Opis funkcjonalny projektu w ramach przedmiotu Ochrona Danych w Systemach Informatycznych

Jan Machowski, 325492

sem. 2025Z

1 Wstęp

W ramach projektu zaliczeniowego zostanie stworzona aplikacja w języku Python do bezpiecznej wymiany wiadomości między dwoma użytkownikami, z możliwością wysłania załączników. Aplikacja będzie umożliwiać rejestrację i logowanie użytkownika oraz zaszyfrowane przesłanie wiadomości, wykorzystując aktualne sposoby szyfrowania i zachowania bezpieczeństwa informacji. Aplikacja zostanie napisana w architekturze MVC z wykorzystaniem framework'u Flask oraz potrzebnych do zabezpieczenia technologii opisanych dalej w tym dokumencie.

2 Planowany stos technologiczny

W ramach stworzenia aplikacji wykorzystane zostaną następujące technologie:

- **Python** - język programowania, w którym zostanie napisany backend aplikacji. Szczególnie, zostaną wykorzystane następujące technologie:
  - *Flask* - framework w którym zostanie napisana aplikacja;
  - *Pycryptodome* - biblioteka, z której zostaną wykorzystane funkcje szyfrujące;
  - *html_sanitizer* - biblioteka do sanityzacji danych wprowadzonych przez użytkownika z negatywnym nastawieniem;
  - textitpasslib.hash - moduł biblioteki passlib służący do obsługi kryptograficznych funkcji skrótu.
- **SQLite** - silnik bazodanowy w którym przechowywane będą dane potrzebne do prawidłowego funkcjonowania aplikacji.
- **nginx** - silnik serwera WWW wykorzystywanego jako pośrednika, umożliwiającego korzystanie z aplikacji.
- **docker** - cała aplikacja będzie się znajdować w kontenerze docker w celu łatwej obsługi.

3 Planowana architektura aplikacji

Architektura aplikacji będzie zawierać następujące części:

3.1 Baza danych

W skład bazy danych wchodzić będą następujące tabelki:

1. *users* - Tabela zawierająca informacje o użytkownikach, w tym:
   - nazwa użytkownika
   - ID użytkownika
   - sól + hasło zaszyfrowane funkcją skrótu
   - publiczny klucz RSA
   - ID wiadomości wysłanych
   - ID wiadomości otrzymanych
2. *messages* - Tabela zawierająca przesłane przez użytkowników, zaszyfrowane wiadomości. Jako rekord tej tabelki rozumie się zaszyfrowaną wiadomość oraz jej identyfikator.

3.2 Aplikacja w Python

W ramach projektu planowane jest stworzenie bibliotek odpowiadających za poszczególne funkcjonalności aplikacji. W skład tych bibliotek wchodzić będą:

- *main.py* - plik w framework'u Flask będący kontrolerem między serwerem proxy a backend'em aplikacji;
- *usermgmt.py* - biblioteka do zarządzania kontami użytkowników - tworzenia kont, tworzenia soli dla użytkowników, sprawdzania siły hasła, etc.;
- *attachments.py* - biblioteka do obsługi dodawania załączników - sprawdzania rozmiaru pliku, czy jest odpowiedniego rozszerzenia, sanityzacji pliku, etc.;

3.3 Serwer proxy

Aby umożliwić działanie przeglądarkowe aplikacji wykorzystany zostanie serwer proxy w technologii nginx. Serwer zostanie skonfi

4 Decyzje projektowe

4.1 Struktura wiadomości

Aby zapewnić deterministyczny sposób zarządzania wiadomościami i załącznikami, struktura wiadomości będzie wyglądała następująco:

```
podpis nagłówek wiadomość <|tytył_załącznika|załącznik>
```

4.1.1 Nagłówek

Nagłówek wiadomości będzie bajtem zawierającym m.in. informację o tym, czy wiadomość została przeczytana oraz czy ma załącznik.

4.1.2 Wiadomość

Wiadomość będzie ciągiem znaków w formacie UTF-8, o maksymalnej długości 4096 bajtów. Tekst wiadomości jest integralną częścią przesłania wiadomości.

4.1.3 Załącznik

Jeżeli nadawca będzie chciał wysłać załącznik wraz z wiadomością, zostanie on dołączony w postaci kodu w formacie **base64**, z dwoma znakami — oddzielającymi tytuł od wiadomości oraz załącznik od tytułu. Dzięki temu zabiegowi, po odszyfrowaniu payload'u, będzie można w prosty sposób odróżnić dane należące do wiadomości w porównaniu z tymi, które są załącznikiem.

4.2 Szyfrowanie i podpisywanie wiadomości i załączników

W aplikacji projektowej szyfrowanie wiadomości i załączników będzie wykonywane za pomocą algorytmu **RSA** z wykorzystaniem paddingu **OAEP**, zaimplementowanych w bibliotece *PyCryptodome*. Payload zostanie zaszyfrowany kluczem publicznym odbiorcy, i nie będzie możliwe odczytanie wiadomości wysłanej przez nadawcę.

Przed szyfrowaniem, wiadomość zostanie podpisana za pomocą hasha funkcji **SHA256**. Podpis zostanie doklejony do wiadomości jako pierwsze 256 bajtów.

4.3 Wkład od użytkownika

Każdy wkład od użytkownika zostanie poddany sanityzacji za pomocą biblioteki *html_sanitizer* w celu ochrony przed atakiem XSS. Dodatkowo, nieszyfrowane lub niehashowane dane wprowadzane do baz danych będą parametryzowane w celu ochrony przed atakami SQL Injection.

4.4 Kontrola siły hasła

Aplikacja nie pozwoli utworzyć użytkownika z hasłem, które nie będzie spełniało przynajmniej jednego z poniższych kryteriów:

1. Hasło musi składać się ze znaków należących do dozwolonego alfabetu:
   - Alfanumeryczne: `0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ`
   - Specjalne: `!"#$%&'()*+,-./:;<=>?@[\]^_`
2. Hasło powinno posiadać co najmniej 2 wielkie litery, 1 cyfrę, 2 znaki specjalne;
3. Hasło powinno posiadać co najmniej 12 znaków;
4. Hasło nie może znajdować się na liście 100 000 najpopularniejszych haseł;

4.5 Przechowywanie hasła

Hasło użytkownika będzie przechowywane w następującej formie 64-bajtowego ciągu:

```
sól hash
```

gdzie sól to 16-bajtowy ciąg składający się ze znaków z alfabetu hasła, hash to wynik kryptograficznej funkcji skrótu **argon2** o długości 48 bajtów.

4.6 Opóźnienia i limity prób

W celu ochrony przed atakami brute-force, w aplikacji okreslony zostanie limit 5 prób zalogowania się, po przekroczeniu którego na użytkownika zostanie nałożona 5-minutowy okres oczekiwania.