1-1-2017

Analysis on the Security and Use of Password Managers

Carlos Luevanos
Willamette University

John Elizarraras
North Star Charter High School

Khai Hirschi
Capital High School

Jyh-Haw Yeh
Boise State University

 doi: 10.1109/PDCAT.2017.00013

*Analysis on the Security and Use of Password Managers**

Carlos Luevanos[1], John Elizarraras[2], Khai Hirschi[3], and Jyh-haw Yeh[4]

[1]Dept. of Computer Science, Willamette University, CA
[2]North Star Charter High School, Eagle, ID;
[3]Capital High School, Boise, ID
[4]Dept. of Computer Science, Boise State University, ID

***Abstract*— Cybersecurity has become one of the largest growing fields in computer science and the technology industry. Faulty security has cost the global economy immense losses. Oftentimes, the pitfall in such financial loss is due to the security of passwords. Companies and regular people alike do not do enough to enforce strict password guidelines like the NIST (National Institute of Standard Technology) recommends. When big security breaches happen, thousands to millions of passwords can be exposed and stored into files, meaning people are susceptible to dictionary and rainbow table attacks. Those are only two examples of attacks that are used to crack passwords. In this paper, we will be going over three open-source password managers, each chosen for their own uniqueness. Our results will conclude on the overall security of each password manager using a list of established attacks and development of new potential attacks on such software. Additionally, we will compare our research with the limited research already conducted on password managers. Finally, we will provide some general guidelines of how to develop a better and more secure password manager.**

***Index Terms*— Password Managers, Password Authentication**

I. INTRODUCTION

In this paper, we will be discussing three open-source password managers: Passbolt [24], Padlock [8], and Encryptr [6]. We have chosen these three due to each unique quality they carry; we will mention them here once and then bring them up again in their separate sections. Passbolt was chosen for its unique property in that its full potential and benefits are reached when utilized by teams, companies, and closed groups of people who trust each other with sensitive information; additionally, Passbolt runs on OpenPGP [7], a secure email encryption standard founded by Phil Zimmerman. Padlock was chosen for its use of the Electron and Polymer developer environment as well as being a minimalist password manager. Lastly, Encryptr was chosen for its brow-raising qualities; all data is stored in the cloud, and it uses Crypton, a cryptography framework that implements a no-knowledge proof system, a somewhat fancier way of saying end-to-end encryption. To conclude our paper, we will compare and contrast all the features between these password managers and more popular ones in order to have a standard of security for this type of software in general.

*This work was partially funded by the NSF REU Software Security Site and NASA ISGC High School Summer Research Experience grants.

[4]The corresponding author, jhyeh@boisestate.edu

The organization of this paper starts with the related works in Section II. Following that we give a brief rundown and history of password managers. We then discuss each password manager in their own sections; we will go over Passbolt, Encryptr, and Padlock in that order. Each section contains subsections discussing the details of the password managers, reported security flaws already found for them, our reported flaws, a review and critique about the password manager, and then some potential solutions to the vulnerabilities. The paper concludes with suggestions of what an ideal password manager should have; we review some features of more popular password managers and look at the pros and cons of both open-source and closed-source password managers.

II. RELATED WORK

The niche of research on password managers has direct ties to that of applied cryptography and penetration testing, so the body of our work will go back to referencing a mix of academic papers on the security of more popular password managers and penetration testing write ups performed by security auditing teams, although we will reference some papers that display proof of concept or have contributed to the field in some other way. Additionally, we have tried our best to extend any and all audits performed on our three researched password managers. The earliest work for password managers comes from Luo and Henry in 2003 [20], who demonstrated a proof of concept and implementation of a more effective password manager, compared to Microsoft Passport. In 2005, Halderman et al's work [19] comprised the proof of concept and implementation of a password manager in the web-browser, where an example implementation in Firefox was given in their work.

Moving on to the trend of security analysis, we are given insight on popular password managers such as LastPass and Roboform by Li, He, Song, and Akhawe [18]. Silver, Jana, Chen, Boneh, and Jackson [3] made outstanding contributions to the auto-fill feature found in popular password managers such as LastPass, KeePass, and those implemented in web-browsers such as Google Chrome and Safari. They found critical vulnerabilities that abused the auto-fill feature; such attacks include iFrame sweep attacks, password sync exploitation, and injections. Their work would help greatly influence the policies auto-fill executes. To show some of the root problems as to why password managers are seeming to

become more and more necessary, the work of Gaw and Felten [16] would contribute statistical analysis of surveys performed at Princeton University. Gaw and Felten found that participants often reused passwords for less important websites and predict this trend would grow as more on-line accounts accumulate. Participants were found to be ignorant to the security risks that this trend brings. They also found a feeling of indifference towards the use of password managers. Looking at the rising trend of cloud computing, the work of Zhao, Yue, and Sun [14] contributed to vulnerability analysis of LastPass and Roboform; they were able to detect threats such as credentials being stored in plain-text on cloud servers and offered suggestions to both product makers on how to better secure their data and product.

To conclude the related work we mention Gasti and Rasmussen [13]; their contributions include a forefront on the analysis of password manager database formats, as their paper's title suggests: "On The Security of Password Manager Database Format." What Gasti and Rasmussen found was that despite a number of password managers being different from each other, each pretty much used the same database format. They also found several vulnerabilities in each password manager they investigated.

III. OVERVIEW OF PASSWORD MANAGERS

A. Quick Rundown

First and foremost, we must define what a password manager is. Password managers are programs used to generate, encrypt, and store passwords for a client-side user. All that is required of a user is to remember one master password and user name. It is believed that using such software will increase security. Typically passwords will be stored on the local machine itself or on some hosted server. In some cases, they may be hosted on cloud servers to ensure more security for the parties involved; i.e., the user and the host of the servers/proprietors of a password manager. There is some variety in the types of password managers available to the public; some are built into web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge, while others serve as standalone programs with the capability of web-browser integration. Some strictly enforce strong master passwords while others do not. More notably there are a few that have integrated multi-factor authentication, which is very beneficial to security. Given a brief rundown of such software, let us now delve into some history.

B. History of Password Managers

Not much is known about password managers; the first successful implementation of an effective password manager open to the public (to our knowledge), after the work of Luo and Henry [20], was KeePass, developed by Dominik Reichl [21], with an initial launch back in 2003. KeePass is a minimalist password manager that runs on Windows, Mac OS, and Linux, along with unofficial imports to Android, IoS, and Blackberry. The latest versions of KeePass use AES-256 bit encryption [40] along with ChaCha20-256 bit encryption [41]; however, KeePass does allow you to use other algorithms if you wish to do so, but we shall not go over technical specifications. Following a mixed response from the launch of KeePass, it would be a few years until commercial success for password managers took off; LastPass, Dashlane, and Roboform serve as prime examples, with the LastPass being the most popular, having a reported 7 million users as of 2015 [22]. By the 2010's it seems as if some kind of password manager mania has taken off. Many developers have begun focusing their time on password managers, and to increase popularity they made their work open-source, giving the public a chance to use their product and view their code, giving the technical community a chance to shape a password manager to their liking by giving feedback on what can be added onto or improved, or by exposing vulnerabilities [23]. This method seems to be useful in the fact that it is essentially a free security audit. Of course, the time it would take to report to developers would be much longer.

IV. OPEN-SOURCE PASSWORD MANAGERS

Now focusing our attention to the bulk of our work, let us recall our three password managers: Passbolt, Padlock, and Encryptr. Each of these password managers is open-source. We chose to look at open-source password managers for multiple reasons. First of all, because they are open-source, we can look at the source code ourselves. Additionally, this also allows us to set up our own servers to test on. Each password manager was chosen for their own unique properties. We shall go into technical detail about each one, along with pointing out vulnerabilities we have found in each one and referencing previous vulnerabilities exposed by others.

A. PASSBOLT

Passbolt is an open-source password manager initially developed by Kevin Muller, Diego Lendoiro, Remy Bertot, and Cedric Alfonsi, with later work of Passbolt being supported by the GitHub community. Passbolt's core user-base includes development teams and companies, adopting the philosophy that company password policies can be shoddy or annoying, which in turn creates a less efficient workspace and perhaps some security vulnerabilities. The Passbolt development team believes their product can be adopted to ease the process of sharing passwords among peers and coworkers in an easier and much more secure way [24].

1) Overview of Passbolt: Currently, Passbolt only runs in a browser. More specifically, it only runs on Firefox and Google Chrome. This is reportedly due to them still being in the alpha development of Passbolt [25]. Passbolt was written in JavaScript, PHP, and Shell, and it currently uses OpenPGP for their encryption standard. Passwords stored in Passbolt are encrypted, and the database used by the client can also be encrypted to improve security. However, user names are not

encrypted and are stored in plain text. One existing problem for Passbolt includes the use of a bad pseudo-random number generator [25]. There is also no current way to change your master password or use multi-factor authentication. There is the option of emailing a copied list of your encrypted passwords to yourself should the option for email notification be enabled. Passbolt also boasts the use of a color security token which should prevent phishing, however, we believe we have found a way to bypass this feature which we will go into detail in our discovered flaws section. Another current flaw reported by the Passbolt team is the predicament of the client and server trusting all keys; although they admit to this being a flaw, they wish to fix this error in the future [25].

2) Reported Security Flaws: Found early in the surveillance of Passbolt, it was discovered by Wigginton et al that the use of the PHPseclib has the potential to default to the use of ECB encryption [4]. While this isn't a direct flaw in Passbolt itself, it is still a flaw to consider. Reported problems by Passbolt include server integrity problems, DDoS attacks, server information leaks, key revocation, the potential of authentication cookies being stolen if SSL is broken, and the potential to mimic server keys [26]. It should be noted that Passbolt currently only uses MySQL servers, which have had reported problems that were recently fixed by Golunski [2]. Given that Passbolt is only in alpha and developed by a single team rather than a company, they do not have the resources to perform a full-stretched security audit; most vulnerabilities have been found by the developers themselves or by the GitHub community. The team's use of cryptographic functions (by use of OpenPGP) has been reviewed by security audit team Cure53 [27]. The team was able to find several vulnerabilities in the OpenPGP library but we shall omit the details.

3) Our Discovered Flaws: To begin testing on Passbolt we opted to not perform any attacks already conducted, so we tried some of our own attacks. Before testing, we set up a private server using the Hamachi Virtual Private Network so that we could as closely as possible simulate a work environment typical to what Passbolt should be used for. The tools involved were all on Kali Linux, with the exception of some attacks written in C# and tested on Windows machines. It was very easy to see that a key-logger on an unsuspecting user would give us the master password. Passbolt was also found to be susceptible to a clipboard attack; given the generated passwords can be somewhat hard to read, a user will opt to just copy and paste rather than manually type their passwords, making strong individual passwords to websites and services seem useless if an attacker is successful.

Passbolt has a special feature to prevent phishing attacks; a user will remember a color assigned to them and it will be present when they attempt to log-in. We conducted an attack on their use of this color key token; by grabbing a live copy of a user's session we were able to find the lines of code that gave us the color. Simply editing these lines we now had an exact copy of what a user would think is the login page, all working as if it were the real thing. With this, an attacker could potentially perform a phishing attack in which they can steal a user's master password. See Figure 1 and Figure 2 for more detail.

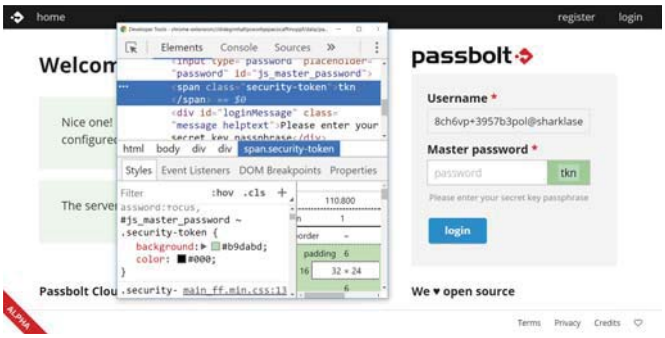

Fig. 1: Here we can see the hex value of the color a user's security token

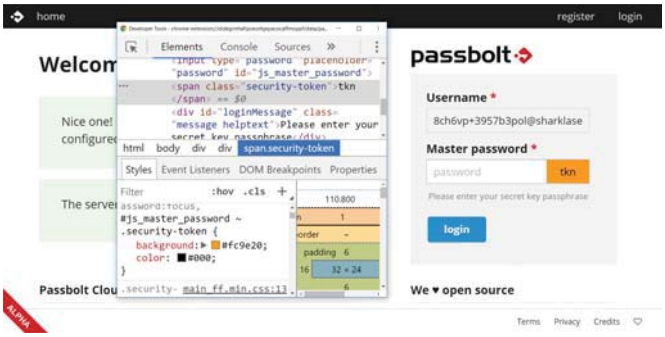

Fig. 2: By editing this hex value, we can replace the color with any color of our choice

One other attack we created was a custom user-script. A user-script is a custom script that users can install to gain extra functions to websites. Users will typically not look over all of the code in the user-script, so it would be easy to hide two lines of malicious code in a script that looks innocent. We created a script (see Appendix) that would secretly replace all links that download the Firefox Passbolt extension with another random extension. All the links appear to go to the original website, as seen in Figure 3. Because of the way Firefox installs extensions, the user will get a pop-up that says the website itself wants the user to download the fake extension, making the extension seem trusted (See Figures 4 and 5). The Passbolt extension itself can't be modified, however, as Passbolt detects any changes and disables itself if it finds any. Downloading a fake extension would bypass this, and since the extension is open-source, mimicking the extension would be very simple. A proposed method to get the user to download this user-script is to hide the code inside a good user-script. As mentioned previously, the code would most likely go unchecked. Another method would be to hide it in an extension. The extension would then inject the JavaScript into the website.

4) Review and Critique: Overall, Passbolt was not a very user-friendly password manager and we question the integrity of the product. They have no reported security audits on their

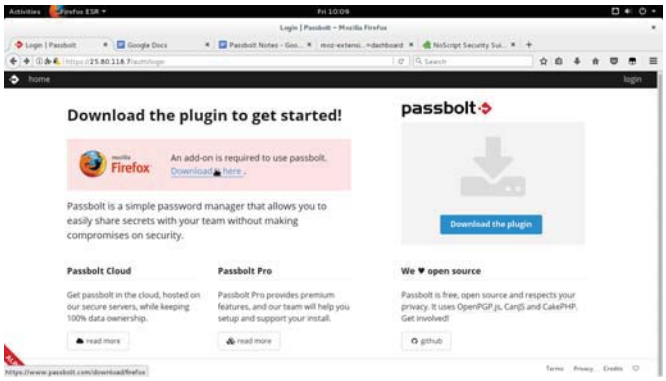

Fig. 3: Notice how the link to download the extension appears to go to passbolt.com

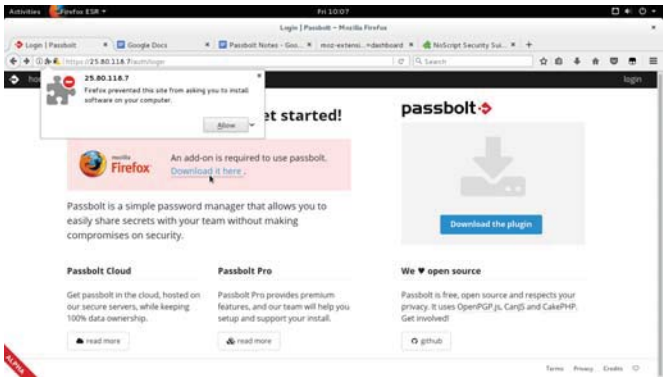

Fig. 4: Firefox shows the server itself asking you to download the extension

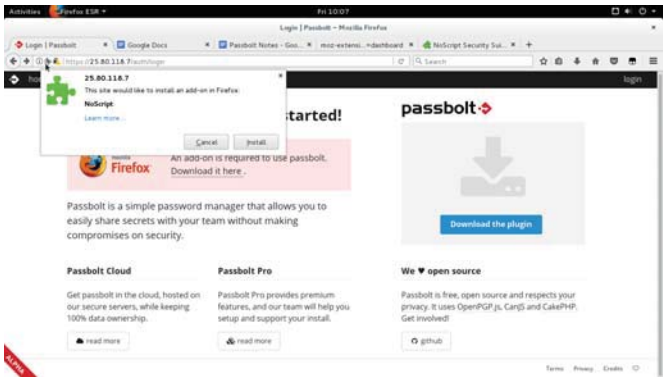

Fig. 5: The link actually downloads an arbitrary extension called NoScript

product except a reference to the audit of OpenPGP, and they lack many key features other password managers have, which brings into question why they would even release an alpha version of their product. Additionally, their demo page is quite shoddy and the intended use of the password manager seems to often lead attackers to use cunning phishing attacks and DDoS attacks. The design also increases the risk of an attack on the main administrator, since they are at the core of how Passbolt should be used by a company. It should also be noted there is no enforcement of a strong master password, which is in itself a big security risk.

5) Solutions to Some Vulnerabilities: We suggest that the Passbolt team develop some features such as typing obfuscation and auto-fill to protect against key-logger and clipboard attacks. We also encourage the use of other types of servers besides MySQL. While not all servers are perfect, giving user options can potentially increase the overall security of the company using Passbolt. The user-script vulnerability is something that would be hard for Passbolt to stop, but they should at least use an HTML content security policy that would block scripts loaded from an off-line source, most of the time [12]. We would also recommend some way to either obfuscate the security token or develop some other method in order to prevent our proposed phishing attack.

B. ENCRYPTR

Encryptr is an open-source password manager initially developed by Tommy Williams and then bought out by SpiderOak [28], a company focused on building services that feature no-knowledge frameworks.

1) Overview of Encryptr: Encryptr is a cross-platform password manager, e-wallet, and note-holder written in JavaScript, HTML, CSS, JSON, and XML. Its encryption standard was built using the Crypton framework, created by SpiderOak. Crypton is an open-source framework developed in JavaScript with a primary goal to store information on a server without the server ever knowing what is stored [1]. Crypton's back-end uses PostgreSQL [36], Redis [37], Node.js [38], and Docker [39]. Encryption and decryption are assumed in AES-256 using Galois/Counter Mode. For more specifics, ElGamal encryption [35] and ECDSA (Elliptic Curve Digital Signature Algorithm) [34] are used for signature verification, elliptic curve cryptography is used for key generation, and such ciphers can be switched for others if a user decides to do so [1]. The strength of Crypton is the protection of user data and data sharing, it is the direct belief of end-to-end encryption that users may feel more secure from attackers and the company hosting such a service. Crypton also uses SRP (Secure Remote Password) authentication, which reportedly limits data compromise, with the only supposed attack that being brute forcing AES keys [1]. Some weaknesses of Crypton include the ability of peer graph analysis and container access frequency analysis. What we mean by the former statement is that user-names are stored in plain text, so it is possible for database records to be analyzed in order to find connections between users and perform some intelligence gathering, leading to more potential attack vectors [1]. As for the latter, containers in Crypton can sometimes be created, updated, and accessed deterministically, which can lead to a potential brute force attack. However, this can be remedied using a strong password [1]. One last note about

Crypton and other SpiderOak applications is the use of the clipboard, which is reported by them to be safer than just typing all your information.

2) Reported Security Flaws: It should be noted in this section that no official security audit of Encryptr has been performed. Furthermore, to our knowledge no security audit has been performed on other SpiderOak services. Only two official security audits were performed and published on Crypton. The main issues reported by Leviathan Security Group include [10]:

1) An account's public key is not verified against the decrypted private key. This could result in a user encrypting something that cannot be decrypted.
2) The public signing key is not verified against the private signing key.
3) A container by the name of containerNameHMacKey is not verified before decryption, so the server could replace it with a different known container and encrypt a new symmetric key to the user's public key.

A look into the report by Least Authority gave us some more insight into the security of Crypton which includes [11]:

1) Server information forgery: attackers with access to the server can overwrite and forge data on a user's account.
2) Guessable private keys: an attacker with server access can grab copies of cipher-text and read the plain-text.
3) An attacker with server access can disclose the encryption key, essentially making all container contents available to them.

Having read both reports we noticed plenty of the attacks included DDoSing. While not the largest security threat to worry about, it can still cost companies quite a bit of money and reputation. The reports failed to include auditing web-based attacks such as XSS attacks (Cross-Site Scripting), CSRF attacks (Cross-Site Request Forgery), Man-In-the-Browser attacks, and SQL injections.

3) Our Discovered Flaws: Our reported findings of vulnerabilities for Encryptr include:

1) High security threat with clipboard attacks (copying is the easiest way to transfer passwords from Encryptr to a form).
2) Susceptible to key-loggers when typing your master password or when transferring passwords from Encryptr to a form without copy-paste.

4) Review and Critique: Encryptr was the most minimal of the three open-source password managers reviewed by far; it was incredibly simple, it could be used on almost all platforms and did not require the use of an email, yet you could still retrieve the same data from other devices. One critique of Encryptr is that after some further investigation it was discovered some code is still obfuscated. It is also noted that there is no strict enforcement of strong passwords and generated passwords have a default length of 12 characters.

5) Solutions to Some Vulnerabilities: Like for Passbolt, we recommend SpiderOak implement an auto-fill feature for passwords and credit-card information, along with typing obfuscation should user's create their own passwords instead of generating one.

C. PADLOCK

Padlock is a minimalist, open-source password manager developed by Martin Kleinschrodt using the Electron and Polymer frameworks, so like Passbolt and Encryptr, it was all written in JavaScript, HTML, and CSS, and for the most part, all code is available to the public. Padlock is multi-platform and can be used on Windows, MacOS, Android, iOS, and some time in the future, Linux.

1) Overview of Padlock: Similar to Passbolt and Encryptr, Padlock uses a copy/paste function to quicken the process of using one's passwords. As noted earlier, this leads to very big security problems. One of the notable features of Padlock is that the application automatically logs you out of your vault in one minute if there is no user activity detected. This feature can be changed to a maximum of ten minutes or can even be disabled if the user wishes to do so. Padlock does include its own password generator, however, its weakness is that 7 character passwords with at least one uppercase letter, lowercase letter, and special character are considered very strong by Padlock. By the 2017 NIST standards, these generated passwords are not considered secure [29].

2) Reported Security Flaws: Surprisingly, the creator of Padlock has a repository for penetration testing of his own application. Furthermore, the penetration team Cure53 was hired to do even more extensive testing. The reports detail some of the following vulnerabilities:

- Tap-jacking [9].
- Exposed authentication tokens during API requests, leading to Man in the Middle attacks [9].
- Permanent DoS attack on mobile devices: an attacker with server access can increase the number of iterations, essentially making the CPU do a job it cannot do, and ultimately having to make the user reset Padlock if they want to use the application on their phone again. However, resetting Padlock will delete all information stored [9].
- DoS email attacks [9].

3) Our Discovered Flaws: Since Padlock uses the clipboard like Passbolt and Encryptr, along with no auto-fill feature, it was easy to discover that Padlock is susceptible to clipboard attacks and key-loggers. Additionally, we wrote a script (see Appendix) that would be able to reset a user's account, all that is required is the user click a button, thus deleting all their saved passwords and information. The attack involves a user installing a script or extension which would then inject JavaScript into the Padlock web page. For testing purposes, we did this using our own personal server, but we

believe it can be easily adapted to any other server using Padlock. We also used a user-script (defined in subsection IV-A.3) to manipulate a vulnerability in Padlock. Padlock has an on-line dashboard where users can change what devices have access to their account, as well as reset their data–with no password. To abuse this fault, we made a script that would reset the user's data the instant they logged into the dashboard. It should be noted that Padlock does try to prevent this. It uses content security policies that block off-line scripts [12]. This method will break certain user-script managers (extensions that install and inject the user-scripts) but some managers are able to bypass the security policy. This is something Padlock can't fix but they should ask for more verification, such as a password, to reset all of the user's data. It is just two clicks to clear all of the passwords and devices on an account. It should be noted that if you don't reset the client, your passwords are safe, but if the cloud was the only place where the passwords were stored, this method will make the passwords unrecoverable. Another script (see Appendix) we made would revoke all of the devices, meaning that the user would lose access to the cloud on their devices. This isn't a huge issue, as the user can just reconnect, but it demonstrates Padlock's vulnerability to script attacks.

4) Review and Critique: Padlock lived up to its name of being a minimalist password manager that got the job done and we were quite pleased with the initial security audits that were reported on the application. Overall it was easy to use and the ability to use a custom server was a nice addition. However, we did not like the minimum security standards of generated passwords, nor was there any strict enforcement of strong master passwords.

5) Solutions to Some Vulnerabilities: Similar to Passbolt and Encryptr, we suggest the creation of an auto-fill feature as well as typing obfuscation to prevent key-logger and clipboard attacks. Additionally, we suggest that the reset feature be removed, as loss of all passwords in one swoop can be easy; whether they be by someone with direct access to the local machine or by some social-engineering attack.

V. CONCLUSION: WHAT MAKES A PASSWORD MANAGER MORE SECURE

In this section, we will review the strengths and weaknesses of all the password managers we looked at in comparison with the strengths and weaknesses of more popular password managers in order to envision what a more secure password manager would look like.

A. Strengths of Open-Source Password Managers

Looking back at Passbolt, Encryptr, and Padlock, one of the greatest strengths they commonly share is the fact that they are open-source. This allows for consumers to examine the code and report any vulnerabilities and bugs themselves to quicken the refinement process. This method of putting trust into the consumer eases some of the burdens for developers and, of course, saves money; however, this does come at the cost that bugs and vulnerabilities may be found at a much slower pace. The option of users setting up their own servers is also a nice feature that can potentially increase security for users who know what they are doing, but this can be detrimental when attackers gather knowledge of targets using their own servers. It should be noted that the use of end-to-end encryption by Encryptr, i.e. Crypton, is a very desirable feature, especially for those who want the utmost privacy they can get. Such no-knowledge features were even commented on by Edward Snowden, who is pushing for end-to-end encryption to become a more standard feature in cloud storage [30].

B. Weaknesses of Open-Source Password Managers

Being open-source allows people to find vulnerabilities, but not everyone will report the security problems they find. An attacker can keep a vulnerability they found secret and use it in a future attack. Additionally, quite a few open-source password managers do not have many features that strengthen security like more well-known, closed-source password managers do. In order for an open-source product to be successful, it must have a strong group of supporters that review the code and make suggestions, and a development team that listens to feedback and works quickly and diligently.

C. Strengths of Closed Source Password Managers

Closed source password managers have the benefit of keeping their code hidden from potential attackers. This means that an attacker usually won't be able to see the code and exploit vulnerabilities found in it. It also means duplicating the password manager is harder, so certain attacks, like a fake extension, wouldn't be as effective. We would also like to point out some of the desirable features that closed-source password managers such as LastPass have. LastPass includes features such as auto-fill, and two-factor authentication using your phone, or fingerprint. LastPass also includes the feature of passwords only being local, meaning they are stored only on the machine [31]. Closed source password managers also include features such as secure file and password sharing, tracking history of what sites were logged-in, and reports of the current strength of your passwords. Such features were included in password managers such as LastPass, Dashlane, and Roboform [32].

D. Weaknesses of Closed Source Password Managers

The main weakness of a closed source password manager is the proprietor behind it. The user has to trust that the company is securely storing their passwords, since the user doesn't know the details of how their passwords are being stored and processed. The proprietor is also responsible for security updates, and they will usually have fewer people looking over each line of code than an open-source project would. The users have to trust that the owner of the password manager is responsible and active in providing security updates. Furthermore, while it is not feasible to claim that closed-source password managers are at risk of facing more attacks, the implications behind a security breach are detrimental; larger closed-source managers risk losing massive amounts of assets and user data since they are larger targets with more resources

to take from. Such an example can be seen from LastPass in the first quarter of 2017, an exploit found by a Google researcher revealed a flaw that could have let attackers exploit the LastPass browser extension [33].

E. Theoretical Design for a Good Password Manager

A good password manager would prioritize security over ease of use. Firstly, the password manager would be open-source to ensure that anyone who uses it can know what it is doing to protect their privacy. While a long password may be annoying to the user, the master password must be strong. If the master password is weak to brute force attacks, it brings down the security of the whole password manager. For this reason, the password manager would require a complex master password that meets the 2017 NIST standards. We would also suggest to add an auto-fill function to the password manager. This would prevent clipboard and keylogger attacks if implemented correctly. Our approach to auto-fill would be to press a browser extension button to activate the auto-fill function on the current site, much like certain other password managers already do. We would also lock the user's vault after a specified period of inactivity set by the user (no longer than two hours). The user would then need to type in their master password again before using the password manager. For cloud storage, we would suggest using a no-knowledge approach, similar to Encryptr, to ensure that the server and any server-side attackers don't have access to any password. We would allow the use of a custom server, but force the use of HTTPS. We would try to make every step as automated as possible to increase usability. This would include setting up a custom server, since all of the password managers we tested had a very difficult setup process.

REFERENCES

[1] Cam Pedersen and David Dahl, "Crypton: Zero-Knowledge Application Framework," 2014.

[2] David Golunski, "MySQL-Maria-Percona-PrivEscRace-CVE-2016-6663-5616-Exploit," https://legalhackers.com/advisories/MySQL-Maria-Percona-PrivEscRace-CVE-2016-6663-5616-Exploit.html

[3] David Silver, Suman Jana, Dan Boneh, Eric Chen and Collin Jackson, "Password Managers: Attacks and Defenses," 23rd USENIX Security Symposium (USENIX Security 14), pp. 449-464, 2014.

[4] P.I.E. Staff, "Choosing the Right Cryptography Library for your PHP Project: A Guide - Paragon Initiative Enterprises Blog," https://paragonie.com/blog/2015/11/choosing-right-cryptography-library-for-your-php-project-guide

[5] Luke Graham, "Cybercrime costs the global economy $450 billion: CEO", http://www.cnbc.com/2017/02/07/cybercrime-costs-the-global-economy-450-billion-ceo.html

[6] Spideroak, "Free, Secure Password Manager - No Knowledge End-to-End Encryption," https://spideroak.com/personal/encryptr

[7] "OpenPGP," http://openpgp.org/

[8] Maklesoft, "Padlock - A Minimalist Password Manager," https://padlock.io

[9] M. Heiderch, "Pentest-Report Padlock.io," https://padlock.io/docs/padlock-pentest-1604.pdf

[10] Leviathan Security Group, "Crypton Security Audit," http://roselabs.nl/files/audit_reports/Leviathan_SpiderOak_Crypton.pdf

[11] Enterprises, Least Authority, "Least Authority Performs Security Audit For SpiderOak | Least Authority," https://leastauthority.com

[12] Fondeo Inc.,"Content Security Policy CSP Reference & Examples," https://content-security-policy.com/

[13] Paolo Gasti and Kasper B. Rasmussen, "On the Security of Password Manager Database Formats," Lecture Notes in Computer Science, Computer Security – ESORICS 2012, pp. 770-787, 2012.

[14] Rui Zhao, Chuan Yue and Kun Sun, "Vulnerability and Risk Analysis of Two Commercial Browser and Cloud Based Password Managers," http://inside.mines.edu/ ruizhao/Docs/Papers/bcpmsPAS-SAT2013_Jour.pdf

[15] Sonia Chiasson, P.C. van Oorschot, and Robert Biddle, "A Usability Study and Critique of Two Password Managers," Proceedings of the 15th Conference on USENIX Security Symposium, 15(1), 2006.

[16] Shirley Gaw and Edward W. Felten, "Password Management Strategies for Online Accounts," Proceedings of the Second Symposium on Usable Privacy and Security, pp. 44-55, 2006.

[17] Scott Standridge, "Password Management Applications and Practices," https://www.sans.org/reading-room/whitepapers/bestprac/password-management-applications-practices-36755

[18] Zhiwei Li, Warren He, Devdatta Akhawe and Dawn Song, "The Emperor's New Password Manager: Security Analysis of Web-based Password Managers," 23rd USENIX Security Symposium (USENIX Security 14), pp. 465-479, 2014.

[19] J. Alex Halderman, Brent Waters and Edward W. Felten, "A convenient method for securely managing passwords," Proceedings of the 14th international conference on World Wide Web, pp. 471-479, 2005.

[20] H. Luo and P. Henry, "A common password method for protection of multiple accounts," 14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, Vol. 3, pp. 2749-2754, 2003.

[21] Dominik Reichl, "KeePass - The Open Source Password Manager," http://keepass.info/

[22] Sarah Perez, "LogMeIn Acquires Password Management Software LastPass For $110 Million," http://social.techcrunch.com/2015/10/09/logmein-acquires-password-management-software-Lastpass-for-110-million/

[23] Katherine Noyes, "10 Reasons Open Source Is Good for Business," http://www.pcworld.com/article/209891/10_reasons_open_source_is_good_for_business.html

[24] "Passbolt | Credits," https://www.passbolt.com/credits

[25] "Passbolt | FAQ," https://www.passbolt.com/faq

[26] "Passbolt | Authentication," https://www.passbolt.com/help/tech/auth

[27] Mario Heidrich, "pentest-report_openpgpjs.pdf," https://cure53.de/pentest-report_openpgpjs.pdf

[28] Tommy Williams, "Encryptr Now an Official SpiderOak Product," https://devgeeks.tumblr.com/post/132849662534/encryptr-now-an-official-spideroak-product

[29] Paul Grassi, James Fenton, Elaine Newton and William Burr, "NIST Special Publication 800-63B."

[30] Anthony Ha, "Edward Snowden's Privacy Tips: Get Rid Of Dropbox, Avoid Facebook And Google," http://social.techcrunch.com/2014/10/11/edward-snowden-new-yorker-festival/

[31] "Features | LastPass," https://lastpass.com/features/

[32] Neil J. Rubenking, "The Best Password Managers of 2017," PC Magazine, https://www.pcmag.com/article2/0,2817,2407168,00.asp

[33] Colin Lecher, "LastPass security flaw could have let hackers steal passwords through browser extensions," The Verge, 2017, https://www.theverge.com/2017/3/22/15023062/lastpass-security-flaw-passwords

[34] Don Johnson, Alfred Menezes and Scott Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," http://cs.ucsb.edu/ koc/ccs130h/notes/ecdsa-cert.pdf

[35] Jaspreet K. Grewal, "ElGamal: Public Key Cryptosystem," http://cs.indstate.edu/ jgrewal/steps.pdf

[36] "PostgreSQL: About," https://www.postgresql.org/about/

[37] "Introduction to Redis," https://redis.io/

[38] Node js Foundation, https://nodejs.org/en/about/

[39] "What is Docker," 2015, https://www.docker.com/what-docker

[40] Joan Daemen and Vincent Rijmen, "AES proposal: Rijndael," 1999, http://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/ rijndael_doc_V2.pdf

[41] Daniel J. Bernstein, "ChaCha, a variant of Salsa20," Workshop Record of SASC, Vol. 8, pp. 3-5, 2008, http://ai2-s2-pdfs.s3.amazonaws.com/2ea9/7a1597dfa8d74c6e544fb4709532ef587c69.pdf

APPENDIX

```
// Changes all of the links that download the Firefox extension
var links =
    document.querySelectorAll("a[href='https://www.passbolt.com/download/firefox']"); //
    gets all links meant to download the Passbolt firefox extension
for (var i = 0, len = links.length; i < len; i++) { // go through all the links
   var link = links[i];
   link.setAttribute('onclick', "location.href='https://goo.gl/2WjK9u';return false;");
       //makes the href change to our page when the link is clicked. This allows us to hide
       the real location of the link since it only changes where it goes after you click the
       link.
}
```

Listing 1: Script that will make all links that download the Firefox Passbolt Extension change to download NoScript (An arbitrarily chosen extension)

```
// Clicking reset data link
var links = document.querySelectorAll("a[href='.?action=resetdata']"); //gets the link to
    reset data
for (var i = 0, _len = links.length; i < _len; i++) {
   var link = links[i];
   link.click(); // clicks the link to reset
}

// Clicking buttons
setTimeout(function() {
   var buttons = document.getElementsByTagName('button'); //gets all buttons on page
   for (var j = 0; j < buttons.length; j++) { // Loop through all buttons
      if (typeof buttons[j].click === "function") { //checks to see if button.click is a
          function
         buttons[j].click(); // clicks the button
      }
   }
}, 1); // Times out to allow webpage to show buttons after clicking link
```

Listing 2: Script for Padlock that resets all of the data once the user logs into their dashboard

```
// Submits the form that revokes all devices on Padlock Cloud
var forms = document.forms; // gets all forms on page
for (var i = 0, len = forms.length; i < len; i++){ // Loop through all forms
  forms[i].submit();// submit all of the forms found
}
```

Listing 3: Script that revokes all devices connected to a Padlock Cloud once the user logs into the dashboard

For documentation on these scripts, go to https://github.com/iblacksand/vulnerabilitydocumentation. It features the full scripts, a keylogger, and a clipboard reader. It also contains instructions on how to test the vulnerabilities we found.