CNS LAB

ASSIGNMENT - 4

IP/ICMP Attacks

NAME : SIRI S

SEMESTER : 5

SECTION :H

SRN : PES1UG19CS485

LAB SETUP:

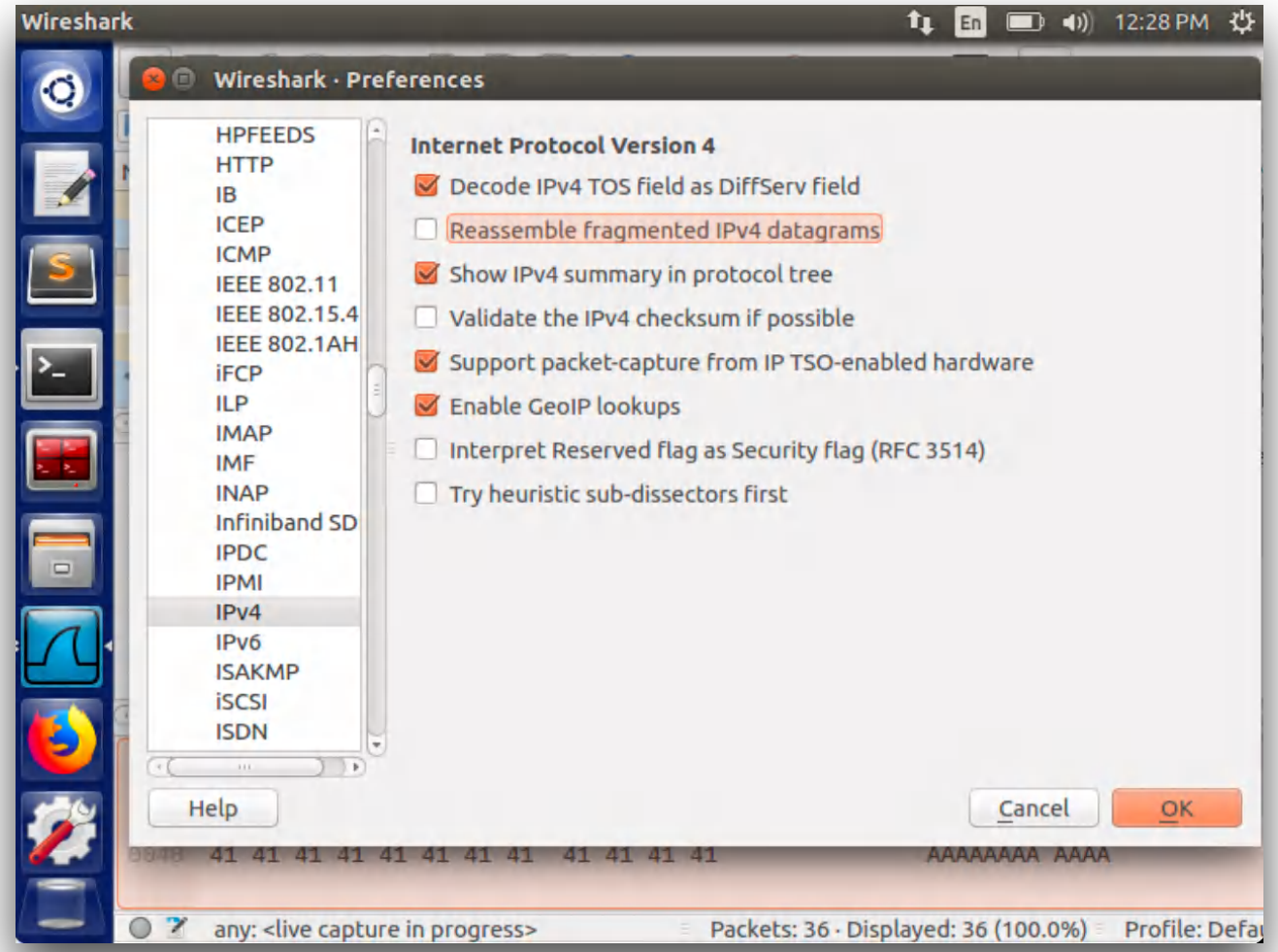
UDP SERVER: 10. 0. 2. 11

UDP CLIENT: 10. 0. 2. 10

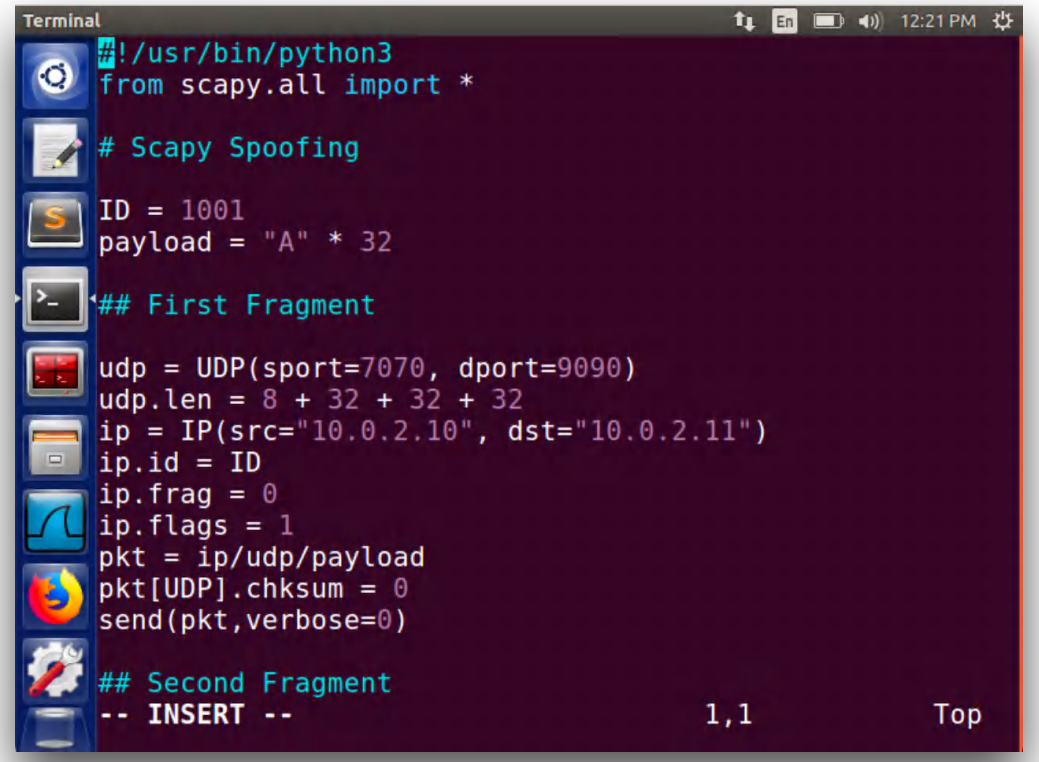
Task 1A: Conducting IP Fragmentation

We are going to spoof IP fragments using scapy. We will do this by building multiple IP fragments and sending it across the UDP server.

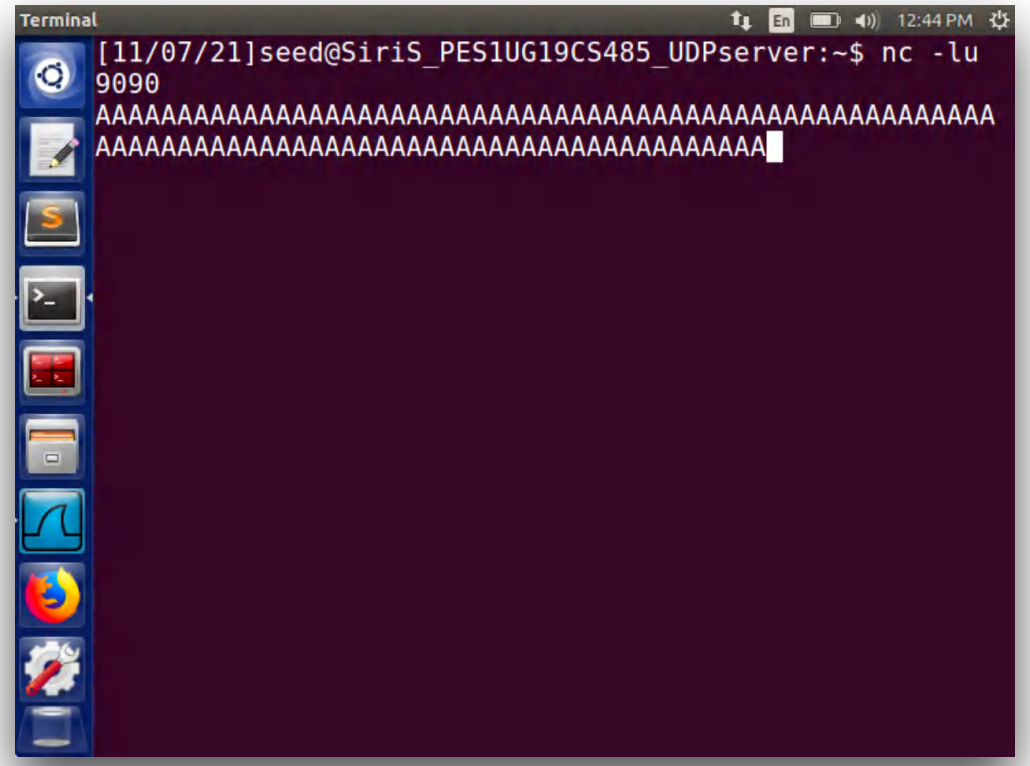
Before starting, we will uncheck the "Reassemble fragmented IPv4 datagrams" option on both VMs as shown below as we need to analyse the packets in fragments:



The code used to perform this task:

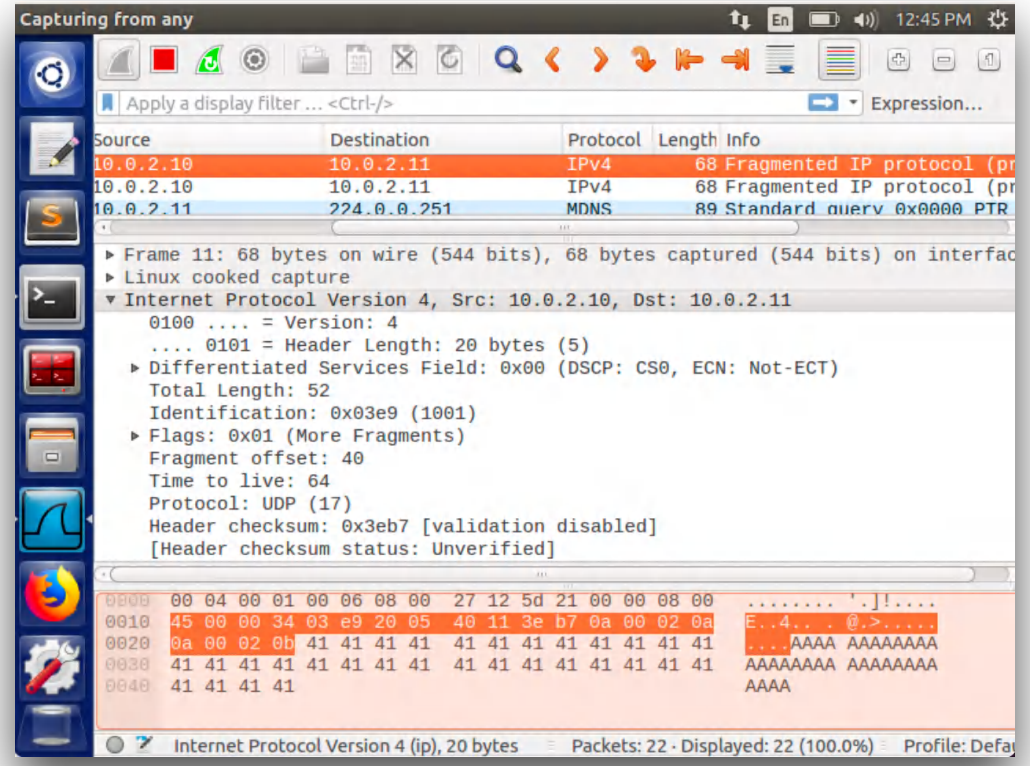


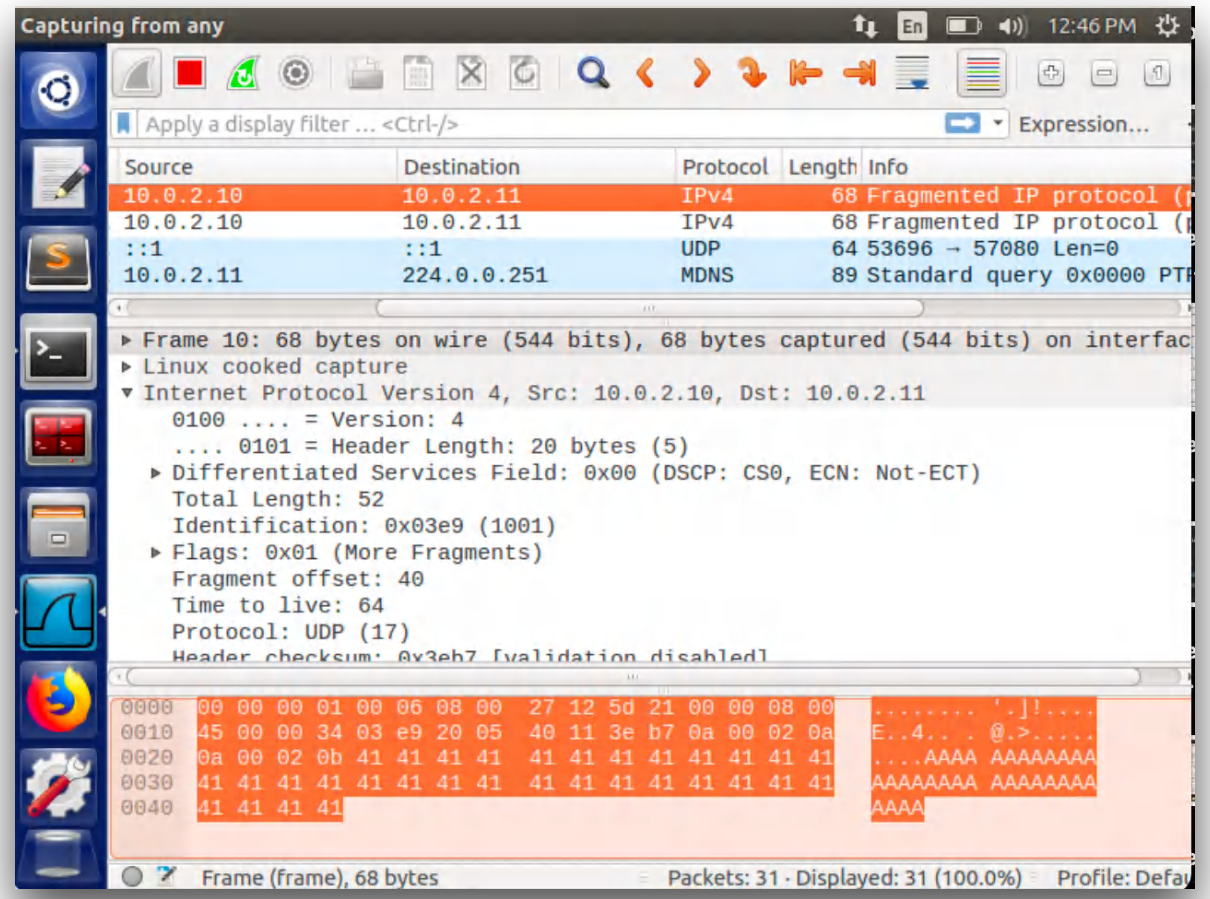
We will first run the command '*nc -lu 9090*' on the Server VM and then run the code on the Client VM:



The output received

The Wireshark screenshots:



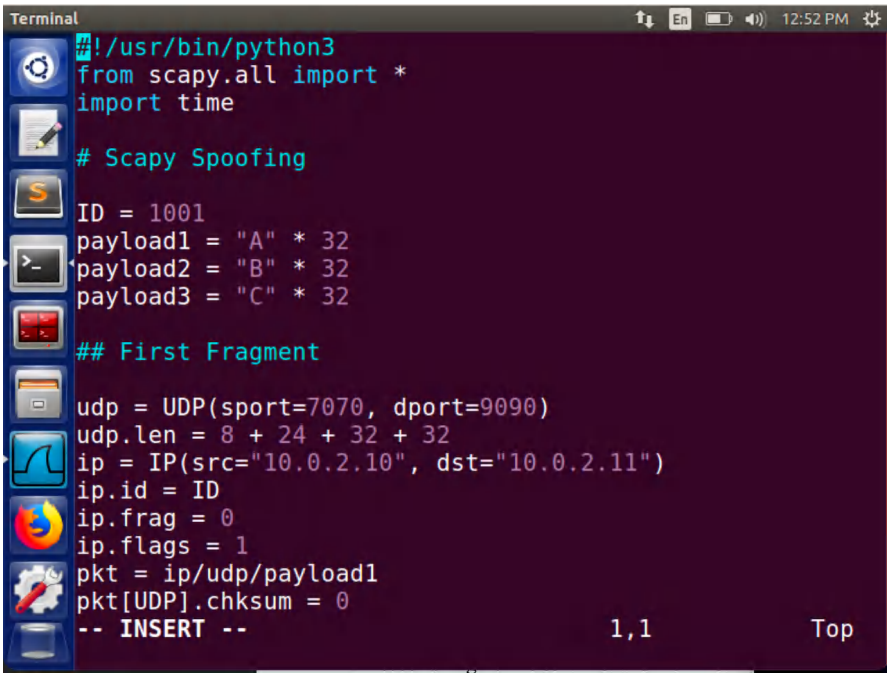


A UDP packet is sent to the UDP server in 3 packets each of 32 bytes as shown in the code. In the wireshark capture we can see that the data length for the UDP packet is 96 bytes. On the server machine we can see that 96 A's are printed. All 32 bytes packets are assembled together. The fragments are sent from VM1 (client) to VM2 (server).

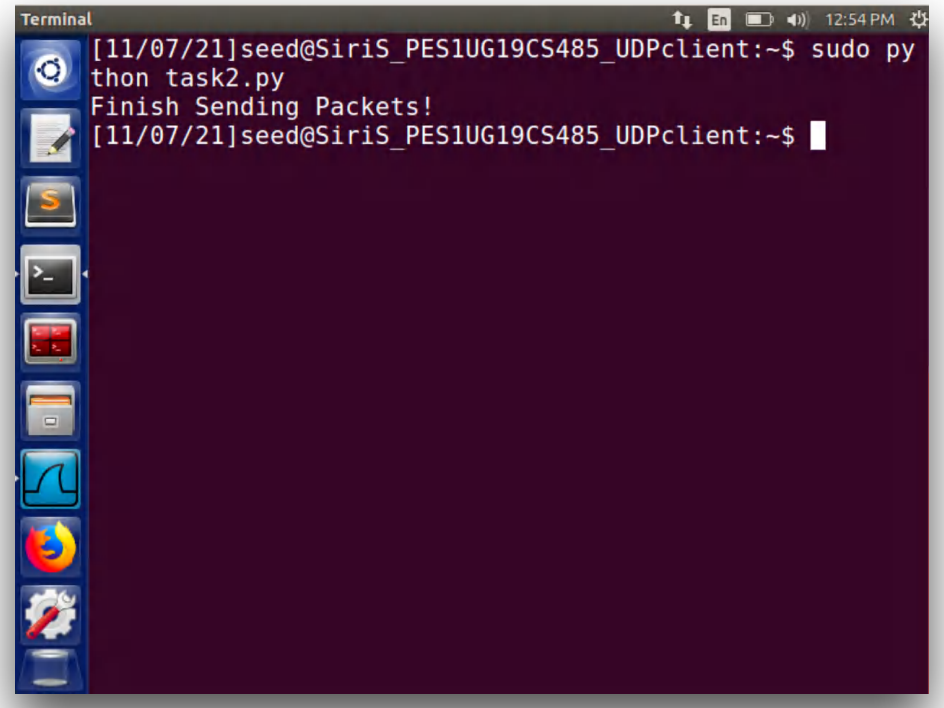
Task 1B: IP Fragments with Overlapping Contents

Step 1: Again we need to send fragments to UDP server but this time the contents of fragments need to overlap.

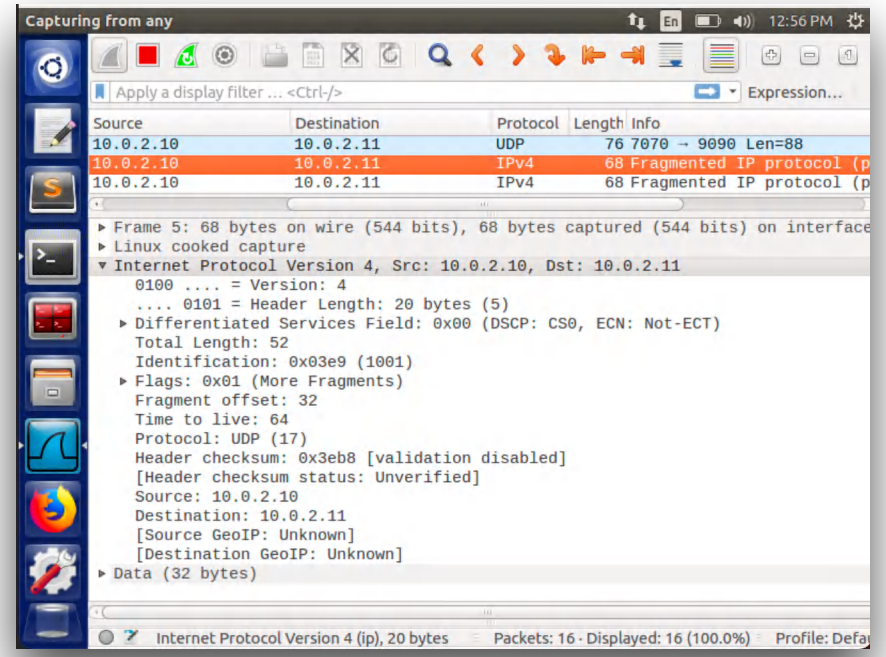
The code used to perform this task:



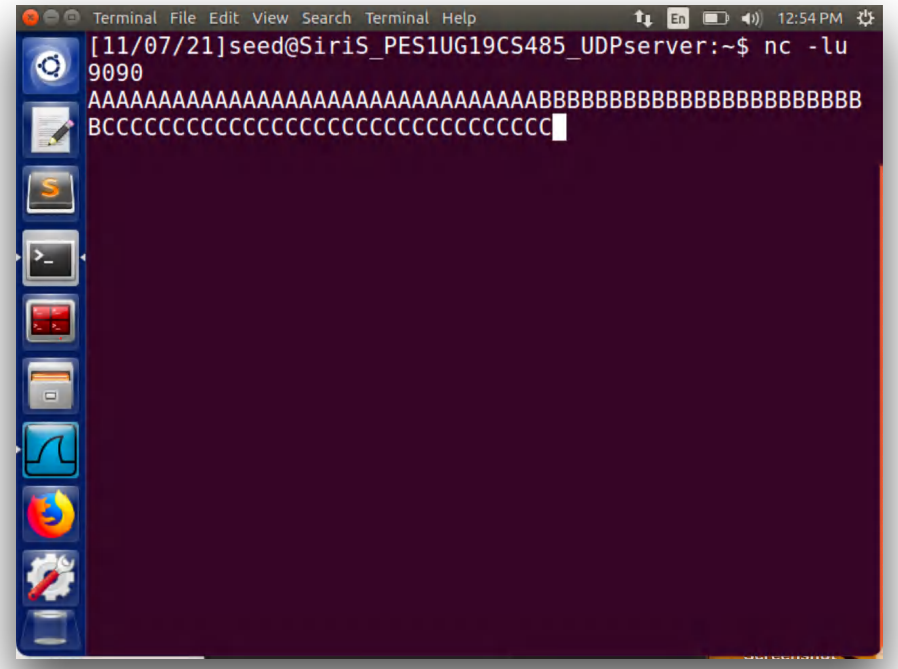
Upon running the code on the Client:



The Wireshark screenshot:



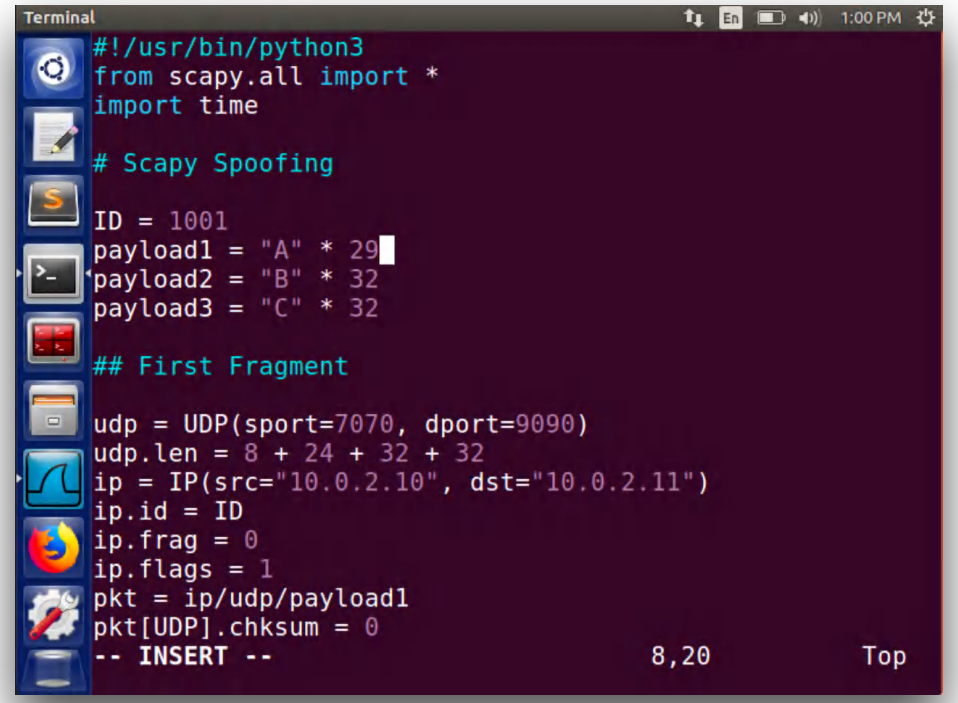
The output on the Server machine:



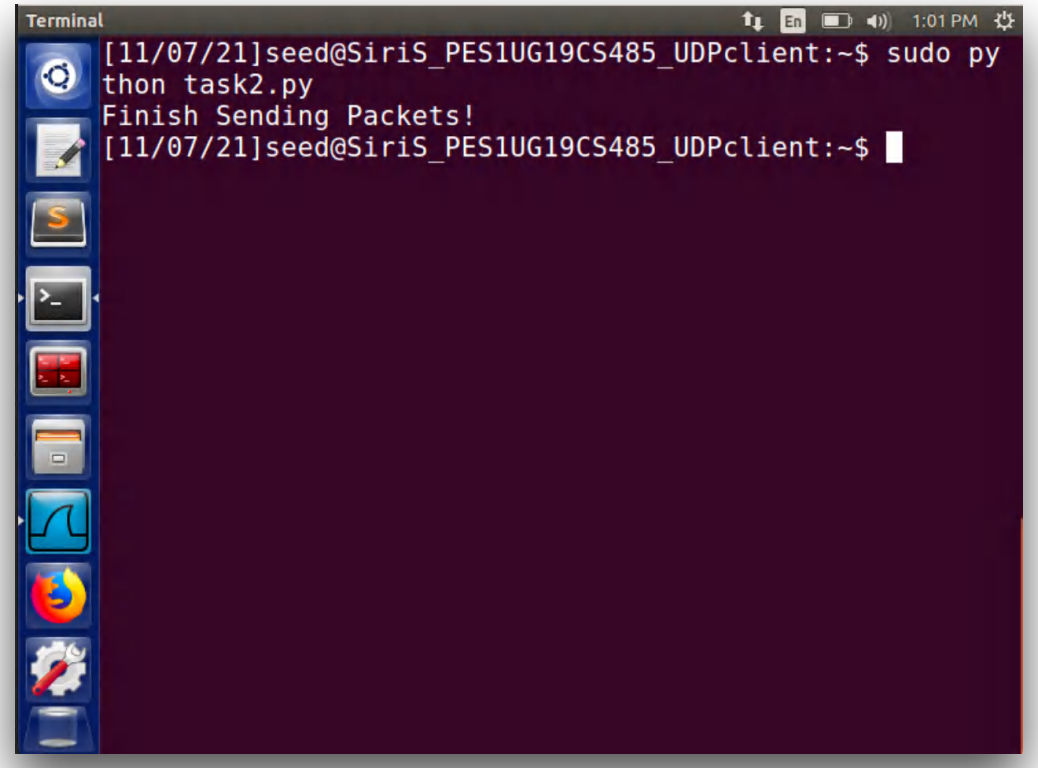
Only the 1st and 3rd fragments data is displayed. Due to the id.frag value set, third fragment continues right after the second, and the 2nd fragment is not displayed. The second fragment overwrites the first , hence data is discarded and the first one is kept.

Step 2: The end of the first fragment and the beginning of the second fragment should have K bytes of over-lapping, i.e., the last K bytes of data in the first fragment should have the same offsets as the first K bytes of data in the second fragment.

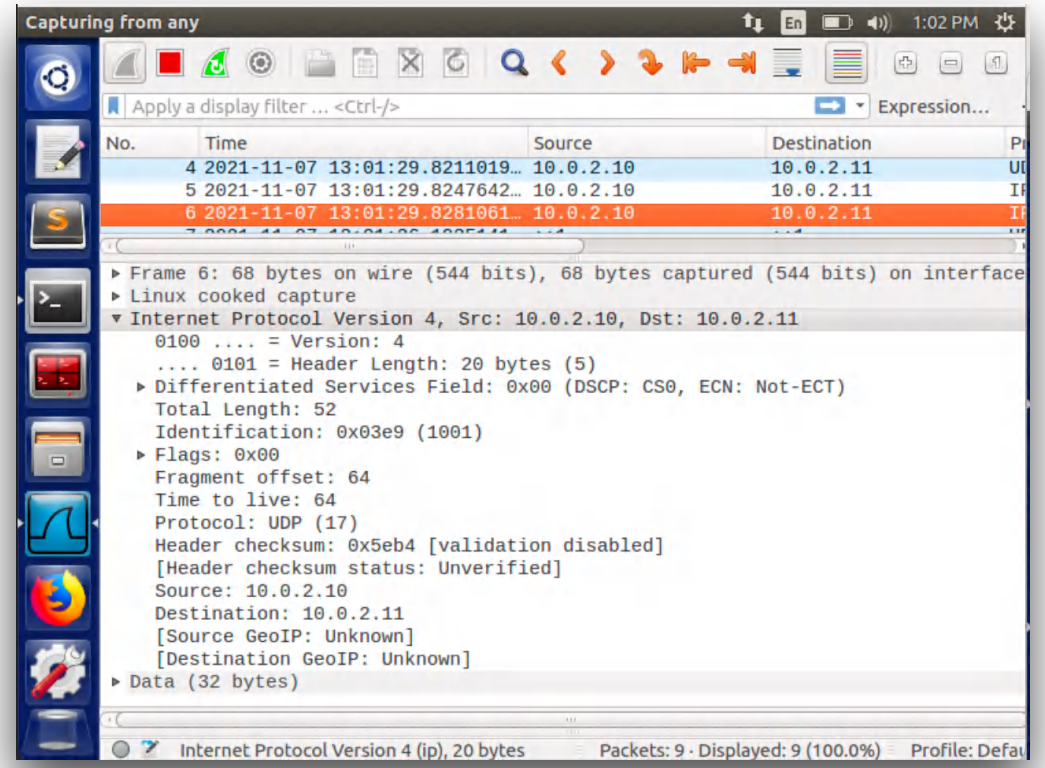
The code used to perform this task:

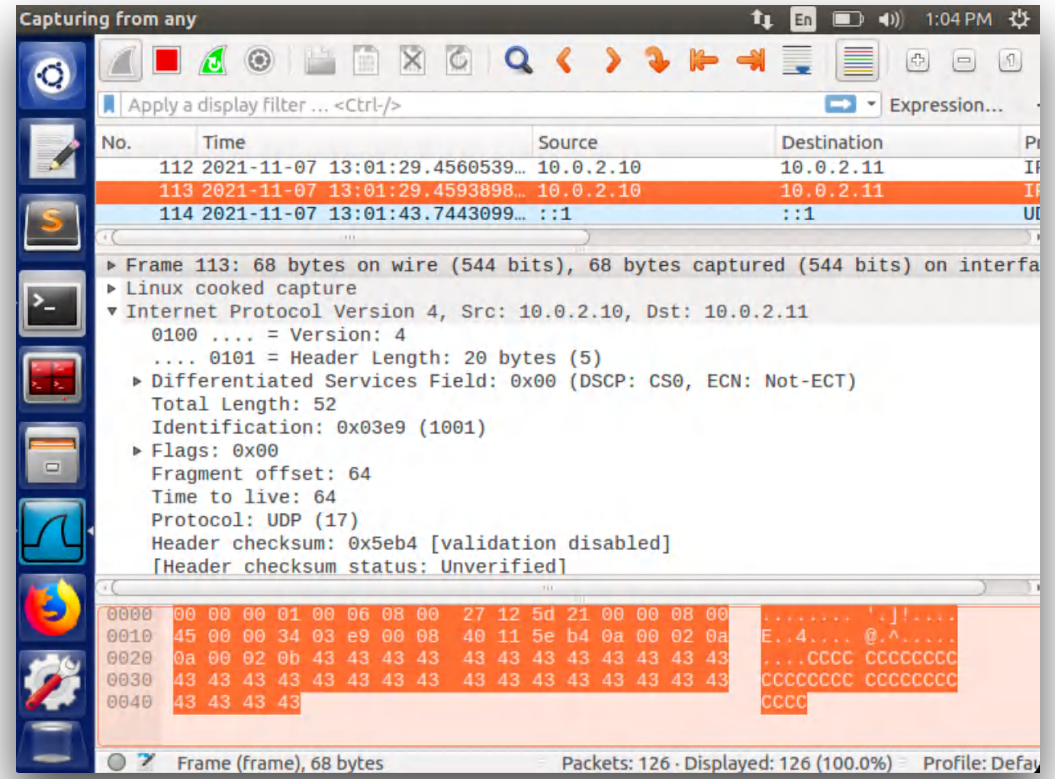


Upon running the code on the Client:

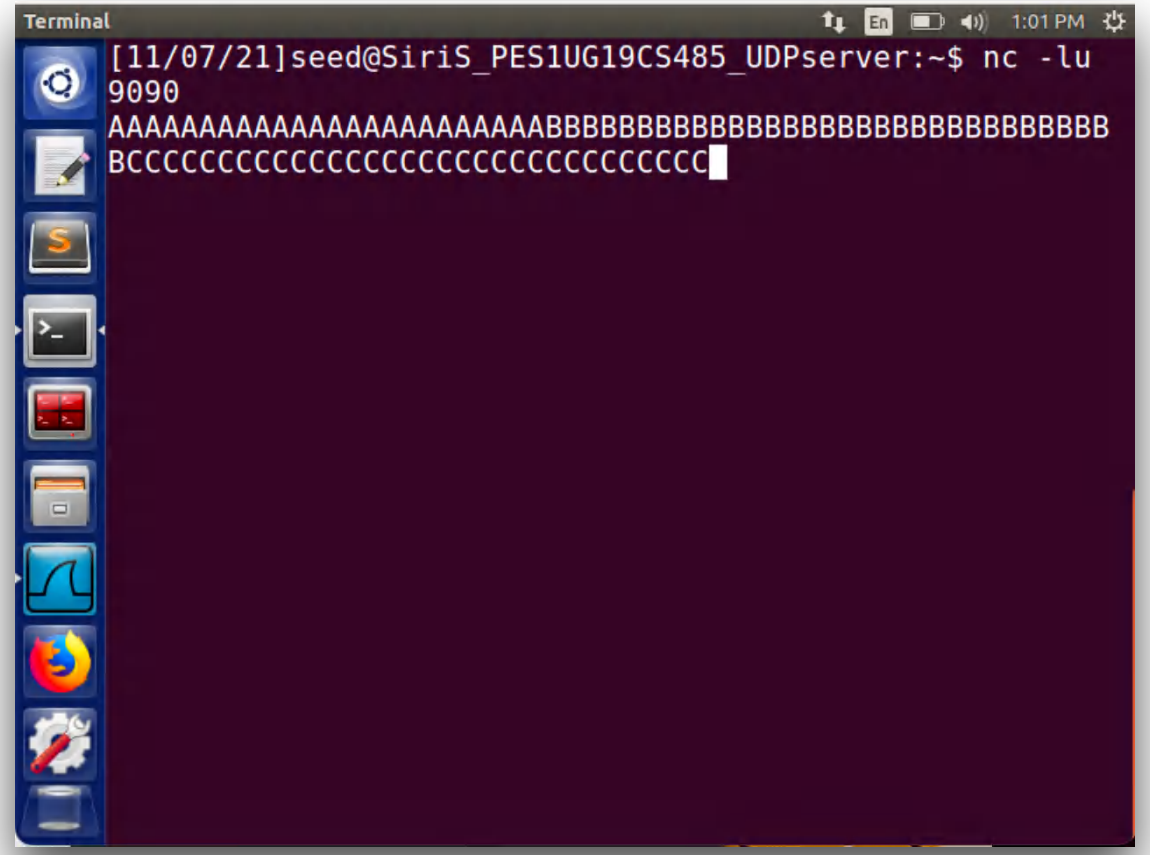


The Wireshark screenshot:





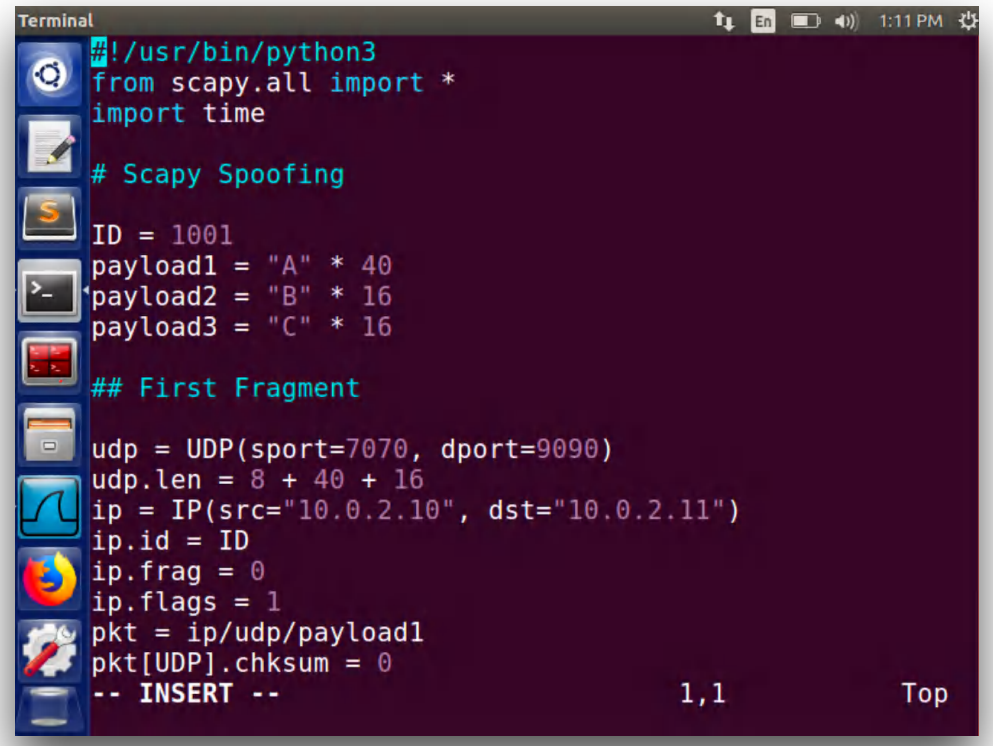
The output on the Server machine:



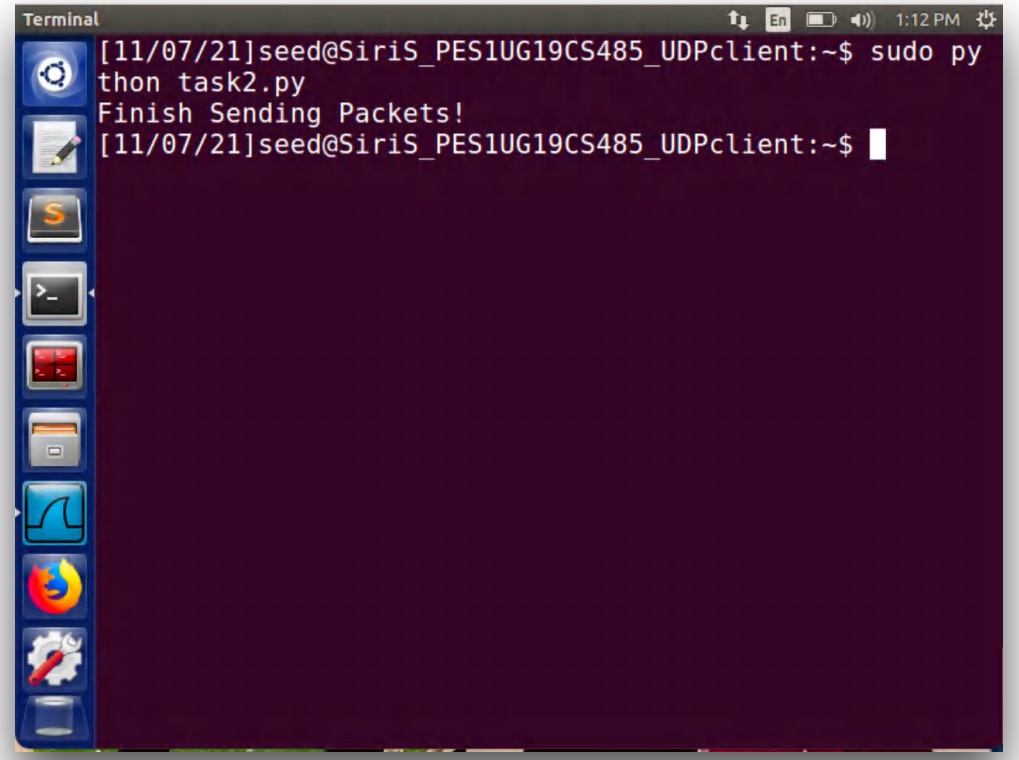
The first fragment and the beginning of the 2nd fragment overlap by **5 bytes**. The first fragment ends at 37 bytes but the 2nd starts at 32 hence it overlaps, as clearly seen in the server screenshot above and Wireshark.

Step 3: The second fragment is completely enclosed in the first fragment

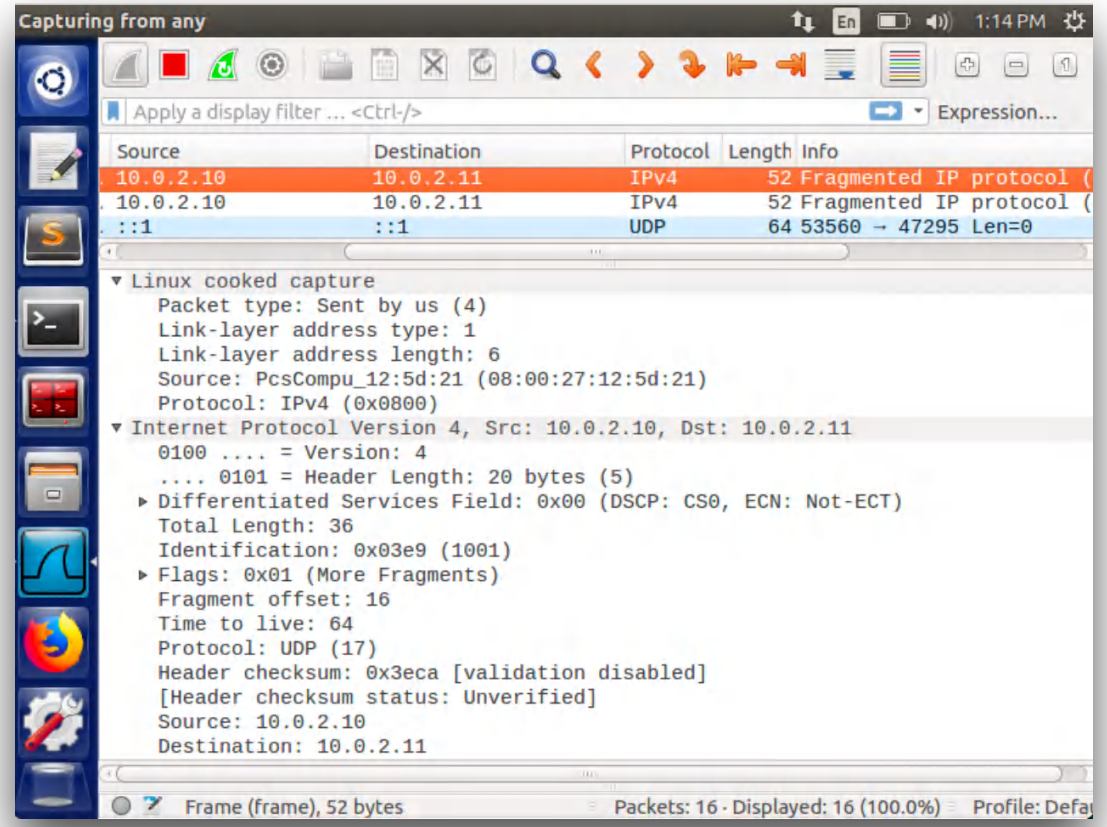
The code used to perform this task:

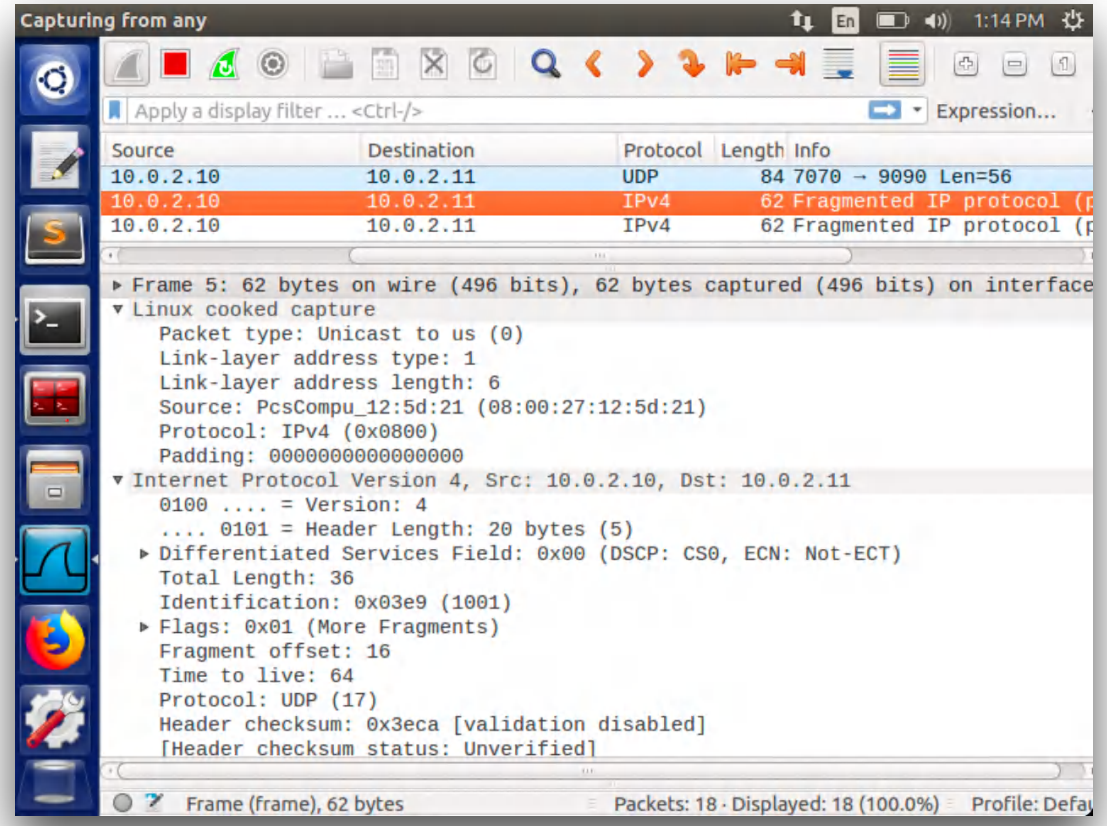


Upon running the code on the Client:

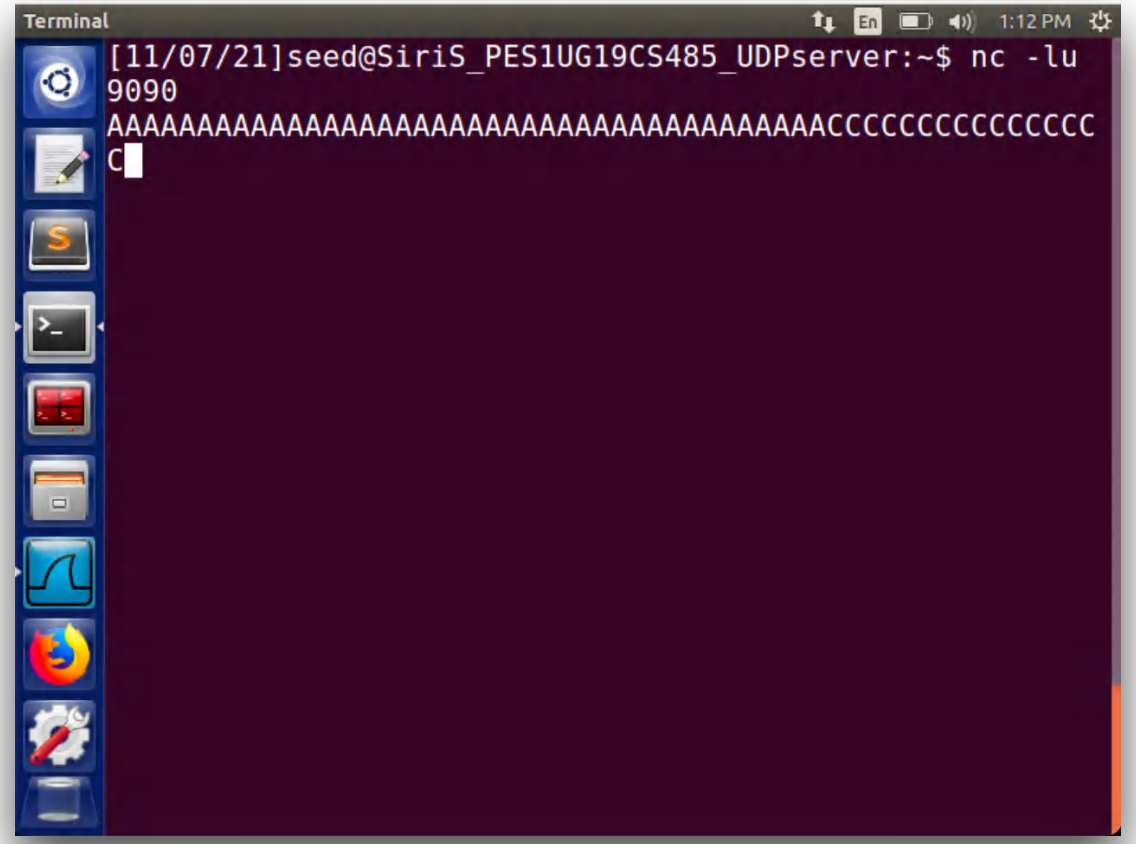


The Wireshark screenshots:





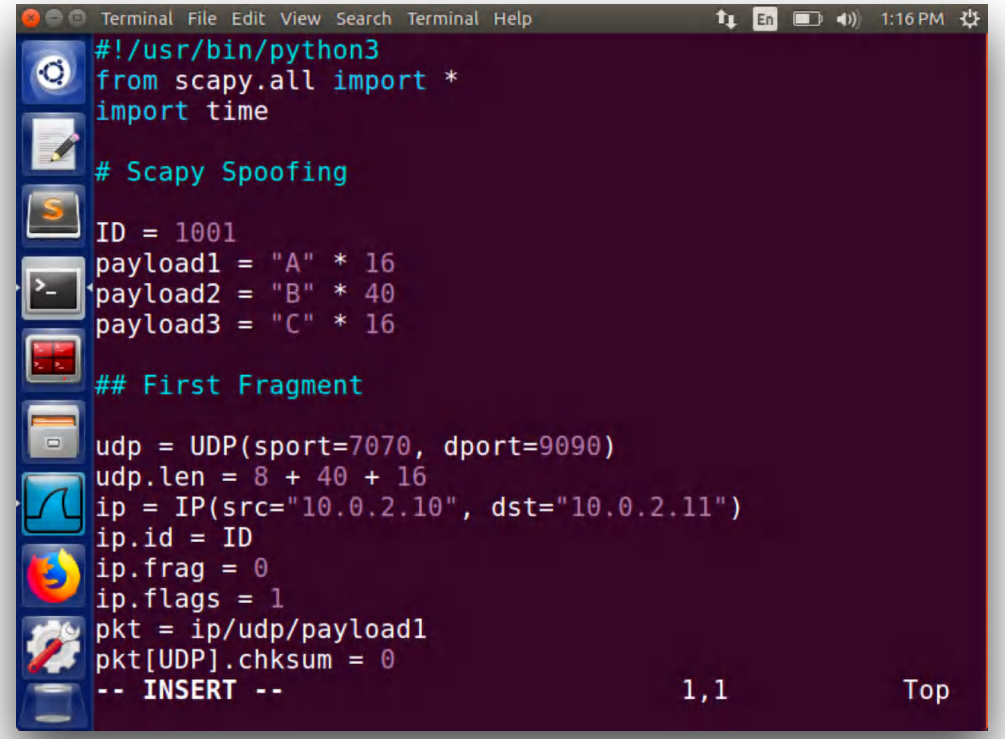
The output on the Server machine:



Here the 2nd fragment is entirely enclosed in the first fragment as the size of the second fragment must be smaller than the first fragment. In the code it is seen that the offset of fragment 3 as the end of fragment1, hence 3 continues right at the end of fragment 1. Here fragment 2 is skipped entirely.

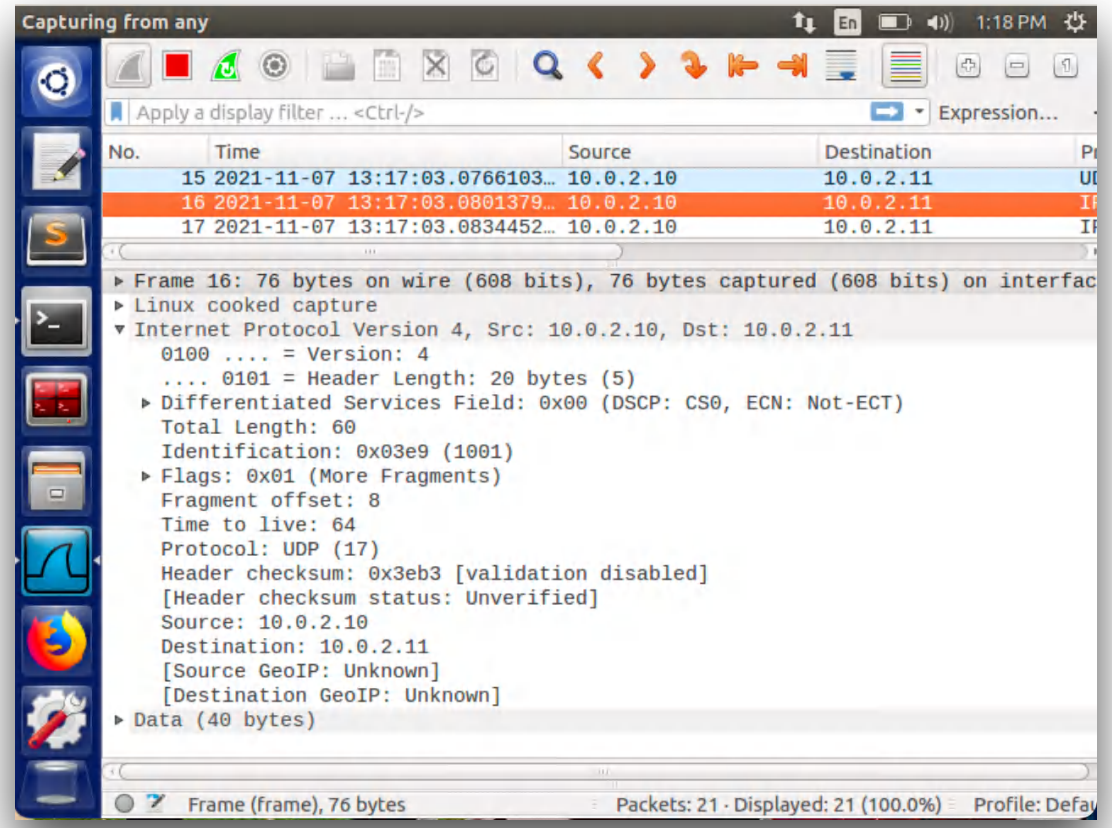
Step 4: If second fragment is sent first and then first fragment in scenario 2 and 3 then what are the observations?

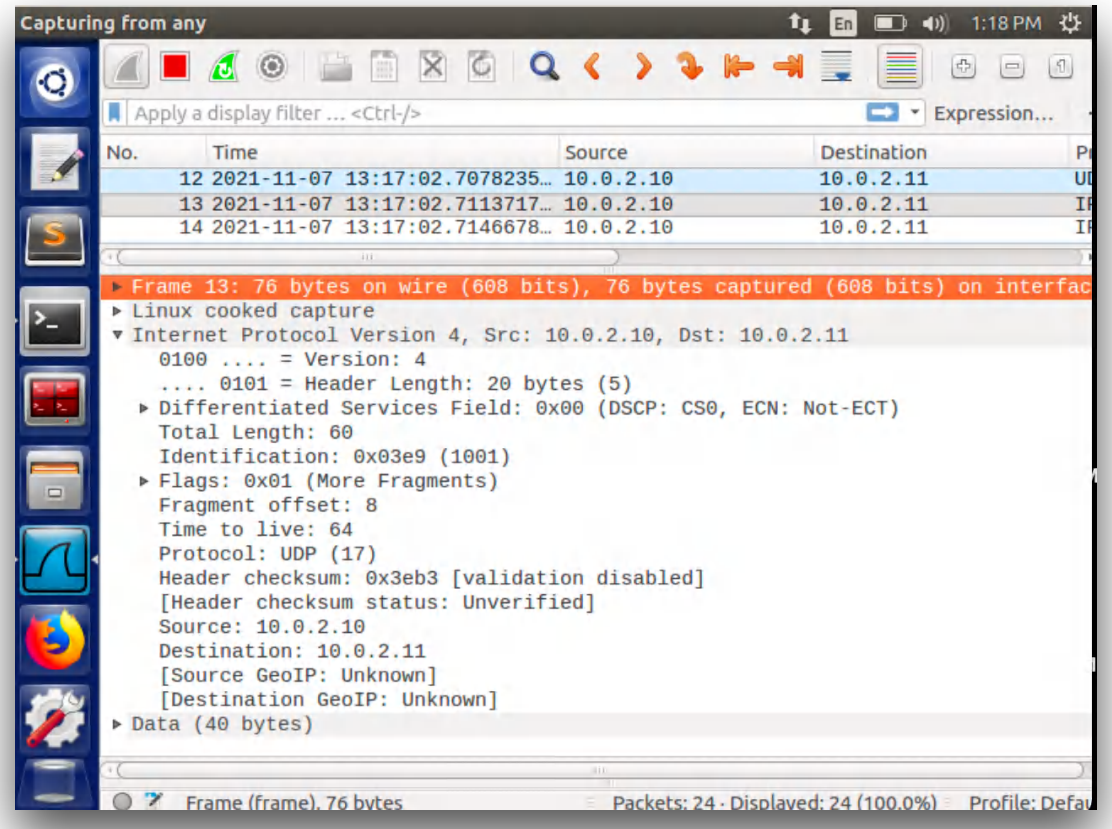
The code used to perform this task:



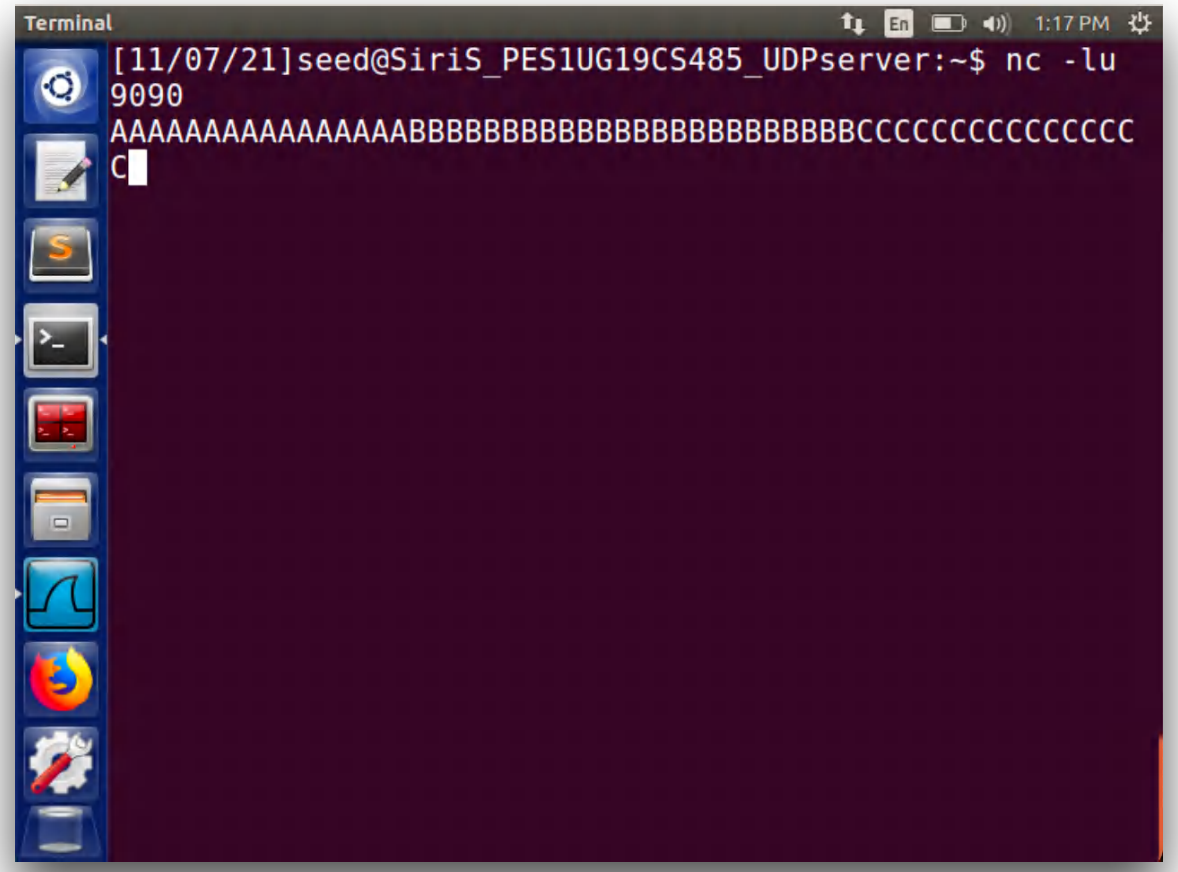
Upon running the code on the Client:

The Wireshark screenshots:





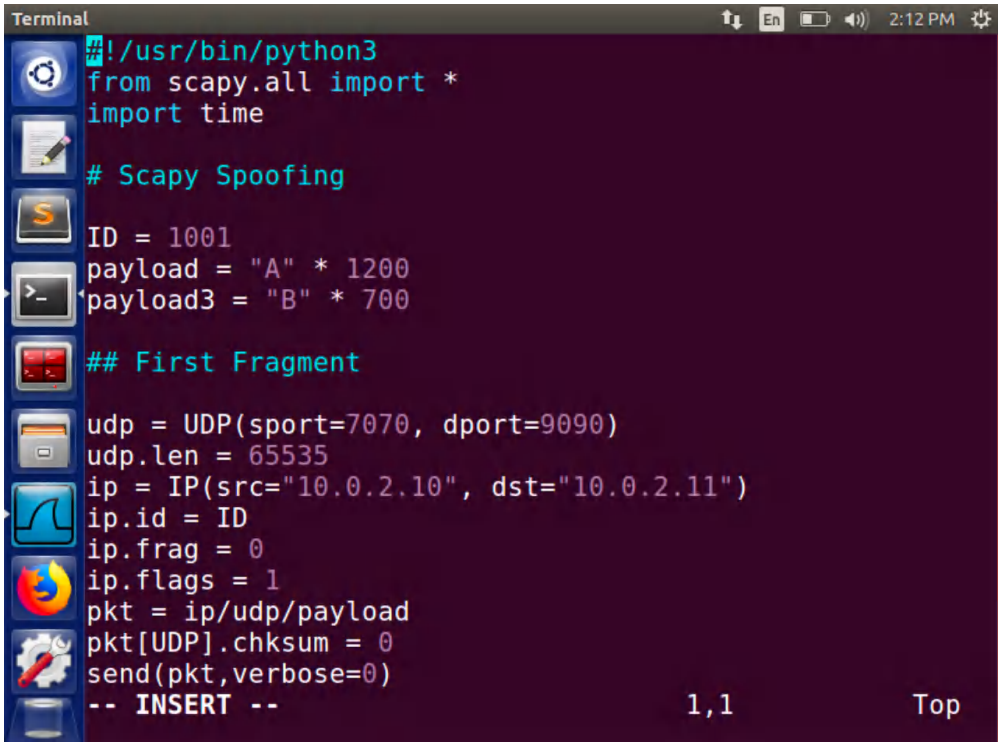
The output on the Server machine:



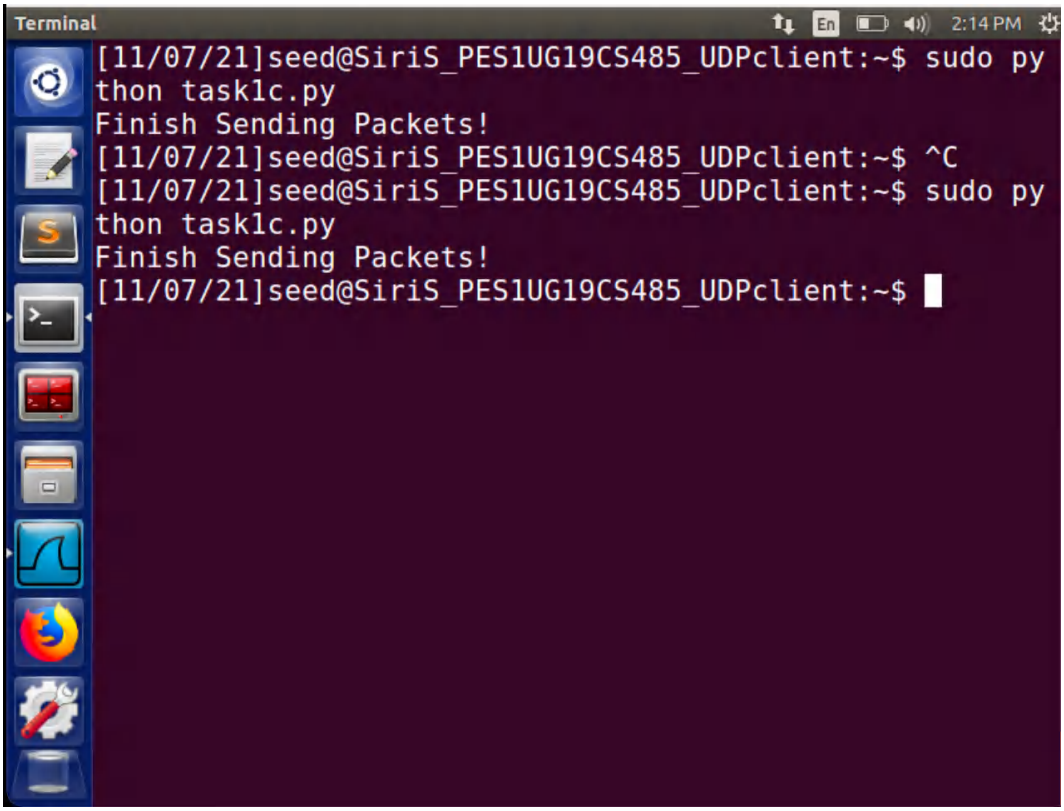
The length of the data is 56 and the first fragment is completely enclosed in the second fragment. This is done by setting fragment offset so that 1 is enclosed in fragment 2 by directly starting fragment 1, after fragment 2 has started and ending it before fragment 2 ends. However, fragment 1 is not overwritten.

Task 1C: Sending a Super-Large Packet

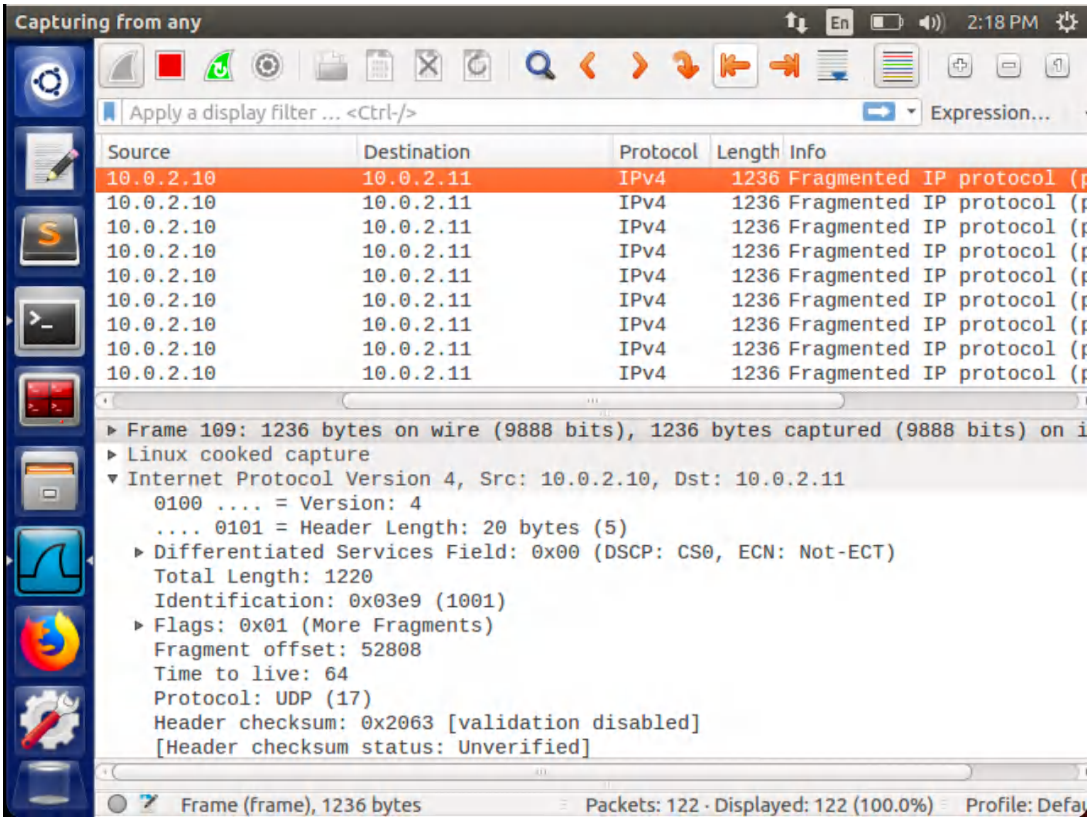
The code used to perform this task:

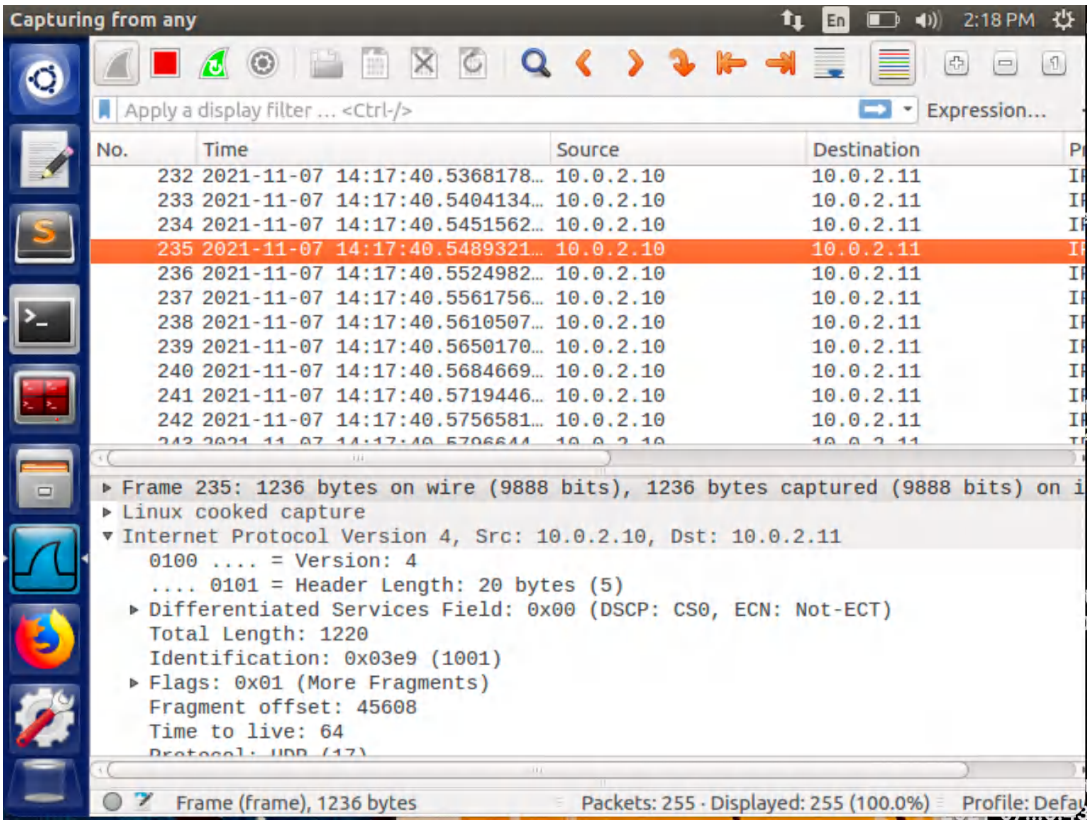


Upon repeatedly running the code on the Client:



The Wireshark screenshots:





The output on the Server machine:

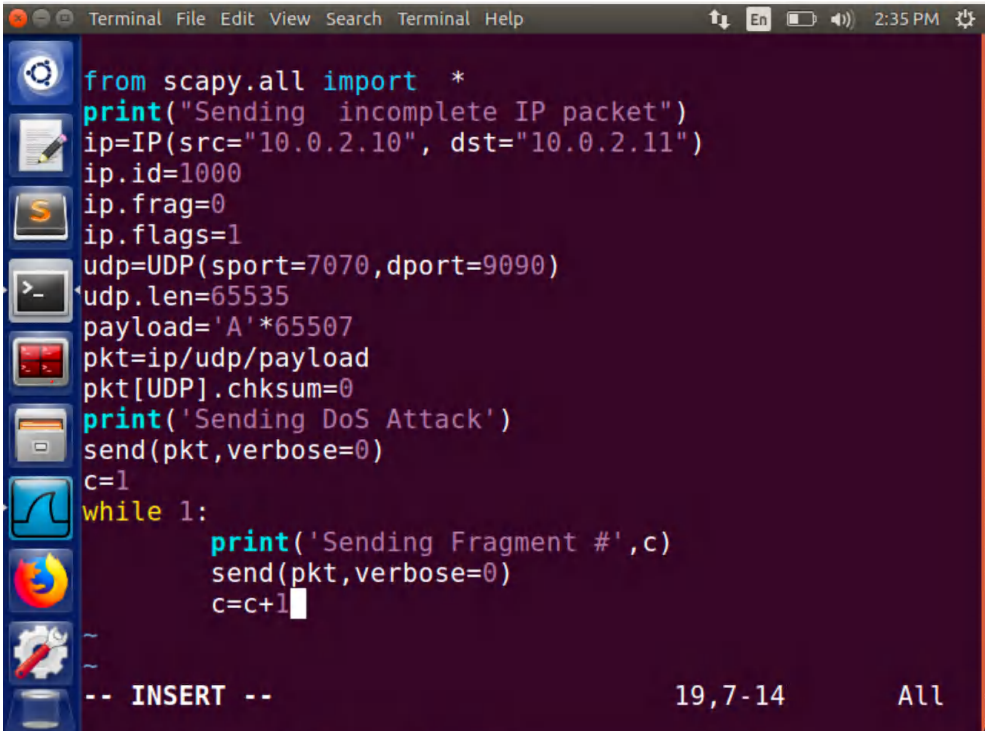
The packet exceeds the limit of the packet length. The maximum length in the code is set to 65535 and anything greater than that is not transmitted. However, here, packets of 800 bytes each are sent, thus exceeding the max packet length.

Hence, the server does not print anything because the packet is super-larger and much greater than the supported packet length.

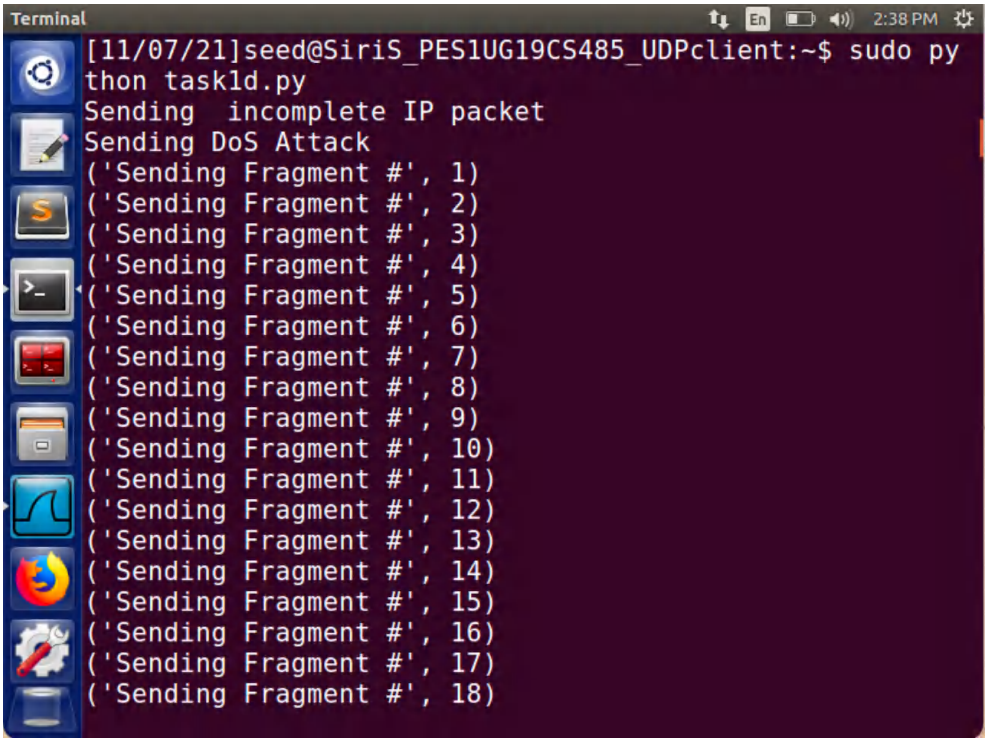
Task 1D: Sending Incomplete IP Packet

A DoS (Denial-of-Service) attack is launched on VM 2 using fragmentation.

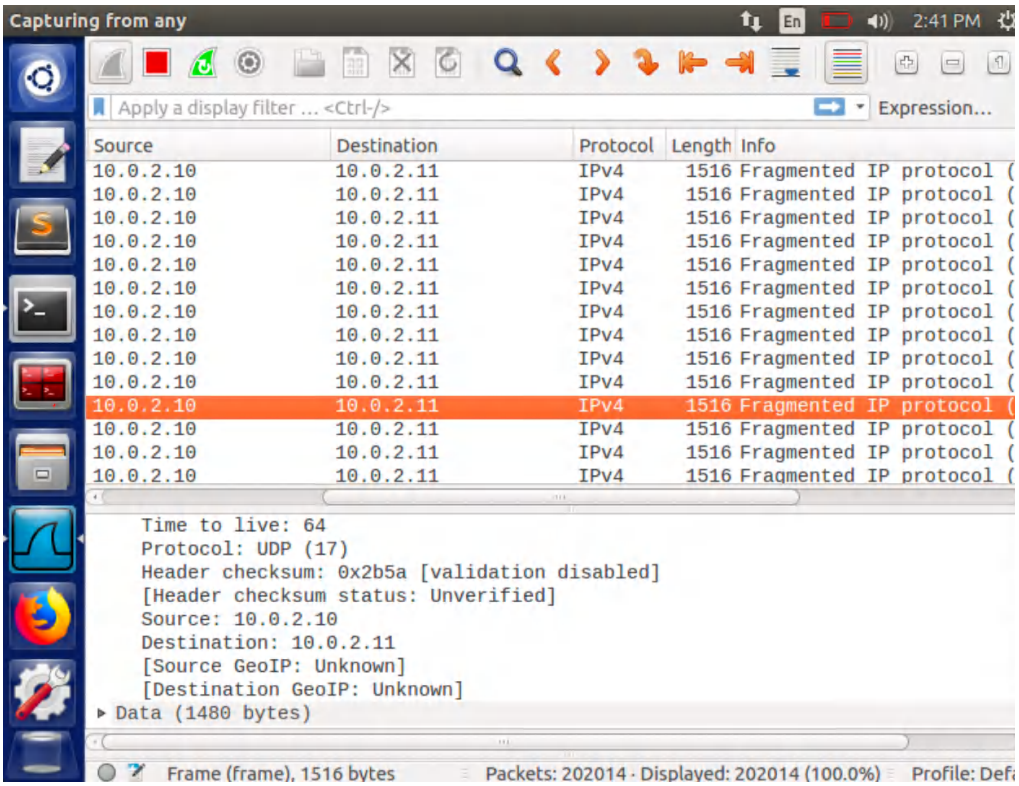
The code used to perform this task:

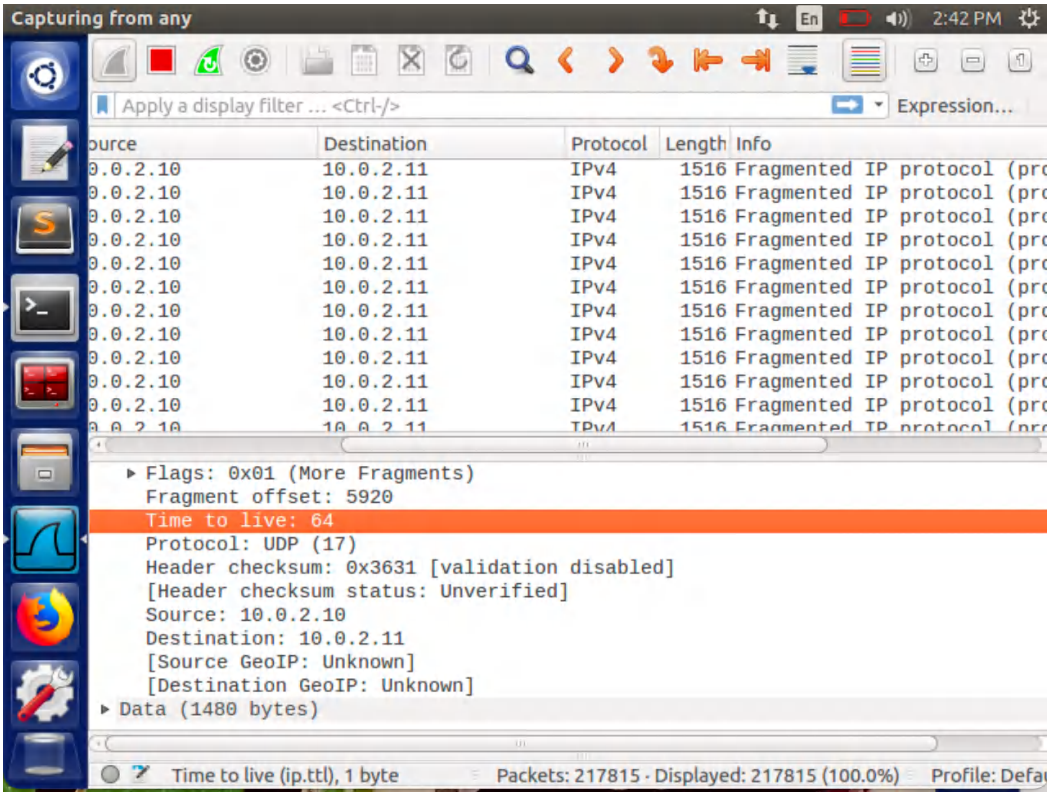


Upon running the code on the Client:



The Wireshark screenshots:



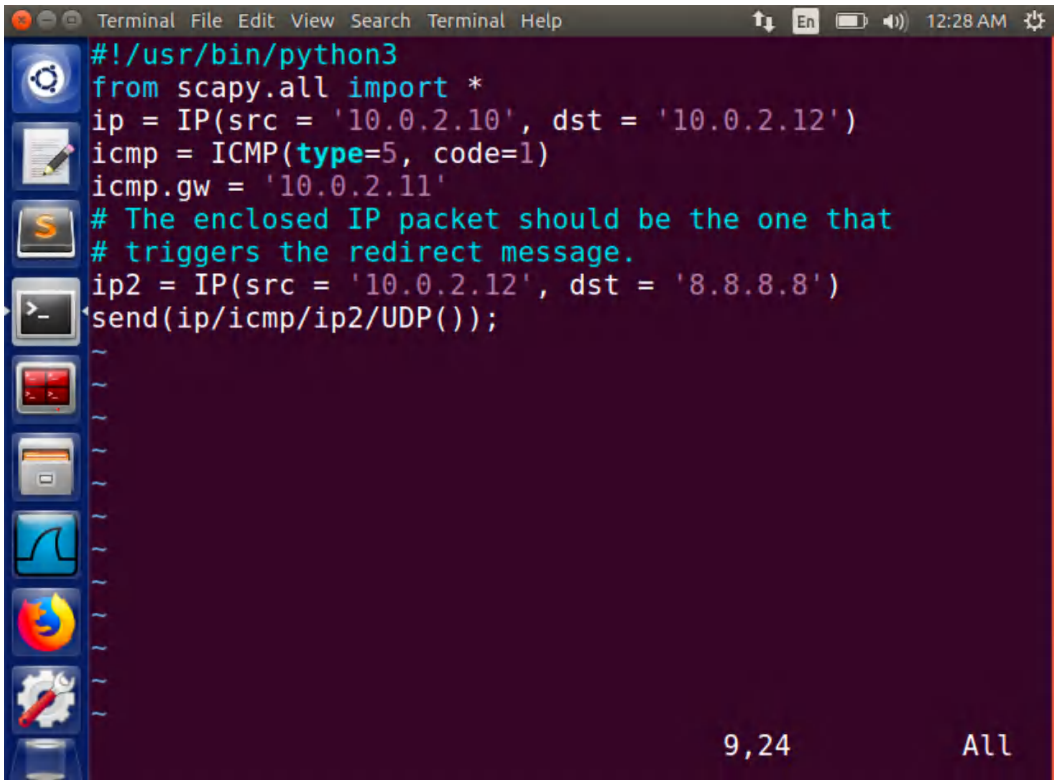


The output on the Server machine:

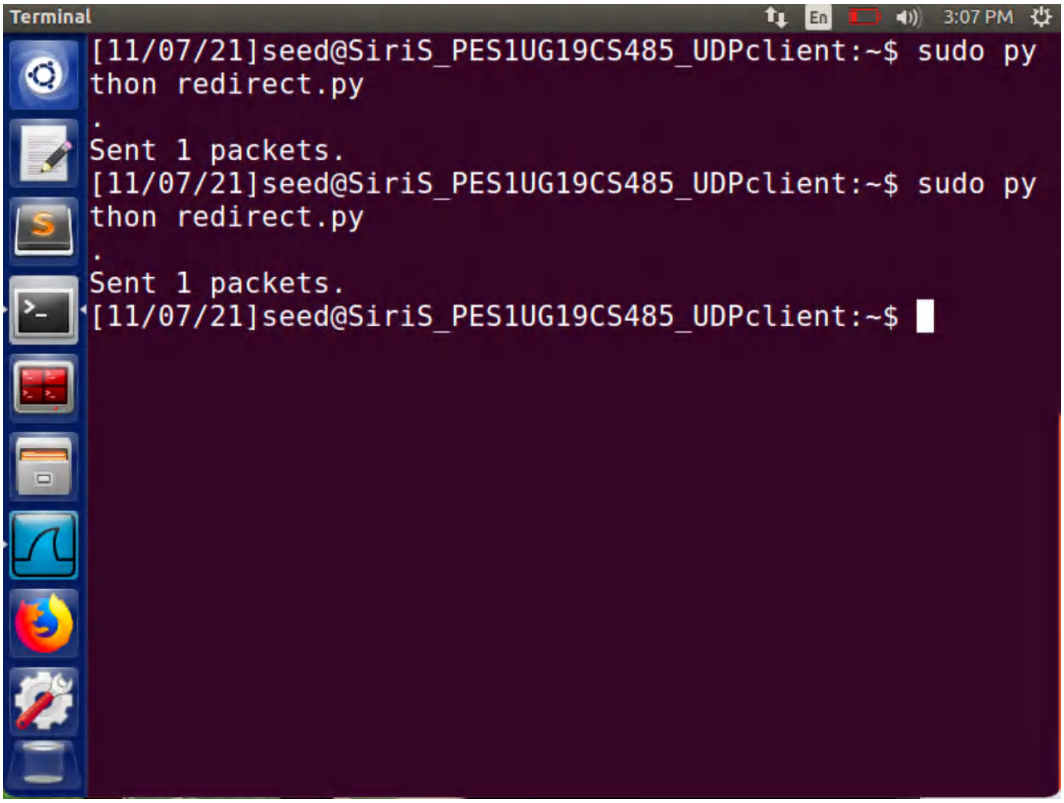
Multiple fragments are sent to VM 2 of multiple packets. What we expect is that the receiver stores these fragments in the kernel memory until it receives all the fragments. However, here it does not receive all the packets, hence it times out.

Task 2: ICMP Redirect Attack

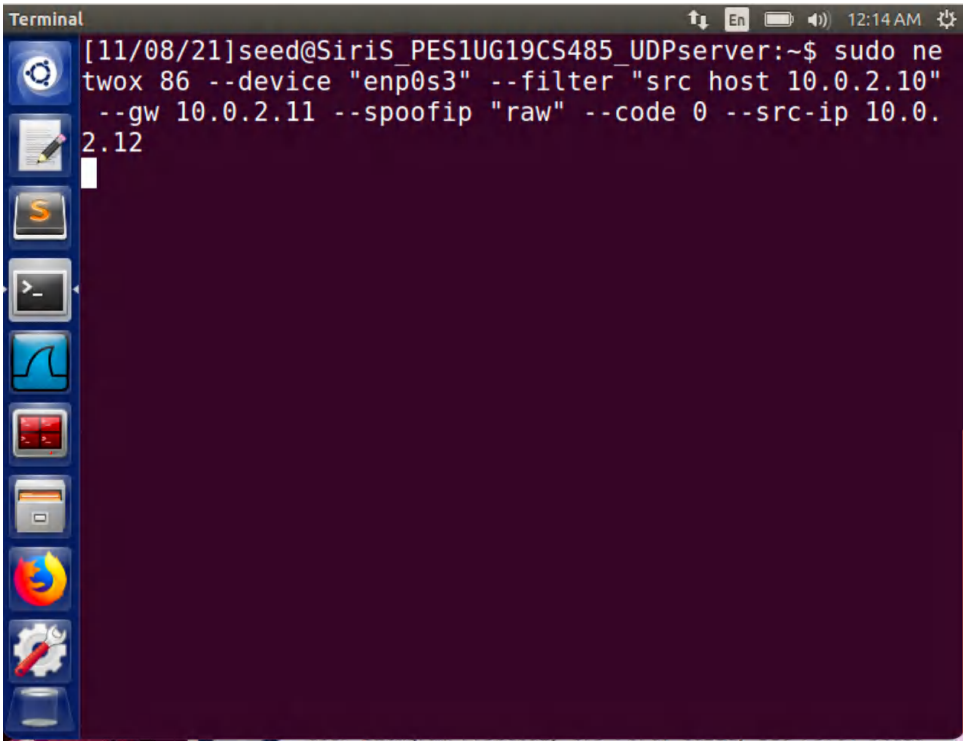
The code used to perform this task:



Upon running the code:

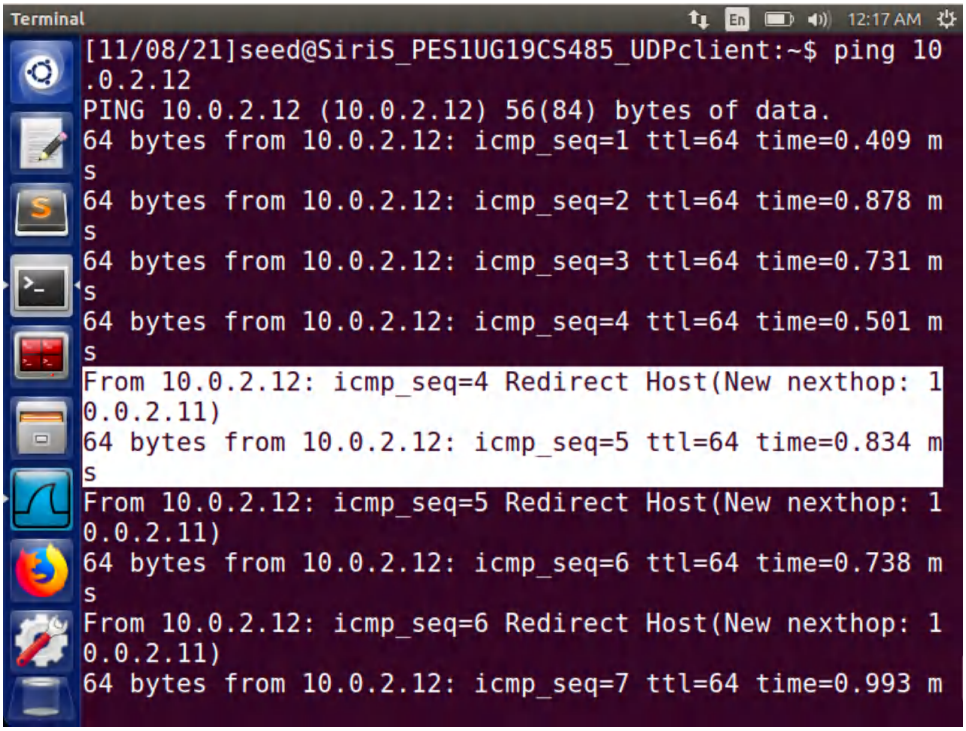


Upon performing the attack:

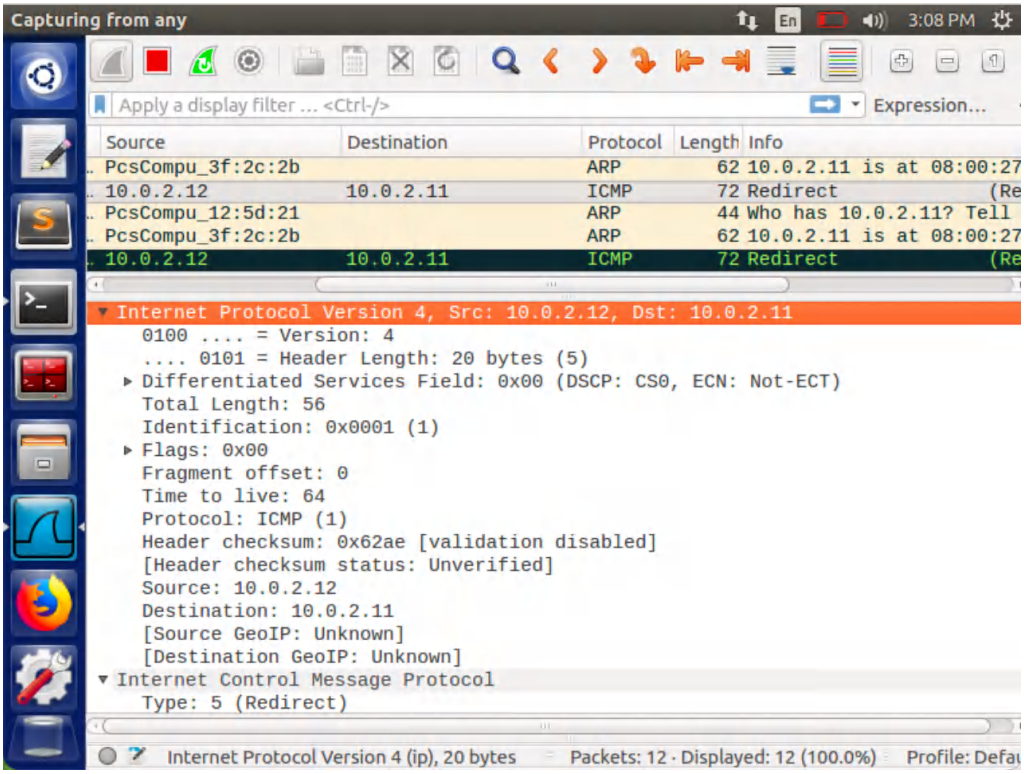


We remove the countermeasure:

When we ping 10. 0. 2. 12, first the packets go through normally, however after performing the attack, the packets are redirected as shown bellow:



Wirershark screenshot:



Therefore, we redirect the IP such that it has the attacker machine's IP gateway. This ensures that the packet gets redirected to the attacker machine and hence the changes are made accordingly. Even in the client VM screenshot we can see the redirect messages being shown.