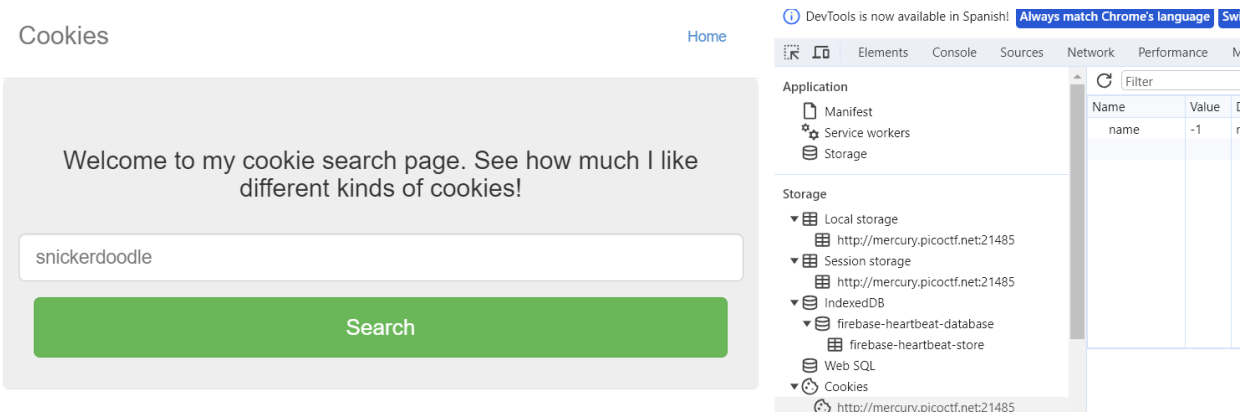
Gabriel Alexander Pástor Villacís - 00211253

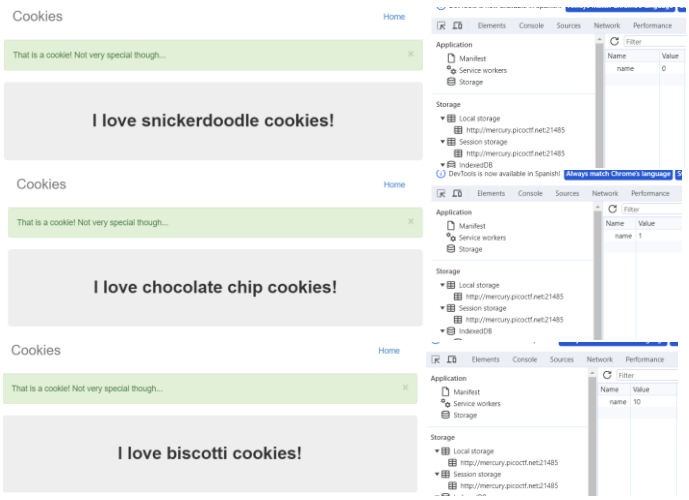
Computer Security HW3 - PicoCTF

Cookies:

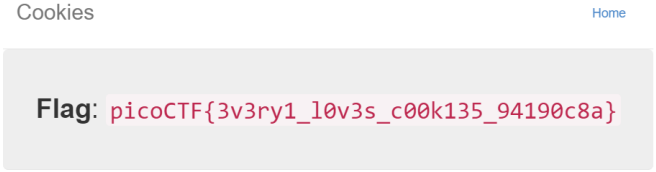
Its implementation was a little bit interesting, mainly by the name of the problem. Although the point was to get in the inspection page (Developer or Inspect section), the answer was in the Application tab. The point of cookies here is the value element inside it. Like data saved in a DB, with its respective elements, here the point was the variation of that value.



Based on the image, the point was that for no cookies the value was negative, but then for the first case: snickerdoodle, the value turned to 0. Considering that, there should be an ID to deploy the rest of the cookies, and the brute force was the only one I could consider for it.



Of course, there was a limit of 29 values, meaning to the cookie names. Then, the iteration through each cookie name let me for the value 18 get a different result. And then, there was a flag.



Flag found at second page of the website, with cookie value = 18

Gabriel Alexander Pástor Villacís - 00211253

Scavenger hunt:

The case here began when the page mentioned that the program was done with html, I considered to look at the source code, and then the hint appeared by itself.

```
      <div id="tabintro" class="tabcontent">
        <h3>How</h3>
        <p>How do you like my website?</p>
      </div>

      <div id="tababout" class="tabcontent">
        <h3>What</h3>
        <p>I used these to make this site: <br/>
          HTML <br/>
          CSS <br/>
          JS (JavaScript)
        </p>
    <!-- Here's the first part of the flag: picoCTF{t -->
      </div>

    </div>

  </body>
</html>
```

Flag (1) found at source code, in the comments part

In the same way, the code also includes css and js files, the point of inspecting implied to analyze them also. Then there was a result in the css file, but the js file, let me analyze something about indexing on Google.

```
.tablink {
    background-color: #555;
    color: white;
    float: left;
    border: none;
    outline: none;
    cursor: pointer;
    padding: 14px 16px;
    font-size: 17px;
    width: 50%;
}

.tablink:hover {
    background-color: #777;
}

.tabcontent {
    color: #111;
    display: none;
    padding: 50px;
    text-align: center;
}

#tabintro { background-color: #ccc; }
#tababout { background-color: #ccc; }

/* CSS makes the page look nice, and yes, it also has part of the flag. Here's part 2: h4ts_4_l0 */
```

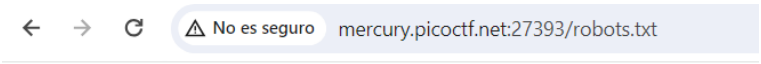
```
function openTab(tabName,elmnt,color) {
    var i, tabcontent, tablinks;
    tabcontent = document.getElementsByClassName("tabcontent"
    for (i = 0; i < tabcontent.length; i++) {
        tabcontent[i].style.display = "none";
    }
    tablinks = document.getElementsByClassName("tablink");
    for (i = 0; i < tablinks.length; i++) {
        tablinks[i].style.backgroundColor = "";
    }
    document.getElementById(tabName).style.display = "block";
    if(elmnt.style != null) {
        elmnt.style.backgroundColor = color;
    }
}

window.onload = function() {
    openTab('tabintro', this, '#222');
}

/* How can I keep Google from indexing my website? */
```

Flag (2) found at mycss.css, with hint of (3) in myjs.js

Of course, the thing is that a page should be visited or analyzed by a Google crawler, on its content and information. The point was that, pages are indexed as blocked or allowed to access in a robots.txt component, which is a file that defines which URLs are blocked to be crawled. Then the main link here: https://support.google.com/webmasters/answer/7645831?hl=en. In that way, to access there, the only thing was to replace the subdirectory and add in the domain URL the extension of that file only.

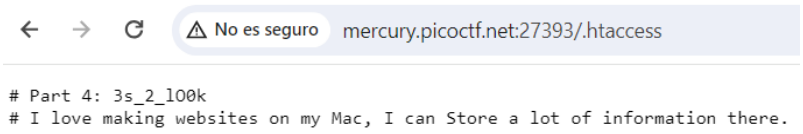



```
User-agent: *
Disallow: /index.html
# Part 3: t_0f_pl4c
# I think this is an apache server... can you Access the next flag?
```

Flag (3) found at robots.txt file, in the comments

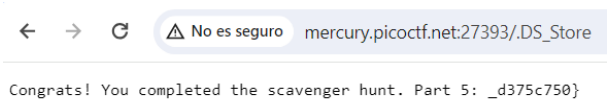
Then, considering the Apache hint, I tried to search for information about files of Apache that could give me an idea. Then, I found that .htaccess file implies the access to a website, and that name is applied to

manage pages, directories and subdirectories, being “special files” for that role. Then the link was: https://httpd.apache.org/docs/2.4/configuring.html.



Flag (4) found at .htaccess file, in the comments

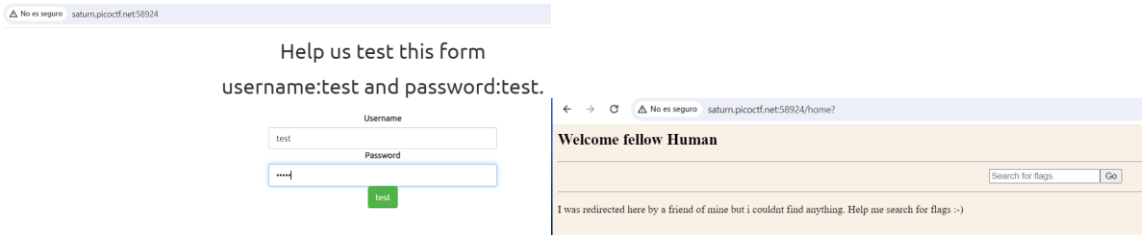
As it mentioned Mac, there should be taken in count that for Mac, the .DS_Store is applied to save information of a website. Then, the final result appeared.



Flag (5) found at .DS_Store file

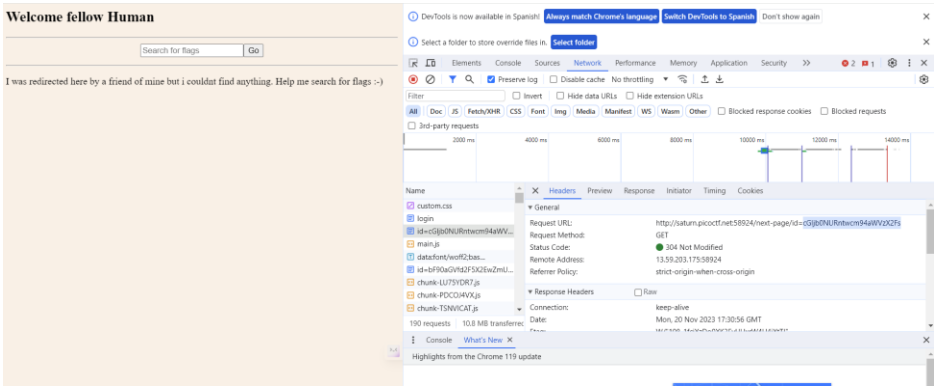
Find me:

This case implied a website, where you enter the credentials and then access to the main page.

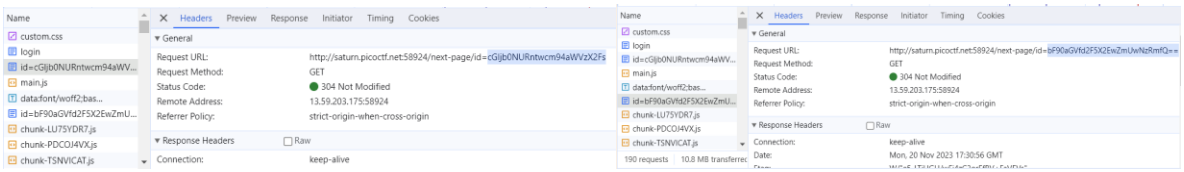


In one part, the search textbox looks important to search for a flag, but then trying to get something there was nothing to do. However, as the text of the page mentions, there was redirection. Then, I applied ‘Preserve log’ to retrieve information about the pages form where it was redirected.

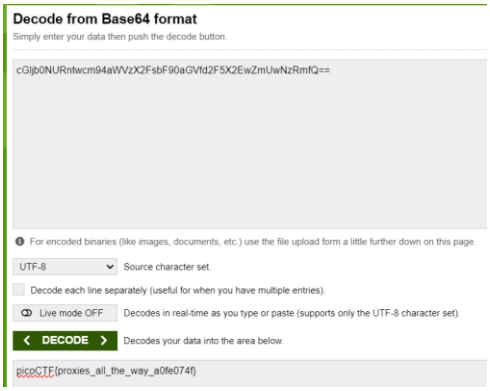
When I moved back and to the main page after login, then I could get the information of different components, since visual elements to other things. However, I could look for other pages that could be in the process.



With those pages, I could see that there is nothing there that could give me some data, but the point was then in the id. As I could see, the id codes are in the base64 format. As base 64 implies encoding and decoding with ASCII, then I could consider a decoder for that action, although in this case a decoder webpage could solve this rapidly now.



Then, with the ids and the decoder page, the result appeared.

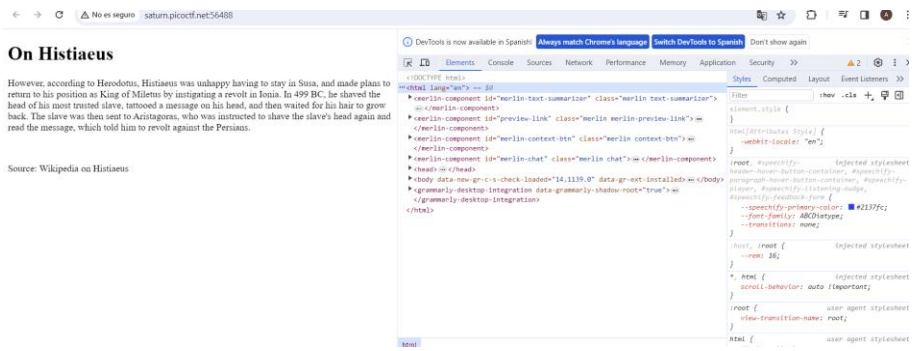


Flag found at redirect pages (their IDs), in base64 format (then, decodified to get the real one)

Inspect HTML:

These cases implied a simple webpage where a text was implemented, then I moved to the inspector section.

At the first moment, nothing interesting appeared, the point then was to analyze what could be there in the Elements tab section, and maybe the body part could give some information.



Finally, just getting inside the body structure, in the html code of the page, just a simple comment gave the answer for this problem, just about inspecting the HTML.

Flag found at the source code page, in the body section, in the comments

It is my Birthday:

Based on the case presented, it could imply a hash implementation, or well, an algorithm that could help me to verify if a file was modified. When I meant to that, it's important to apply a hash algorithm or at least one that could be important, and one of the most common. Then, considering those points, MD5 could be taken in count.

For MD5, the implementation of it and mainly of hashing in general, implies that hash values of two documents are compared to verify corruption on them (based on the idea of that both are the same file but one of them after travelling on Internet). One of the main things that hashing has is that, if both hash values are the same, we have a collision then, and the MD5 algorithm is not resistant to that.

So then, considering the case of Wang and Yu, there could be a form to generate or obtain two different files with the same hash value. The case here is that I considered files found in a link to verify it: https://www.mscs.dal.ca/~selinger/md5collision/.

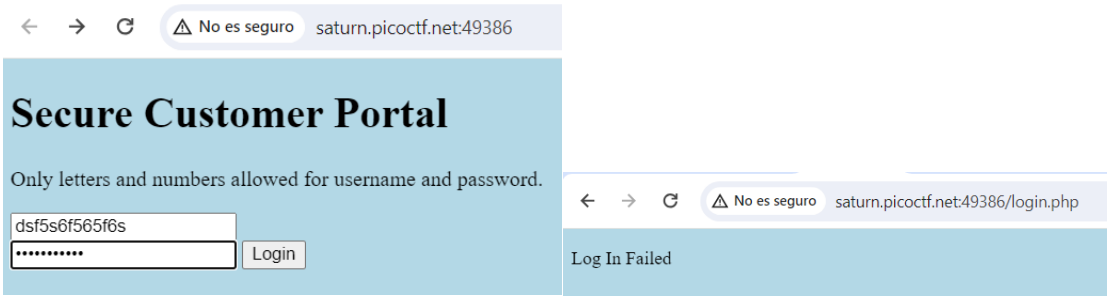
Flag found after submitting two pdf files with same hash, the flag was in the redirected page, where the source code was

Then, I could get something interesting there, the flag, and with the collision the result was there.

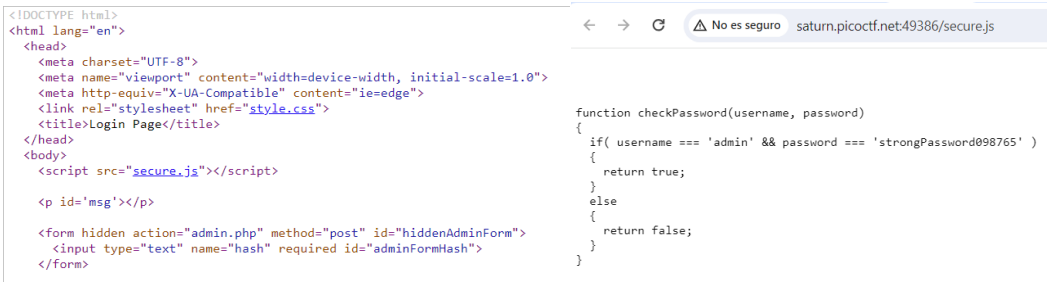
Gabriel Alexander Pástor Villacís - 00211253

Local Authority:

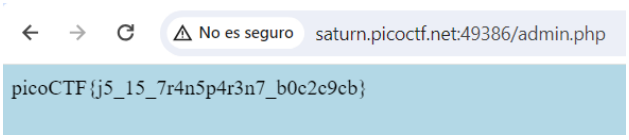
This case implied a simple website, for login mainly. At the beginning, the webpage shows the page to enter the credentials and a message for a login failed, if the credentials are wrong.



However, as it was done in previous problems, this could be analyzed in the inspection part, or in the source code. As I could see, there was not enough information about a possible flag, and then in the files linked to the code, I could see something unexpected in the secure.js file.



One of the worst or bad security implementation in web applications is the exhibition of credentials, mainly of admin credentials, then given the username and password, I applied them to login. Finally, I could get the flag.

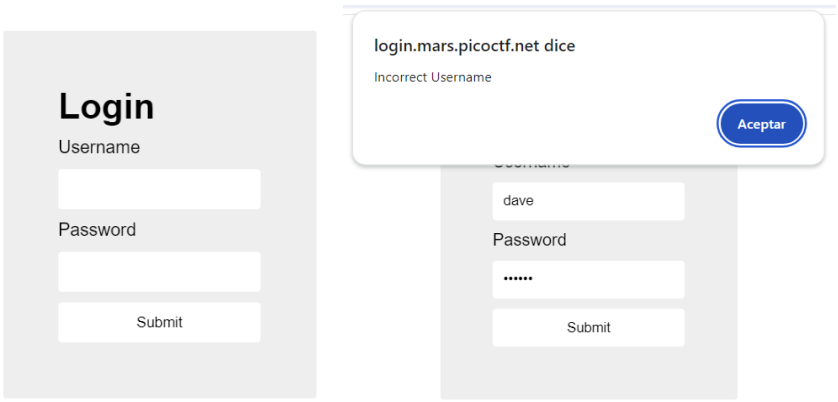


Flag found after the login page, with the admin credentials

Login:

The next webpage, as it was in previous problems, implies the insertion of credentials. It doesn't matter how many times or different SQL injections I could try, anything could give me a clear hint there.

Gabriel Alexander Pástor Villacís - 00211253

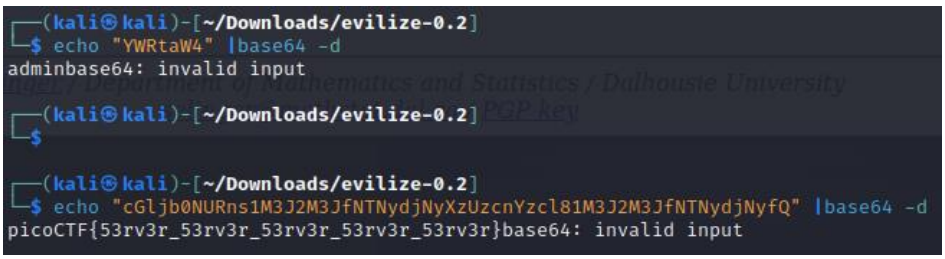


Looking at the source code, there were different source files, but mainly the index.js file. In this file, the implementation at the end implied a verification of credentials to login.

```
<html>
    <head>
        <link rel="stylesheet" href="styles.css">
        <script src="index.js"></script>
    </head>
    <body>
        <div>
          <h1>Login</h1>
          <form method="POST">
            <label for="username">Username</label>
            <input name="username" type="text"/>
            <label for="username">Password</label>
            <input name="password" type="password"/>
            <input type="submit" value="Submit"/>
          </form>
        </div>
    </body>
</html>
```

The thing there was the string to compare, and then I considered this case as base64 again, but in this case for decoding.

With that in count, the decoding could give me a hint about the credentials and mainly to login and get the key. However, the password looked to be the flag.

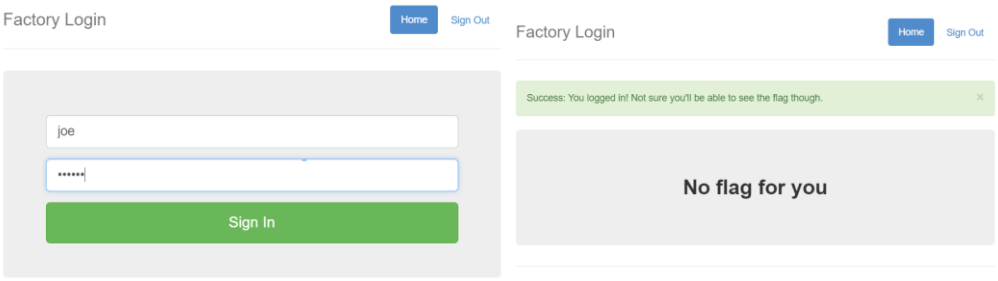


Flag found after applying base64 decoding to comparison strings to verify login access in website, in the webpage source code

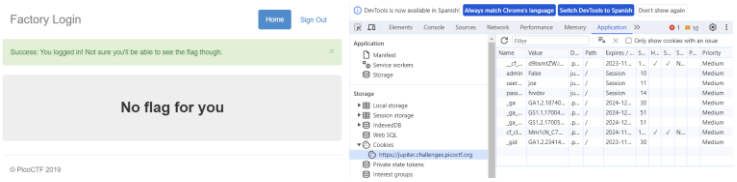
Gabriel Alexander Pástor Villacís - 00211253

Logon:

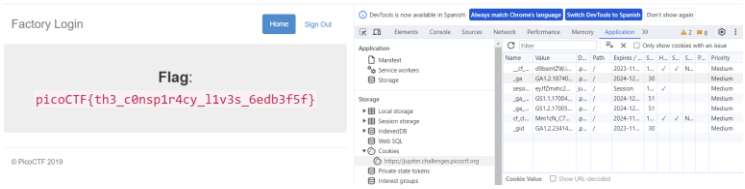
Maybe as different cases before, this could imply a simple case of inspection in the website. Logged in as Joe, the message of no flag appears, and with another credential it is the same.



Considering the case of websites and credentials, I could consider the case of cookies, which are implemented here to save information about accounts of a webpage.



The case here is that, with the cookies tab opened, there was an element that looked strange, and that was an admin Boolean. As a Boolean if the account was not of the admin, the web app will show every time the same message. Changing the value then I could look for what could happen in the page, and then I could get the flag. Considering that, an implementation of components like that one in the cookies could make a website vulnerable to any kind of access.



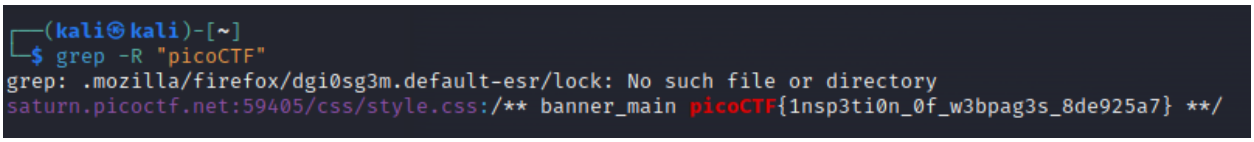
Flag found after login, modifying the admin value in cookies section of the page, as 'True'

Search source:

At the first sight, it looks like a simple webpage, really defined. And in the source code there was a hint that tells me that the try to look for the flag was in failure. Although there was a hint about mirror a webpage.

The case here is that mirror a webpage could be considered as copying or downloading a website, with all its source files and other elements related. Then, for linux the command was clearly defined to save a website, so that could let me get all the elements related to it.

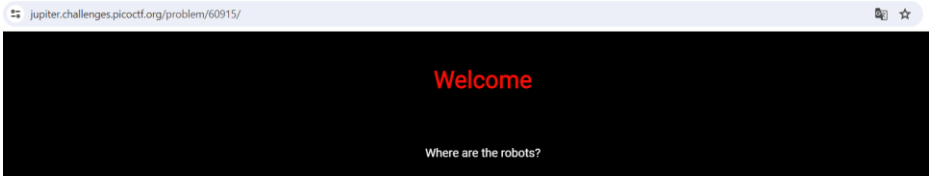
In the same way, with all the source code files in a directory, the use of grep command could let me search for a possible flag, taking in count that all the flags begin with “picoCTF”. Finally, the flag was found.



Flag found after webpage mirroring, and then being searched in the hole source code of the webpage (using grep)

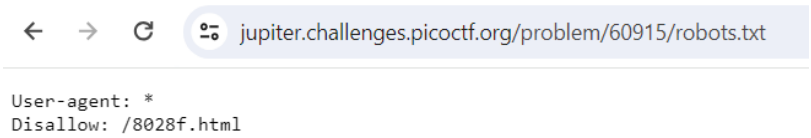
Where are the robots:

As a simple page, it doesn’t tell me something about a flag at the beginning. However, the problem was not in the source code or the inspection part, it was in the problem name directly.

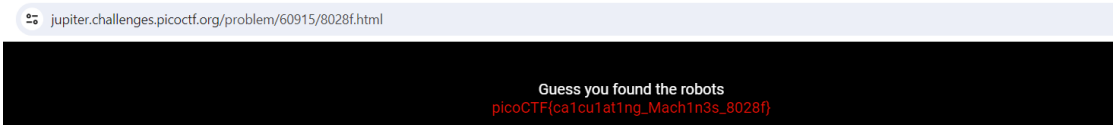


As I could see in previous problems, there was a file to define pages and content to be indexed of excluded of a website, and that was useful to define the rules for crawlers to access in a website. If the case was about robots.txt, then it could help me to get the flag.

Gabriel Alexander Pástor Villacís - 00211253



As it seems, there was not exactly something that could directly tell me about the flag, but as I could see there was a disallow to a page in the site. Maybe the point here was to access on it, because there could be the flag. Finally, the flag was gotten.



Flag found at the disallowed page of the website, defined in the robots.txt file for the website