Documenting My Journey: Root Me Challenges

Author: El Bettioui Reda

Contents

1 Introduction

Root Me is an online platform designed for cybersecurity enthusiasts to test their skills across a wide range of challenges. This document focuses on challenges from the **Web - Client** category, describing the methods, tools, and steps taken to solve them.

2 Challenge Logs

2.1 Challenge 1: HTML - Boutons Désactivés

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/HTML-boutons-desactives`
Objective: The form on this page is disabled and cannot be used. The task is to find a way to enable the form and solve the challenge.

2.1.1 Initial Observation

Below is an image of the challenge webpage. We see an input field and a button, both disabled. To solve this challenge, we need to enable them.

Figure 1: Disabled input and button elements.

2.1.2 Steps to Solve

1. **Open Inspect Element**
   Using the browser's developer tools, I opened the "Inspect Element" feature to view the webpage's source code.

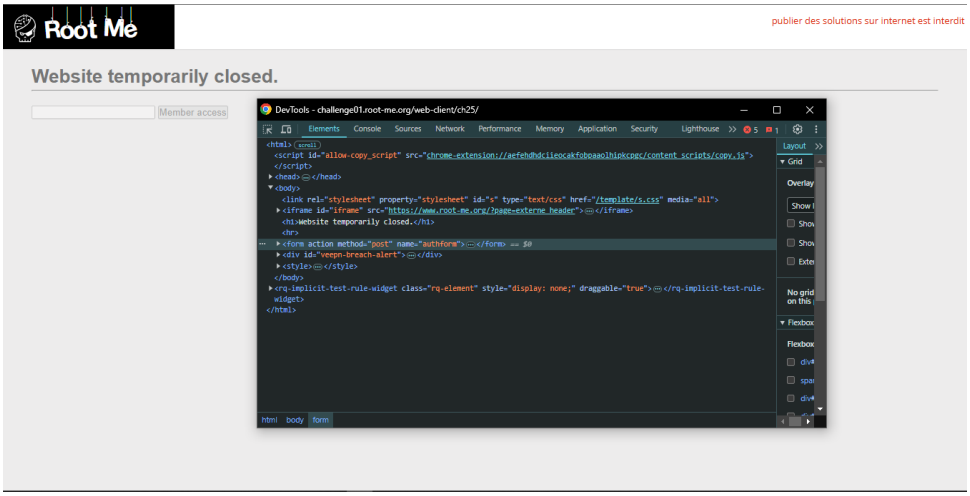

Figure 2: Inspect Element opened on the page.

2. **Locate the Input and Button Elements**
   I searched for the input and button elements in the HTML structure.

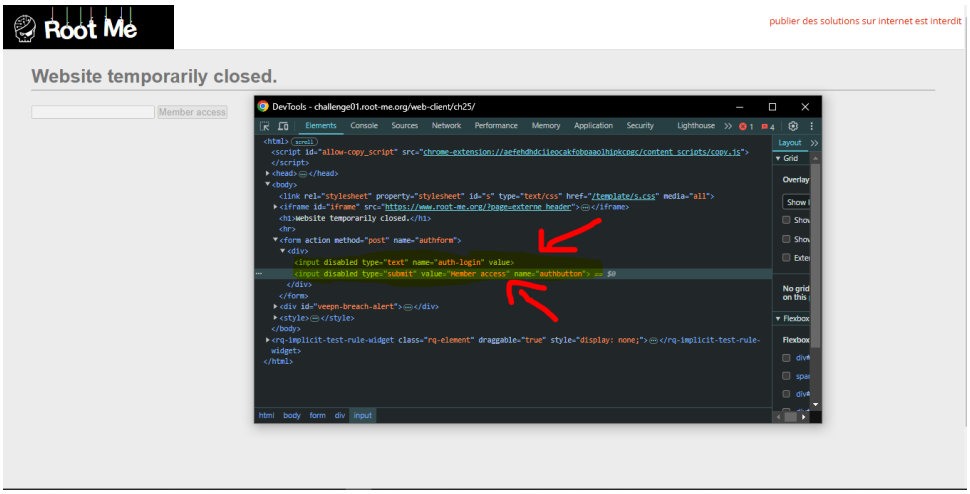

Figure 3: Input and button elements located in the code.

3. **Identify the Disabled Attribute**
   Both elements had the `disabled` attribute, which was preventing them from being enabled.

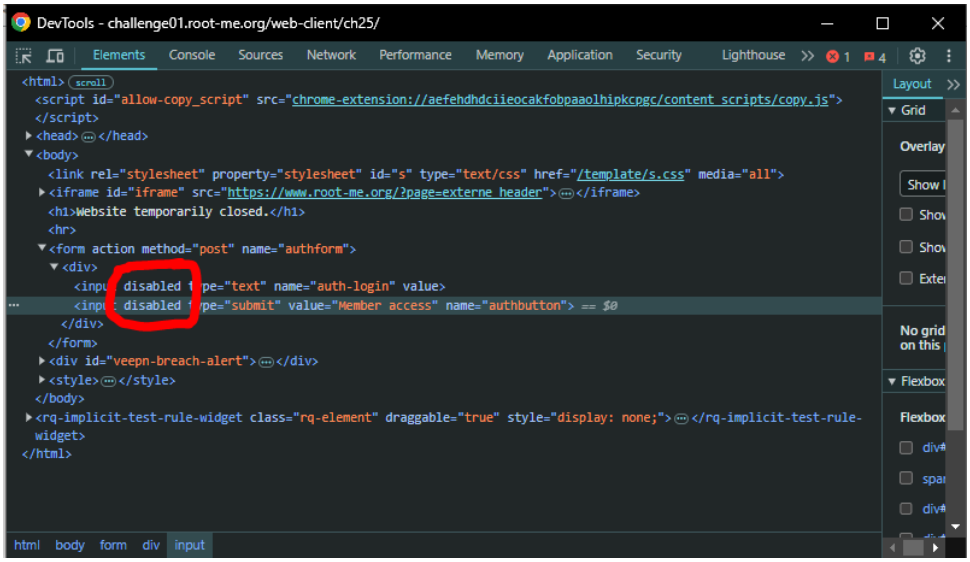

Figure 4: The `disabled` attribute on the elements.

4. **Remove the Disabled Attribute**
   By double-clicking on each element in the developer tools, I removed the `disabled` attribute.

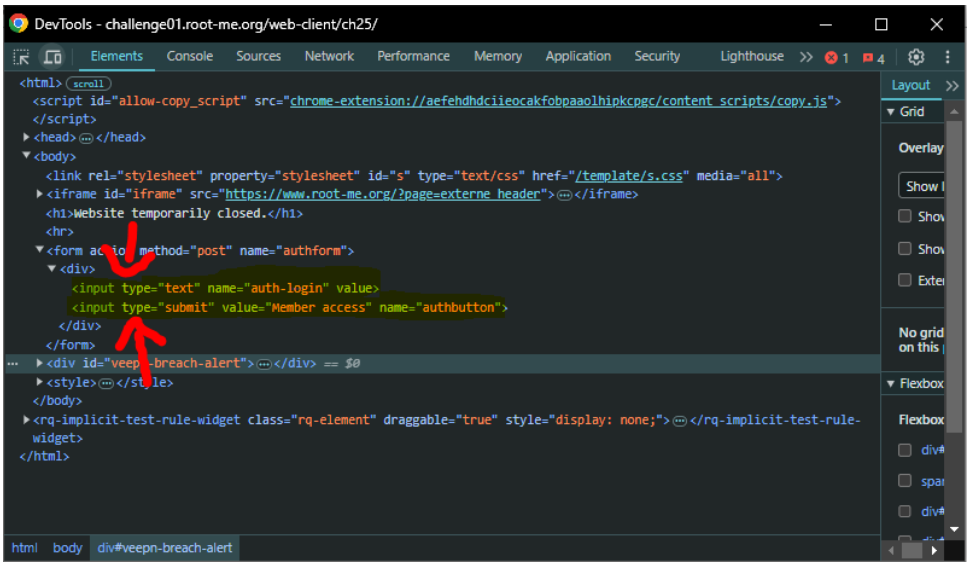

Figure 5: Disabled attribute removed from the elements.

5. **Test the Form**
   After removing the `disabled` attribute, I went back to the webpage. The input field and button were now enabled, and I could interact with them.

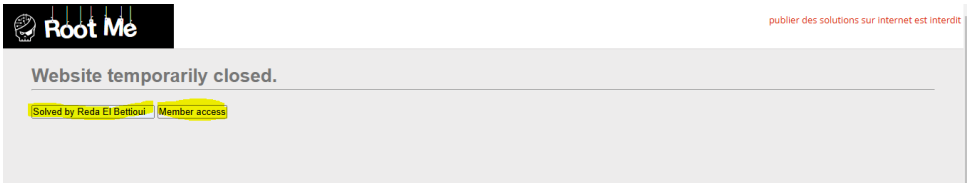

Figure 6: The form is now enabled and functional.

6. **Submit the Solution**
   The password displayed in the enabled form was **********. I entered this password into the challenge submission box and successfully completed the challenge.

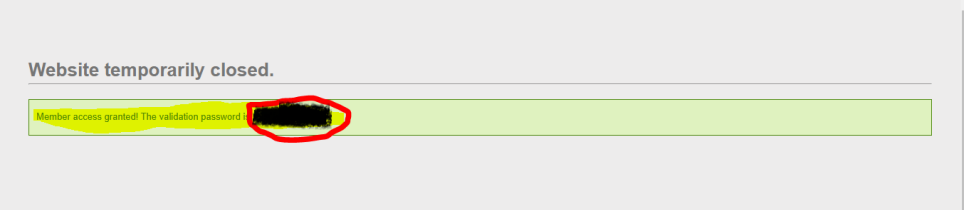

Figure 7: The solved challenge with the password.

2.1.3 Results

Password: **********
Points Earned: 5
Congratulations on solving the challenge!

2.2 Challenge 2: Javascript - Authentification

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Authentification`
Objective: Find the username and password to authenticate successfully.

2.2.1 Initial Observation

The challenge involves a webpage where the user must input the correct username and password. The goal is to find these credentials using browser developer tools.

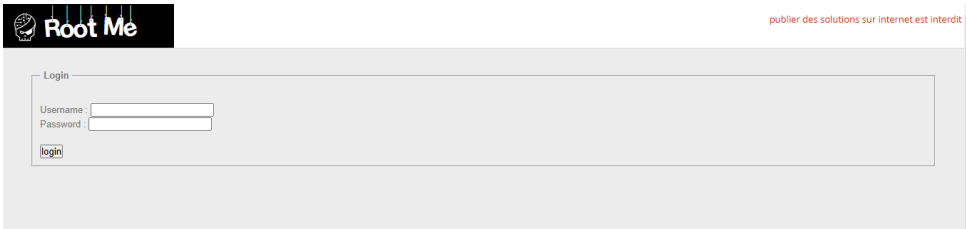

Figure 8: Find the username and password to authenticate.

2.2.2 Steps to Solve

1. **Open Inspect Element**
   First, I opened the "Inspect Element" feature to analyze the HTML and JavaScript source code for any clues.

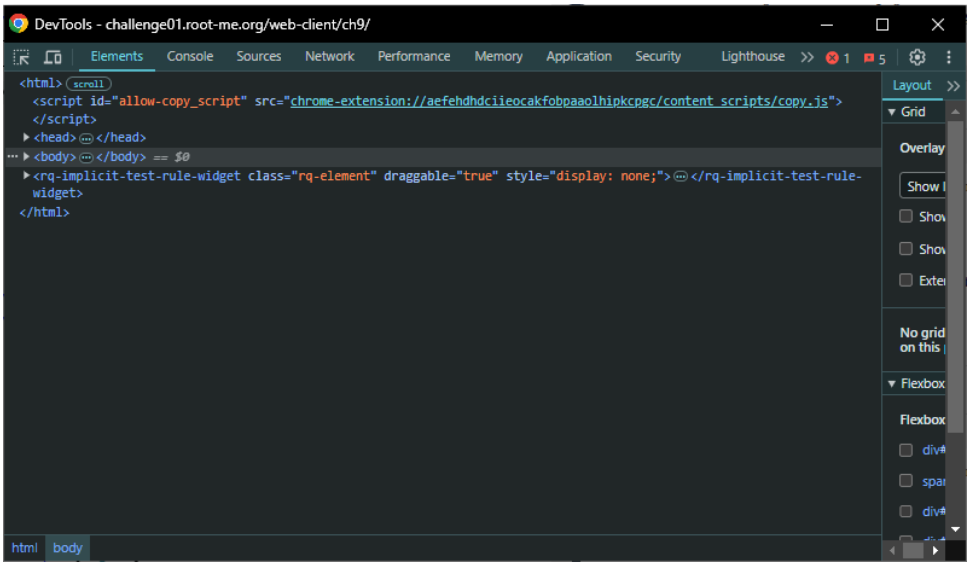

Figure 9: Inspect Element opened to analyze the source code.

2. **Analyze the HTML**
   In the HTML code, I didn't find any helpful comments or clues in the 'Elements' section. However, the submit button revealed a form linked to a 'Login()' function.

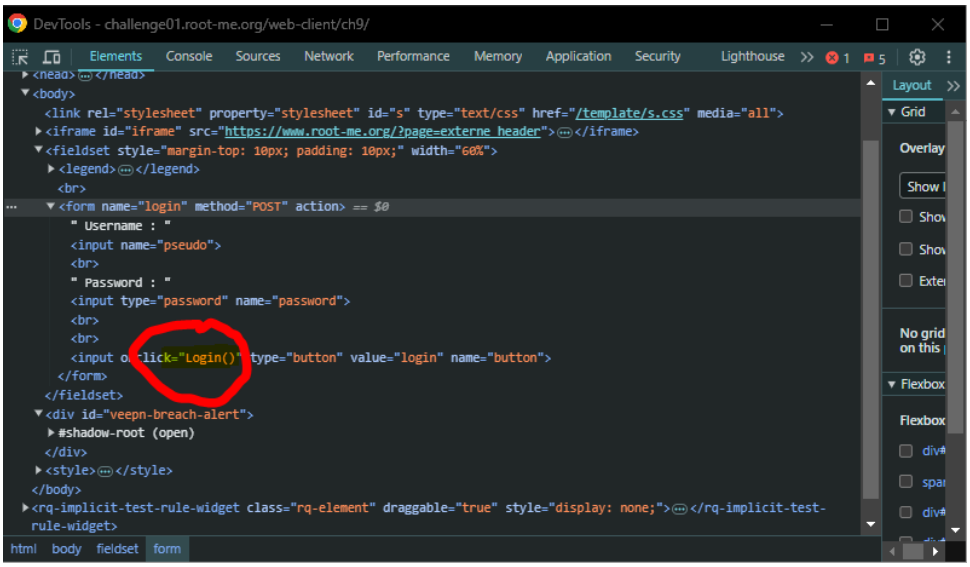

Figure 10: HTML form linked to the `Login()` function.

3. **Check JavaScript Files**
   Next, I navigated to the 'Sources' tab and explored the available JavaScript files.

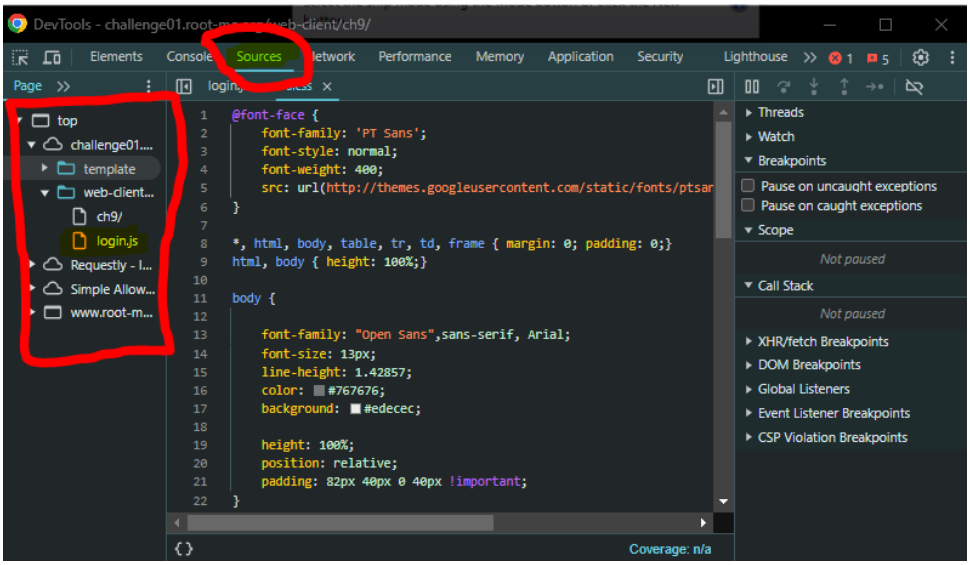

Figure 11: JavaScript files listed in the Sources section.

4. **Locate the Login Function**
   Upon opening the relevant JavaScript file, I found the 'Login()' function. This function checks whether the username and password entered by the user match predefined values.

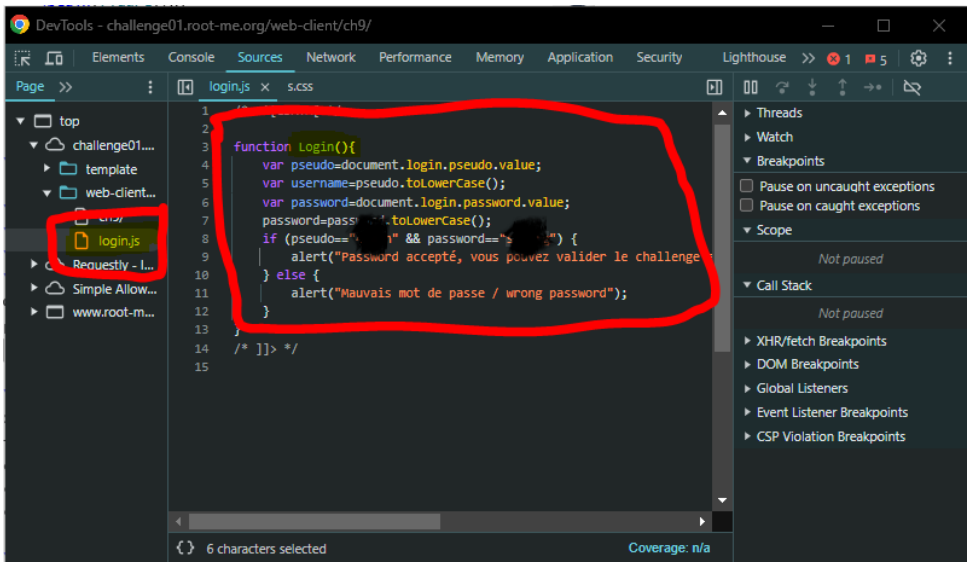

Figure 12: JavaScript file showing the `Login()` function.

5. **Extract Credentials**
   Inside the 'Login()' function, the predefined username and password were clearly visible. The username is ******, and the password is ******.

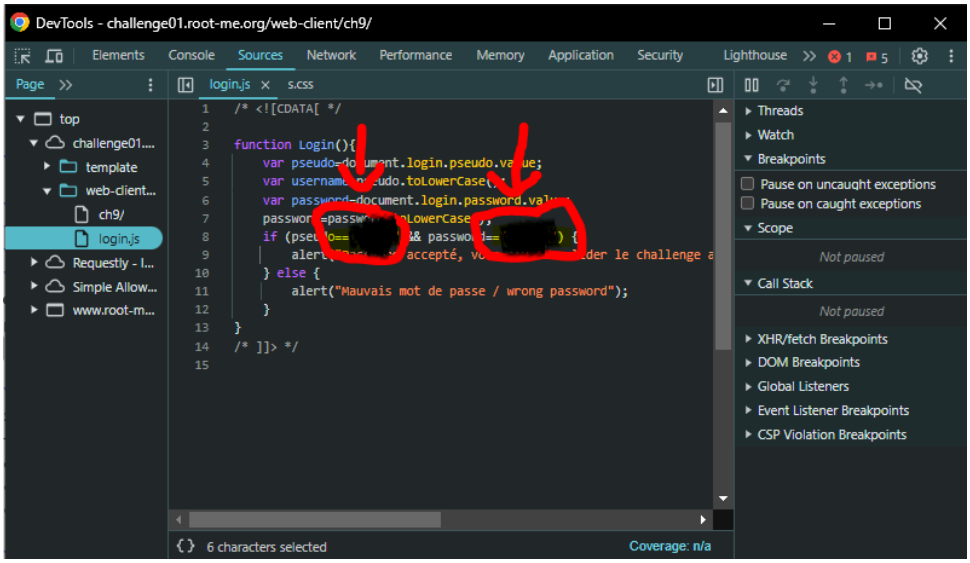

Figure 13: Credentials found in the `Login()` function.

6. **Test the Credentials**
   After copying the username and password, I entered them into the webpage challenge. An alert message confirmed that the challenge was successful.

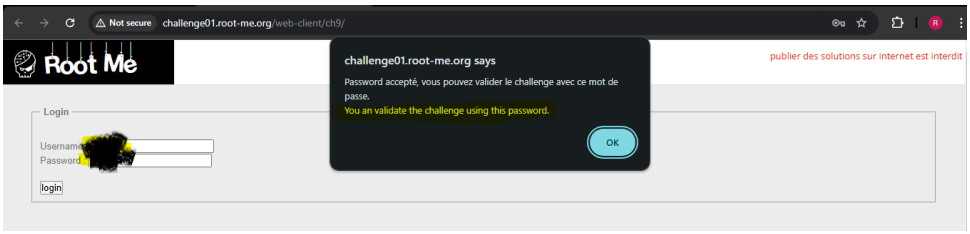

Figure 14: Challenge success message after entering the credentials.

2.2.3 Results

Username: ******
Password: ******
Points Earned: 5
Congratulations on solving this challenge and earning 5 points!

2.3 Challenge 3: Javascript - Source

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Source`
Objective: Find the password to authenticate successfully.

2.3.1 Initial Observation

The challenge involves analyzing the webpage's source code to find the correct password and successfully authenticate.

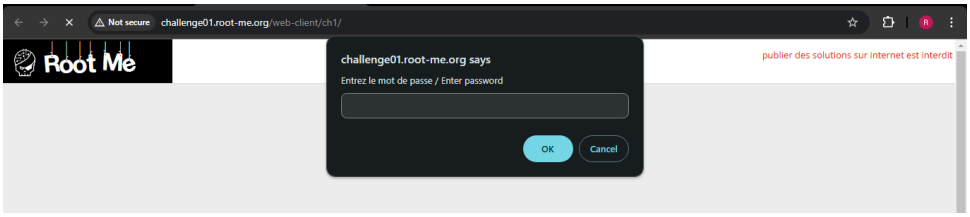

Figure 15: Find the password.

2.3.2 Steps to Solve

1. **Open Inspect Element**
   Like before, I opened the "Inspect Element" feature to analyze the HTML code for any helpful hints. However, there was nothing useful in the 'Elements' section. So, I moved to the 'Sources' tab.

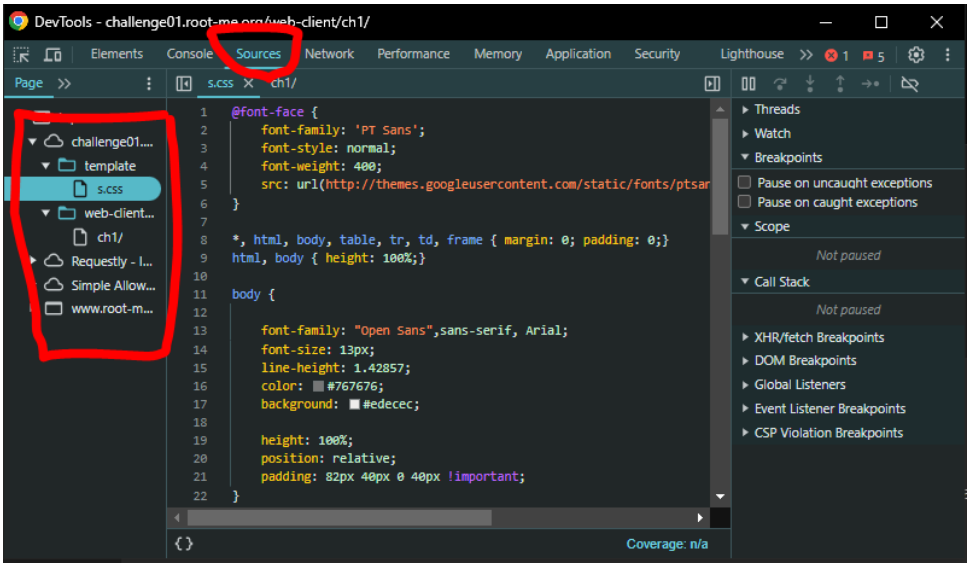

Figure 16: Inspecting the Sources section for useful files.

2. **Locate the Relevant File**
   In the 'Sources' section, I found a file named `ch1`. Opening it revealed some interesting code.

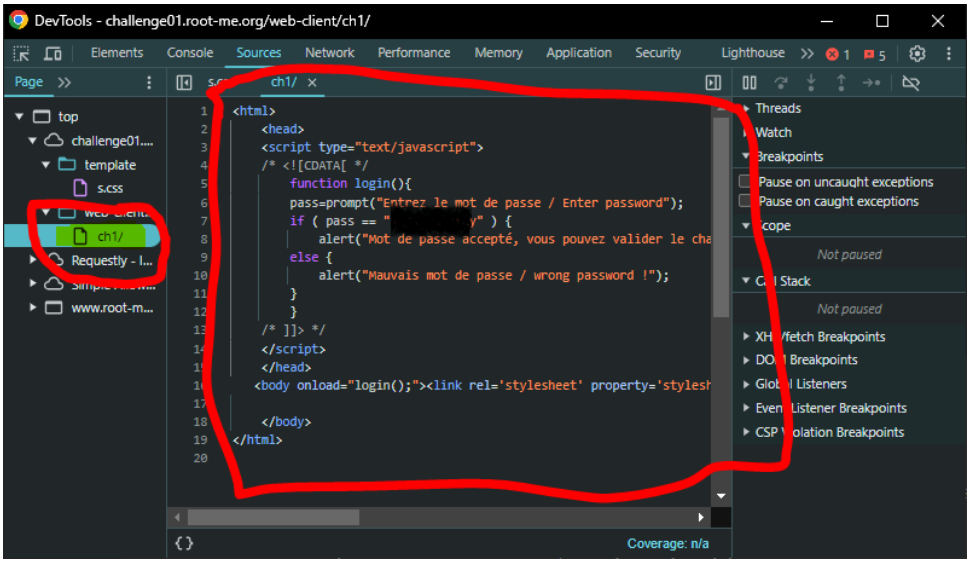

Figure 17: The `ch1` file contains a script tag with JavaScript code.

3. **Analyze the Script Code**
   Inside the file, there was a JavaScript function that validates the password entered by the user. The condition checks if the password matches a specific value.

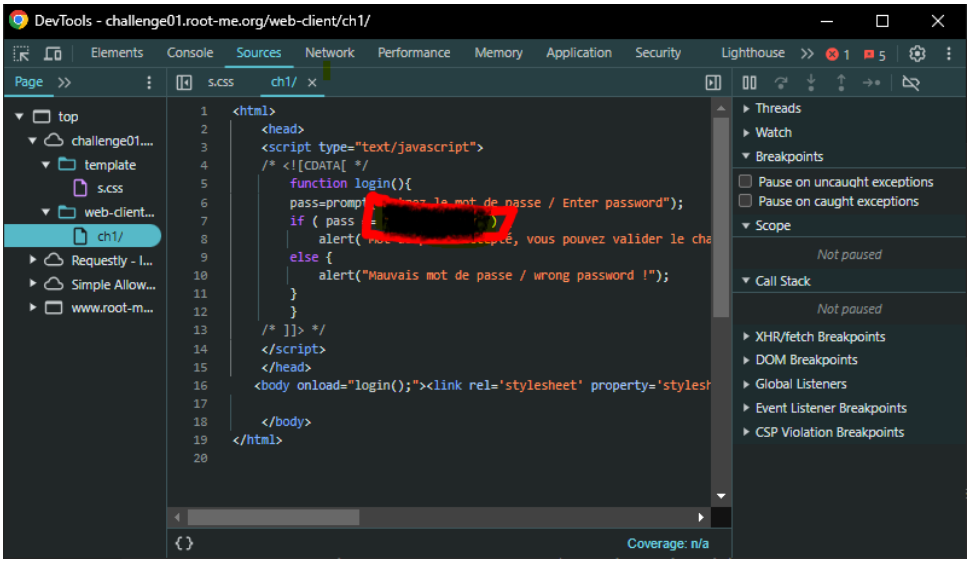

Figure 18: JavaScript code showing the validation logic and the hardcoded password.

4. **Extract and Test the Password**
   I identified the password, ******, inside the 'if' condition. I refreshed the page to reset it and entered the extracted password.

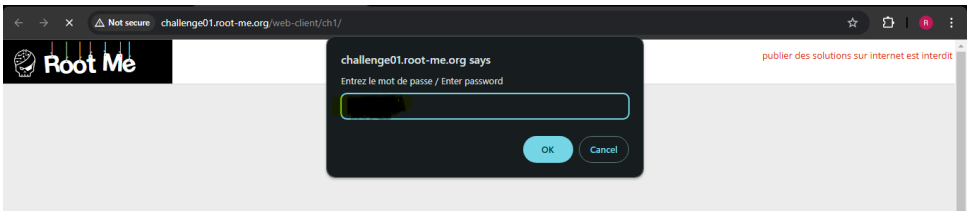

Figure 19: Entering the extracted password in the input field.

5. **Verify the Solution**
   After entering the password, the webpage displayed a success message, confirming that the password was correct.

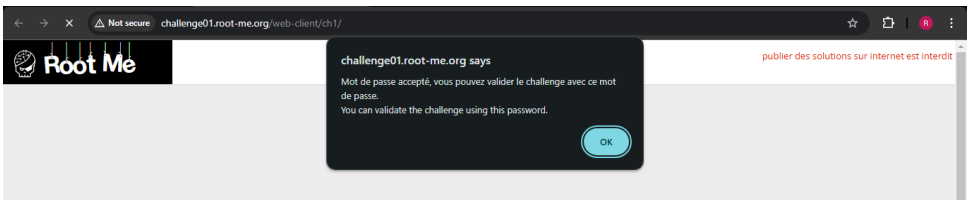

Figure 20: Success message confirming the correct password.

2.3.3 Results

Password: ******
Points Earned: 5
Congratulations on solving this challenge and earning 5 points!

2.4 Challenge 4: Javascript - Authentification 2

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Authentification-2`
Objective: Find the username and password to authenticate successfully.

2.4.1 Initial Observation

The challenge involves finding the correct username and password. Using browser developer tools, we aim to analyze the webpage's elements and source code.

Figure 21: Initial inspection of the challenge webpage.

2.4.2 Steps to Solve

1. **Analyze the HTML Code**
   I started by opening "Inspect Element" to analyze the HTML code. I found an input button linked to the `connexion()` function, hinting at further investigation in the 'Sources' tab.

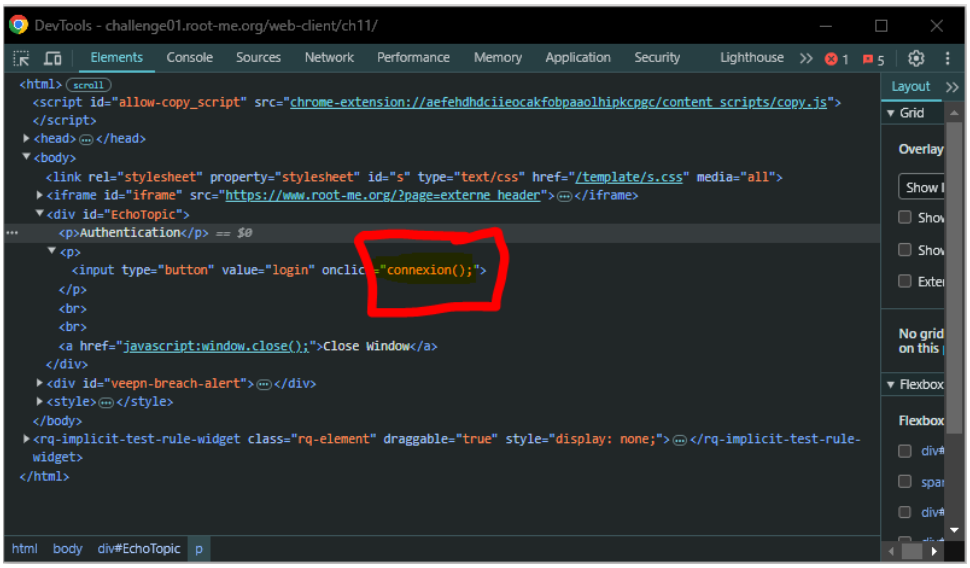

Figure 22: HTML code showing the `connexion()` function.

2. **Check JavaScript Files**
   In the 'Sources' tab, I located the JavaScript file containing the `connexion()` function. This function performs user authentication.

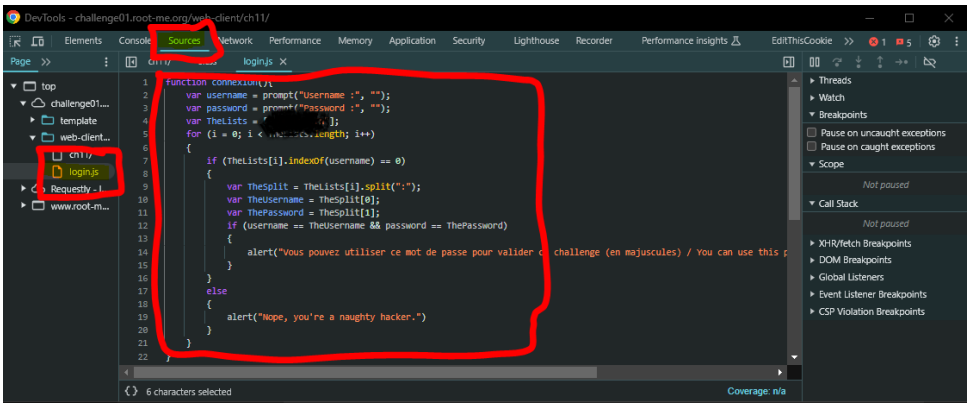

Figure 23: JavaScript file containing the `connexion()` function.

3. **Understand the Function**
   The `connexion()` function prompts the user to input a username and password. It then compares the inputs against a predefined list. If a match is found, the user is alerted that the challenge is solved.

4. **Extract Credentials**
   The predefined list contains the credentials: `******` as the username and `******` as the password. These values are used for successful authentication.

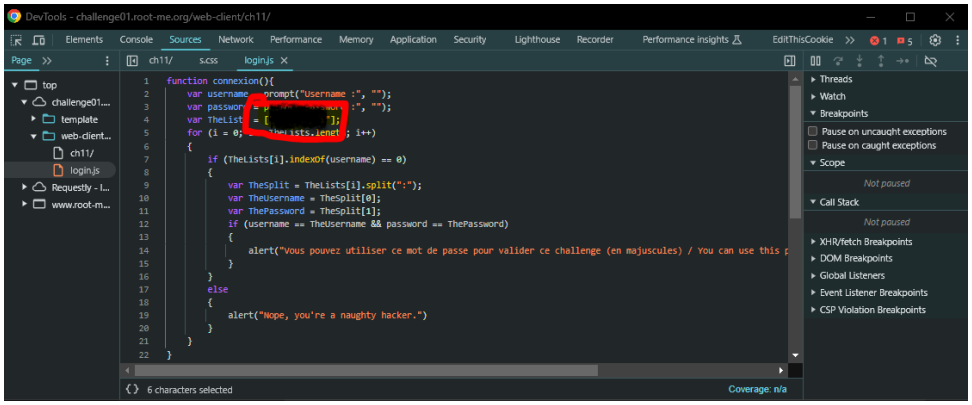

Figure 24: Extracted credentials: username and password.

5. **Test the Credentials**
   Returning to the challenge webpage, I clicked "Login" and entered the extracted credentials. The challenge was successfully solved.

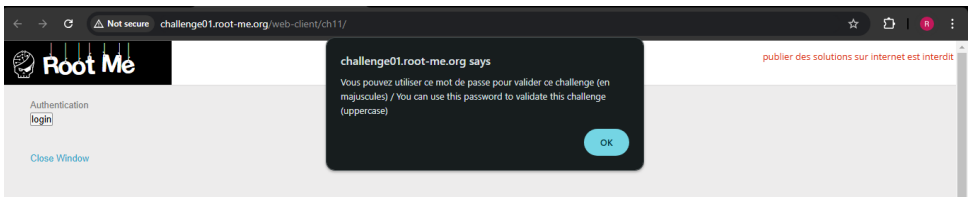

Figure 25: Challenge solved with the correct credentials.

2.4.3 Results

Username: ******
Password: ******
Points Earned: 10
Congratulations on solving this challenge and earning 10 points!

2.5 Challenge 5: Javascript - Obfuscation 1

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Obfuscation-1`
Objective: Find the password to solve the challenge.

2.5.1 Initial Observation

The goal is to analyze the webpage and source code to uncover the password.

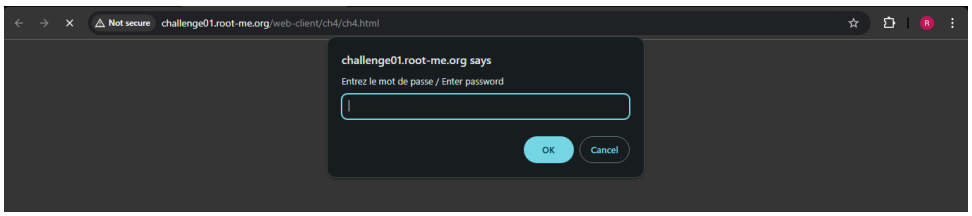

Figure 26: Initial view of the challenge webpage.

2.5.2 Steps to Solve

1. **Inspect Element for Clues**
   I started by analyzing the HTML code in "Inspect Element." However, nothing significant was found here.

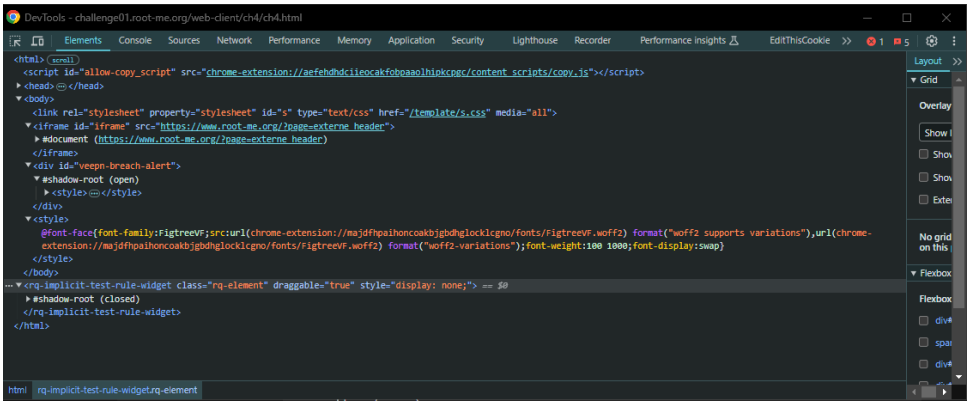

Figure 27: HTML inspection showing no valuable clues.

2. **Analyze the Sources Section**
   Moving to the 'Sources' tab, I found a file named `ch4.html` containing HTML code. This file had a JavaScript section that revealed an encoded password.

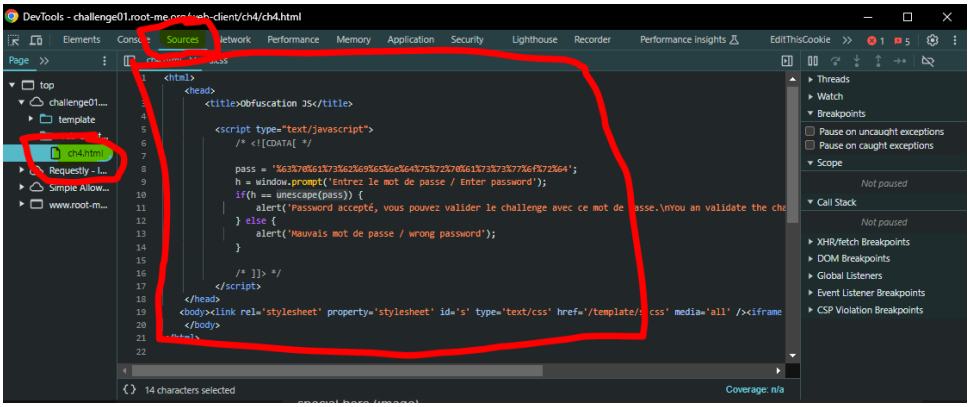

Figure 28: File `ch4.html` showing the encoded password.

3. **Understand the Encoding Logic**
   The JavaScript code uses the `unescape()` function to decode an obfuscated password. The encoded string is: `**********`

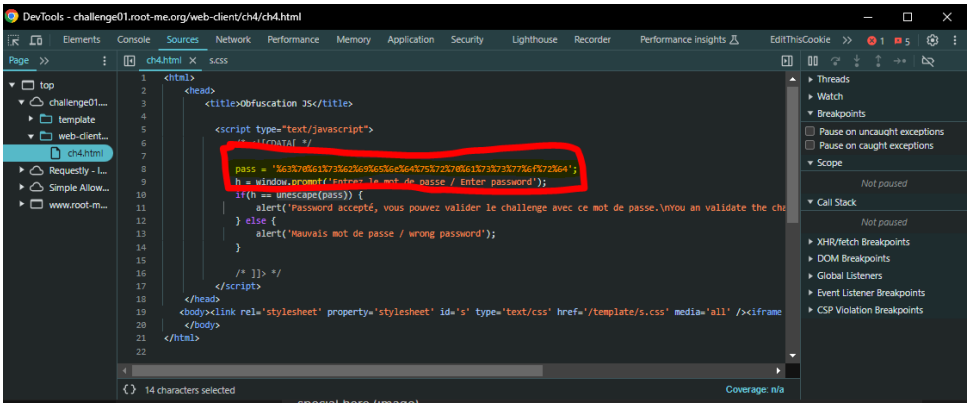

Figure 29: Encoded password found in the JavaScript code.

4. **Decode the Password**
   To decode the password, I ran the following command in the browser's Console:

   ```
   console.log(****************);
   ```

   The output revealed the password: ******.

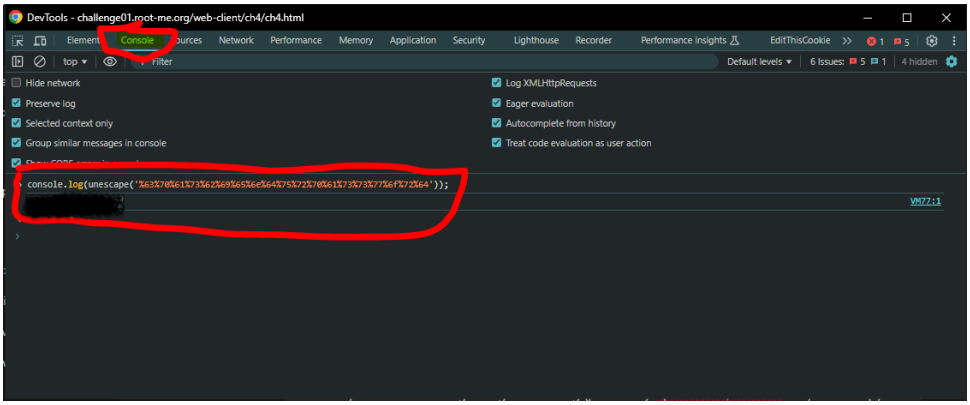

Figure 30: Decoded password: ******.

5. **Test the Password**
   Returning to the challenge webpage, I refreshed the page, entered the password, and successfully solved the challenge.

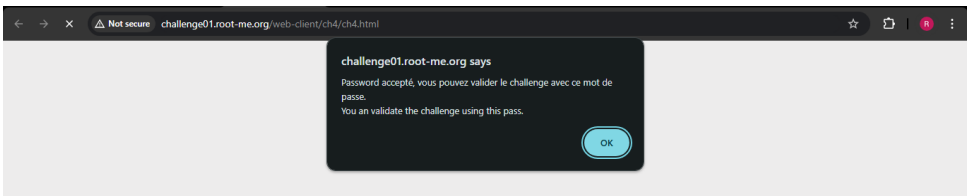

Figure 31: Challenge solved using the decoded password.

2.5.3 Results

Password: ******
Points Earned: 10
Congratulations on solving this challenge and earning 10 points!

2.6 Challenge 6: Javascript - Obfuscation 2

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Obfuscation-2`
Objective: Find the password to solve the challenge.

2.6.1 Initial Observation

The challenge presents an empty page, so all the analysis must be done in the browser's developer tools.

Figure 32: Initial view of the challenge webpage.

2.6.2 Steps to Solve

1. **Inspect Element for Clues**
   As usual, I started by analyzing the HTML code in the "Inspect Element" section. However, there was nothing special found in the HTML.

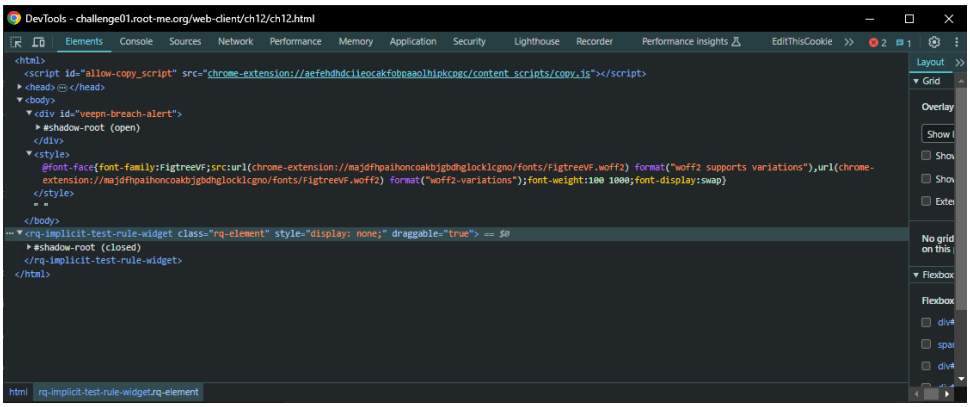

Figure 33: HTML inspection showing no useful information.

2. **Analyze the Sources Section**
   Moving to the 'Sources' tab, I found a file named `ch12.html`. Inside this file, there was a JavaScript code that revealed the password.

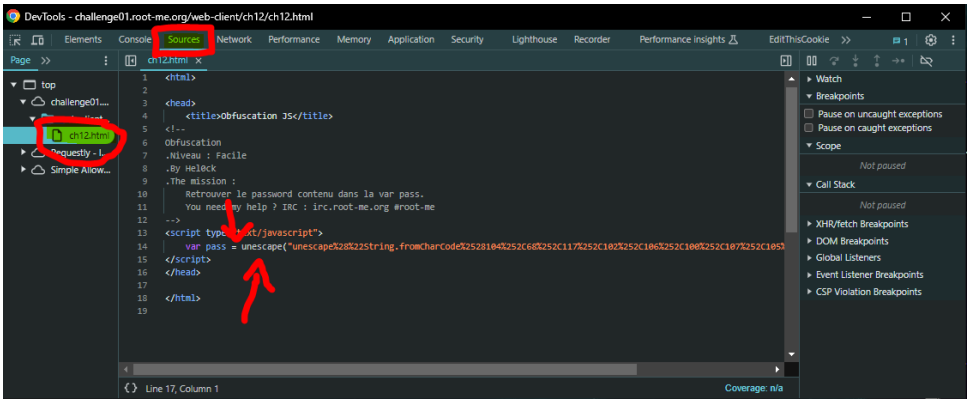

Figure 34: File `ch12.html` containing the obfuscated password.

3. **Decode the Password in the Console**
   Since the password was encoded, I decoded it using the browser's Console section. The decoding process involved running the following command:

   ```
   console.log(unescape("***************"))
   ```

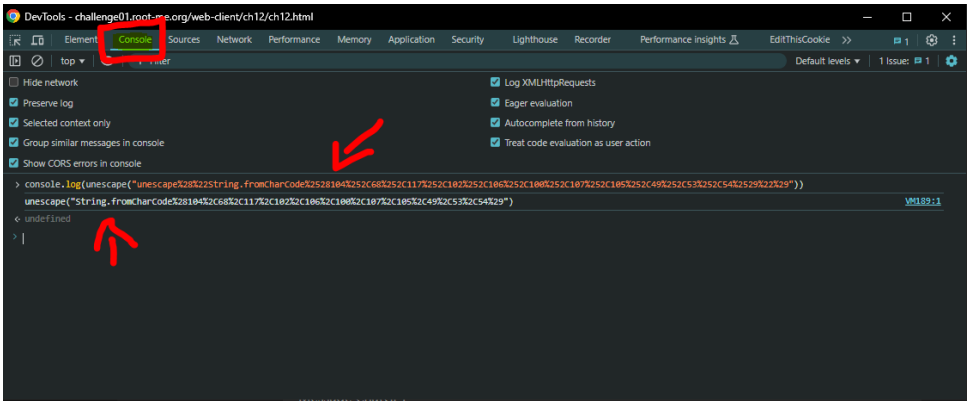

Figure 35: Running the decoding command in the Console.

4. **Iterative Decoding Process**
   The output of the first decoding attempt was not a string, so I repeated the process using the new output each time until the final decoded string was obtained. After multiple iterations, the password was revealed as: ******.

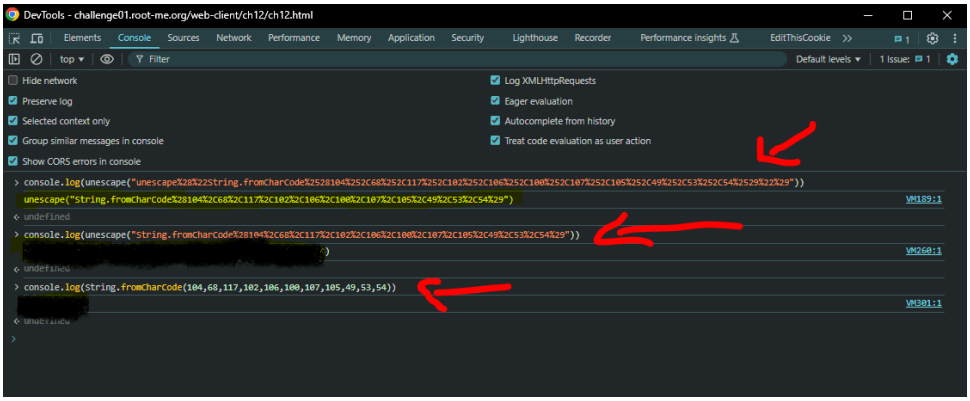

Figure 36: Final decoded password: ******.

2.6.3 Results

Password: ******
Points Earned: 10
Congratulations on solving this challenge and earning 10 points!

2.7 Challenge 7: Javascript - Native Code

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Native-code`
Objective: Find the password to solve the challenge.

2.7.1 Initial Observation

Upon loading the challenge page, an alert appears asking for a password.

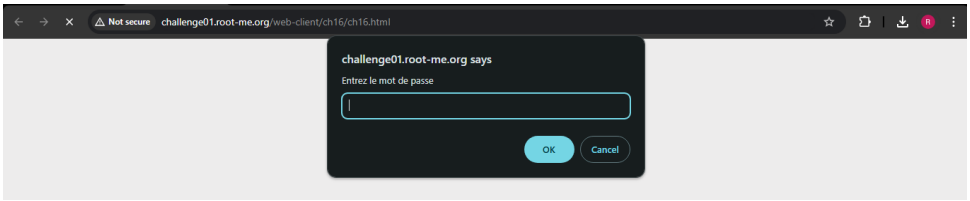

Figure 37: Initial alert on the challenge page asking for a password.

2.7.2 Steps to Solve

1. **Inspect the Code for Clues**
   Opening the "Inspect Element" section reveals an encrypted script. This script might contain the password. To proceed, I copied the encrypted script by double-clicking on it.

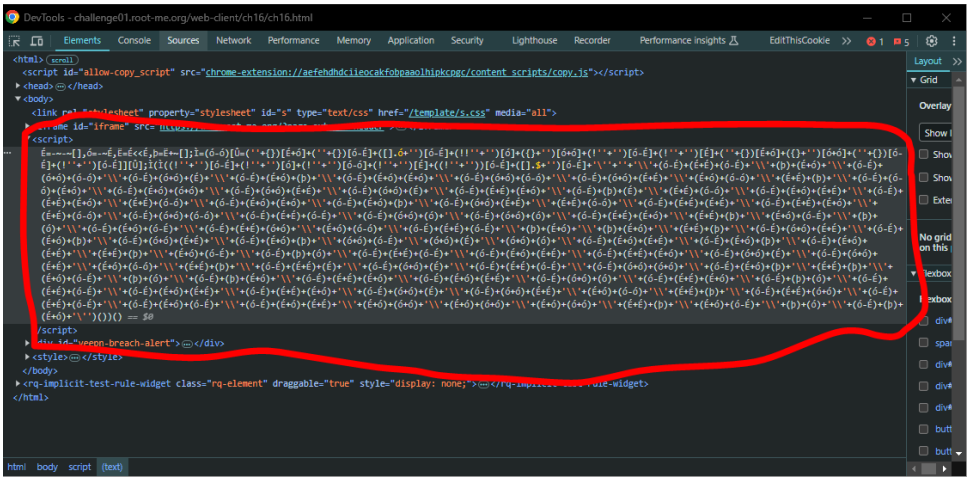

Figure 38: Encrypted script found in the page's source code.

2. **Run the Script in the Console**
   Next, I pasted the script into the "Console" section and ran it. Upon execution, another alert appeared asking for the password.

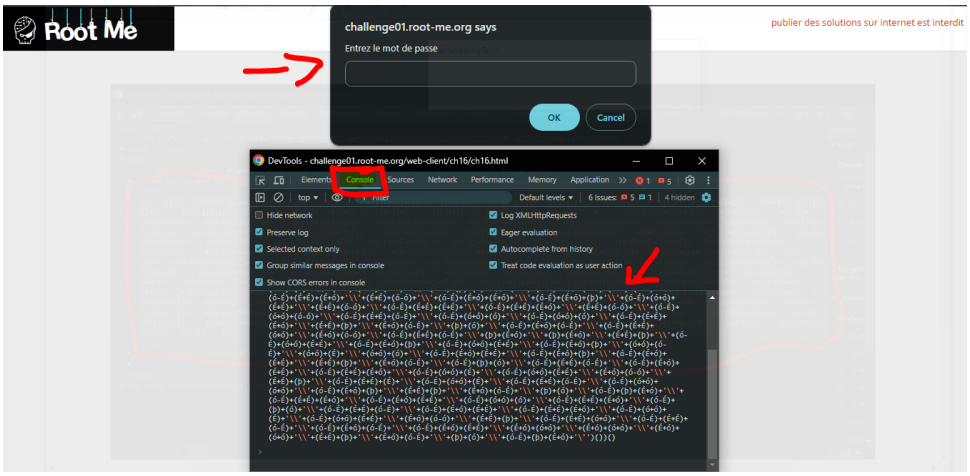

Figure 39: Running the encrypted script in the Console section.

3. **Decrypt the Script for Insights**
   To decrypt the script, I modified it by removing the '()' at the end. I then ran the modified script in the Console section, revealing the decrypted content.

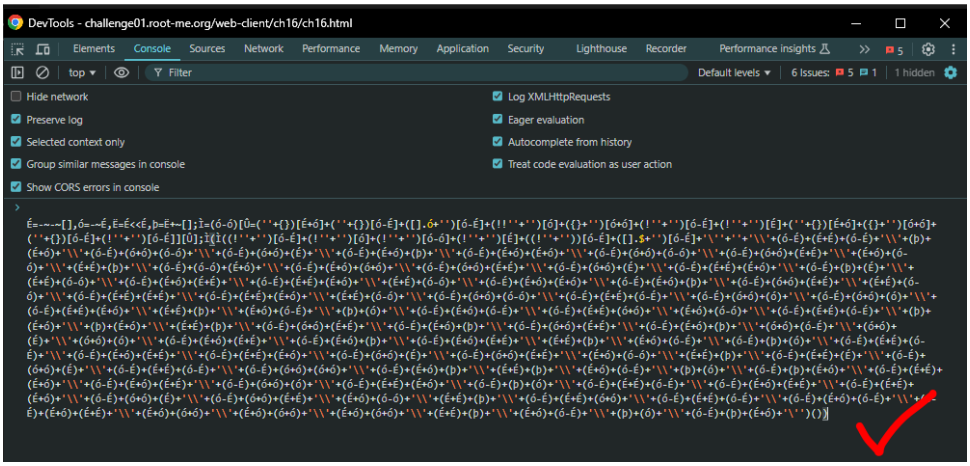

Figure 40: Decrypted content of the script in the Console.

4. **Extract and Test the Password**
   From the decrypted content, I found the password: ******. I returned to the challenge page, refreshed it, and entered the password. Success!

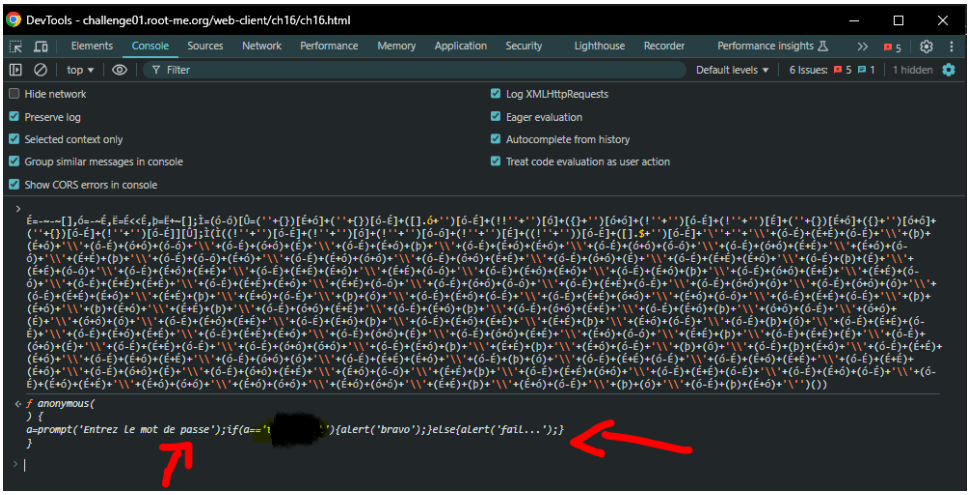

Figure 41: Challenge solved using the password: ******.

2.7.3 Results

Password: ******
Points Earned: 15
Congratulations on solving this challenge and earning 15 points!

2.8 Challenge 8: Javascript - Webpack

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Webpack`
Objective: Find the flag hidden in the code.

2.8.1 1. Initial Observation

The challenge involves a web application with multiple pages. Upon initial inspection using the browser's developer tools, no useful information was found in the HTML or JavaScript files.

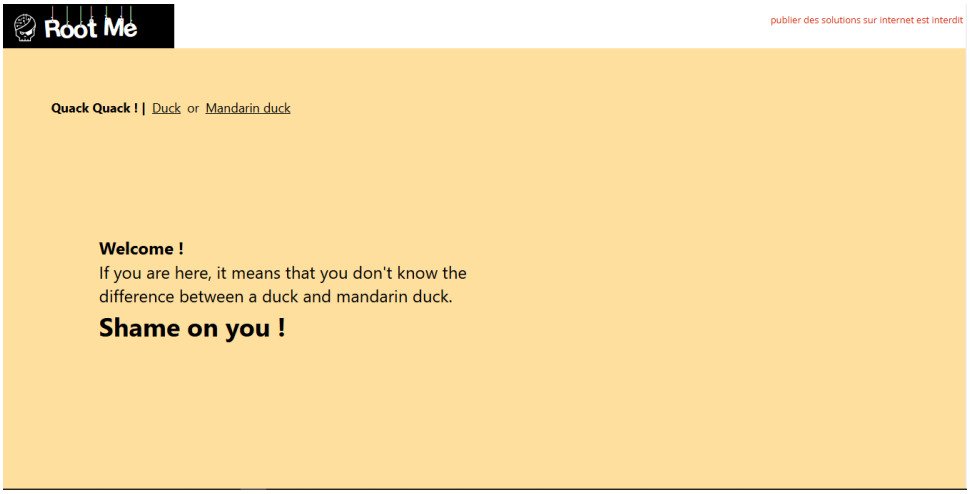

Figure 42: Initial view of the web application.

2.8.2 2. Steps to Solve

1. **Explore the Source Section**
   Since this challenge is about Webpack, I navigated to the "Sources" section in the developer tools. I found three JavaScript files. However, the files were minified or transpiled, making them hard to read.

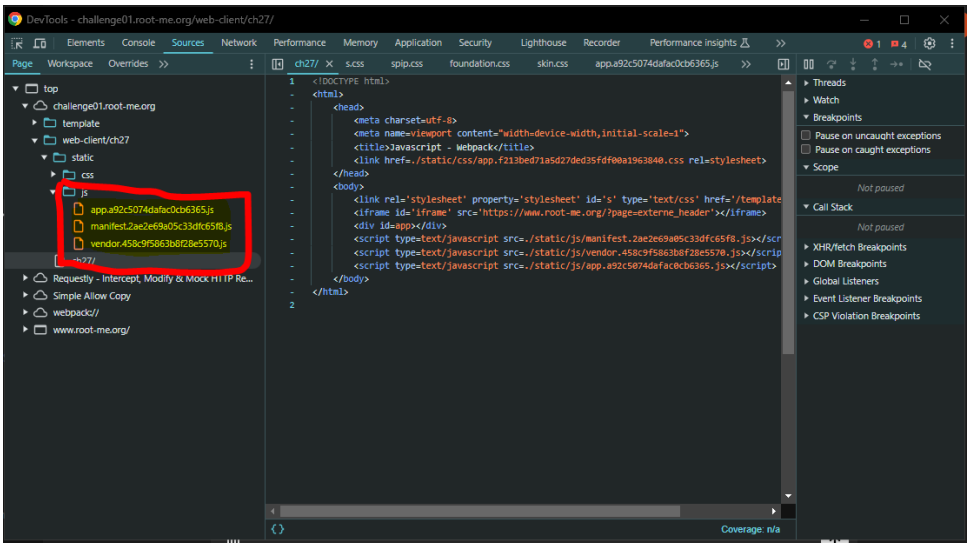

Figure 43: Minified JavaScript files in the Sources section.

2. **Identify the Source Map**
   Reading about Webpack and source maps, I knew to look for the 'sourceMappingURL', typically located at the bottom of the file. Scrolling to the bottom of one file, I found the following path:

```
//# sourceMappingURL=app.a92c5074dafac0cb6365.js.map
```

   This file maps to the original source code.

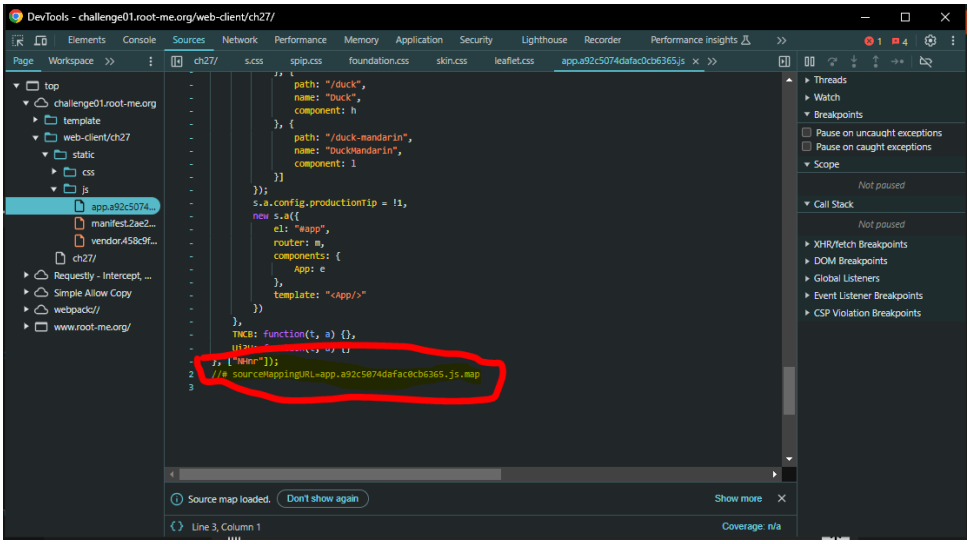

Figure 44: Path to the original file.

3. **Download the Source Map**
   Using the path, I constructed the complete URL to download the source map file: http://challenge01.root-me.org/web-client/ch27/static/js/app.a92c5074dafac0cb6365.js.map Opening this URL downloaded the file, which I opened in Visual Studio Code.

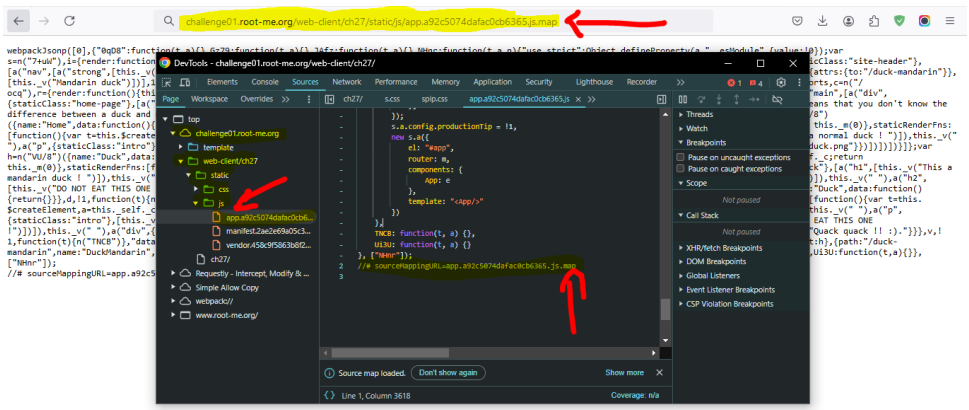

Figure 45: Downloaded source map file.

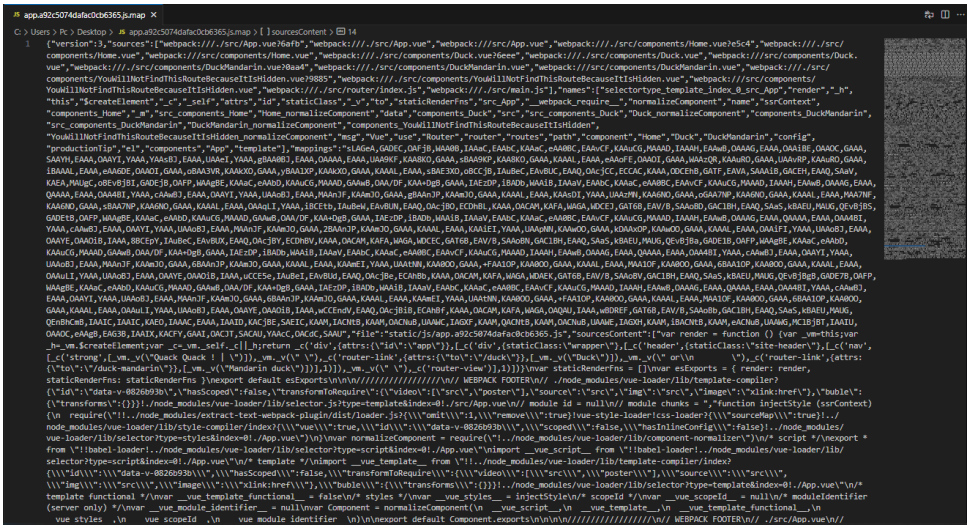

Figure 46: Opened in Visual Studio Code.

4. **Search for the Flag**
   Searching through the source map file for common keywords like "password" yielded no results. However, searching for "flag" revealed the flag:

   ******

Figure 47: Flag found in the source map file.

2.8.3 3. Results

Password: ******
Points Earned: 15
Successfully solved the challenge by finding the hidden flag in the source map file.

2.9 Challenge 9: Javascript - Obfuscation 3

Link to Challenge:
`https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Obfuscation-3`
Objective: The task is to find the password that is obfuscated within the JavaScript code.

2.9.1 Initial Observation

In this challenge, the objective was to find a password that had been obfuscated using JavaScript. Upon inspecting the webpage, I found that the password was hidden within a JavaScript function. The task was to analyze the code and extract the correct password.

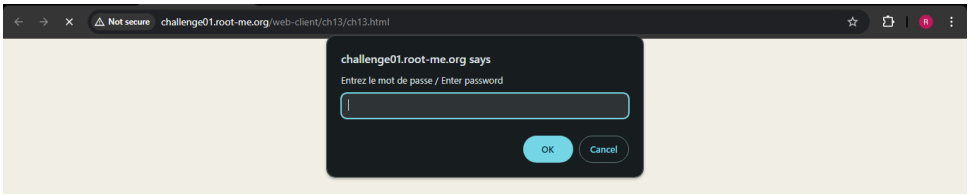

Figure 48: Find the password

2.9.2 Steps to Solve

1. **Inspect the Page**
   I opened the "Inspect Element" tool in my browser to analyze the HTML and JavaScript code.

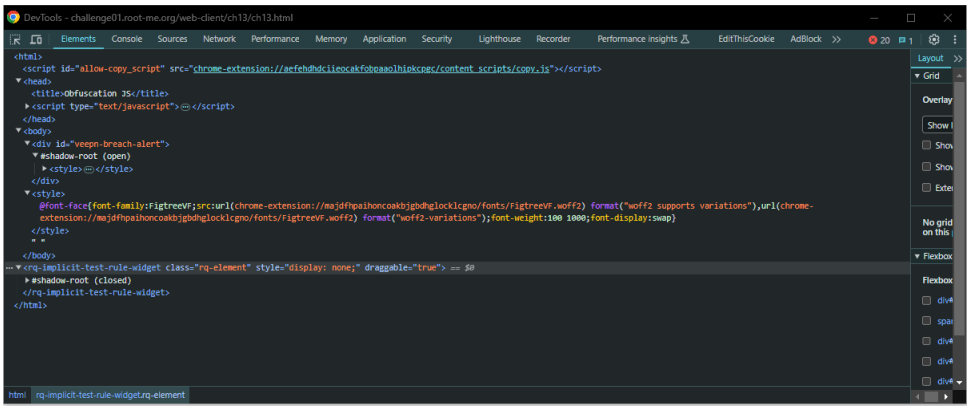

Figure 49: Inspecting the page's HTML code.

2. **Identify the JavaScript Function**
   I found a JavaScript function `dechiffre` in the source code. This function takes an encoded password and uses ASCII code conversions to decode it. However, the initial password attempt didn't work because of a hidden trick in the challenge.

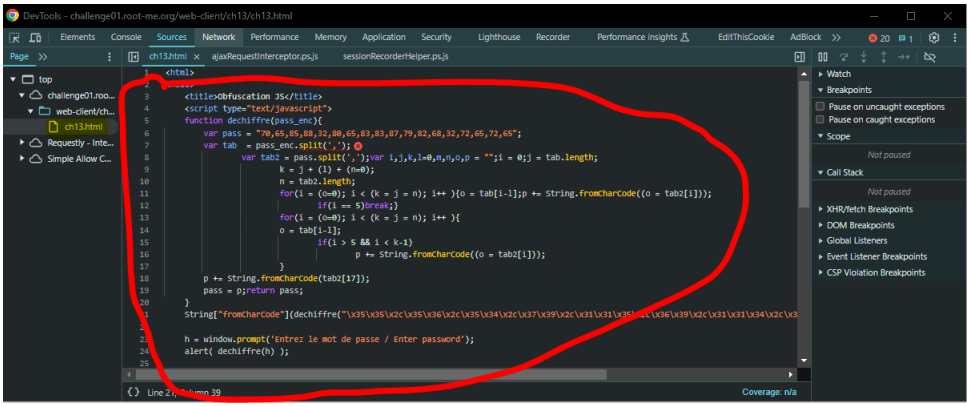

Figure 50: Attempting to decode the password using the provided function.

3. **Recognize the Trick**
   I realized that no matter what password I entered, the result would always show as "incorrect." This was part of the challenge's trick to mislead the solver into thinking the password was wrong.

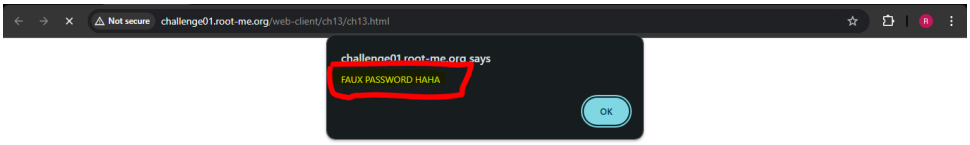

Figure 51

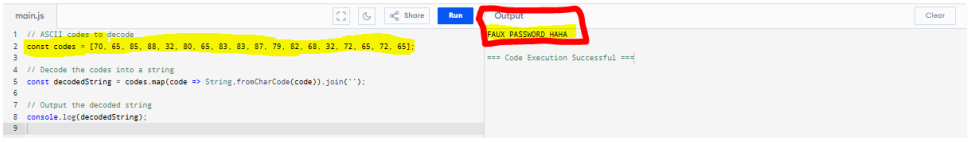

Figure 52: Decoded password.

4. **Find the Hidden Code**
   While further analyzing the script, I noticed an additional piece of code that revealed a new clue. By decoding this new string, I was able to uncover the correct password.

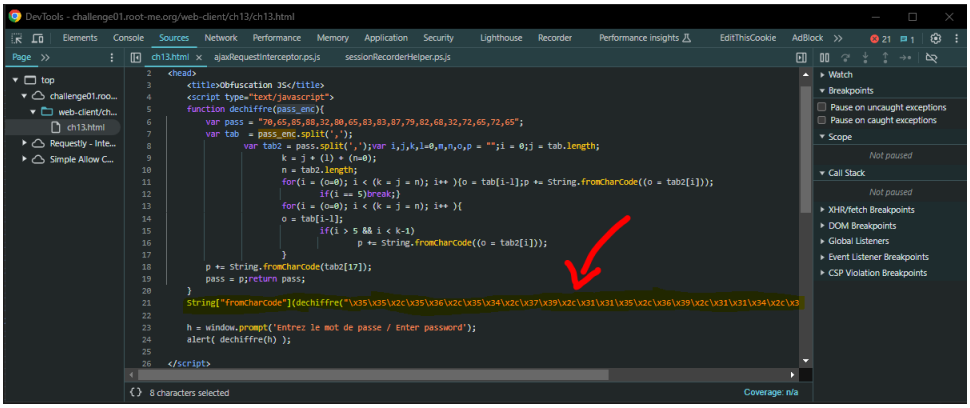

Figure 53: Piece of code

5. **Convert the Output to ASCII**
   I converted the decoded output into ASCII characters and found that the correct password was *****.

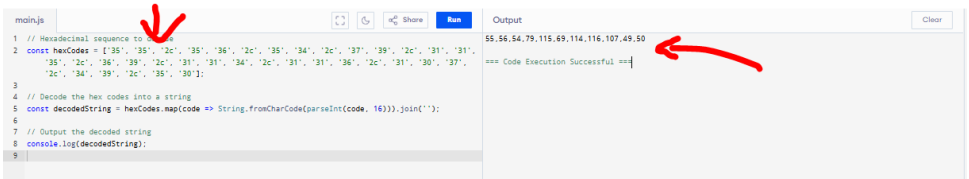

Figure 54

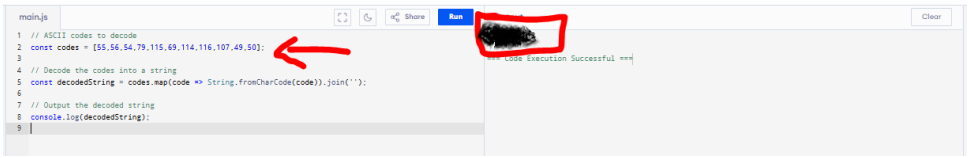

Figure 55: The final decoded output leading to the correct password.

6. **Test the Password**
   I entered the decoded password ******** into the challenge submission box, which was successfully accepted, completing the challenge.

2.9.3 Results

Password: ********
Points Earned: 30
Congratulations on solving the challenge!

2.10 Challenge 10: XSS - Stockée 1

Link to Challenge:
`http://challenge01.root-me.org/web-client/ch18/`
Objective: Steal the administrator's session cookies and use them to validate the challenge.

2.10.1 Initial Observation

The challenge is to exploit a stored Cross-Site Scripting (XSS) vulnerability to steal the administrator's session cookies. The image below shows the challenge's webpage where user input can potentially be injected into the site.

Figure 56: Challenge webpage for testing XSS vulnerability.

2.10.2 Steps to Solve

1. **Test for XSS Vulnerability**
   I injected a simple JavaScript code snippet, `<script>alert("TEST")</script>`, into the message box to check for XSS vulnerability.

Figure 57: Testing XSS by injecting a JavaScript alert.

2. **Successful XSS Execution**
   The JavaScript executed successfully, confirming that the site is vulnerable to XSS.

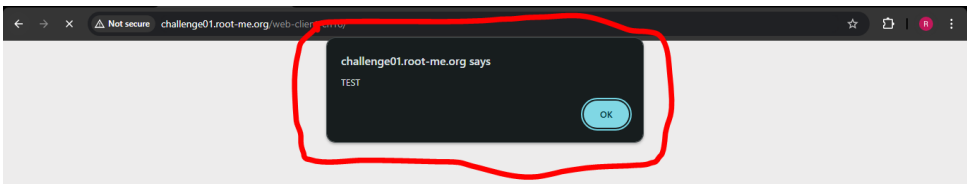

Figure 58: JavaScript alert executed successfully, confirming XSS vulnerability.

3. **Set Up Webhook for Logging Requests**
   I used `webhook.site` to generate a unique URL that logs all incoming requests. This URL will help capture the administrator's cookie when they visit the site.

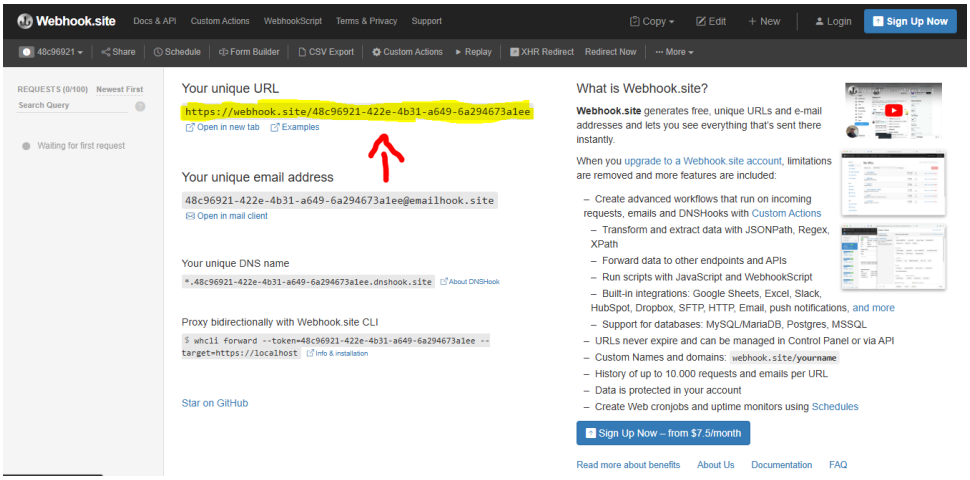

Figure 59: `webhook.site` interface showing the unique URL.

4. **Inject JavaScript for Cookie Stealing**
   I injected a JavaScript snippet to redirect users to the webhook URL. This script includes the administrator's cookies as part of the query string. The yellow-highlighted part in the script was replaced with the generated URL from `webhook.site`.

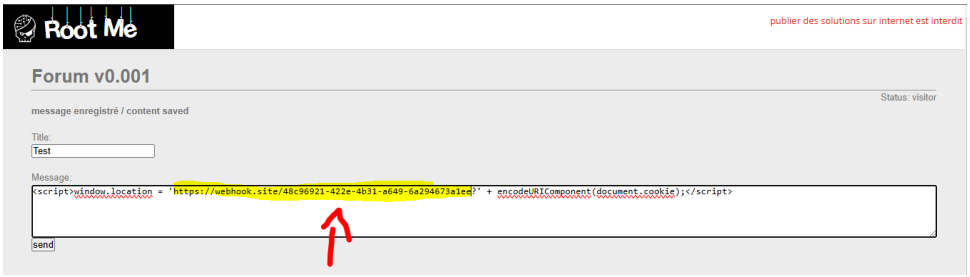

Figure 60: JavaScript injected to redirect users and steal cookies.

5. **Verify Redirection**
   After posting the comment with the script, I was redirected to the `webhook.site` URL, confirming that the injection worked as expected.

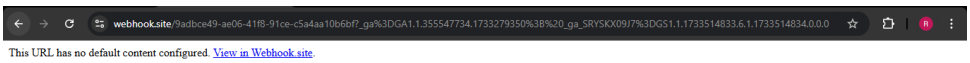

Figure 61: Redirection to `webhook.site` after JavaScript injection.

6. **Wait for Administrator's Visit**
   Back on `webhook.site`, I waited for a new request from the administrator. After a short while, a request containing the administrator's session cookies appeared.

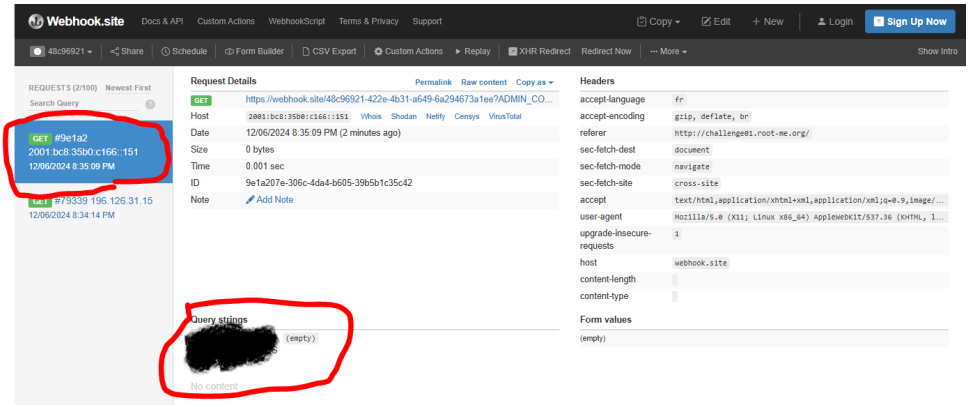

Figure 62: Administrator's request containing session cookies.

7. **Use the Cookies to Complete the Challenge**
   I copied the session cookie value and used it to validate the challenge on Root Me. This completed the challenge successfully.

2.10.3 Results

Points Earned: 30
Congratulations on solving the challenge!

3 References

Root Me Challenge: `https://www.root-me.org`