Project Design

CS576

Group Members:

Georgios kapoglis

Johann Lau

Vladislav Ligai

**Description:**

Our password manager “Enclosed” as stated in the proposal will help users store multiple passwords on a secure system and access them from multiple devices (desktop, laptop etc). We will keep our password manager simple and user-friendly while at the same time implement security features to keep our password manager safe and secure.

**Requirements:**

(A)Must have.

Our system will have four main components to it: **client, server, database** and **communication** between both. The server will manage the **database**, which will store the account name, username, and **encrypted** password data**.** Client will handle the input on the user side, and it also will handle the **Blowfish encryption**. Communication channel will be responsible for **client server authentication**, which will be securely encrypted using **Blowfish encryption**. The server is used to parse commands that is being sent from the client and it send the information from the database to the client.

(B)Should have, if at all possible.

Our goal is to make our password manager as secure as possible, so we will be adding additional security features in addition to Blowfish encryption. After we are done with implementing main security features we will attempt to add **two-factor authentication(2FA)** and **complexity requirements** for the user-generated master password. 2FA will consist of master password and security question. Complexity requirements for master password will be: password must be at least eight characters long, it must contain at least one uppercase letter, one number and one special character. If at all possible we will also try to implement **user offline access** for better availability.

(C)Would be nice to have, but not necessary.

Some of our features will not be required for our manager to function properly and will not be implemented unless our system is fully functional. Some of those features include: **random password generator**, **mobile access** and **password autofill option**. Random password generator will allow users to use randomly generated passwords when registering for online services, it will generate passwords randomly with few restrictions: no three same characters in the row, more than eight characters, but less than 16 characters long. Mobile access will allow users to view their passwords from their devices. Password autofill option will allow users to autofill user and password field on the appropriate website with a single button. We also would like to make password manager user friendly, so we will be implementing **graphics user interface** for the ease of use.

**Implementation:**

Alpha System:

The Alpha System of Enclosed that we have designed is comprised of a client. server, our own implemented database. We created our own database that will allow us to store passwords and retrieve them without the use of a SQL database. The passwords that we are retrieved from database is encrypted with MD5, however it will be replaced with a stronger hashing function that will provide enhanced security to our system.

Beta System:

The Beta System of Enclosed will be comprised of the security and bug testing of the system. We secured the communication of the system by using Blowfish encryption to encrypt the data that is transferred between the client and the server. By using the SHA-256 hash of the master password as the key to the secure our system, we encrypted our connection so that the keys are unique every single time. We also implemented an unique database for each user and their master passwords so when a user logs in with his credentials he is able to search through all his stored passwords.

Final System:

The Final System of Enclosed will be the last iteration of Enclosed and will be bug free and stable for use for users. This means that the client will be able to use a GUI to log in with their credentials to obtain their stored passwords. The data is encrypted on the client side so it is secured before it is sent. While the client retrieves the user’s passwords, the data is encrypted using **BLOWFISH** and when the data arrives from the database to the client it is decrypted using the key that is only available on the client side. We also connected the database with the server so that the server creates a new database for each user and master password. To use the passwords that are obtained from the server, we implemented the clipboard to copy the password onto it. After the client disconnects from the server, the clipboard erases itself so that the password is securely deleted from it thus preventing attackers from stealing the passwords.

**Threat Model:**

A password manager has a lot of threats to take into consideration and prevent. This is because a PM store a number of passwords for different users. Each user has passwords for different services stored. Thus a PM could be targeted by a lot of attackers who will try to either obtain a specific password from a specific user or attack the entire manager and obtain as many passwords as they can in order to expose the PM security.

A common attack against Password Managers is sweep attacks. These attacks take advantage of vulnerabilities in autofill completion of passwords. This type of attack can compromise all the passwords of a user through a MITM attack. We will try to implement all the security features needed to prevent such attacks in our PM. If during our evaluation we confirm that our PM is safe from this kind of attack then we will implement autofill on Enclosed.

One of the most common threats that needs to be taken into consideration is MiTM attack which can occur when the user connects to the server from an unsecure network. For example if a user wants to retrieve a password from the server while he or she is on public network(a coffee shop WiFi etc) he or she is vulnerable to these kind of attacks. The communication between the server and the client will be encrypted in order to address this kind of attacks as much as we can.

A vulnerability that we also cover with our security implementations is authorization vulnerabilities of a password manager. Sensitive data are exchanged between the server and the client when a user requests a password. Thus we must ensure that the client is authorized correctly through Public Key Infrastructure that we will be using.

Another security issue is the initial exchange of the cryptographic key between the server and the client. We will address this issue by implementing a secure key exchange protocol.

We assume that the server which will be in Linux-lab will be safe and won't be vulnerable to any kind of attacks that is why we focus on attacks in communication and on the client side.

At last even if an attacker can have access to the user's device where our client will be stored he cannot gain access to the passwords stored in the device or the server(Physical attack). This is because the user has to enter his master password to access his stored passwords. The master password must be kept secret and if the user needs to store it in order to remember it, it must be stored in a secure place that only the user has access to.

Our client and server model protects against many of these attacks. Using a secured encryption and hashing algorithm we are able to encrypt our data so that if an attacker can watch over the connection, they will not be able to decrypt any of our data. Our data is secured using these security features and it will be difficult for attackers to steal any password data. Also the passwords are secured when they are transferred from the client to the user’s destination. The user do not have to worry about any memory left after using their password.

**Evaluation plan:**

To test the Alpha version of the program we will simply run both client and server and establish connection between them. Once we have established connection we will test the password retrieval functions to send the unencrypted password from the server to the client. If the connection is successful and client receive plaintext passwords from the server then alpha version is complete.

While testing the Beta version of Enclosed we will follow the same testing procedure that we used in the alpha version, however we are going to implement more security aspects such as asymmetric key encryption and BLOWFISH. Instead of unsecure channels and plaintext passwords, the Beta version will need to use asymmetric encryption to secure link between client and server as well as encrypted passwords using BLOWFISH. We will also move everything from the command line to the GUI, which needs to be tested thoroughly.

To test out Final version we will need to establish a secure connection between the server and multiple clients and send BLOWFISH passwords, which will need to be decrypted on the client side. After we established that system of retrieval and decryption we will need to test if the database is capable of storing additional passwords and if the client can encrypt and decrypt the passwords securely.

After completing functionality tests, we will test our security, by trying various attacks, such as brute force to make sure of the security of the product.