Augmenting Centralized Password Management with Application-Specific Passwords

Trevor Smith
Brigham Young University
tsmith@isrl.byu.edu

Scott Ruoti
MIT Lincoln Laboratory
scott@ruoti.org

Kent Seamons
Brigham Young University
seamons@cs.byu.edu

ABSTRACT

Password authentication is the most prevalent form of authentication; however, passwords have numerous usability issues. For example, due to the large number and high complexity required of passwords, users frequently reuse and choose weak passwords. One way to address these problems is to centralize password management by using a password manager or single sign-on. While this centralizing approach can improve a user's security, it also magnifies the damage caused by a compromise of the user's master password. In this paper, we describe a new approach to enhance centralized password management using application-specific passwords. This approach prevents the compromise of a master password from immediately compromising all associated applications and instead, requires the attacker to conduct further online attacks against individual applications. We detail five possible system designs for application-specific passwords and describe our plans for user studies to test the acceptance and usability of this approach.

1. INTRODUCTION

Passwords are the most common form of authentication, and their problems are legion. Still, their combination of usability and deployability have made it difficult for more secure alternatives to displace them [11].

Users have a large number of applications, both local and remote, that they authenticate to using a password. Ideally, a user would have a unique password for each application that was secure enough to survive an offline attack. Unfortunately, the sheer number of applications and limitations on human memory makes this ideal intractable. Instead, users commonly choose weak passwords and reuse their passwords across many applications.

Two common solutions to this problem involve centralizing passwords, either through a password manager or a single sign-on (SSO) system. Password managers (e.g., 1Password, LastPass) generate random passwords and manage those passwords for users, only requiring the user to remember a single master password to unlock their password manager. SSO (e.g., Google OAuth 2.0, Facebook Connect) allows a user to authenticate to many different applications using a single global account and its associated password. In both cases, the hope is that if users need to select fewer passwords, then those passwords will be much stronger than the passwords currently selected by users.

While password managers and SSO both significantly improve the usability of passwords, they introduce a single point of failure where a system compromise has the potential to be much more damaging than the compromise of a single-application account. Users are cognizant of this limitation, as demonstrated by a recent usability study of authentication methods conducted by Ruoti et al. [20]. During the study, users indicated that they liked SSO, but that they were reluctant to trust a single third party (i.e., password manager or SSO identity provider) with information and access to potentially all of their accounts.

Several users suggested that SSO could be augmented to include an additional application-specific password for high-value applications. The benefit of this augmentation is that even if a user's SSO account is compromised, an attacker does not automatically have access to the user's high-value applications. While the suggestion was made regarding SSO, it is equally applicable to password managers. Interestingly, the suggested augmentation represents a middle-of-the-road solution between the extremes of requiring each application to have its own unique, user-memorized password and centralizing all passwords into a single password manager or SSO account.

In this paper, we describe our attempts to design systems that would augment centralized password management with application-specific passwords. Additionally, we describe our plans for user studies to test the acceptance and usability of this approach. At the workshop, we look forward to feedback and discussion of this approach to authentication, our system designs, and the proposed user study.

2. BACKGROUND

In this section, we describe password managers and single sign-on (SSO). We also describe the threat model that motivates our work.

2.1 Password Managers

Password managers serve three purposes. (1) They help users generate random passwords that satisfy specific

Who Are You?! Adventures in Authentication (WAY 2017) July 12, 2017, Santa Clara, California.

password policies while also maximizing entropy. (2) The password manager stores passwords for the user—both generated passwords and those set by the user—in a password vault. This functionality is essential, as it is infeasible for users to memorize many of the randomly generated passwords. To access the passwords in the user's vault, they must input the master password that was used when the vault was created. (3) The password manager can automatically enter the appropriate password into applications the user is authenticating to. All password managers support (2), most support (1), and there is an ongoing debate whether (3) should be supported as it has various security implications [17]. Many password managers store the password vault in the cloud, allowing users to synchronize this vault to all of their devices. There are many password managers, including password managers built into browsers (e.g., Chrome, Edge, Firefox), browser extensions (e.g., LastPass [7]), and standalone applications (e.g., LastPass [7], 1Password [1]).

2.2 Single Sign-on (SSO)

Single sign-on allows users to authenticate to a relying party (RP) by way of an identity provider (IDP). First, a user authenticates to their IDP and then specifies the RP to which they want to prove their identity. Second, the IDP creates a signed attestation of the user's identity—this attestation is time-bounded and specifies the RP it is intended for—and sends it to the user. Third, the user presents the signed attestation to the RP. Finally, the RP validates the attestation using a long-term secret shared between the RP and IDP. There are many SSO systems [2, 6, 8, 3, 10, 16, 9, 15], with the two most widely-deployed SSO systems being Google OAuth 2.0 [5] and Facebook Connect [4].

2.3 Threat Model

Our threat model includes four parties: the user, the relying party, an identity provider or cloud-based password manager, and the adversary. The adversary's goal is to impersonate the user to the relying party, without alerting any other party to this impersonation. There are several approaches the adversary can use to accomplish this goal:

1. The adversary can directly steal one or more of a user's passwords (e.g., phishing).
2. The adversary can steal the password database from a relying party and conduct an offline attack against the user's password. If password sharing occurs, an attack against one relying party might compromise multiple relying parties.
3. The adversary can steal the password database from the identity provider or cloud-based password manager and conduct an offline attack against the user's master password.

In each case, we consider attacks where the attacker has long-term access to the other parties as out of scope (e.g., key logger, insider threat at the identity provider). While centralizing password management mitigates the second attack vector, it potentially magnifies the damage of the first and third attack vectors. Specifically, if the attacker obtains the master password for the password manager or SSO, they will gain access to a multitude of applications. Alternatively, if the identity provider or cloud-based password manager are not fully trusted, it is also possible for these services to impersonate the user.

3. APPLICATION-SPECIFIC PASSWORDS

If a password manager's password or an account password for SSO—hereafter also referred to as a master password for consistency—is compromised, all of the user's other applications immediately become compromised.[1] This is a natural consequence of consolidating all authentication data into a central party.

To partially mitigate the damage of a compromised master password, it is possible to decrease the level of centralization, and introduce an *additional* application-specific password for high-value sites. With application-specific passwords, the compromise of a user's high-value site only occurs when there is a compromise of both the master password and the site's application password. The design of any application-specific password system must ensure that the loss of a master password does not allow an offline attack against the application-specific password. Instead, the attacker will need to perform an online attack, which can be easily detected, or to also compromise the reliant application. Importantly, this means that the application-specific password must be remembered by the user and cannot be stored by a password manager. Also, application-specific passwords should ideally ensure that any loss of data by the central party does not allow an offline attack against the application-specific password.

Application-specific passwords are a middle-of-the-road approach between fully centralized (Password Managers, SSO) and fully decentralized password management (every website has its own user-generated and remembered password). Application-specific passwords are only intended to augment centralized approaches in order to limit the effect of a compromised central party by requiring additional online attacks to be carried out against the high-value applications. Specifically, these passwords will likely be much shorter than master passwords, with the user's authentication security being predominantly derived from the protection gained from the central party.

Florêncio et al. [14] describe the large chasm that exists between the complexity of passwords that are strong enough to resist an online attack compared to those that only need to resist an off-line attack. Our design addresses these differences by requiring the central party to generate strong passwords resistant to offline attacks and relying on the user to select passwords that thwart online attacks. This is less demanding on the user than requiring them to manage numerous strong passwords and does not require that they place inordinate trust in the central party.

3.1 System Designs

We have identified five possible designs for adding an application-specific password to password managers and/or SSO.

[1]For simplicity, we assume that all accounts are managed by either a password manager or SSO.

3.1.1 Application Password

After authenticating to an application using a password manager or SSO, the application itself could present an entry form for the additional application-specific password. While this approach requires changing the application, it does not require changing password managers or SSO systems. This design does provide protections against a stolen master password or a compromised central party. The biggest drawback of this approach is that applications have to self-identify as high-value, and users will likely not have control of which applications offer this feature and which do not.

3.1.2 Password Manager + Hashing

The password manager could be modified to require users to enter two passwords for high-value applications. These two passwords would then be hashed together, and the resulting hash presented to the application. Importantly, the password manager can generate and/or store only one of the two passwords. The user *must* generate and remember the application-specific password. This design requires modifying the password manager, but not the applications. It provides protections against a compromised master password or password vault. If an RP's password database is stolen, it might be possible to perform an offline attack against that specific password if both the password manager-generated and user-generated passwords are weak.

3.1.3 Password Manager + User

Instead of changing the password manager to mix the application password with the vault-stored password, the user could handle this responsibility. When authenticating, the password manager would fill in the application's password as normal, but then the user would append the application-specific password onto the end of the auto-filled password. The user could also adopt a more complex strategy to mix in the application-specific password. This approach has the same properties as *Password Manager + Hashing*, except that it does not require modifying the password manager. There is the potential for significant usability challenges, as the password manager might attempt to remember the full password (vault-stored password + user-entered password) in order to auto-fill that value in the future.

3.1.4 Modified SSO Protocol

The SSO protocol can be modified to incorporate information about the application-specific password into the SSO signed attestation from the identity provider. The application could then use a verifier stored locally, and *not* at the SSO, to verify the application-specific part of the signed attestation. This approach requires modifying both the application and the SSO server. This approach provides protection against theft of the master password or compromise of the SSO server. Unlike password manager-based approaches, even if the data stored at the SSO is stolen, it cannot be used by itself to conduct an offline attack against the user's credentials for that application.

3.1.5 SSO + Challenge

The SSO authentication flow could be modified to include an application-specific password entry screen following entry of the SSO password. The identity provider presents the screen and verifies the application-specific password. This approach obviates the need to modify applications. While it does protect against the loss of a master password, compromise of the identity provider makes it possible for an attacker to conduct an offline attack against both the master password and application-specific passwords.

4. ANALYSIS

In this section, we analyze the the security[2], deployability, and usability of the five design alternatives. Table 1 summarizes the key differences between these approaches.

4.1 Security

All five system designs succeed at preventing a lost master password from immediately compromising high-value applications. In each case, after the compromise of a master password, the attacker will still need to conduct an online attack against the application-specific passwords. The security of the application would then depend on its ability to detect and defend against online attacks.

Of the five systems, only `SSO + Challenge` fails to protect high-value sites when the password database is stolen from the SSO. In this case, the stolen password database can be used to perform an offline attack against the application-specific passwords. This limitation is a trade-off for this design not requiring modification of applications.

While outside the threat model, our system designs also help prevent against a malicious central party impersonating the user. In this case, the central party must conduct an online attack against the application-specific passwords (except in the case of `SSO + Challenge`). Similarly, an attacker with non-administrative access to a user's unlocked device (e.g., laptop left open at a table) cannot access the user's high-value sites because they are protected by an application-specific password that cannot be "remembered" by the system for later use.

Password manager-based approaches (including existing password managers) have a limitation in that a compromise of a given application is sufficient to allow impersonation to that application. Specifically, an adversary that steals a password database can conduct an offline attack to guess the users' passwords. These stolen passwords can then be used to impersonate users at the previously-compromised application. Password managers still improve on vanilla passwords in that the compromised password will not be used for other applications. Also, in many cases password managers will have generated passwords that can survive most offline brute-force attacks. Regardless, this is weaker than SSO-based approaches which do not rely on the security of individual applications to protect authentication.

Though centralized authentication systems create a single point of failure, they also provide a single point of entry, meaning that access to the central store is necessary to access any associated account. Security enhancements to the central store necessarily increase the security of all associated accounts. Analyzing the specific effects of centralized security modifications on associated accounts is beyond the scope of this proposal and is left for future investigation. However, we note that mitigating the consequences of a single point of

[2]The security analysis assumes adherence to best practices when choosing passwords resilient to online attacks.

System Design	Requires no change to applications	Requires no change to password managers	Requires no change to SSO	Stolen master password does not compromises all applications	Compromised central party does not compromise all applications	Stolen application data does not allow impersonation to that application	Notes
(§3.1.1) Application Password	○	●	●	●	●	○/●[1]	
(§3.1.2) Password Manager + Hashing	●	○	—	●	●	○	
(§3.1.3) Password Manager + User	●	●	—	●	●	○	Potential usability hurdles
(§3.1.4) Modified SSO Protocol	○	—	○	●	●	●	
(§3.1.5) SSO + Challenge	●	—	○	●	○	●	All verifiers stored at SSO

[1] Empty dot if used with password managers and full dot if used with SSO.

Table 1: System Design Comparison

failure with an application-specific password does not remove the security benefits gained with a single point of entry.

4.2 Deployability

One hurdle to deploying application-specific passwords is the need to modify existing systems. The first set of columns in Table 1 demonstrates the different deployability characteristics of each system design.

The `Application Password` design can be unilaterally deployed by applications without changing password managers or SSO systems. The password manager designs work with existing applications, and the `Password Manager + User` design also requires no modification to existing password managers either. This comes with a trade-off that the password manager might try to memorize the application-specific password, a potentially significant usability issue. Finally, the SSO designs require changing SSO, but by relaxing the security guarantees `SSO + Challenge` can be implemented without a change to existing applications.

The deployment of both `Application Password` and `Modified SSO Protocol` are limited by the need to modify applications. As there are a large number of applications, it is unlikely that all will be modified. Partial deployment limits user choice. Users may be unable to protect all of their high-value sites. Also, some applications may not let users elect to forgo the additional password for less important accounts.

4.3 Usability

There are many potential usability hurdles for application-specific passwords. For example, while it was users that suggested application-specific passwords, do they find these extra passwords overly burdensome to use on a consistent basis? Alternatively, how do users select their application-specific passwords? Are they unique from each other? How much entropy do they provide? We propose several user studies to answer these questions and evaluate their security implications.

4.3.1 Attitude and Acceptability Study

We plan to conduct user interviews that explore user's attitudes towards centralized password management and application-specific passwords. Additionally, we will review the five possible designs with the users, and determine which they find most acceptable. We will also work with users to establish possible UI designs. Finally, we will validate the results from the interviews with a large Mechanical Turk survey.

4.3.2 Laboratory Usability Studies

We will conduct a laboratory user study to validate that users truly do prefer application-specific passwords to existing systems and to obtain user feedback on prototype implementations of the five designs. These usability studies will leverage the methodology of Ruoti et al. [20]—multi-round, within-subject evaluations of multiple systems, where systems are compared using the System Usability Scale (SUS) [12]. In these studies, we will ask post-study questions related specifically to application specific passwords.

4.3.3 Longitudinal Studies

There is a strong risk that users might enjoy application-specific passwords in a laboratory setting but would not use them in a real-world setting. To evaluate the long-term acceptability of application-specific passwords, we plan to conduct a longitudinal study of the system design rated as most usable in the laboratory studies. We will have participants use the system over the course of a month, measuring how many sites they protect with an application-specific password and how often they authenticate to those sites. During the study, we will ask users to report their experiences in an authentication journal. We will also conduct post-study debriefing interviews to gather valuable qualitative feedback.

5. RELATED WORK

Ruoti et al [20] evaluated several single sign-on systems (Google OAuth 2.0, Facebook Connect, and Mozilla Personas). They found that users preferred these systems to alternative authentication schemes. Still, participants in the

study expressed concern that a compromise of the identity provider would result in the immediate compromise of all of their other applications. While participants were willing to accept that for low-value applications, they indicated that they wanted more security for their high-value applications. Earlier, Sun et al. [21] conducted a user study of OpenID, a single sign-on system. Some participants in that study also expressed concern that the identity provider represents a single point of failure.

Ross et al. [19] introduced PwdHash, a password scheme that hashes a user's master password with the domain name, producing a unique password for each domain. To trigger this hashing, users preface their password with "@@". This approach does not prevent offline attacks against the user's password. In contrast, our system design uses a second user-supplied application-specific password, and not a static domain name, to augment the master password.

Llewellyn-Jones and Rymer [18] describe an approach to defeating PwdHash's client-side hashing of the user's password and the domain name of the website. Our `Password Manager + Hashing` system design will need to consider how to prevent a similar attack.

Chiasson et al. [13] explored the usability of password managers by conducting a 26-person user study comparing PwdHash and Password Multiplier. The study discovered significant usability issues with both systems even though usability was considered during system design, illustrating the importance of user studies. The goal of our proposed user study is to determine whether our proposed systems have similar usability flaws.

As indicated by Bonneau et al.[11], Two-factor authentication systems offer similar and in many cases stronger security protections; however, two-factor authentication can introduce a non-negligible usability burden on users. One of the driving factors for using an application specific password was to retain as much usability gained by password managers and SSO over traditional passwords. Ultimately, a compromise must be made between usability, deployability, and security. Our proposed systems sacrifice potential security benefits to retain significant usability and deployability characteristics.

6. CONCLUSION

While the centralization of password management into password managers and single sign-on (SSO) comes with strong benefits, it also magnifies the damage of a user having their remaining password—their master password—compromised (e.g., phishing attack). Enhancing centralized password management with application-specific passwords for high-value applications can mitigate this damage. This approach prevents the loss of a master password from compromising all applications, instead of requiring that the attacker conducts online attacks against individual applications.

In this paper, we describe five possible system designs for application-specific passwords—`Application Password`, `Password Manager + Hashing`, `Password Manager + User`, `Modified SSO Protocol`, and `SSO + Challenge`. We also evaluate the security and deployability trade-offs for each of these designs. Finally, we describe several planned users studies to explore the acceptance and usability of application-specific passwords.

7. REFERENCES

[1] 1password. https://1password.com/.

[2] CardSpace. http://msdn.microsoft.com/CardSpace.

[3] Central Authentication Service. http://www.jasig.org/cas.

[4] Facebook connect. https://developers.facebook.com/blog/post/2008/05/09/announcing-facebook-connect/.

[5] Google OAuth 2.0. https://developers.google.com/accounts/docs/OAuth2/. [Online; accessed 2014/11/20].

[6] Higgins: Open Source Identity Framework. http://www.eclipse.org/higgins/.

[7] Lastpass. https://www.lastpass.com/.

[8] Liberty Alliance Project. http://projectliberty.org/.

[9] OpenID. http://openid.net/.

[10] Shibboleth. http://shibboleth.internet2.edu/.

[11] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Symposium on Security and Privacy (SP)*. IEEE, 2012.

[12] J. Brooke. SUS—a quick and dirty usability scale. In *Usability Evaluation in Industry*. CRC Press, Boca Raton, FL, 1996.

[13] S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *15th USENIX Security Symposium*, 2006.

[14] D. Florêncio, C. Herley, and P. C. Van Oorschot. An administrator's guide to internet password research. In *28th Large Installation System Administration Conference (LISA14)*, 2014.

[15] E. Hammer-Lahav, D. Recordon, and D. Hardt. The oauth 2.0 authorization protocol. *draft-ietf-oauth-v2-18*, 8, 2011.

[16] J. T. Kohl, B. C. Neuman, and T. Y. Ts'o. The Evolution of the Kerberos Authentication Service. In *Spring EurOpen Conference*, 1991.

[17] Z. Li, W. He, D. Akhawe, and D. Song. The emperor's new password manager: Security analysis of web-based password managers. In *USENIX Security*, 2014.

[18] D. Llewellyn-Jones and G. Rymer. Cracking pwdhash: A bruteforce attack on client-side password hashing. In *The 11th International Conference on Passwords (Passwords 2016)*. Springer, 2017.

[19] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *USENIX Security*, 2005.

[20] S. Ruoti, B. Roberts, and K. Seamons. Authentication melee: A usability analysis of seven web authentication systems. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015.

[21] S.-T. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey, and K. Beznosov. What makes users refuse web single sign-on?: An empirical investigation of OpenID. In *Symposium on Usable Privacy and Security*. ACM, 2011.