**247CTF Web Challenge: TRUSTED CLIENT**

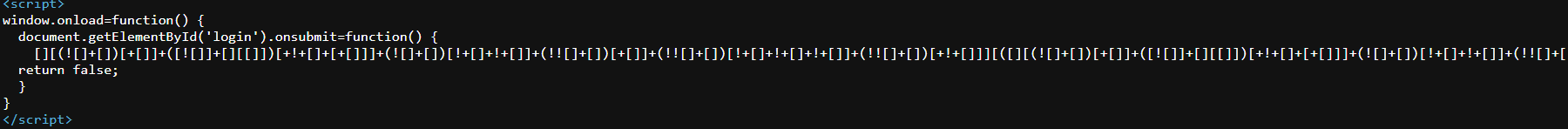
**Description**

Developers often prototype code quickly, sometimes storing credentials or sensitive data on the client side, assuming obfuscation alone will protect this information. This challenge explores the implications of such practices.

**Challenge Details**

The task is to assess a web application where credentials are stored on the client side. The challenge involves finding and accessing these credentials, which have been obfuscated.

**Steps to Solve**

1. **Access the Application:**
   * Start by accessing the web application provided for the challenge.
2. **View source code :**
   * + Ctrl + U to view source code , as we can the java script code is obfuscated 
3. **Look for Obfuscated Data:**
   * + By copyin a part of the code and sending it to <https://www.dcode.fr/> for the auto [**Cipher Identifier**](https://www.dcode.fr/cipher-identifier) , found that the language name is “JSFuck” .
     + The script is too large to be decoded by an online tool , so researching how this obfuscation works might help to create a decoder script but I’ve found a written code , I used <https://chatgpt.com/> to help me understand how the script works by adding comments :
     + //The python code explained :
     + import subprocess
     + import os
     + def unjsfuck(jsfuck\_code):
     + """
     + Decodes JSFuck code using Node.js.
     + Parameters:
     + - jsfuck\_code (str): The obfuscated JSFuck code to decode.
     + Returns:
     + - str: The decoded JavaScript code, or None if an error occurred.
     + """
     + try:
     + # Write the JSFuck code to a temporary JavaScript file.
     + # The code will use `eval` and `String.fromCharCode` to decode the JSFuck code.
     + with open('temp.js', 'w') as temp\_js\_file:
     + temp\_js\_file.write(f"console.log(eval(String.fromCharCode.apply(null, {jsfuck\_code})))")
     + # Run the temporary JavaScript file using Node.js.
     + # `subprocess.check\_output` executes the command and captures its output.
     + decoded\_js = subprocess.check\_output(["node", "temp.js"])
     + # Decode the output from bytes to string and remove any leading/trailing whitespace.
     + return decoded\_js.decode('utf-8').strip()
     + except subprocess.CalledProcessError as e:
     + # Handle any errors that occur during the execution of the Node.js script.
     + print(f"Error during execution: {e}")
     + return None
     + finally:
     + # Clean up by removing the temporary JavaScript file.
     + if os.path.exists('temp.js'):
     + os.remove('temp.js')
     + if \_\_name\_\_ == "\_\_main\_\_":
     + # Path to the input file containing the JSFuck code.
     + input\_file = "/path\_to/jsfucked/file.txt"
     + try:
     + # Open and read the JSFuck code from the input file.
     + with open(input\_file, 'r') as file:
     + jsfuck\_code = file.read()
     + # Decode the JSFuck code using the `unjsfuck` function.
     + result = unjsfuck(jsfuck\_code)
     + if result:
     + # Print the decoded JavaScript code.
     + print("Decoded JavaScript code:")
     + print(result)
     + else:
     + print("Failed to decode the JSFuck code.")
     + except FileNotFoundError:
     + # Handle the case where the input file is not found.
     + print(f"File not found: {input\_file}")
     + except Exception as e:
     + # Handle any other exceptions that may occur.
     + print(f"An error occurred: {e}")

Just create a file and paste the obfuscated script and save it as .txt and replace the path in the code to decode it , heres the result : if (this.username.value == 'the\_flag\_is' && this.password.value == '247CTF{xxxx}'){ alert('Valid username and password!'); } else { alert('Invalid username and password!'); }