PRACTICAL - 1

AIM:

Perform 5 different types of (port) scanning using nmap on a single port and capture the packets using wireshark and analyze the output.

Tool Introduction:

1. **Nmap:**
   - Nmap is a free and open-source network scanner created by Gordon Lyon. Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses.
   - Nmap provides a number of features for probing computer networks, including host discovery and service and operating system detection.
   - These features are extensible by scripts that provide more advanced service detection, vulnerability detection,and other features.
   - Nmap can adapt to network conditions including latency and congestion during a scan.
   - Nmap started as a Linux utility and was ported to other systems including Windows, macOS, and BSD. It is most popular on Linux, followed by Windows.

2. **Wireshark:**
   - Wireshark is a free and open-source packet analyzer.
   - It is used for network troubleshooting, analysis, software and communications protocol development, and education.
   - Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.
   - Wireshark is cross-platform, using the Qt widget toolkit in current releases to implement its user interface, and using pcap to capture packets; it runs on Linux, macOS, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows.
   - There is also a terminal-based (non-GUI) version called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of version 2 of the GNU General Public License.

Port Scanning:

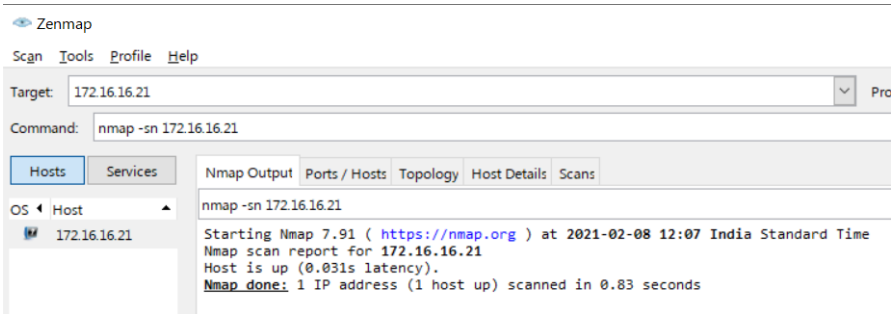
1. Host Discovery / Ping Scan:

Ping scan in nmap is done to check if the target host is alive or not. As we know that ping by default send the ICMP echo request and gets an ICMP echo reply if the system is alive.

Command: nmap -sn ip domain_name ip_range subnet - 172.16.1.1/24

Example: nmap -sn 172.16.16.21

nmap Screen Shot:



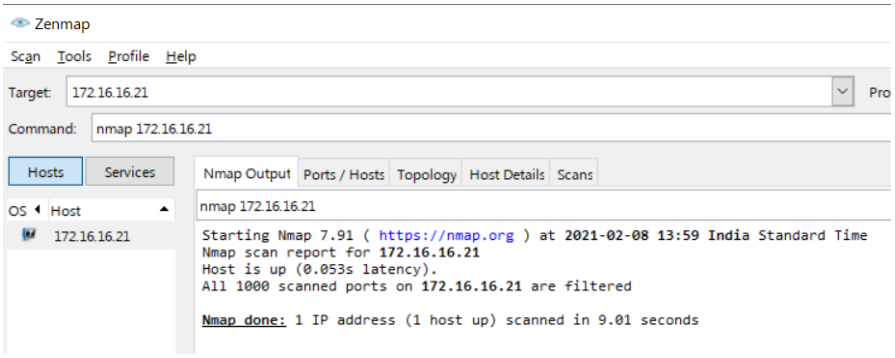
2. Default Scan:

By default, Nmap scans the most common 1,000 ports for each protocol. This option specifies which ports you want to scan and overrides the default. Individual port numbers are OK, as are ranges separated by a hyphen (e.g.1-1023).

Command: nmap ip

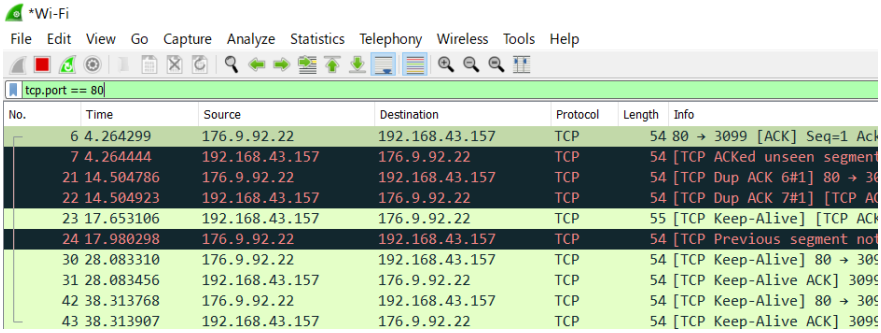
Example: nmap 172.16.16.21

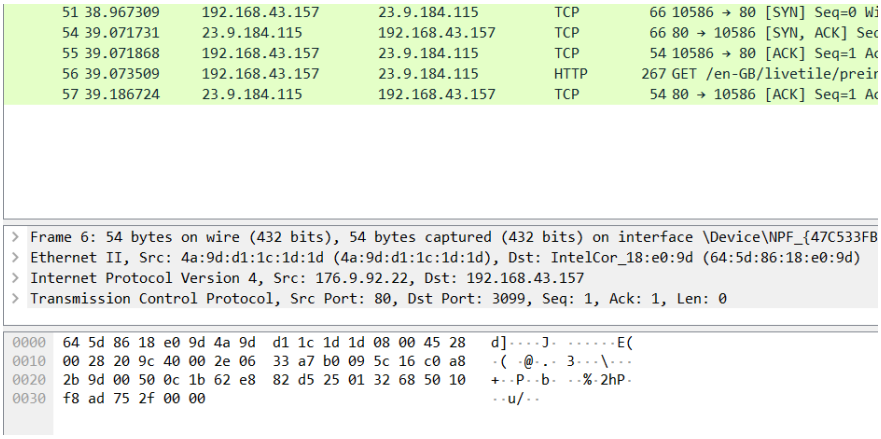
nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:





By default, Nmap performs a SYN Scan, though it substitutes a connect scan if the user does not have proper privileges to send raw packets (requires root access on Unix).not have proper privileges to send raw packets (requires root access on Unix).

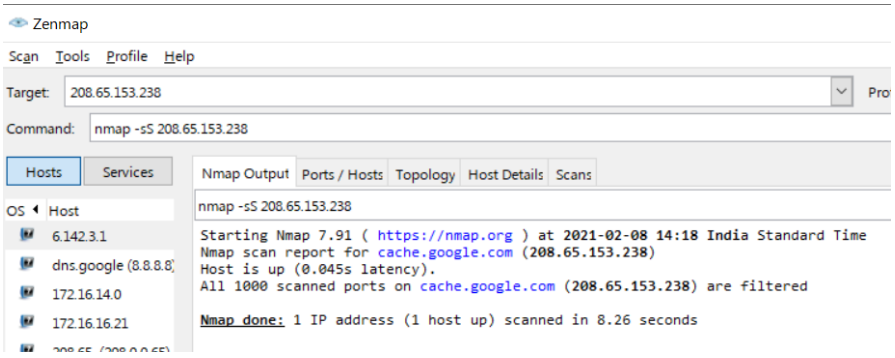
3. TCP SYN Scan:

SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections. SYN scan works against any compliant TCP stack rather than depending on idiosyncrasies of specific platforms as Nmap's FIN/NULL/Xmas, Maimon and idle scans do. It also allows clear, reliable differentiation between the open, closed, and filtered states.

Command: nmap -sS ip

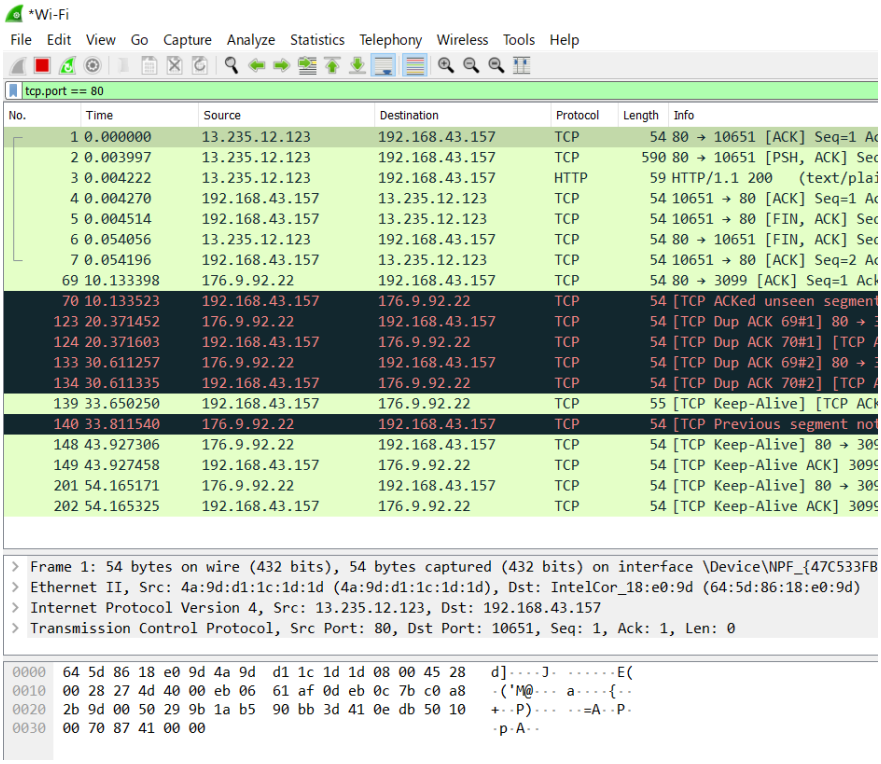
Example: nmap -sS 208.65.153.238

nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:



You send a SYN packet, as if you are going to open a real connection and then wait for a response. A SYN/ACK indicates the port is listening (open), while a RST (reset) is indicative of a non-listener.

If no response is received after several retransmissions, the port is marked as filtered.

The port is also marked filtered if an ICMP unreachable error (type 3, code 0, 1, 2, 3, 9, 10, or 13) is received. The port is also considered open if a SYN packet (without the ACK flag) is received in response.

4. TCP Connect Scan:

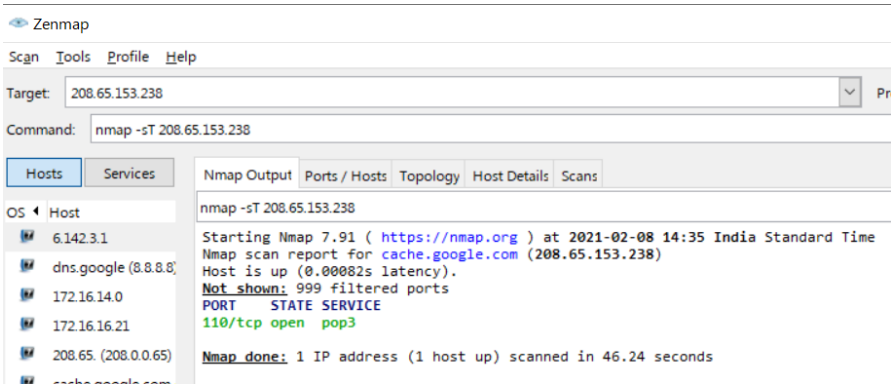
TCP connect scan is the default TCP scan type when SYN scan is not an option. This is the case when a user does not have raw packet privileges. Instead of writing raw packets as most other scan types do, Nmap asks the underlying operating system to establish a connection with the target machine and port by issuing the connect system call. This is the same high-level system call that web browsers, P2P clients, and most other network-enabled applications use to establish a connection.

It is part of a programming interface known as the Berkeley Sockets API. Rather than read raw packet responses off the wire, Nmap uses this API to obtain status information on each connection attempt.

Command: nmap -sT ip

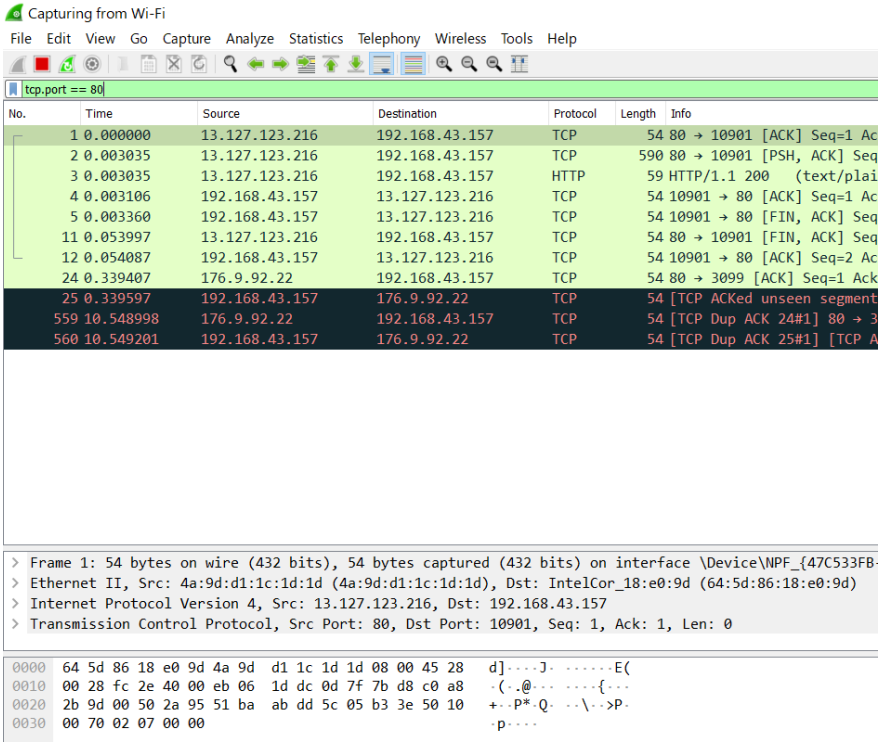
Example: nmap -sT 208.65.153.238

nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:



The system call completes connections to open target ports rather than performing the half-open reset that SYN scan does. Not only does this take longer and require more packets to obtain the same information, but target machines are more likely to log the connection.

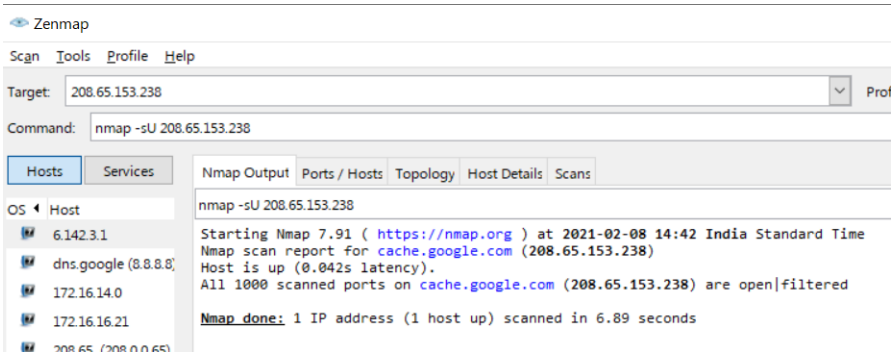
5. UDP Scan:

While most popular services on the Internet run over the TCP protocol, UDP services are widely deployed. DNS, SNMP, and DHCP (registered ports 53, 161/162, and 67/68) are three of the most common. Because UDP scanning is generally slower and more difficult than TCP, some security auditors ignore these ports. This is a mistake, as exploitable UDP services are quite common and attackers certainly don't ignore the whole protocol. Fortunately, Nmap can help inventory UDP ports.

Command: nmap -sU ip

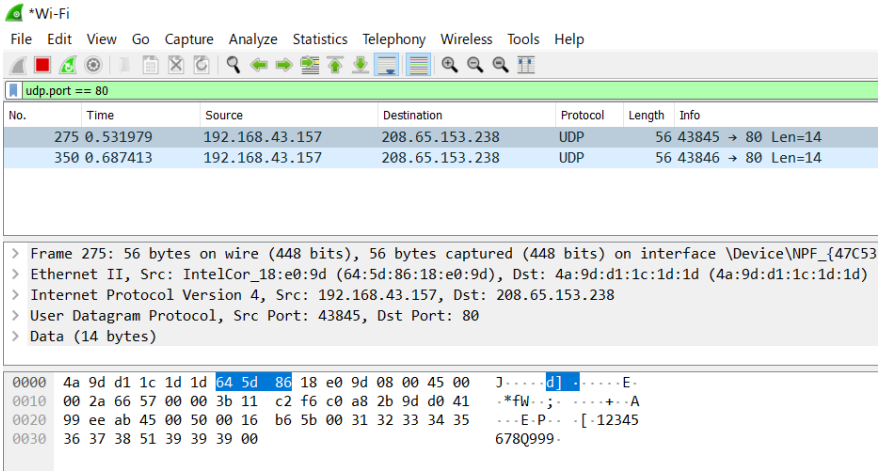
Example: nmap -sU 208.65.153.238

nmap Screen Shot:



Filter: udp.port == 80

Wireshark Screen Shot:



UDP scan works by sending a UDP packet to every targeted port. For some common ports such as 53 and 161, a protocol-specific payload is sent to increase response rate, but for most ports the packet is empty unless the --data, --data-string, or --data-length options are specified.

If an ICMP port unreachable error (type 3, code 3) is returned, the port is closed. Other ICMP unreachable errors (type 3, codes 0, 1, 2, 9, 10, or 13) mark the port as filtered. Occasionally, a service will respond with a UDP packet, proving that it is open.

If no response is received after retransmissions, the port is classified as open/filtered.

This means that the port could be open, or perhaps packet filters are blocking the communication.

6. TCP Null, TCP Xmas, TCP FIN Scan:

These three scan types (even more are possible with the --scanflags option described in the next section) exploit a subtle loophole in the TCP RFC to differentiate between open and closed ports. Page 65 of RFC 793 says that "if the [destination] port state is CLOSED .... an incoming segment not containing a RST causes a RST to be sent in response."

Then the next page discusses packets sent to open ports without the SYN, RST, or ACK bits set, stating that: "you are unlikely to get here, but if you do, drop the segment, and return."

When scanning systems compliant with this RFC text, any packet not containing SYN, RST, or ACK bits will result in a returned RST if the port is closed and no response at all if the port is open.

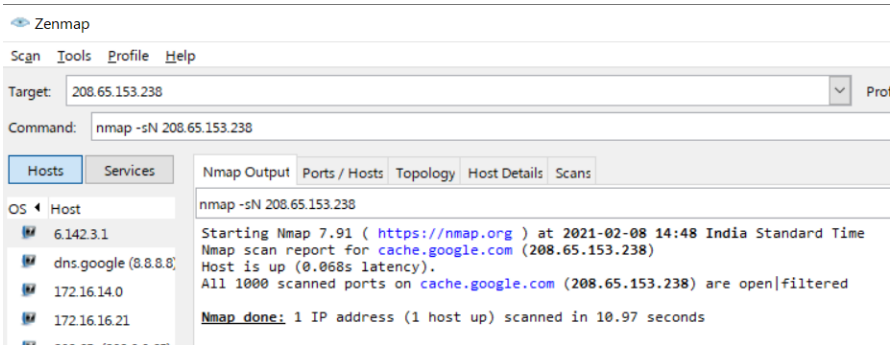
As long as none of those three bits are included, any combination of the other three (FIN, PSH, and URG) are OK.

1) Null

Command: nmap -sN ip

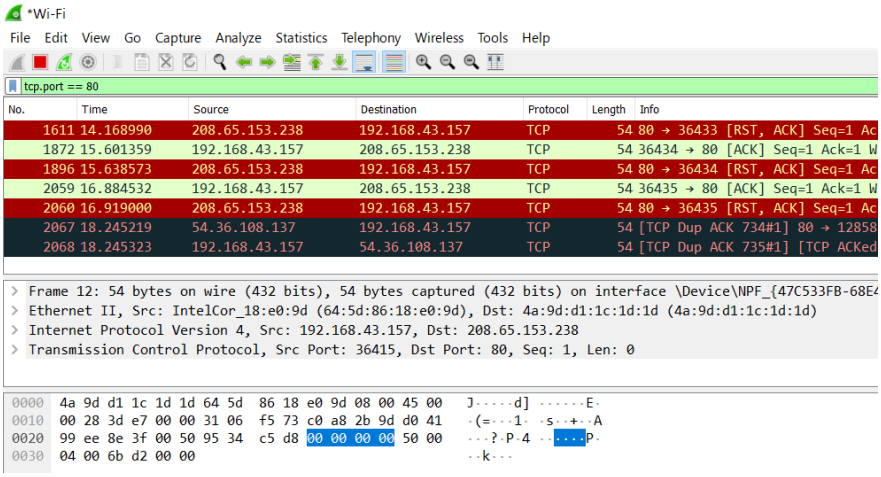
Example: nmap -sN 208.65.153.238

nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:

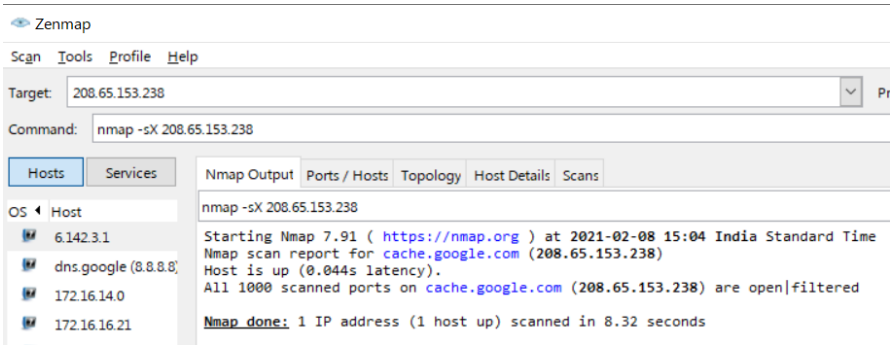


2) Xmas

Command: nmap -sX ip

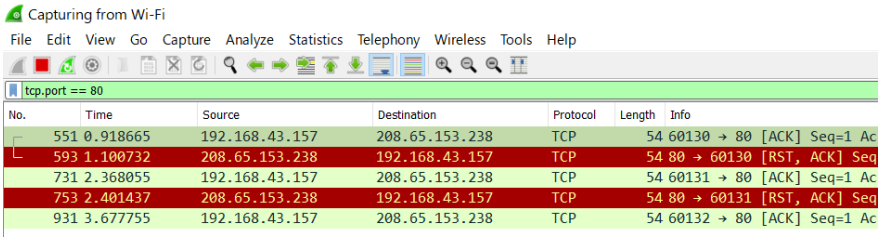
Example: nmap -sX 208.65.153.238

nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:



```
> Frame 551: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{47C533F
> Ethernet II, Src: IntelCor_18:e0:9d (64:5d:86:18:e0:9d), Dst: 4a:9d:d1:1c:1d:1d (4a:9d:d1:1c:1d:1d)
> Internet Protocol Version 4, Src: 192.168.43.157, Dst: 208.65.153.238
> Transmission Control Protocol, Src Port: 60130, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
```

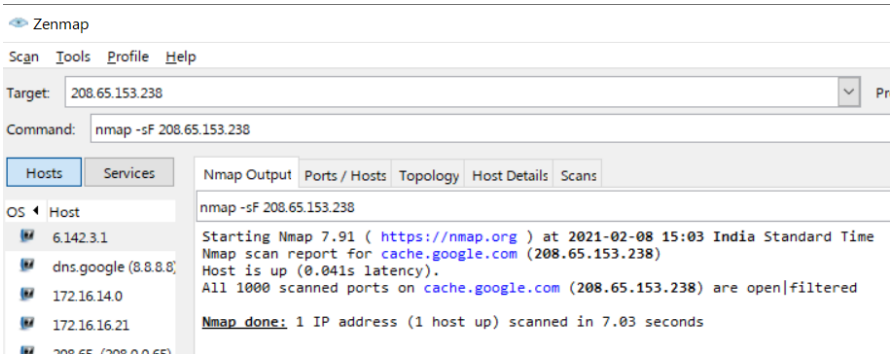
```
0000  4a 9d d1 1c 1d 1d 64 5d  86 18 e0 9d 08 00 45 00   J·····d] ····· E·
0010  00 28 72 ad 00 00 39 06  b8 ad c0 a8 2b 9d d0 41   ·(r···9· ····+··A
0020  99 ee ea e2 00 50 00 00  00 00 ea 06 dc 7a 50 10   ·····P·· ····· zP·
0030  04 00 a3 aa 00 00                                  ·····
```

3) FIN

Command: nmap -sF ip

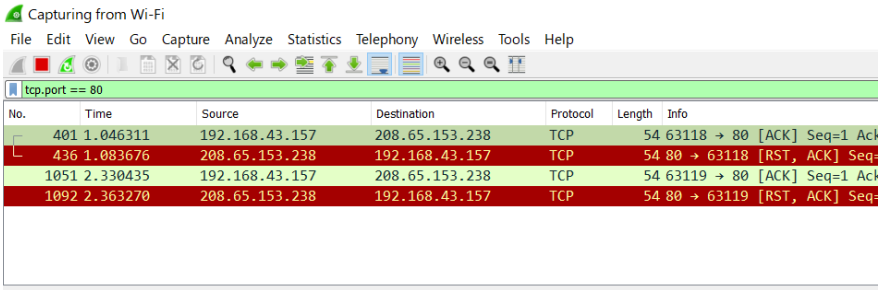
Example: nmap -sF 208.65.153.238

nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:




```
> Frame 401: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{47C533F
> Ethernet II, Src: IntelCor_18:e0:9d (64:5d:86:18:e0:9d), Dst: 4a:9d:d1:1c:1d:1d (4a:9d:d1:1c:1d:1d)
> Internet Protocol Version 4, Src: 192.168.43.157, Dst: 208.65.153.238
> Transmission Control Protocol, Src Port: 63118, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
```

```
0000  4a 9d d1 1c 1d 1d 64 5d  86 18 e0 9d 08 00 45 00   J·····d] ····· E·
0010  00 28 2c 1c 00 00 2b 06  0d 3f c0 a8 2b 9d d0 41   ·(,···+· ·?··+··A
0020  99 ee f6 8e 00 50 00 00  00 00 14 c9 f5 84 50 10   ·····P·· ····· P·
0030  04 00 54 32 00 00                                  ··T2··
```

These three scan types are exactly the same in behavior except for the TCP flags set in probe packets. If a RST packet is received, the port is considered closed, while no response means it is open/filtered. The port is marked filtered if an ICMP unreachable error (type 3, code 0, 1, 2, 3, 9, 10, or 13) is received.

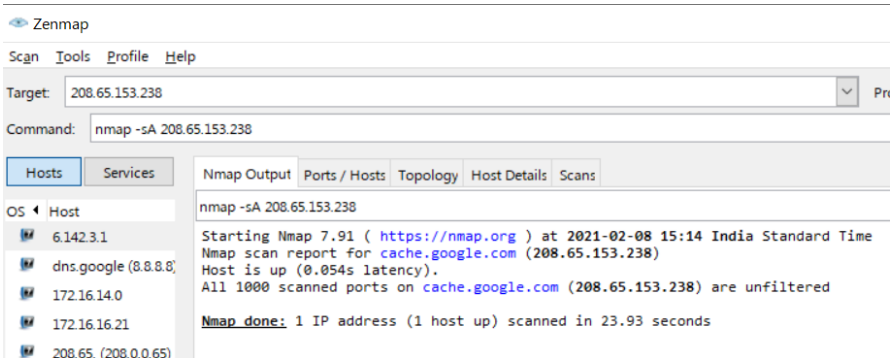
7. TCP ACK Scan:

This scan is different than the others discussed so far in that it never determines open (or even open/filtered) ports. It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered.

Command: nmap -sA ip

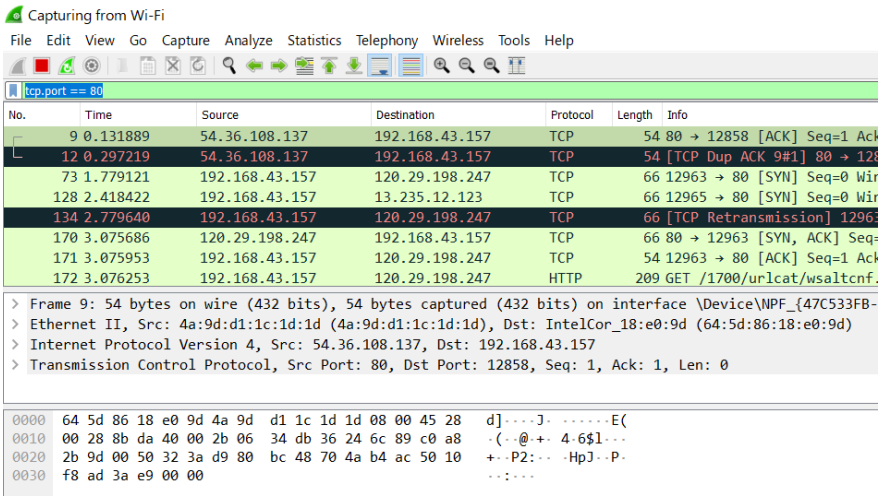
Example: nmap -sA 208.65.153.238

nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:



The ACK scan probe packet has only the ACK flag set (unless you use --scanflags). When scanning unfiltered systems, open and closed ports will both return a RST packet.

Nmap then labels them as unfiltered, meaning that they are reachable by the ACK packet, but whether they are open or closed is undetermined. Ports that don't respond, or send certain ICMP error messages back (type 3, code 0, 1, 2, 3, 9, 10, or 13), are labeled filtered.

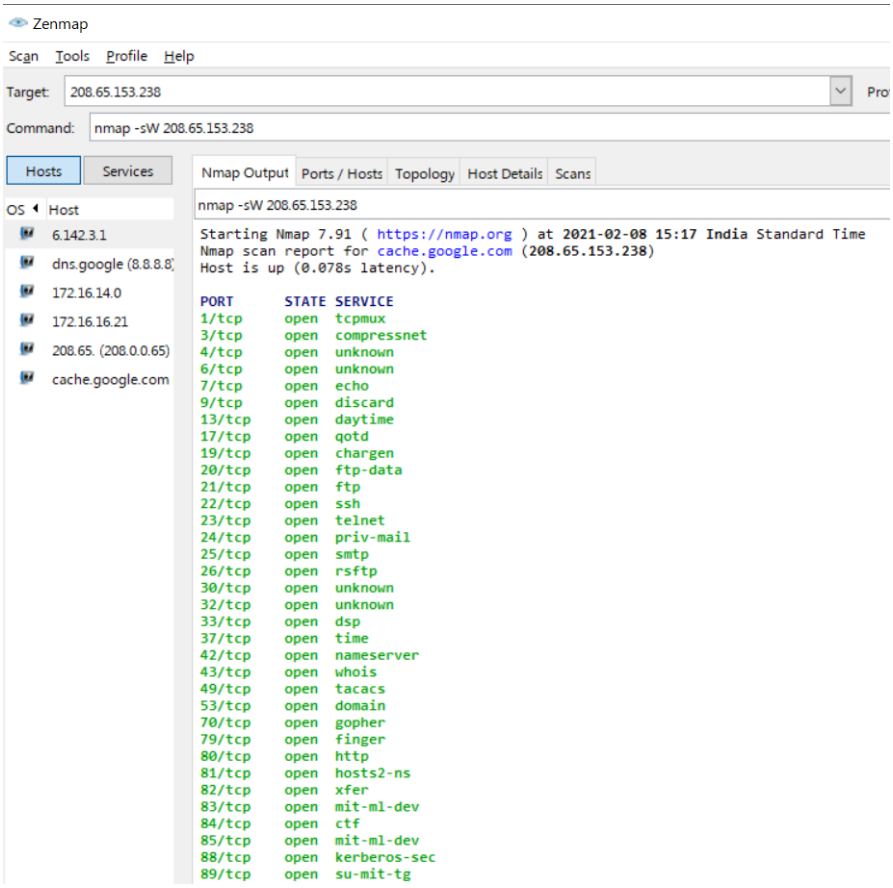
8. TCP Window Scan:

Window scan is exactly the same as ACK scan except that it exploits an implementation detail of certain systems to differentiate open ports from closed ones, rather than always printing unfiltered when a RST is returned. It does this by examining the TCP Window field of the RST packets returned. On some systems, open ports use a positive window size (even for RST packets) while closed ones have a zero window. So instead of always listing a port as unfiltered when it receives a RST back, Window scan lists the port as open or closed if the TCP Window value in that reset is positive or zero, respectively.

Command: nmap -sW ip

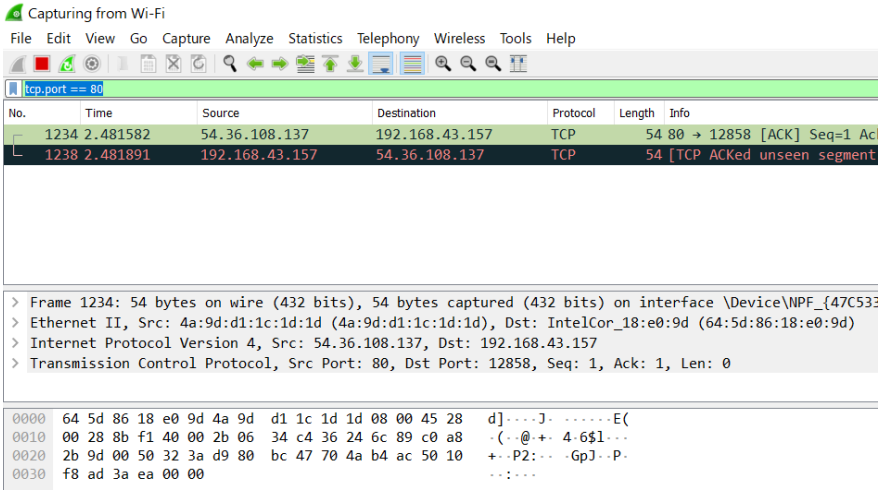
Example: nmap -sW 208.65.153.238

nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:



This scan relies on an implementation detail of a minority of systems out on the Internet, so you can't always trust it. Systems that don't support it will usually return all ports closed. Of course, it is possible that the machine really has no open ports.

If most scanned ports are closed but a few common port numbers (such as 22, 25, 53) are filtered, the system is most likely susceptible. Occasionally, systems will even show the exact opposite behavior. If your scan shows 1,000 open ports and three closed or filtered ports, then those three may very well be the truly open ones.

9. TCP PSH Scan:

Truly advanced Nmap users need not limit themselves to the canned scan types offered. The -scanflags option allows you to design your own scan by specifying arbitrary TCP flags.

Let your creative juices flow, while evading intrusion detection systems whose vendors simply paged through the Nmap man page adding specific rules!

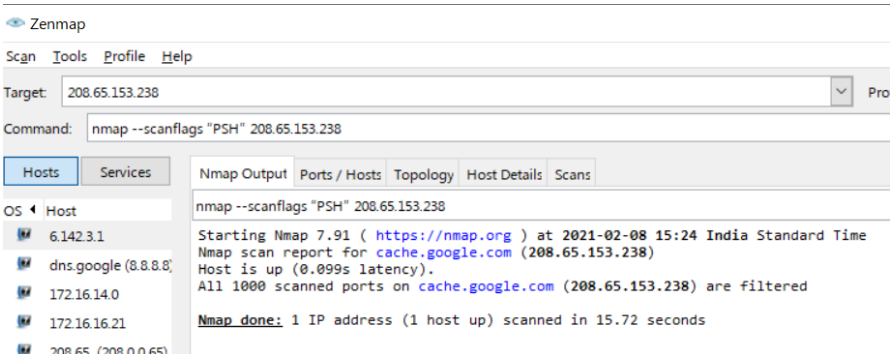
The -scanflags argument can be a numerical flag value such as 9 (PSH and FIN), but using symbolic names is easier.

Just mash together any combination of URG, ACK, PSH, RST, SYN, and FIN. For example, -scanflags URGACKPSHRSTSYNFIN sets everything, though it's not very useful for scanning.

Command: nmap -scanflags flag ip

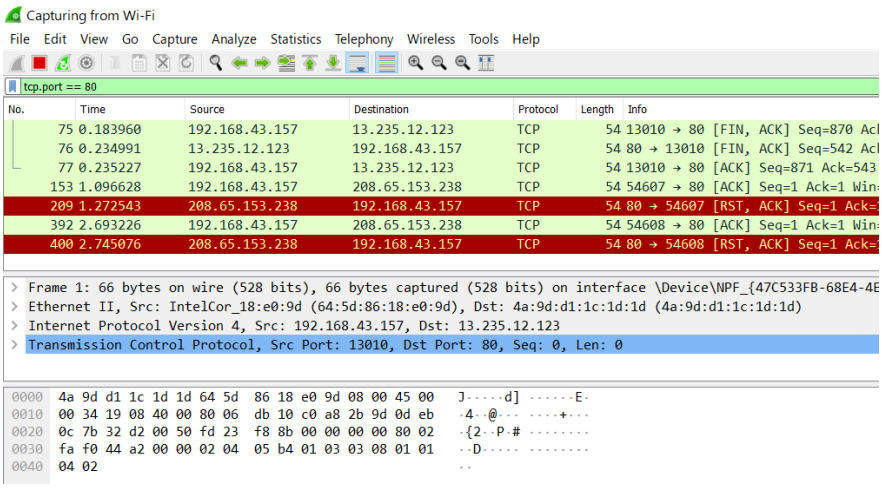
Example: nmap -scanflags “PSH” c

nmap Screen Shot:



Filter: tcp.port == 80

Wireshark Screen Shot:



A SYN scan considers no-response to indicate a filtered port, while a FIN scan treats the same as open/filtered.

10. Other:

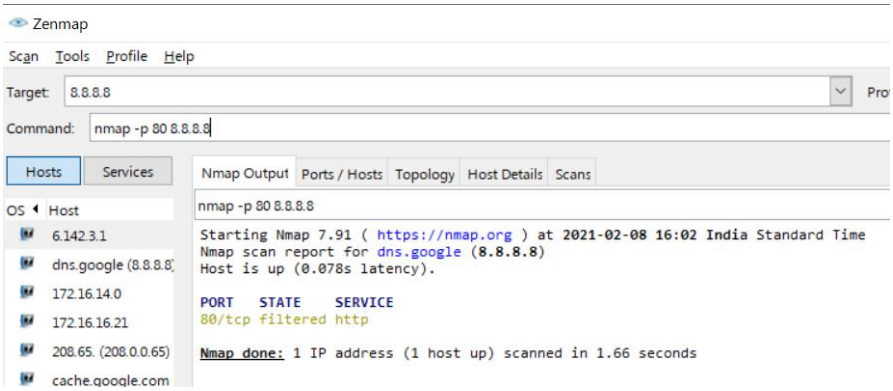
1) **-p:**

   This option specifies which ports you want to scan and overrides the default. Individual port numbers are OK, as are ranges separated by a hyphen (e.g. 1-1023).

   The beginning and/or end values of a range may be omitted, causing Nmap to use 1 and 65535, respectively.

   So you can specify -p- to scan ports from 1 through 65535. Scanning port zero is allowed if you specify it explicitly.

For IP protocol scanning (-sO), this option specifies the protocol numbers you wish to scan for (0–255).



2) -p*:

This option will scan all the reachable ports of the target.

3) -PO:

One of the newer host discovery options is the IP protocol ping, which sends IP packets with the specified protocol number set in their IP header.

The protocol list takes the same format as do port lists in the previously discussed TCP, UDP and SCTP host discovery options.

If no protocols are specified, the default is to send multiple IP packets for ICMP (protocol 1), IGMP (protocol 2), and IP-in-IP (protocol 4). The default protocols can be configured at compile-time by changing DEFAULT_PROTO_PROBE_PORT_SPEC in nmap.h.

Note that for the ICMP, IGMP, TCP (protocol 6), UDP (protocol 17) and SCTP (protocol 132), the packets are sent with the proper protocol headers while other protocols are sent with no additional data beyond the IP header (unless any of --data, --data-string, or --data-length options are specified).

4) -T:

While the fine-grained timing controls discussed in the previous section are powerful and effective, some people find them confusing.

Moreover, choosing the appropriate values can sometimes take more time than the scan you are trying to optimize.

Fortunately, Nmap offers a simpler approach, with six timing templates. You can specify them with the -T option and their number (0–5) or their name. The template

names are paranoid (0), sneaky (1), polite (2), normal (3), aggressive (4), and insane (5).

The first two are for IDS evasion. Polite mode slows down the scan to use less bandwidth and target machine resources. Normal mode is the default and so -T3 does nothing.

Aggressive mode speeds scans up by making the assumption that you are on a reasonably fast and reliable network.

Finally insane mode assumes that you are on an extraordinarily fast network or are willing to sacrifice some accuracy for speed.

5) -A:

This option enables additional advanced and aggressive options. Presently this enables OS detection (-O), version scanning (-sV), script scanning (-sC) and traceroute (--traceroute).

More features may be added in the future. The point is to enable a comprehensive set of scan options without people having to remember a large set of flags.

However, because script scanning with the default set is considered intrusive, you should not use -A against target networks without permission.

This option only enables features, and not timing options (such as -T4) or verbosity options (-v) that you might want as well.

Options which require privileges (e.g. root access) such as OS detection and traceroute will only be enabled if those privileges are available.

6) -v:

Increases the verbosity level, causing Nmap to print more information about the scan in progress.

Open ports are shown as they are found and completion time estimates are provided when Nmap thinks a scan will take more than a few minutes.

Use it twice or more for even greater verbosity: -vv, or give a verbosity level directly, for example -v3.

7) -d:

When even verbose mode doesn't provide sufficient data for you, debugging is available to flood you with much more!

As with the verbosity option (-v), debugging is enabled with a command-line flag (-d) and the debug level can be increased by specifying it multiple times, as in -dd, or by setting a level directly.

For example, -d9 sets level nine. That is the highest effective level and will produce thousands of lines unless you run a very simple scan with very few ports and targets.

Conclusion:

Through this PRACTICAL - I learned about different scanning using nmap and capture the packets in the wireshark.

PRACTICAL - 2

AIM:

a) Perform Port Scanning, File Transfer, Client-server chat and Basic Webserver implementation using netcat.
b) Find the service running on the particular port using netcat.

TOOL INTRODUCTION:

➢ **NETCAT**

- Netcat (often abbreviated to nc) is a computer networking utility for reading from and writing to network connections using TCP or UDP.
- The command is designed to be a dependable back-end that can be used directly or easily driven by other programs and scripts.
- At the same time, it is a feature-rich network debugging and investigation tool, since it can produce almost any kind of connection its user could need and has a number of built-in capabilities.

1. Chat Server using TCP Connection.

THEORY:

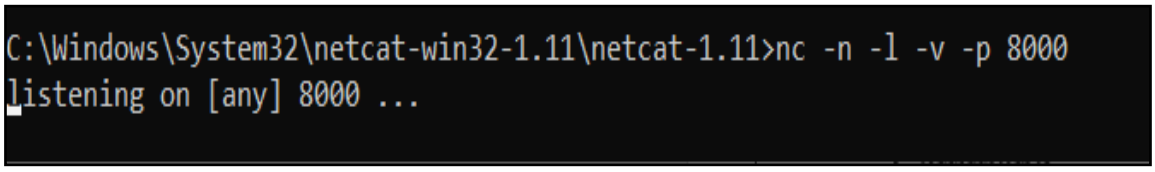
- You can use **Netcat** to create a simple command-line messaging server instantly.
- **Nc** must be installed on both systems used for the chat room.
- On the system, run the following commands to create the chat server listening on port 8000.

Implementation:

Step1: To make windows OS as listener

Command: **nc -n -l -v -p 8000**

Snap Shot:



Step 2: To make Kali Linux as client

Command: **nc -n 192.168.56.1 8000**

Snap Shot:

```
┌──(root㉿kali)-[~]
└─$ nc -n 192.168.56.1 8000
```

Step3: Message passing:

Write in Linux terminal and result will be displayed in windows terminal also.

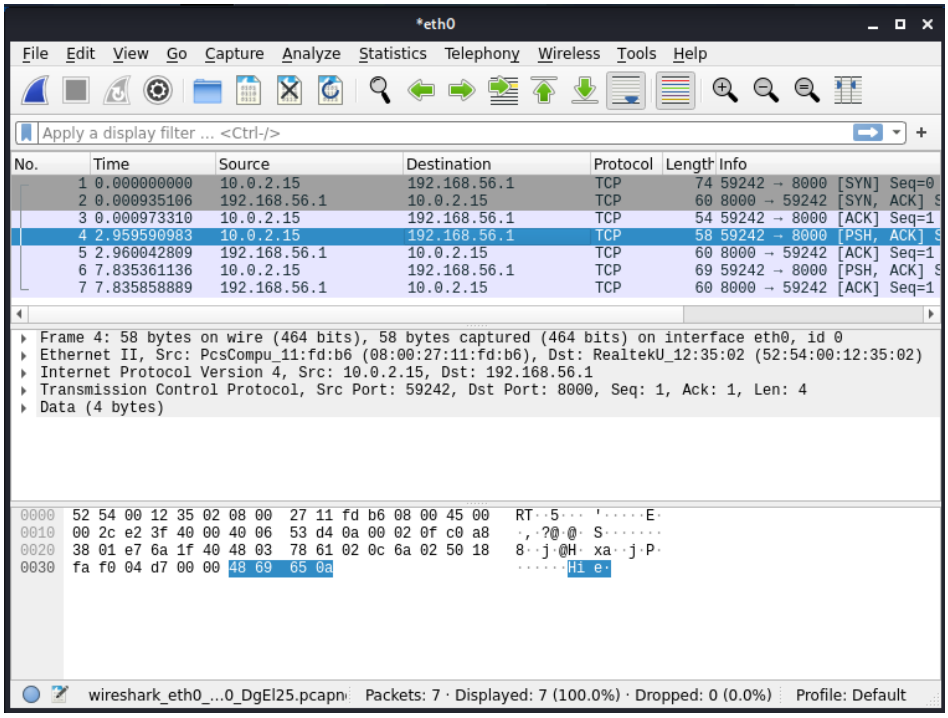
Snap Shot:

```
┌──(root㉿kali)-[~]
└─$ nc -n 192.168.56.1 8000
Hie
this is R User
```

```
C:\Windows\System32\netcat-win32-1.11\netcat-1.11>nc -n -l -v -p 8000
listening on [any] 8000 ...
connect to [192.168.56.1] from (UNKNOWN) [192.168.56.1] 59464
Hie
this is User
```

ANALYSIS:

Snap shot of Wire Shark:



The above connection is done using TCP thus resulting in sending and receiving ACK signal which are captured by wireshark.

2. Chat Server using UDP Connection

Implementation:

Step1: To make windows os as listener

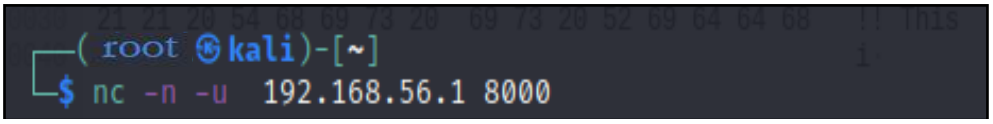
Command: **nc -n -u -l -v -p 8000**

Snap Shot

Step 2: To make Kali Linux as client

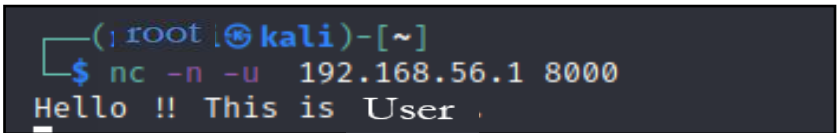
Command: **nc -n -u 192.168.56.1 8000**

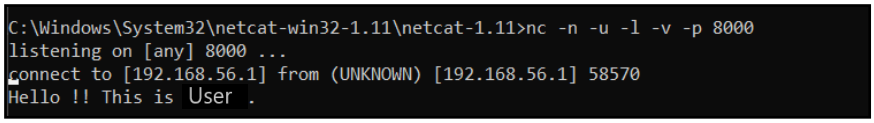
Snap Shot



Step3: Message passing

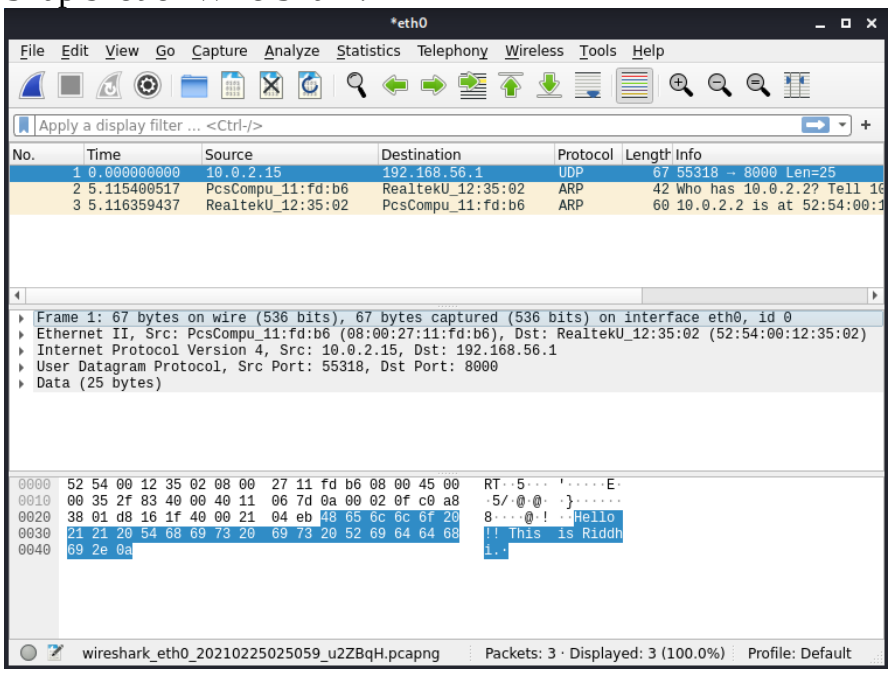
Snap Shot





ANALYSIS:

Snap shot of Wire Shark:



The above connection is done using UDP thus no ACK signal.

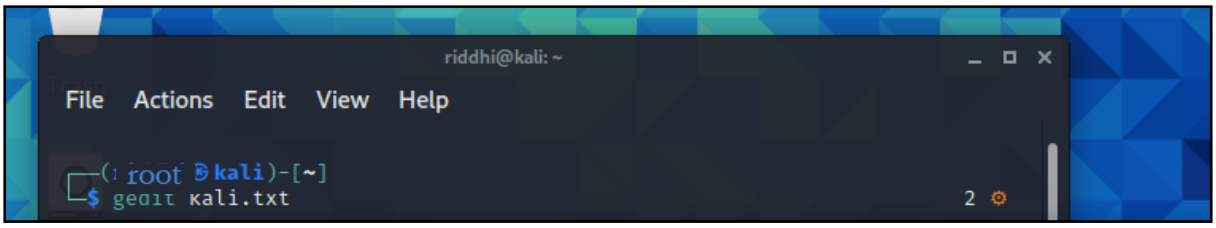
3. **FILE TRANSFER**

Introduction:

- The nc (netcat) command can be used to transfer arbitrary data over the network.
- It represents a quick way for Linux administrators to transfer data without the need for an additional data transfer services such as FTP, HTTP, SCP etc.

Implementation:

Step 1: Create a file named kali.txt at sender side.



Step 2:

Run command **nc -n -l -v -p 8000 >windows.txt**

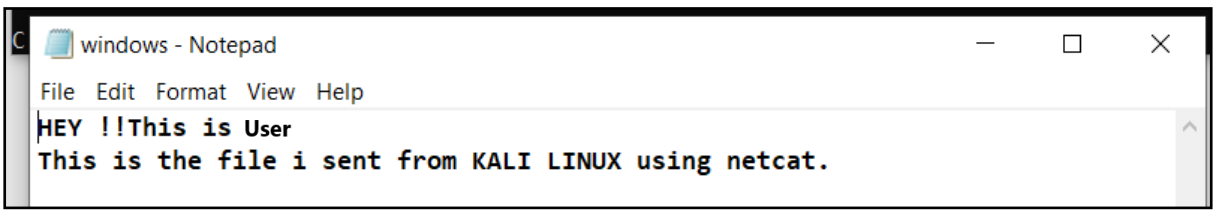
will begin listening on port 8000.

Step 3:

Run command **nc -n 192.168.56.1 8000 < kali.txt**

will connect to the receiver and begin sending file.

Step 4: Now, check your received file from received machine i.e. Windows here.



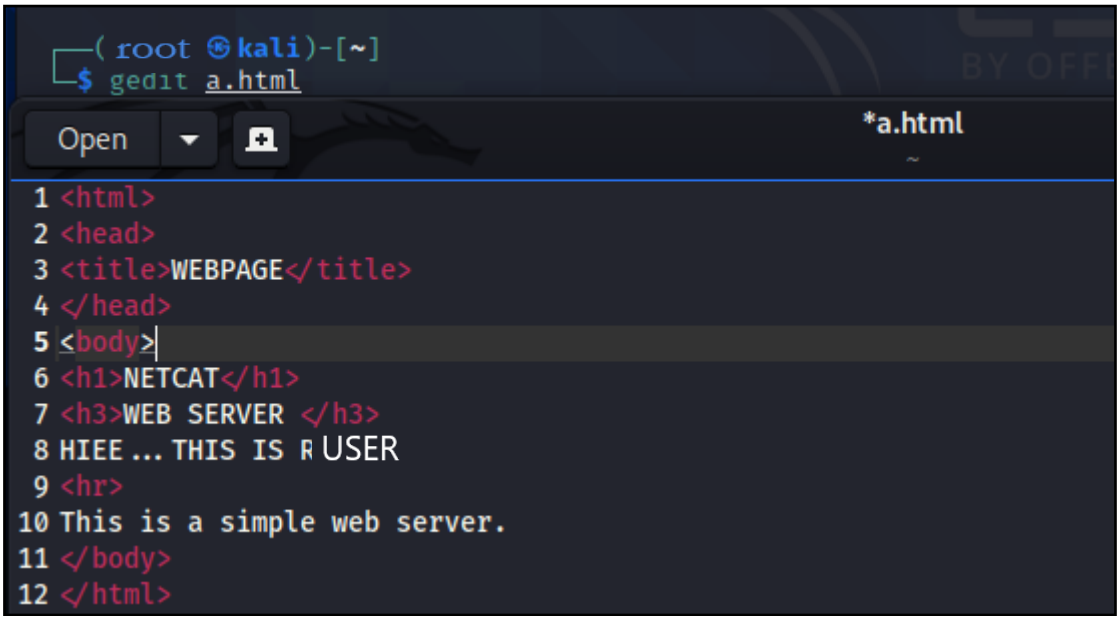
4. WEB SERVER

Introduction :

- The netcat tool nc can operate as a TCP client. Because HTTP works over TCP, nc can be used as an HTTP server!
- Because nc is a UNIX tool, we can use it to make custom web servers: servers which return any HTTP headers you want, servers which return the response very slowly, servers which return invalid HTTP, etc.

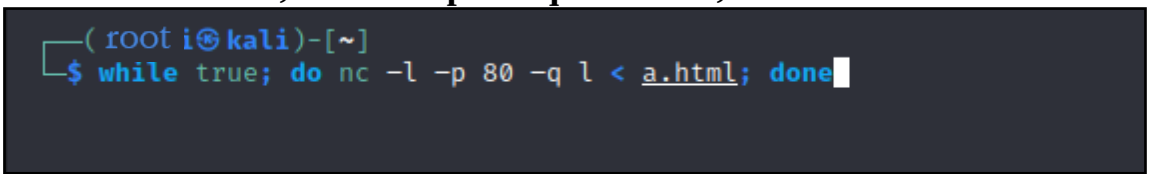
Implementation:

Step 1: Create a new text document on your local system and make sure to use valid HTML tags. Then save the file as **"a.html"**

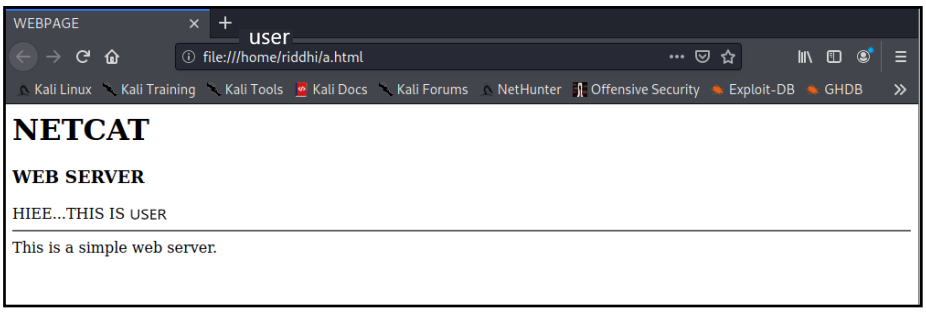


Step 2:

Command: **while true; do nc –l –p 80 –q l < a.html; done**



Step 3 : Open your file in web browser entering the required filepath.



5. WINDOWS OS HACK

Introduction:

- Most common use for Netcat when it comes to hacking is setting up reverse and bind shells, piping and redirecting network traffic, port listening, debugging programs and scripts and banner grabbing.
- System hacking is a vast subject that consists of hacking the different software-based technological systems such as laptops, desktops, etc.

Implementation:

Step 1:

Command: **nc -n -l -v -p 8000 -e cmd.exe**

```
C:\Windows\System32\netcat-win32-1.11\netcat-1.11>nc -n -l -v -p 8000 -e cmd.exe
listening on [any] 8000 ...
```

Step 2:

Command: **nc 192.168.56.1 8000**

```
┌──(root㉿kali)-[~]
└─$ nc 192.168.56.1 8000
```

Step 3: ACCESS TRANSFERED

```
┌──(root㉿kali)-[~]
└─$ nc 192.168.56.1 8000
Microsoft Windows [Version 10.0.19042.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\System32\netcat-win32-1.11\netcat-1.11>
```

CONCLUSION

Thus, in this PRACTICAL - I have performed File Transfer, Client-server chat, Basic Webserver implementation and even OS hacking using netcat.

PRACTICAL - 3

AIM:

Use Wireshark and netsniff to perform Cryptanalytic attacks using packet sniffing.

TOOL INTRODUCTION:

➢ **NETSNIFF-NG**

- It is a fast, minimal tool to analyze network packets, capture pcap files, replay pcap files, and redirect traffic between interfaces with the help of zero-copy packet sockets.
- It is Linux specific, meaning there is no support for other operating systems.
- It uses both Linux specific RX_RING and TX_RING interfaces to perform zero-copy.
- It also supports early packet filtering in the kernel. It has support for low-level and high-level packet filters that are translated into Berkeley Packet Filter instructions.
- Since netsniff-ng does not establish any state or perform reassembly during packet dissection, its memory footprint is quite low, thus, making netsniff-ng quite efficient for offline analysis of large pcap files as well.

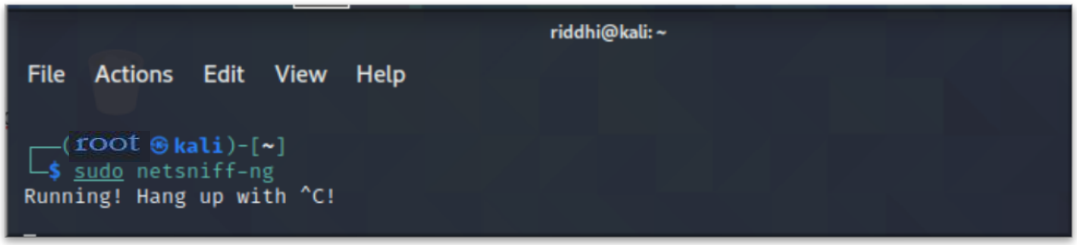
- Use of Netsniff-ng
  - ✓ Debug netlink traffic.
  - ✓ Supports analysis, replaying, and dumping of raw 802.11 frames.

IMPLEMENTATION:

1. Start the netsniff-ng tool

   Command: **netsniff-ng**

   SNAPSHOT:



2. Running basic command to get the MAGIC CODE:

   Command: **netsniff-ng -D**

   SNAPSHOT:

```
┌──(root㉿kali)-[~]
└─$ sudo netsniff-ng -D
tcpdump-capable pcap:
  magic: 0×a1b2c3d4 (swapped: 0×d4c3b2a1)
  features:
    timeval in us
    packet length
    packet cap-length
tcpdump-capable pcap with ns resolution:
  magic: 0×a1b23c4d (swapped: 0×4d3cb2a1)
  features:
    timeval in ns
    packet length
    packet cap-length
Alexey Kuznetzov's pcap:
  magic: 0×a1b2cd34 (swapped: 0×34cdb2a1)
  features:
    timeval in us
    packet length
    packet cap-length
    packet ifindex
    packet protocol
    packet type
netsniff-ng pcap:
  magic: 0×a1e2cb12 (swapped: 0×12cbe2a1)
  features:
    timeval in ns
    timestamp source
    packet length
    packet cap-length
    packet ifindex
    packet protocol
    hardware type
    packet type
```

3. Running basic netsniff command to capture packets

   Command: **sudo netsniff-ng --in eth0 --out demo.pcap -T 0xa1e2cb12 -b 0 tcp or udp**

   SNAPSHOT:

```
┌──(root㉿kali)-[~]
└─$ sudo netsniff-ng --in eth0 --out demo.pcap -T  0×a1e2cb12 -b 0 tcp or udp
Running! Hang up with ^C!

> eth0 70 1617225409s.530496228ns #1
 [ Eth MAC (08:00:27:11:fd:b6 ⇒ 52:54:00:12:35:02), Proto (0×0800, IPv4) ]
 [ Vendor (Cadmus Computer Systems ⇒ Locally Administered) ]
 [ IPv4 Addr (10.0.2.15 ⇒ 192.168.29.1), Proto (17), TTL (64), TOS (0), Ver (4), IHL (5), Tlen (56), ID (56454), Res (0), NoFrag (1), MoreFrag (0), FragOff (0), CSum (0×7476) is ok ]
 [ UDP Port (37225 ⇒ 53 (domain)), Len (36 Bytes, 28 Bytes Data), CSum (0×e9ed) ]
 [ Chr $...........google.com..... ]
 [ Hex  24 c9 01 00 00 01 00 00 00 00 00 00 06 67 6f 6f 67 6c 65 03 63 6f 6d 00 00 01 00 01 ]

> eth0 70 1617225409s.530550016ns #2
 [ Eth MAC (08:00:27:11:fd:b6 ⇒ 52:54:00:12:35:02), Proto (0×0800, IPv4) ]
 [ Vendor (Cadmus Computer Systems ⇒ Locally Administered) ]
 [ IPv4 Addr (10.0.2.15 ⇒ 192.168.29.1), Proto (17), TTL (64), TOS (0), Ver (4), IHL (5), Tlen (56), ID (56455), Res (0), NoFrag (1), MoreFrag (0), FragOff (0), CSum (0×7475) is ok ]
 [ UDP Port (37225 ⇒ 53 (domain)), Len (36 Bytes, 28 Bytes Data), CSum (0×e9ed) ]
 [ Chr 2...........google.com..... ]
 [ Hex  32 b5 01 00 00 01 00 00 00 00 00 00 06 67 6f 6f 67 6c 65 03 63 6f 6d 00 00 1c 00 01 ]
```

```
< eth0 166 1617225418s.605385866ns #24
 [ Eth MAC (52:54:00:12:35:02 ⇒ 08:00:27:11:fd:b6), Proto (0×0800, IPv4) ]
 [ Vendor (Locally Administered ⇒ Cadmus Computer Systems) ]
 [ IPv4 Addr (192.168.29.1 ⇒ 10.0.2.15), Proto (17), TTL (64), TOS (0), Ver (4), IHL (5), Tlen (152), ID (1441), Res (0), NoFrag (0), MoreFrag (0), FragOff (0), CSum (0×8afc) is ok ]
 [ UDP Port (53 (domain) ⇒ 59769), Len (132 Bytes, 124 Bytes Data), CSum (0×106a) ]
 [ Chr 0:...........174.199.58.216.in-addr.arpa.............#....bom05s08-in-f174.1e100.net.........#....bom05s08-in-f14.1e100.net. ]
 [ Hex  30 3a 81 80 00 01 00 02 00 00 00 00 03 31 37 34 03 31 39 39 02 35 38 03 32 31 36 07 69 6e 2d 61 64 64 72 04 61 72 70 61 00 00 0c 00 01 c0 0c 00 0c 00 01 00 00 cc 23 00 1c 10 62
   05 31 65 31 30 30 03 6e 65 74 00 c0 0c 00 0c 00 01 00 00 cc 23 00 1b 0f 62 6f 6d 30 35 73 30 38 2d 69 6e 2d 66 31 34 05 31 65 31 30 30 03 6e 65 74 00 ]

        24  packets incoming (0 unread on exit)
        24  packets passed filter
         0  packets failed filter (out of space)
    0.0000% packet droprate
        18  sec, 292154 usec in total
```

4. Passing packets to source destination.

   Command: **ping -c 10 google.com**

   SNAPSHOT:

```
┌──(root㉿kali)-[~]
└─$ ping -c 10 google.com
PING google.com (172.217.26.238) 56(84) bytes of data.
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=1 ttl=114 time=20.0 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=2 ttl=114 time=23.0 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=3 ttl=114 time=19.5 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=4 ttl=114 time=20.4 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=5 ttl=114 time=21.0 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=6 ttl=114 time=23.3 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=7 ttl=114 time=20.6 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=8 ttl=114 time=20.0 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=9 ttl=114 time=20.5 ms
64 bytes from bom05s09-in-f14.1e100.net (172.217.26.238): icmp_seq=10 ttl=114 time=20.7 ms

--- google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 19.515/20.896/23.269/1.181 ms
```

5. Sniffing TCP packets:

   Command:

   sudo netsniff-ng --in eth0 --out demo1.pcap -s -T 0xa1e2cb12 -b 0 tcp

   or udp

   It specifies that no information will be printed on terminal.

   This command will capture TCP traffic from the networking device eth0 into the pcap file named demo1.pcap, which has netsniff-ng specific pcap extensions.

   SNAPSHOT:

```
┌──(root㉿kali)-[~]
└─$ sudo netsniff-ng --in eth0 --out demo1.pcap -s -T 0xa1e2cb12 -b 0 tcp or udp
Running! Hang up with ^C!
         24  packets incoming (0 unread on exit)
         24  packets passed filter
          0  packets failed filter (out of space)
     0.0000% packet droprate
         15  sec, 715522 usec in total
```

CONCLUSION:

Hence,we learnt about packet sniffing using netsniff-ng.

PRACTICAL - 4

AIM:

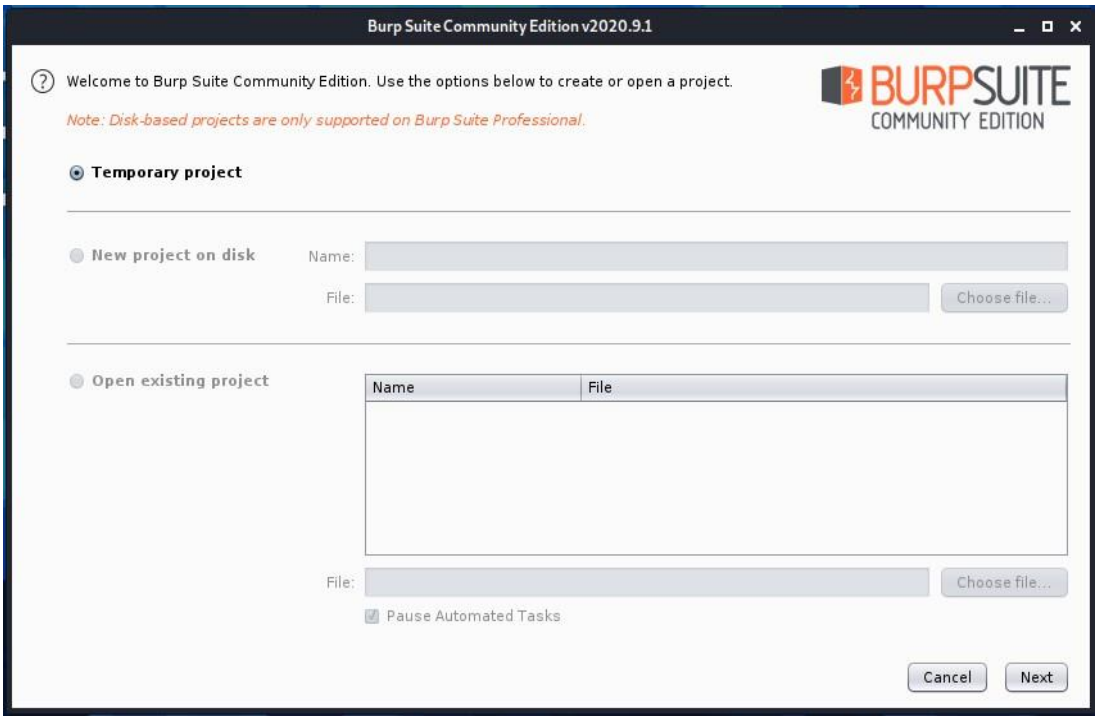
Perform automated testing using BurpSuite or ZAP proxies.

TOOL INTRODUCTION:

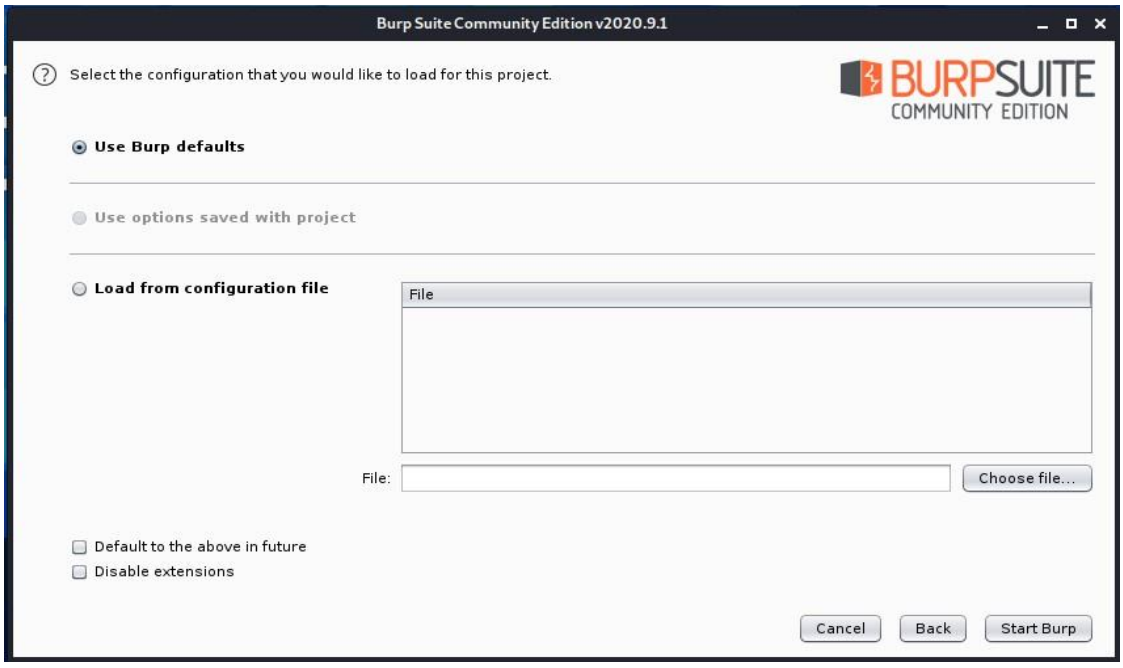
- **BURPSUITE:**
  - Burp Suite is designed to be a hands-on tool, where the user controls the actions that are performed.
  - At the core of Burp's penetration testing workflow is the ability to pass HTTP requests between the Burp tools in order to carry out particular tasks.
  - You can send messages from the "Proxy" > "Intercept", "HTTP history", or "Site map" tabs, and indeed anywhere else in Burp that you see HTTP messages.
  - To do this, select one or more messages, and use the context menu to send the request to another tool.
  - The Burp tools you will use for particular tasks are as follows:
    - ✓ Scanner - This is used to automatically scan websites for content and security vulnerabilities.
    - ✓ Intruder - This allows you to perform customized automated attacks, to carry out all kinds of testing tasks.
    - ✓ Repeater - This is used to manually modify and reissue individual HTTP requests over and over.
    - ✓ Collaborator client - This is used to generate Burp Collaborator payloads and monitor for resulting out-of-band interactions.
    - ✓ Clickbandit - This is used to generate clickjacking exploits against vulnerable applications.
    - ✓ Sequencer - This is used to analyze the quality of randomness in an application's session tokens.
    - ✓ Decoder - This lets you transform bits of application data using common encoding and decoding schemes.
    - ✓ Comparer - This is used to perform a visual comparison of bits of application data to find interesting differences.

IMPLEMENTATION:

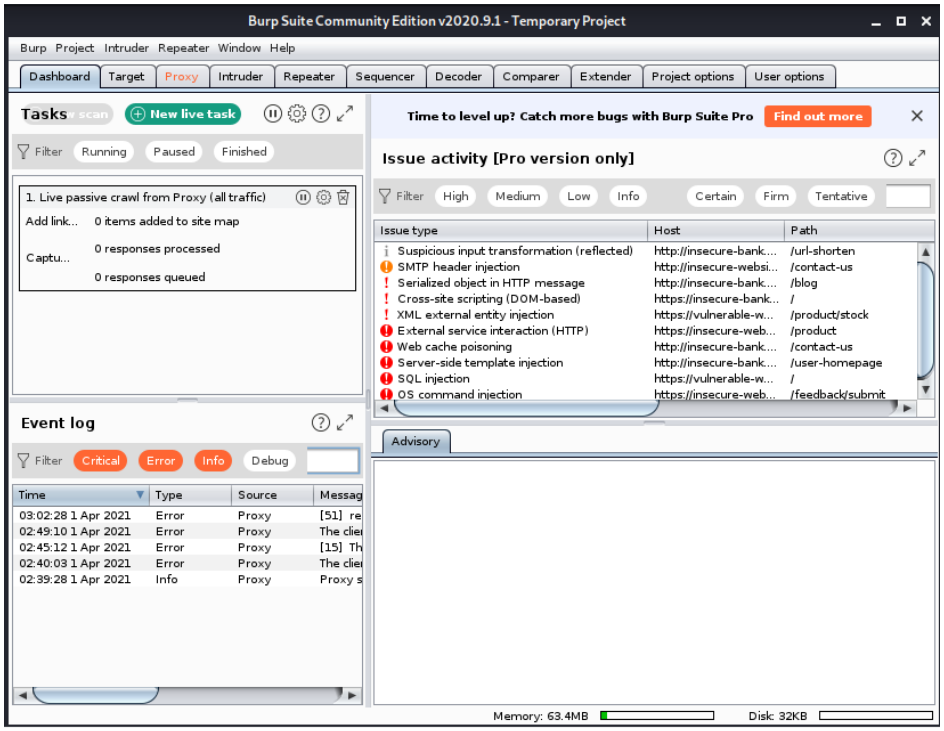
1. Open BurpSuite in Kali Linux by searching.



2. Create a temporary project and press next.



3. Use default settings for now in order start Burp.

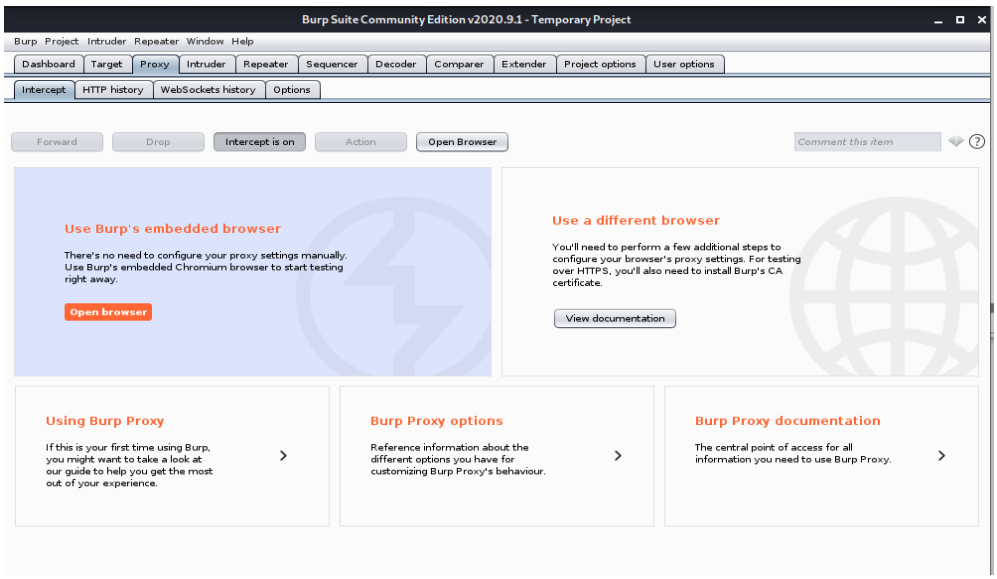


4. Dashboard will be appeared when we enter Burp. One can perform many tasks in burp easily. One of them is explained below.

Using Proxy:

- A proxy server acts as a gateway between you and the internet.
- We can use manual proxy from BurpSuite for browser too.
- We need to change the setting of proxy in browser too.
- Then when 'intercept is on', all the HTTP requests will only be forwarded if user permits them from BurpSuite.

SNAPSHOT:

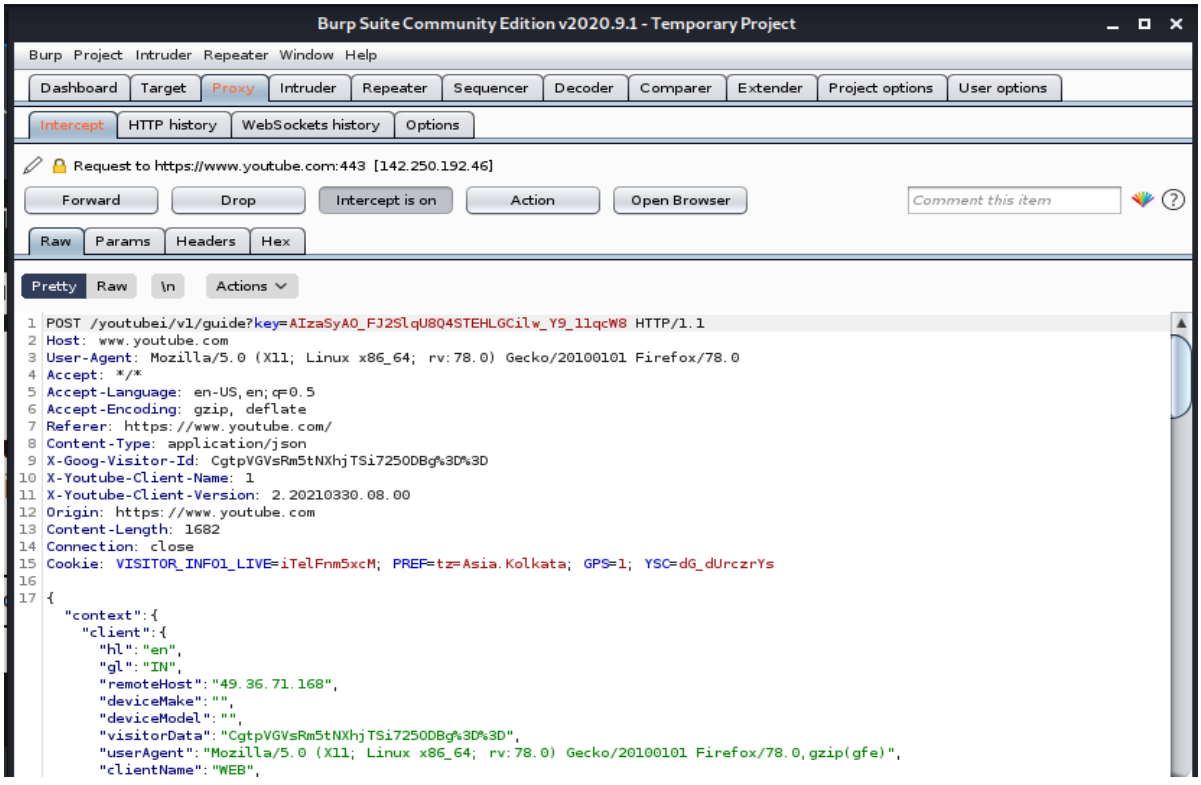


- If we try to open any HTTP website using our browser, it will show the request like below in BurpSuite.

SNAPSHOT:

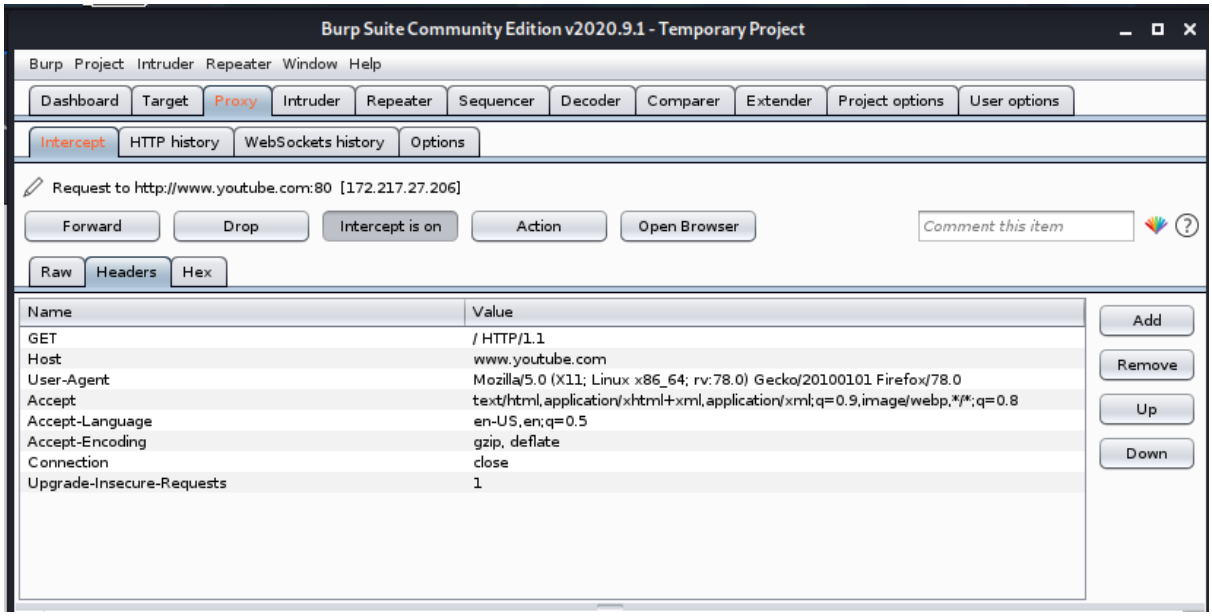
- We can forward them if we want or drop them.

SNAPSHOT:



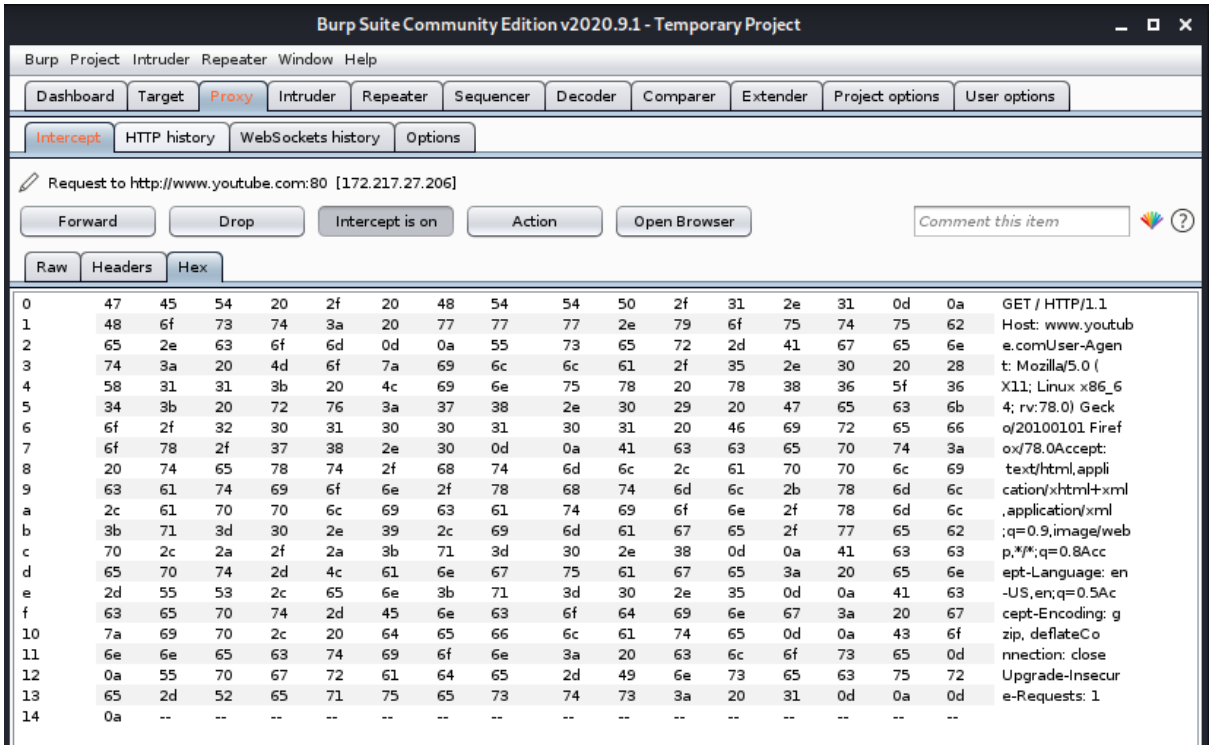
- We can look at the header of it by Header tab.

SNAPSHOT:



- We can see its Hex string in Hex tab.

SNAPSHOT:



CONCLUSION:

In this PRACTICAL -, we learned about BurpSuite tool and used its different services to perform different tasks from easy tasks like decoding and comparing to high level tasks like setting proxy.

PRACTICAL - 5

AIM:

Access remote terminal using SSH without password.

Experiments:

Firstly, there is no password less authentication

To check that,

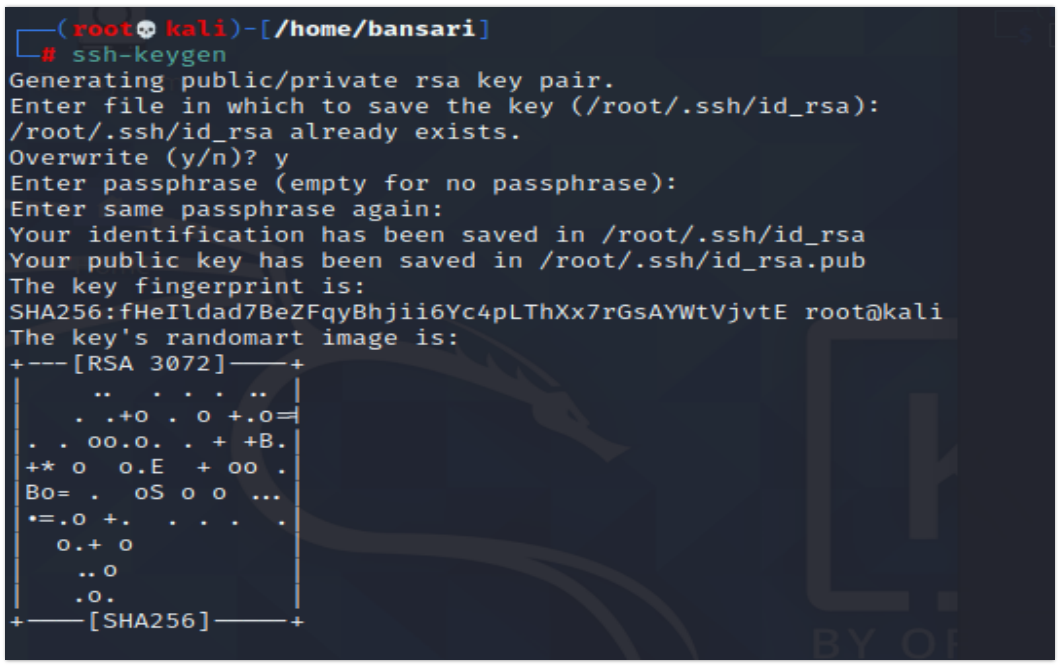
ssh root@ip_address_of_serverPC

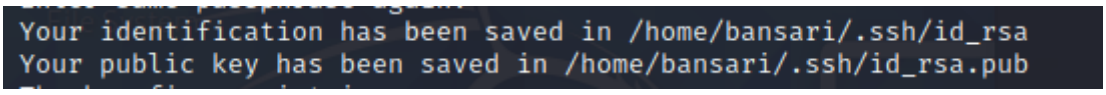
It will asked the password of server PC(2nd Linux Server)

Create a remote connection using openssh.

Step 1: Generate the Private and Public Keys in Linux machine 1.

Command is ssh-keygen





Step 2 : The keys are stored in /root/.ssh

Private Key : id_rsa

Public Key : id_rsa.pub

Step 3: Share the public key of remote machine (Linux server 2)

You can manually share the public key but we will do it via command:

ssh-copy-id –i /root/.ssh/id_rsa.pub "2nd machine username"

Hit enter

First time it will ask you the password

2nd time it won't ask the password

If you want to close the connection then type exit and hit enter

Step 4: Login to remote-host without entering the password

ssh "remote-host"

If you want to close the connection then type exit and hit enter

CONCLUSION:

In this PRACTICAL -, we learned Access remote terminal using SSH without password.

PRACTICAL - 6

AIM:

In computers, Footprinting is the process of accumulating data regarding a specific network environment, usually for the purpose of finding ways to intrude into the environment. Footprinting can reveal system vulnerabilities and improve the ease with which they can be exploited. Use the given approach to implement Footprinting: Gathering Target Information making use of following tools:

Dmitry – Deepmagic

UA Tester

Whatweb

Tool Introduction:

1. Dmitry - Deepmagic:
    - DMitry (Deepmagic Information Gathering Tool) is a UNIX/(GNU)Linux Command Line Application coded in C. DMitry has the ability to gather as much information as possible about a host.
    - Base functionality is able to gather possible subdomains, email addresses, uptime information, tcp port scan, whois lookups, and more.
    - The following is a list of the current features:
        i. An Open Source Project.
        ii. Perform an Internet Number whois lookup.
        iii. Retrieve possible uptime data, system and server data.
        iv. Perform a SubDomain search on a target host.
        v. Perform an E-Mail address search on a target host.
        vi. Perform a TCP Portscan on the host target.
        vii. A Modular program allowing user specified modules

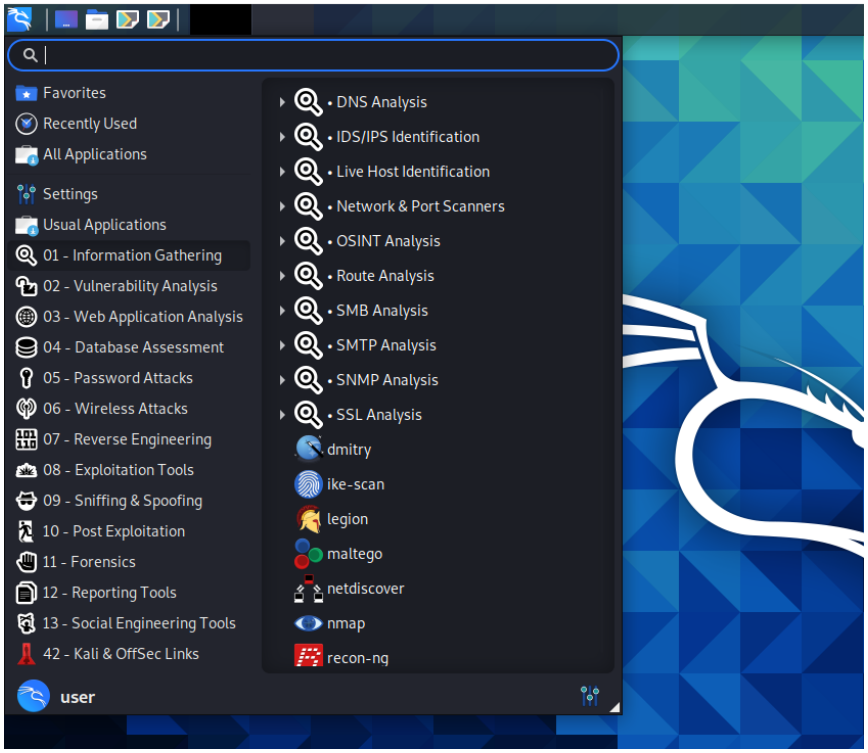
2. UA-Tester:
    - This tool is designed to automatically check a given URL using a list of standard and non-standard User Agent strings provided by the user (1 per line).
    - The results of these checks are then reported to the user for further manual analysis where required. Gathered data includes Response Codes, resulting URL in the case of a 30x response,
    - MD5 and length of response body, and select Server headers.
    - Results: When in non-verbose mode, only values that do not match the initial reference connection are reported to the user.
    - If no results are shown for a specific user agent then all results match the initial reference connection.
    - If you require a full output of all checks regardless of matches to the reference, please use the verbose setting.

3. Whatweb:

- WhatWeb identifies websites.
- Its goal is to answer the question, “What is that Website?”.
- WhatWeb recognizes web technologies including content management systems (CMS), blogging platforms, statistic/analytics packages, JavaScript libraries, web servers, and embedded devices. WhatWeb has over 1700 plugins, each to recognise something different.
- WhatWeb also identifies version numbers, email addresses, account IDs, web framework modules, SQL errors, and more.
- WhatWeb can be stealthy and fast, or thorough but slow.
- WhatWeb supports an aggression level to control the trade off between speed and reliability.
- When you visit a website in your browser, the transaction includes many hints of what web technologies are powering that website.
- Sometimes a single webpage visit contains enough information to identify a website but when it does not, WhatWeb can interrogate the website further.
- The default level of aggression, called ‘stealthy’, is the fastest and requires only one HTTP request of a website.
- This is suitable for scanning public websites. More aggressive modes were developed for use in penetration tests.

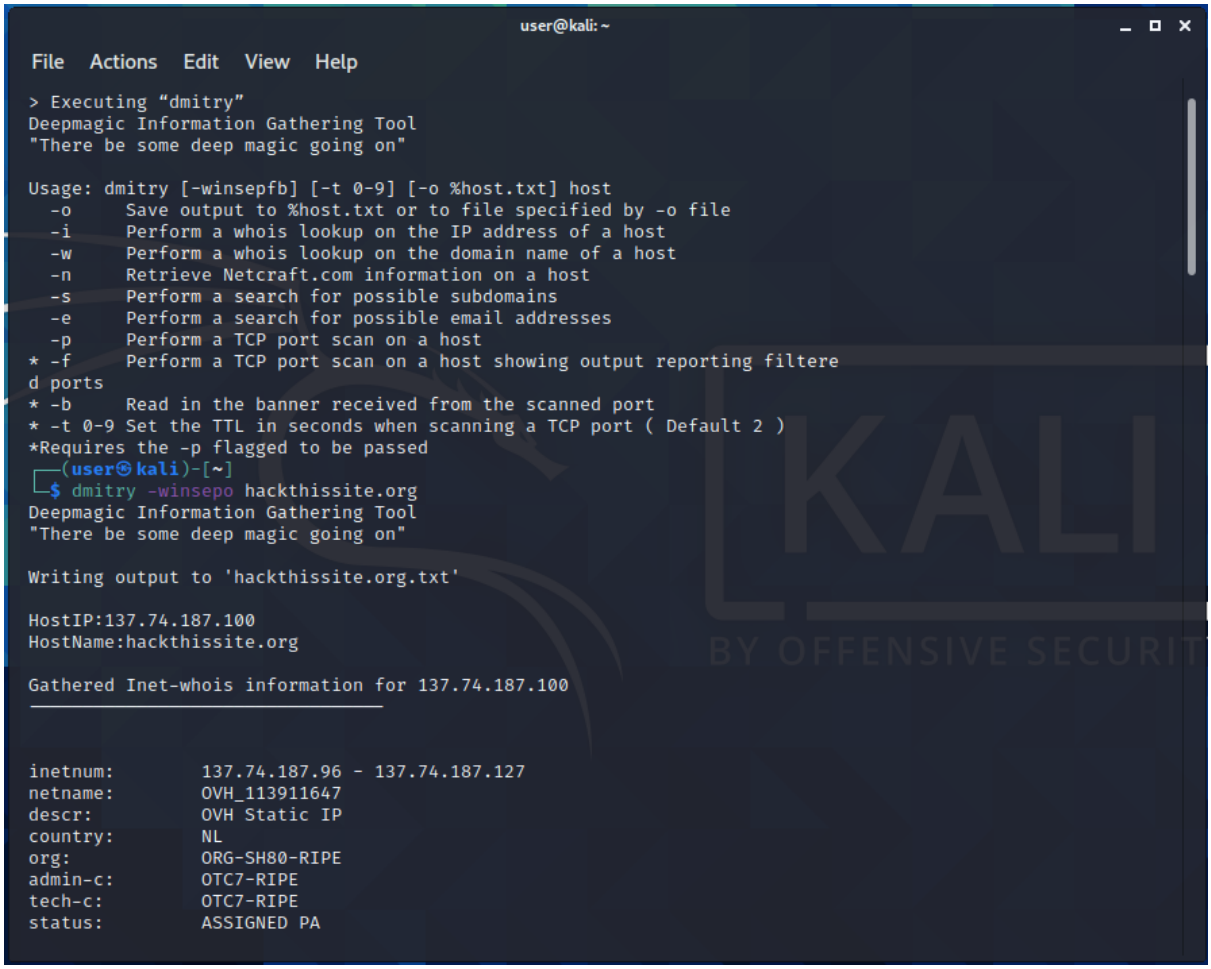
Dmitry - Deepmagic:

We can find Dmitry in Information Gathering section.

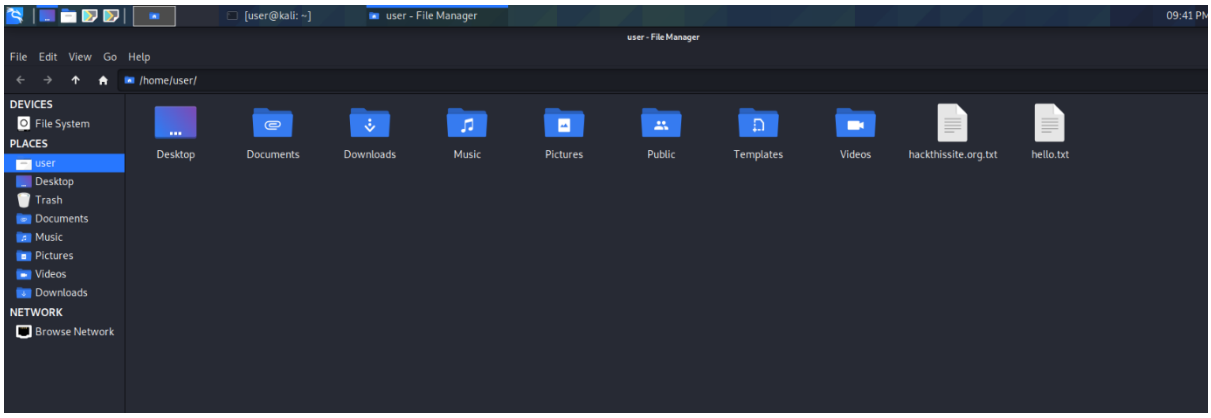


Command: Dmitry -winsepo hackthissite.org

We can use any website as target.



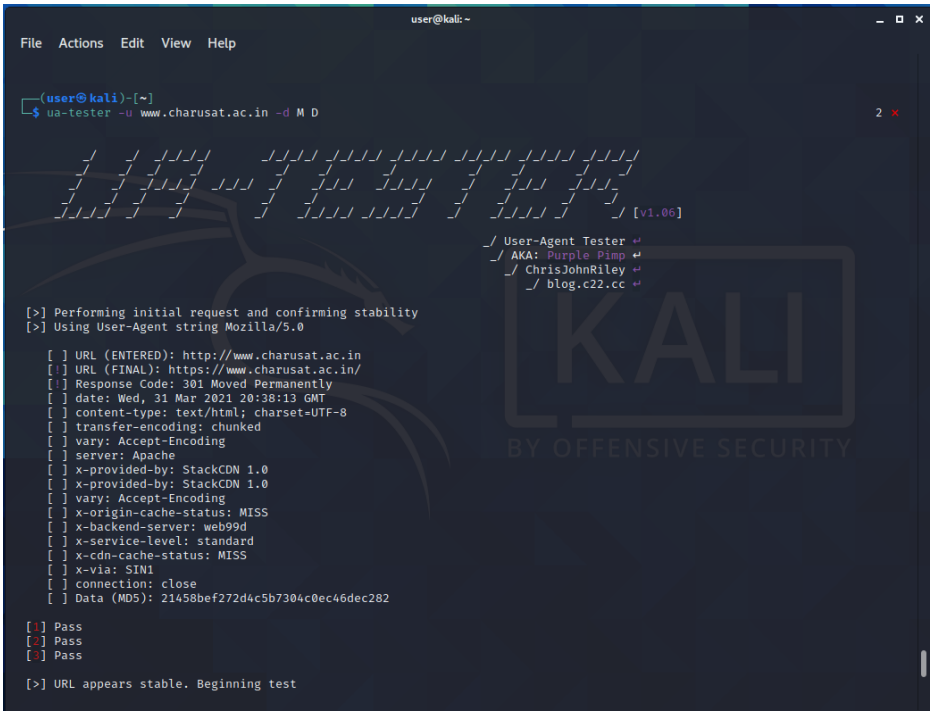
The output is stored in "hackthissite.org.txt" file.



UA Tester:

We can use UA tester directly from terminal.

Command: ua-tester -u www.charusat.ac.in -d M D

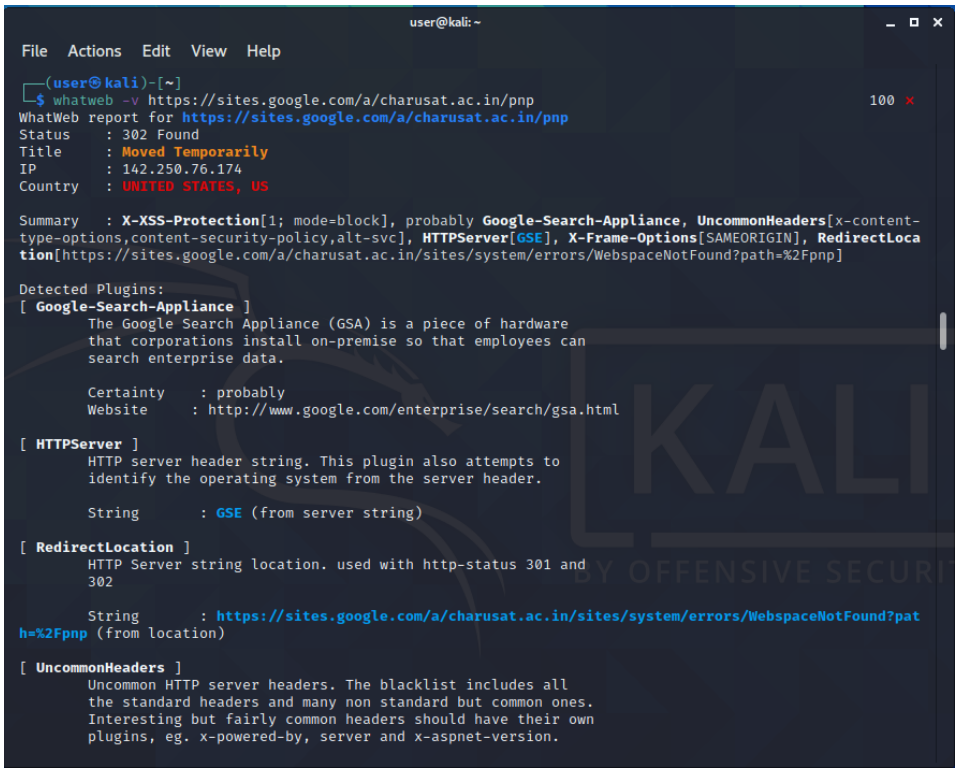


Whatweb:

We can also use Whatweb directly from terminal.

Command: whatweb -v https://sites.google.com/charusat.ac.in/pnp

We can use any website we want as target.



CONCLUSION:

In this PRACTICAL -, we learned tools like Dmitry, UA-tester and Whatweb for information gathering.

PRACTICAL - 7

AIM:

The transmission of information needs to be secure over the communication channel and the data has to be confidential. Study and implement the PRACTICAL - approach for Steganography.

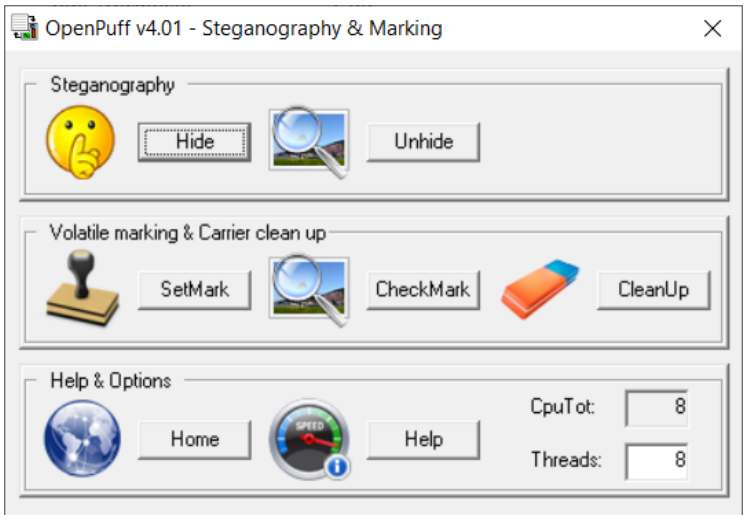
-Using DOS commands
-Using OpenPuff Tool

TOOL INTRODUCTION:

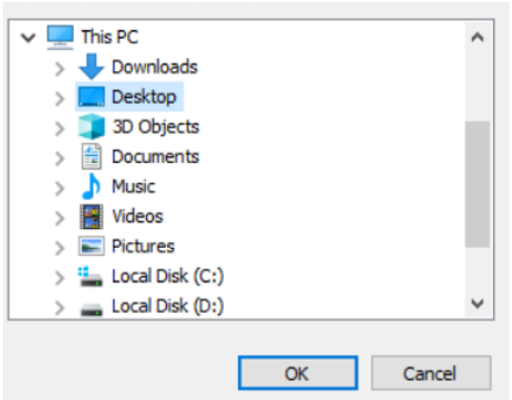
4. OpenPuff –
    - OpenPuff Steganography and Watermarking, sometimes abbreviated OpenPuff or Puff, is a free steganography tool for Microsoft Windows created by Cosimo Oliboni and still maintained as independent software. The program is notable for being the first steganography tool (version 1.01 released on December 2004) that:
        i. let's users hide data in more than a single carrier file. When hidden data are split among a set of carrier files you get a carrier chain, with no enforced hidden data theoretical size limit (256MB, 512MB, ... depending only on the implementation)
        ii. implements 3 layers of hidden data obfuscation (cryptography, whitening and encoding)
        iii. extends deniable cryptography into deniable steganography
    - Last revision supports a wide range of carrier formats
        i. Images Bmp, Jpg, Png, Tga
        ii. Audios Aiff, Mp3, Wav
        iii. Videos 3gp, Mp4, Mpeg I, Mpeg II, Vob
        iv. Flash-Adobe Flv, Pdf, Swf
    - OpenPuff is used primarily for anonymous asynchronous data sharing:
        i. the sender hides a hidden stream inside some public available carrier files (password + carrier files + carrier order is the secret key)
        ii. the receiver unhides the hidden stream knowing the secret key
    - The advantage of steganography, over cryptography alone, is that messages do not attract attention to themselves. Plainly visible encrypted messages — no matter how unbreakable — will arouse suspicion, and may in themselves be incriminating in countries where encryption is illegal. Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties.
    - Watermarking is the action of signing a file with an ID or copyright mark. OpenPuff does it in an invisible steganographic way, applied to any supported carrier. The invisible mark, being not password protected, is accessible by everyone (using the program).

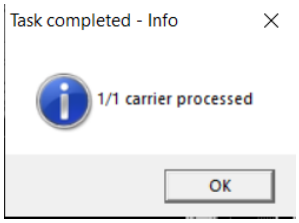
Using OpenPuff:

- OpenPuff provides more services that command prompt.
- Instead of using commands, we can directly use GUI in OpenPuff.

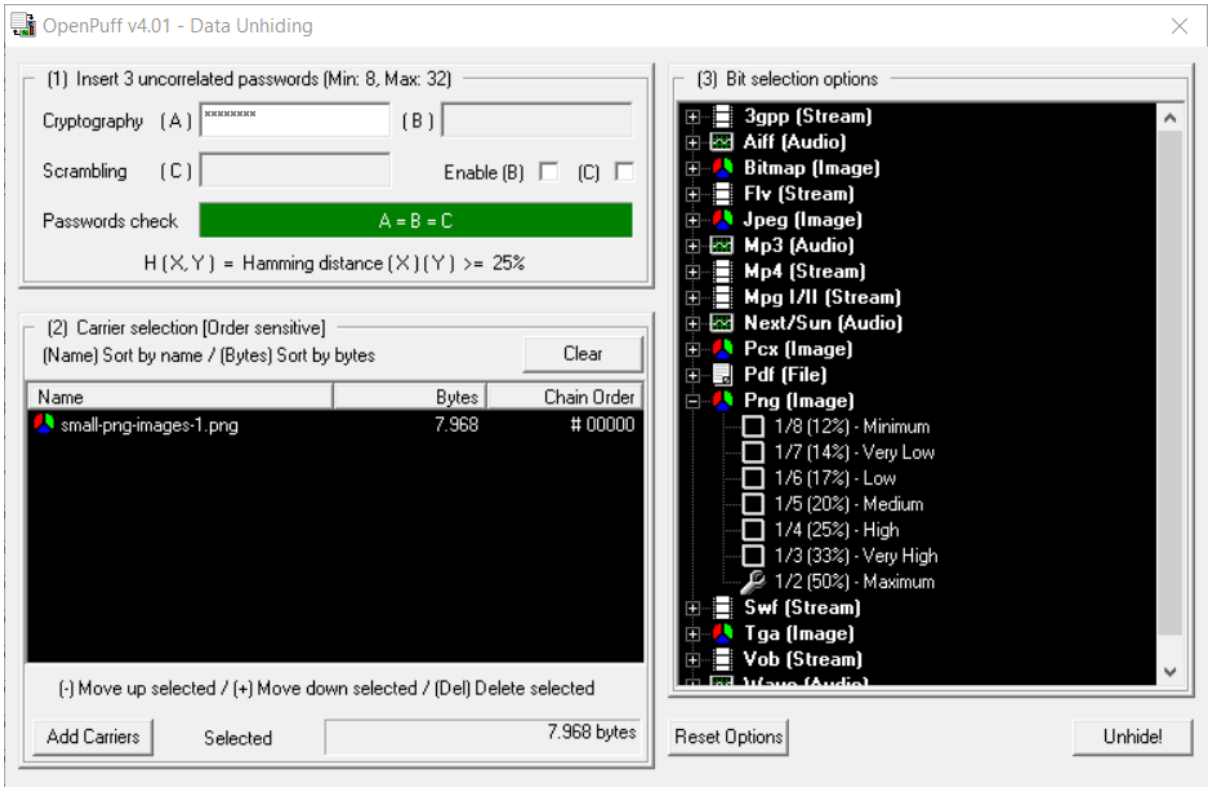


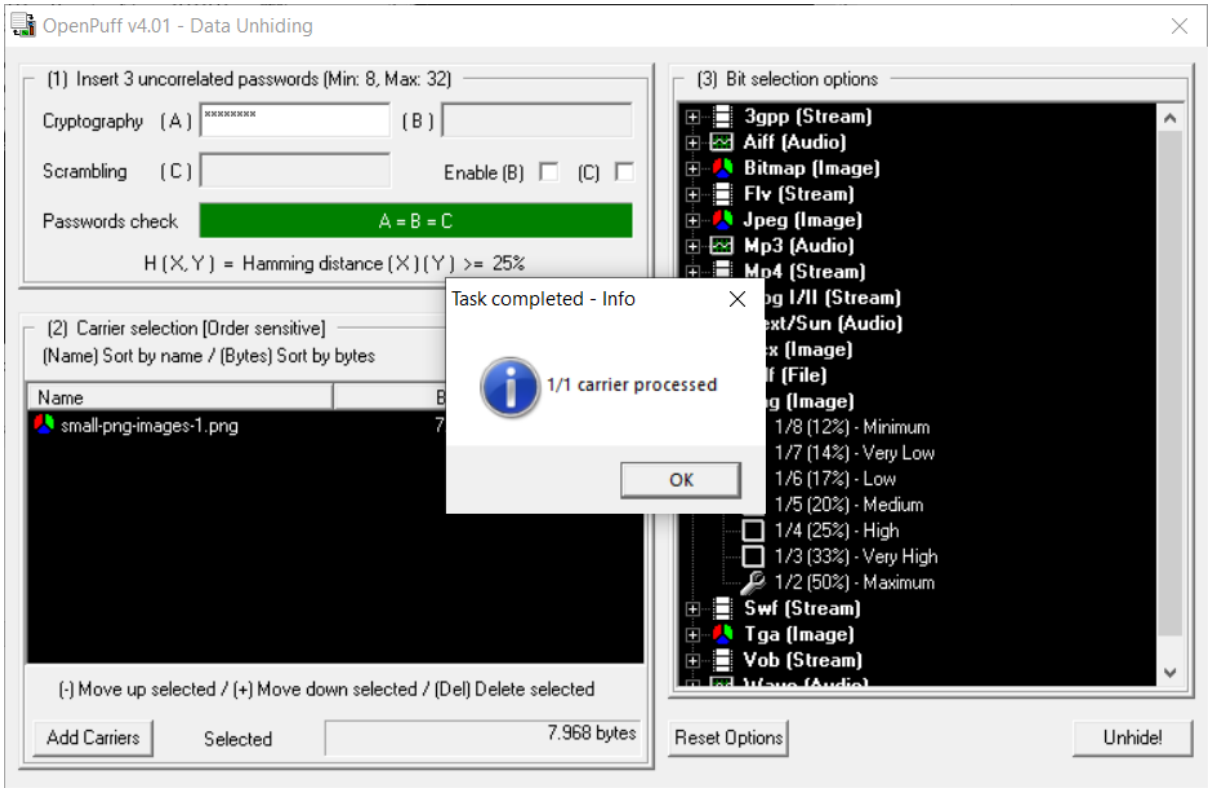
- We can select Hide option to hide our text file into image file.
- We can set password and give image and text files as input.
- On pressing “Hide Data!” we can select the location to save our output.



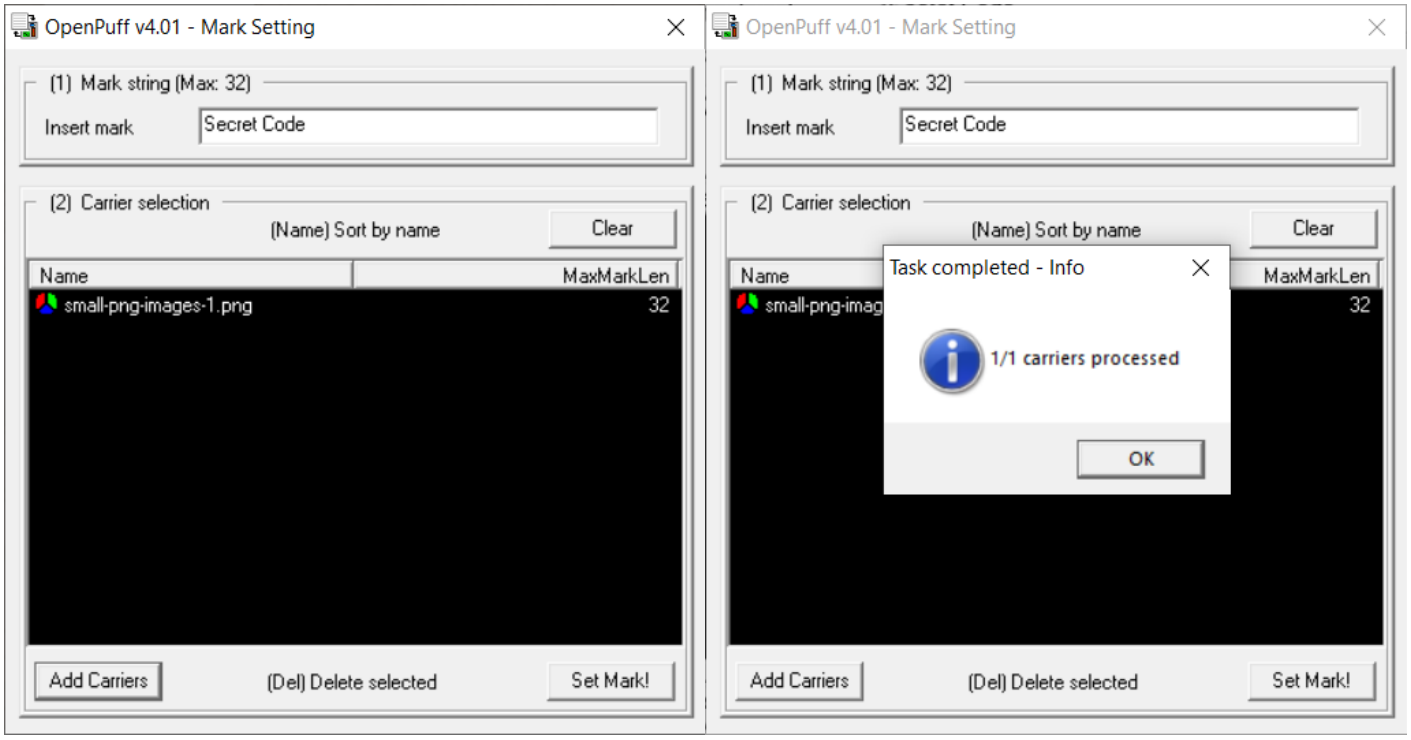


- We can recover the secret message by giving the image and password we used earlier.

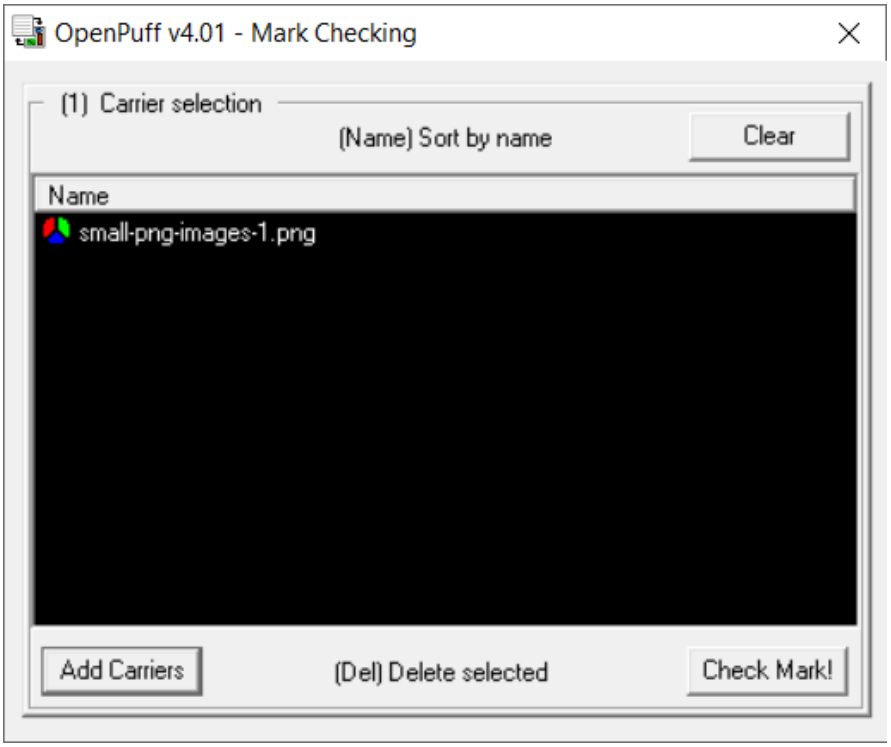


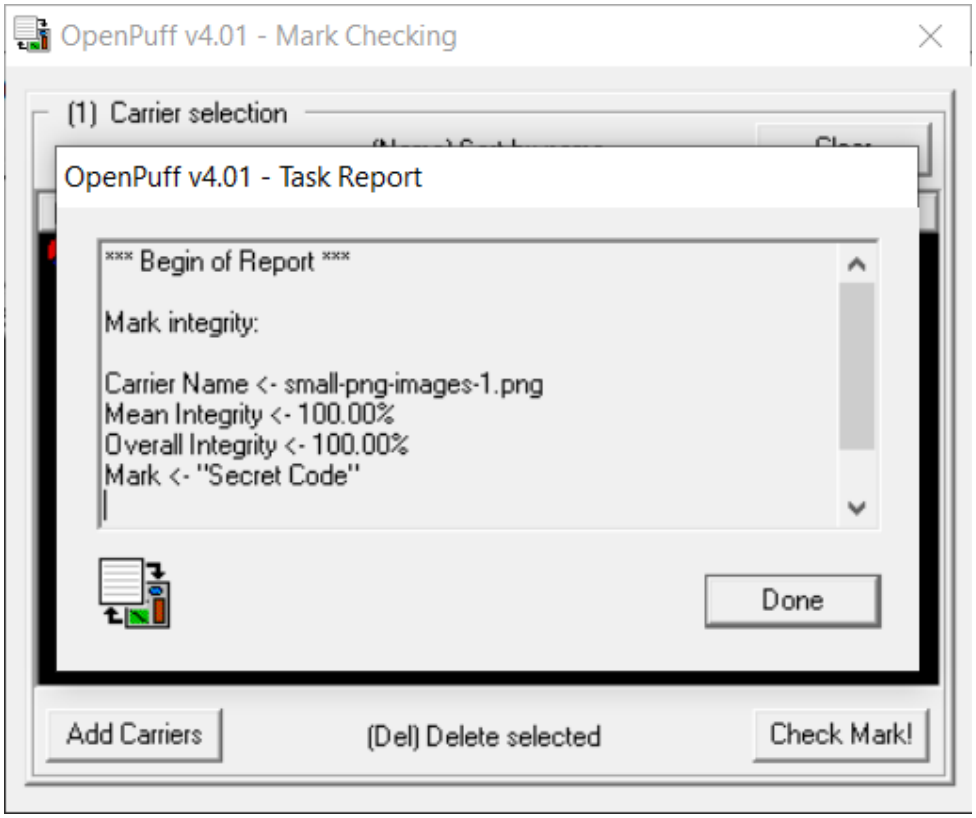


- OpenPuff can also be used for Watermarking.
- We can easily create watermarking using “Set Mark” option.

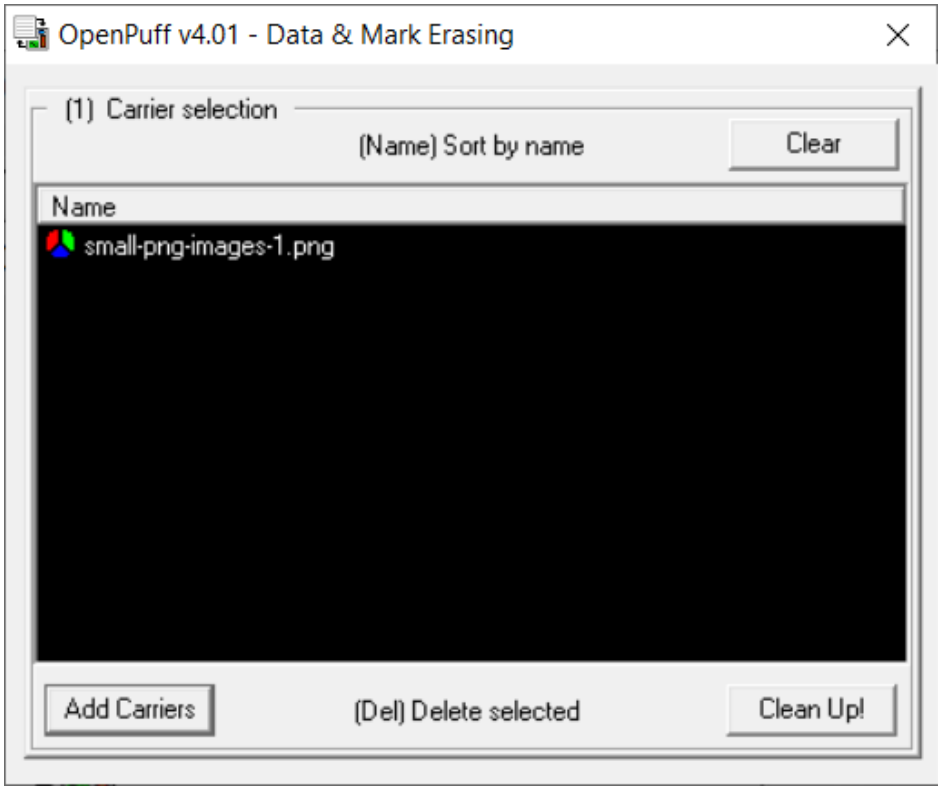


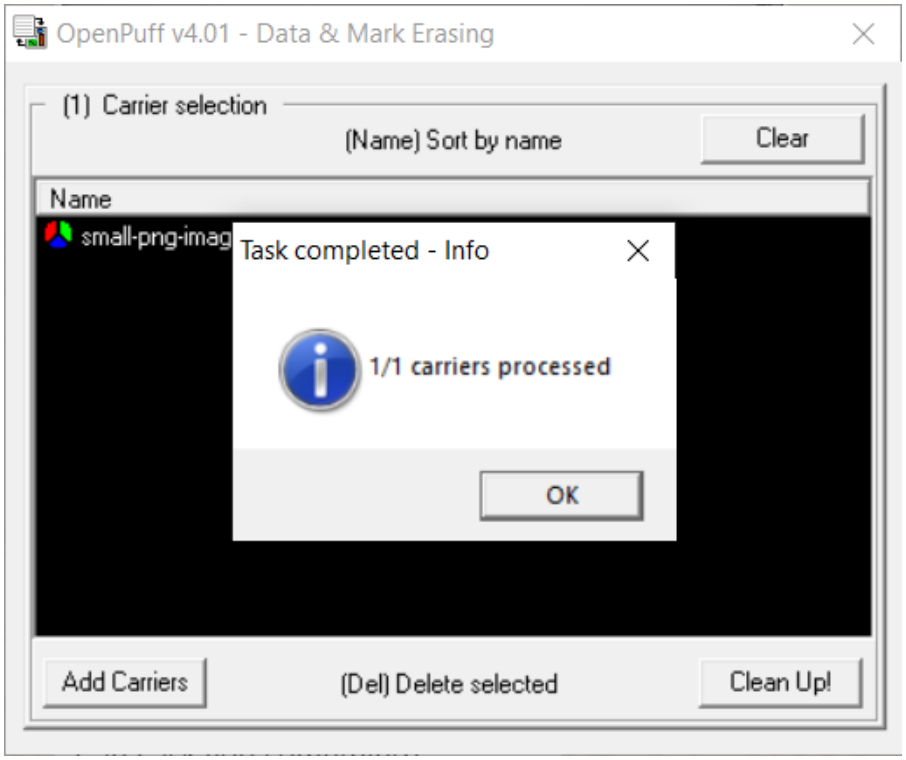
- We can check if any image contains watermark using “Checkmark” option.





- We can permanently remove watermark using “Cleanup” option.





Using DOS Command:

- Easy way to use steganography is using DOS command.
- We can hide a text message into any image using following command.

• Command: echo secret message>image
• Ex: echo A secret message by 18DCS118>secret.jpg

- We can get the secret message back using notepad.

• Command: notepad image
• Ex: notepad secret.jpg

```
D:\>echo A secret message by 18DCS118>secret.jpg

D:\>notepad secret.jpg
```

Similar Tools as OpenPuff:

- **ImageSpyer G2:**
  - The utility allows you to hide information in images using techniques of modern steganography, and the data will be encrypted by two-layer

cryptographic protection, which including more than twenty block ciphers (for data protection) and powerful stream cipher to encrypt stego.
    - Mechanisms used in the program, aimed at not only to hide existence of the data, but also guaranteed to protect it from possible attacks.
    - Through the use of a powerful compression algorithm, the available capacity is significantly increased.
    - Besides powerful cryptographic security is provided by a number of privacy settings, without knowing that an attacker cannot determine the presence of hidden information.
- MP3Stego:
    - MP3Stego is a typical steganographic tool for MP3 audio, which embeds secret message into MP3 audio according to the parity of the block length. In this paper, a detection method for MP3Stego based on the differential statistics of quantization step is present.
    - By analyzing the algorithm of MP3Stego, we find that the quantization step which is an important parameter of MP3 encoding is affected during embedding.
    - The standard deviation of the second-order differential sequence from quantization step is adopted as the classification feature.
    - Experimental results indicate that the proposed algorithm is effective in detecting MP3Stego and can achieve better detection performance than the existing algorithms.
- S-Tools:
    - S-Tools is an extremely simple tool that can be used to hide text files in images.
    - The tool is present in zip format, just extract it and create a simple text file which has the data you want to hide.
    - Save a cover photo in which the data will be hidden. Just run the .exe file and drag the BMP file to that window.
    - Then drag the text file over the image and select the encryption algorithm to embed the file.
    - The options available are IDEA, DES, triple DES etc. Select a passphrase that will be used to reveal the hidden files from the images.
    - Just use the same process to reveal the contents that have been hidden in the file.
- StegoShare:
    - StegoShare is a steganography tool that allows embedding of large files into multiple images.
    - It may be used for anonymous file sharing. This software can be easily used for anonymous file sharing.
    - An uploader downloads legal images from a public photo hosting site, and embeds the censored file into those images.
    - The uploader then uploads pictures to the public photo torrent tracker and puts the links referencing the stego pictures with censored file's description on a forum or blog. Downloaders, seeders, and public photo trackers, if caught distributing illegal files, are protected from legal prosecution, because they can

always use plausible deniability, saying that they knew nothing about the illicit file in the images.
    - This is impossible to prove otherwise, as the human eye cannot differentiate between an ordinary image and a picture with hidden embedded file.
- BMPSecrets:
    - HexaStego–BMP is a very small steganography tool (only 10KB in size) for hiding a message in unrelated information.
    - It can hide relatively small files (around 50KB) within .bmp graphic file. The image is slightly altered throughout the process, and this change is only noticeable if the modified and original files are viewed simultaneously.
    - The hidden file can be extracted using the same utility.

CONCLUSION:

In this PRACTICAL -, we learned about steganography and watermarking. We implemented them in real image using command prompt and OpenPuff. We also got to know about different tools as OpenPuff.

PRACTICAL - 8

AIM: Using OWASP-ZAP tool to find out Web Application Vulnerability.

TOOL INTRODUCTION:

1. Owasp zap:

Software security testing is the process of assessing and testing a system to discover security risks and vulnerabilities of the system and its data. There is no universal terminology but for our purposes, we define assessments as the analysis and discovery of vulnerabilities without attempting to actually exploit those vulnerabilities. We define testing as the discovery and attempted exploitation of vulnerabilities.

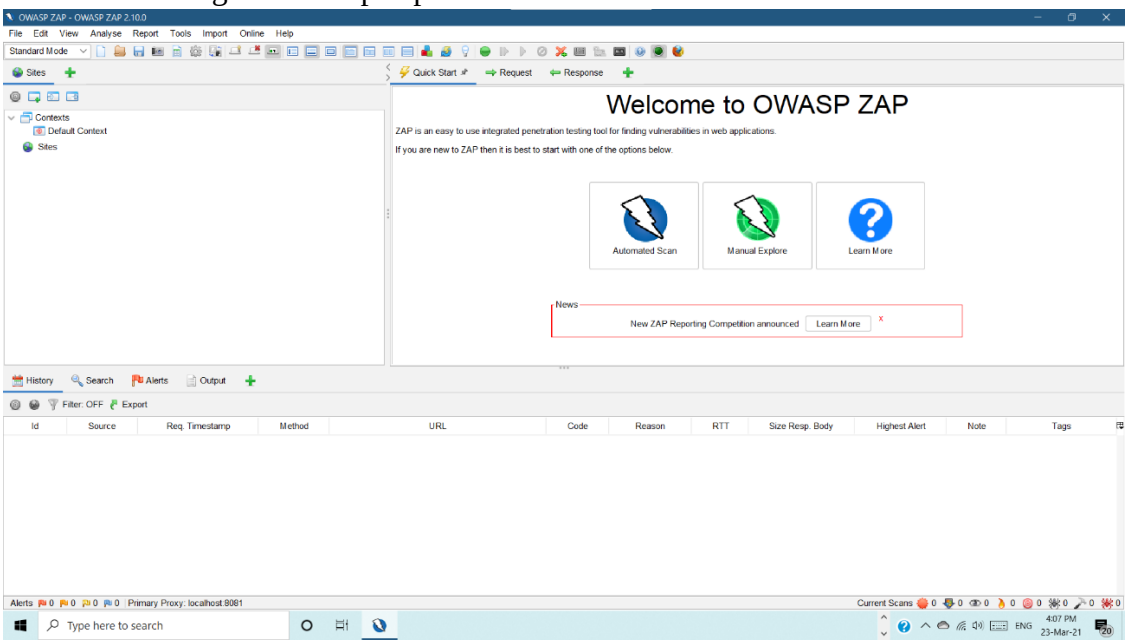
Security testing is often broken out, somewhat arbitrarily, according to either the type of vulnerability being tested or the type of testing being done. A common breakout is:

- **Vulnerability Assessment** – The system is scanned and analyzed for security issues.
- **Penetration Testing** – The system undergoes analysis and attack from simulated malicious attackers.
- **Runtime Testing** – The system undergoes analysis and security testing from an end-user.
- **Code Review** – The system code undergoes a detailed review and analysis looking specifically for security vulnerabilities.

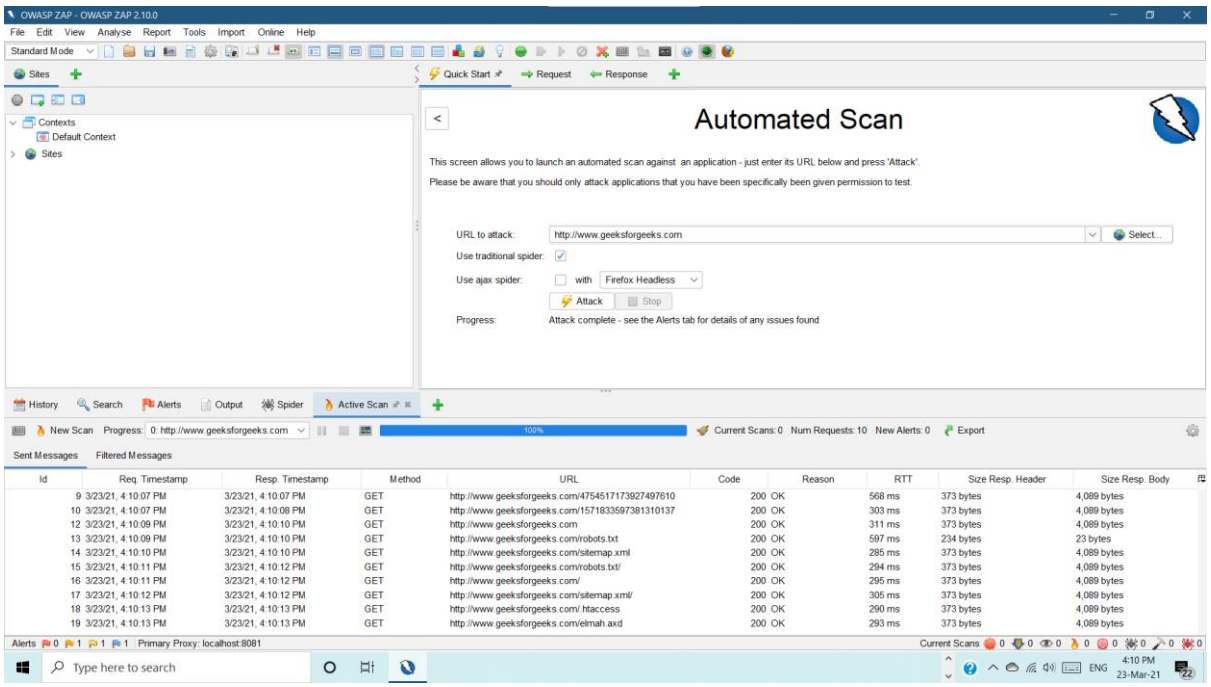
Note that risk assessment, which is commonly listed as part of security testing, is not included in this list. That is because a risk assessment is not actually a test but rather the analysis of the perceived severity of different risks (software security, personnel security, hardware security, etc.) and any mitigation steps for those risks.

Using Owasp zap:

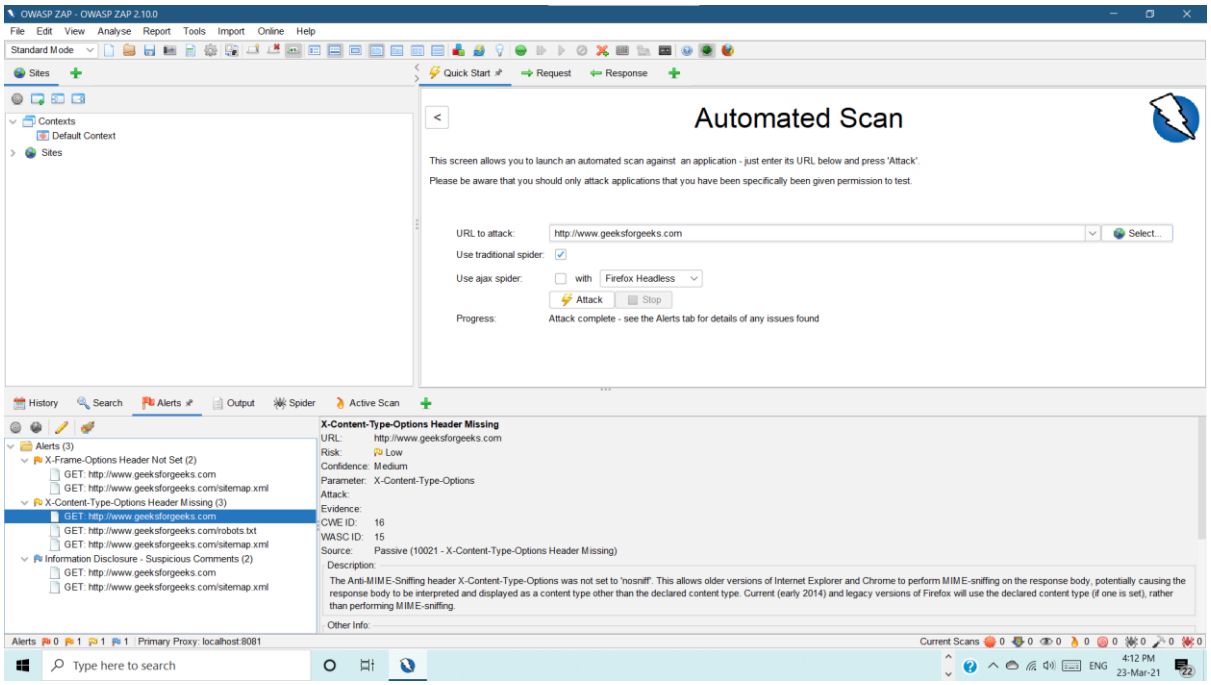
- The gui of owasp zap



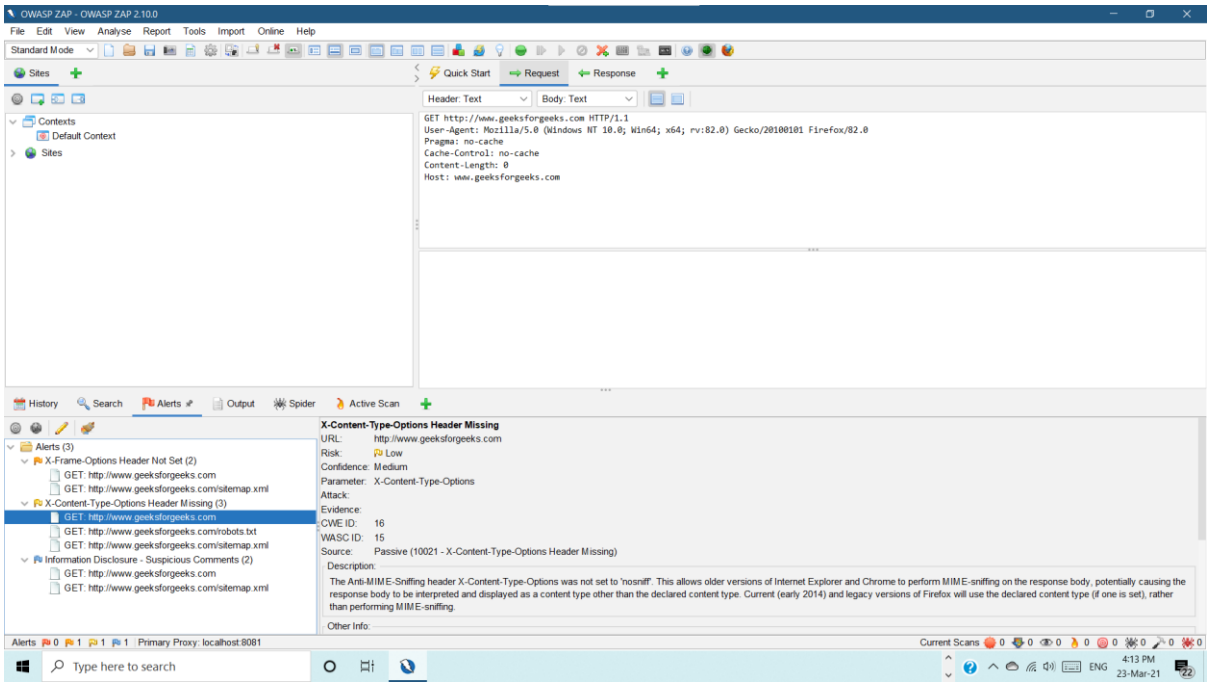
- The automatd scan done on geeksforgeeks website.



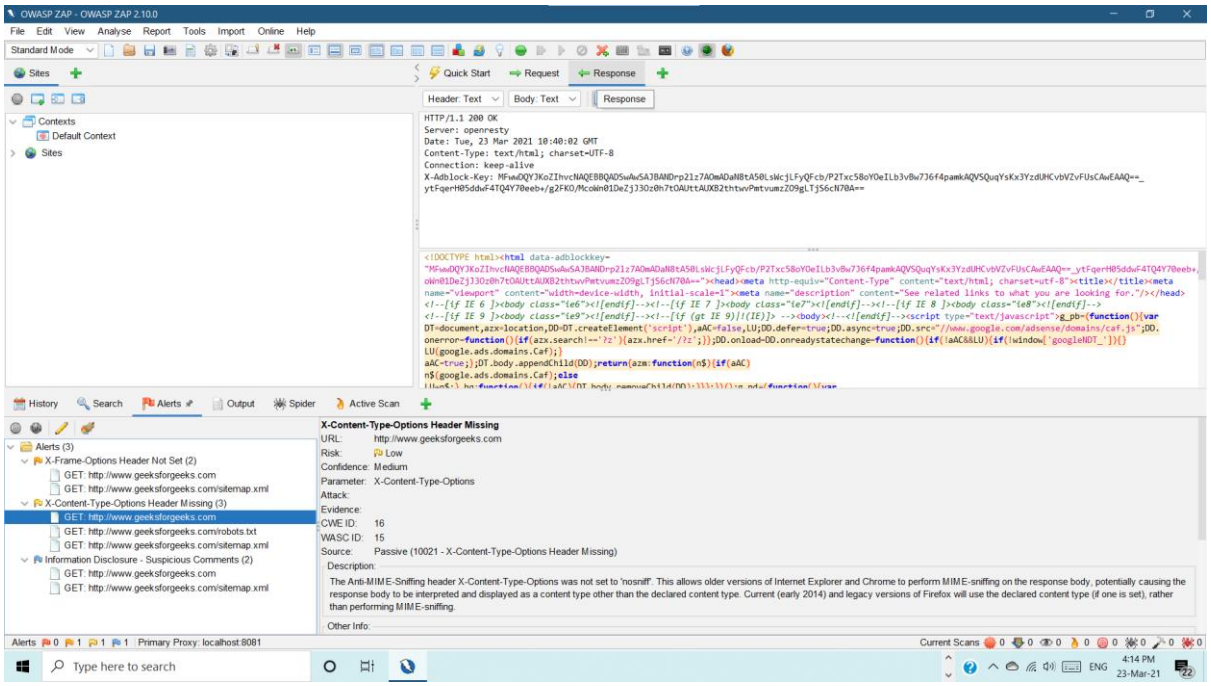
- The risks detected by owasp :



- Request :



- Response:



CONCLUSION:

In this PRACTICAL -, we learned about owasp zap tool which detects Web Application Vulnerability
And also provides solution to resolve them.

PRACTICAL - 9

AIM: Implement GPG for windows and Linux.

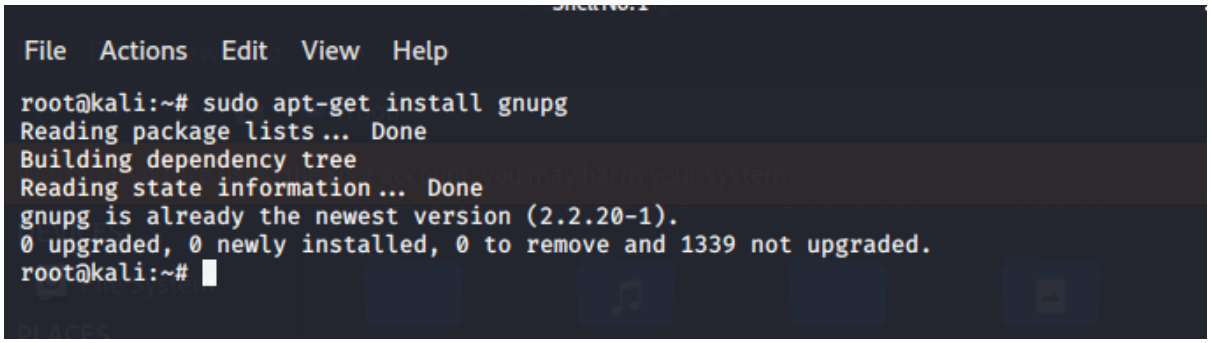
TOOL INTRODUCTION:

1. Gpg linux:

This software is used to encrypt and decrypt the file.

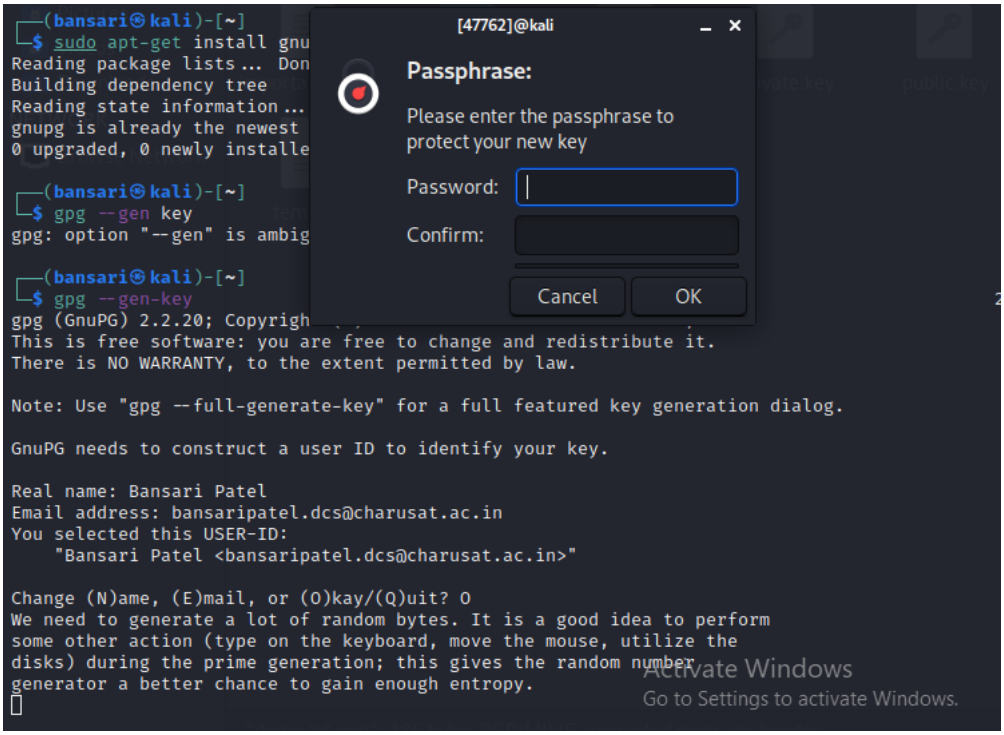
First of all to install the gpg software type this command:-

```
sudo apt-get install gnupg
```



now create a key using command:-

```
gpg –gen-key
```



```
┌──(bansari㉿kali)-[~]
└─$ gpg --gen-key
gpg (GnuPG) 2.2.20; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Bansari Patel
Email address: bansaripatel.dcs@charusat.ac.in
You selected this USER-ID:
    "Bansari Patel <bansaripatel.dcs@charusat.ac.in>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 1AEB8B1B8ABAD926 marked as ultimately trusted
gpg: revocation certificate stored as '/home/bansari/.gnupg/openpgp-revocs.d/31B13900E9E4E12DDB4
165C51AEB8B1B8ABAD926.rev'
public and secret key created and signed.

pub   rsa3072 2021-03-11 [SC] [expires: 2023-03-11]
      31B13900E9E4E12DDB4165C51AEB8B1B8ABAD926
uid                      Bansari Patel <bansaripatel.dcs@charusat.ac.in>
sub   rsa3072 2021-03-11 [E] [expires: 2023-03-11]

┌──(bansari㉿kali)-[~]
└─$
```


now export public key using command:-

gpg –export –a “User Name” > Public.key

```
┌──(bansari㉿kali)-[~]
└─$ gpg --export -a Bansari Patel > Public.key

┌──(bansari㉿kali)-[~]
└─$ ls
demo.txt      Documents          index.html  Pictures     public.key  temp.txt
demo.txt.gpg  Downloads          INS.txt     private.key  Public.key  Videos
Desktop       important.txt.txt  Music       Public       Templates

┌──(bansari㉿kali)-[~]
└─$
```


Now export private key using command:-

gpg –export-secret-key –a “User Name” > Private.key

```
┌──(bansari㉿kali)-[~]
└─$ gpg --export-secret-key -a Bansari Patel > Private.key

┌──(bansari㉿kali)-[~]
└─$ ls
demo.txt  demo.txt.gpg  Desktop  Documents  Downloads  important.txt.txt  index.html  INS.txt  Music  Pictures  private.key  Private.key  Public  public.key  Public.key  Templates  temp.txt  Videos

┌──(bansari㉿kali)-[~]
└─$
```

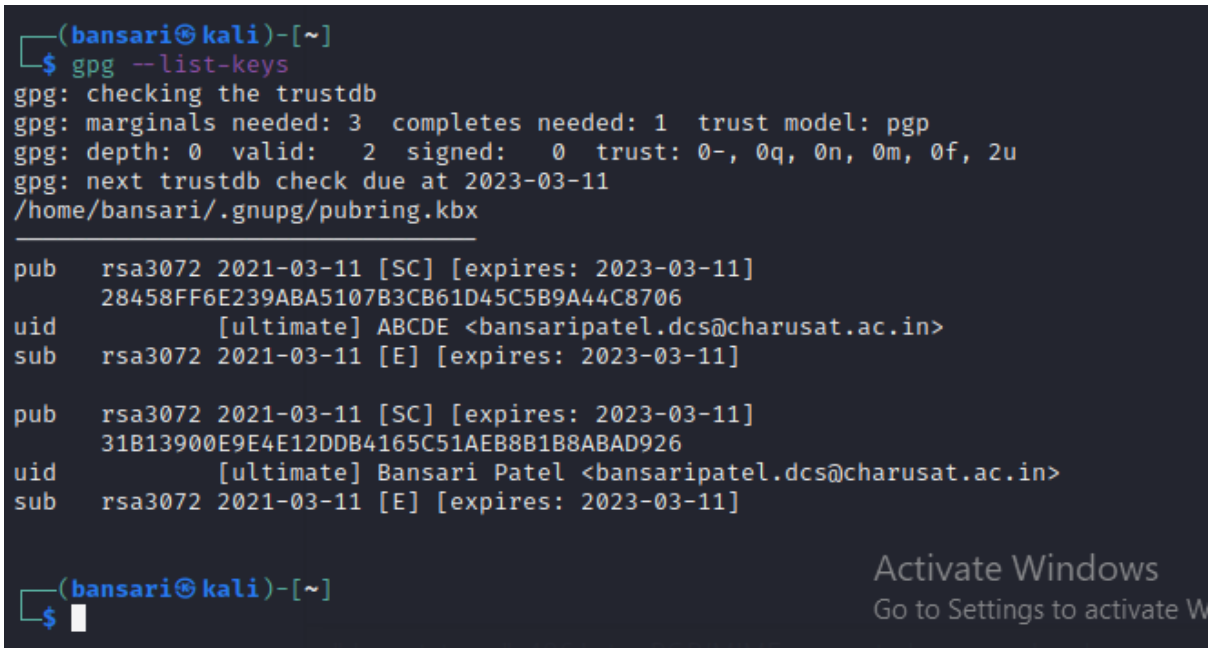
The same way importing public and private key using following commands:-

gpg - -import Public.key

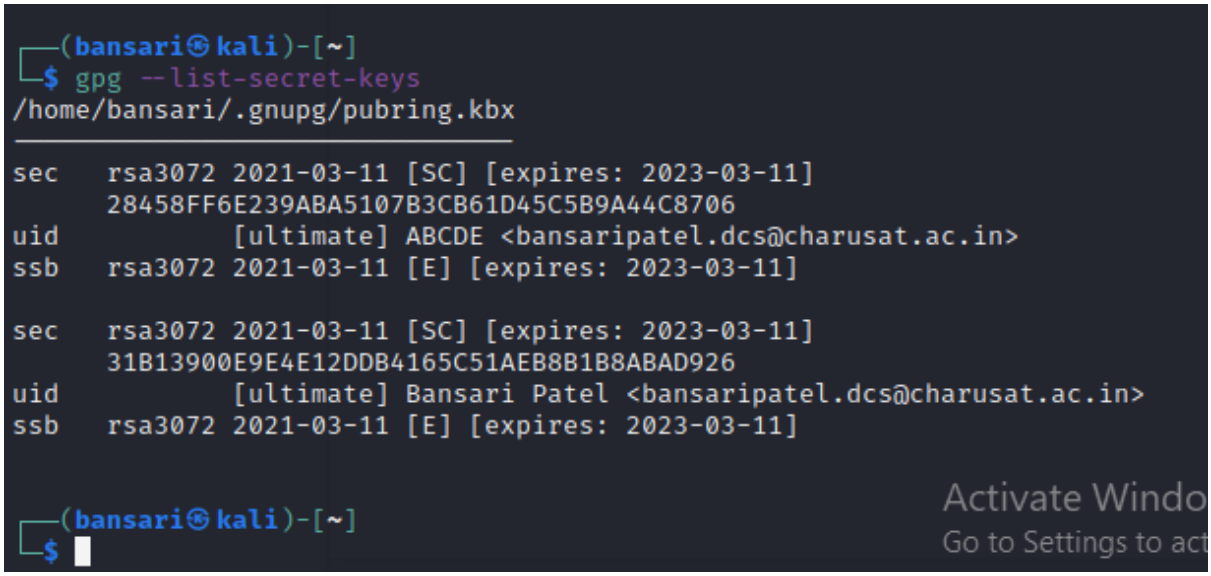
gpg - -allow-secret-key-import - - import Private.key

listing public and private keys using commands:-

gpg - -list-keys



gpg –list-secret-keys

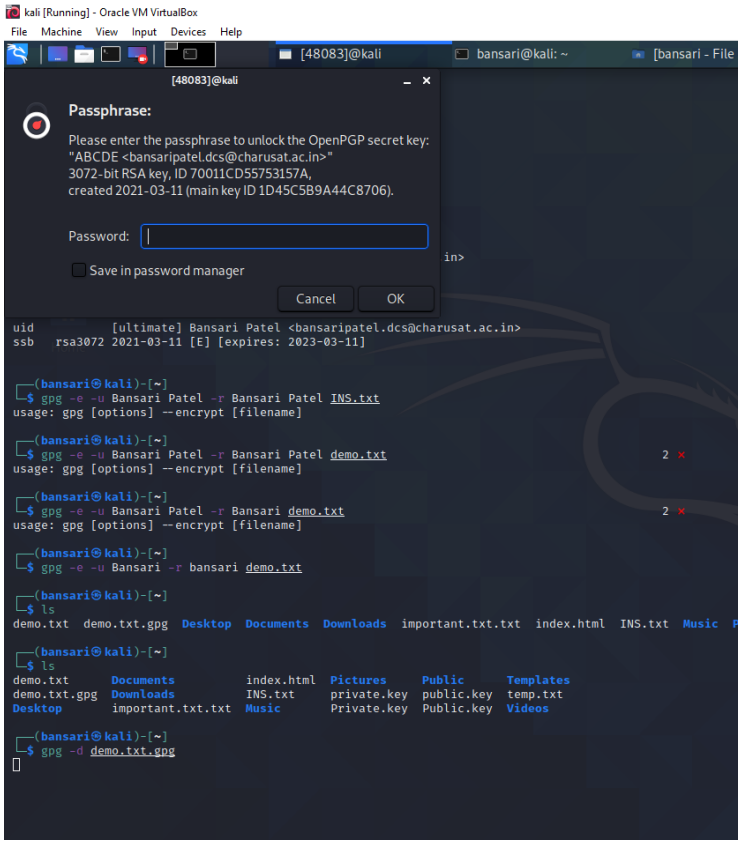


Now incrypting file name demo.txt using command.

gpg –e –u “Sender User Name” –r “Receiver User Name” file name

```
┌──(bansari㉿kali)-[~]
└─$ gpg -e -u Bansari -r bansari demo.txt

┌──(bansari㉿kali)-[~]
└─$ ls
demo.txt  demo.txt.gpg  Desktop  Documents  Downloads  important.txt.txt  index.html  INS.txt  M

┌──(bansari㉿kali)-[~]
└─$ ls
demo.txt       Documents          index.html  Pictures     Public      Templates
demo.txt.gpg   Downloads          INS.txt     private.key  public.key  temp.txt
Desktop        important.txt.txt  Music       Private.key  Public.key  Videos

┌──(bansari㉿kali)-[~]
└─$
```


Now decrypting file using command:-

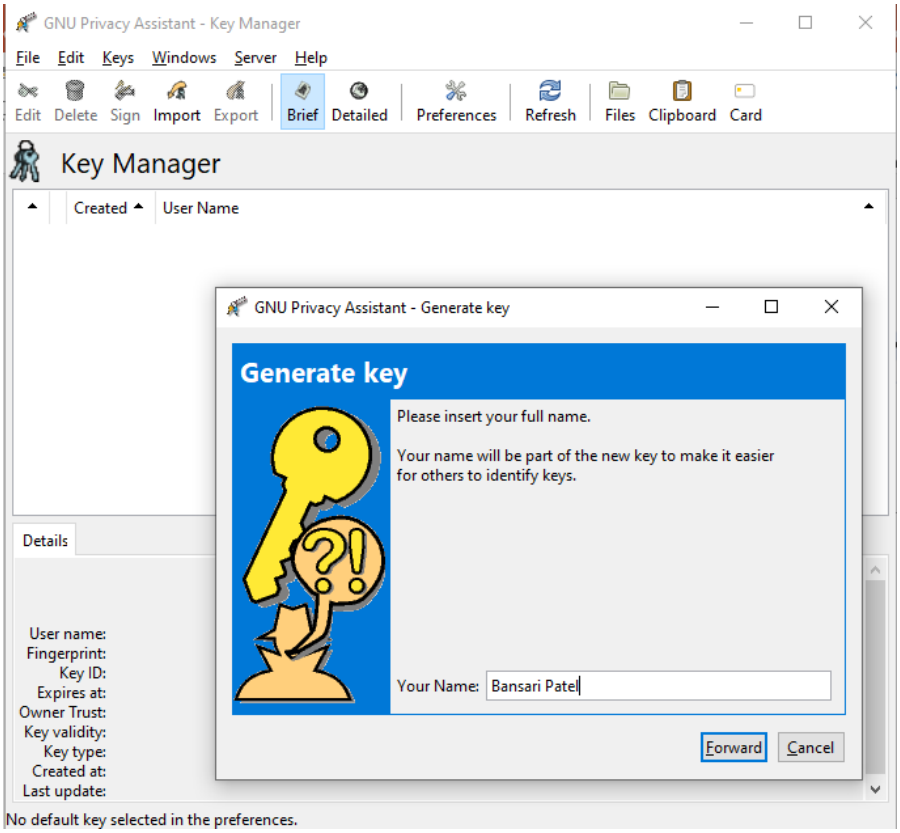


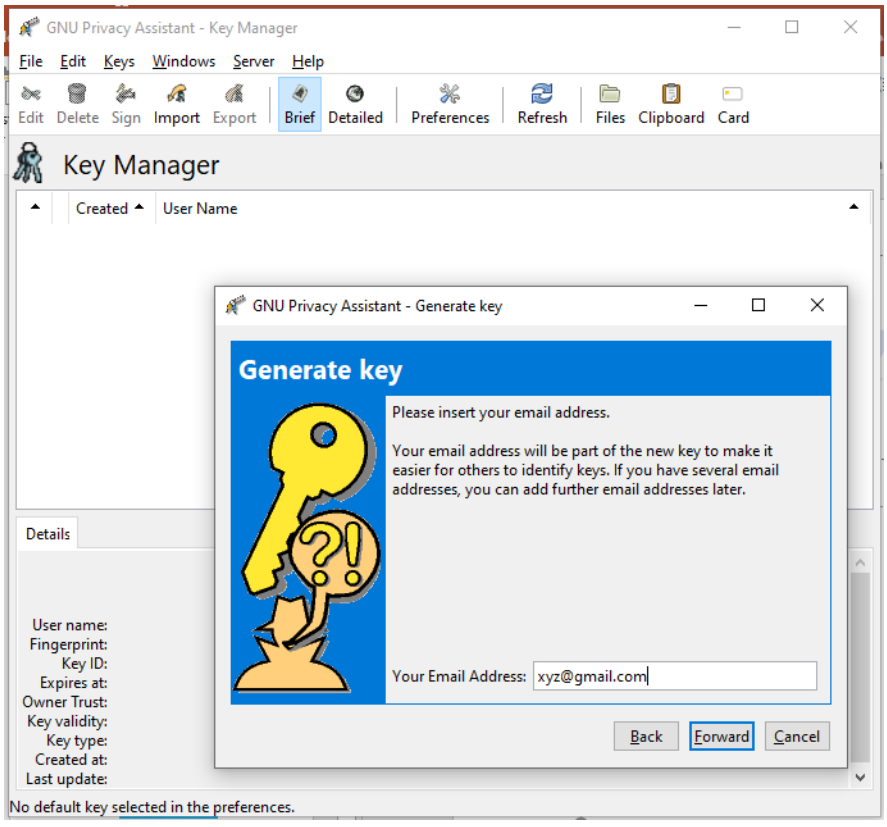

```
┌──(bansari㉿kali)-[~]
└─$ gpg -d demo.txt.gpg
gpg: encrypted with 3072-bit RSA key, ID 70011CD55753157A, created 2021-03-11
      "ABCDE <bansaripatel.dcs@charusat.ac.in>"
This
is
a
demo
file

┌──(bansari㉿kali)-[~]
└─$
```

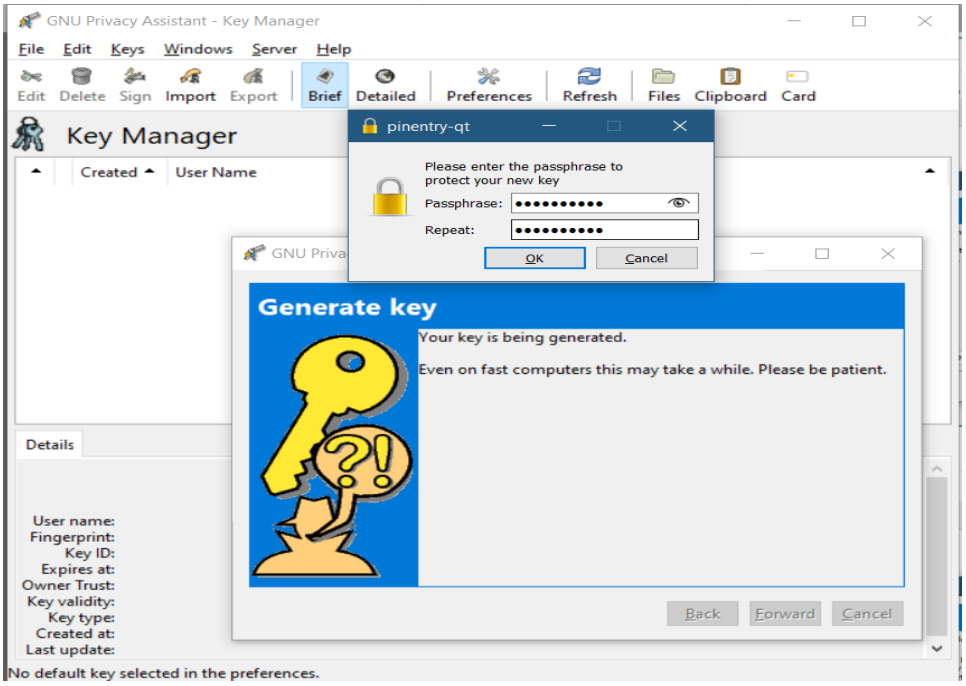
2. Gpg windows:

Install this software for windows from site www.gpg4win.org

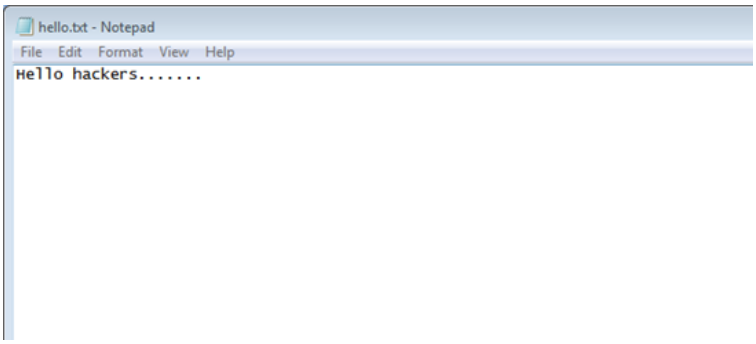
Here also we have to first login using name and email id and create password



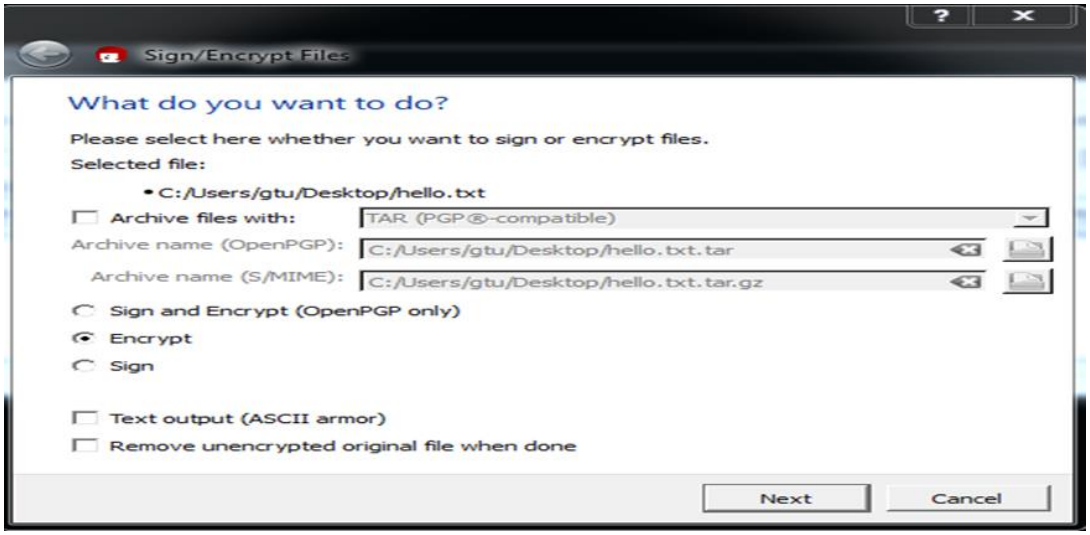


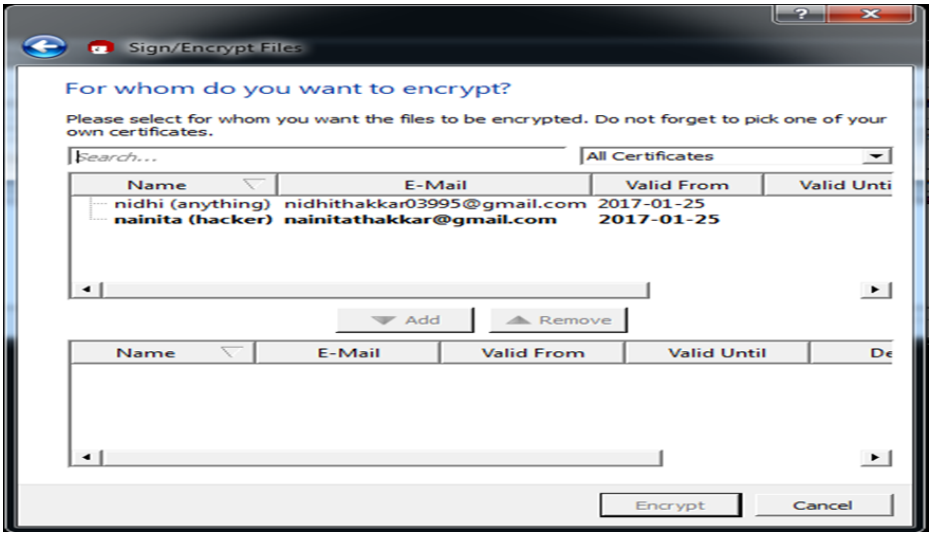


then for encrypting file select file

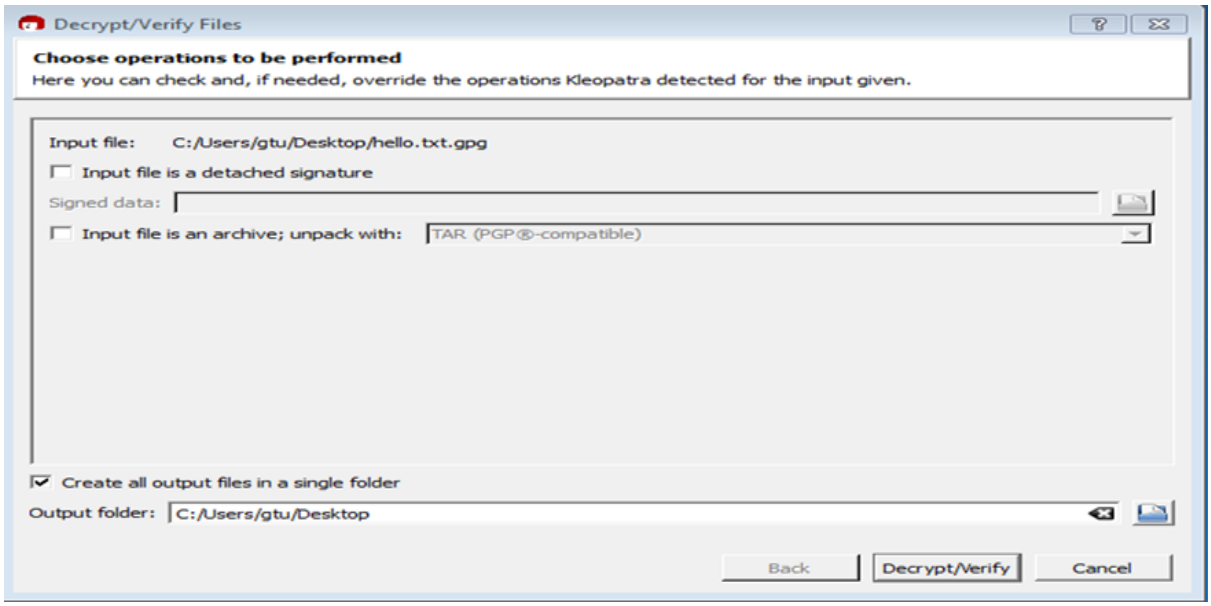


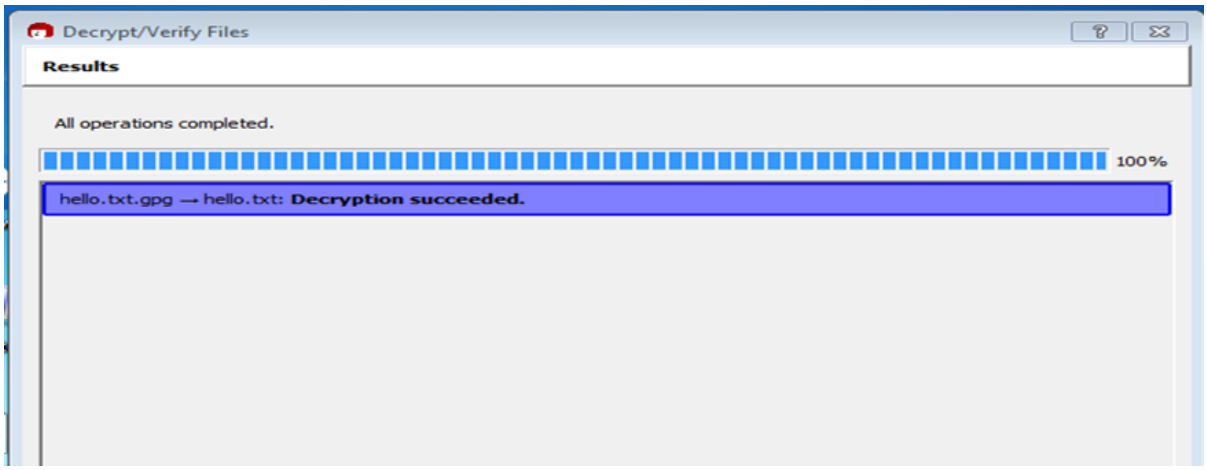
encrypting file using following steps:





For decryption:





CONCLUSION:

In this PRACTICAL -, we learned about gpg tool and we encrypt and decrypt the file and also learned about public and private key.

PRACTICAL --10

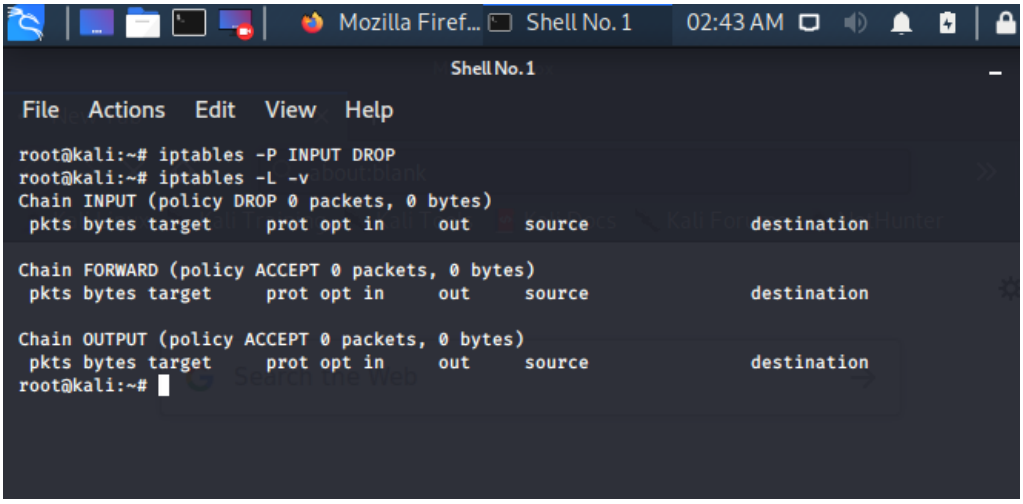
AIM: Setup Firewall that
a) Allow all outgoing connection
b) Block all unwanted incoming connection

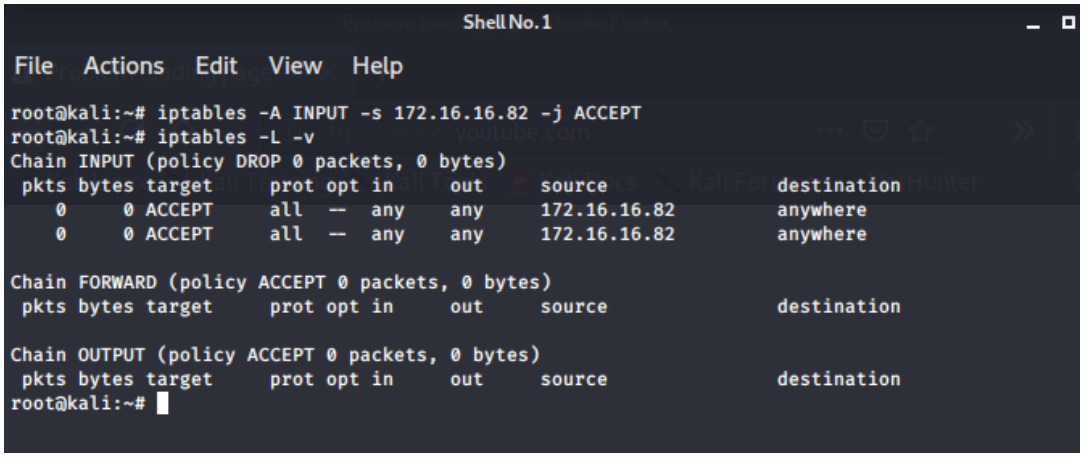
Setup firewall:

- help options

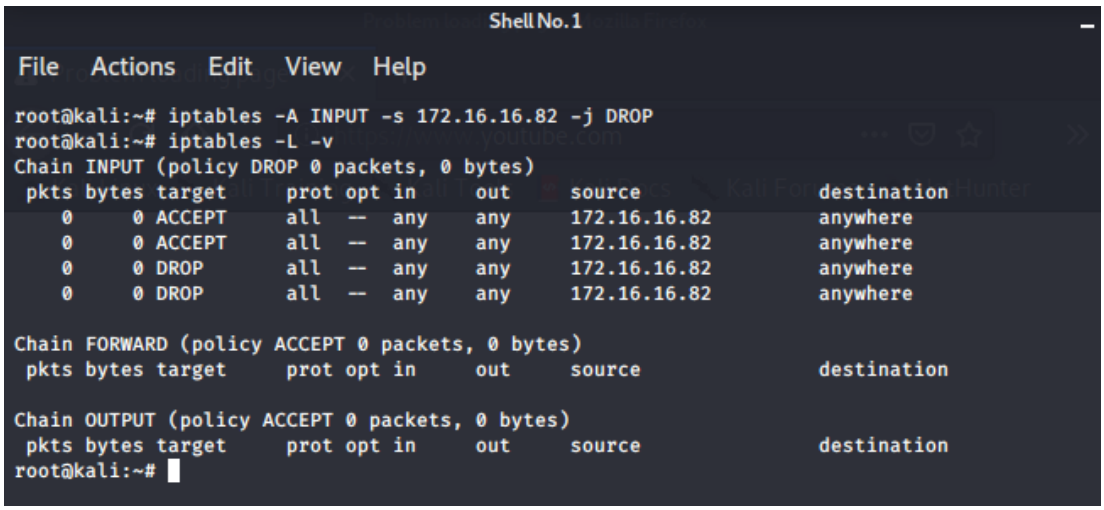
```
root@kali:~# iptables -h
iptables v1.8.3

Usage: iptables -[ACD] chain rule-specification [options]
       iptables -I chain [rulenum] rule-specification [options]
       iptables -R chain rulenum rule-specification [options]
       iptables -D chain rulenum [options]
       iptables -[LS] [chain [rulenum]] [options]
       iptables -[FZ] [chain] [options]
       iptables -[NX] chain
       iptables -E old-chain-name new-chain-name
       iptables -P chain target [options]
       iptables -h (print this help information)

Commands:
Either long or short options are allowed.
  --append  -A chain            Append to chain
  --check   -C chain            Check for the existence of a rule
  --delete  -D chain            Delete matching rule from chain
  --delete  -D chain rulenum
                                Delete rule rulenum (1 = first) from chain
  --insert  -I chain [rulenum]
                                Insert in chain as rulenum (default 1=first)
  --replace -R chain rulenum
                                Replace rule rulenum (1 = first) in chain
  --list    -L [chain [rulenum]]
                                List the rules in a chain or all chains
  --list-rules -S [chain [rulenum]]
                                Print the rules in a chain or all chains
  --flush   -F [chain]          Delete all rules in  chain or all chains
  --zero    -Z [chain [rulenum]]
                                Zero counters in chain or all chains
  --new     -N chain            Create a new user-defined chain
  --delete-chain
            -X [chain]          Delete a user-defined chain
  --policy  -P chain target
                                Change policy on chain to target
  --rename-chain
            -E old-chain new-chain
                                Change chain name, (moving any references)
Options:
    --ipv4      -4              Nothing (line is ignored by ip6tables-restore)
    --ipv6      -6              Error (line is ignored by iptables-restore)
[!] --proto     -p proto        protocol: by number or name, eg. `tcp'
[!] --source    -s address[/mask][...]
                                source specification
[!] --destination -d address[/mask][...]
                                destination specification
[!] --in-interface -i input name[+]
                                network interface name ([+] for wildcard)
 --jump -j target
                                target for rule (may load target extension)
  --goto      -g chain
                               jump to chain with no return
  --match       -m match
                                extended match (may load extension)
  --numeric     -n              numeric output of addresses and ports
[!] --out-interface -o output name[+]
                                network interface name ([+] for wildcard)
  --table       -t table        table to manipulate (default: `filter')
  --verbose     -v              verbose mode
  --wait        -w [seconds]    maximum wait to acquire xtables lock before give up
  --wait-interval -W [usecs]    wait time to try to acquire xtables lock
                                default is 1 second
  --line-numbers                print line numbers when listing
  --exact       -x              expand numbers (display exact values)
[!] --fragment  -f              match second or further fragments only
  --modprobe=<command>          try to insert modules using this command
  --set-counters PKTS BYTES     set the counter during insert/append
[!] --version   -V              print package version.
root@kali:~#
```

- allowing all outgoing connections:-





- Block all unwanted incoming connection:-



CONCLUSION:

In this PRACTICAL -, we learned about firewall and in that we learn to allow connections and to drop unwanted connections.