Advanced Software Engineering
Data Encryption & Password Management System

Team Matrix

Mausam Sanjay Agrawal - msa2213

Nirvishi Shrivastava - ns3499

Rishabh Gupta - rg3334

Rohan Kumar Sachdeva - rks2158

Part 1:

Nothing has changed.

Continuing with the following setup:
Programming language - Python
Platform - Mac
Github: https://github.com/RohanKumarSachdeva/COMSW4156-TeamMatrix

Part 2:

Functionality:
Our service is a data encryption and password management service. It enables the user to do the following:

1. Authenticate and login to our service for accessing user specific information.
2. Encrypt their data and store it in our backend which can be retrieved later.
3. Can generate a random password for an application they are using, which is then saved in our backend in encrypted format.
4. Retrieve, update and delete previously generated passwords.
5. Users can also provide their own password and we inform them how strong it is.

For users concerned with privacy of their data, we let the users provide an encryption key of their choosing (or create a new one using our service) and

encrypt their data/passwords using that. This will make sure only users can decrypt their information with their secret key.

Users: The users of our service can be standalone internet users who want to save their data in encrypted format and/or keep a track of their passwords for the various services they use. Our service can also be used by enterprises who collect user data and do not want to get into the hassles of managing its encryption.

Data Creation: The password creation and management part of our service would simply be creating passwords for end users that they will make use of while logging into the different accounts that they use.
The data encryption component of our service will be used by users/enterprises to encrypt different sorts of data. For enterprises this data could be their proprietary information or the data concerned with their users. For individual users of our service, they can store their personal data such as images, videos, financial documents, etc.

Part 3:

We will test that our service does what it is supposed to do by doing integration testing of all the APIs that we expose. For example, if we perform data encryption of a test file using a key supplied by the user, we will decrypt it and then do a comparison of the 2 files. This will essentially involve calling multiple units/subroutines of our implementation.

We ensure that our service will not behave badly when clients use it unintended ways while implementing our various subroutines. For example, if a user asks us to encrypt a password which is greater than the maximum characters we expect for a password, the application won't proceed with that input. We will do similar checks in our implementation for different functionalities that we expose.

We guarantee that the user data is not even accessible to use if the encryption was done using a key provided by the user. Our design is such that it is computationally impossible for us to decrypt data encrypted using an external key. Also, the data is stored in such a way that if our database is compromised, the attacker won't be able to map the saved information to a particular user.