HackerOne-Postbook

PostBook is a vulnerable web application of Hacker101 platform, the goal is to find 7 flags using owasp top 10.

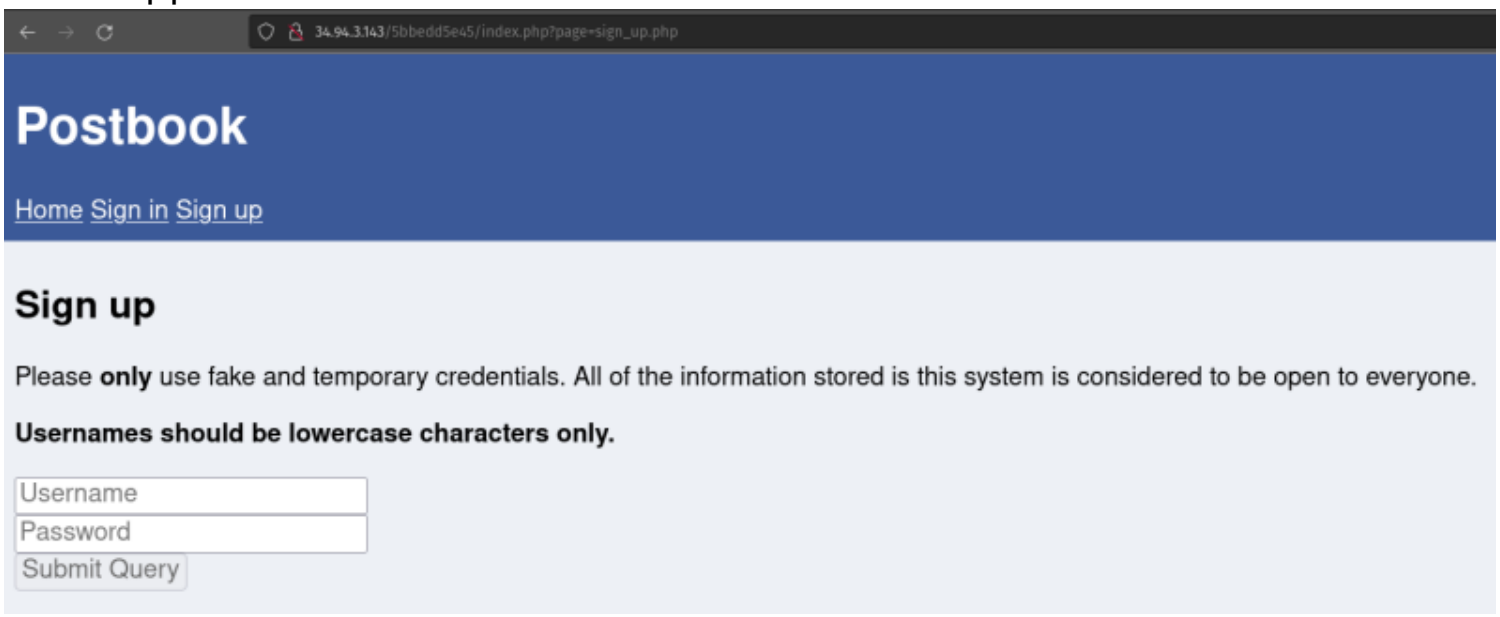
Goal: Find 4 flags

Before trying anything with the web page it is very important to know how it works, what we can do as any user, review the source code of each of the sections of the page.

What is postbook?

It is a web application where we can register, log in and publish things, and there are two types of publications: public and private.

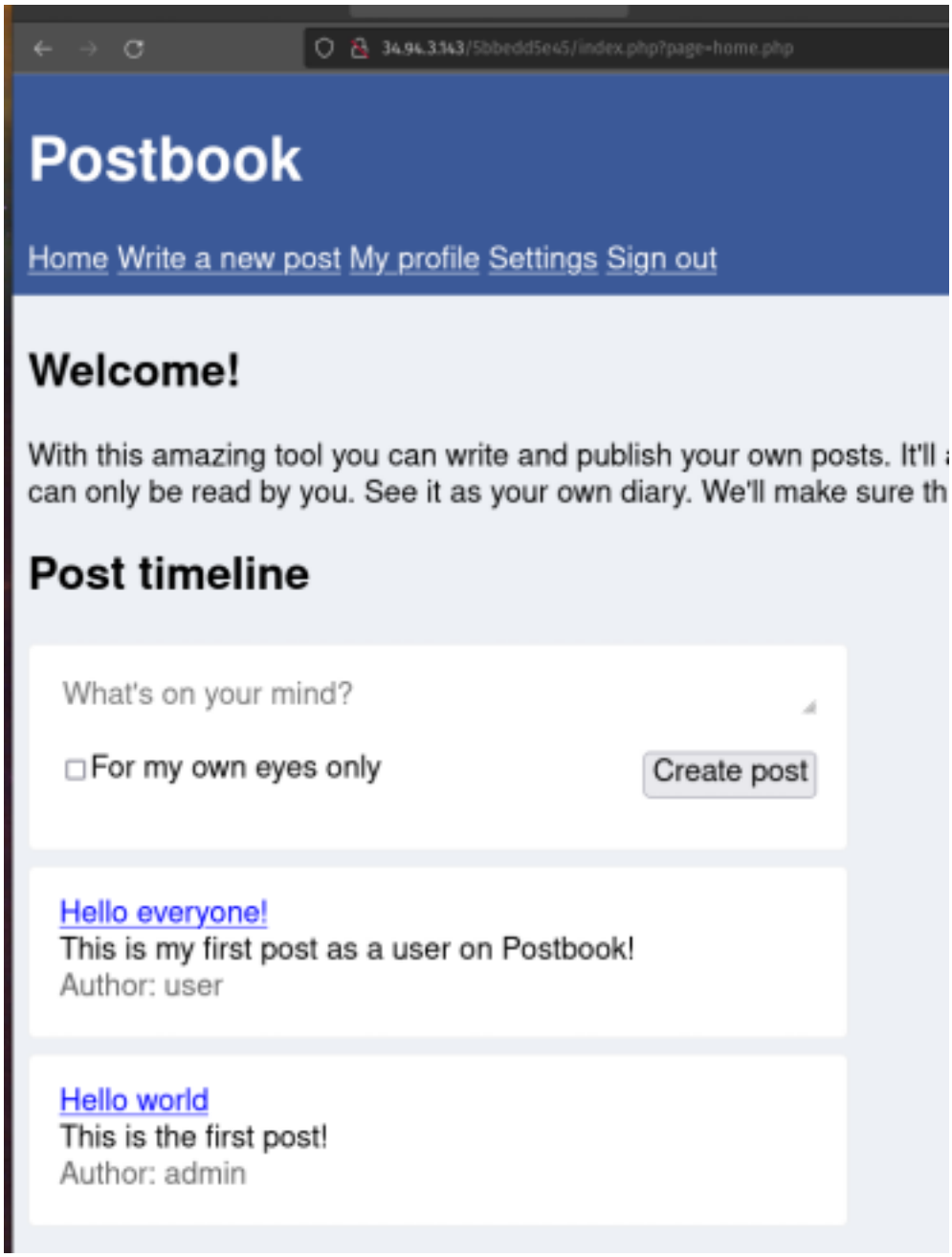
Something that catches my attention is the message in the following screenshot, "all information stored in this system is considered open to all". When we register we are asked to log in, now I will be able to know what else is in the application.



Once we log in we can see many more options, we can now write posts, visit our profile, change settings and log out.

We can also see the authors of published posts, among them:

→ admin
→ user



Once we log in we can see many more options, we can now write posts, visit our profile, change settings and log out.

Flag 0:

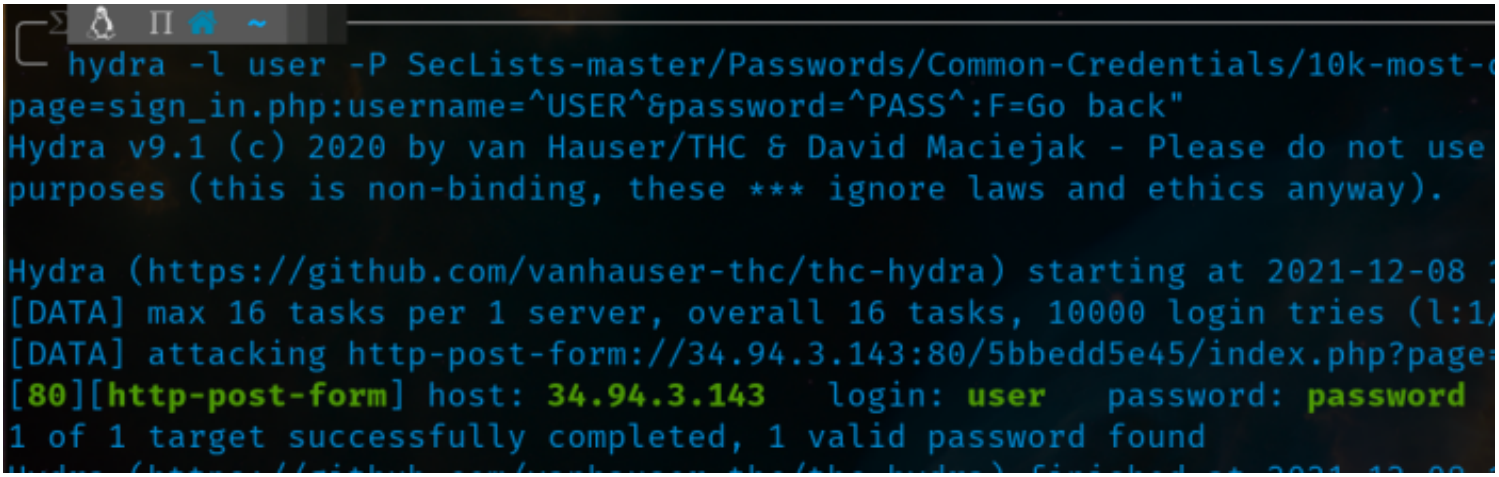
We can also see the authors of published posts, among them:

- admin
- user

Here we can start on passwords, will they use weak credentials? We can run a hydra attack to confirm this!

Thanks to the following command we know that the user "user" has a password that is really easy to obtain.

hydra -l user -P SecLists-master/Passwords/Common-Credentials/10k-most-common.txt 34.94.3.143 http-post-form "/5bbedd5e45/index.php?-page=sign_in.php:username=^USER^&password=^PASS^:F=Go back"

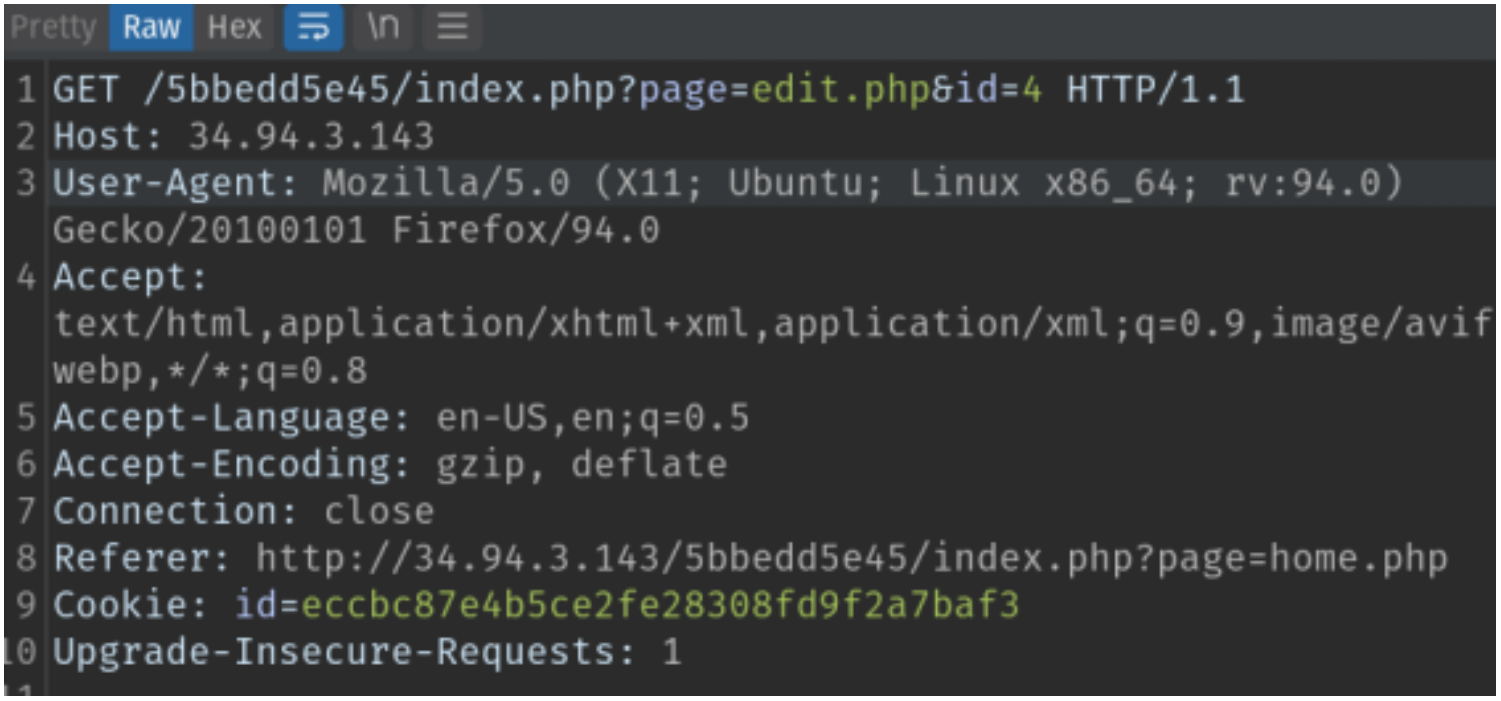


Once we log in with the user "user" we can obtain the first flag!

Flag 1:

According to the first tip on this challenge, we should focus on the posts and look closely at the parameters that are sent to the server.

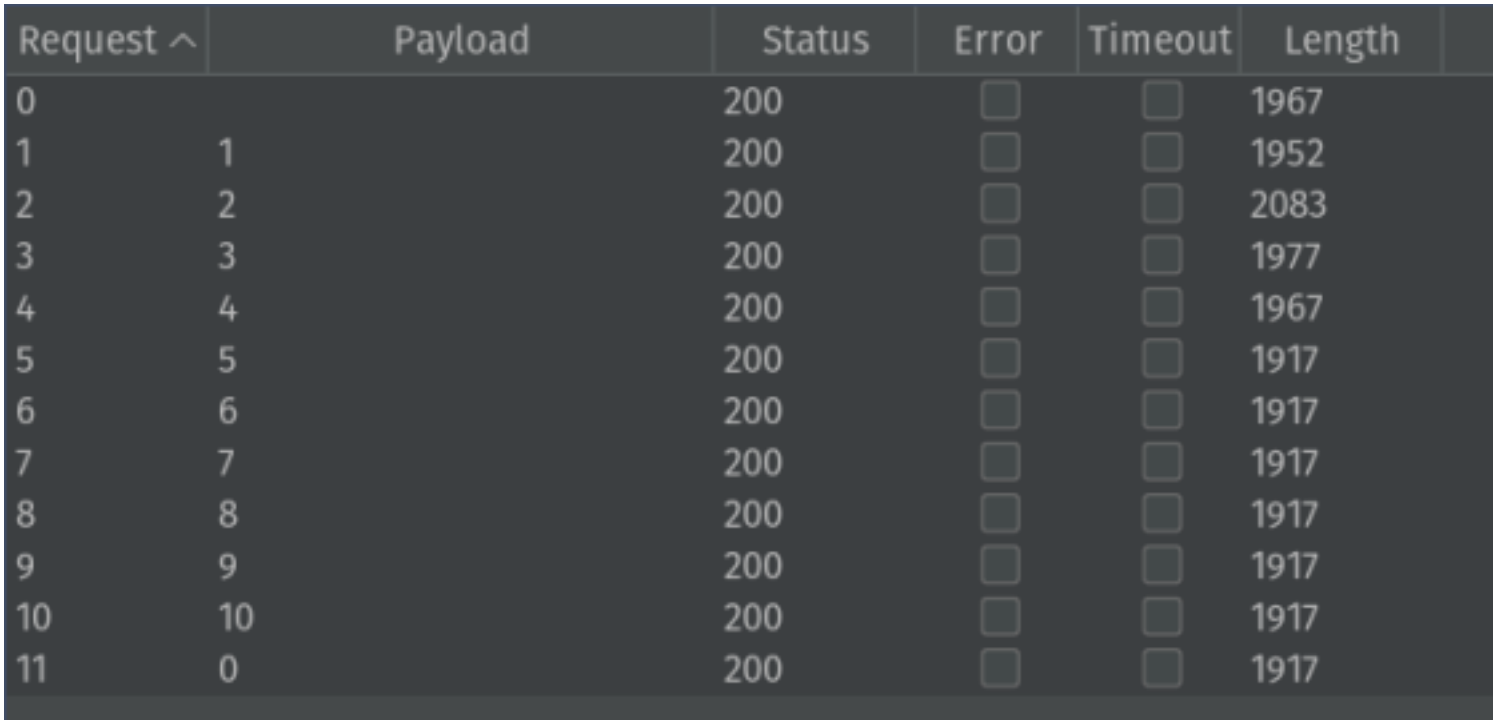
For the inputs it is always good to ask ourselves, can we inject something, are there any parameters that are not sanitized, what if...?



When intercepting the post edit request with BurpSuite there are two parameters that are being sent to me:

→ page
→ id

But for now I will focus on the second one as I believe this determines the post identifier, what would happen if I change it?



Request	Payload	Status	Error	Timeout	Length
0		200			1967
1	1	200			1952
2	2	200			2083
3	3	200			1977
4	4	200			1967
5	5	200			1917
6	6	200			1917
7	7	200			1917
8	8	200			1917
9	9	200			1917
10	10	200			1917
11	0	200			1917

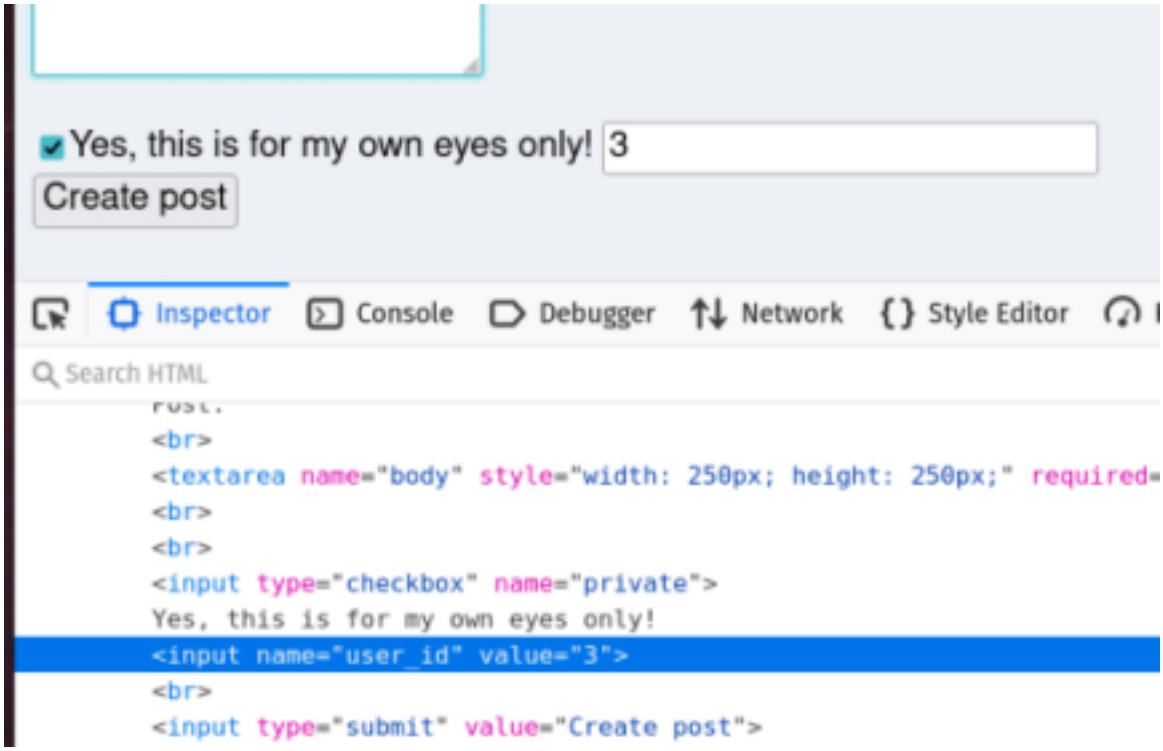
When I modified the ID parameter with intruder, I noticed that all the requests had the same length except for one, the payload with the id number 2!

I can see the post made by the "admin" user and modify it so that ALL users of the platform can read it. Now thanks to this I got the flag!

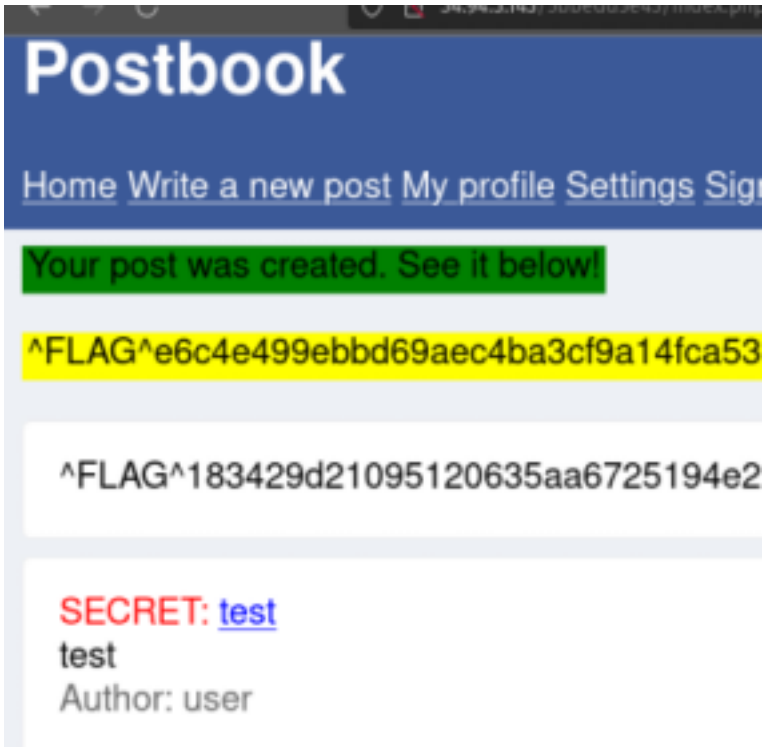

34.94.3.143/5bbedd5e45/ind
Postbook
Home Write a new post My profile S
Your post was created. See it below
^FLAG^8cf4f965ddf02a7300eb00ca
Dear diary...
I am so glad that I am on Postboo
down my thoughts and no one ca
tomorrow. Yours truly, admin
Author: admin

Flag 2 & 4:

Upon closer inspection of the form to create a new post, reviewing the source code I discovered a "hidden" input when I removed that option, there is a parameter that uses an ID, so I think that by modifying it I could hide or make all posts visible to the user.



By modifying it we get the flag and create a new post!

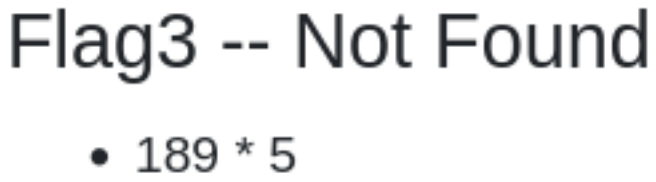


I got two flags! Since by modifying the ID parameter I can also edit the POSTs

of other users!

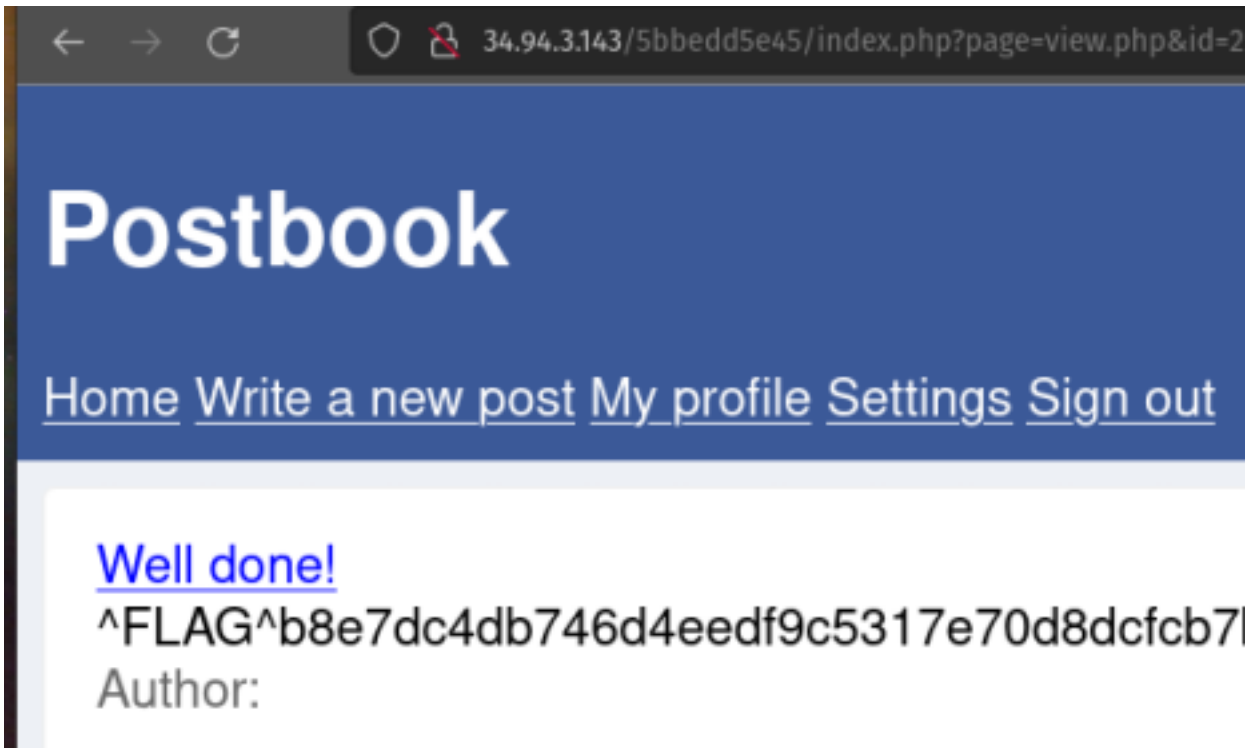
Flag 3:

The truth is that for flag 3 I had no idea where to look, and I got the following advice:



When solving the operation I get the result 945, so I thought it might be the ID of some POST or user!

It worked! When trying to see another user's post I was able to modify the ID and see the "secret" POST and get the flag!



Flag 6:

For flag number 5 I decided to get some advice as I had no idea what to do and I knew I had to do something with the cookies, when I looked at them I realized it is a simple hash, but what algorithm did they use?

I used this page and found out they used MD5!

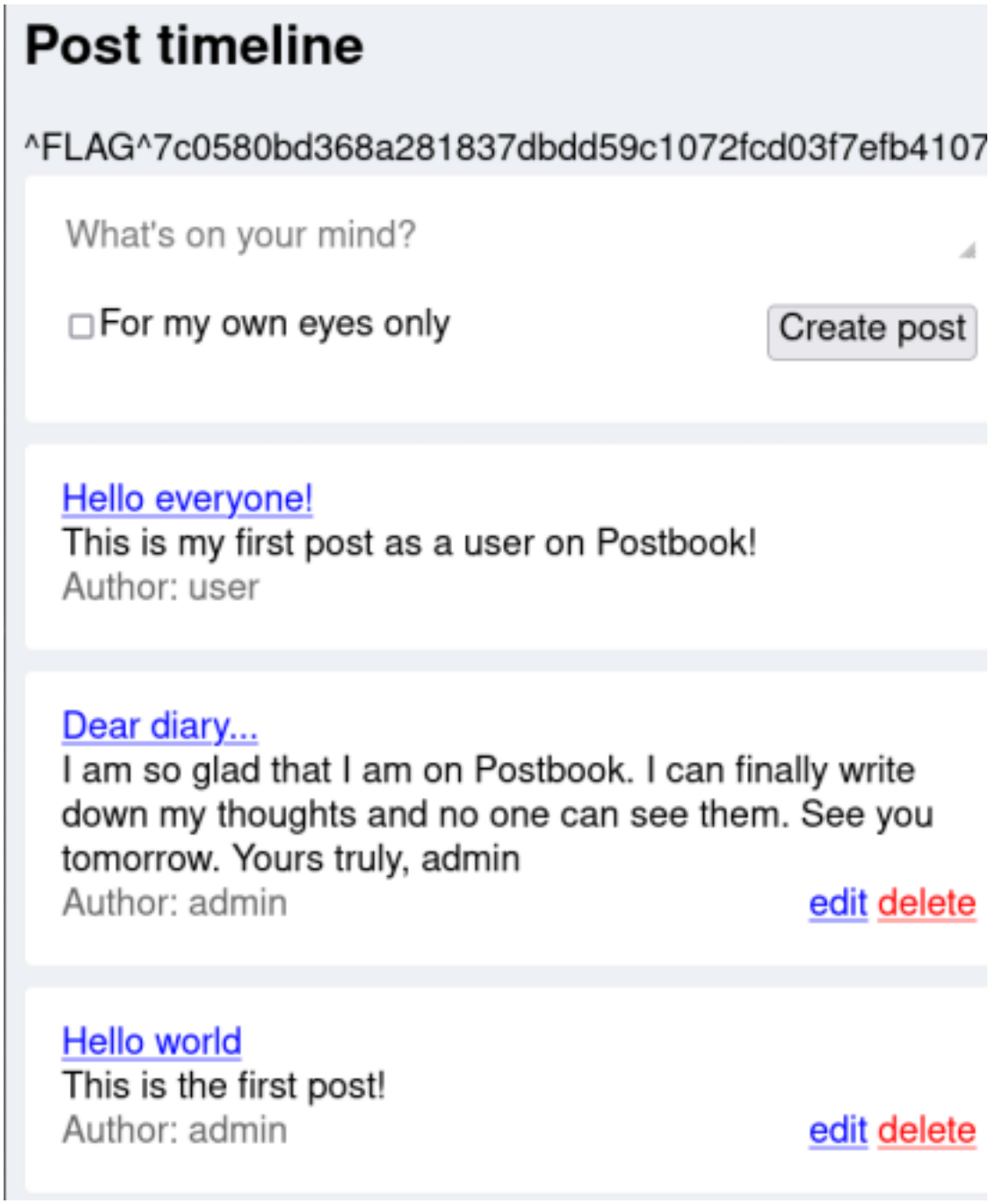
→ link: https://hashes.com/en/tools/hash_identifier

To know the value of the hashes I used crackstation and I realized that they only take the ID of the user and with MD5 they obtain the hash, so to obtain the hash of the user with ID 1 I only have to do that operation.

c81e728d9d4c2f636f067f89cc14862c	md5	2
eccbc87e4b5ce2fe28308fd9f2a7baf3	md5	3

To create the hash use cyberchef and just pass the ID number 1, obtaining the following result:

To log in as the admin user, you can use the cookie editor extension and modify the cookie!



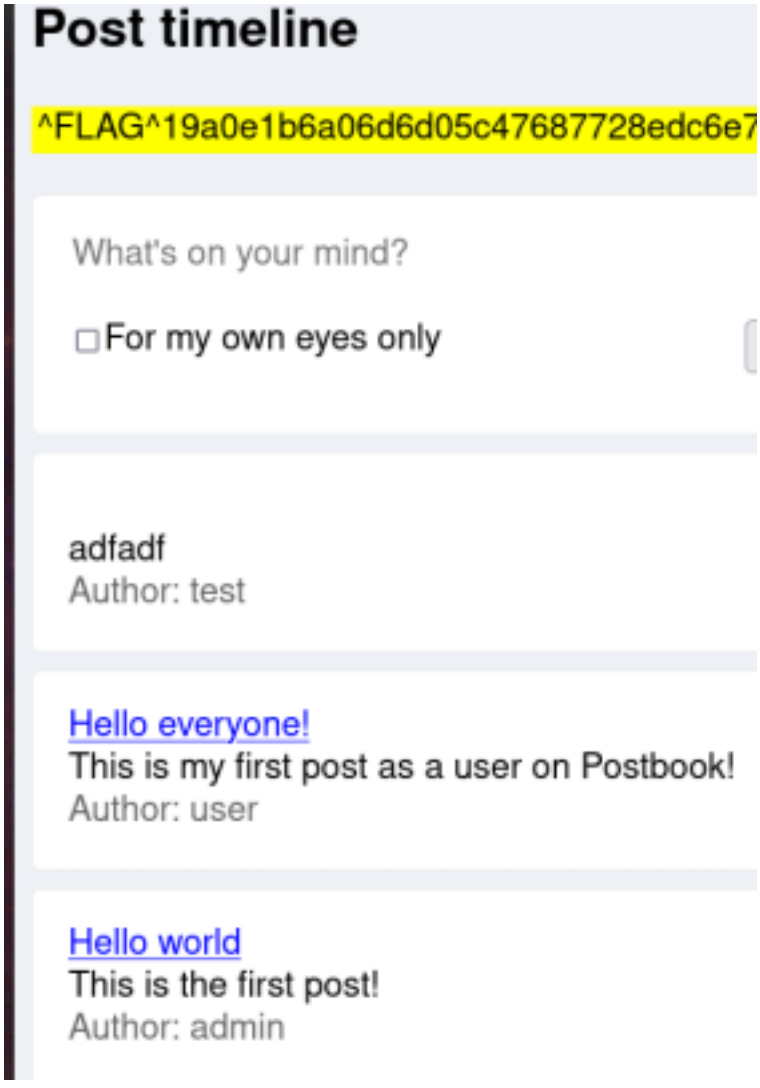
Now we got the flag and to check it we can edit the posts of the admin user.

Flag 7:

For the last flag, we must always intercept the actions that a user can do, the actions to delete a post must take something as a parameter to indicate the post to delete it!

```
Pretty Raw Hex \n
1 GET /5bbedd5e45/index.php?page=delete.php&id=
  c9f0f895fb98ab9159f51fd0297e236d HTTP/1.1
2 Host: 34.94.3.143
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86
  Gecko/20100101 Firefox/94.0
4 Accept:
  text/html,application/xhtml+xml,application/xml
  webp,*/*;q=0.8
```

As with cookies it takes an MD5 HASH with the value of the index of the POST! so to delete a post I must create an MD5 hash with the value of the index assigned to the post I want to delete!



In my case I decided to delete the POST with ID number 2, the deleted post was the "hidden" message from the admin user!

What I learned?

This CTF showed me the importance of knowing how the web application works, not just what you can and can't do!
It is also important to know how the application handles the data to be able to make modifications, in this case it was the parameter ID!
The importance of cookies and how they can be obtained in a relatively simple way.