

Network Security

LAB 1: DHCP Starvation Attack

DHCP stands for Dynamic Host Configuration Protocol which is a connectionless network protocol which allows the server to dynamically assign IP address to the incoming hosts and allows them to communicate with other IP networks. Apart from this, it also assigns various other important information to the client system which include subnet mask and gateway.

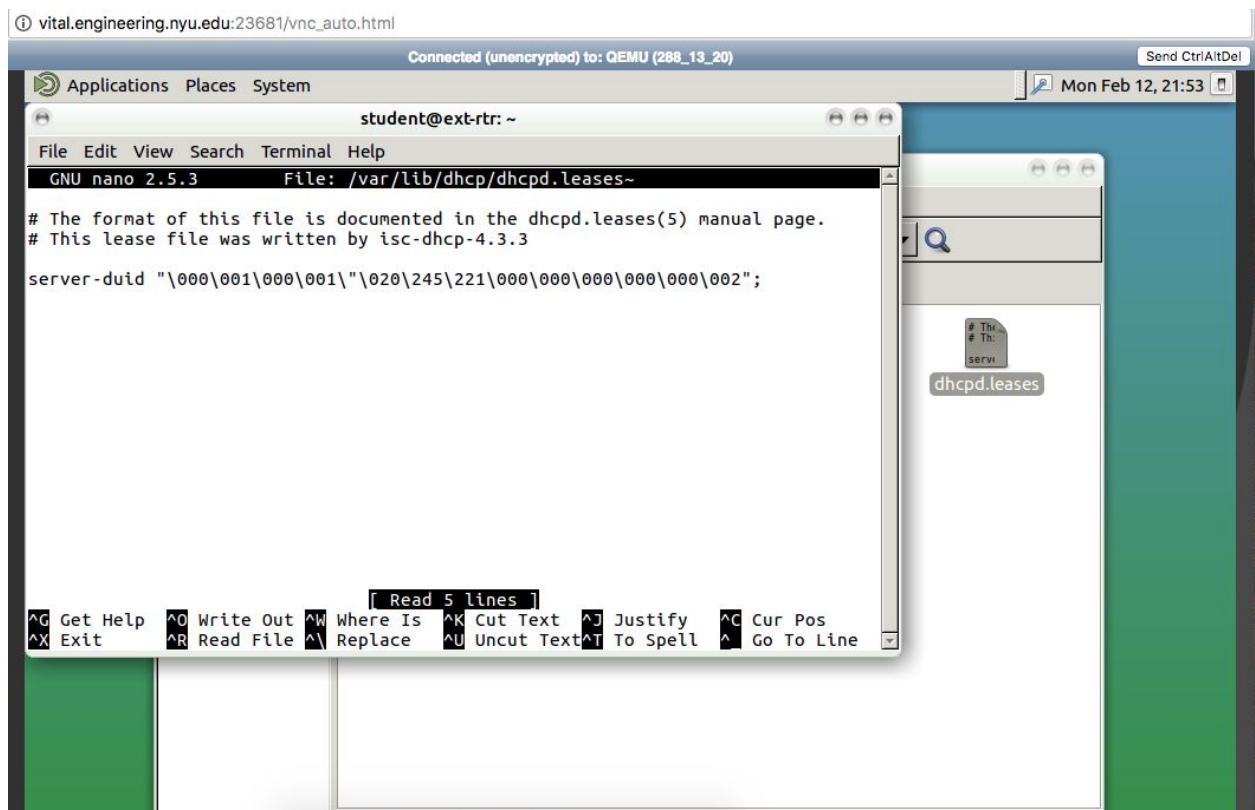
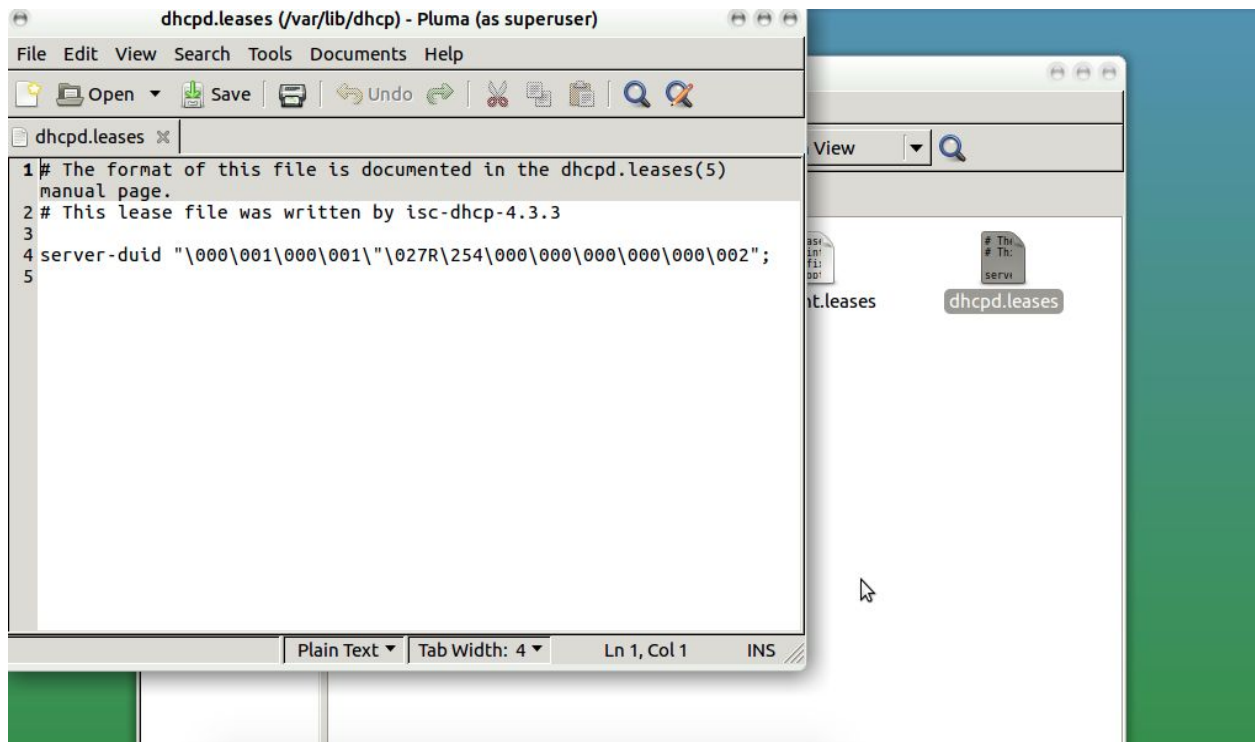
DHCP process to configure client:

1. **Discover** : Client broadcasts a request to search for the DHCP server.
IP address for client is given as 0.0.0.0 and for server it is given as 255.255.255.255. Client sends its physical address as source MAC (Media Access Control) address and destination as FF.FF.FF.FF IP address.
2. **Offer** : Offer packet is transmitted by the DHCP server to the client with the destination address as its own IP address so that the requester client can only accept it.
3. **Request** : When the client got the offer from DHCP server, it issues a request packet to get the IP address.
4. **Acknowledge** : After getting the request packet, DHCP server then acknowledges the message after receiving the request packet and assigns the IP address to the client with other important informations.

Steps for DHCP starvation attack:

1. Import scapy.all package to include DHCP, IP, sniff and other necessary commands.
2. Write a Python code to exhaust all the available IP addresses which include functions:
 - a. **Storage_DHCP()** to store MAC address after being spoofed and IP addresses from 10.10.111.100 to 10.10.111.200 excluding 10.10.111.101 and 10.10.111.1 is the DHCP server.
 - b. **req_Maintain()** to maintain the ACK and NAK for all requested IPs'.
 - c. **Sniff()** to perform sniffing of UDP traffic.
 - d. **proc_start()** for the process to start.
 - e. **starvation()** to start the actual starvation process.
3. Setup the SFTP to transfer the DHCP python code from my local machine to the VM by accessing the root/home folder.
4. Re-image the external router machine in-order to have an empty **dhcpcd.leases** file. We need to make sure we do not have any cached binding in the file. Also, make sure to

have no entries in the **dhcpd.leases~** file which can be accessed and modified using the terminal (delete if any entries exist from both the files).



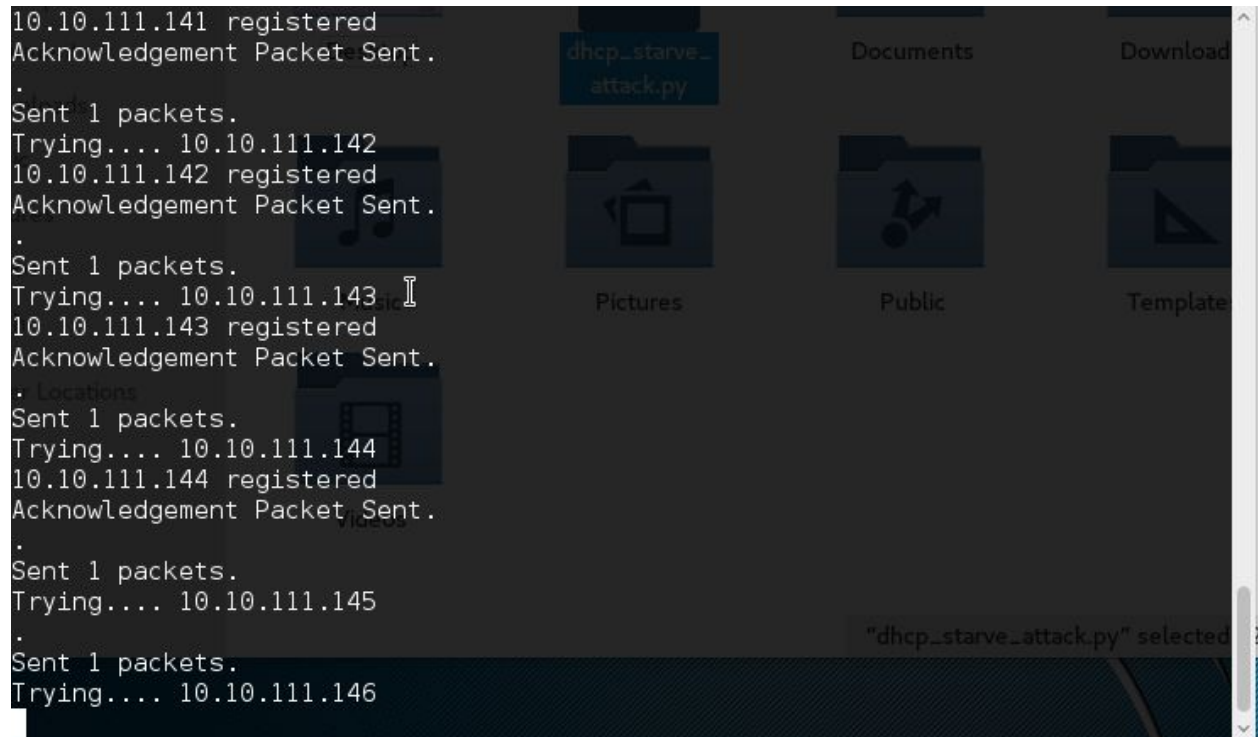
5. Start the Kali Machine and then, open the terminal. We type the command **sudo ifconfig** in the terminal to get the IP address assigned to the Kali machine which we will need to hardcode in the Python code.

```
student@kali:~$ sudo ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.111.100 netmask 255.255.255.0 broadcast 10.10.111.255
    inet6 fe80::200:ff:fe00:3 prefixlen 64 scopeid 0x20<link>
    ether 00:00:00:00:00:03 txqueuelen 1000 (Ethernet)
    RX packets 93 bytes 13218 (12.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 36 bytes 4429 (4.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

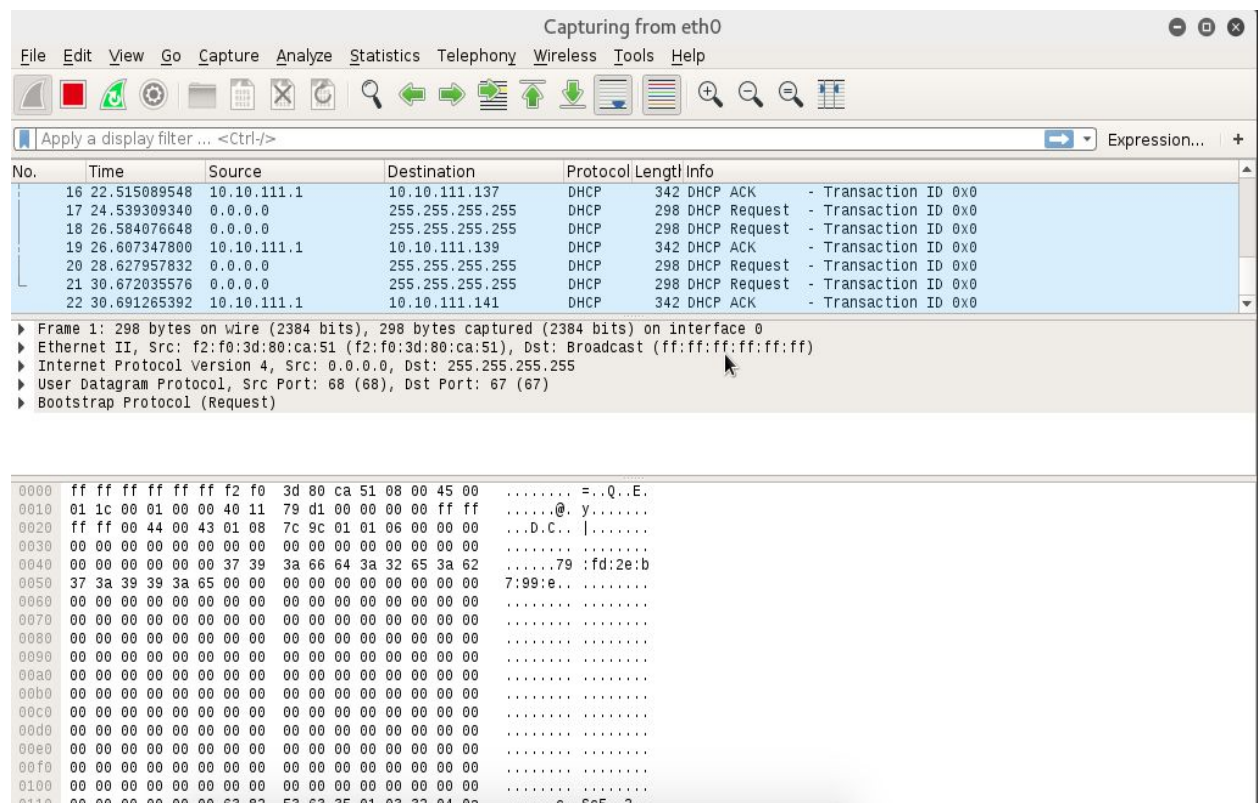
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 0 (Local Loopback)
    RX packets 26 bytes 1612 (1.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 26 bytes 1612 (1.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

student@kali:~$
```

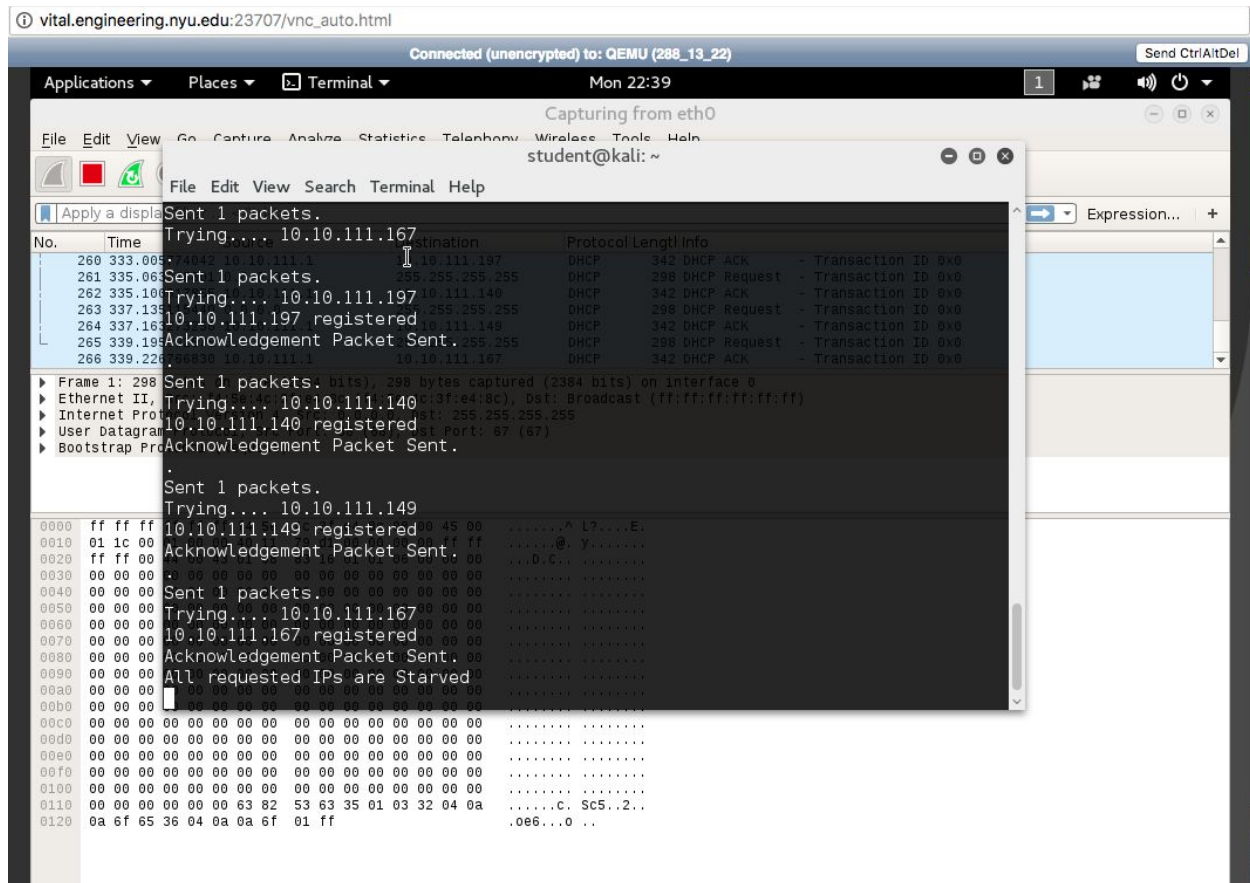
6. Next step is to get the Python code from my personal Machine to the Kali VM using the **SFTP server**. We need to make sure that we are connected to the NYU server and then write the get command to get the python code to the VM.
7. Python code is then executed in the Administrator mode using the command **sudo python dhcp_starve_script.py**
Python code on execution assigns all the available IP addresses dynamically.



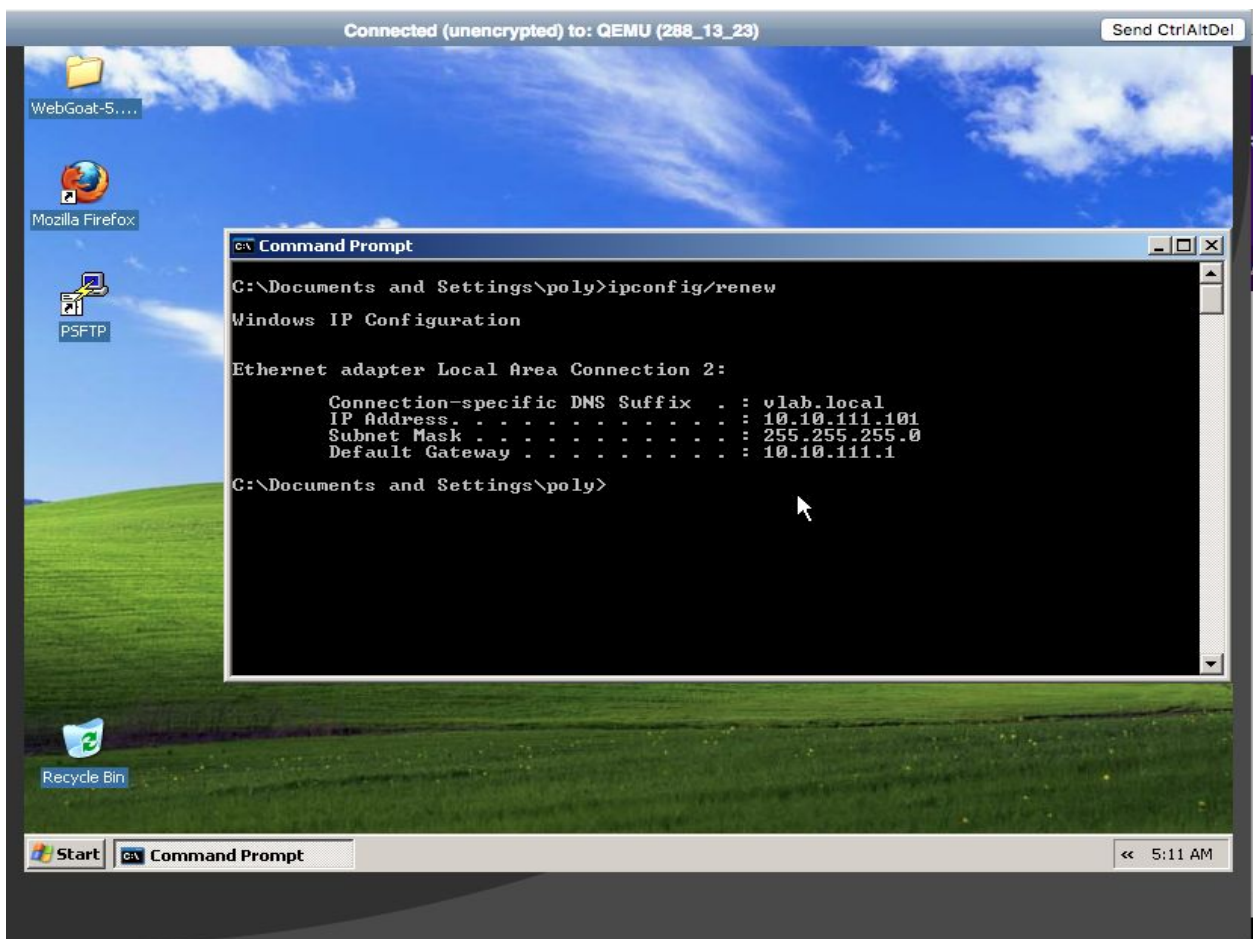
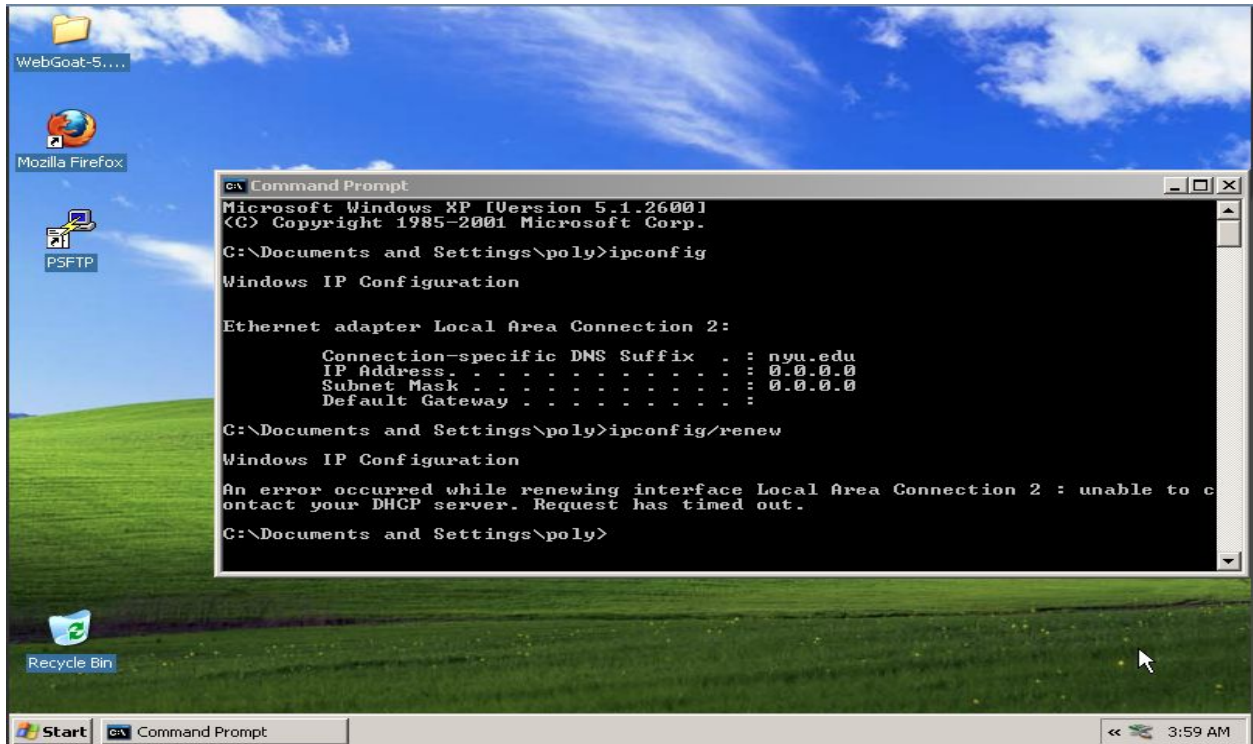
8. Wireshark is then started to capture the packets like ACK, NAK, Requests, etc.



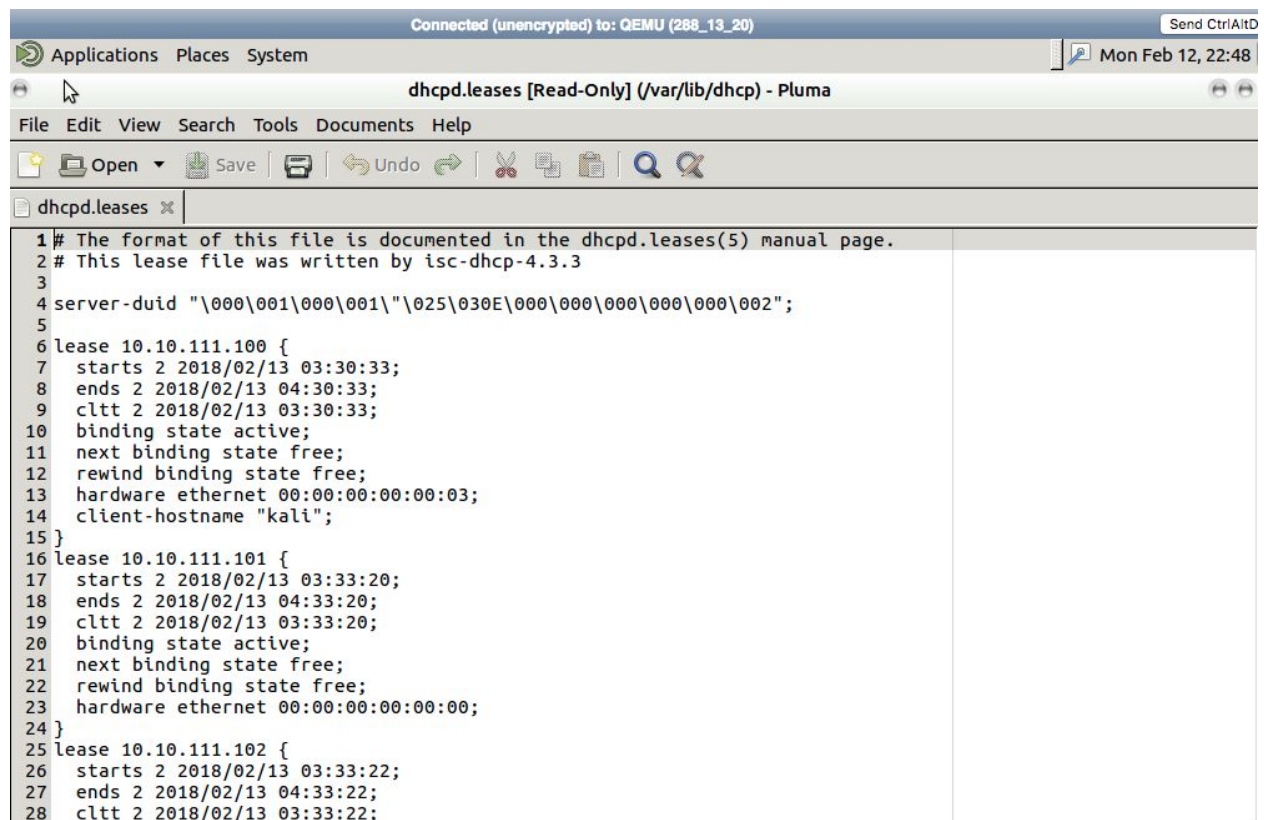
9. After all the IP addresses are assigned then we find that all the IPs' are starved.



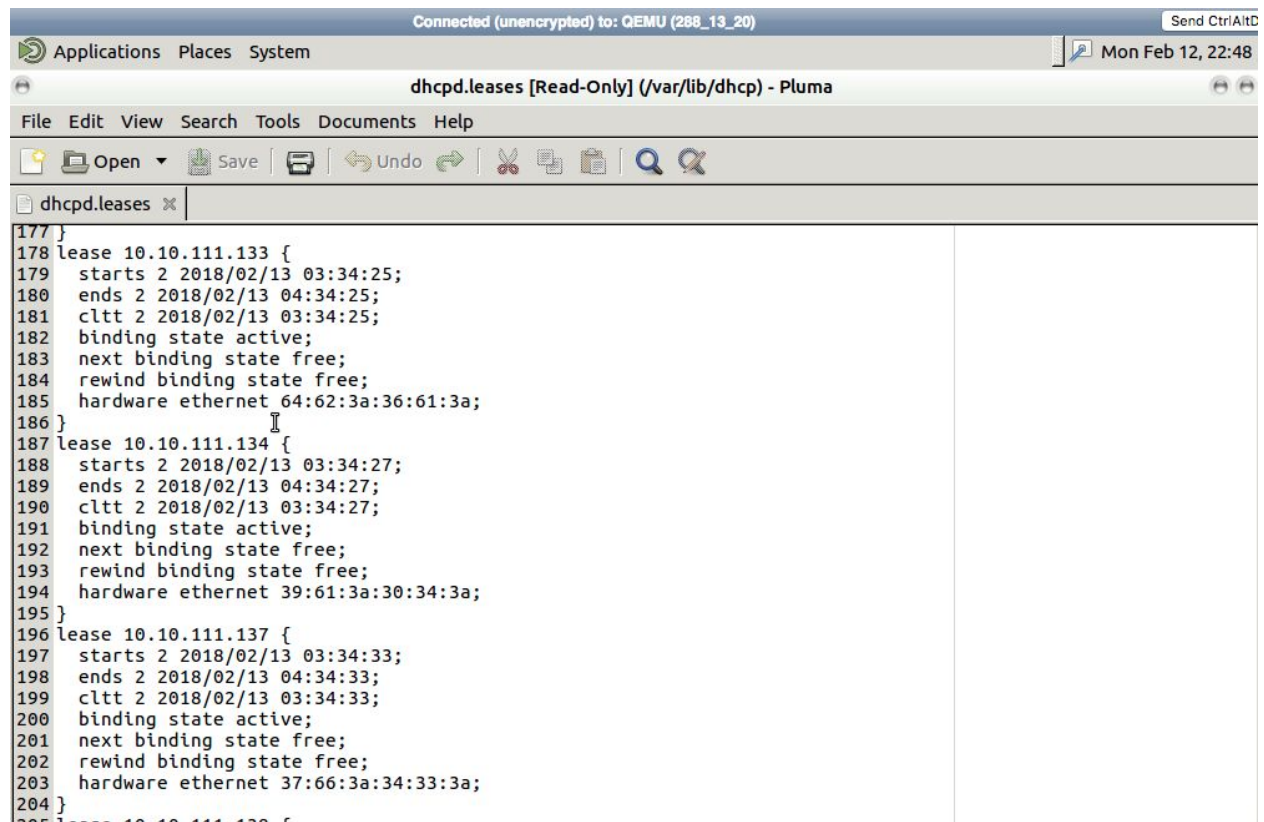
10. We then have to start the Windows machine and check the IP address of the Windows VM by the command **ipconfig**. If the Windows machine has an IP assigned to it then we need to execute **ipconfig/release** command in order to release the assigned IP address. Further, the command **ipconfig/renew** is executed to get the new IP address.



11. Dhcpd.leases file after the starvation attack



```
1 # The format of this file is documented in the dhcpd.leases(5) manual page.
2 # This lease file was written by isc-dhcp-4.3.3
3
4 server-uid "\000\001\000\001\"025\030E\000\000\000\000\000\002";
5
6 lease 10.10.111.100 {
7   starts 2 2018/02/13 03:30:33;
8   ends 2 2018/02/13 04:30:33;
9   cltt 2 2018/02/13 03:30:33;
10  binding state active;
11  next binding state free;
12  rewind binding state free;
13  hardware ethernet 00:00:00:00:00:03;
14  client-hostname "kali";
15 }
16 lease 10.10.111.101 {
17   starts 2 2018/02/13 03:33:20;
18   ends 2 2018/02/13 04:33:20;
19   cltt 2 2018/02/13 03:33:20;
20   binding state active;
21   next binding state free;
22   rewind binding state free;
23   hardware ethernet 00:00:00:00:00:00;
24 }
25 lease 10.10.111.102 {
26   starts 2 2018/02/13 03:33:22;
27   ends 2 2018/02/13 04:33:22;
28   cltt 2 2018/02/13 03:33:22;
```



```
177 }
178 lease 10.10.111.133 {
179   starts 2 2018/02/13 03:34:25;
180   ends 2 2018/02/13 04:34:25;
181   cltt 2 2018/02/13 03:34:25;
182   binding state active;
183   next binding state free;
184   rewind binding state free;
185   hardware ethernet 64:62:3a:36:61:3a;
186 }
187 lease 10.10.111.134 {
188   starts 2 2018/02/13 03:34:27;
189   ends 2 2018/02/13 04:34:27;
190   cltt 2 2018/02/13 03:34:27;
191   binding state active;
192   next binding state free;
193   rewind binding state free;
194   hardware ethernet 39:61:3a:30:34:3a;
195 }
196 lease 10.10.111.137 {
197   starts 2 2018/02/13 03:34:33;
198   ends 2 2018/02/13 04:34:33;
199   cltt 2 2018/02/13 03:34:33;
200   binding state active;
201   next binding state free;
202   rewind binding state free;
203   hardware ethernet 37:66:3a:34:33:3a;
204 }
205
```

Python Code Details :

1. To add time delay and threading operations, we import **Sleep** and **Thread packages**.
2. Empty mac and IP list is created.
3. Object of **Thread** class with the name 'thread' is created which has argument as 'target = sniffing' to sniff UDP traffic. Function proc_start() will start the attack by executing the thread.start() function.
4. proc_start() calls the starvation() function where spoof_mac addresses send requests to the DHCP server. The function is called till the time all of the IPs get registered with the DHCP server.
5. starvation() function starts the actual starvation process. Hundred spoof_mac addresses are generated using the standard function RandMAC(), and are then appended to the mac list. Loop is executed for the range(0,101).
6. self.mac is used for any duplicate mac addresses.
7. starvation() checks for IP addresses which are assigned by the DHCP server using the for loop and appends those addresses to the IP list defined in the third step.
8. sendp(pkt) is used to send request for IP and pkt stores the client and server information in various layers.
9. If IP is already assigned to the spoof_mac addresses, the function req_Maintain() will return NAK or else will return ACK.
10. Object of class DHCP_Store is declared and function proc_start() is called to start the starvation process.