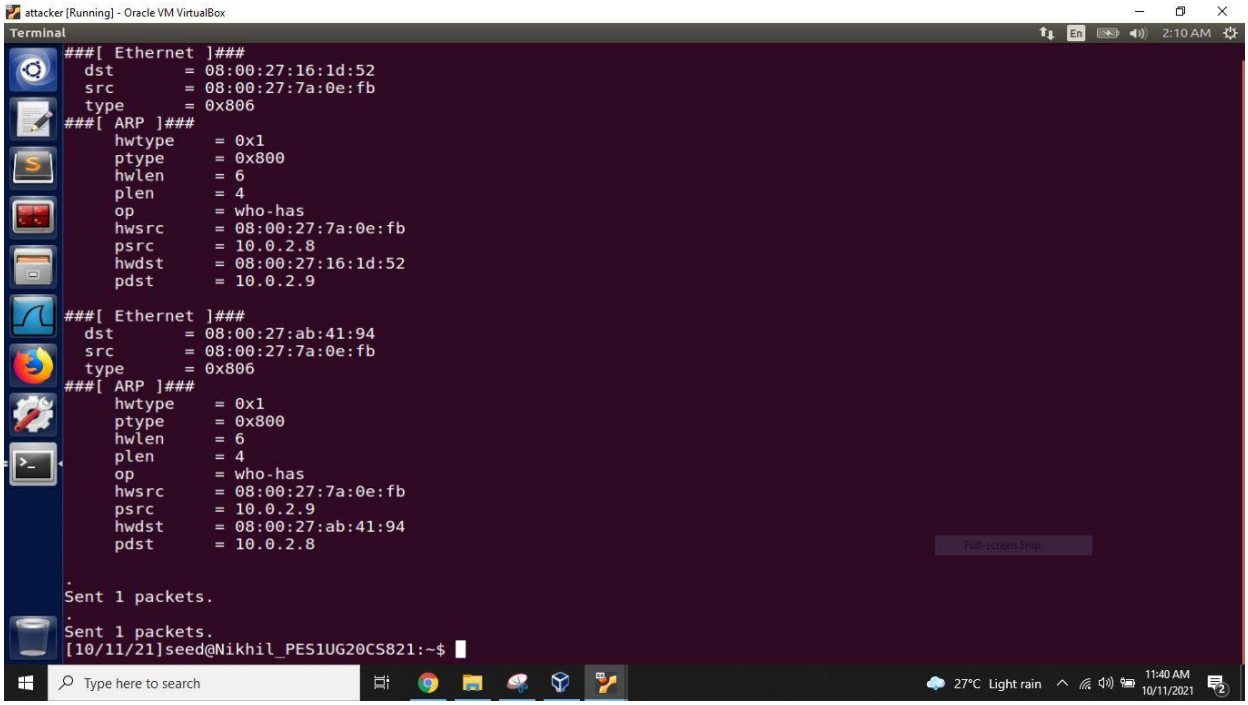
wire shark observation in ubuntu2



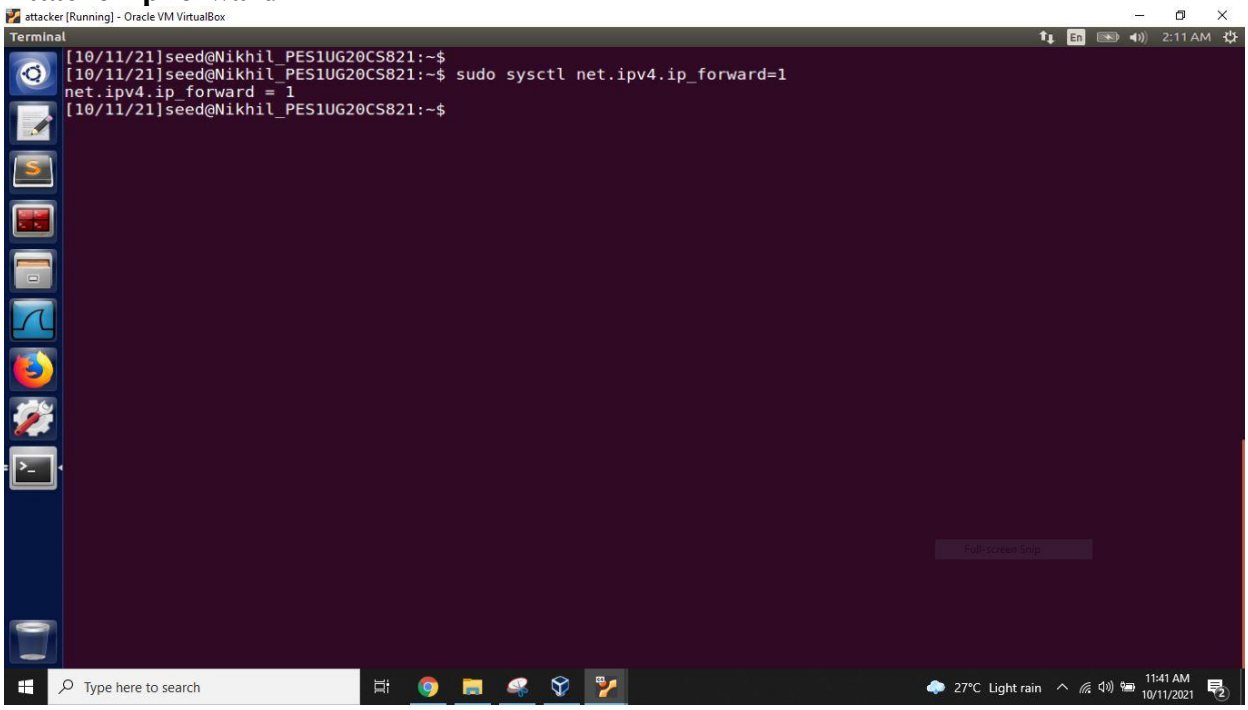
We observe no response is obtained in the wireshark

Step 3 (Turn on IP forwarding)

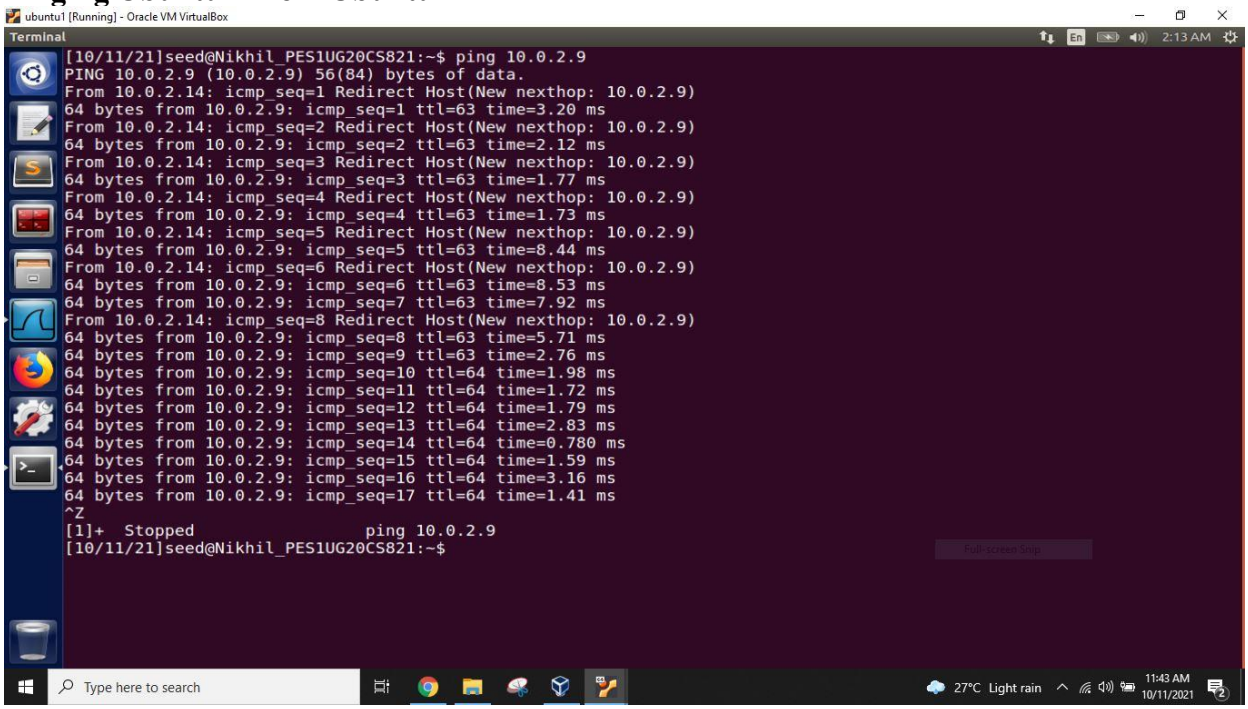
Attacker code



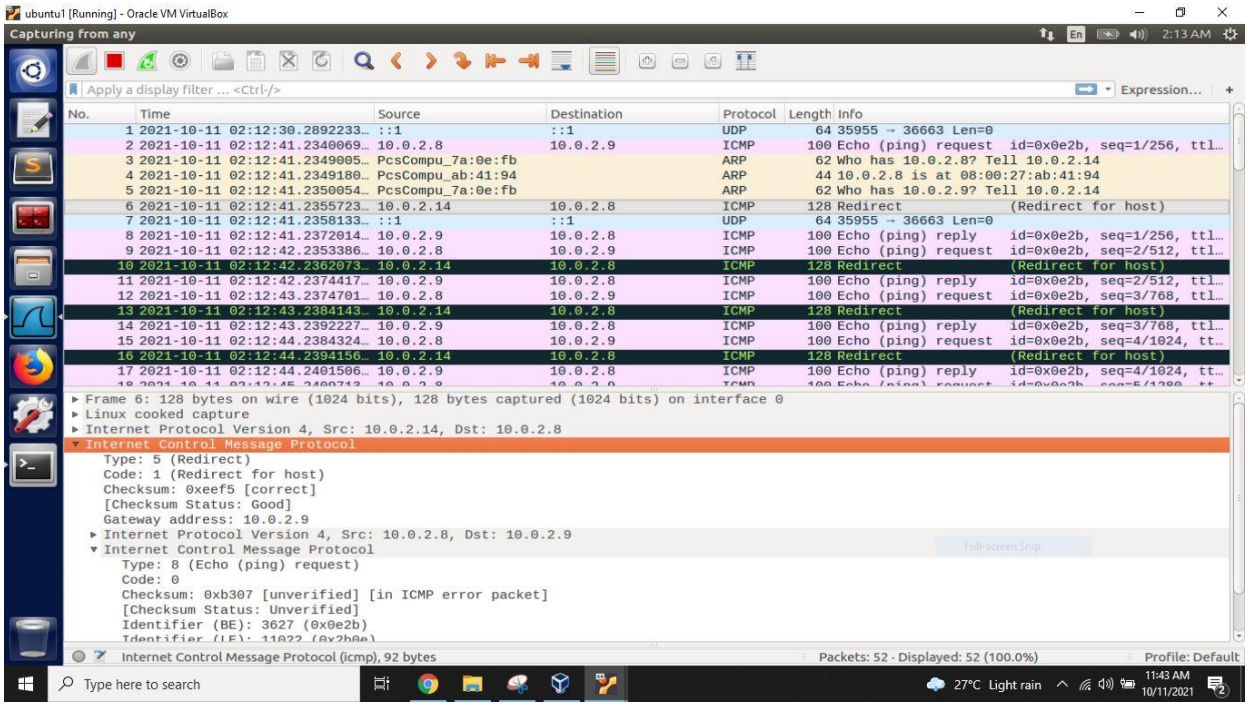
Attacker ip forward



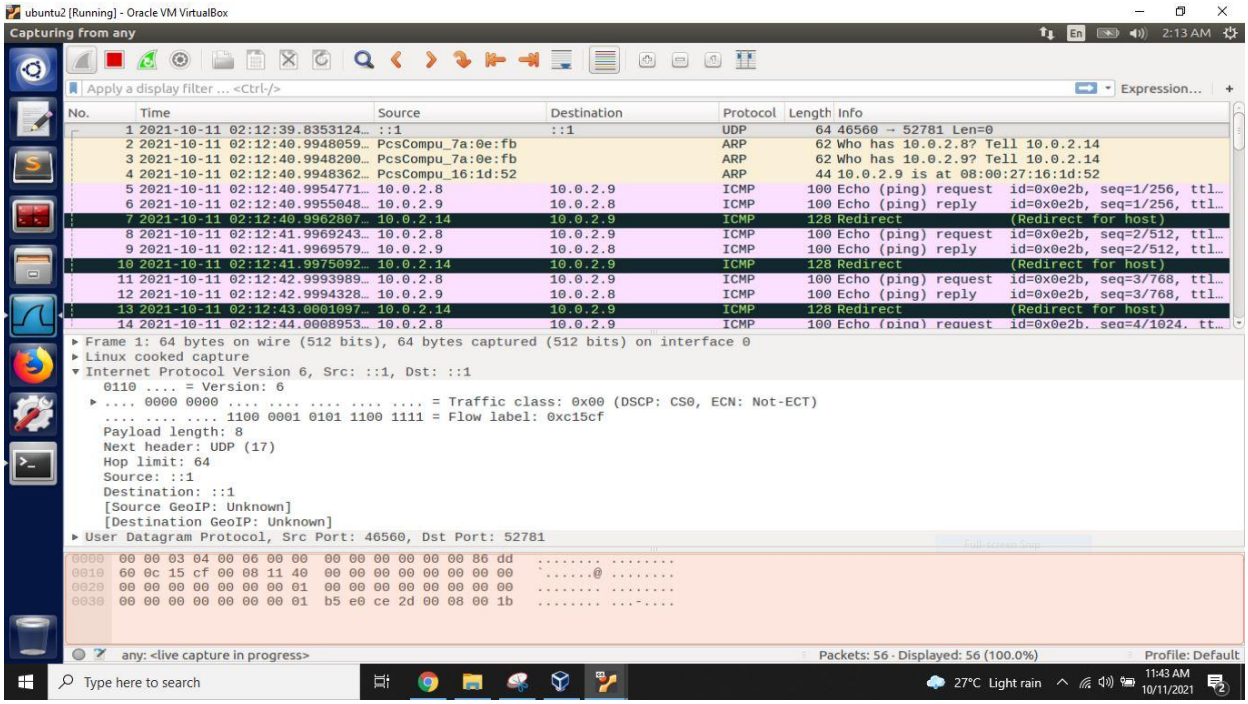
Pinging Ubuntu 2 from Ubuntu 1



wire shark observation in ubuntu1



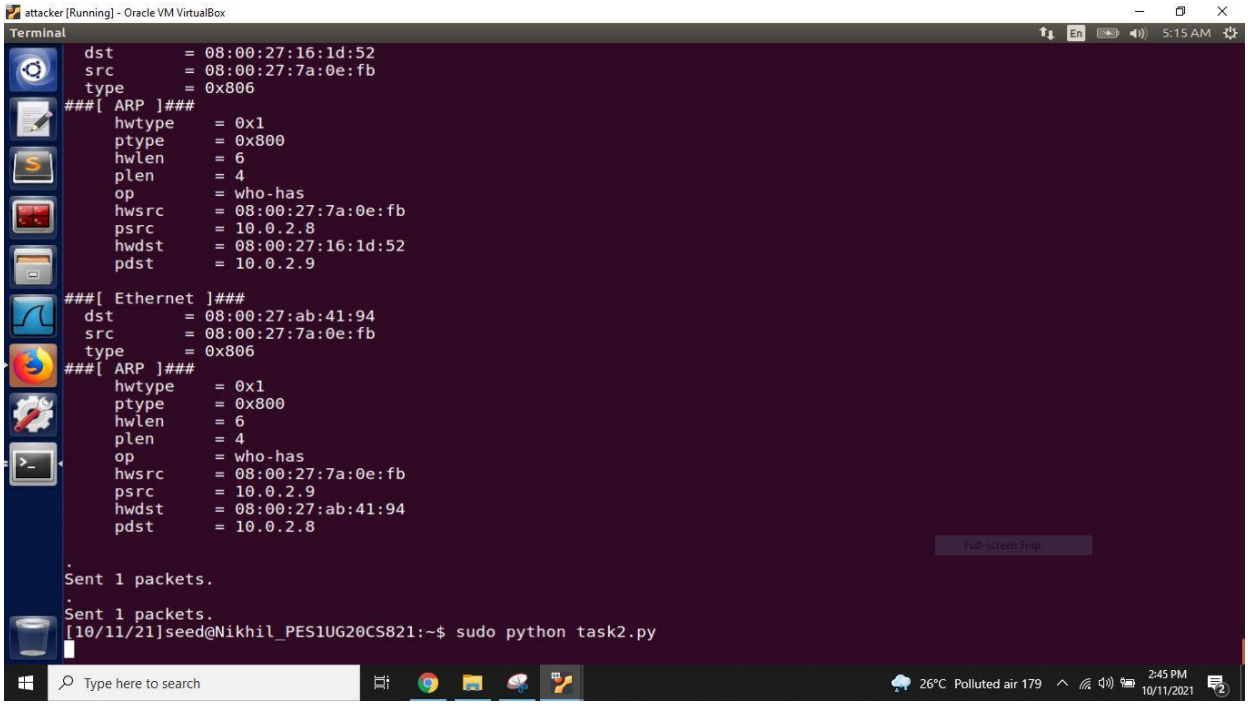
wire shark observation in ubuntu2



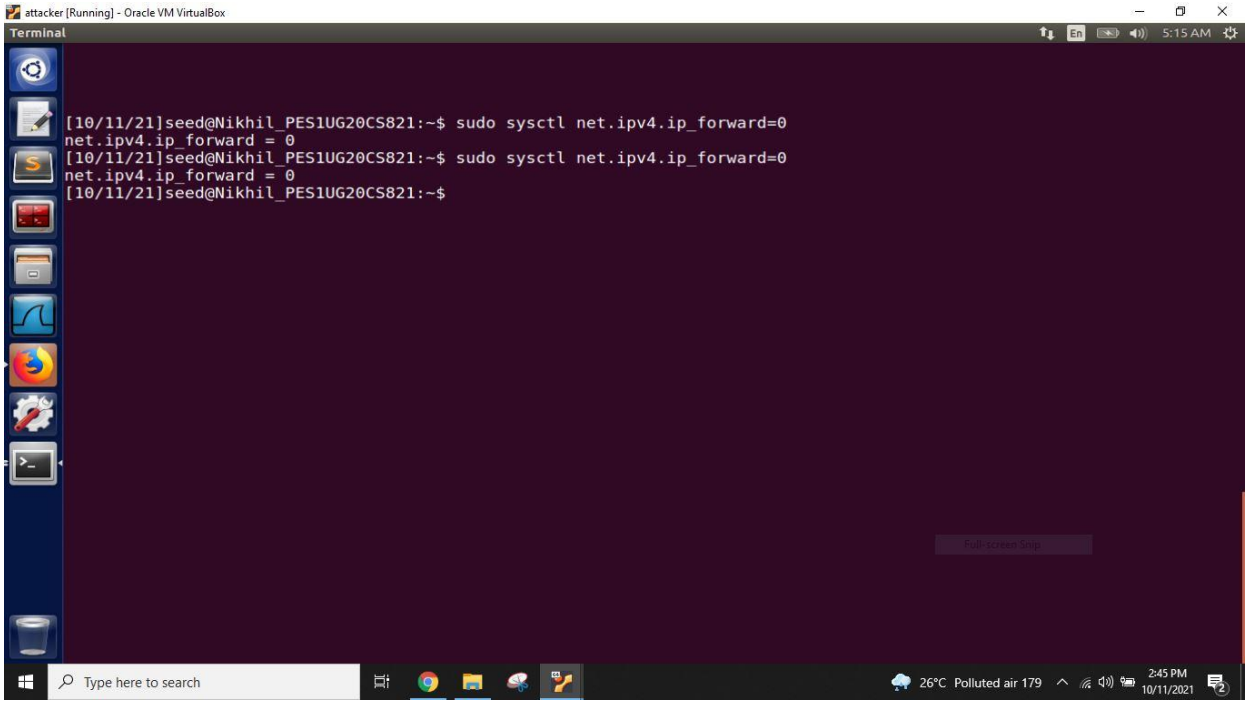
Same above step is repeated with addition of setting the ip forward and redirect can be seen in the wireshark observation.

Step 4 (Launch the MITM attack)

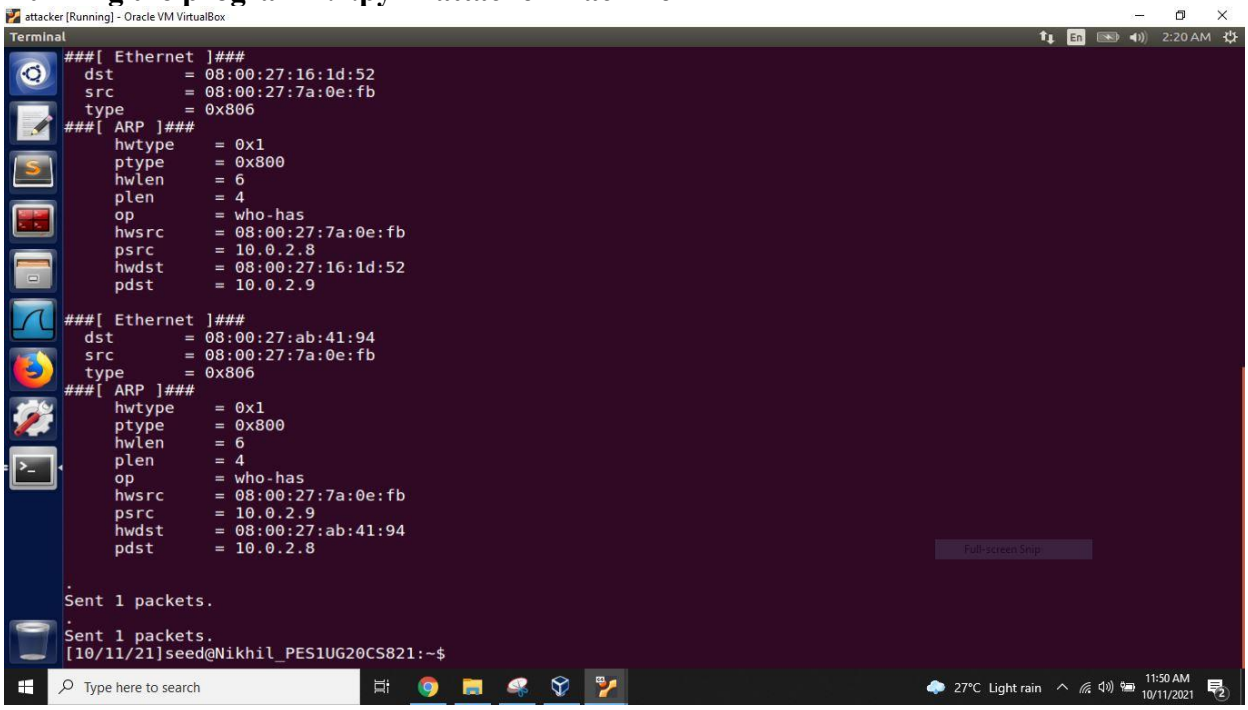
Attacker



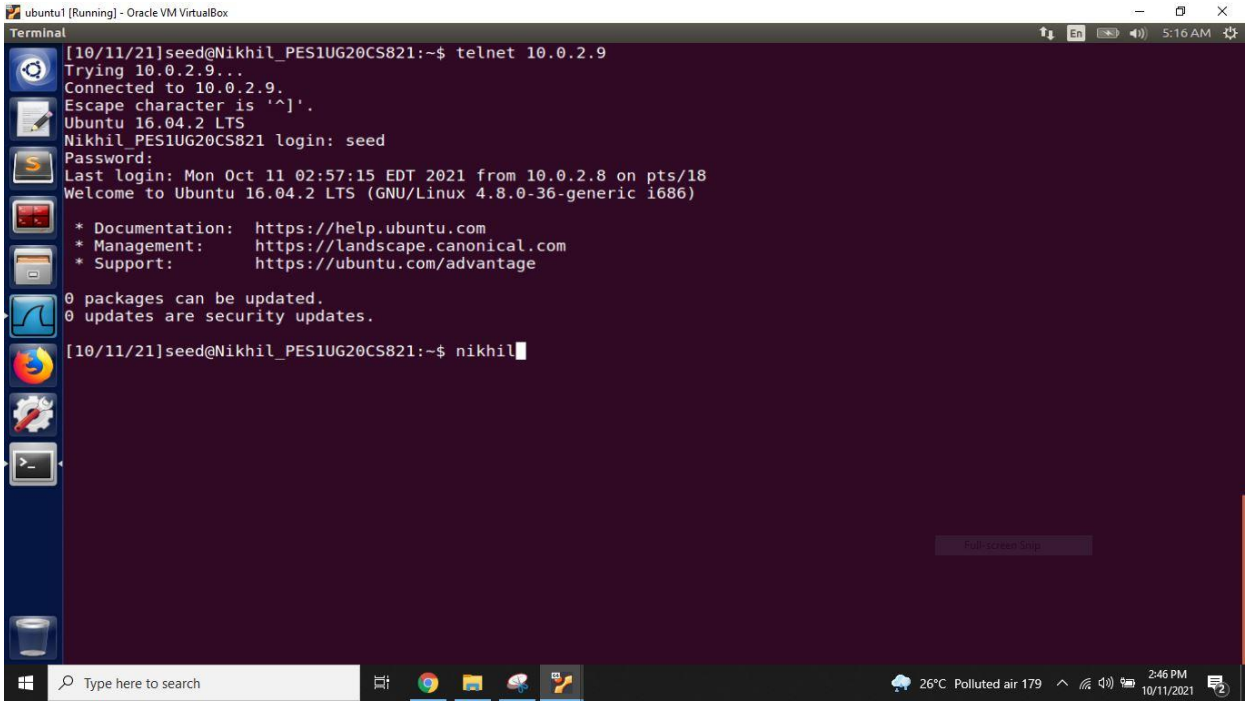
IP Forward set to 0



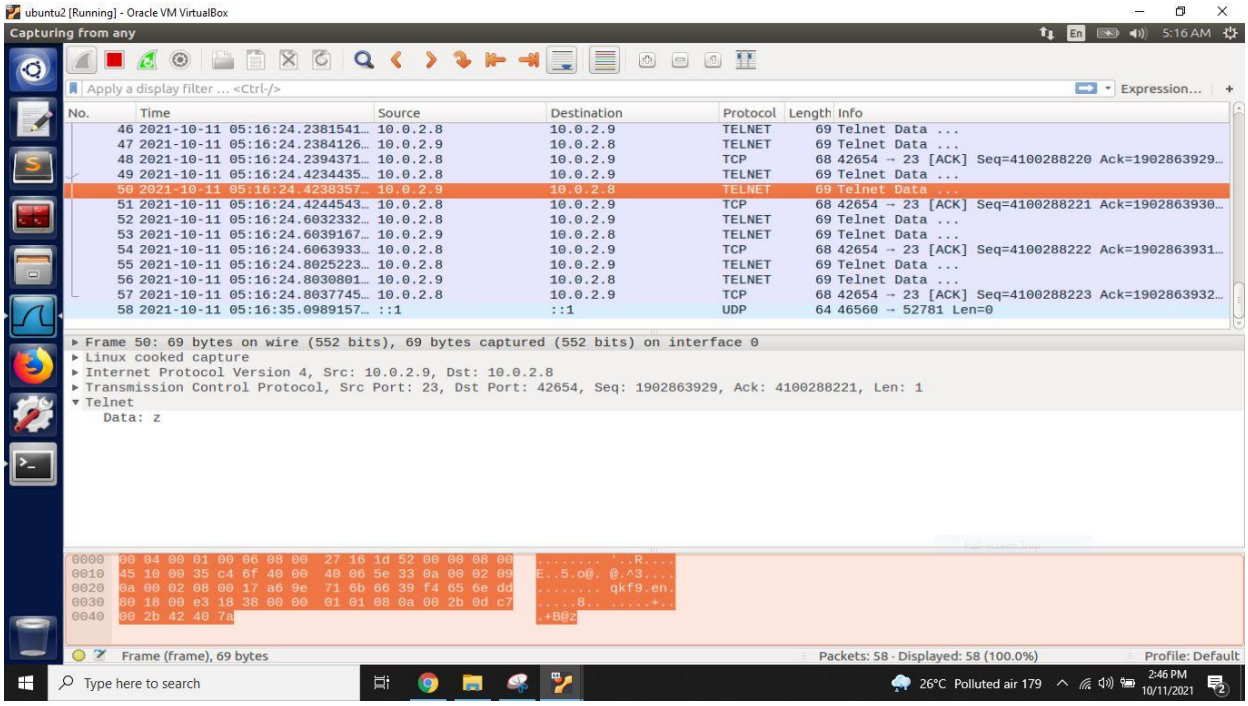
Running the program 2.1.py in attacker machine



Getting connected to Ubuntu 2 from Ubuntu 1



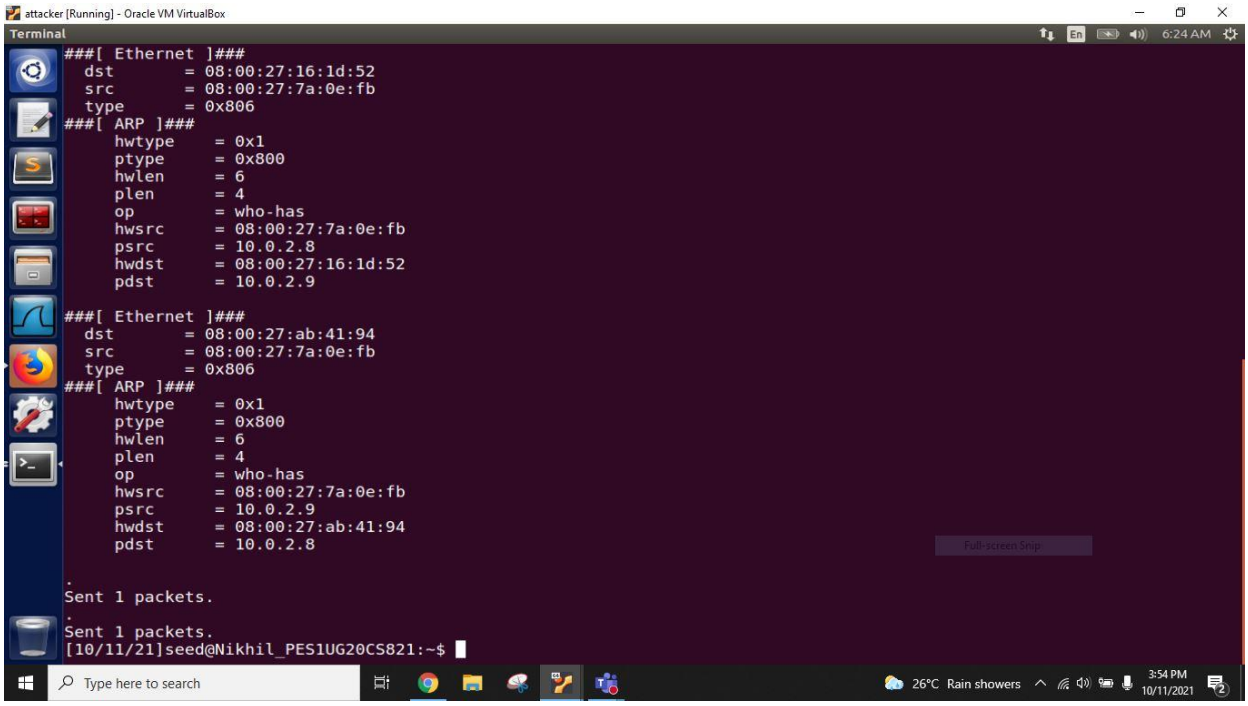
wire shark observation in Ubuntu 2



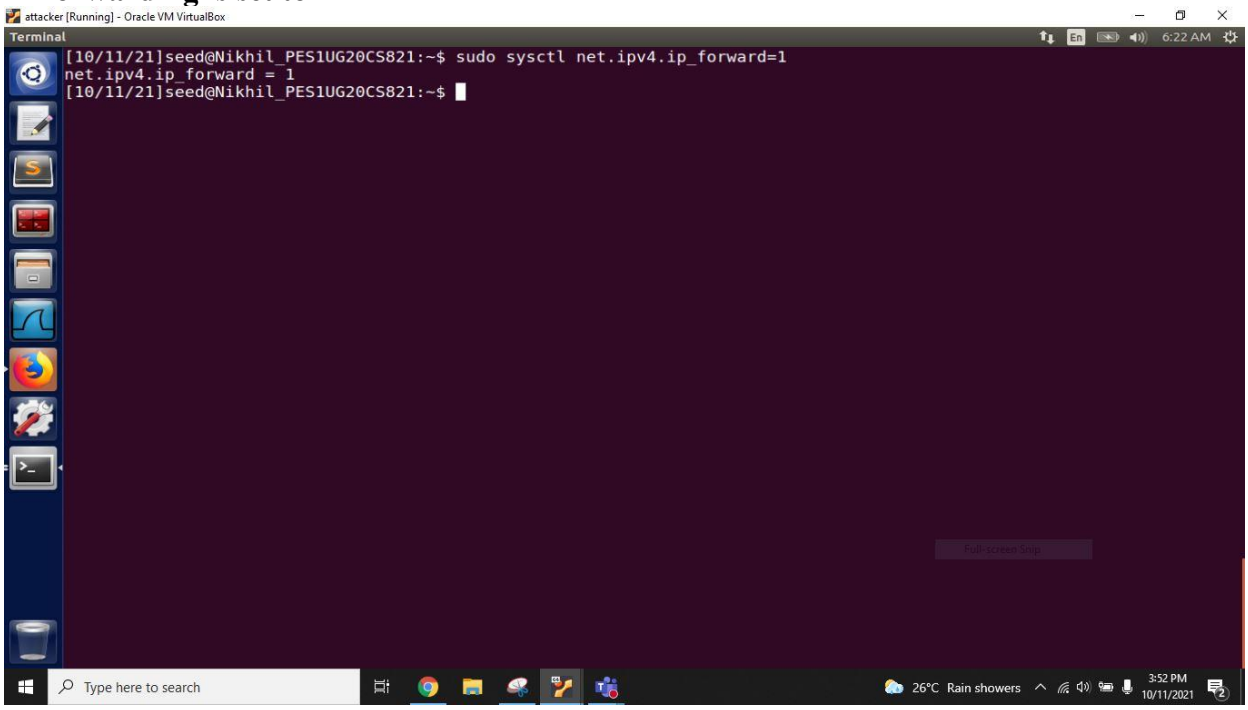
Here we turn off the ip forwarding and spoofing is done when the ubuntu1 connects to ubuntu2 using telnet the attackers performs spoofing and whatever the data typed in by the ubuntu1 user gets replaced by the constant charater ‘z’ which can be seen in the wireshark in ubuntu2.

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

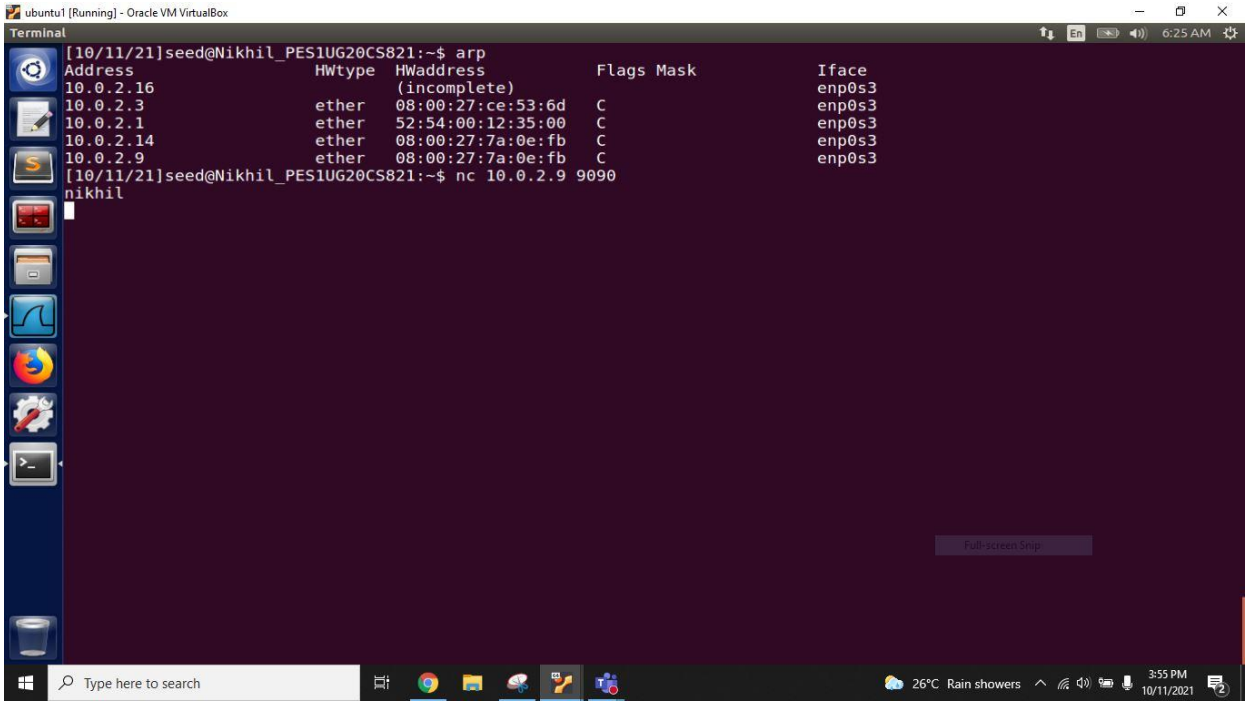
Attacker



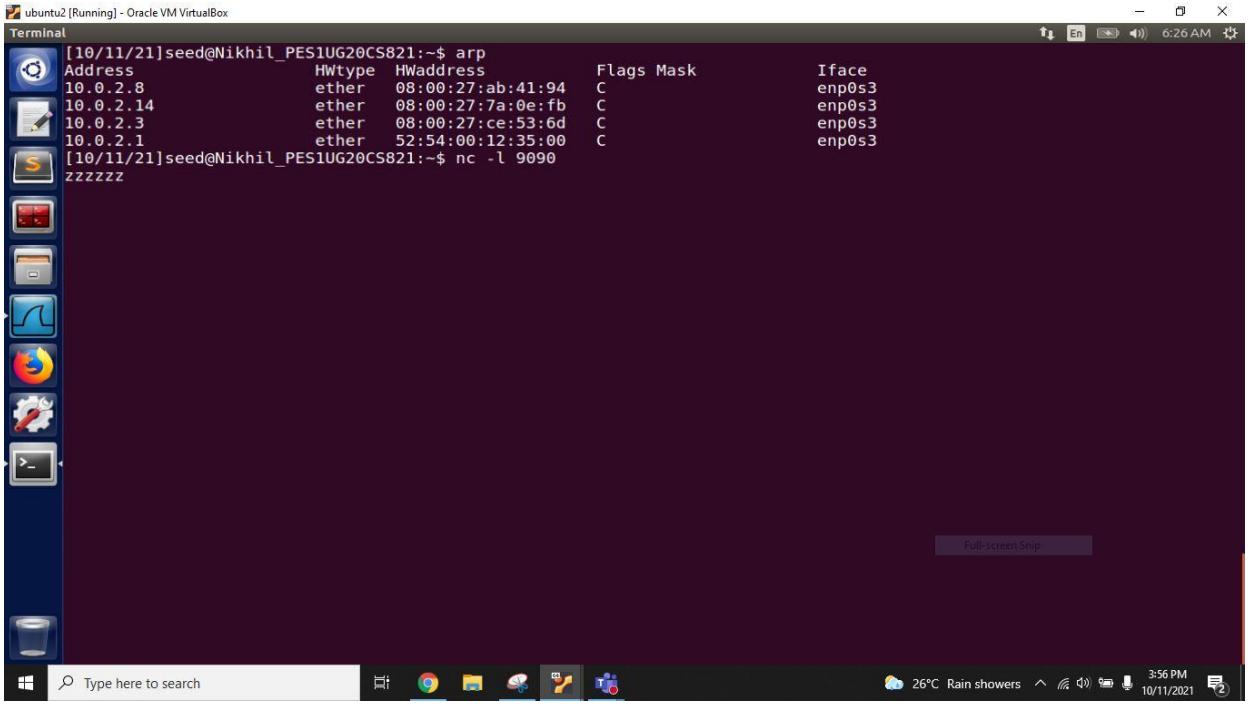
IP forwarding is set to 1



ARP table in Ubuntu 1



ARP table in Ubuntu 2



This as same objective of the task 2 but the ubuntu1 does not gets connect to through telnet whereas netcat is used when the ubuntu1 user tries to communicate with the ubuntu2 through the port 9090 the characters are replaced by the constant character ‘z’ by the attacker.