Security Analysis of Modern Password Managers

Student Name: Jia Xiu Sai
Supervisor Name: Ehsan Toreini
Submitted as part of the degree of BSc Computer Science to the
Board of Examiners in the Department of Computer Sciences, Durham University

Abstract—The rapid growth of today's internet usage makes online security more important than ever. With passwords being the primary authentication method on the internet, they need to be resilient against malicious attacks to protect the valuable assets of internet users. Password managers are advertised as a solution for users looking for a one-stop solution for their password needs to improve their online security. Security analysis looks at the current password managers on the market to find any security vulnerabilities that can be exploited to compromise its users. The objective of this paper is to improve password managers by analysing their state of security and providing suggestions based on industry standards. A list of tests was devised from prior academic and non-academic sources on the topic to evaluate the password managers on their three main features; password generation, password storage, and password autofill. Analysis and comparison are made based on the results from the tests on seven popular password managers. The results reveal that improvements have been made since previous security analyses but there still exist some security flaws in today's password managers. The security flaws include the lack of settings for common use cases, insecure design choices, and vulnerable autofill policies. Based on our results, we identified the current industry standard of password managers, provided recommendations to improve password managers, reported crucial vulnerabilities to the developers and demonstrated that password managers improve users' online security.

Index Terms—Authentication, Hardware/Software Protection, Security, Verification

1 Introduction

Passwords have been the primary authentication method on the Internet, and this trend will continue into the foreseeable future. Passwords come with well-known security and usability drawbacks, often being the weak link in online security. In response to the significant advancements in attackers' capabilities to compromise passwords, password policies have gradually become increasingly complex. Complex password policies require longer and more random passwords, giving rise to a new problem: the human memory is only capable of remembering sequences of around seven items, plus or minus two [1]. With the average user actively using 16 to 26 password-authenticated accounts [2], creating and remembering random and secure passwords for each account proves to be a difficult task. Users often cave into insecure practices for convenience by using weaker but more memorable passwords or reusing a password across different accounts [3]. Insecure practices lead to more catastrophic effects if one of the user's accounts is compromised as the attacker can access other accounts sharing the password.

Security experts, including The National Cyber Security Centre in the UK [4], recommend using password managers to generate and store random and secure passwords. Security vulnerabilities in these password managers can make them a single point of failure, where if attackers gain access to the passwords, the user will have all their passwords compromised. Password managers have to keep a high level of security to protect sensitive information by implementing strict measures to prevent the user from misusing the features of the password manager, which is a trade-off between security and usability.

Are today's popular password managers secure enough to benefit their users? This project aims to answer this question through three major objectives. The first objective is to carry out a comprehensive analysis and comparison of different features and design choices in at least five popular password managers on the market. The analysis and comparison focus on three key features of password managers: password generation, password storage, and password autofill. The purposes of this objective include uncovering the current industry standard, highlighting password managers that lag behind these standards, uncovering the changes made after previous security analyses, and providing a snapshot of the current industry standard for future security analyses. The second objective is to carry out a forensic analysis of the current implementations by using a set of tests curated from published papers and incorporating novel methods that have not been used by previous security analyses on password managers. The third objective is to discover a novel security vulnerability and make a report to the developers of the password managers with the suggestion of possible mitigation or solutions.

Our contributions include:

1) This paper is the first security analysis focusing on the evaluation of default settings used in the three key features of password managers; password generation, password storage, and password auto-

fill. For example, the default settings of password generation refer to the default password compositions set by the developers. Developers need to be cautious when setting these default settings to make sure that it fits the use cases of most users while maintaining a high level of security. Although password compositions are modifiable in all extension-based password managers, this is not the case in the browser-based password managers evaluated by us. Unmodifiable password compositions require the users to manually edit the password generated if they wish to have a different password composition or length.

2) For our analysis of password generation, we generated a password corpus using the default password compositions for each password manager, creating a total of 720,000 unique passwords. We evaluated the password corpus using methods from Oesch and Ruoti's work [5] (Shannon entropy, χ^2 test and zxcvbn) and put together a novel evaluation framework incorporating deep machine learning password cracking, PassGAN. Using these methods, we evaluate the password corpus we collected to highlight non-random patterns in the generated passwords and estimate their resilience against password cracking attacks.
3) The most recent security analysis of password managers is dated two years old and others are more than seven years old. Our work found that minor improvements have been made compared to two years ago and there still exist some security flaws in today's password managers. For example, Dashlane and Mozilla Firefox automatically autofill the login field on insecure websites. We responsibly disclosed the security vulnerability in two password managers, Dashlane and Mozilla Firefox. This security vulnerability has been promptly acknowledged by the developers of Dashlane and Mozilla Firefox. Our analysis also found insecure design choices made by browser-based password managers, namely in the password generation compositions and key derivation function for password vault.

1.1 Password Managers

Password managers are applications that generate and store login credentials, usernames/emails, and passwords to all the different accounts a user may need. Password managers help users reduce the cognitive burden of remembering long and complicated passwords, promote the use of more secure passwords, and prevent password reuse. Password managers come with basic features that help generate, store, and autofill passwords. Advanced features are available behind a paywall in some password managers and extend the basic functionalities of password managers, often with more niche use cases.

1.1.1 Basic features

Password Generation: When the user visits a login page on a website, the password manager detects the login fields and prompts the user to allow the password manager to create a new password or autofill the stored login credentials. Password managers generate new passwords based on their password composition, the requirements/combinations of numbers, symbols, uppercase, and lowercase alphabets, with some password managers allowing the user to choose which classes to include/exclude. Although the length and complexity of these passwords depend on the password managers, the default lengths of the passwords generated by the popular password managers can be from 12 to 20 characters. Some password managers have more options for this feature, allowing the user to choose the number of characters for the password generated, remove similar-looking characters to allow for more readable passwords, etc.

Password Storage: Once the password manager has created a new password for a new account, the user's device encrypts the login credentials before the password manager stores them. If the password vault is compromised, the encryption of the password ensures that attackers will not gain access to the login credentials without decoding the passwords, which is a challenging task without the master password. Depending on the password manager, the password vault can be an encrypted vault on the user's device or the password manager's cloud servers. The master password is the only key that can access the password vault and decode its contents.

Password Autofill: When the user visits a login page, the password manager detects the login fields and starts the autofill process if it has login credentials stored. The user's device fetches the encrypted login credential from the password vault and decrypts it before the password manager autofills the login fields. There are two types of autofill:

1) **Automatic autofill** removes the need for the user to choose or confirm the password manager to fill in the login forms. Autofill policies can overwrite and disable this feature due to several factors including, the protocol (HTTP/HTTPS), HTML login input attribute (`autocomplete="off"`), login form action used on the current page relative to the protocol, etc [6]. Some password managers take this feature further by submitting the login form.
2) **Manual autofill** requires the user to press a prompt for the password manager to fill in the login forms. Often due to security concerns or the user has more than one login credential saved for that website.

Autofill policies are rules that password managers follow to decide whether to autofill the login credentials when presented with a login page. Password managers decide whether to autofill a login page that is different from the login page when saving the password as the user would want to autofill a login page that is different but still secure. Therefore, password policies need to have a good balance between security and usability. Password managers have non-identical autofill policies decided by their developers, resulting in different autofill decisions.

1.2 Advanced Features

Password Sharing: This feature is exclusive to password managers that have business versions. Employees can access company accounts by sending a request through the password manager.

Web Monitoring: This feature searches the web to check if the password vault stores any compromised passwords, informing the user to change any compromised password. Some password managers take this feature further by checking the dark web.

File Storing: As government agencies adopt online services, users need to upload important identification when using these online services. This feature ensures secure storage of digital versions of sensitive documents like the user's passport, identification card, driver's licence, etc.

1.3 Types of password managers

Password managers are categorised based on their level of integration with the browser [5].

1.3.1 Browser-based

Browser-based password managers come built into web browsers. These password managers support the basic functionalities and password sync across devices with the browser. Browser-based password managers often provide all their features for free and are available on both the desktop and mobile versions of the web browsers.

1.3.2 Extension-based

Extension-based password managers require users to install their extensions on web browsers for the password manager to work. Extension-based password managers operate on a freemium business model by providing a limited-time free trial or restricting some of the more advanced features.

1.3.3 App-Based

App-based password managers require the user to install the application on their desktop devices for the password manager to work. Users would need to install an extension on the browsers to support websites.

1.4 Architecture of password managers

Bitwarden is an open-source password manager that allows anyone to review their codebase by making their security white papers and documentation available to the public, providing a clear overview of their design choices and security principles [7]. Although other password managers published their documentation on the architecture of their password managers, none has reached the level of detail by Bitwarden.

1.4.1 Overview of User Account Creation

The user's experience of using a password manager starts when they create an account. This process includes creating a Master Password for logging in and unlock the items stored in the vault. Bitwarden's sign up system measures the Master Password's strength, warning the user if the Master Password is predictable. The user can turn on additional authentication measures like two-factor authentication using email or authenticator apps such as Google Authenticator for an extra layer of security.

Once the user has chosen a Master Password, Bitwarden creates an asymmetric key and a Master Key. An asymmetric key is an RSA Key Pair containing a public and private key, only used if the user creates an organisation that shares data between different users. As illustrated in Figure 1, Bitwarden carries out 100,000 iterations of Password-Based Key Derivation Function 2, creating the salted value of a 256-bit Master Key. (PBKDF2-SHA256) using the Master Password and the email address for the account as the salt. This operation is carried out on the user's devices and will not be transmitted or stored in the cloud. The Master Key has two applications:

1) Master Password Hash: The Master Key is encrypted using one iteration of PBKDF2-SHA256 and the master password as the salt, creating the Master Password Hash. The Master Password Hash is securely transmitted from the user's device to Bitwarden's database through HTTPS. Upon reaching Bitwarden's servers, the Master Password Hash is encrypted again using 100,000 iterations of PBKDF2-SHA256 using a random salt generated by a Cryptographically Secure Pseudorandom Number Generator (CSPRNG).
2) Stretch Master Key: The Master Key is stretched into 512 bits using the Hash-Based Message Authentication Code (HMAC) -based Extract-and-Expand Key Derivation Function (HKDF) to create the Stretch Master Key. The Master Password and Stretched Master Key are not transmitted to Bitwarden servers or stored on the user's device.

For future authentication purposes, Bitwarden uses CSPRNG to generate a 512-bit Symmetric Key and an Initialisation Vector, as illustrated in Figure 2. The Symmetric Key is encrypted with AES-256 bit encryption using the Stretched Master Key and the Initialisation Vector, creating the Protected Symmetric Key. The Protected Symmetric Key is stored in the Bitwarden cloud database and used to identify the user when they log in in future instances.

An RSA Key Pair is generated when the user registers their account but it is only used if the user creates an organisation. Bitwarden supports data sharing within users that are in an organisation, allowing a secure system of administrating and usage of passwords for shared accounts. As illustrated in Figure 3, CSPRNG generates an Organisation Symmetric key that is encrypted using the Public Key from the Generated RSA Key Pair, creating the Protected Organisation Symmetric Key. The Private Key from the Generated RSA Key Pair is encrypted with your Generated Symmetric Key using AES-256, creating the Protected Private Key. The Protected Organisation Symmetric Key is similar to the Protected Symmetric Key but instead of identifying an individual user, it identifies an organisation. The Protected Organisation Symmetric Key and Protected Private Key are transmitted and stored in the Bitwarden cloud database.

1.4.2 User Login and Decryption

The vault contains all the sensitive information the user has stored and can be accessed once the user has been authenticated. As illustrated in Figure 4, the Master Key is addi-

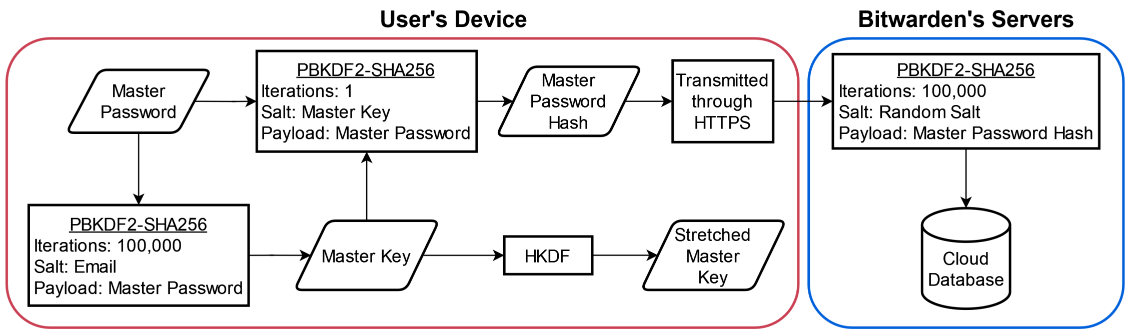


Fig. 1. Diagram describing the process of generating a Master Password Hash

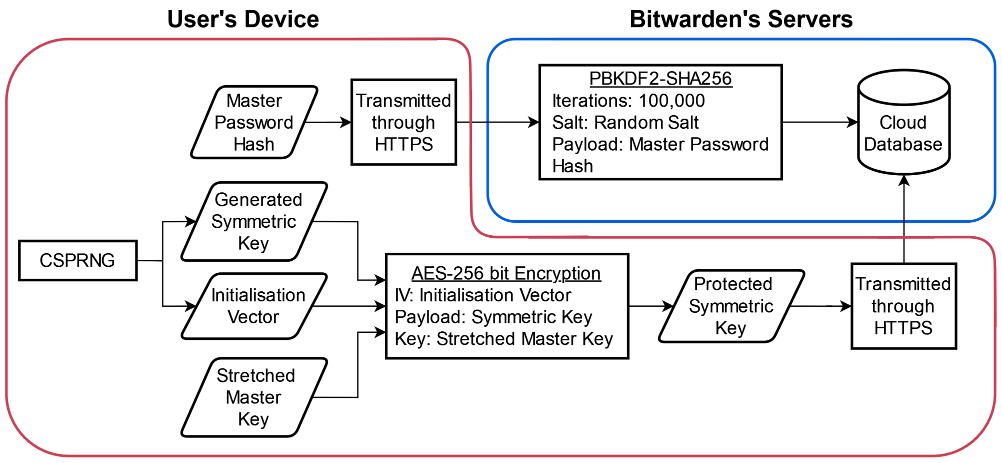


Fig. 2. Diagram describing the process of Password Hashing, Key Derivation, and Encryption

tionally stretched to 512 bits in length using HMAC-based HKDF to create the Stretched Master Key. The Stretched Master Key decrypts the protected Symmetric Key using the Stretched Master Key to get the Symmetric key. The Symmetric Key decrypts the items in the user's vault. The Master Password, Master Key, or Stretch Master Key are not transmitted to Bitwarden servers or stored on the user's device during this process.

When the user enters the email address and Master Password, the same process described in Section 1.4.1 creates the Master Password Hash of the Master Password entered by the user. The newly created Master Password Hash is sent to Bitwarden servers for authentication with the stored Master Password Hash upon account creation.

2 Related Work

Maintenance of security in software is a collective effort by software developers and their users. Password managers have bug bounty programs that incentivise users to report novel security vulnerabilities by providing monetary rewards or recognition. Several academic studies have looked at various aspects of security in password managers.

2.1 Prior Security Analysis on Password Managers

Oesch and Ruoti [5] carried out a security analysis of password managers in 2020. The study is the most recent academic paper on this topic, with the previous academic papers being from 2016 and before. By compiling past security analysis, they carried out their analysis on 13 different app-based, extension-based, and browser-based password managers. They have demonstrated that while password managers have improved in the half-decade before their analysis, there are still significant issues. The password analysis proved that password generation in password managers created random weak passwords. They suggested a password filter serving as a final check before presenting the new password to the user.

Li et al. [8] carried out a security analysis of five popular web-based password managers, namely, LastPass, RoboForm, My1login, PasswordBox, and NeedMyPassword in 2014. Out of 5, only two password managers, LastPass and Roboform, are still available. My1login has shifted its focus to Single Sign-On authentication and enterprise password managers. PasswordBox (Acquired by Intel in 2014 [9]) and NeedMyPassword password manager are no longer available. They found vulnerabilities in different aspects of the password managers, including the user interface, authorisation, bookmarklet, and web applications. These vulnerabilities have a root cause of a range of logic mistakes and misunderstandings about the web security model, in addition to the typical vulnerabilities like Cross-Site Request Forgery (CSRF) and Cross-site scripting (XSS).

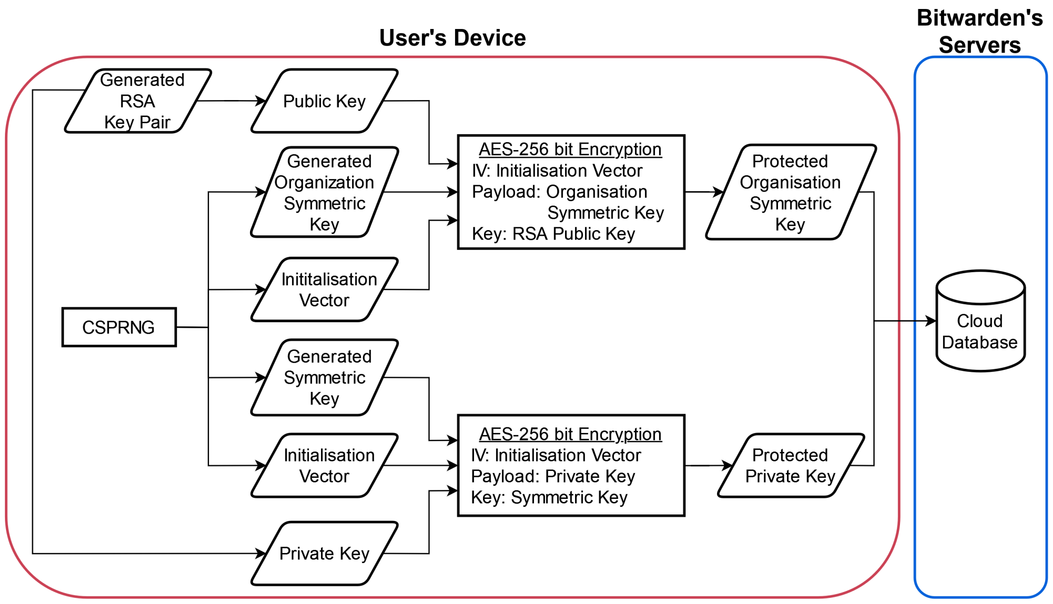


Fig. 3. Diagram describing the process of generating various keys when registering a new Bitwarden account

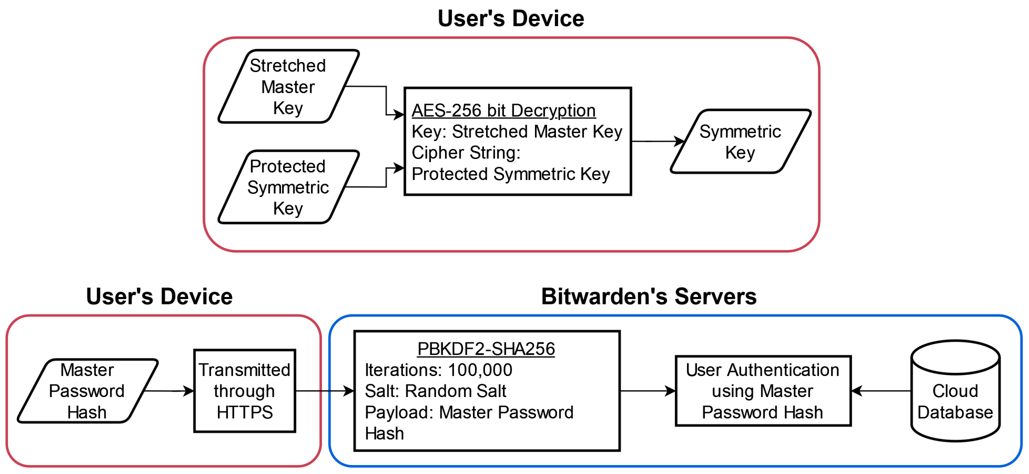


Fig. 4. Diagram describing the process of the User Login and Decryption of vault items

2.1.1 *Bookmarklet Exploits*

A bookmarklet allows the users of a password manager to log in to web applications without needing to install any extensions. Li et al. [8] have found three critical vulnerabilities, including an exploit with the browser request allowing an attacker to include a script of any origin and execute it on the website. The malicious script can extract the encrypted credentials and decrypt them using the user's devices, stealing all the login credentials stored in the password vault. LastPass has since discontinued the bookmarklet feature in 2020.

2.2 Attacks and Defenses of Password Managers

Silver et al. [6] demonstrated ways to attack and enhance the security of password managers. In the paper, they focused on exploits that targeted password autofill. The attacks proved that automatic autofilling without strict autofill policies can have exposed all the user's passwords stored in their password vault. Password sync magnifies the damage, compromising more passwords across different devices. The paper prompted several changes to autofill policies, raising the standard of autofill policies for password managers. The changes included password managers no longer autofilling password fields in iFrames or HTTP pages.

2.2.1 *Sweep Attack*

Sweep attacks steal the credentials for multiple sites at once without the user visiting any of the victim's sites. Attacks take advantage of a login page loaded over HTTP to modify a page on the victim site by tampering with network traffic. These attacks work best on password managers with automatic autofill enabled, highlighting the fundamental danger of this feature. Different techniques including redirects, popup windows, and iFrames are used to implement sweep attacks.

2.2.2 Invisible Forms

Invisible login forms are often used in sweep attacks to avoid alerting the victims. Attacks make login forms invisible by exploiting the login form's CSS display or opacity attribute (either directly or inherited from a parent). Password managers can check if the login form has invisible CSS attributes, display (None) or opacity (0). Attackers can bypass this by using a low value for those attributes to achieve the same results when rendering the login forms. Therefore, password managers should consider this when creating their autofill policies.

2.3 Password Managers Storage

Gasti and Rasmussen [10] are the first to extensively study the storage formats in popular password managers. They demonstrated how practical attacks exploit specific vulnerabilities in databases by designing two security models to represent the capabilities of real-world adversaries. Their study proved that even weak adversaries can break most password manager storage formats.

Even with industry-standard practices at the time of investigation, such as AES-CBC, password databases are not secure against determined attackers. A passive attacker may extract all the passwords to carry out impersonation or fraud, while an active attacker may modify or delete the login credentials to prevent the user from logging into their accounts. Some of the password managers depend on the storage mechanism of the password files and not the encryption of the contents of the login credentials, which poses a serious threat to the security of the login credentials stored. In their work, they created database formats that provide security, usability, and low computation and storage overhead, using standard cryptography tools.

2.4 Passwords

The generation of secure passwords by password managers is essential in ensuring that the main features of password managers remain secure. Password strength can be measured by looking at information entropy and analyzing the results from password cracking tools [11], [12].

2.4.1 Password Analysis Methods

zxcvbn: is a password strength estimator that uses pattern matching and conservative estimations [13]. It uses data of 30k common passwords, common names, and surnames according to US census data, common English words from Wikipedia and US television and movies, and other common patterns like dates, repeats (e.g. aaa), sequences (e.g. abcd), keyboard patterns (e.g. first row of the QWERTY keyboard layout - qwertyuiop), and leetspeak (password becomes p4s5w0rd). With password composition policies, users often create passwords that meet the bare minimum of password composition policies, resulting in more predictable passwords. zxcvbn eliminates fixed rules in password composition policies, making it more secure, flexible, and usable. zxcvbn requires minimal computing power to be run on websites and mobile applications to give instant targeted feedback to guide users to augment their passwords to be less guessable.

2.4.2 Password Cracking

Password policies are based on naive metrics like length, presence of non-alphabetic and uppercase/lowercase characters, resilience to common attacks like brute force and dictionary [14] but this does not include more advanced cracking techniques. State-of-the-art password-guessing tools, such as HashCat and John the Ripper, enable users to check millions of passwords per second against password hashes. In addition to performing straightforward dictionary attacks, these tools can expand password dictionaries using password generation rules, such as concatenation of words and leetspeak.

Brute Force Attack: uses trial-and-error to guess login credentials by working through all possible combinations. This method is naive and ineffective against passwords with longer lengths.

Dictionary Attack: uses a library of curated natural language words and common passwords that are used in passwords [14]. This technique works best on weak passwords but applying mangling rules to these libraries can generate a more extensive list. The mangling rules can include the juxtaposition of words, appending or prepending sequences of digits or symbols to passwords, or capitalising words.

Markov Chain-based Attack: utilises statistical analysis of a list of words and the probability of each character appearing from the characters before it [14]. For example, the letter 't' or 'r' may have a higher probability of appearing after 'ca' than 'z'. Therefore, password-cracking software like John the Ripper may try guessing 'cat' and 'car' before 'caz'. This is a more efficient method when compared with the methods mentioned above.

PassGAN: is a novel approach for using deep machine learning for password cracking introduced in 2019 by B. Hitaj et al [15]. HashCat and John the Ripper work well in practice, but expanding them to model further passwords is a laborious task that requires specialised expertise. Contrarily to password-cracking software, PassGAN uses a Generative Adversarial Network (GAN) to autonomously learn the distribution of real passwords from actual password leaks and to generate high-quality password guesses. PassGAN replaces human-generated password rules with theory-grounded machine learning algorithms. PassGAN is trained on real passwords from actual password leaks to generate high-quality password guesses. The paper reports the performance of PassGAN by using two large datasets, RockYou Dataset [16] and LinkedIn Dataset. Results from HashCat and PassGAN generated 51%-73% more matching passwords than with HashCat alone.

3 Methodology

3.1 Experimentation Setup

All password managers are tested on Microsoft Windows 10 Home Operating System (Version 10.0.19043 Build 19043). The extension-based password managers are installed on Google Chrome and the specific versions are presented in Table 1 with the abbreviations that are used in this paper. A study [17] found that users use password managers due to usability and not security benefits. Therefore, we assume that the typical user will not change their settings, all tests are done with the default settings of the respective password managers.

TABLE 1
Password Managers Versions

Password Manager	Abbreviation	Version
Google Chrome	GC	95.0.4638.69
Microsoft Edge	ME	95.0.1020.53
Mozilla Firefox	MF	94.0.1
1Password	1P	2.1.4
Bitwarden	BW	1.54.0
Dashlane	DL	6.2142.2
LastPass	LP	4.84.0

3.1.1 *Password Managers Selection Process*

We chose the browser-based password managers based on their web-browser market share globally [18] and selected the top three market shares. We have decided to exclude Safari as it is only available for the Apple ecosystem. As for extension-based password managers, we made our choices based on their popularity and ratings on the Chrome Web Store, selecting the extensions with more than 1 million users and three or more star ratings.

Google Chrome: Developed by Google, the Google Chrome browser has a built-in password manager. Google Chrome is the most popular web browser by a big margin, with more than 65% of the global browser market share. [18]. Google Chrome uses Chromium, a free and open-source web browser project developed and maintained by Google.

Microsoft Edge: Developed by Microsoft to replace Internet Explorer, the Microsoft Edge browser comes with a built-in password manager. Edge has some similar features to Google Chrome as it uses the same code base, Chromium [19].

Mozilla Firefox: Developed by Mozilla, Mozilla Firefox is a free and open-source web browser with a built-in password manager, Firefox Lockwise (also known as Lockbox). Firefox Lockwise comes built in the web browser and is available as a mobile application that can sync together.

1Password: Available as a desktop application, browser extension, or mobile app, 1Password is a freemium password manager. 1Password reports that they have more than 15 million users on their website [20].

Bitwarden: Available as a desktop application, browser extension, mobile app, or command-line interface, Bitwarden is a free and open-source password management service and provides all its functionality to individual users with unlimited passwords and synced devices with non-paid accounts. It is hard to estimate the number of users Bitwarden has, as it is open-source, and users can host the program on their local servers.

Dashlane: Available as a browser extension or mobile app, Dashlane is a freemium password manager. Dashlane discontinued its desktop application in January 2022, opting for web-based applications as their only option for desktop users. In 2021, Dashlane is estimated to have 15 million users [21].

LastPass: Available as a desktop application, browser extension or mobile app, LastPass is a freemium password manager. LastPass reported more than 25 million users in 2020 [22] and is the most popular password manager for Google Chrome with more than 10 million users with 4.5 stars average with around 30k ratings on Google web store [23].

3.2 Password Generation Analysis

When generating passwords, there is a possibility the password manager generating a weak password, e.g. repeating textual patterns, common words, etc. Although the probability of generating these passwords is slim at longer lengths, it is still possible. To evaluate the passwords generated by the password managers, we collect a large number of passwords from the password managers and carry out tests using information entropy techniques and other existing tools, zxcvbn and PassGAN.

3.2.1 *Password Corpus*

We have collected 100,000 passwords from each of the password managers using their default password compositions. We aim to maximise replicability but the methods of collection vary based on the functionality of the password managers. We collected passwords from password managers with a command-line interface (BW and LP) by piping the output of the CLI into a text file. For password managers with no command-line interface (1P and DL), we wrote Python scripts to run Selenium, a browser automation tool, to simulate mouse clicks to collect passwords from web pages. For browser-based password managers (GC, ME and MF), we collected the passwords using macros that simulate mouse clicks, as Selenium doesn't work on the browser's native prompts.

3.2.2 *Password Evaluation*

After collecting the password corpus, we evaluate the strength of the passwords of each password manager by looking at how random or guessable they are. We utilised information entropy techniques from Oesch and Ruoti's work [5], zxcvbn, and assembled a novel evaluation framework incorporating deep machine learning password cracking, PassGAN.

Shannon Entropy: To check for abnormalities in the frequency of characters in the passwords, we use Shannon entropy to measure the average minimum number of bits needed to encode a string of symbols based on the frequency of their occurrence. This is calculated using the formula:

$$\sum_i p_i log_b(p_i)$$

Shannon entropy measures the number of distinct characters that can be present in a string and their relative frequency within each password set. This measure is not affected by the password length [5].

χ^2 Test: The randomness of the passwords is measured using a simple statistical test, χ^2 test. We determine whether the difference between the two distributions can be explained by random chance by evaluating the password corpus independently using the χ^2 test. The p-values are corrected using a Bonferonni correction to account for the multiple statistical tests for the same family of characters.

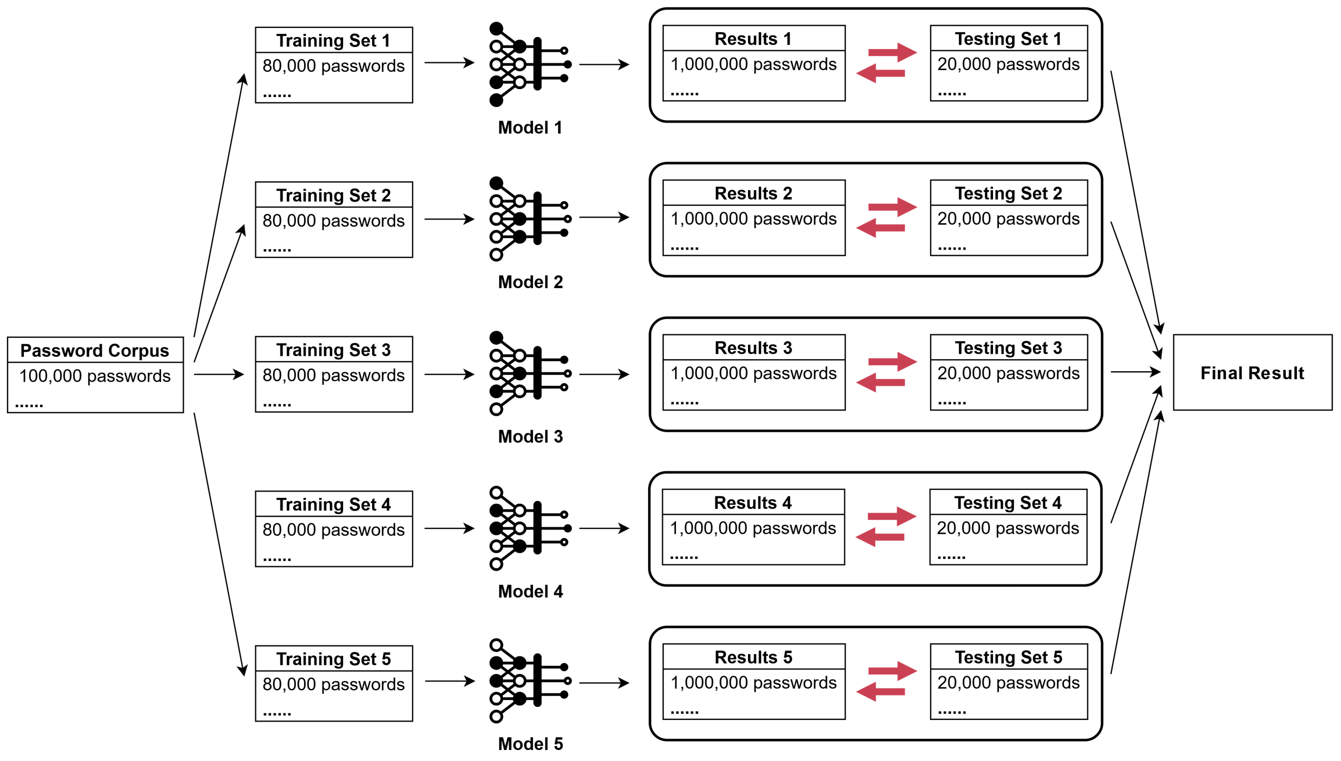


Fig. 5. Diagram describing the evaluation framework using PassGAN

The correction multiplies each p-value by 147 and caps the maximum value at 1.00 [5].

To supplement the Shannon entropy and χ^2 test, we carried out a character frequency ratio to explain any abnormal results we may find. We calculate how often, on average, each character is used in each password. This is calculated using the formula:

$$length \times \frac{1}{|characters_{all}|}$$

Zxcvbn: We used zxcvbn to evaluate the strength of passwords as it is a quick and efficient way of estimating passwords' strength. Zxcvbn rates each password based on the number of guesses required to crack the password for both online and offline attacks, categorising passwords that require more than 10^6 guesses to be resilient against online attacks and passwords that require more than 10^{10} guesses to be resilient against offline guessing attacks [13]. The tool provides warnings and feedback on weak passwords (less than 10^8 number of guesses), a feature used by websites to warm its users during the sign-up process. The result reveals the most insecure password generated by each password manager and how the password managers performed on average.

PassGAN: For more subtle patterns in the passwords generated that are not detected by zxcvbn, we implemented a 5-fold cross-evaluation algorithm that uses PassGAN to evaluate the passwords as illustrated in Figure 5. By separating the 100,000 passwords into a 20/80 train-test split, we train PassGAN on 80% of the training split passwords using its default parameter for 50,000 iterations. Once done training, we use each PassGAN model to crack the remaining 20% of the testing split passwords the model has not seen before using 1 million guesses, and we record the number of passwords the model guessed correctly. We repeated this process five times for each password manager by rotating the 20% of the passwords from the testing split to the training split and averaged the results to get the final results of this algorithm.

The RockYou Dataset [16] is a collection of 14,341,564 unique passwords that were leaked or stolen from various sites. The collection of passwords is preprocessed by the author, removing any names, emails, or personal information. To compare the passwords generated by password managers and humans, we utilised the publicly available RockYou Dataset by randomly selecting 100,000 passwords from the dataset and repeating the algorithm to have a baseline result for human-created passwords.

3.3 Password Storage Analysis

Although password managers utilise both cloud and local storage, our analysis only examined the local storage of the password vault on the user's devices, as we do not have access to cloud storage. We replicated the analysis by Oesch and Ruoti [5] to discover any changes made to the storage methods recorded. We manually inspected each password manager's local files to examine which information is available and not encrypted. Furthermore, we compare each password manager's storage methods by looking at their latest white papers that are publicly available.

3.4 Password Autofill Analysis

Password managers need to be able to differentiate between a legitimate and malicious website before autofilling the saved login credentials. If a website is compromised by attackers, they can insert malicious JavaScript code that can steal the user's password when it is entered. Automatic autofill makes the task of the attacker easier as the malicious JavaScript code would only need to overcome the password manager without user interactions to carry out their attack.

We reconsolidated the testing websites made public by Oesch and Ruoti [5] that leveraged attacks identified by previous security analyses [6], [8], [24]. The testing websites identify the autofill method used for HTML form attributes (autocomplete, actions, and login field name/types), HTTPS/HTTP, invisible login forms, website redirects, and IFrames. We removed unnecessary tests (i.e. low opacity login forms) and added links to the testing websites to make the testing cycle more efficient. This allows us to examine any changes made to the autofill policies, highlight any existing security vulnerabilities and if any changes are needed.

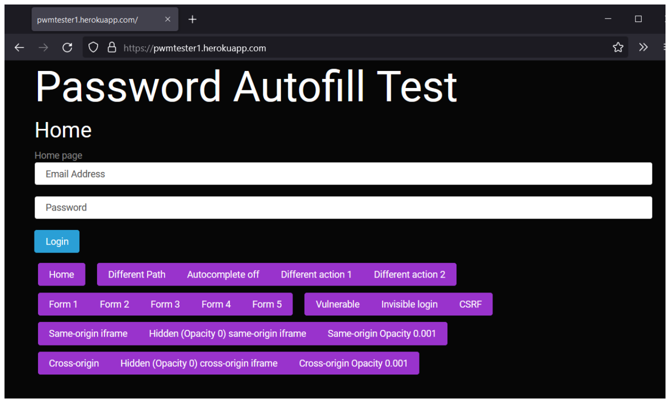


Fig. 6. Autofill testing site

3.5 Ethics and Responsible Disclosure

The aims of the project are purely academic. Ethical considerations are upheld throughout the project. All testing and experimentation are carried out on the author's account. All security vulnerabilities discovered in the project are responsibly disclosed to the appropriate professional or the organisation of that password manager if it poses an actual security threat to current users.

4 RESULTS

4.1 Password Manager Features

As presented in Table 2, all password managers provide the basic features, while the advanced features are only available in extension-based password managers.

4.1.1 Cloud Sync for Settings

Browser-based password managers have the edge over extension-based password managers by syncing settings across all the devices using the same account. Browser-based password managers achieve this by incorporating their settings as part of the browsers' settings, while extension-based password managers have their settings limited to the individual devices due to implementation limitations in extensions. Browser-based password managers

TABLE 2
Summary of Password Managers Features

	GC	ME	MF	1P	BW	DL	LP
Supports password generation	✓	✓	✓	✓	✓	✓	✓
Supports autofill	✓	✓	✓	✓	✓	✓	✓
Support password sharing	✗	✗	✗	✓	✓	✓	✓
Support web monitoring	✓	✓	✓	✓	✓	✓	✓
Support non-password	✓	✓	✓	✓	✓	✓	✓
Cloud sync for settings	✓	✓	✓	✗	✗	✓	✗
Cloud sync for vault	✓	✓	✓	✓	✓	✓	✓
CLI support	✗	✗	✗	✗	✓	✗	✓
Supports MFA	✓	✓	✓	✓	✓	✓	✓
Lockable Vault	✗	✗	★	✓	✓	✓	✓
Master password requirements	✗	✗	✗	✓	✓	✓	✓
Login on separate tab or app	✓	✓	✓	✓	✗	✗	✗
Has assessment tool	✓	✓	✗	✓	✗	✓	✓
Clears clipboard	✗	✗	✗	✗	✗	✗	✗
Open Source	★	★	✓	✗	✓	✗	✗

* ✓ = Feature is available
✗ = Feature is not available
★ = Special case, explained in the following section

provide the bare minimum control for the users to include the settings of the password managers in the browser's settings. In browser-based password managers, browsers only allow the user to turn on or off half a dozen basic functionalities. Whereas extension-based password managers allow personalised customisation for each case, thereby improving user experience and security. DL syncs their settings by implementing the settings as part of their dashboard on their website. This complicates the process of changing settings as the user is prompted to log in again when the user is redirected to the dashboard.

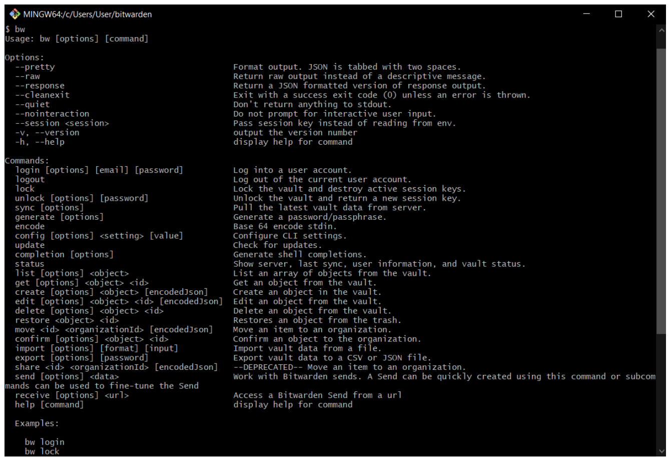
4.1.2 Command-line interface Support

Command-line interface (CLI) is a text-based user interface that can be used to run password managers. It is a powerful tool often used by administrators to supervise all the accounts in the organisation. Without any graphical user interface, it provides a faster and more efficient method of using the features provided by the password managers. All features of the password managers are available in the CLI and can be easily utilised by entering simple commands, often with faster completion times, as it doesn't require any graphical feedback or animations.

Out of the seven password managers we analysed, only Bitwarden and LastPass provide their users with CLIs. Bitwarden password manager, illustrated in Figure 7, was released in 2018 [25] and is available on their official GitHub page. The CLI codebase is constantly updated and improved by Bitwarden's developers and open-source contributors. LastPass' open-sourced CLI was first created in 2014 [26] and is available on their official GitHub page [27]. The LastPass developers have stopped updating the CLI codebase since 2019, although other contributors are still maintaining the codebase.

4.1.3 Lockable Vault

GC and ME prompt users to enter the device's password when they try to copy, view, edit, or autofill, but no password is needed to delete any login credentials. Once the user enters the password, the password vault is open for 1 minute or the current session (depending on which



```
$ bw
Usage: bw [options] [command]

Options:
  --pretty                                   Format output. JSON is tabbed with two spaces.
  --raw                                      Return raw output instead of a descriptive message.
  --response                                 Return a JSON formatted version of response output.
  --cleanexit                                Exit with a success exit code (0) unless an error is thrown.
  --quiet                                    Don't return anything to stdout.
  --nointeraction                            Do not prompt for interactive user input.
  --session <session>                        Pass session key instead of reading from env.
  -v, --version                              output the version number
  -h, --help                                 display help for command

Commands:
  login [options] [email] [password]         Log into a user account.
  logout                                     Log out of the current user account.
  lock                                       Lock the vault and destroy active session keys.
  unlock [options] [password]                Unlock the vault and return a new session key.
  sync [options]                             Pull the latest vault data from server.
  generate [options]                         Generate a password/passphrase.
  encode                                     Base 64 encode stdin.
  config [options] <setting> [value]         Configure CLI settings.
  update                                     Check for updates.
  completion [options]                       Generate shell completions.
  status                                     Show server, last sync, user information, and vault status.
  list [options] <object>                    List an array of objects from the vault.
  get [options] <object> <id>                Get an object from the vault.
  create [options] <object> [encodedJson]    Create an object in the vault.
  edit [options] <object> <id> [encodedJson] Edit an object from the vault.
  delete [options] <object> <id>             Delete an object from the vault.
  restore <object> <id>                      Restores an object from the trash.
  move <id> <organizationId> [encodedJson]   Move an item to an organization.
  confirm [options] <object> <id>            Confirm an object to the organization.
  import [options] [format] [input]          Import vault data from a file.
  export [options] [password]                Export vault data to a CSV or JSON file.
  share <id> <organizationId> [encodedJson]  --DEPRECATED-- Move an item to an organization.
  send [options] <data>                      Work with Bitwarden sends. A Send can be quickly created using this command or subcom
mands can be used to fine-tune the Send
  receive [options] <url>                    Access a Bitwarden Send from a url
  help [command]                             display help for command

  Examples:

    bw login
    bw lock
```

Fig. 7. Bitwarden's command-line interface outputting its options, commands, and examples

occurs first) and the user has full access to all the login credentials in the password vault. The password vault will prompt the user for the devices' password once the 1-minute window is up or if it is a new session. GC and ME do not have any options to block unauthenticated users from viewing the websites of the login credentials saved in the password vault. Chromium's backlog has issues opened in 2008 [28], [29] for the developers to implement an option to set a master password. Chromium developers stated that a master password would not improve security and labelled the issues as "WontFix (Closed)".

MF is the only password manager we analysed that does not lock the password vault by default. MF provides an option to turn on a master password to lock the password vault on the local machine, but no password policy is required for the master password. The lockable vault setting is not synced with other devices connected to the same account, meaning other devices can keep the password vault open or set up a different master password. When the user turns this option on, the browser will prompt the user to fill in their master password whenever they want to edit, delete, view, or autofill a password, blocking any unauthenticated user from viewing the contents of the password vault. The user would only need to authenticate once each browser session.

Extension-based password managers use the same password as the account for the password vault. Out of the four extension-based password managers, BW has the most relaxed password policy for master passwords, only requiring eight or more characters with no other password requirements. Although it warns about weak and common master passwords when signing up, it does not stop the user from continuing to use it. Other extension-based password managers have more complex master password policies, at least one character of letters, numbers and symbols and stops users from using weak passwords with common sequences.

4.1.4 *Open Source vs. Closed Source*

For the password managers we looked at, only MF and BW are open-sourced, while GC and ME are partially open-source. Although GC and ME's codebases are not public, they are based on Chromium, which is open-source. Most of the GC and ME's codebases are built on top of Google's free and open-source software project Chromium, but they are licensed as proprietary freeware (closed-source but free to download/use). Microsoft's decision to make ME use Chromium aims to benefit both the users and the developers by using the existing Chromium codebase [30]. By utilising Chromium, it reduces complexity and overhead for web developers that are already testing their web applications on Chrome. With most of the Chromium-based browsers sharing a good number of common features and user-interface components, this provides users with a more consistent user experience when switching between browsers.

4.2 Password Analysis

Our analysis of password generation in password managers measures the randomness and strength of the password generated.

4.2.1 *Password Generation Features*

Password managers use different password policies and methods to generate secure and random passwords. Table 3 presents the different features of each password manager. Extension-based password managers allow users to customise the length and password composition with different character sets: letters, symbols, and digits, see Table 4. By default, all password managers except LP and DL remove characters that are difficult to differentiate from the character sets. For example, MF removes the letters I, O, l, o, and numbers 0 and 1. Although LP and DL provide the user with the option to enable this feature, browser-based password managers do not support customisation for password generation.

Browser-based password managers do not allow users to customise the password policies, restricting the length and compositing to the default. Although this is not a problem for GC and ME, as the password generated is 15 characters in length with all character compositions, satisfying most if not all common use cases. MF is the only password manager that does not have the option to generate passwords with symbols. An issue was opened on Mozilla's issue-tracking system (Bugzilla) in 2019 [31] for the developers to implement passwords generation that includes symbols. The developers have stated that the issue can be overcome by users manually editing the passwords generated and flagging the issue as "wontfix" for previous versions of Firefox. This issue causes inconvenience to users when the website requires passwords that include symbols and passes the responsibility of creating passwords that meet the password policy to the users. These inconveniences can be easily fixed by adding settings for the length and password compositions to browser-based password managers.

4.2.2 *Password Randomness*

The results of our analysis on randomness are summaries in Tables 5, 6, 7, 8. Shannon's entropy, Table 5, suggests that the password managers exhibit a more balanced mix of characters from both the character sets and passwords themselves when compared to passwords from the RockYou dataset. The results for character frequency ratios (see Table 6) demonstrates that characters are used equally frequently

TABLE 3
Password Generation Features

	GC	ME	MF	1P	BW	DL	LP
Supported lengths	15	15	16	8-50	5-128	4-40	1-99
Default length	15	15	16	20	14	16	12
Avoid difficult characters	✗	✗	✓	✓	✓	✓	✓
Default composition*	all	all	ld	all	ld	all	ld
Available composition*	N/A	N/A	N/A	all	all	all	all

* Password compositions: l = letters, ld = letters and digits, ls = letter and symbols, sd = symbols and digits, all = combination of character sets listed previously

TABLE 4
Password Generation Symbol Sets

	Count	Symbols
GC	22	!#$%&'()*+,-./:;=?@^_~
ME	5	!-.:_
MF	-	N/A
1P	19	!#%)*+,-.:=>?@]^_}~
BW	8	!#$%&*@^
DL	6	!#$&?@
LP	32	!"#$%&'()*+,-./:;<=>?@[\]^_'{|}~

in each password manager with a standard deviation of between 0 and 0.002. This result is significantly better than the RockYou dataset, passwords created by humans and the minimum character frequencies for all password sets created by password managers are non-zero, signifying that passwords have at least one character from each character set. The results from the χ^2 test 7 suggest that only LP has no statistically significant results. The result can be attributed to the password corpus collected on the default settings, length 12, password composition ld and no difficult characters removed. Table 8 reveals that the characters from the smaller character set appear more frequently in the passwords generated. This is especially prevalent in BW where all nine numbers are used most frequently.

TABLE 5
Entropy

	Set	Password	Char Count
GC	6.274	94.281	78
ME	5.908	88.961	61
MF	5.800	87.110	56
1P	6.266	125.336	77
BW	5.823	81.660	57
DL	6.069	97.399	68
LP	5.954	71.450	62
RockYou	3.342	91.978	95

TABLE 6
Characters Frequency Ratios

	Mean	σ	Min	Max
GC	0.013	0.002	0.012	0.018
ME	0.016	0.0	0.015	0.025
MF	0.018	0.002	0.017	0.023
1P	0.013	0.0	0.013	0.014
BW	0.018	0.002	0.016	0.023
DL	0.015	0.002	0.013	0.022
LP	0.0016	0.0	0.016	0.016
RockYou	0.006	0.009	0.0	0.044

TABLE 7
χ^2 Test

	Random*	p	χ^2
GC	✗	0.0	25687
ME	✗	0.0	52259
MF	✗	0.0	17466
1P	✗	0.0	1247
BW	✗	0.0	19706
DL	✗	0.0	45631
LP	✓	0.79	52
RockYou	✗	0.0	19365152

* ✓ = No statistically significant results
✗ = Statistically significant result

4.2.3 Password Strength

Our analysis of zxcvbn results found that all passwords generated by password managers except LP are resilient to online and offline guessing attacks, requiring more than 10 guesses. The weakest passwords of each password manager presented in Table 9 suggests that LP has generated the weakest password overall. The weakest password generated by LP, "k5yeroOmFoUr", is a leetspeak of the words key, room, and four. Zxcvbn rated the password to be guessable within 20 hours in an offline slow hashing attack (1^4 guesses per second). Out of 100,000 passwords collected from LP, 11 passwords are rated to be suboptimal, containing different sequence patterns including, brute force, repeat, and dictionary. This can be explained by the default password composition LP uses, making it statistically generate weaker passwords. LP can fix this by implementing a weak password filter or changing their default password composition to length 14 or longer.

The results from PassGAN did not detect any subtle patterns that zxcvbn could not detect as summarised in Table 10. The models trained on the training set of our password corpus were not able to guess any of the passwords in the testing set. However, the model was able to guess 926 passwords on average when trained and tested on the baseline dataset, the reduced RockYou dataset. This proves that PassGAN is highly more likely to generate human-created passwords due to the patterns in these passwords and the passwords generated by password managers do not contain such patterns.

4.3 Password Storage Analysis

Table 11 summarises the results of our manual examination of password manager storage and documentation [7], [20], [32], [33], [34], [35]. All password managers encrypt their databases using AES-256. Chromium browsers, GC and ME, utilise the operating system to encrypt the password, but the method varies with the operating system. All extension-based password managers except DL use PBKDF2-SHA256, with more than 100,000 rounds. DL uses 3 rounds of Argon2D, a modern and secure key derivation function resistant against attacks utilising ASICs (Application Specific Integrated Circuits) and GPUs (Graphics Processing Units) [36]. ASICs and GPUs are hardware capable of intensifying hash cracking speeds, increasing the speed of cracking password hashes.

In prior security analysis, MF handled the encryption of storage itself by using 3DES to encrypt passwords and a

TABLE 8
Characters sorted by frequency

	Characters (Highest to lowest frequency)
GC	56837294,=&*:'?-@$%M+.)#~!/x;^(wgYEX_dyChVUteviGDScJmBAfkLzFjQTNqaprsbPKuZHRWn
ME	_:!-.29543876PuxDYVzydimXLQNqwkSFJfKbCMGHjTceEvRsphZUgatBArnW
MF	58794326fwUFstqQBeWyjZDvpKaCbTmhXPVJErgSxnLckdAuHzGiMNRY
1P	3890627514:?>F%#te@q^da_u*!}U+sPrvc,.zkbE~JgM=Do)-RL]ZGTKNnCYjiQwyAXhVBfmHpWx
BW	35768942WRvZBAVXNDdYQwKaHTmFiSUbyzfPCGkEJMguhetcLopsqjrnx
DL	$?#@!&9734518260aITEcHoLGOArjkdUJtwyfXmWKRlCNgxqVQDFevzSBsYMpibuPnhZ
LP	7Zu0nx4Qob51cyOCfdpVGSAi3IMFKjJelsNL6mBrtwahzYHqDXPvgRW9E2T8kU

TABLE 9
Weakest Passwords Generated

	Password Generated	Guesses*
GC	Aw&h;N:uM%82323	1.0×10^{13}
ME	YsxP_:4sMiThqP8	3.0×10^{12}
MF	KVe2JhgC9etEmEm	1.8×10^{12}
1P	cQH~PmoNK3yBa!dc^2s:	2.7×10^{17}
BW	semaJ3JU6FDtHD	1.0×10^{11}
DL	XH$AOc2?@1?neLeh	3.0×10^{13}
LP	k5yeroOmFoUr	7.2×10^{8}

* Estimated number of guesses by zxcvbn

TABLE 10
PassGAN Results

	GC	ME	MF	1P	BW	DL	LP	RockYou
Matches	0	0	0	0	0	0	0	925.6

single round of SHA-1 to derive the encryption key [5]. The design choice was insecure as it allows password-cracking methods like brute force attacks, and dictionary attacks to be viable. Although MF now uses AES-256 with 10,000 rounds of PBKDF2, it is still one-tenth of the 100,000 rounds used in extension-based password managers as shown in Table 11.

4.3.1 Metadata Encryption

Compared to Oesch and Ruoti's [5] observations, there have not been any significant changes to the protection of metadata. Browser-based password managers do not encrypt any metadata; the URL, username/email, the number of times the password has been autofilled and UNIX timestamps of password creation and modification dates are stored on the users' devices in an SQLite (Chromium) or JSON (MF) file unencrypted. The local folders of the browsers contain the user's email and settings in a JSON (Chromium) or JavaScript (MF) file. Attackers with access to the users' devices can easily access these files, exposing the browsing habits of the users.

BW uses their icon server to provide the delivery endpoint for website icons of the password saved [37]. If the user is using website icons on their devices, BW will issue requests to icons.bitwarden.net for each login in the password vault that has a URI that resembles a website. The request for an icon contains the hostname of the website stored in the password vault, exposing which websites the user has passwords stored for. This option is turned on by default but can be turned off by the user in the settings to prevent the requests for icons from exposing which websites the user visits.

4.4 Password Autofill Analysis

All the password managers we evaluated support autofill in browsers. Table 12 presents the results of our analysis of the password managers' autofill policies. All password manager autofill policies we looked at define criteria where the default autofill policy would apply, and this policy would turn stricter if a criterion is not met. Browser-based password managers automatic autofill by default, but if the initial website redirects to another website, GC will continue automatic autofill while ME switches to manual autofill and MF turns off the autofill feature. These behaviours are decided by the developer to trade usability for security at times when the website contains potentially malicious elements. Browser-based password managers do not allow any customisation to their autofill policies and the only option available for users is to turn the autofill feature off completely.

Overall, extension-based password managers except DL have default password policies that keep the user safe while providing convenience on websites without any potential malicious changes. 1P and DL have a safer approach by having manual autofill as the default and providing users with the option of switching to automatic autofill. All extension-based password managers provide users with the option to customise autofill policies, giving the user the choice between automatic and manual autofill for different use cases. This gives the users the ability to customise the autofill policies to be more secure or convenient if they wish to do so.

4.4.1 HTML autocomplete Attribute

The HTML `autocomplete` attribute allows web developers to specify the permissions of automated assistance in interacting with the form field, which includes password managers. We tested if password managers detected this parameter and disabled their features on login forms with `autocomplete` disabled. Our analysis found that all password managers did not offer to create passwords and require user interaction before storing the passwords when `autocomplete` is disabled. However, once the user has stored their login credentials, all password managers, except 1P and BW, ignore the `autocomplete` attribute and autofill the login forms.

4.4.2 Different Login Forms

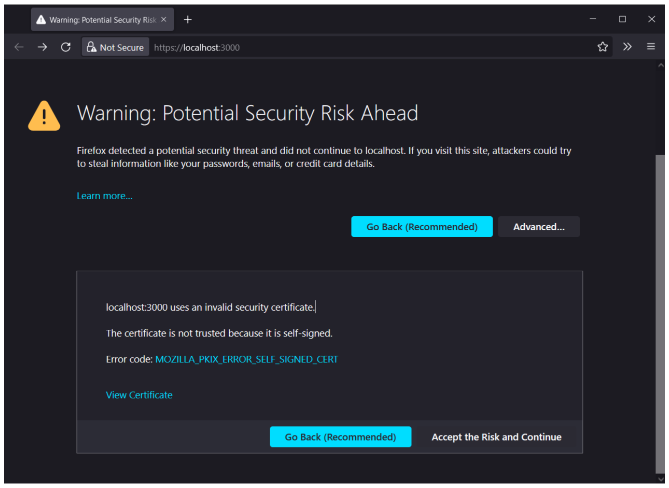
Password managers identify which login credentials to autofill based on the URLs and the login forms. When the login form uses a different action, i.e. sending the request to a different address, 1P and LP disable their autofill feature,

TABLE 11
Password Manager Storage Features

		GC	ME	MF	1P	BW	DL	LP
Storage	Encryption	OS	OS	AES-256	AES-256	AES-256	AES-256	AES-256
	Key Derivation Function (KDF)	N/A	N/A	PBKDF2	PBKDF2	PBKDF2	Argon2d	PBKDF2
	KDF Rounds (Client-side)	N/A	N/A	10,000	100,000	100,001	3	100,100
Metadata Encryption	URL	✗	✗	✗	✓	✓	✓	✓
	Icon	✗	✗	✗	✓	★	✓	✓
	Username/Email	✗	✗	✓	✓	✓	✓	✓
	Creation date	✗	✗	✗	✓	✓	✓	✓
	Modification date	✗	✗	✗	✓	✓	✓	✓
	Last use date	✗	✗	✗	✓	✓	✓	✓
	Fill count	✗	✗	✗	✓	✓	✓	✓
	User's email	✗	✗	✗	✗	✗	✗	✗
	User's settings	✗	✗	✗	✗	✓	✓	✗

✓ = Encrypted ✗ = Not encrypted ★ = Special case, explained in the following section

preventing the user from autofilling potentially malicious login forms. However, they fail to detect the change if the malicious login form is dynamically changed by the JavaScript code after the website is loaded. This is hard to detect as it requires the password manager to constantly check the login form's attributes as the JavaScript code can change the action right before the request is sent. Other password managers do not adjust their autofill policies for this situation. All password managers suitably adjust their autofill policies when a potentially malicious change is detected. For example, when text inputs are presented as the login form, GC, ME, MF, and LP do not recognise the input as a login form. To balance security and usability, 1P, BW and DL require the users to manually autofill the text input.



Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to localhost. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

localhost:3000 uses an invalid security certificate.

The certificate is not trusted because it is self-signed.

Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

Fig. 8. Mozilla Firefox warning user about invalid self-signed certificate.

4.4.3 HTTPS Certificates

When the HTTPS certificates are valid, all password managers use their default autofill policies; GC, ME, MF, and DL use autonomic autofill and 1P, BW and LP use manual autofill. When using an unsecured connection (i.e. HTTP), websites are more prone to network injection attacks. All password managers have taken precautions in this situation and adjusted their autofill policies. All browser-based password managers turn off their autofill features, while extension-based password managers opt for the more secure, manual autofill.

Browsers prompt a warning to the users to alert them about the security risk if they proceed to a website that is insecure as illustrated in Figure 8. Similar warnings are prompted due to invalid certificates for different reasons including, self-signed, expired, corrupted, etc [38]. When the user chooses to proceed despite the warning prompted by the browsers, MF and DL automatically autofill the login field if the user has saved any login credentials. When this similar situation occurs, other browsers like GC and ME disable their password manager and password managers like BW, 1P, and LP require manual autofill. All password managers should disable their autofill if the HTTPS certificates are invalid.

4.4.4 Redirects and IFrames

When a website requests the users to log in, often the website redirects the user to a login page, or presents an IFrame. However, autofilling passwords within IFrames can be especially dangerous, even with manual autofill [24]. For example, a malicious website can trick users into providing the necessary user interaction to autofill by placing invisible items over essential buttons like ”Enter”. For same-origin IFrame, GC, ME, MF, and LP use automatic autofill, while 1P, BW, and DL use manual autofill. DL is the only password manager we examined that adjusted its autofill policy for same-origin IFrames, changing from automatic autofill to manual autofill.

Attackers can use cross-origin IFrames to extract passwords from other sites that the current user is not visiting. These attacks require the attacker to have access to the user's Wi-Fi access point to conduct a network injection attack and can be carried out within minutes if automatic autofill is enabled, stealing all passwords the user has stored. Although the attackers cannot pinpoint which website the users have passwords saved for, the attack includes presenting login pages for all common email, banking, and social media websites, relying on automatic autofill to do the job. All password managers we examined adjusted their autofill policy for cross-origin IFrame, requiring manual autofill.

TABLE 12
Password Manager Autofill Policies

	GC	ME	MF	1P	BW	DL	LP
Store autocomplete=”off”	★	★	★	★	★	★	★
Autofill autocomplete=”off”	✓	✓	✓	★	★	✓	✓
Different URL path	✓	✓	✓	★	★	✓	✓
Different login form action	✓	✓	★	✗	★	✓	✗
Dynamically changed form action	✓	✓	✓	★	★	✓	✓
Remove method/action	✓	✓	✓	★	★	✓	✓
Remove field name	✓	✓	✓	★	★	✓	★
Change field name	✓	✓	✓	★	★	✓	★
Input field type=”text”	✗	✗	✗	★	★	★	✗
Minimal form	✓	✓	✓	★	★	✓	★
Saved HTTPS, Broken HTTPS	✗	✗	✓	★	★	✓	★
Saved HTTPS, HTTP	✗	✗	✗	★	★	★	★
Invisible login	✓	✓	✓	★	★	★	★
Cross-site redirect	✓	★	✗	★	★	✓	✗
Same-origin IFrame	✓	✓	✓	★	★	★	✓
Cross-origin IFrame	★	★	★	★	★	★	★

✓ = Password manager automatic autofill page onload
✗ = Password manager does not autofills page
★ = Password manager maunally autofills or requires user interaction

5 Evaluation

Our analysis demonstrates that both browser-based and extension-based password managers provide a secure and convenient solution for users that use password-authenticated websites regularly. Overall, users that require a convenient tool that generates random and secure passwords, stores, and autofill them when needed can use any of the password managers we looked at. However, the results demonstrated a difference in the standard of security and usability in browser-based and extension-based password managers. Understandably, extension-based password managers have password-management software as their main product, while browser-based password managers have password management as an additional feature to their larger product, browsers.

5.1 Communication with Developers

We have responsibly disclosed the security vulnerability of password managers, MF, and DL automatic autofill webpages with invalid HTTPS certificates, as mentioned in Table 12 and Section 4.4.3, to the developers and have received acknowledgements.

The issue is currently still open on Mozilla’s bug tracking system [39] and Mozilla developers have commented on the issue. Mozilla has been considering removing automatic autofill altogether, but the special case we reported is worth considering if they do decide to keep automatic autofill. Below are the original responses to the issue:

Sergey Galich (Lead Engineer, Password Management at Mozilla Corporation): Thank you for filing this bug. We are considering to disable autofill. It’s adding just a bit of convenience at the larger cost of security. Once bug 1427543 is resolved, the issue you describe will go away.

Daniel Veditz (Security Lead at Mozilla Corporation): We have known for a long time that autofill is risky in general (the link you’re following might contain an XSS payload, for example) and there’s several bugs on that. ”After a certificate error override” is a special case, but certainly one worth considering if we weren’t already trying to remove the entire foot-gun. (We might have a duplicate of this, too.)

DL’s security engineer Samuel Gravis has responded to our email and stated that Dashlane is aware of this vulnerability and is already working on releasing a fix.

5.2 Recommendations

Browser-based Password Managers

Based on the three browser-based password managers we looked at, the settings available for the three features, password generation, password storage, and password autofill, are very limited. The lack of settings for users to correctly configure their password managers inhibits the ability of these password managers, making more technically proficient users to look for alternatives. We recommend browsers adopt stricter default autofill policies by switching to manual autofill as the default and adding options for password generation like password composition and length.

The possibility of starting a partnership between browsers and password manager companies to integrate extension-based password managers into browsers would be mutually beneficial for both parties. Extension-based password managers would have access to a larger customer base and browsers would be able to have a more complete password management solution for their users.

Extension-based Password Managers

Overall, all the extension-based password managers are safe to use with a few minor improvements that can be easily implemented to further improve the password managers. We recommend LastPass implement a password filter for their password generation to remove any weak passwords before using them. This idea was first introduced by Oesch and Ruoti [5] and all other password managers we evaluated have implemented it.

5.3 Future Works

Our analysis only covers a specific type of password manager from an individual user’s perspective. With the increasing adoption of password managers, future work could include security analysis of:

Mobile-based Password Managers

Our analysis only looked at password managers that interact with browsers, but password managers can be installed on a desktop or mobile phone for local applications. Mobile-based password managers can be developed by manufacturers of smartphones like Samsung and Google that are built into their smartphones.

Advanced Features

More advanced features like password sharing and file storing are designed for business use cases that require more than one user, making the security analysis complicated and difficult to execute. However, the security analysis of these features is important as a security flaw in these features has the potential to cause more damage to an organisation.

Memory Forensics

Volatile memory forensics methodologies and tools can be used to examine the artefacts left in the main memory by password managers. Memory forensics has been used to evaluate private browsing before and there have been security vulnerabilities found exposing users' private browser activity [40], [41]. We have replicated the methods used by private browsing memory forensics but found no evidence of artefacts of passwords but found user information like emails that are stored in their local database files.

More Advanced Autofill Analysis

Websites use redirects or cross-origin iFrames to allow accounts to be linked to social media or emails (Open Authentication). Our autofill analysis tests for the generic iFrames autofill policies, same and cross-origin iFrames. Future analysis could include more complex autofill tests for iFrames.

6 Conclusion

In this project, we conclude that all seven password managers we examined provide a secure and convenient solution for users that value their online security and users can use any of the password managers we looked at with confidence. With password managers becoming more prevalent, password managers need to ensure that users are getting what they expect from them, which is security and convenience. Users need to take the initiative to understand password managers to correctly use and configure a password manager that can provide them with both security and convenience. Therefore, we believe it is important that users, researchers, and developers continue to work together to contribute to the improvement of future password managers by balancing the security and usability of password managers.

References

[1] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *The Psychological Review*, vol. 63, no. 2, pp. 81–97, March 1956. [Online]. Available: http://www.musanim.com/miller1956/

[2] M. Zviran and W. J. Haga, "Password security: An empirical study," *J. Manage. Inf. Syst.*, vol. 15, no. 4, p. 161–185, Mar. 1999. [Online]. Available: https://doi.org/10.1080/07421222.1999.11518226

[3] R. Wash, E. Rader, R. Berman, and Z. Wellmer, "Understanding password choices: How frequently entered passwords are re-used across websites," in *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*. Denver, CO: USENIX Association, Jun. 2016, pp. 175–188. [Online]. Available: https://www.usenix.org/conference/soups2016/technical-sessions/presentation/wash

[4] E. W., "What does the ncsc think of password managers?" [Online]. Available: https://www.ncsc.gov.uk/blog-post/what-does-ncsc-think-password-managers

[5] S. Oesch and S. Ruoti, "That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2165–2182.

[6] D. Silver, S. Jana, D. Boneh, E. Chen, and C. Jackson, "Password managers: Attacks and defenses," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 449–464. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/silver

[7] "Bitwarden security whitepaper." [Online]. Available: https://bitwarden.com/help/article/bitwarden-security-white-paper/

[8] Z. Li, W. He, D. Akhawe, and D. Song, "The emperor's new password manager: Security analysis of web-based password managers," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 465–479.

[9] "Intel acquires passwordbox, an award-winning digital identity manager," 2014. [Online]. Available: https://newsroom.intel.com

[10] P. Gasti and K. Rasmussen, "On the security of password manager database formats," in *ESORICS 2012: Computer Security – ESORICS 2012*, 09 2012, pp. 770–787.

[11] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 523–537.

[12] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean, and accurate: Modeling password guessability using neural networks," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 175–191. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher

[13] D. L. Wheeler, "Zxcvbn: Low-budget password strength estimation," in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. Sec'16. Usa: USENIX Association, 2016, p. 157–173.

[14] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password strength: An empirical analysis," in *2010 Proceedings IEEE INFOCOM*, 2010, pp. 1–9.

[15] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, "Passgan: A deep learning approach for password guessing," 2019.

[16] R. Mutalik, D. Chheda, Z. Shaikh, and D. Toradmalle, "Rockyou," 2021.

[17] M. Fagan, Y. Albayram, M. M. H. Khan, and R. Buck, "An investigation into users' considerations towards using password managers," *Human-centric Computing and Information Sciences*, vol. 7, no. 1, pp. 1–20, 3 2017.

[18] "Browser market share worldwide." [Online]. Available: https://gs.statcounter.com/browser-market-share

[19] "Download the new microsoft edge based on chromium." [Online]. Available: https://support.microsoft.com

[20] *1Password Security Design*. 1Password, 2021. [Online]. Available: https://1password.com/security/

[21] I. Lunden, "Dashlane taps jd sherman, ex-hubspot coo, as new ceo, as co-founder emmanuel schalit steps aside," Jan 2021. [Online]. Available: https://techcrunch.com/

[22] L. Corbett, "Lastpass celebrates 25 million users," Sep 2020. [Online]. Available: https://blog.lastpass.com/2020/09/lastpass-celebrates-25-million-users/

[23] "Lastpass: Free password manager." [Online]. Available: https://chrome.google.com/webstore/detail/lastpass-free-password-ma/hdokiejnpimakedhajhdlcegeplioahd

[24] B. Stock and M. Johns, "Protecting users against xss-based password manager abuse," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 183–194. [Online]. Available: https://doi.org/10.1145/2590296.2590336

[25] K. Spearrin, "The bitwarden command-line tool," 2018. [Online]. Available: https://bitwarden.com/blog/cli-tool-released/

[26] A. Steel, "Open sourced lastpass command line application now available," 2014. [Online]. Available: https://blog.lastpass.com/2014/10/open-sourced-lastpass-command-line-application-now-available/

[27] LastPass, "Lastpass cli," 2019. [Online]. Available: https://github.com/lastpass/lastpass-cli

[28] "Issue 53: No master password option," 2008. [Online]. Available: https://bugs.chromium.org/p/chromium/issues/detail?id=53

[29] "Issue 1397: Master password is missing," 2008. [Online]. Available: https://bugs.chromium.org/p/chromium/issues/detail?id=1397

[30] M. Edge, "Microsoft edge and chromium open source: Our intent," 2018. [Online]. Available: https://github.com/MicrosoftEdge/MSEdge

[31] "Bug 1559986: Add special characters/symbols to generated passwords," 2019. [Online]. Available: https://bugzilla.mozilla.org/show_bug?id=1559986

[32] *Dashlane: Security White Paper*. Dashlane, 2021. [Online]. Available: https://www.dashlane.com/security/

[33] *LastPass: Technical Whitepaper*. LastPass. [Online]. Available: https://support.logmeininc.com/lastpass/help/lastpass-technical-whitepaper

[34] A. Barr, "Microsoft edge password manager security," 2021. [Online]. Available: https://docs.microsoft.com

[35] M. Kelly, "New password security features come to firefox with lockwise," 2019. [Online]. Available: https://blog.mozilla.org/en/products/firefox/password-security-features/

[36] C. Rahalkar and D. Gujar, "A secure password manager," *International Journal of Computer Applications*, vol. 178, no. 44, pp. 5–9, 2019.

[37] "Privacy when using website icons." [Online]. Available: https://bitwarden.com/help/website-icons/

[38] "What do the security warning codes mean?" 2022. [Online]. Available: https://support.mozilla.org/en-US/kb/what-does-your-connection-is-not-secure-mean

[39] "Bug 1755669: Password manager automatically autofilling on broken https pages," 2022. [Online]. Available: https://bugzilla.mozilla.org/show_bug?id=1755669

[40] G. Horsman, B. Findlay, J. Edwick, A. Asquith, K. Swannell, D. Fisher, A. Grieves, J. Guthrie, D. Stobbs, and P. McKain, "A forensic examination of web browser privacy-modes," *Forensic Science International: Reports*, vol. 1, p. 100036, 2019.

[41] K. Satvat, M. Forshaw, F. Hao, and E. Toreini, "On the privacy of private browsing — a forensic approach," in *Revised Selected Papers of the 8th International Workshop on Data Privacy Management and Autonomous Spontaneous Security - Volume 8247*, 2013, p. 380–389.