**Aleena**

**20BCE0200**

Contents –

1. Abstract
2. Introduction

* Background and Motivation
* Objectives

1. Related Work

* Literature Survey
* Research Gap
* Problem Statement

1. Design Approach
   1. System Architecture
   2. Class Diagram
   3. Use case Diagram
2. Methodology
3. Implementation
4. Results and Discussion
5. Conclusion
6. References

Objectives

The objective of this project is to develop a secure and user-friendly password manager that secures user credentials using encryption and multi-factor authentication. The application aims to provide users with the ability to safely store, retrieve, and manage their passwords and user IDs, while also offering robust recovery mechanisms in case of forgotten passwords or lost access. The system will include features for securely adding, updating, and deleting credentials, along with mechanisms to provide limited access through alternate login method that is used for emergencies. The project will prioritize user experience and data security, ensuring that users have reliable access to their credentials while minimizing the risk of unauthorized access.

Literature Survey

1. **Systematization of Password Manager Use Cases and Design Paradigms**

*Simmons, J., Diallo, O., Oesch, S., & Ruoti, S. (2021, December). Systematization of password manageruse cases and design paradigms. In Proceedings of the 37th Annual Computer Security Applications Conference (pp. 528-540).*

The paper aims to address the usability of password managers by identifying and categorizing their use cases and design paradigms. Through reviewing password management documentation and literature, the authors systematized seventeen use cases and seventy-seven design paradigms. They conducted cognitive walkthroughs on eight popular desktop managers to evaluate their usability across these use cases. The study revealed significant usability issues such as difficulties in entering credentials on secondary devices, fatiguing setup processes, and challenges with interface designs and credential linking. Key observations include the limited functionality and security of browser-based managers compared to extension-based managers. The study highlights the need for further research to improve password manager designs and usability. A major gap identified is the lack of comparative studies on design paradigms, preventing a comprehensive understanding of their strengths and weaknesses. The authors suggest a transition strategy from browser-based to extension-based managers to enhance security and functionality for users.

1. **“It Basically Started Using Me:” An Observational Study of Password Manager Usage**

*Oesch, S., Ruoti, S., Simmons, J., & Gautam, A. (2022, April). “It Basically Started Using Me:” An Observational Study of Password Manager Usage. In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (pp. 1-23).*

This paper investigates the real-world usage and rationale behind password manager use through observational interviews with 32 users, employing grounded theory for data analysis. The research reveals that many users concurrently utilize both browser-based and third-party password managers, primarily as a backup strategy, and often without initial intention. Users avoid generated passwords due to difficulties in cross-device entry and memory challenges. Additionally, while credential audit tools are rarely used due to overwhelming and confusing outputs, users appreciate simpler and more direct warnings from Chrome’s built-in manager. Mobile password manager usage is minimal due to inconsistent autofill functionality and syncing issues, with users preferring single sign-on (SSO) solutions. Adoption is driven by work requirements, ease of credential entry, and improved password quality, yet promotion of password manager usage is limited outside immediate family. The study highlights the need for improved usability in password generation, credential audits, and mobile managers. Limitations include a U.S.-centric, tech-savvy sample from MTurk, potentially introducing biases and not capturing the experiences of non-users or those who abandoned managers. Future research should explore diverse populations and employ quantitative methods to validate and expand these findings.

1. **That Was Then, This Is Now: A Security Evaluation of Password Generation, Storage, and Autofill in Browser-Based Password Managers**

*Oesch, S., & Ruoti, S. (2020, August). That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers. In Proceedings of the 29th USENIX Conference on Security Symposium (pp. 2165-2182).*

This study evaluates the current security of 13 popular browser-based password managers, examining all three stages of the password manager lifecycle: password generation, storage, and autofill. The purpose is to determine if these tools have addressed known vulnerabilities from past research conducted five years ago. The researchers generated 147 million passwords to analyze their strength and randomness using various statistical tests, examined local password storage for encryption practices, and tested autofill functionalities against known web attacks. They found improvements in security features such as better encryption and safer autofill processes, but also identified persistent issues like unencrypted metadata, unsafe defaults, and susceptibility to clickjacking attacks. Notably, some generated passwords were still vulnerable to online and offline attacks, and certain managers auto-filled passwords without user interaction, posing security risks. The study highlights the need for users to carefully select and configure password managers, recommends enhancements like filtering weak passwords and improving master password policies, and suggests areas for future research such as mobile password managers and browser-supported password generation. Despite advancements, the study emphasizes that significant gaps remain, requiring ongoing evaluation and improvement to ensure password manager security and usability.

1. **Why people (don’t) use password managers effectively**

*Pearman, S., Zhang, S. A., Bauer, L., Christin, N., & Cranor, L. F. (2019). Why people (don't) use password managers effectively. In Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019) (pp. 319-338).*

The study investigates the adoption and effective use of password managers among users of built-in (browser-based or OS-based) and separately installed password managers through 30 semi-structured interviews. It aims to understand user mindsets, motivations, and barriers related to password management practices. Findings reveal that users of built-in managers prioritize convenience, often resulting in weaker password practices like reuse, while users of separately installed managers prioritize security with stronger, unique passwords. The study identifies barriers such as security concerns, lack of awareness, and usability issues. However, it acknowledges gaps in addressing specific user frustrations and suggests the need for more tailored designs and improved usability testing to enhance adoption among diverse user groups, particularly those less inclined towards security-centric behaviours. Future research should focus on bridging these gaps to effectively promote secure password management practices universally.

1. **Why Older Adults (Don’t) Use Password Managers**

*Ray, H., Wolf, F., Kuber, R., & Aviv, A. J. (2021). Why older adults (Don't) use password managers. In 30th USENIX Security Symposium (USENIX Security 21) (pp. 73-90).*

This study explores the adoption of password managers (PMs) among older adults (>60 years), and compares motivations and barriers with younger cohorts. Through semi-structured interviews with n = 26 participants, it identifies key differences: older adults exhibit higher mistrust of cloud storage and cross-device synchronization but show favourability towards PMs when recommended by family. Findings highlight barriers like perceived complexity and lack of urgency, suggesting advocacy and education as strategies for promoting PM adoption. However, the study acknowledges limitations in sample size and generalizability, particularly in technological literacy among older adults. This points to a need for further research to comprehensively explore these factors across diverse older adult populations and to address potential gaps in understanding how best to encourage PM adoption among this demographic.

1. **Challenges and Opportunities in Password Management: A Review of Current Solutions**

*Fernando, W. P. K., Dissanayake, D. A. N. P., Dushmantha, S. G. V. D., Liyanage, D. L. C. P., & Karunatilake, C. (2023). Challenges and Opportunities in Password Management: A Review of Current Solutions.*

This review paper explores the enduring challenges of password management in computer systems, emphasizing the prevalence of passwords despite numerous security vulnerabilities and malpractices. It evaluates various solutions including traditional, biometric, and PIN-based authentication, concluding that none have supplanted passwords due to their simplicity and universal applicability. The paper extensively discusses password managers (PMs) as a promising solution, categorizing them into software and hardware-based types, each with distinct advantages and limitations. Software PMs offer convenience but face security risks, while hardware PMs enhance security but suffer from usability concerns and potential data loss risks. The review identifies a research gap in the need for fully automating the password management process to address these issues comprehensively, emphasizing the importance of usability, security enhancements, backup mechanisms, and resistance to attacks as crucial areas for future research and development efforts.

1. **Android Password Managers and Vault Applications: Data Storage Security Issues Identification**

*Petrov, M. (2022). Android password managers and vault applications: data storage security issues identification. Journal of Information Security and Applications, 67, 103152.*

In their study titled "Android Password Managers and Vault Applications: Data Storage Security Issues Identification," the authors conduct a comprehensive analysis aimed at evaluating the security of Bitwarden and Keeper, two highly rated Android password manager/vault applications. Their primary objective is to assess the efficacy of these applications in securely storing user-entered data over the long term. The methodology involves advanced reverse engineering techniques, including runtime analysis and debugging, to uncover implementation details beyond what is publicly documented. By scrutinizing artifacts such as application data in persistent storage, crash dumps, and installer files, they aim to identify potential vulnerabilities and security gaps. Through their investigation, the authors successfully uncover several vulnerabilities and demonstrate proof-of-concept attacks that exploit these weaknesses. However, the study is limited in scope to the core protections implemented by the applications themselves, excluding external libraries and network-related security aspects. Despite revealing significant security risks and design flaws, the study provides valuable insights for improving data storage security in password managers and vault applications, suggesting avenues for future research on antipatterns and alternative security approaches.

1. **Enhanced Password Manager using Hybrid Approach**

*Pandare, P., Uniyal, S., Vani, R., Mali, S., & Rumao, P. (2023, April). Enhanced Password Manager using Hybrid Approach. In 2023 International Conference on Inventive Computation Technologies (ICICT) (pp. 1793-1798). IEEE.*

This paper introduces an enhanced password manager extension for web browsers, aiming to provide secure and user-friendly password storage. Using a hybrid approach of SHA-256 and AES encryption, the system ensures robust data security. The methodology involves leveraging Trongate as a development platform with PHP, HTML, CSS, and JavaScript for frontend and backend implementation. Data is stored securely in a phpMyAdmin database hosted on a localhost server, optimizing speed and cost-effectiveness. The system encrypts passwords upon saving, ensuring privacy and protection against unauthorized access. However, while the approach integrates strong cryptographic methods and emphasizes user privacy, potential drawbacks include scalability concerns with local hosting and the dependency on a single database system, which may limit deployment flexibility in larger-scale environments or cloud-based deployments. Future work could focus on enhancing scalability and exploring cloud-based storage options to address these limitations and broaden applicability across different deployment scenarios.

1. **Password Manager with Multi-Factor Authentication**

*Dhanalakshmi, R., Vijayaraghavan, N., Narasimhan, S., & Basha, S. (2023, April). Password Manager with Multi-Factor Authentication. In 2023 International Conference on Networking and Communications (ICNWC) (pp. 1-5). IEEE.*

This paper introduces a password manager enhanced with multi-factor authentication (MFA) to bolster security against data breaches and unauthorized access. The primary goal is to securely store and encrypt passwords using AES-256 encryption and PBKDF2 hashing for the master password. The system offers users the choice between local storage (offline) and cloud-based storage for passwords, each providing distinct advantages in terms of accessibility and security. MFA is implemented using biometric factors like fingerprints and graphical passwords, enhancing authentication security. The architecture involves modules for master password management, vault key generation, MFA, and interaction with a web server or cloud database. Implementation includes a user-friendly interface for managing various credentials securely. However, potential drawbacks include the risk of security bugs due to the complexity of MFA integration and the possibility of compromising the master password hash in cloud-based scenarios. Future work could focus on rigorous security testing, additional hashing techniques, and improving user education on MFA usage to mitigate these risks effectively and enhance overall system resilience.

1. **MonoPass: A Password Manager without Master Password Authentication**

*Jeong, H., & Jung, H. (2021, April). MonoPass: a password manager without master password authentication. In 26th international conference on intelligent user interfaces-companion (pp. 52-54).*

MonoPass is a novel password manager designed to eliminate the use of a master password for authentication, aiming to enhance security against password breaches while maintaining usability. The system employs a password generator that derives unique passwords from a master password using PBKDF2 hashing and HMAC-SHA256/512 encryption, ensuring consistency across devices without storing passwords centrally. Password metadata, including username and policy requirements, is hashed to generate passwords locally on each device, avoiding transmission over the network. The central server facilitates synchronization of metadata between devices via user-generated identification codes. While MonoPass addresses security concerns by decentralizing password storage and eliminating the risk of exposing all passwords through a master password breach, it relies on a central server for metadata synchronization, limiting functionality without an internet connection. Additionally, usability aspects such as manual metadata entry and potential inconvenience in real-world scenarios require further evaluation. Future work includes exploring alternative synchronization methods and conducting comprehensive security and usability assessments to validate MonoPass's effectiveness in real-world usage scenarios.

**Methodology**

The methodology of developing the password manager begins with a thorough planning phase where the project goals, requirements, and constraints were clearly defined. The primary objective was to create a secure and user-friendly application that can manage passwords and critical credentials, incorporating multiple layers of authentication and robust recovery mechanisms.

**I. Initial Setup**

The first-time setup process ensures that the application is correctly configured and ready for use. This process is crucial for establishing a secure environment from the outset.

**1. Master Password Setup:**

The master password setup process is a critical component of the password manager. It involves ensuring that users set a strong, secure master password, which is then hashed and stored securely. This password acts as the primary key to access all other stored credentials.

1. **Prompting the User for a Master Password:**
   * During the initial setup, users are prompted to enter a master password. This is done using a secure input method to ensure the password is not displayed on the screen.
   * The user is also asked to confirm the password to avoid any typographical errors. The entered password and the confirmation password are compared. If they match, the process proceeds to the next step. If not, the user is prompted to re-enter the passwords until they match.
2. **Password Strength Check:**
   * The password is checked against predefined strength criteria to ensure it is secure. These criteria include a minimum length of 12 characters, and inclusion of at least one lowercase letter, at least one uppercase letter, at least one digit, at least one special character.
   * If the password meets all the requirements, it is considered strong. Otherwise, the user is prompted to enter a new password that meets the strength criteria.
3. **Hashing and Storing the Master Password:**
   * Once a strong password is confirmed, it is hashed using the bcrypt hashing algorithm.
     + **bcrypt Hashing**

bcrypt is a password hashing function designed to be computationally intensive and slow. This deliberate slowness helps protect against brute-force attacks, by making it computationally expensive and impractical to hash each password, even with the prevalence of powerful computing resources. It is specifically designed and is widely considered a reliable choice for securely hashing passwords. When a password is hashed using bcrypt, a random salt is generated and combined with the password before hashing. Salting prevents the same password from being hashed to the same value across different users or even multiple uses by the same user, thus preventing attackers from using precomputed hash tables (rainbow tables) effectively. bcrypt uses the Blowfish cipher internally. It applies a key derivation function (KDF) that includes multiple iterations (called rounds) of hashing to slow down the hashing process. The number of rounds is a parameter that can be adjusted to increase the computational workload required to hash passwords. The result of bcrypt hashing is a hashed password string that includes both the salt used and the final hashed password. This string can be safely stored in a database or file. Even if the password file is compromised, the actual passwords remain secure due to the computational effort required to reverse-engineer the hashes.

* + The hashed password is stored securely in a file, ensuring that the plaintext password is never stored, thus reducing the risk of compromise and unauthorized access.

**2. Recovery Code Generation:**

The recovery code generation process is a crucial step to ensure that users can regain access to their accounts if they forget their master password. This process involves generating a secure, random recovery code, hashing it, and storing it securely.

1. **Random Recovery Code Generation:**
   * A random recovery code is generated using a combination of uppercase letters, lowercase letters, and digits. This ensures a strong and unpredictable code that is difficult to guess.
   * After generation, the user will be provided with the original recovery code and instructed to save it securely. This notification ensures that users are aware of the importance of the recovery code and are encouraged to store it in a safe place.
2. **Hashing and securely Storing the Recovery Code:**
   * Once the recovery code is generated, it is hashed using secure bcyrpt hashing algorithm to ensure its security.
   * Hashing the recovery code ensures that even if someone gains access to the stored hash, they cannot easily reverse-engineer the original code.
   * The hashed recovery code is then stored in a secure file.

**3. Key Generation:**

- Cryptographic keys are generated to encrypt and decrypt sensitive data within the application. These keys are stored securely and are essential for maintaining the integrity and confidentiality of the sensitive information stored in the database.

**Key Generation Procedure:**

1. **Generate Symmetric Keys:**
   * Symmetric keys are generated for encrypting different types of data within the application. Symmetric encryption involves using the same key for both encryption and decryption, making the secure handling of these keys vital.
   * Separate keys are generated for different types of data. This compartmentalization enhances overall security by ensuring that a compromise of one key does not affect other data types. Keys are generated for the user’s primary and alternative phone numbers, and for the usernames, passwords and service names that will be stored in the application database.
   * The Fernet symmetric encryption is used to generate most of the encryption keys. Fernet uses AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode with a 128-bit key. AES is a widely accepted and secure encryption standard, ensuring strong data protection. Fernet also uses HMAC (Hash-based Message Authentication Code) to ensure data integrity. This means that any tampering with the encrypted data can be detected. Keys are Base64 encoded, making them safe for storage and transport as ASCII strings.
   * FPE (Format-Preserving Encryption) is used for service names to maintain their format while still encrypting the data. This allows encrypted service names to be used in database queries and lookups without revealing their actual content, thus ensuring both security and functionality.
2. **Secure Storage of Keys:**
   * Each generated key is securely stored in a separate file ensuring that the keys are kept secure and are only accessible by authorized parts of the application.

**4. Phone Number Registration:**

The phone number registration process is an essential part of ensuring secure access to the password manager. The procedure involves collecting, verifying, and securely storing both a primary and an alternative phone number for the user. These phone numbers are critical for OTP (One-Time Password) verification during login and account recovery processes, providing an additional layer of security.

**Key Steps in the Phone Number Registration Process:**

1. **Phone Number Collection:**
   * Users are prompted to enter their primary phone number first.
   * After the primary number is successfully registered and verified, users are asked to enter an alternative phone number.
   * The registration of both a primary and an alternative phone number ensures that there is always a backup option.
   * If the primary phone number is unavailable for any reason, the alternative number can be used to receive OTPs, ensuring continuous access to the account and reducing the risk of being locked out.
2. **OTP Verification:**
   * Upon entering a phone number, an OTP is sent to the provided number using an SMS service (such as Twilio).
   * The user must enter the OTP received on their phone within a specified timeframe to verify their phone number.
   * If the OTP is entered correctly within the allowed time, the phone number is considered verified.
   * This ensures that the phone number provided by the user is valid and accessible, adding an extra layer of security.
3. **Secure Storage:**
   * Once verified, the phone number is encrypted using Fernet symmetric encryption. Encrypting phone numbers with Fernet ensures that any tampering with the stored data can be detected, maintaining data integrity.
   * Separate encryption keys are used for the primary and alternative phone numbers, ensuring that each number is encrypted independently for added security.
   * The encrypted phone numbers are then stored securely in files.

**II. Authentication Mechanisms**

2.1 Master Password Verification

The master password verification ensures that only authorized users can access their stored credentials.

1. **Password Input and Verification:**

The user is prompted to input their master password. The provided password is hashed and compared against the stored hash. If the hashed password matches the stored hash, the user is authenticated and allowed to proceed.

1. **Failure Handling and Lockout:**

The system keeps track of the number of failed password attempts. If the number of failed attempts exceeds a predefined limit, the account is locked for a specified duration. This mechanism helps deter brute force attacks.

2.2 OTP Verification

OTP (One-Time Password) verification adds an extra layer of security.

1. **OTP Generation and Sending:**

An OTP is generated and sent to the user’s registered phone number. The user must enter the OTP correctly within a limited number of attempts and within a specified timeframe.

1. **Failure Handling and Lockout:**

Similar to password verification, failed OTP attempts are tracked. Upon exceeding the allowed attempts, the account is locked for a set period of time to prevent continuous guessing.

2.3 Recovery Code

In case users forget their master password, they can recover their account using the recovery code. Users input their recovery code, which is hashed and compared with the stored hash. If it matches, an OTP is sent to the primary phone number for further verification. The system limits the number of recovery attempts to prevent unauthorized access, and exceeding the allowed recovery attempts triggers a lockout similar to the normal login procedure. The number of attempts and lockout duration may differ from normal login.

2.4 Alternative Login

An alternative login method is provided for situations where the primary phone number is inaccessible. The user inputs their master password, which is verified as described above. After verification, an OTP is sent to the alternative phone number instead of the primary number. The user must enter the correct OTP to gain access. Failed attempts for alternative login are tracked separately from ordinary login. Exceeding the allowed number of failed attempts triggers a lockout for a specified period. The number of attempts and lockout duration may differ from normal login.

Since the secondary phone number is being used for user verification, some additional layers of security are included in case an unauthorized entity has managed to gain access to the master password and alternative phone.

* After verification of master password when OTP is sent to the alternative phone, another OTP is simultaneously sent to the primary phone number. This will notify the authorized user in case they are not the ones attempting to access their Password Manager app, and enable them to inform the admin/customer service to lock the app.
* Most of the features of the app will be disabled in the alternative login mode. Only a few credentials, which were marked as most critical, will be viewable, while others will be inaccessible. Similarly, edit access is also denied.

**III. Credential Management**

The core functionality of the application is credential management, and it is designed to ensure secure handling of user credentials through various operations such as adding, retrieving, updating, deleting, and searching for credentials. Each operation involves a sequence of steps that ensure security and data integrity.

3.1 Adding Credentials

Users are prompted to enter details such as service name, username, password, and criticality of the service. Validation checks ensure the service name adheres to specified constraints (maximum length and valid characters). Users can mark a service as critical, influencing access restrictions during alternative login scenarios. Users are asked to confirm the details before they are saved. Once confirmed, the credentials are encrypted using distinct cryptographic keys for different fields. The encrypted credentials are then stored in the database.

3.2 Retrieving Credentials

The user can request the credentials for a specific service. The stored encrypted credentials are decrypted before being displayed to the user in a secure manner. Additionally, if the user has logged in via alternative method, only the critical flagged credentials can be retrieved, while access to all other credentials will be denied.

3.3 Updating Credentials

Users can choose to update specific fields (e.g., service name, username, password, critical status) or all fields of a service credential. Validation and confirmation steps follow to ensure accuracy and intentional updates. The relevant credentials are encrypted again before updating the database.

3.4 Deleting Credentials

Users can delete credentials for a specific service. Upon confirmation, the credentials are securely removed from the database.

3.5 Searching for Credentials

Users can search for services using a search term. The application fetches all services and decrypts them to find matches. Matching services and corresponding usernames are displayed to the user.

**IV. User Interface and User Experience**

**4.1 Password Generation**

The password generator in this application is designed to create strong and secure passwords based on user-specified criteria. The logic behind the password generator involves:

* **User Customization Input**: The user specifies the desired length of the password and whether special characters should be included.
* **Character Selection**: The application constructs a pool of characters from which the password can be generated. This pool always includes:
  + Uppercase and lowercase letters (string.ascii\_letters)
  + Digits (string.digits)
  + If the user opts to include special characters, these are added to the pool (e.g., [@#$%Z^\_\*-]).
* **Random Selection**: The password is created by randomly selecting characters from the specified pool until the password reaches the desired length. This ensures that the password is both random and strong, reducing the likelihood of it being easily guessed or cracked.

This approach ensures that passwords are both secure and customizable to the user's needs.

**4.2 Command-Line Interface**

The application employs a Command-Line Interface (CLI) designed for simplicity and usability. The logic of the CLI revolves around the following key principles:

* **Clear Instructions**:
  + **Step-by-Step Guidance**: The CLI provides clear and concise instructions at each step, ensuring that the user understands what input is required. For example, during login, the user is prompted to enter the master password and then an OTP, with explicit prompts and feedback at each stage.
  + **Error Handling**: When the user makes an error, such as entering an incorrect password or OTP, the system provides informative messages and the number of remaining attempts.
* **Enhanced Usability**:
  + **Structured Menus**: The CLI is organized into menus and sub-menus that guide users through various functionalities. For example, after logging in, the user is presented with options to add, retrieve, update, or delete credentials, among other actions.
  + **Feedback and Confirmation**: The CLI frequently asks for user confirmation before performing critical actions, such as changing a password or deleting credentials. This helps prevent accidental changes.
  + **Lockout Mechanism**: To enhance security, the CLI includes lockout mechanisms that temporarily block access after too many failed attempts. The user is informed of the lockout and the duration they need to wait before retrying.
  + **Instructions Command**: The CLI includes an option to view detailed instructions on how to use the application, ensuring that users can always refer to guidance if needed.