Group Member 1 : Jaideep Singh (2210110675)

Group Member 2 : Akshit Baliyan (2310110029)

Programming Assignment

(Implementing a Password Manager)

1. Introduction

This report details the design and implementation of a password manager application. The application provides secure storage and retrieval of user passwords for various websites and services. Key features include master password authentication, AES encryption, password generation, and password strength checking.

2. Library Requirements

Pip , tkinter , bcrypt , pyperclip , xclip

3. Encryption Technique

The password manager employs AES encryption via the Fernet library from the `cryptography` package to protect stored passwords. A unique encryption key is derived from the user's master password using PBKDF2HMAC.

*   Key Derivation: PBKDF2HMAC (Password-Based Key Derivation Function 2 with HMAC) is used to generate a strong encryption key from the master password. This involves salting and

hashing the master password iteratively, making it computationally expensive for attackers to derive the key even if they obtain the salt. The salt is randomly generated during the initial setup.

* AES Encryption: The Fernet library provides a high-level interface to AES encryption. It handles the complexities of key management, initialization vectors (IVs), and authenticated encryption. Each password and the entire password store is encrypted using this key.

* Encryption at Rest: All passwords and associated data are stored in an encrypted format on disk. This ensures that even if the storage file is compromised, the passwords remain protected without the master password.

4. Design Choices

* Modular Design: The application is designed with a modular architecture, separating concerns such as authentication, encryption, storage, password generation, and GUI. This enhances maintainability and testability.

* Authentication (auth.py): The `MasterPasswordAuth` class handles user authentication. It stores a salted hash of the master password using bcrypt. Upon successful authentication, an encryption key is derived.

* Encryption (encryption.py): The `PasswordEncryption` class uses the derived key to encrypt and decrypt individual passwords and the entire password store. It utilizes Fernet for AES encryption.

* Password Storage (password_store.py): The `PasswordStore` class manages the storage of encrypted passwords in a file. It provides methods for adding, retrieving, updating, and deleting password entries. Data is serialized to JSON before encryption.

* Password Generation (password_generator.py): The `PasswordGenerator` class generates strong, random passwords based on user-defined criteria (length, character types).

* Password Strength (password_strength.py): The `PasswordStrengthChecker` class assesses the strength of passwords based on length, character diversity, and common patterns.

* GUI (password_manager_gui.py): The `PasswordManagerGUI` class provides a user-friendly interface using Tkinter. It allows users to interact with the password manager, manage their passwords, generate new passwords, and check password strength.

5. Security Considerations

*   Master Password Strength: The security of the entire system relies on the strength of the master password. Users are advised to choose a strong, unique password.

*   Key Derivation: PBKDF2HMAC with a high iteration count is used to make brute-force attacks on the master password more difficult.

*   Memory Security: The application does not store the master password or encryption key in memory longer than necessary.

*   Clipboard Security: When copying passwords to the clipboard, users should be aware of the risk of other applications accessing the clipboard data. The application provides a warning about this.

*   Dependency Management: The application relies on external libraries (`bcrypt`, `cryptography`, `pyperclip`). Keeping these libraries up-to-date is crucial for addressing potential security vulnerabilities.

*   Common Password Check: The application checks against a list of common passwords to prevent users from using easily guessable passwords.

6. Example Usage, Test Cases, and User Interface

This section provides an overview of how to use the password manager, including setting up a master password, adding and retrieving passwords, generating secure passwords, and checking password strength. Additionally, it outlines the tests performed to verify the functionality and security of the application.

Demonstration Video : Link

For example:

- **Setting Up a Master Password:** If the entered password is too weak (e.g., "`abc`"), a warning is displayed, prompting the user to re-enter a stronger password.
- **Adding a New Password:** Users can store credentials (e.g., "`Twitch.com`") and check password strength in real time. A **"Generate Password"** option is available for creating strong passwords.

- **Retrieving a Stored Password:** Selecting a saved entry (e.g., `"Twitch.com"`) and clicking **"View Details"** displays the password, with an option to copy it to the clipboard.
- **Generating a Secure Password:** The password manager generates strong passwords containing uppercase/lowercase letters, digits, and special characters, which can be copied to the clipboard.
- **Checking Password Strength:**
  `"password123"` → **Weak Password**
  `"a1B2c3D4e5F6g7H8i9J0!"` → **Strong Password**

User Interface :

The GUI provides an intuitive way to manage passwords. It includes features such as:

* A login screen for master password authentication.

* A password list view to display stored passwords.

* An add/edit password form for creating and modifying password entries.

* A password generator with customizable options.

* A password strength checker to evaluate password security.

7. Conclusion

This password manager provides a secure and convenient way for users to store and manage their passwords. By using strong encryption and a well-designed architecture, it protects sensitive data from unauthorized access. Future improvements could include features such as two-factor authentication, cloud synchronization, and browser integration.