Jillian Laliberte
CMPT Milestone Project

Abstract

The program developed in this project aims to accept String inputs for Website and Password values from a user and then stores an encrypted version of the password in a hashmap in a separate location. In order to retrieve the values of the password the user must enter a passcode designated by the programmer. Encryption is done by means of a custom symmetrical cipher.

Introduction

This code aims to create a convenient and secure location for the storage and retrieval of passwords for a variety of websites. The relevance of this project lies within increased use of technology within the past decade. Not only does this increase demand for the creation of passwords, it has also consequently led to decreased utilization of previous storage methods. Passwords and other pertinent information are now commonly stored in digital form, rather than written down. Encrypting passwords before storing then prevents this sensitive information from being compromised.

System Description

Classes Encryption and Decryption utilize a custom cipher that is determined by key values only known by the user to alter the contents of a string to be stored within the HashMap storage. Website titles are linked to their respective passwords via a HashMap, however the password stored here is an encrypted version of the String inputted by the user. In order to access previously entered and stored values, the user must input a four digit code to unlock the safe.

Encrypt
- password: String

+ Encrypt ()
+ Encrypt (password)
+ getPassword()
+ encryptString()

Decrypt
- encrypted: String

+ Decrypt ()
+ Decrypt (encrypted)
+ getEncrypted()
+ decryptString()

Requirements

This project addresses one of the most basic aspects of cyber security, password protection. By conveniently storing passwords in an encrypted and locked form, users are able to safeguard information they likely use for more than one website without having to worry about remembering all of their passwords. Utilization of a user determined key during encryption provides increased security as it limits a hacker's ability to guess the code to the cipher.

Literature Survey

Evaluation of literature regarding mechanisms of password protection, the key to safety seems to be never storing passwords as plain text. Regardless of the numeric lock on the safe, passwords can be very easily hacked if not encrypted. Other mechanisms of protection suggest using hashing algorithms rather than encryption. This provides added security because hashing is a one way process as opposed to the two way path of encryption/decryption.

User Manual

Users select an action to use on the password hashmap, their options being to delete, add, or view passwords. The user must input a 4 digit numeric key in order to enact any function. From there, users must input the name of the website and its respective password if they are choosing to store a new input. In addition, the user must enter the encryption key which determines the mechanism by which their password is encrypted and stored.

Conclusion

Convenience and speed are qualities of utmost priority for technology today. Therefore, a simple and easy mechanism of information storage and retrieval is a useful and sought after development. Programs to easily access and store passwords, an aspect of technology used every day, are furthermore a relevant area of study. Via encryption and user specific key inputs, sensitive information is protected and secured within this system.

Bibliography

Sundar, Veera. (2010). Storing passwords in Java web application. Dzone.
https://dzone.com/articles/storing-passwords-java-web