SimPriv API

By: Ayush Kohli
Utsav Dhungel

Content

- Introduction
- Application Features
- API Endpoints
- Workflow Diagrams
- Security Aspect
- Security Vulnerabilities
- Demo

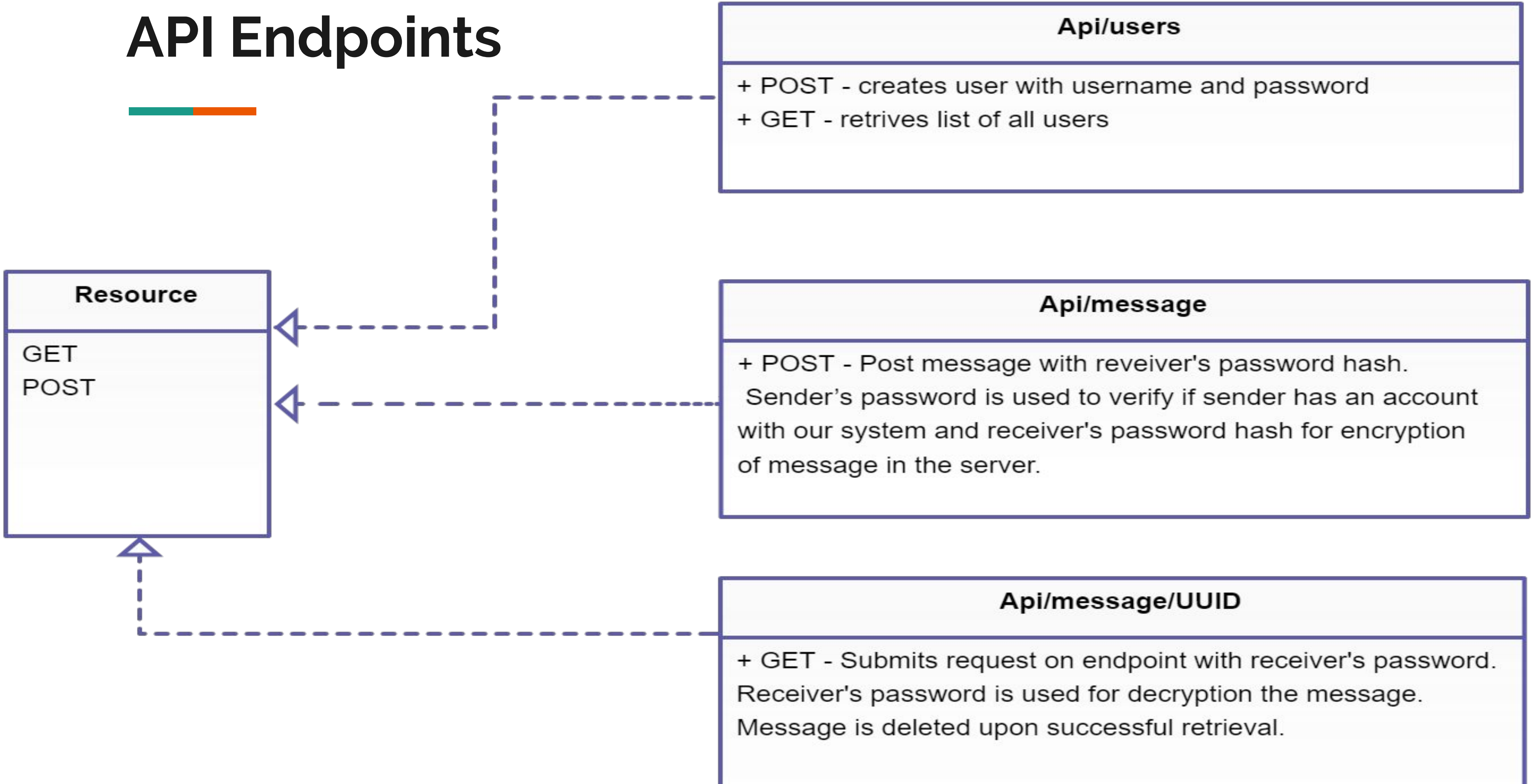
Introduction

- Rest based API for exchanging Private Messages
- JAVA 8 and Spring Boot used for API Development
- Uses SQL Database for storage (H2 database for development)
- Similar services: privnote and snapchat
- Snapchat uses symmetric encryption to encrypt messages and uses the same key for each message between users
- Privnote generates a NoteID after message is created and uses NoteID to encrypt it

Application Features:

- User receives randomly generated password and their username upon signup
- Message is encrypted and decrypted by sender using receiver’s Password hash as key
- Random URL is generated when the encryption is completed
- Receiver decrypts the message with the URL and their own password
- Message gets permanently destroyed when the user retrieves the message.

API Endpoints



Resource

GET
POST

Api/users

+ POST - creates user with username and password
+ GET - retrives list of all users

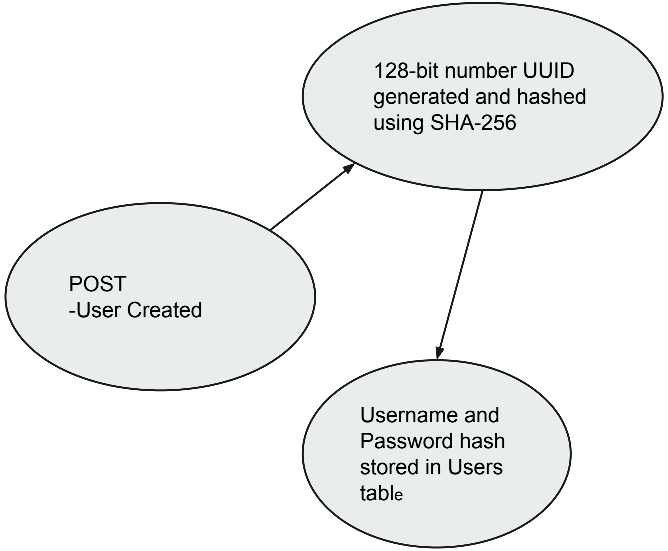
Api/message

+ POST - Post message with reveiver's password hash. Sender’s password is used to verify if sender has an account with our system and receiver's password hash for encryption of message in the server.

Api/message/UUID

+ GET - Submits request on endpoint with receiver's password. Receiver's password is used for decryption the message. Message is deleted upon successful retrieval.

Workflow Diagram:

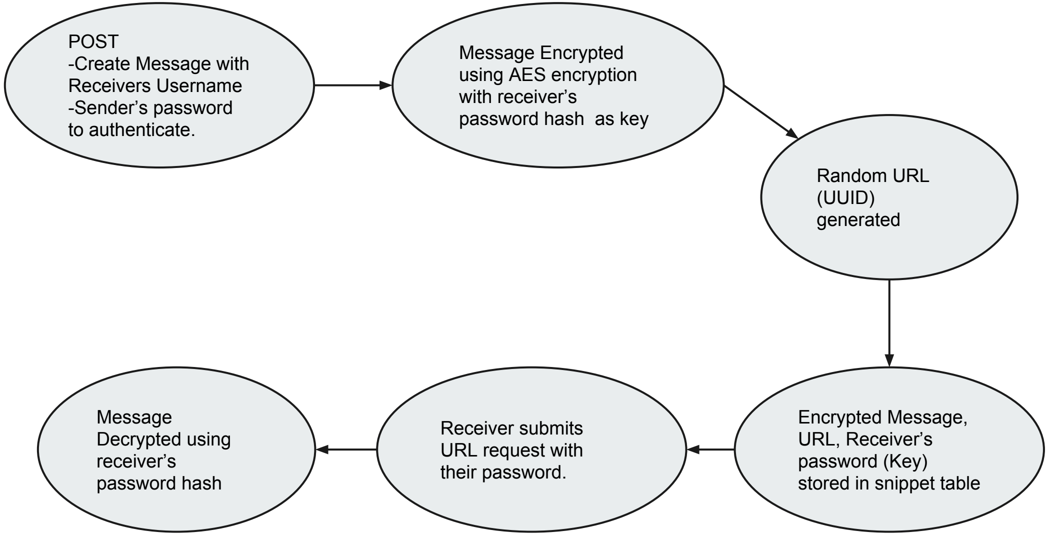


POST
-User Created

128-bit number UUID generated and hashed using SHA-256

Username and Password hash stored in Users table

Message Workflow Diagram



Security Aspect:

- Randomly generated Password key is the 128-bit number UUID
- Password key is hashed using SHA 256 before storing in the database
- Symmetric encryption AES is used to encrypt the message
- Message is encrypted using receiver’s Password hash as a key
- Message is decrypted by the receiver using their username and password
- Message deleted after successful retrieval

Security Vulnerabilities:

- Uses HTTP instead of HTTPS
- AES encryption key is stored in the database
- If the database is compromised then the private message can be decrypted by the attacker
- Random UUID collision
- Brute-Force Attack

API DEMO: