They Would do Better if They Worked Together: The Case of Interaction Problems Between Password Managers and Websites

Nicolas Huaman$^{\mathcal{C}*}$ Sabrina Amft* Marten Oltrogge$^{\mathcal{C}}$ Yasemin Acar$^{\dagger *}$ Sascha Fahl$^{\mathcal{C}*}$

$^{\mathcal{C}}$*CISPA Helmholtz Center for Information Security*
**Leibniz University Hannover*
†*Max Planck Institute for Security and Privacy*

***Abstract*—Password managers are tools to support users with the secure generation and storage of credentials and logins used in online accounts. Previous work illustrated that building password managers means facing various security and usability challenges. For strong security and good usability, the interaction between password managers and websites needs to be smooth and effortless. However, user reviews for popular password managers suggest interaction problems for some websites. Therefore, to the best of our knowledge, this work is the first to systematically identify these interaction problems and investigate how 15 desktop password managers, including the ten most popular ones, are affected. We use a qualitative analysis approach to identify 39 interaction problems from 2,947 user reviews and 372 GitHub issues for 30 password managers. Next, we implement minimal working examples (MWEs) for all interaction problems we found and evaluate them for all password managers in 585 test cases.**

Our results illustrate that a) password managers struggle to correctly implement authentication features such as HTTP Basic Authentication and modern standards such as the autocomplete-attribute and b) websites fail to implement clean and well-structured authentication forms. We conclude that some of our findings can be addressed by either PWM providers or web-developers by adhering to already existing standards, recommendations and best practices, while other cases are currently almost impossible to implement securely and require further research.

I. Introduction

Username and password combinations remain the dominant authentication mechanism on the web. Although significant effort has been invested into developing alternative authentication solutions [36], [41], [52], [65] and helping users to use more secure passwords [34], [69], many users still rely on passwords that are easy to guess [35], [48], [71] and re-use the same password for multiple accounts [46], [58], [72], [73]. Alternative methods like FaceID or fingerprint sensors for authentication also resort to passwords chosen during their setup [38].

Online services are encouraged to deploy multi-factor authentication (MFA) such as timed one-time-passwords (TOTP) [40], [61]–[63] to address the password security problems and strengthen account security, which still rely on passwords as the first factor. To address the need for passwords, the use of password managers (PWMs) is often recommended for end-users. In a nutshell, PWMs are tools to help users deal with credentials and reduce their mental load for password creation, account login and credential updates [1], [8], [12], [14], [15], [20], [22]. PWMs have been researched extensively [47], [51], [56], [59], [60], [64]; previous research on PWMs mostly focuses on PWM security issues and usability and adoption challenges. Multiple studies researched the security of different PWM types, finding that both browser-based and locally installed PWMs are vulnerable to problems such as key theft or secret recovery from temporary files, as well as weaknesses within typical features such as autofill [64]. Other research focused on the usability of PWMs and were able to show that user adoption of PWMs is motivated by convenience of usage and usability [59]. While security benefits can also be a driving factor for PWM adoption, in the majority of cases these where only mentioned for accounts that users perceive as especially important.

Overall, previous work identified significant security and usability challenges PWM providers should address to improve overall password security [39]. However, in addition to the previously identified challenges, PWM usability and adoption also depend on how well PWMs and websites can interact with each other [2], [17], [42]. Websites that do not accept auto-generated passwords, prevent autofill and autologin of stored credentials or make credential storage for accounts complicated contribute to bad PWM usability and thwart their adoption. Examples of poor PWM support in websites are sites that manipulate forms using JavaScript or fail to define input field attributes, making these interactions harder. While these issues are discussed in online forums [2], [17] and blog posts [42] and providers of web browsers suggest novel mechanisms to better support PWMs (e. g. Apple's passwordrules [57]), a systematic analysis of poor interactions between websites and PWMs that make their use unnecessarily complicated or even impossible is missing so far. To the best of our knowledge, our work is the first to investigate those poor interactions reported by real PWM users, to analyze how 15 popular desktop PWMs deal with these circumstances and to propose ideas based on our results for future PWMs and websites for better PWM usability and password security on the web.

In the course of the following work, we address the following research questions:

RQ1: Which interaction patterns on the web are problematic for password managers?

RQ2: How do PWM browser extensions handle these interactions?

RQ3: What can be done to improve the interaction between PWMs and websites?

To the best of our knowledge, this work is the first to investigate interaction challenges between browser-based PWMs and websites that impact both password security and usability and end-user PWM adoption. Based on the above research questions, we make the following contributions.

Systematic Problem Survey — We perform a systematic analysis of 3,319 reviews and issues users of 30 browser-based PWMs report in the Chrome Web Store, and on GitHub. Based on this analysis, we identify 39 interactions that demonstrably hindered end-users using PWMs in real-world settings.

Minimal Working Examples (MWEs)[1] **—** We build a simple website including minimal implementations of the previously identified interactions PWMs struggle with, based on the results of our systematic problem survey. With this website, we can extensively test PWMs regarding their performance and supported features.

PWM Problem Evaluation — We test 15 popular browser-based PWMs on the 39 MWEs we implemented. Our analysis shows that in many cases, PWMs have issues working with websites that include complex or non-standard implementations, but also with standards like "basic" HTTP Authentication.

Recommendations — We investigate existing web standards [3], [7], [10], [18], [44] in regards to the problems we found to be most prevalent in our PWM evaluation, and propose how existing standards and approaches can already be used to solve most of the problems we discover. Furthermore, we propose potential vectors for extensions of standards that solve the remaining issues.

Detailed Replication Information — We provide an extensive replication package including the set of collected user-complaints, the resulting code book, and the 39 MWEs derived from the code book (cf. Section V-D. In addition, we provide the implementations of the MWEs and demo screen captures to replicate our approach (cf. Section V-D) along with our results.

This work focuses on desktop PWMs and their browser extensions. While we considered adding mobile PWMs, these have very different requirements, as they can e. g. use APIs provided by Android [4] and iOS [3] and are focused on user installed apps.

The rest of the paper is organized as follows: We discuss related work and the relevant background in Section II and Section III. In Section IV we describe our systematic problem survey based on real-world end-user feedback. Section V details the analysis of 15 popular PWMs using our set of 39 MWEs. Finally, we discuss our findings and propose strategies to mitigate the problems PWMs struggle with in Section VI.

II. Related Work

While there have been many different attempts to replace passwords as the main authentication method on the web, none of the proposed alternatives were able to supersede them. Moreover, most attempts that improve authentication security come with high costs in terms of decreased usability [37]. PWMs yield a solution to this problem by removing the need to memorize passwords and storing them in a secure manner, which enables users to choose stronger passwords [51]. Previous research already paid attention to several aspects of PWMs. In the following section, we present an overview of these works, focusing on PWM security and vulnerabilities as well as usability and adoption.

PWM Security. Overall, different studies were able to show that many popular PWM solutions are vulnerable to different forms of attacks. In 2013, Zhao et al. [74] analyzed the PWMs included in five different browsers and find severe vulnerabilities as attackers can e. g. steal the files in which keys are stored to decrypt them. As an alternative, they propose a cloud-based approach. Similar to this, Li et al. [50] conducted an analysis of five browser-based PWMs and identify 4 key security concerns within bookmarklets, the UI, and classic web or authorization vulnerabilities. While these studies were focused on browser-based PWMs, in 2016 Gray et al. [47] evaluated the security of KeePass, RoboForm and Password Safe, three locally installed PWM applications. They reveal security problems with e. g. secret information such as the master password or database content recoverable from temporary files, in some cases even after the applications were terminated.

Although most of the research concerning PWM security was conducted more than five years ago, recent work shows that the problems are still persisting. In 2020, Oesch et al. [56] were able to replicate previous studies and evaluate the security of 13 PWMs. While they found improvements for several features such as password storage or autofill security, they were also able to reproduce severe vulnerabilities especially within the password generation.

In 2014, Silver et al. [64] were able to show that especially the autofill functionalities of PWMs are often vulnerable. They review different types of PWMs and their various policies to recognize relevant form fields, finding them susceptible to attacks via e. g. modified web forms or unencrypted connections.

Furthermore, Zhao et al. [68] evaluated the top 4,000 Alexa pages and found 86.3% of websites offering login fields vulnerable to XSS-based attacks. As a solution, they proposed an alternative approach in which only dummy input is entered into the web form and the real password directly transmitted in the respective HTTP request.

While previous work focuses on the security within PWM applications, our research concentrates on the usability of PWMs and their ability to adapt to frequent problems or edge cases.

PWM Usability and Adoption. Several previous works focus on the usability and adoption motivations for PWM usage. In 2006, Chiasson et al. evaluated the usability of two PWMs that previous research proposed. Both showed significant problems that did not only limit the usability, but in some cases lead

[1] In software development, MWEs are used to illustrate problems or bugs within code while only consisting of the most relevant features [27].

to security issues that lead to e. g. user misconceptions on whether or not their passwords were protected [39].

Another usability study was conducted in 2010 by Karole et al., who examined three different types of PWMs. They chose a locally installed LastPass, a mobile KeePass version as well as an USB-based RoboForm approach, finding that users preferred the mobile variants despite the reduced usability when compared to the local LastPass [49].

A series of 30 semi-structured interviews conducted by Pearman et al. found differences in the user motivation to adopt a PWM depending on its type. Their work suggests that users of browser-based PWMs rated the convenience higher while the users of separate stand-alone clients valued security benefits more [59]. In 2020, Ray et. al. replicated this interview study with a focus on older adults (>60 years old), finding that older adults had higher mistrust of cloud storage and online services in general, and noting that recommendations from trusted people like family and friends provide motivators for those older people [60].

Other recent work by Maclean et al. suggests that three key factors influence an user's decision to adopt a PWM: The expectation that the tool performs well and improves their workflow, the generation of a habit due to continued use and trust in the security mechanisms within the PWMs [53].

While most works focus on novice users, Stobert et al. [67] interviewed security experts about their own habits and usage patterns and found similar motives on websites the participants deemed less relevant. Secure behavior was often only shown for accounts they perceived as important, suggesting that even with expert users, PWM usage is not widely adopted.

Most studies that researched the usability of PWMs found that convenience and usability were the primary deciding factors for adoption. These can be decreased by PWMs that do not behave in the intended way, e. g. due to problems recognizing relevant information such as password input fields. To mitigate this, Stajano et al. proposed a html-based specification of semantic labels for input fields to help PWMs find relevant fields and correctly classify their main purpose [66].

To the best of our knowledge, our work provides the first evaluation of PWMs based on real world problem reports. Our research examines how well PWMs can deal with existing edge cases. We evaluate the behavior of 15 major PWMs on a set of MWEs for the previously analyzed problem reports. In contrast to Stajano et al. who focused on the recognition of input fields, our work aims to identify broader shortcomings of PWMs by identifying widespread issues and recommending possible solutions.

III. Background on Password Managers

At their core, PWMs are tools to handle user credentials (e. g. usernames and passwords) and to reduce their users' mental load during password generation, account registration and login, as well as credential updates. PWMs aim to improve online account security by generating, storing and autofilling distinct secure passwords for online accounts and therefore compensating for end-users' difficulties with creating and remembering a myriad of secure and distinct passwords.

However, since handling user authentication is a security critical problem, PWMs have been a focus for security research, and prior work has discovered a multitude of issues across all common PWMs [47], [50], [56], [64], [74]. Furthermore, the multitude of approaches to implement password based authentication on the web led to a number of usability problems that limit the adoption of PWMs. Here, related work has presented us with the requirements and expectations that users have for PWMs [39], [49], [53] and discovered what features of PWMs motivate users to use them [59], [60], [67]. The main reasons to adopt PWMs were discovered to be the convenience of not having to remember passwords and the time saved when a PWM supports the user by autofilling the passwords. To provide these features however, a PWM has to analyze a website, identify it and interact with the authentication mechanisms of the website. As we will demonstrate in our user complaints evaluation (cf. section IV), PWMs currently have to handle a large range of possible interaction patterns to authenticate on websites.

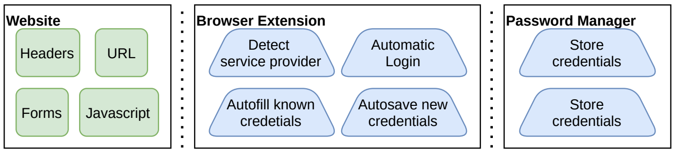


Fig. 1. Interaction between PWMs and websites through the browser extension.

A. Interaction with websites

Most desktop PWMs interact with websites through a browser extension. They usually cooperate with a standalone PWM application, or connect to a web service. The application or web service handles the storage of user credentials; the browser extension interact with the website. In this work we focus on the interaction between PWMs and websites to identify obstacles that can decrease usability.

Web Authentication. PWMs commonly provide the following features in order to support users during their authentication workflow:

- **Service Detection**
- **Credential Storage** and *autosave*
- **Providing Credentials** and *autofill*
- **Automatic Login** or *autologin*
- **Secure Credential Generation**

Additionally, as we discovered during our analysis, one obvious requirement for PWMs is to not interfere with a website in a way that prevents users from accessing its contents. In the following, we provide background information and examples for each of these features, and how we handle them in our work. We provide an overview in Figure 1.

Service Detection. This feature refers to the way password managers determine which authentication data to use. The most important factor is the URL of a website, which is usually matched on domain-name level (e. g. example.com). PWMs may or may not match different subdomains to the same set of credentials (e. g. sub1.example.com, sub2.example.com) or offer to store multiple accounts for different paths (e. g. example.com/service1, example.com/service2).

Credential Storage. Storage of credentials represents the core feature of password managers. From an usability perspective, password managers can provide automatic creation of new credentials when they detect them being entered on a website, which we refer to as *autosave*. It can also offer to update credentials when it notices different credentials being entered for already stored accounts.

Providing Credentials. From a convenience point of view, this is the second main feature password managers can provide to support users. When a user accesses a service that the password manager has stored credentials for, the password manager can offer these credentials automatically. To provide further convenience, the password manager may even attempt to fill the credentials in the appropriate form-fields on a website or provide them for authentication headers [44], commonly refereed to as *autologin*.

Automatic Login. One rarer functionality across password managers is to automatically log in (*autologin*) users without being triggered at all when the user visits a website that the password manager has stored credentials for. For the purpose of our analysis, this feature includes everything required for *autologin* and to submit the login request, without any previous interaction by the user except to visit the site.

Secure Credential Generation. The final feature we want to talk about for this analysis is the option to create credentials for the user. Password managers can generate secure passwords and offer them to the user during registration or password change processes. From a user experience perspective, the main features here relate to password policies, detecting and enforcing them for the generated passwords. -

IV. User Complaints Evaluation

To identify interaction problems between PWMs and websites users encounter, we collected and analyzed user feedback for all PWMs listed in Table I. We used the identified list of interaction problems to build MWEs for our second study in the following section.

A. Choosing Password Managers (PWMs)

Since Chrome has the highest market share, we focus our user complaints evaluation on PWMs that offer browser extensions in the Chrome Webstore [21], [28], [32]. We consider all PWMs that were downloaded at least 10,000 times which should cover the majority of users. Table I provides an overview for all PWMs we considered.

TABLE I
PASSWORD MANAGERS FOR WHICH WE COLLECTED FEEDBACK INCLUDING THEIR DOWNLOAD COUNTS.

	Name	Downloads
1	LastPass: Free Password Manager	10,000,000
2	Norton Password Manager	4,000,000
3	Avira Password Manager	3,000,000
4	Dashlane - Password Manager	3,000,000
5	1Password X – Password Manager	600,000
6	Bitwarden - Free Password Manager	500,000
7	RoboForm Password Manager	500,000
8	Keeper® Password Manager & Digital Vault	300,000
9	ThinkVantage Password Manager	200,000
10	Blur	100,000
11	RapidIdentity	100,000
12	Enpass extension (requires desktop app)	100,000
13	SafeInCloud Password Manager	80,000
14	KeePassXC-Browser	60,000
15	Password Depot Extension	50,000
16	Passbolt Extension	50,000
17	NordPass® Password Manager & Digital Vault	30,000
18	Zoho Vault	30,000
19	Password Manager Pro	30,000
20	MYKI Password Manager & Authenticator	20,000
21	Passwordstate	20,000
22	Kee - Password Manager	20,000
23	F-Secure KEY Password Manager	10,000
24	KeePassHelper Password Manager	10,000
25	Devolutions Web Login	10,000
26	SaferPass: Password Manager for Free	10,000
27	Steganos Password Manager	10,000
28	Trezor Password Manager	10,000
29	Advanced Password Manager	10,000
30	1Password X Beta – Password Manager	10,000

After selecting the PWMs, we aimed to find interaction problems between PWM browser extensions and websites (**RQ2**, e. g. non-working autofill of login credentials). We only consider browser extensions, since we expect users to mainly use PWMs within popular web browsers. To identify high impact problems and a measure of frequency or prevalence, we decided to focus on user feedback and reviews for all PWMs in Table I. We collected 3,319 instances of user feedback for these PWMs. Using open coding (cf. Figure 2 for an overview of our coding process), we consolidated feedback to construct MWEs to investigate these problems across all PWMs and develop potential improvements. Below, we present and discuss our user feedback evaluation and the resulting interaction problems we identified.

B. Collecting User Feedback

For diversity and to reach saturation, we collected user feedback from multiple sources. Since we selected PWMs based on the availability of a Chrome Web Store extension, we started collecting feedback from user reviews. Users can post public reviews for each extension. Additionally, users can post public support requests that extension providers can answer or redirect to official support channels. Hence, Chrome Web Store reviews are the main feedback source in this study.

Feedback taken from the Chrome Web Store in most cases stems from end users, therefore containing non-expert opinions and complaints.

To obtain more diverse feedback, we also collected expert feedback containing more technical details and different use-cases. While ticket and support systems for PWMs are usually not publicly available, open source PWMs make detailed histories of issues and solutions accessible in their repositories. These issues are typically not submitted by average end-users but more tech-savvy users and web developers. All open source PWMs in our data set host their code on GitHub [16]. Hence, we included their GitHub issue history in our analysis. We also considered including issues from larger browser vendors, but for this analysis decided to focus on issues within the realm of WebExtensions. In the remainder of this section we provide details on our user feedback collection and evluation.

Chrome Web Store User Reviews. Concerning feedback from the Chrome Web Store, we collected and analyzed both the user reviews and support requests as of 07/16/2020. We collected up to 125 of the most recent reviews and support requests for each PWM browser extension in Table V-C. Only few PWMs had more reviews and we did not find new interaction problems in additional reviews. Using a Python-based Selenium web crawler [25], [26] we collected feedback items and related meta information including the author, date of posting and comments. To guarantee a correct problem assessment, we dropped all reviews that were not in English or German. Overall, we collected 2,947 user reviews and support requests for 30 PWMs. In 15 PWMs did not offer a support section or only linked to their official support system on the Chrome Web Store page. For those, we limited our analysis to user reviews. In total, we acquired 1,895 reviews in 30 PWMs' review sections, but only 1,052 support requests in 15 PWMs' support sections. Since reviews and support requests are structured differently, we collected slightly different data. For both, we collected the text itself as well as any comments left by e. g. other users or support teams, the authors, the date of the postings and which PWM the post refers to. Both reviews and support requests included data unique to their respective type: For reviews, this was a rating between one and five stars awarded by the commenting users, for support requests, users were able to add a title to their post, which we both collected and considered as well.

GitHub Issues. We analyzed both open and closed issues on GitHub that were created between 04/01/2019 and 07/17/2020 as well as updates to them until 08/12/2020. Similar to the Chrome Web Store reviews and support requests, we limited our collection to issues until 01/01/2020, as we did not find new interaction problems in older issues. The issues were collected using a crawler extension for Chrome [29], [30]. We collected issues regarding PWM browser extensions for three open-source PWMs (Bitwarden [12], KeePassXC [13], [14][2], and Passbolt [15]) from five GitHub repositories. We extracted the title, link, whether the issue was open or closed, and date for each issue. Content wise, we looked at the complete thread including issue discussion and resolution.

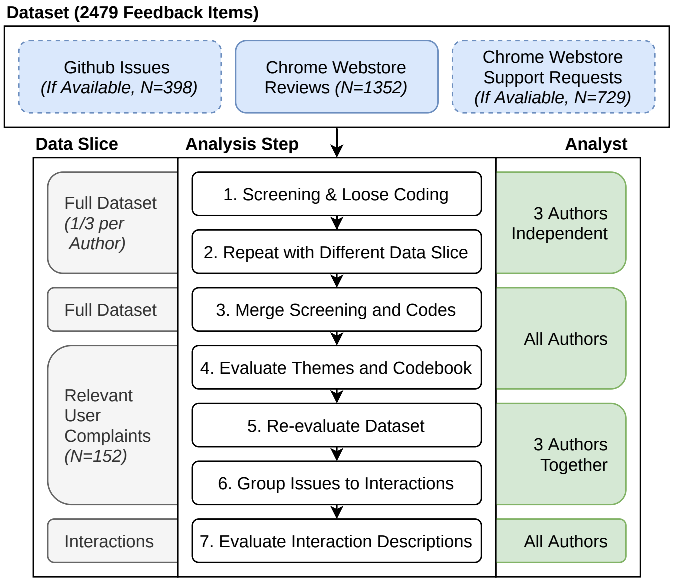

Dataset (2479 Feedback Items)
Github Issues (If Available, N=398)
Chrome Webstore Reviews (N=1352)
Chrome Webstore Support Requests (If Avaliable, N=729)
Data Slice
Analysis Step
Analyst
Full Dataset (1/3 per Author)
Full Dataset
Relevant User Complaints (N=152)
Interactions
1. Screening & Loose Coding
2. Repeat with Different Data Slice
3. Merge Screening and Codes
4. Evaluate Themes and Codebook
5. Re-evaluate Dataset
6. Group Issues to Interactions
7. Evaluate Interaction Descriptions
3 Authors Independent
All Authors
3 Authors Together
All Authors

Fig. 2. Methodology for our qualitative coding research user reports.

Overall, we collected users feedback from 372 issues, 1,895 reviews and 1,052 support requests resulting in a total of 3,319 feedback items we considered and coded in our further analysis.

C. Classification and Problem Case Development

We used an iterative exploratory coding approach [55] for the user feedback analysis. Since we were interested in analysing user complaints regarding issues on the PWMs' interaction with websites we had to filter out unrelated feedback (e. g. "This Password Manager is very good" or "Doesn't work, this sucks") first. Therefore, three authors each reviewed a third of the full set, marking user complaints as *relevant* if they focused on the PWM's interaction with a website (cf. Figure 1). This first pass was also used to assign initial codes to the complaints (Figure 2, step 1). We aimed to identify both the broken functionality as well as the underlying technical cause. To obtain a more diverse view, we then repeated the process with a different third of the data set (step 2). Therefore, two researchers reviewed each feedback item for relevancy and the initial codes. After coding all items we regrouped and merged our findings (step 3), resulting in 150 relevant user complaints. Since all researchers discussed and agreed on the final codes, we did not calculate the inter-rater reliability [54]. Using the merged initial codes, we iteratively developed themes covering issues and areas suitable for the later problem cases. This resulted in a final code book consisting of 59 codes divided into a end-user oriented and a more technical perspective (step 4). We first evaluated the final code book by coding 20% of the items together, before splitting the remaining 80% between three researchers to reassess the entire data set with (step 5). Using the coded data set, we were able to identify complaints that described the same problem even across multiple PWMs. Based on this, we developed the final

[2] KeePassXC contained browser extensions related issues in two repositories for either the extension or client. We filtered for issues labelled as related to the extension.

interactions, which describe technical problems of one or more PWMs with common practices in web development (step 6). In this step, we also decided to filter out 40 reviews we previously considered relevant because they either contained not enough details to be reproducible. While these fit our criteria for relevancy and coding, they where unfit for the MWEs we attempted to build. Finally, three researchers discussed the resulting valid interactions and categories, reviewing and refining the description for each interaction (step 7). In this step, we used the technical codes of the code book to categorize the interactions and the end user oriented side to determine fulfillment criteria, e. g. the requirements a PWM has to pass in order to obtain a positive rating for the respective interaction. We also reviewed the feedback items our further analysis is based on to refine these fulfillment criteria. See Table V-C for the final list of interactions and their descriptions.

D. Limitations

Due to the qualitative nature of our user feedback analysis, this work is not without limitations. First, we only review PWMs that are available as web browser extensions. However, since Chrome has by far the largest market share over all browsers, and most extensions can also be used on e. g. Firefox, we are confident that our selection covers most if not all relevant PWMs. We can only provide limited insight into closed source PWMs and possible causes for their behavior as these are not available publicly. In these cases, we could only review Chrome Web Store reviews and support requests.

While we are confident that we reached saturation during our analysis, we only reviewed a limited number of user feedback sources. Depending on the PWMs popularity and whether or not they provide third party feedback collection e. g. through a ticket system, the number and age of reviews feedback items varies per PWMs. Popular extensions sometimes provided thousands of reviews for which we only sighted the first 100 with all comments, while less widespread ones only had few comments that could span until November 2013. Furthermore, reviews are related to multiple extension versions. We did not review any new versions or feedback items after July 2020, since after that time frame we started coding. Additionally, reviews are subject to a self-reporting bias and the sample of users who report issues is much smaller than the number of downloads and therefore possibly not representative to the user base at large. As described in Section IV-E, we found several issues that were only rarely reported. However, since we aimed for issue diversity, we argue that for the purpose of this work, knowing that an issue exists is sufficient. If a problem description only occurs seldom or if it is related to an outdated extension version, it might still be relevant for other PWMs. Overall, we based all examples of interaction problems between PWMs and websites on our data for issues, reviews and support requests and tested all examples with all 30 PWMs in our list. Finally, our approach is not suitable to investigate the prevalence of these issues, as they are dependent on e. g. users that report them, the popularity of the PWM and potentially private issue trackers we could not access.

E. Interaction Problems

In the following, we present and discuss our findings in alphabetical order, highlighting their relevance and describing the set of interactions they are composed of.

Additional Elements (Auth). When a website contains more input fields than necessary for the current authentication task, even if they were declared correctly and explicitly, PWMs can struggle locating relevant input fields. Examples for this include websites with username and password fields for both account creation and login, or cases in which multiple single-digit fields instead of only one input field are used for e. g. TOTP codes, in which users complained about not being able to auto-fill this additional information:

> "Great application and extension. Pity it can't detect more than 2 fields on a screen. In case (such as banks etc.) you need to enter 3 different strings for identification, the app / extension will grab and use only 2, while you'll have to add the 3rd one manually. [...]" (AA-04, User 1)

Overall, this category requires the additional input fields to be related to the authentication process in some way. In these four cases, the desired authentication process fails.

Additional Elements (Non-Auth). While similar to the previous category, this focuses on input fields that are not related to authentication purposes. This includes unrelated text fields on e. g. CMS that are incorrectly filled with credentials as well as other types of elements such as drop-downs and check boxes. Here, undesired behavior might expose passwords in plain text or even break websites. We identified four cases where additional elements were problematic.

Domain Matching. Before a PWM can perform any detection or interaction on the website, it has to compare the given URL to all stored credentials and their associated service URLs. This is a necessary step to recognize and autofill known credentials as otherwise, authentication-related information might be leaked to unrelated or untrustworthy services. Overall, we identified seven sources of service mismatching. For example, we identified cases where services were not recognized due to e. g. no or wrong differentiation between subdomains or paths on a website, or because the website incorporates redirects or iframes with different origins, all of which complicates the service matching process.

Input Fields. After identifying the website, a PWM needs to locate all input fields that are relevant for the current authentication task. This usually comprises fields for an identifier such as a username and a password, but can also require further personal information or TOTP codes. To find all relevant fields, PWMs usually rely on their declaration, using attributes such as name, type or id to decide whether or not an input field should be filled. In this category, we collect nine interactions where the detection is difficult, e. g. because not all or no relevant attributes are present, or because the attributes used ambiguous or misleading values such as "IDToken1" for username inputs. This was a problem we found frequently within user reviews:

> "[...] Some websites don't define a type attribute for their inputs, while LastPass seems to blindly trust any input element to have a defined type. I think maybe the best solution is to assume that if a type is not set, that the type is then "text"" (I-07, User 1)

Non-Standard Forms. This category includes five in which the form element containing the relevant input field diverged from standards by e. g. omitting a form tag or locating submit buttons in website elements different from the associated form.

JavaScript. We found six problem cases related to scripts on the website that in some way manipulate form elements or user input, as was e. g. described in the following issue:

> "[...] I guess this is because citi.com actually displays 5 different password fields before the password field used by the user. While they're not specificly hidden and just moved out of the viewport, I imagine it to be kind of tricky to add detection for something like this." (J-05, User 3)

This category also includes cases in which relevant input fields are shown one after another, are hidden until the user interacts with other website elements or when pseudo security measures are in place to disrupt automated inputs by e. g. enforcing a keypress event before enabling submit buttons. In all of these cases, JavaScript impedes the work of PWMs, therefore reducing their usability.

Timing. We collected two interactions in which some kind of delay disrupts the PWM workflow. Such problems can occur when scripts on the website cause relevant elements to appear after the initial scan for input fields, or when a PWM generates TOTP codes only once - here, the code can be outdated if there is a delay before submitting the credentials, causing the login to fail.

Web Standards. Finally, we identified two in which PWMs struggle to correctly work with HTTP Basic Authentification [44]. This falls outside of the realm of our previous categories since it does not require extensions to interact with the website, but simply implements a callback to the `onAuthRequired` event [6], [31]. Issues with HTTP Basic Authentication were noteworthy as they were reported multiple times. For example, ten different users showed frustration and complained about the lacking support for basic authentication.

Further Findings. The reported issues in our coding implicate vastly different approaches to automatic storage, filling or full on logging in on websites. These approaches included automatically filling the page when accessing a website, but also the possibility to choose an account when clicking on the input field or having to interact with buttons provided on the browser user interface, not including any convenience. We need to consider these different approaches when testing the PWMs in our next study in order to discover how each of them performs on the different categories of MWEs we test. This way we hope to discover recommendable approaches that solve some of the issues fully automatic PWMs have compared to the ones that require more user interaction. Finally, while we think the interactions we discovered reach saturation concerning the interactions users report, some of them were found for only one or two PWMs, which inhibits reliably reports on their prevalence and how other PWMs might solve them. In a follow-up study described in section V, we test common PWM approaches for these interactions to both obtain a picture of problem frequency and to investigate how different PWMs try to solve them. We further discuss how problematic interactions could be mitigated in the future.

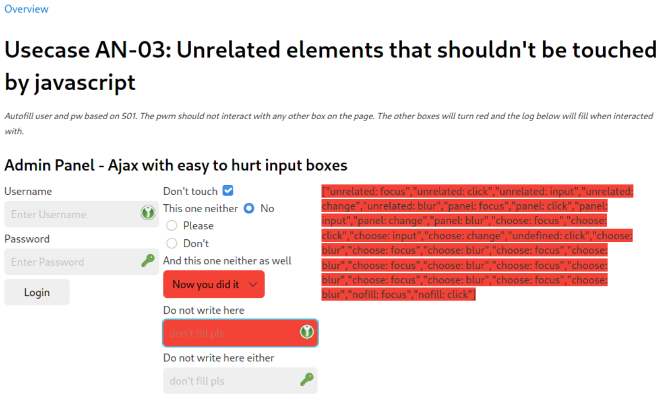
V. Interaction Problem Evaluation

In the second study, we investigate the interactions between 15 PWM browser extensions with previously identified 39 interactions. Since the user feedback we analyzed in Section IV was specific for single PWMs, we aim to investigate how other PWMs deal with the problems we found and provide recommendations to help future PWM and website development. We will detail our methodology and considerations below and then present and discuss the results.

A. Methodology

To test the 15 PWMs, we implemented all problem cases we found in the form of 39 minimal working samples (MWEs). With this, the problems can be presented easily and it is further possible to remove e. g. dependencies as external causes. We therefore chose this form to build working examples from our list of interactions, using a flask web application that simulated logins in different scenarios [11]. In total, we conducted 585 distinct tests. The application spans three domains and two subdomains (D01-D04), using the third domain to simulate the HTTP protocol interaction (D-04). This was necessary as our other domains use HSTS, which prevents an accidental protocol changes on other interactions (D01-D03). We implemented the MWEs as basic websites, consisting of minimal styling for readability, the JavaScript and HTML elements required to reconstruct the interaction and in most cases some basic form of validation checking that allows us to verify correct username and password input by the PWM. They all work on a template, for which we included an example in Figure 3 and each page includes a description of the interaction and instructions on the requirements to pass the interaction. We provide the web applications as part of our replication package in subsection V-D.

Evaluating PWM Interactions. For this analysis, we limited ourselves to the top ten PWMs, mainly because these cover 97.9% of users according to download counts. We added KeePassXC and Passbolt since their open source nature allows for investigation of interactions, bringing our total coverage up to 98.4%. We further added the default browser PWMs for Chrome, Firefox and Edge, since due to their immense user base, their PWMs likely have a larger count of users than any of the top ten PWMs mentioned previously. Other



Overview

Usecase AN-03: Unrelated elements that shouldn't be touched by javascript

Autofill user and pw based on S01. The pwm should not interact with any other box on the page. The other boxes will turn red and the log below will fill when interacted with.

Admin Panel - Ajax with easy to hurt input boxes

Username

Enter Username

Password

Enter Password

Login

Don't touch ☑

This one neither ● No

○ Please

○ Don't

And this one neither as well

Now you did it

Do not write here

don't fill pls

Do not write here either

don't fill pls

```
["unrelated: focus","unrelated: click","unrelated: input","unrelated:
change","unrelated: blur","panel: focus","panel: click","panel:
input","panel: change","panel: blur","choose: focus","choose:
click","choose: input","choose: change","undefined: click","choose:
blur","choose: focus","choose: blur","choose: focus","choose:
blur","choose: focus","choose: blur","choose: focus","choose:
blur","choose: focus","choose: blur","choose: focus","choose:
blur","nofill: focus","nofill: click"]
```

Fig. 3. Our minimal working examples (MWEs). Consist of the interaction that we modeled, a short description and instructions on what is considered a successful interaction. Layout aspect ratio adapted for paper.

browsers, such as Safari, were not reviewed due to a smaller marketshare at the time our tests took place. Overall, we tested 15 PWMs. For all of them we used the premium trial version where available.

We test all PWMs against all 39 problems and distinguish the following test outcomes:

- Seamless: The PWM behaves as expected without any manual interaction (e. g. it intuitively autosaves and autocompletes). Expected behavior is defined per minimal working example on the page.
- Manual: The PWM does not behave as expected, but manual intervention or workarounds lead to the expected result. This includes e. g. using context menus to fill credentials or settings that make the case possible.
- No Solution Found: The PWM is not able to solve the task without additional interaction. Manual interaction with the website or searches on e. g. the respective support pages lead to no solution, which suggests that the case is not supported or malfunctioning.
- Not Applicable: The PWM does not support a required feature such as autosave or multi-factor authentication. It cannot be evaluated for these cases.

Each MWE includes a description how to proceed with the example and which conditions define success or failure. The researchers who tested the PWMs were instructed to follow these descriptions using the interaction model of the respective PWM. If a feature such as autosave was not available or working, we tried to find a solution by using a simple Google query including the name of the PWM, the term “password manager” and the feature in question to find existing settings that resolve the issue. After a first pilot run in which we tested each PWM with all MWEs, we fixed remaining bugs and obtained an overview over the different approaches PWMs deploy in order to finalize our rating. We included these considerations for the ratings in order to prevent unfair treatment of more manual approaches like having to click an input field to trigger the PWM. We also used this pilot to clear up any confusion about interaction descriptions and conditions between the authors.

TABLE II
PASSWORD MANAGER APPROACHES TO IMPLEMENT AUTOSAVE, AUTOFILL AND AUTOLOGIN.

Password Manager	Autosave	Autofill	Autologin
Lastpass	Ask	On Load	Setting
Norton	Setting	Manual	None
Avira	Ask	On Load	None
Dashlane	Ask	On Load	Seamless
1Password X	Manual	Manual	None
Bitwarden	Ask	Manual	None
RoboForm	Ask	Manual	Seamless
Keeper	Ask	Ask	Seamless
Blur	Setting	On Load	None
Enpass	Ask	Manual	Seamless
KeePassXC	Ask	Manual	Setting
Passbolt	None	Manual	None
Chrome	Ask	On Load	None
Firefox	Ask	On Load	None
Edge	Ask	On Load	None

Password Manager Approaches. PWMs differ vastly in the approaches they use to implement autosave, autofill and autologin features. While the terms suggest automatic behavior, they often require manual intervention. An overview of the strategies encountered during the PWM tests is given in Table II. The approaches described for each PWM were used as a baseline to assign codes to each MWE. If a PWM did not support feature e. g. TOTP, we used the code “Not Applicable” in contrast to “No Solution Found” for PWMs that supported the feature, although it was not available or working in both cases. We did however count it as “No Solution Found” when PWMs did not support common approaches like HTTP Basic Authentication (W01 & W02) or simple input field uses like pin entry (AA-01).

As can be seen in Table II, PWMs usually autosave credentials by asking after a login was submitted. In a few cases such as Norton, 1Password or Blur, this feature can be triggered manually or enabled in the settings. Only Passbolt does not offer an autosave feature. For autofill, about half of our tested PWMs filled the stored credentials automatically as soon as the login page is loaded, while the others require manual intervention such as using a context menu or clicking an icon within the input fields. Only Keeper uses an approach in between by automatically asking if credentials should be autofilled, but still requiring the user to manually agree. Finally, we found that as few as four out of our 15 PWMs support an autologin feature. Although two others, LastPass and KeePassXC, allow to control autologin in their settings, the remaining nine PWMs do not offer the feature at all.

Study Protocol. Following the study protocol, two authors reviewed all PWMs they did not review during the piloting. For each test, we re-installed the extensions and started the PWMs with fresh user profiles, to ensure a clean testing environment. We went through the entire set of MWEs, followed the instructions given on each test case page and based our rating on the conditions we set for a seamless interaction. After rating all managers this way, the two involved authors merged the results together with the third author who conducted the piloting for that PWM. We used this very elaborate approach in order to capture as many workarounds and additional settings as possible, detect inconsistent behavior and prevent unfair ratings. Since some of the problematic interactions we examine can be considered edge cases, we refrain from ranking the tested PWMs or apply similarly subjective measures and comparisons. This is especially important as we do not consider the actual real world frequency of the issues found in our set of interactions.

B. Limitations

Our analysis was conducted between 11/13/2020 and 11/18/2020 with at that time most recent versions for each PWM. As PWMs might have been updated after our analysis, more recent versions could behave differently due to added or modified support for features or known issues. Similar, user feedback for more recent versions of a PWM might raise new issues we did not include in our analysis. We used four different labels to distinguish how well a PWM was able to deal with the different interactions. Although all authors discussed the fulfillment criteria for each interaction together, the individual ratings might be subjective and can depend on the respective settings and overall setup. However, since we tested all PWMs with three authors, we are confident to have correctly estimated the capabilities of each. We further argue that if a setting was available, but hard to find and not regarded in our evaluation, this decrease in usability can also lead to a rating such as 'No solution found'. Additionally, some settings might depend on settings only available in e. g. beta builds of a PWM which were not used in our analysis. Furthermore, PWMs might include manual or hardcoded workarounds for certain websites, which would not necessarily trigger in our MWEs and lead us to mark the respective interaction as failed. While we base all interactions on real user feedback and our implementations follow examples given in reviews, support requests and GitHub issues, we cannot guarantee that all examples are reflective of common real world issues. However, all problems depicted in our MWEs are valid problems that represent obstacles PWMs face. In many cases, we prioritized having more different interactions to increase problem diversity over testing multiple variants of the same issue. Since PWMs use a variety of approaches to implement different features such as autofill or autosave, this might have led to our evaluation favoring certain PWMs.

C. Results

The results of our evaluation are summarized in table V-C. In the following, we will describe interesting findings and interactions per category.

Additional Elements (Authentication Related). In this category, our MWEs were largely unsupported. AA-01 (pin field required for authentication) proved to be especially difficult, with only two PWMs receiving conditional ratings at best. This is likely due the five input fields included, that are combined to an array and therefore use the same `name`-attribute. The example further contains no field with the type `password`, which might disturb PWM field detection. Only AA-03 (login page contains both login and registration forms), received seven (46%) seamless PWM ratings. Three other cases, including tests for additional required authentication fields (e. g. for pin codes or last names), received up to three seamless, requiring workarounds or outright failing with most PWMs. This is due to poor or missing support for additional or custom authentication fields. Most PWMs did not offer to store information other than a username and a password.

Additional Elements (Not Authentication Related). This category contains cases with website elements that are not directly part of the authentication process such as radio buttons or checkboxes (e. g. *Remember Login* buttons), but also unrelated form elements (e. g. setting panels). In interaction AN-01 (admin panel with multiple user authentication fields), seven (46%) PWMs got a seamless rating, successfully filling admin credentials while leaving out credential fields for other users. Another problematic interaction was AN-02, which contained a *Remember Me* checkbox and a panel consisting of multiple radio buttons representing choices that a PWM should remember. Only one PWM provided a seamless experience by directly storing the choices with the credentials, therefore demonstrating an exemplary solution to this autosave problem, while two being able to store custom fields on manual interaction. Interaction AN-03 (multiple input elements outside of the login form) includes scripts that report when a PWM interacts with more than the necessary input fields by checking if events on the other elements are triggered. Here, ten PWMs (66%) correctly avoided field interactions. This case only concerned PWMs that scan and potentially test (e. g. click or send a keyboard event) all page input fields. Surprisingly, all default browser PWMs failed this test, interacting with each input field while submitting the form. Finally, we reconstructed upload fields that some users reported to be obstructed by the PWM extension in AN-04. However, we found them to work with all PWMs, representing one of two interactions that received only seamless rating.

Domain Matching. This category covered some of the most commonly reported issues such as account usage across multiple subdomains (D-01) or multiple redirects after login, obstructing autosave functionalities (D-05). Both performed noticeably better than others, with nine (60%, D-01) and ten (66%, D-05) seamless interactions. This implies that these are common issues which are likely solved by defaults similar to the same-origin policy [19], [23], [24]. Another interesting interaction is D-02 (account management across multiple different second-level or top-level domains), inspired by an issue with e. g. ShareLaTeX's move to Overleaf [9], or domain moves and multi-service domains like discord.gg and discord.com. In all of these cases, multiple different domains point towards the same service. D-02 was not seamlessly handled by any PWM, likely because there is no reliable way to automatically detect if a service or an authentication

TABLE III

RESULTS FOR OUR EVALUATION TESTING PASSWORD MANAGERS AGAINST THE PREVIOUSLY GENERATED MINIMAL WORKING EXAMPLES.

Interaction Legend:

⬜ Seamless , 🟨 Manual , 🟥 No Solution Found , ⬛ Not Applicable

Color scheme optimized for color blindness and b/w printouts.

			Password Manager														
Nr	Description	#(Seamless)	LastPass	Norton	Avira	Dashlane	1Password X	Bitwarden	RoboForm	Keeper	Blur	Enpass	KeepassXC	Passbolt	Chrome PWM	Firefox PWM	Edge PWM
Additional Elements (Auth)																	
AA-01	Multiple input fields for single input, e.g. 5 input fields for a 5-letter PIN	0	🟥	🟥	🟥	🟥	🟥	🟨	🟥	🟥	🟥	🟨	🟥	🟥	🟥	🟥	🟥
AA-02	Multiple login buttons (e.g. user & password fields, Google signin, SSO)	2	🟨	⬛	⬛	⬜	⬛	⬛	🟥	🟥	⬛	⬜	🟨	⬛	⬛	⬛	⬛
AA-03	Site includes more authentication-related forms than necessary (e.g. login & registration)	7	🟥	⬜	🟥	⬜	⬜	🟥	⬜	⬜	🟥	⬜	🟨	⬜	🟥	🟥	🟥
AA-04	Site includes more authentication fields than user & password (e.g. lastname)	3	🟨	🟥	🟥	⬜	🟨	🟨	⬜	🟨	🟨	⬜	🟨	🟥	🟥	🟥	🟥
Additional Elements (Non-Auth)																	
AN-01	Site includes an admin panel with multiple user authentication fields	7	🟥	⬜	🟥	🟨	⬜	⬜	⬜	🟨	🟥	⬜	⬜	⬜	🟥	🟨	🟥
AN-02	Site includes checkbox and radio buttons that the PWM is supposed to remember (e.g. ”Remember Login”)	1	🟥	🟥	🟥	🟥	🟨	🟨	🟥	🟥	🟥	⬜	🟥	🟥	🟥	🟥	🟥
AN-03	Site with interactable elements such as drop-downs unrelated to authentication, but affected by the PWM	10	⬜	⬜	⬜	⬜	⬜	🟥	⬜	⬜	⬜	🟥	⬜	⬜	🟥	🟥	🟥
AN-04	Site with form submits unrelated to authentication	15	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
Domain Matching																	
D-01	Base domain with subdomains using the same credentials	9	⬜	⬜	⬜	🟨	⬜	⬜	⬜	⬜	⬜	⬜	🟨	🟨	🟨	🟨	🟨
D-02	Multiple distinct domains (e.g. ShareLaTeX & Overleaf) using the same authentication realm	0	🟥	🟥	🟨	🟥	🟨	🟨	🟨	🟥	🟥	🟨	🟨	🟨	🟥	🟥	🟥
D-03	Base domain with subdomains using separate services	4	⬜	🟨	🟨	🟨	🟨	🟨	🟨	🟥	🟥	🟨	⬜	⬜	🟨	⬜	🟨
D-04	Login is available via both HTTP (port 80) and HTTPS (port 443)	4	🟨	🟥	🟨	🟨	🟨	⬜	⬜	🟨	⬜	⬜	🟨	🟨	🟥	🟥	🟥
D-05	Multiple redirects after login submission that obstruct auto-save	10	⬜	🟨	⬜	⬜	⬛	⬜	⬜	⬜	🟥	⬜	🟨	⬛	⬜	⬜	⬜
D-06	Login form in an iframe which loads a different website	3	🟥	🟥	🟨	🟨	⬜	🟥	🟨	🟥	🟥	⬜	🟨	🟥	🟨	⬜	🟨
D-07	Base domain with several paths or subpages with seperate services	3	⬜	🟨	🟨	🟥	⬜	🟨	🟨	🟥	🟥	🟨	⬜	🟨	🟥	🟥	🟥
Input Fields																	
I-01	Input field definition includes e.g. ”code” substring, can be confused with TOTP codes.	12	⬜	⬜	⬜	⬜	⬜	🟥	⬜	🟨	🟥	⬜	⬜	⬜	⬜	⬜	⬜
I-02	Input field has misleading or unusual name, e.g. ”auth” or ”IDToken1” for username fields	13	🟨	⬜	⬜	⬜	⬜	⬜	⬜	🟨	⬜	⬜	⬜	⬜	⬜	⬜	⬜
I-03	Input field has misleading or unusual type, e.g. ”password” for TOTP fields	11	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟨	⬜	⬜	🟥	🟥	⬜	🟥	⬜
I-04	`autocomplete` tags are used within input fields (e.g. `autocomplete=username`)	9	⬜	⬜	🟥	🟥	⬜	🟥	🟨	🟥	🟥	⬜	⬜	⬜	⬜	⬜	⬜
I-05	No information on type of input field (i.e. no hints in ID, name, type or other attributes)	5	🟥	⬜	🟨	⬜	🟥	🟥	⬜	🟥	⬜	⬜	🟥	🟥	🟥	🟥	🟥
I-06	Form with <textarea> for username instead of input field	3	🟥	🟥	🟨	🟥	🟥	🟥	⬜	🟥	⬜	⬜	🟥	🟥	🟥	🟥	🟥
I-07	Input field has no type attribute	7	⬜	⬜	🟨	⬜	🟥	⬜	⬜	🟥	⬜	⬜	🟥	🟥	🟥	🟥	🟥
I-08	Input field has a max-length smaller than the pre-generated password	1	🟥	🟥	🟥	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
I-09	Website manipulates input data to a semantical equivalent (e.g. changes email to uppercase)	10	⬜	⬜	⬜	🟥	⬛	⬜	🟥	⬜	⬜	⬜	⬜	⬜	🟥	⬜	🟥
JavaScript																	
J-01	Hidden password field (e.g. display:none HTML-style)	6	⬜	🟥	⬜	⬜	🟨	⬜	🟥	🟥	⬜	🟥	🟨	🟥	🟨	⬜	🟨
J-02	Submit button is only enabled after registering a keypress event	12	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟥	🟨	⬜	🟨
J-03	Multiple login steps on one page: only user field initially visible, password shows up in later step.	0	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥	🟨	🟨	🟨	🟨	🟨	🟨	🟨
J-04	Multiple login steps over multiple pages: only user field initially visible, password shows up in later step.	0	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
J-05	Site manipulates input in some way (e.g. substitutes with ****, adds whitespaces, deletes automated inputs)	7	⬜	🟥	🟥	🟨	⬛	⬜	🟥	⬜	⬛	🟥	⬜	⬛	⬜	⬜	⬜
J-06	Site with a dummy password field that is swapped with an initially hidden real one on interaction	3	🟥	⬜	🟥	🟥	🟥	🟥	⬜	⬜	🟥	🟥	🟥	🟥	🟨	🟥	🟨
Non-Standard Forms																	
N-01	No form tag around the login fields that the (login-)data via Ajax	9	⬜	🟨	⬜	⬜	⬛	🟥	⬜	⬜	⬛	⬜	🟥	⬛	⬜	⬜	⬜
N-02	Form element is a custom WebComponent-Tag	15	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
N-03	Submit button is a div with role=button	8	⬜	🟥	⬜	🟨	⬛	🟥	⬜	⬜	⬛	🟥	⬜	⬛	⬜	⬜	⬜
N-04	Form submit button is not part of the form itself	9	⬜	🟨	⬜	⬜	⬛	🟥	⬜	⬜	⬛	⬜	🟥	⬛	⬜	⬜	⬜
N-05	Hyperlink instead of submit button that triggers JavaScript	10	⬜	🟨	⬜	⬜	⬛	⬜	⬜	⬜	⬛	⬜	🟥	⬛	⬜	⬜	⬜
Timing																	
T-01	Delay initializing authentication fields (pages load slowly)	9	⬜	🟥	⬜	⬜	⬜	⬜	⬜	🟨	🟨	⬜	🟥	⬛	⬜	🟥	⬜
T-02	Delay between generation of TOTP and input submission (e.g. due to users waiting too long)	0	⬛	⬛	⬛	🟨	🟥	🟥	⬛	🟨	⬛	🟨	🟨	⬛	⬛	⬛	⬛
Web Standards																	
W-01	HTTP basic authentication as login method	4	🟥	🟥	🟥	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟨	🟥	⬜	⬜	⬜
W-02	Multiple files behind HTTP basic authentication triggering multiple authentication requests that might fail due to request-reply mismatches	3	🟥	🟥	🟥	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟨	🟥	🟨	⬜	⬜

realm is the same across multiple domains. PWMs commonly solved this by letting users search accounts and displaying a warning to make them aware of potential phishing attacks. After confirming the warning, the credentials could be autofilled. Another interesting case is D-04 where both HTTP and HTTPS protocols were used to access a website. We rate the interaction as seamless if PWMs allowed to insert the same credentials to the HTTP page, which is critical from a security perspective. Common solutions we observed included PWMs offering autofill with warnings and requiring additional interaction. This seems to be a sensible compromise until enforcing HTTPS has finally become the default solution.

Input Fields. This category covers atypical or confusing input fields in forms. Within our tests, I-08 revealed to be the most challenging, being our only input field example including a type of password policy by enforcing a maximal length in the password field. Only one PWMs handled this case seamlessly, which we defined as limiting the filled password to the allowed number of symbols and offering to save this as a new password as this meant that the PWM detected the different input. Most PWMs ignored the constraint, inserting their full-length passwords, which might lead to rejected requests or other problems after submitting. We did not expect this result, because the `max-length` attribute is machine-readable and therefore should help the PWMs instead of posing a difficult interaction. This finding is in line with our results for I-04, in which the `autocomplete`-attribute is used. This allows websites to specify the expected type of input, which was a new password in this case. The other interactions in this category focused on different levels of noise that obstruct the PWM. While I-01 to I-03 named input fields in unusual ways and posed no problem to most (eleven to thirteen) of our tested PWMs, interactions I-05 to I-07 omitted information such as `type=password` or used unusual input types like `textarea`. This introduced security risks such as autofilling passwords into cleartext fields. Only between three and seven PWMs achieved seamless interactions.

JavaScript. This category focuses on JavaScript as part of the login process of websites. Here, J-02 is the best working interaction, which included a submit button that was disabled until a keypress event is registered. We found that twelve PWMs (80%) provided a seamless interaction and two more send a somewhat delayed event. Further interesting cases were logins with multiple steps (J-03 & J-04), e. g. by showing a username field and a button to continue to the password input. Multiple widespread service providers like Google and Microsoft utilize this authentication workflow, and while almost all PWMs had workarounds or very simple fixes using page re-scans to support them, no PWM provided a seamless experience. This meant that the PWMs with autologin features were not able to utilize them, and the PWMs with autofill left fields empty and required manual interaction. Overall, this demonstrates the difficulty of these interactions, especially in J-04, where the password field was hidden behind another GET-request, therefore requiring an additional page load, the PWM has no way of detecting if this form will be used for authentication. Finally, J-05 and J-06 directly manipulate input fields after they are filled. In J-05, inputs are replaced with asterisks (*) while J-06 swaps a hidden real password field with the initially visible one when it is focused. Both mechanisms were employed as security mechanisms and were problematic for PWMs. While J-05 achieved seven seamless interactions as the substitution did not obstruct the PWMs, only three PWMs detected the field replacement in J-06 and filled the correct one.

Non-Standard Forms. We further investigated how well PWMs are suited to work with non-standard forms, which includes input fields or submit buttons that deviate from web standards by e. g. using custom tags (N-02) or placing the submit button outside of the authentication form (N-04). As PWMs rely on certain criteria such as web standards to detect relevant authentication input fields, they can struggle with detection if a website deviates from widespread best practices. Overall, we found that most tested PWMs were able to detect malformed login forms. Within our sample, seven (46%) PWMs were able to fulfill our requirements seamlessly. Three received ratings of non-applicable for most interactions in this category as they do not support autosave. The only exception is N-02 (custom WebComponent-tag as form element), with which every PWM was able to interact without problems. Here, we only required autofill instead of autosave due to our endeavor to stay close to the examples we found in the user feedback evaluation. We argue that PWMs use similar approaches to detect input fields, which should therefore not affect the performance of autosave or autofill functionalities.

Timing. This category dealt with two test cases in which some sort of delay can obstruct PWMs as relevant information is not (yet) available. Between both cases, support differed vastly: While nine (60%) of the tested PWMs had no problem with T-01 (input fields appear after a few seconds), none were able to succeed in T-02 (invalid TOTP code due to delay before submission) without any additional user intervention. For T-01, we observe three PWMs that were unable to fill the input fields without manual interaction or a re-scan of the website. Two PWMs require the user to e. g. click on the icon of the PWM, which triggers the autofill. We suspect that the PWMs without solution only scan the page once as soon as it is available, and therefore miss fields that become accessible at a later time. Interaction T-02 expects the PWMs to fill a TOTP code besides username and password. This does not only require them to be able to store and generate TOTPs, but to also be able to fill and update them if they turn invalid. We found that most (nine, 60%) tested PWMs do not support TOTP in general, rendering us unable to test this MWE with them. For the remaining PWMs, none were able to update the code while four PWMs can generate and autofill the currently valid TOTP. Finally, two only generate a TOTP and require users to copy and paste it. We consider these to be unable to succeed in this case.

Web Standards. Finally, we included two Web Standard MWEs that both included HTTP Basic Authentication. [44].

Here, almost all PWMs behaved in the same way in both MWEs, with the exception of Chrome, where W-01 (HTTP Basic Authentication as login method) worked flawlessly, while autosaving credentials in W-02 (multiple files on the website are secured with HTTP Basic Authentication, resulting in multiple authentication requests) required explicit user approval and reloading the website. Although HTTP Basic Authentication is one of the oldest and most basic login mechanisms, most PWMs (ten, 66%) struggle with detecting its pop-ups as login input fields, with only one of them warning the user that autofill is not possible and that they should copy and paste their credentials manually. Within the remaining PWMs, four are able to detect and fill the form, while one offers a setting to activate basic authentication support.

D. Replication Package

To support the replication of our work, we make the following item available as part of our replication package: We include the set of collected GitHub issues (cf. IV-B), Chrome Web Store user reviews and support requests (cf. IV-B) as well as the code book created as a result of our coding process (cf. IV-C Step 4) and the resulting analysis results (cf. Table V-C).[3] The implementation of our MWEs is included as a web application, supported by sample videos of screen casts documenting our analysis process and the set of extension packages used during our analysis.

VI. Discussion

In this section, we discuss our findings and develop recommendations to help PWMs and websites to improve authentication experience, as well as pointing existing shortcomings.

A. User complaints

To address *RQ1: Which interaction patterns on the web are problematic for PWMs?* we decided to investigate user feedback, in order to collect feedback for real-world deployments in uncontrolled environments. We reviewed the related websites and interactions, sorted them into categories and used these findings to develop MWEs that reflected interaction problems. During this process, we found that almost no user feedback covered the password generation. This is surprising because from a security perspective, this is one of the main advantages of PWMs, and the few interactions we found concerning this feature (like I-08: max-length attribute for a password field) indicate that it should be a common issue across PWMs. We reason that we found only very little user feedback for this feature because users expect this to be the website's fault or might not be using the feature. This could hint at an awareness problem concerning the security benefits and possibilities of PWMs. The number of failures means that password generation itself is problematic. The low amount of user feedback we found for this large field of password rules indicates that perhaps there are other problematic issues that users are unlikely to report (at least to our chosen PWM feedback sources).

B. PWM Evaluation

To investigate *RQ2: How do PWM browser extensions handle these interactions?* we collected the interactions of 15 PWMs with the MWEs we developed previously. Here, we found numerous interaction problems, pointing us towards the most problematic areas for PWMs that we need to work on. In summary, only two interactions were seamless for all tested password managers: N-02 (the custom web component tag) and AN-04 (form not at all related to authentication), both found on PWMs with less than 10,000 downloads, likely indicating ”non-issues” that should simply be fixed by the PWMs they were reported for. On the other hand five interactions where not solved seamlessly by any PWM we tested, indicating a gap in possible approaches to solve the problems represented by these MWEs. For these and any other difficult or important case, we would like to formulate recommendations in the following section.

C. Recommendations

This sections aims to use our findings in the previous studies in order to answer *RQ3: What can be done to improve the interaction between PWMs and websites?* We discuss potential improvements for the interaction problems we found and discuss their feasibility. While we try to focus on improvements that are possible on the web right now, we found a few gaps that need to be filled by the major web standards before websites or PWMs can address them. We highlight these responsible actors at the heading of the following recommendation paragraphs.

Support for Using Credentials in Multiple Environments (Webstandards). One strong pointer here is found in interactions D-01 to D-04 and D-07: a service that spans or does not span across multiple domains, paths and protocols. The underlying problem is most likely the lack of standardization: Since there is no best practice for PWMs how to match different origins or how to detect whether or not different URLs serve the same service and should therefore be provided with the same credentials, and since posting username and password combinations to the wrong website poses a critical security risk, PWM have adopted different default strategies that mostly lead to non-recognition of URLs that deviate from the stored websites. Additionally, some of these cases require opposing behaviors: Fulfilling all interactions seamlessly would require PWMs to identify e. g. subdomains as both the same or different services that they should or should not match at the same time. Upon investigating how open source PWMs solve these interactions we found a list of equivalent websites within the repository of one of the open source [5]. This indicates a gap within standards, since websites likely also want to indicate this kind of behavior to any authentication mechanism. To solve this, a website could use a process similar to the Cross Origin Resource Sharing header, perhaps in combination with the content security policy. Currently no fitting standards exists however, so this would need further investigation and discussion In the case

[3] The replication package is provided at https://publications.teamusec.de/passwordManagers

of D-04 (protocol switch from HTTP to HTPSc) a seamless interaction represents a security risk, which means that this rating is undesirable. However, this case represents a usability and security trade-off, as the conditional interactions we found mostly rely on warnings, which in the past have proven to be an ineffective security measure on their own [33]. Furthermore, HTTPS is likely to become a web default [43], [45]. Websites will need to comply or become effectively inaccessible, which in principle solves this problem. Furthermore, hard coding information that could change in the future, such as domains, can lead to security issues later on, as domain owner changes are possible.

Better Support for Multipage Login (Websites). Other examples for predefined data include rules to improve support for logins that span multiple pages as tested in J-03 & J-04. For popular services like Google, we found that they rely on lists of predefined sites in PWMs that require special treatment. We found a list of websites with username-only logins within the repository of one of the open source PWMs [70]. Similar to hard coded domains, this information can change if a website changes its authentication mechanism which can cause the respective PWM to malfunction. Websites could potentially use the `autocomplete`-attribute [18] to indicate the need to fill username on one page and password for the account on the other, but this needs widespread adoption to prevent the need for hard coded lists.

Support for Custom Fields (PWMs, Websites). We further were able to observe a lack of support for custom fields within login forms. Many of the PWMs we tested were unable to detect, store or autofill them, while several others could only do so after manual interaction. This also often required a tedious search for the correct required settings within the PWM, which we deem to be an usability shortcoming. We argue that while it might not always be possible to automatically detect all relevant custom fields, PWMs should offer an easily accessible way to store additional data for all credentials. Furthermore, the HTML `autocomplete`-attribute [18] already offers a standardized way to pre-define identity-related content types that an input field expects. While we cannot judge the prevalence of this attribute on websites, we suggest this as a solution that website managers can adopt to help PWMs correctly detect and autofill additional information.

Avoid Obscure JavaScript (Websites). Another striking finding in our evaluation of user feedback was the frequent mentions of websites that showed some kind of odd behavior, obstructing the PWM. In these cases, the websites made use of JavaScript to implement e. g. security measures, which are mostly meant as security measures to avoid automatic interaction with the website. This includes websites that substitute the password with asterisks after submission, delete automated input, or use a hidden input field for passwords that only appears if the dummy field is focused. Features like this are aimed at increasing security and were mostly found on e. g. banking websites, however, we argue that while there is a fitting threat model, this type of attack is very unlikely and does not justify the decreased usability by essentially blocking PWMs. We further found that some PWMs, especially those build into common browsers, use JavaScript to interact with website elements. While this is usually done to detect all possibly relevant input fields on a page, we also found PWMs who manipulate the website after credentials were entered or after submitting a login. Manipulating websites in this way can cause the website to malfunction, as it might trigger events linked to the respective website elements and is further not necessary to detect input fields.

Support for HTTP Basic Authentication (Password Managers). While basic authentication is not frequently used anymore, we argue that as a web authentication standard it should be supported by PWMs. Basic authentication is not part of the website, but a mechanism provided by the respective browser, and as such requires different detection approaches than those used to identify input fields. However, most current browsers provide APIs that PWMs could use to access and therefore support basic authentication forms [6], [31].

Better Support for TOTP (Password Managers). Finally, we found that almost none of the PWMs we investigated supports the automatic generation and filling of TOTP. This can be regarded as a double-edged sword: While the adoption of a TOTP feature increases convenience and usability of a password manager by easing the process for users, it also removes the multiple factors as everything is stored in one place, rendering the PWM a single point of failure. A way to mitigate this issue is the addition of a separate smartphone application that is necessary to use TOTP, which is already offered by some PWMs such as Dashlane. This could further be used to increase security by allowing the PWM to send push notifications with authentication requests to a user's phone, adding a second factor without necessarily adopting a TOTP mechanism.

VII. Conclusion

This work identified several shortcomings of both modern desktop PWMs as well as current authentication implementation on websites, which lead to interaction problems between them. While our work provides the first systematic analysis of interaction problems between desktop PWMs and websites, we identified different areas for future work. Although we collected and labelled Chrome Web Store reviews and GitHub issues until we reached saturation, our list of interaction problems is not exhaustive. Further work could review additional resources such as issue trackers by browser vendors. Furthermore, our work focuses on desktop PWMs. However, mobile PWMs are getting more important and bring their own opportunities and limitations. Hence, a similar analysis could be helpful for mobile PWMs.

References

[1] 1password x – password manager. https://1password.com. Last visited: 11/23/2020.

[2] Angular forms and password managers - stack overflow. https://stackoverflow.com/questions/53911864/angular-forms-and-password-managers/53956890#53956890. Last visited: 12/01/2020.

[3] Apple password autofill. https://developer.apple.com/documentation/security/password_autofill. Last visited: 12/03/2020.

[4] Autofill framework — android developers. https://developer.android.com/guide/topics/text/autofill. Last visited: 12/03/2020.

[5] Bitwarden: Added discord to global equivalent domain #752. https://github.com/bitwarden/server/pull/752/commits/c4164f5703babc5d11d8cf1e8426c442b4969573. Last visited: 12/04/2020.

[6] chrome.webrequest - google chrome. https://developer.chrome.com/extensions/webRequest#event-onAuthRequired. Last visited: 12/03/2020.

[7] Content security policy level 3. https://www.w3.org/TR/CSP3/. Last visited: 12/03/2020.

[8] Dashlane - password manager. https://www.dashlane.com. Last visited: 12/03/2020.

[9] Exciting news — sharelatex is joining overleaf - overleaf, online latex editor. https://www.overleaf.com/blog/518-exciting-news-sharelatex-is-joining-overleaf. Last visited: 12/03/2020.

[10] Fetch standard - 3.2. cors protocol. https://fetch.spec.whatwg.org/#http-cors-protocol. Last visited: 12/03/2020.

[11] Flask. https://palletsprojects.com/p/flask/. Last visited: 12/03/2020.

[12] Github - bitwarden/browser: The browser extension vault (chrome, firefox, opera, edge, safari, & more). https://github.com/bitwarden/browser. Last visited: 12/03/2020.

[13] Github - keepassxreboot/keepassxc-browser: Keepassxc browser extension. https://github.com/keepassxreboot/keepassxc-browser/. Last visited: 12/03/2020.

[14] Github - keepassxreboot/keepassxc: Keepassxc is a cross-platform community-driven port of the windows application "keepass password safe". https://github.com/keepassxreboot/keepassxc/. Last visited: 12/03/2020.

[15] Github - passbolt/passbolt_browser_extension: Passbolt browser extensions (firefox & chrome). https://github.com/passbolt/passbolt_browser_extension/. Last visited: 12/03/2020.

[16] GitHub main page. https://github.com/. Last visited: 12/03/2020.

[17] How is a two-step login better than single-step when you have a password manager? https://ux.stackexchange.com/questions/124021/how-is-a-two-step-login-better-than-single-step-when-you-have-a-password-manager. Last visited: 12/01/2020.

[18] Html 5.2: 4.10. forms. https://www.w3.org/TR/html52/sec-forms.html#autofilling-form-controls-the-autocomplete-attribute. Last visited: 12/03/2020.

[19] Html standard - 7.5 origin. https://html.spec.whatwg.org/#origin. Last visited: 12/03/2020.

[20] Lastpass: Free password manager. https://www.lastpass.com/. Last visited: 11/23/2020.

[21] Netmarketshare: Browser market share. https://netmarketshare.com/browser-market-share.aspx. Last visited: 12/03/2020.

[22] Norton password manager. https://norton.com/feature/password-manager. Last visited: 12/03/2020.

[23] Same origin policy - web security. https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy. Last visited: 12/03/2020.

[24] Same-origin policy - web security — mdn. https://www.w3.org/Security/wiki/Same_Origin_Policy. Last visited: 12/03/2020.

[25] Selenium with python — selenium python bindings 2 documentation. https://selenium-python.readthedocs.io/. Last visited: 12/03/2020.

[26] Seleniumhq browser automation. https://www.selenium.dev/. Last visited: 12/03/2020.

[27] Stackoverflow: How to create a minimal, reproducible example. https://stackoverflow.com/help/minimal-reproducible-example. Last visited: 12/03/2020.

[28] Statcounter: Browser market share. https://gs.statcounter.com/browser-market-share#monthly-202010-202010-bar. Last visited: 12/03/2020.

[29] Web scraper - free web scraping. https://chrome.google.com/webstore/detail/web-scraper-free-web-scra/jnhgnonknehpejjnehehllkliplmbmhn. Last visited: 12/03/2020.

[30] Web scraper - the #1 web scraping extension. https://webscraper.io/. Last visited: 12/03/2020.

[31] webrequest.onauthrequired - mozilla — mdn. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest/onAuthRequired. Last visited: 12/03/2020.

[32] Wikimedia foundation: User agent breakdowns. https://analytics.wikimedia.org/dashboards/browsers/#all-sites-by-browser. Last visited: 12/03/2020.

[33] D. Akhawe and A. P. Felt. Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness. In *Proc. 22nd Usenix Security Symposium (SEC'13)*. USENIX Association, 2013.

[34] I. Becker, S. Parkin, and M. A. Sasse. The rewards and costs of stronger passwords in a university: linking password lifetime to strength. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 239–253, 2018.

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

[36] 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 *Proc.\ 33rd IEEE Symposium on Security and Privacy (SP'12)*. IEEE, 2012.

[37] 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 *2012 IEEE Symposium on Security and Privacy*, pages 553–567. IEEE, 2012.

[38] I. Cherapau, I. Muslukhov, N. Asanka, and K. Beznosov. On the impact of touch ID on iphone passcodes. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 257–276, Ottawa, July 2015. USENIX Association.

[39] S. Chiasson, P. C. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *USENIX Security Symposium*, volume 15, pages 1–16, 2006.

[40] J. Colnago, S. Devlin, M. Oates, C. Swoopes, L. Bauer, L. Cranor, and N. Christin. "it's not actually that horrible" exploring adoption of two-factor authentication at a university. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2018.

[41] J. S. Conners and D. Zappala. Let's authenticate: Automated cryptographic authentication for the web with simple account recovery. *Who Are You*, 2019.

[42] H. de Vries. Making password managers play ball with your login form. https://hiddedevries.nl/en/blog/2018-01-13-making-password-managers-play-ball-with-your-login-form. Last visited: 12/01/2020.

[43] Evolving Chrome's security indicators, May 2017. visited.

[44] R. T. Fielding and J. Reschke. Hypertext Transfer Protocol (HTTP/1.1): Authentication. RFC 7235, June 2014.

[45] Firefox 83 introduces HTTPS-Only Mode , Nov. 2020. visited.

[46] M. Golla, M. Wei, J. Hainline, L. Filipe, M. Dürmuth, E. Redmiles, and B. Ur. " what was that site doing with my facebook password?" designing password-reuse notifications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1549–1566, 2018.

[47] J. Gray, V. N. Franqueira, and Y. Yu. Forensically-sound analysis of security risks of using local password managers. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 114–121. IEEE, 2016.

[48] M. Harbach, S. Fahl, and M. Smith. Who's Afraid of Which Bad Wolf? A Survey of IT Security Risk Awareness. In *Proc. 27th Computer Security Foundations Symposium (CSF'14)*. IEEE, 2014.

[49] A. Karole, N. Saxena, and N. Christin. A comparative usability evaluation of traditional password managers. In *International Conference on Information Security and Cryptology*, pages 233–251. Springer, 2010.

[50] Z. Li, W. He, D. Akhawe, and D. Song. The emperor's new password manager: Security analysis of web-based password managers. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 465–479, 2014.

[51] S. G. Lyastani, M. Schilling, S. Fahl, M. Backes, and S. Bugiel. Better managed than memorized? studying the impact of managers on password strength and reuse. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 203–220, 2018.

[52] S. G. Lyastani, M. Schilling, M. Neumayr, M. Backes, and S. Bugiel. Is fido2 the kingslayer of user authentication? a comparative usability study of fido2 passwordless authentication. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 268–285. IEEE, 2020.

[53] R. Maclean and J. Ophoff. Determining key factors that lead to the adoption of password managers. In *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, pages 1–7. IEEE, 2018.

[54] N. McDonald, S. Schoenebeck, and A. Forte. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for cscw and hci practice. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–23, 2019.

[55] J. Neale. Iterative categorization (ic): a systematic technique for analysing qualitative data. *Addiction*, 111(6):1096–1106, 2016.

[56] S. Oesch and S. Ruoti. That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers. In *Proc. of USENIX Security Symp*, 2020.

[57] Password Rules Validation Tool. visited.

[58] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget. 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*, pages 295–310, 2017.

[59] S. Pearman, S. A. Zhang, L. Bauer, N. Christin, and L. F. Cranor. Why people (don't) use password managers effectively. In *Fifteenth Symposium On Usable Privacy and Security (SOUPS 2019). USENIX Association, Santa Clara, CA*, pages 319–338, 2019.

[60] H. Ray, F. Wolf, R. Kuber, and A. J. Aviv. Why older adults (don't) use password managers. *arXiv preprint arXiv:2010.01973*, 2020.

[61] K. Reese, T. Smith, J. Dutson, J. Armknecht, J. Cameron, and K. Seamons. A usability study of five two-factor authentication methods. In *Fifteenth Symposium on Usable Privacy and Security ({SOUPS} 2019)*, 2019.

[62] J. Reynolds, N. Samarin, J. Barnes, T. Judd, J. Mason, M. Bailey, and S. Egelman. Empirical measurement of systemic 2fa usability. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 127–143, 2020.

[63] J. Reynolds, T. Smith, K. Reese, L. Dickinson, S. Ruoti, and K. Seamons. A tale of two studies: The best and worst of yubikey usability. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 872–888. IEEE, 2018.

[64] D. Silver, S. Jana, D. Boneh, E. Chen, and C. Jackson. Password managers: Attacks and defenses. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 449–464, 2014.

[65] F. Stajano. Pico: No more passwords! In *International Workshop on Security Protocols*, pages 49–81. Springer, 2011.

[66] F. Stajano, M. Spencer, G. Jenkinson, and Q. Stafford-Fraser. Password-manager friendly (pmf): Semantic annotations to improve the effectiveness of password managers. In *International Conference on Passwords*, pages 61–73. Springer, 2014.

[67] E. Stobert and R. Biddle. Expert password management. In *International Conference on Passwords*, pages 3–20. Springer, 2015.

[68] B. Stock and M. Johns. Protecting users against xss-based password manager abuse. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 183–194, 2014.

[69] J. Tan, L. Bauer, N. Christin, and L. F. Cranor. Practical recommendations for stronger, more usable passwords combining minimum-strength, minimum-length, and blocklist requirements. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1407–1426, 2020.

[70] K. Team. Add fidelity.com to Predefined Sites - KeepassXC Github Repository, 11 2020. https://github.com/keepassxreboot/keepassxc-browser/blob/develop/keepassxc-browser/common/sites.js (visited on 03/12/20, commit 7ee83ede26fb974d6366a64e6ef15e703eb6166d).

[71] B. Ur, F. Noma, J. Bees, S. M. Segreti, R. Shay, L. Bauer, N. Christin, and L. F. Cranor. "i added '!'at the end to make it secure": Observing password creation in the lab. In *Symposium on Usable Privacy and Security (SOUPS)*, 2015.

[72] K. C. Wang and M. K. Reiter. How to end password reuse on the web. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[73] R. Wash, E. Rader, R. Berman, and Z. Wellmer. Understanding password choices: How frequently entered passwords are re-used across websites. In *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)*, pages 175–188, 2016.

[74] R. Zhao and C. Yue. All your browser-saved passwords could belong to us: A security analysis and a cloud-based new design. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 333–340, 2013.