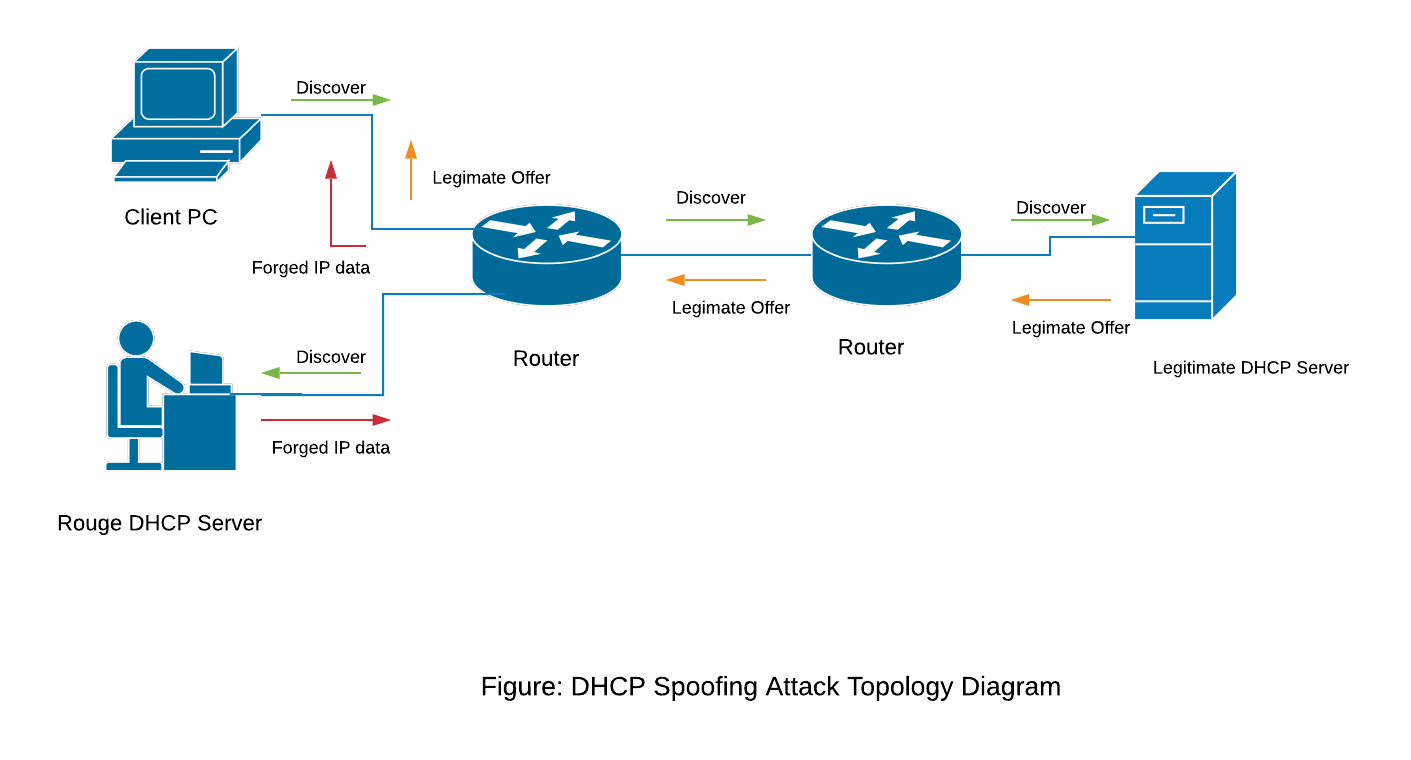
**DHCP Spoofing Attack**

(Shamiul Hasan - 1505038)

**Definition:**

In ***DHCP Spoofing Attack***, An attacker can spoof the DHCP server and send forged replies to the client with fake network settings allowing the attacker to intercept upcoming client’s communication.



**Steps of Attacks:**

1. At first, the dhcp\_spoofer.py code is run. This code completes the DHCP spoofing attack. This file sniffs the packets at 67 and 68port of the attacker PC and send appropriate replies to them to behave as a DHCP server.
2. Dhcp\_spoofer.py at first looks for ‘DHCP discover’ packets from the victim and if it finds it, it replies with a rogue offer packet based on the received packet.

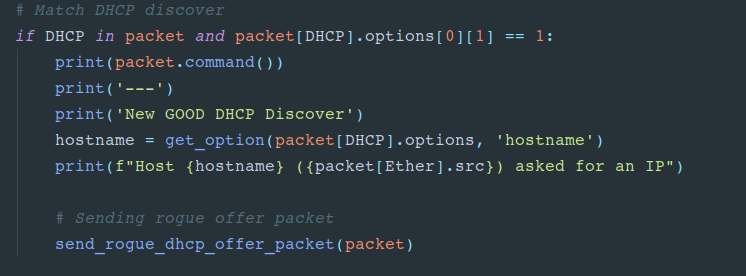


Figure: DHCP Discover sniffing

1. When the victim device receives the DHCP Offer packet, it sends the request packet to the rogue DHCP server and the Rogue server replies with an appropriate ACK packet and thus the connection is established.

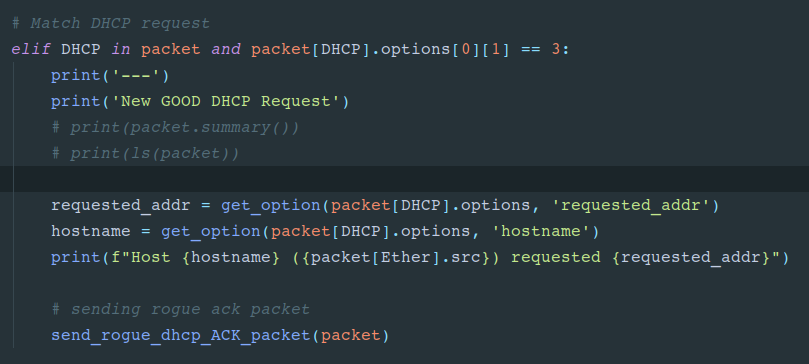
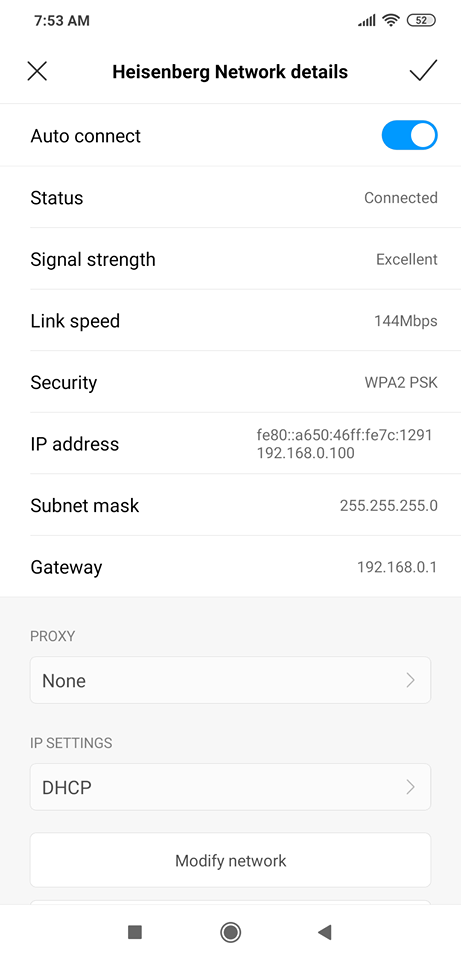


Figure: DHCP request checking and replying with rogue ACK

1. In this whole process, attacker set the gateway of the victim PC as its own IP. So, once the connection is established, whenever victim passes a packet, it will go through Rogue PC.

**Attacker and victim snapshots:**

****

**Figure:**  My android device at normal condition, connected to the good dhcp server and default gateway is 192.168.0.1 and IP is 192.168.0.100

Now we run the dhcp\_spoofer.py to perform the attack.

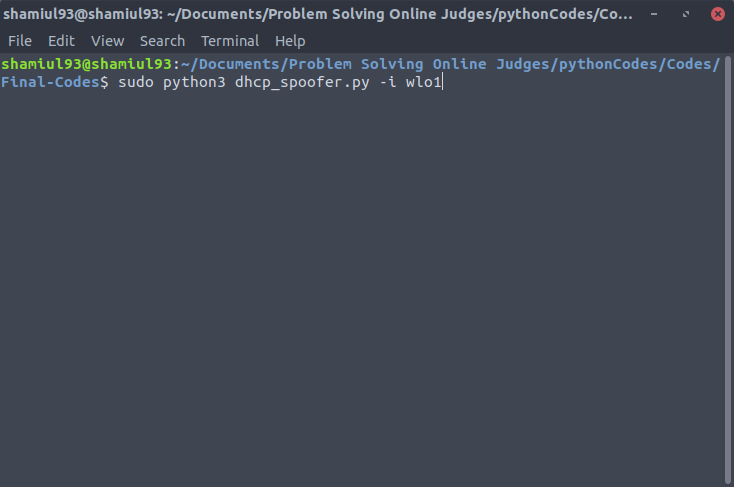


Figure: Command to run the dhcp\_spoofer.py

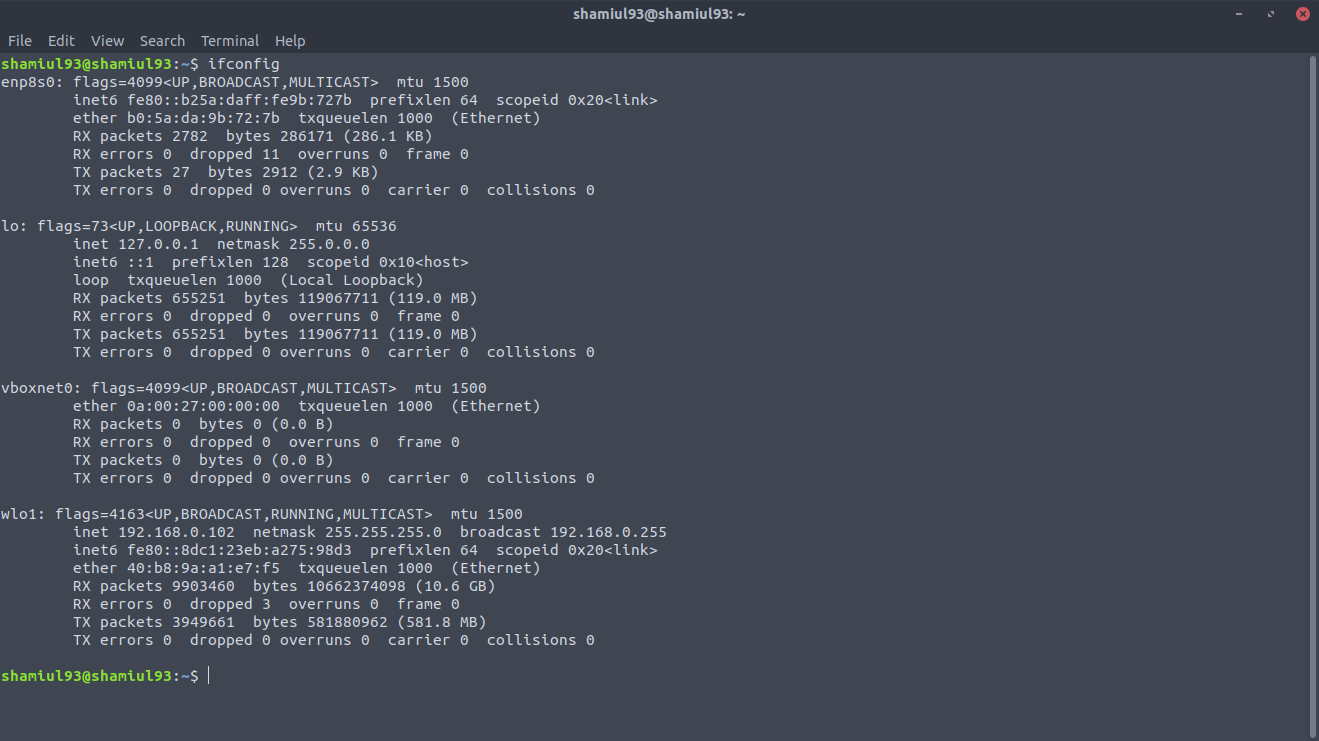


Figure: Attacker PC IP configuration

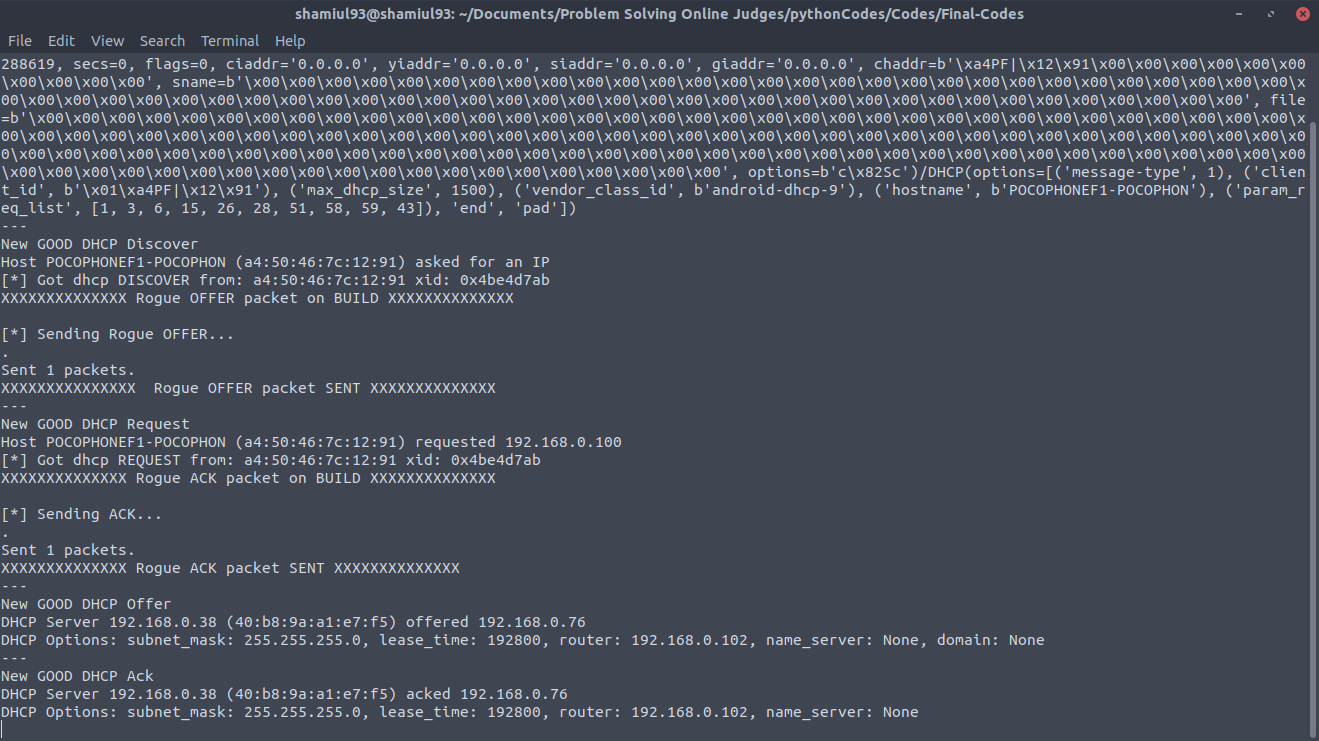
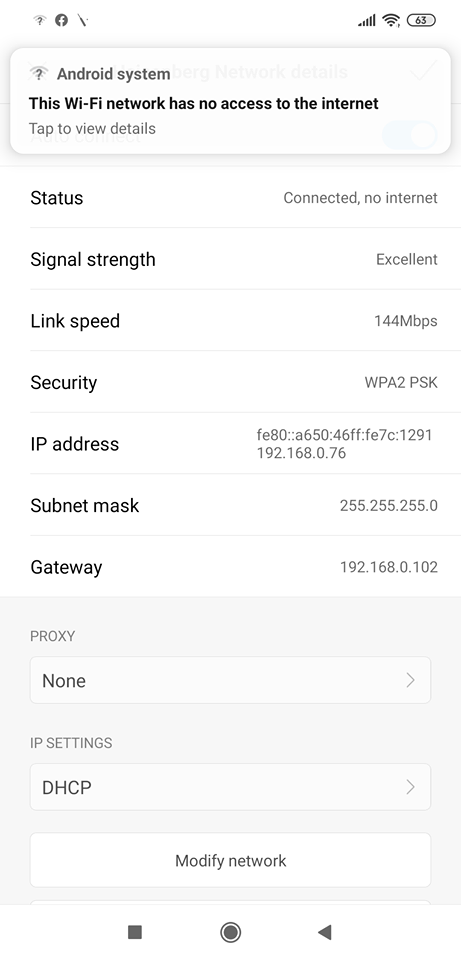


Figure: dhcp\_spoofer.py is sniffing for DHCP packets after running



In this image, an android phone tried to get connected to the ‘Heisenberg’ wifi network. Default gateway set by our real DHCP server is 192.168.0.1 and the ip pool of the good dhcp server is from 192.168.0.100-192.168.0.110.

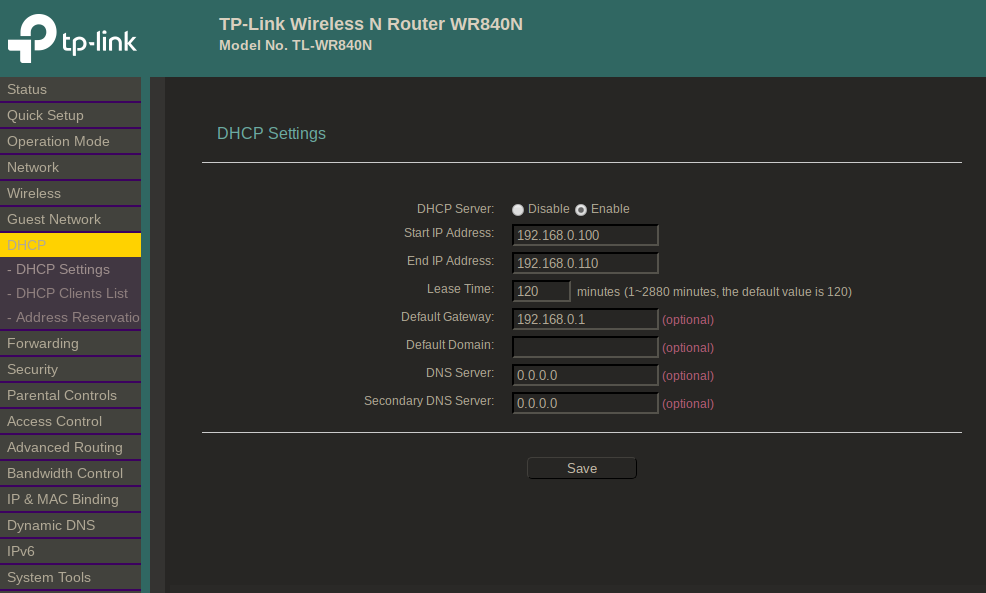


Figure: DHCP configuration of the router.

But if we look at the android device wifi configuration, we can see that, the assigned IP address is 192.168.0.76 which is out of good dhcp server’s ip pool and set by us. The default gateway is also set as 192.168.0.102 which is attacker PC’s ip assigned by good DHCP server instead of 192.168.0.1. So, we can say that our attack was successful and spoofing is done.

**Problem faced:**

I tried this attack on various devices and got good results on android devices and other devices who aren’t connected to the wifi network now and weren’t connected either in about previous half an hour or so. I looked into the matter and found out, modern OS and some modern mobile devices takes a countermeasure against DHCP spoofing by skipping Discover and Offer steps and directly sending Request packets to the trusted DHCP server with the last IP the device was assigned. Even if attacker replies with a proper ACK packet, modern device will ignore it totally. But if the PC is getting connected to the network for the first time in a while, my attack works successfully on it.

**My attempt to solve this problem:**

I tried to run a DHCP starvation attack on the router (good DHCP server) to fill up it’s ip pool so that even if victim device sends request packets to the good dhcp server, good dhcp server has no more ip to assign. So, the victim will definitely have to connect with rogue server. But the problem is, after starvation, router is overloaded and it gets down. It’s access point is vanished and can’t be seen from other devices to connect. As the attacker server works only when the victim tries to connect to the router, our attack can’t be done if the router itself can’t be found.

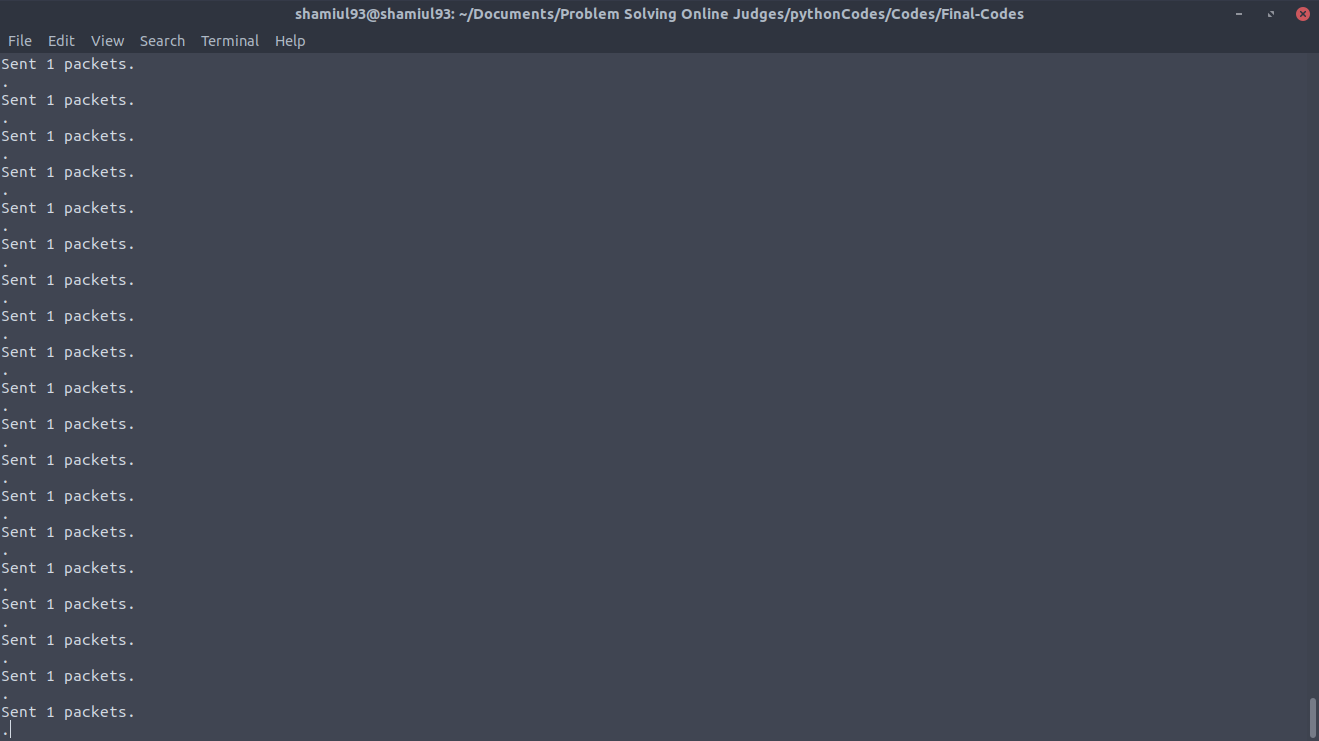


Figure: Request\_starve.py is running DHCP starvation on the good dhcp server.

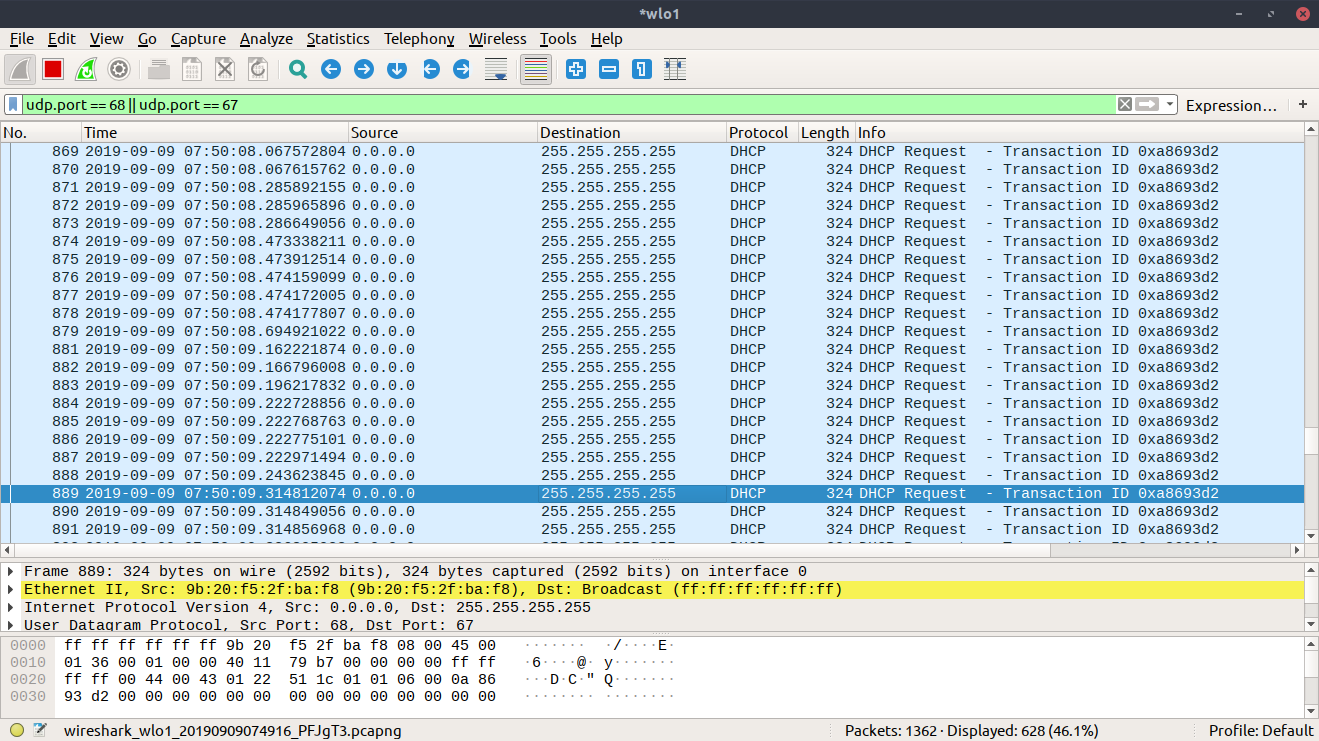


Figure: After starvation attack thousands of DHCP request goes to the good dhcp server and makes it busy.

After this attack, the router stops internet connection to all the connected devices and doesn’t show up in other devices’ available wifi network list.

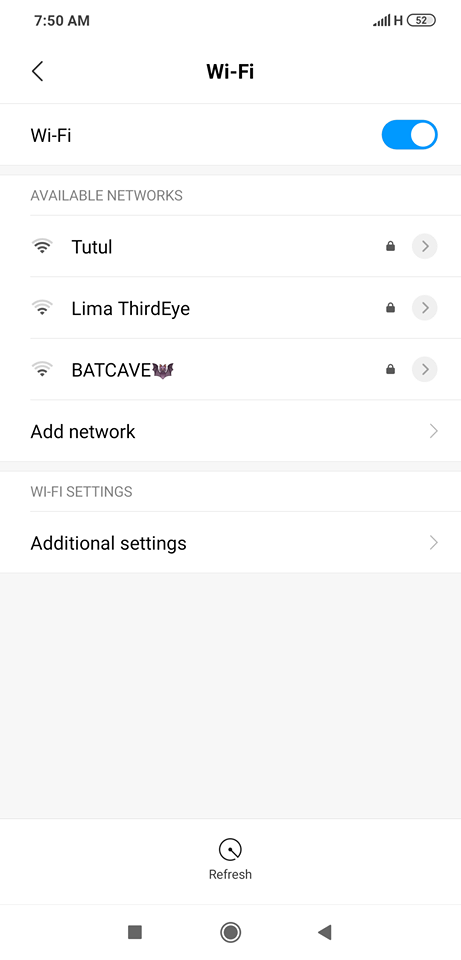


Figure: Wifi ‘Heisenberg’ is vanished from the available wifi list after starvation attack.

**Countermeasure:**

I made a naive but working countermeasure against it. The idea is implemented on the victim side. Victim will send a DHCP discover packet in the network. Now it will listen at the 53 port. Both good and bad (if any) DHCP servers will reply to the Discover packet with an Offer packet. Victim will count these offer packets and if the offer count is more than 1, victim can be sure that there are two DHCP servers in the network. So, there might be a possibility of DHCP attacks. It can be cautious to not getting connected to the network.

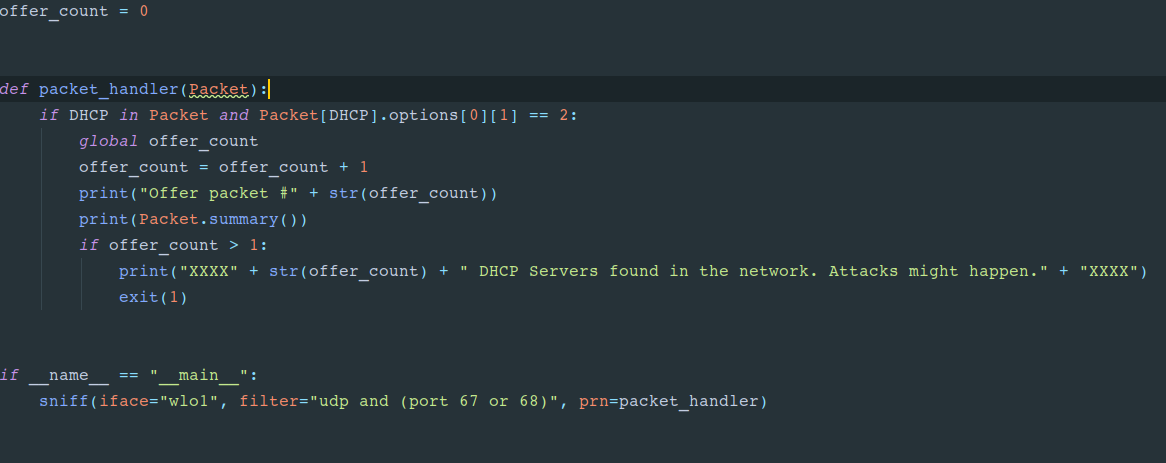


Figure: code snippet from counter\_measure.py

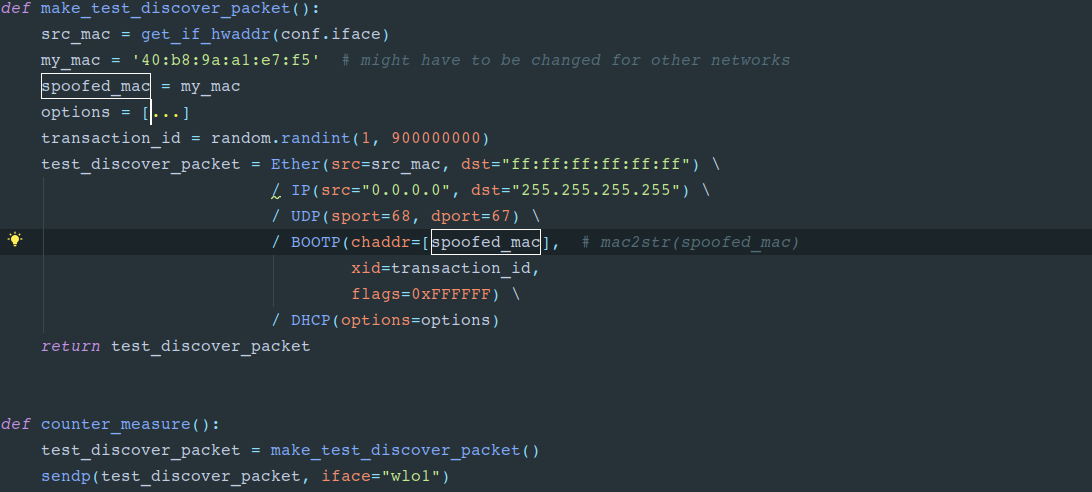


Figure: code snippet from send\_discover.py

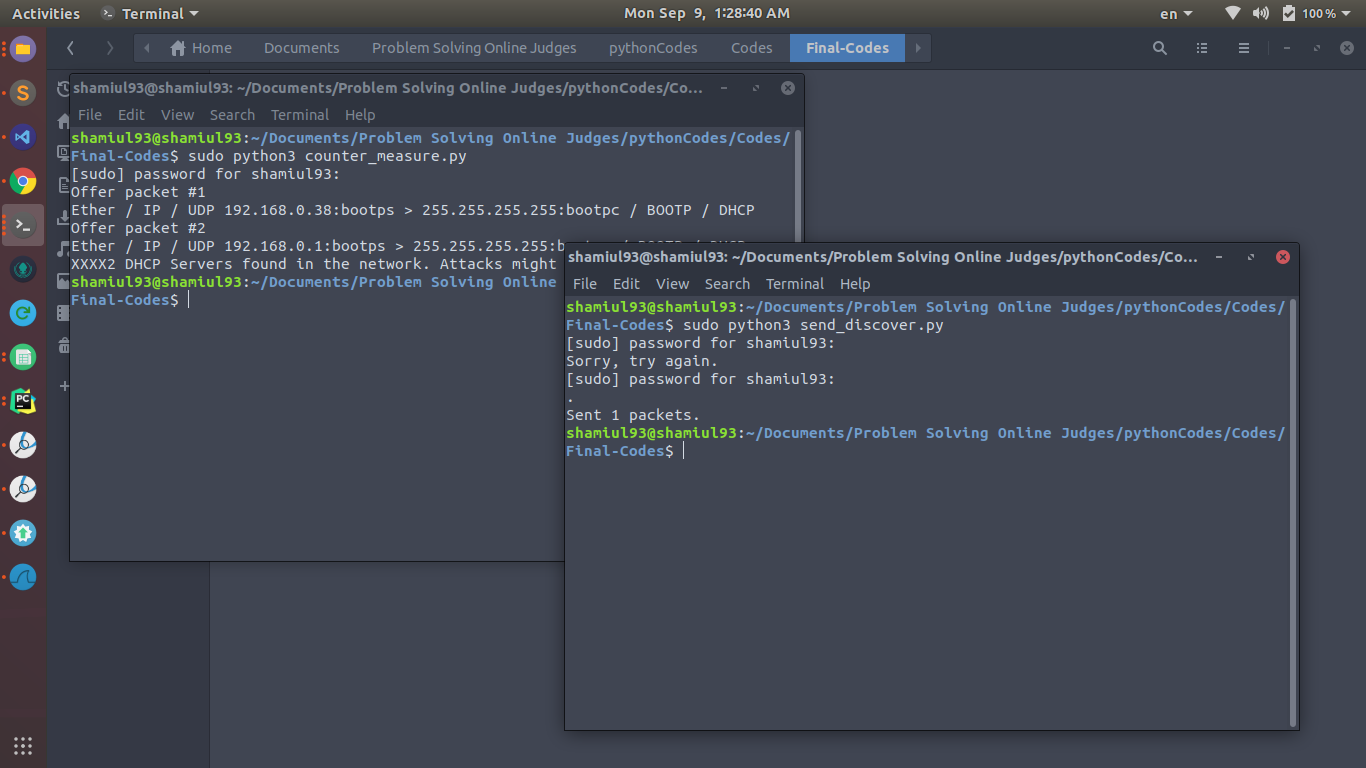


Figure: Countermeasure against the DHCP spoofing attack.

Here, counter\_measure.py listens on the 53 port of victim and increases the counter whenever it gets an offer packet. Send\_discover.py just sends a discover packet to the network. We can see, counter\_measure.py found out 2 DHCP offer packets where one if from 192.168.0.1 (good dhcp server) and another one from 192.168.0.38 which is attacker’s fake ip assigned by dhcp\_spoofer.py.