Logical and Security Testing Soar Flask App

Security Vulnerabilities

/client_registeration

Weak Password Policy

- **Description**: Users are allowed to set weak passwords (e.g., without requiring a mix of characters, length, or complexity), and attackers can easily crack them using brute-force attacks.
- **Check**: Test the registration flow by entering weak passwords (e.g., "1234", "password").
- **Severity**: Medium
- **Risk**: 6/10
- **Recommendation**: Implement a stronger password policy with requirements for length, complexity, and special characters.

Account Enumeration via Different Error Messages

- **Description**: Different error messages are returned for non-existent vs. existing users, which allows an attacker to enumerate valid usernames.
- **Check**: Test the /client_registeration API with various non-existent and existing usernames, and check if the error messages differ.
- **Severity**: High
- **Risk**: 8/10
- **Recommendation**: Return generic error messages to prevent account enumeration.

Lack of Rate Limiting

- **Description**: No rate limiting on the registration endpoint, making it vulnerable to attackers flooding the API with requests, potentially causing a denial of service (DoS).
- **Check**: Attempt multiple rapid registrations in a short period to see if the system rate limits these requests.
- **Severity**: Medium

- **Risk**: 7/10
- **Recommendation**: Implement rate limiting to prevent abuse.

/client_login

Unencrypted Password Storage

- **Description**: Passwords are stored in plaintext in the database, making them accessible if the database is compromised.
- **Check**: Check if the password is being hashed before storage. You can check the backend code or database schema.
- **Severity**: Critical
- **Risk**: 10/10
- **Recommendation**: Implement password hashing for secure password storage.

Session Management Vulnerabilities

- **Description**: The session token is not properly secured or invalidated, allowing attackers to hijack or reuse sessions.
- **Check**: After logging in, test the session cookie/token management. Ensure it's stored securely (HTTPOnly, Secure, SameSite flags for cookies) and invalidated after logout.
- **Severity**: High
- **Risk**: 9/10
- **Recommendation**: Ensure session tokens are securely stored and properly invalidated upon logout.

Brute Force Protection on Login

- **Description**: The login endpoint does not prevent multiple failed login attempts, making it vulnerable to brute-force attacks.
- **Check**: Test if the login API allows for multiple failed login attempts without delay or lockout.
- **Severity**: High
- **Risk**: 8/10
- **Recommendation**: Implement account lockout or CAPTCHA after multiple failed login attempts.

Cross-Site Scripting (XSS) in Login Fields

- **Description**: The login form is vulnerable to XSS attacks, allowing attackers to inject malicious scripts into the application.
- **Check**: Test the login form by inputting a script tag like <script>alert('XSS')</script> in the username or password field.
- **Severity**: Medium
- **Risk**: 7/10
- **Recommendation**: Sanitize and validate all user inputs to prevent XSS vulnerabilities.

Logical Vulnerabilities

/client_registeration

Duplicate Registration

- **Description**: The system allows the client to register multiple times with the same username and it may cause issues with authentication or duplicate data.
- **Check**: Attempt to register the same client multiple times using the same username.
- **Severity**: Medium
- **Risk**: 6/10
- **Recommendation**: Enforce unique email/username validation during registration.

No Proper Validation for the Form Fields

- **Description**: The system allows the client to register with invalid data such as email does not have any validation.
- **Check**: Attempt to register the client using email as a ‘test’.
- **Severity**: Medium
- **Risk**: 6/10
- **Recommendation**: Enforce proper data validation for all the form fields.

/client_login

Login Success on Incorrect Credentials

- **Description**: The system incorrectly logs in users even with invalid usernames or does not show an appropriate error message which can lead to attackers gaining unauthorized access.
- **Check**: Test with an incorrect username and observe the response for inconsistencies.
- **Severity**: Critical
- **Risk**: 10/10
- **Recommendation**: Ensure proper validation of credentials during login and return a consistent error message for failed logins.