# ByPass: Reconsidering the Usability of Password Managers

Elizabeth Stobert[1], Tina Safaie[2(✉)], Heather Molyneaux[3], Mohammad Mannan[2], and Amr Youssef[2]

[1] Carleton University, Ottawa, ON, Canada
elizabeth.stobert@carleton.ca
[2] Concordia University, Montreal, QC, Canada
t_safaie@encs.concordia.ca,m.mannan@concordia.ca,
youssef@ciise.concordia.ca
[3] National Research Council of Canada, Fredericton, NB, Canada
heather.molyneaux@nrc-cnrc.gc.ca

**Abstract.** Since passwords are an unavoidable mechanism for authenticating to online services, experts often recommend using a password manager for better password security. However, adoption of password managers is low due to poor usability, the difficulty of migrating accounts to a manager, and users' sense that a manager will not add value. In this paper, we present ByPass, a novel password manager that is placed between the user and the website for secure and direct communication between the manager and websites. This direct communication allows ByPass to minimize the users' actions needed to complete various password management tasks, including account registration, logins, and password changes. ByPass is designed to minimize errors and improve usability. We conducted a usability evaluation of ByPass and found that this approach shows promising usability, and can help users to better manage their accounts in a secure manner.

**Keywords:** Authentication · Usable security · Password manager · API

## 1 Introduction

Password-based online services are ubiquitous, despite many known security and usability limitations of passwords [10,19]. Password managers can alleviate some of these drawbacks by removing the need for people to memorize a large number of strong passwords, helping users cope with different password policies, creating unique passwords and providing protection features against some attacks (e.g., phishing). However, adoption of password managers is low, and even among password manager users, many still do not use them effectively [27].

Previous studies of password managers [3,5] show that usability issues can contribute to low adoption. While several studies [7,26] have investigated different ways to improve the usability of password managers, the basic wallet


N. Park et al. (Eds.): SecureComm 2020, LNICST 335, pp. 446–466, 2020.
https://doi.org/10.1007/978-3-030-63086-7_24

structure of the password manager has remained the same. In all of them, a password manager is regarded as client-side software that helps users store and fill passwords on a targeted website without the website recognizing whether it is the user who is filling the password field or a software program.

In this work, we investigate how the design of password managers can be rethought to facilitate adoption and minimize usability problems. We design and develop ByPass, a new password manager that sits between the user and the website, reducing friction resulting from usability problems. ByPass uses an API for direct communication between the website and password manager, which allows the password manager to not only directly send credentials to the website, but also to query the website for information such as the password policy. In addition, this communication channel allows the introduction of innovative features, such as automated password changes, and account creation/deletion through the password manager. The primary goal of ByPass is to provide a more usable password management system that encourages users to behave securely. ByPass integrates nudges for secure behaviour, and is designed to make tasks such as account migration and password changes as simple as possible.

No work so far has focused on using APIs within the password manager development. Our key contribution is developing a password manager with new capabilities resulting from different API calls from the websites. Although this approach requires buy-in from websites, it results in increased password manager usability resulting from fewer actions required by the user. Assuming the integration of password-based online services, in ByPass, we shift the focus to comprehensive *account management* instead of current approaches to *password management*, primarily limited to create/save/fill passwords. We address critical security limitations of current password managers, and at the same time, improve ease of use and control over numerous accounts that a typical user needs to manage.

In this paper, we focused on ByPass's impact on end users. We evaluated the usability of our prototype implementation of ByPass by conducting a study with 20 participants. We found that participants were able to quickly and easily add new accounts to ByPass, migrate existing accounts into the manager, and change their passwords. Users were generally positive about the features in ByPass, but expressed concerns about the ways in which ByPass moves the locus of control.

With ByPass, our goal is not to create yet another password manager. We want this new design, along with the results of our prototype evaluation, to act as a vehicle for investigating more fundamental questions about account management. Password managers, while seeming to be one of the most accessible solutions to the password problem, have not been widely adopted. This suggests that a new approach to password management is needed: users need to be empowered with tools to effectively access and manage the security of their accounts by relieving them of tasks more easily completed by a computer.

## 2 Background

Password managers are software tools primarily designed to save users' passwords. Modern password managers range in sophistication from browser-based

managers that save and fill users' passwords, to more complex standalone programs (e.g., 1Password[1], and Lastpass[2]) that integrate features such as customizable password generation, security audits, and family password sharing. Almost all managers are designed based on a *wallet* model, where the central function is to manage a list of passwords, protected by a master password. The security and usability of password managers have been studied extensively.

### 2.1 Security of Password Managers

By saving all passwords in one place, password managers create a central point of failure in a user's security ecosystem. Although various proposals have been made for password managers that mitigate this problem [16,29,34,36], the majority of commercially-available password managers are vulnerable in this way. The prevalence of such attacks is unclear, but it is compounded by the likelihood that users will choose insecure and easily-guessed master passwords [38].

Silver et al. [33] studied auto-fill password policies among different types of password managers. Although the auto-fill function can increase the usability of password managers, poor implementation of this function can lead to exposure of users' credentials. They also described the *sweep* attack, which requires the attacker to have control over the user's WiFi (e.g., in a public hotspot run/controlled by the attacker); the attacker can extract the user's credentials by injecting malicious JavaScript to the website without the user noticing.

Li et al. [22] analyzed five third-party web-based password managers, revealing four security vulnerabilities—including classical web application vulnerabilities such as Cross Site Scripting (XSS) and Cross Site Request Forgery (CSRF). They also analyzed authorization and user interface vulnerabilities, and proposed guidelines to mitigate these identified issues.

Recently, Carr and Shahandashti [11] revisited various known attacks on password managers, and analyzed whether password managers are still vulnerable to those attacks. They also identified four new vulnerabilities: phishing attacks, clipboard vulnerability, PIN brute force vulnerability (for applications), and brute force via an extension (evaluated against five popular commercial password managers). Their results showed that all of the tested password managers were vulnerable to at least one of the attacks they considered.

### 2.2 Usability of Password Managers

Chiasson et al. [14] conducted an early usability evaluation of two types of password manager, and found that users' poor mental models of password managers caused them to make dangerous errors. Karole, Saxena and Christin [21] compared a third-party password manager to a phone-based manager and a USB manager. They found that users preferred the control implicit in having passwords stored on a local device.

[1] https://www.1password.com/.
[2] https://www.lastpass.com/.

There are a variety of factors leading to the low adoption of password managers. Aurigemma et al. [5] found that insufficient time for installation, lack of immediacy, and users' feeling that using a password manager needs an additional effort that they don't want to spend contributed to password manager non-adoption. Maclean and Ophoff [25] found that trust, habit, and performance expectancy are the three factors that lead to the use of password managers. Pearman et al. [27] interviewed 30 participants, including people who choose not to use a password manager, about their password manager use (and non-use). They found that factors such as lack of awareness and poor understanding of how password managers work prevent people from adopting password managers. They found that users could be divided into two categories: people for whom convenience is a priority, and people for whom security is a priority.

Seiler-Hwang et al. [31] analyzed the usability of smartphone password managers and suggested security, user interaction, and integration with external applications as three key areas for improvement. They mentioned that only knowing about the password manager's existence is not enough to motivate an individual to use it. Users need to be encouraged to install and try password managers. Alkaldi and Renaud [3] identified three phases to user adoption of password managers: searching, trialling, and deciding. They later explored how self-determination theory can be used to encourage users to adopt password managers, and found that recommender tools can cause users to at least search for and try password managers [4].

Lyastani et al. [24] studied whether using a password manager can influence password strength and re-use. They used a browser extension to collect data on different password entry methods. Participants who used technical support for password creation had stronger passwords. They concluded that password generators improve security, but are under-adopted, highlighting the need to rethink password manager workflows.

### 2.3 Proposals for New Password Managers

A number of studies have designed new tools to address existing problems with password management. McCarney et al. [26] created Tapas, a dual possession, theft-resistant password manager in which an adversary needs to gain access to both devices in order to read the saved credentials. In their design, both devices – a mobile phone and a desktop computer – must be available in order to use the password manager, and the user does not need to have any master password. Barbosa et al. [7] designed UniPass, a password manager for visually impaired users. UniPass allowed users with visual impairment to access their accounts and passwords more effectively and to use public computers. Although not positioned directly as a password manager, Ruoti and Seamons [30] propose a system in which passwords are stored in the operating system, and verified (using zero-knowledge proofs) by an independent password checking service. The system minimizes the interaction between users and passwords, as well as the attack surface for passwords.

## 3 ByPass: Design and Implementation

ByPass is designed to address usability problems that prevent users from adopting password managers while maintaining good security properties. In this section, we discuss the design of ByPass, features supported and introduced by it, and our prototype implementation.

### 3.1 Design Overview and Goals

ByPass is designed to reduce friction in password managers resulting from usability problems. Our insight is that many of the usability problems with password managers result from the password manager functioning as an intermediary between the user and the regular login page. Copy/paste errors, form fills happening to the wrong field, leakage of passwords from the paste buffer, all result from placing the password manager as an external resource, accessible only by the user and not the website. Traditional password managers offer little conceptual advantage beyond a list of written passwords, and in doing so, miss an opportunity to genuinely address the usability problems with passwords and add value beyond other password storage mechanisms.

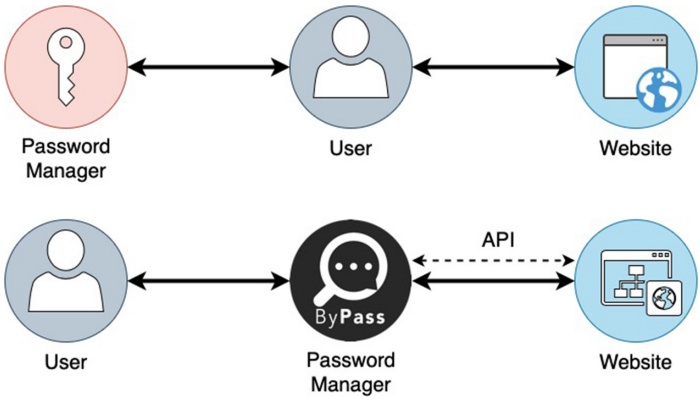


**Fig. 1.** Top: current password managers rely heavily on users for account interaction, other than storing the passwords. Bottom: ByPass is placed directly between the user and the website, allowing it to mediate communications between them.

As shown in Fig. 1, ByPass communicates directly with websites, thus avoiding errors resulting from user manipulation of passwords and forms. It provides an API for websites to use so that it can pass credentials directly to these websites. This allows users to skip the manual login procedure when using ByPass, improving usability and avoiding errors. Our goal is to design a mechanism to reduce the number of clicks and actions required to complete account/login management tasks.

ByPass is designed to nudge users toward choosing secure options. For example, password generation features in password managers remain mostly

unused [24,27,37]. ByPass creates secure randomly-generated passwords by default, and only allows users to select their own passwords by clicking through multiple steps. These generated passwords are secure and unique, and because users usually do not need to interact with these passwords directly, they are encouraged to take advantage of this functionality.

The usability advantages of ByPass are only realized if the website chooses to implement the ByPass API. This is a strong requirement, and we did not want to create a tool that was only usable with affiliated websites. Thus, we designed ByPass to extend the functionality of password managers without taking anything away. ByPass can be used as a repository for copying and pasting passwords in the same way that existing managers can, but adds functionality for websites that implement the needed APIs. Note that when we discuss usability and security of ByPass, we focus on the new design, not this backward-compatibility feature which shares the usual drawbacks of traditional password managers.

Moving the password manager to sit between the user and the website also creates an opportunity to consider other functions that password managers typically exclude, including account creation on third-party websites, account deletion, and automated password changes. The security of ByPass is discussed in Sect. 4, but user authentication to the manager is handled exactly as it is by traditional password managers, i.e., with a username and master password.

### 3.2 ByPass Features

The key features of ByPass include third-party website account creation, account migration, direct account logins, password changes, and account deletion; more features can also be easily accommodated. Figure 2 shows a flow diagram of user interactions when setting ByPass up for a new account.

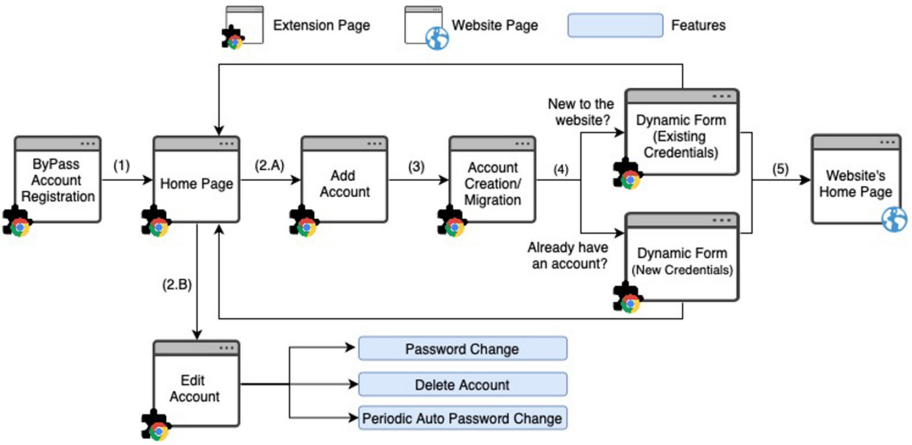


**Fig. 2.** User flow diagram of ByPass.

**Adding Accounts.** ByPass provides two options for users when adding new accounts: to register a new account, or to enroll an existing one; both these tasks are abstracted into the same function: "Adding an account". We used a wizard to guide users through the process of choosing what kind of accounts they are adding to ByPass, and to ensure that they enter the correct information.

Because ByPass communicates directly with websites, all user interactions with accounts can be moved to ByPass, including account registration. When users want to create an account on a new website, they search for the website name in ByPass, and ByPass queries the website for the registration fields they want. Users fill these fields, and ByPass automatically populates the password field with a random password conforming to the password policy (also sent by the website). Users can view and regenerate the random password, but must go through an additional confirmation to do so.

To add an existing account, after searching for the website name in ByPass, users simply enter their username and password, similar to standard password managers. If the user enters the name of a website that does not have the ByPass API installed, the password manager will act like a regular password manager, and store credentials for the user to manually enter.

**Account Login.** After adding an account to ByPass, the user can log into the website by opening ByPass to the account page, and clicking "login". ByPass sends the user's credentials to the website via secure API calls, and opens the site's main page in a browser. The user avoids any direct interaction with login pages, e.g., navigating to intended URLs, and typing usernames and passwords.

**Password Changes.** Password changes are frequently avoided by users due to serious usability drawbacks [20,37]. While password expiration policies are no longer recommended by NIST [2], they are frequently mandated by organizations. Password changes are particularly onerous to users of password managers, who typically must open the manager to copy the old password, navigate to the password change menu, generate a new password in the manager, ensure it is saved in the password manager account entry, and copied into the website. Because passwords are generally masked to avoid shoulder surfing attacks, it can be difficult to know which string is copied into which field, inviting errors. In ByPass, the direct communication between manager and website elides all of these steps, and makes password changes into a one step process for the user. On the edit account page, users can view their password, change it with the (re)generate password function, and set a timer for automatic password changes.

**Account Deletion.** Although not typically considered as part of password management, another opportunity afforded by the ByPass architecture is to easily delete website accounts from the password manager. Account deletion is frequently made difficult by website designers [17], but privacy rights (such as the right to be forgotten [6]) indicate that account deletion should be an available straightforward process. In ByPass, account deletion is accessed through the "Edit account" page, and includes a two-step confirmation process.

### 3.3 Implementation Details

The ByPass implementation has two components: the password manager itself which is implemented as a Chrome browser extension (using JavaScript and HTML, and IndexedDB to store data), and the API that allows secure communication between websites and the password manager. Figure 3 shows the ByPass API interaction between the user's browser and the website backend.

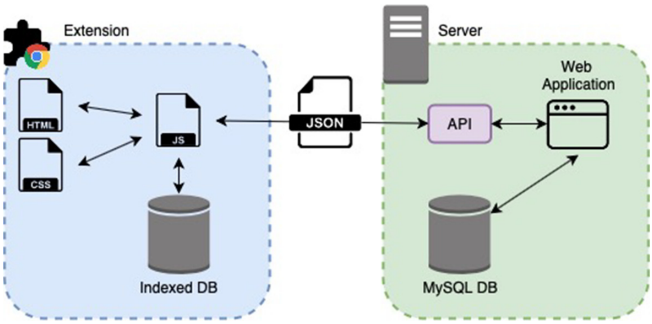


**Fig. 3.** ByPass API interaction between the user browser and website backend.

We created two fictional websites (an e-commerce portal and a webmail platform) for our study, using a Microsoft IIS server application to handle user requests and communicate with the backend database. The password manager side is a browser extension that directly requests and sends information to and from the website, receives responses, and displays the results in the browser.

We used a REST API for communicating with website backends as it improves the performance, has a simple interface, allows independent modification of the components, and requires low bandwidth [15]. API calls are made via HTTPS using JSON ByPass enables various account management functions, and currently implements API calls for creating an account, logging into an account, editing, and deleting an account. Our prototype API was written in C# using the .NET framework, using the System.Web.Http library.

In our fictional websites, we used JSON Web Tokens [18] for authentication purposes. The tokens contain three parts: header, body, and cryptographic signature. The header consists of a type of the token and signing algorithm, the body contains the claim, which is the value that we want to secure, and the cryptographic signature is formed using the header and body. Our TLS connections for both websites were set up through CloudFlare Keyless SSL [1,35].

The implementation details described here are specific to our prototype, but ByPass is designed to work with a wide variety of website setups. The API is packaged as a library that can be included by web developers, and the manager is database-agnostic. The password database is stored offline in this particular implementation, which allows users to make easy backups of the (encrypted)

password list, but limits portability. In this work, our primary focus was to understand the usability of ByPass, and both the feasibility of this approach, and issues such as portability between devices will be explored in future work.

## 4 Security and Attack Mitigation

We first discuss the security of the ByPass database, and then explain some attacks that can be mitigated through our design.

Our current ByPass implementation uses an offline database to store account information. As a result, this database can be subject to offline attacks (e.g., via guessing the master password). Currently, we encrypt the database using AES-256, where the key is derived via PBKDF2 [42] from a user-chosen master password, a random salt, and 100,000 iterations. The encryption key is kept in memory as long as the user session remains active. Existing measures (see e.g., [12,13,32]) can be adopted to enhance resistance against offline attacks.

In terms of reducing attacks, since we perform all the communications between the password manager and websites through TLS, remote network attacks, and SSL stripping attacks are mitigated; a remote attacker cannot intercept and/or modify our communications.

Since ByPass communicates directly with web servers (instead of filling forms on client-facing pages), encouraging users to use ByPass for completing account management functions can mitigate HTTP(S) auto-fill vulnerabilities and subdomain equivalence attacks [9,11]. HTTP(S) vulnerabilities happen when the password manager fills the credentials on an HTTP version of the website while the credentials were saved on the HTTPS version, and in this way the password manager makes an opportunity for the attacker to extract the user's credentials from non-HTTPS pages of the website. Avoiding auto-fill also helps us avoid the *sweep* attacks [33], where credentials are filled to invisible fields in a malicious webpage, and exfiltrated using JavaScript.

Copy to clipboard is a feature that most password managers utilize if they cannot auto-fill the credentials. If the password manager doesn't support enough protection for the credentials that were copied to the clipboard, it will lead to clipboard vulnerabilities, e.g., exposing passwords to other sites/processes; see e.g., [11]. In ByPass, passwords and other information are communicated directly to the website, and no password copying/pasting is used for login or other account-related tasks, which avoids leaking passwords from the paste buffer.

User interface-based password brute-force attacks are another vulnerability that affect extension-based password managers [11] if attackers gain access to the password manager user interface. To reduce this vulnerability, we add a delay based on the specific number of wrong master passwords entered by the user (similar to some leading commercial password managers).

## 5 Usability Evaluation

ByPass is designed to address usability issues that prevent end-users from adopting and using password managers. We conducted two usability evaluations of

ByPass: an inspection-based evaluation of an early prototype, and a lab-based user study of the higher fidelity prototype.

### 5.1 Cognitive Walkthrough

After creating the first prototype implementation of ByPass, we wanted feedback on the usability of the manager. At that point, the prototype included the main functions (adding accounts, logging in, and a version of password changing), but was not sufficiently functional for a user study. We chose to conduct a cognitive walkthrough [39] because of its focus on learnability and because it allowed us to include the perspective of novice users.

We conducted a pluralistic walkthrough with five evaluators, including the project team leads, the developer, and two volunteers playing the part of a novice ByPass user. One of these volunteers had longtime experience using a password manager, and the other had no password manager experience. We walked through the process of creating a new user account on ByPass, migrating an existing account, registering for a new third-party account, changing a password, and logging into a website.

In general, our novice participants found interacting with the prototype confusing, and were unsure what steps to first take, and what information to enter where. This led us to redesign much of the user interface, and include stronger markers of flow between steps, e.g., the account-adding wizard and more prominent navigation buttons.

Another issue that arose in the walkthrough was that the inexperienced participant seemed to have few mental models for common password tasks, and in particular, had no language for describing them. The abstraction taking place in the password manager was confusing to them, and they had multiple problems entering the right (fictitious) credentials, or choosing the correct menu option.

One of our central questions was whether the features in ByPass made sense to users – would using ByPass seem jarring, given that it departs from the usual interaction with websites? We expected the volunteer with previous password manager experience to question how ByPass worked, but they did not comment until prompted by us. When queried, they spoke to the intuitiveness of the feature set, and said that they did not question how those features were working *because they just worked.*

### 5.2 User Study

The second phase of our usability evaluation was to conduct an in-lab user study to evaluate the usability of the ByPass prototype with unbiased users. Following the cognitive walkthrough, the ByPass prototype was completely redeveloped into a higher fidelity prototype (described in Sect. 3.3).

The goal of our user study was to gain insight into how easily users learned to use ByPass, their reactions to the functions, and assess how using an API can help the user to deal with password-protected websites. We did not include

**Table 1.** Descriptive statistics for task completion time in seconds.

| Tasks | Mean | SD | Median | Minimum | Maximum | Range |
|---|---|---|---|---|---|---|
| New ByPass account | 150.0 | 125.5 | 112.5 | 53.0 | 578.0 | 525.0 |
| New web account | 188.1 | 95.4 | 166.0 | 73.0 | 535.0 | 462.0 |
| Account migration | 60.8 | 25.3 | 50.5 | 33.0 | 128.0 | 95.0 |
| Login via ByPass | 4.4 | 0.8 | 4.2 | 3.2 | 6.7 | 3.5 |
| New account via ByPass | 161.4 | 104.5 | 120.5 | 52.0 | 502.0 | 450.0 |
| Password change | 73.1 | 65.8 | 43.5 | 14.0 | 218.0 | 204.0 |
| Account deletion | 19.6 | 13.1 | 13.0 | 7.0 | 50.0 | 43.0 |

a control condition in this evaluation, because we did not feel that existing managers provided a meaningful comparison to the novel features offered in ByPass. Our goal was to evaluate the usability of ByPass (including the problems that arose). We chose an in-lab study so that we could observe participants' interactions with ByPass, and ask follow-up questions about their experience.

Each study session lasted between 30 min and one h, and was divided into three parts. Participants first completed a pre-test questionnaire asking about existing password habits and demographics. After that, we provided them with a short introduction to password managers and a brief overview of ByPass. They were asked to complete six tasks using ByPass, and then to complete the post-test questionnaire. Participants were paid 15 CAD. The study was approved by our institution's ethics board.

We designed the study to emulate a plausible first-time experience with ByPass. To improve the ecological validity of the study, we created two fictional websites (discussed above) to implement the API and be used for study tasks. These websites were open-source versions of a webmail platform and an e-commerce website, and we gave participants a handout to use in the study with the website URLs and credentials to use in the study. We reminded participants never to use their own passwords in the study.

**Participants.** We recruited 20 participants (12 female) by posting posters around our university campus. Participants ranged in age from 18 to 46 with a median age of 24. 18 participants were students, and 2 university employees. 15 participants (75%) reported having previous experience with a password manager, though only three participants reported that a password manager was their primary means of saving a password.

## 6 Results

We structured our usability evaluation around the ISO 9241 definition of usability [8], and evaluated ByPass using three measures: efficiency, effectiveness, and satisfaction. We recorded task completion times, success rates, and errors via instrumented data collection in the prototype, and measured satisfaction using

Likert scale questions on the post-test questionnaire. We evaluated efficiency using the time spent on each task. For effectiveness, we assessed the number and types of errors that occurred during the study, and for satisfaction, we considered findings from the Likert scale questions. We also recorded observations related to task completion, participants' comments, problems, and recommendations.

### 6.1 Time

Table 1 shows descriptive statistics for the duration of each study task, and Fig. 4 shows the distributions of completion time for all tasks. Times were recorded in the manager logs from the appearance of the first screen to successful completion of the task, and include the time participants spent making errors.

Although there were outliers, the median completion times were less than three minutes for all tasks. Keeping in mind that all participants were completely new to ByPass and were completing all tasks for the very first time, these results are encouraging. Variance was relatively low for most tasks, particularly account migration and login, which we think could form the majority of the users' tasks when setting up the manager for the first time.

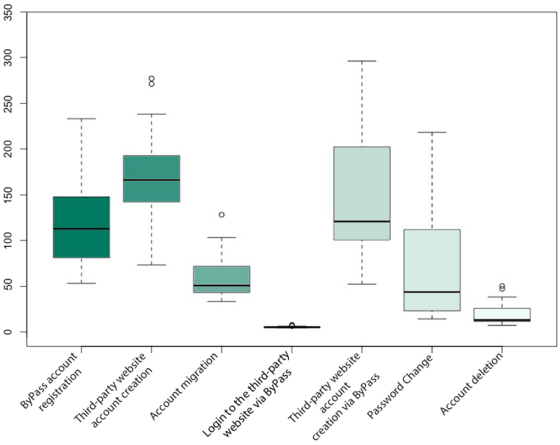


**Fig. 4.** Boxplots showing the distributions of task completion time in seconds.

ByPass account registration involves the user entering their email address, and choosing a master password. To enhance the ecological validity of the evaluation, we included this task in the study and used a strength meter to encourage participants to choose a strong master password. As mentioned above, participants were warned not to use their real passwords. Ten participants had trouble picking a suitable master password, and the median completion time was 113 s. As the ByPass registration process is no different than regular account creation, participants were not asked to recall this password during the study.

The second task had users register directly on one of our external websites, so that they would have an existing account to add to ByPass in a later step. This step also gave us a baseline for the length of time needed to register on a "regular" website. This task had the highest median completion time at 166 s, and a few participants took much longer to complete it.

Later in the study, participants added that account to ByPass, and this process was considerably faster; the median time to add an account (through ByPass) was just 51 s. Following migration, participants were instructed to log into that account through ByPass and participants had no trouble doing this very quickly, in a median of just 4 s.

The process of creating a new website account via ByPass was new to all users, and the median completion time was 121 s. There was a large interquartile spread, and a few participants struggled considerably with this task, but we expect that times might decrease as users grew more familiar with the manager. In any case, an average of two minutes to register on a new website seems to be an acceptable time. According to our logs, 95% of the participants used the password automatically generated by ByPass, which means that our nudges were successful in encouraging users to use a secure password while keeping convenience.

Participants were asked to change the password on one of their accounts on ByPass, and the median completion time was 44 s. 75% of the participants used the password generation function to change their password and choose a new one. The variance for this task was very low, and participants generally did not encounter any problems. One thing that may have inflated times was that participants clicked on the re-generate password button an average of 4.7 times. It was unclear exactly why participants were doing this, but for some, they seemed to want to demonstrate to themselves that the password was actually changing. Others were looking for a password that appealed to them.

The password change page also included a periodic password auto-change feature, allowing the user to pick a period of 30, 60, and 90 days, so the password manager will automatically change the password as the period ends. We did not include this feature in a task, but most participants expressed interest in this feature, and discussed it in relation to the annoyance of regular password changes. Time spent on these conversations may have also artificially increased the time spent on the password change tasks.

The final study task was to delete an account. At this point, participants were relatively familiar with ByPass, and the median completion time for this task was only 13 s.

### 6.2 Errors

We were particularly interested in the kinds of errors that participants made while using ByPass, as dangerous errors in computer security can be difficult to undo and can open the user to serious vulnerabilities [41]. Most participants did not make any errors: the median number of errors per participant was zero,

**Table 2.** Total number of errors committed by usability study task.

| Task | Number of errors |
|---|---|
| ByPass account registration | 34 |
| Third-party website account creation | 4 |
| Third-party website account creation via ByPass | 2 |
| Account migration | 1 |
| Password change | 0 |
| Account deletion | 0 |
| Login to third-party website via ByPass | 0 |

and the maximum was four. We examined errors on two dimensions: the type of error committed, and the incidence of those errors.

Errors were logged by the ByPass software, and then grouped thematically for this analysis. Some errors may have been excluded during this process; e.g., repeated regeneration of a password was not counted as an error, though it is not the intended behaviour.

Table 2 shows the total number of total by task. By far the most error-prone task was the ByPass account registration, incurring a total of 34 errors.

The majority of these errors were password mismatches (i.e., the confirmed password not matching the created one), difficulties in choosing a sufficiently strong master password, and filling in all fields appropriately. The irony that this task had the most errors is not lost on us: we included this step in the evaluation only for the purposes of realism, but we cannot ignore the fact that authenticating to the password manager itself was the most problematic part of the process. The next most error-prone task was the account creation on a third-party website, where participants had a few problems with password mismatches.

Encouragingly, there were few errors in the tasks that actually involved using ByPass. Below is a description of each error type, with the total number of times it occurred in parentheses:

**Password Mismatch on the Registration Page (21).** This error was unique to the ByPass registration page, and it occurred when the entered master password and the confirm master password were not matched. This error was replicated 21 times in the study with a median of 1 time per participant. The maximum number of errors made by one individual was 3.

**Wrong Password (11).** We did not expect participants to perform anything other than the tasks described in the study, but some participants took steps to confirm whether or not the tasks they completed in ByPass were really accomplished on the websites. In doing so, these participants made some incorrect password errors when logging into the websites managed through ByPass.

There were a total of eleven incorrect password entries. Participants particularly wanted to check the password change task by visiting the website and

testing the new and old passwords. Since they used the auto-password generation function for changing the password, they sometimes entered their previous password mistakenly, leading to this error.

**All Fields Required (10).** Although this validation was included on all the pages containing different fields, all of these errors happened on the ByPass registration page, and these errors were caused by our design of the login page. Our extension's window range was not wide enough to accommodate all the fields on the registration page when the user was typing into the master password field. As the user typed, the password strength meter gave more feedback and expanded in size, shifting the password confirmation box below the page break. Although we included a hint encouraging users to scroll down, several participants missed it.

**Choose a Strong Password (4).** For study realism, we included a password strength meter (based on `zxcvbn` [40]) on the ByPass account creation page, and required participants to choose a master password with a "strong" rating. Four users attempted to use a password that did not fill this requirement, and these problems stemmed from not paying attention to the instruction reminding them to pick a strong password, from a lack of understanding of what forms a good password, and from ignoring the feedback from the password strength meter.

**Incorrect Master Password (3).** Following creation of their master password, participants were asked to use it to log into ByPass. Three participants made mistakes in entering their password during login.

**Bad Request (1).** This error category encompasses several different errors: when ByPass sends packets containing user information to websites, it waits for an HTTP response status of 200 (OK) as an approval. Incorrect email addresses, incorrect passwords, and request timeouts belong to this type of error. In our study, this happened only once when a participant mistyped an email address.

**Unsupported Website (1).** Features like creating an account, changing passwords, or deleting an account, can be done only for websites for which we have their API. This error occurs if the user tries to do any of these functions on an unsupported website. One participant had a typo while they were entering the name of the webmail service for the account creation function on ByPass, and because we did not support that name, this error appeared.

### 6.3 Usability Perceptions

We were also interested in how our participants perceived the usability and security of ByPass. We asked participants for their responses to 12 Likert-scale questions, asking about the ease of use, perceived security, and desire to use ByPass in future. Participants were asked to rate their agreement on a 7 point scale where 7 was most positive. Figure 5 shows the distribution of responses for selected questions.

Participants were universally positive about the ease of use of the password change process ($med = 7$) and website login processes ($med = 7$) in ByPass.

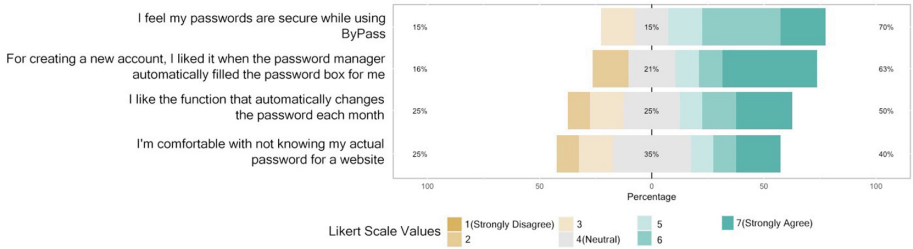


**Fig. 5.** Responses to Likert scale questions asking about participants' interest in ByPass features.

They were also positive about the ease of creating new accounts through ByPass ($med = 7$) and adding existing accounts ($med = 7$), as well as the ease of deleting accounts ($med = 7$). Participants also found it easy to migrate a third-party account to ByPass ($med = 7$).

The median agreement score for the perceived security of ByPass was 6, indicating general satisfaction with the security of ByPass, though participants expressed more frustration with the process of choosing a master password ($med = 5$). In the discussion, some participants mentioned that they would trust ByPass more if it were a software application from a well-recognized organization like Google. A few of the more technical participants commented that they liked the fact that there is no server-side component to the manager.

Participants were most negative about features that related to user control, see Fig. 5. They were not positive about the concept of not knowing the actual password ($med = 4.5$), and displayed mixed responses about the password generation and auto-change features. 31% of the participants wanted to use their own password, instead of relying on a password generator to create one for them.

## 7 Discussion

Although widely recommended as a simple step that users can take towards improving their password security, few users adopt password managers. Adoption problems and the extra work of using a manager are thought to be part of the reason that users do not turn to these tools [3,4]. In this paper, we designed and evaluated ByPass, a password manager that rethinks the users' interactions with the password manager, thereby encouraging adoption and encouraging users to make use of security features. ByPass uses an API for secure communication between the password manager and websites, freeing the user from creating and avoiding errors resulting from copying and pasting. ByPass is built to extend traditional password manager features and supports new functionality such as automated password changes and account deletion. ByPass nudges users toward using secure randomly generated passwords.

The downside of the ByPass approach to password management is that it requires buy-in from websites, who must implement the API. We hope that the

promise of increased security compliance from users might motivate websites to include the ByPass API, and that in turn, this might encourage users to adopt ByPass over other password managers. We acknowledge the uphill nature of this process, while leaving it somewhat out of scope for this paper; we think it is worthwhile to investigate how an architecture such as that of ByPass impacts users even without knowing how uptake might look. We specifically designed ByPass to also be backwards compatible with websites that do not implement the API, and designed the API so that it can be included as a library by website developers. In future work, we plan to further study the feasibility of ByPass's implementation, as well as the implications for web developers, and how they can be supported in implementing ByPass.

We conducted two early evaluations of ByPass's usability, and found that users were able to understand and use the features in ByPass. Most participants did not encounter major problems, though there is undoubtedly still room for improvement in the ByPass user interface. The results of this study will be used to improve the prototype development. However, through the process of designing, implementing and evaluating ByPass, we made a number of observations that affect the design of not only ByPass, but password management tools in general.

### 7.1 An Abstraction Layer for Accounts

ByPass adds a layer of abstraction between the user and the website, where all account-management related interactions take place within the password manager. Adding this abstraction layer brings up various design questions: Where should the password manager sit in the physical space of the web browser? How to instrument the browser extension to "correctly" interrupt interactions with websites? How to train users to go to the password manager first for account management-related tasks? What kind of language should be used to correctly convey password tasks when they are de-situated from their website contexts? Our evaluation suggests that while most participants were able to interpret what was happening, some had great problems.

Our goal was to create a space where security could be the users' primary task, and allow them to focus cleanly and consistently on account management tasks. The constancy of the ByPass interface is intended to allow users a greater sense of control over their passwords and accounts. By using the API to move account interactions into this space, we hoped to create an interface where users knew where to address security concerns, and access the controls to address those concerns. Current password managers hint at this functionality (and include innovative tools, such as security audits) but their placement outside the authentication interaction hampers the functionality they are able to support.

### 7.2 Control vs Automation

ByPass re-architects the password manager to adjust the locus of control for the user. The user is given more control over some aspects of their password management tasks (password changes, account deletion), but less control over passwords themselves. In ByPass, passwords are generated randomly by default, and obscured to the user.

Our contention in ByPass is that the user does not need to know their passwords. With the addition of the API, this is functionally true, but it does not seem to fulfill users' sense of security self-efficacy [28]. In our study, users expressed unhappiness about not knowing their passwords, both in comments and in the post-test questionnaire. We also observed them engaging in "epistemic actions" [23] – actions that serve only to understand a situation rather than to advance a goal, such as repeatedly regenerating passwords, or double-checking on the website that a password had actually changed. Some of these reactions may be due to unfamiliarity, but they echo the findings of previous studies where users have expressed both a desire for control [4,27] and a corresponding sense of responsibility for security [27].

Automating security is often tricky. Solutions such as TLS certificates automate nearly all of the security interaction, turning to the user only when a certificate is not validated, but demonstrate failures when users are unprepared to cope with these situations. Conversely, passwords leave nearly all of the control in the hands of the users, expecting them to exert individual responsibility for all aspects of the password management task. ByPass attempts to find a middle ground for users, removing tasks that needlessly involve users (e.g., reading the password policy, choosing a password that conforms to it). In designing ByPass, we became aware of the myriad corner cases in which the user might need to exert control, and we attempted to leave accessible controls for situations such as assigned passwords, and other unusual contexts.

### 7.3 Testing a Password Manager

In evaluating ByPass, one difficulty we encountered was creating a realistic testing scenario. Ecological validity, or the realism of the study situation, is of the utmost importance in security studies. Tasks that seem easy or manageable as primary tasks (such as remembering a password) are not always manageable when happening in the context of another more important task. Simulating this for a password manager is difficult – the gold standard evaluation would seem to be a field study where participants use the manager for their own accounts. We would want to collect instrumented data from such an evaluation, and privacy could be a significant concern (not only for data collection, but potentially biasing participants' behaviour). For ByPass, a study of this type carried the additional challenge of needing websites to be implemented with the API, further restricting our ability to have users test the manager with their own accounts.

## 8 Conclusion

The usability of password managers is a key issue, since there is no benefit in developing a secure password manager when users cannot make use of it. In this paper, we design and implement ByPass, a password management software offering new features to users by reducing the number of required actions for specific task completion. The key idea of our proposed password manager is that API-enabled secure communication between the password manager and websites allows various password management tasks to be streamlined for the end user. ByPass supports third-party account creation, password change, account deletion directly through the password manager.

We constructed a prototype implementation of ByPass and evaluated it in a user study with 20 participants. The results show that the participants found ByPass easy to use, and our concept is effective both in terms of usability and security. ByPass successfully nudged participants towards using automatically generated passwords, and most of the participants were able to learn how to use ByPass efficiently, while making few errors.

In future work, we plan to integrate the findings of this usability evaluation into the ByPass user interface, integrate new features to support users, and further test ByPass in more ecologically valid scenarios. We also plan to address elements of password manager design that were left out of scope in this early prototype, such as portability to multiple devices.

ByPass raises important questions about where security controls should be placed for end-users. Users desire control, but this may be at odds with good usability. The abstraction of password tasks in password manager creates an extra management step for users, and managers must be carefully designed so that users are supported in understanding this abstraction. However, we think that redesigning the password manager could be key to seeing its wide adoption.

## References

1. CloudFlare - The Web Performance & Security Company. https://www.cloudflare.com/en-ca/
2. NIST Special Publication 800–63b: Digital Identity Guidelines. SP-800-63b Section 5.1.1.2
3. Alkaldi, N., Renaud, K.: Why do people adopt, or reject, smartphone password managers. EuroUSEC'16 (2016)
4. Alkaldi, N., Renaud, K., Mackenzie, L.: Encouraging password manager adoption by meeting adopter self-determination needs. In: Hawaii International Conference on System Sciences, pp. 4824–4833 (2019)
5. Aurigemma, S., Mattson, T., Leonard, L.: So much promise, so little use: what is stopping home end-users from using password manager applications. 50th Hawaii International Conference on System Sciences (2017)
6. Ausloos, J.: The 'right to be forgotten' - worth remembering. Comput. Law Secur. Rev. **28**(2), 143–152 (2012)

7. Barbosa, N.M., Hayes, J., Wang, Y.: UniPass: design and evaluation of a smart device-based password manager for visually impaired users. In: ACM UbiComp (2016)
8. Bevan, N., Carter, J., Harker, S.: ISO 9241-11 Revised: what have we learnt about usability since 1998? In: Kurosu, M. (ed.) HCI 2015. LNCS, vol. 9169, pp. 143–151. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20901-2_13
9. Blanchou, M., Youn, P.: Password managers: exposing passwords everywhere. White Paper, iSEC Partners, pp. 1–6 (2013)
10. Bonneau, J., Herley, C., Van Oorschot, P.C., Stajano, F.: The quest to replace passwords: a framework for comparative evaluation of web authentication schemes. In: IEEE Symposium on Security and Privacy, pp. 553–567. IEEE (2012)
11. Carr, M., Shahandashti, S.F.: Revisiting security vulnerabilities in commercial password managers. In: Hölbl, M., Rannenberg, K., Welzer, T. (eds.) SEC 2020. IAICT, vol. 580, pp. 265–279. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58201-2_18
12. Chatterjee, R., Bonneau, J., Juels, A., Ristenpart, T.: Cracking-resistant password vaults using natural language encoders. In: IEEE Symposium on Security and Privacy. San Jose, CA, USA, May 2015
13. Cheng, H., Zheng, Z., Li, W., Wang, P., Chu, C.H.: Probability model transforming encoders against encoding attacks. In: USENIX Security (2019)
14. Chiasson, S., van Oorschot, P.C., Biddle, R.: A usability study and critique of two password managers. In: USENIX Security, vol. 15 (2006)
15. Doglio, F.: Pro REST API Development with Node. js. Apress, New York (2015)
16. Golla, M., Beuscher, B., Dürmuth, M.: On the security of cracking-resistant password vaults. In: ACM CCS. ACM, Vienna Austria (2016)
17. Habib, H., et al.: It's a scavenger hunt": usability of websites. In: ACM SIGCHI, Opt-Out and Data Deletion Choices (2020)
18. Haekal, M., et al.: Token-based authentication using JSON web token on SIKASIR RESTful web service. In: 2016 International Conference on Informatics and Computing (ICIC), pp. 175–179. IEEE (2016)
19. Herley, C., Van Oorschot, P.: A research agenda acknowledging the persistence of passwords. IEEE Secur. Priv. **10**(1), 28–36 (2011)
20. Inglesant, P.G., Sasse, M.A.: The true cost of unusable password policies: password use in the wild. In: CHI'10, pp. 383–392 (2010)
21. Karole, A., Saxena, N., Christin, N.: A comparative usability evaluation of traditional password managers. In: Rhee, K.H., Nyang, D. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 233–251. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24209-0_16
22. Li, Z., He, W., Akhawe, D., Song, D.: The emperor's new password manager: security analysis of web-based password managers. In: USENIX Security (2014)
23. Liu, Z., Nersessian, N., Stasko, J.: Distributed cognition as a theoretical framework for information visualization. IEEE Trans. Visual. Comput. Graph. **14**(6), 1173–1180 (2008)
24. Lyastani, S.G., Schilling, M., Fahl, S., Backes, M., Bugiel, S.: Better managed than memorized. In: USENIX Security, Studying the Impact of Managers on Password Strength and Reuse (2018)
25. Maclean, R., Ophoff, J.: Determining key factors that lead to the adoption of password managers. In: 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC). IEEE (2018)

26. McCarney, D., Barrera, D., Clark, J., Chiasson, S., Van Oorschot, P.C.: Tapas: design, implementation, and usability evaluation of a password manager. In: ACSAC'12, pp. 89–98 (2012)
27. Pearman, S., Zhang, S.A., Bauer, L., Christin, N., Cranor, L.F.: Why people (don't) use password managers effectively. In: SOUPS'19. USENIX (2019)
28. Rhee, H.S., Kim, C., Ryu, Y.U.: Self-efficacy in information security: its influence on end users' information security practice behavior. Comput. Secur. **28**(8), 816–826 (2009)
29. Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J.C.: Stronger password authentication using browser extensions. In: USENIX Security, p. 15 (2005)
30. Ruoti, S., Seamons, K.: End-to-end passwords. In: NSPW. ACM (2017)
31. Seiler-Hwang, S., et al.: " I don't see why I would ever want to use it" analyzing the usability of popular smartphone password managers. In: ACM CCS'19 (2019)
32. Shirvanian, M., Jareckiy, S., Krawczykz, H., Saxena, N.: SPHINX: a password store that perfectly hides passwords from itself. In: International Conference on Distributed Computing Systems (ICDCS'17). Atlanta, GA, USA, Jun 2017
33. Silver, D., Jana, S., Boneh, D., Chen, E., Jackson, C.: Password managers: attacks and defenses. In: USENIX Security (2014)
34. Smith, T., Ruoti, S., Seamons, K.: Augmenting centralized password management with application-specific passwords. In: SOUPS'17. USENIX (2017)
35. Stebila, D., Sullivan, N.: An analysis of TLS handshake proxying. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 1, pp. 279–286. IEEE (2015)
36. Stobert, E., Biddle, R.: A password manager that doesn't remember passwords. In: NSPW. ACM (2014)
37. Stobert, E., Biddle, R.: The password life cycle. ACM Trans. Priv. Secur. (TOPS) **21**(3), 1–32 (2018)
38. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: an underestimated threat. In: ACM CCS. Vienna Austria (2016)
39. Wharton, C., Bradford, J., Jeffries, R., Franzke, M.: Applying cognitive walkthroughs to more complex user interfaces: experiences, issues, and recommendations. In: ACM SIGCHI (1992)
40. Wheeler, D.L.: `zxcvbn`: low-budget password strength estimation. In: USENIX Security (2016)
41. Whitten, A., Tygar, J.D.: Why johnny can't encrypt: a usability evaluation of PGP 5.0. In: USENIX Security (1999)
42. Yao, F.F., Yin, Y.L.: Design and analysis of password-based key derivation functions. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 245–261. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_17