Secure Authentication System Development

Iteration Summary and Future Plans

Faizan ul Haq(21i1771), Mubashir Zaidi(21i1764), Maria Khan(21i1352) DS-M

October 15, 2024

Contents

1 Introduction

This report outlines the progress made in Iteration 1 of the Secure Authentication System Development project and provides an overview of the upcoming work planned for Iterations 2, 3, and 4.

2 Iteration 1: Environment Setup and Basic User Authentication

2.1 Summary of Work Completed

In Iteration 1, we established the foundational components of the secure authentication system:

- **Database Configuration**:
  - Initialized a SQLite database to store user data.
  - Defined the `users` table schema with fields for username, email, and securely hashed passwords.
- **Basic User Authentication**:
  - Implemented user registration functionality that collects username, email, and password.
  - Ensured passwords are securely stored using bcrypt hashing.
  - Developed user login functionality that authenticates users based on email and password.
  - Included basic session management to keep users logged in after authentication.
  - Added logout functionality to allow users to end their sessions.
- **Frontend Implementation**:
  - Incorporated custom HTML and CSS for the login and registration pages, enhancing the user interface.
  - Ensured forms capture user input securely and interact correctly with backend routes.
- **Application Routing**:
  - Configured the Flask application to land on the login page by default.
  - Set up navigation between login, registration, and dashboard pages.

2.2 Key Achievements

- Successfully set up a secure development environment.
- Implemented secure password storage using bcrypt.
- Created a user-friendly interface with custom styling.
- Established session management for user authentication.

3 Iteration 2: Email Verification and Enhanced Session Management

3.1 Planned Work

In Iteration 2, the focus will be on enhancing the authentication system by adding email verification and improving session management:

- **Email Verification**:

  - Implement email sending functionality.
  - Modify the registration workflow to send a verification email containing a confirmation link.
  - Develop a route to handle account activation when the user clicks the confirmation link.
  - Update the user model to include an activation status, preventing login until the email is verified.

- **Enhanced Session Management**:
  - Ensure that session data is securely stored and managed.
  - Implement measures to protect session cookies from being compromised.
  - Provide clear messages to users about the status of their account and the need for email verification.
  - Redirect users appropriately based on their activation status.

- **Frontend Updates**:
  - Update templates to inform users to check their email for the verification link.
  - Add pages or messages for account activation success or failure.
  - Display informative errors if users attempt to log in without verifying their email.

4 Iteration 3: OTP Integration for Two-Factor Authentication

4.1 Planned Work

In Iteration 3, we will integrate One-Time Password (OTP) functionality to add an extra layer of security:

- **OTP Generation and Verification**:
  - Implement a secure method to generate OTPs.
  - Send OTPs to users' registered email addresses after successful password authentication.
  - Allow users to request a new OTP if the previous one expires.

- **Login Workflow Enhancement**:
  - Modify the login process to include OTP verification as a second step.
  - Update the frontend to include an OTP input field after password authentication.
  - Ensure that the OTP verification process is user-friendly and secure.

- **Security Enhancements**:
  - Securely store and handle OTPs to prevent reuse or interception.
  - Implement safeguards against OTP brute-force attacks.

5 Iteration 4: Security Features, Testing, and Documentation

5.1 Planned Work

In Iteration 4, the focus will be on bolstering security, thorough testing, and preparing final documentation:

- **Advanced Security Measures**:
  - Implement rate limiting on login attempts and OTP requests using tools like `Flask-Limiter`.
  - Use parameterized queries to prevent SQL injection attacks.
  - Ensure sensitive data is not exposed in logs or error messages.

- **Comprehensive Testing**:
  - Perform functional testing of all user flows to ensure they work as intended.
  - Conduct security testing, including penetration testing to identify and fix vulnerabilities.
  - Verify that password hashing and OTP handling are secure.
- **Documentation and Final Deliverables**:
  - Finalize the codebase with comments and documentation.