EECS 4482: Network Security Assignment 1

**My Scapy TraceRoute**

1.1)

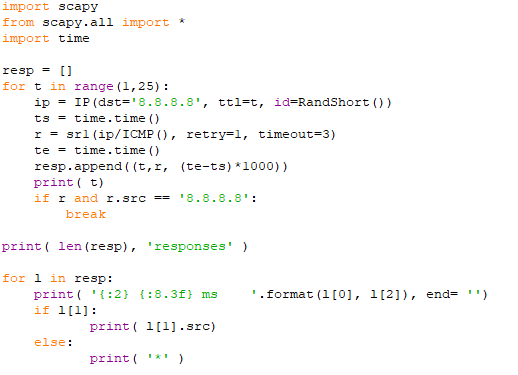


Figure 1 MyTraceRoute completed Scapy code

1.2)

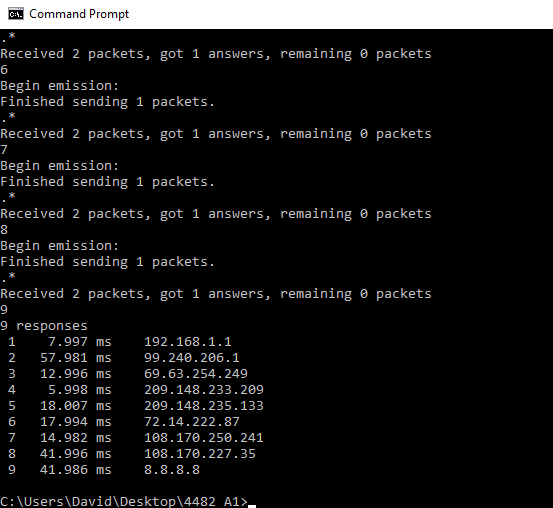


Figure 2MyTraceRoute sample output

1.3.a)

The way this program works, from my understanding, is that it iterates through a for loop from 1 to 25 and sends a packet each iteration with a TTL of the corresponding value. When it reaches the target then the whole traceroute path will be sent pack and the final t value will be the number of packets it took to get to that packet (e.g. In this case, 9 responses so it iterated until 9 and the t value was 9).

While testing out the functionality of this piece of code I found that iterating through 5 to 25 instead, for example, would only give me the traceroute starting from packet 5 (instead of 1). Therefore, it would make no sense to iterate through 25 to 5, and upon trying it I received no packets in my traceroute as it only takes 9 packets to get to 8.8.8.8 from here, and the end range is lower than the starting range.

1.3.b)

The purpose of using a random distribution (e.g. RandShort()) is to generate an ID field that could be identified by the user to determine if the packets sent are separate from each other or are part of the same content and need to be put together in a certain way.

The code would work whether or not a default value was assigned to the ID field because for this traceroute you are just trying to see which IP each router reaches and return the value you back, so they don’t need to know whether are not these packets are related to each other.

**IP Packet Steganography**

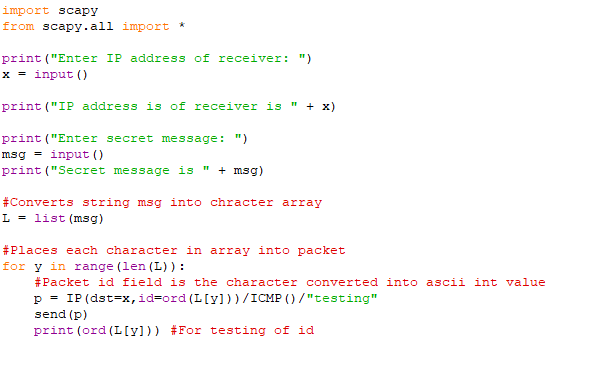


Figure 3 Steganography Sender

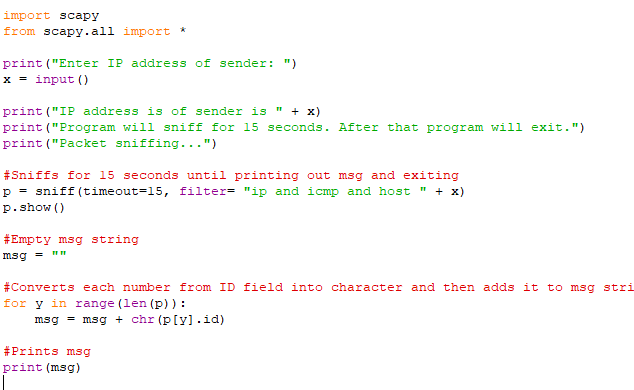


Figure 4 Steganography Receiver

**ARP Poisoning – Forensic Analysis**

2.1)

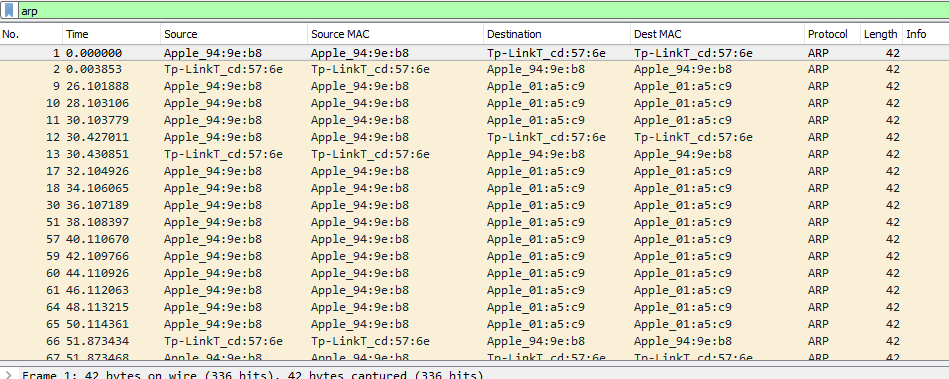
The MAC address of the machine that has conducted the ARP poisoning attack is **APPLE\_94:9e:68 (7c:d1:c3:94:9e:68).** I determined this by filtering packets with the keyword “ARP” to find all the ARP packets. In frame 1 the attacker sends an ARP request and in frame 2 the another machine sends an ARP reply. From that point on there are a flood of ARP replies detected from the attackers MAC address making it likely that this is the attacker.

Figure Flood of ARP replies after the initial ARP request and reply in frame 1 and 2

Furthermore, in frame 9 you can see that the attacker is now spoofing the IP of the other machine for the flood of ARP replies.

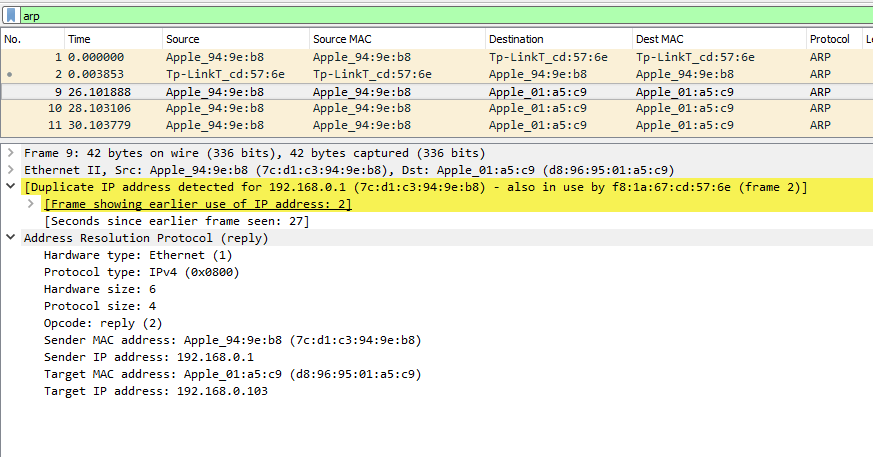


Figure In frame 9 the first instance of duplicate IP addresses appears, which suggests IP spoofing by the attacker.

2.2)

The IP address of the attacker machine is likely **192.168.0.100**.

Using the previous information figured out it is apparent that the attack is coming from the apple MAC address stated in 2.1. To determine the attackers IP address I looked for the packet that did not appear with a duplicate IP. This is found in frame 1, which is the initial ARP request of the attacker and shows his IP address.

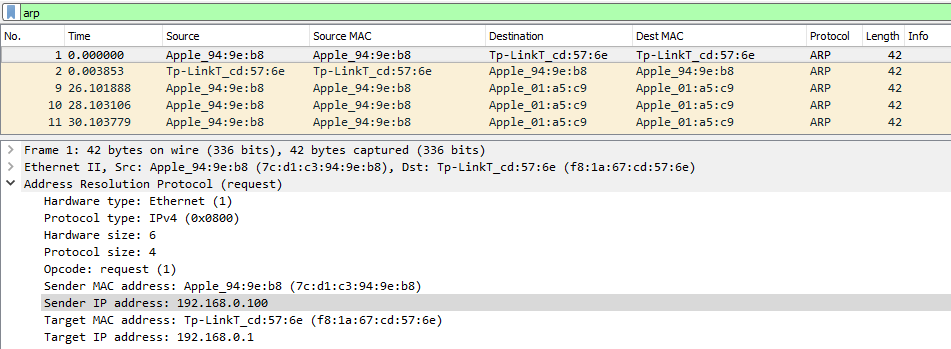


Figure The first appearance of a packet that does not happen to have a duplicate IP address is frame 1, which displays the attackers actual IP address.

2.3)

The MAC address of the of the victim machine is **Apple\_01:a5:c9 (7c:d1:c3:94:9e:b8)**. The IP address of the victim machine is **192.168.0.103**.

With the information from 2.2 and 2.3 it is clear who is the victim. The victim is the machine that is being bombarded with ARP replies from the attacker using the spoofed IP address. This flood of ARP replies is in order to ensure that the victims ARP table stays changed with the spoofed IP address. The victims MAC address and IP address was found by looking at any one of the many ARP replies that were being sent.

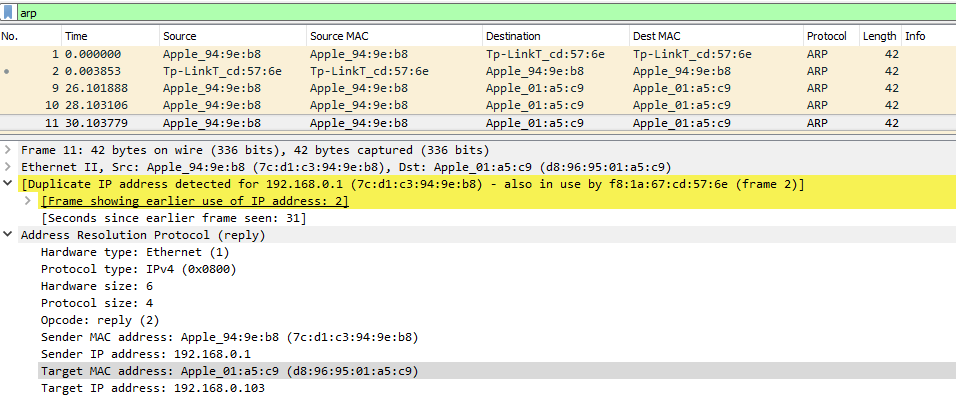


Figure Frame 11 is one of the many examples where you can see the victims MAC address (Target) and IP address (Target).