Systematization of Password Manager Use Cases and Design Paradigms

James Simmons
jsimmo58@vols.utk.edu
The University of Tennessee
Knoxville, Tennessee, USA

Oumar Diallo
osouleym@vols.utk.edu
The University of Tennessee
Knoxville, Tennessee, USA

Sean Oesch
toesch1@vols.utk.edu
The University of Tennessee
Knoxville, Tennessee, USA

Scott Ruoti
ruoti@utk.edu
The University of Tennessee
Knoxville, Tennessee, USA

ABSTRACT

Despite efforts to replace them, passwords remain the primary form of authentication on the web. Password managers seek to address many of the problems with passwords by helping users generate, store, and fill strong and unique passwords. Even though experts frequently recommend password managers, there is limited information regarding their usability. To aid in designing such usability studies, we systematize password manager use cases, identifying ten essential use cases, three recommended use cases, and four extended use cases. We also systematize the system designs employed to satisfy these use cases, designs that should be examined in usability studies to understand their relative strengths and weaknesses. Finally, we describe observations from 136 cognitive walkthroughs exploring the identified essential use cases in eight popular managers. Ultimately, we expect that this work will serve as the foundation for an explosion of new research into the usability of password managers.

CCS CONCEPTS

• **Security and privacy** → **Authentication**; **Usability in security and privacy**.

KEYWORDS

password managers, systematization, expert review

ACM Reference Format:
James Simmons, Oumar Diallo, Sean Oesch, and Scott Ruoti. 2021. Systematization of Password Manager Use Cases and Design Paradigms. In *Annual Computer Security Applications Conference (ACSAC '21), December 6–10, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3485832.3485889

1 INTRODUCTION

Despite efforts to replace password, they remain the primary form of authentication on the Web [2]. Still, passwords have many limitations: for example, users create easily guessed passwords [5, 23] and reuse the same password across accounts [4, 10, 20, 33]. Password managers seek to address these problems by helping users generate strong passwords (avoiding weak passwords), storing those passwords (encouraging unique passwords at every website), and filling those passwords (ensuring passwords are only sent to the correct website).

There have been several studies examining the usability of password managers [3, 8, 13, 16, 17, 21, 22, 29, 31]. However, as we will show in this work, they do not cover the full range of use cases supported by modern managers. Furthermore, most of these studies focus on high-level evaluations of password management, with few [3, 31] specifically identifying and studying the high-level designs (hereafter referred to as design paradigms) used to create password managers. This gap in the research literature means that design paradigms' relative strengths and weaknesses are unclear, preventing an informed approach to creating password managers.

To help guide the design of usability studies that address these gaps in the research, we systematize the use cases and design paradigms for password management. This systematization is guided by a review of password management documentation, examining 12 desktop and 12 mobile password managers, and reviewing the literature. Our systematization identifies seventeen use cases, categorized into essential use cases supported by all password managers, recommended use cases supported by a plurality of managers, and extended use cases supported by a minority of users or represent non-critical use cases. We also identified 77 design paradigms that can facilitate these use cases, 65 of which are built into deployed managers, three proposed in the research literature, and nine recommended based on our experiences evaluating managers.

While it is certainly possible that a different set of authors would end up with a somewhat different taxonomy, this is true of nearly all systematizations published at top security conferences. Regardless, this systematization is sufficient to help guide future usability studies exploring the usability of use cases that have not been previously studied. Similarly, we expect that our enumeration of design paradigms will make it easier for researchers to identify design paradigms to study and compare against each other in future studies.

As a first step in this direction, we used our systematization to perform cognitive walkthroughs of eight popular desktop managers, evaluating seventeen tasks covering the essential and recommended uses cases identified in our systematization. While not a replacement for user studies, these cognitive walkthroughs help identify low-hanging usability issues in the studied managers [14, 34]. Observations from these walkthroughs include significant issues entering credentials when autofill is unavailable, confusing interface designs, and challenges linking credentials to multiple websites.

In short, our contributions are,

(1) **Identification of password management use cases (§2)**. We identify seventeen different sets of tasks (i.e., use cases) that password managers support. We further categorize these use cases based on their importance to the password management experience. These use cases and their categorizations identify which functionality needs to be examined in user studies of password managers and can help inform the design of scenarios to use in those studies. Furthermore, we demonstrate that the majority of these use cases have not been previously studied.

(2) **Enumeration of design paradigms found in password managers and the research literature (§3)**. For each use case, we identify the system designs (i.e., design paradigms) used to support that use case. These design paradigms include those used in existing password managers, those proposed in the research literature, and several novel paradigms we propose in this work. This list of design paradigms can inform the creation of future user studies that compare these paradigms, revealing information about their relative advantages and disadvantages.

(3) **Initial exploration of usability challenges in password managers using cognitive walkthroughs (§4).** In this paper, we demonstrate that most use cases, and by extension design paradigms, have not previously been studied. As a first step, we performed cognitive walkthroughs of eight desktop password managers, completing tasks related to the majority of the use cases. Observations from these walkthroughs help identify low-hanging usability issues that need to be addressed.

2 USE CASES

Understanding how managers are intended to be used—i.e., their use cases—is critical in identifying what tasks should be studied in usability studies. While at a high level, the password manager life cycle can be described as supporting password generation, storage, and autofill [19], this list of activities is overly vague and does not fully cover the myriad ways in which modern password managers are used. In this section, we address this gap by providing a complete taxonomy of password manager use cases.

To identify use cases, we considered three sources. First, we read the documentation for password managers, identifying what companies consider to be use cases for their products. Second, we examined the most popular and downloaded publicly available desktop and mobile password managers (see Tables 1 and 2, respectively), identifying uses cases not described in the documentation. Third, we reviewed the research literature (though this did not reveal any use cases not discovered in the previous two sources).

Manager	Version
1Password X	1.17.0
Bitwarden	1.38.0
Chrome	7.1.0
Dashlane	6.1908.3
Edge	42.17134
Firefox	64
KeePassX	2.0.3
KeePassXC	2.3.4
LastPass	4.24.0
Opera	58.0.3135
RoboForm	8.5.6.6
Safari	12.0

Table 1: Examined desktop managers

Manager	iOS Version	Android Version
1Password	7.47	7.4
Bitwarden	2.3.1	2.2.8
Dashlane	6.2013.0	2.2006.3
Enpass	6.4.2	6.4.0
iCloud Keychain	13.3.1	—
Keeper	14.9.1	14.5.20
LastPass	4.8.0	4.1..4
Norton	6.8.78	6.5.2
RoboForm	8.9.2	8.10.4
SafeInCloud	20.0.1	20.2.1
Smart Lock	—	9.0
StrongBox	1.47.4	—

Table 2: Examined mobile managers

In total, we identify seventeen use cases. We group these use cases into three categories: essential (10), recommended (3), and extended (4). In the remainder of this section, we describe each use case in more depth. We also provide example tasks that could be leveraged in a usability study to examine these use cases.

2.1 Essential Use Cases

Essential use cases focus on the core password management lifecycle, and each is supported in some fashion by all the managers we examined.[1]

(E1) Setup manager: Users need to install and configure their password manager for first-time use. Sometimes this involves installing new software (an app or a browser extension), but it can also include enabling functionality built into the browser or

[1]The one exception to this rule is the Edge browser manager, which does not support password generation.

operating system. For most managers, it also includes setting up an online account to support credential syncing.

Tasks to explore this use case include setting up a new manager for the first time and setting it up on secondary devices.

(E2) Register credential: Users need to register credentials (username and password) and associated domain within their manager for later retrieval. Users may also need to link a credential with multiple apps or domains (e.g., `bitbucket.com` and `atlassian.com` use the same backend authentication system).

Tasks include directly registering a credential in the manager, logging in with credentials not stored in the manager (triggering autodetect mechanisms), or linking an already registered credential to additional domains or apps.

(E3) Update credential: After registering credentials, users need the ability to update those credentials. Updates can occur due to account recovery (e.g., needing to access the account when the manager was unavailable) or mandatory password resets.

Tasks include directly updating the credential in the manager or logging in with credentials different than those stored in the manager.

(E4) Remove credential: Users may wish to remove credentials that they no longer wish to have stored by the manager. They may also need to mass delete credentials if they are planning to migrate away from the manager.

Tasks include manually removing an obsolete credential or having the user migrate to a new manager.

(E5) Autofill credential: One key benefit of password managers is that the manager can automatically enter and submit credentials, obviating the need to enter credentials manually.

The primary task is to log in to a website/app. Furthermore, tasks could explore situations where there are multiple credentials associated with the website/app or multiple credentials for different subdomains of a common parent domain.

(E6) Manually enter credential: Users need to enter credentials on a range of devices, some of which may not have the manager available—for example, entering Netflix credentials on a smart TV or game console—or for which the user does not have a manager installed—for example, a work computer or phone. In these cases, they need to enter credentials stored in their manager manually. Additionally, even if the manager is available, autofill can fail, necessitating the manual entry of credentials.

Tasks should include manually typing credentials as well as copying and pasting credentials. The task of manually typing credentials can be conducted on a range of devices, with studies comparing the dynamics of entering different types of passwords—i.e., human-generated, simple machine-generated, and complex machine-generated—on these different devices.

(E7) Generate password: Generating credentials helps ensure that users have strong and unique credentials. Credentials could also be tailored to meet user needs, making them more memorable or easy to enter on different devices.

Tasks that could prompt password generation include creating a new account updating an existing account, or creating a PIN/password for a separate use case (e.g., generating a PIN for a credit card).

(E8) Sync credentials: Users often have multiple devices where they need to access stored credentials, such as desktops, laptops, tablets, and phones. Managers support this by allowing users to synchronize credentials between these devices.

Tasks include setting up a secondary device or creating/updating a credential on one device then immediately using it on another device.

(E9) Lock manager and (E10) Unlock manager: Users may need to lock their manager to prevent other users of their computer from accessing their credentials—for example, before letting a friend borrow their laptop. They may also automatically set their manager to deactivate after a set period or when some event occurs, such as closing the browser. Eventually, they will need to reactivate the manager before continuing to use it.

For each use case, there is a singular task to deactivate and activate the manager, respectively. Studies could also explore what situations would cause a user to feel the need to deactivate their manager. Similarly, longitudinal studies could seek to understand user perceptions of autolock by asking users about it during the unlock process.

2.2 Recommended Use Cases

Recommended use cases identify use cases that we believe—based on our review of the literature and personal experience—significantly improve the usability and utility of managers. Unlike the essential use cases, the uses cases are not supported by all managers.

(R1) Audit credentials: Auditing stored credentials can help users identify reused passwords, weak passwords, or credentials included in a password leak (commonly referred to as a "health check"). This service is beneficial for passwords manually created and stored by the user, as these are much more likely to be weak or reused than generated passwords.

Tasks include asking users to periodically examine their credentials or check credentials based on news of a password leak. When exploring this use case, it is essential to consider the case where the user stores primarily human-generated credentials [8, 16, 21], causing the credential audits to return a large number of results.

(R2) Modify settings: Users modify settings to customize the manager for themselves and to ensure secure behavior. For example, Oesch and Ruoti [19] found that in some managers, users need to manually require user interaction before autofill to prevent a range of credential scraping and XSS attacks.

Tasks include updating settings as part of the initial setup or disabling specific unsafe settings. Importantly, this latter should also explore how users find relevant settings, understand what they need to do, and ultimately change those settings.

(R3) Recover access: If users lose access to their password manager (e.g., forget their master password), this represents a significant challenge as they will lose access to all their accounts and need to reset those passwords (where that is even possible).

Managers can provide options to help users recover access to their accounts in these situations. Note, the password manager's security will only be as good as the security of the recovery mechanism.

The singular task is to recover access when access has been lost. In practice, it may be necessary to study this property in a longitudinal study as many of the recovery mechanisms we observed do not work in laboratory settings.

2.3 Extended Use Cases

Extended uses cases include situationally useful use cases, but for which we find no evidence that they are widely used in practice.

(X1) Migrate manager: Over time, users may switch between managers, such as moving from a browser-based manager to a more feature-rich and secure extension-based manager. In this case, they will need to export credentials from their old manager and then import them into the new manager. They will also need to discontinue the use of their old manager safely.

Tasks include migrating to a new tool or creating a backup of the credential vault (effectively migrating from the current manager to an offline store).

(X2) Share credentials: Users may need to share credentials with each other—for example, family members that share access to common video services like Netflix or Disney+. While this can be done by manually sending credentials across secondary communication channels (e.g., email), this use case focuses on manager-supported sharing.

Tasks include sharing a credential as well as updating, removing, and using shared credentials.

(X3) Manage identities: Managers can allow credentials to be segmented between multiple identities, only allowing access to the credentials for the currently select identity. Segmentation can include allowing a single user to separate credentials based on context—for example, one identity for work and one for home. Alternatively, multiple users could use this to share a single password manager account—for example, a wife and husband who only want to pay for one subscription to a password manager. Within this use case, we include the creation and modification of identities and switching between identities.

Tasks include creating initial identities to separate credentials, adding/removing identities, moving credentials between identities, and switching identities to log in to various websites/apps.

(X4) Store sensitive data: In addition to storing credentials, users may need to store other sensitive information such as addresses and payment information. Storage could include unstructured data (i.e., storing arbitrary strings) or structured data (e.g., phone number, address).

Tasks include entering new sensitive information, updating that information, viewing that information, and filling that information into forms.

2.4 Coverage in Prior Work

We analyzed prior work to determine which use cases they had considered in their studies. For the software security research, most research has focused on the security of *(E5) Autofill credential* [9, 15, 18, 19, 30, 32], though some have also considered *(X4) Store sensitive data* [11, 15, 19] and *(E7) Generate password* [19]. Interviews of password manager users [8, 21, 22] touch on a wide variety of topics, but the analysis of this data primarily focuses on three uses cases: *(E2) Register credential*, *(E5) Autofill credential*, and *(E7) Generate password*.

Finally, usability studies cover the broadest range of use cases, with their coverage summarized in Table 3. While there is consistent coverage of *(E2) Register credential*, *(E3) Update credential*, and *(E5) Autofill credential*, coverage for the remaining use case is either rare—three essential use cases are only covered by a single study—or completely absent—including three essential use cases. This lack of coverage clearly highlights a need for additional usability studies exploring the understudied use cases. In particular, there is a critical need for studies examining *(E6) Manually enter credential*, *(E7) Generate password*, *(E8) Sync credentials*, *(R1) Audit credentials*, and *(R3) Recover access*.

3 DESIGN PARADIGMS

In the last section, we established the need for additional studies of password manager use cases. However, new studies will be most impactful if they compare and contrast the ways—hereafter referred to as *design paradigms*)—in which these use cases can be satisfied. Comparing design paradigms makes it possible to identify their relative strengths and weaknesses, forming the scientific basis for designing and implementing modified and new password managers. As an example of the importance of studying design paradigms, we note that while the usability problems with secure email were long known [27, 34], it was not until research focused on comparing and contrasting design paradigms [1, 24, 25] that secure email was finally made usable [26].

To help with the creation of such studies, we set out to systematize the design paradigms. This systematization was done by thoroughly analyzing the designs used in 12 popular desktop managers and 12 popular mobile managers (see Tables 1 and 2, respectively). We also reviewed the literature to identify any design paradigms discussed there that might not be reflected in deployed managers. When identifying design paradigms, we are primarily concerned with the high-level design, not individual implementation details for each manager.

Tables 4 and 5 summarized the design paradigms identified in evaluation. It also includes a mapping showing which paradigms are supported by the eight popular desktop managers evaluated in our cognitive walkthroughs (see §4). This mapping helps to demonstrate which paradigms are widely supported, which are supported by a small number of managers, and which paradigms often appear together. In the remainder of this section, we discuss these paradigms in greater depth.

3.1 Essential Use Case Paradigms

(E1) Setup manager: There are three types of password managers, each of which use a different setup paradigm: extension-based—*(P2) install an extension*, thick client—*(P1) install an app*, and browser-based—*(P3) built into the browser*. The most common setup paradigm for extension-based and thick client

	Authors	Year	Citation	(E1) Setup manager	(E2) Register credential	(E3) Update credential	(E4) Remove credential	(E5) Autofill credential	(E6) Manually enter credential	(E7) Generate password	(E8) Sync credentials	(E9) Lock manager	(E10) Unlock manager	(R1) Audit credentials	(R2) Modify settings	(R3) Recover access	(X1) Migrate manager	(X2) Share credentials	(X3) Manage identities	(X4) Store sensitive data
Security	Gasti and Rasmussen	2012	[11]																	✔
	Fahl et al.	2013	[9]					✔												
	Li et al.	2014	[15]					✔												✔
	Silver et al.	2014	[30]					✔												
	Stock and Johns	2014	[32]					✔												
	Oesch and Ruoti	2020	[19]					✔		✔										✔
	Oesch et al.	2021	[18]					✔												
Interview	Fagan et al.	2017	[8]		✔			✔		✔										
	Pearman et al.	2019	[21]		✔			✔		✔										
	Ray et al.	2021	[22]		✔			✔	✔	✔										
Usability	Lyastani et al.	2017	[16]		✔	✔		✔		✔										
	Seiler-Hwang et al.	2019	[29]	✔	✔	✔		✔							✔					
	Huaman et al.	2021	[13]		✔	✔		✔												
	Chiasson et al.	2006	[3]		✔	✔		✔	✔											
	McCarney et al.	2012	[17]	✔	✔			✔												
	Stobert et al.	2020	[31]		✔	✔	✔	✔												

Table 3: Use cases examined in prior usability studies

managers is installing an extension; however, this switches to installing an app on mobile devices. Some desktop managers support installing both an app and extension, providing additional features and security if both setup paradigms are used. In some cases, the password manager is *(P4) built into the operating system*, with a browser serving as an interface for that password manager (e.g., macOS Keychain and Safari).

Finally, a manager often *(P5) requires a cloud account* to be created and used to access the password manager or sync passwords. In browsers, this is done using the same account used to log in to the browser. While browsers do support using the password manager without a linked account, this will prevent credential syncing and may disable other features.

(E2) Register credential: All managers support *(P1) manual registration* of credentials within the manager's UI, with most also able to *(P2) auto-detect registration* and offer to save the detected credential. On desktop, this detection is primarily limited to use with websites in a browser, whereas on mobile, detection works in both browsers and apps. Stobert et al. [31] proposed and tested a design paradigm where an *(P3) internal registration tool* can create online accounts, though this has not been implemented in any deployed manager. Finally, some managers allow users to *(P4) link additional domains or apps* to a credential, addressing the case where the same authentication backend is used by multiple domains/apps—for example, LAN websites using a common LDAP backend. This linking is helpful to avoid credentials being marked as reused in a credential audit.

(E3) Update credential: As with registering credentials, managers support *(P1) manual update* and may be able to *(P2) auto-detect update* as well. Some managers also provide an *(P3) internal update tool* that provides a one-click method for a user to update their credentials, both changing the credential in the manager and at the website/app.

(E4) Remove credential: All managers support *(P1) manual removal* of credentials. Unlike registration and updating credentials, managers do not support *(E4) Remove credential*, though it is easy to imagine how such a feature could be implemented. Stobert et al. [31] proposed and tested a design paradigm where an *(P3) internal registration tool* can delete online accounts, though this has not been implemented in any deployed manager. Some managers also allow users to entirely *(P4) wipe the credential vault*, allowing them to discontinue usage of the manager quickly.

(E5) Autofill credential: One key benefit of password managers is that they allow the manager to fill the credentials, with all managers supporting *(P1) autofill w/ interaction* with many also supporting the less secure *(P2) autofill w/o interaction* [15, 18, 19, 32]. Some managers also provide an *(P3) internal login tool* that will open the appropriate website and

		Desktop		Mobile	
Use Case	Paradigm	Count	Percent	Count	Percent
(E1) Setup manager	(E1-P1) Install an app	6	50%	10	83%
	(E1-P2) Install an extension	6	50%	0	0%
	(E1-P3) Built into the browser	4	33%	0	0%
	(E1-P4) Built into the operating system	1	8%	2	17%
	(E1-P5) Requires a cloud account	7	58%	11	92%
(E2) Register credential	(E2-P1) Manual registration	9	75%	11	92%
	(E2-P2) Auto-detect registration	11	92%	10	83%
	(E2-P3) Internal registration tool †	0	0%	0	0%
	(E2-P4) Link additional domains or apps	3	25%	2	17%
(E3) Update credential	(E3-P1) Manual update	12	100%	11	92%
	(E3-P2) Auto-detect update	11	92%	1	8%
	(E3-P3) Internal update tool	3	25%	0	0%
(E4) Remove credential	(E4-P1) Manual removal	12	100%	11	92%
	(E4-P2) Auto-detect removal *	0	0%	0	0%
	(E4-P3) Internal removal tool †	0	0%	0	0%
	(E4-P4) Wipe the credential vault	4	33%	0	0%
(E5) Autofill credential	(E5-P1) Autofill w/ interaction	11	92%	11	92%
	(E5-P2) Autofill w/o interaction	6	50%	1	8%
	(E5-P3) Internal login tool	5	42%	11	92%
	(E5-P4) Separate subdomains	5	42%	3	25%
	(E5-P5) Group subdomains	7	58%	0	0%
	(E5-P6) Relies on trusted path †	0	0%	0	0%
(E6) Manually enter credential	(E6-P1) Show stored credentials	11	92%	11	92%
	(E6-P2) Obfuscate password characters	11	92%	10	83%
	(E6-P3) Distinguish password characters	2	17%	7	58%
(E7) Generate password	(E7-P1) Manual generation	8	67%	10	83%
	(E7-P2) Auto-detect and generate	8	67%	1	8%
	(E7-P3) Manually set the PCP	7	58%	10	83%
	(E7-P4) Auto-detect the PCP	1	8%	0	0%
	(E7-P5) Download the PCP *	0	0%	0	0%
(E8) Sync credentials	(E8-P1) Fully automated sync	10	83%	10	83%
	(E8-P2) Manually copy the vault file	2	17%	0	0%
	(E8-P3) Partially automated sync *	0	0%	0	0%
(E9) Lock manager	(E9-P1) Manual lock	8	67%	10	83%
	(E9-P2) Timed auto-lock	8	67%	10	83%
	(E9-P3) Logout of the browser to lock	6	50%	11	92%
(E10) Unlock manager	(E10-P1) Unlock with master password	7	58%	10	83%
	(E10-P2) Unlock with biometric	2	17%	10	83%
	(E10-P3) Unlock with 2FA	9	75%	5	42%
	(E10-P4) Log into browser to unlock	4	33%	0	0%
	(E10-P5) Log into OS to unlock	1	8%	1	8%

*—Novel paradigm introduced in this paper †—Paradigm proposed in the research literature
Counts are of the number of managers that we evaluated which implement these design paradigms.

Table 4: Password Manager Design Paradigms—Essential Use Cases

complete the login operation. On mobile managers, this commonly happens within a custom browser built into the manager.

When multiple credentials are shown are linked to a given domain/app, the manager shows a selection dialog for users to select which credentials to use. When matching domains with credentials, some managers *(P4) separate subdomains* from each other, where others *(P5) group subdomains* together as a single domain.

Ruoti and Seamons [28] propose using a trusted pathway to improve the security of filled credentials. While these types of pathways have been explored for browser-based authentication [6, 7], they have never been integrated with a password manager.

(E6) Manually enter credential: To be able to enter credentials manually, it is critical that managers *(P1) show stored credentials* to users, allowing them to type those credentials elsewhere as well as copy and paste the credentials. Many managers *(P2) obfuscate password characters* displayed credentials, requiring users to click a button to reveal the plaintext credential. A more helpful design paradigm is to *(P3) distinguish password characters* when displaying the password—for example, highlight digits one color and characters another to help distinguish between `0` and `O` or `1` and `l`—making it easier to enter complex passwords correctly.

(E7) Generate password: Most managers support *(P1) manual generation* of passwords, though some also *(P2) auto-detect and generate* and fill passwords when needed during account creation. In terms of identifying the password composition policy (PCP), most managers allow users to *(P3) manually set the PCP*. Uniquely, Chrome attempts to *(P4) auto-detect the PCP*, though success is limited. While not supported by any managers, a paradigm allowing managers to *(P5) download the PCP* for websites/apps would help ensure that generators only create compliant passwords, improving their usability.

(E8) Sync credentials: Most managers provide *(P1) fully automated sync* of the password vault (i.e., credential store). For security reasons, several managers eschew the use of cloud-stored vaults, instead require users to *(P2) manually copy the vault file* between devices to sync the managers. For these security-conscious users, we propose a new paradigm providing a *(P3) partially automated sync*, wherein (a) the credential vault is encrypted with a key generated on the source device, (b) the encrypted vault is upload online (temporarily), (c) the destination device downloads the encrypted vault, and (d) the user enters the encryption key on the new device (e.g., scanning a QR code). We believe that the proposed paradigm satisfies the security requirements of non-cloud-based managers while being significantly more usable than relying on exporting and importing the vault file (even when that process is aided by other sync software such as Dropbox).

(E9) Lock manager: All non-browser-based managers require users to perform a *(P1) manual lock* of the password vault, preventing access to the credentials stored there until it is unlocked. These managers commonly also supported a *(P2) timed auto-lock* that triggers after some period of inactivity. Browser-based managers do not provide manual or timed lockouts, instead requiring users to *(P3) logout of the browser to lock* the vault.

(E10) Unlock manager: Users have a variety of methods for unlocking their password vault. Most commonly, users *(P1) unlock with master password*—a single, strong password or passphrase chosen by the user. Some managers also allow the user to *(P2) unlock with biometric* in place of the master password. For additional security, managers call also require users to *(P3) unlock with 2FA*. For browser-based and OS-based managers (i.e., KeyChain), users need only *(P4) log into browser to unlock* the vault or *(P5) log into OS to unlock* the vault, respectively.

3.2 Recommended Use Case Paradigms

(R1) Audit credentials: Credential audits (i.e., health checks) are intended to help users monitor the security of their stored credentials. They can *(P1) identify weak passwords*, *(P2) identify reused passwords*, *(P3) identify old passwords*,[2] and *(P4) identify compromised passwords*, helping users address the most significant issues with their passwords. We also propose a design paradigm to *(P5) identify unused passwords*, helping users identify online accounts that they no longer use and which could be deleted to reduce the user's online footprint and help protect their privacy.

Managers can require users to manually trigger the credential audit and *(P6) display a summary of audit results* when the audit is finished. Alternatively, managers can be continually running audit and *(P7) prompt with audit results* with results as they are identified, though currently, this is only done for compromised credentials in Chrome. As some users may be overwhelmed with the number of results in an audit—for example, a user that just adopted the manager and uploaded all their human-generated passwords to the vault—and so we propose a design paradigm to *(P8) prioritize audit recommendations*, preventing users from being overwhelmed and helping them focus on the most important items.

(R2) Modify settings: To increase the usability and correct usage of settings [25], many managers provide *(P1) inline setting documentation*, including details on how they work and the security implications of changing them.[3] Managers may also provide *(P2) searchable settings* allowing users to find relevant settings quickly. We suggest an additional paradigm that would *(P3) audit settings*, helping users identify ways they could further secure their manager (e.g., disabling password autofill without user interaction [15, 18, 19, 30, 32]).

(R3) Recover access: For many managers, if they have forgotten or lost access to their master password or other authentication factors, they cannot regain access to their vault. This sacrifices usability (people do forget and lose important things) to significantly improve the security of stored credentials. One way to address this problem is to allow users to print or otherwise store in a safe place a cryptographically secure *(P1) recovery code* that can be used to unlock the vault and allow users to reset their master password. Alternatively, as browser-based managers are tied to the browser's online account, they can use the browser's online account recovery mechanisms to restore access to the manager: *(P2) email/Phone-based recovery* and *(P3) customer service-based recovery*.

3.3 Extended Use Case Paradigms

(X1) Migrate manager: When migrating between managers, it is helpful if users can bring over their old credentials. This migration can be done by first performing a *(P1) manual export of credentials* in one manager and then a *(P2) manual import of credentials* in

[2]Old and weak credentials are more likely to be compromised and changing old credentials can remove access to an attacker if the credential had been previously compromised. Still, the importance of flagging old credentials is unclear and seems to run counter to updated NIST guidelines on password expiration. As such, we believe more research is needed on this topic.

[3]Managers also provide traditional, external documentation. Still, as this documentation is frequently ignored by users [25] and because it is not a part of the manager, we choose not to include it as a design paradigm.

Use Case	Paradigm	Desktop		Mobile	
		Count	Percent	Count	Percent
(R1) Audit credentials	(R1-P1) Identify weak passwords	6	50%	7	58%
	(R1-P2) Identify reused passwords	5	42%	8	67%
	(R1-P3) Identify old passwords	2	17%	0	0%
	(R1-P4) Identify compromised passwords	6	50%	8	67%
	(R1-P5) Identify unused passwords *	0	0%	0	0%
	(R1-P6) Display a summary of audit results	6	50%	8	67%
	(R1-P7) Prompt with audit results	2	17%	0	0%
	(R1-P8) Prioritize audit recommendations *	0	0%	0	0%
(R2) Modify settings	(R2-P1) Inline setting documentation	7	58%	8	67%
	(R2-P2) Searchable settings	4	33%	1	8%
	(R2-P3) Audit settings *	0	0%	3	25%
(R3) Recover access	(R3-P1) Recovery code	2	17%	0	0%
	(R3-P2) Email/Phone-based recovery	4	33%	2	17%
	(R3-P3) Customer service-based recovery	4	33%	0	0%
(X1) Migrate manager	(X1-P1) Manual export of credentials	11	92%	2	17%
	(X1-P2) Manual import of credentials	11	92%	4	33%
	(X1-P3) Automatic import of credentials	6	50%	0	0%
	(X1-P4) Disable the prior manager *	0	0%	0	0%
(X2) Share credentials	(X2-P1) Share credentials with other users	5	42%	6	50%
	(X2-P2) Share credentials with non-users *	0	0%	2	17%
	(X2-P3) Apply access control	5	42%	6	50%
(X3) Manage identities	(X3-P1) Manage identities	3	25%	2	17%
	(X3-P2) Protect an identity with a PIN *	0	0%	0	0%
(X4) Store sensitive data	(X4-P1) Store unstructured data	8	67%	10	83%
	(X4-P2) Store structured data	11	92%	11	92%
	(X4-P3) Autofill for structured data	9	75%	11	92%

*—Novel paradigm introduced in this paper †—Paradigm proposed in the research literature
Counts are of the number of managers that we evaluated which implement these design paradigms.

Table 5: Password Manager Design Paradigms—Recommended and Extended Use Cases

the second manager. Users can also leverage manual export and import of credentials to allow them to create offline backups of their vault. To automate this process, the destination manager detects the source manager and executes an *(P3) automatic import of credentials* from the source manager. We also propose a paradigm where the new manager would *(P4) disable the prior manager*, preventing it from interfering with the new manager.

(X2) Share credentials: Users can share their credentials using various standard channels (e.g., email, texting, verbally). However, managers can also support sharing by allowing users to *(P1) share credentials with other users* of the same manager. While not currently supported, it is easy to imagine a design paradigm where users *(P2) share credentials with non-users*—for example, by sending non-users a link that lets them access a cloud portal with the stored credentials. If the recipient uses a password manager, it could also be designed to ingest these URLs and make them directly accessible in the recipient's vault, even though the two users have different managers. The benefit of sharing credentials using the manager is that it allows users to *(P3) apply access control* to shared credentials.

(X3) Manage identities: Managers can allow users to *(P1) manage identities*, named groupings of credentials with only credentials from the currently selected identity being accessible in the manager's interface. To help increase the security of credentials stored associated with identities, we propose a design paradigm that would allow users to *(P2) protect an identity with a PIN*. For example, this could be used on a family password manager account to segment and protect credentials related to shopping from accounts kids can access.

(X4) Store sensitive data: In addition to credentials, managers can allow the users to *(P1) store unstructured data* (e.g., text blobs) or *(P2) store structured data* (e.g., phone numbers, addresses). For structured data, managers can also support *(P3) autofill for structured data* for websites and apps.

3.4 Discussion

In this subsection, we identified a large number of design paradigms (77) that have either been used in deployed managers (65), detailed in the research literature (3), or which we recommended based on our analysis of the existing paradigms (9). While we believe this list of paradigms to be complete, we recognize that it will not necessarily stay that way as researchers study these existing paradigms and then develop and study new paradigms. We do not view this as a flaw with this research artifact but rather as its intended purpose (i.e., to spur new research into paradigms).

Critically, we note that while there have been usability studies of password managers exploring a handful of these paradigms, the majority of paradigms have not been critically examined in the research. Even when paradigms have been studied, they have not been studied comparatively, leaving their relative strengths and weaknesses unclear. For all these reasons, we expect that this paradigm list can spur a substantial body of impact research into the usability and utility of password managers.

There are no mutually exclusive paradigms, though it might not be possible to use both simultaneously in some cases. For example, it is not possible to *(P4) separate subdomains* and *(P5) group subdomains*, though a manager could allow users to select which they prefer. However, if a manager supported too many paradigms simultaneously, this would likely lead to confusion. Similarly, some paradigms might have unexpected interactions. More research is needed to identify the ideal set of default paradigms, select the paradigms that should be offered as options, and determine which should not be used together.

Comparing the percentage of managers that implement a paradigm on mobile and desktop, we find that differences are primarily attributable to the types of managers examined on each. For example, we do not examine the built-in managers for any mobile browsers. Still, we observed a couple of interesting points when comparing paradigm usage between desktop and mobile implementations for the same manager. First, we find that mobile managers often have fewer features than their desktop counterparts. Second, when both the desktop and mobile managers share a design paradigm, the implementation is usually identical, with little to no customization made to address mobile devices' unique constraints and abilities. We hypothesize that these two issues help explain the poor usability of mobile password managers found by Seilew-Hwang et al. [29]. Future research is needed to investigate how existing paradigms can be better tailored to support mobile apps or whether new, mobile-specific paradigms are needed.

4 COGNITIVE WALKTHROUGHS

As part of our systematization of password manager use cases and design paradigms, we conduct an initial assessment of the strengths and weaknesses of the identified paradigms. We do this using by conducting cognitive walkthroughs [14], a form of expert review, for eight popular desktop managers: browser-based (Chrome, Edge, Firefox, Safari), extension-based (1PasswordX, Dashlane, LastPass), and app-based (KeePass). We evaluate these managers across 17 different tasks (a total of $8 * 17 = 136$ evaluations), covering all the essential and recommended use cases identified in our systematization. Full descriptions of each task and its associated use cases are given in Appendix A.

We choose to investigate desktop managers as there have not been prior usability studies of the user interface for desktop managers. We decided to conduct cognitive walkthroughs for two reasons. First, cognitive walkthroughs are highly effective at identifying low-hanging usability issues [34]. Second, cognitive walkthroughs allow for exploring a much larger collection of tools and use cases than would be feasible in most user studies—for example, we evaluated eight managers across 17 tasks, far exceeding the number of systems and tasks tested in most user studies.

In the remainder of this section, we detail our methodology, share observations and lessons learned from the walkthroughs, and identify topics needing further research.

4.1 Methodology

Cognitive walkthroughs are a form of expert review in which a usability expert completes a given task with an assigned tool. While completing this task, the evaluator will role-play, responding to the tool's interface and taking actions only as the role-played user would [14]. As they complete the tasks, the evaluators apply a think-aloud protocol, describing what they see, identifying how they discover features, and describing any confusion they encounter as they complete the task. Finally, the expert evaluators would be debriefed periodically during this process by the entire research team, allowing for further probing of their experiences.

In this work, two members of our research team (the first two authors) conducted these reviews. Both had received training on conducting cognitive walkthroughs, and all walkthroughs were reviewed by the research team, ensuring their quality. While completing tasks, the evaluators were instructed to role-play mildly technical users (roughly what could be expected of a college graduate). While this role is far from representative of all users, we believe it roughly represents the demographic targeted by most managers.

For all managers but Safari, tasks were completed on a virtual machine running Windows 10. The virtual machines used snapshots to ensure a consistent starting state for evaluating each manager and task. For Safari, we used a MacBook Air running macOS 10.15, taking care to reset its state between each set of tests. Walkthroughs were recorded (audio and video) and used as part of the analysis of results.

4.2 Observations

Below we discuss observations and lessons learned from the cognitive walkthroughs.

4.2.1 Using Credentials on Secondary Devices. In one task, evaluators needed to enter a generated password stored in their desktop manager into a website on a mobile device (*(E6) Manually enter credential*). Due to the difficulty of reading and entering these passwords, they noted that this was the most difficult and annoying of all the tasks they completed. Only 1Password X aided this process, highlighting characters based on character class (*(E6-P3) Distinguish password characters*), making it easier to read the password to be entered on the mobile device. Even in this case, entering generated passwords was still challenging due to the use of symbols, which require additional effort to enter on mobile keyboards.

At first glance, it might seem possible to address these challenges by having the user install their manager on the mobile device. While this approach works in that situation, it is impossible to install a password manager on the myriad of users of smart devices and IoT devices where users need to enter credentials. Similarly, users also need to enter passwords on shared computers where it is not feasible to install the manager.

As such, there is a need for more research to address this use case. Research could explore why more devices do not provide password-alternative authentication mechanisms, such as providing users with a login URL that they can visit on their phone (which does support a manager) to log in to the device. Also, research could investigate password generation that factors in the devices where the password will be entered, making it easier to enter generated passwords. This research could be modeled after and extend the work of Greene et al. [12], which examined generating passwords that were easier to enter on mobile keyboards.

4.2.2 Fatiguing Setup. When setting up the managers, evaluators found the extension-based managers to have the most complicated setup process (*(E1) Setup manager*). In addition to downloading an extension or desktop app (*(E1-P1) Install an app* or *(E1-P2) Install an extension*), the evaluators also need to register an online account (*(E1-P5) Requires a cloud account*) and create their initial credential store. This process can take an extended period, with the evaluators noting fatigue during this process. Such fatigue could turn some users away from completing the setup of extension-based managers. This dropout would be problematic, as extension-based managers are more secure than browser-based managers [19], with anything that inhibits migration from browser-based managers to extension-based managers being problematic. While it is unclear exactly how to solve this problem, it deserves more attention in future usability studies.

4.2.3 Credential Linking Challenges. When evaluators completed a credential audit (*(R1) Audit credentials*), several passwords were marked as reused. However, some of these passwords were not reused, it was just the case that websites at different domains used the same authentication backend (e.g., atlassian.com and bitbucket.org) and thus had the same stored password. Such false positives in an audit are disadvantageous, as not only were they frustrating to the evaluators, but they could also cause users to not trust audit results and overlook actual issues. This issue could be resolved in the two managers that supported the *(E2-P4) Link additional domains or apps* paradigm, though finding this functionality was difficult and it was hard to use. This paradigm needs to be implemented more widely and surfaced to users more effectively.

4.2.4 Interface Designs. While completing tasks, evaluators were consistently disappointed with various aspects of the managers' interfaces. Most commonly, problems arose due to confusion when attempting to locate features or settings (*(R2) Modify settings*). This difficulty was often caused by a combination of those features and settings being deeply nested in menus or named using non-obvious vernacular. For example, Chrome groups manager features under the label "autofill", which may not be a meaningful term to many users (as it was not initially to our evaluators), as opposed to "privacy and security", where many users might expect it. Issues such as these caused our evaluators to spend considerable time locating these items, and for some users may prevent them from realizing the feature or setting exists. This problem was somewhat alleviated in managers that supported *(R2-P2) Searchable settings*, but even in this case, non-obvious wording could make it difficult for users to find relevant settings.

Another point of confusion in some managers was an over-reliance on icons in place of text. While this saves screen real estate, it was not always obvious what icons meant. Together, these results demonstrate that more research is needed to explore how best to communicate the manager's features to the user.

On a positive note, the evaluators noted that several managers used password strength meters within their password generators (*(E7) Generate password*). These meters included visual indicators and colors to describe the strength of the password. The evaluators noted that this gave them confidence that their selected generation settings were secure. Ideally, managers could find ways to incorporate more of these simple, easy-to-understand indicators through the manager to help users identify when they are using the managers correctly.

4.2.5 Operating System-Based Manager. Of the tested managers, only Safari uses a manager provided by the operating system: Keychain Access. Passwords can be managed directly in Keychain Access or indirectly through Safari, with both programs exposing slightly different functionality. This is an example of the paradigm (*(E1-P4) Built into the operating system*). Our cognitive walkthroughs revealed that MacOS' and Safari's implementation of this paradigm is flawed. The primary issue is that when using KeyChain Access through Safari, dialog messages consistently refer to keychains, but never describe what a keychain is, what role it plays in password management, or how to access it. While this is unlikely to cause an issue for technical users already aware of KeyChain Access, less technical users could be confused by these messages and are more likely to build incorrect mental models of how password storage works, potentially leading to security issues in some edge cases. For example, users may not realize that even if they log out Safari, the usernames and domains associated with stored credentials can still be viewed through KeyChain Access. While the passwords are not visible in KeyChain Access without the account password, this might reveal more information that the user intends (e.g., if they share their computer with a friend).

4.2.6 Browser-Based Managers. Through the cognitive walkthroughs, evaluators noticed that the browser-based managers had the most limited functionality—i.e., implementing the smallest number of design paradigms. Even when they did support a given design paradigm, they often did so in a less feature-rich fashion. While the evaluators were happy with the browser-based managers overall, they felt they were especially weak compared to the other managers tested. These results regarding limited functionality mirror prior results that show that modern browser-based managers are often less secure than other managers [19]. Considering these results, we believe that while browser-based managers serve as an easy transition into password management, users should be steered to more feature-rich and secure managers over time.

4.2.7 *Browser-Based Manager Vulnerability.*

To lock the credential vault for browser-based managers (*(E9) Lock manager*), it is necessary to log out of the browser itself (*(E9-P3) Logout of the browser to lock*). When doing so, the evaluators noticed an edge case in which a user's credentials would remain accessible in the browser even after logging out of the browser. First, users would need to begin using the browser's manager without logging into the browser. Doing so creates a local credential store for the user's credentials. Next, the user would need to log into the browser, causing the local store to be synced with the cloud store. Surprisingly, the local store is not removed, only hidden; moreover, the local store is kept in sync with the cloud store. Lastly, if a user logs out of their browser, access to the cloud store will be revoked, but the local store (which contains credentials synced from the cloud store) will become unhidden and accessible to anyone that opens the browser. While it is unclear how often this edge case would be encountered in real-world usage, it is still something that browser-based managers should address.

4.3 Limitations

Our cognitive walkthroughs suffer the same challenge to ecological validity faced by all forms of expert review: the evaluators are not the users that they role-play. There will always exist differences between how the evaluators think and use tools as compared to actual users. Additionally, as evaluators complete more tasks with additional tools, it becomes harder to think and act like new users. As our cognitive walkthroughs intend to provide an initial, non-definitive evaluation of design paradigm usability, we think this is an acceptable limitation. Indeed, future research should replicate and expand upon our evaluation, making liberal use of user testing. We hope that this paper serves as a basis to encourage and promote such research.

5 RELATED WORK

We are unaware of other efforts to systematize password management use cases or design paradigms. Instead, in this section, we summarize efforts to measure human perceptions and motivations regarding password management as well ass discuss usability studies of password managers. We also describe how this type of research could benefit from our systematization.

5.1 User Perceptions and Motivations

There has been a line of research examining users' perceptions and behaviors related to password management [8, 21, 22]. First, Fagan et al. [8] surveyed both users and non-users of managers, trying to understand why they had or had not adopted it. They found that adopters primarily focused on usability benefits, and those that did not adopt managers had security concerns. Similarly, Pearman et al. [21] interviewed both users and non-users of password managers. They found that adopters of browser-based managers favored convenience, whereas adopters of extension-based managers were driven more by security. Ray et al. [22] replicated the work of Pearman et al. but changed the study population to focus on adults at least 60 years. These older adults indicate that fear of a single point of failure, desire to retain control of personal information, and concern about not having the manager available when needed impeded their adoption of password managers. In contrast, recommendations from family members were often crucial to adoption by these older users.

Relationship to our work: While these works lightly touch on many use cases, they primarily focus on three: *(E2) Register credential*, *(E5) Autofill credential*, and *(E7) Generate password.* We find no evidence that these studies ask users about the recommended or extended uses cases identified in our systematization. Future studies on user perceptions and behaviors could use our list of use cases to help increase the coverage of password management-related topics discussed in the interviews, helping better understand how users view and engage with the full range of functionality provided by modern managers.

5.2 Usability Studies

There are three usability studies of password managers deployed in the wild. Lyastani et al. [16] instrumented a manager to measure usage in the wild, gathering data about registering, updating, autofilling credentials, as well as generating passwords. Their results demonstrated positive security outcomes for adopting a password manager, but that many users eschewed using the password generator. Huaman et al. [13] investigated usability issues reported on GitHub. They found that password managers sometimes do a poor job of identifying password fields and that many websites implement non-standard interfaces making this process even more challenging. Seiler-Hwang et al. [29] conducted a laboratory user study exploring the usability of four smartphone password managers. They found significant usability issues, particularly regarding poor integration of the manager with apps and browsers, with users rating the apps as barely acceptable.

There are also three usability studies of password manager research prototypes. In 2006, Chiasson et al. [3] evaluated the usability of two password manager prototypes, prototypes that had little to no resemblance to modern managers. Their study found that while users reported tasks as being "very easy" to complete, users also failed to complete tasks correctly, suggesting an inaccurate mental model of the design paradigms used in these prototypes. McCarney et al. [17] proposed the design of a password manager that secures the credential vault using a mobile device. They conducted a usability study of this manager, finding that users preferred it to the Firefox password manager. Stobert et al. [31] proposed the design of a password manager that moved all account management tasks into the browser—e.g., users could create, updated, and even delete their online accounts from within the manager. An initial user study of this manager found that users viewed this approach favorably and believed it could help them more securely manage their accounts. Design paradigms *(E2-P3) Internal registration tool* and *(E4-P3) Internal removal tool* are derived from this research.

Relationship to our work: As shown previously in Table 3, these studies only consider a minority of the use cases in our systematization. This indicates a substantial need for additional studies of password manager usability covering these use cases. Additionally, our systematization of design paradigms identifies fruitful possibilities for conducting studies that compare and contrast the strengths and weaknesses of these paradigms.

6 CONCLUSION

In this paper, we described a systematization of password manager use cases and design paradigms. As a result of this effort, we discovered that most use cases had not been previously examined in a usability study. Moreover, we are unaware of any research comparing design paradigms or multiple implementation approaches for a single paradigm, meaning that these design's relative strengths and weaknesses and the best way to implement them remain unknown. In making this observation, we are not denigrating the excellent existing research into password manager usability but instead noting that this is an area rife with research opportunities. Within this environment, our systematization serves as an invaluable reference for constructing future usability studies.

Based on our systematization, we conducted cognitive walkthroughs of eight different managers, covering all essential and recommended use cases. Observations from these walkthroughs include significant issues typing credentials when autofill is unavailable, confusing interface designs, and challenges linking credentials to multiple websites.

We conclude with an observation taken from considering both our systematization and our cognitive walkthroughs. We find that browser-based managers have limited feature sets, failing to implement many of the design paradigms provided by extension-based paradigms, limiting the usability and security of browser-based managers. However, browser-based managers are more trivial to set up and begin using than the more feature-rich extension-based managers. This situation could explain why convenience-oriented users are more likely to adopt and continue using browser-based managers [8, 21].

Still, this situation is problematic as the lackluster security of browser-based managers leaves user credentials at risk [19]. While one approach focuses on making browser-based managers more secure or extension-based managers easier to set up (both of which are admirable goals), we instead recommend that browser- and extension-based managers work together to help users adopt and transition to secure password managers. In this approach, browser-based managers would continue to provide a seamless onboarding process, maximizing the number of users who adopt password managers. After users become accustomed to using a password manager, the browser-based managers would begin nudging users to adopt more functional and secure extension-based browsers. The extension-based browsers would automate the transition from the browser-based manager to the extension-based manager, helping users recognize and leverage the increased utility and security of the extension-based managers. Ultimately, we hypothesize that such a two-step process would be more successful than having any single manager optimize for both adoption and long-term use simultaneously.

REFERENCES

[1] Wei Bai, Moses Namara, Yichen Qian, Patrick Gage Kelley, Michelle L Mazurek, and Doowon Kim. 2016. An inconvenient trust: User attitudes toward security and usability tradeoffs for key-directory encryption systems. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*. 113–130.

[2] Joseph Bonneau. 2012. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*. IEEE, 538–552.

[3] Sonia Chiasson, Paul C van Oorschot, and Robert Biddle. 2006. A Usability Study and Critique of Two Password Managers. In *USENIX Security Symposium*, Vol. 15. 1–16.

[4] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. 2014. The Tangled Web of Password Reuse. In *NDSS*, Vol. 14. 23–26.

[5] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. 2010. Password strength: An empirical analysis. In *2010 Proceedings IEEE INFOCOM*. IEEE, 1–9.

[6] Rachna Dhamija and J Doug Tygar. 2005. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 symposium on Usable privacy and security*. 77–88.

[7] Saba Eskandarian, Jonathan Cogan, Sawyer Birnbaum, Peh Chang Wei Brandon, Dillon Franke, Forest Fraser, Gaspar Garcia, Eric Gong, Hung T Nguyen, Taresh K Sethi, et al. 2019. Fidelius: Protecting user secrets from compromised browsers. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 264–280.

[8] Michael Fagan, Yusuf Albayram, Mohammad Maifi Hasan Khan, and Ross Buck. 2017. An investigation into users' considerations towards using password managers. *Human-centric Computing and Information Sciences* 7, 1 (2017), 12.

[9] Sascha Fahl, Marian Harbach, Marten Oltrogge, Thomas Muders, and Matthew Smith. 2013. Hey, you, get off of my clipboard. In *International Conference on Financial Cryptography and Data Security*. Springer, 144–161.

[10] Dinei Florencio and Cormac Herley. 2007. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 657–666.

[11] Paolo Gasti and Kasper B Rasmussen. 2012. On the security of password manager database formats. In *European Symposium on Research in Computer Security*. Springer, 770–787.

[12] Kristen K Greene, John Michael Kelsey, and Joshua M Franklin. 2016. *Measuring the usability and security of permuted passwords on mobile platforms*. US Department of Commerce, National Institute of Standards and Technology.

[13] N. Huaman, S. Amft, M. Oltrogge, Y. Acar, and S. Fahl. 2021. They Would do Better if They Worked Together: The Case of Interaction Problems Between Password Managers and Websites. In *2021 2021 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1626–1640. https://doi.org/10.1109/SP40001.2021.00094

[14] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research methods in human-computer interaction*. Morgan Kaufmann.

[15] Zhiwei Li, Warren He, Devdatta Akhawe, and Dawn Song. 2014. The Emperor's New Password Manager: Security Analysis of Web-based Password Managers. In *USENIX Security Symposium*. 465–479.

[16] Sanam Ghorbani Lyastani, Michael Schilling, Sascha Fahl, Michael Backes, and Sven Bugiel. 2018. Better managed than memorized? Studying the Impact of Managers on Password Strength and Reuse. In *27th USENIX Security Symposium*. 203–220.

[17] Daniel McCarney, David Barrera, Jeremy Clark, Sonia Chiasson, and Paul C Van Oorschot. 2012. Tapas: design, implementation, and usability evaluation of a password manager. In *Proceedings of the 28th Annual Computer Security Applications Conference*. 89–98.

[18] Sean Oesch, Anuj Gautam, and Scott Ruoti. 2021. The Emperor's New Autofill Framework: A Security Analysis of Autofill on iOS and Android. *CoRR* abs/2104.10017 (2021). arXiv:2104.10017 https://arxiv.org/abs/2104.10017

[19] Sean Oesch and Scott Ruoti. 2020. That Was Then, This Is Now: A Security Evaluation of Password Generation, Storage, and Autofill in Browser-Based Password Managers. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/usenixsecurity20/presentation/oesch

[20] Sarah Pearman, Jeremy Thomas, Pardis Emami Naeini, Hana Habib, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Alain Forget. 2017. Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 295–310.

[21] Sarah Pearman, Shikun Aerin Zhang, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2019. Why people (don't) use password managers effectively. In *Fifteenth Symposium On Usable Privacy and Security (SOUPS 2019). USENIX Association, Santa Clara, CA*. 319–338.

[22] Hirak Ray, Flynn Wolf, Ravi Kuber, and Adam J. Aviv. 2021. Why Older Adults (Don't) Use Password Managers. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/ray

[23] Shannon Riley. 2006. Password security: What users know and what they actually do. *Usability News* 8, 1 (2006), 2833–2836.

[24] Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark O'Neill, Elham Vaziripour, Justin Wu, Daniel Zappala, and Kent Seamons. 2016. " We're on the Same Page" A Usability Study of Secure Email Using Pairs of Novice Users. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 4298–4308.

[25] Scott Ruoti, Jeff Andersen, Travis Hendershot, Daniel Zappala, and Kent Seamons. 2016. Private Webmail 2.0: Simple and easy-to-use secure email. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 461–472.

[26] Scott Ruoti, Jeff Andersen, Tyler Monson, Daniel Zappala, and Kent Seamons. 2018. A comparative usability study of key management in secure email. In *Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018)*. 375–394.
[27] Scott Ruoti, Jeff Andersen, Daniel Zappala, and Kent Seamons. 2015. Why Johnny still, still can't encrypt: Evaluating the usability of a modern PGP client. *arXiv preprint arXiv:1510.08555* (2015).
[28] Scott Ruoti and Kent Seamons. 2017. End-to-end passwords. In *Proceedings of the 2017 New Security Paradigms Workshop*. ACM, 107–121.
[29] Sunyoung Seiler-Hwang, Patricia Arias-Cabarcos, Andrés Marín, Florina Almenares, Daniel Díaz-Sánchez, and Christian Becker. 2019. "I don't see why I would ever want to use it:" Analyzing the Usability of Popular Smartphone Password Managers. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1937–1953.
[30] David Silver, Suman Jana, Dan Boneh, Eric Yawei Chen, and Collin Jackson. 2014. Password Managers: Attacks and Defenses. In *USENIX Security Symposium*. 449–464.
[31] Elizabeth Stobert, Tina Safaie, Heather Molyneaux, Mohammad Mannan, and Amr Youssef. 2020. ByPass: Reconsidering the Usability of Password Managers. In *International Conference on Security and Privacy in Communication Systems*. Springer, 446–466.
[32] Ben Stock and Martin Johns. 2014. Protecting users against XSS-based password manager abuse. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*. ACM, 183–194.
[33] Ke Coby Wang and Michael K Reiter. 2018. How to end password reuse on the web. *arXiv preprint arXiv:1805.00566* (2018).
[34] Alma Whitten and J Doug Tygar. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *USENIX Security Symposium*, Vol. 348. 169–184.

A COGNITIVE WALKTHROUGH TASKS

Below are the tasks used in our cognitive walkthroughs:

(1) **First time setup (E1):** The evaluator was required to set up the password manager. This included downloading the manager (if applicable), installing it, creating an online account, and finishing any remaining setup tasks. Evaluators avoided using external documentation, focusing instead on how the manager's UI supported this use case. This task was always the first task completed for each manager.
(2) **Registering a credential in the manager (E2):** The evaluator needed to register a credential (an account name and password) in the manager. They were free to complete this task in any way they wanted so long as they did not leave the manager's interface. This task explores design paradigm *(E2-P1) Manual registration.*
(3) **Log in with an unstored credential (E2):** The evaluator would log into a test website we created with credentials not already stored in the manager. This would trigger the manager to suggest saving the credentials. The task was complete once the credentials were saved. This task explores design paradigm *(E2-P2) Auto-detect registration.*
(4) **Updating a credential in the manager (E3):** The evaluator needed to update a credential previously stored in the manager. They were free to complete this task in any way they wanted so long as they did not leave the manager's interface. This task explores design paradigm *(E3-P1) Manual update.*
(5) **Login with updated credentials (E3):** The evaluator would log into a test website with credentials different than those stored in the manager. This would trigger the manager to suggest saving the updated credentials. The task was complete once the updated credentials were saved. This task explores design paradigm *(E3-P2) Auto-detect update.*
(6) **Removing a credential stored in the manager (E4):** The evaluator would use the manager to remove a target credential.
(7) **Log in with credentials stored in the manager (E5):** The evaluator would log into a test website that has associated credentials stored in the manager. This would trigger the autofill process (*(E5-P1) Autofill w/ interaction* or *(E5-P2) Autofill w/o interaction*). The task was complete once the evaluator finished the autofill process and was logged into the website.
(8) **Log in from multiple subdomains (E5):** The evaluator would sequentially log into two test websites we had created. These websites shared the same parent domain. The task was completed once both websites were logged in. This task compares design paradigms *(E5-P4) Separate subdomains* and *(E5-P5) Group subdomains.*
(9) **Log in from a mobile device (E6):** The evaluator would log into a test website using their mobile device. This device did not have a manager installed, and the evaluator was not allowed to install one. Instead, they would need to view the credential in the desktop manager then manually enter it into the mobile device.
(10) **Create an account (E7):** The evaluator would need to create a new online account, making sure to use a generated password. What approach they took in generating the password was left up to the evaluator, though they did examine the default settings and modified them if they wished to.
(11) **Set up and sync on a secondary device (E1, E8):** The evaluator was tasked with setting up the manager again on a second device. After setting up the manager, the evaluator would ensure that their credentials were synchronized between the two managers, and if not, they needed to figure out how to resolve the situation.
(12) **Lock manager (E9):** The evaluator would need to lock the manager.
(13) **Unlock manager (E10):** The evaluator would need to unlock the manager.
(14) **Complete a credential audit (R1):** The evaluator would need to find the credential audit tool, execute the audit, and respond to any warnings presented. The vault was filled with several problematic credentials to ensure that results would be returned from the audit.
(15) **Ensure safe settings (R2):** The evaluator would need to ensure that user interaction was required before credentials would be autofilled. This included identifying the appropriate setting and changing it if necessary.
(16) **Recover access to the manager (R3):** Evaluators would act as if they had lost their credentials and try to determine if they could regain access to their vault. If it were possible, they would take the necessary steps to do so.
(17) **Store non-credential data in manager (X4):** The evaluator would be tasked with storing a phone number and address in the manager. They would then be asked to enter this information into a website. They were free to enter the information using autofill (*(X4-P3) Autofill for structured data*) if supported.