PASSWORD MANAGERS: COMPARATIVE EVALUATION, DESIGN, IMPLEMENTATION AND EMPIRICAL ANALYSIS

by
Daniel McCarney

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTERS OF COMPUTER SCIENCE

AT

CARLETON UNIVERSITY
OTTAWA, ONTARIO, CANADA
AUG 26TH, 2013

Dedicated to my parents, John and Cheryl.

Table of Contents

List of Tables

List of Figures

Abstract

Passwords continue to prevail on the web as the primary method for user authentication, despite well-known security and usability drawbacks. Password managers are known to offer some improvement without the deployment barrier of server-side changes. This thesis examines password managers to alleviate some of the security and usability deficits of password authentication, while retaining the deployability advantages of passwords

In order to provide more fine-grained comparative evaluation of password managers, we extend the Usability-Deployability-Security (UDS) framework of Bonneau *et al.* (IEEE Symposium on Security and Privacy, 2012), by adding additional evaluation properties which allow differentiation of password managers by important characteristics not measured by the more general UDS.

We introduce and evaluate the security of *dual-possession authentication*, an authentication approach offering encrypted storage of passwords and theft-resistance without the use of a master password. We further introduce Tapas as a concrete implementation of dual-possession authentication leveraging a desktop computer and a smartphone. Tapas requires no server-side changes to websites, no master password, and protects all the stored passwords in the event either the primary or secondary device (*e.g.,* computer or phone) is stolen.

To empirically evaluate the viability of Tapas as an alternative to traditional password managers, we perform a 30 participant user study comparing Tapas to two configurations of Firefox's built-in password manager. We found users significantly preferred Tapas. We then improve Tapas by incorporating feedback from this study, and reevaluate it with an additional 10 participants.

Acknowledgements

The author acknowledges David Barrera, Dr. Jeremy Clark, Dr. Sonia Chiasson, and Dr. Paul Van Oorschot for their input and contributions to both the Tapas password manager and this thesis. Further acknowledgements are due to Dr. Robert Biddle for his guidance on the statistical analysis performed within this work. The author also acknowledges Dr. Carlisle Adams for his role on the defence committee. Finally, the author wishes to acknowledge Robin Langerak for her continued support throughout the composition of this thesis.

Chapter 1

Introduction

A large number of research contributions have been made toward increasing the security and usability of password-based authentication [12]. Many of these attempts require account providers to change how they handle authentication by augmenting or outright replacing passwords; *e.g.,* one-time passwords, dual-factor, single-sign on, biometrics, graphical passwords, *etc.* Recently, researchers have argued that despite the wide-held sentiment from the security and usability communities that passwords need to be replaced, the incumbency, familiarity, and low cost of traditional passwords continues to hamper widespread adoption of an alternative, as well as a lack of consensus on what exactly the alternative should provide [32].

We are interested in practical solutions combining easy deployability with security and usability. For this reason in this thesis we exclude from interest proposals requiring server-side changes. Previous research under this constraint focuses on storing and retrieving passwords for users (*e.g.,* password managers), strengthening password quality (*e.g.,* randomly-chosen, cryptographically processed, or site specific passwords), and encoding alternative authentication mechanisms into passwords (*e.g.,* graphical or object-based passwords). These three classes of solutions tend to address orthogonal issues and can be complementary. We focus on the first, not necessarily excluding the others.

Password managers are designed to relieve password fatigue, reduce cognitive load on users, and reduce log-in time. They can also indirectly facilitate better password quality and a reduction in password reuse. A naive password manager simply stores the passwords, while security-conscious managers lock the stored passwords under a master password. Password managers may also integrate other techniques to strengthen or encode passwords, including those mentioned above.

Password managers have certain drawbacks. To use a password manager, existing

accounts must be migrated to use the manager, potentially requiring current passwords be changed. In the event an adversary gains access to the manager's storage, a naive password manager offers no protection, making it a high value target. With a master password, the manager provides at best a level of protection dependent on the strength of the master password against an offline attack, provided the attacker has access to the manager's storage. This is assuming the theft does not occur when the manager has unlocked the passwords for the duration of a session, in which case the protection offered is greatly reduced. Password managers that maintain unprotected passwords during use do not always clearly indicate to the user the current state (locked or unlocked) of the system.

Password managers can be divided into subcategories based on the credential management approach taken. These categories motivate a natural taxonomy of password managers. In this thesis we employ the Usability-Deployability-Security (UDS) framework of Bonneau *et al.* [12] in order to evaluate a collection of password managers, comparing the resulting evaluation to the taxonomy presented. Insights into the divergence of password manager approaches is offered. Further extensions to UDS are introduced to allow fine-grained differentiation within the category of password managers, and applied to several password managers.

In this thesis, we also present a type of password manager that combines usability advantages of the naive password manager with protected storage. Passwords are protected against offline attacks with a strong encryption key which the user need not remember, and decryption requires the control of two independent devices (*e.g.,* a desktop PC and a smartphone). Operation of this type of manager requires no master password, only control of both devices. If any one of these two devices is stolen, the adversary cannot recover the passwords in practice. In the general case we refer to this design of password manager as implementing *dual-possession authentication*, a form of authentication requiring access to two distinct devices (*something you have*).

We consider a specific instantiation of this type of manager, Tapas,[1] and present its design, implementation, and analysis. Tapas is a smartphone-assisted password manager for a desktop computer that requires no server-side changes from account

[1] Tap-based authentication using a smartphone

providers. It maintains security of the managed passwords by encrypting and storing the encrypted passwords on a smartphone, keeping the decryption key inside the browser on the paired computer. Tapas is resistant to theft in the following sense: an adversary must steal both the smartphone and the user's computer to gain access to managed credentials. Tapas is designed to provide a simple mental model of sending the password from the phone to the login screen on a separate device, maintaining no cached master password, and not storing any credentials on the local device's long-term memory. Unlike a hashing-based solution [16], Tapas does not preclude memorization of passwords; login outside of the Tapas system is thus possible.

We also present the results of a 30-participant user study evaluating a Tapas prototype and comparing it to the built-in Firefox password manager both with and without the use of a master password. Our study found that in general users have little knowledge of the benefits password managers provide or the means by which they protect passwords. We believe this leads to an underutilization of browser password managers and low enrollment in opt-in master password protection. Participants selected to use Tapas rated their enjoyment of using Tapas higher than participants' ratings for the Firefox password manager (both with and without a master password). Further they were able to utilize Tapas successfully and without error to store credentials and log into websites.

1.1 Contributions

The primary contributions of this thesis are as follows:

1. We provide a background review, and introduce a taxonomy of password managers, contrasting the strengths and weaknesses of the subcategories. Illustrative examination of several existing password managers are provided as background.

2. We introduce an extension to UDS [12] in order to improve its granularity such that meaningful comparsion between password managers can be performed. Discussion on the results of performing an application of these extensions to existing password managers is included to illustrate usage and results.

3. We present and study the notion of *dual-possession authentication* which has received little attention in the literature. We develop a threat model for using it in conjunction with a password manager and find it offers a practical set of security and usability properties.

4. To allow concrete evaluation of this notion, we design and implement a dual-possession password manager (Tapas) using a Firefox extension on a primary device and an Android app on a secondary device. Although the idea of requiring two devices for secure password retrieval is simple, the implementation involves several novel and subtle security and networking details. Tapas requires no server-side changes, no master password, and offers theft-resistance for the managed passwords.

5. We validate the viability in practice of Tapas through an in-person user study with 30 participants comparing Tapas to two browser password managers. Users of Tapas were successful in using the system, even without prior knowledge of password managers. Using insights from the initial study we improve the Tapas design and then conduct a 10 participant follow-up study to evaluate it, finding it improves user's understanding of the system.

Parts of items 3, 4, and 5 have appeared as a refereed publication [45].

Chapter 2

Password Managers, Background and Related Work

2.1 Problems with regular passwords

Despite the prevalence of authentication schemes designed to replace passwords, their status as the de-facto authentication scheme is undeniable. Though some progress has been made in deploying two factor authentication within enterprise settings, authentication on the web remains dominated by passwords. With ubiquitous support from both users and service providers one might ponder why there is a demand to replace or otherwise augment passwords. From an adoption standpoint password authentication has succeeded tremendously. Unfortunately from a security standpoint password authentication leaves much to be desired.

Low Entropy. The strength of a password authentication system is inherently limited by the strength of the password, relative to the threat of guessing attacks. If an adversary is able to guess a user's password or otherwise quickly enumerate all possible password choices the authentication is bypassed, allowing an adversary to impersonate a valid user and acquire the privileges assigned to them.

The two primary means by which an adversary is able to accomplish this task are dictionary attacks and brute force attacks. In a dictionary attack a list of common password choices is combined with a standard dictionary to provide a source of password guesses. An attacker is able to enumerate this list until a guess is successful or the list is expended. A brute force password cracking attempt differs in that rather than sourcing guesses from a list of likely passwords forming a subset of all available passwords, the attacker instead enumerates the entire password space preferably in some deterministic ordering based on probability. In this attack strategy every potential password is tried in a systematic way by composing password guesses informed by the available character set and the maximum password length.

In order for a password to be resistant to a dictionary attack it must not be a common word or easily guessable permutation of such (*e.g.,* a dictionary word with a single digit appended). For a password to be resistant against brute force attack it must be both composed of characters from a large character set (*e.g.,* letters of mixed case, numbers and symbols) and additionally of sufficient length to ensure the adversary must expend an infeasibly large amount of time and computational resources to exhaust all possible choices. These requirements encourage long and unwieldy password choices that are both difficult for users to create and even more difficult for them to remember.

The ramifications of user password selection on security are not theoretical topics. Large scale analysis of password choice has estimated many user-chosen passwords have extremely low entropy against both online and offline attacks [11] (10 and 20 bits respectively). Analysis of large scale password breaches has allowed researchers to study password choices of real users from an ecologically sound dataset not tainted by study design. Examination of these data sets confirms users choose predictable passwords [58], a large percentage of which are cracked with a small investment of computational resources.

Memory Demands. Having account providers generate strong random passwords for users rather than allowing human chosen passwords addresses the difficulty in creating secure passwords but does not address memorization difficulties. Password policies further complicate matters by requiring users to choose difficult to remember passwords containing symbols and numbers. The benefit of these policies is questionable [30] but clearly places higher demand on user memory.

Issues of password choice are further aggravated by the sheer number of passwords a user is expected to manage. As the number of accounts associated with a user grows so too does the number of passwords. Some studies estimate the average user has a username/password combination for 6 separate services [21], others as high as 25 [20], each of which is ideally unique and secure. Studies highlighting the manner in which accounts comprising a user's "digital footprint" accrue over time [39] estimate the total number of accounts even higher. A tempting strategy employed by users to alleviate the memory load caused by this is password reuse across accounts.

Unfortunately this too has security implications as the compromise of one account and its associated password can ripple outwards resulting in the compromise of disparate accounts and services for which the password was reused. One study estimates the average user is estimated to have 6.5 unique passwords, each of which is shared across 3.9 sites [20], indicating that password reuse is the norm.

Phishing. Common use of passwords offers no resistance to phishing. If a user can be lured to a fake website and convinced to enter their username and password the adversary has gained all of the information required to hijack the user's account, if that password is the only mechanism used by the corresponding service. Once a password is leaked to a phishing attack few remediation options are available to a user outside of a password reset.

User Familiarity. Users of every imaginable demographic are comfortable with the paradigm of username and password authentication and have amassed significant experience accessing services protected in this way. The tangible cost to protect a resource with a username and password is nearly zero. No new hardware is required for users or the service, no software requires expensive adaptation, and resources need not be allocated to acclimatize users to the use and operation of passwords.

Comparing many alternative authentication schemes to passwords helps indicate why few have obtained meaningful adoption on a wide scale [12]. Many schemes propose radically different usage scenarios than users expect, eschewing decades of user expectations. Others require costly new hardware to be purchased and distributed to end users (*e.g.,* client-end biometric input, two factor authentication involving password generator devices). Most proposed post-password authentication schemes require intrusive backend/infrastructure changes by service providers in order to allow for adaptation by end users, further hampering the likelihood of wide spread adoption.

2.2 Password managers overview

Recently, researchers have been encouraged to consider that the persistence of passwords is not incidental [32]: their advantages as a well-known, firmly entrenched incumbent (*e.g.,* widespread familiarity, marginal cost per user) outweigh the costs of implementing an alternative, and this is asserted to be unlikely to change in the near-term. Despite a wide-held sentiment from the security and usability communities that passwords must be replaced, there is little consensus on the actual harm incurred by password breaches [30] (passwords may not be the last line of defence), what fraction of breaches is attributable to each threat vector, and thus, what alternative schemes should prioritize.

Password managers (client-side tools to assist password-based authentication) offer the promise of a compromise between addressing the drawbacks of password authentication and respecting the factors that prevent outright replacement. Current security advice given to users regarding passwords is untenable [21, 20, 57]. Users are expected to create unique, strong passwords for each of their individual accounts and these passwords must be protected indefinitely against compromise and loss. Evaluating the burden of these requirements next to the potential gain they provide has been asserted by some to indicate that rejection of password advice by users is entirely rational [30].

The need to alleviate the memory burden placed on users by this unrealistic advice motivates password managers as a class of potential solution. Password managers all endeavour to reduce the number of passwords a user must retain in their memory. Some password managers accomplish this by acting as a memory aid, committing passwords to a storage medium other than the human mind. Others accomplish a reduction of cognitive load by strengthening, or stretching, one memorized password into many unique passwords.

Besides the usability benefits offered by reducing memory demands, many password managers also provide great deployability advantages compared to an outright password replacement. Password managers can transparently act as a middle-man between the user and the service to which they are providing a password, requiring

no code changes or alternative authentication options be provided by the end service. The down-side is that preserving passwords as the underlying authentication mechanism prevents password managers from addressing all of the security failings of passwords, instead electing for greater backwards compatibility with existing services. The details of password manager implementations may differ, but at the end of the login process the service being authenticated with will always receive a password.

Password managers also leverage user familiarity with passwords to lessen the learning curve for adoption. The incumbency of passwords as the dominant authentication mechanism for the past several decades has created large psychological barriers [13] providing resistance to the adoption of radically different authentication technologies. Both users and service providers are most comfortable using password authentication for reasons varying from cultural to technological. An outright replacement of passwords will incur education and support costs that may be prohibitively expensive for small providers or free services. Password managers retain much of the familiarity of password authentication (*e.g.,* usernames, passwords) and can be adopted on a per-user basis. Their biggest disadvantage is, depending on the implementation details, often limited security improvement.

2.3 Generative password managers

Generative password managers typically operate by taking one secret and strengthening or stretching it through multiple iterations of a cryptographic hash function [27]. To produce site-specific passwords generative password managers will include metadata such as the username or site URL as input to the password generator function, thereby achieving differing output for each website while requiring only one secret be provided as input by the user. Generally the input secret provided by the user is called a *master password* or *master secret*, one hopefully strong secret in the form of a password that is used by the users' device to generate site specific passwords. Other approaches replace this input secret with a graphical password [8], or a digital object [9, 42, 44]. Where possible we refer to the input secret (password or otherwise) as the *master secret*. The choice of how the master secret is used to generate site-specific passwords is the primary differentiator between generative password managers.

By including domain information as well as the username/master password as input to the hash function, generative password managers can achieve phishing protection [52], addressing a common password authentication security concern. If a user is lured to a website with a domain name closely matching one at which they have a managed account, the site-specific generated password will not be equal to the real password used for the account at the legitimate domain. As the protection is based on domain name, it does not address pharming attacks [56] that trick users by manipulating the domain name resolution system.

Generative password managers offer increased usability by freeing users of the requirement to remember individual passwords. Further, security advantages are gained by removing the possibility of both poorly chosen, low entropy site passwords as well as removing password reuse across multiple sites. Generative password managers also benefit from a lack of stored state; no passwords must be maintained or stored between logins. This helps to prevent issues of theft or offline attack common to managers that must safely store password state between sessions. A lack of state additionally grants users the ability to change machines, only requiring the user to install the software used to generate site-specific passwords from the master secret and site metadata onto each machine. No export/import process must be undertaken in order to accomplish the transition from one computer to another. Some generative password managers (*e.g.,* [52]) offer a website that can be used to generate site-specific passwords per the generative password manager's algorithm for use without having the software installed.

The primary detriment of generative password managers is the amount of work required by a user to make the initial transition to using the tool [59]. For each of the accounts the user wishes to manage they must perform a password change from the password they are currently using (and have presumably memorized) to the site-specific generated password that the manager produces. For a user with many accounts this is a prohibitively time intensive step. Similarly, anytime the user wishes to change their master secret they must again update all managed accounts as the derived passwords will now differ. This seems sufficiently cumbersome to more or less make such managers non-viable in practice.

Generative password managers also require users to cede full control of their password management to the tool. While retrieval password managers allow users to maintain existing passwords they have previously memorized by importing them into the manager, generative password managers require that these passwords be changed. Existing examinations of the usability of password managers in the literature [16, 35] emphasize that users dislike giving up control of their authentication experience.

The master secret employed by generative password managers to produce site-specific passwords is also a single point of failure. If a user forgets the master secret they will be unable to use the tool to generate the correct password for any of the managed accounts. In the absence of a master secret recovery (which itself could be the source of many security issues) the user is forced to employ the password reset functionality specific to each of the managed accounts that must be recovered.

Without careful construction a generative password manager may be vulnerable to offline attack on the master secret. By capturing one site specific password an adversary may be able to combine the URL of the site for which the password is captured, and a candidate master secret guess. By following the known hashing procedure used by the targeted generative password manager the adversary is able to compute a candidate password to compare against the site specific password that was captured. When the generated password matches the captured password the adversary has verified the candidate master secret guess and can proceed to generate site specific passwords for all other managed websites. Given the poor entropy [11] of user chosen passwords, if a master password is used as the master secret it may thus not be resistant to an offline attack.

In the event the master password of a generative password manager is disclosed to an adversary, that adversary is then able to generate all current and future site-specific passwords. This is in contrast to *retrieval* password managers which require the adversary capture both the master password and the stored passwords protected with the master password. With generative password managers the adversary can perform the hashing process used by the manager (generally a well known implementation detail) to combine the captured master password with site-specific information.

If the master secret is a password, inadvertent disclosure could occur due to user

misunderstanding or as the result of a phishing attack. For example if a user incorrectly enters their master password to a website login form rather than the password manager tool [16] the website gains all of the information required to derive the passwords for the user's other managed accounts. Inadvertent disclosure may also occur from a user mistakenly typing the master password into the username field. Similarly, if an adversary presents a user with a fake password manager master secret entry screen they may be able to convince the user to disclose their master secret (password or otherwise).

The lack of user control in the password generation process employed by many generative password managers can also introduce negative usability properties. Often the output of a generative password manager's site-specific password generation routine does not meet the password policy requirements of the website. If the website requires that passwords contain a certain number of characters that are numbers or symbols, or if the policy precludes the use of some characters, then the output of the generative password manager must be altered to respect these constraints. Similarly, the deterministic nature of the password generation may prevent the user from changing one site-specific password without changing the master secret. If the username used to authenticate with the website is not included in the generation process, or if the generation process does not salt the input, it may also be impossible to maintain more than one password for a given website (*i.e.,* to support multiple accounts) as all generated passwords for the domain will be identical.

2.4 Retrieval password managers

Retrieval password managers operate by prompting for, or benevolently capturing, a user's existing passwords as they are used. The passwords are then stored by the manager such that they can later be retrieved and used to log in to websites. Typically passwords are stored on permanent local storage alongside other site specific metadata such as the account username and the URL used to log in. The category of retrieval password managers can be further subdivided based on whether the stored passwords are encrypted or not. Storing many passwords all in one location could pose a security threat, offering an attractive target for an adversary. To alleviate these

concerns many retrieval password managers rely on a *master password* that must be provided to unlock (by means of decryption) the stored passwords. When the master password is entered it is used to derive a key used both to encrypt new passwords to be stored and to decrypt previously stored passwords. Some other retrieval password managers implement password encryption by using the user's operating system account password as the master password, or by saving passwords in a central operating system provided storage mechanism (often referred to as a *keyring*) locked to the user's operating system account.

In practice, as with generative password managers, retrieval password managers require the user only remember the master password in order to authenticate with all of the accounts registered in the manager. Retrieval password managers provide phishing protection by preventing password entry for associated accounts at the incorrect URL. When an account is imported into the retrieval password manager the associated URL is typically imported in addition to the username and password to allow auto-login behaviour when the manager detects the user is at the associated URL. If the user attempts to login (via the password manager) at a phishing website the URL of the browser will not match the URL stored in the manager for the targeted account. As with generative password managers any phishing protection relying on accurate domain names will remain vulnerable to pharming attacks [56] that manipulate DNS responses.

Many retrieval password managers are implemented as browser extensions [38, 1, 10, 7], capturing user account information as the user logs into the account for the first time. Automatically importing user passwords as they are used provides retrieval password managers with strong deployability advantages compared to generative password managers, which require an "all or nothing" adoption strategy. The usability gain is traded for decreased security, as user chosen passwords are likely to be much weaker than those generated by best-of-class software. In contrast to generative password managers, password reuse is possible within a retrieval password manager. This again aids in adoption and fosters feelings of control at the cost of overall security.

In the most naive case, a retrieval password manager need not protect (in the

sense of encryption) stored passwords at all [34]. In this case an adversary need only gain access to the retrieval password manager's storage[1] in order to compromise all associated accounts. This is the default behaviour of Firefox [48] for instance. Typically, retrieval password managers will protect, or offer to protect, the stored credentials with a master password.

Retrieval password managers protected with a master password have several drawbacks. First, many implementations do not use encryption correctly—many password wallets on mobile devices have been demonstrated to be insecure [3]. A second drawback is that a user-chosen password may not resist an offline attack if the wallet is stolen. If the adversary is able to capture the encrypted storage of the password manager they will be able to attack the master password directly, performing a dictionary or brute force attack. A human chosen master password may offer as few as 20 bits of security in practice [11]. To address this issue, Bojinov *et al.* [10] propose the use of password decoys to force the adversary back to using online attacks.

In practice, retrieval password managers protected by master passwords prompt the user for their master password only once per browsing session, with a session typically defined as lasting from when the browser application is first loaded until the application is explicitly closed by the user. Generally master password prompting will occur the first time the user requires access to any credentials protected by the manager. After entering the master password the protected passwords are decrypted and typically available in the clear for the duration of the browsing session. This behaviour improves usability by allowing the user to log into websites without entering their master password repetitively. This usability advantage comes at the cost of security, as it is not clear whether end users understand this *session model*, or when their passwords are in fact protected by encryption.

If the master password protecting the stored passwords is disclosed (accidentally or through malicious means such as phishing or malware) an adversary with access to the manager's storage will be able to access all of the associated accounts. Unlike generative password managers, knowledge of the master password alone is not sufficient to compromise the associated accounts. The adversary must somehow capture

[1] If the unencrypted retrieval password manager storage is located on an encrypted filesystem it must be acquired from the running machine while the unencrypted filesystem is mounted.

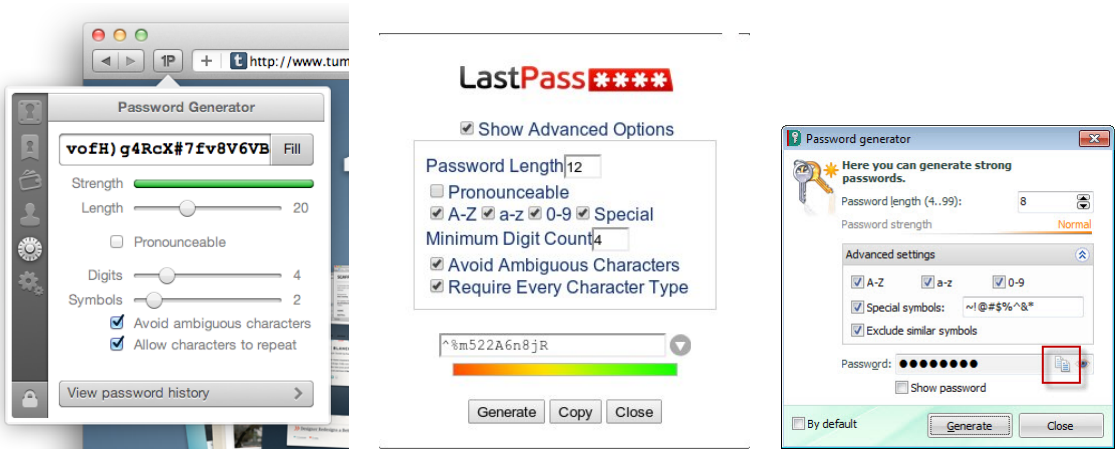
the ciphertext stored by the retrieval password manager in addition to the master password in order to learn site specific passwords.

As retrieval password managers rely on having access to a protected storage of user passwords to operate, they are by definition stateful. In addition to the manager software, the stored passwords must be present on all machines that the user wishes to authenticate from. This lack of portability may complicate authentication from more than one machine without support for a secure synchronization process. Cloud storage of managed credentials increases usability at the cost of security, allowing an adversary to target the trusted cloud provider to potentially access stored credentials for all of the users employing the service.

2.5 Auxiliary features of password managers

Password managers often provide a variety of auxillary features above and beyond the simple management of passwords. Commercial managers such as LastPass [38], 1Password [1], and Kaspersky Password Manager [36] (see Figure 2.1) use these features to differentiate their offering from competitors, providing additional value to attract consumers. These features, while useful, are not intrinsicly linked to the management of account credentials are thus considered to be auxiliary features.

Backups. Users of password managers are asked to place a large amount of trust in the password manager of their choice. Often without access to the password manager the user will no longer know the password associated with an account and must either return to the password manager or reset the password. To mediate concerns of data loss many tools allow the user to export their saved passwords, creating a backup that can be stored offline, and later used for import back into the password manager. If the backup is to remain secure it must be encrypted, for example with a key derived from the master password; unfortunately this prevents restoration from backup in the event the user has forgotten their master password. Backups that are not encrypted with the master password offer a disaster recovery mechanism for users that may have forgotten their master password, but must be carefully guarded. If the unencrypted backup is accessible to an adversary, the adversary is able to compromise all of the



(a) 1Password [1] (b) LastPass [38] (c) Kasperksy password manager [36]

Figure 2.1: Random-password generators.

accounts included in the backup. The process of creating a backup requires a variable amount of work depending on the tool. Some, such as Firefox, are entirely a manual effort.[2] Frequently the password manager tool will explicitly support the backup process, as is the case with LastPass[3] and 1Password.[4]

Synchronization. Another common feature requested by users of password managers is support for allowing authentication with stored credentials from any of the machines a user frequently employs in their day to day web browsing, not just one primary machine where the manager was first installed. Rather than require users install the password manager software and import a backup of registered passwords on all machines they wish to use, some password manager software allows cloud synchronization of password data. 1Password [1] indirectly allows synchronization by storing the password database in existing cloud file synchronization software (*e.g.*, Dropbox, Apple Cloud, *etc.*). LastPass [38] supports cloud synchronization directly, automatically sharing the encrypted password database with all installed copies of the password manager extension.

[2] `http://support.mozilla.org/en-US/kb/back-and-restore-information-firefox-profiles`
[3] `https://lastpass.com/support.php?cmd=showfaq&id=1206`
[4] `http://help.agilebits.com/1Password3/data_backup.html`

Random Password Generation. Some password managers address a common weakness with retrieval password managers by offering features available to improve upon the security characteristics inherent in a user's imported human-chosen passwords. One way is by offering generation of random passwords (in the sense of high-entropy). Frequently [38, 1, 51, 36] this is implemented in a way that allows the user to select parameters controlling the password generation that include length and the character classes (*e.g.,* letters, numbers, symbols) used during generation. This can be seen in the three representative screenshots in Figure 2.1. Allowing the user to select character classes enables the generated passwords to meet any password composition policies that may be required by the account provider for which the password is being generated. For both 1Password [1] and LastPass [38] the generators allow for pronounceable password generation (see Figure 2.1(a) and 2.1(b)), generating passwords that are easier to pronounce to aid password sharing and memory recall.

Password Quality Monitoring. In a similar vein some managers pro-actively monitor the passwords in use by the user in order to measure password quality. Potentially weak passwords are flagged and the user can be notified that these passwords should be replaced/strengthened. LastPass [38] additionally monitors for password reuse,[5] notifying the user when the same password is detected as being used across more than one website/account.

Untrusted Login. Untrusted computers pose a problem for password entry in general [43], but especially for many retrieval password managers, as keyboard logging malware can easily gain access to the master password. Some managers offer a means by which users can authenticate with the password manager without divulging their master password. These auxiliary authentication methods are designed to allow the user to access their stored passwords from a machine that they may not be comfortable typing their master password into. LastPass [38] implements this feature using one-time passwords[6] which can be printed out ahead of time and used as required.

[5] `http://blog.lastpass.com/2013/03/lastpass-now-warns-you-when-youre-using.html`
[6] `https://helpdesk.lastpass.com/security-options/one-time-passwords/`

Kaspersky Password Manager [36] opts instead to provide a virtual keyboard[7] which can be used to enter the master password via clicking on an on-screen keyboard, avoiding keyboard logging malware.

Secure Notes. While the main focus of almost all password managers is the storage of account credentials for websites, *i.e.,* a three tuple $\langle url, username, password \rangle$, some managers support secure storage of non-website credentials that do not have a URL (or potentially a username, *e.g.,* a door access code). Often this feature is presented as a "secure note" function.[8] LastPass [38] includes the notion of a *note type* presenting specific fields for credit cards, bank accounts, social security numbers, and WiFi access credentials (amongst others). Supporting secure storage of non-website related credentials allows the password manager to operate as a general secret storage tool.

Web Access. A related feature is web access, which can be used from computers not owned or maintained by the user without requiring any software to be installed. For generative password managers this can be implemented as a dedicated website capable of following the same site specific password generation as the password manager, by providing server-based generation rather than by software installed on the client machine. PwdHash [52] proposed web access in this fashion, allowing the user to enter a site URL, their username and their master password in order to generate the site specific password for the provided website. In the case of LastPass [38] a hosted website is offered to allow access to managed passwords using knowledge of the master password and a vanilla web browser by visiting `lastpass.com`. A synchronized encrypted copy of the LastPass password database is maintained by `lastpass.com` and decrypted when the user visits and provides the master password. In all cases use of web access to managed passwords (or generated passwords for a generative password manager) requires trust in the web site not to divulge the master password or any generated/retrieved credentials. In the case of a retrieval password manager the security impact of inadvertent disclosure of the master password (*e.g.,*

[7] `http://support.kaspersky.com/5059`

[8] `https://helpdesk.lastpass.com/password-manager-basics/secure-notes/`

to malware or a phishing site) is amplified by web access, removing the requirement that an adversary capture the stored credentials in addition to the master password.

2.6 Comparison and taxonomy of password managers

The category of password managers is broad and contains many different, generally complimentary, techniques. Examples of security-related services they provide are password strengthening through iterated hashing [27, 52, 14], phishing protection through site-specific passwords [52, 59], and providing strong passwords derived from non-password input [42, 54, 8]. A taxonomy of password managers is presented in this section to help distinguish between the approaches used.

The major distinguishing factor between password managers is whether individual passwords are: 1) stored and maintained by the password manager, or 2) generated on-demand. We distinguish these two approaches by calling the former *retrieval password managers* and the latter *generative password managers*. Figure 2.2 shows a visual breakdown of the password manager taxonomy we present.

The category of generative password managers is further subdivided into those that use a text-based password and those that use a non-text password. Non-text passwords are divided into graphical password and digital object passwords. The category of retrieval password managers is subdivided into those that use encrypted storage and those that do not. Encrypted storage is subdivided into categories for master password encryption password managers, login password and keyring password managers, and dual-possession authentication password managers.

Interestingly, outside of academia we see no generative password managers, supporting the belief that the adoption cost (*i.e.,* changing all existing passwords one-by-one) is too high for use in practical settings. Examining the auxiliary features of password managers in Section 2.5 almost all are found only in retrieval password managers and could not be implemented for a generative password manager (*e.g.,* Secure Notes, Untrusted Login, Password Quality Monitoring, and Backups). Some, like quality monitoring and backups, directly address issues of registered account password quality and state transfer not relevant in generative password managers.

Most (if not all) publicly available password managers conform to a retrieval style

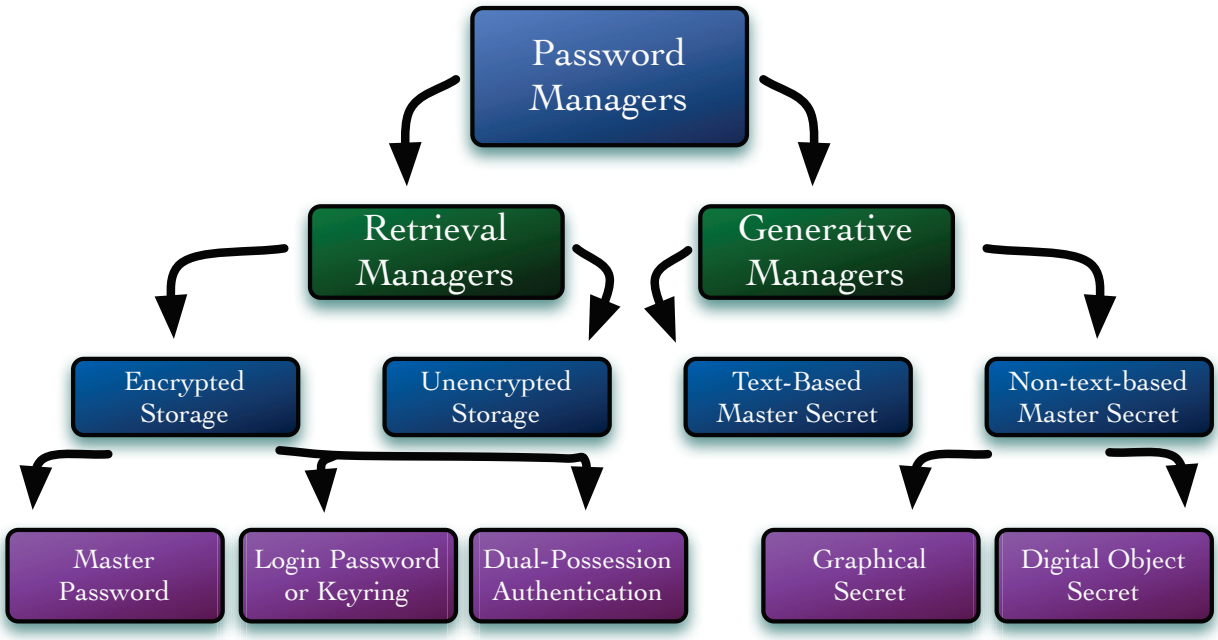


Figure 2.2: A Taxonomy of password managers. Divided into two main categories: generative password managers and retrieval password managers, each with respective subcategories.

of password management. Providing users a means to import their existing passwords for a gradual transition to the password manager decreases the initial cost of adoption at the price of decreased security. Given the goals of the academic security community (mainly, improving security) it is not surprising that the focus of academic password manager work is largely on the side of generative password managers. Conversely, retrieval password managers emphasize usability and simplicity, making them a more appealing choice to end users. Further, modern browsers all support a retrieval password manager without requiring any additional software to be installed, making a retrieval password manager available at the user's fingertips whether they are aware of it or not.

2.7 Related work

2.7.1 Password managers

PwdHash. An early generative password manager, PwdHash [52] is implemented as a browser extension for both Internet Explorer [47] and Firefox [48]. PwdHash uses the general notion of a generative password manager and describes challenges encountered in implementing the password derivation inside a web browser. The authors specifically address concerns related to phishing, malicious JavaScript, and the difficulty of encoding derived passwords such that they meet the individual requirements of various password composition policies. The paper itself can also be viewed as a tutorial on how to avoid JavaScript keyloggers.

PwdHash generates individual passwords using the following construction:

$$hash(pwd, dom) = PRF_{pwd}(dom) \tag{2.1}$$

Here *pwd* is the user's master password, *dom* is the domain of the site for which the password will be generated, and PRF is a pseudo random function [23]. In this construction *dom* is used as the salt, precluding easy generation of multiple passwords for one website, or the use of unique passwords for multiple accounts on one domain, and *pwd* is the (secret) seed for the pseudo-random function. To activate the PwdHash software application, the user must first enter a *Password Prefix* which engages defenses against JavasScript attacks and enables a user to authenticate using a generated password. While a small usability study is presented by the authors, a more in-depth follow up by Chiasson *et al.* [16] found major usability issues with PwdHash as implemented in the academic work [52]. Many users were unable to correctly operate PwdHash, in some cases resulting in disclosure of their master password.

Password Multiplier. Another generative password manager, Password Multiplier [27] is similar in both design and implementation to PwdHash [52]. The authors present Password Multiplier and discuss many of the benefits of generative password managers, *e.g.,* portability, reduced password reuse, increased password quality, *etc.*. In contrast to PwdHash, Password Multiplier is only implemented for Firefox [48].

PasswordMultiplier generates individual passwords using the following construction:

$$V = f^{k_1}(username : master\_password) \tag{2.2}$$

$$site\_password = f^{k_2}(site\_name : master\_password : V) \tag{2.3}$$

Here V is an intermediate variable that may be computed once and cached on the end-user device, $f(x)$ is a secure hash function, and k_1, k_2 are the number of iterations to be used for V and *site_password* respectively, while ':' denotes concatenation. Unlike PwdHash, PasswordMultiplier considers the username to be the salt, and not the site name. A primary differentiating factor between PwdHash and PasswordMultiplier is the introduction of the k_i parameters, an iterated hashing scheme, and two-step computation (where one computation is performed infrequently).

With PwdHash an adversary capable of gaining access to one site-specific password can perform an offline brute force attack on the master password, potentially gaining access to all other passwords. In this case the adversary is limited only by the time cost of the hashing algorithm. Password Multiplier uses iterated hashing algorithms and recommends an iteration count that requires about 100 seconds for the generation of V and a tenth of a second for the generation of each *site_password*. This small increase in computation time (for *site_password*) for the user will be transparent, but greatly impede an attacker's ability to quickly try candidate master passwords.

Passpet. Utilizing the same site specific generation procedure as PasswordMultiplier [27] (see Equation 2.2 and 2.3), the Passpet password manager [59] aims to further improve the usability and phishing resistance of generative password managers. Passpet relies on user supplied site-specific labels and a custom UI in order to prevent phishing attacks. Passpet addresses some of the usability concerns related to master password entry that resulted in both both PwdHash [52] and PasswordMultiplier [27] having unsafe usage [16] in empirical trials.

While Passpet uses the same site specific password generation procedure as PasswordMultiplier [27] it does not fix the k_1 value. Instead the number of iterations of the hash algorithm applied to generate V is increased automatically on the master password generation screen. Interactive visual feedback is given as the number of

iterations is increased, encouraging users to wait at the generation screen for a longer period of time to achieve greater security. The generation screen updates in real-time displaying to the user the estimated time to crack the master password given the current iteration value.

ObPwd. While most generative password managers use a master password as the secret that is *stretched* or *strengthened* into site-specific passwords, ObPwd [9, 44] uses digital objects. A digital object is a file chosen by the user from a collection of digitally represented files such a photos, music, or videos. These object files may be stored locally, or accessed remotely. Avoiding text typing during password entry makes ObPwd especially well suited to smartphones and other devices with limited text input [44]. ObPwd generates passwords using a user's own files. By relying on the inherent meaning and attachment a user has with digital objects such as photos, ObPwd is asserted [9] to be a usable, secure alternative to manual password creation.

ObPwd generates individual passwords using the following construction:[9]

$$site\_password = Base64(SHA1(objectBytes : domain)) \tag{2.4}$$

Here $objectBytes$ are the bytes of the digital object (generally constrained to $160 < byte\_length\_of(objectBytes) < 100000$), and $domain$ is an optional salt.

Usability testing [9] of both the desktop ObPwd and the mobile implementation show strong user affectation. Users were shown to be able to successfully log in to websites with ObPwd without suffering performance or security related issues.

gridWordX, iPMAN, and GPEX. Within the password manager taxonomy (see Section 2.6) gridWordX [17], iPMAN [6], and GPEX [7] all fall within the non-text-based secret generative password manager category – specifically, the graphical secret subcategory. GPEX is a generative password manager that converts a click-based graphical password into site specific passwords. iPMAN does the same using an icon-based graphical password. Lastly, gridWordX has the user choose from a grid of text words to create a passphrase without requiring character-by-character text entry.

[9] Notation simplified to match other related work.

Kamouflage. In contrast to the academic offerings in the generative password manager space, Kamouflage [10] is one that is a retrieval password manager. The primary goal of Kamouflage is to prevent offline attacks both against the master password and the stored passwords. The intent is to do this by removing an attacker's ability to verify candidate master passwords offline by checking whether a guessed candidate successfully decrypts the stored passwords. In contrast to traditional retrieval password managers, Kamouflage stores one true password database (S_0) in addition to a large set of decoy password databases ($S_1, ..., S_N$). An adversary that is able to steal the Kamouflage databases must perform on average $N/2$ online attempts to verify the master password (where the authors suggest typical N values to be approximately 10,000).

Much of the Kamouflage proposal deals with the difficulty of generating believable decoy password sets that will not leak any information useful to an adversary for verification purposes. The decoy passwords must appear to be human memorable, and not truly random. Further, human passwords are often related to one another, another property that should be maintained for realism. A proof of concept implementation for Firefox [48] is discussed, detailing Kamouflage's implementation of the `nsILoginManagerStorage` interface[10] provided by Firefox to aid in implementing third party password managers (*i.e.,* password managers built to replace the built in browser manager).

Browser Managers. Firefox [48], Chrome [25], Internet Explorer [47], and Safari [2] all offer some form of built-in password management. It is not clear whether users are aware that the *"save password"* functionality many rely on is in fact a password manager. All of the major browsers implement a retrieval password manager for password management. The browser managers offer few of the auxiliary features discussed in Section 2.5. For example, none of the above four browser password managers offer random password generation, password quality monitoring, or untrusted login (*i.e.,* features for login from untrusted machines). Interestingly, the protection of the stored passwords is dependent not only on the choice of web browser, but also

[10] `https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsILoginManagerStorage`

on the operating system it is installed on.

On a Microsoft Windows machine Internet Explorer [47], Safari [2], and Chrome [25] default to encrypting the stored passwords using the Windows Data Protection API (DPAPI).[11] The encryption used by DPAPI is keyed using the user's Windows account password, allowing decryption without password prompts when the user is logged in to the OS. This provides a seamless encryption process that protects passwords stored by IE, Safari and Chrome from being accessed in plaintext from the on-disk password database by default, without requiring the user set a master password for the manager.

A consequence to using DPAPI is that once the user has logged into their operating system account their stored passwords are readily available to an adversary capable of gaining access to the account, or to a user who finds the machine in an unlocked state. A roommate or spouse that finds the computer unlocked may simply open the web browser and immediately use and access all of the encrypted credentials without having to revalidate knowledge of the user's account password or a master password for the manager. This provides better usability, at some loss in security relative to such threat vectors. DPAPI has also been shown vulnerable to offline attack [50], further weakening the security of browser password managers that rely on it for encrypting the password database. Additionally the design of DPAPI may allow other user-installed applications access to the encrypted credentials saved by the browser managers using DPAPI, potentially decreasing the difficulty of writing password stealing malware.

On an OSX machine both Safari [2] and Chrome [25] rely on the OSX Keychain service[12] to securely store passwords to be managed. Similar to DPAPI, in the default configuration the password used to unlock the keychain is the same as the user's login password. The keychain is unlocked when the user logs into their OSX user account on the machine. While possible to change the password used to unlock the keychain such that it is not equal to your account password, these steps must be manually undertaken and are therefore unlikely to be done by average users. The keychain

[11] `http://msdn.microsoft.com/en-us/library/ms995355.aspx`
[12] `http://pdox.ca/osxkeychain`

integration chosen by Safari and Chrome for the password manager storage is vulnerable to the same issues mentioned for DPAPI on Windows — mainly unattended access by a casual passer by and inter-application access.

On a Linux machine, Chrome [25] operates similarly to how it does on OSX. This design is complicated by the fragmentation of Linux desktop environments (and by association, Keychain provider applications). Chrome automatically determines whether it should integrate with GNOME Keyring,[13] KWallet,[14] or store passwords unencrypted based on the desktop environment in use and the Keychain software availability.[15] Notably, if the user does not have GNOME Keyring or KWallet installed the Chrome manager defaults to storing passwords unencrypted.

Mozilla Firefox [48] is the only browser with a password manager that operates identically across all operating systems. By default, the Firefox password manager does not have a master password and does not encrypt the user's stored credentials. An adversary capable of accessing the on-disk representation of the password database with user-level permissions can compromise all managed accounts. Similar to DPAPI and Keychain integration, if the user's browser is accessible in an unlocked desktop session a passer by can access and utilize the stored credentials without having to enter any password or authenticate with the manager.

Firefox does allow for users to elect to set a master password.[16] If the master password is set then all stored credentials are encrypted using a triple-DES key derived from that password.[17] After a master password is set the user must enter the password the first time within a session that any encrypted credentials are accessed. As setting a master password is not mandatory and must be undertaken under the user's own volition it is not clear how often a master password is used in practice.

LastPass. The LastPass [38] password manager is a full-featured commercial password manager with both free and paid versions available. LastPass may be installed as a browser plugin for all of the major browsers on Linux, OSX and Windows or

[13] `https://live.gnome.org/GnomeKeyring`
[14] `http://utils.kde.org/projects/kwalletmanager/`
[15] `https://code.google.com/p/chromium/wiki/LinuxPasswordStorage`
[16] `http://support.mozilla.org/en-US/kb/use-master-password-protect-stored-logins`
[17] `http://support.mozilla.org/en-US/questions/824120`

can be accessed in a web browser through the hosted `LastPass.com` cloud panel. LastPass is implemented as a retrieval password manager, and mandates the use of a master password by default.

LastPass supports many of the auxiliary features discussed in Section 2.5 including cloud synchronization, encrypted backups, one-time passwords, and web or *cloud* access. LastPass pro-actively offers to generate random high-entropy passwords for the user but does not force users to use randomly generated passwords. LastPass also monitors the quality and reuse of the user's existing passwords suggesting the user replace passwords found to be weak or shared between multiple accounts.

Prior to implementing Password Based Key Derivation Function 2 (PBKDF2) [33] the LastPass password database was not resistant to offline attack [3]. It has since switched to using 1000 iterations of PBKDF2 on the master password to derive the database encryption key. Metadata is encrypted, but LastPass does not use an authenticated ciphermode.

1Password. The 1Password [1] password manager is a competitor to LastPass [38] in the commercial password manager space. 1Password has both a free 30 day trial and a full version available for a fixed cost ($49.99 USD as of May 2013). 1Password is available on OSX, Windows, iOS and Android.

The auxiliary features offered by 1Password overlap with LastPass significantly. Both offer many of the same features related to password generation, backups, and secure notes. Notably 1Password highlights giving users a greater level of control over their data by default by not requiring cloud synchronization but instead making it an optional opt-in feature. This is in contrast to LastPass which synchronizes the encrypted password database to LastPass.com for web/cloud access.

While too numerous to cover exhaustively, many other password managers exist (*e.g.,* KeePass,[18] Roboform[19]) that are not discussed here. Most frequently these other retrieval password managers operate near identically to LastPass and 1Password at their core (*e.g.,* master password protected encrypted password storage and retrieval).

[18] `http://keepass.info/`
[19] `http://www.roboform.com/`

2.7.2 Usability evaluations and comparison frameworks

A number of papers have compared password managers through surveys and user studies. Gaw and Felten [21] surveyed users on password use and found few users employed a password manager, instead of relying on memory alone. Chiasson *et al.* [16] examined two password managers, finding significant usability and security failings related to entry of the master password as well as inaccurate/incomplete mental models of the software. Bicakci *et al.* [5] examined the user interface of browser-based managers and the tendency of users to inadvertently save private information on a public computer.

Karole *et al.* [35] performed a comparative user study between an online, a mobile, and a portable-USB password manager. They found non-technical users preferred keeping their credentials on mobile phone based password managers, but had difficulty entering passwords of sufficient strength on the mobile device.

Biddle *et al.* [9] examine the use of digital objects, *e.g.,* desktop files such as photos and MP3s, as passwords. They introduce a novel generative password manager based on digital objects (ObPwd [9]) and evaluate it using a 30 participant user study. Their results indicated a positive user experience using ObPwd and performance in line with existing schemes.

Bonneau *et al.* [12] propose the UDS framework for evaluating authentication solutions based on usability, security and deployability properties. They rank 35 representative schemes (including 2 password managers: Firefox [48] and LastPass [38]), but do not themselves carry out any user studies. We evaluate a variety of password managers in addition to our own proposal, Tapas [45], using this UDS framework as extended in Chapter 5.

2.7.3 Device-based authentication

Several papers have explored *device-based authentication*, *i.e.,* authentication (password or otherwise) aided by possession of a device (*i.e.,* something you have); we restrict our coverage of these primarily to such mechanisms involving possession of a smartphone. With dual-factor authentication, a secondary factor (often a hardware

token) is required in addition to a password. One use of a smartphone in authentication is to have the smartphone generate such tokens (*e.g.,* Google Authenticator[20]) or to receive them *e.g.,* as one-time passwords over SMS. Phoolproof [49] uses a smartphone as an authentication token to augment traditional password authentication with the goal of preventing phishing through the use of public key cryptography and end-to-end TLS. Pico [57] is an unimplemented proposal suggesting the use of a cluster of devices, including smartphones and other smart devices, in proximity to each other to allow authentication. All of these require server-side changes.

The use of untappable ("out-of-band" or OOB) channels for authentication has been explored in the Seeing-is-Believing [46] work in which 2D barcodes are utilized for a one-way authentication process. Further work extends the use of OOB channels for two-way authentication [53] and explores minimal sensor implementations that do not rely on high fidelity camera hardware. Exploration into the use of OOB pairing procedures by disabled users [54] indicates the use of phone-based pairing, while not applicable to blind users, may of great use to other disabled users due to the high availability and accessibility offered by the phones.

Using mobile devices to aid in secure password reset was examined in Mercury [41]. Employing a trusted mobile device to allow safe authentication using an untrusted terminal was the basis of MP-Auth [43], Session Juggler [15], and Phoolproof [49]. Each of these approaches (except for Session Juggler) require server-side changes. Additional evaluation of device-aided authentication was provided by Bonneau *et al.* [12].

[20] `http://code.google.com/p/google-authenticator/`

Chapter 3

Tapas Password Manager

Tapas is a retrieval password manager designed to offer password management with resilient encrypted password storage, without requiring the burden of a master password. Tapas aims to be a highly usable password manager supporting gradual adoption of existing user chosen passwords and near-effortless login. At a high level Tapas associates the user's desktop with a smartphone, sharing the responsibility of password management with both devices. Saving account details and later retrieving these details for use in authentication is done in-browser on the user's desktop, but must be confirmed through touch input (tapping) on the smartphone.

By securely introducing the user's smartphone and desktop computer via an out-of-band channel (*i.e.,* pairing) the two may exchange data over an insecure network in the future with both privacy and authentication. In general terms Tapas splits the storage and use of a user's credentials, requiring simultaneous possession of the desktop PC and the paired smartphone for operation. Passwords to be stored are encrypted by the desktop PC and securely communicated to the smartphone for storage. A user initiated tap gesture on the smartphone returns the stored credentials to the desktop computer for use. The design of Tapas first appeared as a refereed publication [45].[1]

Before examining the Tapas password manager in detail we first present the constraints and motivations that influenced the overall design. We created Tapas by first establishing a core set of motivating problems and then devising a solution that would solve these problems within the constraints we identified.

[1]While the author of this thesis lead the Tapas project, contributions on the project and subsequent refereed publication are graciously acknowledged. The in-lab user study for Tapas was designed in collaboratio with David Barrera, who carried out the study. The security analysis and security protocols for dual-possession authentication was aided by Jeremy Clark.

3.1 Design motivations and requirements

Many alternative authentication schemes require server-side changes in order to be supported [49, 43, 57, 15, 18]. This prevents security conscious users from adopting the scheme independent of the services they wish to authenticate with, since such users alone can not influence what is supported by webservers. Adoption related issues are further compounded by the large number of accounts each user must maintain (*e.g.,* about 25 providers per one study [20]). It is clear we can't expect widespread adoption of an authentication technology requiring buy-in from all account providers. This restriction works against outright replacement of passwords, as they are the only authentication method offered by the vast majority of service providers. Even augmenting passwords with a separate authentication factor (*e.g.,* two-factor authentication) is challenging as the server must be modified to be aware of the additional factors. We restrict our attention in this thesis to discussion of authentication solutions that can be adopted independent of service providers.

Password managers avoid requiring server-side changes, instead producing a password to be used for authentication on behalf of a user. By removing the constraint that a user remember each of their account passwords, a password manager enables strong unique per-account passwords. As the sole method of authentication remains a password, no server-side changes must be made to support additional factors or novel authentication technology. This does constrain the security improvements that can be made by a password manager as they may improve upon the basic situation where the stored passwords may be subject to attack during transmission to the server (in the absence of SSL) or at the service provider (online or offline). We chose to implement Tapas as a password manager opting to favour the deployability advantages of not requiring server-side changes over the open-ended clean-slate design flexibility and security advantages that server-side modifications might otherwise allow.

A unified collection of user authentication credentials makes for an attractive target for an adversary. In the naive case where a password manager does nothing to protect the managed credentials an adversary need only gain access to the stored password collection in order to compromise all managed accounts. Our proposed solution acknowledges the importance of adequately protecting the stored credentials; Tapas

is resistant to mobile device theft, and protects stored credentials from unauthorized access or brute-force attack.

Many password managers [10, 38, 25, 47, 2, 36] protect the user's stored credentials by using a master password to generate or unlock a symmetric key used to encrypt the passwords. This reduces the number of passwords a user has to remember from n to 1, but provides no aid in remembering the master password. Further, if the user chooses a poor master password the stored credentials may be vulnerable to an offline dictionary attack. Disclosure of the master password, either accidentally or through a phishing attack, could potentially nullify the protection of stored credentials, making it a single point of failure. In light of these drawbacks we set as a design requirement that stored credentials must be protected by encryption without requiring the use of a master password.

Our design was also motivated by the prevalence of smartphones as a nearly ubiquitous secondary computing device. As users increasingly carry a smartphone, designing a solution that assumes this device is present was determined to be a reasonable constraint.

3.2 Dual-possession authentication

As opposed to remembering passwords in your head — which is generally called "something you know" — storing passwords, whether software-based or by a post-it note with passwords written on it, is based on the principle of authentication by "something you have": the contents of the password 'wallet.' The primary security vulnerability of an unprotected wallet is theft. This is traditionally addressed by adding a master password, something you know, for additional protection. However this protection is best considered a deterrent, as theft allows offline attacks on such a master password. Given a user-chosen master password this may mean fewer than 20 bits of security [11]. By contrast, password management that requires simultaneous access to multiple paired devices offers a level of theft-resistance.

Strictly speaking, requiring access to multiple paired devices is not dual-*factor* authentication because the factors are of the same type (*i.e.,* something that you have). *Dual-possession authentication*, as we define it, involves two applications, a Manager

and a Wallet, on different devices. The Manager acts to retrieve passwords to be provided to a service while the Wallet acts as the password storage. A dual-possession authentication implementation offers the three following protocols for managing the passwords: Pair (Protocol 1), Store (Protocol 2), and Retrieve (Protocol 3). These protocols are designed to achieve a relatively simple goal: by stealing the data of either the Manager or the Wallet, an adversary cannot determine the stored password for any given account with any greater success than attacking the account directly. This is achieved by encrypting each password with a key held by the Manager and storing the resulting ciphertext on the Wallet. By stealing the Manager, the adversary obtains the decryption key but not the ciphertexts to decrypt, and by stealing the Wallet, the adversary only has a set of ciphertexts resistant to offline attacks.[2] The effect of malware which remains resident on the Manager is discussed in Section 3.4.

To ensure these devices can run Store and Retrieve over a potentially hostile network, we require Pair to be performed on an authenticated and secret out-of-band (AS-OOB) channel [28]. The pairing is essentially an assignment of public keys that will be used by each device to authenticate the other during network communication. In Tapas we instantiate the AS-OOB channel by having the Manager display a QR code to be scanned by the Wallet containing a public key for the Manager and a public and private keypair for the Wallet. Once paired, the devices will establish a mutually-authenticated end-to-end secure channel (*e.g.,* TLS with a Diffie-Hellman key exchange[3]) before exchanging any encrypted passwords. This allows the devices to securely tunnel their communication through various network devices that may assist them in establishing a connection.

3.3 Tapas

We instantiate the protocols and general notion of dual-possession authentication (Section 3.2) to construct Tapas. In Tapas, password management is handled across both the user's desktop PC and a paired smartphone. In this Section we describe the

[2]For two devices, this approach seems more straight-forward than using distributed/threshold decryption with key shares.

[3]The RSA-based key exchange in TLS does not provide perfect forward secrecy, which is necessary for security as discussed in Section 3.4.

Protocol 1: Pairing Manager and Wallet

User action: Upon a user choosing to set-up a new Wallet, the following protocol is initiated by the Manager.

Communication channel: A one-way authenticated and secret out-of-band (AS-OOB) channel from the Manager to the Wallet.

1. The Manager generates an authentication key pair for itself $\langle pk_m, sk_m \rangle$ and sends its public key pk_m to the Wallet.
2. The Manager generates an authentication key pair for the Wallet $\langle pk_w, sk_w \rangle$ and sends the pair to the Wallet.
3. The Manager generates a symmetric secret key k for an authenticated encryption scheme $\mathsf{Enc}_k()$.

Output: The Manager stores $\langle pk_m, pk_w, sk_m, k \rangle$ and erases sk_w. The Wallet stores $\langle pk_m, pk_w, sk_w \rangle$.

implementation details of Tapas, and explain how the 3 protocols of dual-possession authentication are implemented.

While we have chosen to implement the components of Tapas using Mozilla Firefox [48] and the Google Android [24] platform, the architecture is independent of these choices. We expect that an extension for Chrome [25], Safari [2], and other extensible browsers could be developed for users who do not use Firefox as their primary browser. Similarly, non-Android smartphone platforms could be used. In fact, following our description of Tapas as one instance of a dual-possession authentication scheme, the second device need not be a smartphone at all. New classes of consumer electronics such as smart watches, or heads-up-display glasses could be used as the second device in place of a smartphone, in a manner similar to Pico [57].

Our initial prototype Tapas implementation supports one computer and one smartphone, a decision made to reduce the complexity of the initial implementation. We viewed this restriction as acceptable based on our observations that (1) users generally have a small set of computers they use to log in to online services; and (2) the proliferation of smartphones means that many users already carry one of these devices, so for these users there is little extra cost. The first observation is echoed by existing user studies [29] finding near 100% of password events occur on either the users' home or work machines. In a comparative usability study of online, desktop

Protocol 2: Storing a Password

User action: Upon a user choosing to save a password p_i, the following protocol is initiated by the Manager.

Communication channel: A mutually-authenticated secure channel with perfect forward secrecy between the Manager and the Wallet, who respectively identify themselves with pk_m and pk_w.

1. The Manager takes user password p_i (entered by user) and site information s_i and computes $c_i = \mathsf{Enc}_k(p_i : s_i)$. **Note:** ":" denotes concatenation.
2. The Manager sends $\langle c_i, s_i \rangle$ to the Wallet.
3. The Wallet prompts the user to create a tag t_i for referencing the site, using s_i to suggest a value for the tag.

Output: The Manager erases $\langle p_i, s_i, c_i \rangle$. The Wallet stores $\langle t_i, c_i \rangle$ and erases s_i.

and mobile password managers [35] users were found to prefer the mobile phone based managers citing a greater feeling of control. These findings help support our choice to employ the user's smartphone in the password management process.

3.3.1 Components of Tapas

Firefox Extension. In our implementation, the Manager device is implemented as a Firefox browser extension on the users's desktop PC. It is written in Javascript and XML User Interface Language (XUL), utilizing interfaces exposed by Firefox for use by extensions. Internally Firefox relies on the Network Security Services (NSS) library[4] to implement SSL/TLS and all cryptographic primitives. Since NSS is not exposed to browser extensions Tapas includes a Javascript *ctypes* wrapper for the portions of NSS required to implement symmetric encryption/decryption, public key generation and mutually authenticated SSL. The Tapas extension is multi-platform and requires no native code that is not included in the Firefox installation. The extension can be installed on Windows, Linux, OSX and all other platforms supported by Firefox.

Android Application. The Wallet device is implemented as an application for the Android smartphone platform. We emphasize that this is a password Wallet and has

[4] `http://www.mozilla.org/projects/security/pki/nss/`

Protocol 3: Retrieving a Password

User action: Upon a user choosing a password for retrieval, the following protocol is initiated by the Wallet.

Communication channel: A mutually-authenticated secure channel with perfect forward secrecy between the Manager and the Wallet, who respectively identify themselves with pk_m and pk_w.

1. The Wallet retrieves the c_i value associated with the tapped t_i, and sends c_i to the Manager.
2. The Manager decrypts and authenticates c_i to retrieve s_i and p_i.
3. The Manager checks that s_i matches the site information for the current site that the browser is visiting.
4. The Manager transfers the user password p_i to the site over the connection to the webserver.

Output: The Manager erases $\langle p_i, s_i, c_i \rangle$.

no relation to mobile payment functionality. Our prototype Wallet is written using Java for devices running Android versions 2.3 and above. Based on platform distribution statistics[5] Tapas is compatible with over 95% of Android devices worldwide (as of June, 2013). The Tapas application operates with minimum privileges, requesting only `INTERNET`, `WAKE_LOCK` and `C2DM RECEIVE` permissions.

Rendezvous Server. In order to allow direct communication between two devices potentially located on separate networks, the Tapas architecture employs a hosted server application we refer to as the Rendezvous Server to facilitate network address translation (NAT) traversal and hole punching. This allows the Manager and the Wallet to communicate even when associated with different networks, or behind a firewall/NAT translation. Each client makes an outbound request to the public Rendezvous Server which negotiates a direct connection between the two clients. The Rendezvous Server is considered untrusted and external to the management of passwords; no unprotected data is transmitted through it.

In addition to negotiating network connections, the Rendezvous Server is responsible for federating communication with the Google Cloud to Device Messaging (C2DM)

[5]Google platform versions distribution: `http://goo.gl/rQ2gv`

service. Google requires all applications utilizing C2DM to pre-register with the service to obtain an API authentication token allowing access to the service. In order to avoid embedding a C2DM API token into the Manager extension we defer C2DM pushes to the Rendezvous Server, allowing the Manager to send a push message to the paired device without requiring the C2DM token. Tapas relies on C2DM strictly as a means of launching the Wallet application automatically without requiring a long-running listener service on the smartphone. To avoid reliance on Google, alternative implementations can rely on a perpetual service listening for Wallet import events on the network. When a request is received the service can launch the Wallet automatically acting as a functional replacement for the Android-specific C2DM service at the cost of slightly higher battery usage (from running the persistent service).

Design Alternatives. Rather than implement the Wallet on a smartphone one could imagine an alternative design instead opting for a USB Memory Stick (*e.g., thumb-drive*) based Wallet. While at first glance this appears to provide a simpler solution, such a USB-based design suffers from at least two limitations. First, while plugged into the desktop machine a USB Memory Stick based Wallet exposes the entire collection of encrypted passwords to the host machine. In comparison, the smartphone Wallet used in Tapas discloses passwords to the desktop machine one-by-one on a per-use basis, providing some security against malware (see Section 3.4, *Resistance to Malware*). Second, while we could encourage users to remove the USB Memory Stick from the system when not in use it is extremely likely that many users would leave the device plugged in persistently in order to ease usability. In comparison, Tapas communicates over the network, removing the need for the user to remember to unmount or securely disconnect Tapas from the host system when not in use.

3.3.2 Setup

To set up Tapas, the user installs the Firefox extension and the Android app using the standard software installation procedure for each respective platform. Once installed, the devices are paired using Protocol 1. The Manager generates the authentication

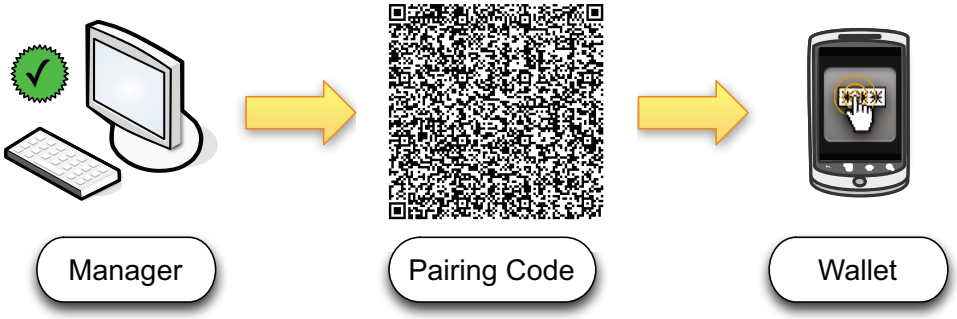


Figure 3.1: Setting up an out-of-band communication channel initiated (depicted by the checkmark) by the Manager, for pairing the devices. The checkmark indicates which device initiates the procedure.

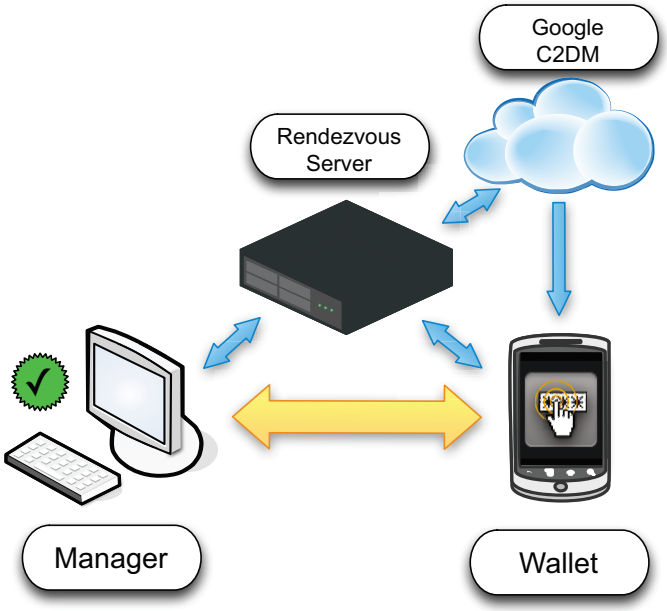


Figure 3.2: Setting up a two-way network communication channel initiated by the Manager, for password storage on the Wallet. The checkmark indicates which device initiates the procedure.

key pairs and a self-signed TLS certificate for both public keys. It embeds networking information (IP address and port number), a fingerprint of its own certificate, and the Wallet's certificate and corresponding secret key into a QR code. This QR code is displayed on the computer screen, forming a unidirectional AS-OOB channel (Figure

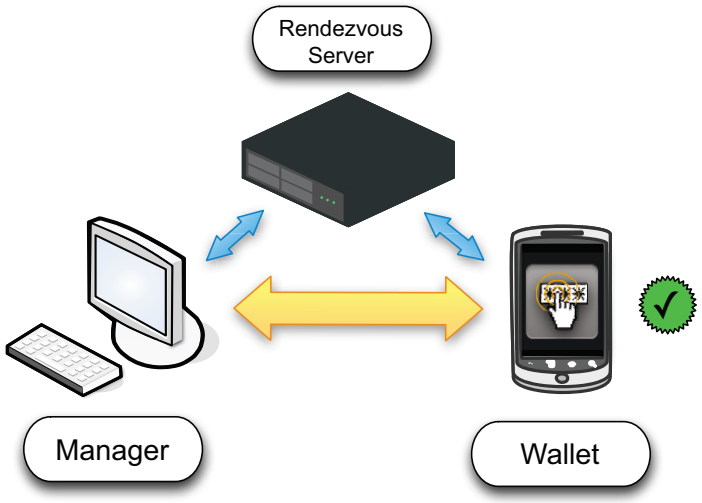


Figure 3.3: Setting up a two-way network communication channel initiated by the Wallet, for password retrieval from the Wallet. The checkmark indicates which device initiates the procedure.

3.1). It is thus assumed that an adversary does not have visibile access to this screen.

The user now opens the Wallet app on the smartphone. The Android app defaults to displaying a "begin pairing" screen until the user successfully pairs (*i.e.,* establishes secure mutually authenticated communication channel) the application with an instance of the Firefox browser extension. When the user presses the "pair" button on the Wallet app a QR code scan is initiated via the ZXing QR Code application.[6] ZXing utilizes vibration and auditory feedback to inform the user when a code has been successfully scanned in order to help make the process intuitive. After the Wallet app reads the QR code, the Wallet decodes from it the IP address and listening port of the Manager browser extension as well as the certificate material. Subsequent communication between the Wallet and the Manager occurs over the network through a mutually authenticated SSL channel.

[6] `http://code.google.com/p/zxing/`

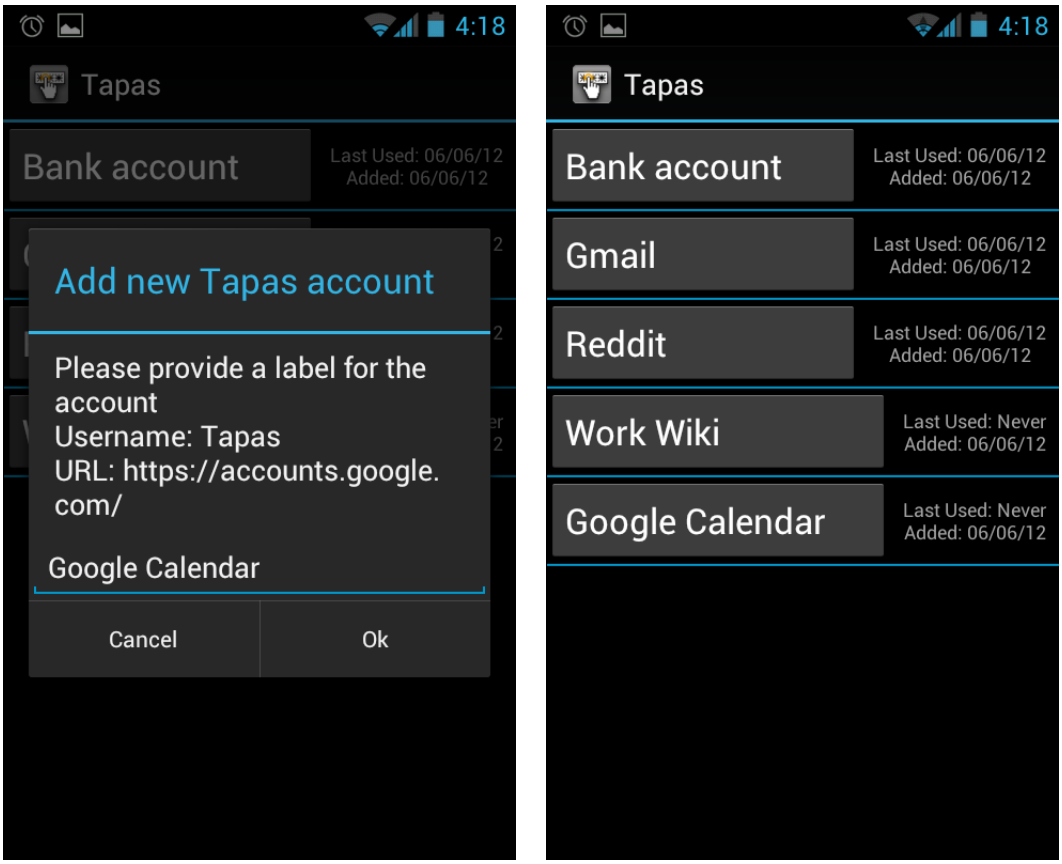
3.3.3 Account Import

Login events are detected using the Firefox `nsILoginManager` API[7] as well as some simple heuristics looking for HTML login form elements. When the Manager detects a username/password being submitted to a website, it temporarily saves the values as they are submitted and offers the user a chance to store the account credentials in the Wallet. This is done by presenting a non-obtrusive drop-down notification similar to those used by the built-in Firefox password manager. If the user accepts the offer then the Manager stores the credentials in the Wallet as follows: the Manager contacts the Rendezvous Server to initiate a C2DM push to launch the Wallet application on the paired smartphone (Figure 3.2). The smaller arrows in the figure represent communication used to launch the Wallet automatically (via C2DM and the Rendezvous Server) and to negotiate a direct network connection between the Manager and the Wallet (the direction connection itself is pictured as the larger arrow). Both the Manager and the Wallet rely on outgoing connections to the Rendezvous Server to negotiate direct communication through NAT, similar to traditional NAT hole punching techniques involving a third party.

At this point the Manager and Wallet follow Protocol 2 to securely transfer encrypted credentials. First the Manager encrypts the site information (URL, username, password) using AES in GCM mode with a symmetric encryption key known only to it. The encrypted ciphertext is then transmitted from the Manager to the Wallet over a mutually-authenticated TLS connection (using a Diffie-Hellman ciphersuite) where both certificates are pinned to the device certificates previously established during pairing. Data communicated between the Manager and Wallet is encapsulated in a simple JSON notation to provide structure to the data and allow for easy serialization/de-serialization.

When new account information is transmitted by the Manager to the Wallet the user is presented a chance to provide a meaningful label for the account (see Figure 3.4(a)). By default the label text is populated with the site URL; the suggested default label may be renamed by the user (*e.g.,* for privacy reasons). Each account in

[7] `https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsILoginManager`



(a) Save account (b) Account list

Figure 3.4: Screenshots of the Tapas Wallet.

the Wallet has a large touch region displaying the user-chosen label for the account. Additionally, each account displays the date on which it was last used, and the date on which the account was added to the Wallet. Accounts are listed in order of most recent use (see Figure 3.4(b)).

3.3.4 Password retrieval

When the user taps an account label in the Wallet on their smartphone, the stored credential associated with the account is transmitted to the paired Manager by Protocol 3. Figure 3.3 shows how the communication channel is set up. The Manager and the Wallet rely on communication with the Rendezvous Server (smaller arrows) to negotiate a direct network connection between one another (larger arrow). Assuming

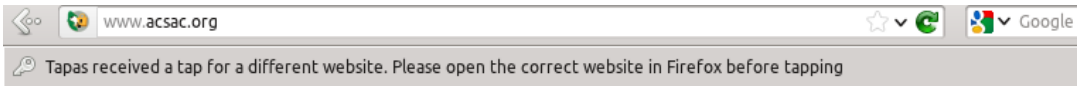


Figure 3.5: The notification the Tapas extension displays for mismatched user intent.

the user's browser is open to the correct website (*i.e.,* is viewing the URL associated with the tapped account) then the username and password field on the website are filled by the Manager and submitted. The result of the login process is returned to the Wallet in order to display meaningful status messages to the user via the Wallet UI. All communication between the Manager and the Wallet is carried out over a mutually-authenticated TLS connection.

Tapas requires the user to signal their intent on both the browser and the smartphone before a login can occur. When a user's smartphone transmits account credentials from the Wallet to the Manager, the latter decrypts the account information and verifies that the decrypted URL matches the currently open web page before filling the username and password with the decrypted credentials. If a URL other than the one contained in the decrypted ciphertext is open in the browser the Manager displays a message indicating that the correct URL must be opened before a login can occur (see Figure 3.5). This prevents accidental logins or a situation in which the user is away from their computer and accidentally triggers a login to a website by tapping their smartphone.

3.3.5 Limitations

Network. Tapas requires an internet connection both on the desktop computer and the smartphone. If the Manager is not connected to the internet it will be unable to store credentials to the Wallet, or receive stored credentials to decrypt for a login. Similarly, if the Wallet is unable to connect to the internet it will be unable to receive ciphertext to store or to transmit any stored ciphertext back to the Manager. If the Wallet is using a cellular data connection (*i.e.,* not the same network connection as the Manager) password management may be hampered by low signal strength. We consider this a reasonable limitation as Tapas was designed for use with internet facing public websites, which themselves would be unreachable without a working and stable

internet connection. We emphasize that due to the Rendezvous Server the Wallet and Manager may be on separate networks without issue.

Rendezvous Server. Tapas also relies on the presence of an accessible Rendezvous Server in order to support network traversal between the Wallet and Manager. In the event no Rendezvous Server is reachable Tapas logins and setup will still function, if the two devices are on the same network and can communicate directly without requiring a third party to mediate the connection.

Paired Devices. In order to use Tapas, both the paired devices must be present and in a usable state. In the case of the smartphone Wallet this means the battery must be charged. Our initial prototype implementation of Tapas allows pairing between only one computer and one smartphone, preventing use with multiple machines. Implementation of a full fledged secret sharing scheme [55] could address the multiple device scenario. In the event either device is lost the user would be forced to reset the password for each of their stored accounts using the password reset functionality offered by the individual websites. An encrypted backup solution for the Tapas Wallet could mitigate the frustration imposed by this limitation.

Web Accounts. Tapas is implemented as a browser extension, limiting its applicability primarily to storing credentials for logging in to websites. It is not currently possible to use Tapas to manage passwords for non-web accounts such as Wifi networks, or operating system accounts. Our design was focused on optimizing for the most common use case (web accounts) rather than to support all password storage scenarios.

3.4 Security evaluation

We evaluate the security of Tapas relative to other types of password wallets, both with and without a master password. We assume the existence of an adversary with the ability to intercept, record, and modify any communication between the Manager and the Wallet except the one-time pairing process (Protocol 1) conducted over an

AS-OOB channel (implemented in Tapas as a visible QR code). The pairing process allows the devices to obtain each other's public keys for authentication enabling the devices to communicate confidently later in the presence of an active adversary on standard communication channels (as in Protocols 2 and 3). In addition to assuming access to the communication channel between the Manager and Wallet, we assume the adversary may have physical possession (theft) of either the Manager or the Wallet hardware. Tapas offers no security against a loss of both.

Resistance to Theft If a device with an unprotected password wallet is physically lost, there is no inherent protection of the passwords stored in the wallet. The use of a master password offers some protection, however the adversary may still be able to conduct an offline attack that will recover all the passwords if the master password is not strong by attempting decryption of the protected credentials using candidate passphrases, confirming trial decryption results against the expected structure of the cleartext data (*i.e.,* presence of ASCII characters, words, urls). On the other hand, a strong master password introduces usability issues related to memorability and accurate entry. In Tapas, theft-resistance (in this sense) is provided against offline attacks without the user having to remember any passwords.

Smartphones (which hold the Wallet in Tapas) are frequently lost and stolen. Passwords in the Wallet are encrypted in such a way as to be indistinguishable from randomness without the decryption key. This is a consequence of using the AES GCM mode of operation which provides indistinguishability under chosen plaintext attacks. The randomly generated 128 bit AES decryption key is held by the Manager and not contained on the smartphone, therefore the stored passwords are protected against even an offline attack. Further, aside from the user-chosen tag, all metadata corresponding to the stored passwords is also encrypted, providing privacy against individuals with passive access to the smartphone.

The Wallet also contains a wallet authentication key. Disclosure of this key to an adversary would allow the adversary to masquerade as the Wallet. The Wallet's only functionality is receiving and pushing encrypted passwords to and from the Manager. The ciphertext of each stored password is authenticated by the Manager's decryption key — a feature of GCM that prevents the decryption of any modified ciphertext. If

a modified ciphertext caused the password portion to be submitted to a non-HTTPS site or one controlled by the adversary, the adversary could learn it. GCM does not allow the plaintext to be manipulated in structured ways, unlike other modes (*e.g.,* ECB or CBC). More generally, authenticated encryption ensures that the Manager cannot be a useful decryption oracle to the adversary.

The user's computer (which hosts the Manager in Tapas) may also be lost, stolen, or given away without the proper deletion of memory. In this case, the adversary recovers the symmetric AES encryption key k. k would allow the adversary to recover each password p_i given its ciphertext c_i, however the set of c_i are stored by the Wallet (*i.e.,* on the Wallet device). Recall our assumption that the adversary can store all past communications observed over the secure channel in Protocols 2 and 3. In order to prevent such an adversary from learning the full set of c_i, an essential design requirement is that the encrypted passwords are communicated over an encrypted channel even though they are themselves already encrypted. Further, the adversary would also learn the authentication key sk_m. If the design of the secure channel provided only authentication and encryption (using *e.g.,* the RSA-based ciphersuites in TLS), sk_m would be sufficient to derive the session key used in past executions of Protocol 2 and 3, allowing the adversary to recover the set of c_i. To thwart this line of attack, the secure channel in Protocols 2 and 3 is designed to have perfect forward secrecy (using a Diffie-Hellman key exchange in TLS) to ensure past session keys cannot be derived from a compromised sk_m.

Resistance to Malware Like other password managers, Tapas cannot protect stored passwords from persistent malware on the user's computer. If current websites are not altered in any way then the passwords must, at some point, be in plaintext for submission to the web service as per the current design of most web services (this is true if users memorize their passwords as well). With a traditional browser based password manager, malware resident on the client can immediately recover all the stored passwords as soon as the master password is entered. With Tapas, individual site passwords can only be recovered as they are used. If the malware is detected and removed, unused passwords will remain safe by repeating Protocol 1. Tapas also prevents specific forms of attack like hardware keystroke loggers and shoulder surfing

as no site passwords or master password must be entered during authentication once the credentials are stored in the Wallet.

Chapter 4

Tapas Usability Evaluation

Prior work [16, 35] strongly demonstrates the need for password managers to be usable both in order to achieve meaningful deployment in the real world and to ensure secure operation. We elected to evaluate the usability of Tapas with a user study soliciting input from users without a strong technical background. It was our hypothesis that Tapas would be equivalently usable as the Firefox browser-based password manager, demonstrating error free setup and usage.

To evaluate the usability of Tapas, we conducted an in-lab user study with 30 participants. Using the results of this initial study we altered the Tapas implementation to address implementation drawbacks highlighted by participant feedback. The improved Tapas prototype was used to conduct a follow-up study with 10 additional participants evaluating the changes made. Our study design was approved by the Carleton University Ethics Review Board and participants were required to fill out an informed consent form (See Appendix C).

4.1 Overview

We selected a between-subjects design where participants were randomly assigned to one of three conditions: Firefox with no master password (NMP), Firefox with a user-chosen master password (MP), and Tapas. Each participant was asked to complete a set of core tasks using the assigned password manager (see Section 4.5). We collected data through observation of the participants' interaction with the password manager as well as through questionnaires before and after each participant's session. We did not mention to participants that Tapas was our own application, in an attempt to avoid biasing participants.

We opted for an in-person study rather than an Amazon Mechanical Turk[1] study

[1] `http://aws.amazon.com/mturk/`

for two reasons. First, Tapas requires the use of an Android phone, and installation of an app not available in the standard Android market. In our study, we provided participants in the Tapas condition with an Android phone pre-loaded with the application, and a Firefox browser with the Tapas extension pre-installed. It was not our intention to test the software installation aspects. In the future both the Tapas Android application and browser extension can be made readily available through the standard software installation process for their respective platforms.

Second, conducting an in-person study allowed for direct observation of user behaviour when using the password managers. Participants were encouraged to think aloud, describing their thought processes as they interacted with their respective password manager. It would have been difficult, if not impossible, to gain as much information about the participant interaction with the password manager using a study performed out of laboratory.

4.2 Participant demographics

We recruited a total of 30 participants (17 males, 13 females) through posters around the university campus and mailing lists (See Appendix E and F). Most (age 18 to 42, $\bar{x} = 24.13$) were university students or staff. Participants had a wide range of backgrounds including accounting, psychology, theoretical physics, criminology, music, computer science and math.

Devices, operating systems and browsers. The majority of participants described themselves as Windows users (86%) and Google Chrome users (76%). A smaller number used MacOS and Linux regularly and one participant did not know how to tell what operating system he/she used. Chrome was the most popular browser, followed by Firefox which was used regularly by 50% of participants. Internet Explorer, Opera, and Safari were less popular. 28 participants owned a cell phone or smartphone. Smartphone OSs were approximately evenly split among Android, iOS and Blackberry.

Passwords and password managers. When asked to describe their use of passwords on the Internet, participants reported having between 3 and 40 ($\bar{x} = 11.53$) accounts that require passwords, and between 2 and 25 ($\bar{x} = 5.73$) unique passwords for those accounts, implying password reuse. 70% reported changing their passwords very rarely or never. Two of 30 participants commented that the only time they change their passwords is if they are forgotten.

Participants in general had a poor understanding of the term *password manager*. Only 2 reported using a password manager, but several explained during their session that they do in fact use the browser's built-in password manager (though not calling it by that name).

Online Activities. We asked participants if they ever purchased goods online, or use online banking. All but one reported being active users of online banking, 80% using these services at least a few times per month. For online purchases, all but two reported buying something online. Of the 28 participants who had bought something online, 73% reported doing so several times a year.

4.3 Study setup

An Ubuntu Linux[2] computer with Firefox pre-installed was used to perform the study. The Firefox history and settings were restored to defaults between sessions, so every participant saw the same "clean install" version of the browser. Firefox was configured to open in full-screen mode, obscuring any OS elements. This was a concious effort to prevent users unfamiliar with the Ubuntu environment from being confused or otherwise distracted. The computer itself was set up in a quiet and isolated room within a university setting.

Tapas requires that both a Firefox extension be installed on the desktop PC and an Android application be installed on the smartphone. We chose to avoid testing this common software installation process, and focused on the initial (post-install) setup and use. Thus, the Tapas Manager extension and Android Wallet app were installed, but not configured, before user sessions.

[2] `http://www.ubuntu.com/`

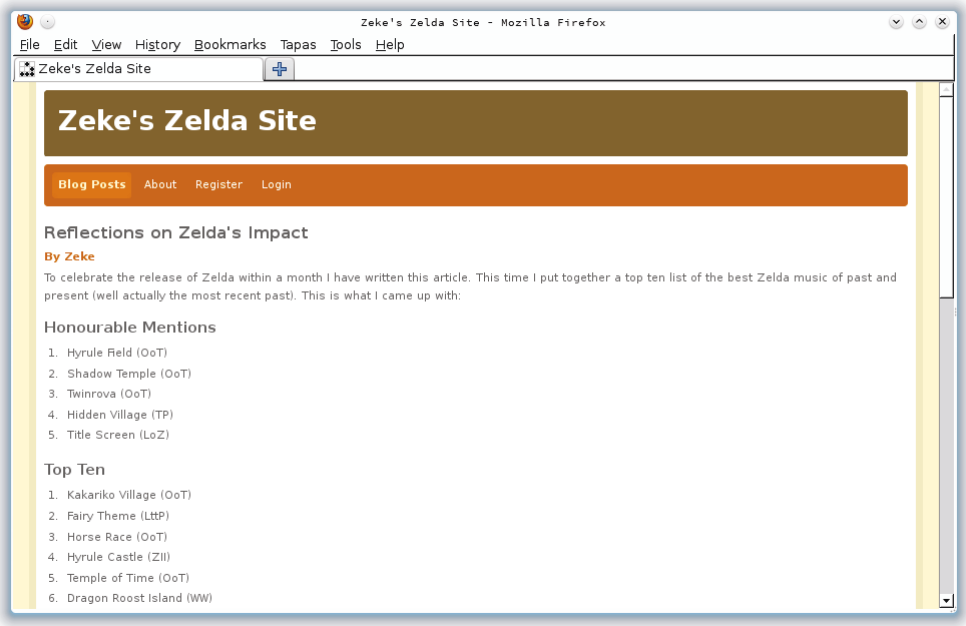


Figure 4.1: User Study Blog A – Zeke's Zelda Site

Three blog websites were created for the study (hereafter referred to as blog A, B and C). A minimal custom PHP[3] backend was created that allowed for only three actions: account registration, login, and leaving a comment. This functionality was written as small PHP fragments that could be included directly into each of the test websites. Account information and comments were saved to a SQLite database by the PHP code. The comment functionality was implemented to require the user to log in prior to being allowed to add a comment to a blog. Each of the three blog websites imported the registration, login and comment code in order to maintain a consistent experience across the experiment. The websites were styled uniquely using a combination of custom XHTML, CSS and fake content. We additionally included fake comments to give participants more content to respond to.

The title and content of each blog were as follows:

A. Zeke's Zelda Site – A video game fan website dedicated to Nintendo's Legend of Zelda franchise. Blog A had fake content describing the author's opinion on the best songs from the franchise. Pictured in Figure 4.1.

[3] `http://php.net/`

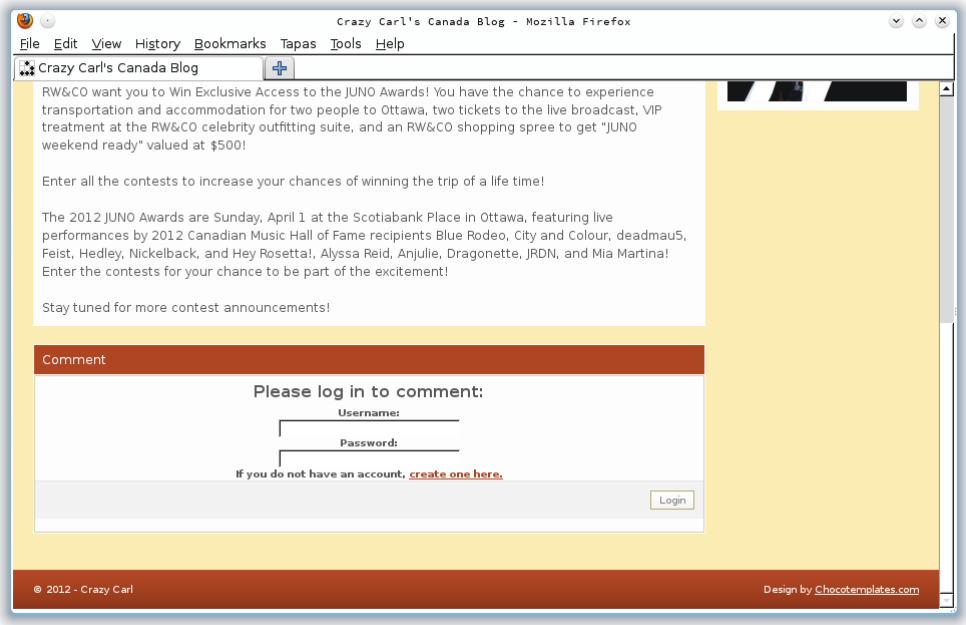


Figure 4.2: User Study Blog B – Crazy Carl's Canada Blog

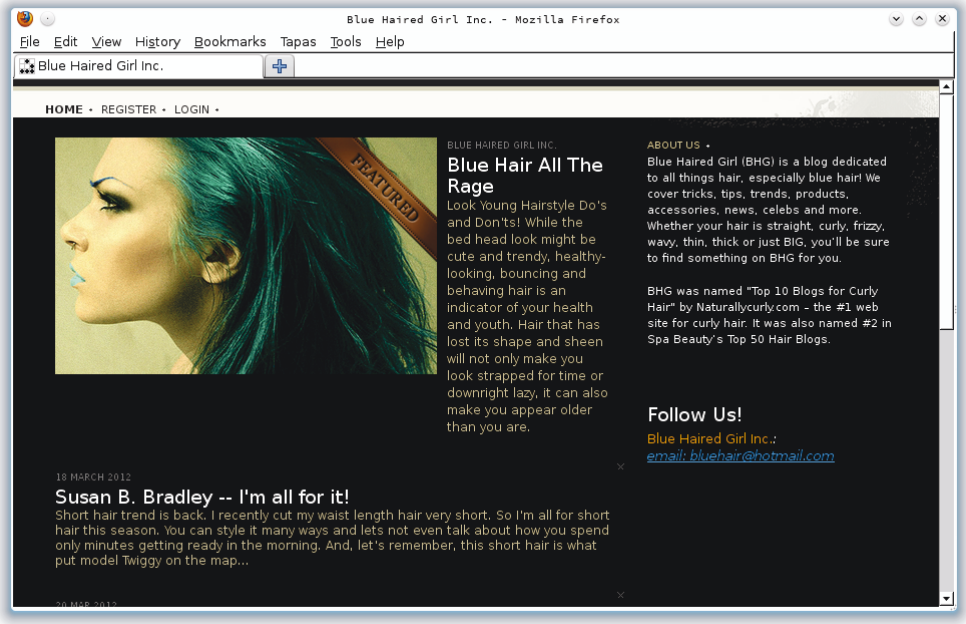


Figure 4.3: User Study Blog C - Blue Haired Girl Inc.

B. Crazy Carl's Canada Blog – The blog of an enthusiastic Canadian music fan, Blog B contained a fake post detailing a contest to win tickets to an award show. Pictured in Figure 4.2.

C. Blue Haired Girl Inc. – A style website with fake content describing popular trends in women's hair. Pictured in Figure 4.3.

By editing the `/etc/hosts` file on the experiment machine, Blog A, B and C were given unique domain names (*zekeszelda.com, carlsblog.ca* and *bluehairedgirl.com*). The blog domains were only accessible in the experiment environment, masking the fact that the websites were created and hosted for the purpose of the experiment. On the experiment machine an instance of the Apache Webserver[4] was installed and configured with three virtual hosts,[5] one for each of the blog domains in the hosts file. This resulted in a self-contained experiment environment that mimicked three websites being externally hosted by three separate entities.

The Logkeys[6] keystroke recovery software was installed and running in the background during each session, recording all keystrokes typed by the user. This was of particular importance for sessions in the MP condition, allowing us to avoid the work of instrumenting Firefox to save the master password. Website usernames and passwords were stored in cleartext in a SQLite database by the PHP blog code.

4.4 Session description

In-person sessions lasted 25 minutes on average and participants were paid $15. Participants were asked to read and sign an informed consent form which stated that their passwords would be logged, but not disclosed. Each participant was asked to read a short explanation of password managers in general, followed by a description of the specific password manager selected for their session. These text descriptions (see Appendix B) were written with the objective of helping the user build an accurate mental model of the password manager rather than focusing on technical accuracy. For example, for the MP and Tapas participants the general concept of a retrieval

[4] `http://httpd.apache.org/`
[5] `http://httpd.apache.org/docs/2.2/vhosts/`
[6] `https://code.google.com/p/logkeys/`

password manager with an encrypted password database is described in terms of a key, a lock, and a safe. Participants were also given a reminder that when required to choose passwords, such passwords should be strong enough that they should not be easily guessable but should be such that they remain memorable to the participant after about a week.

4.5 Tasks within each session

During each session, participants were asked to perform the following tasks, after being given verbal instructions only at the start of each task (the examiner's involvement being minimal thereafter). The full script used by the examiner is reproduced in Appendix D.

T1 — Configure password manager: If applicable (*i.e.,* in the MP and Tapas conditions), perform initial configuration.

For MP, this consisted of the following: enable the master password protection in the Firefox settings and create a master password. Participants were asked to choose a strong password that would be difficult to guess, but one they could still remember after about a week.

For Tapas, this consisted of the following: scan the QR code displayed in the Tapas → Preferences pairing screen.

T2 — Create and store accounts into the password manager: Visit blogs A and B, find the "register" or "create account" section and select a username and password. When prompted by the software, save the account into the password manager.

T3 — Migrate an existing account into the password manager: Participants were given a username and password and asked to pretend they already had an account on blog C. Proceed to log in to blog C and save the account into the password manager. Log out of blog C.

T4 — Log in to blogs: After a distraction task,[7] participants were asked to log into and comment on the three blogs, in the following order. First log in to blog C. Next, close the application and reopen Firefox. Next, log in to blog A, followed by blog B. Closing and reopening Firefox was done to help users (particularly those in NMP and MP) identify when their passwords were accessible. Firefox, when configured with a master password, prompts the user for the master password on the first login after the browser is restarted.

4.6 Results of first user study

After completing the in-lab tasks, participants were given a post-test questionnaire designed to capture their comments and experience while interacting with the password manager. This section presents the questionnaire results and observations made by the examiner during the sessions.

Statistical tests. The Kruskal-Wallis non-parametric one way analysis of variance test [37] was applied with a p-value of 0.05 considered significant to determine whether responses from participants in each condition were independent for a given question. If this test yielded a statistically significant p-value (*i.e.,* less than 0.05), one of the conditions was independent. To determine which condition(s) were independent, individual pairs were further analyzed with a non-parametric Mann-Whitney test [40] corrected for repeated measures using the Bonferroni correction [19] to identify the specific conditions for which the results differed.

Box-and-whisker plots. Participant responses for the Likert scale questions are visualized in Figures 4.4, 4.5(a), 4.5(b), 4.6(a), 4.6(b), 4.7, and 4.8 as box-and-whisker plots. Each box segment displays the lower to upper quartile (25 to 75 percent) of responses, with a black line representing the median. The minimum and maximum are shown as lines, or whiskers, extending from the box. Outlier responses are shown as single points outside of the box-and-whiskers.

[7] The distraction task had participants count down from 100 in decrements of 3. This was intended to help remove the recently created passwords from their working memory.

4.6.1 Post-test questionnaire

Perceived usability of password manager setup. Overall, no issues were raised with the setup process in any of the three conditions. Participants in all conditions rated the ease of setup (on a 4-point Likert scale) as either easy or very easy with no obvious trend. Application of the Kruskal-Wallis test found no significant evidence that Tapas differed from the other managers in ease of setup.

Participants were asked if they thought they would be able to complete the initial password manager configuration on their own. In all conditions participants responded positively. Participants in the MP condition noted that while the setup process was easy, finding the master password checkbox in the Firefox preferences was not straightforward, and that if the written session information did not guide them to the right setting it would have been more difficult.

Comments from Tapas participants included the following:

- "This was a really easy step, I had never done it before but it was extremely simple"
- "It was pretty straightforward. It is easy to use"
- "The use of the QR code was a great tool to pair the devices. The set up was easy and quick".

Participants in the NMP condition (no setup required) were also allowed to enter comments regarding setup. A few voiced concerns about the simplicity of setup, stating that it was almost "too simple", and that you may actually end up accidentally saving passwords with the manager you didn't intend to save, a concern echoed in the literature [5]. Comments like these support the Tapas design feature that requires signalled intent on both devices prior to saving account information or logging in using the password manager.

The QR code pairing method was found by participants to be very intuitive and the audio and vibration feedback was verbally noted by some participants as useful. Tapas users mentioned feelings of accomplishment, as though they had achieved something complicated with little effort. On the other hand, the Firefox master password setup screen displays a password meter which no user was able to fill. Some

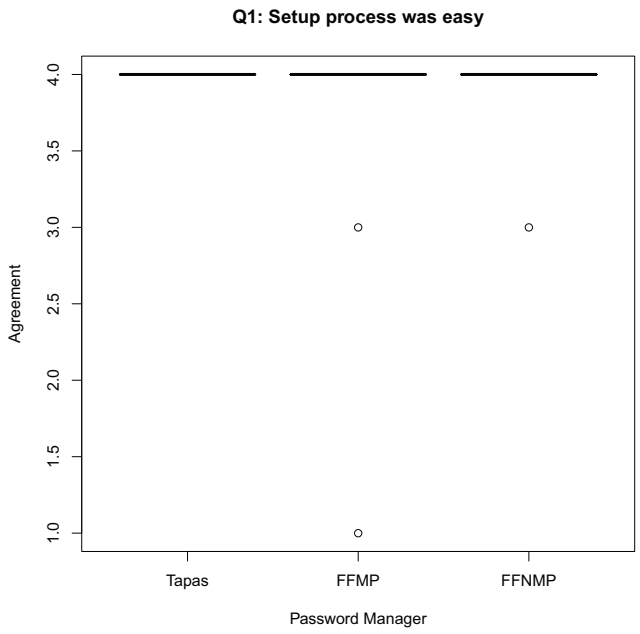


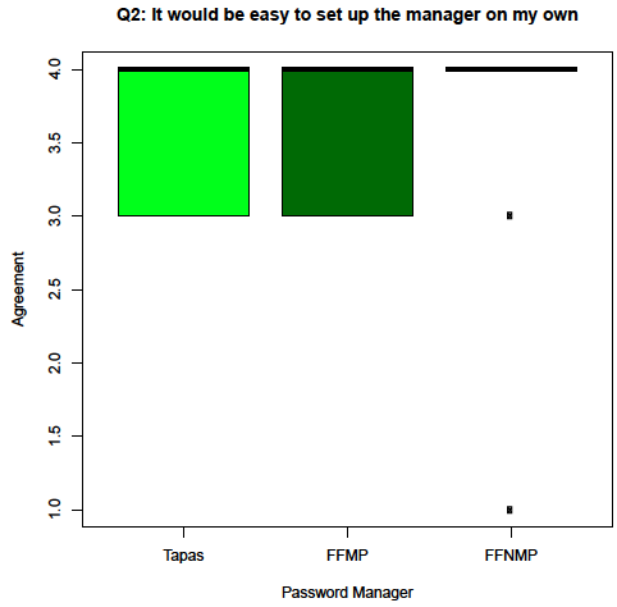
Figure 4.4: Box plot for Q1. A 1 on the Likert scale represents Strongly Disagree and 5 represents Strongly Agree.

users typed in two or three different passwords to try to increase the measure of the password strength bar.

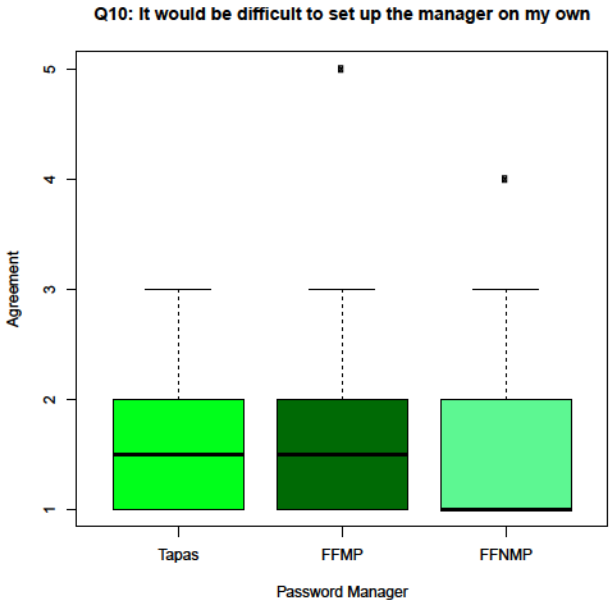
Figure 4.6 shows box plots for questions related to the saving of accounts under tasks T2 and T3 (see Section 4.5).

Perceived usability of password saving. We asked participants to rate their agreement (on a 5 point Likert scale, with 1 as Strongly Disagree and 5 as Strongly Agree) with the following statement: Saving a password was easy when I created a new account and migrated an existing account". Participants did not find Tapas any more difficult than the other two conditions, although some participants had to be reminded to complete the saving process on the phone after clicking the save button in the Tapas extension.

Perceived usability of logins using the password manager. We also asked participants if they thought that using the password manager they were assigned made logging in easier than logging in without one. Based on verbal feedback from

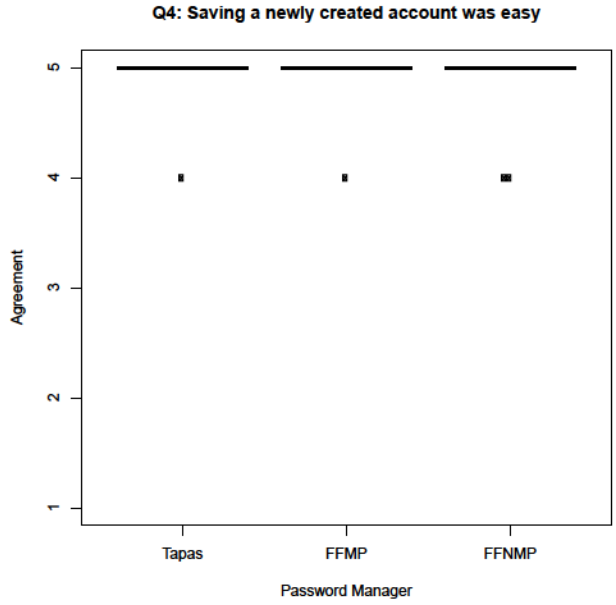


(a) Box plot for Q2

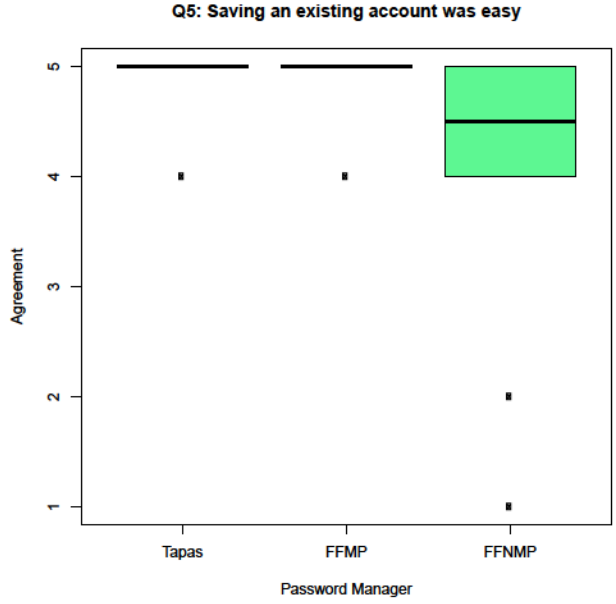


(b) Box plot for Q10

Figure 4.5: Box plots for Q2 and Q10 (See Appendix G). Both questions ask if participants think they could set up the password manager on their own. (a) is phrased positively and (b) is phrased negatively. In both cases 1 is Strongly Disagree and 5 is Strongly Agree.



(a) Box plot for Q4



(b) Box plot for Q5

Figure 4.6: Box plots for questions related to saving (a) newly created accounts into the password manager, and (b) existing accounts. A 1 on the Likert scale represents Strongly Disagree and 5 represents Strongly Agree.

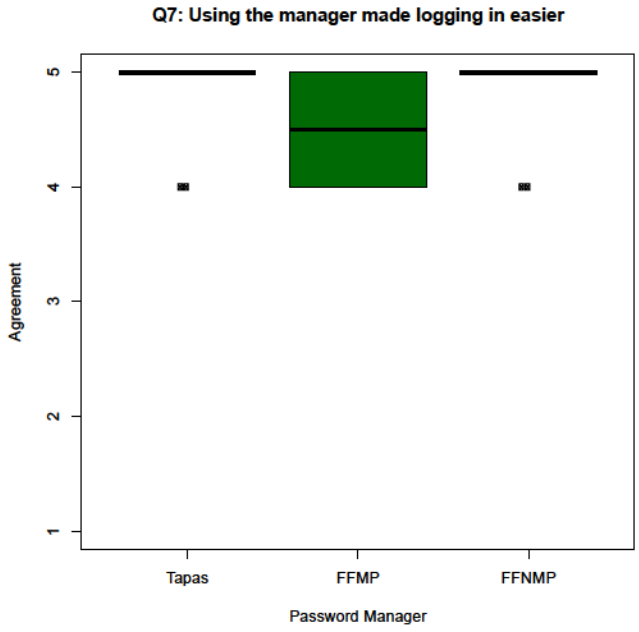


Figure 4.7: Box plot for Q7

participants in the MP condition it appears some users perceive lower ease of use due to the master password being requested when the password manager is invoked for the first time. For the Tapas condition, one participant verbally noted that logging in with Tapas would take longer since you would have to take out your phone and launch the app every time you log in (a step avoided in the account import process by automatic app launch).

User affectation. We asked users to rate how much they enjoyed using the password manager overall. Participants liked Tapas more than MP, and liked MP more than NMP. For this question, the Kruskal-Wallis rank sum test found one or more conditions to be statistically significant ($p = 0.04891$). The Kruskal-Wallis test was followed up by application of pair-wise Mann-Whitney tests adjusted for multiple comparisons using the Bonferroni correction resuling in statistically significant difference between the Tapas condition and both the MP and NMP conditions ($p = 0.01365$ and $p = 0.01742$ respectively). For the NMP condition, 6 participants reported enjoying the password manager and one participant highly disliked it. In the MP condition,

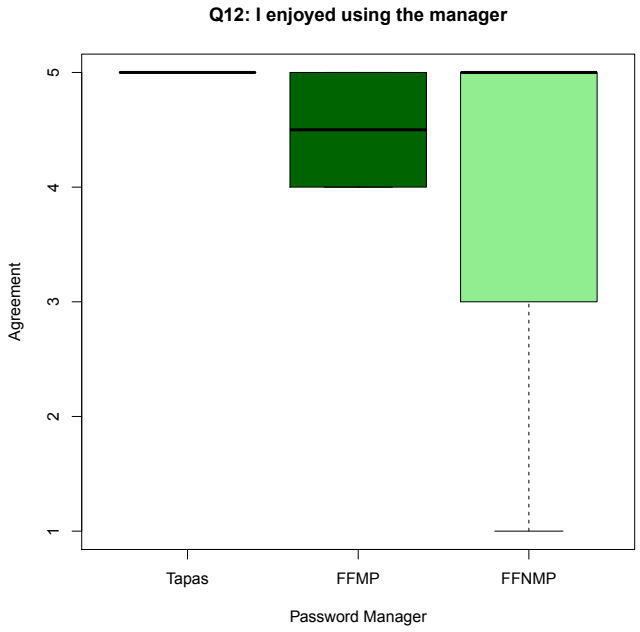


Figure 4.8: Box plot for Q12. A 1 on the Likert scale represents Strongly Disagree and 5 represents Strongly Agree.

5 participants enjoyed using the system, and 5 "somewhat enjoyed" it. Participants in the Tapas condition universally (10/10 participants) rated their enjoyment at the highest level of the Likert scale, demonstrating high user affectation both alone, and in comparison to both the MP and NMP conditions.

Understanding of the password manager mental model. Questions related to attacks on password managers (see Appendix G) also received a wide range of answers, and no condition showed increased understanding of the risks. Participants in all conditions were unable to identify what would have to be stolen by an adversary to successfully impersonate the user. One participant said the adversary would "need my brain to impersonate me", perhaps not understanding what impersonation meant in the context of the question.

Applying a qualitative analysis technique such as Grounded Theory [22] to the participant answers is a potential area for future work. Identifying categories across the responses provided by participants may lead to an increased understanding of how best to explain the threats to password managers.

General participant observations. In the free-form comment field at the end of the survey, several participants across all conditions expressed a desire to know where the passwords were stored. Some participants in the Tapas condition did not notice the pop-down message asking them to save their passwords. Considering this input, we modified Tapas as described in Section 4.7.

A second major theme of comments was related to losing access to the password manager. Several users stated that they probably would never use a password manager because if they lost access to it, they would lose access to all their accounts. While in reality users could still employ the password recovery mechanisms offered by individual websites, these comments highlight the importance of addressing a loss-of-access scenario. This motivates future work to enable an encrypted backup feature for Tapas.

4.7 Improving Tapas and follow up user study

We revised the Tapas Firefox extension incorporating feedback from our 30 participant user study, specifically addressing issues with the poor visibility of the pop-down messages. For the message offering the user the chance to save an account (user ID, password) with Tapas, the background was changed from gray to blue, and the label was changed from "Save with Tapas" to "Save to phone". The error condition pop-down messages were changed to have a red background. We revised the help text used in the Android Wallet application to clarify the goal of the pairing process.

To test the usability of the revised version of Tapas, we recruited 10 (6 female, 4 male) additional participants for the Tapas condition only. The study methodology was identical to the previous study, with the two changes being the updated Tapas Firefox extension, and the updated Tapas help text. Participants in the new study provided further confirmation of the earlier ease of use and affectation findings, additionally providing evaluation of the implementation revisions.

User attention. Observing participants during the second study confirmed that the new blue message background better attracted participant's attention to the "save password" prompt. One participant remarked that the font size for the message was

too small (the default Firefox font size on Ubuntu was used). Only one participant had to be reminded to look for the pop-down message after registering an account.

Mental model. The post-test survey attempted to capture the mental model participants had while using the modified Tapas password manager. All 10 participants in the second study answered correctly when asked "Where were your passwords stored?". In the first study, only 6 of 10 participants in the Tapas condition correctly mentioned the phone. While not statistically significant with 10 participants, we believe a larger sample would likely demonstrate that renaming the save button to "Save to phone" had a strong impact on the users' understanding of how the password manager works. The updated button label clearly explains where passwords are going when the button is clicked. In contrast, participants in the NMP and MP conditions answered this question correctly 50% of the time. Incorrect answers included some participants stating passwords were stored "in cyberspace", "on the website memory" and "no idea". We attributed this to Firefox's ambiguous "Remember password" button label.

4.8 Ecological validity

Regarding demographics, most participants of both studies reported using Chrome as their primary browser, as well as not using a password manager. Thus, the lab study introduced these participants to both a new browser and a new password manager. This may have overloaded participants' memory, moving their attention away from the password manager or otherwise introducing a confounding effect negatively influencing results

The websites used were purpose-built blogs, and thus the security of accounts created for the study was not likely highly valued by users. Participant interaction with these sites, particularly in relation to password choice, may have been influenced by the lack of personal relevance, or perceived value in the blogs offered.

Some participants mentioned that the websites used in the study behaved strangely while logging in. Our sites were designed with minimal functionality. Thus, when a user successfully logged in, the login form would be replaced with a message saying

"successfully logged in", rather than returning to the password protected resource. Some participants failed to notice the change in login status, and were confused because they thought the login had failed.

4.9 Summary

Acknowledging the need for password managers to be usable [16], Tapas was evaluated in a laboratory study with 30 participants, and a follow-up study with 10 additional participants. Without prior knowledge or excessive training, users of a variety of backgrounds were able to correctly setup and use Tapas. Participants voiced positive opinions about Tapas in comparison to the Firefox browser based password manager. Overall knowledge, understanding, and use of password managers was found to be low within study participants, hinting that more work needs to be done to educate users on the topic of password managers. The Tapas design was improved using insight gained from the first user study and reevaluated to measure the results of these improvements.

Chapter 5

Usability-Deployability-Security Framework (UDS)

Proposed by Bonneau *et al.* [12], the Usability-Deployability-Security (UDS) framework is a set of 25 criteria meant to be used to evaluate user authentication schemes. The criteria span potential benefits related to usability of a proposed system or tool, traits required to have the system see wide deployment, and positive security properties the system may offer. For each criterion a proposed system can fully offer the benefit, almost offer the benefit, or not provide the benefit at all. The authors explicitly avoid ranking benefits or providing a numerical score for each property. They indicate that ratings are likely to be highly environment and/or application specific.

The authors use the 25 UDS criteria to perform a comprehensive evaluation of 35 representative authentication schemes, across a number of categories including password managers, graphical password schemes, device assisted proposals, and biometrics (amongst others). For convenient reference we reproduce the description text for each of the 25 UDS criteria in Appendix A. One of the goals of this evaluation was to help explore why despite a great number of proposed alternatives, password authentication remains the dominant incumbent authentication technology.

Examining the results of their comprehensive evaluation the authors note a strong (and in retrospect obvious) pattern. While some schemes improve on the usability or security of passwords, none achieve as many of the deployability properties. This helps to understand why the research community has had no large successes in replacing password authentication.

5.1 Evaluation of password managers

Password managers were identified in Chapter 2 as improving on some of the usability and security properties of passwords while maintaining high deployability. It is noted by Bonneau *et al.* [12] that one of the primary downsides of password managers is

the reliance on password authentication as the underlying technology, preventing any instances within the category from achieving all of the security benefits.

The only browser-based password manager evaluated by Bonneau *et al.* [12] was the Firefox [48] browser-based password manager. In their evaluation Firefox was assumed to be configured to use a master password. We complement this evaluation through application of UDS to browser password managers by evaluating Firefox in the default configuration (*i.e.,* without a master password) in addition to the password managers within Internet Explorer [47], Safari [2], and Chrome [25]. In addition to LastPass [38], which was evaluated by Bonneau *et al.*, we also evaluate 1Password [1], Kamouflage [10], PwdHash [52], Password Multiplier [27], PassPet [59], gridWordX [17], GPEX [7], and iPMAN [6]. The authors of ObPwd have also applied the UDS framework to evaluate both the desktop version of ObPwd [9] and the mobile version [44]; we include their evaluation herein with slight modification for consistency in evaluation (see discussion in Section 5.1.2). For our evaluations we examine the currently available implementations of each tool, in as close to a default configuration as possible. The results of this evaluation are presented in Table 5.1 with an empty cell representing that the scheme does not provide the property, an empty circle (○) indicating that the scheme nearly provides the property, and a full circle (●) indicating that the scheme fully provides the property. We present an evaluation of these password managers with additional evaluation properties tailored for password managers in Section 5.2.

5.1.1 Retrieval password managers

Firefox. Surprisingly in our evaluation Firefox [48] without a master password and Firefox with a master password rate equivalently using the UDS framework except for two properties: Memorywise-Effortless and Physically-Effortless (see Appendix A for property definitions). With a master password Firefox receives an empty circle for both properties rather than a filled circle due to the recall/entry of the master password. While a master password increases the security of stored passwords in the event of unauthorized access (*e.g.,* computer theft, in-person use, or through exposed backups) the UDS framework is not fine-grained enough to distinguish this security

Scheme	U1 — Memorywise-Effortless	U2 — Scalable-for-Users	U3 — Nothing-to-Carry	U4 — Physically-Effortless	U5 — Easy-to-Learn	U6 — Efficient-to-Use	U7 — Infrequent-Errors	U8 — Easy-Recovery-from-Loss	D1 — Accessible	D2 — Negligible-Cost-per-User	D3 — Server-Compatible	D4 — Browser-Compatible	D5 — Mature	D6 — Non-Proprietary	S1 — Resilient-to-Physical-Observation	S2 — Resilient-to-Targeted-Impersonation	S3 — Resilient-to-Throttled-Guessing	S4 — Resilient-to-Unthrottled-Guessing	S5 — Resilient-to-Internal-Observation	S6 — Resilient-to-Leaks-from-Other-Verifiers	S7 — Resilient-to-Phishing	S8 — Resilient-to-Theft	S9 — No-Trusted-Third-Party	S10 — Requiring-Explicit-Consent	S11 — Unlinkable
	Usability								Deployability						Security										
*Passwords			●		●	●	○	●	●	●	●	●	●	●		○						●	●	●	●
Firefox [48] (NMP)	●	●	○	●	●	●	●		●	●	●		●	●	○	○					●	●	●	●	●
*Firefox [48] (MP)	○	●	○	○	●	●	●		●	●	●		●	●	○	○					●	●	●	●	●
IE [47][1]	●	●	○	○	●	●	●		●	●	●		●	●	○	○					●	●	●	●	●
Safari [2]	●	●	○	○	●	●	●		●	●	●		●	●	○	○					●	●	●	●	●
Chrome [25][2]	●	●	○	○	●	●	●		●	●	●		●	●	○	○					●	●	●	●	●
*LastPass [38]	○	●	○	○	●	●	●	○	●	○	●		●		○	○	○	○			●	●		●	●
1Password [1]	○	●	○	○	●	●	●		●	○	●		●		○	○	○	○		○	●	●	●	●	●
Kamouflage [10]	○	●	○	○	●	●	●		●	●	●			●	○	●		○			●	●	●	●	●
PwdHash [52]	○	●	○		●	●			●	●	●			●	○	○	●	●		●	●	●	●	●	●
PMult [27][3]	○	●	○		●	●			●	●	●			●	○	○	●	●		●	●	●	●	●	●
PassPet [59]	○	●	○		●	●	●		●	●	●			●	○	○	●	●		●	●	●	●	●	●
ObPwd-D [9][4]	○	●			●	●	●			●	●			●	●	●	●	●		●	●	●	●	●	●
ObPwd-M [44][5]	○	●			●	○	○			●	●			●	●	●	●	●		●	●	●	●	●	●
gridWordX [17]	○	●	○		●		●			●	●			●	○	○	●	●		●	●	●	●	●	●
GPEX [7]	○	●	○		●		●			●	●			●	○	○	●	●		●	●	●	●	●	●
iPMAN [6]	○	●	○		●		●			●	●			●	○	○	●	●		●	●	●	○	●	●
Tapas [45]	●	●	○	○	●	○	●		○	●	●			●	●	○					●	●	●	●	●

[1] **I**nternet **E**xplorer
[2] Assuming presence of keyring integration. See Browser Managers discussion in Section 2.7.1.
[3] **P**assword **Mult**iplier.
[4] ObPwd desktop [9].
[5] ObPwd mobile [44].

Table 5.1: Evaluating password managers using the Usability-Deployability-Security framework [12] for comparative evaluation of password alternatives. The rows preceeded by * (*Passwords*, *Firefox (MP)*, and *LastPass*) are reproduced unchanged from Bonneau *et al.* [12] for reference comparison. Schemes are grouped according to the taxonomy presented in Section 2.6. The UDS benefit descriptions are reproduced in Appendix A for quick reference.

benefit. The Firefox password manager (with or without a master password) does not support or mandate generating strong passwords for the user, precluding delivery of the following benefits: Resistance to Throttled Guessing, Resistance to Unthrottled Guessing, and Resilient to Leaks from Other Verifiers.

IE, Safari, Chrome. Internet Explorer [47], Safari [2] and Chrome [25] are identical to Firefox with a master password except for the Memorywise-Effortless property. For these ratings we assumed a configuration that encrypts stored passwords (*i.e.,* Chrome must be installed with keyring integration as discussed in Section 2.7.1). We give IE, Safari and Chrome the Memorywise-Effortless property because they achieve encrypted storage by using the user's desktop login password for the encryption key. Since no other schemes penalize the user for having to remember the password for the computer/account on which they've installed the authentication tool/software we say that it is Memorywise-Effortless. Firefox with a master password only gets a quasi rating for Memorywise-Effortless because the master password must be memorized in addition to the user's operating system account login password. Similar to Firefox, none of the other browser based password managers support generating strong passwords for users. Because an attacker can target the login password, the browser based password managers are awarded a quasi rating on Resilient to Targeted Impersonation.

LastPass. We retain the same ratings for LastPass [38] as Bonneau *et al.* [12]. In the usability category, similar to Firefox with a master password, LastPass receives a quasi rating for Memorywise-Effortless, as the user must remember a master password distinct from the system password. It receives quasi for the Easy Recovery from Loss property as a byproduct of being able to print one-time-passwords usable in the event the master password is forgotten (See *Untrusted Login* in Section 2.5). In the deployability category LastPass receives a quasi rating in the Negligible Cost Per User category as there is both a free and paid subscription version, with some features (*e.g.,* Mobile access – which is not one of the UDS properties) restricted to the subscription version. LastPass is closed source and in comparison to Firefox does not receive the Non-proprietary property. For the Security category LastPass receives a quasi rating

for Resilient to Throttled and Unthrottled Guessing because while LastPass supports random password generation, it does not force users to use it. This leaves any user-chosen passwords vulnerable to throttled and unthrottled guessing outside of any protection offered by the master password by attacking the accounts at the service provider directly. LastPass requires a trusted third party in order to support Web Access (See Section 2.5). Because users may reuse passwords across multiple websites if they do not elect to use the random password generation features, LastPass receives only a quasi rating for Resilient to Leaks from Other Verifiers. Because an attacker can target the master password LastPass is awarded a quasi rating on Resilient to Targeted Impersonation.

1Password. The evaluation of 1Password [1] very closely follows that of LastPass and Firefox with a master password. Unlike LastPass, 1Password does not receive a quasi rating in the Easy Recovery from Loss property as it does not support the generation of one-time-passwords for use in place of a forgotten master password. In the event a user forgets their master password, and they do not have a usable backup they must reset the passwords for each of the managed accounts.[1] Similar to LastPass, 1Password receives a quasi in Negligible Cost Per User and does not receive the Non-Proprietary property. 1Password does receive the full No Trusted Third Party property as it by default does not use any form of cloud synchronization or retain a copy of your encrypted passwords. 1Password is rated equal to LastPass in the Resilience to Throttled Guessing, Resistance to Unthrottled Guessing, Resilient to Leaks from Other Verifiers, and Resilience to Targeted Impersonation properties by the same rationale.

Kamouflage. Interestingly, within the granularity of measurement available in UDS, Kamouflage [10] evaluates identically to Firefox with a master password. Like the Firefox manager, Kamouflage does not support or require generating strong passwords on behalf of the user preventing it from gaining a quasi or full rating on the Resilient to Throttled and Unthrottled Guessing properties. Like Firefox it is free, earning the full Negligible Cost Per User property, and open source, earning the full

[1] `http://help.agilebits.com/1Password3/forgot_password.html`

Non-Proprietary property.

5.1.2 Generative password managers

PwdHash. In the Usability category, PwdHash [52] ranks nearly identically to Firefox with a master password. Unlike Firefox, PwdHash does not receive the Infrequent Errors property on the basis of user study findings [16]. PwdHash does not receive the Physically Effortless property as the master password must be typed in by the user for every login event. In the Deployability category PwdHash ranks nearly equivalently to the browser based password managers, and the freely available retrieval password managers. PwdHash does not have a significant non-academic userbase and therefore does not receive the Mature property. In the Security category PwdHash receives a Quasi rating on Resilient to Physical Observation and Resilient to Targeted Impersonation as the master password can be observed or targeted directly in an efficient manner. Because PwdHash forces strong machine generated passwords it receives a full rating in Resilient to Throttled and Unthrottled Guessing as well as Leaks from Other Verifiers.

Password Multiplier. The Password Multiplier [27] password manager is evaluated to have the exact same UDS properties as PwdHash. Like PwdHash, Password Multiplier was found to have frequent errors in use by participants in a userstudy [16].

PassPet. The PassPet [59] password manager had many of the same properties as PwdHash and Password Multiplier. Notably, according to usability analysis by Yee *et al.* [59] PassPet offered an improved interface that prevented many of the user errors found in PwdHash and Password Multiplier. These improvements give PassPet a full circle in the Infrequent Errors property. In all other categories PwdHash and Password Multiplier are evaluated equivalently under UDS.

ObPwd. The ObPwd password manager was first introduced as a desktop application [9], and later implemented for use on mobile devices [44]. The authors of ObPwd have evaluated [44] both versions using the UDS framework, comparing to ordinary web passwords in a desktop and mobile environment. In contrast to other generative

password managers, neither configuration of ObPwd receives the Nothing to Carry property, as in both cases a device with the user's password objects must be carried. In contrast to its authors' rating, we choose not to give ObPwd the Easy Recovery from Loss property. The authors argue that if the user loses their object password they may reset the individual managed passwords the same as regular lost passwords. This is true for many other managers (*e.g.,* LastPass, Firefox) but the original UDS rankings do not award these managers the property. In order to maintain consistency we therefore do not give either configuration of ObPwd the Easy Recovery from Loss property. We also diverge from the authors rating of the desktop version of ObPwd by declining to award it the Mature property. It is not clear how large the ObPwd non-academic userbase is and only small scale user testing has been performed.

Graphical generative password managers. Two of the Graphical generative password managers (gridWordX [17] and GPEX [7]) ranked equivalently under UDS. The third, iPMAN [6] differed in only one property (No Trusted Third Party). In contrast to other generative password managers, none of the graphical generative password managers receive the Efficient to Use or Physically Effortless property due to the graphical input mechanisms they rely on. Similarly, they are precluded from receiving the Accessible property as it is not clear how blind users will operate the tools (both ObPwd configurations fail to receive this property for the same reason). Like the other generative password managers, all three graphical approaches receive full circles in Resilient to Throttled Guessing, Resilient to Unthrottled Guessing, and Resilient to Leaks from Other Verifiers. The iPMAN password manager introduces what the authors refer to as a *semi-trusted online signature server* (see [6, §6.2]) used to generate salt in a device transportable fashion. For this reason it receives a quasi rating for the No Trusted Third Party property.

5.1.3 Tapas

Tapas is a novel password manager design and implementation described fully in Section 3.3, and is part of the contribution of this thesis. Referring to the password manager taxonomy in Section 2.6, Tapas is a retrieval password manager implementing

encryption using dual-possession authentication. Tapas addresses the usability issue of recalling an ever-growing number of passwords, without resorting to reuse. Relative to ordinary unmanaged passwords this benefit comes at the cost of interacting with a smartphone. In the event that access to the smartphone is lost, passwords need to be individually recovered using existing recovery mechanisms. Adding accessibility features to Tapas for disabled users is future work. When using a password manager, the stored passwords themselves can always be attacked directly. For this reason, password managers cannot improve on certain security properties of passwords. Tapas does provide phishing protection by ensuring stored passwords are only ever submitted to the legitimate site (as determined by the site's URL and SSL/TLS certificate) they were registered with. Additionally the password cannot be observed externally when a user logs in with Tapas as it is not displayed or entered by the user directly at any point during account login.

Relative to Firefox configured with a master password, Tapas does not require a master password to protect the stored credentials but does require interaction with a smartphone. On security, Tapas offers Resilience-to-External-Observation. The framework does not distinguish that the stored ciphertexts in Tapas are resilient to an offline attack if the wallet is stolen, whereas in Firefox an offline attack on the master password coupled with access to the browser reveals all stored passwords at once. Similar to Firefox with a master password, Tapas receives an empty circle in the Physically-Effortless[2] and the Nothing-to-Carry[3] columns. Additionally, malware capable of recovering the Firefox master password can immediately learn all stored passwords, while malware on a client device running Tapas results only in the gradual disclosure of passwords only as they are used. Tapas bears some similarity to proxy-based authentication systems, specifically URRSA [31], but receives additional usability properties (full U3, partial U4, and partial U6) as a result of the tap-based authentication design in comparison to the printed one-time-codes used by URRSA.

For many cases, Tapas does not preclude composition with other mechanisms for

[2]We consider scrolling equivalent to pushing a button per the Physically-Effortless definition. Removing the phone from pocket is equivalent to removing a YubiKey or similar dongle.

[3]We consider the Tapas desktop component to be independent of the Nothing-to-Carry property, just as Pico-siblings are likewise ignored.

improving security. For example, it can be used for per-site hash-based passwords, randomly generated passwords, or remembering the password portion for dual-factor authentication. Additonally, the Tapas prototype could be easily modified to generate a backup of the wallet's stored ciphertexts encrypted with a password-derived key for recovery.

5.1.4 Discussion

Upon reviewing the evaluations of the various password managers under UDS, several themes emerge.[4] First, ignoring a few minor differences, the retrieval password managers all rate nearly identically. In the usability category the only striking difference across retrieval password managers is whether they are fully or quasi Memorywise Effortless. By relying on the user's desktop password as the master password several browser password managers receive full Memorywise Effortless. This design decision sacrifices some security, but it is a subtle enough difference as to not be captured in any of the UDS security properties.

In the deployability category we mainly see differences based on the cost, openness, and maturity. For-profit retrieval password managers receive only quasi rankings in Negligible Cost Per User if they offer a free version with reduced functionality, and none receive Non-Proprietary. These differences are largely based on business model alone. If LastPass or 1Password released their source code openly for free the ratings would change to match the rest of the retrieval password managers. Free versions of the for-profit managers and open source alternatives offer many of the same core features as non-free options. All for-profit retrieval password managers were evaluated in Table 5.1 using the feature-limited free versions.

In the security category we see that all of the retrieval password managers that do not support cryptographically random password generation score the same, while those that support it receive a quasi rating in Resilience to Throttled Guessing, Resilience to Unthrottled Guessing, and Resilience to Leaks from Other Verifiers. No retrieval password manager receives a higher than quasi rating in these categories due

[4]This section discusses the original UDS properties proposed by Bonneau *et al.* [12]. Section 5.2 discusses the additional password manager specific properties we have introduced beyond the original 25 properties.

to random password generation being opt-in.

Looking at the generative password manager category we see that many of the generative password managers share the same usability properties. For instance, they all receive a quasi rating for Memorywise Effortless as each requires some form of master password to be memorized. Interestingly we see that while the schemes that replace a master password with some other form of secret (graphical password, or a digital object) do so expressly to improve memorability, the UDS framework is not granular enough to note the associated usability benefits when compared to the master password generative password managers. It does however capture the negative aspects of using a non-keyboard based secret as many of the non-password based generative password managers suffer in the Efficient to Use category. All of the generative password managers receive identical ratings in Deployability except where a less accessible input medium dependent on sight was relied on, precluding the Accessible property.

Surprisingly, in the security category we rated all of the generative password managers equivalently (excluding ObPwd). *We see that the UDS framework is not granular enough to illustrate the more subtle security differences between the various generative password managers.* For instance, the UDS framework is unable to capture the security benefits that the iterated hashing approach used by Password Multiplier and PassPet offer in comparison to PwdHash. ObPwd differs from the other generative password managers by offering stronger guarantees against Physical Observation and Targeted Impersonation.

Comparing the UDS evaluation of the retrieval password managers to the generative password managers we can see patterns supporting the taxonomy discussion in Section 2.6. Comparing the Usability of retrieval password managers and generative password managers we see that, as a category, retrieval password managers are a more usable solution to password management, requiring less effort to use and having fewer errors in practice. Conversely, generative password managers offer a greater number of the security properties. By forcing users to use strong machine generated passwords the generative password managers all receive a full circle in Resilience to Throttled Guessing, Resilience to Unthrottled Guessing, and Resilience to Leaks from

Other Verifiers.

Coming back to the discussion in Section 2.6 on the surprising lack of successful, widely deployed generative password managers we see that none score even a quasi rating in the Mature property. While it is likely that this is the case due to the large amount of work required for a user to initially transition to a generative password manager (*i.e.,* the all or nothing problem), these difficulties are not reflected in the Deployability properties. Both generative password managers and retrieval password managers score similarly in the Deployability category, hiding the larger initial cost of switching to a generative password manager.

5.2 UDS extensions for password managers

After applying the UDS framework to a number of password managers from the password manager taxonomy (recall Section 2.6) it is apparent that within the broad category of password managers, vanilla UDS is not sufficient to differentiate within them. On the positive side, the existing UDS properties help highlight the broad differences between generative password managers and retrieval password managers, but on the negative side the level of detail is not well suited to differentiating two generative password managers, or two retrieval password managers from one another. If one's goal is to distinguish the advantages/disadvantages of one password manager versus another a more detailed set of criteria is required.

We propose the following set of seven additional properties for use to help distinguish password managers at a finer level of detail. We introduce two new Usability properties, one new Deployability property and four new Security properties:

1. **U9** – Reduces-Recall-Burden – The password manager does not burden the user with pure recall-based memory tasks for the master secret.

2. **U10** – Supports-on-Mobile-Authentication – The password manager can be used to authenticate with account providers from a mobile device.

3. **D7** – Supports-Existing-Passwords– The password manager supports adoption of the user's existing passwords into the tool, allowing the user to migrate

individual accounts to the manager one by one, without necessarily forcing password changes for each such account.

4. **S12** – Credential-Use-Authenticated – The password manager requires authentication upon each occurrence of credential retrieval/generation.

5. **S13** – Resilient-to-Offline-Attacks-on-Master-Secret– The password manager resists offline attack against a master secret that may be captured from persistent long term storage.

6. **S14** – Resilient-to-Ciphertext-Tampering – The password manager employs authenticated encryption, or a message authentication code, detecting ciphertext manipulation.

7. **S15** – Protects-Metadata – The password manager encrypts account metadata in addition to passwords. *e.g.,* Stored domains or usernames cannot be accessed in the clear by an adversary without the master secret.

The first Usability property, U9, adds greater granularity to the Memorywise-Effortless property of UDS by indicating which type of memory the manager invokes. Graphical passwords are often categorized into recall, cued recall, and recognition memory tasks [8]. Recall is often considered to be the most difficult memory task [26], requiring the user to dredge the secret from memory without any cues or outside stimuli. Cued-recall tasks present the user with a stimulus and requires them to recall some secret based on or related to the stimulus. Finally, recognition has the user view one or more stimuli in an attempt to recognize a previously associated stimulus, or to distinguish it from decoys. It is commonly agreed upon in the literature that a cued-recall or recognition task can be performed with greater ease than a pure recall task [26]. Therefore we allocate a full rating to password managers that index recognition based memory, and a quasi rating to password managers that employ cued-recall. A password manager that relies on pure recall does not receive the benefit.

The second Usability property, U10, addresses authentication from mobile devices. While some password managers (*e.g.,* Tapas) involve the use of a mobile device in an

auxiliary role, not all support authenticating to account providers when the mobile device is the user's primary device. The limited input mechanisms available on mobile devices constrains reasonable expectations for password input. When combined with the increased proliferation of smartphones password managers suitable for use on devices with constrained input is an important area of research.

We explicitly add a Deployability property, D7, addressing incremental adoption as it represents a non-trivial barrier to adopting generative password managers that is not adequately captured by the existing UDS Deployability properties. Requiring users to simultaneously change the password of all manager-associated accounts in order to allow adoption of a generative password manager is, in our view, likely to be among the major reasons generative password managers have not seen greater adoption and general usage outside of academia.

The first new Security property, S12, relates to authenticated credential use and differentiates retrieval password managers that rely on the user's operating system account password, or the user's login keyring, to protect the stored passwords. Systems that rely on this form of master password protection may leave credentials vulnerable due to an unlocked workstation, auto-login configuration for the user's account in the operating system, or misunderstanding of the "session model" by the user. Password managers that employ a master secret that is cached or remembered for some duration after the initial entry are precluded from receiving the full rating in this category.

The second new Security property, S13, addresses the master secret's resilience to offline Attack. While the existing UDS framework has the Security properties Resilient-to-Throttled-Guessing and Resilient to Unthrottled Guessing, they are defined in such a way as to preclude differentiating between the security of the generated passwords/managed accounts against throttled and unthrottled guessing, and the security of the master secret and password storage against such guessing. It is possible that a password manager that does not mandate generated passwords (and is thus not eligible for the full circle for the UDS throttled and unthrottled guessing properties) could be implemented such that the master secret and stored (encrypted) passwords are not vulnerable to throttled or unthrottled guessing (*e.g.,* Kamouflage receives the new master secret property but not the existing UDS throttled and unthrottled

guessing properties).

The third new Security property, S14, addresses the threat of ciphertext tampering. This property is a response to several security flaws identified in previous versions of LastPass and 1Password [4]. Unless the password manager uses authenticated encryption or computes a Message Authentication Code (MAC) the ciphertext corresponding to encrypted account information can be modified without knowledge of the master secret, potentially opening the password manager up to attack (*e.g.,* changing login URLs, exploiting password manager vulnerabilities with malicious data).

The fourth new Security property, S15, addresses exposure of account metadata. If the password manager does not protect the metadata (*e.g.,* username, login URL) associated with a user account it may be possible to learn the usernames/websites a user has stored in the manager without knowing the master secret, aiding targeted online attacks and violating the users' privacy. It may also be possible to capture stored credentials by altering an account's associated login URL [4].

5.3 Extended evaluation

We revisit the password managers evaluated using the vanilla UDS framework in Section 5.1 in order to evaluate them with our extended-UDS property set specific to password managers. The results of the evaluation are presented in Table 5.2. Discussion of the evaluation and rationale follows.

5.3.1 Retrieval password managers

Firefox. With no master password Firefox [48] does not require any memory task giving it the full Reduced-Recall-Burden property. While a mobile version of the Firefox browser is available for Android[5] it does not presently support the use of a master password[6], precluding Firefox with a Master Password from receiving the Supports-on-Mobile-Authentication property. Firefox without a master password receives the quasi rating for U10, as it supports synchronization of stored desktop passwords to

[5] `http://www.mozilla.org/en-US/mobile/`

[6] `https://support.mozilla.org/en-US/questions/837679`

Scheme	U9 — Reduced-Recall-Burden	U10 — Supports-on-Mobile-Authentication	D7 — Supports-Existing-Passwords	S12 — Credential-Use-Authenticated	S13 — Resilient-to-Offline-Attack-on-Master-Secret	S14 — Resilient-to-Ciphertext-Tampering	S15 — Protected-Metadata
	Usability		Deployability	Security			
Firefox [48] (NMP)	●	○	●				
Firefox [48] (MP)			●	○			
IE[1][47]			●				
Safari [2]		●	●				
Chrome[2][25]		○	●				
LastPass [38]		●	●	○	●		○
1Password [1]		●	●	○	●	●	●
Kamouflage [10]			●	○	●	●	●
PwdHash [52]				●		●	●
PMult[3][27]				●	●	●	●
PassPet [59]				●	●	●	●
ObPwd D[4][9]				●	●	●	●
ObPwd M[5][44]		●		●	●	●	●
gridWordX [17]	○			●		●	●
GPEX [7]				●		●	●
iPMAN [6]	●			●	●	●	●
Tapas [45]	●		●	●	●	●	●

[1] **I**nternet **E**xplorer

[2] Assuming presence of keyring integration. See Browser Managers discussion in Section 2.7.1.

[3] **P**assword **Mult**iplier.

[4] ObPwd desktop [9].

[5] ObPwd mobile [44].

Table 5.2: Evaluting password managers under the Extended UDS-Framework for Password Managers.

the mobile Firefox, but does not allow management of credentials from the mobile version. Both configurations of the manager earn the full Supports-Existing-Passwords property; in fact, all retrieval password managers get this property, D7, and none of the generative password managers do.

Without a master password the Firefox password manager does not require any authentication to use stored credentials. Passwords are always accessible to an adversary that can gain access to the machine or the storage. Similarly there is no protection against offline attack or to the metadata stored by the manager, and no protection against tampering with stored ciphertext.

With a master password the Firefox password manager receives a quasi rating for the Credentials-Use-Authenticated property, as with a master password configured the user must enter the master password the first time access to the stored credentials is required. It is not merely sufficient to have gained access to the operating system account that the user has installed Firefox under.

IE, Safari, Chrome. The built-in Internet Explorer [47], Safari [2], and Chrome [25] retrieval password managers all receive equivalent ratings within our extended properties. This is to be expected since all three offer identical features and password storage mechanisms (assuming Chrome is installed on a system that allows keyring integration; see Section 2.7.1). Slight differences arise between the browser managers support for On-Mobile-Authentication. Safari for iOS fully supports mobile password management and authentication receiving the full U10 property while Chrome only supports synchronization of passwords saved on the desktop, earning a quasi rating. Notably, Internet Explorer on Windows Phone does not support any password management, precluding receiving the U10 property. All three browser-based password managers receive the same rating as Firefox without a master password for the U9 property as they do not require memorization of an additional secret. Notably IE, Safari, and Chrome all fail to receive the Credentials-Use-Authenticated property. Because each uses the user's operating system account password as the master secret (*i.e.,* DPAPI on Windows), or the system keyring protected by the user's operating system account password for credential storage (*e.g.,* Safari and Chrome), an adversary need only compromise the running system when the user is logged in. This can

be achieved through malware, or by physically acquiring access to the computer when it is logged in and unlocked. Similarly, if the user has not set an account password or if the operating system is configured to autologin the user when the machine boots without prompting for a password the stored credentials will be unprotected. None of the browser managers evaluated currently prevent ciphertext tampering or encrypt account metadata.

LastPass. The LastPass [38] password manager requires the user to set a master password, which must later be recalled to gain access to the stored credentials, precluding LastPass from receiving the Reduced-Recall-Burden property. LastPass earns the full Supports-Existing-Passwords property like the other retrieval password managers. LastPass is available on both iOS and Android, supporting On-Mobile-Authentication and receiving the full U10 property. LastPass receives a quasi rating for the Avoids Unlocked State property because it requires the user to enter their master password the first time they wish to access the protected credentials. Similar to Firefox with a master password this password must be provided in addition to the one used to log into the operating system account. LastPass receives a full circle for Resilient-to-Offline-Attack-On-Master-Secret since adopting PBKDF2 [33] with a default of 500 iterations for the master password hash.[7] LastPass does not use authenticated encryption or compute a message authentication code on the ciphertext, precluding the Resilient-to-Ciphertext-Tampering property. LastPass does encrypt metadata, earning the full Protected-Metadata property.

1Password. The 1Password [1] password manager is rated identically to LastPass except for the Protected-Metadata and Resilient-to-Ciphertext-Tampering properties. 1Password encrypts all metadata in addition to using a MAC construction over the ciphertext.[8]

[7] `https://helpdesk.lastpass.com/security-options/password-iterations-pbkdf2/`
[8] `http://pdox.ca/tlm2w`

Kamouflage. The Kamouflage [10] password manager receives the strongest rating of all the retrieval password managers. As compared to LastPass and 1Password Kamouflage receives a full circle for the Resilient-to-Ciphertext-Tampering and Protected-Metadata properties. Kamouflage uses authenticated encryption that detects modification to stored account ciphertext. Further, Kamouflage hides the user's encrypted metadata amongst a large set of decoys earning the full S13 property. Kamouflage is only usable to authenticate on a desktop computer and does not support on-mobile authentication, precluding receiving either the quasi or full rating for the U10 property.

5.3.2 Generative password managers

PwdHash. The PwdHash [52] password manager requires the user to enter a master password each time they wish to access a site managed by the tool. For this reason PwdHash does not receive the Reduced-Recall-Burden property. There is no mobile version of PwdHash, precluding the U10 property. Like all other generative password managers, PwdHash does not receive the Supports-Existing-Passwords property. PwdHash does receive a full circle for the Credential-Use-Authenticated property as well as the Resilient-to-Ciphertext-Tampering and Protected-Metadata properties. PwdHash does not store any state to be unlocked, requiring the master password to be entered for all actions. For similar reasons it is not vulnerable to ciphertext tampering or metadata leaks. Pwdhash does not receive the Resilient-to-Offline-Attack-on-Master-Secret property because PwdHash does not employ iterated hashing, allowing quick validation of candidate master passwords. Further, combined with one or more generated passwords PwdHash is vulnerable to an enumeration attack.

Password Multiplier. The Password Multiplier [27] password manager is rated identically to PwdHash save for the Resilient to Offline-Attack-on-Master-Secret property. Password Multiplier receives a full circle for this property because of the iterated hashing that it employs to create site-specific passwords, requiring an adversary to expend a large amount of computational resources to verify each candidate master

password.

PassPet. PassPet [59] is evaluated equivalent to Password Multiplier across the five properties.

ObPwd. Both ObPwd Desktop [9] and ObPwd Mobile [44] are evaluated equivalent to Password Multiplier and Passpet. As indicated by the name, ObPwd Mobile supports On-Mobile-Authentication earning the full U10 property.

Graphical generative password managers. The graphical password generative password managers (gridWordX [17], iPMAN [6] and GPEX [7]) rate similarly to one another under the extended UDS framework. None of the graphical generative password managers are available for use on a mobile device, precluding the U10 property. Notably gridWordX using a cued-recall memory task as the master secret earns a quasi rating for the Reduced-Recall-Burden property. The iPMAN password manager uses a recognition based memory task (selecting icons from a static set) earning the full Reduced-Recall-Burden property. Only iPMAN receives the Resilient-to-Offline-Attack-on-Master-Secret property due to it's unique salting and the use of iterated hashing.

5.3.3 Tapas

Unlike many of the other password managers, Tapas does not require the user to remember any additional secrets, receiving the full Reduced-Recall-Burden property. Tapas, while using a mobile device in an auxiliary role, does not support On-Mobile-Authentication highlighting a weaknesses of Tapas. In contrast to the generative password managers, but similar to the retrieval password managers, Tapas allows users to import their existing passwords, earning a full circle for the Supports-Existing-Passwords property. In contrast to other retrieval password managers Tapas receives a full circle in each of the added security properties. Tapas never leaves unprotected credentials accessible, requiring authentication and user intentionality for all actions, and encrypts all metadata and passwords with an authenticated ciphermode (see Section 3). Unlike the retrieval password managers, Tapas does not rely on a master

password for the master secret; instead a strong random public key pairing model is used, preventing efficient offline attack (See Section 3.4).

5.3.4 Discussion

Looking at the extended evaluation results in Table 5.2 we can see that the introduction of the new Usability property, U9, primarily highlighted the type of memory (recognition, cued-recall, or recall) invoked by the manager's master secret.

The password managers that receive a full or quasi rating for the Reduced-Recall-Burden property all strive to reduce the difficulty associated with recalling a master secret. In the most trivial case, Firefox without a master password, this comes at the cost of all of the security properties (S12 — S15). Tapas, iPMAN, and gridWordX are able to remove or reduce the requirement for the user to recall a master secret while still protecting stored credentials. Support for On-Mobile-Authentication is sparse, with few of the generative password managers offering mobile versions. Firefox and Chrome both offer only limited On-Mobile-Authentication, requiring the use of a desktop computer for adding and managing saved credentials to be synchronized with a mobile device.

The new deployability property related to incremental adoption highlights the high adoption cost of generative password managers. Not a single generative password manager supports any incremental adoption scheme. This serves to further separate generative password managers and retrieval password managers in the deployability category. A wide deployability gap between these two categories of password managers more closely matches the adoption patterns we have seen (*i.e.,* no generative password managers receive the Mature property in the vanilla UDS evaluation presented in Table 5.1).

Using the new security properties we see a more clear distinction between the browser managers (Firefox, IE, Safari, Chrome) and the dedicated password managers (LastPass, 1Password, Kamouflage). The browser managers that rely on the user's OS account password or a keyring protected by the user's OS account password are weaker in the security category than the third party password managers, a finding that was not apparent in the vanilla UDS evaluation scheme. We also see some difference

between the LastPass, 1Password and Kamouflage related to their resilience to direct attack on the password manager storage (or master secret).

The generative password managers largely differ from the retrieval password managers in the new security property ratings based on their resilience to offline attack on the master secret. Solutions that do not offer iterated hashing fail to receive the property, displaying a security weakness that would not have been noted in the vanilla UDS framework.

Chapter 6

Conclusion

Authentication on the web is dominated by passwords, mandating that users choose strong passwords (most often manually) to achieve security. As the number of accounts each user is meant to support continues to grow, it is intractable for users to maintain secure account management practices without aid. Coupled with the increasing ubiquity of devices with limited screen real estate and touchscreen keyboards it is exceedingly difficult for users to accurately enter the passwords they choose, especially random passwords with special characters and numbers. These trends, combined with an acknowledgement of the importance of usability and deployability in addition to security, motivates study and development of password managers.

Development of a taxonomy of password managers helped to illustrate the approaches to password management available today. This systemization of password managers increased our ability to evaluate the relative strengths and weaknesses inherent in the subcategories available. Using the Usability-Deployability-Security framework we have presented the properties of password managers as proposed password authentication replacements, with emphasis on how these properties relate to the categories present in the password manager taxonomy. While the UDS evaluation suggests password managers as a viable near-term solution to many of the burdens imposed by password authentication it alone was not sufficient in fully distinguishing the Usability, Deployability and Security properties of one password manager from another. Providing the extended UDS properties specific to password managers enabled this fine-grain inter-category comparison. Using the extended evaluation offered insight into the reasons that retrieval password managers have seen greater adoption outside of the academic community than generative password managers, despite the latter's prevalence in the literature.

Building on insight gained from a fine-grain comparison of password managers we

presented the concept of dual-possession authentication, and a threat model for developing password managers based on the notion of requiring possession of two independent devices in order to authenticate with stored credentials. We further introduced Tapas, a concrete implementation of a dual-possession authentication based retrieval password manager that achieves a unique set of Usability-Deployability-Security properties differentiating it from the other retrieval password managers evaluated.

We designed Tapas to be compatible with password-based authentication, while relieving users of traditional memory burdens. Tapas avoids the use of a master password — a security mechanism which users found difficult to locate on existing password managers, is not touchscreen friendly, does not offer strong protection against offline attacks, and may be inadvertently disclosed by users. Additionally with Tapas, users can walk away from their computers without exposing stored passwords that may be temporarily unlocked—every login requires an explicit action by the user.

In laboratory user testing of Tapas established its viability for use as a password authentication replacement by users without sophisticated knowledge of password managers, or prior exposure to Tapas. Participants were able to successfully, and without error, use Tapas to manage account credentials and authenticate with service providers. Building on participant feedback from the initial round of user study we addressed common sources of confusion, creating an iterated version of Tapas that was reevaluated using a second user study with positive results.

The limitations of Tapas as currently implemented suggests that future work addressing secure backup of managed credentials would increase robustness of the manager, helping avoid a time intensive password reset process in the event of device malfunction or loss. Similarly, extending Tapas to be usable for on-mobile authentication would increase the number of scenarios in which Tapas could be considered a viable alternative to password authentication. This could be achieved by using the smartphone app previously acting as the Wallet as the Manager and offloading the Wallet storage to a NFC-tag, or similar low powered secondary device. Tapas and other password managers designed for usability and security offer a realistically achievable way to improve account authentication for general users given the current state of authentication on the web. Further study to understand common user

concerns preventing adoption of password managers may aid in providing security improvement by encouraging adoption of password managers.

Appendices

Appendix A

UDS Benefits Reference

For convenient reference we reproduce the original 25 Usability-Deployability-Security definitions provided by Bonneau *et al.* [12]. We defer more in-depth discussion of the properties to the original authors.

A.1 Usability benefits

U1 *Memorywise-Effortless*: Users of the scheme do not have to remember *any* secrets at all. We grant a *Quasi-Memorywise-Effortless* if users have to remember *one* secret for everything (as opposed to one per verifier).

U2 *Scalable-for-Users*: Using the scheme for hundreds of accounts does not increase the burden on the user. As the mnemonic suggests, we mean "scalable" only from the user's perspective, looking at the cognitive load, not from a system deployment perspective, looking at allocation of technical resources.

U3 *Nothing-to-Carry*: Users do not need to carry an additional physical object (electronic device, mechanical key, piece of paper) to use the scheme. *Quasi-Nothing-to-Carry* is awarded if the object is one that they'd carry everywhere all the time anyway, such as their mobile phone, but not if it's their computer (including tablets).

U4 *Physically-Effortless*: The authentication process does not require physical (as opposed to cognitive) user effort beyond, say, pressing a button. Schemes that don't offer this benefit include those that require typing, scribbling or performing a set of motions. We grant *Quasi-Physically-Effortless* if the user's effort is limited to speaking, on the basis that even illiterate people find that natural to do.

U5 *Easy-to-Learn*: Users who don't know the scheme can figure it out and learn it without too much trouble, and then easily recall how to use it.

U6 *Efficient-to-Use*: The time the user must spend for each authentication is acceptably short. The time required for setting up a new association with a verifier, although possibly longer than that for authentication, is also reasonable.

U7 *Infrequent-Errors*: The task that users must perform to log in usually succeeds when performed by a legitimate and honest user. In other words, the scheme isn't so hard to use or unreliable that genuine users are routinely rejected.

U8 *Easy-Recovery-from-Loss*: A user can conveniently regain the ability to authenticate if the token is lost or the credentials forgotten. This combines usability aspects such as: low latency before restored ability; low user inconvenience in recovery (e.g., no requirement for physically standing in line); and assurance that recovery will be possible, for example via built-in backups or secondary recovery schemes. If recovery requires some form of re-enrollment, this benefit rates its convenience.

A.2 Deployability benefits

D1 *Accessible*: Users who can use passwords are not prevented from using the scheme by disabilities or other physical (not cognitive) conditions.

D2 *Negligible-Cost-per-User*: The total cost per user of the scheme, adding up the costs at both the prover's end (any devices required) and the verifier's end (any share of the equipment and software required), is negligible. The scheme is plausible for startups with no per-user revenue.

D3 *Server-Compatible*: At the verifier's end, the scheme is compatible with text-based passwords. Providers don't have to change their existing authentication setup to support the scheme.

D4 *Browser-Compatible*: Users don't have to change their client to support the scheme and can expect the scheme to work when using other machines with

an up-to-date, standards-compliant web browser and no additional software. In 2012, this would mean an HTML5-compliant browser with JavaScript enabled. Schemes fail to provide this benefit if they require the installation of plugins or any kind of software whose installation requires administrative rights. Schemes offer *Quasi-Browser-Compatible* if they rely on non-standard but very common plugins, *e.g.*,, Flash.

D5 *Mature*: The scheme has been implemented and deployed on a large scale for actual authentication purposes beyond research. Indicators to consider for granting the full benefit may also include whether the scheme has undergone user testing, whether the standards community has published related documents, whether open-source projects implementing the scheme exist, whether anyone other than the implementers has adopted the scheme, the amount of literature on the scheme and so forth.

D6 *Non-Proprietary*: Anyone can implement or use the scheme for any purpose without having to pay royalties to anyone else. The relevant techniques are generally known, published openly and not protected by patents or trade secrets.

A.3 Security benefits

S1 *Resilient-to-Physical-Observation*: An attacker cannot impersonate a user after observing them authenticate one or more times. We grant *Quasi-Resilient-to-Physical-Observation* if the scheme could be broken only by repeating the observation more than, say, 10–20 times. Attacks include shoulder surfing, filming the keyboard, recording keystroke sounds, or thermal imaging of keypad.

S2 *Resilient-to-Targeted-Impersonation*: It is not possible for an acquaintance (or skilled investigator) to impersonate a specific user by exploiting knowledge of personal details (birth date, names of relatives etc.). Personal knowledge questions are the canonical scheme that fails on this point.

S3 *Resilient-to-Throttled-Guessing*: An attacker whose rate of guessing is constrained by the verifier cannot successfully guess the secrets of a significant fraction of

users. The verifier-imposed constraint might be enforced by an online server, a tamper-resistant chip or any other mechanism capable of throttling repeated requests. To give a quantitative example, we might grant this benefit if an attacker constrained to, say, 10 guesses per account per day, could compromise at most 1% of accounts in a year. Lack of this benefit is meant to penalize schemes in which it is frequent for user-chosen secrets to be selected from a small and well-known subset (low min-entropy [11]).

S4 *Resilient-to-Unthrottled-Guessing*: An attacker whose rate of guessing is constrained only by available computing resources cannot successfully guess the secrets of a significant fraction of users. We might for example grant this benefit if an attacker capable of attempting up to 2^{40} or even 2^{64} guesses per account could still only reach fewer than 1% of accounts. Lack of this benefit is meant to penalize schemes where the space of credentials is not large enough to withstand brute force search (including dictionary attacks, rainbow tables and related brute force methods smarter than raw exhaustive search, if credentials are user-chosen secrets).

S5 *Resilient-to-Internal-Observation*: An attacker cannot impersonate a user by intercepting the user's input from inside the user's device (e.g., by key-logging malware) or eavesdropping on the cleartext communication between prover and verifier (we assume that the attacker can also defeat TLS if it is used, perhaps through the CA). As with *Resilient-to-Physical-Observation* above, we grant *Quasi-Resilient-to-Internal-Observation* if the scheme could be broken only by intercepting input or eavesdropping cleartext more than, say, 10–20 times. This penalizes schemes that are not replay-resistant, whether because they send a static response or because their dynamic response countermeasure can be cracked with a few observations. This benefit assumes that general-purpose devices like software-updatable personal computers and mobile phones may contain malware, but that hardware devices dedicated exclusively to the scheme can be made malware-free. We grant *Quasi-Resilient-to-Internal-Observation*

to two-factor schemes where both factors must be malware-infected for the attack to work. If infecting only one factor breaks the scheme, we don't grant the benefit.

S6 *Resilient-to-Leaks-from-Other-Verifiers*: Nothing that a verifier could possibly leak can help an attacker impersonate the user to another verifier. This penalizes schemes where insider fraud at one provider, or a successful attack on one back-end, endangers the user's accounts at other sites.

S7 *Resilient-to-Phishing*: An attacker who simulates a valid verifier (including by DNS manipulation) cannot collect credentials that can later be used to impersonate the user to the actual verifier. This penalizes schemes allowing phishers to get victims to authenticate to lookalike sites and later use the harvested credentials against the genuine sites. It is not meant to penalize schemes vulnerable to more sophisticated real-time man-in-the-middle or relay attacks, in which the attackers have one connection to the victim prover (pretending to be the verifier) and simultaneously another connection to the victim verifier (pretending to be the prover).

S8 *Resilient-to-Theft*: If the scheme uses a physical object for authentication, the object cannot be used for authentication by another person who gains possession of it. We still grant *Quasi-Resilient-to-Theft* if the protection is achieved with the modest strength of a PIN, even if attempts are not rate-controlled, because the attack doesn't easily scale to many victims.

S9 *No-Trusted-Third-Party*: The scheme does not rely on a trusted third party (other than the prover and the verifier) who could, upon being attacked or otherwise becoming untrustworthy, compromise the prover's security or privacy.

S10 *Requiring-Explicit-Consent*: The authentication process cannot be started without the explicit consent of the user. This is both a security and a privacy feature (a rogue wireless RFID-based credit card reader embedded in a sofa might charge a card without user knowledge or consent).

S11 *Unlinkable*: Colluding verifiers cannot determine, from the authenticator alone, whether the same user is authenticating to both. This is a privacy feature. To rate this benefit we disregard linkability introduced by other mechanisms (same user ID, same IP address, etc).

Appendix B

Password Manager Introduction Script

During the password manager user study (see Section 4) each participant was made to read a short introduction to the study and password managers. The content of the introduction text is included below:

B.1 Password managers

A password manager is an application that stores login information, usernames and passwords, for websites. Once these passwords are stored, the password manager will allow you to automatically fill in the username/password fields on websites you visit, eliminating the need for you to remember your password for each of your accounts. Today you'll be using a password manager

B.2 Firefox (FF-NMP)

The Firefox password manager is enabled by default on current versions of the Firefox browser. As users log into different sites, Firefox will prompt them to save their passwords and a button labeled "Remember" will store login information. The next time the page is loaded in the browser, Firefox will offer to automatically fill in the stored username and password.

There is no initial configuration required, but we'll pause briefly to let you ask any questions before you start using the password manager.

B.3 Firefox (FF-MP)

The Firefox password manager is enabled by default on current versions of the Firefox browser. As users log into different sites, Firefox will prompt them to save their passwords and a button labeled "Remember" will store login information. The next

time the page is loaded in the browser, Firefox will offer to automatically fill in the stored username and password.

Firefox allows users to configure a master password that will lock all their saved passwords. This prevents unauthorized access to their login information by anyone who does not know their master password. Imagine that the master password is a key to a lock and all website passwords are saved in a box and locked. The master password must be entered to unlock the box, before passwords can be automatically filled in.

To configure a master password, open the Firefox preferences and enable "use a master password". As a reminder, pretend that this is your browser, so pick a password that you think others can't guess, and that you could remember a week from now. Please take a few moments to look at the settings. Feel free to ask any questions.

B.4 Tapas

Tapas is a password manager that uses a smartphone for securely storing passwords. Tapas is installed as an app on a smartphone. The user's computer is configured to use the app (through an extension added to the Firefox browser). As users log into different sites, Firefox will prompt them to save their passwords with a button labeled "Save with Tapas." Pressing the button will send the login information to the Tapas app on the phone for storage. The next time the page is loaded in the browser, a user can tap on the account name in the Tapas app on the phone and the information will be sent to the computer. The browser will automatically fill in the stored username and password and log the user into the website.

Tapas protects the stored passwords. Imagine that Firefox generates the key to a lock and stores all website passwords in a locked box. The locked box is then sent to the phone for storage. When a user taps an account on their phone, the locked box is sent from the phone to the computer, which is the only entity that has the key to unlock it and pull out the username and password.

Appendix C

Informed Consent Form

In order to gain ethics approval at Carleton University for the password manager user study (see Section 4)) each participant had to sign an informed consent form reproduced below:

This study compares the security and usability of password managers. A password manager remembers the passwords you use to log in to sites so you don't have to. The password managers you'll help us evaluate today also help you fill in password boxes to make signing in easier for you.

As a participant you'll be helping us identify security and usability problems with the password managers. You'll be trying out two password managers: X and Y. You'll configure the password managers to remember you passwords, and then create accounts on three websites. You'll then log in to the websites.

As a participant in this study, you will create accounts on fake websites and use password managers. You will also be asked to complete questionnaires to give your opinion and perception of the software. We will record your mouse movements and keystrokes typed INCLUDING THE PASSWORDS YOU CHOOSE TO USE. The session will take at most one hour.

Data collected during the study will be stored electronically on our research servers with password protection. Access to this data is restricted to those researchers involved in the study.

At the end of the session, you will receive $15 as compensation for your time participating in this study. You will receive compensation even if you withdraw from the study.

There are no known risks associated with this study. As a participant, you may choose to be notified when the results of this study are published. If you wish to be

notified, please provide your email address below.

[_] I wish to be notified when the results of this study are published. Please contact me at the following email address _____

[_] I do not wish to be notified when the results of this study are published.

This project was reviewed and recieved ethics clearance by the Carleton University Ethics Board (REB). Contact information for the REB Chair follows:

Professor Antonio Gualtieri, Chair
Research Ethics Board
Carleton University Research Office
Carleton University
1125 Colonel By Drive
Ottawa, Ontario K1S 5B6
Tel: 613-520-2517
E-mail: ethics@carleton.ca

The principal researchers involved in this research are:

Daniel McCarney, MCS Student
School of Computer Science
Carleton University
1125 Colonel By Drive
Ottawa, Ontario K1S 5B6
Tel: +1-905-220-2721
Email: dmccarney@ccsl.carleton.ca

David Barrera, PhD Student
School of Computer Science
Carleton University
1125 Colonel By Drive
Ottawa, Ontario K1S 5B6
Tel: +1-613-261-9144
Email: dbarrera@ccsl.carleton.ca

Sonia Chiasson, Assistant Professor
School of Computer Science
Carleton University
1125 Colonel By Drive
Ottawa, Ontario K1S 5B6
Tel: (613) 520-2600 ext 1656
Email: chiasson@scs.carleton.ca

I, _____ volunteered to participate in this study on evaluating the usability and security of password managers

---
Signature of participant

Date

Signature of researcher

Date

Appendix D

User Study Experimenter Script

D.1 Create new account

First you'll be creating a new account on a website without using a password manager. The website you'll be registering on is loaded on the screen in front of you. Try to pretend like this is a real site and you don't want others to guess your password or log in on your behalf.

D.2 Tapas participants – explanation and setup

The password manager you'll be using is called Tapas (Stands for Tap Authentication with a Smartphone). Tapas stores part of your passwords on your smartphone, and part on your browser. This way, if someones steals your computer but not your phone, they won't be able to retriever your passwords. Similarly, if someone steals your phone but not your computer, they won't be able to log in.

We're providing a phone that has the Tapas app preinstalled. The browser you'll be using is Firefox, and it already has the Tapas extension needed for the phone and the browser to communicate. Follow the on-screen instructions to pair the phone to the browser.

D.3 FFMP participants – explanation and setup

Firefox includes a built-in password manager. When enabled, it offers to remember passwords for you and allows you to optionally lock your passwords with one single master password.

You will now enable the Firefox password manager and create a master password. Follow the on-screen instructions and choose a strong master password. Something that others can't guess, but you could still remember a week from today.

D.4 Create new account with manager

Now you'll register for a new account on $WEBSITE. Visit the address given to you and sign up for a new account. Make sure you allow the password manager to remember the password for you.

D.5 Migrate account into manager

You'll be putting the first password you signed up with into the password manager. We call this migrating from no password manager to the password manager. Visit the first site and try to log in. Complete the on-screen instructions to save your password into the password manager.

D.6 Change account password

$WEBSITE has had a breach and their entire password database has been stolen. You need to change your website on the site immediately. Ensure that the password manager saves the new password for $WEBSITE.

D.7 Log in using manager

Now you'll need to log in to $WEBSITE. Please visit the address given to you and log in. Remember the password manager has already saved your passwords, so try to use it.

Appendix E

Sample Recruitment Poster

Recruitment for the password manager study (see Section 4) was done in part through posters on the Carleton University campus. The content used for this poster is reproduced below:

Experiment title: Usability testing of Password Managers
Experimenters: Daniel McCarney and David Barrera
dmccarney@ccsl.carleton.ca & dbarrera@ccsl.carleton.ca
password-study@ccsl.carleton.ca

Location of Experiment: HP 5145

Brief Description:

Password managers are used to aid users in remembering and filling in website login information. Many password managers are available, each with differing strengths and weaknesses. The goal of this study is to investigate the usability and security of four popular password managers.

Participants will use different password managers during a half an hour session to provide their opinion and feedback. Individuals will receive $10 for their participation.

Interested parties should schedule an appointment by contacting Daniel McCarney or David Barrera at: password-study@ccsl.carleton.ca

To be eligble, participants should:

- Be 18 years of age or older

Ethical review: This research has been reviewed and approved by the Carleton

University Ethics Board.

Appendix F

Sample Recruitment Email

Recruitment for the password manager study (see Section 4) was done in part through email messages sent to recruitment mailing lists created for the purpose of recruiting Carleton University staff and students for studies with ethics approval. The content used for these emails is reproduced below:

Subject: Usability testing of Password Managers
Body text:

RE: Looking for volunteers

We are currently conducting a research study evaluating password managers and looking for interested volunteers. Password managers are used to aid users in remembering and filling in website login information. Many password managers are available, each with differing strengths and weaknesses. The goal of this study is to investigate the usability and security of four popular password managers.

Participants will use different password managers during a half an hour session to provide their opinion and feedback. Individuals will receive $10 for their participation.

To be eligble, participants should be 18 years of age or older and capable of travelling to the Carleton University campus for their designated time slot.

Ethical review: This research has been reviewed and approved by the Carleton University Ethics Board.

Interested parties should e-mail: password-study@ccsl.carleton.ca to schedule a study appointment.

Appendix G

Participant survey

Participant Questionnaire

Age ☐ Sex ☐

Education level (check all that apply)
- ☐ Completed high school
- ☐ Some university or college
- ☐ Some post-graduate degree (Masters, Diploma, PhD)

In what field did you study?

☐

If you are currently working (full or part time), indicate your job sector:

☐

What is your mother tongue:

☐

What other languages are you fluent in

☐

Online experience

How often do you use online banking? (e.g., bill payments, transfers, etc.)
○ Never ○ Daily ○ Weekly ○ Monthly ○ A few times a year ○ About once a year

How often do you buy things online?
○ Never ○ Daily ○ Weekly ○ Monthly ○ A few times a year ○ About once a year

Computer experience

What operating system(s) do you use on a regular basis? (check all that apply)
☐ Windows ☐ Mac OS ☐ Linux ☐ Don't know
Other: ☐

What web browser(s) do you use on a regular basis? (check all that apply)
☐ Chrome ☐ Firefox ☐ Internet Explorer ☐ Opera ☐ Don't know
Other: ☐

Mobile device experience

Do you have a personal cell phone/smartphone? ○ Yes ○ No

If you have a smartphone, what OS does it run?
○ Blackberry ○ iPhone ○ Android ○ Symbian ○ Windows Phone ○ Don't know
Other ☐

How would you rate your skills in using mobile phones?

Unfamiliar			Expert
1	2	3	4

Do you have a tablet? ◯Yes ◯No

If you have a tablet, what OS does it run?
◯Blackberry ◯iPhone ◯Android ◯Don't know
Other

How would you rate your skills in using tablets?

Unfamiliar Expert
1 2 3 4

Have you ever scanned a QR Code (2D barcode)
◯Yes ◯No ◯Unsure

Password experience

How many accounts do you have that require passwords?
◯0-10 ◯11-20 ◯21-30 ◯31-40 ◯41 or more ◯Don't know

How many different unique passwords do you have?
◯1-3 ◯4-6 ◯7-9 ◯10 or more ◯Don't know

How often do you change your password(s)?
◯Weekly ◯Monthly ◯A few times a year ◯Once a year ◯Less than once a year ◯Never

Are you concerned about the security of your passwords?
◯Yes ◯No ◯Unsure

Do you currently use a password manager?
◯Yes ◯No

If yes, which one?

Session Questionnaire

Experience with the password manager

Select the most appropriate response for the following statements:

Saving a password into the password manager was easy when:

-I created a new account

Disagree	1	2	3	4	Agree	N/A

-I was changing my password

Disagree	1	2	3	4	Agree	N/A

-I was migrating an existing password into the password manager

Disagree	1	2	3	4	Agree	N/A

Using the password manager interfered with:

-Login

Disagree	1	2	3	4	Agree	N/A

-Password change

Disagree	1	2	3	4	Agree	N/A

Using the password manager made logging in easier

Disagree	1	2	3	4	Agree	N/A

I selected stronger passwords when I knew they would be saved in a password manager

Disagree	1	2	3	4	Agree	N/A

A password manager makes my passwords more secure

Disagree	1	2	3	4	Agree	N/A

I would have a difficult time setting up the password manager on my own

Disagree	1	2	3	4	Agree	N/A

Were you surprised when you were prompted for you master password?

◯ Yes ○ No

Explain:

I enjoyed using the password manager

Disagree	1	2	3	4	Agree	N/A

I would use the password manager on my own computer

Disagree	1	2	3	4	Agree	N/A

In your own words, how would you describe your experience using the password manager

Mental model

Why do you think you are prompted for you master password?
Explain:

When you don't have a password manager, your passwords are stored in your brain. When you used the password manager, where were your passwords stored? (check all that apply)

Saved passwords for logging into websites
- ☐ On the computer
- ☐ On the internet
- ☐ On the phone
- ☐ It was not stored
- ☐ Not sure

Explain:

What would an attacker need to do to log in to one of your saved sites if they took your computer while it was turned off?
Explain:

Once all your passwords are saved into the password manager, how many times would you have to enter your master password?
- ☐ Every time I log in to a website that I had saved
- ☐ When I launch my browser (Firefox)
- ☐ The first time I log in to a saved site after launching my browser (Firefox)
- ☐ Never
- ☐ Not sure

If someone sat at your computer, under what circumstances could they log in to your saved sites?

☐ If I had already typed in my master password
☐ If they guessed my master password
☐ They can't log in at all.
☐ Not sure

If someone stole your computer while it was off, how confident are you that your saved passwords are safe?

Very confident	1	2	3	4	Not confident

You're using this password manager on a laptop and your laptop gets stolen within 5 minutes of you last using the password manager. Can the thief log in to your accounts using the saved passwords if:

☐ Your browser (Firefox) is open
☐ Your browser (Firefox) is closed but you are logged in to the computer
☐ You're logged out of the computer but your computer is on
☐ Your computer is turned off
☐ Not sure

The following questions are only for participants who used our password manager (Tapas)

Please rate how easy the pairing process (involving the 2D barcode) was:

Very easy	1	2	3	4	Very difficult

Bibliography

[1] AgileBits. 1Password Password Manager. `https://agilebits.com/onepassword`, 2013. [Online; accessed Feb 20th, 2013].

[2] Apple Inc. Safari Web Browser. `http://www.apple.com/ca/safari/`, 2013. [Online; accessed Feb 20th, 2013].

[3] A. Belenko and D. Sklyarov. Secure Password Managers and Military-Grade Encryption on Smartphones: Oh, Really? In *Blackhat Europe*, 2012.

[4] K. Bhargavan and A. Delignat-Lavaud. Web-based attacks on host-proof encrypted storage. In *Proceedings of the 6th USENIX Workshop on Offensive Technologies*, WOOT'12, pages 10–10, Berkeley, CA, USA, 2012. USENIX Association.

[5] K. Bicakci, N. B. Atalay, and H. E. Kiziloz. Johnny in internet café: user study and exploration of password autocomplete in web browsers. In *ACM Digital Identity Management*, 2011.

[6] K. Bicakci, N. B. Atalay, M. Yuceel, and P. C. van Oorschot. Exploration and field study of a password manager using icon-based passwords. In *Workshop on the Future of User Authentication and Authorization on the Web, Financial Cryptography and Data Security*, FC'11, pages 104–118, Berlin, Heidelberg, 2012. Springer-Verlag.

[7] K. Bicakci, M. Yuceel, B. Erdeniz, H. Gurbaslar, and N. B. Atalay. Graphical passwords as browser extension: implementation and usability study. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS '09, pages 39:1–39:1, 2009.

[8] R. Biddle, S. Chiason, and P. C. van Oorschot. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys*, 44(4):1–41, 2012.

[9] R. Biddle, M. Mannan, P. C. van Oorschot, and T. Whalen. User study, analysis, and usable security of passwords based on digital objects. *Trans. Info. For. Sec.*, 6(3):970–979, Sept. 2011.

[10] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh. Kamouflage: Loss-resistant password management. In *ESORICS*, 2010.

[11] J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy*, 2012.

[12] 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 *IEEE Symposium on Security and Privacy*, 2012.

[13] J. Bonneau and S. Preibusch. The password thicket: technical and market failures in human authentication on the web. In *Workshop on the Economics of Information Security*, 2010.

[14] X. Boyen. Halting password puzzles – hard-to-break encryption from human-memorable keys. In *USENIX Security*, 2007.

[15] E. Bursztein, C. Soman, D. Boneh, and J. C. Mitchell. Sessionjuggler secure web login from an untrusted terminal using session hijacking. In *World Wide Web(WWW)*, April 2012.

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

[17] U. Cil and K. Bicakci. gridWordX: Design, Implementation, and Usability Evaluation of an Authentication Scheme Supporting Both Desktops and Mobile Devices. In *Mobile Security Technologies Workshop (MoST)*, MOST '13. IEEE.

[18] M. Dietz, A. Czeskis, D. Balfanz, and D. S. Wallach. Origin-bound certificates: a fresh approach to strong client authentication for the web. In *USENIX Security*, 2012.

[19] O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[20] D. Florencio and C. Herley. A large-scale study of web password habits. In *WWW*, 2007.

[21] S. Gaw and E. W. Felten. Password management strategies for online accounts. In *SOUPS*, 2006.

[22] B. G. Glaser and A. L. Strauss. *The discovery of grounded theory: strategies for qualitative research*. Aldine Transaction, 2008.

[23] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, Aug. 1986.

[24] Google. The Android Mobile Operating System. `http://www.android.com/`, 2008. [Online; accessed Apr 5th, 2013].

[25] Google. Chrome Web Browser. `https://www.google.com/intl/en/chrome/browser/`, 2013. [Online; accessed Feb 20th, 2013].

[26] F. Haist, A. P. Shimamura, and L. R. Squire. On the Relationship Between Recall and Recognition Memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18:691–702, 1992.

[27] J. A. Halderman, B. Waters, and E. W. Felten. A convenient method for securely managing passwords. In *WWW*, 2005.

[28] T. Halevi and N. Saxena. On pairing constrained wireless devices based on secrecy of auxiliary channels: the case of acoustic eavesdropping. In *CCS*, 2010.

[29] E. Hayashi and J. Hong. A diary study of password usage in daily life. In *CHI*, 2011.

[30] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *NSPW*, 2009.

[31] C. Herley and D. Florencio. One Time Password Access to Any Server without Changing the Server. In *Proceedings of the 2008 Information Security Conference*, ISC '08. Springer, 2008.

[32] C. Herley and P. C. van Oorschot. A research agenda acknowledging the persistence of passwords. *IEEE Security & Privacy*, 10(1):28–36, 2012.

[33] IETF. PKCS #5: Password-based Cryptography Specification Version 2.0. `http://tools.ietf.org/html/rfc2898`, 2000. [Online; accessed Feb 28th, 2013].

[34] W. Jones. As Seen on TV: A So-Called Computer Security Product. `http://spectrum.ieee.org/riskfactor/telecom/security/as-seen-on-tv-a-socalled-computer-security-product`, Jan. 2013. [Online; accessed Apr 13th, 2013].

[35] A. Karole, N. Saxena, and N. Christin. A comparative usability evaluation of traditional password managers. In *ICISC*, 2011.

[36] Kaspersky Lab. Kaspersky Password Manager. `http://www.kaspersky.ca/products-services/home-computer-security/password-manager`, 2013. [Online; accessed Apr 9th, 2013].

[37] W. H. Kruskal and W. A. Wallis. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[38] LastPass Inc. LastPass Password Manager. `https://lastpass.com/`, 2013. [Online; accessed Feb 20th, 2013].

[39] M. E. Locasto, M. Massimi, and P. J. DePasquale. Security and privacy considerations in digital death. In *Proceedings of the 2011 workshop on New security paradigms workshop*, NSPW '11, pages 1–10. ACM, 2011.

[40] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[41] M. Mannan, D. Barrera, C. D. Brown, D. Lie, and P. C. van Oorschot. Mercury: Recovering forgotten passwords using personal devices. In *Financial Cryptography*, 2012.

[42] M. Mannan and P. van Oorschot. Digital objects as passwords. In *USENIX HotSec*, 2008.

[43] M. Mannan and P. van Oorschot. Using a personal device to strengthen password authentication from an untrusted computer. *Journal of Computer Security*, 19(4):703–750, 2011.

[44] M. Mannan and P. van Oorschot. Passwords for Both Mobile and Desktop Computers: ObPwd for Firefox and Android. In *USENIX ;login*, volume 37, pages 28–37, August 2012.

[45] D. McCarney, D. Barrera, J. Clark, S. Chiasson, and P. van Oorschot. Tapas: Design, Implementation, and Usability Evaluation of a Password Manager. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12. ACM, 2012.

[46] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, 2005.

[47] Microsoft. Internet Explorer Web Browser. `http://windows.microsoft.com/en-CA/internet-explorer/download-ie`, 2013. [Online; accessed Feb 20th, 2013].

[48] Mozilla Foundation. Mozilla Firefox Web Browser. `http://www.mozilla.org/en-US/firefox/`, 2013. [Online; accessed Feb 20th, 2013].

[49] B. Parno, C. Kuo, and A. Perrig. Phoolproof phishing prevention. In *Financial Cryptography*, 2006.

[50] J.-M. Picod and E. Bursztein. Reversing DPAPI and Stealing Windows Secrets Offline. In *BlackHat DC 2010*, Febuary 2010.

[51] D. Reichl. KeePass Password Safe. `http://keepass.info/`, 2013. [Online; accessed Apr 9th, 2013].

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

[53] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel: Design and usability study. *IEEE TIFS*, 6:306–313, 2011.

[54] N. Saxena and J. H. Watt. Authentication technologies for the blind or visually impaired. In *HotSec*, 2009.

[55] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[56] T. V. Srivastava and J. Purcell. Phishing and pharming – the deadly duo. `http://www.sans.org/reading_room/whitepapers/privacy/phishing-pharming-evil-twins_1731`, Jan. 2007. [SANS Information Security Reading Room, Online; accessed May 12th, 2013].

[57] F. Stajano. Pico: no more passwords! In *Proc. Sec. Protocols Workshop, LNCS*. Springer, 2011.

[58] M. Weir, S. Aggarwal, M. Collins, and H. Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 162–175, 2010.

[59] K.-P. Yee and K. Sitaker. Passpet: convenient password management and phishing protection. In *SOUPS*, 2006.