Report on Secure Password Checker Code

Overview

The Secure Password Checker is a Python-based program designed to help users determine if their passwords have been compromised in known data breaches. The script uses the Have I Been Pwned API to compare user-entered passwords against a database of leaked password hashes. This report explains the key functionalities, logic, benefits, potential drawbacks, and improvements for the code.

Functionality

1. Hashing Passwords

The program leverages the hashlib library to hash the user's password using the SHA-1 algorithm. This ensures the password is never sent in plain text, thereby enhancing privacy. The password is hashed into a hexadecimal representation, converted to uppercase, and split into two parts:

Prefix: The first 5 characters of the hash.

Suffix: The remaining characters.

This separation enables querying the API without exposing the full hash, maintaining additional layers of security.

2. Checking Password Breaches

The program interacts with the Have I Been Pwned API through HTTP requests:

The API is queried using the prefix of the SHA-1 hash.

The response contains a list of hash suffixes and the corresponding breach counts.

The program scans the API response to check if the suffix of the user's hashed password exists. If found, it indicates the password has been compromised and displays the breach count.

3. User Interaction

The program has a loop where the user is prompted to enter a password:

If the password is found in breaches, the user is warned and asked to choose another password.

If the password is not found in breaches, the user is informed it is safe, and the process terminates.

The program's main function ensures a seamless user experience by handling initialization and guiding users through the checking process.

Benefits

1. Security

The program uses SHA-1 hashing and only sends the prefix of the hash to the API, protecting sensitive user data. The API's design ensures that partial data is shared, minimizing the risk of exposing passwords.

2. Ease of Use

The program offers a user-friendly interface:

Simple prompts guide the user through entering and verifying their password.

Clear feedback is provided about whether the password is safe or breached.

3. Reliability

The program relies on the Have I Been Pwned service, a widely recognized and trusted database for tracking data breaches. This enhances the reliability and accuracy of results.

Limitations

1. Use of SHA-1

Although the use of SHA-1 is common for compatibility with the Have I Been Pwned API, it is not cryptographically secure by modern standards. It is vulnerable to collision attacks and should not be used for critical systems requiring strong hashing.

2. Limited Scope

The program only checks if a password exists in known breaches. It does not evaluate:

Password strength (e.g., length, complexity).

Real-time password security features like brute force resistance or entropy.

3. Reliance on External API

The program's functionality depends on the availability of the Have I Been Pwned API. If the API is unavailable or if rate limits are exceeded, the program cannot function as intended.

Potential Improvements

1. Password Strength Evaluation Incorporating a password strength checker alongside the breach check would enhance the program. Tools like zxcvbn could assess password complexity and offer suggestions for stronger passwords.

2. Alternative Hashing Though SHA-1 is suitable for this API, adopting more secure hashing algorithms, such as SHA-256 or Argon2, for internal handling could align the program with modern security standards.

3. Enhanced User Interface

Adding a graphical interface (GUI) could make the program more accessible for non-technical users.

Providing tips on creating secure passwords would be beneficial.

4. Offline Capabilities Downloading a subset of the Have I Been Pwned database to allow offline checking (e.g., for frequently queried hashes) could improve efficiency and reduce reliance on external services.

5. Error Handling Improving error handling for API failures, network issues, or invalid inputs would make the program more robust.

Conclusion

The Secure Password Checker is a practical tool for identifying potentially compromised passwords. By leveraging the Have I Been Pwned API, it provides users with actionable insights into the safety of their credentials. While the program is effective and secure, implementing enhancements such as password strength evaluation, improved hashing methods, and offline capabilities could further increase its utility and appeal.