Security Project

As working on the project by my self I decided to create a command line interface application for this Master Password project. It is programmed using python and uses mariadb for it's database.

Outline of features

- Command Line Interface
- Show/save entries after verifying Master password
- Copy password to clipboard
- Generate random password (alphabets A – Z, numbers and special chars)

Security steps and it's explanation

Master password is first inputted while in configuration, and the hash of it is saved in a file.

When the master password is first inputted during configuration, it is not stored in plain text, but hashed. This adds a layer of security, as hashing is a one-way process, meaning it's computationally infeasible to reverse the hash and retrieve the original password. In the event of a data breach, attackers would only have access to the hash.

Device secret is randomly generated and stored in a file.

This randomness adds unpredictability to the secret, making it difficult for an attacker to guess or predict. By storing this secret securely, it is kept separate from the master password, providing an additional layer of protection. Even if an attacker gains access to one part of the system, they would still need the device secret to derive the master key. It will be combined with the master password to create a master key.

Master password and Device Secret is passed into a hashing function (pbkdf) to create a valid key for AES-256. This is called a MASTER KEY.

PBKDFs are designed to slow down brute-force and dictionary attacks. They repeatedly hash the input data, which makes it computationally expensive for attackers to guess the correct input (i.e., the master password and device secret). A unique salt can be added to a master password and device secret before hashing. This furthermore helps us against dictionary and rainbow attacks.

AES-256 is a strong encryption algorithm, and the master key is used as the basis for encryption and one of commonly used ways to create a strongly encrypted key. I mainly went my recommendation and usage, as my abilities to truly understand which method would be more secure is limited.

The Master key is used to encrypt/decrypt new entries.

Without the master key, an attacker cannot access the stored passwords, even if they have access to the hashed master password and device secret. This is the main point of a password manager.

Different fields that we can input:

Here we encrypt fields that are necessary for signing into specific places and encrypt them for and extra layer of protection while we don't repeat this process for fields that are used to identify our passwords.

Encrypted fields – Email, username and password.

Normal fields – site name and url.

Process of adding a new entry

- Ask for Master Password
- Validate master password by hashing and checking with existing hash
- Make hash (Device secret and master password) into a master key
- Input data
- Encrypt email, username and password with master key and save the fields into the database.

Process of getting and entry

- Input the field to search for. Like site name, site url, email and username.
- Display all the entries that match with search.
- Have password hidden as when displaying a list of sites in command line.
- If user asks to get password with -p flag then ask for master password.
- Validate master password by hashing and checking with existing hash.
- Make hash (Device secret and master password) into a master key.
- Decrypt the password for the specific site and copy it to clipboard.

A detailed guideline of how to run and use the program are in the READ.md file, which describe in much more detail how to use the application.