Route: `/client_registeration`

1. **SQL Injection - Email Check**
    - **Description**: The SQL query to check for an existing email uses string concatenation, making it vulnerable to SQL injection.
    - **Risk Score**: High
    - **Recommendation**: Use parameterized queries & sanitize input to prevent SQL injection.

2. **SQL Injection - User Registration**
    - **Description**: The SQL query that inserts a new user uses string concatenation, making it vulnerable to SQL injection.
    - **Risk Score**: High
    - **Recommendation**: Use parameterized queries & sanitize input to prevent SQL injection.

3. **Password in Plain Text**
    - **Description**: Passwords are stored in the database without hashing or encryption, posing a security risk if the database is compromised.
    - **Risk Score**: Critical
    - **Recommendation**: Store passwords securely using hashing algorithms (e.g., bcrypt, Argon2).

4. **Missing Input Validation for email**
    - **Description**: The email address is not validated for proper format, leading to potential issues with invalid email entries.
    - **Risk Score**: Medium
    - **Recommendation**: Add proper email format validation.

5. **Missing Input Validation for phone**
    - **Description**: The `phone` field is not validated for format or required length, which can result in inconsistent or malformed data being stored.
    - **Risk Score**: Low
    - **Recommendation**: Validate the phone number format.

6. **Weak Password Enforcement**
    - **Description**: No password strength checks are performed, allowing weak or common passwords.
    - **Risk Score**: High
    - **Recommendation**: Enforce strong password policies (such as minimum length, mix of characters).

7. **Lack of HTTPS Enforcement**
    - **Description**: The route is vulnerable to man-in-the-middle attacks as it does not enforce HTTPS for secure transmission of sensitive data.
    - **Risk Score**: High
    - **Recommendation**: Enforce HTTPS for all routes.

8. **Sensitive Data Exposure (No Encryption for Database)**
    - **Description**: Sensitive data such as passwords are stored in plain text within the database without encryption.
    - **Risk Score**: Critical
    - **Recommendation**: Encrypt sensitive data before storage.

9. **No Rate Limiting**
    - **Description**: The registration endpoint is vulnerable to brute force attacks due to lack of rate limiting.
    - **Risk Score**: High
    - **Recommendation**: Implement rate limiting to prevent potential DDoS & brute force attacks.

10. **No Logging of Failed Registration or Login Attempts**
    - **Description**: Failed registration and login attempts are not logged, making it difficult to detect abuse or suspicious behavior.
    - **Risk Score**: Medium
    - **Recommendation**: Log failed registration and login attempts to monitor for potential attacks.

11. **No Email Confirmation**
    - **Description**: There is no email verification step, meaning users can register with invalid or fake emails.
    - **Risk Score**: Medium
    - **Recommendation**: Implement email confirmation for user registration.

12. **No User Input Sanitization**
    - **Description**: User input is not sanitized, allowing potential XSS (Cross-Site Scripting) attacks.
    - **Risk Score**: Medium
    - **Recommendation**: Sanitize user inputs before storing them in the database.

Route: `/client_login`

1. **SQL Injection - Email and Username Check**
    - **Description**: Both the email and username-based SQL queries are vulnerable to SQL injection due to string concatenation.
    - **Risk Score**: Critical
    - **Recommendation**: Use parameterized queries & sanitize input to safely query the database.

2. **Password in Plain Text**
    - **Description**: Passwords are transmitted and checked in plain text, which can be intercepted by attackers.
    - **Risk Score**: Critical
    - **Recommendation**: Always use HTTPS and hash passwords before transmission.

3. **JWT Secret Hardcoded**
    - **Description**: The secret key used to sign JWT tokens is hardcoded in the code, making it susceptible to leakage and attack.
    - **Risk Score**: High
    - **Recommendation**: Use environment variables or a secure key management solution for JWT secrets (such as AWS Secrets Manager or Azure Key Vault).

4. **No JWT Expiry**
    - **Description**: The JWT token does not have an expiration time, making it vulnerable to replay attacks if intercepted.
    - **Risk Score**: High
    - **Recommendation**: Set an expiration time for the JWT token.

5. **JWT No Signature Verification**
    - **Description**: The JWT token is decoded without verifying its signature, which could allow attackers to modify the payload.
    - **Risk Score**: Critical
    - **Recommendation**: Always verify the JWT signature before decoding the payload.

6. **No User Lockout Mechanism**
    - **Description**: The app does not implement an account lockout mechanism after multiple failed login attempts, allowing brute force attacks.
    - **Risk Score**: High
    - **Recommendation**: Implement a lockout mechanism after several failed login attempts. Add logging to be able to more reliably identify and stop malicious users.

7. **No Rate Limiting for Login**
    - **Description**: The login route is vulnerable to brute force attacks due to the lack of rate limiting.
    - **Risk Score**: High
    - **Recommendation**: Implement rate limiting for login attempts. Add logging to be able to more reliably identify and stop malicious users.

8. **Potential for Unauthorized Access (Email/Username Mix)**
    - **Description**: The app allows login using either email or username, but fails to clearly separate the two methods, potentially allowing unauthorized access.
    - **Risk Score**: Medium
    - **Recommendation**: Clearly define and validate the use of either email or username for login. Also ensure during registration that there is uniqueness of both email and username if using either for login.

9. **No Two-Factor Authentication (2FA)**
    - **Description**: The login route does not support two-factor authentication (2FA), which increases the risk of unauthorized access.
    - **Risk Score**: High
    - **Recommendation**: Implement two-factor authentication for added security.

10. **Session Management Missing**
    - **Description**: The app does not manage sessions after successful login, allowing for unauthorized access if the JWT is intercepted.
    - **Risk Score**: High
    - **Recommendation**: Implement session management for authenticated users.

11. **No Logging of Failed Login Attempts**
    - **Description**: Failed login attempts are not logged, making it difficult to detect brute force or unauthorized access attempts.
    - **Risk Score**: Medium
    - **Recommendation**: Log failed login attempts for monitoring purposes.

12. **No Secure Cookie Settings for JWT**
    - **Description**: JWT tokens may be stored in an insecure cookie without the HttpOnly and Secure flags set, making them vulnerable to theft via XSS.
    - **Risk Score**: Critical
    - **Recommendation**: Set the HttpOnly and Secure flags on JWT cookies.

13. **No Security Headers**
    - **Description**: The app does not set important HTTP security headers like `Content-Security-Policy`, `Strict-Transport-Security`, etc.
    - **Risk Score**: Medium
    - **Recommendation**: Set appropriate security headers for all responses.

14. **No Anti-CSRF Mechanism**
    - **Description**: The login process lacks protection against Cross-Site Request Forgery (CSRF), making it vulnerable to attacks from malicious sites.
    - **Risk Score**: High
    - **Recommendation**: Implement CSRF protection mechanisms for login and other sensitive actions, potentially add CSRF token.