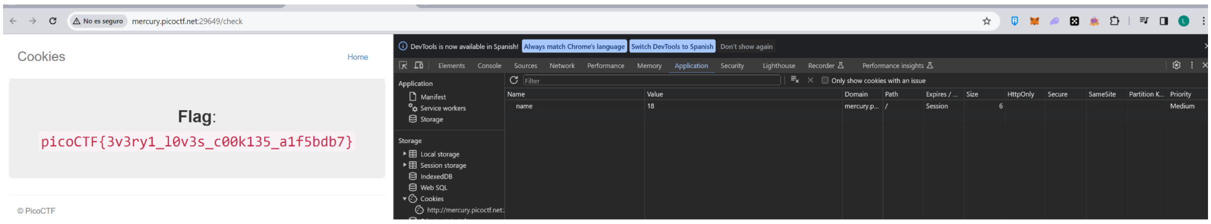
Homework 3 Computer Security

Luis Astudillo
00211000

1. Cookies

For the first Cookies exercise, I accessed the website's application section using the inspect tools. I went to the cookies section and manually changed the value of the variable name, refreshing the page each time, until I reached the value 18, at which point the flag appeared.

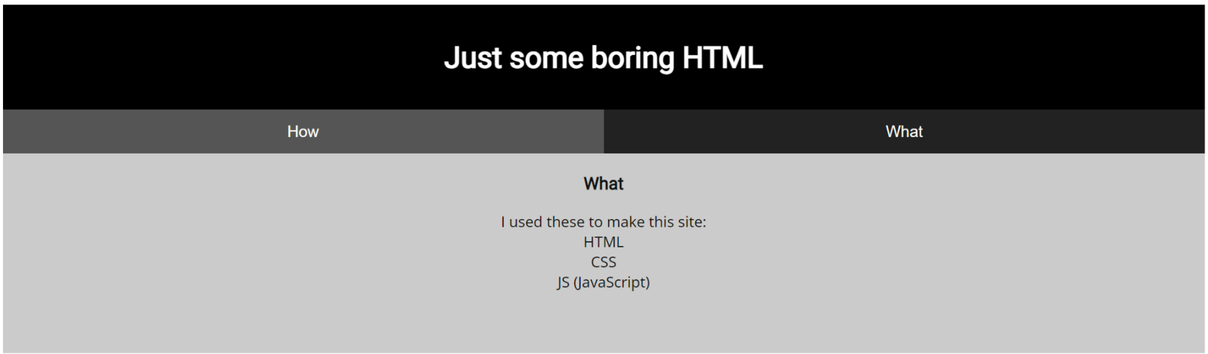


Flag:
`picoCTF{3v3ry1_l0v3s_c00k135_a1f5bdb7}`

Name	Value	Domain	Path	Expires / ...	Size	HttpOnly	Secure	SameSite	Partition K...	Priority
name	18	mercury.p...	/	Session	6					Medium

2. Scavenger Hunt

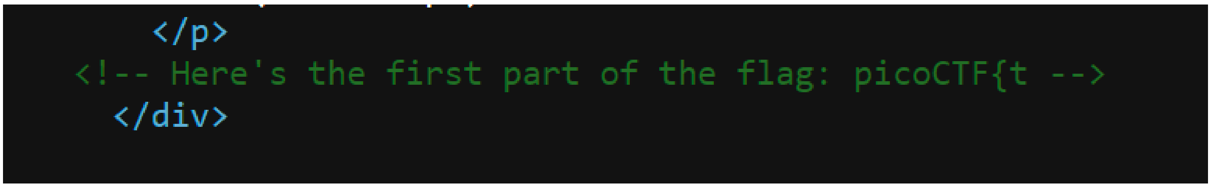
I started by examining the HTML, CSS, and JavaScript files on a webpage, where each file revealed a segment of the puzzle. Intriguingly, the JavaScript file led me to the robots.txt file, commonly used to direct search engine crawlers. Here, I found another piece of the flag and a hint to look into the .htaccess file, essential for setting access rules on Apache servers. My curiosity peaked as I discovered the next clue in the .htaccess file, directing me to the .DS_Store file, specific to Mac systems and storing folder configurations. Each step unveiled a part of the flag, culminating in the discovery of the complete flag: picoCTF{th4ts_4_l0t_0f_pl4c3s_2_lO0k_fa04427c}.



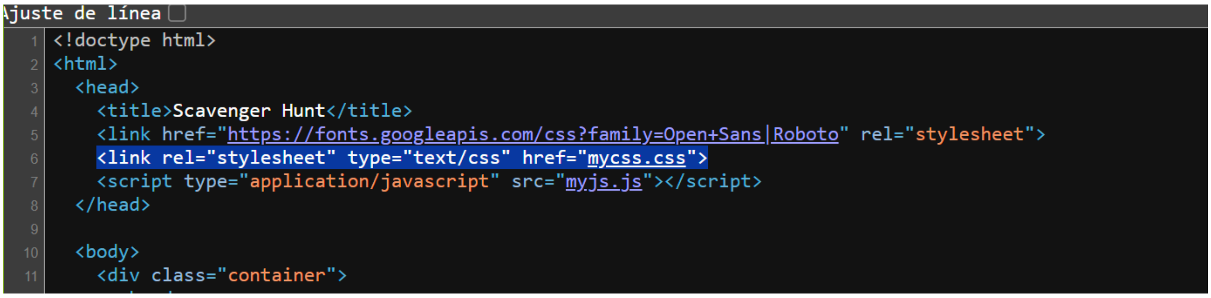
Just some boring HTML

What

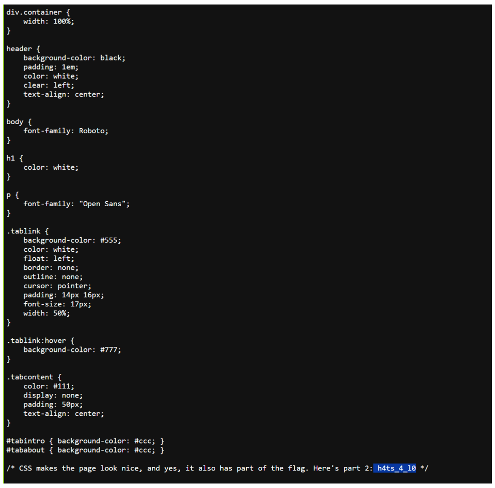
I used these to make this site:
HTML
CSS
JS (JavaScript)



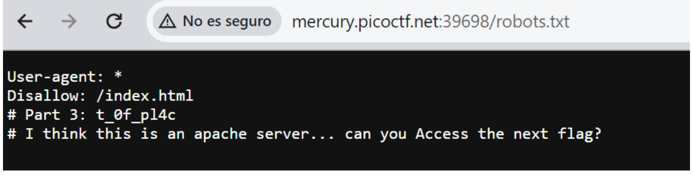
```
      </p>
  <!-- Here's the first part of the flag: picoCTF{t -->
    </div>
```



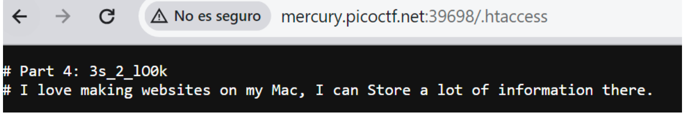
```
<!doctype html>
<html>
  <head>
    <title>Scavenger Hunt</title>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans|Roboto" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="mycss.css">
    <script type="application/javascript" src="myjs.js"></script>
  </head>

  <body>
    <div class="container">
```



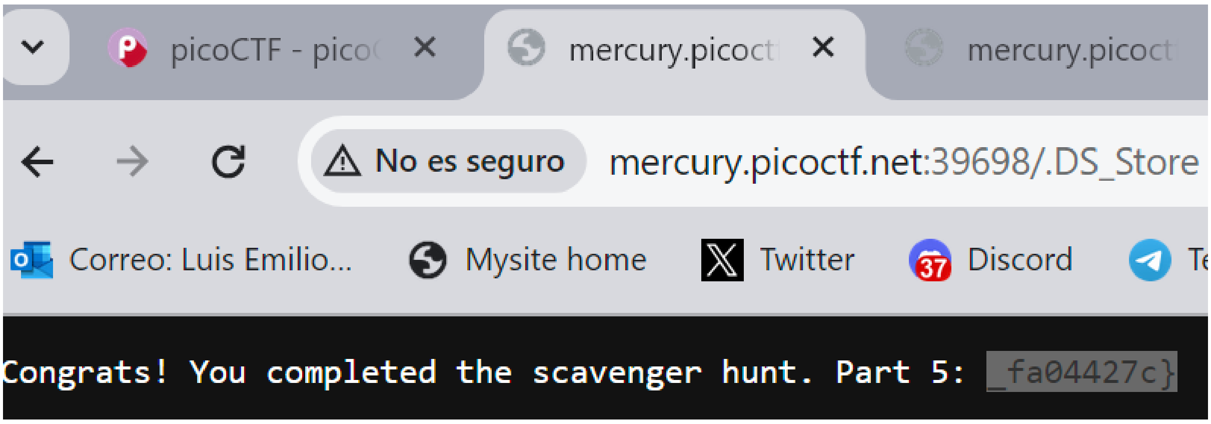
```
div.container {
    width: 100%;
}

header {
    background-color: black;
    padding: 1em;
    color: white;
    clear: left;
    text-align: center;
}

body {
    font-family: Roboto;
}

h1 {
    color: white;
}

p {
    font-family: "Open Sans";
}

.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 */
```



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



```
# Part 4: 3s_2_l0ok
# I love making websites on my Mac, I can Store a lot of information there.
```



```
Congrats! You completed the scavenger hunt. Part 5: _fa04427c}
```

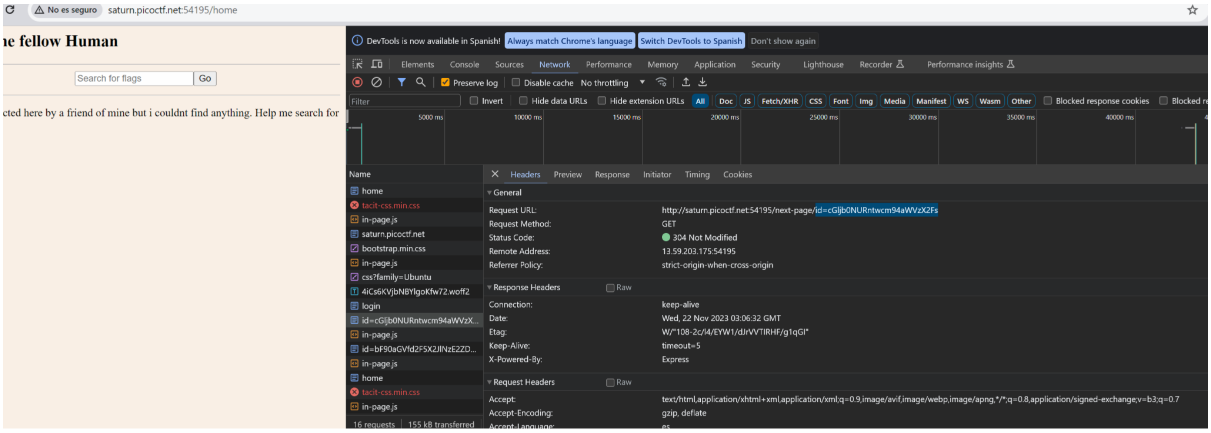
picoCTF{th4ts_4_l0t_0f_pl4c3s_2_lO0k_fa04427c}

3. findme

I first logged into a "login" page. This action automatically redirected me to a "home" page. Curious about the mechanics of this redirection, a concept I'd seen in picoCTF challenges, I decided to dig deeper. Using the web inspector tool, I focused on the network activity, particularly the transition sequence from the "login" to the "home" page.

This scrutiny revealed an interesting detail: the process involved passing through two intermediary pages, each identified by a unique URL. These URLs caught my attention due to their distinct structure, specifically the inclusion of an 'id' parameter.

Upon closer examination, I recognized that these IDs were likely encoded in Base64, hinted by the '==' at the end of one of them - a typical Base64 encoding feature. Intrigued by this, I decoded the Base64 IDs



No es seguro saturn.picoctf.net:54195/home

ıe fellow Human

Search for flags Go

cted here by a friend of mine but i couldnt find anything. Help me search for

DevTools is now available in Spanish! Always match Chrome's language Switch DevTools to Spanish Don't show again

☑ Preserve log ☐ Disable cache No throttling

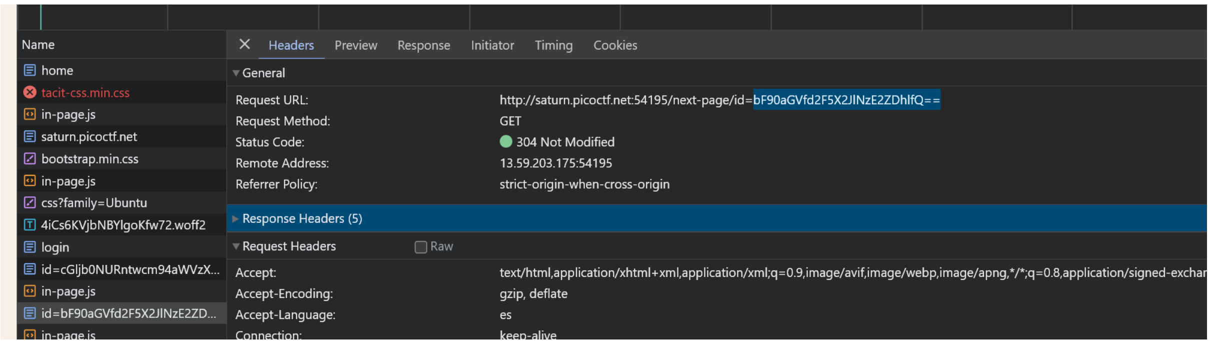
☐ Invert ☐ Hide data URLs ☐ Hide extension URLs All Doc JS Fetch/XHR CSS Font Img Media Manifest WS Wasm Other ☐ Blocked response cookies ☐ Blocked re

Name
home
tacit-css.min.css
in-page.js
saturn.picoctf.net
bootstrap.min.css
in-page.js
css?family=Ubuntu
4iCs6KVjbNBYlgoKfw72.woff2
login
id=cGljb0NURntwcm94aWVzX...
in-page.js
id=bF90aGVfd2F5X2JlNzE2ZD...
in-page.js
home
tacit-css.min.css
in-page.js
16 requests | 155 kB transferred

General
Request URL: http://saturn.picoctf.net:54195/next-page/id=cGljb0NURntwcm94aWVzX2Fs
Request Method: GET
Status Code: 304 Not Modified
Remote Address: 13.59.203.175:54195
Referrer Policy: strict-origin-when-cross-origin

Response Headers ☐ Raw
Connection: keep-alive
Date: Wed, 22 Nov 2023 03:06:32 GMT
Etag: W/"108-2c/l4/EYW1/dJrVVTIRHF/g1qGI"
Keep-Alive: timeout=5
X-Powered-By: Express

Request Headers ☐ Raw
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: es

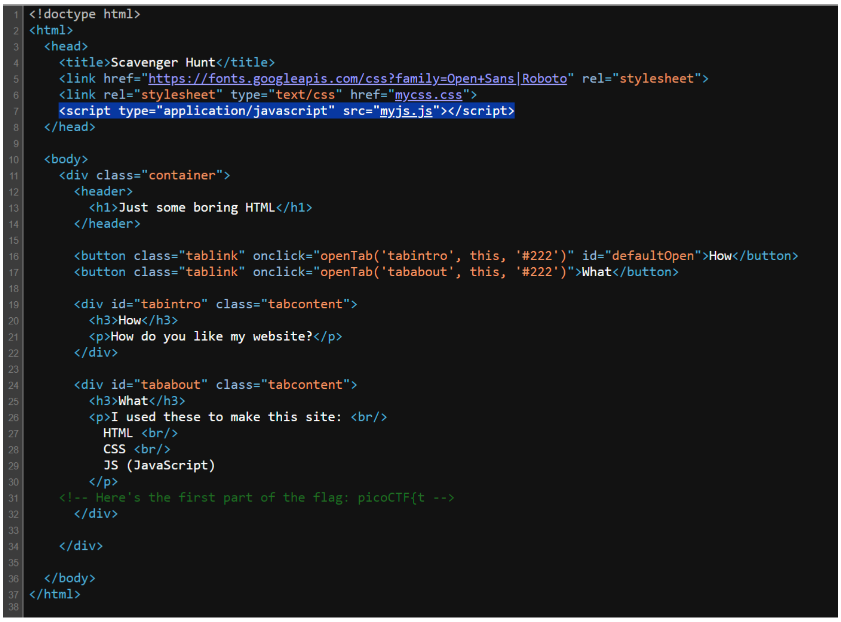


Name
home
tacit-css.min.css
in-page.js
saturn.picoctf.net
bootstrap.min.css
in-page.js
css?family=Ubuntu
4iCs6KVjbNBYlgoKfw72.woff2
login
id=cGljb0NURntwcm94aWVzX...
in-page.js
id=bF90aGVfd2F5X2JlNzE2ZD...
in-page.js

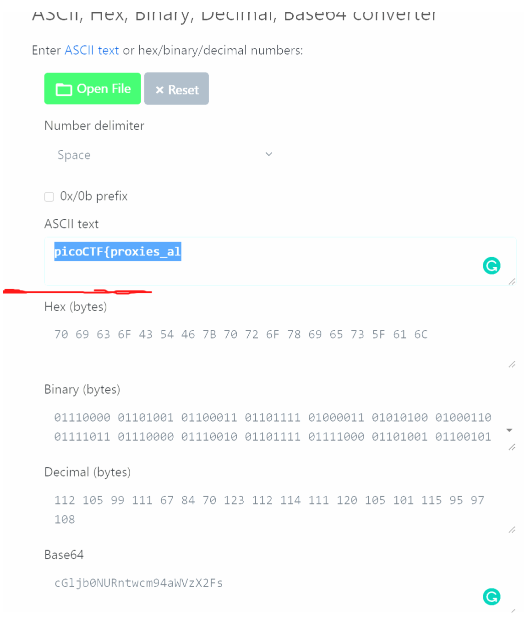
General
Request URL: http://saturn.picoctf.net:54195/next-page/id=bF90aGVfd2F5X2JlNzE2ZDhlfQ==
Request Method: GET
Status Code: 304 Not Modified
Remote Address: 13.59.203.175:54195
Referrer Policy: strict-origin-when-cross-origin

Response Headers (5)

Request Headers ☐ Raw
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchar
Accept-Encoding: gzip, deflate
Accept-Language: es
Connection: keep-alive



```
<!doctype html>
<html>
  <head>
    <title>Scavenger Hunt</title>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans|Roboto" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="mycss.css">
    <script type="application/javascript" src="myjs.js"></script>
  </head>

  <body>
    <div class="container">
      <header>
        <h1>Just some boring HTML</h1>
      </header>

      <button class="tablink" onclick="openTab('tabintro', this, '#222')" id="defaultOpen">How</button>
      <button class="tablink" onclick="openTab('tababout', this, '#222')">What</button>

      <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>
```



ASCII, Hex, Binary, Decimal, Base64 converter

Enter ASCII text or hex/binary/decimal numbers:

Number delimiter

Space

□ 0x/0b prefix

ASCII text

```
picoCTF{proxies_al
```

Hex (bytes)

```
70 69 63 6F 43 54 46 7B 70 72 6F 78 69 65 73 5F 61 6C
```

Binary (bytes)

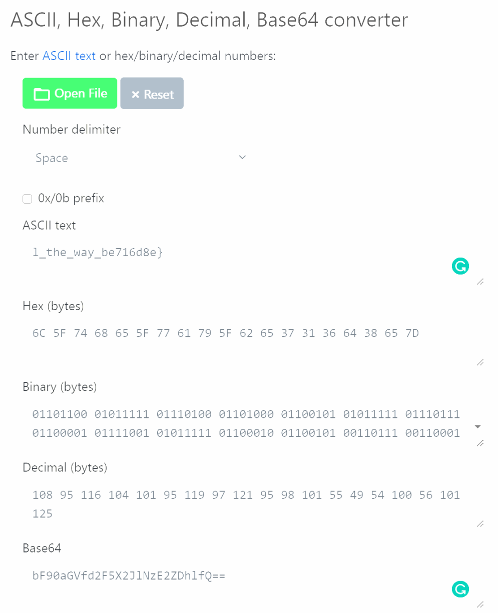
```
01110000 01101001 01100011 01101111 01000011 01010100 01000110
01111011 01110000 01110010 01101111 01111000 01101001 01100101
```

Decimal (bytes)

```
112 105 99 111 67 84 70 123 112 114 111 120 105 101 115 95 97
108
```

Base64

```
cGljb0NURntwcm94aWVzX2Fs
```



ASCII, Hex, Binary, Decimal, Base64 converter

Enter ASCII text or hex/binary/decimal numbers:

Number delimiter

Space

□ 0x/0b prefix

ASCII text

```
l_the_way_be716d8e}
```

Hex (bytes)

```
6C 5F 74 68 65 5F 77 61 79 5F 62 65 37 31 36 64 38 65 7D
```

Binary (bytes)

```
01101100 01011111 01110100 01101000 01100101 01011111 01110111
01100001 01111001 01011111 01100010 01100101 00110111 00110001
```

Decimal (bytes)

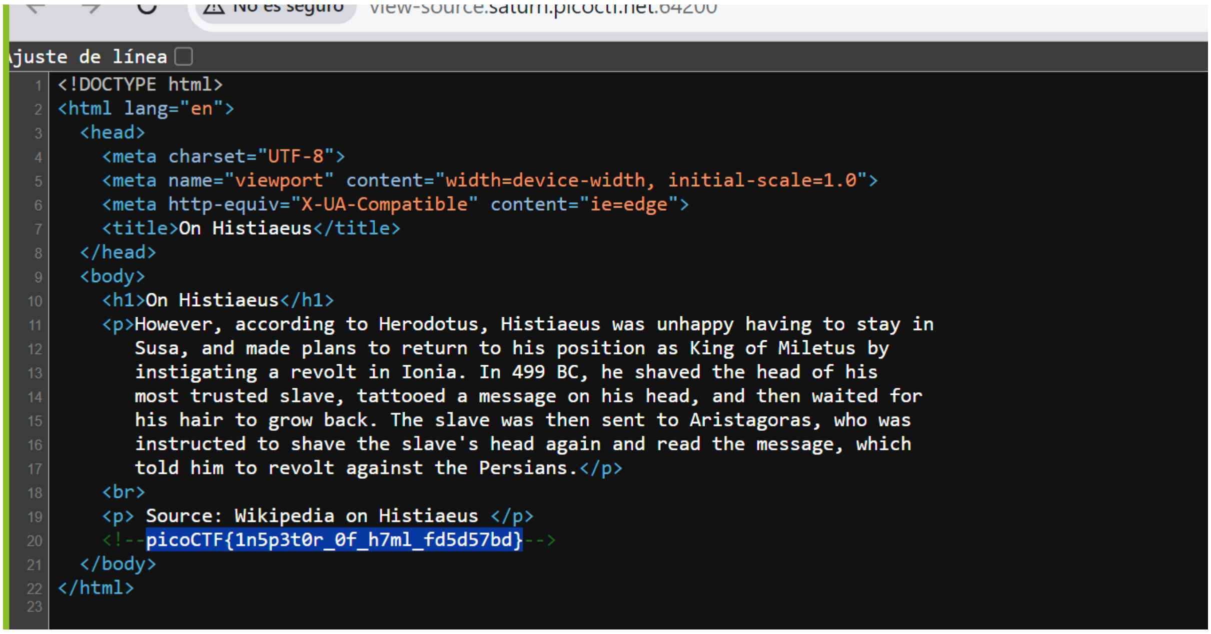
```
108 95 116 104 101 95 119 97 121 95 98 101 55 49 54 100 56 101
125
```

Base64

```
bF90aGVfd2F5X2JlNzE2ZDhlfQ==
```

4. Inspect HTML

   I just inspect the html

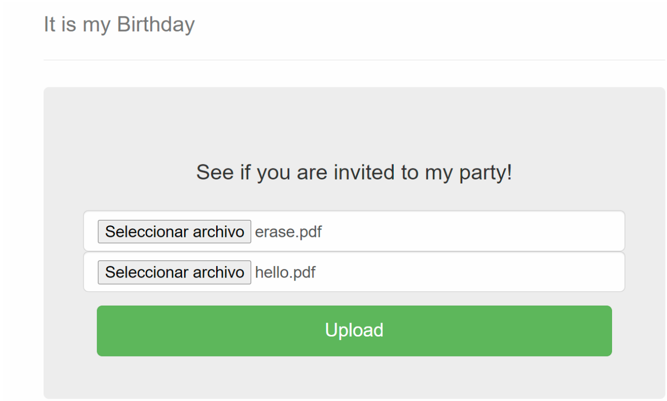


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>On Histiaeus</title>
  </head>
  <body>
    <h1>On Histiaeus</h1>
    <p>However, according to Herodotus, Histiaeus was unhappy having to stay in
       Susa, and made plans to return to his position as King of Miletus by
       instigating a revolt in Ionia. In 499 BC, he shaved the head of his
       most trusted slave, tattooed a message on his head, and then waited for
       his hair to grow back. The slave was then sent to Aristagoras, who was
       instructed to shave the slave's head again and read the message, which
       told him to revolt against the Persians.</p>
    <br>
    <p> Source: Wikipedia on Histiaeus </p>
    <!--picoCTF{1n5p3t0r_0f_h7ml_fd5d57bd}-->
  </body>
</html>
```

5. **It is my Birthday**

   To tackle the exercise, I utilized two unique PDF documents, named "erase.pdf" and "hello.pdf," which I discovered through an online search. These documents were intriguing because, despite their differences, they shared an identical MD5 hash. This similarity was due to a precalculated MD5 hash collision, a concept that perfectly aligned with the requirements of the challenge I was working on.

   Since the MD5 collision was already in place, I didn't need to make any modifications to these files. They were ready to be used as they were. I downloaded both "erase.pdf" and "hello.pdf" and then proceeded to upload them to the specific website that was part of the challenge.

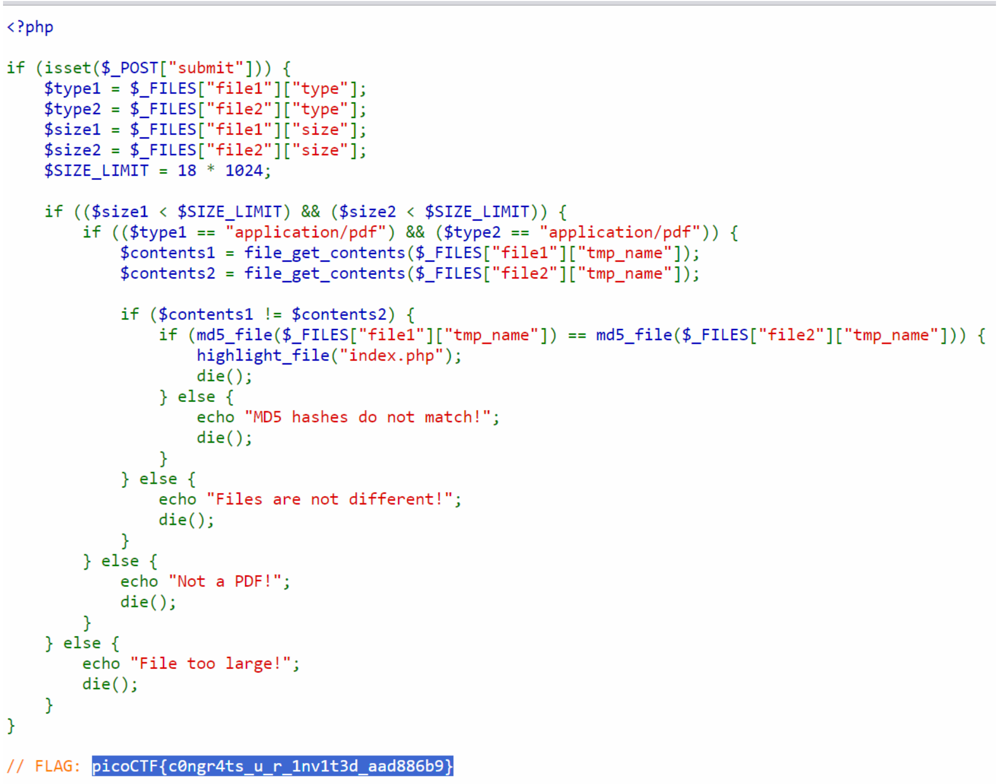


It is my Birthday

See if you are invited to my party!

Seleccionar archivo erase.pdf

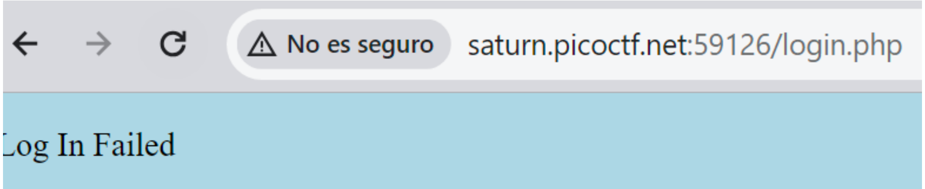
Seleccionar archivo hello.pdf



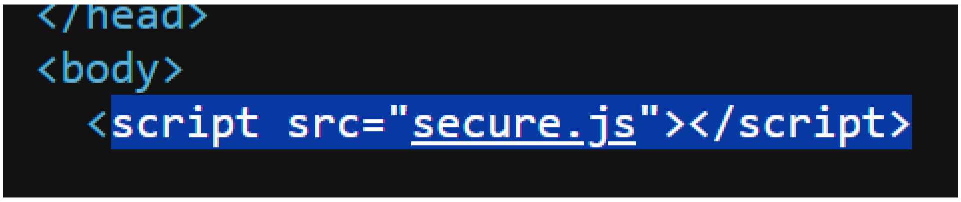
```php
<?php

if (isset($_POST["submit"])) {
    $type1 = $_FILES["file1"]["type"];
    $type2 = $_FILES["file2"]["type"];
    $size1 = $_FILES["file1"]["size"];
    $size2 = $_FILES["file2"]["size"];
    $SIZE_LIMIT = 18 * 1024;

    if (($size1 < $SIZE_LIMIT) && ($size2 < $SIZE_LIMIT)) {
        if (($type1 == "application/pdf") && ($type2 == "application/pdf")) {
            $contents1 = file_get_contents($_FILES["file1"]["tmp_name"]);
            $contents2 = file_get_contents($_FILES["file2"]["tmp_name"]);

            if ($contents1 != $contents2) {
                if (md5_file($_FILES["file1"]["tmp_name"]) == md5_file($_FILES["file2"]["tmp_name"])) {
                    highlight_file("index.php");
                    die();
                } else {
                    echo "MD5 hashes do not match!";
                    die();
                }
            } else {
                echo "Files are not different!";
                die();
            }
        } else {
            echo "Not a PDF!";
            die();
        }
    } else {
        echo "File too large!";
        die();
    }
}

// FLAG: picoCTF{c0ngr4ts_u_r_1nv1t3d_aad886b9}
```

6. Local Authority

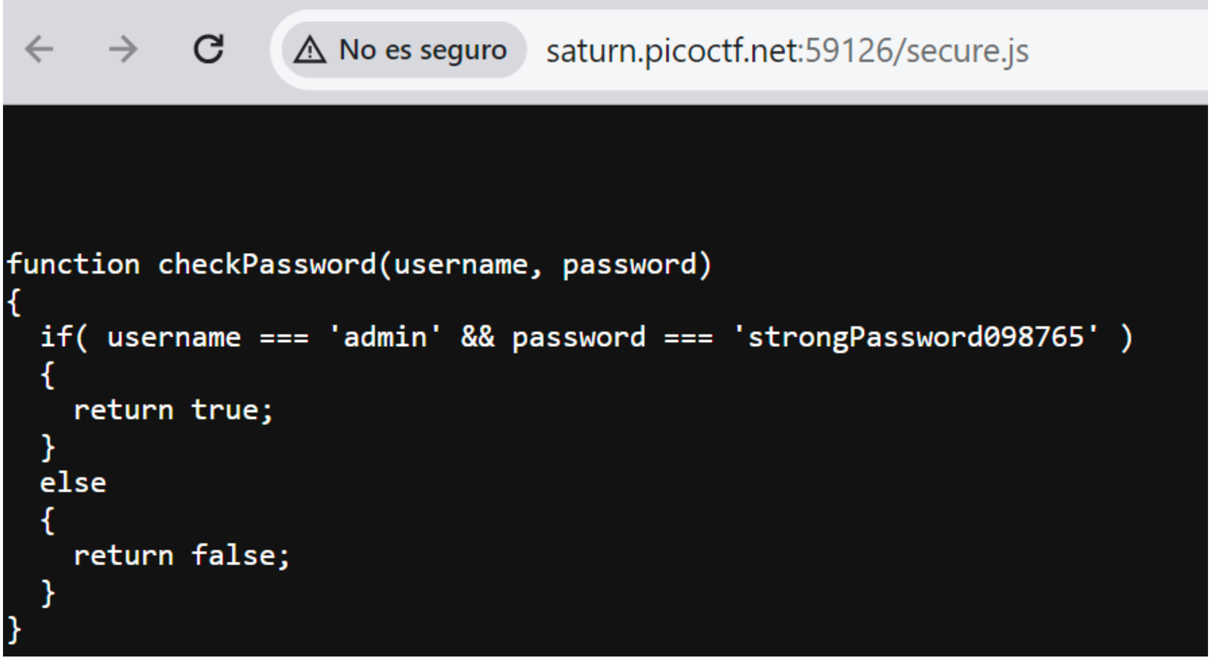
In solving the challenge, I focused on understanding the mechanics of password verification. This required inspecting the webpage closely, where I discovered a PHP file linked to the login process. Testing this, I logged in with arbitrary credentials, which predictably resulted in a redirection to the PHP file and an accompanying error message. This redirection was key, as it offered a potential insight into the password verification method. However, I also found a 'secure.js' file on the site. Intriguingly, this file contained the actual username and password needed for the challenge. Armed with this information, I returned to the login portal, entered the correct credentials, and was successfully redirected to the page displaying the flag.



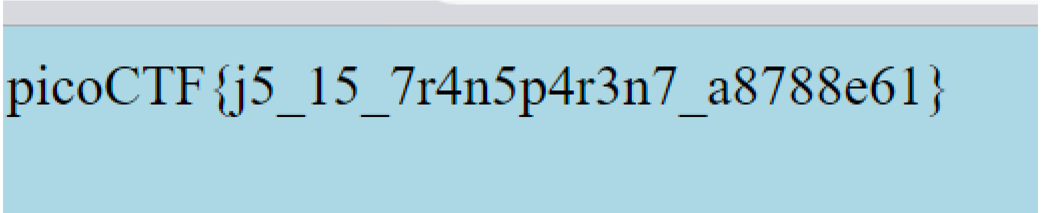
Log In Failed



```html
</head>
<body>
  <script src="secure.js"></script>
```



```
function checkPassword(username, password)
{
  if( username === 'admin' && password === 'strongPassword098765' )
  {
    return true;
  }
  else
  {
    return false;
  }
}
```



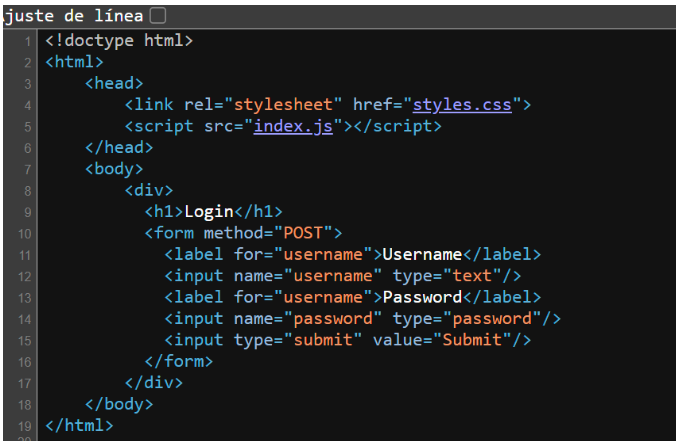
picoCTF{j5_15_7r4n5p4r3n7_a8788e61}

7. Login

I scrutinized the website's source code and zeroed in on a particular JavaScript file, which seemed pivotal for the login mechanism. Delving into the file, I found a function written to handle the events of a web form. This function was designed to intercept the form submission, encode the entered username and password into base-64 format, and then check these against pre-set values.

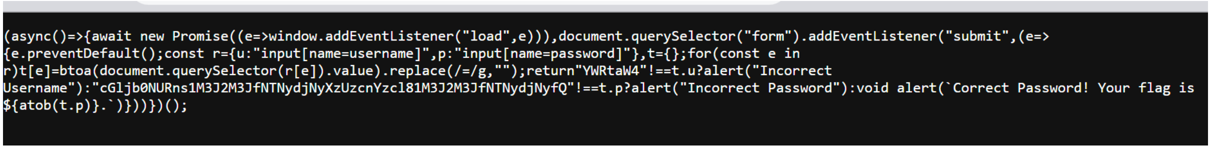
The comparison was straightforward: if the encoded username wasn't equivalent to a specific base-64 string, an alert for "Incorrect Username" would be triggered. Similarly, if the encoded password didn't match its corresponding pre-set base-64 string, it would prompt an "Incorrect Password" alert. However, in the event of a match, the function would decode the password, revealing the flag.

I realized that I could leverage this function by decoding the provided base-64 strings to reveal the correct username and password, which would lead me directly to the flag.

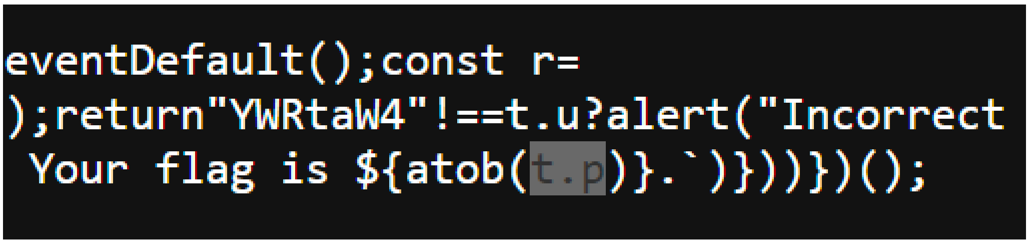


juste de línea □

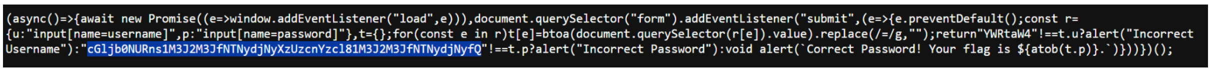
```html
<!doctype html>
<html>
    <head>
        <link rel="stylesheet" href="styles.css">
        <script src="index.js"></script>
    </head>
    <body>
        <div>
          <h1>Login</h1>
          <form method="POST">
            <label for="username">Username</label>
            <input name="username"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



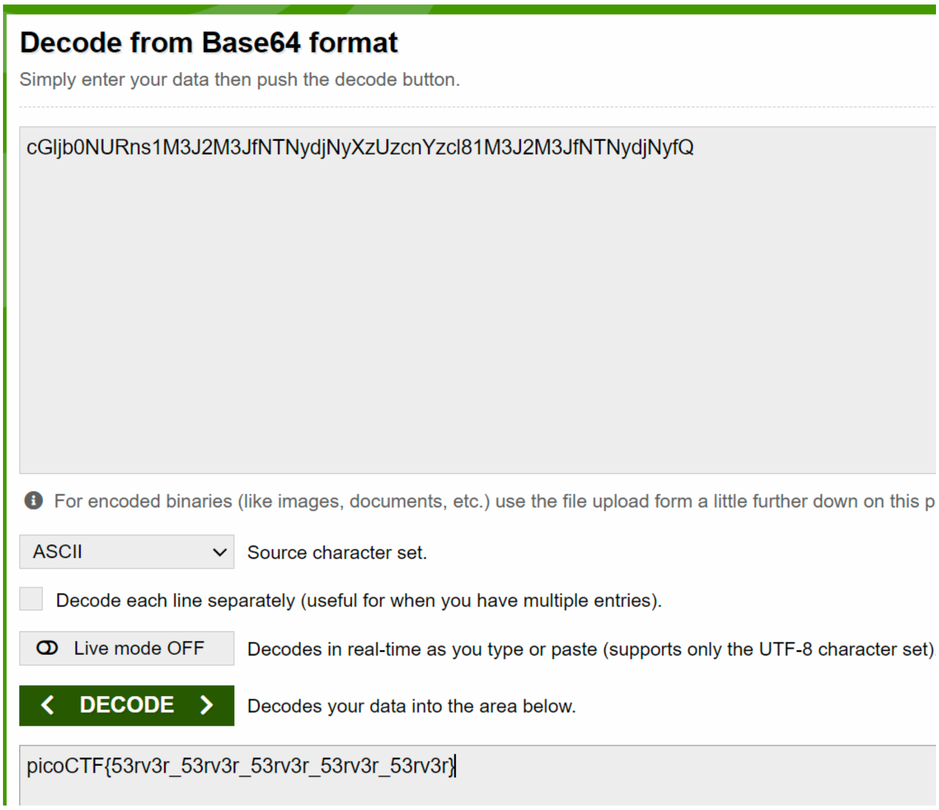
```
(async()=>{await new Promise((e=>window.addEventListener("load",e))),document.querySelector("form").addEventListener("submit",(e=>
{e.preventDefault();const r={u:"input[name=username]",p:"input[name=password]"},t={};for(const e in
r)t[e]=btoa(document.querySelector(r[e]).value).replace(/=/g,"");return"YWRtaW4"!==t.u?alert("Incorrect
Username"):"cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ"!==t.p?alert("Incorrect Password"):void alert(`Correct Password! Your flag is
${atob(t.p)}.`)}))})();
```



```
eventDefault();const r=
);return"YWRtaW4"!==t.u?alert("Incorrect
 Your flag is ${atob(t.p)}.`)}))})();
```



```
(async()=>{await new Promise((e=>window.addEventListener("load",e))),document.querySelector("form").addEventListener("submit",(e=>{e.preventDefault();const r=
{u:"input[name=username]",p:"input[name=password]"},t={};for(const e in r)t[e]=btoa(document.querySelector(r[e]).value).replace(/=/g,"");return"YWRtaW4"!==t.u?alert("Incorrect
Username"):"cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ"!==t.p?alert("Incorrect Password"):void alert(`Correct Password! Your flag is ${atob(t.p)}.`)}))})();
```



Decode from Base64 format

Simply enter your data then push the decode button.

cGljb0NURns1M3J2M3JfNTNydjNyXzUzcnYzcl81M3J2M3JfNTNydjNyfQ

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this p

ASCII — Source character set.

□ Decode each line separately (useful for when you have multiple entries).

Decodes in real-time as you type or paste (supports only the UTF-8 character set)

Decodes your data into the area below.

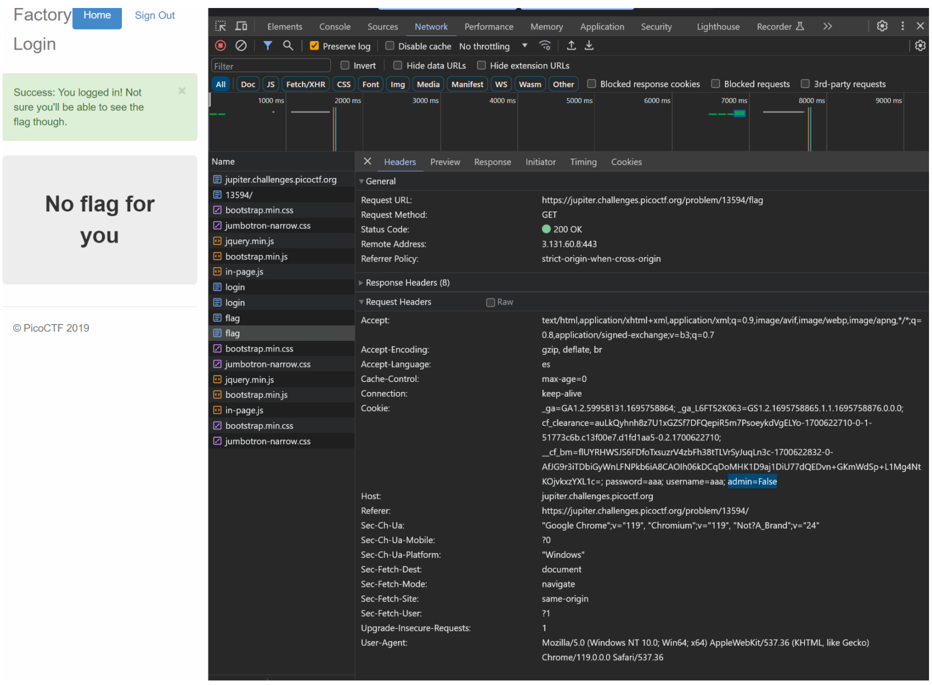
picoCTF{53rv3r_53rv3r_53rv3r_53rv3r_53rv3r}

8. Logon

During the logon challenge, I initially input arbitrary credentials on the login screen. Following this, I found myself on the /flag page, which confirmed my successful login but indicated that no flag was available for me. To understand why, I initiated a deep dive into the page's inspection tools.

My exploration led me to the network section, where, nested within the /flag page's details, I stumbled upon a cookie setting that defined the 'admin' as false. Recognizing that this setting was likely preventing me from accessing the flag, I navigated to the Applications tab within the inspection tools. There, I located the cookies pertaining to the domain in question.

Within the cookies details, I modified the 'admin' value from false to true. With this change made, I refreshed the /flag page. To my satisfaction, the elusive flag was now presented to me, successfully concluding the challenge.



Success: You logged in! Not sure you'll be able to see the flag though.

No flag for you

© PicoCTF 2019

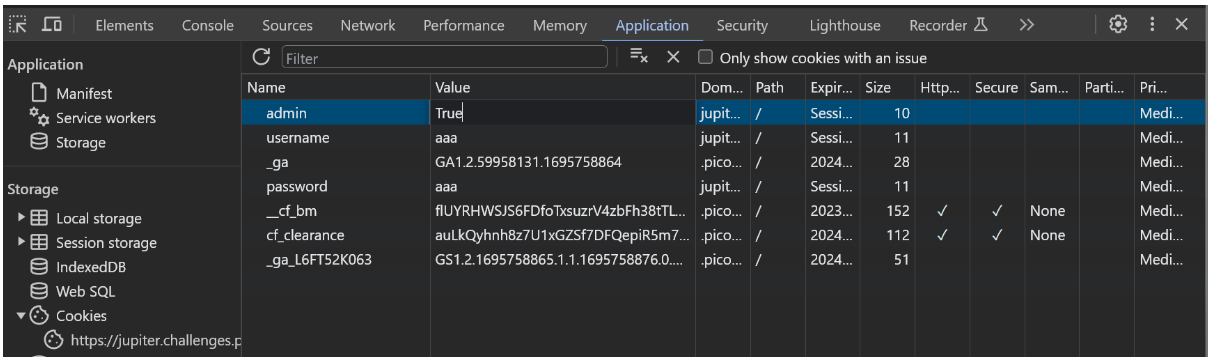
General

Request URL:	https://jupiter.challenges.picoctf.org/problem/13594/flag
Request Method:	GET
Status Code:	● 200 OK
Remote Address:	3.131.60.8:443
Referrer Policy:	strict-origin-when-cross-origin

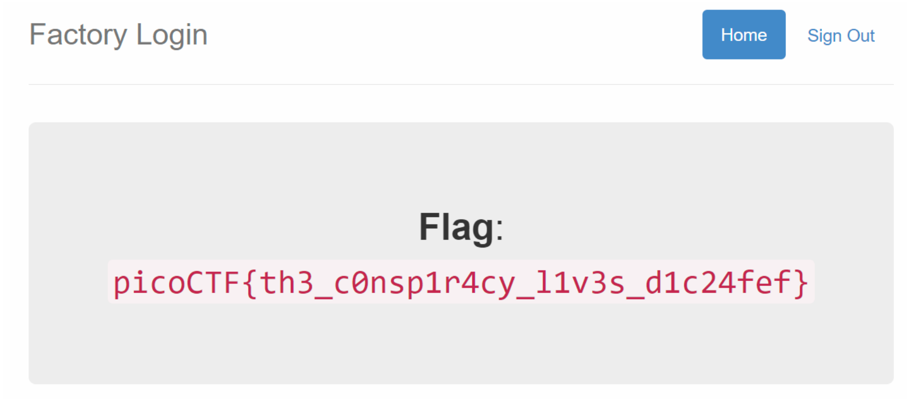
Response Headers (8)

Request Headers

Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding:	gzip, deflate, br
Accept-Language:	es
Cache-Control:	max-age=0
Connection:	keep-alive
Cookie:	_ga=GA1.2.59958131.1695758864; _ga_L6FT52K063=GS1.2.1695758865.1.1.1695758876.0.0.0; cf_clearance=auLkQyhnh8z7U1xGZSf7DFQepiR5m7PsoeykdVgELYo-1700622710-0-1-51773c6b.c13f00e7.d1fd1aa5-0.2.1700622710; __cf_bm=flUYRHWSJS6FDfoTxsuzrV4zbFh38tTLVrSyJuqLn3c-1700622832-0-AfJG9r3iTDbiGyWnLFNPkb6iA8CAOIh06kDCqDoMHK1D9aj1DiU77dQEDvn+GKmWdSp+L1Mg4NtKOjvkxzYXL1c=; password=aaa; username=aaa; admin=False
Host:	jupiter.challenges.picoctf.org
Referer:	https://jupiter.challenges.picoctf.org/problem/13594/
Sec-Ch-Ua:	"Google Chrome";v="119", "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile:	?0
Sec-Ch-Ua-Platform:	"Windows"
Sec-Fetch-Dest:	document
Sec-Fetch-Mode:	navigate
Sec-Fetch-Site:	same-origin
Sec-Fetch-User:	?1
Upgrade-Insecure-Requests:	1
User-Agent:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36



Name	Value	Dom...	Path	Expir...	Size	Http...	Secure	Sam...	Parti...	Pri...
admin	True	jupit...	/	Sessi...	10					Medi...
username	aaa	jupit...	/	Sessi...	11					Medi...
_ga	GA1.2.59958131.1695758864	.pico...	/	2024...	28					Medi...
password	aaa	jupit...	/	Sessi...	11					Medi...
__cf_bm	flUYRHWSJS6FDfoTxsuzrV4zbFh38tTL...	.pico...	/	2023...	152	✓	✓	None		Medi...
cf_clearance	auLkQyhnh8z7U1xGZSf7DFQepiR5m7...	.pico...	/	2024...	112	✓	✓	None		Medi...
_ga_L6FT52K063	GS1.2.1695758865.1.1.1695758876.0....	.pico...	/	2024...	51					Medi...

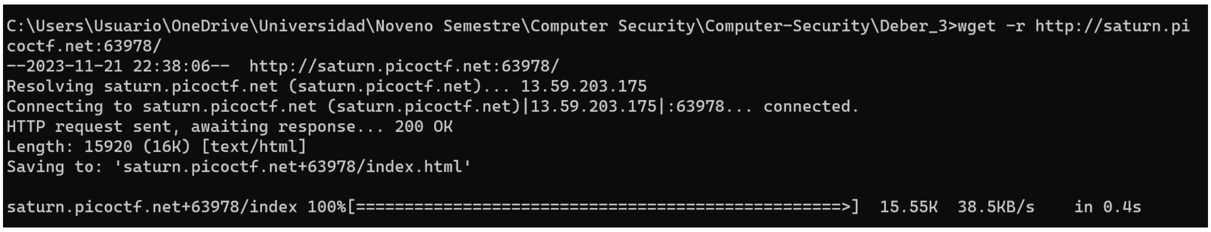


Flag:
picoCTF{th3_c0nsp1r4cy_l1v3s_d1c24fef}

9. Search source

To approach the challenge more effectively, I decided to create a local mirror of the website. This would allow me to utilize advanced search capabilities not feasible within the constraints of the web interface.

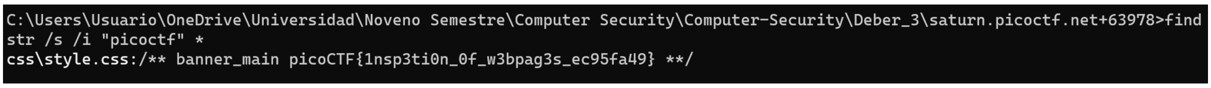
I opened my terminal and used the wget command with the recursive option to pull all the content from the website onto my machine:



```
C:\Users\Usuario\OneDrive\Universidad\Noveno Semestre\Computer Security\Computer-Security\Deber_3>wget -r http://saturn.pi
coctf.net:63978/
--2023-11-21 22:38:06--  http://saturn.picoctf.net:63978/
Resolving saturn.picoctf.net (saturn.picoctf.net)... 13.59.203.175
Connecting to saturn.picoctf.net (saturn.picoctf.net)|13.59.203.175|:63978... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15920 (16K) [text/html]
Saving to: 'saturn.picoctf.net+63978/index.html'

saturn.picoctf.net+63978/index 100%[================================================>]  15.55K  38.5KB/s    in 0.4s
```

This command methodically fetched the website's files and directories, replicating them locally. After the download was complete, I navigated to the local directory that now housed the website's files.

I then executed a search command tailored for the task:



```
C:\Users\Usuario\OneDrive\Universidad\Noveno Semestre\Computer Security\Computer-Security\Deber_3\saturn.picoctf.net+63978>find
str /s /i "picoctf" *
css\style.css:/** banner_main picoCTF{1nsp3ti0n_0f_w3bpag3s_ec95fa49} **/
```

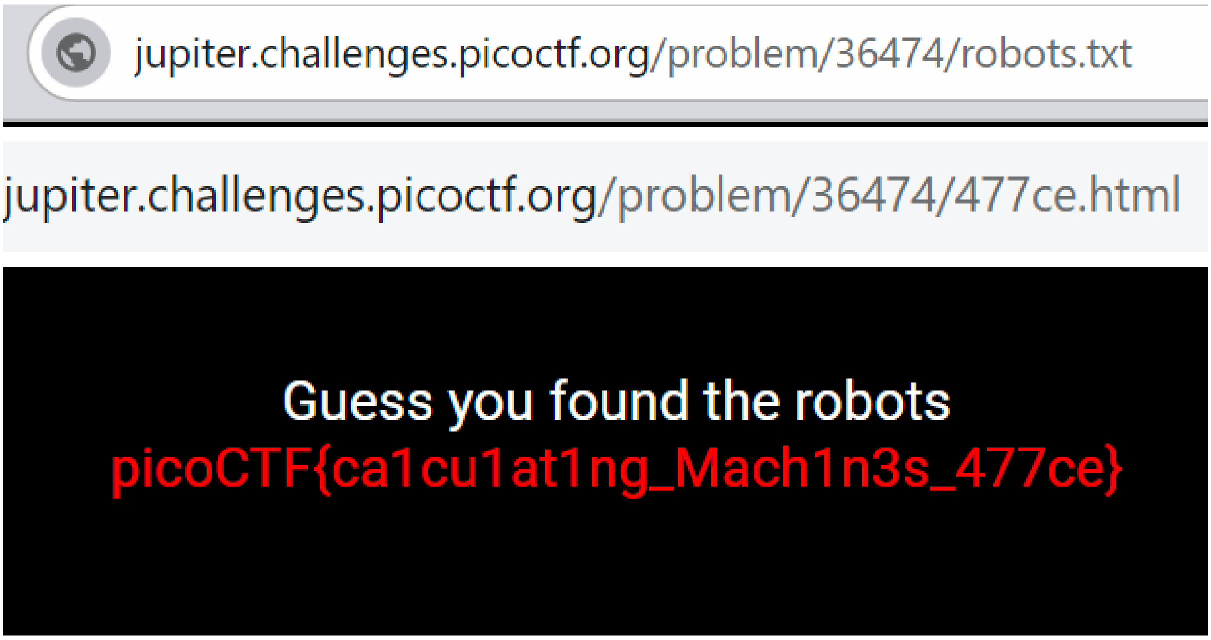
This command combed through every file and subdirectory in the local copy of the website, looking for instances of the string "pico" irrespective of the case. The search was thorough, and it led to the discovery of the flag, which was also accompanied by details on its original location on the website. This method of mirroring the website and searching locally proved to be a decisive strategy in uncovering the hidden information.

10. where are the robots

Upon receiving the page's message suggesting a search for 'robots', I considered the robots.txt file, a standard on websites that guides search engines on what paths to exclude from their indexing. To inspect this, I simply added '/robots.txt' to the base URL.

In the robots.txt file, I identified a disallowed path which indicated a place on the website the creator preferred to remain unindexed. To follow this clue, I modified the URL, replacing '/robots.txt' with the specific path I found, which was '/477c.html'.

When I navigated to this updated URL, it led me to the intended hidden location, thus uncovering the flag. The robots.txt file played its part perfectly, pointing me to where the website's secrets, or 'robots', were kept.



jupiter.challenges.picoctf.org/problem/36474/robots.txt

jupiter.challenges.picoctf.org/problem/36474/477ce.html

Guess you found the robots

picoCTF{ca1cu1at1ng_Mach1n3s_477ce}