**Dominik Olewiński**

**Nr albumu 159811**

**Dokumentacja aplikacji SecurityApp**

**Opis**

Aplikacja **SecurityApp** demonstruje zastosowanie szyfrowania AES oraz różnice między podatnym i zabezpieczonym podejściem do logowania w bazie danych w kontekście SQL Injection. Kod składa się z dwóch głównych modułów:

1. **AES**: Implementacja szyfrowania i deszyfrowania danych.
2. **SQLInjectionDemo**: Pokaz mechanizmów logowania z wykorzystaniem SQL w wersji podatnej i zabezpieczonej.

**Struktura aplikacji**

**AES (Szyfrowanie i deszyfrowanie)**

* **generateKey()**: Generuje klucz AES-128.
* **encrypt(data, key)**: Szyfruje dane za pomocą podanego klucza.
* **decrypt(encryptedData, key)**: Deszyfruje dane za pomocą podanego klucza.

**SQLInjectionDemo (Demonstracja SQL Injection)**

* **setupDatabase()**: Tworzy bazę danych w pamięci i wypełnia ją przykładowymi danymi.
* **loginVulnerable(username, password)**: Logowanie podatne na SQL Injection (używa dynamicznych zapytań SQL).
* **loginSecure(username, password)**: Zabezpieczone logowanie (używa przygotowanych zapytań).

**Potencjalne problemy i rozwiązania**

**Problem 1: Nieprawidłowe zarządzanie kluczem AES**

* **Opis**: Zgubienie klucza szyfrowania uniemożliwi odszyfrowanie danych.
* **Rozwiązanie**: Zaimplementować bezpieczne przechowywanie kluczy, np. w menedżerze kluczy lub bezpiecznym repozytorium.

**Problem 2: SQL Injection w loginVulnerable()**

* **Opis**: Funkcja **loginVulnerable** jest podatna na ataki SQL Injection, ponieważ bezpośrednio wstawia dane użytkownika do zapytania SQL.
* **Rozwiązanie**: Zawsze używać zapytań przygotowanych, jak w funkcji **loginSecure**.

**Problem 3: Brak obsługi wyjątków w operacjach SQL**

* **Opis**: Problemy z połączeniem z bazą danych lub błędy SQL mogą powodować wyjątki.
* **Rozwiązanie**: Dodać odpowiednie bloki try-catch oraz logowanie błędów.

**Problem 4: Brak walidacji danych wejściowych**

* **Opis**: Użytkownik może wprowadzać dane w nieoczekiwanym formacie.
* **Rozwiązanie**: Wprowadzić walidację danych wejściowych, np. poprzez sprawdzanie długości, znaków specjalnych itp.

**Problem 5: Możliwy brak bezpieczeństwa przy obsłudze kluczy szyfrowania**

* **Opis**: Klucz AES jest generowany i przechowywany w pamięci podczas działania aplikacji.
* **Rozwiązanie**: Usuwać klucz z pamięci po zakończeniu użycia lub używać bezpiecznego środowiska do operacji kryptograficznych.

**Zalecenia dotyczące ulepszeń**

1. Zaimplementować testy jednostkowe dla obu modułów.
2. Rozważyć zastosowanie mechanizmu logowania błędów z wykorzystaniem bibliotek takich jak Log4j.
3. Rozbudować funkcjonalność o interfejs użytkownika (np. graficzny lub webowy).