Password Manager Security

Nathan Weckwerth, Brian Xia, Jenny Zhang
{nweck, bxia, beining}@mit.edu

May 12, 2020

1 Introduction

The internet has become ubiquitous in our everyday life. There are billions of people who use the internet every day, and that number is only increasing. Companies like Facebook, Google, and Amazon boast hundreds of millions or even billions of users. But most people don't limit their internet usage to a few large sites; in fact, the average American has about 200 different accounts on various websites [1]. This large number of accounts corresponds to a similarly large number of passwords; it's clearly desirable to use a different password for each service to keep any successful attacks localized, and various password restrictions will force at least some differences between passwords. Since it's not reasonable to remember hundreds of distinct passwords, it's common to use some sort of password manager to keep track of them. However, many people may understandably have some hesitation in providing all of their passwords to a password manager.

In this project, we examine the security of various password managers, including those which are built into popular browsers as well as some of the most common external options. Overall, we find that many of these password managers are seriously lacking in security, and many of them have made little to no progress in recent years.

2 Background

Here, we will discuss previous work that has been done on the security of password managers. In particular, we will focus on a study done by Gasti and Rasmussen in 2012 [2]. In their study, they introduce two security definitions, which we will delineate and utilize in our own analyses.

2.1 Related Work

Since Gasti and Rasmussen's study, there have been numerous other evaluations done on the security of password managers. We highlight three in particular in this section. For our project, we focused on many of the same vulnerability areas as in these prior studies, extended their methods to other password managers, and evaluated new security procedures these managers have introduced since they were studied in these works.

Li et al. have recently conducted a security analysis of various web-based password managers, particularly LastPass, RoboForm, My1login, PasswordBox, and NeedMyPassword [3]. The employed threat model revolved around a web attacker with access to multiple web servers and DNS domains. They defined four key vulnerabilities: bookmarklet, web, authorization, and user interface. Briefly, bookmarklet vulnerabilities lie in malicious APIs that are typically available to web applications. Web vulnerabilities lie in management of its serviced web applications and potentially untrustworthy sites. Authorization vulnerabilities lie in unintentional authorization leading to credential access. User interface vulnerabilities lie in phishing attacks. All of the aforementioned web-based password managers were found to suffer from one or more vulnerabilities, highlighting their insecurity.

Another recent study by Silver et. al. examined the autofill policies of many password managers, including Chrome, Firefox, Safari, 1Password, and LastPass [4]. When users visit a website for which they have password information saved in a password manager, the password manager must decide at which point to automatically fill in the user information. They discovered that the autofill policies of the various password managers actually varied significantly, and that many of the policies were vulnerable to network attacks that could extract the password information without any input from the user, simply by fooling the password manager into thinking the user was attempting to visit various websites and to log in to their accounts.

Finally, a study by Carr and Shahandashti focused on vulnerabilities of commercial password managers, external services such as 1Password separate from in-built functions in browsers - Dashlane, LastPass, Keeper, 1Password, and Roboform [5]. They investigated both the browser and Android versions of these managers and found many were vulnerable to phishing attacks where an attacker poses as a legitimate application or webpage, and the password manager can be tricked into inputting information into the malicious site. They also identified vulnerabilities associated with copying data to the clipboard and insecure PINs vulnerable to brute force attacks.

2.2 Security Definitions

Here, we will review the security definitions constructed by Gasti and Rasmussen [2].

2.2.1 IND-CDBA

A challenger $\mathtt{Ch}$ interacts with $\mathtt{Adv_r}$ as follows:

- $\texttt{Adv}_\texttt{r}$ outputs two record-sets RS_0, RS_1
- $\texttt{Ch}$ selects a bit b uniformly at random and the database $DB_b \leftarrow \texttt{Create(}RS_b\texttt{)}$ is returned to $\texttt{Adv}_\texttt{r}$
- $\texttt{Adv}_\texttt{r}$ outputs bit b'
- The game outputs 1 iff $b' = b$

We say that $\texttt{Adv}_\texttt{r}$ wins the IND-CDBA game if it can cause it to output 1.

2.2.2 MAL-CDBA

A challenger $\texttt{Ch}$ interacts with $\texttt{Adv}_\texttt{rw}$ as follows:

- $\texttt{Adv}_\texttt{rw}$ adaptively outputs n record-sets RS_i and receives, from $\texttt{Ch}$, the corresponding databases $DB_i \leftarrow \texttt{Create(}RS_i\texttt{)}$
- $\texttt{Adv}_\texttt{rw}$ outputs DB'
- The game outputs 1 iff $\texttt{Valid(}DB'\texttt{) = 1}$ and $DB \neq DB_i$ for $i \leq n$

We say that $\texttt{Adv}_\texttt{rw}$ wins the MAL-CDBA game if it can cause it to output 1.

3 Database Format Vulnerabilities

In the following subsections, we describe the database formats and encryption schemes for various password managers, and we discuss how well they meet our security definitions.

3.1 Google Chrome

Format Description

Google Chrome allows users to store usernames and passwords locally in the form of an SQLite database. In particular, the database consists of three fields: origin_url, username_value, and password_value. The origin_url and username_value fields are stored as plaintext while the password_value field is encrypted using Window's `CryptProtectData` function [6]. `CryptProtectData` encrypts passwords using Triple DES in CBC mode and a SHA-1 based HMAC. Importantly, the symmetric key used with Triple DES is derived from a randomly-generated MasterKey using PBKDF2 [7].

Security Analysis

Google Chrome is not IND-CDBA secure. In the IND-CDBA game, $\texttt{Adv}_\texttt{r}$ creates two record-sets RS_0 and RS_1 that differ in at least one origin_url. $\texttt{Adv}_\texttt{r}$ can then easily identify whether DB_b originates from RS_0 or RS_1 since the origin_url fields are stored simply as plaintext.

Chrome is not MAL-CDBA secure. In the MAL-CDBA game, $\texttt{Adv}_\texttt{rw}$ generates any nonempty RS and receives the corresponding DB from $\texttt{Ch}$. $\texttt{Adv}_\texttt{rw}$ can then generate DB' from DB by replacing any of the origin_url fields with another valid origin_url field.

3.2 Firefox

Format Description

Like Chrome, Firefox stores passwords and other autofill information in a local SQLite database. Passwords are kept in the `logins.json` file. The `logins.json` contains fields:

- `id`
- `hostname`
- `httpRealm`
- `usernameField` (encrypted)
- `passwordField` (encrypted)
- `guid`
- `encType`
- `timeCreated`
- `timeLastUsed`
- `timePasswordChanged`
- `timesUsed`

Keys used for encryption are stored in `key4.db`. Additional form autocomplete information is kept in `formhistory.sqlite` [8].

Passwords are stored encrypted, but an attacker with access to the laptop can view all passwords. Users can optionally set a master password to prevent this attack.

For password encryption, the both the master password and passwords are encrypted with 3DES-CBC. The initialization vector for the master password is derived from the master password, and the entropy of the password is bounded 112 bits. The master password is not repetedly hashed. For individual passwords, the IV is in the metadata and the generation method cannot be guessed.

Security Analysis

As in 2012, Firefox is neither IND-CDBA secure nor MAL-CDBA secure for the same reasons Chrome is not. In the IND-CDBA game, $\texttt{Adv}_\texttt{r}$ can create two record-sets RS_0 and RS_1 that differ in at least one origin_url. $\texttt{Adv}_\texttt{r}$ can then easily identify whether DB_b originates from RS_0 or RS_1 since the origin_url fields are stored simply as plaintext.

In the MAL-CDBA game, $\texttt{Adv}_{\texttt{rw}}$ can generate any nonempty RS and receives the corresponding DB from `Ch`. $\texttt{Adv}_{\texttt{rw}}$ can then generate DB' from DB by replacing any of the origin_url fields with another valid origin_url field. Firefox is thus still vulnerable to the man-in-the-middle attack described in [2], where an adversary can modify the URL to cause the user to submit sensitive information to an adversary-controlled site.

3.3 Internet Explorer

Format Description

Internet Explorer stores user information in a local registry. Each entry in the registry corresponds to a website; the name of each entry is just a hash of the website url. The value of each entry is the username and password for the corresponding website, encrypted with the native `CryptProtectData` function which was described in section 3.1 of this paper [9].

Security Analysis

Internet Explorer is not IND-CDBA secure. $\texttt{Adv}_{\texttt{r}}$ can create two record-sets RS_0 and RS_1 that differ in at least one url. $\texttt{Adv}_{\texttt{r}}$ can then easily identify whether DB_b originates from RS_0 or RS_1 since they can simply check if there is an entry in the registry with the hash of the distinct url.

Internet Explorer is not MAL-CDBA secure. $\texttt{Adv}_{\texttt{rw}}$ can generate any nonempty RS and receives the corresponding DB from `Ch`. $\texttt{Adv}_{\texttt{rw}}$ can then generate DB' from DB by renaming any of the entries in the registry to the hash of a different website's url.

3.4 Microsoft Edge

Microsoft Edge utilizes the same format as Internet Explorer. Thus, Microsoft Edge shares the same security flaws as Internet Explorer.

3.5 Safari

Format Description

Safari stores user information in a local 'keychain' file which can be directly converted to an XML file. In this file, there is a set of entries, with each entry corresponding to a different account on a different website. Various metadata is stored, along with the encrypted password and username. While the password and username are encrypted with `CryptProtectData` as described in section 3.1, the rest of the information is just present in plaintext [10]. In particular, the website url is not encrypted.

Security Analysis

Safari is not IND-CDBA secure. $\texttt{Adv}_\texttt{r}$ can create two record-sets RS_0 and RS_1 that differ in at least one url. $\texttt{Adv}_\texttt{r}$ can then easily identify whether DB_b originates from RS_0 or RS_1 since they can simply check if the distinguishing url appears in plaintext in the database.

Safari is not MAL-CDBA secure. $\texttt{Adv}_\texttt{rw}$ can generate any nonempty RS and receives the corresponding DB from $\texttt{Ch}$. $\texttt{Adv}_\texttt{rw}$ can then generate DB' from DB by editing the url and other metadata for some entry in the database to information corresponding to a different website, since all this metadata is unencrypted.

3.6 Roboform

Format Description

Roboform stores all usernames, passwords, url, and metadata in a single encrypted file [11]. This file is encrypted with AES-256. The key for this encryption is generated from a master password selected by the user by adding a 32 byte random salt and running PBKDF2 for a large number of iterations [12].

Security Analysis

Roboform is IND-CDBA secure. To determine the source of DB_b, the adversary $\texttt{Adv}_\texttt{r}$ must gain some partial information from the databases about their contents, which is impossible since they are encrypted with AES-256. Since the key for this encryption is selected by PBKDF2, the adversary will be unable to find the key or break the encryption and thus unable to determine the source of the database.

Roboform is MAL-CDBA secure. Without knowing the key for the AES encryption, which is impossible since it is selected by PBDKF2, the attacker will be unable to forge any file encrypted by AES-256 with that key, and thus can never forge a false database.

3.7 1Password

Format Description

1Password uses JSON files to store passwords. There is a `profile.json` file containing the master and overview keys. The overview key is used to encrypt the `folders.json` file, which defines user-created folders. Finally, records are split into 16 band files based on each item's UUID. PBKDF2 is used for key derivation, in getting the encryption and MAC keys from the master password. [13].

In December 2012, 1Password moved from the Agile Keychain format to the OPVault format for syncing with iCloud and Dropbox. With this change, they now encrypt all information, including Location and Title fields of records, except the following metadata: folder, category, creation time, modify time, and last sync time. Previously, 1Password does not have to be "unlocked" for potential website matches to be returned; now, the user must input their master password to see any of the records' content, including title of websites. Additionally, 1Password now uses Encrypt-then-MAC authenticated encryption using HMAC-SHA256 and AES-CBC with 256 bit keys. HMACs are calculated over elements in each item.

Security Analysis

1Password is not IND-CDBA secure, but it is safe in the MAL-CDBA game.

$\mathtt{Adv_r}$ can win the IND-CDBA game by creating two same-size record-sets RS_0 and RS_1 that differ in at least one of the unencrypted fields (folder, category, creation time, modify time, last sync time). As these fields are not encrypted, the adversary can determine bit b with probability 1 by testing which record belongs to DB_b.

Despite 1Password not being IND-CDBA, it's worth noting that compared to when 1Password still utilized Agile Keychain in 2012, an adversary with access to the user's 1Password database can gain significantly less information now. An adversary now can still discover when the user was creating and modifying records. They can potentially guess the website due to the category and folder, combined with any other data they may have discovered about the user. However, they now cannot determine with certainty which record belongs to which website, even with some of the metadata being unencrypted. An attacker can still learn information about the user's browsing patterns, which is undesirable. Compared to other password managers like Chrome and Firefox, 1Password does better in protecting the security of records and with regards to encrypting sensitive information.

The added authentication protects 1Password from adversaries in MAL-CDBA. Due to the sensitive information in each record being encrypted and an HMAC being calculated over each record, an adversary is unlikely to win the game as that would require generating a valid HMAC for their modified entry.

3.8 PasswordSafe

Format Description

PasswordSafe v4 continues to have one file with a header, containing the keys used to encrypt passwords, and the body, which contains the encrypted passwords. The file contains an HMAC at the end [14].

Security Analysis

As shown in [2], PasswordSafe is IND-CDBA and MAL-CDBA secure. The design flaw brought up in the same paper has not been addressed in all versions. In the Java implementation, keys K and L are regenerated and the passwords reincrypted when the master password is changed [15]. Thus, depending on the version, PasswordSafe is still vulnerable to the same attack in which an attacker who discovers the user-set master password and thus retrieves K and L will have continuous access to the passwords even after the password is updated.

3.9 LastPass

Format Description

LastPass has its users create a master password consisting of upper case, lower case, numeric, and special character values that has at least 12 characters. Encryption keys, generated from the master password using PBKDF2, are used in 256-bit AES encryption [16]. Each entry in LastPass consists of 9 fields:

- `folder`
- `aid`
- `name`
- `extra`
- `grouping`
- `username`
- `password`
- `url`
- `urid`

The `folder`, `aid`, `extra`, and `urid` fields are extraneous metadata that are not particularly useful to an adversary in a practical sense. Importantly, the name, grouping, user, and password fields are encrypted using the aforementioned 256-bit AES encryption, but the `url` is simply the hexadecimal encoded string of the plaintext url.

Security Analysis

LastPass is not IND-CDBA secure. In the IND-CDBA, $\texttt{Adv}_{\texttt{r}}$ can create two record-sets RS_0 and RS_1 that differ in at least one `url`. $\texttt{Adv}_{\texttt{r}}$ can then easily identify whether DB_b originates from RS_0 or RS_1 since the `url` fields are stored simply as hexadecimal encoded strings.

LastPass is not MAL-CDBA secure. In the MAL-CDBA game, $\texttt{Adv}_{\texttt{rw}}$ can generate any nonempty RS and receives the corresponding DB from `Ch`. $\texttt{Adv}_{\texttt{rw}}$ can then generate DB' from DB by replacing any of the `url` fields with another valid `url` field. The valid `url` field can easily be generated by taking a plaintext url and converting it to a hexadecimal encoded string.

	IND-CDBA	MAL-CDBA
Chrome	×	×
Firefox	×	×
Internet Explorer	×	×
Microsoft Edge	×	×
Safari	×	×
Roboform	✓	✓
1Password	×	✓
PasswordSafe	✓	✓
LastPass	×	×

Table 1: A summary of our analysis. The password managers are split into browsers and external tools, and for each password manager we note whether it is IND-CDBA and MAL-CDBA secure.

4 Conclusion

Table 1 summarizes the results of our security analysis. Of the password managers we have studied, the majority are vulnerable to attack in the IND-CDBA or the MAL-CDBA game. In particular, all of the browsers are vulnerable in both settings; in general, they all fail to provide adequate security because the url or a hash of the url is available in plaintext to the attacker for each website stored in the password manager. Overall, the external password managers do tend to perform better than the browsers. Roboform and PasswordSafe pass both security definitions, and 1Password passes MAL-CDBA and only fails to pass IND-CDBA due to leaking certain metadata. LastPass, however, unfortunately falls prey to the same attacks that affect the browser-based password managers.

If we compare to the previous results from 2012 by Gasti and Rasmussen, we can notice that both Roboform and 1Password improved in this time frame and became more secure according to our definitions; PasswordSafe was already secure in these settings in 2012, and LastPass was not studied.

Although many of the password managers do not meet our strict standards of security, this does not mean they are inherently flawed. Many users may not be too worried about security of their password managers; if they never let any attacker access their machine at all, they avoid the attacks we describe. Furthermore, most of these password managers, especially those which are browser-based, don't make any claims about security, merely presenting themselves as a convenience for their users. However, for users who are concerned about these attacks, there are options in the external password managers which do provide security which meets our strict definitions.

References

[1] Tim Johnson. *How Many Passwords Can You Remember?* McClatchy Washington Bureau, 2018.

[2] Pablo Gasti and Kasper Rasmussen. *On The Security of Password Manager Database Formats.* UC Irvine, 2012.

[3] Li et al. *The Emperor's New Password Manager: Security Analysis of Web-based Password Managers* UC Berkeley, 2014.

[4] Password Managers: Attacks and Defenses. `https://crypto.stanford.edu/~dabo/pubs/papers/pwdmgrBrowser.pdf` Usenix Security, 2014.

[5] Michael Carr and Simak F. Shahandashti. Revisiting Security Vulnerabilities in Commerical Password Managers. 2020. `https://arxiv.org/pdf/2003.01985.pdf`

[6] Jason Faulkner. *How Secure are Your Saved Chrome Browser Passwords?* `https://www.howtogeek.com/70146/how-secure-are-your-saved-chrome-browser-passwords/`. How-To Geek, 2013.

[7] Windows Data Protection. `https://docs.microsoft.com/en-us/previous-versions/ms995355(v=msdn.10)?redirectedfrom=MSDN`. Microsoft, 2010.

[8] Mozilla. Firefox Password Manager `https://support.mozilla.org/en-US/kb/password-manager-remember-delete-edit-logins`.

[9] Exposing the Password Secrets of Internet Explorer. `https://securityxploded.com/iepasswordsecrets.php` SecurityXploded, 2020.

[10] Exposing the Password Secrets of Apple Safari. `https://securityxploded.com/safari-password-secrets.php` SecurityXploded, 2020.

[11] Where is my Roboform for Business Data Stored? `https://help.roboform.com/hc/en-us/articles/360000701611-Where-is-my-RoboForm-for-Business-data-stored-` Roboform, 2020.

[12] Roboform Security White Paper. `https://www.roboform.com/pdf/RoboForm_Security_White_Paper.pdf` Roboform, 2018.

[13] 1Password OPVault design. `https://support.1password.com/opvault-design`

[14] Password Safe. `https://pwsafe.org/`

[15] Java PasswordSafe. `https://sourceforge.net/projects/jpwsafe/`

[16] Secure product architecture. `https://www.lastpass.com/enterprise/security`. LastPass, 2020.

[17] PSA: LastPass Does Not Encrypt Everything In Your Vault. `https://hackernoon.com/psa-lastpass-does-not-encrypt-everything-in-your-vault-8722d69b2032`. Hacker Noon, 2017.