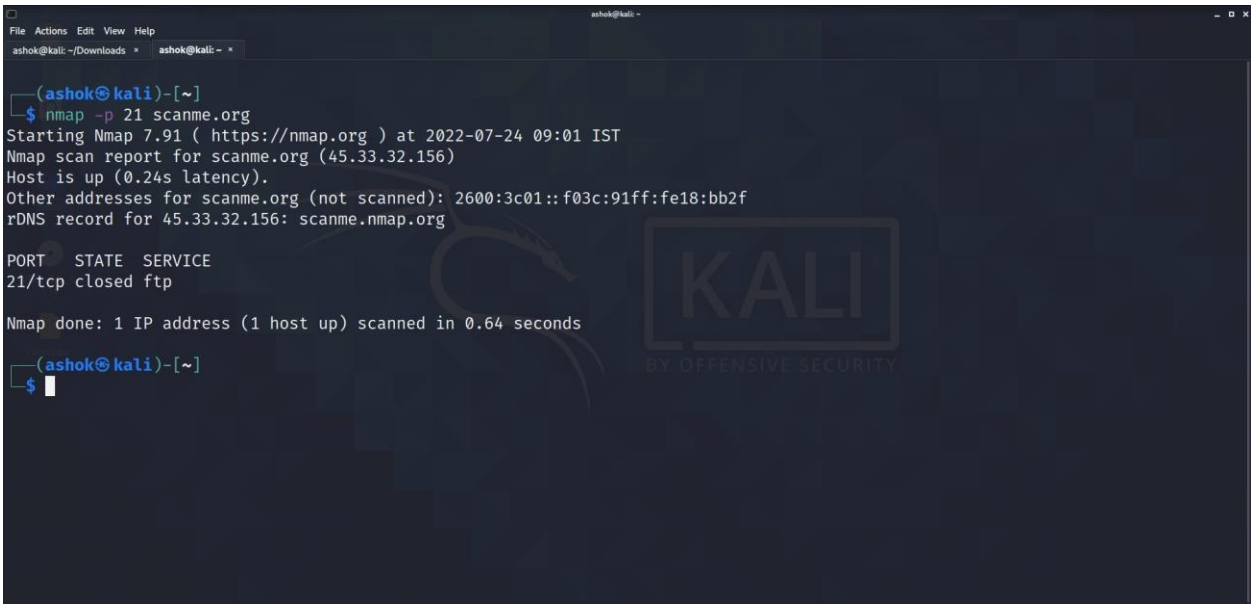
Experiment 1: Perform an Experiment for port scanning with NMAP.

Aim: To remotely testing numerous ports to determine what state they are in

Nmap is a network scanner utility used for port mapping, host discovery and vulnerability scanning. Most of its functions are based on using IP packet analysis to detect and identify remote hosts, operating systems and services.

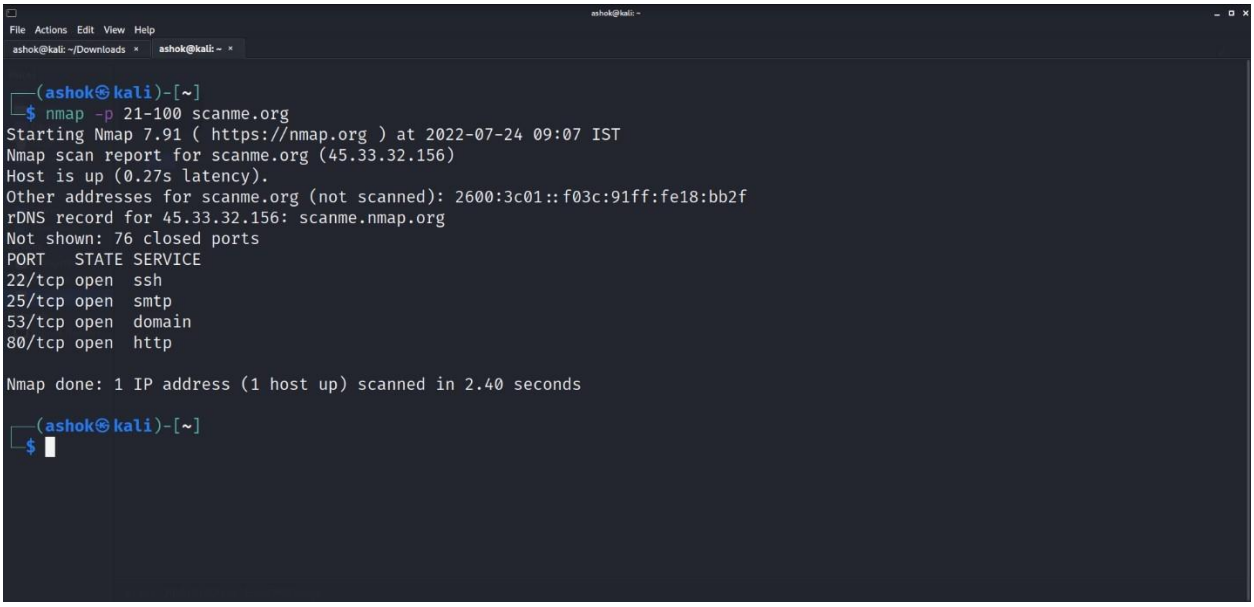
Step 1: Port scan for port 21

Command: $ nmap -p 21 scanme.org



Step 2: Port scan for port range

Command: $ nmap -p 21-100 scanme.org

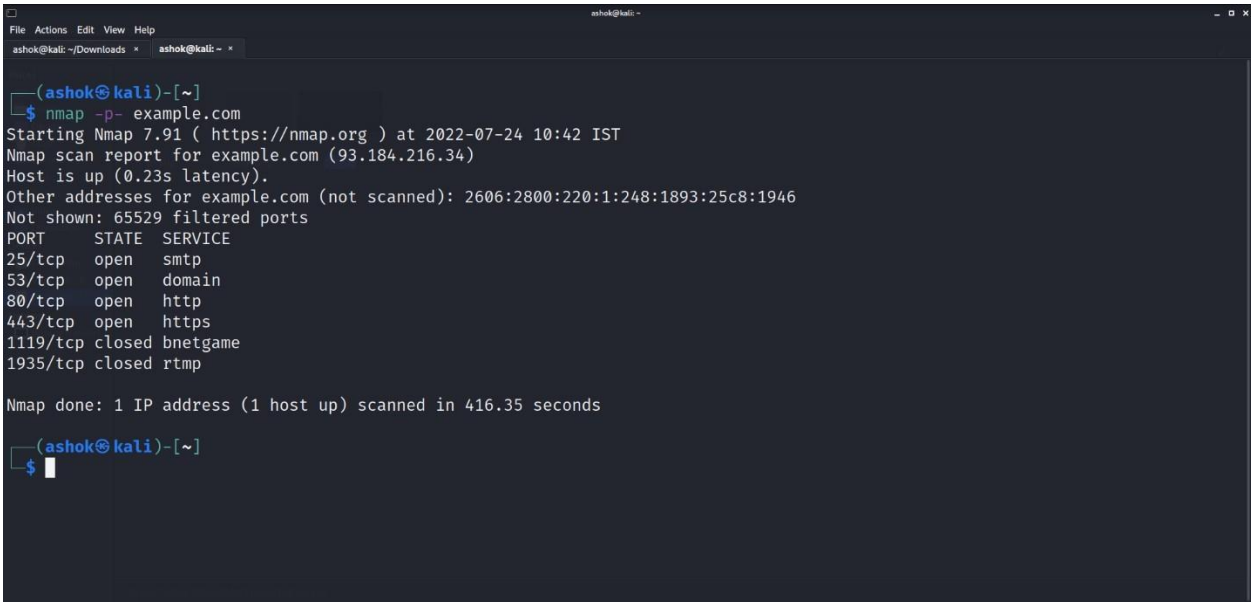


Step 3: Port scan for multiple TCP and UDP ports

Command: $ nmap -p U:53, T:21-25,80 scanme.org

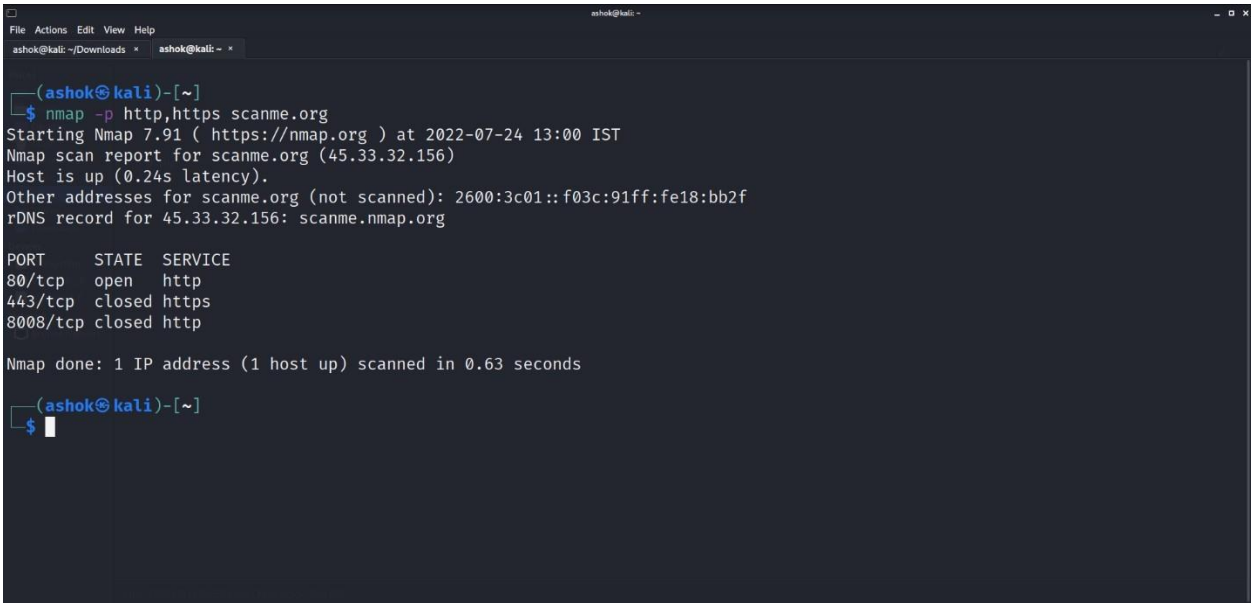
Step 4: Port scan for all ports

Command: $ nmap -p- example.com



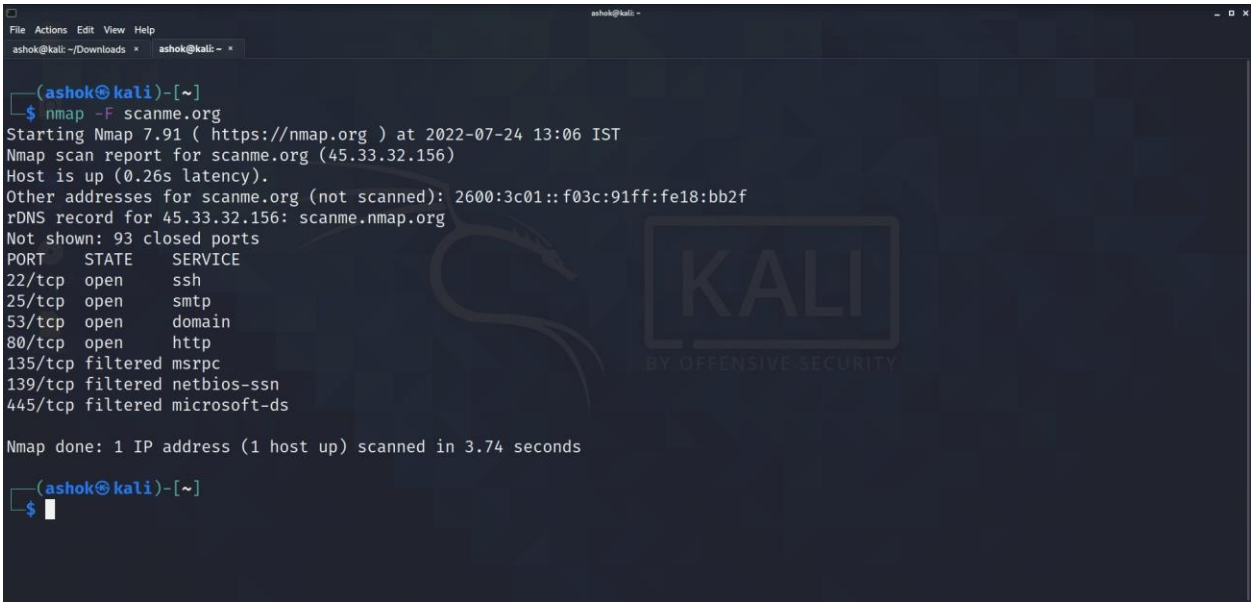
Step 5: Port scan for service name

Command: # nmap -p http, https scanme.org



Step 6: Fast port scan (100)

Command: $ nmap -F scanme.org



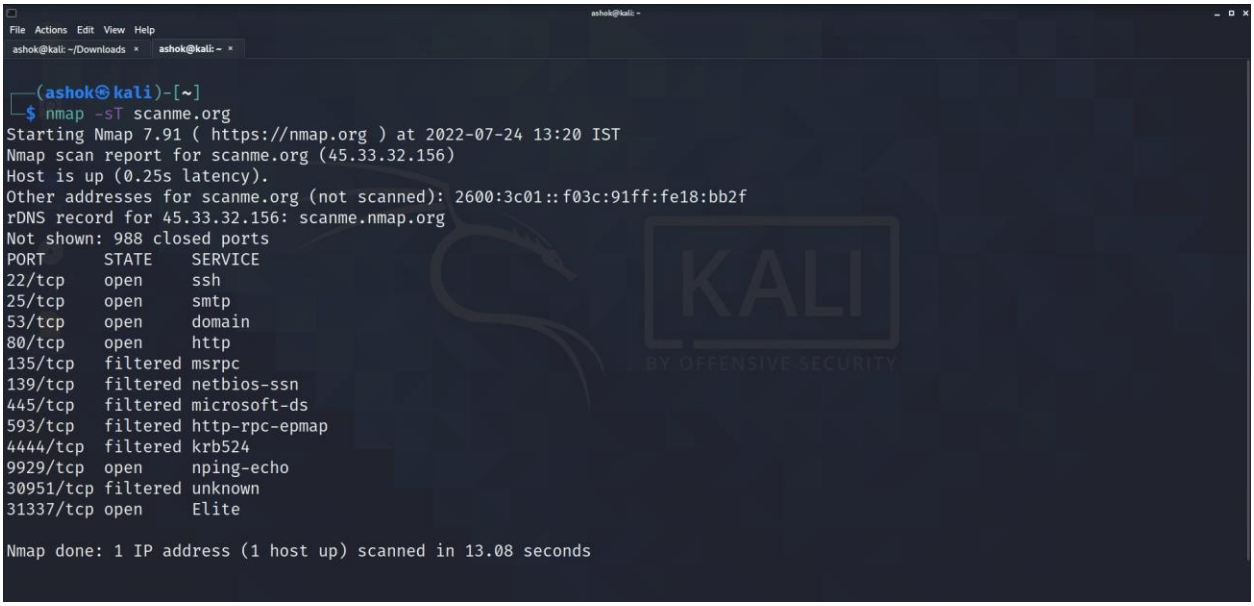
SCAN TECHNIQUES:

Step 1: TCP SYN port scan

Command: $ nmap -sS scanme.org

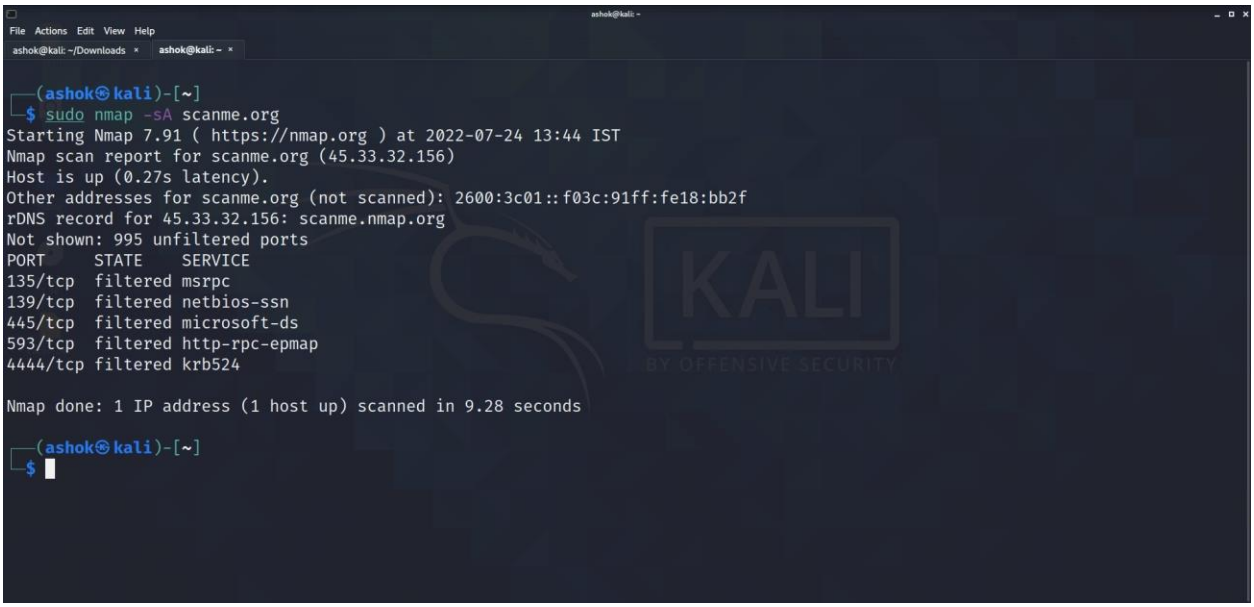
Step 2: TCP Connect port scan (without root privileges)

Command: $ nmap -sT scanme.org



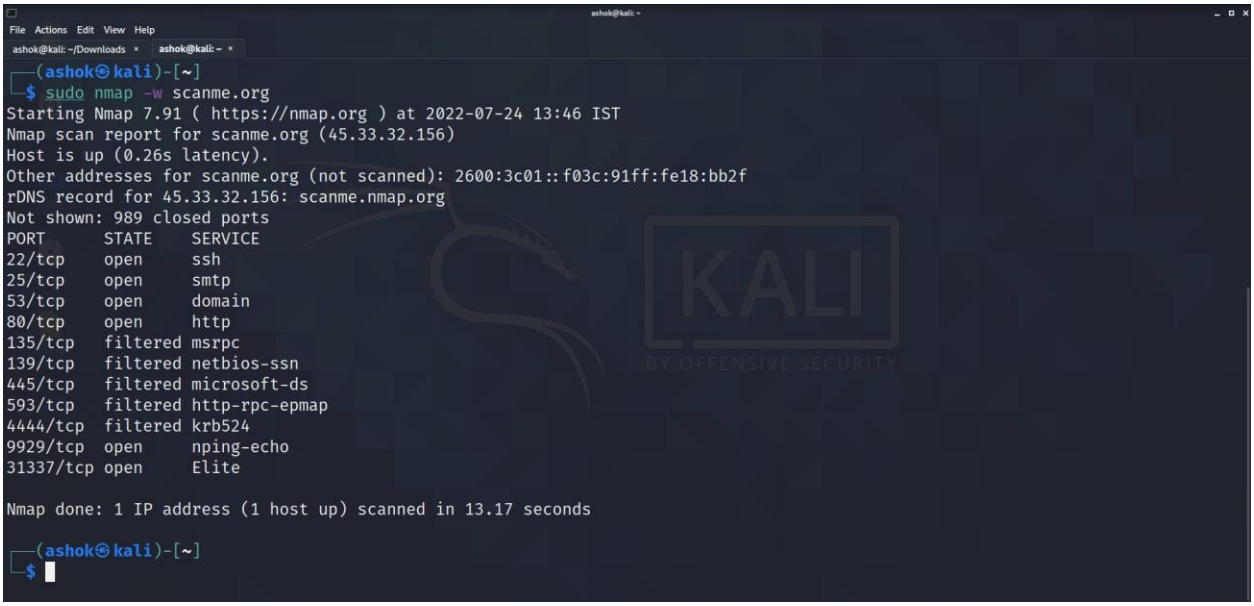
Step 3: TCP ACK port scan

$ nmap -sA scanme.org



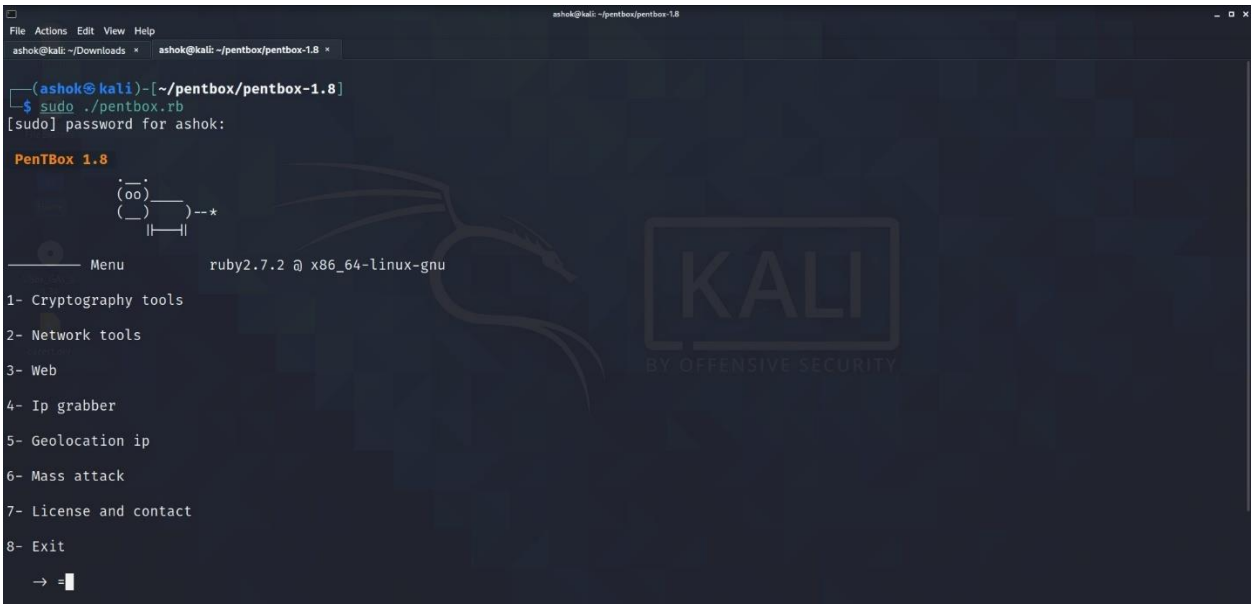
Step 4: TCP window port scan

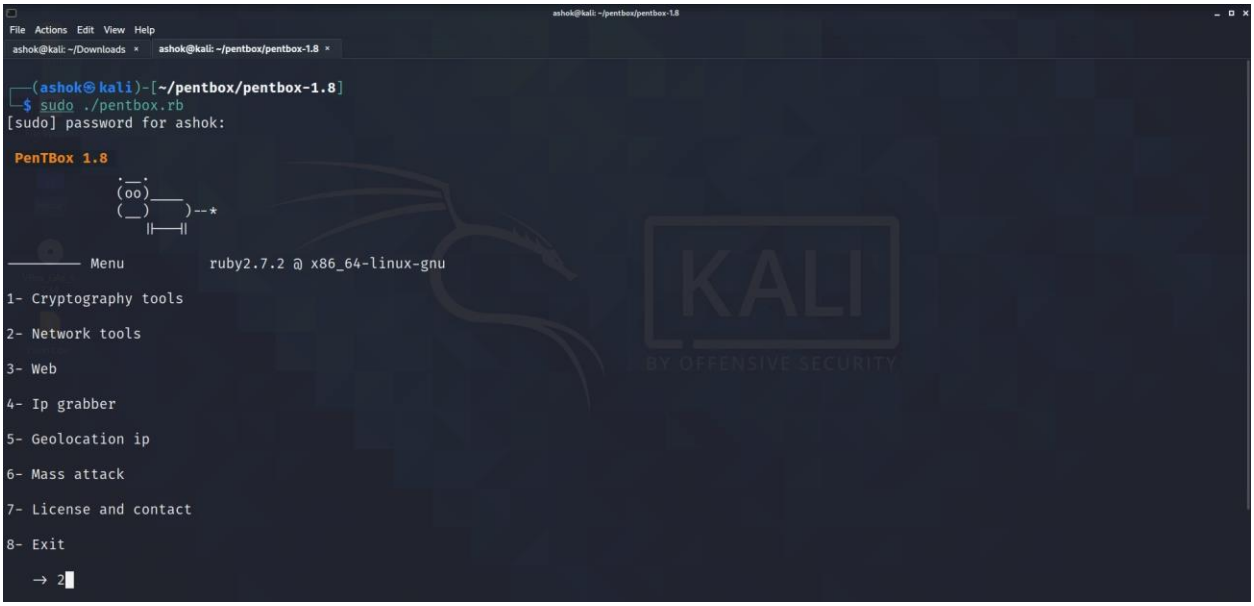
Command: $ namp -w scanme.org

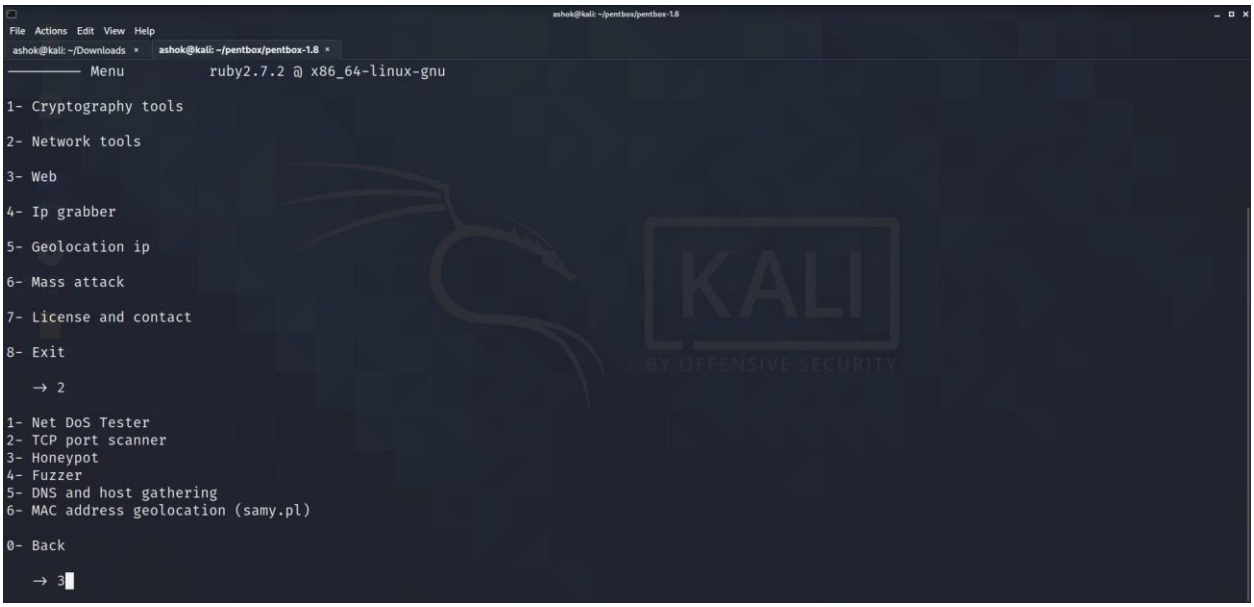

ashok@kali: ~
File Actions Edit View Help
ashok@kali: ~/Downloads
ashok@kali: ~
(ashok㉿kali)-[~]
$ sudo nmap -w scanme.org
Starting Nmap 7.91 (https://nmap.org) at 2022-07-24 13:46 IST
Nmap scan report for scanme.org (45.33.32.156)
Host is up (0.26s latency).
Other addresses for scanme.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
rDNS record for 45.33.32.156: scanme.nmap.org
Not shown: 989 closed ports
PORT STATE SERVICE
22/tcp open ssh
25/tcp open smtp
53/tcp open domain
80/tcp open http
135/tcp filtered msrpc
139/tcp filtered netbios-ssn
445/tcp filtered microsoft-ds
593/tcp filtered http-rpc-epmap
4444/tcp filtered krb524
9929/tcp open nping-echo
31337/tcp open Elite
Nmap done: 1 IP address (1 host up) scanned in 13.17 seconds
(ashok㉿kali)-[~]
$

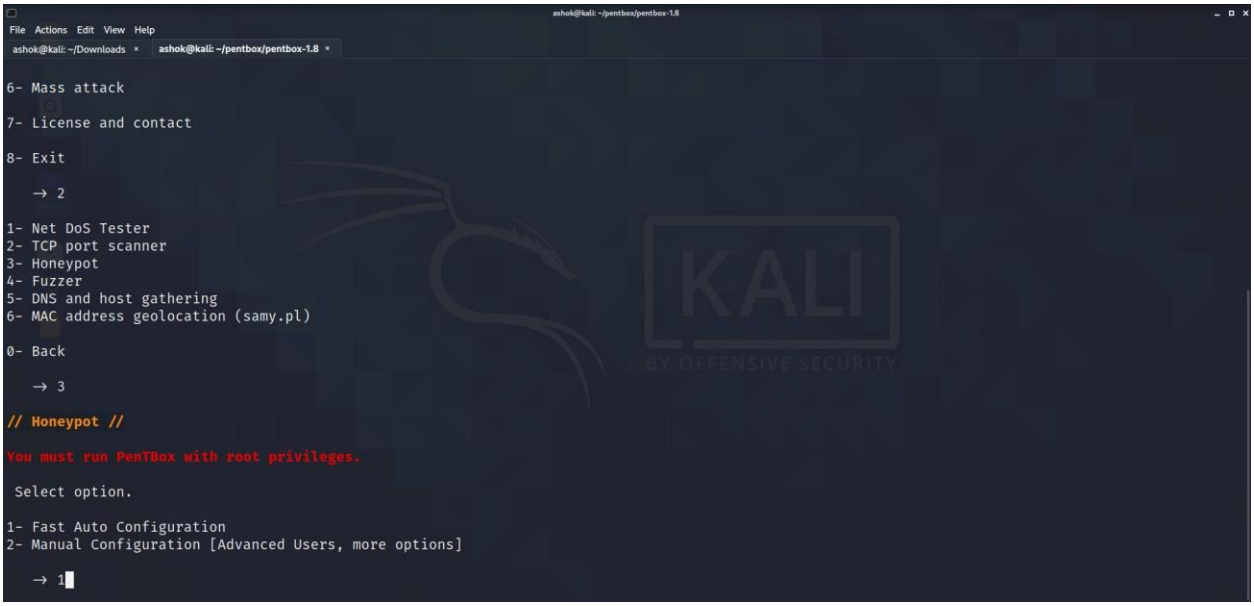
Experiment 2: Set up a honeypot and monitor the honeypot on the network

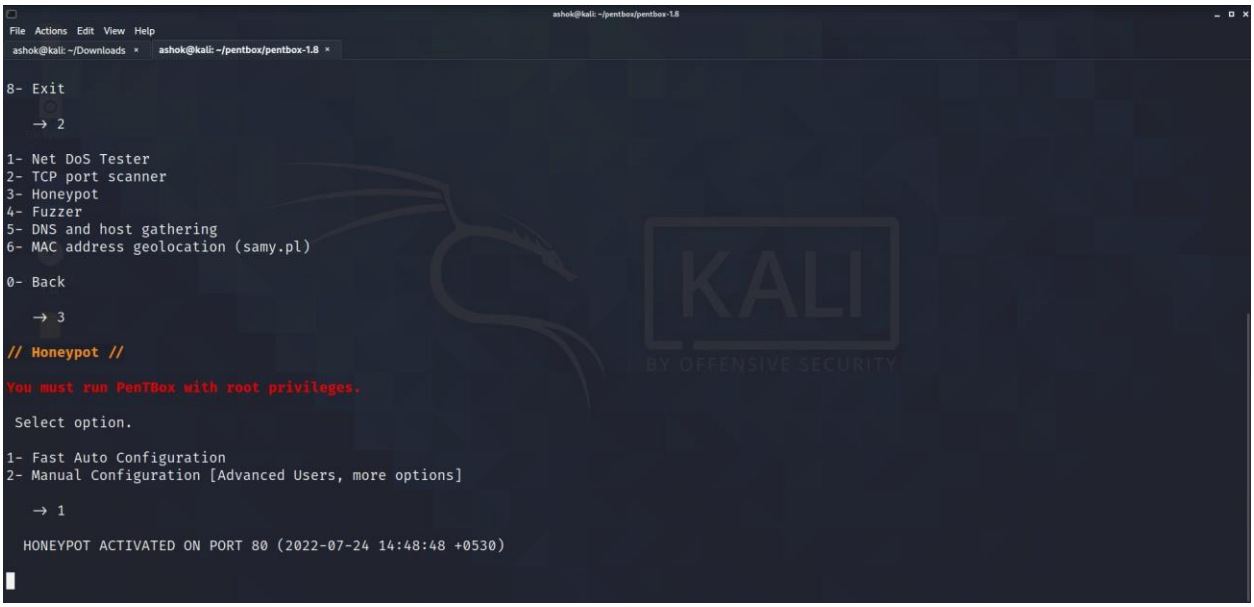
Aim: To lure cyber attackers and detect, deflect and study hacking attempts to gain unauthorized access to information systems.

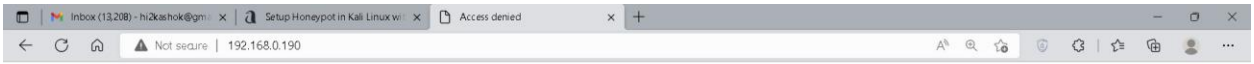



ashok@kali: ~/pentbox/pentbox-1.8
File Actions Edit View Help
ashok@kali: ~/Downloads
ashok@kali: ~/pentbox/pentbox-1.8
┌──(ashok㉿kali)-[~/pentbox/pentbox-1.8]
└─$ sudo ./pentbox.rb
[sudo] password for ashok:
PenTBox 1.8
(oo)
(__)
Menu
ruby2.7.2 @ x86_64-linux-gnu
1- Cryptography tools
2- Network tools
3- Web
4- Ip grabber
5- Geolocation ip
6- Mass attack
7- License and contact
8- Exit
→ 2


ashok@kali: ~/pentbox/pentbox-1.8
File Actions Edit View Help
ashok@kali: ~/Downloads
ashok@kali: ~/pentbox/pentbox-1.8
Menu
ruby2.7.2 @ x86_64-linux-gnu
1- Cryptography tools
2- Network tools
3- Web
4- Ip grabber
5- Geolocation ip
6- Mass attack
7- License and contact
8- Exit
→ 2
1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)
0- Back
→ 3


ashok@kali: ~/pentbox/pentbox-1.8
File Actions Edit View Help
ashok@kali: ~/Downloads
ashok@kali: ~/pentbox/pentbox-1.8
6- Mass attack
7- License and contact
8- Exit
→ 2
1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)
0- Back
→ 3
// Honeypot //
You must run PenTBox with root privileges.
Select option.
1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]
→ 1

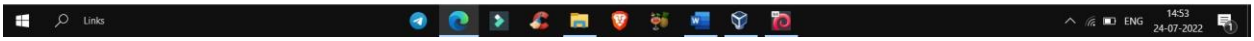

ashok@kali: ~/pentbox/pentbox-1.8
File Actions Edit View Help
ashok@kali: ~/Downloads
ashok@kali: ~/pentbox/pentbox-1.8
8- Exit
→ 2
1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)
0- Back
→ 3
// Honeypot //
You must run PenTBox with root privileges.
Select option.
1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]
→ 1
HONEYPOT ACTIVATED ON PORT 80 (2022-07-24 14:48:48 +0530)

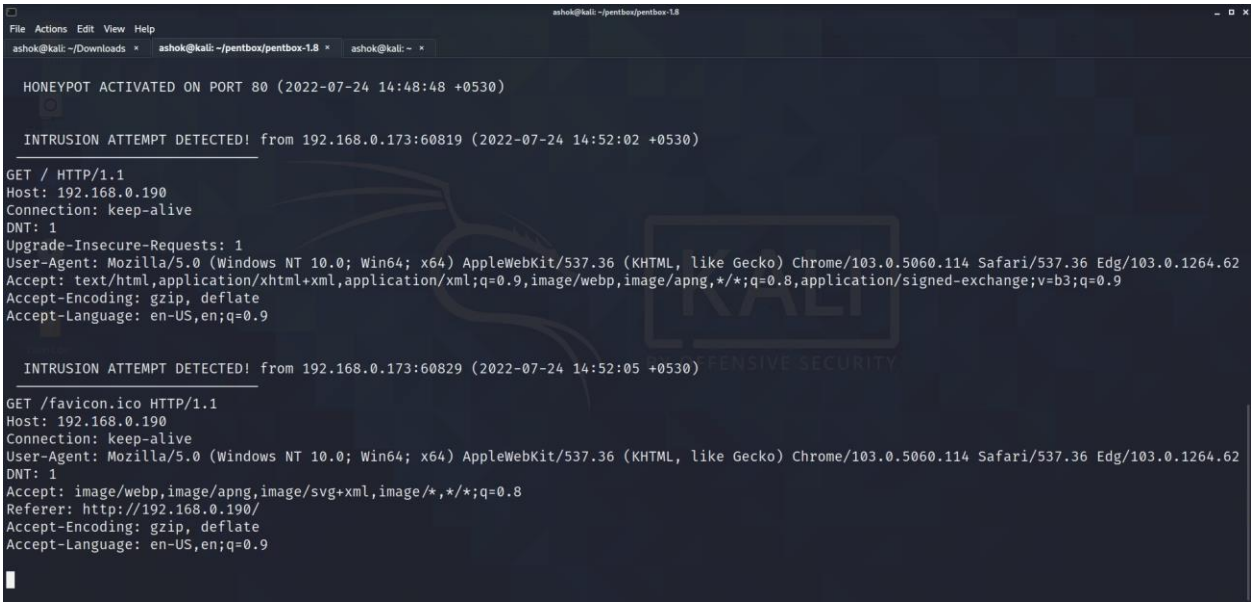


Access denied

IP Address login failed

2022-07-24 14:48:48 +0530



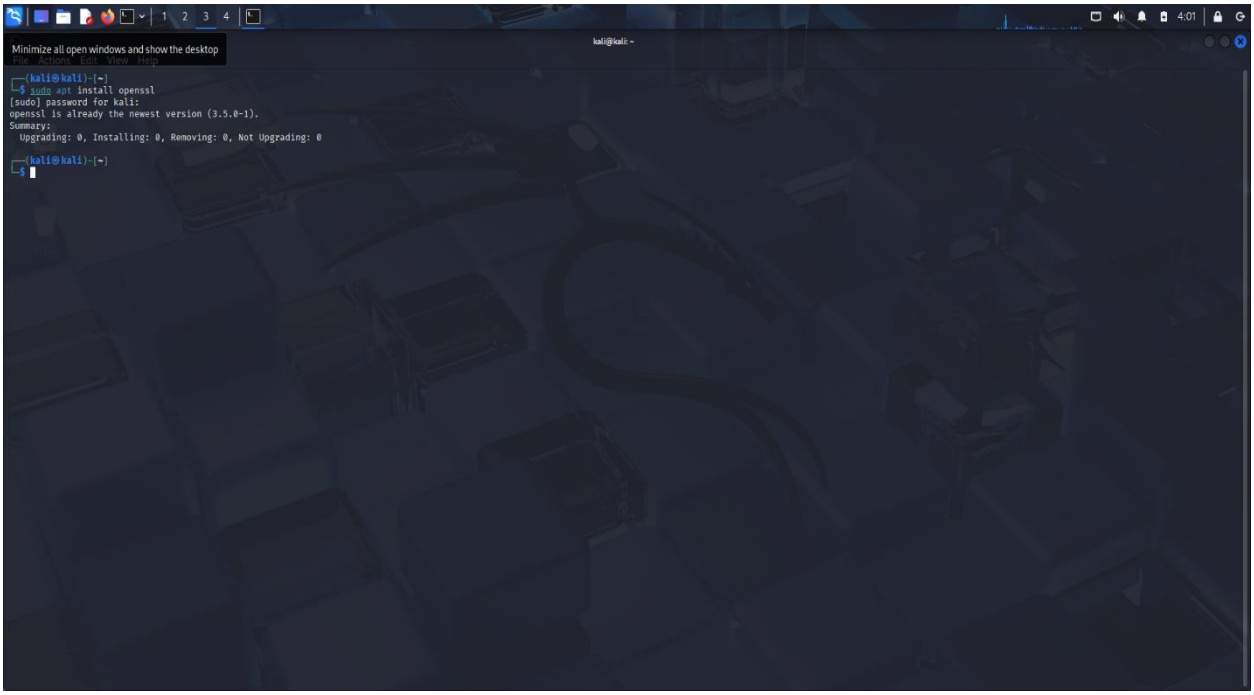


Experiment 4: Generate minimum 10 passwords of length 12 characters using **openSSL** Command.

Aim: To generate strong passwords of different characters of length 12.

Step 1: Install openssl

Command: #sudo apt install openssl



Step 1:

```
$ openssl rand -base64 16
```

Decode:$ echo "B3ch3m3e35LcCiRQiqI=" | base64 -d | wc -c 14

Experiment 5: Perform practical approach to implement Foot Printing-Gathering target information using Dmitry-Dmagic / UAtester.

Aim: To gather target information

Step 1: Install Dmitry

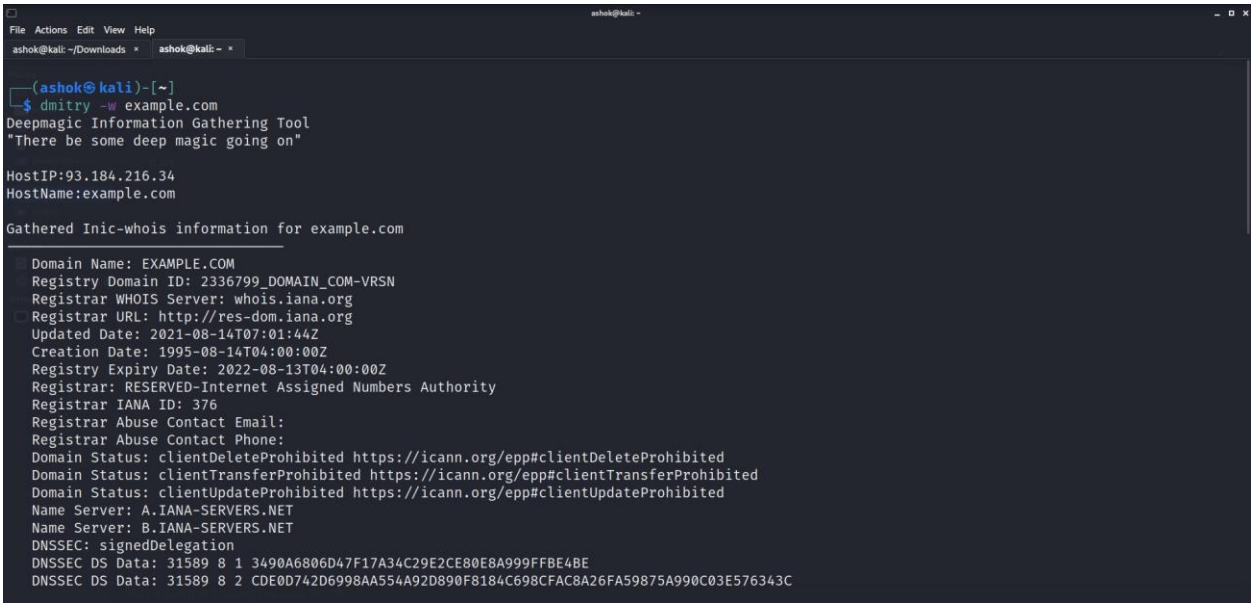
Command: #sudo apt install dmitry

dmitry Usage Example

Run a domain whois lookup (w), an IP whois lookup (i), retrieve Netcraft info (n), search for subdomains (s), search for email addresses (e), do a TCP port scan (p), and save the output to example.txt (o) for the domain example.com:

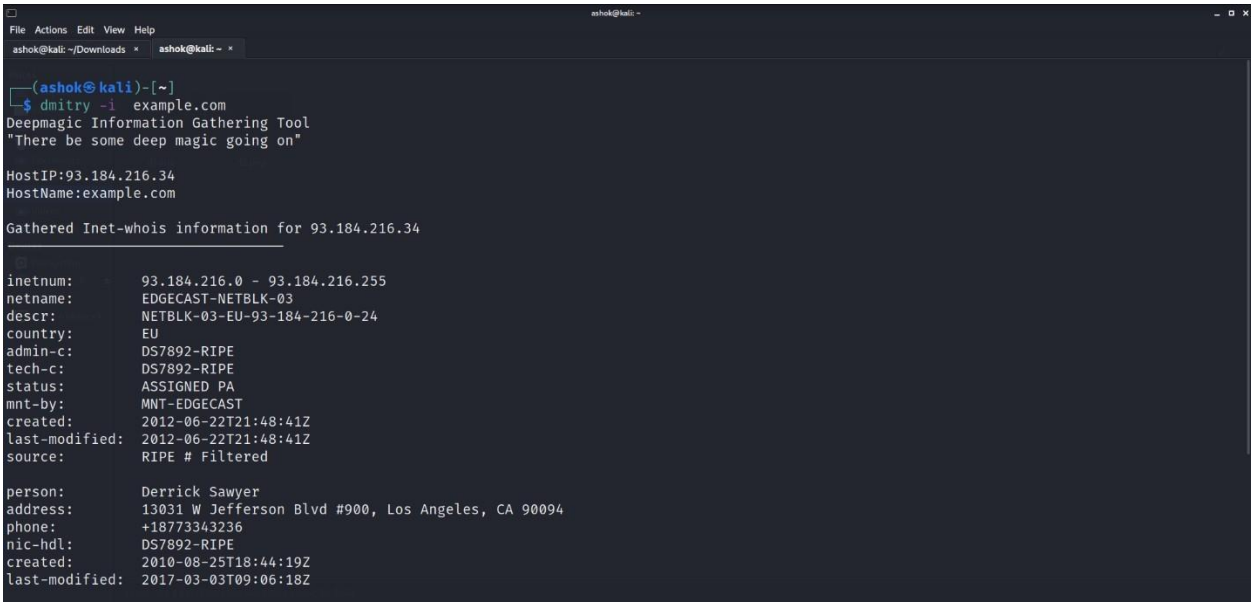
Step 2: Run the tool and type the following command to gather WHOIS information.

Command: # ./dmirty -w example.com



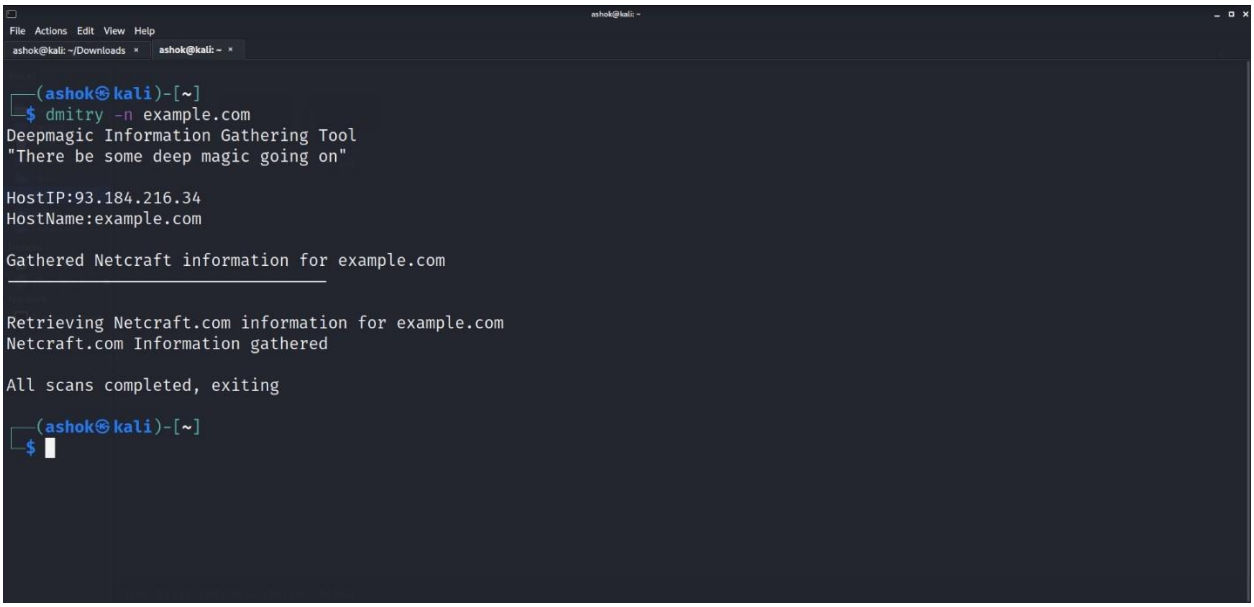
Step 3: Run the tool and type the following command to gather Inet-WHOIS information.

Command: # ./dmirty -i example.com



Step 4: Run the tool and type the following command to gather netcraft information.

Command: # ./dmirty -n example.com



Step 5: Run the tool and type the following command to gather email information.

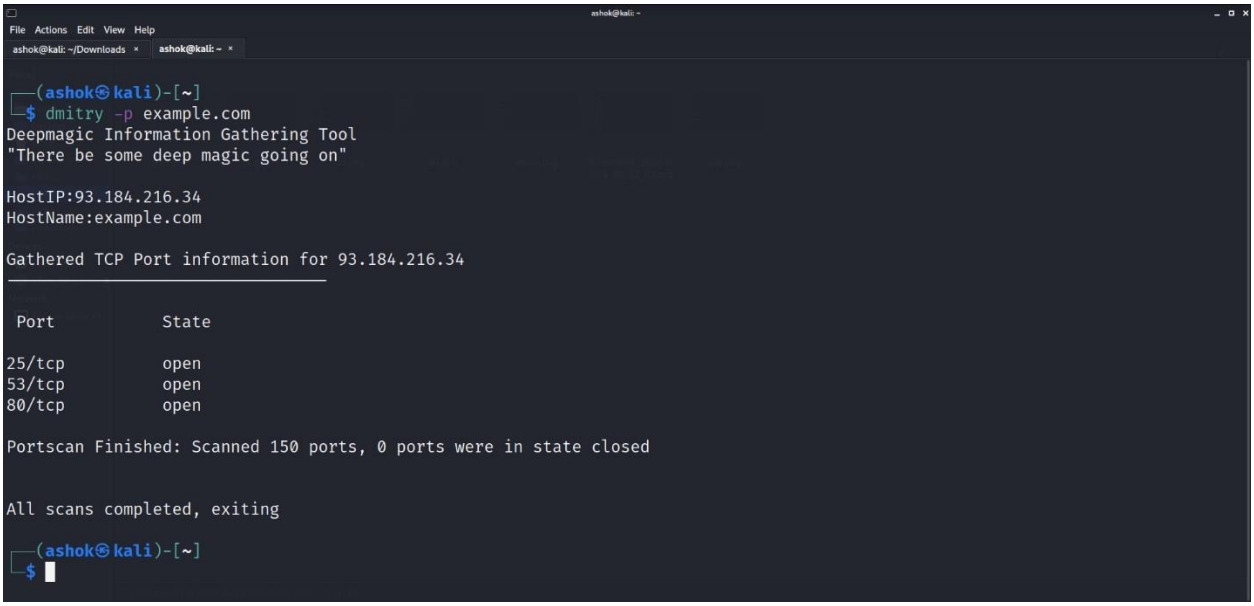
Command: # ./dmirty -e example.com

Step 6: Run the tool and type the following command to gather subdomain information.

Command: # ./dmirty -s example.com

Step 7: Run the tool and type the following command to gather port information.

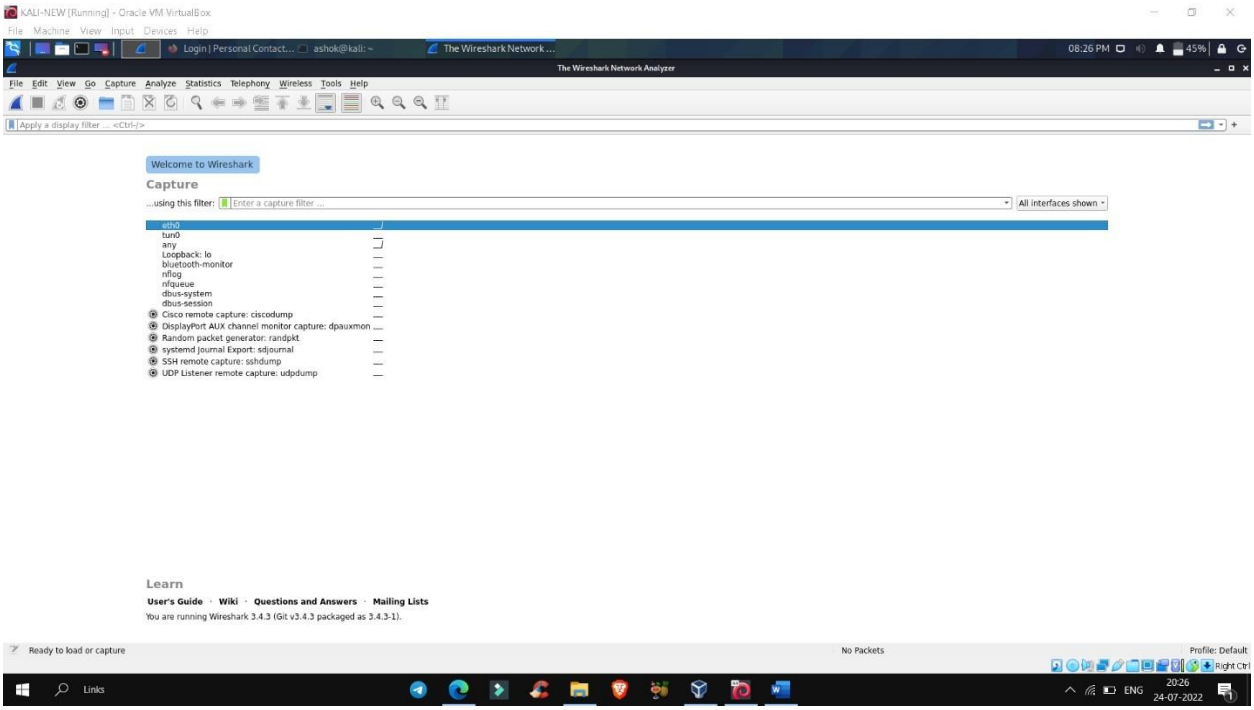
Command: # ./dmirty -p example.com


ashok@kali: ~
File Actions Edit View Help
ashok@kali: ~/Downloads
ashok@kali: ~
┌──(ashok㉿kali)-[~]
└─$ dmitry -p example.com
Deepmagic Information Gathering Tool
"There be some deep magic going on"
HostIP:93.184.216.34
HostName:example.com
Gathered TCP Port information for 93.184.216.34
Port State
25/tcp open
53/tcp open
80/tcp open
Portscan Finished: Scanned 150 ports, 0 ports were in state closed
All scans completed, exiting
┌──(ashok㉿kali)-[~]
└─$

Experiment 6: Working with sniffers for monitoring network communication (Wireshark)

Aim: To monitor network communication.

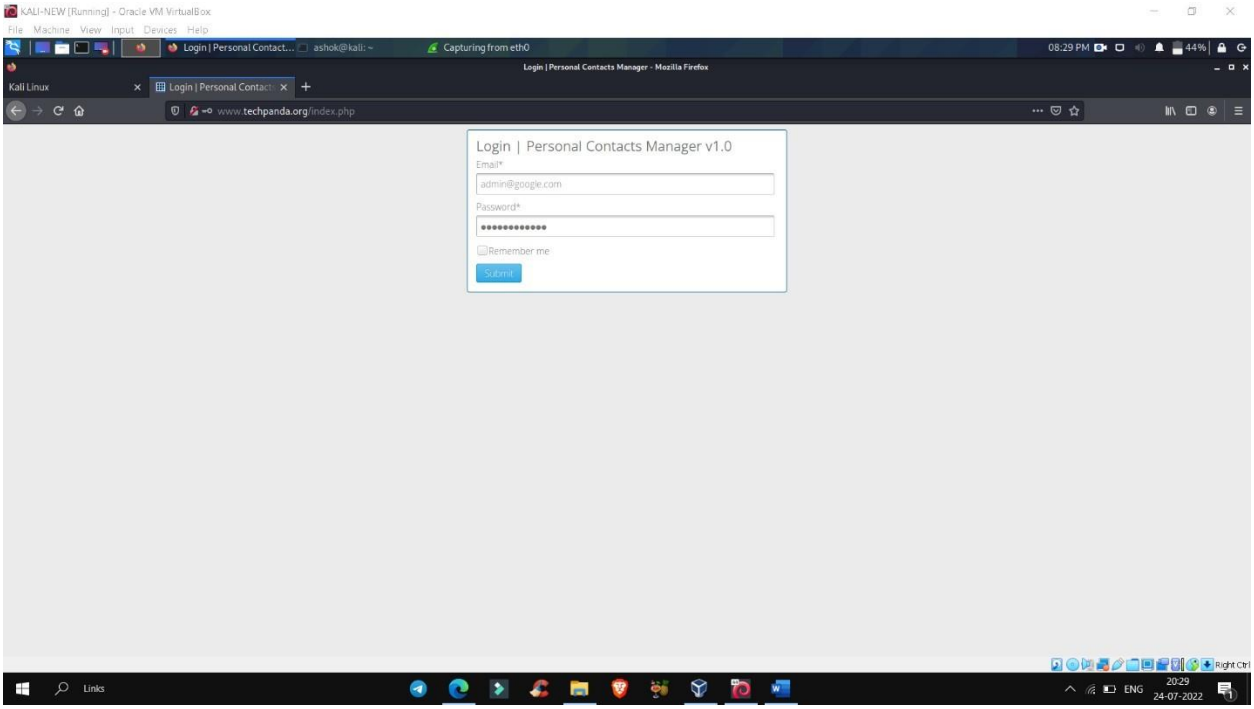
Step 1: Open Wireshark, select Interface and click on start button to capture packets.



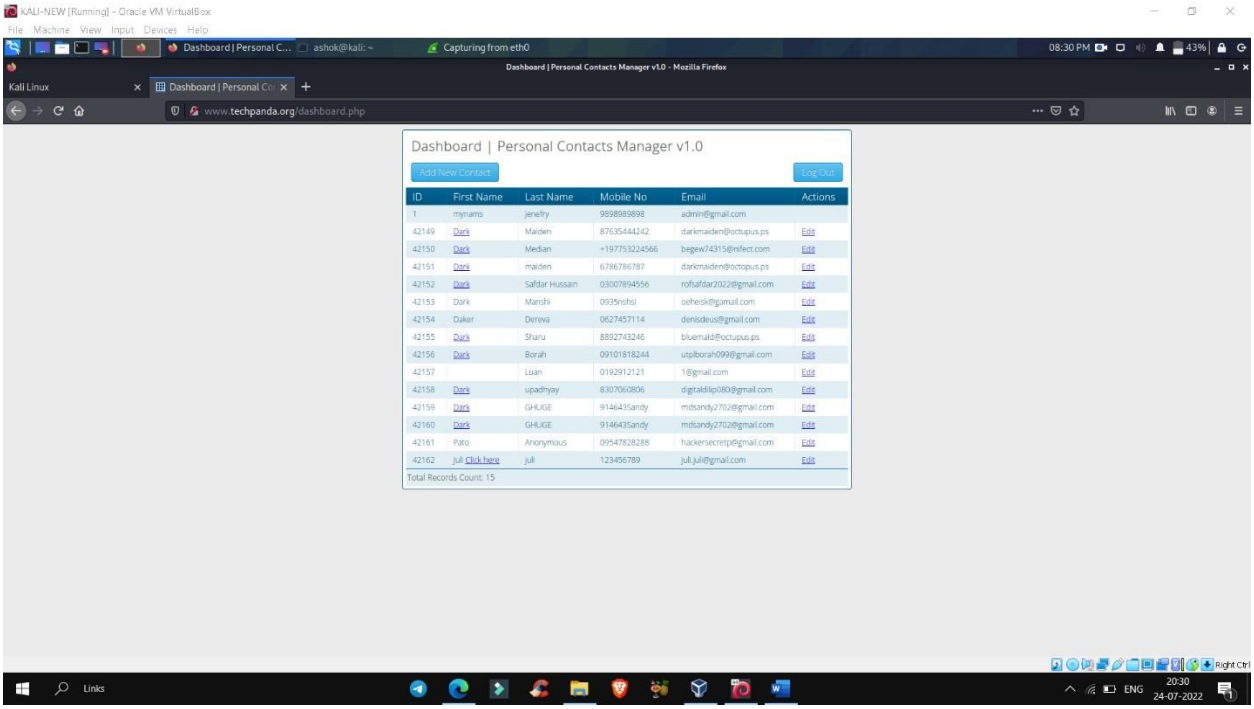
Step 2: Open http://www.techpanda.org in any web browser and enter credentials.

Email: **admin@google.com**

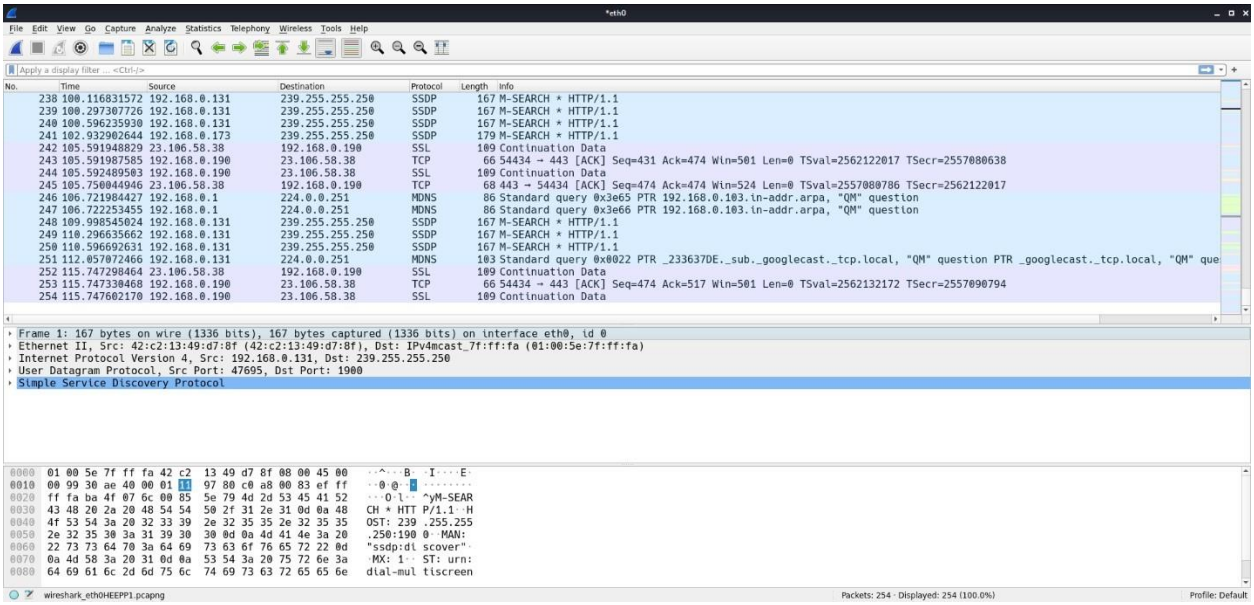
Password: Password2010



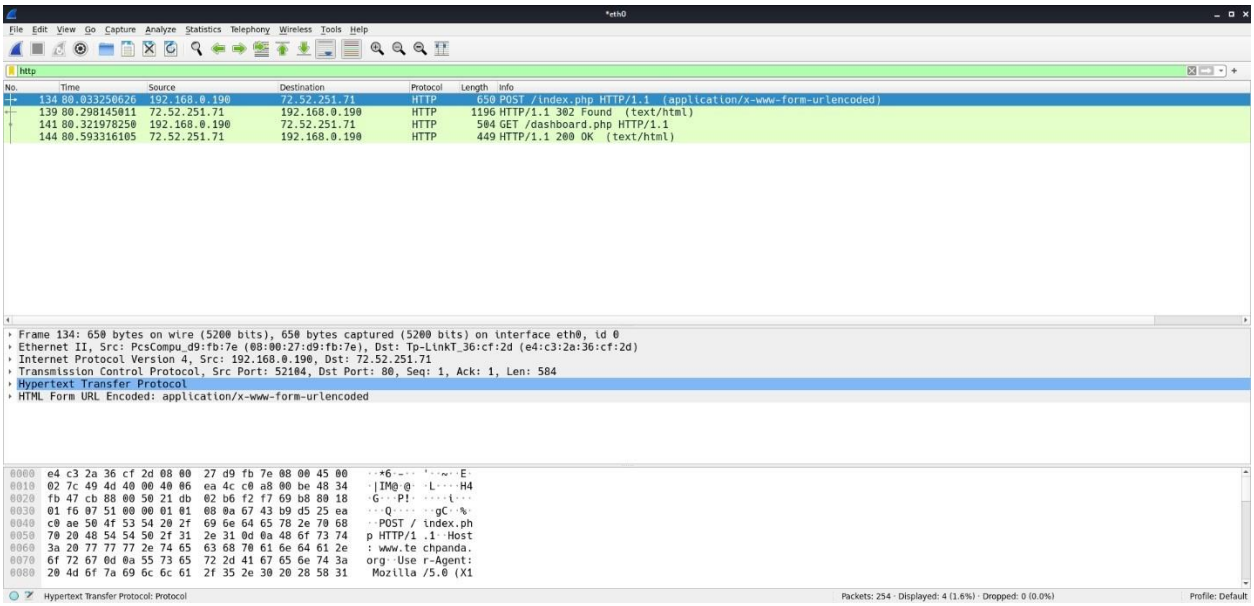
Step 3: Click on submit button and it will display the below screen



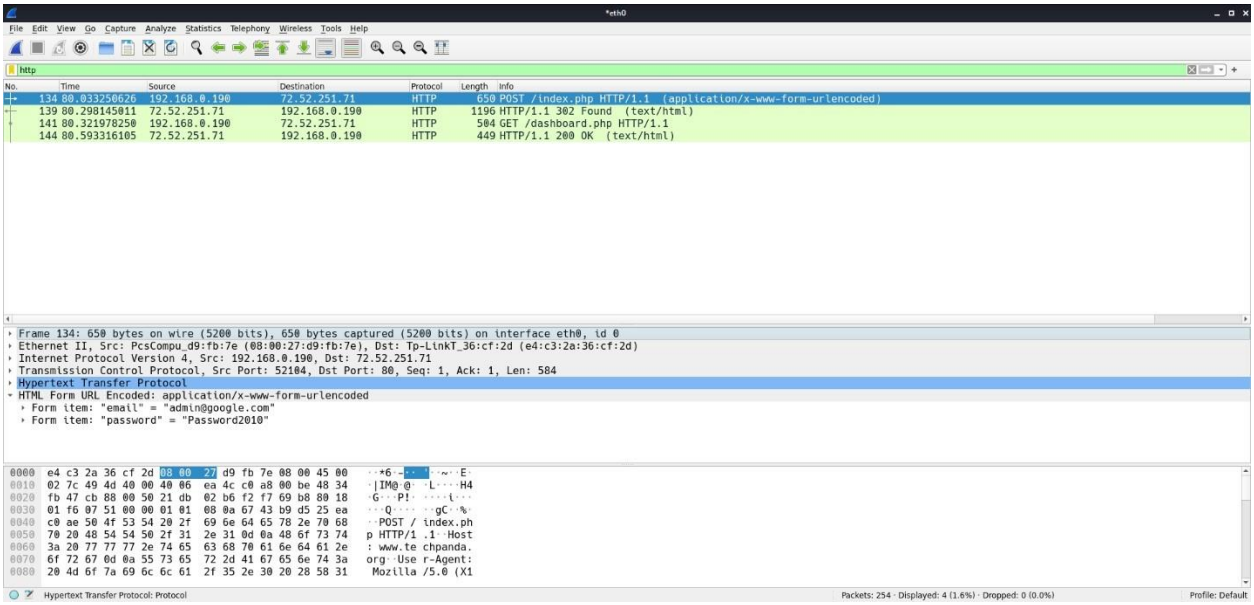
Step 4: Next go to the Wireshark and monitor the traffic



Step 5: Now stop the live capture and filter HTTP protocol. Locate the Info column and look for entries with the HTTP verb POST and click on it.



Step 6: We should be able to view the plaintext values of all the POST variables submitted to the server via HTTP protocol.


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
134 80.033250626 192.168.0.190 72.52.251.71 HTTP 650 POST /index.php HTTP/1.1 (application/x-www-form-urlencoded)
139 80.298145011 72.52.251.71 192.168.0.190 HTTP 1196 HTTP/1.1 302 Found (text/html)
141 80.321978250 192.168.0.190 72.52.251.71 HTTP 504 GET /dashboard.php HTTP/1.1
144 80.593316105 72.52.251.71 192.168.0.190 HTTP 449 HTTP/1.1 200 OK (text/html)
Frame 134: 650 bytes on wire (5200 bits), 650 bytes captured (5200 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_d9:fb:7e (08:00:27:d9:fb:7e), Dst: Tp-LinkT_36:cf:2d (e4:c3:2a:36:cf:2d)
Internet Protocol Version 4, Src: 192.168.0.190, Dst: 72.52.251.71
Transmission Control Protocol, Src Port: 52104, Dst Port: 80, Seq: 1, Ack: 1, Len: 584
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "email" = "admin@google.com"
Form item: "password" = "Password2010"
0000 e4 c3 2a 36 cf 2d 08 00 27 d9 fb 7e 08 00 45 00
0010 02 7c 49 4d 40 00 40 06 ea 4c c0 a8 00 be 48 34
0020 fb 47 cb 88 00 50 21 db 02 b6 f2 f7 69 b8 80 18
0030 01 f6 07 51 00 00 01 01 08 0a 67 43 b9 d5 25 ea
0040 c0 ae 50 4f 53 54 20 2f 69 6e 64 65 78 2e 70 68
0050 70 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74
0060 3a 20 77 77 77 2e 74 65 63 68 70 61 6e 64 61 2e
0070 6f 72 67 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a
0080 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 58 31
Hypertext Transfer Protocol: Protocol
Packets: 254 · Displayed: 4 (1.6%) · Dropped: 0 (0.0%)
Profile: Default

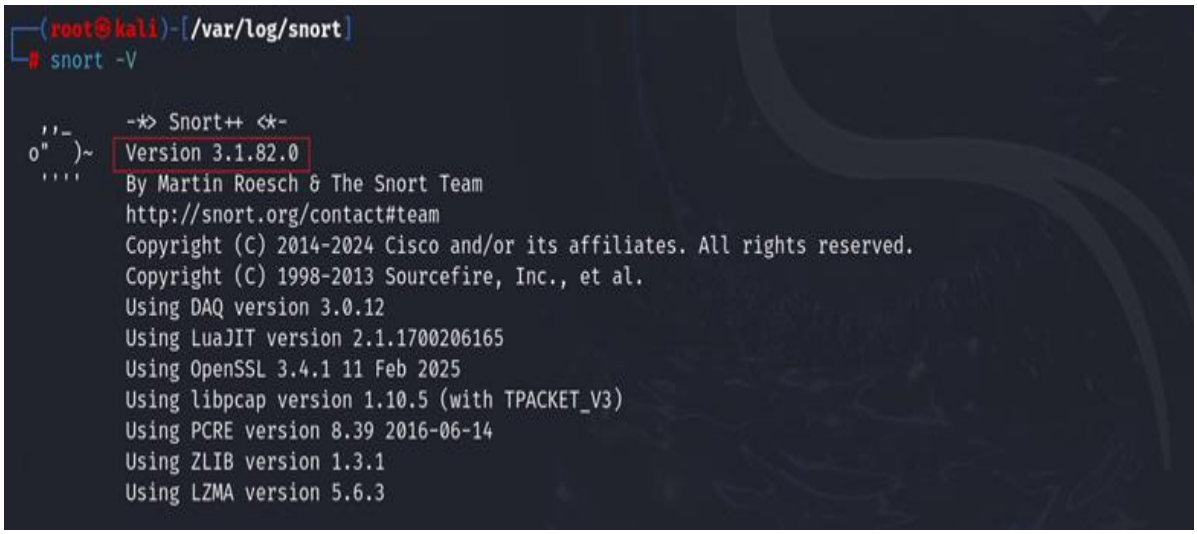
Experiment 7: Using snort, Perform real time traffic analysis and packet logging.

Aim: To perform real time traffic analysis and packet logging.

Installing Snort

Command: sudo apt install snort

To check your Snort version, use the command: snort -V



Choosing the Network Interface

Snort defaults to your primary network interface, but you can explicitly choose one with -i:

To sniff on a specific interface:

Command: sudo snort -i eth0

- **-i eth0:** Use interface eth0 (replace with your active interface name).

```
┌──(root㉿kali)-[/home/kali]
└─# sudo snort -i eth0

o")~   Snort++ 3.1.82.0

pcap DAQ configured to passive.
Commencing packet processing
++ [0] eth0

Packet Statistics

daq
                received: 204
                analyzed: 200
             outstanding: 4
          outstanding_max: 4
                   allow: 200
                rx_bytes: 46632

codec
                   total: 200          (100.000%)
                 discards: 7           (  3.500%)
                     eth: 200          (100.000%)
                   icmp4: 6            (  3.000%)
                   icmp6: 1            (  0.500%)
                    ipv4: 199          ( 99.500%)
                    ipv6: 1            (  0.500%)
                     tcp: 175          ( 87.500%)
                     udp: 11           (  5.500%)
```

To find your active interfaces:

Command: ip a

```
┌──(root㉿kali)-[/home/kali]
└─# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:92:28:d0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.194.132/24 brd 192.168.194.255 scope global dynamic noprefixroute eth0
       valid_lft 1706sec preferred_lft 1706sec
    inet6 fe80::f500:ce82:7d:9015/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:85:18:1a:e0 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

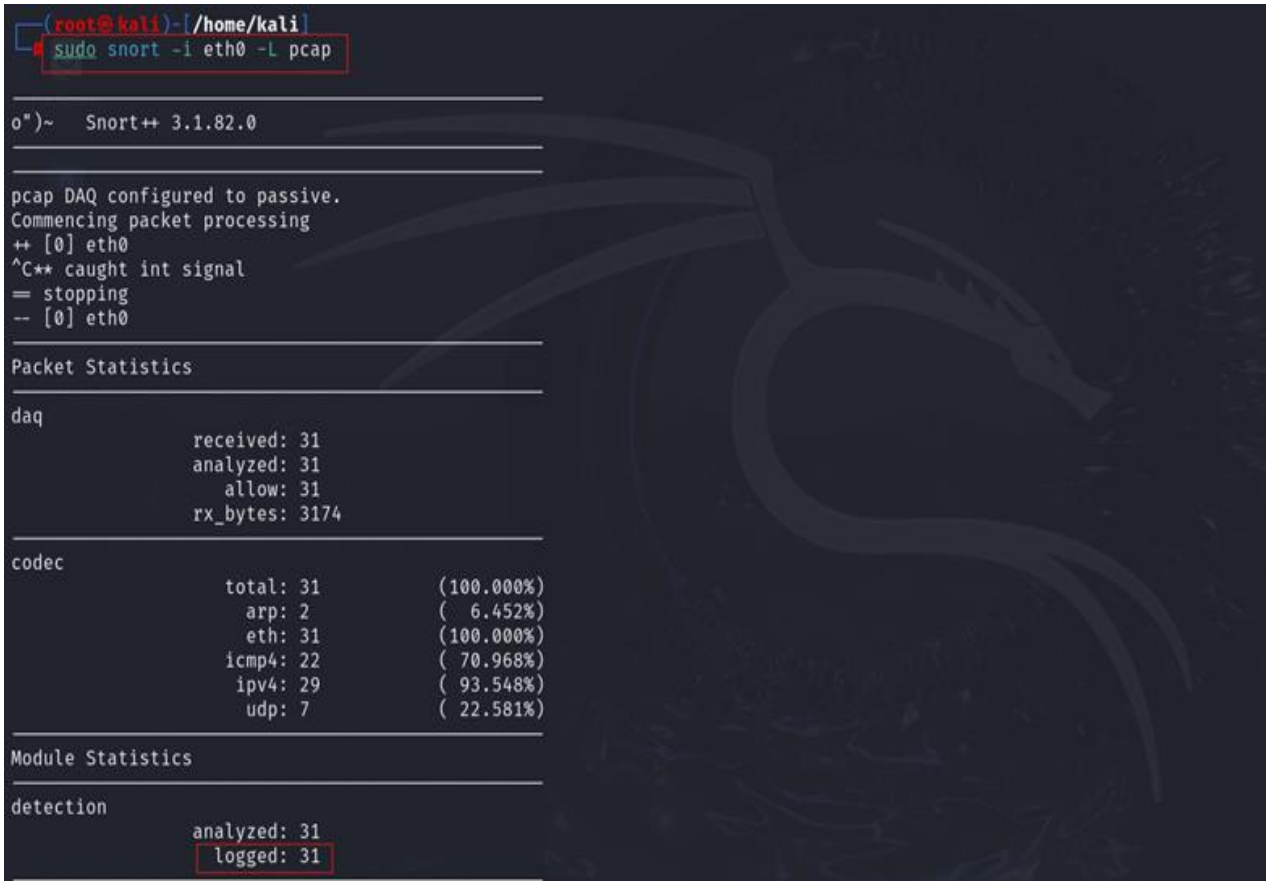
Logging Packets

If you want to save what you sniff for later analysis.

Command: sudo snort -i eth0 -L pcap

- **-i eth0:** Sniff on interface eth0.
- **-L pcap:** Log packets to a pcap file.

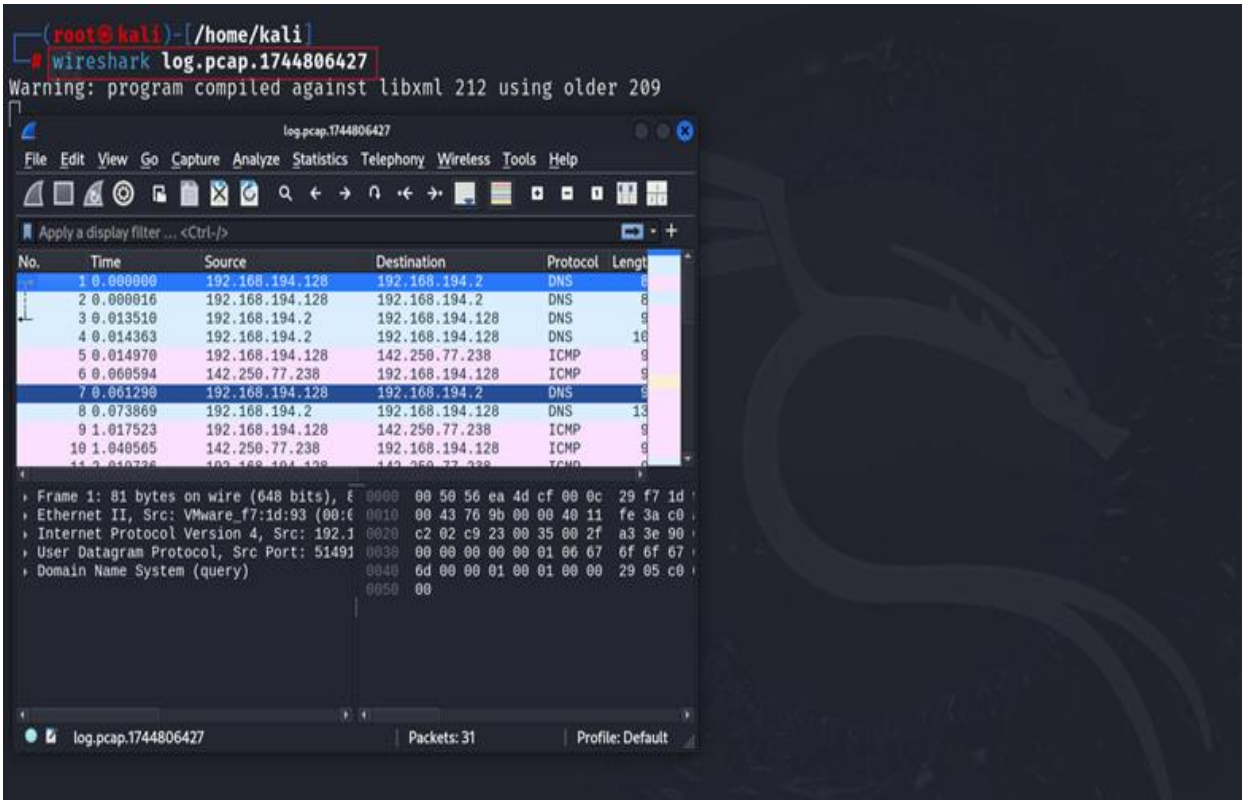
The filename will be log.pcap.TIMESTAM



This creates timestamped log files you can analyze later using tools like tcpdump or Wireshark.

Command for seeing the sniffed packets

Command: wireshark log.pcap.1744806427


(root㉿kali)-[/home/kali]
wireshark log.pcap.1744806427
Warning: program compiled against libxml 212 using older 209
log.pcap.1744806427
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Lengt
1 0.000000 192.168.194.128 192.168.194.2 DNS
2 0.000016 192.168.194.128 192.168.194.2 DNS
3 0.013510 192.168.194.2 192.168.194.128 DNS
4 0.014363 192.168.194.2 192.168.194.128 DNS
5 0.014970 192.168.194.128 142.250.77.238 ICMP
6 0.060594 142.250.77.238 192.168.194.128 ICMP
7 0.061290 192.168.194.128 192.168.194.2 DNS
8 0.073869 192.168.194.2 192.168.194.128 DNS
9 1.017523 192.168.194.128 142.250.77.238 ICMP
10 1.040565 142.250.77.238 192.168.194.128 ICMP
Frame 1: 81 bytes on wire (648 bits), 8
Ethernet II, Src: VMware_f7:1d:93 (00:0
Internet Protocol Version 4, Src: 192.1
User Datagram Protocol, Src Port: 51491
Domain Name System (query)
0000 00 50 56 ea 4d cf 00 0c 29 f7 1d
0010 00 43 76 9b 00 00 40 11 fe 3a c0
0020 c2 02 c9 23 00 35 00 2f a3 3e 90
0030 00 00 00 00 00 01 06 67 6f 6f 67
0040 6d 00 00 01 00 01 00 00 29 05 c0
0050 00
log.pcap.1744806427
Packets: 31
Profile: Default