Password Manager Development and Security Analysis

Zhifan Liang   
Department of Electrical and Computer EngineeringWashington State University, Pullman, WA, USA  
zhifan.liang@wsu.edu

*Abstract*—Passwords are an essential method to verify the user on the internet. More passwords are required to obtain for each person online nowadays. Password manager is a good solution to solve the reuse password problem and improve the user's ability to maintain more passwords. A password manager should be secure enough for store all the sensitive data. In this paper we develop a password manager and perform security analysis on this system. We find the vulnerabilities and propose methods to improve the password manager, as well as provide general security analysis aspects for the password managers.

# Introduction

Passwords are a very effective way to verify the owner of a certain account. With the widely use of internet service, the users are required to obtain more passwords. In order to remember all password without compromise the security, many people use the password manager to organize their password.

## Research Problem

According to Turner [1], average person has 100 passwords. It is very hard for most people remember this many complex passwords, but reuse the password can bring users security issues [2]. As the password managers are used by more people than before, it introduces the concern that how secure is the password managers and how to identify the secure password managers.

Different password manager has different methods to encrypt, store and sync data. User experience will also be influenced by those methods. It is significant to find a way to evaluate those aspects and determine whether we can trust a password manager or not.

## Proposed Solution

In this paper, we develop a basic password manager and use this password manager to analyze the security of password managers. We developed an iOS app as the main password manager, then use AES to encrypt the data in the data. We also proposed several methods to sync the data with the database and find the vulnerabilities of those methods. According to the vulnerabilities we found in the app and the sync methods, we provide improvements and methods to mitigate.

After the development, security analysis and propose mitigation, we compare several popular password managers with the information we have and provide some possible improvements and approaches to evaluate a password manager.

## Contributions

In this paper, we develop an iOS password manager app with different methods of syncing. With the app we developed, we go through the security analysis, find the vulnerabilities of the system and propose the methods to mitigate the issues. After all those practices we provide a basic password manager app, the security analysis can also be used with any other password manager, as a model to check the security.

## Outline of this paper

This paper is organized as follows. Sections II contains the summary of previous work on password manager security analysis and password security discussions. Section III provides a detailed description of the password manager app we developed with design ideas and explanation. This section also contains the discussion of different server sync methods. In section IV, we analyze the password manager in section III, provide methods to mitigate the issues we found. Section V provides the aspects we found important to analyze the security of the password manager. Section VI presents the conclusion we get and the future work. References are listed in the end.

# Related Work

Han et al. [2] point out there are a large number of users reusing their password at multiple websites. They found the shadow attack may effectively attack the user accounts with reusing the passwords. They suggest the user should avoid reusing the password, and if it is too hard to remember, password manager can be a solution.

Current popular password managers bring some security concern, Carlos Luevanos et al. [3] discussed about the strengths as well as the weakness of open-source password managers and closed source password managers. They analyzed three different open-source password managers, found more than one password managers use pseudo random in their generator, and clipboard can also be attacked as there is no protect to stop the third party software to read the clipboard.

Kamat et al. [4] analyzed the techniques of the password manager with some p2p network models implemented in developing the password managers. They discussed the drawbacks of the p2p models, furthermore, how can the password managers benefit from the p2p network architecture. Kamat et al. [4] also compared some popular password managers in the aspects of storage type and the advantages, disadvantages of those password managers.

Fowler[5] discussed about the major flaws and benefits of using a password manager. Fowler[5] claimed though use password manager is not one hundred percent save but it will make the user account harder to crack. Password manager making the accounts harder to hacked and this makes them less likely to be hacked.

Saicheur et al. [6] provided a method to implement AES-128 and AES-512 on iOS devices, they built a structure to implement the AES as well as an app to test the function if their AES implementation.

# Password Manager Development

This section explains the design of the password manager and the security analysis of this system.

## Password Application

The app uses Core Data framework to manage SQLite as the database, the data model contains the account, password, tag, description and a binary data to contain the image allow users to add a thumbnail. After user add account data, the password manager encrypt the password part with AES 256 and store this new data to database. Then, if the user wants to check the data, after tap the data from the main view with a list of all data, the app will show a detail view with the decrypted password. The app uses a preset key for the AES encryption, which is not ideal at the time.

## Implementation of AES

To encrypt the data in the app, we decide to use AES encryption. The preset key is 128 bits for practicing AES-128. The encryption algorithm contains four steps, SubBytes, ShiftRows, MixColumns and Add-round key according to J. Daemen and V. Rijmen [7]. AES turn the clear text of data to a matrix and encrypt the matrix with several rounds. The rounds depend on the size of key shown in Table I. After AES encryption we store the output as a string inside the data model then store the whole data model including unencrypted parts (account string, tag string, description string and image binary data) into the database.

In the Fig. 2 we can see the password part shown in Fig. 1 is encrypted in the database. In order to show how data store in the database, we only encrypted the password part of the data.

1. Rounds related to key size

|  |  |  |
| --- | --- | --- |
|  | Key Size (bit) | Rounds |
| AES-128 | 128 | 10 |
| AES-192 | 192 | 12 |
| AES-256 | 256 | 14 |

Graphical user interface, application

Description automatically generated

Fig. 1. Detail view of password manager, with the password decrypted.

Table

Description automatically generated

Fig. 2. Data in the database

## Server Sync Methods

To sync the data with a server, we tried three methods.

1. Self-host CouchDB server

This server is very vulnerable, with HTTP connection is very hard to achieve an acceptable level of password manager security. We can easily catch the data during transmitting with WireShark [11], that exposes the data entirely. Chordiya et al. [8] provided the method to perform the MITM attack based on the tools with the ability to hijack the data transmitting with HTTP. They showed the traffic can be rerouted from the authenticated server, shows HTTP is not safe enough for transmitting data. Another way to attack the database is XSS, with the right IP address of server it will not be difficult to change the data on the server.

1. Sync the whole database with third party cloud drive

This solution can provide a good enough security. Cloud drives like Google Drive and One Drive require the user to login with their own server security authentication. This authentication introduced an extra layer of security. To ensure security, the app can encrypt the whole database before upload, decrypt the database after download. This will cost more time to process depend on the phone performance and size of database, but for password managers, we think security is more important than user experience.

1. Sync with iCloud

This method is easy to implement with a decent security. Database can be sync through iCloud in the background so encrypting the database will not influence user experience in the most circumstance. The sandbox architecture of the phone system can prevent most unpredictable access to the database and encrypting the database will add an extra layer of security. The major drawback of this method is the limitation of iCloud, it limits the user to access the database from another platform. This limitation could damage the user experience and increase the possibilities to leak the data from the user side, for example, sometimes they have to input the password by hand while having the password show in clear text for a long time.

# Analysis And Mitigation

This password manager system is functional, but not secure enough for practical use. Lacking main password to lock the app will provide access of the database to anyone who has access to the user's phone. There is only one key for AES encryption. In this case, anyone who can access to the source code the password manager has the ability to quickly break into the database. Clipboard can also be a problem. After user copy and paste the password, the password stays on the clipboard. In this situation, any other app can read the clear text password from the clipboard.

To mitigate those issues, to allow the user to setup a custom main password should be the most priority. This improvement will mitigate the first two issues mentioned above. The main password to lock the app can prevent unexpected people to access the app, and this main password with can be the key to encrypt the data. The app stores the hash of the main password, only temperately store the main password after hash verified it and let the user unlock the app, erase the main password after user killed app in the background can prevent the leaking of the main password. To prevent the potential vulnerability of clipboard, the app should override the clipboard after a period of time. 15 second is recommend for the time to clear the clipboard, which is short enough to clear it before the user opens another app without annoying user with instant clipboard cleaning.

# Aspects to Evaluate Password Manager

## Open-source and Closed source

Open-source password managers have their code checked by public, according to Luevanos et al. [3], open-source can help developers to decrease developing costs but may have a slower bug fix and expose the weakness to the attackers.

Closed source may reduce some kinds of attacks as the source code is unavailable to the attackers but users have to trust the developers at the same time. Closed source password manager with outside founding may cause worries, for example, DHH [9] shared his worry when 1password raised money from the outside [10].

## Sync Methods

Self-host server by user with decent database encryption can be a reliable method. This method grants user full control of their data, but it may reduce the sync stability of the password manager. Company host server provided by the developer can enhance the consistency of the syncing experience and in some situations this method can provide abilities to sync between multiple platforms. This method grants the company the power to access the user's database. Without end-to-end encryption to prevent the company from reading data, this method will not provide enough security.

## Overall Security Design of the Application

The app should provide a design to prevent the unauthorized users to access the data inside app. Such as a mandatory main password, sealed password text by default and 2FA. To prevent the other apps from accessing the data is also significant. The app should clear the clipboard after a reasonable period of time and store the data encrypted and inaccessible by another app on the same device.

# Conclusion and Future Work

Although we should use password manager to enhance our internet account's security, we cannot ignore the weakness of password manager. Some vulnerabilities of password manager can be catastrophic, so the user should perform a security analysis before using the service.

The next step should be continuing improve the password manager we develop, mitigate the issues the detected with the methods we proposed. In addition, it is also important to find several widely used open-source password manager to examine their code with security analysis, this analysis will provide more idea to improve the password manager and have a better understanding of the evaluation of security.

##### References

1. J. Turner, “Study Reveals Average Person Has 100 Passwords to Remember,” Tech.co, 26-Oct-2020. [Online]. Available: <https://tech.co/news/average-person-100-passwords>. [Accessed: 09-Dec-2020]
2. W. Han, Z. Li, M. Ni, G. Gu and W. Xu, "Shadow Attacks Based on Password Reuses: A Quantitative Empirical Analysis," in *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 2, pp. 309-320, 1 March-April 2018, doi: 10.1109/TDSC.2016.2568187.
3. C. Luevanos, J. Elizarraras, K. Hirschi and J. Yeh, "Analysis on the Security and Use of Password Managers," 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), Taipei, 2017, pp. 17-24, doi: 10.1109/PDCAT.2017.00013.
4. A. Kamat, C. Tomar, A. Tainwala and S. Akram, "Performance analysis and survey on security of password managers and various schemes of p2p models," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2018, pp. 23-26, doi: 10.1109/RTEICT42901.2018.9012612.
5. G. Fowler, “Review | Password managers have a security flaw. But you should still use one.,” 19-Feb-2019. [Online]. Available: <https://www.washingtonpost.com/technology/2019/02/19/password-managers-have-security-flaw-you-should-still-use-one/>. [Accessed: 09-Dec-2020]
6. V. Saicheur and K. Piromsopa, "An implementation of AES-128 and AES-512 on Apple mobile processor," 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Phuket, 2017, pp. 389-392, doi: 10.1109/ECTICon.2017.8096255.
7. J. Daemen and V. Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard. Springer Science & Business Media, 2002.
8. A. R. Chordiya, S. Majumder and A. Y. Javaid, "Man-in-the-Middle (MITM) Attack Based Hijacking of HTTP Traffic Using Open Source Tools," 2018 IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, 2018, pp. 0438-0443, doi: 10.1109/EIT.2018.8500144.
9. DHH, “Twitter,” 14-Nov-2019. [Online]. Available: <https://twitter.com/dhh/status/1195009383349903361?lang=en>. [Accessed: 10-Dec-2020]
10. M. A. Azevedo, “In Its First Funding In 14 Years, Toronto's 1Password Raises $200M Series A Led By Accel,” Crunchbase News, 14-Nov-2019. [Online]. Available: <https://news.crunchbase.com/news/in-its-first-funding-in-14-years-1password-raises-200m-series-a-led-by-accel/>. [Accessed: 10-Dec-2020]
11. Chapter 4. Capturing Live Network Data. (n.d.). Retrieved December 11, 2020, from https://www.wireshark.org/docs/wsug\_html\_chunked/ChapterCapture.html