Bezpieczeństwo systemów informatycznych
Projekt menadżera haseł

Grzegorz Nieużyła
Dawid Karolewski

1 Wprowadzenie

Mnogość różnych aplikacji internetowych używanych przez przeciętnego użytkownika Internetu wymaga korzystania z różnych i silnych haseł żeby zapewnić bezpieczeństwo danych przed nieuniknionymi atakami i wyciekami danych. Jako, że pamięć ludzka jest zawodna, pomocne jest użycie zewnętrznej aplikacji oferującej generowanie i bezpieczne przechowywanie skomplikowanych haseł. Ten projekt ma na celu stworzenie oprogramowania które spełnia te warunki.

2 Zakres projektu

Celem tego projektu jest stworzenie aplikacji zawierającej dwa moduły – aplikację okienkową zapewniającą odpowiednie zabezpieczenie danych i możliwość edycji danych po podaniu hasła oraz rozszerzenie do przeglądarki mające ułatwić korzystanie z oprogramowania bez konieczności wyszukiwania rekordów i ręcznego kopiowania hasła.

2.1 Aplikacja okienkowa

Aplikacja powinna spełniać następujące wymagania:

- Bezpieczne przechowywanie danych (szyfrowanie za pomocą silnego algorytmu)
- Dostęp do danych za pomocą hasła głównego
- Możliwość generowania haseł (z uwzględnieniem wyboru zestawu znaków)

- Możliwość wklejania hasła do schowka
- Analiza złożoności podanych haseł

2.2 Rozszerzenie

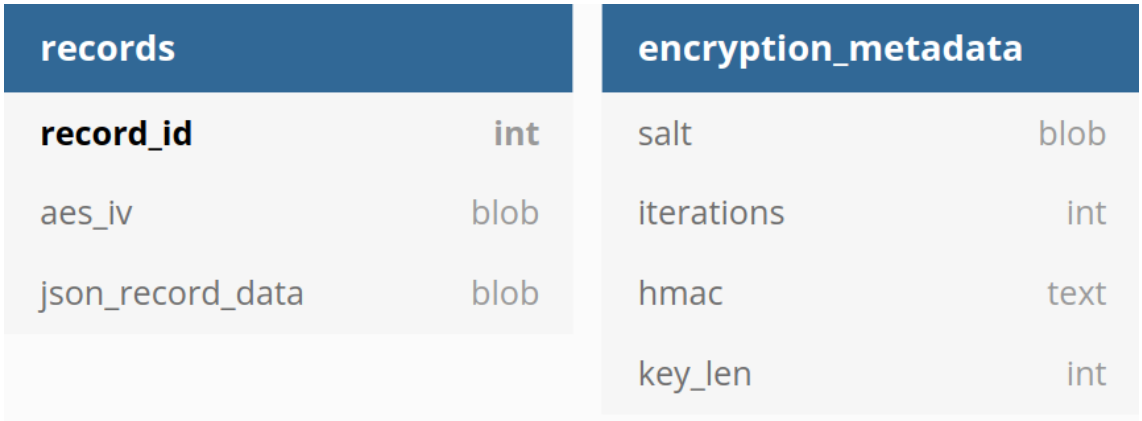
Rozszerzenie do przeglądarki powinno pozwalać na:

- Autozapełnianie pól logowania na stronach internetowych dla których zdefiniowane jest hasło w aplikacji
- Stworzenie nowego rekordu z losowym hasłem dla danej strony
- Automatyczne wklejenie do schowka hasła jeśli autozapełnianie się nie powiodło

3 Implementacja

3.1 Warstwa danych i szyfrowania

Dane są przechowywane lokalnie w postaci pliku bazy danych SQLite3. Schemat bazy danych przedstawiony jest na rysunku 1.



Rysunek 1: Diagram ERD aplikacji

Tabela *encryption_metadata* zawiera dane konfiguracyjne konieczne do działania procedur kryptograficznych. Dane te są wykorzystywane przy użyciu funkcji PBKDF2 – umożliwiającej bezpieczne tworzenie klucza na podstawie hasła. Przeznaczenie parametrów jest następujące:

- *salt* – ciąg bajtów wykorzystywany do dodatkowego zabezpieczenia hasła.
- *iterations* – liczba iteracji algorytmu – konieczne jest użycie na tyle dużej liczby aby spowolnić działanie w celu przeciwstawienia się przed atakami typu *brute force*.
- *hmac* – nazwa wykorzystywanej funkcji skrótu
- *key_len* - długość wynikowego klucza.

Tabela *records* przechowuje właściwe dane użytkownika. Składa się z dwóch pól danych. Pierwszym jest wektor inicjalizujący algorytm AES CBC *aes_iv* przeznaczony do symetrycznego szyfrowania danych. Żeby utrudnić próby złamania algorytmu konieczne jest użycie różnych wektorów dla każdego osobnego użycia algorytmu. Ta wartość może być przechowywana w postaci jawnej[1].

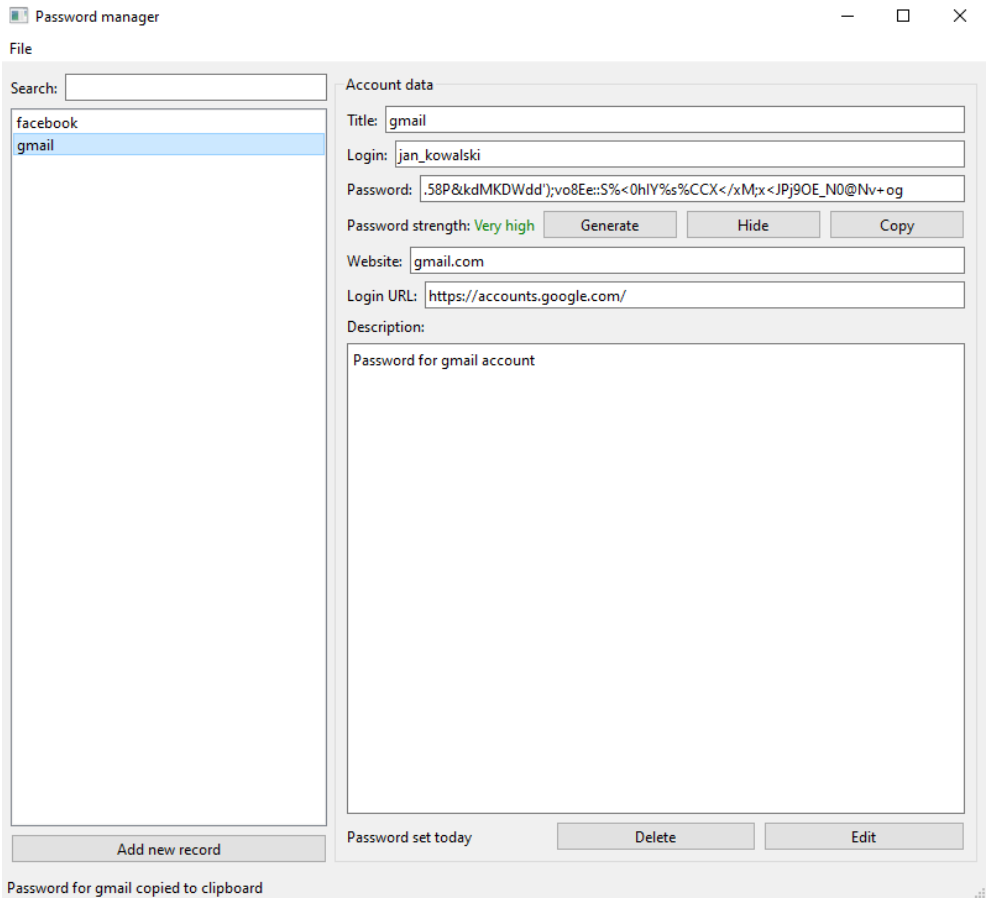
json_record_data	
title	string
website	string
loginUrl	string
login	string
password	string
description	string
modificationDate	integer

Rysunek 2: Struktura obiektu JSON danych użytkownika

Drugim polem są dane właściwe w postaci zaszyfrowanego obiektu JSON. Takie podejście sprawia że nie ma konieczności szyfrowania całego pliku bazy danych.

Struktura obiektu JSON w postaci niezaszyfrowanej jest przedstawiona na rysunku 2. Znajdują się w nim dane potrzebne do logowania (*login*, *password*), dane umożliwiające przypisanie rekordu do danej strony internetowej przy korzystaniu z rozszerzenia (*website*, *loginUrl*) oraz dane dodatkowe opisujące dany rekord.

3.2 Aplikacja



Rysunek 3: Główne okno aplikacji

Aplikacja na komputery osobiste została skonstruowana w bibliotece

pyQt umożliwiającej tworzenie zaawansowanych interfejsów użytkownika. Składa się z widoku logowania, tworzenia nowej bazy, widoku głównego umożliwiającego odczyt i modyfikację danych oraz paneli generacji hasła i ustawień.

Po włączeniu aplikacji użytkownik uzyskuje opcję wybrania istniejącej bazy danych i zalogowania się do niej za pomocą wcześniej wybranego hasła lub stworzenie nowej. Wszystkie dane zostaną zapisane w jednym pliku o rozszerzeniu *.pmdb* w lokalizacji podanej przez użytkownika.

Z poziomu okna głównego (pokazanego na rysunku 3) użytkownik może odczytywać, dodawać, edytować i kasować rekordy. Przy tworzeniu hasło może być wpisane ręcznie (pokazywany jest poziom bezpieczeństwa hasła) lub przez wbudowany generator umożliwiający zdefiniowanie długości i zbioru znaków.

3.3 Rozszerzenie

Rozszerzenie będące częścią projektu tego projektu jest przeznaczone na przeglądarki oparte na Chromium. Do działania wymaga otwartej bazy w aplikacji okienkowej (dane muszą być w danym momencie odszyfrowane).

Schemat jego działanie jest następujący: po wejściu na stronę sprawdzane jest czy istnieją dane użytkownika dla tej strony, jeżeli występują, następuje próba wpisania danych do pól logowania. Jeżeli próba jest nieudana, to hasło jest kopiowane do schowka i użytkownik jest o tym informowany. Możliwe jest także ręcznie ponowienie tej funkcji przez menu kontekstowe.

W celu tworzenia rekordu należy w polach rejestracji lub logowania wpisać wybrany login i wybrać odpowiednią opcję w menu kontekstowym. Nastąpi wtedy zapytanie do aplikacji, wygenerowane zostanie hasło i zwrócone do przeglądarki i nastąpi próba wpisania do formularza.

Aby wykryć pola formularza stosowane są następujące heurystyki: w przypadku pola hasła jest to stosunkowo proste – wystarczy znaleźć elemeenty przez selektor $input[type =' password']$. Dla pól loginu różne strony mogą definiować różne identyfikatory i znalezienie odpowiedniego elementu jest bardziej skomplikowane.

Po przeanalizowaniu kilku popularnych stron zastosowane zostałe następujące heurystyki wyszukiwania pola loginu:

- elementy *input* o właściwości *type* oznaczonej jako *email* lub *login*
- elementy *input* o właściwości *name* oznaczonej jako *email*, *login*, *username*, *user*, $user[login]$, $user[username]$, $user[email]$

- elementy *input* umiejscowione bezpośrednio przed polem hasła

Tak skonfigurowane rozszerzenie pozwala na automatyczne wpisywanie danych na znacznej ilości stron.

3.4 Integracja

Aplikacja łączy się z bazą danych SQLite3 za pomocą biblioteki SQLAlchemy. Dodatkowo wystawiony jest serwer HTTPS w sieci lokalnej komputera za pomocą frameworku Flask, który następnie jest używany przez rozszerzenie.

Dokładne interakcje i przesyłane dane są przedstawione na diagramie DFD (rysunek 4). Aplikacja jest odpowiedzialna za przetwarzanie i szyfrowanie danych, w bazie danych przechowywane są tylko metadane i zaszyfrowane rekordy, które nie mogą być odczytane poza aplikacją. Tak samo rozszerzenie ma dostęp do danych tylko przez wystawiony interfejs.

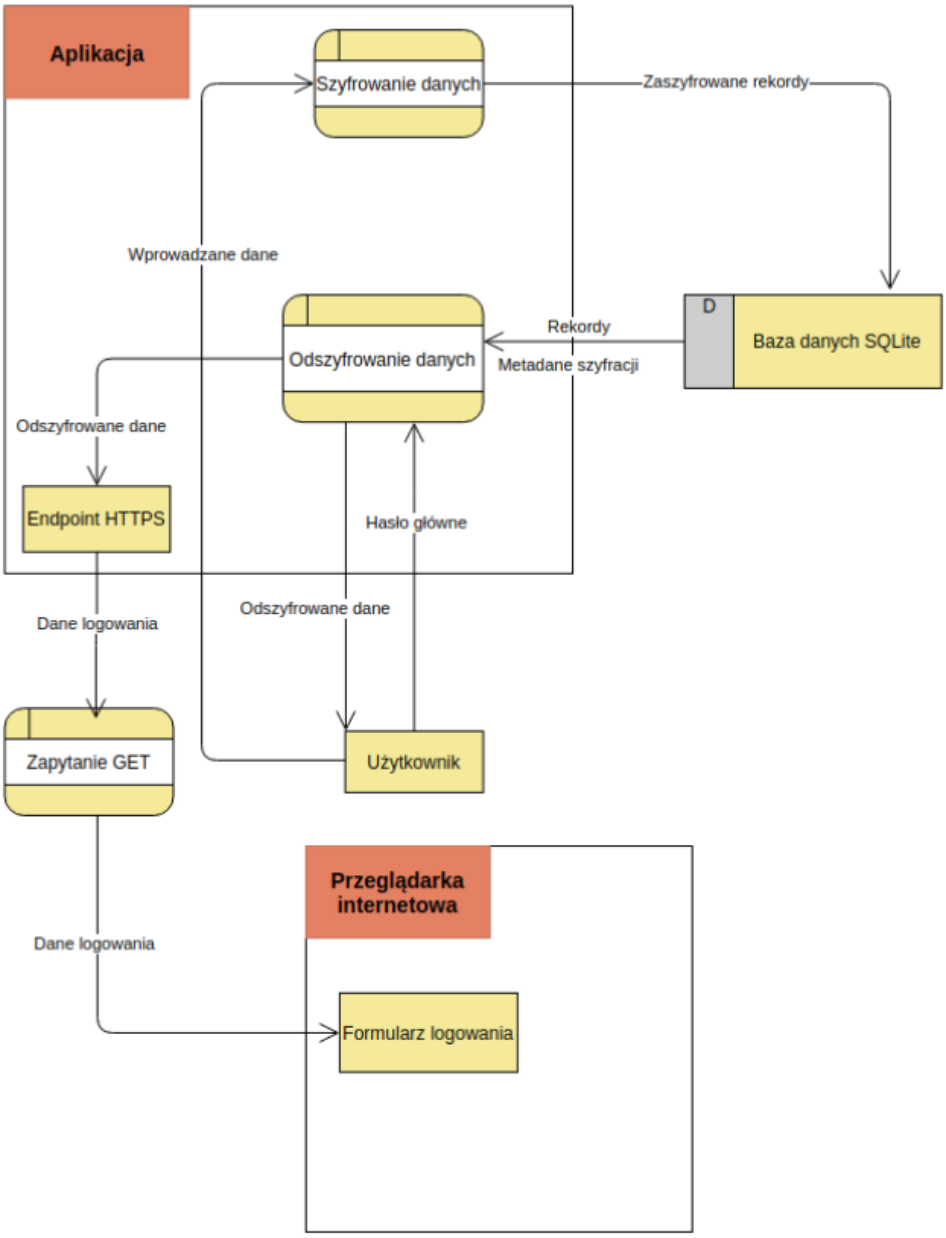
Serwer HTTPS wystawia następujące endpointy:

- /v1/api/sites – zwraca listę stron internetowych dla których zdefiniowane są dane logowania

```
[
  "site1",
  "site2",
  ...
]
```

- /v1/api/password?url={url} – zwraca listę danych logowania dla danej strony:

```
[
  {
    "login": "login1",
    "password": "password1"
  },
  {
    "login": "login2",
    "password": "password2"
  }, ...
]
```



Rysunek 4: Diagram DFD

- /v1/api/createpassword?url={url}&login={login} – Tworzy rekord z losowo wygenerowanym hasłem i zwraca dane logowania:

```
{
  "login": "login1",
  "password": "password1"
}
```

4 Testy

W celu zapewnienia jakości i bezpieczeństwa programu skonstruowane zostały następujące zbiory testów:

- Testy jednostkowe

  Testują pojedyncze komponenty/klasy. Największy nacisk został położony na przetestowanie szyfrowania danych i generacji kluczy/haseł. Dodatkowo testowane są kontrolery odpowiedzialne za poprawne dostarczanie i pobieranie danych z komponentów GUI oraz różnorakie metody pomocnicze wymagane do poprawnego działania systemu.

- Testy integracyjne

  Przeznaczone są do testowania serwera HTTPS w gotowej aplikacji. Testowane są odpowiedzi serwera na zapytania i sprawdzane jest czy opcje kreacji rekordów poprawnie zapisują dane w bazie

- Testy systemowe (razem z GUI)

  Te testy sprawdzają działanie aplikacji z perspektywy użytkownika końcowego. Interakcje są symulowane przez framework testowy QTest. Pokrywają wszelkie możliwe czynności możliwe do wykonania w programie jak: tworzenie bazy danych, logowanie, dodawanie, modyfikacja, usuwanie rekordów, generowanie haseł, kopiowanie haseł do schowka oraz reakcje systemu na błędy użytkownika.

5 Podsumowanie

Na podstawie wcześniej zdefiniowanych wymagań udało się wykonać w pełni funkcjonalny program umożliwiający bezpieczne lokalne przechowywanie i

generowanie silnych haseł. Uwzględnienie rozszerzenia umożliwia łatwe używanie programu na różnych stronach internetowych bez konieczności ręcznego kopiowania z aplikacji.

Przechowywanie danych w pojedynczym, zaszyfrowanym pliku pozwala na łatwe tworzenie kopii zapasowych, które mogą być umieszczone także w niezaufanych lokalizacjach np w dyskach internetowych.

References

[1] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation Methods and Techniques*. en. 2001.