Python Projekt Passwortmanager

an der Dualen Hochschule Baden-Württemberg Ravensburg Campus Friedrichshafen

von

Simon Blum, TIT

Maximilian Rodler, TIT

Ruth Fröhlich, TIK

01.09.2024
Bearbeitungszeitraum 10 Wochen

Inhaltsverzeichnis

1 Einleitung

Der Passwortmanager ist ein Python-basiertes Programm, welches Benutzern die sichere Verwaltung ihrer Passwörter zu ermöglichen. Diese Dokumentation beschreibt den Aufbau und die Funktionalität des Passwort Managers, der in Python entwickelt wurde. Sie umfasst eine detaillierte Beschreibung der Funktionen, eine Architekturbeschreibung, das Speicherformat, das Nutzerinterface, den Programmablauf sowie die Ergebnisse der durchgeführten statischen und dynamischen Analysen.

2 Funktionsbeschreibung

Der Passwortmanager bietet eine Reihe von Funktionen, die es dem Benutzer ermöglichen, Passwörter sicher zu speichern, zu verwalten und bei Bedarf abzurufen. Zu den Hauptfunktionen gehören:

2.1 Erstellen von Benutzern/ Registrieren

Die Registrierungs-Funktion ermöglicht es neuen Benutzern, sich im Passwortmanager ein Konto zu erstellen und ihre Anmeldeinformationen sicher zu speichern. Dabei wird der Nutzt er aufgefordert, Benutzername sowie Masterpasswort anzugeben. Um Tippfehler zu vermeiden, muss das Masterpasswort zweimal bestätigt werden.

Die Eingabe erfolgt über das Text User Interface (TUI), um sicherzustellen, dass die Eingabe der Passwörter nicht auf dem Bildschirm angezeigt wird, wird eine Input-Validation von den Textboxen in 'Curses' verwendet.

Um die Eingaben zu validieren, wird zunächst überprüft, ob der eingegebene Benutzername den vorgegebenen Anforderungen entspricht (min. 4, max. 20 Zeichen). Das Masterpasswort wird anschließend auf Sicherheit überprüft, indem es gegen bestimmte Kriterien getestet wird, wie z. B. Mindestlänge, Verwendung von Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen. Eine zu einfache Masterpasswortwahl wird abgelehnt. Dazu dient die validate_password_saferty Methode, welche anhand dieser Kriterien safetypoints vergibt. Wenn der Wert drei erreicht ist, wird das Masterpasswort als sicher genug eingestuft. Schließlich müssen das eingegebene Masterpasswort und die Bestätigung übereinstimmen, um Tippfehler auszuschließen.

Daraufhin wird überprüft, ob der eingegebene Username bereits in der Datenbank vorhanden ist, und gegebenenfalls wird eine Warnung ausgegeben, dass der Name bereits vergeben ist.

Bevor der Benutzername und das Masterpasswort in der Datenbank gespeichert werden, werden beide Daten durch den Hash-Algorithmus SHA-256 gehashed. Diese verschlüsselten Zugangsdaten gewährleisten, dass nur autorisierte Benutzer auf ihre gespeicherten Passwörter zugreifen können.

2.2 Login

Die Login-Funktion ermöglicht Benutzern den Zugriff auf das Passwort-Management-System. Auch hier wird der Benutzer aufgefordert, seinen Benutzernamen und sein Masterpasswort über das Text-User-Interface (TUI) einzugeben.

Im nächsten Schritt wird der Benutzername in der Datenbank gesucht, um festzustellen, ob der Benutzer existiert. Wenn der Benutzername existiert, wird das eingegebene Masterpasswort mit dem in der Datenbank gespeicherten verschlüsselten Masterpasswort verglichen. Dabei wird das eingegebene Masterpasswort mit dem gleichen Algorithmus verschlüsselt, der bei der Registrierung verwendet wurde, und das Ergebnis mit dem gespeicherten Hash verglichen.

Wenn der Benutzername und das Masterpasswort übereinstimmen, wird der Benutzer erfolgreich angemeldet und auf das Hauptmenü weitergeleitet. Falls der Benutzername nicht existiert oder das Masterpasswort falsch ist, wird eine Fehlermeldung angezeigt und der Benutzer kann den Anmeldeprozess erneut versuchen.

2.3 Sichere Verschlüsselung

Die Verschlüsselungs-Funktion ist ein zentraler Bestandteil des Passwortmanagers und stellt sicher, dass alle Passwörter, die in der Datenbank gespeichert werden, sicher verschlüsselt sind. Dies wird durch den Einsatz des AES-256-Verschlüsselungsalgorithmus gewährleistet. AES-256 steht für „Advanced Encryption Standard“ mit einer Schlüssellänge von 256 Bit, was eine starke kryptografische Sicherheit bietet. Ein Initioalisierungs-Vektor (IV) sorgt dafür, dass zwei gleiche Passwörter bei jeder Verschlüsselung unterschiedliche Ergebnisse liefern, indem er eine zusätzliche Zufälligkeit hinzufügt. Dies verhindert, dass zwei identische Passwörter den gleichen verschlüsselten Wert erzeugen. Vor der Verschlüsselung wird mit Hilfe einer KDF (Key Dervation Function) aus dem eingegebenen Passwort ein Schlüssel abgeleitet. Als KDF wird SScrypt"verwendet. Scrypt leitet aus dem Masterpasswort einen Schlüssel ab, welcher dann zur Verschlüsselung mit AES verwendet werden kann
Die encrypt aes-Methode übernimmt die Verschlüsselung von Passwörtern. Sie nimmt das zu verschlüsselnde Passwort und den abgeleiteten Schlüssel aus der KDF als Eingabe und gibt die verschlüsselten Daten als Ausgabe zurück. Innerhalb dieser Methode wird der bereits angesprochene IV generiert, der zusammen mit dem Schlüssel verwendet wird, um die Verschlüsselung durchzuführen. Der IV sorgt dafür, dass selbst bei gleichem Klartext und Schlüssel unterschiedliche verschlüsselte Ausgaben erzeugt werden.
Mit der decrypt aes-Methode dagegen, werden die Passwörter wieder entschlüsselt. Sie nimmt das verschlüsselte Passwort und den Schlüssel als Eingabe und gibt das entschlüsselte (ursprüngliche) Passwort als Ausgabe zurück. Um das Passwort korrekt wieder herzustellen, muss derselbe Schlüssel, der auch für die Verschlüsselung verwendet wurde, genutzt werden.
Alle anderen Daten werden mit Fernet ver- und entschlüsselt.

2.4 Passwortverwaltung

Nach erfolgreicher Registrierung und Anmeldung können Benutzer ihre Passwörter innerhalb der Anwendung verwalten. Diese Verwaltung basiert auf den grundlegenden CRUD-Operationen (Create, Read, Update, Delete):

- **Erstellen eines neuen Passworts (Create):** Benutzer können neue Passwörter für verschiedene Dienste und Konten hinzufügen. Dabei wird ein neuer Eintrag erstellt, der den Benutzernamen, die zugehörige URL oder Anwendung und das verschlüsselte Passwort enthält. Vor dem Speichern wird das Passwort durch die bereits beschriebene Verschlüsselungs-Funktion gesichert. Der Eintrag wird dann zusammen mit dem verschlüsselten Passwort in der Datenbank abgelegt.

- **Abrufen eines gespeicherten Passworts (Read):** Wenn der User sich im Passwortmanager anmeldet, werden ihm die gespeicherten Passwörter anzeigen. Dazu werden die Passwörter aus der Datenbank abgerufen und mit der decrypt aes-Methode entschlüsselt.

- **Aktualisieren eines bestehenden Passworts (Update):** Benutzer haben die Möglichkeit, bereits gespeicherte Passwörter zu ändern. Dies ist nützlich, wenn ein Benutzer seine Passwörter regelmäßig ändern möchte, um die Sicherheit zu erhöhen. Das neue Passwort wird verschlüsselt und der entsprechende Datenbankeintrag wird mit den neuen verschlüsselten Daten aktualisiert.

- **Löschen eines Passworts (Delete):** Benutzer können Passwörter aus der Datenbank löschen. Der entsprechende Eintrag wird dann aus der Datenbank gelöscht.

Auch ein zentrales Element bei der Verwaltung der Nutzer ist das Masterpasswort, was als Hauptschlüssel zur Ver- und Entschlüsselung der gespeicherten Passwörter dient. Wenn ein Benutzer sein Masterpasswort ändert, erfordert dies eine Neuverschlüsselung aller gespeicherten Passwörter. Da das Masterpasswort der Schlüssel für die AES-256-Verschlüsselung ist, müssen alle Passwörter im Passwortmanager mit dem neuen Masterpasswort neu verschlüsselt werden. Dazu werden erst alle Passwörter mit dem alten Masterpasswort entschlüsselt. Nachdem die Passwörter im Klartext vorhanden sind wird das neue Masterpasswort verwendet, um einen neuen Schlüssel zu generieren, um die Passwörter neu zu verschlüsseln. Am Ende werden die verschlüsselten Passwörter wieder in der Datenbank gespeichert und ersetzten somit die alten Einträge.

3 Architektur

Der Passwort Manager ist in einer modularen Struktur aufgebaut, die eine klare Trennung der Verantwortlichkeiten zwischen den verschiedenen Komponenten ermöglicht. Dies erleichtert die Wartung, Erweiterung und das Testen des Systems. Die Ordnerstruktur des Projekts aus fünf Hauptmodulen, welche die aufgaben innerhalb des Systems getrennt halten:

```
src/
|-- controller/
|-- crypto/
|-- model/
|-- tui/
+-- main.py
```

controller: Die Controller sind für die Kommunkation zwichen Datenbank und Anwendung verantwortlich.

crypto: Das Modul stellt Funktionen zur Verschlüsselung, Entschlüsselung und zum Hashing von Passwörtern bereit:

- die AES-256-Verschlüsselung, zur sicheren Verschlüsselung von Passwörtern.
- das Hashing, es generiert kryptographische Hashes von Passwörtern, z.B. für das Masterpasswort.
- eine Schlüsselableitung, um aus einem Passwort sichere Verschlüsselungsschlüssel zu erzeugen.

model: Das Model definiert die Datenstrukturen, sie im System verwendet werden. Es enthält die Klassen, welche die Benutzter- und Passwortinformationen repräsentieren. Dazu gehören:

- Die Klasse User welche einen Benutzer repräsentiert. Sie enthält Informationen wie den Benutzernamen und das verschlüsselte Passwort.
- Die Klasse Password repräsentiert ein einzelnes Passwort, das vom Benutzer gespeichert wird. Sie wird von der Klasse PasswordInformation umschlossen, welche alle restlichen Daten enthält.

tui: Das Text User Interface implementiert das Interface für den Nutzer. Es ermöglicht die Interaktion mit dem System in einer Kommandozeilenumgebung. Zu den Hauptkomponenten gehören:

- Views: verschiedene Menüs für Aktionen wie Login, Registrierung und Passwortverwaltung
- Eingabebefehle: Ermöglichung der Eingaben wie Name und Passwort
- Anzeigen: zeigt Informationen an, wie z.B. Passwortlisten und Fehlermeldungen

4 Speicherformat

Die Daten werden in einer SQLite-Datenbank gespeichert, die als Datenbanklösung für den Passwort Manager dient. Sie besteht aus zwei Tabellen, die jeweils die Daten zu vorhandenen Nutzern und deren Passwörtern speichern.
Der Benutzername wird als Hash gespeichert und identifiziert den Benutzer eindeutig innerhalb des Systems. Diese Spalte ist einzigartig, sodass kein Benutzername doppelt

vorkommen kann. Das Masterpasswort wird als Hash gepeichert.
In der Passwörtertabelle wird eine Id für jedes Passwort abgespeichert. Diese Spalte ist der Primärschlüssel der Tabelle und wird automatisch inkrementiert. Als Fremdschlüssel, der auf die Benutzer-Tabelle verweist, dient der Hash des Benutzernamens. Dieser Fremdschlüssel stellt eine Beziehung zwischen den gespeicherten Passwörtern und dem zugehörigen Benutzer her, sodass jedes Passwort eindeutig einem Benutzer zugeordnet werden kann. Die restlichen Daten werden verschlüsselt in ihren eigenen Spalten gespeichert.

5 Nutzerinterface

Das Nutzerinterface des Passwort Managers basiert auf einer textbasierten Oberfläche, die durch das curses-Modul in Python realisiert wird. Diese Oberfläche ermöglicht eine benutzerfreundliche Interaktion mit dem Programm über einfache Tasteneingaben. Die TUI ist darauf ausgelegt, dem Benutzer eine klare und intuitive Navigation zu bieten, während sie gleichzeitig auf die verschiedenen Funktionen des Passwort Managers zugreift. Die Navigation durch die Benutzeroberfläche erfolgt über die Pfeiltasten auf der Tastatur. Diese Tasten ermöglichen dem Benutzer, zwischen verschiedenen Menüoptionen zu wechseln. Dazu gehören die Pfeiltaste, welche den Fokus auf die jeweilige Menüoption nach oben oder unten, links oder rechts bewegen, und die Enter Taste, welche die markierte Option auswählt und die zugehörige Aktion ausführt, wie z.B. das Einloggen oder das Hinzufügen eines neuen Passworts. Mit der Tab taste, kann zwischen den Views /TODO gewechselt werden.
Am unteren Rand des Bildschirms werden einfache Textanweisungen angezeigt, die dem Benutzer die verfügbaren Steuerungsmöglichkeiten erläutern. Die Menüs und Auswahlmöglichkeiten sind so gestaltet, dass der Benutzer leicht zwischen verschiedenen Funktionen wie Passworterstellung, -verwaltung und -anzeige wechseln kann

6 Programmablauf

Der Programmablauf des Passwortmanagers ist strukturiert und folgt einer klaren Sequenz von Schritten, um dem Benutzer eine intuitive und sichere Verwaltung von Passwörtern zu ermöglichen.
Beim Start der Anwendung wird zunächst eine Verbindung zur SQLite-Datenbank hergestellt. Dies geschieht über die Funktion `connect_to_db`, die sicherstellt, dass die Datenbankdatei existiert und geöffnet wird. Anschließend wird das Text-basierte User Interface (TUI) mithilfe der curses-Bibliothek initialisiert. Hierbei werden die Farbschemata festgelegt und das Hauptfenster (Window-Objekt) er- stellt, das als zentrale Schnittstelle für die Benutzerinteraktionen dient. Das erscheinende Startmenü bietet die folgenden Optionen:

Login:
Ermöglicht es dem Benutzer, sich mit seinen bestehenden Anmeldeinformationen anzumelden.

Registrierung:
Erlaubt neuen Benutzern, ein neues Konto zu erstellen. Je nach Auswahl wird der Benutzer zur entsprechenden Funktion weitergeleitet. Nach erfolgreicher Anmeldung findet sich der Nutzter im Hauptmenü wieder, dort werfden die Passwörter, die der User gespeichert hat angezeigt, mit den jeweiligen Zugehörigkeiten/Notizen. Der Benutzer hat nun Zugriff auf die folgenden Funktionen der Passwortverwaltung:

Passwort hinzufügen:
Der Benutzer kann neue Passwörter eintragen, indem er einen Namen für das Passwort (z.B. „E-Mail-Konto“) und das eigentliche Passwort angibt. Dabei wird das Passwort zusammen mit Metadaten (z. B. Notizen, Kategorien) gespeichert. Alternativ können Passwörter importiert werden.
Hierfür kann sich an den Dateien innerhalb des `import\_export` Ordners orientiert werden

Passwort ändern:
Der Benutzer kann ein bestehendes Passwort ändern. Das neue Passwort wird verschlüsselt und der Datenbankeintrag wird aktualisiert.

Passwort löschen:
Der Benutzer kann Passwörter aus der Datenbank entfernen. Der entsprechende Eintrag wird aus der Datenbank gelöscht.
Jede dieser Aktionen wird durch die TUI gesteuert, die Eingaben des Benutzers aufnimmt und entsprechende Datenbankoperationen durchführt.

7 Statische und dynamische Analysen

Für das Projekt wurde sowohl mit statischer als auch dynamischer Codeanalyse gearbeteitet.

7.1 Mypy

Für Type-Checking wurde **MyPy** Version 1.10.0 mit der vorgegebenen Konfiguration genutzt:

```
[mypy]
warn_return_any = True
warn_unused_configs = True
disallow_untyped_defs = True
disallow_untyped_calls = True
disallow_incomplete_defs = True
```

Für das Aufrufen der Analyse wird der Befehlt `mypy src main.py` verwendet. Hierbei treten keine Fehler in den 71 analysierten Dateien auf.

7.2 Pylint

Für die Statische Codeanalyse wurde **Pylint** in der Version 3.2.3 verwendet. Die Verwendete Konfiguration kann in der Datei `.pylintrc` gefunden werden.
Für den Aufruf der Analyse wird der Befehl `pylint src main.py` verwendet. Auch hier werden keine Warnungen ausgegeben und der Code wird mit 10/10 Punkten bewertet.

Bei Pylint gibt es zu beachten, dass in der Datei `src/import_export/import_data.py` die Warnung **E1101** deaktiviert wurde. Mehr Details hierzu sind in der Datei zu finden.

7.3 Coverage und Unittest

Die Module Coverage (Version 7.5.3) und Unittest wurde für die dynamische Codeanalyse verwendet. Um die Tests aufzurufen und daraufhin einen Report in der Konsole auszugeben können folgende Befehle genutzt werden:

```
coverage run -m unittest discover -v
coverage report -m --skip-empty
```

Aufgrund schwindender Zeit und der Komplexität der Benutzeroberfläche wurde jedoch eine Gesamtabdeckung von 11% erreicht. Der vollständige Bericht kann in der Datei `coverage_report.txt` gefunden werden.

8 Versionen der genutzten Bibliotheken

Wir nutzt eine Reihe von Bibliotheken, um einige Funktionalitäten des Passwortmanagers zu gewährleisten. Diese Abhängigkeiten werden in der Date `requirements.txt` definiert.

8.1 Standardbibliotheken (Python 3.11.5)

- `json`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.

- `random`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.

- `datetime`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: Zur Zeitmessungen und Verzögerungen verwendet. Sie stellt Funktionen bereit, um die aktuelle Zeit zu ermitteln, Wartezeiten zu implementieren, und einfache Zeitmessungen durchzuführen.

- `os`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.

- `sqlite3 (für das Database-Modul):`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: Diese Bibliothek wird zur Verwaltung und Speicherung der Benutzer- und Passwortdaten in einer SQLite-Datenbank verwendet. sqlite3 stellt die Verbindung zur Datenbank her und ermöglicht die Ausführung von SQL-Operationen zur Verwaltung der Daten.

- `sys`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.

- `asyncio`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.

- `pickle (für das Model-Modul):`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: Wird verwendet, um Python-Objekte (wie die Password-Instanz) zu serialisieren und zu deserialisieren. Dies ermöglicht das Speichern und Wiederherstellen komplexer Datenstrukturen.

- `base64`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.

- `hashlib (für das Crypto-Modul):`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: hashlib wird zur Implementierung von Hashing-Algorithmen wie SHA-256 verwendet.

- `secrets`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: Zum kryptografisch sichere Zufallszahlen generieren, für die Erstellung von Passwörtern

- `string`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.

- `enum`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: Wird zur Definition von Aufzählungstypen (Enums) verwendet, die symbolische Namen für eine Gruppe von Werten bereitstellen.

- `unittest`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: Ein Framework zum Schreiben und Ausführen von Tests in Python. Es ermöglicht das Erstellen von Testfällen, das Gruppieren von Tests und das automatische Ausführen und Überprüfen von Testergebnissen. Im

Passwortmanager-Projekt wird unittest verwendet, um sicherzustellen, dass die einzelnen Komponenten des Programms korrekt funktionieren und die erwarteten Ergebnisse liefern.

- `typing`
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: Um Typannotationen im Code hinzuzufügen. Dies verbessert die Lesbarkeit und Wartbarkeit des Codes und ermöglicht es Entwicklern, die erwarteten Datentypen für Variablen, Rückgabewerte und Parameter von Funktionen klar zu definieren. Im Projekt wird typing verwendet, um Typensicherheit zu gewährleisten und die Entwicklung zu unterstützen.

- **curses (für das TUI-Modul):**
  Version: Teil der Python Standardbibliothek, Version 3.11.5.
  Verwendung: Diese Bibliothek ermöglicht es, im Terminal textbasierende interaktive Oberflächen zu erstellen, die auf Benutzerinteraktionen reagiert. Ermöglicht das Rendering von Menüs, Eingabefeldern und andere GUI-ähnliche Elemente im Terminal zu implementieren.

8.2 Externe Bibliotheken

- `requests`
  Version: 2.32.3
  Verwendung: Senden von HTTP-Anfrangen an die "haveibeenpwnedÄPI.

- `cryptography`:
  Version: 3.11.5
  Verwendung: Bietet umfassende Bibliothek für kryptografische Operationen, darunter AES-256-Verschlüsselung, die zur Sicherung der gespeicherten Passwörter verwendet wird.

- `python-dotenv`
  Version: 1.0.1
  Verwendung: Umwandeln einer `.env` Datei in Umgebungsvariablen. Wird im aktuellen Version nur zur Definition des Datenbankpfades genutzt.