JWKS Server Project 3 - Documentation

This project implements a secure JSON Web Key Set (JWKS) server to manage RSA key pairs and user authentication.

Project Summary

* Encrypt RSA private keys using AES before storing them.
* User registration via the /register API.
* User authentication and JWT token issuance via the /auth API.
* Log authentication activities, including IP addresses and timestamps.
* Enforce rate limiting: Maximum of 5 authentication attempts per minute.
* Serve public keys through the /.well-known/jwks.json endpoint.

API Routes

1. User Registration

Endpoint: POST /register

Request Body:

{

"username": "yourusername",

"email": "youremail@example.com"

}

Response:

{

"password": "generated-uuid-password"

}

2. User Authentication

Endpoint: POST /auth

Request Body:

{

"username": "yourusername",

"password": "yourpassword"

}

Response:

{

"token": "signed-jwt-token"

}

* When too many requests are made, a 429 Too Many Requests error is returned.

3. Retrieve Public Keys

Endpoint: GET /.well-known/jwks.json

Provides public RSA keys in JWKS format.

Database Overview

* keys table: Encrypted private keys with expiration times.
* users table: Stores user credentials with hashed passwords.
* auth\_logs table: Records login attempts along with IP addresses and timestamps.

Technologies Used

* Flask — Web application framework
* SQLite — Lightweight database engine
* Cryptography — AES encryption for private keys
* Argon2-cffi — Secure password hashing
* PyJWT — JWT creation
* Flask-Limiter — Request rate limiting

Installation and Running

1. Install dependencies:

pip install flask flask-limiter cryptography pyjwt argon2-cffi

1. (Optional) Set an AES key environment variable:

export NOT\_MY\_KEY="your-32-byte-secret-key"

If not set, a default key is used.

1. Start the application:

python app.py

The server will be available at: http://localhost:8080

Important Points

* Use the /register endpoint to create users before authenticating.
* JWTs are signed using an active (non-expired) private key.
* If no active key exists, /auth will return a 404 error.
* Private keys are AES-encrypted using PKCS7 padding.
* Rate limiting restricts /auth to 5 requests per minute per IP address.

Author: Sahil Chaudhari

Developed for CSCE 3550 - Project 3.