Part 3: Secure programming

Secure programming recommendation 1: Encryption for User Credentials

Problem:

The client uses the GET method to send the new usernames and passwords to the server in text format without encryption, per the brief. Because it makes it simple to steal the data while it is being transmitted, it is not safe.

Solution:

Encrypt data between the client and server using HTTPS to fix this. To send sensitive data outside of the URL, use the POST method instead of the GET method. To enforce secure connections, install a TLS certificate on the server and include security headers like Strict-Transport-Security.

Secure programming recommendation 2: Storing of Passwords

Problem:

The brief states that passwords are currently kept in the database in plain text, which makes it extremely vulnerable to hacking. The application should store passwords as cryptographically strong one-way salted hashes in accordance with secure coding practices.

Solution:

Passwords should be stored as secure hashed values with a distinct random salt. Make use of robust hashing techniques like bcrypt. Make sure the hashing takes place on the server, generate a new salt for every password, and use a reliable library to manage the hashing during registration and login.

Secure programming recommendation 3: Validation of Input

Problem:

The brief stated that a login page uses a database to verify the username and password, but it made no mention of whether the input is verified for security. Attackers may enter malicious data that could harm the database if the input is not properly verified.

Solution:

The app should verify the username and password to make sure they only contain safe characters to fix this. It ought to make use of secure techniques like parameterized queries. To guarantee that only legitimate data is processed, it should reject any input that contains malicious characters or commands.

Secure programming recommendation 4: Improvement of Authentication

Problem:

The brief states that the application checks the username and password using a login page without any protections against account hacking or brute force attacks.

Solution:

To solve this, enforce strong passwords and implement account lockouts following multiple unsuccessful login attempts. Keep track of unsuccessful login attempts and, if too many are made, temporarily lock accounts. Ensure that all passwords contain capital and lowercase letters, numbers, and special characters. Notify users of unsuccessful login attempts to alert them to potential illegal access.

Secure programming recommendation 5: Management of sessions

Problem:

The brief states that the application checks usernames and passwords using a database via a login page, but it makes no mention of how session management is managed.

Solution:

This can be fixed by marking session identifiers as HttpOnly and Secure and storing them in secure cookies. After logging in, modify the session identifier to stop hackers from taking control of it. Configure session management to automatically expire sessions after a period of inactivity and to include security flags in cookies. Using a library with robust session management features can help ensure that sessions are handled correctly.