

Evaluation of Professional Cloud Password Management Tools

Daniel Schougaard¹, Nicola Dragoni^{1,2}(✉), and Angelo Spognardi¹

¹ DTU Compute, Technical University of Denmark, Lyngby, Denmark
ndra@dtu.dk

² Centre for Applied Autonomous Sensor Systems, Örebro University,
Örebro, Sweden

Abstract. Strong passwords have been preached since decades. However, lot of the regular users of IT systems resort to simple and repetitive passwords, especially nowadays in the “service era”. To help alleviate this problem, a new class of software grew popular: password managers. Since their introduction, password managers have slowly been migrating into the cloud. In this paper we review and analyze current professional password managers in the cloud. We discuss several functional and non-functional requirements to evaluate existing solutions and we sum up their strengths and weaknesses. The main conclusion is that a silver bullet solution is not available yet and that this type of tools still deserve a significant research effort from the privacy and security community.

1 Introduction

For many years, IT professionals have preached the importance of strong passwords. Many publications exist, describing exactly what defines a strong password and user habits [1]. The general consensus is that it needs *at least* both upper- and lower-case letters, digits and preferably also symbols (*#, -, etc.*). Additionally, it should not be a word, or a word where an L is replaced by a 1. And of course it has to be at least 8 characters long. More importantly, the user is not supposed to use the same password for more than one service. With all of these rules for strong passwords, it comes as no surprise that many low-security-educated users of IT services resort to simple and repetitive passwords.

To help alleviate this problem, a new class of software grew popular: Password managers. Those are simple tools, usually protected by a single master password, able to generate and store in a secure manner, distinct and hardly-to-guess passwords in place of the user herself. A lot of the IT professionals took these tools to their heart, despite their inherent —very often hidden— flaws.

As with many other contexts in modern society, the users crave convenience. In particular, tools storing an encrypted file with all the password locally, was no longer sufficient, as the majority of users began to use multiple devices and needed to have passwords available in all of them. Hence, the password managers slowly migrated into the cloud. This also saved the users from the hassle of managing their passwords, themselves: the users unload some of the “responsibilities” onto third parties and their data are kept for them, available at all times, from any device.

While the cloud *does* come with its benefits, especially convenience, it has its own drawbacks as well, primarily *trust*. When uploading data into the cloud, the user is effectively trusting the service provider. She is trusting that the provider is completely honest about the inner working of its service, mainly regarding what it can and can not access. Users are trusting the providers when they say that they do not share their information to third parties. Unfortunately, sometimes this trust is betrayed, mainly when service providers experience technical incidents. In the context of cloud password managers, for example, it is well known the involving LastPass company in 2015¹. As many IT professionals had feared, the online password manager had a breach. Panic arose and LastPass almost forced their users to change their passwords.

However, even if trust is a general issue with the cloud, in the case of password managers it is particularly critical, as the user trusts a service to store confidential information that give access to, potentially, all the other services the user everyday accesses. Thus, it is ultimately important to have a detailed knowledge and a objective security assessment of the password manager services available in the cloud.

Contribution and Outline of the Paper. The main contribution of this paper is a comparative and critical security analysis of the different alternatives available for the user, with the final aim to understand if a suitable manager already exists or if (as it is) further efforts are required to provide adequate protection to users' passwords. In particular, in this paper

- we consider and discuss functional and non-functional requirements for password manager services in the cloud;
- we survey and perform a usability and security assessment of 14 typologies of professional password manager tools available in the cloud;
- we compare the results of the assessment and focus on the main weaknesses

We think that the final outcome of our analysis will raise the awareness of the less-security-aware users and will call the IT community for a higher effort to face the password management in the cloud. We want to stress that the paper is focused on available professional password manager tools, while purely academic approaches are left as future work.

The rest of the paper is organized as follows: next section contains the analysis of the functional and non-functional requirements a password manager service in the cloud should guarantee. Section 3 is focused on the description of the password manager services considered for this paper, while Sect. 4 contains the comparison of the obtained results. Section 5 concludes the paper with some final future directions.

2 Functional and Non-functional Requirements

In this section we report and briefly describe the most desirable requirements a cloud password manager service should have. We distinguish between functional

¹ <http://krebsonsecurity.com/2015/06/password-manager-lastpass-warns-of-breach/>.

and non-functional requirements [20]. The former define the expected functioning of the system, namely what the system is expected to do, while the latter refer to qualities of the system, including performance, usability, reliability and so on. In the next subsection we identified 17 functional requirements, as desirable features of the system.

2.1 Functional Requirements

The first mandatory functionality a cloud password manager solution should have is the accessibility: namely, it should be possible to have access to the passwords from several physical devices. We say that it has to have a *distributed password database*. Another desired functionality is that it should support *multiple* —individual— users. It should also be possible to distinguish between *administrators* and regular users. In order to better restrict outside access, the admin will have to create a new user. This can be done either with the admin actually setting up the user, or an invite to registration.

It should be possible to organize passwords in a *structured way* and in multiple levels, customizable by the individual users, for the best user experience. For convenience, it should also be possible to selectively *share passwords*, according to the user needs.

The desirable solution should be *platform agnostic*, and should not be limited to one specific server software. In particular, the user should be able to choose what type of underlying storage/*database*, he or she prefers to use. This would also make possible to run it on low powered devices.

No password —or any other sensitive data— should *ever be present unencrypted* anywhere else, than a local device. This ensures that even if another part of the solution is somehow compromised, data is not revealed on that device.

The users should be able to audit access to their personal data including, but not limited to, *retrieving* passwords, adding/*changing* passwords, and *deleting* passwords. This should be done leveraging a *logging system*, able at least to record detailed access time and the remote host. This ensures that a user can detect if, when and from where unauthorized accesses have occurred.

Access to the system should be protected by the users *master password*, and it should be *possible to change* it. Enabling and using a *two-factor authentication* mechanism should be a possible option. Finally, to protect the availability of the system, we would require that the client-side of the system should *automatically restart* after a hardware reboot.

2.2 Non-functional Requirements

Considering non-functional requirements, we selected 7 desirable properties of password manager services. Firstly, we would require that there is the option to store the passwords where the user has *control over*. This would make the system more flexible, since it would open the way for a password manager in a private cloud.

In order to promote further development, allowing for use of various open source frameworks and libraries, the solution should be *open source* and licensed with an appropriate license (MIT for instance). The solution should be scalable, namely able to store at least *million of password* entries, spread across all users. The *encryption used for storing* the passwords should be of industry standard, and should be viable for at least 5 years. The same goes for the *encryption used for communication*. For maximum security, the solution should only accept and use *TLS version 1.2* connections, with a limited cipher suite.

Finally, for the best user experience, all the interaction with the user interface should be realized with a *latency never exceeding 500ms*. Any longer, and the user will grow tired of using the software, because of its sluggish feel.

3 Tools

In this section we briefly introduce 14 different available password manager tools, detailing the most relevant features and postponing in Sect. 4 a more thorough analysis. We considered only real systems already usable to final users, as listed in Table 1. In the last part of this section, we also report a concise survey of proposals coming from the literature and not available as usable tools.

Table 1. Password managers considered in the analysis

1. In-Browser built-in	6. Zoho Vault	11. SimpleVault
2. LastPass (and similar)	7. TeamPasswordManager	12. RoboForm
3. KeePass (and similar)	8. Passwordstate	13. Vaultier
4. Rattic	9. Simple Safe	14. TeamPass
5. Encryptr	10. PassWork	

In the following sections we briefly describe each of the considered solutions, with also a critical eye towards the user experience and the usability: if the solution is not user friendly, the users will not use it and then it is effectively worthless.

1. In-Browser Password Managers. The most used password managers are probably the ones built-in into the various browsers. This is a feature most major browsers have adopted: Chrome, FireFox, Edge (new name of Internet Explorer), Safari and Opera. Almost all of the most recent versions of the mentioned browsers can sync their passwords between different devices, but this requires to upload the passwords to one of the corporations' Web sites. Additionally, built-in password managers have one big limitation: they only work within web sites accessed through that *specific* type of browser, i.e. only in Chrome browser. Passwords for other applications (like email clients, development suites and so on) cannot be easily retrieved.

In [27] it is presented an analysis of the storage formats for the different browsers' password managers. While their results are for probably outdated versions (for example the analyzed version of Chrome was *v.21.0*, while at the time of writing, the current newest version is *v.47.0*), their primary concern is the encryption methods used by the web browsers to store the passwords. At the time of their analysis, only Firefox and Opera were supporting a master password to enable the access to the stored passwords.

2. LastPass, and Similar Solutions. LastPass², PassPack³, DashLane⁴, and many others are smartphone apps coupled with plug-in browser enabling the user to access the passwords from several devices. We refer only to LastPass as a representative of this group, it being the most well-known.

LastPass uses 256-bit AES encryption for the communications and applies PBKDF2, as the hashing function, in order to make it difficult to crack stored data. Both encryption and decryption are performed client side [10], as to avoid transferring the actual password, unencrypted, to their servers. Encryption and decryption are done using the master password, which is never actually sent to their servers. Finally, as is to be expected, all connections to LastPass' servers, are TLS 1.2 encrypted.

Regarding the usability, LastPass allows the user to organize passwords in folders, creating the tree-like structure. For devices without a browser supporting plug-ins, LastPass offers a so-called bookmarklet [9]. A bookmarklet is a bookmark, which essentially contains JavaScript code, in order to add previously unobtainable features, in a browser. While this on the surface seems like a nifty feature, work in [12] discusses an attack on LastPass, exploiting the users bookmarklet, to gain access to virtually all of the users stored credentials. Finally, it is work mentioning that there has been a recent leak from LastPass [25], that leads to even more users to look suspicious of their services.

3. KeePass, and Similar Solutions. KeePass⁵ gained fame after the LastPass data breach. Differently to this latter, KeePass allows the user to store the passwords in a local file. While there exists a plethora of tools similar to KeePass, it will be used as a representative of this group.

Version 2.x of KeePass uses AES-256 encryption, but it can also apply additional algorithms through plug-ins [8]. This enables users to tailor the encryption security, to their own requirements. KeePass features a tree-like structure, in order to completely organize passwords and also has a fully customizable password generator, where the user can also choose the character sets.

KeePass lacks of usability, since it does not support password distribution. Since KeePass works on a local file, it would only inherently work on a *single* device.

² <https://lastpass.com/>.

³ <https://www.passpack.com/>.

⁴ <https://www.dashlane.com/>.

⁵ <https://keepass.org>.

Should one wish to distribute it, another tool has to be involved to save the file in the cloud. Additionally, there is the lack of cross-platform compatibility, since KeePass only supports Windows.

4. Rattic. Rattic [7] is a self-hosted password manager, in the so-called private cloud. Rattic can be considered a password management database, with a special focus on managing passwords for a team [7]. Since Rattic *is* meant for teams it has multi-user support and makes the distinction between admin and regular users. It organizes passwords and users in groups, for easy access control, where a group is a collection of users which can access the same passwords. Additionally it supports tags for their passwords, allowing for even further organization, for their users allowing quick access to similar passwords, from across different groups. However, the fact that Rattic is team-oriented, the user cannot simply create “private” passwords, but it needs to manually create a group with a single user. Rattic also provides a password generator and makes possible to download passwords in the KeePass format, making it available for later offline use.

Regarding the technical aspects, Rattic surprisingly does *not* encrypt passwords stored in the database and highly recommends storing the database on an encrypted drive, to ensure database protection. Clearly, this does mean that a system administrator can access *all* passwords, should he or she have the encryption key for the drive. As a positive note, Rattic is developed in Python, using the Django framework and tested on the Apache server.

5. Encryptr. Bordering between the type of LastPass and Rattic, Encryptr [3] relies on the Crypton [6] backend [4]. Crypton is an application framework and backend service to develop applications, providing the required primitives for cryptography. Encryptr can host the passwords on a third party cloud service (namely SpiderOak⁶), but makes also possible to run a dedicated Crypton backend, like a the private cloud. However, this requires a high level of technical skills, including editing source files [5], apply patches, compile and fine setting. This severely affects the usability of the solution. Moreover, the user interface is *very* minimalist and sleek, while passwords are stored in one unique, single list.

Despite its complexity, the Crypton backend stands for its zero-knowledge security [19]: according to the authors, it is impossible to obtain the unencrypted data on their servers, without actually getting hold of the users’ private encryption keys. The Crypton backend is open source and uses AES-256.

6. Zoho Vault. Zoho Vault [15] relies on the storage within proprietary cloud and aims at enterprise customers, providing interesting features, such as LDAP integration. Vault organizes passwords in so called “chambers” and each password can be added to one or more chambers. While this approach sounds a valid alternative to the classic tree-style organization, it does not add any real benefit

⁶ <http://spideroak.com>

to the final user. Zoho Vault uses a combination of RSA and AES. To enable sharing, a common AES encryption key is retrieved using RSA keypairs [16].

7. Team Password Manager. Team Password Manager [26] is a tool that is fairly similar to Rattic, being enterprise-oriented and organizing passwords in a tree-like hierarchy of “projects” and sub-projects. It can be privately-hosted, and is platform independent, requiring Apache, PHP and MySQL.

Team Password Manager uses AES-256 and bcrypt as its key derivation function, to slow down brute force attacks against password hashes. It also supports Google’s Authenticator, for two-factor authentication. Being enterprise-oriented, it is very complex and can keep away low-security-wise regular users.

8. Passwordstate. Passwordstate from ClickStudios [17] is another enterprise-oriented solution. It has Active Directory support, and other built-in options, as High Availability, a fully customizable password generator and several options for two-factor authentication. Passwordstate is self-hostable, but requires a Windows platform and the IIS server (it is not platform agnostic). Passwordstate has a very intuitive UI and it is able to give the most important information, the most screen-space. The tree-like structure that creates an excellent organizational options for the users. However, its enterprise-oriented engine presents the user with many technical options, reducing its usability for a regular user.

Passwordstate encrypts all passwords using AES-256 [18] and protects the backend services (web service and database) employing code obfuscation of the pages and encryption of the data.

9. Simple Safe. Simple Safe [14] is another self-hosted team password manager solution. The user can choose to organize passwords into “groups”, that are accessed through a menu, which is auto hidden. Unfortunately, changing between groups takes a long time, in the order of seconds. This causes a bad experience for the user: while generally the developers of Simple Safe have been very generous with the animations, this causes a heavy latency that adds to the overall sluggish feel of the system. Considering the possibility to organize the passwords, it is worth citing as the user can customize the fields in the different groups. For instance, they allow the user to store SSH keys, in dedicated field type, making possible to separate passwords that require keys/certs, from traditional username/password logins.

The technical details about Simple Safe are very few. The only available information regarding security or encryption is the following quote: “*Simple Safe utilizes a 256 bit encryption method. Each password has a unique private salt along with a master salt stored separately to the database. Only encrypted passwords are stored within the database.*” [13].

10. PassWork. PassWork [21] is yet another enterprise-oriented solution, available as both a remote and a self-hosted, both of which comes with a price tag. It

organizes passwords in “groups” and sub-groups with an associate list of users currently allowed to access that passwords. Interestingly, it offers the option of setting permissions to users, namely when adding a user it is possible to give different permissions like “Full Access”, “Edit”, “Read” and so on.

Being a closed source solution, PassWork’s developers do not describe and document their solution adequately. Something as simple as figuring out which platform it supports, prior to purchasing is not possible. Additionally, they simply state of using “256-bit passwords” and RSA keypairs for sharing passwords.

11. SimpleVault. SimpleVault [2] encrypts each individual password with a different “passphrase”. As the name suggests, SimpleVault is much more a proof of concept, with an extremely minimalist interface: adding a password is done in a page having two fields completely not described. It provides a powerful password generator, with increasingly “rare” symbols. Retrieving a password, requires the user to type in the passphrase. The choice of this per-password passphrase, will undoubtedly only result in the user using the same password over and over again. However, this choice *does* allow for the system to be used by multiple users, each just using their own master passphrase for all of their passwords.

The security of SimpleVault is again described with few details. The authors state that at some point or other, the encrypted data are eventually stored in clear text on the remote machine. Moreover, they state that when adding a new entry, the password is sent without encryption to the server, which in turn encrypts it, only relying on the HTTPS. Additionally, SimpleVault does *not* use a database system, but stores all contents in a `.txt` file.

12. RoboForm. RoboForm [24] is somewhat a hybrid between KeePass and LastPass: it locally stores the password and is available to the user as a web browser extension. Additionally, it offers “RoboForm Everywhere” cloud synchronization of the encrypted file. This does however, entail trusting them with safe-keeping the encrypted file. Using RoboForm on a single device, renders it as a tool very similar to KeePass, albeit with “better” browser integration.

13. Vaultier. Vaultier [22] is self-hosted, has a imperceptible latency for any UI actions, and has a clean design of the front page. Passwords are organized in vaults, where cards are placed and, inside of these cards are the passwords. However, the UI showing inside a single vault, only shows six cards at a time. Similarly, at most two passwords can be displayed on screen, at a time. It also has a password generator, with limited customizable options.

Vaultier is open source, since it has a the git repository’s history publicly shown. However, it seems like the project is in a idle state, since the last update was done on the 17th of April, 2015 [23].

14. TeamPass. TeamPass [11] ensures platform agnosticism due to being implemented in PHP, and only requiring an Apache server and MySQL server. It features a series of APIs to permit an access to TeamPass Items from a third party application. However, it only relies on HTTPS to transmit the passwords that are, indeed, sent using the GET HTTP method. Additionally, based on the description of the APIs, it appears clear as the password is decrypted server-side.

4 Analysis

While the previous section provided a high level description of the considered solutions, here we detail how each of them deal with the requirements introduced in Sect. 2, providing a unified picture. The analysis is shown in Fig. 1.

In order to better investigate some of the requirements, we provide a closer look to some of the most desirable functional requirements. Figure 2 distinguishes between solutions that are able to be run in the private cloud, and those who are not. Focusing only on those which actually can be hosted in a private cloud, Fig. 3 shows their default organization with respect to the passwords. While all of the systems support multiple users, they all do not consider personal passwords.

This is primarily a result of nearly all of these being marketed towards enterprises and team-oriented development. However, for almost all of the solutions, it is possible to solve this shortcoming, creating a new personal group for each individual user, in which personal passwords can be stored. Unfortunately, this must be considered as a workaround, with the only exception of PassWork, which creates this group automatically.

Looking at the more technical aspects of the assessment, in Fig. 4 we report the agnosticism of the compared solutions. Most of the password manager service are able to run on multiple platforms, being implemented in PHP. Hence, they only require an Apache (or nginx) Web server to run. Fortunately, both of these have versions for all the major operating systems. Considering the database agnosticism, instead, there is only Rattic that actually offers different options. However, even in this case, the developers warn that different options are done at the user's risk, since all the alternatives have to be considered experimental.

As our analysis shows, *no* tool covers all of the requirements. Although some tools might come close, they still lack too much to be considered a viable option.

5 Conclusion

Cloud password managers are becoming increasingly popular, helping users to protect their services with stronger passwords, relieving them from the extra effort required. The number of possible professional alternatives is high, as this paper shows, but also a perfect solution is still missing. We have considered 17 functional requirements and 7 non-functional requirements as the desirable features a cloud password manager should possess. Consequently, we have surveyed 14 professional password manager solutions and critically analyzed against the considered requirements. We focused on professional tools only, as we aimed

at getting a clear picture of available implemented cloud password managers. What emerges is that a silver bullet application is still lacking, as no professional password manager fully satisfies the set of functional and non-functional requirements. Consequently, research work has still to be done to propose a

	In-Browser Managers	LastPass, and Similar	KeePass, and Similar	Rattic	Encryptr	Passwordstate	Vault (Zoho)	TeamPasswordManager	Simple Safe	PassWork	SimpleVault	RoboForm	TeamPass	Vaultier	Desirable solution
Functional															
F-Req #1	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
F-Req #2	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F-Req #3	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
F-Req #4	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
F-Req #5	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
F-Req #6				✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
F-Req #7	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
F-Req #8			✗	✗	✗	✗	✗	✗	✗		✓	✓	✗	✗	✓
F-Req #9	✓	✓	✓	✗							✗	✗	✗		✓
F-Req #10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F-Req #11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F-Req #12	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F-Req #13	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓
F-Req #14	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
F-Req #15	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F-Req #16	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
F-Req #17				✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Non-Functional															
NF-Req #1	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓
NF-Req #2	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓
NF-Req #3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NF-Req #4	✓	✓	✓	✗	✓	✓	✓						✓	✓	✓
NF-Req #5	✓	✓		✓	✓	✓	✓	✓	✓	✓					✓
NF-Req #6	✗	✗		✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
NF-Req #7	✓	✓	✓	✓	✓		✓	✓	✗	✓	✓	✓	✓	✓	✓
<div>✓ Yes.</div> <div>✓ Yes, with a remark. Only true using a workaround, possibly going against other requirements, or it works less than ideally</div> <div>✗ No.</div> <div></div> Not applicable to the solution. <div></div> No information is available, to indicate an answer.															
Functional requirements (F-Req)															
1. Distributed password database	10. Permission for new passwords														
2. Multi-user support	11. Permission for reading passwords														
3. Support admin and regular users	12. Permission for deleting passwords														
4. Hierarchical password organization	13. Extensive auditing														
5. Password sharing	14. Single master password														
6. Admin user management	15. Single master password modification														
7. Platform agnostic	16. Support two-factor authentication														
8. Database agnostic	17. Auto-start after reboot														
9. Encrypted storage															
Non-functional requirements (NF-Req)															
1. User-controlled storage	5. Strong communication encryption														
2. Open Source License	6. Only use TLS 1.2 or higher														
3. At least 1 million passwords	7. Latency below 500ms														
4. Strong storage encryption															

Fig. 1. Analysis of requirements and available professional password manager

Name	Web Based	Self-Hostable
LastPass	Yes	No
KeePass	No	N/A
Rattic	Yes	Yes
Encryptr	No	Yes ^a
Passwordstate	Yes	Yes
Vault (<i>Zoho</i>)	Yes	No
TeamPasswordManager	Yes	Yes
Simple Safe	Yes	Yes
Passwork	Yes	Yes
SimpleVault	Yes	Yes
RoboForm	No	N/A
TeamPass	Yes	Yes
Vaultier	Yes	Yes

^aRequires editing source files and running own Crypton backend.

Fig. 2. Comparison of access method and ownership model, of the available solutions.

Name	Supports Multiple Users	Personal Passwords	Password Sharing
Rattic	Yes	Yes ^a	Yes
Passwordstate	Yes	Yes ^a	Yes
TeamPasswordManager	Yes	Yes ^a	Yes
Simple Safe	Yes	Yes ^a	Yes
Passwork	Yes	Yes	Yes
SimpleVault	Yes ^b	Yes ^b	Yes ^c
TeamPass	Yes	Yes ^a	Yes
Vaultier	Yes	Yes ^a	Yes

^aThrough password grouping.

^bBy each user, using their own unique passphrase for encrypting their personal passwords.

^cThrough a commonly used master passphrase, acting as a workaround.

Fig. 3. Comparison of password ownership in self-hostable solutions.

Name	Platform Agnostic	Database Agnostic
Rattic	Yes	Yes ^a
Passwordstate	No	No
TeamPasswordManager	Yes	No
Simple Safe	Yes	No
Passwork	N/A ^b	N/A ^b
SimpleVault	Yes	No ^c
TeamPass	Yes	No
Vaultier	Yes	No

^aDatabases other than MySQL receives less testing.

^bNo information available to determine.

^cDoesn't use a database. Stores passwords in a .txt file.

Fig. 4. Comparison of agnosticism of self-hostable solutions.

solution that is secure, effective and efficient at the same time. Thus, we call the security community for a higher effort to face the password management in the cloud, providing adequate protection to users' passwords.

References

1. Amico, M.D., Michiardi, P., Roudier, Y.: Password strength: an empirical analysis. In: 2010 Proceedings of INFOCOM, pp. 1–9. IEEE, March 2010
2. Brugger, R.: Simplevault - password manager. <http://simplevault.sourceforge.net/>. Accessed 09 Dec 2016
3. Crypton: Encryptr - powered by crypton. <https://crypton.io>. Accessed 05 Dec 2016
4. Devgeeks: About encryptr - powered by crypton. <https://encryptr.org/#about>. Accessed 05 Dec 2016
5. Devgeeks: Add server software for self-hosting. issue #156 spideroak/encryptr. <https://github.com/SpiderOak/Encryptr/issues/156>. Accessed 05 Dec 2016
6. Devgeeks: Crypton - build private applications. <https://encryptr.org>. Accessed 05 Dec 2016
7. Hall, D.: RatticDB. <http://www.ratticdb.org>. Accessed 05 Dec 2016
8. KeePass: Keepass. <http://keepass.info/help/base/security.html>. Accessed 05 Dec 2016
9. LastPass: Bookmarklets — user manual. <https://helpdesk.lastpass.com/bookmarklets/>. Accessed 10 Dec 2016
10. LastPass: How it works — lastpass. <https://lastpass.com/how-it-works/>. Accessed 10 Dec 2016
11. Laumail, N.: Teampass. <http://teampass.net/>. Accessed 07 Jan 2016
12. Li, Z., He, W., Akhawe, D., Song, D.: The emperors new password manager: security analysis of web-based password managers. Technical report, University of California, Berkely (2014)
13. Ltd., H.P.: Frequently asked questions. <https://www.simplesafe.net/faqs/>. Accessed 08 Dec 2016
14. Ltd., H.P.: Team password management — simplesafe - self-hosted password sharing. <https://www.simplesafe.net/>. Accessed 08 Dec 2016
15. Ltd., Z.C.P.: Password manager, password management software - Zoho vault. <https://www.zoho.com/vault/>. Accessed 05 Dec 2016
16. Ltd., Z.C.P.: Secure sharing of secrets. <https://www.zoho.com/vault/secure-sharing-of-secrets.html>. Accessed 05 Dec 2016
17. Passwordstate: Enterprise password management with passwordstate. <http://www.clickstudios.com.au/>. Accessed 08 Dec 2016
18. Passwordstate: Secure code, secure data. <http://www.clickstudios.com.au/about/secure-code-data.html>. Accessed 08 Dec 2016
19. Pedersen, C., Dahl, D.: Crypton: Zero-knowledge application framework. Technical report, SpiderOak. <https://crypton.io/crypton.pdf>. Accessed 01 Sep 2016
20. Pressman, R.: Software Engineering: A Practitioner's Approach. McGraw-Hill higher education, New York (2005)
21. Primepix, Konfeta: Team password manager. collaboration and password sharing, API, hostable version. <https://passwork.me/>. Accessed 05 Dec 2015
22. RightClick: Collaborative password manager & file storage. <https://www.vaultier.org>. Accessed 17 Dec 2015
23. RightClick: rclick / vaultier — gitlab. <http://git.rclick.cz/rclick/vaultier>. Accessed 17 Dec 2015
24. Siber Systems, I.: World's best password manager. <http://www.roboform.com/>. Accessed 05 Dec 2015
25. Siegrist, J.: Lastpass security notice. <https://blog.lastpass.com/2015/06/lastpass-security-notice.html/>. Accessed 05 Dec 2015

26. TeamPasswordManager: Secure sharing of secrets. <http://teampasswordmanager.com/>. Accessed 05 Dec 2015
27. Zhao, R., Yue, C.: All your browser-saved passwords could belong to us: a security analysis and a cloud-based new design. In: Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy (CODASPY 2013). ACM (2013)

Current Trends in Web Engineering

ICWE 2016 International Workshops, DUI, TELERISE,
SoWeMine, and Liquid Web, Lugano, Switzerland, June
6-9, 2016. Revised Selected Papers

Casteleyn, S.; Dolog, P.; Pautasso, C. (Eds.)

2016, XX, 209 p. 68 illus., Softcover

ISBN: 978-3-319-46962-1