CTF 1

FLAG 1: <FLAG_E4rMfZ>

In the home page of the Cake Shop some users had posted personal information such as birthdates, name of relatives, and locations. Additionally, the website automatically provides you with the username of each person who posts a review. For this flag I chose to smart brute-force attack user `cristiano` in the login page.

After gathering some personal information about them I generated a word list that I will use to guess their password.

```
    "cristiano": [
        "portugal","madeira","02","feb","february","05","5","5th",
        "maria","1985","85","cristiano","jr","georgina","viva","madrid",
    ],
```

I wrote a small python script that would generate all possible permutations of the wordlist and try it in the `localhost:3000/login` page. After some fine tuning I was able to return the password for `cristiano` - `mariamadrid1985jr`. When logged it, I got the key.

“Pasted image 20241019092910.png” could not be found.

FLAG 2: <FLAG_WDzR9j>

When I logged in as `cristiano` the first thing I noticed was the page `orders`. Which had a search bar in it, and the individual orders below the search bar. The vulnerability here was that the search bar was SQL injectable.

“Pasted image 20241019093131.png” could not be found.

After some playing around I realized that the orders shown are returned by an SQL query which from the visible elements (order number, price number of items, customer) returns four elements. Therefore, I tried some SQL injections into the search bar to see if I could get it to return information from other tables about the users.

```
' UNION SELECT id, NULL, username, password FROM users -- -
```

I tried this injection which then returned the username and password of every user. The username that I was interested in was `root` since this would give me admin access to the website.

`root` - `GTfCR38n3A`

After signing in as root I got the second flag.

“Pasted image 20241019093714.png” could not be found.

FLAG 3 <FLAG_8EPwdd>

This flag I found using a tool called `FFUF`. I was looking for any hidden links, or stuff that perhaps was left there unintentionally. I found this `common.txt` Common Words on GitHub to try first.
Using this command I was able to fuzz for any possible hidden paths.

```
ffuf -u http://127.0.0.1:3000/FUZZ -w common.txt -mc 200,403
```

This command will return any paths that are either open to any user (200), or they have forbidden access (403).

“Pasted image 20241019095219.png” could not be found.

When I ran the command I was able to get a few open links, but the one that we are interested in for this flag is secret. When navigating to `localhost:3000/secret` we get the third flag.

“Pasted image 20241019095325.png” could not be found.

FLAG 4: <FLAG_Np6XMD>

The vulnarability for this flag was identified by checking the `robots.txt` file. By navigating to `localhost:3000/robots.txt` I was able to find paths the developers did not want the crawlers to access.

```
User-agent: Googlebot
Disallow: /orders

User-agent: Googlebot
User-agent: AdsBot-Google
Disallow: /files/index

User-agent: *
Disallow: /
```

This reveals to us a hidden path `files/index`. Which if you try to access with just as an `anon` or a normal user like `crsitiano` you will get an error `403`.

“Pasted image 20241019095754.png” could not be found.

However since we have gotten access to root account, we can navigate to this while signed in as root. This is what you see when you go to `localhost:3000/files/index` as root:

```
Index of static/

- images
    - background.jpg
    - background_grey.jpeg
    - background_studio.JPG
    - benz.jpg
    - cr7.jpeg
    - messi.jpeg
    - ney.jpg
    - off.jpg
- robots.txt
- secret
    - not-for-public
        - flag.txt
- styles
    - style.css
```

As we can see there is a secret path at `localhost:3000/static/secret/not-for-public/flag.txt`. I also found out that if you were able to identify this path somehow you do not need root access to navigate there. The link is reachable even as `anon`.

"Pasted image 20241019100320.png" could not be found.

FLAG 5: <FLAG_UAfGM1>

When you click on details for any order, you will see the URL changes to `orders/#` where # is the number of the order. `cristiano` has the orders 3, and 4, however if you are signed it as `cristiano` and you manually alter the URL to say point to order 1, it will direct you to that page.
The vulnerability is no permission check for manual URL input.

"Pasted image 20241019100704.png" could not be found.

Good job! FLAG 6: <FLAG_NFVw2p>

If we log in as `root` user, we will see there is an extra page that is only available to `root` called `Reports`. Which has a `<textarea>` element for creating reports. The vulnerability with this element is that it is susceptible to `XSS` attacks. Therefore I tried the most basic script just to see what filters it has.

```
<script> alert("xss") </script>
```

This returned an alert, meaning that the script worked, and afterwards I got the last flag.

“Pasted image 20241019101212.png” could not be found.

This was the most fun assignment I have done in 4 years of Computer Science :)