CryptNote Secure Note Access System

Project Report

Overview

The Secure Note Access System is a web application that will enable users to create, encrypt, and access their notes securely. Every note is password-protected and encrypted using the Fernet symmetric encryption scheme. Some salient features include note expiration, destruction after reading, user authentication, and other features put in place to ensure the security and privacy of data. The solution was developed using Django as the system backend, focusing on the simplicity-security-usability paradigm.

Issues Faced

1. **Encryption and Key Management**: Handling encryption keys generated from users password was a very big task. There were cases such as the wrong keys or wrong password to lead to failure in decryption. This was resolved by implementing a utility function, derive_key(), to consistently derive keys and adding error handling to manage invalid keys.
2. **Expiry and Destruction Logic**: The lifecycle handling-expiration or destruction-of the notes was done for with and without expiry dates, and also the case of multiple parallel accesses. The is_expired method was implemented to check for expiry, and those without expiry were destroyed upon access.
3. **SMTP for Notifications**: This involved the configuration and debugging of SMTP so that, on note expiration or note destruction, email notifications could be sent out. The functionality can be achieved using the Django email framework; test emails would ensure it's reliable.
4. **Concurrency Issues**: Inconsistent states may arise, such as many users decrypting a destroyed note while multiple users access the same note at the same instant. While some basic-level protections were implemented, an advanced solution like database level locking was left for future work
5. **Testing and Debugging**: Simulation scenarios such as an expired note or invalid decryption key took rather more time. Heavy unit tests and use of test tools from Django were inevitable parts of the validation and making this module reliable.

Lessons Learned

1. **Cryptography for Note Encryption**: Implementing secure workflows of encryption and decryption with Fernet symmetric encryption has given very practical experience. It was essential to learn key management and proper handling of sensitive data.
2. **SMTP for Notifications**: SMTP setup for notifications was another important thing learned. Fixing issues like delivery failure and integration with the Django email framework added good experience in handling emails.
3. **Secure Coding Practices**: Quite some attention has had to be paid to secure coding practices, including password hashing, sensitive information handling, and error handling. These practices lay the bedrock for any web application to be secure.
4. **Database State Management**: The note lifecycle states, such as marking notes as expired or destroyed, required atomic updates and database consistency.
5. **Security and Usability**: The system had to be designed so as not to compromise its security but still be usable. In this regard, it used features such as human-readable time for expiry and clear messages for error messages.
6. **Testing**: The system was required to undertake tests about encryption workflows regarding their expiry and testing of edge cases to make them reliable. django had lots of built-in tools with which we built our test, pretty efficiently I'd say.

Remaining Bugs or Limitations

1. **Concurrency Issues**: If many users try to access the same note at the same time, minor delays or inconsistencies may arise. Higher-order solutions could be achieved by using database-level locking.
2. **Password Attempts Rate Limiting**: In its present state, the system has no limit for the rate at which passwords can be tried; thus, brute-force attacks cannot be stopped.
3. **Scalability**: It's not optimized for a lot of traffic. Database query optimization will go a long way.

Conclusion

The Secure Note Access System provides a secure but easy-to-use platform for both creating and accessing encrypted notes. It showcases strong encryption workflows, effective expiry and destruction logic, and appropriate user feedback.
Key takeaways include the implementation of cryptography, configuration of SMTP for notifications, and secure coding practices. Some limitations still exist, but the system is a very strong foundation to build upon. With some planned enhancements like rate-limiting, notifications, and improvements in scalability, this system has great potential for real-world adoption.