EVALUATING THE THEORETICAL SECURITY OF VARIOUS OPEN-SOURCE PASSWORD MANAGERS

KEVIN FANG, JASON KUNG, NADIA WAID, AND BRANDON YUE

Abstract. We stress test and evaluate three well known open source password managers for security and ease of use: KeePass, BitWarden, and TeamPass. Specifically, we analyze and report potential vulnerabilities in the programs' password generation, data management, and autofill feature. We also surveyed 24 people on how difficult each of the three password managers were to use as well as what they value in password managers. We found that people generally valued secure storage and easy access to their passwords over other security features.

Contents

1. Introduction

As people start to put more personal information onto the web, they increasingly expect better security and accountability from the applications they use. Open-source password managers have become an appealing solution to both of these issues, as they entail more frequent security updates made by known contributors – they can be checked by anyone for security vulnerabilities and violated privacy policies, making open source password managers ostensibly more trustworthy than closed source alternatives. Three such password managers include KeePass, BitWarden, and TeamPass which each have specific applications. KeePass is used for local storage and generation of passwords, BitWarden is used for local generation and online storage of passwords, and TeamPass has similar functionality to KeePass but is made for organizations rather than individuals. While their intended functionality is different, each of these password managers can be evaluated in much the same way: how random passwords are, if we can tell the difference between passwords from different password managers, how secure the data handling and storage schemes are, if the auto fill feature is secure, and how usable their interfaces are. While the first three features under scrutiny directly pertain to security, we believe the last one to be as, if not more, important than the prior three because increased accessibility will naturally increase security as users of different technical backgrounds secure their passwords. We'll be looking specifically at open source password managers, such as KeePass, TeamPass, and Bitwarden; these applications should not require any special permissions, as they all are/can be self-hosted. We will be analyzing the source code and extracting relevant submodules to test directly. We will also look into relevant papers the publishing team or security analysts have put out within the last five years.

Our paper will mainly consist of analyzing the listed password managers and our attempts at breaking the security of the functions listed previously: password generation, stored data, and autofill.

2. Background

2.1. **KeePass.** KeePass is a free, open-source password manager designed for Windows but with support for Linux-based operating systems through external tools or various extensions of KeePass [8]. First released in November 2003, KeePass has been continuously updated up to the publishing of this paper and has garnered widespread acclaim for having security on par with that of paid alternatives [8]. KeePass does not require the users to directly install to the OS, has auto-type/drag-and-drop accessibility features, and also securely handles the Windows clipboard by clearing it after the clipboard is used for password transfer.
KeePass operates on a single user's desktop with no connection to their browser – it operates solely through their system, unlike password managers like the extension connected to Google Chrome. KeePass can be used by regular users, however we believe its more accessible to developers, or users more familiar with code, as its functionality often requires more effort on the user's part than other password managers.

2.2. **BitWarden.** Bitwarden is a free, open-source password manager that offers a varety of password vault applications, like desktop apps (for both Windows and Linus OS), browser extensions, mobile apps, and more. Bitwarden first released a limited set of these applications in August 2016, and since then has continuously increased their scope. *US News & Report* named it "Best Password Manager" in January 2021[3]. With many of the same features as KeePass (including frequent software updates), we chose to evaluate Bitwarden because of its increased reach and accessibility across more platforms.
Bitwarden is also the only application we're evaluating that offers actual pricing tiers with its cloud storage capabilities – therefore, we are of the opinion that Bitwarden is more targeted towards businesses and consumers.

2.3. **TeamPass.** TeamPass was again selected for being free and open-source, but it crucially differs in that it allows for password management for groups of individuals, differing in scope from the prior two password managers. Each user has defined access rights that allow them to only access appropriate passwords and data[5].

3. Password Generation

Along with storing passwords, KeePass, BitWarden, and TeamPass can also generate passwords for the user. This works by having the user specify certain requirements for generated passwords and the password manager then randomly generates a string with the user-defined restrictions. For example, users can specify a password length and a set of characters to include in the randomly generated string. The strength of the generated passwords is important to many users – around 50% of responses in our password manager usability survey (section 6) listed "Generating strong passwords" as a highly valued feature.

In our analysis, we will define a "strong password" as a password whose characters appear to be drawn from a uniformly random distribution. In other words, the "stronger" a password manager's password generation is, the closer the characters in its generated passwords appear to be drawn uniformly at random.

To analyze the strength of KeePass, BitWarden, and TeamPass's password generation, we generated 100,000 passwords from each program with password lengths of 16 and character sets including alphanumeric and special characters.

3.1. **Password Generation Options.** Each of the 3 open source password managers (as well as other proprietary password managers on the market – Lastpass + 1Password), has the ability to generate a password with certain parameters and character sets. All password managers were able to generate variable length passwords. However, all password managers except for Keepass had limited options in choosing the security component of each generated password. Both Bitwarden and Teampass had only one "special" character option, and the actual character set were only a subset of what Keepass had. Bitwarden also has the ability to specify the minimum amount of characters from each character set (minimum numbers or special characters in the password). All 3 password managers allow for users to remove "ambiguous" characters (such as O and 0, l and 1), however each password manager has its own set of ambiguous characters.

3.2. **RNN Analysis.** We used recurrent neural networks (RNN) to analyze the generated passwords. RNNs can learn underlying patterns in sequences, so we wanted to see if we can use them to learn any patterns between the characters of generated passwords.

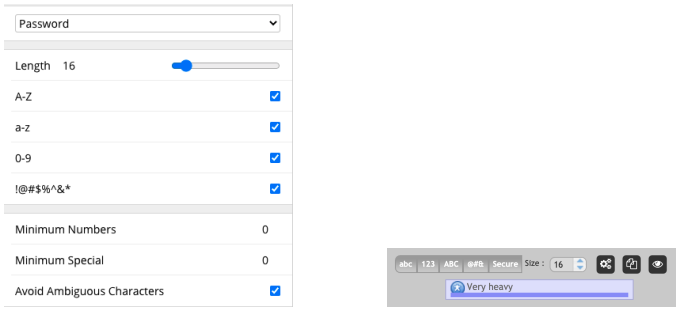

FIGURE 1. BitWarden (left) and TeamPass (right) Password Generation Screens

3.2.1. *Password Manager Classification.* We designed an RNN to see if we can tell which password manager different passwords were generated by. In this task, because we have 3 password managers, we expect the model to achieve a classification accuracy of about 33%, meaning the model would be guessing uniformly at random for each password. If our model can achieve an accuracy significantly higher than 33%, it could imply that there are underlying patterns in the generated passwords that differ between each password manager.

The structure of the RNN was a LSTM model with standard dropout, linear parameters.

```
PasswordManagerClassifier(
  (embedding): Embedding(94, 200)
  (lstm): LSTM(200, 500, num_layers=5, batch_first=True, dropout=0.3, bidirectional=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (lstm2): LSTM(200, 500, num_layers=5, batch_first=True, dropout=0.3, bidirectional=True)
  (dropout2): Dropout(p=0.3, inplace=False)
  (linear): Linear(in_features=1000, out_features=3, bias=True)
)
```

FIGURE 2. Password Manager Classifier Model Structure

Originally our tests concluded that the model had a 97% accuracy in predicting which password manager generated each specific password. However after more careful analysis it was determined that the model essentially learned which characters were unique to each password manager, giving it a high probably chance of correctly predicting the correct password manager. For example, if KeePass is the only password manager that includes ; in its character set, the model would be able to classify passwords that have a ; as coming from KeePass with high probability.

We then obtained another 100,000 passwords from each password manager on only alphanumeric characters to standardize the character sets used by each password manager. The model only achieved a 33% accuracy in the prediction, thus confirming that when all the password managers use the same character sets, it is impossible to distinguish passwords from any two password managers using our RNN. This could imply that with standardized character sets, there are no underlying patterns within generated passwords that differ between the KeePass, BitWarden, and TeamPass.

However, due to the increase in the special character requirements, password generators are less likely to use the same character sets because they offer different sets of special characters. This could be a potential security vulnerability where, if an adversary was able to obtain plaintext email/passwords combinations, they would be able to learn which

password manager (if at all) users were using. The adversary could then create a focused channel of attack, either at the specific manager or by prioritizing which users to go after should another vulnerability in a specific password manager be revealed.

3.2.2. *Password Character Prediction.* We also used RNNs to see if we could predict characters that are generated by each password manager. The model would take in a sequence of characters that were generated by the password manager and predict the next character that would be generated. In this task, we expect the model to achieve a prediction accuracy of $\frac{1}{|charset|}$ for each password manager. This would mean that at best, the model would be predicting characters uniformly at random over the entire set of possible characters. If the model can achieve an accuracy significantly above this for a specific password manager, this could imply that there is some underlying pattern between the characters of a password generated by the password manager. For example, if the model can achieve above expected prediction accuracy for KeePass passwords, this could imply that KeePass's password generation has some pattern that can be learned to allow an adversary to guess passwords better than guessing at uniformly random.

The structure of this RNN was a bidirectional LSTM with dropout and linear layers.

```
PasswordPredictor(
  (embedding): Embedding(94, 500)
  (lstm): LSTM(500, 500, num_layers=5, batch_first=True, dropout=0.2, bidirectional=False)
  (dropout): Dropout(p=0.2)
  (linear): Linear(in_features=500, out_features=94)
)
```

FIGURE 3. Password Predictor Model Structure

Running the prediction model on passwords from each password manager, we observed the results in the tables 1 and 2.

TABLE 1. Character Prediction Accuracies (Non-Standardized Charsets)

	KeePass	BitWarden	TeamPass
Expected Accuracy	0.011	0.014	0.012
Test Accuracy	0.011	0.018	0.013

TABLE 2. Character Prediction Accuracies (Standardized Charsets)

	KeePass	BitWarden	TeamPass
Expected Accuracy	0.016	0.016	0.016
Test Accuracy	0.016	0.019	0.017

We can see that when we run the prediction model on KeePass passwords with the non-standardized (alphanumeric and special characters) and standardized (alphanumeric characters only) character sets, our test accuracy is the same as our expected accuracy. For TeamPass passwords, we notice that our test accuracy is greater than expected by 0.001 in

both cases, but this difference is relatively small and may not be significant enough to indicate any underlying patterns in the generated passwords. However, for BitWarden, we notice that our test accuracy is greater than expected by 0.004 and 0.003 for non-standardized and standardized character sets respectively. This difference is relatively large, and may indicate that BitWarden password generation does not necessarily choose characters in a password from uniformly random.

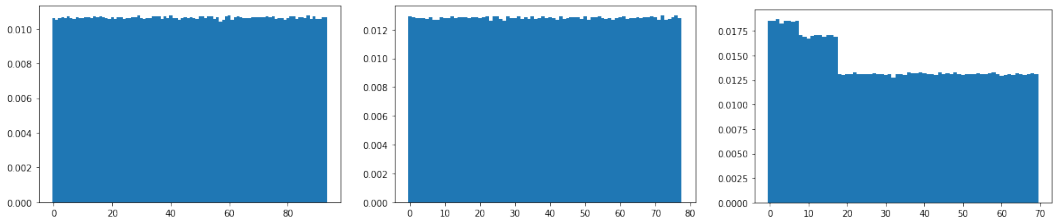

FIGURE 4. Character frequencies of KeePass (left), TeamPass (middle), and BitWarden (right). In each graph, characters from left to right are in the order of special, numbers, and letters.

We can get further insight on our observed prediction accuracies by looking at the frequencies at which each password manager used each character throughout all 100,000 generated passwords (fig. 4). KeePass and TeamPass both have a relatively flat distribution, meaning throughout all the generated passwords, each character was chosen with approximately the same probability (uniform). However, BitWarden's distribution has 3 sets of frequencies, with special characters being chosen with the greatest probability, numbers being chosen with the second greatest probability, and letters being chosen with the lowest probability. This could explain our prediction accuracies where BitWarden was the only password manager that could achieve an accuracy significantly higher than expected – when we look at a large set of passwords, BitWarden does not generate characters uniformly at random so an adversary could guess characters in the generated passwords with greater probability than uniform.

After analyzing the source code within Bitwarden, we found that even though users can require all passwords to not have a minimum requirement of symbols and numbers, if the user selects the 'symbols' or 'numbers' character set, the minimum defaults to 1. This ensures that for any password Bitwarden generates that at least one number and symbol is present in every password. This is not true for Teampass nor Keepass and explains the reasoning non-uniform character frequencies because there are fewer special characters than numbers and fewer numbers than letters.

3.3. **Randomness Tests.** In addition to our RNN analyses, we perform additional tests of randomness on the generated passwords.

3.3.1. χ^2-*Test for Randomness.* We perform χ^2-tests on each generated password from each password manager. Specifically, we test if the character distributions of a single password is uniform. Because our password length is 16, we can say that each password has 16 possible characters, so our test uses 15 degrees of freedom. At $\alpha = 0.05$, our critical value is 7.261, meaning we reject our null hypothesis when our χ^2 test statistic is greater than 7.261.

Our null hypothesis in this case is that the character c_i at index i has a $\frac{1}{16}$ probability of being the specific character k (uniform over the possible characters in the password). We

TABLE 3. χ^2-Test Null and Alternative Hypothesis

$$H_0: \quad P(c_i = k) = \frac{1}{16}$$
$$H_a: \quad P(c_i = k) \neq \frac{1}{16}$$

expect most passwords to fail to reject H_0, meaning the χ^2 statistic is less than our critical value 7.261. When we run this test, we get the results in table 4.

TABLE 4. χ^2-Test Results

	KeePass	BitWarden	TeamPass
Proportion Failed to Reject H_0	1	1	1
Maximum Test Statistic	1.5	1.625	1.625

From our test, we can see that when we run χ^2 test on each individual password, KeePass, BitWarden, and TeamPass all seem to draw characters from a uniform distribution because every password failed to reject H_0. This suggests that all three password managers have strong password generation when we only look at individual passwords whereas when we looked at 100,000 passwords before, BitWarden's password generation appeared to be weaker. It may be important to note that because we run the tests on each individual password, the sample size for each test is relatively small.

3.3.2. *zxcvbn Algorithm.* We also tested each individual password using *zxcvbn* [10]. This is an algorithm that uses pattern matching and conservative estimates to evaluate the security of passwords. Furthermore, *zxcvbn* recognizes and weighs 30,000 common passwords, common names according to US census data, popular words from Wikipedia and US television, repeated patterns and sequences, and other common trends in passwords in its evaluation [10]. We will be looking at *zxcvbn*'s score metric, which is an integer 0-4 that indicates password strength (4 being the strongest, requiring $\geq 10^{10}$ guesses), and its estimate of the number of attempts needed to guess the password (in $\log_{10}$).

TABLE 5. *zxcvbn*-Analysis Results

	KeePass	BitWarden	TeamPass
Mean *zxcvbn*-Score	4.0	4.0	4.0
Mean Required Guesses (in $\log_{10}$)	15.982	15.971	15.983

From our results of running *zxcvbn* on each password (table 4), we can see that the passwords generated by KeePass, BitWarden, and TeamPass are all strong, requiring on the order of 10^{15} guesses on average. This means if an adversary can guess at a rate of 1000 guesses per second, it would take more 31688 years on average to guess the password. This supports the findings from our χ^2-tests, suggesting that the individual passwords generated by all three password managers are still strong. Even so, we can see that BitWarden passwords are estimated to require fewer guesses than KeePass and TeamPass, requiring on the order of 10^{14} fewer guesses on average. This aligns with the previous analyses that suggested BitWarden's password generation is relatively weaker than those in KeePass and TeamPass.

3.4. **Recommendation.** Although the character sets of the passwords generated by Bitwarden, Teampass and Keepass are different, we don't believe that there is a significant security threat posed by knowing the password manager of the generated password. We believe that all three produce passwords that are secure and inherently random. Even though Bitwarden has a higher probability rate of guessing the next character, the percent increase only marginally reduces the passwords necessary to check for longer passwords.

4. Password and Information Storage

4.1. **Introduction.** If the account data isn't stored securely, then the password manager could open the client to more security vulnerabilities. The database becoming compromised at any point would give the attacker enough details to infiltrate the client's accounts on various websites. Some password managers even migrate information to a different server, meaning that a poor communication scheme could also give an adversary attacking through the network direct access to sensitive information.
Therefore, we chose to evaluate the data storage scheme and not necessarily the specific encryption schemes used in storing the data, as BitWarden, TeamPass, and KeePass all utilize reliable and provably secure one-way encryption schemes. For the password vaults that use non local storage, namely TeamPass and BitWarden, we also evaluated the communication protocols between the browser and the server in which data was stored. To do this, we read their documentation and assessed the code for further information and confirmation.

4.2. **KeePass.** We analyze KeePass's password and information storage by evaluating their methods of handling data within the program and identifying any potential vulnerabilities. KeePass uses a combination of the Advanced Encryption Standard as well as ChaCha20 to encrypt entire password databases (passwords, usernames, notes, etc.) [8]. SHA-256 (a secure one-way hash function) is then used to hash the master key components and the output is run through a key derivation function to generate a secret key, making the use of pre-computed and guessing attacks much more difficult.

KeePass only manages sensitive data encryptedly in process memory, including master keys and passwords – it does not encrypt user names, notes, file attachments, etc. This means that if the process memory of KeePass is dumped to disk, all the master keys and passwords would be safe, but other information would be leaked in plaintext. KeePass deems this is acceptable because it assumes that non-sensitive data like names and notes cannot be used to compromise the credentials of the user. However, we believe that this may not necessarily be the case. For example, if a user misuses the notes section of KeePass's entries, they might put information like recovery codes or password hints in plaintext. Because this is not encrypted in process memory, an adversary could then read the notes of a password entry and learn something to aid in guessing the user's password. We believe that other similar attacks exploiting user error could be done using these plaintext fields.

There are also some operations for which KeePass must make sensitive data available unencryptedly in process memory, such as showing an unhidden password to the user in the UI. Furthermore, KeePass's storage and encryption heavily relies on Windows and .NET, using features like Windows Data Protection API. Because of this, Windows and .NET could potentially make copies of data in the process memory that cannot be erased by KeePass. This means that if a user runs on a machine where Windows or .NET are compromised, an adversary could read and copy the unencrypted sensitive data, allowing them to learn the master keys and passwords of the user. Under KeePass's assumptions, this is not a concern

because KeePass is run in a secure environment, however, user errors could lead to this attack being feasible in practice.

4.3. **BitWarden.** The BitWarden Google Chrome Extension stores account information both on the Microsoft Azure cloud and on Google Chrome storage. It utilizes AES-CBC 256-bit encryption for the stored data and PBKDF2 SHA-256 to get the encryption key. With these encryption schemes, BitWarden acts as a zero-knowledge solution, only allowing decryption through the user's master password.

To analyze the security of their data storage, we did a code analysis to evaluate what information BitWarden encrypted and how information was sent to the cloud. We were unable to of course evaluate the Cloud itself, although BitWarden claims that, once its servers receive your data, the server again cryptographically salts and hashes the value to its servers.

We found that BitWarden's data storage scheme encrypts each field, including the website, username, password, and any other important information, before sending it to cloud storage. In cloud storage, this encryption is unable to be decrypted without the user's masterkey. The only data stored on Google Chrome's local storage is the master key, which is also encrypted before storage, meaning that the data can be decrypted only when its been sent back to the user's browser on a fetch. Assuming that the encryption schemes are implemented correctly, BitWarden's data storage security seems be sound and to follow the security protocols they describe.

4.4. **TeamPass.** TeamPass stores all information on a server that users can access via login, utilizing the encryption scheme AES-256-CTR[5].

After analyzing the code, we found that, when migrating the information to the central server, TeamPass makes an API call that includes a URL that contains all information, including the password. None of this information is encrypted before being sent the server, meaning that even the password is stored in plaintext in the URL. After the server receives the password, it encrypts and hashes the password, and any customized fields, before storing the information on a MySQL database.

TeamPass's communication protocol with its server is vulnerable to sniffing. If an adversary were to intercept a message between the user and the server, it would have direct, non-encrypted access to all important information, including login credentials and the respective website URL. This means they would direct access to the client's account, and considering that TeamPass is used to share account information between users, this would impact all users associated with that account.

TeamPass's data storage protocol relies on the assumption that the other information does not contain critical information and wouldn't be enough to compromise the user's account. In order to encrypt these other fields, the user would have to create custom fields. However, these fields could contain important personal information that could leave a user vulnerable in the case that this information becomes accessible to an adversary. We would like to note that the extra hashing on the password provides authenticity to the encryption in the case of data corruption.

4.5. **Comparison of Password Storage Schemes.** When comparing the encryption schemes, we found that BitWarden is the most secure in terms of its data storage scheme. BitWarden is the only cloud-based password vault that we analyzed that provides full end-to-end encryption. Furthermore, BitWarden's decryption is only accessible via the user's local machine.

Assuming that the encryption schemes are implemented correctly, BitWarden provides security from start to finish in the password management process.
Unlike BitWarden, both KeePass and TeamPass don't automatically encrypt all information that it stores, meaning that some fields are accessible in the case that the database or process memory is leaked to an attacker. Furthermore, TeamPass has a seemingly insecure communication protocol between the client and the server, leaving it vulnerable to various network attacks.

5. AutoFill

5.1. **Introduction.** Autofill is a function that makes the user's life convenient and is becoming increasingly used by the everyday user. It prevents unnecessary repetitiveness in filling out user information and allows users to adopt more complex passwords, as the user no longer need to remember passwords themselves.
To analyze the autofill features available in these password vaults, we performed a code analysis and some basic tests, looking specifically for autofill phishing attacks. By "autofill phishing attacks," we mean those that edit the HTML to collect information without the user's knowledge nor consent. An example would be an input field for the user's address with a text box size of zero – in this case, the user would be handing over their personal information with no knowledge of such, as it wouldn't be evident from the UI. We analysed KeePass and Bitwarden for this functionality, but not TeamPass, as TeamPass does not provide this functionality directly. TeamPass does provide autofill capabilities through various extensions. However, considering that extensions are often developed by developers outside the main product, we chose to prioritize staying strictly within our chosen applications.

5.2. **KeePass Autofill Analysis.** While KeePass does not have an actual autofill function, it has several functions which can approximate autofill. Users are able to use copy paste, drag drop, and auto-type to more conveniently fill in their passwords. KeePass' copy paste function uses the system clipboard, but for security purposes the clipboard gets cleared after approximately 12 seconds. Drag drop uses the Windows' drag and drop to transfer passwords. KeePass' autotype uses a keyboard macro to input the user's username and password.
We found KeePass's autofill approximations to be secure against most attacks, mostly because of KeePass' assumption that the user's environment is free from malware/spyware - this shifts the onus of auto-type attacks from KeePass to the user. As a result, users of KeePass are highly vulnerable to social engineering attacks by virtue of how KeePass handles autofill: it is entirely on the user how and where to enter in their sensitive information, as KeePass doesn't check if where the user is inputting the login credentials matches the respective website; KeePass merely provides the tools required for users to do so in a secure way. Since KeePass contends that "neither KeePass nor any other password manager can magically run securely in a spyware-infected, insecure environment", this makes KeePass' autofill feature secure based on its assumptions.
Because of KeePass's use of the system to provide autofill functionality, it is secure against autofill phishing attacks, meaning that it won't fill information that the user does not intend to.

5.3. **Bitwarden Autofill.** BitWarden allows users to autofill account credentials. In our code analysis, we found that BitWarden generally prevents most autofill phishing attacks that hide fields. BitWarden checks to see if the field is not hidden and greater than 10px in

width and height. Therefore, Bitwarden is secure against autofill phishing attacks that deal with setting fields to be hidden or of extremely small size.
However, we found that BitWarden does not check for fields with zero opacity or fields that are covered by other other objects. This means that BitWarden isn't secure against all phishing attacks.
After assessing the code to find this information, we also verified this by utilizing the HTML of Viljami Kuosmanen's proof-of-concept autofill phishing attack [1]. We verified that BitWarden prevented the phishing attacks described, and after editing the HTML, we confirmed that BitWarden was vulnerable to other kinds of phishing attacks.
Like KeePass, Bitwarden also allows users to copy and paste passwords in plain-text to input into whatever field they desire. This function also uses the clipboard, and although Bitwarden offers the capabilities to automatically clear the clipboard after a user-defined number of sections, we noticed that its default setting is "Never". Without the user explicitly changing that setting, the clipboard keeps the password for an undefined period of time, meaning that Bitwarden's default copy and paste function is susceptible to user error targeted attacks – the user could very possibly paste the password accidentally into unintended places, and an adversary could exploit these human errors.

5.4. **Comparison of Autofill Functions.** KeePass appears to have the most secure autofill functionality, although it's also the more involved of the two, requiring the user to use hotkeys to start the auto-type process or actions of drag and drop or copy and paste. KeePass only operates through the local machine, meaning that it has no access to the browser except through the user's keyboards.
BitWarden has a more commonly known autofill function, meaning that its open to autofill phishing attacks. Currently, it isn't secure against all phishing attacks, although preventing these kinds of attacks would be very possible. However, regardless BitWarden interacts with the website directly, meaning that its vulnerable to unknown autofill attacks and future autofill attacks, unlike KeePass.

6. Password Manager Usability

We also began preliminary analysis of the password manager's usability. To analyze usability, we asked volunteers to complete a survey that contained some questions regarding general motivations behind using password managers, as well as specific questions regarding our study's password managers. In regards to the password managers we're using, we asked users to complete set-up, generate a password, and fill in credentials feature. After completing these actions, we gauged them for usability on a scale of 1 to 10 where, in each case, 1 was "Easy" and 10 was "Difficult."

We interviewed 24 people on the usability of the three password managers, and found the results to be rather disheartening: in general many of the respondents tended to dislike using a password manager, finding it difficult to both set up and store passwords. Coupled with the fact that many of the respondents weren't already using a password manager and don't plan to after the survey, open-source password managers still have quite a ways to go to make their product accessible beyond simply making the password manager free.

[1] found at https://anttiviljami.github.io/browser-autofill-phishing/

6.1. **General Survey Results.** When looking for a good password manager to use, 100% of respondents looked for the ability to sync passwords between devices and over 90% of respondents wanted a secure storage of passwords. It appears that the frustration of not being able to log on to specific websites at will is the main priority of users when looking for a password manager. Users did not seem to find the ability to auto-fill as important, nor the aesthetic of the password manager.

Almost half of the respondents were already using a password manager, and some of the concerns raised for not using a password manager is the added work needed to switch to a new method of storing and generating passwords.

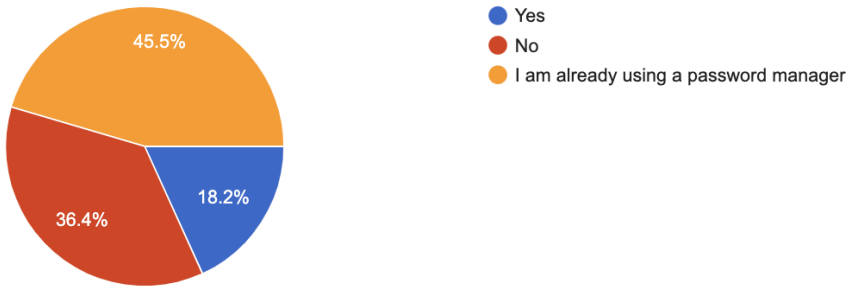

FIGURE 5. Current password usage: Breakdown of responses to the question "If you didn't use a password manager before, would you use one now?"

6.2. **Usability of Bitwarden.** Participants that evaluated Bitwarden found the UI to be fairly easy to navigate, as Bitwarden had the lowest scores to all survey questions. Specifically, Bitwarden has the lowest scores in the questions about difficulty of set up (1.75 out of 10), password storage (2 out of 10), the filling-in of credentials (2.25 out of 10), and navigation (2.5 out of 10). In regards to setup, one user noted that the "download process was very quick" and that Bitwarden's video guide was "helpful for setup." For password generation, the respective UI was described as "straightforward" and "smooth." However, some noted that some of the extra features, like setting up authenticator keys, were confusing. Navigation and autofill were generally described as "streamlined, " "manageable," and "self-explanatory."

Overall, Bitwarden appeared to design with the user in mind at every step. Although the usage of extra features and meaning of some icons were unclear, the features we were evaluating were generally easy to navigate.

6.3. **Usability of KeePass.** KeePass scored relatively well in it's set up process, scoring a 3.33, however its general navigation appeared to frustrate users(6.66). The set up was described as "simple", vastly contrasting the responses to the navigation. Users found themselves having to navigate through "trial and error" and described KeePass's UI as "outdated" and "non-intuitive."

In the other two questions about credential fill-in and password storage, users found themselves in the middle about it's usability, both being scored at a 4.66.

6.4. **Usability of TeamPass.** Our study found that users struggled the most with using TeamPass, scoring the worst on almost all questions. Users particularly struggled with password storage (8.2) and general navigation (7.8). When storing a new password, one user received an ambiguous error that read "Hacking Attempt" and the respective UI was generally described as "confusing." Users were especially frustrated by the significant amount unlabelled buttons and unclear actions.
Users seemed to find set up and filling-in credentials relatively easier than the other two actions, scoring them as 4 and 4.6 respectively. It is worth noting that TeamPass's main set up involves setting up a server, which we didn't ask our users to attempt – instead, we set up the server prior to the study. However, logging in credentials was generally described as "easy" even though TeamPass does not provide an autofill feature.

6.5. **Password Manager Recommendations.** Although this study is preliminary, we believe that users enjoyed using Bitwarden the most due to its ease of use and simple design. A common theme of Keepass and Teampass was confusion in both the language and the location of certain buttons to accomplish certain tasks.
While the functionality for all three password managers are there, KeePass and Teampass would require a longer learning period in order to become accustomed to the program.

7. Related Work

Password managers are both highly valued and scrutinized in the technical world: a web search for 'Password Manager Security Analysis' brings up several papers of interest. While there has been a lot of research and testing done on these password managers, all three password managers are under active development which could introduce new vulnerabilities to the codebase.

There have been previous works analyzing the password generation of other password managers. In [7], they found that the password generation of RoboForm did not choose characters uniformly at random, selecting "Z", "z" and "9" less frequently than other characters. This means that an adversary could guess RoboForm's generated passwords with higher accuracy than guessing at uniformly random. This is similar to our findings with BitWarden passwords, suggesting that this weakness in password generation may not be unique to BitWarden.

There have also been previous works that found vulnerabilities in other password managers' autofill feature. For example, in [7], they found that Firefox's autofill feature was vulnerable to harvesting attacks. This means if a user visits a website compromised by an adversary, the adversary could insert `iframes` that trick Firefox's autofill feature to fill out the user's credentials, all without the knowledge of the user.

While the security analysis of password managers is by no means a novel idea, we have not seen a paper tuned towards the analysis of password managers from a user-sided perspective. For example, in the paper *Password Managers: Attacks and Defenses*, different attacks on a broad range of password managers are discussed and analyzed (including an autofill analysis), but the scope of said password managers is undefined.

There's also been much interest in security of all the major web browsers currently available to consumers. Recently, there were a few major vulnerabilities discovered by researchers in the chrome browser (with one exploit specific to autofill) that allowed the attacker to arbitrary execute code.

Multiple security companies have also audited these password managers, usually with recommendations on improvements needed to ensure the security of the managers. In a 2018 report [2], a German security company Cure 53 found multiple vulnerabilities in login urls and also their autofill code. There are also audits on keepass which produced no significant security vulnerabilities as well [1].

Even in the history of 6.857, past student project have focused on the security of password managers. One analyzed the passwoord managers connected to Google Chrome, FireFox, Internet Explorer, and Microsoft Edge and external password managers, like Roboform, 1Password, PasswordSafe, and LastPass [9]. They found that most of the listed password managers were vulnerable, often providing the plaintext of the url or the hash of the url directly to the hacker. However, they recognize that this only becomes an issure in the instance that the user's local machine is compromised. Another student research project looked into the security of autofill in google Chrome, FireFox, and LastPass[6]. They noted some successful breaches in the three applications, specifically noting that they were able to use a module to decrypt the ciphertext of the file containing the passwords on Google Chrome and similarly able to decrypt the master key and login data of FireFox.

Finally there have been many usability studies as well on how to best create a system where users would be comfortable in using a password manager [4].

8. Conclusion

We are mostly confident in the security of Bitwarden, Teampass, and Keepass. However, TeamPass has clear vulnerabilities in their communication protocol between the client and the server, and Bitwarden isn't entriely secure against all phishing attacks. Regarding usability, open-source applications still have a ways to go, but Bitwarden appears the most promising and user friendly. It is worth noting that TeamPass is coming out with a new version, which may introduce changes that could greatly improve their functionality.

In terms of active development, all three projects boast an active community and continue to have updates to the existing codebase. At the end of the day, what is important is that users don't use the same password for all their websites.

9. Acknowledgments

We'd like thank the 6.857 staff for all the hard work they put into making this semester fun and interesting. We'd especially like to thank Andres our favorite 6.857 project advisor for helping provide great advice.

References

[1] Brinkmann, M. Keepass audit: no critical security vulnerabilities found - ghacks tech news, Nov 2016.
[2] Brinkmann, M. Results of bitwarden security audit published - ghacks tech news, Nov 2018.
[3] Colby, C. The best password managers of 2021 and how to use them, Apr 2021.
[4] Karole, A., Saxena, N., and Christin, N. A comparative usability evaluation of traditional password managers. In *Proceedings of the 13th International Conference on Information Security and Cryptology* (Berlin, Heidelberg, 2010), ICISC'10, Springer-Verlag, p. 233–251.
[5] Laumaillé, N. Teampass documentation, 2021.
[6] Lin, S., Baral, A., Vadari, M., and Maccow, S. *Security Analysis of Browser Auto-fill and Password Managers*. PhD thesis, 2020.
[7] Oesch, S., and Ruoti, S. That was then, this is now: A security evaluation of password generation, storage, and autofill in thirteen password managers. *CoRR abs/1908.03296* (2019).
[8] Reichl, D. Keepass security.

[9] Weckwerth, N., Xia, B., and Zhang, J. *Password Manager Security*. PhD thesis, 2020.
[10] Wheeler, D. L. zxcvbn: Low-budget password strength estimation. In *25th USENIX Security Symposium (USENIX Security 16)* (Austin, TX, Aug. 2016), USENIX Association, pp. 157–173.

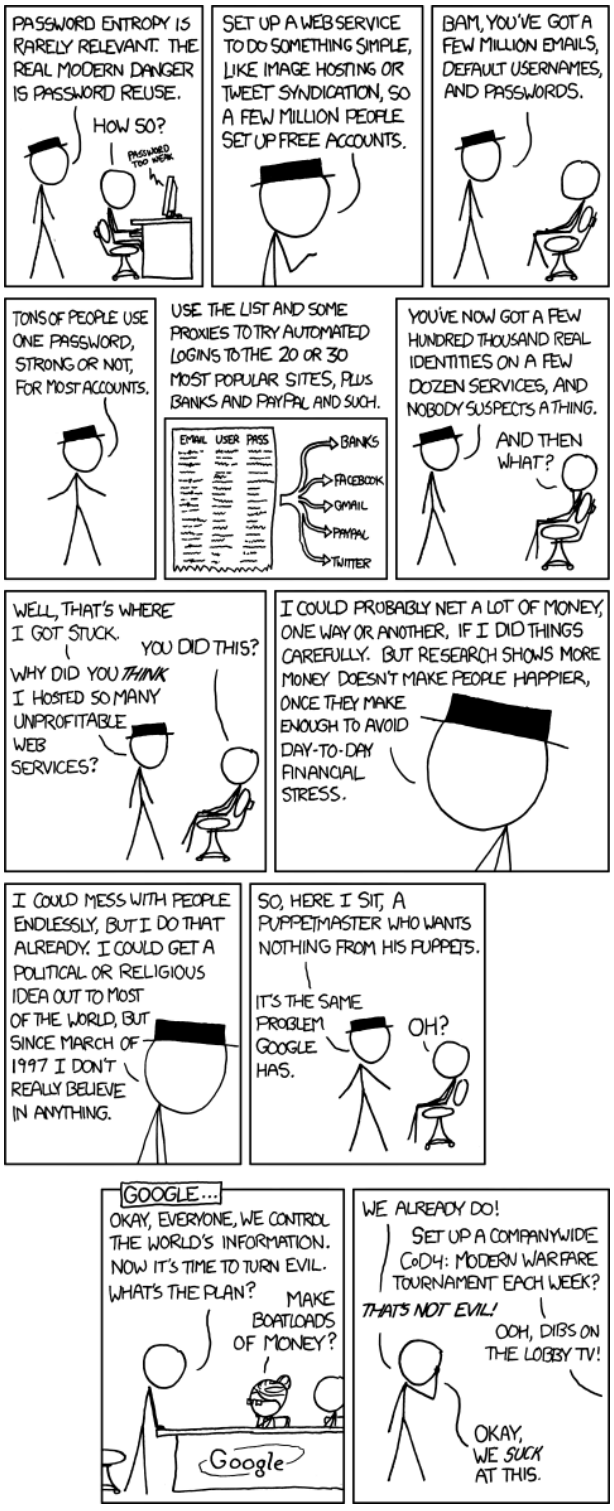

FIGURE 6. XKCD 792