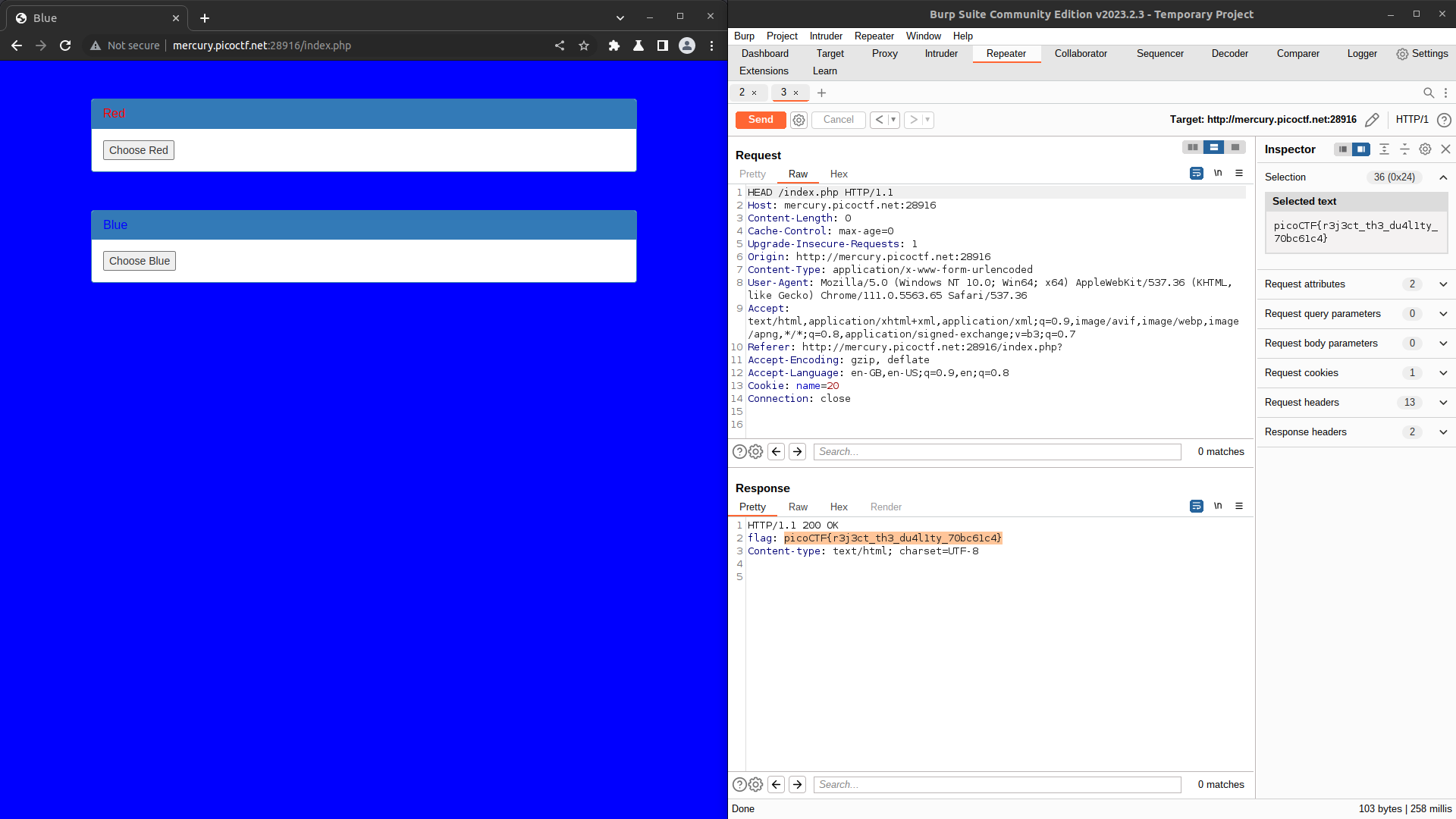
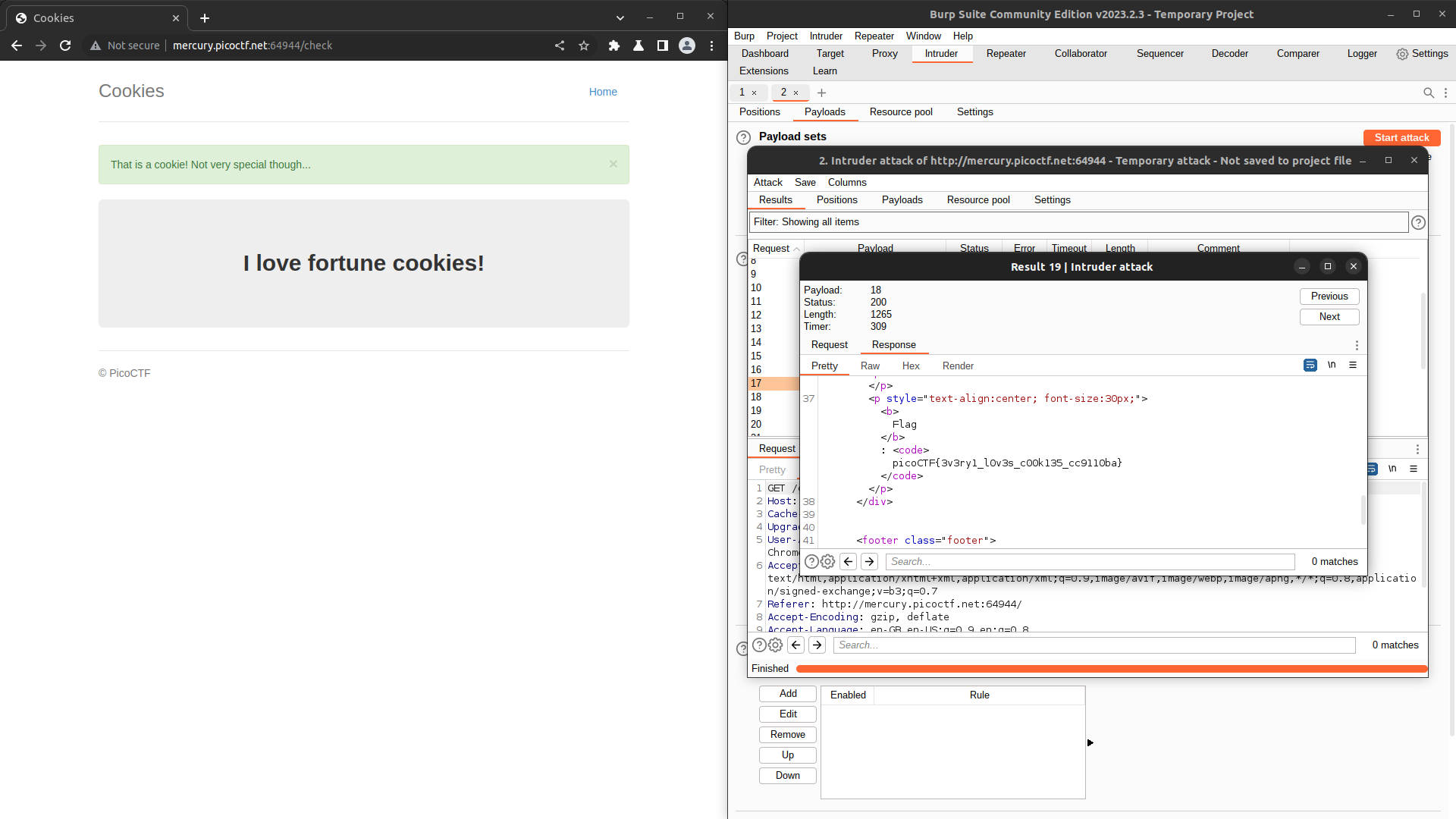
**GET aHEAD**

1. Given the hit, I opened up the url in burpsuite and intercepted both the requests – GET & POST.

2. Learning about different requests and with the help of a writeup, found out that you can change the header request to HEAD which was obvious from thetitle and sending it back would give you the flag.

**COOKIES**

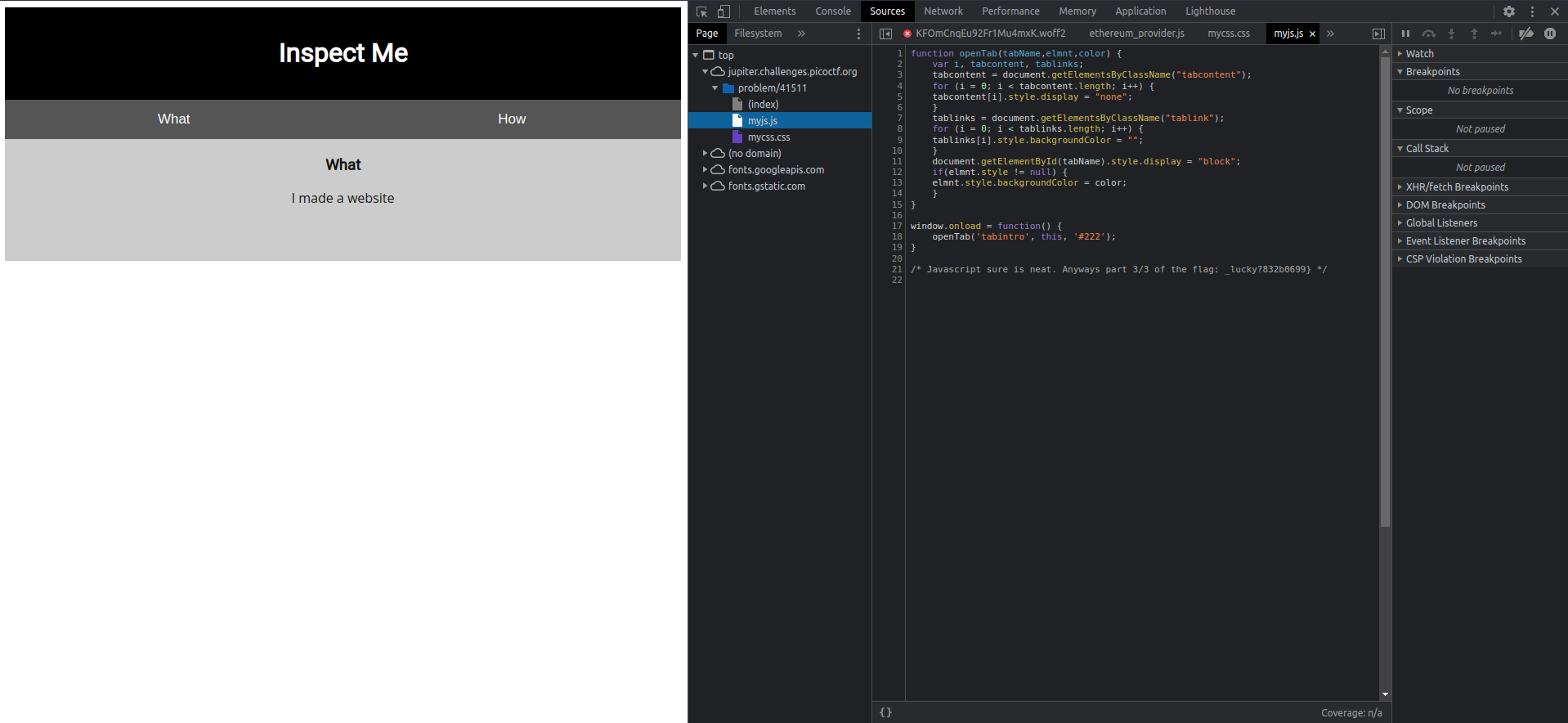
1. Given the title, it had to be about cookies which is confirmed after opening the cookie editor and changing its values.

2. Using a bit of hints from other writeups, it was clear that different kind of cookies is assigned different cookie value.

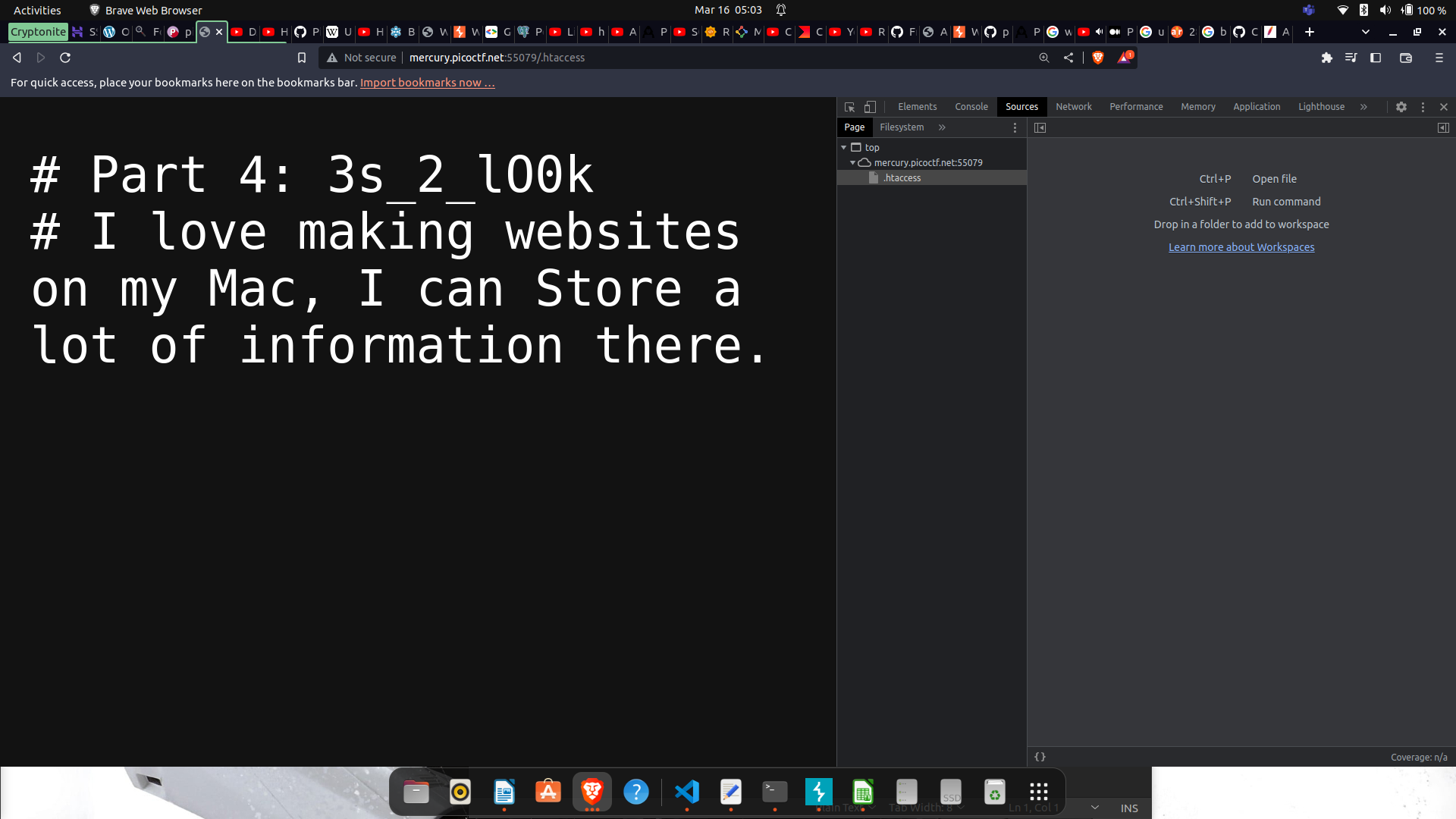
3. Instead of checking it manually, I used BurpSuite to intercept and automatically check for all cookies, which got me few abnormal results.

4. One of this was cookie value=18 which got me the flag.

**INSP3CT0R**

1. A simple task which just required me to inspectthrough the source files – index, mycss.css, myjs.js which had different parts of the flag commented out.

**Scavenger Hunt**

1. The first two flags can be found in the html and css file.

2. The js file containedan hint on web crawlers which got me the third part of the flag by accessing the robots.txt file.

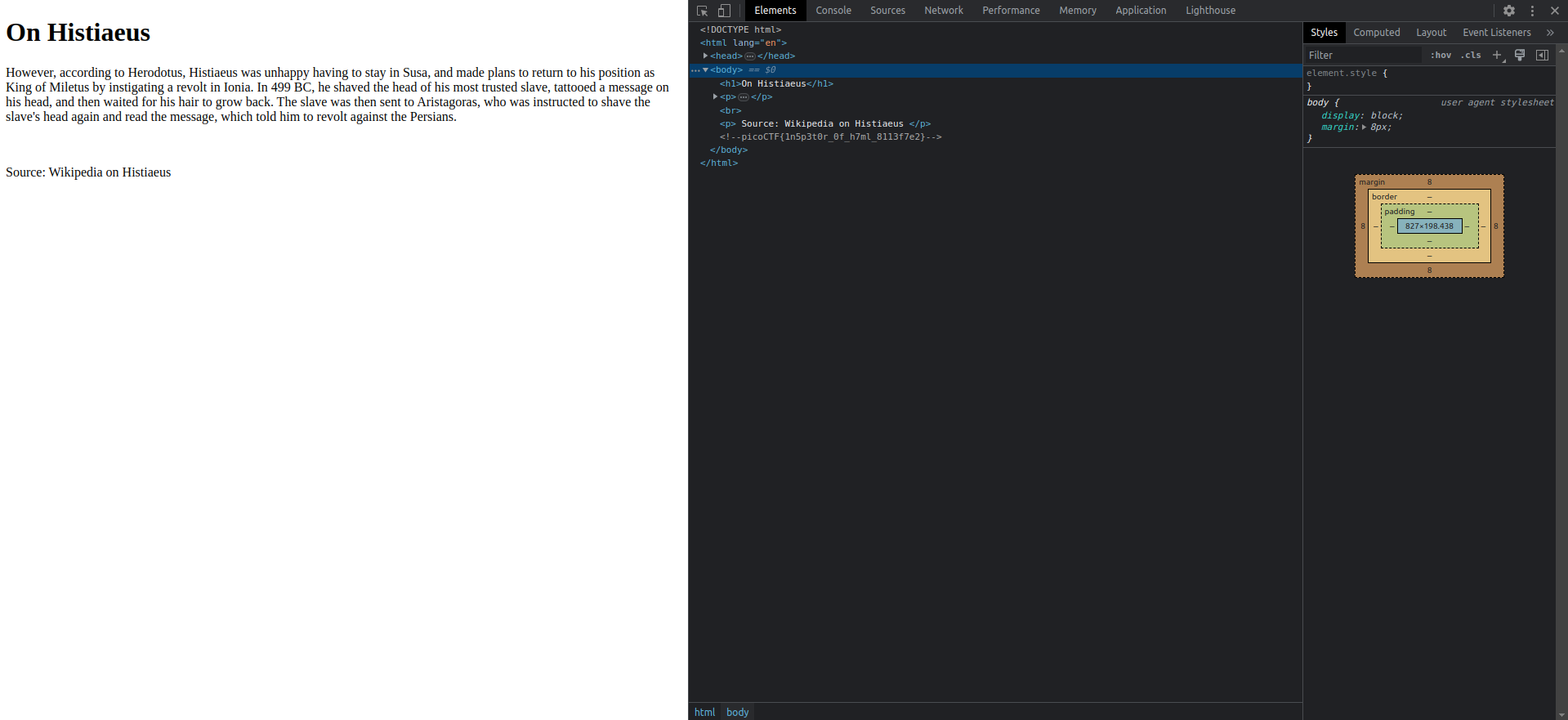
3. The next hint on apache server got me to search up its access file term (.htaccess) and accessing this got me the fourth part of the flag.

4. Onto the next hint, had to look up for this one, we gotta access the DS\_Store which is a common macos file which stores the attributes of its containing folder.

5. Accessing .DS\_Store give us the last part of the flag.

**Some Assembly Required 1**

**Inspect HTML**

****

The flag can be found by inspecting the html source code.

**Power Cookie**

1. The website leads you to a basic html page with an option “Continue as Guest”

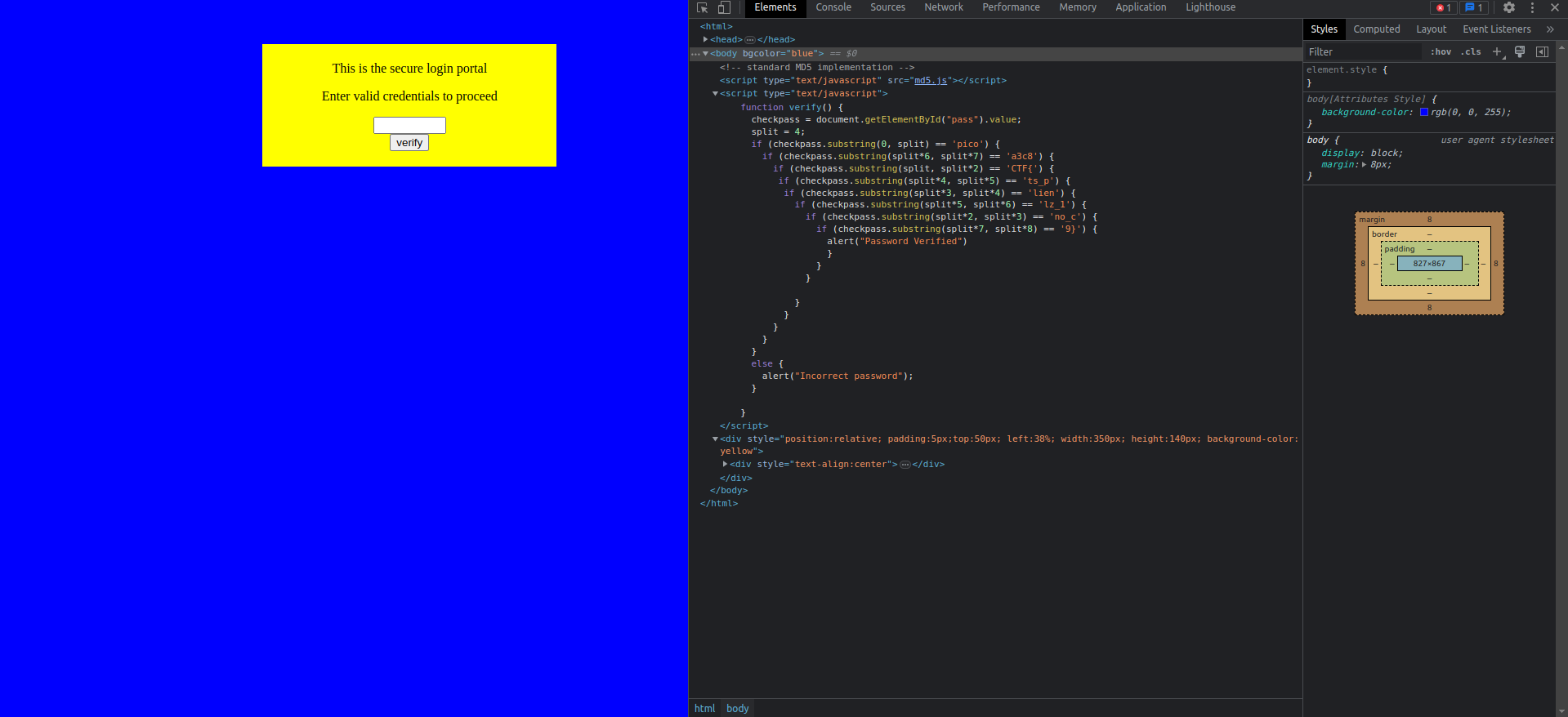
2. Looking at the question, a hint is given in form of “Power Cookie”.

3. So as we search for cookies now, we find none.

4. Although a cookie “”isAdmin” pops out as you click on “Continue as Guest”.

5. If we change the value of “isAdmin” to ‘1’, we get the flag required.

**Don’t use client side**



1. Visiting the website greets us with a login page.

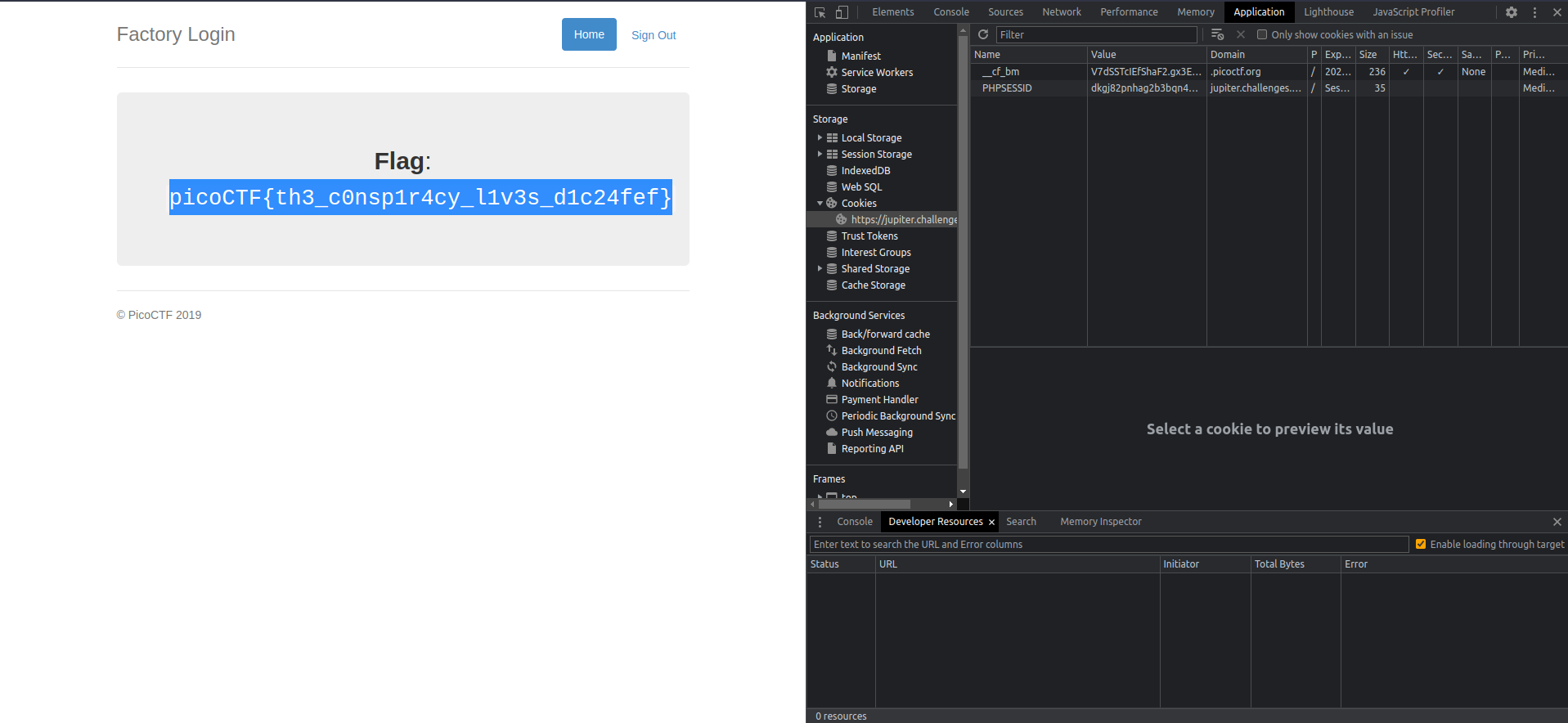
2. We can assume the credentials to be verified locally.

3. It can be seen that the checkpass variable holds the password.

4. Hence, we get the required flag.

**Client Side Again**

**LOGON**

****

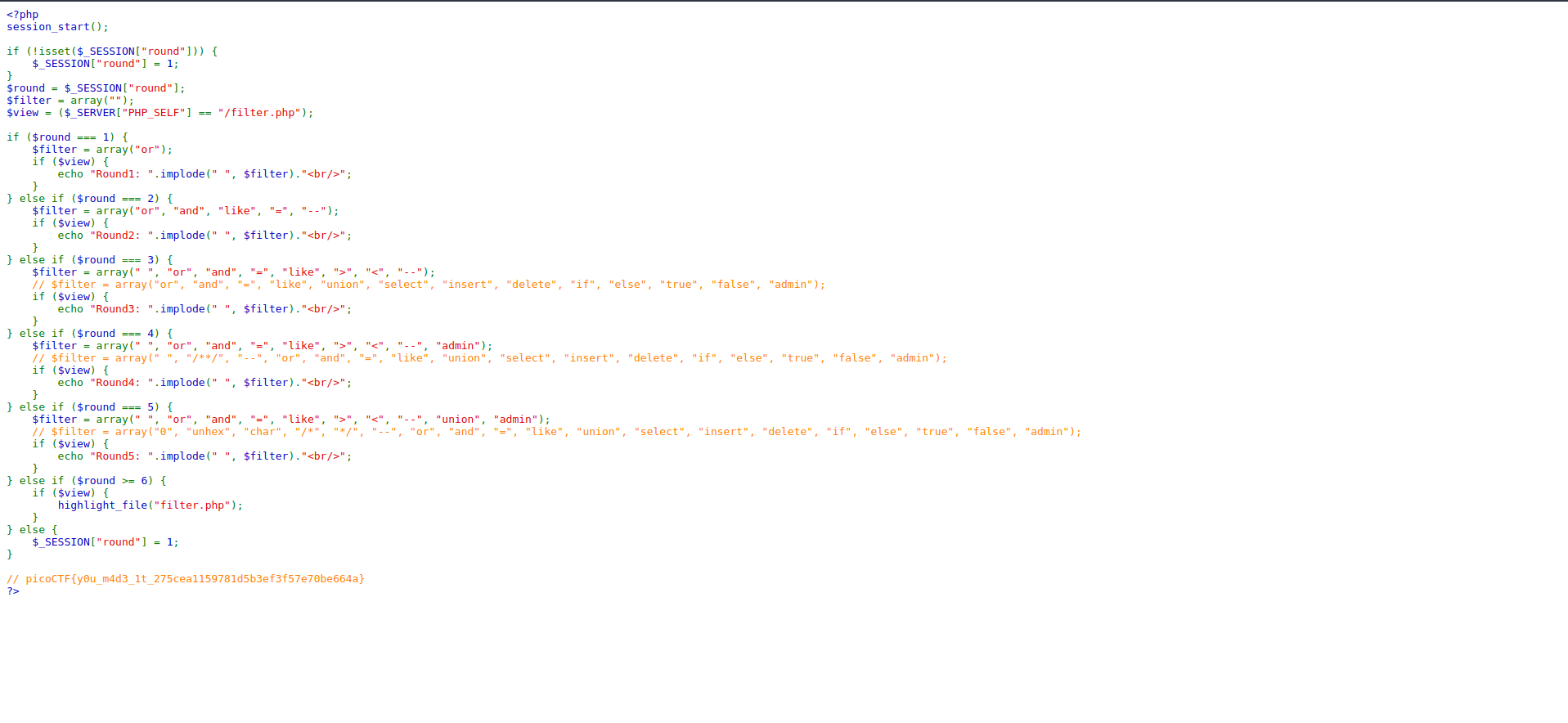
1. User: Joe and a random password gives the prompt that it failed.

2. Using the hint and inputting a random user and password logs us in but there isn’t any flag present.

3. Browsing through the cookies, we find an admin cookie set to ‘False’, and this confirms that the site uses a cookie to separate admin access.

4. Setting the value to True and reloading gives us the flag.

**Web Gauntlet**

****

1. SQL Injection: Username: admin’ --

2. Username: admin’ /\*

3. Username: admin’;

4. Username: admi’||’n’/\*

5. Username: admi’||’n’/\*

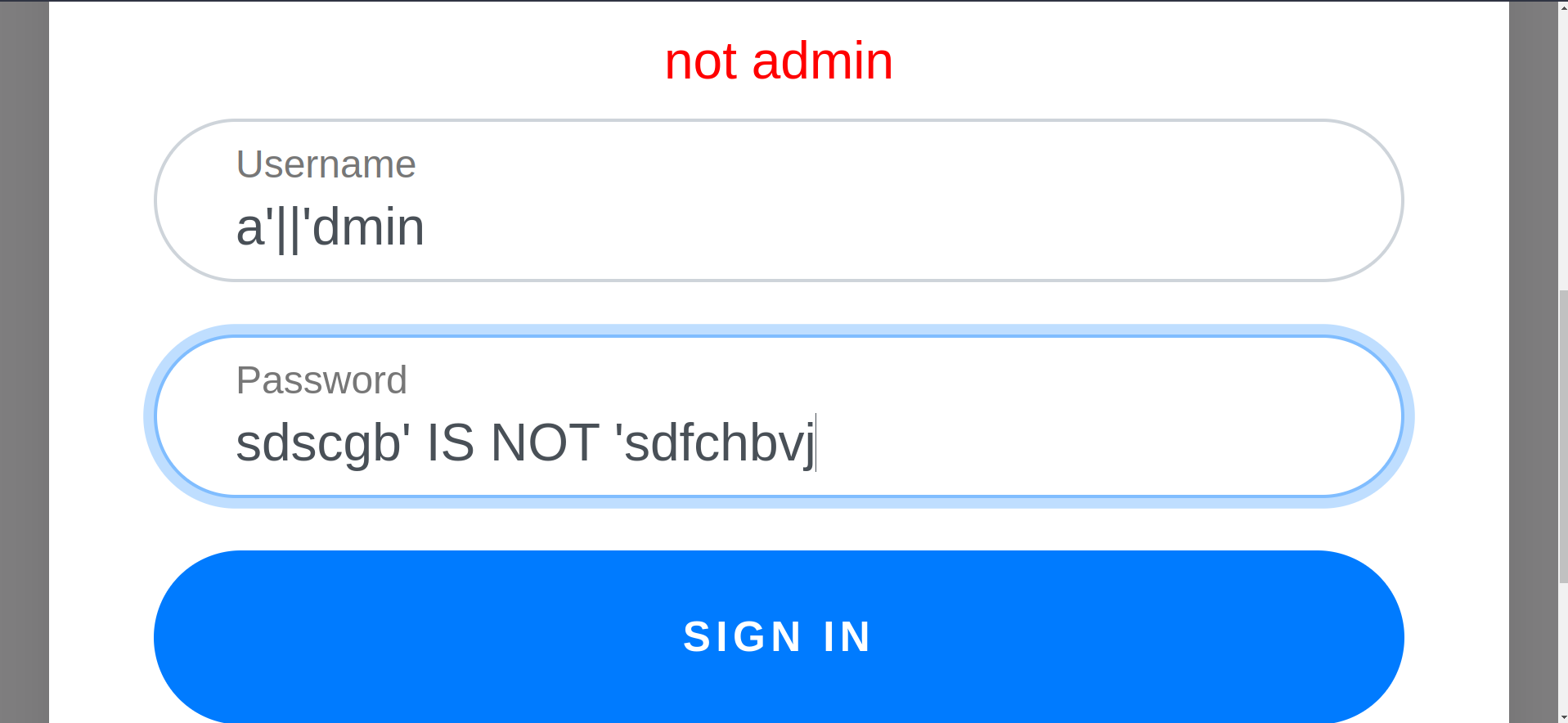
Reference: https://zacheller.dev/pico-web-gauntlet

**Web Gauntlet – 2**

1. Concatenate is not filtered.

2. SELECT username, password FROM users WHERE username='ad'||'min' AND password='efw'

This shows that the username part is True, now we gotta turn password prompt to true without using any of the filtered injections.

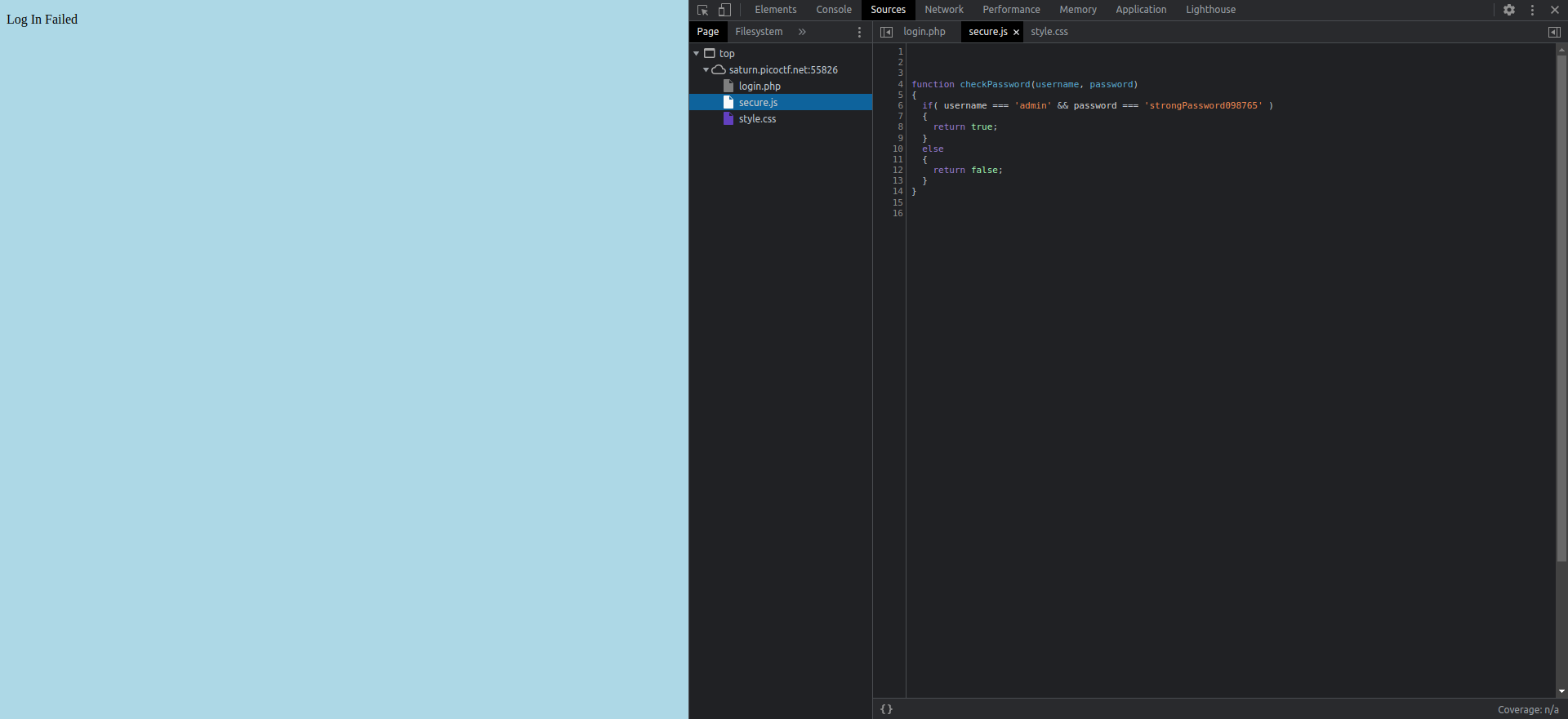
3. Username: a’||’dmin

Password: random’ IS NOT ‘sometyhing

**Web Gauntlet – 3**

1. The same injection as we used for Gauntlet-2 works.

**Local Authority**

****

1. The given website is a login page with two required credentials - username and password.

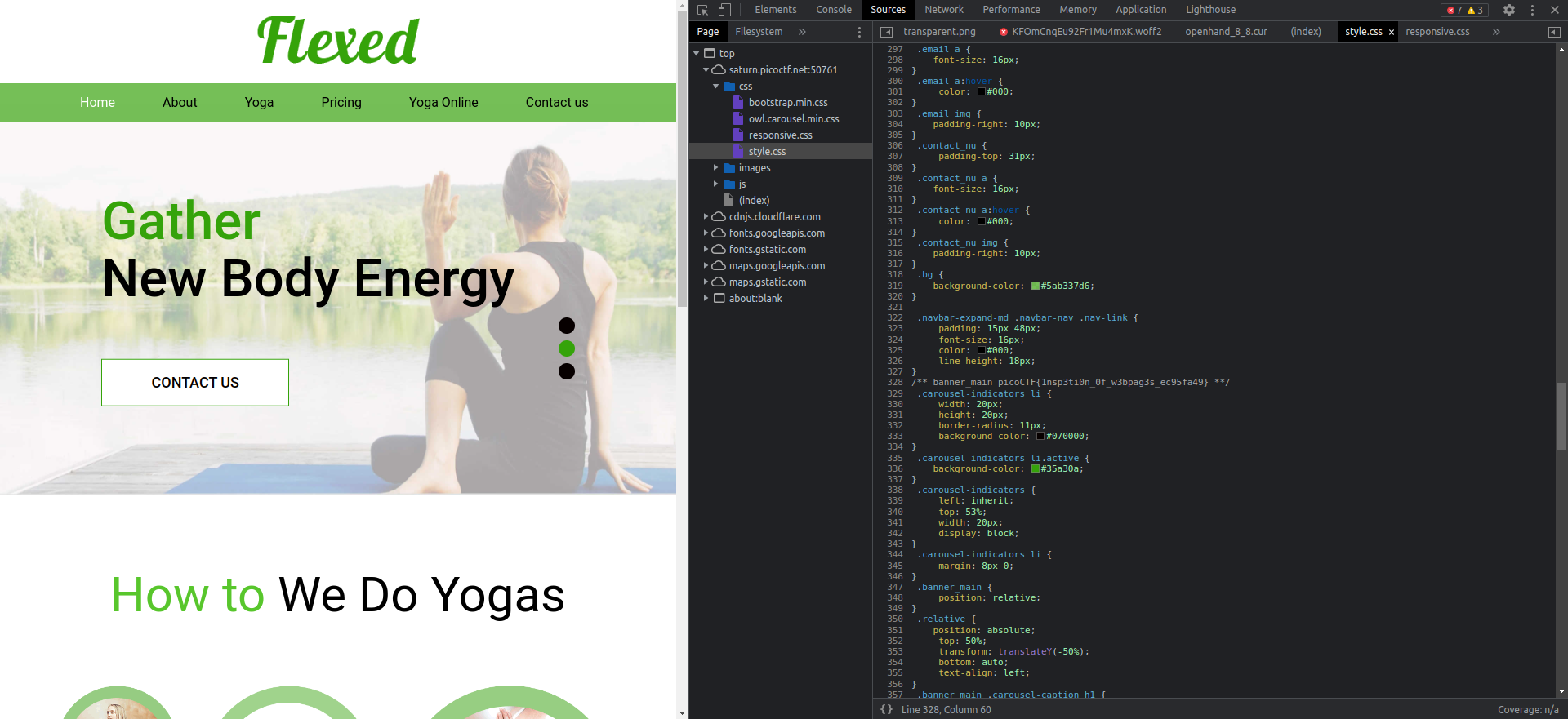
2. Jumping onto inspect element, we find nothing in the html source code.

3. To find the sources involved, I give in the most probable credentials “admin” “password”.

4. Immediately, a js script file “secure.js” pops out which explicitly includes the credentials “admin” and “strongPassword098765”.

5. Inputting the credentials would give us the flag.

**Search Source**

****

1. By visiting the website and going to the sources, we find numerous files involved.

2. By manually going through each one of them, we can find the flag in style.css file.

**SQLDIRECT**

1. Login to the PostGre SQL server

2. View databases by \l

3. Connect to ‘pico’ database \c pico

4. \dt to view tables in ‘pico’

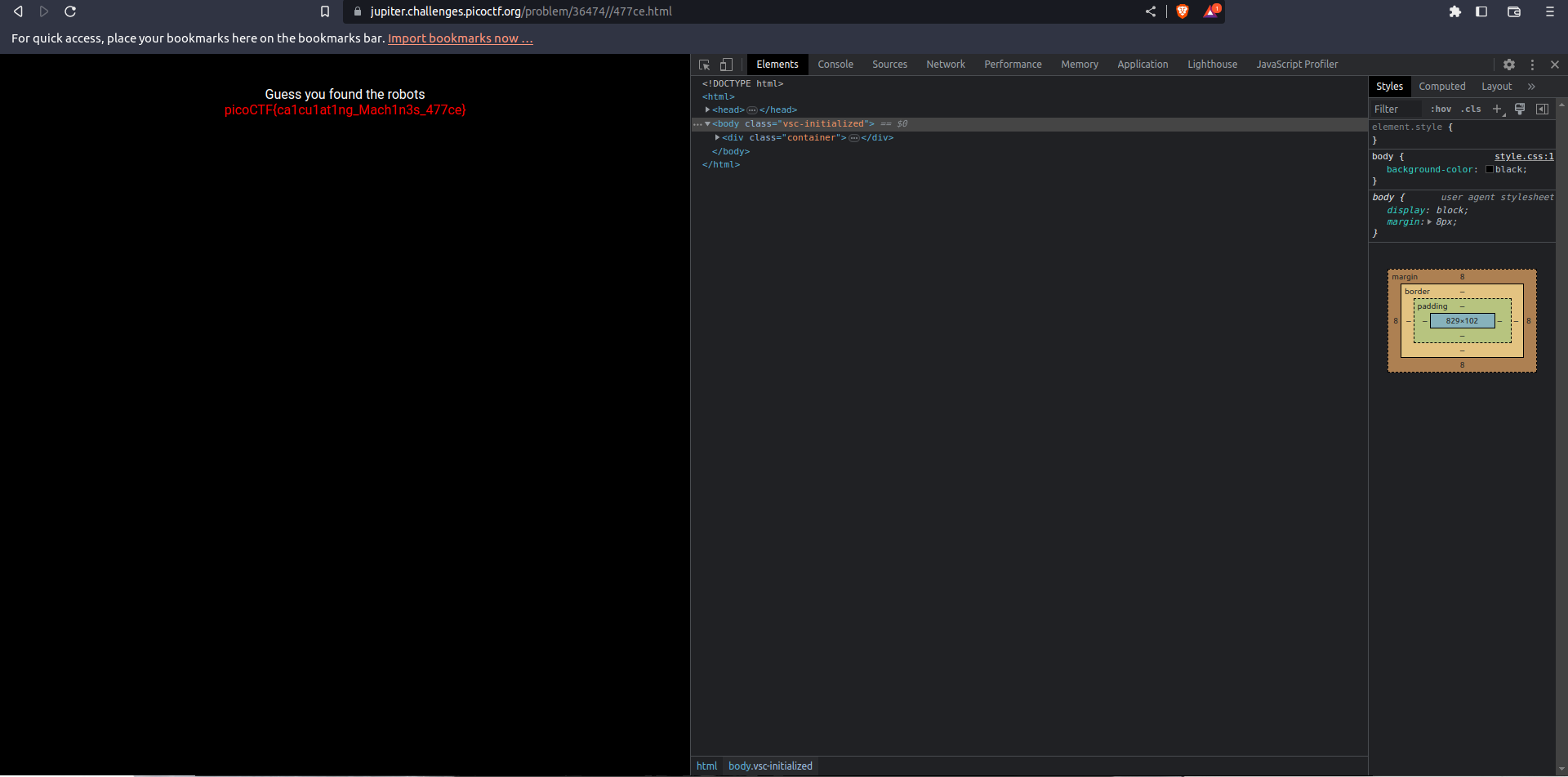
5. SELECT \* from flags;

6. Poof! Theres your flag.

**Includes**

1. Both the parts of the flag are found in the style.css and script.js files.

**Where are the Robots**

****

1. Accessing the robots.txt file, we get a hidden directory ‘/477ce.html’.

2. This gives us the flag.

**Client Side again**

**SQLLite**

**Irish Name Repo – 1**

1. Visiting the website and brwsing through it gives us a few irish gentlemen and a beautiful irish lady.

2. That de-tour aside, when we visit the login page, it’s a bit obvious given the hint is leading us towards sql injections.

3. A basic injection “admin’ --” is enough to bypass this.

**Irish Name Repo – 2**

1. Similar to the first part, “admin’ --” was enough.

**Irish Name Repo – 3**

**JaWT ScratchPad**

JSON Web Token is an internet standard for creating data with optional sign using a private secret or a public key.

Contains Header, Payload, Signature

**Picobrowser**

rajeev@rajeev-ubuntu:~$ curl --user-agent "picobrowser" "https://jupiter.challenges.picoctf.org/problem/26704/flag"

<!DOCTYPE html>

<html lang="en">

<head>

<title>My New Website</title>

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css" rel="stylesheet">

<link href="https://getbootstrap.com/docs/3.3/examples/jumbotron-narrow/jumbotron-narrow.css" rel="stylesheet">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container">

<div class="header">

<nav>

<ul class="nav nav-pills pull-right">

<li role="presentation" class="active"><a href="/">Home</a>

</li>

<li role="presentation"><a href="/unimplemented">Sign In</a>

</li>

<li role="presentation"><a href="/unimplemented">Sign Out</a>

</li>

</ul>

</nav>

<h3 class="text-muted">My New Website</h3>

</div>

<!-- Categories: success (green), info (blue), warning (yellow), danger (red) -->

<div class="alert alert-success alert-dismissible" role="alert" id="myAlert">

<button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span></button>

<!-- <strong>Title</strong> --> picobrowser!

</div>

<div class="jumbotron">

<p class="lead"></p>

<p style="text-align:center; font-size:30px;"><b>Flag</b>: <code>picoCTF{p1c0\_s3cr3t\_ag3nt\_e9b160d0}</code></p>

<!-- <p><a class="btn btn-lg btn-success" href="admin" role="button">Click here for the flag!</a> -->

<!-- </p> -->

</div>

<footer class="footer">

<p>&copy; PicoCTF 2019</p>

</footer>

</div>

<script>

$(document).ready(function(){

$(".close").click(function(){

$("myAlert").alert("close");

});

});

</script>

</body>

</html>

**SQLiLite**

\*\* Was done with the challenge earlier but as I’m writing this, the server instances are down for this challenge