Nuclear Code

1. Challenge Overview:

The challenge presents a **web application** that may contain hidden data or vulnerabilities. The task is to identify and extract this hidden information, or exploit the vulnerabilities in the web application to retrieve a flag.
First, I proceed to explore the web application by accessing the provided URL and inspecting the HTML source code for any unusual clues or hidden data.

- **Challenge Type**: Web Application
- **Challenge Type**: Web Forensics (likely involving **SQL injection**, **XSS**, **hidden files**, etc.)

Test for Vulnerabilities:

The next step is to check for any common **web application vulnerabilities** that could lead to the extraction of hidden data or the flag.

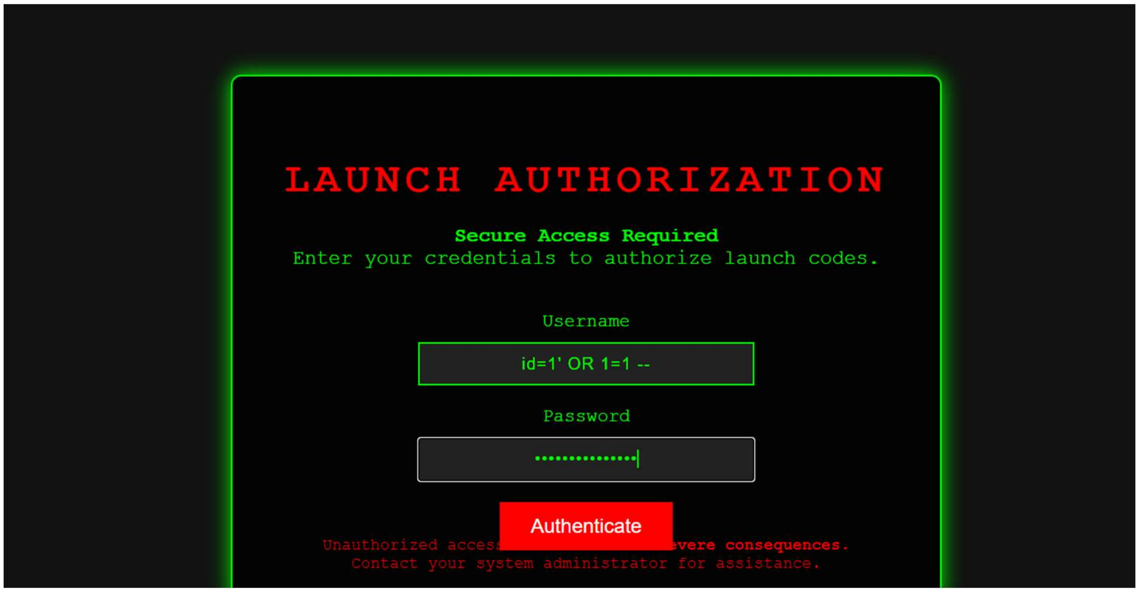
2. SQL Injection:

Check if the web application is vulnerable to **SQL Injection** by manipulating URL parameters.

Action:

- Look for URL parameters like id, search, user, etc.
- Try common **SQL Injection payloads**:
    - id=1' OR 1=1 --
    - id=1' UNION SELECT null, username, password FROM users --

Result from injection : id=1' OR 1=1 --
Then I also tried several creds such as ‘or’’=’ , admin:password and failed miserably!



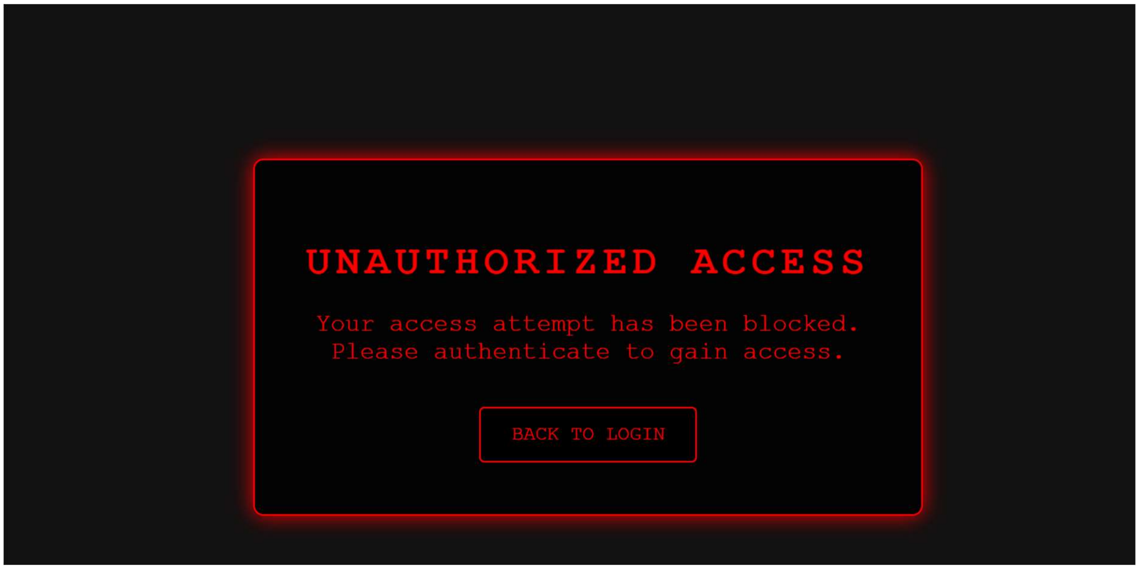
LAUNCH AUTHORIZATION

Secure Access Required
Enter your credentials to authorize launch codes.

Username: id=1' OR 1=1 --

Password: •••••••••••••••

Unauthorized acces **evere consequences.**
Contact your system administrator for assistance.



UNAUTHORIZED ACCESS

Your access attempt has been blocked.
Please authenticate to gain access.

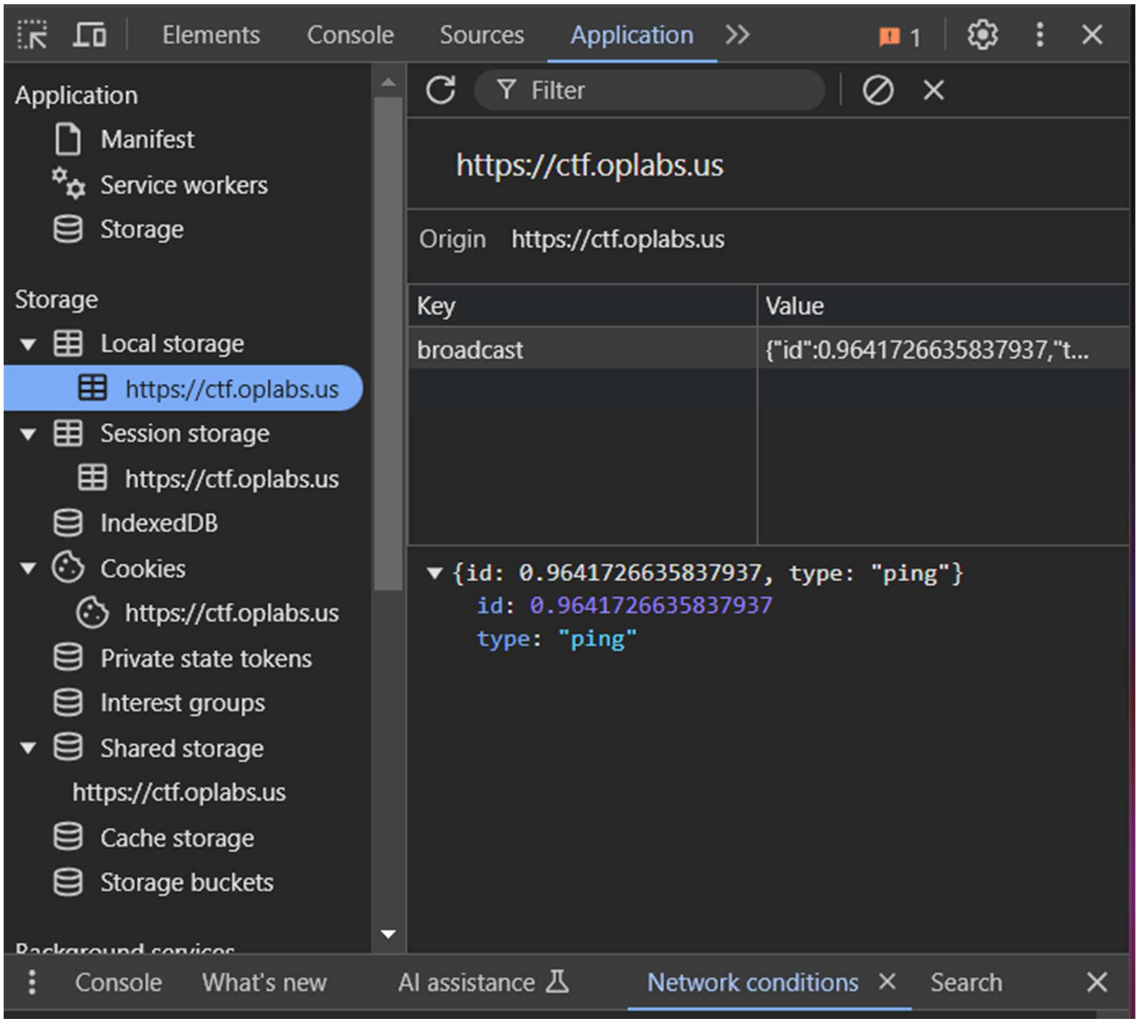
3. Inspect Cookies and LocalStorage:

Sometimes flags are hidden in **browser storage** mechanisms like **cookies** or **localStorage**.

Action:

- Open the **Developer Tools** in your browser (F12 or Ctrl + Shift + I).
- Go to the **Application** tab.
  - Check **Cookies** and **LocalStorage** for any strange keys or values.

Still negative result:



https://ctf.oplabs.us

Origin https://ctf.oplabs.us

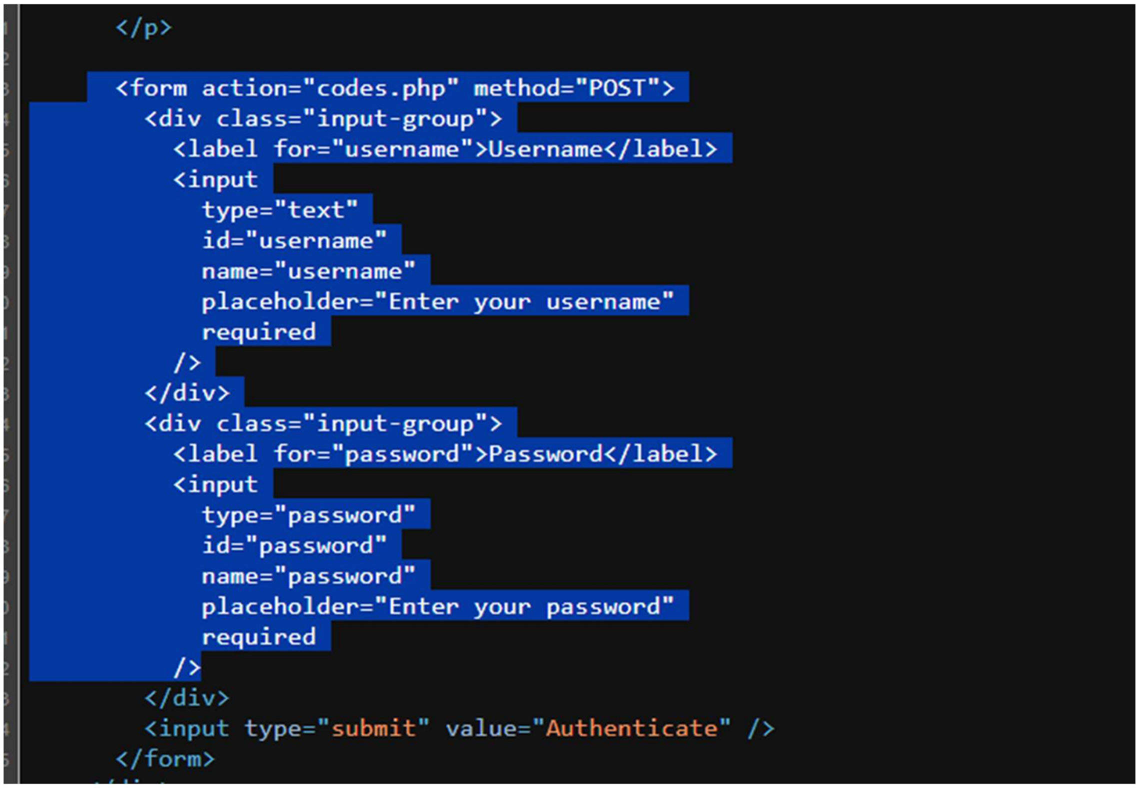
Key	Value
broadcast	{"id":0.9641726635837937,"t...

```
▼{id: 0.9641726635837937, type: "ping"}
   id: 0.9641726635837937
   type: "ping"
```

4. Inspect the Page Source:

After loading the page, I inspect the **HTML source code** for any clues or hidden data. Sometimes, flags or hints are stored directly in the HTML, or comments may contain valuable information.

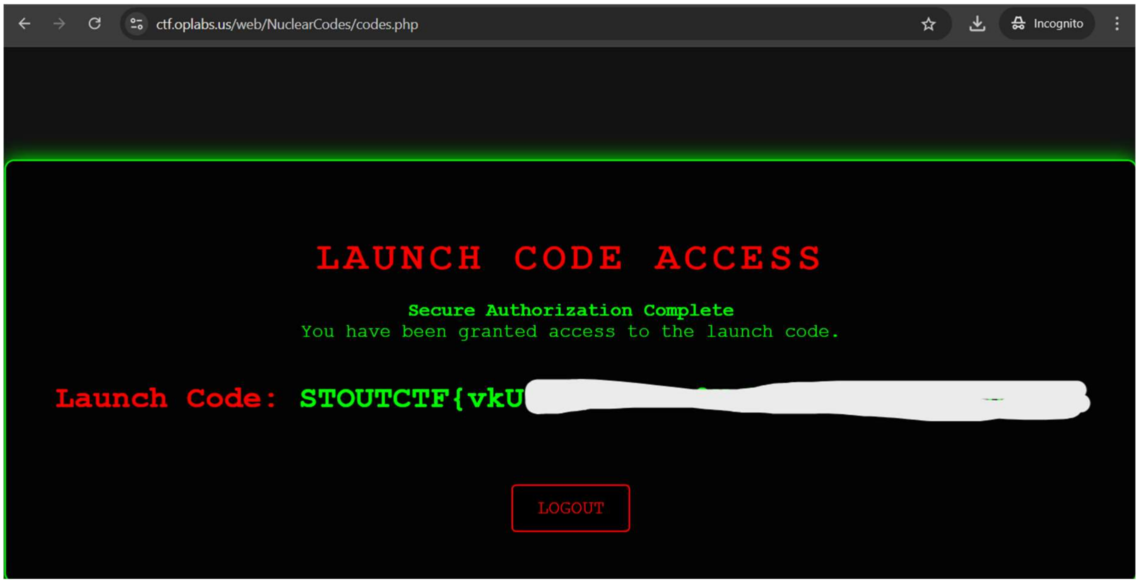
- **Command** (Browser): Right-click → **View Page Source**
- **Result**: While inspecting the page, I find an unusual line which sending all creds to the php page and I think this is not coincidence that they don't hide this one!



```
    </p>

    <form action="codes.php" method="POST">
      <div class="input-group">
        <label for="username">Username</label>
        <input
          type="text"
          id="username"
          name="username"
          placeholder="Enter your username"
          required
        />
      </div>
      <div class="input-group">
        <label for="password">Password</label>
        <input
          type="password"
          id="password"
          name="password"
          placeholder="Enter your password"
          required
        />
      </div>
      <input type="submit" value="Authenticate" />
    </form>
```

My curiosity got me there I tried browsing to https://ctf.oplabs.us/web/NuclearCodes/codes.php and the launch code is here!



LAUNCH CODE ACCESS

Secure Authorization Complete

You have been granted access to the launch code.

Launch Code: STOUTCTF{vkU[redacted]

Conclusion

This challenge was straightforward and demonstrated the importance of always **viewing the source code** of a web page, as flags are sometimes **deliberately exposed** there for CTF participants to find.