Opis projektu
Aplikacja do wymiany zaszyfrowanych wiadomości z weryfikacją autentyczności

Marcel Opałko
Indeks: 331722

1 Cel i przedmiot projektu

Przedmiotem projektu jest zaprojektowanie oraz implementacja prostej aplikacji webowej umożliwiającej wymianę wiadomości zaszyfrowanych, wraz z zapewnieniem dowodu autentyczności (weryfikowalny podpis nadawcy), analogicznie do rozwiązań klasy *secure mail*. System ma obsługiwać zakładanie kont użytkowników, logowanie z uwzględnieniem dwuetapowej autentykacji (2FA: TOTP lub HOTP), a także bezpieczne operacje na wiadomościach (podgląd, oznaczanie jako odczytane, usuwanie) i załącznikach.

Załączniki stanowią integralną część wiadomości, co oznacza, że są objęte tym samym modelem poufności oraz integralności co treść wiadomości (szyfrowanie i weryfikacja spójne dla całego ładunku wiadomości).

2 Zakres minimalny funkcjonalny

Minimalny zakres funkcjonalny aplikacji obejmuje:

- rejestrację konta użytkownika;
- logowanie użytkownika;
- dwuetapową autentykację (TOTP lub HOTP);
- wysyłanie zaszyfrowanej wiadomości do co najmniej jednego użytkownika, wraz z załącznikami;
- usuwanie otrzymanej wiadomości;
- oznaczanie otrzymanej wiadomości jako odczytanej;
- podgląd otrzymanej wiadomości oraz pobieranie dołączonych załączników;
- weryfikację autentyczności wiadomości (weryfikacja podpisu cyfrowego nadawcy).

3 Wymagania techniczne

Projekt spełnia następujące wymagania techniczne:

- skonteneryzowanie aplikacji przy użyciu **Docker** (uruchomienie i konfiguracja w środowisku wielokontenerowym);
- wykorzystanie relacyjnej bazy danych **SQL**, przy czym w projekcie przyjęto **SQLite**;
- zapewnienie bezpiecznego połączenia z aplikacją poprzez **HTTPS** z wykorzystaniem serwera WWW (**NGINX**) działającego jako reverse proxy.

4 Proponowany stos technologiczny

W projekcie przyjęto następujące technologie i narzędzia (z możliwością równoważnej zamiany na zgodne funkcjonalnie komponenty):

- **Backend:** Python + FastAPI (API typu REST, walidacja danych wejściowych przez Pydantic);
- **Baza danych:** SQLite (relacyjny model danych, przechowywanie metadanych i zaszyfrowanych ładunków);
- **Reverse proxy i TLS:** NGINX (terminacja TLS, przekazywanie żądań do backendu);
- **Konteneryzacja:** Docker + Docker Compose;
- **2FA:** biblioteka `pyotp` (TOTP/HOTP);
- **Hasła:** Argon2id (`argon2-cffi`);
- **Kryptografia:** biblioteka `cryptography` (szyfrowanie AEAD oraz podpisy cyfrowe).

5 Architektura systemu

System ma architekturę warstwową:

1. **Warstwa kliencka (UI):** interfejs użytkownika realizujący rejestrację, logowanie z 2FA, skrzynkę odbiorczą oraz proces wysyłania i odczytu wiadomości.
2. **Warstwa serwerowa (API):** logika aplikacyjna obejmująca autentykację, autoryzację, obsługę wiadomości i załączników oraz operacje kryptograficzne wymagane przez protokół aplikacji.
3. **Warstwa danych (SQLite):** przechowywanie informacji o użytkownikach, kluczach publicznych, metadanych oraz zaszyfrowanych treści wiadomości i załączników.

Dostęp do backendu odbywa się wyłącznie przez NGINX w trybie TLS; ruch nieszyfrowany jest blokowany lub przekierowywany do HTTPS.

6 Model kryptograficzny i decyzje projektowe

6.1 Przechowywanie haseł

Hasła użytkowników nie są przechowywane w postaci jawnej. Zastosowano funkcję mieszającą przeznaczoną do haseł:

- **Argon2id** z losową solą oraz parametrami kosztu obliczeniowego i pamięciowego.

Dodatkowo implementowana jest kontrola siły hasła (np. minimalna długość i podstawowe reguły złożoności) w celu ograniczenia ryzyka wynikającego z haseł słabych.

6.2 Dwuetapowa autentykacja (2FA)

System obsługuje 2FA w postaci:

- **TOTP** (Time-based One-Time Password) oraz alternatywnie **HOTP** (HMAC-based One-Time Password).

Sekret 2FA jest przypisany do konta użytkownika i przechowywany w bazie z odpowiednimi ograniczeniami dostępu aplikacyjnego.

6.3 Szyfrowanie wiadomości i załączników (poufność i integralność)

Wiadomości oraz załączniki są szyfrowane z użyciem trybu uwierzytelnionego szyfrowania:

- **AEAD** (np. AES-256-GCM lub równoważny algorytm zapewniający poufność i integralność).

Dla każdej wiadomości generowany jest losowy klucz sesyjny K_{msg} oraz nonce/IV. Treść wiadomości i załączniki są traktowane jako ładunek logicznie spójny: ich zaszyfrowane postacie są przechowywane i udostępniane w sposób umożliwiający integralną weryfikację.

6.4 Dystrybucja klucza sesyjnego (szyfrowanie hybrydowe)

W celu umożliwienia odszyfrowania wiadomości przez odbiorcę stosowany jest schemat hybrydowy:

- klucz sesyjny K_{msg} jest szyfrowany asymetrycznie kluczem publicznym odbiorcy (np. RSA lub mechanizm oparty o krzywe eliptyczne, zależnie od implementacji);
- w bazie przechowywany jest ciphertext treści i załączników oraz zaszyfrowany K_{msg} przeznaczony dla danego odbiorcy.

6.5 Autentyczność wiadomości (podpis cyfrowy)

W celu zapewnienia dowodu autentyczności i nienaruszalności pochodzenia wiadomości stosowany jest podpis cyfrowy:

- nadawca podpisuje zaszyfrowaną postać wiadomości (ciphertext) oraz istotne metadane;
- odbiorca weryfikuje podpis na podstawie klucza publicznego nadawcy.

Pozytywna weryfikacja podpisu stanowi podstawę do uznania wiadomości za autentyczną oraz nienaruszoną.

7 Autentykacja, autoryzacja i zabezpieczenia endpointów

7.1 Autoryzacja dostępu

Każdy endpoint operujący na wiadomościach i załącznikach jest zabezpieczony mechanizmem autentykacji oraz autoryzacji. Autoryzacja wymusza, aby:

- dostęp do wiadomości otrzymanej miał wyłącznie jej odbiorca;
- dostęp do informacji o wiadomości wysłanej był ograniczony do nadawcy (w zakresie przewidzianym funkcjonalnie).

Usuwanie oraz oznaczanie jako odczytane realizowane jest per odbiorca (np. poprzez flagi w encji relacyjnej), aby spełnić wymaganie usunięcia wiadomości w skrzynce odbiorcy bez wpływu na inne konta.

7.2 Walidacja danych wejściowych

Wprowadzono walidację danych wejściowych zgodnie z zasadą podejścia restrykcyjnego (*deny-by-default*):

- ograniczenia długości, formatów i typów danych;
- limity rozmiaru przesyłanych załączników;
- odrzucanie pól nieoczekiwanych oraz niezgodnych ze schematem API.

7.3 Ograniczanie prób i opóźnienia

W celu utrudnienia ataków słownikowych i brute-force zastosowano:

- rate limiting dla logowania i weryfikacji 2FA;
- opóźnienia narastające po kolejnych nieudanych próbach;
- jednolity sposób raportowania błędów autentykacji (bez ujawniania szczegółów).

7.4 Minimalizowanie informacji w komunikatach błędów

Komunikaty błędów są projektowane tak, aby nie ujawniać informacji ułatwiających enumerację użytkowników lub wnioskowanie o stanie konta (np. rozróżnienie „nie istnieje użytkownik” vs. „błędne hasło”).

8 Zarys modelu danych (SQLite)

Proponowany zarys encji bazy danych obejmuje:

- **users:** identyfikator, login, hash hasła (Argon2id), sekret 2FA, klucz publiczny, daty utworzenia/aktywności;
- **messages:** identyfikator, nadawca, znacznik czasu, ciphertext treści, podpis cyfrowy;

- **message_recipients:** powiązanie wiadomość–odbiorca, zaszyfrowany klucz sesyjny dla odbiorcy, flagi odczytania/usunięcia;
- **attachments:** powiązanie z wiadomością, nazwa pliku, typ MIME, ciphertext danych, rozmiar.

9 Uruchomienie projektu (Docker) i prezentacja działania

Projekt jest uruchamiany z użyciem Docker Compose. Konfiguracja obejmuje kontenery:

- **nginx:** reverse proxy oraz obsługa połączeń TLS;
- **backend:** aplikacja API (np. uruchomiona przez serwer produkcyjny ASGI);
- **storage/db:** warstwa danych oparta o SQLite (wolumen dla trwałości danych).

Podczas prezentacji przewidziane jest przedstawienie pełnego scenariusza: rejestracja konta, logowanie z 2FA, wysłanie wiadomości z załącznikiem, odbiór, weryfikacja podpisu, oznaczenie jako odczytana oraz usunięcie wiadomości przez odbiorcę.

10 Rozszerzenia (opcjonalnie)

W przypadku rozszerzenia zakresu projektu możliwe jest uwzględnienie:

- ochrony CSRF/XSRF (w zależności od przyjętego modelu sesji i tokenów);
- procedury odzyskiwania dostępu poprzez generację linku resetu hasła (symulacja wysyłki);
- monitorowania logowań oraz sygnalizowania nowych urządzeń/IP;
- honeypotów rejestrujących podejrzaną aktywność;
- polityki Content-Security-Policy (CSP) oraz dodatkowych nagłówków bezpieczeństwa.

11 Bibliografia (propozycja)

- OWASP Cheat Sheet Series (Authentication, Password Storage, Secure Headers, Rate Limiting).
- NIST SP 800-63B: Digital Identity Guidelines.
- RFC 6238 (TOTP), RFC 4226 (HOTP).
- Dokumentacja: Docker, NGINX, FastAPI, `argon2`, `pyotp`, `cryptography`.