Secure Authentication System Development

Department of Data Science

Information Security (CS3002)

Group Members:

Muhammad Shamil Umar (21i-1786)

Hamiz Ahmed Siddiqui (21i-1678)

Afnaan Asif (21i-1387)

Contents

Chapter 1

Objective

This report aims to display and outline the structure of the database and describe the workflow for the implementation of this Secure Authentication application using Two-Factor Authentication (2FA for short). The application will ensure that only the verified users can access the website, with a simple yet complex 2FA setup as well as other security measures.

1.1 System Overview

- **Database Design:**

  In this iteration, focus is on creating the ground level structure for the application's data. This involves defining the necessary entities, such as tables for users, roles, permissions, and sessions, and establishing their relationships. Indexes are designed for efficient querying, particularly for key fields like user IDs and session tokens. Considering security of data, encryption of sensitive data like password will be implemented.

- **Front End:**

  During the front-end development iteration, our primary goal will be to build a responsive and user-friendly interface which will be implemented using React and Tailwind CSS. Forms for user registration, login, and profile management will be implemented. Front-end validation and error handling are also part of this iteration to ensure smooth user experiences.

- **Backend (Flask API):**

  The backend development iteration focuses on designing server API using Flask. API

routes will be created for user management tasks, such as registration, login etc. Data validation and error handling are also part of this iteration to ensure accurate API responses. Authentication is handled using token-based methods which ensures secure access to the application.

- **Two-Factor Authentication (2FA):**

  This iteration enhances the application's security by integrating two-factor authentication. Users are provided with options like Time-based One-Time Passwords (TOTP) via email for additional protection during login. Fallback mechanisms, such as recovery codes, are implemented to ensure users can regain access if they lose their 2FA device. Sensitive 2FA-related data is securely stored, and logging is set up to track 2FA activities for auditing purposes, ensuring a secure and seamless experience for users.

- **Session Management & Rate Limiting:**

  This iteration focuses on improving both security and performance. Secure session tracking is implemented, using HTTP-only cookies to manage user sessions, with expiration policies to automatically log out inactive users. Rate limiting is introduced to prevent API abuse, setting limits on the number of requests per user or IP address, especially for sensitive actions like login attempts.

Chapter 2

System Overview

The system includes three main stages:

1. **User Registration and Email Verification:** Users will register by providing their emails, and once that email is verified and the registration process is completed, the users will then be able to access the website.
2. **Account Creation:** After the email has been verified, the user's details are transferred to the main user database.
3. **Two-Factor Authentication (2FA):** After a successful login has been made, users will be required to verify their identity using a token sent via email.

2.0.1 Database Structure (Current Iteration)

The following database schema has been designed to support the workflow described above:

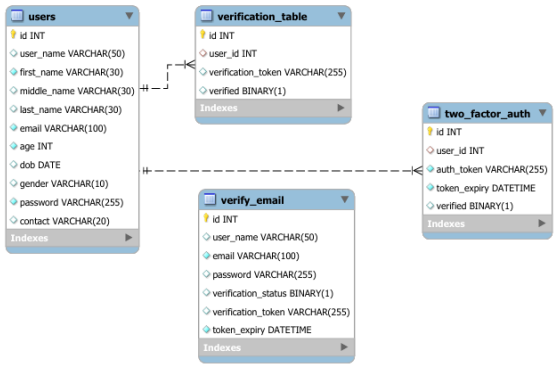

Figure 2.1: Your image caption here

verify_email Table (Temporary Storage)

- This table temporarily stores user details (email, username, password) when they first register.
- A verification email is sent with a token to confirm that the user has provided a valid email.

users Table (Main User Table)

- Once a user has verified their email, their details will be moved to this table.
- This table stores complete user information such as name, age, and gender.

verification_table (Email Verification Token Table)

- This table stores verification tokens for users who have successfully registered but need to verify their email address.

two_factor_auth (Two-Factor Authentication Table)

- This table stores details related to 2FA for users. After a user logs in, a 2FA token is generated and sent to their email. The user must provide the correct token to proceed.

2.1 Workflow

2.1.1 User Registration and Email Verification

1. **Registration:** Users submit a username, email, and password. This data is temporarily stored in the verify_email table.
2. **Email Verification:** A verification email is sent with a unique token. The token and its expiry are stored in verify_email.
3. **Verification Confirmation:** Once the user has verified their email, a token will be generated and compared with the verify_email table. If it is valid the user will be verified and fully registered.

2.1.2 Login and Two-Factor Authentication

1. **Login:** The user provides their email and password. After successful authentication, a 2FA token is generated and sent via email.
2. **Two-Factor Authentication:** The user must enter the 2FA token. The token is verified against the two_factor_auth table, and if the token is valid and not expired then the user will be logged in.

2.2 Security Considerations

- **Password Hashing:** Passwords in both the verify_email and users tables are stored as hashed values to ensure security.
- **Token Expiry:** Both the email verification tokens and 2FA tokens have expiry times to limit their validity and prevent misuse.
- **Unique Tokens:** Tokens for email verification and 2FA are generated as unique, random values to ensure that each user has a different token.

2.3 Conclusion

This database design aims to secure the web application by creating a secure layer of protection with email verification and then 2FA. Only the secure and verified emails will be able to actually

use the website and those users passing 2FA will be able to log into the website. This will allow the system to be secure against any sort of threats of insecure access.