Password Managers and Vault Application Security and Forensics: Research Challenges and Future Opportunities

Aleck Nash and Kim-Kwang Raymond Choo(✉)

University of Texas at San Antonio, San Antonio, TX 78249, USA
Aleck.Nash@my.utsa.edu, raymond.choo@fulbrightmail.org

Abstract. Password manager and vault applications can be used by users to select strong passwords as well as storing user credentials locally or in the cloud. Such apps have been studied by various security researchers, for example in identifying potential vulnerabilities and bugs, as well as proposing techniques to forensically recover artifacts of interest/relevance to an investigation, which is also the focus of this paper. Specifically, we review the extant literature on the security and forensics of password manager and vault applications with the objective of identifying existing limitations and challenges.

Keywords: Password Manager · Vault Application · Forensic Analysis

1 Introduction

Passwords continue to be the dominant method for online authentication, say in e-commerce or many other electronic services. It is generally recommended to choose long and complex passwords (e.g., combination of numbers, letters, and special characters) to minimize the risk of password guessing and brute-force attacks (Fei Yu & Hao Yin 2021). However, passwords that are efficiently long and random are often difficult to remember by most users. In addition, in our increasingly digitalized society, it is not realistic to expect users to have different long and complex passwords for each account. Even if such passwords are used, they are still vulnerable to phishing and harvesting attacks (Rui Zhao & Chuan Yue & Kun Sun 2013a, b). Due to our limited capacities for remembering large sets of long and complex passwords, a typical user would tend to re-use the same passwords across their different accounts web applications (Ben Stock & Martin Johns 2014).

Users can use password manager (PM) applications (apps) to help them select strong passwords, store passwords for different accounts locally on their device or in the cloud, and/or be entered into password-matching dialogs (Ben Stock & Martin Johns 2014). There are many available open-source and commercial PM apps for desktops, as browser plug-ins (e.g., built-in feature), and for mobile devices (e.g., mobile apps). (Rui Zhao & Chuan Yue & Kun Sun 2013a, b). Table 1 provides a summary of several popular PM apps and their features.

S. Goel and P. R. Nunes de Souza (Eds.): ICDF2C 2023, LNICST 571, pp. 31–53, 2024.
https://doi.org/10.1007/978-3-031-56583-0_3

Table 1. Summary of most popular apps, based on Google Play Store

Password manager	Number of downloads, if applicable (as of 12/30/2022)	Open-Source	Features		
			2FA	Auto-fill	Zero Knowledge
Keeper	10M+	X	x	x	x
LastPass	10M+	X	x		x
Bitwarden	1M+	X	x	x	x
KeePass	1M+	X		x	
Dashlane	5M+		x	x	x
RoboForm	500K+		x	x	x
1Password	100K+		x	x	x
Passbolt	5K+	X	x	x	
PassPack	NA		x	x	
Encryptr	NA				x
Padlock	NA	x			

While there are many benefits in using such apps, the latter is also a single source/point of attack and any vulnerability in these apps (or their implementation) could have a cascading consequence. For example, the compromise of the master password for the app could reveal all passwords for many different accounts stored in the app (Chaudhary et al., 2019). The master password is generally 'transformed' into a key with many iterations of OBKDF2 key-generating algorithms, and the key is recomputed whenever the master password is entered. Another commonly used key generation approach is PassPack, which uses its own custom key generation algorithm that involves taking every other byte of a SHA-256 hash and doing several rounds of salted hashing. The master key is used to encrypt the account data to be stored. Most password managers use AES-256 to encrypt the vault, and others such as MSecure use Blowfish-256 (Elizabeth Walkup, 2016).

Vault apps (VAs) are mobile apps that store and hide sensitive files such as documents, photos, and other apps on the user's phone. The increased use of mobile devices that carry sensitive data reinforces the importance of protecting such data by saving them in the VA. However, studies such as those of (Gilbert & C. Seigfried-Spellar (2022), suggested that about over 30% Android vault apps stored passwords in cleartext, while others did not encrypt the photos. Such apps can also be used to store incriminating data such as terrorism and child abuse materials. In other words, both PM and vault apps are an invaluable source of evidence in digital investigations.

Hence, in this paper, we will review the extant review on password managers and vault applications, specifically focusing on the security and forensics aspects.

2 Research Methodology

In our literature review, we searched for and located articles using Google Scholar and other academic databases (e.g., IEEEXplore, ACM Digital Library, ScienceDirect and SpringerLink) using keywords and operators such as forensics AND "password managers", forensics AND "password vault", and vulnerability AND ("authentication app" OR "password manager").

Although during the initial search we located hundreds of papers, a closer read suggested that many of these papers are not relevant and subsequently excluded from our review. In the second pass, a total of 100 papers were downloaded and scanned more closely for relevance to our study. This yielded a total of 33 papers selected for the literature review. We then decided to expand the scope of the review by including articles related to user behavior and secure app development, of which we found 7 of them in the initially downloaded papers. This yielded a final total of 40 papers selected for our review.

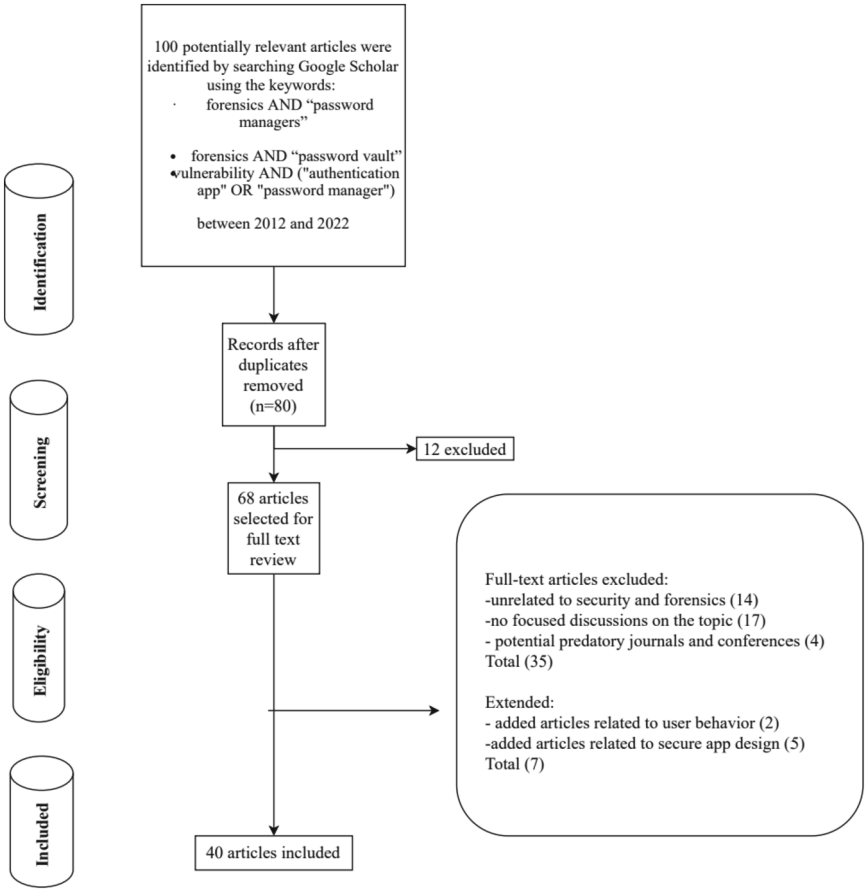

Fig. 1. Paper selection methodology

We also located one other literature review paper with overlapping focuses. Chaudhary et al., (2019) conducted a systematic literature review of 32 articles that dealt with security and usability problems related to PMs. The goal of the review is to provide guidance for PM developers, focusing specifically on the (in)security aspects. In our paper, however, we focus not only on the security of PM and vault apps, but also on the forensics aspects of these apps (Fig. 1 and Table 2).

Table 2. Summary of literature review articles

Study	Number of Papers	Scope
Chaudhary et al., (2019)	32 papers published between 2006 and 2015	A systematic review of the security and usability issues of PMs, specifically targeting PM developers
Our study	40 papers published between 2012 and 2022	A literature review of the security and forensics aspects of PMs and vault apps

In the next section, we will introduce the review of the security literature relating to PMs and vault apps.

3 Security Vulnerabilities

Here, we will discuss the existing literature relating to the potential attack types, the varying level of attacker expertise, and the adoption barriers.

3.1 Attack Types

A summary of the attack types is presented in Table 3, followed by detailed discussions in the subsections that follow.

3.2 Phishing, Clipboard Attack, and Keylogging Attacks

To examine the security of password managers, it is important to study the functionalities of these applications as well as their security policies. Silver et al., (2014), for example, conducted a study that examined ten browser built-in, third-party, and mobile password managers. These password managers include Chrome, Safari, LastPass, 1Passowrd, KeePass, and Keeper. All examined password managers offer either automatic or manual autofill functionality. The former functionality (automatic autofill) populates the username and password fields without requiring user interaction, and manual autofill requires user interaction prior to auto-filling by requiring the user to click the username field for example. In addition, all examined password managers allow password autofill pages within the same domain as pages from which the password was initially saved.

Table 3. Summary of attack types

Attack Types	Potential Attack Methods	Key vulnerability(ies)
Phishing	Sweep attacks Illegitimate/fake app Read-write Trojan-horse	Attackers' ability to access stored information utilizing the auto-fill functionality
Clipboard	Extracting data from the clipboard	
Cross-scripting	Cross-site scripting against the browser	
Key-logging	Clipboard and login page key logging	The ability of the attacker to capture keystrokes or information input by the user. While this seems to contradict the above observation, implementing the auto-fill functionality may prevent such an attack
Offline Brute Force	• Brute-force the master password • Automated brute-force against the master password • Brute force against authentication pin	The ability of the attacker to try different password combinations against locally stored password
Local decryption	• Monitoring HTTPS requests • Calling decryption functions	• Storing credentials without encryption • Transmitting in plain-text
Sensitive data extraction	• Extracting saved passwords from password managers using automated tools along with web crawlers	• Not implementing SSL • exposing secure login forms on HTTPS pages

This feature can be exploited by an attacker to target the password manager on the least secured page within the domain. Additionally, all password managers except Chrome are found to be vulnerable to iFrame sweep attacks, where a hotspot landing page contains invisible iFrames that point to random pages at various target sites. The attacker controlling the router (e.g., man-in-the-middle attacker) can inject a login form and JavaScript into each iFrame loaded by the browser. As each iFrame loads, the password manager will automatically populate the corresponding field with the user's password, which allows the injected JavaScript in each iFrame to steal these credentials. Another type of this sweep attack is called the Window sweep attack, in which windows are utilized to trick the user into disabling the popup blocker. Consequently, such an attack allows the landing page to open all of the victim pages each in a separate window. The JavaScript injected in each page will steal the passwords and hide the pages' content to

make them appear as blank windows to the user as well as closing them immediately once the passwords are stolen. The authors found that nearly all the automatic autofill password managers they examined, with the exception of LastPass, are vulnerable to window-based attacks. They also uncovered vulnerabilities that allow a public wi-fi network attacker to leverage the policies of password managers to steal stored passwords without the user's awareness.

Carr and Shahandashti, (2020) analyzed the security of five popular commercial password managers (i.e., LastPass, Dashlane, Keeper, 1Password, and RoboForm), where they found both 1Passwore and LastPass to be vulnerable to a phishing attack as a result of weak matching criteria used to determine which saved credentials to recommend for autofill. This allows a malicious program to masquerade as a lawful one by using the same package name. The authors developed a malicious app with the package com.google and a login page that looks like the official login screen. When this malicious program is started, LastPass offers to autofill the login page with Google credentials stored in the user's vault, because LastPass' matching criteria are based solely on the application package name. When the autofill feature fails, password managers frequently provide the option to copy credentials to the clipboard. Apart for 1Password, the authors discovered that the evaluated PMs do not provide sufficient security while copying sensitive things to the clipboard. Although locking the system would reduce such risk, Windows 10 permits access to the clipboard of a locked machine, which allows an adversary with physical access to the machine or an application running on the machine to paste the value of the clipboard in plaintext. Even if the attacker is unaware of the account linked with the stolen password, they can test the credentials against a pre-compiled list of websites where autofill is known to fail.

Similarly, Aonzo et al., (2018) analyzed the auto-fill mechanism of Android password managers. Their strategy was to install a genuine Facebook app and enter the credentials into a password manager. Then, they removed the real app, deleted the legitimate app, and installed malicious software with the same package name as the legitimate Facebook app. The purpose of that is to check whether the auto-fill mechanism of the PM would suggest the credentials of the legitimate app. Then, if susceptible, they built a mechanism to exploit the vulnerability of a certain mapping implementation. The approach was successful for a number of PMs such as Keeper, Dashlane, LastPass, and 1Password.

Luevanos et al., (2017) were able to perform an attack that enabled them to capture a live copy of the user's session, which allowed them to edit coding lines and simulate the login screen as if it was real. This gives a potential for a phishing attack that allows the attacker to steal the master password of the user. The authors performed a custom user-script attack by hiding malicious code in a program that a user can install to get additional website functionalities. Because users rarely read all of the code, dangerous code lines might be hidden in the script of such a program. A good way to carry out this attack is to conceal a decent user script in an extension that injects JavaScript into the page. Some PMs, like Passbolt, contain a mechanism that prevents the extension from being edited by detecting any modifications and disabling the app if any modifications are found. However, installing a phony extension would bypass such feature because open-source extensions are easy to mimic. They authors also discovered that Passbolt is difficult to use and has no published security audits. Additionally, several major features

seen in other password managers are missing from this program. The design of Passbolt makes it highly vulnerable to attacks on the main administrator. Encryptr was found to be particularly vulnerable to clipboard attacks since copying is the most convenient way to send data from the app to any form. The authors wrote a script that can reset any user account with a click of a button, which would subsequently delete all saved data such as passwords and other sensitive information. Padlock was also found to be vulnerable to clipboard attacks since it uses the clipboard without an autofill feature. Encryptr and Pablo were also found to be vulnerable to keyloggers when typing the master password and copying a password from the app to a form.

Gasti and Rasmussen, (2012) analyzed the storage formats used by 13 popular password managers by creating two adversaries, one which has read access to the password database, and another that has read-write access. The analysis of the selected password managers revealed that all of the examined PMs with the exception of PasswordSafe V3 are vulnerable to read-write attackers. Although PasswordSafe V3 was found to be secure in the authors' adversary and system mode, it had some interesting design flows. Li et al. (2014) conducted a comprehensive security analysis of five popular PMs (LastPass, RoboForm, My1login, PasswordBox, needMyPassword) and found them vulnerable to a threat model that allows the attacker, who controls one or several web servers and DNS domains, to trick the user into visiting the domains controlled by the attacker. Similarly, Zhao & Yue, (2013a, b) analyzed the vulnerabilities of existing browser-based Password Managers. The basic threat model they used allows the attacker to download malware, such as Trojan horses, temporarily on the victim's machine. Such malware can help the attacker collect the login information stored by the PM. The analysis of the source code of the examined PMs revealed that Firefox and Opera encrypt user passwords with three key triple-DES algorithms. The encrypted credentials are then saved into a database file and The triple-DES keys are saved into a binary file. The authors were able to verify that the attacker can simply decrypt the credentials by stealing both files.

3.3 Cross-Site Scripting Attacks

Stock & Johns (2014) assessed the potential threat of cross-site scripting attacks against browsers' PMs by analyzing the current features of password generation and password fields available in popular websites (Google Chrome version 31, Mozilla Firefox version 25, Opera version 18, Safari version 7, Internet Explorer version 11, and the Maxthon Cloud Browser version3). They discovered that among all of the examined web browsers, only Internet Explorer fills forms on any area of the same web application while the application bounds are controlled by the same Origin Policy. The Maxthon Cloud Browser only populates credentials when the same second-level domain is accessed, which bypasses the protocol and the port resources and allows the attacker to steal passwords from its storage. With respect to matching criteria, most browsers were found to be very relaxed. All examined browsers with the exception of Internet Explorer fill in the passwords only if they match the origins, while Maxthon Cloud Browser only considers the second-level domain. In such cases, the attacker can design a simple form to deceive the PMs into revealing the stored password from sites where two input fields are used for the login dialogue.

Oesch and Gautam and Ruoti (2021) evaluated the mobile autofill frameworks of iOS and Android devices by examining their capability of achieving significant improvements over desktop frameworks. The authors selected 14 mobile PMs for framework evaluation, and the three frameworks are iOS password autofill, iOS app extensions, and Android autofill service. The authors found that all frameworks require user involvement prior to autofill, and that iOS password autofill feature completely secures the autofill process for native UI app elements. Nevertheless, all mobile password manager frameworks fail to validate credential mapping accurately, and none of them successfully secure filled credentials, which makes them less secure than the weakest desktop password managers in many cases.

3.4 Brute Force and Local Decryption Attacks

Because the master password is so crucial in the concept, it must be strong in order to safeguard users' sensitive information including other passwords, files, and notes. Yu and Yin (2021); Zhao and Yue and Sun (2013a, b); examined the authentication mechanisms of well-known commercial PMs such as 1password, LastPass, and KeePass. They found that the authentication mechanisms of 1password, LastPass, and KeePass all use the same model that includes the master password or username as well as some database properties such as the KDF parameters. Yu and Yin (2021) used the brute force approach to crack the master password on the PMs, using a GPU platform (Graphic Processor Unit) due to its high parallelism and low energy consumption. They discovered that cracking the master password is far more difficult than cracking a basic hash like mdf($password). 1password, LastPass, and KeePass were found to have the same security level (Yu and Yin, 2021). However, LastPass's master password mechanism has serious security flaws. For example, the user is automatically authenticated when LastPass is re-used. This is because the master password is saved into an SQLite database table named LastPass-SavedLogins2. Regardless of encryption, locally stored master passwords are subject to local decryption attacks, which can be carried out by external attackers who have advanced computational and/or client-side stealing capabilities. Even if Lastpass does not store the master password locally, it is still vulnerable to brute-force attacks by external attackers. Internal attackers who are capable of server-side monitoring can also launch brute-force attacks on the master password (Zhao & Yue & Sun, 2013a, b). Similarly, Keeper, 1Password, and Dashlane are vulnerable to UI-driven brute force attacks. There was no defense mechanism in place to stop the authentication process after 10 failed login attempts, which implies that a dictionary attack on a user's account is feasible (Carr & Shahandashti, 2020).

The techniques utilized by PMs to implement autosave, autologin, and autofill functions vary greatly. Although the terms imply automated activity, these functions frequently require user participation. After a login submission, PMs usually autosave credentials by prompting the user after a login submission. In some applications, such as 1Password, Norton, and Blur, this functionality can be activated or removed via the settings. Other applications such as Passbolt, do not have an autosave function. Some PMs save credentials automatically when the login page loads, while others require user involvement like clicking a button. An in-between approach is utilized by Keeper

where the user is asked to manually decide on whether credentials should be auto-filled (Huamanc et al., 2021). Some PMs allow the user to establish a four-digit PIN to ease authentication to Android applications, which eliminates the need to input a complex and lengthy master password when entering the vault. RoboForm and Dashlane Android applications do not implement a permanent counter on the number of times an invalid PIN can be entered to access the app. Both applications (RoboForm and Dashlane) employ a four-digit long PIN and thus have 10,000 combinations. The manual testing of these applications reveals that a randomly selected pin can be cracked in less than three hours (Carr and Shahandashti, 2020). When the master password is not used, RoboForm provides no security to local storage. The user's website credential is not encrypted, but rather encoded without the usage of a cryptographic key. Therefore, external attackers with client-side stealing capabilities can obtain the file containing the website credentials of users who do not use master passwords. The attackers can then call the function RfUngarbleStringW() to retrieve the website's credentials entirely. Additionally, external attackers can still launch brute force attacks when the master password is used. Furthermore, RoboForm does not encrypt source code and traffic, instead, they are transmitted in plain text through HTTPS communication. Although HTTPS encrypts client-server communication to prevent MiTM attacks, inside attackers can get all credentials by monitoring incoming HTTPS requests and waiting for them to be decrypted on the server side. (Zhao and Yue and Sun, 2013a, b).

Oesch and Ruoti, (2020) evaluated the security improvements of thirteen popular password managers compared to their security prior to 2020, taking into account the three stages of the password manager lifecycle, password generation, autofill, and storage. They found extension and application-based password managers had improved since they were analyzed in previous years. They have addressed specific vulnerabilities related to securing metadata stored in password vaults as well as preventing password harvesting attacks by limiting auto-filling capabilities. In terms of security and functionality, browser-based PMs were found to lag behind extension and application-based PMs. Users must still verify that app and extension and application-based PMs are correctly configured, as neither LastPas nor Dashlane require user participation before auto-filling website passwords. Chatterjee et al., (2015) examined the development of vault applications that can prevent offline cracking attempts. The purpose of their "Kamofulage" system was to force the attacker to try login with a password from each of the vaults to identify the actual one. The authors aimed to discover a vulnerability in this approach and developed an attack to exploit it. They developed an ML-based cracking attack and used a number of datasets to train and test the model. They primarily used three large-scale password leaks which are Rock.You, Myspace, and Yahoo. They also simulated Kamouflage and found two vulnerabilities and their attack and exploited them. They were able to demonstrate how conventional techniques for cracking-resistant vaults actually weaken security through the attacks. The main strategy of Kamouflage, which is to store a list of cipher texts with one genuine vault buried amid several decoys created as a function of the real master password, offers attackers with an offline speedup.

3.5 Crawling, and Sensitive Data Extraction

Gonzalez & Chen and Jackson, (2013) presented a tool named Lupin that automatically saves passwords in password managers without the contest of the user. A network adversary can use the tool to extract credentials if the login form is on a non-HTTPS website. The authors used a web crawler that surveyed 45000 popular websites from Alexa's top website list. They found that more than 25% of these sites are vulnerable to such attacks. Examples of these vulnerable sites include Facebook, LinkedIn, and Twitter. It was also observed that about 12% of surveyed websites implemented SSL and about 27% exposed secure login forms on HTTPS pages. Stock and Johns (2014) performed a crawl of the top 4000 sites from Alexa. They discovered that less than 300 of the 2,143 examined domains do not utilize autocomplete characteristics, which prevents password managers from saving the credentials in the password manager's store.

3.6 Attacker's Level of Expertise

Ruffin et al., (2022) performed an empirical analysis of 20 popular vault applications to identify the vulnerabilities and expertise level required to exploit them. The expertise level consists of three categories, novice, intermediate, and expert. The beginner level reflects an attacker with little to no technical understanding of Android systems, which can only help in determining whether a vault program is installed on the mobile device. The intermediate level reflects an attacker who is familiar with the Android system and can access secret files saved by an app. The expert level reflects an attacker which advanced knowledge of Android systems, which allows the attacker to effectively use adp (Android Debug Bridge) to retrieve files and decompile apks (file format or distribution and installation used in Android). The authors determined that the majority of the tested vault applications may be located using noise level analysis by evaluating the program names and icons and matching them with other apps in the app library. They also found that an adversary with an elementary knowledge of Android systems can recover private files from 15 out of the 20 apps easily without rooting the mobile device. They also discovered that an attacker with a novice level may readily recover private files from about 75% of the apps without rooting the mobile device. It was also discovered that 5 apps' files cannot be recovered without root. Advanced-level adversaries can root the victim's mobile device and retrieve the data for two of the applications since these applications simply hide the contents by altering the file extension and adjusting the header bytes. Additionally, only 5 out of the 20 apps were found to have some encryption methods to protect the stored files. One of the common approaches is storing the files in a hidden folder. The authors, however, believe that this strategy is not safe since attackers may simply enable the option to view hidden files.

3.7 Adoption Barriers

The potential risk of losing all passwords is a significant barrier to using password managers. Some users are concerned that service providers would be able to access their passwords. A comparison of security and privacy issues reveals that security-related

concerns outweigh privacy-related concerns in terms of adoption. The adoption of password managers can also be predicted while observing the dissatisfaction with the way passwords are handled without the use of PMs. This dissatisfaction mainly stems from the expense of having to memorize or save these credentials on notebooks or smartphones, which means that users need to look up the password every time they need to log in. Users' opinions of the benefits of PMs compared suggest a willingness to adopt this technology. Thus, it is critical to stress the benefits of using PMs in order to influence their adoption. Furthermore, the perception of the efficiency of PMs in protecting users' credentials also influences the adoption of this technology (Alkaldi & Renaud, 2022). The majority of issues in existing PMs relate to some features and functionalities such as optional auto-fill, and weak master passwords, while other issues can be resolved easily if the developers remain cautious and perform tests and trials with the established security methods, algorithms, and principles to correctly implement them. Additional unresolved challenges related to users' suspicion of PM software include the gap between the user and the developer, and improper web design practices, all of which are time-consuming and difficult to overcome. (Chaudhary et al., 2019).

Oesch et al., (2022) interviewed users of over 20 different Password Managers to identify how they practically utilize their password manager and to investigate underused capabilities and unexpected consumption behavior. They discovered that experts frequently use multiple password managers, which includes using browser-based managers such as Chrome in addition to an external manager such as LastPass. This practice was inspired by three important considerations. The first was a concern about losing passwords saved in the external manager, thus the user browser's manager was utilized as a backup. The second is that users are used to clicking through dialogue presented by the browsers, and the third is that participants utilized both managers such that if one failed to autofill a certain webpage, the other could. Although an unwillingness to use password generation was observed, there are two primary reasons behind that. The first is that participants often find generated passwords to be difficult to input if not auto-filled by a password manager, and the second is that participants worry that their manager fails to locate the generated password and subsequently locks out their accounts. While some users are more concerned about convenience than security, most users find breach notifications helpful and that they are willing to update their breached accounts' passwords. Participants favored Chrome's account breach notifications over those offered by external managers. Participants in the study, on the other hand, disliked the credential audit service offered by many external managers. It was also observed that some PMs fail to identify login forms and modify credentials or save them. Participants indicated that such issues are widespread. Additionally, mobile PMs usability is a big barrier to their use and adoption in the real world.

Michael Fagan et al., (2017) performed an online survey of 137 users and 111 non-users to evaluate differences between PM users and non-users, which included questions on emotion given its psychological significance when making decisions. The authors found that users of PMs are likely to have greater computer skills, better password managers, and better computer security in general. To these users, password managers are useful and convenient. Non-Users, on the other hand, are likely to rate their computer professionally and have fewer accounts. These non-users do not view PMs as an effective

way to protect their accounts. While convenience was found to be important for many users, security concerns are the primary reasons why non-users do not utilize PMs. A combination of both quantitative and qualitative results suggests that non-users do not generally regard PMs as secure, which is not a true generalization considering that some PMs do provide good security if used properly.

4 Forensic Analysis

(See Table 4).

Table 4. Summary of artifacts across different apps

Source of evidence	Type of artifacts that could potentially be recovered	Reasons
Device-level (Main Memory)	Credentials (stored password's hash value and pin)	Credentials stored in plain-text Screen lock leaves the pin in the memory
Device-level (Login Screen)	Accessible files once the login screen protection is bypassed such as photos, videos, notes, and logged-in accounts	The ability of the investigator to make unauthorized changes to security settings such as Disabling the Screenshot prevention mechanism disabling password field visibility protection Bypassing login security
App-Level (Database)	Recovery of Photo, video, and timestamps that provide an indication of various app-related activities such as installation and uninstallation	The key used for encrypting these files was hard-coded in the source code
Server-level (Cloud server)	Credentials (clear text passwords, authentication tokens, and pin)	TLS is not implemented, which allows credential discovery when performing MiTM attack

4.1 Memory Forensics

Sabev and Petrov conducted a series of studies on two popular Android PMs (Keeper and Bitwarden). In 2021, they Analyzed the runtime behavior of two popular Android Password managers/vault applications (Keeper and Bitwarden) from a security and forensics perspective. The authors conclude that Keeper gives a large time frame for possible

attackers to acquire the user's Master Password and data through inspecting Keeper's main memory. Moreover, Keeper retains any entrusted data in cleartext in main memory for the duration of its process. Unlike Keeper, Bitwarden's Master Password was only discovered in its _MD2. Additionally, Bitwarden only loads a tiny part of the data entrusted to it in clear text into the main memory. Based on this, Bitwarden is believed to take stronger security measures to restrict data exposure in main memory verses Keeper (Sabev and Petrov, 2021a, b).

Barten (2019) used File system-based and memory-based for client-side attacks on the LastPass extension for the Chrome browser. Because both file system and memory-based attacks take advantage of the local storage of the encrypted vault, the vault may be quickly decrypted if the password is memorized. Thus, disabling the offline access is arguably the best practice for the user. However, disabling offline access can only be achieved if a form of multi-factor authentication is enabled and cannot be disabled independently. The authors suggest that offline access is a security flaw since it enhances the opportunity to steal credentials through client-side attacks. The vault key, which is needed to open the vault in both file system and memory-based attacks, may be obtained through the Chrome developer tools. As a result, instead of inspecting the memory dump, an attacker may utilize the Chrome developer tools to retrieve the key. Today, the vault key is always available in memory and may be requested at any moment to decrypt the vault. Even when deleted, the key must be generated from the master password. As a result, the users may be repeatedly prompted for their master password in order to produce the key, which degrades the user experience. (Derk Barten, 2019).

Apostolopoulos et al., (2013) investigated the possibility of detecting authentication credentials stored in the volatile memory of Android mobile devices, using existing tools. They used a Dalvik Debug Monitor Server (DDMS), which is a graphical debugging tool used to examine the running process. They were able to recover their credentials in most of the examined applications because the credentials were stored in plaintext. They were also able to retrieve the passwords of five apps of major Greek banks and the username of four of these apps. In another set of experiments, the authors discovered a unique pattern indicating where the password is saved. For instance, they found the string "password " in the memory dump, which means that a forensic investigator or a malicious actor can look for such patterns to find passwords and usernames in the device. Most Android apps seem to be vulnerable to credential discovery. What is more disturbing is learning that web banking apps were also found to be vulnerable to credential discovery, which leaves their customers' credentials at risk. Similarly, Christoforos Ntantogian et al., (2014) conducted an experimental analysis to investigate the possibility of discovering authentication credentials from the volatile memory of 13 Android mobile devices, using open source tools such as Linux memory exactor (LiME) software, a loadable kernel module that allows recovering volatile memory from Linux and Linux-based devices. The authors were able to recover their own credentials in most of the examined applications as they were found in plaintext.

Gray and Franqueiraand and Yu, (2016) analyzed three popular password managers: KeePass (v2.28) Password Safe (b3.35.1) and RoboForm (v7.912). Their investigation showed that KeePass master password is saved in the hard drive within the page files when the computer capacity is low, which allows the identification of the password

in plain text using keyword search. For Password Safe, the keyword search showed that a copied entry (the password for instance) could be found unencrypted and saved in pagefile (C\fpagefile.sys). As for RoboForm, it was observed that it stores crucial information on the pagefile, which includes name, address, pin number, email, and credit card number. Lee & Chen & Wallach, (2019) selected 11 Android apps (including 4 password managers) to study the password retention problem. The study involved performing a full physical memory dump and per-process dump. The authors found that all tested apps are vulnerable to credential discovery as they successfully received passwords in plain text for all apps. The lock screen system also retains the PIN password inside the memory. Although Android spends a significant effort to protect the PIN password, however, the retention of the PIN password in the memory defeats the purpose of the added security measure completely.

Martini and Do & Choo, (2015) analyzed Universal Password Manager and other apps to determine what kind of sensitive information can be obtained. The authors were able to discover crypto classes instances of UPM extension, which allowed them to locate the 8-character salt used by the app to encrypt and decrypt the database that contains the user's data. Additionally, they were able to find an instance of the java class that contains the user's password in plaintext inside the memory. Because UPM does not directly interact with a server, the only ways to access data from its database are brute-force techniques or other means of acquiring the user's password.

4.2 Investigation of Current Encryption and Security Approaches

Fahl et al., (2013) analyzed 13 free and 8 non-free Android password managers. They found that ost analyzed PMs use Advanced Encryption Standards (AES) within several key lengths. Seven applications improve the security of encrypted credentials by deriving the symmetric encryption key from the user's master secret using a specific key derivation mechanism. Passwords with fewer than 16 characters get appended to strengthen the password. The Android standard browser does not provide encryption for passwords, but rather protects against unauthorized access. The mechanism of Android's Account Manager enables a centralized way to store credentials and provides protection from unauthorized access. Nevertheless, the accounts.db database lacks an additional layer of encryption, making the password vulnerable to forensic investigation. Furthermore, an attacker can discover the services for which a victim has accounts and which services use the same login credentials. This is because certain PMs store databases on the SD card, which is globally accessible without permission on all Android 4.0 and earlier devices. Additionally, the authors discovered that many app developers do not properly implement TLS, leaving their apps vulnerable to MiTM attacks. Although RoboForm and SecureSafe implement TLS, they failed to authenticate the cloud server's TLS certificates, and accept all certificates instead. But this has no security implication for SecureSafe because it implements a session-specific symmetric key during the SRP login to ensure end-to-end encryption for passwords.

Zhang & Baggili & Breitinger, (2017) analyzed 18 Android vault applications and investigated the forensic artifacts they generate. They discovered that the majority of developers used obfuscation and native libraries to safeguard their code. While this method disturbed the process of reverse engineering the apps, examiners can still exploit

these applications. The authors found that photos and videos entrusted in these applications may be undecrypted, encrypted, or deconstructed. While the photos and videos in about a third of the total anlyzed applications were changed or renamed, the examiner can easily recover such files since the data of the media files were not modified. On the other hand, a similar portion of the applications stored deconstructed files by erasing data from the header. Although about a third of the analyzed applications encrypt the photos, the encryption method was insecure because the information used for generating the key was hard coded in the source code. Furthermore, around one-third of the applications saved the password in cleartext on the Android device, while about one-half of the examined applications store the password hash value, allowing the authors to perform brute force and similar password-cracking attacks. Keeper used a salute password hashing function, while Coverme saved the hash value of the key generated by the password rather than the hash value of the password itself. Therefore, the authors regarded these two applications as more secure since they offered a more complicated password system or used a more robust hashing technique. The authors were able to easily recover hidden contents from all vaults with the exception of Keeper and Coverme, and concluded that Coverme offeres a higher chance for brute force attack because it limits the password length to 16 digits.

Sabev & Petrov (2021b) were able to remove password field visibility protection by modifying the security settings in keeper and increasing the auto-lock time. Furthermore, they manipulated the application record by altering the value of link between the fields. They were successful in making Bitwarden's vault constantly open and fully available by switching the vault time from 15 min to "Never". In a different study, Sabev & Petrov (2021a) used the SEC, which is "an extensive body of knowledge about software security that is widely accepted security recommendations and guidelines for building secure software", to evaluate LastPass, Bitwarden, and Dashlane based on public information about their security that is available in the official documentation. They found Bitwarden to provide the highest security among these applications. On the other hand, Laspass appeared to be the least secure app, while Keeper and Dashlane were found to be equal in terms of security. Btiwarden was found to provide better trustworthiness and transparency compared to the other applications. However, the results may change significantly if advanced analysis and debugging techniques were performed.

4.3 Content Hiding App Identification

Although content-hiding applications are considered a good approach for securely storing users data, yet, it can also be a perfect tool for bad actors to hide their illegal data. Peng et al., (2021) created a system that uses deep learning to detect content-hiding applications that can discover and retrieve targeted data from Android devices. The authors deployed a crawler to collect GooglePlay Store apps in order to acquire information about probable content-hiding applications. Their system was able to recognize every vault app that was active on a rooted device in addition to extracting data such as pictures, videos, and text. Although the system successfully recognized all vault apps for all third-party programs in the unrooted device, the extraction of the hidden data was only successful in one of the vault applications.

Gokila Dorai et al., (2020) developed a fast way of identifying vault apps using common keyword search based on Machine Learning-based binary classification to assess whether an app is indeed a vault application. They developed a system that can automatically extract hidden contents from each potential target. This system is called VIDE and is designed for iOS devices. Although existing data extraction tools are capable of extracting all data from smartphones, yet they frequently necessitate the user to categorize and filter the data. Thus, the proposed system can make the process easier for the investigator as it automatically identifies and extracts data from the set of vault applications. Similarly, Gilbert & Seigfried-Spellar, (2022) Conducted a forensic analysis on 5 iOS photo vaults (KeepSafe, Photo Vault, Calculator+, Secret Safe, and Purple photo vault). The authors found that each vault left evidence and images for the investigator and each of the forensic toolkits used in the investigation generated different scan findings. The forensic tools used are UFED Cellebrite (v.7.3), Magnet Axiom (v.3.8.0), and Black Bag Mobilyze 2019 R1. Mobilyze found the least amount of information from the vault applications and was only able to identify 3 out of 5 vault apps, which implies the importance of using more than one forensic software to provide the most correct evidence. Axiom was not able to locate every picture in the vault apps' camera roll, while Cellebrite did. Amongst the three applications, Cellebrite produced the most results and was the only tool to discover one of the passcodes from the "Photo Vault" of photos.

5 Design of Secure Password Managers and Vaults

Decades after inventing them, PMs are still subject to many attacks including online guessing (brute force), offline dictionary, and shoulder surfing. Some of these attacks rely on issues with the nature of the password itself, such as memorability. Users tend to pick unsophisticated passwords that are easy to remember, which carries the risk of easier attack, whereas safe random passwords expose alternative attack venues owing to memorability difficulties where users write such passwords down or store them electronically. Online password managers provide a solution for these problems by encrypting web accounts passwords and storing them in the cloud (Shirvanian et al, 2021). But this carries its own risks as these credentials stored in the cloud or locally are subject to massive password breaches. To mitigate the risk of storing passwords locally or in the cloud, Shirvanian et al., (2021) proposed a cloud-based password manager named HIPPO that does not memorize or save accounts or master passwords. The proposed application relies on the device-enhanced password key exchange cryptographic principle, which has been shown to be resistant to dictionary and online guessing attacks. When HIPPO receives the user's master password, it immediately produces a strong, random account password for each website without saving it. The randomized account password is generated using an obvious-pseud random function (OPRF) protocol, which is a crucial component of the master password. The user (the client) starts the authentication process with the master password, and the cloud-based password manager stores a different key for every service that needs authentication. By using this protocol (OPRF), the password manager is prevented from learning or storing the randomized account password or any passwords that are derived from them. The security of the proposed app is derived from the formal framework and security proofs of the Device Enhanced Password Authenticated Key

Exchange (DE-PAKE) protocol which offers many security benefits such as resilience to both online and offline attacks, and resistance to attacks upon compromised servers. A limitation of this approach is that it may be a target for guessing attempts to log in to the web server because it is implemented as an online service without requiring a user authentication. Y. Li et al., (2017) addressed the same issues by proposing a password manager named BluePass that stores the vault (all encrypted site passwords) locally in mobile devices and the decryption key to the vault on the user's computer BluePass separates the password for a decryption key from the password for the password vault, allowing the password vault to be kept locally while the decryption key is kept on the BluePass server. When the user needs to log in to a website account, the computer will automatically request the password associated with the website account from the mobile device. The computer will then utilize the local decryption key to decode the site password that was received through Bluetooth and automatically fill out the user's online forms. BluePass uses two-factor authentication as its model, requiring a master password and a mobile device to decrypt and retrieve site credentials. Given today's computing power and the weak password selection on the user side, the master password database can almost certainly be cracked by offline attacks. But since BluePass does not store passwords on the server, it provides security against massive data breaches, which can reveal both hashed master passwords and password vaults. However, since this model utilizes the auto-fill feature, all of the security issues related to auto-fill vulnerabilities discussed in the previous sections apply to this model. Additionally, severe security issues can occur when the user fails to capture the correct mental model.

Stobert & Biddle, (2014) Proposed a prototype for a password manager called Versipass that incorporates cued graphical passwords to address the issue of password memorability. According to psychological findings known as the "picture superiority effect", people are more likely to recall pictures than texts. Graphical passwords leverage this result by cueing graphical passwords for more memorable passwords. Verispass does not retain passwords; instead, it remembers picture clues for graphical passwords, which makes it function more like a password cue manager. The app only stores the password cues and enables users to securely create passwords and send them to websites. The password system sends a cue to the user, which helps the user to memorize the original password. This model does not only cue graphical passwords, but more generally, the cue may be visual or audio with the goal of cueing memory and providing context to the user. This approach eliminates the risk imposed by an attacker that can access the user account when accessing Versipass because the passwords are not stored in the app. However, such an attacker can still access personal information such as lists of sites and usernames, which can cause significant disruption for the user if the attacker deletes information or change the category password. Thus, despite the benefits of not remembering passwords, the app is still vulnerable to fishing, guessing, and capture attacks.

AlMuhanna and AlFaadhel and Ara, (2022) presented a system that provides resilience against brute-force attacks by implementing honey encryption (HE) in the password manager. In this system, the password manager hashes the user's inputted

password to produce a group of Honey words that closely resemble the original password. These honey phrases serve as a warning system to catch intruders. OTP encryption, which encrypts all application credentials, adds an extra degree of protection to websites and other application passwords stored in this password manager. The password manager will give the user three attempts to input their credentials before granting access if the provided hashed master password matches the one that is already saved. When an attacker types the incorrect key, the Honey encryption algorithm creates a genuine-looking false message while securely hiding the real message. In a way, the system analyzes the attackers' activity to identify and deceive them. When a collection of automatically created honey words is used to identify brute-force attacks, the system blocks access to the account if one of the words matches the master password that was input. The attacker will be directed to a phony account with fictitious passwords. One of the biggest problems with this method is that it might mistakenly identify a genuine user as an attacker if they make a typo that matches one of the honeywords. To address the same issue, Yuchen & Rui, & Wenchang, (2016) proposed an approach based on hardware trusted platform module (TPM) in order to improve the security of browser password managers. This method employs the master password as the authorization credential for permitted access to TPM and encrypts the user's passwords using keys generated by TM. The experiments performed by the authors show that this system can successfully protect against brute force and password-stealing attacks. One of the limitations of this approach is its inability to prevent keyboard recording attacks from stealing user-entered passwords (Table 5).

Table 5. Summary of design principles

Study	Characteristics	Benefits	Problem addressed	Limitations
Shirvanian et al., (2021)	OPRF: derived from the formal framework and security proofs of the Device Enhanced Password Authenticated Key Exchange	Proven to resist online guessing and dictionary attacks Does not store passwords	Massive password breaches of passwords stored in the cloud or locally by PMs	Susceptible to guessing attack against the web server
Y. Li et al., (2017)	Stores password vault locally while the decryption key is stored in the server of PM	Does not store passwords on the server	Massive data breaches which can leak both password vaults and hashed master passwords	Security issues related auto-fill may also apply to this model User can fail to capture the correct mental model
Stobert and Biddle, (2014)	incorporates cued graphical passwords as well as visual or audio with the purpose of cueing memory and providing context to the user	Remembers image cues for graphical passwords instead of remembering passwords Eliminates the risk of attackers accessing the user account	Password memorability and associating passwords with accounts	Susceptible to fishing, guessing, and capture attacks

(*continued*)

Table 5. (*continued*)

Study	Characteristics	Benefits	Problem addressed	Limitations
AlMuhanna and AlFaadhel and Ara, (2022)	Implements honey encryption in the password manager which hashes the password and creates a set of Honey words that mimic the real password	Provides resilience against brute-force attacks Detects brute force attempts Misleads the attacker by analyzing their behavior	Brute Force attack	Incapable of distinguishing between a brute force attempt and a typographical error by a legitimate user
Yuchen and Rui, and Wenchang (2016)	based on hardware trusted platform module (TPM) that encrypts the user's passwords with keys generated by TM which uses the master password as the credential for authorized access to TPM	Defends against password stealing and brute force attacks	Brute Force attack	Incapable of protecting against key loggers

6 Discussion and Future Research Directions

Username and password are still, by large, the predominant method of authentication. Although password managers provide many benefits to the user by generating sophisticated random passwords and storing user passwords locally or in the cloud, there are important barriers to the adoption of these applications. In this research, we observe that security concerns are important barriers to the adoption as not all users view password managers as a secure method of storing their credentials. Many users have concerns about the security of these credentials stored in the cloud or in the application's server. These concerns can be confirmed by existing research that finds many security problems with PMs. Most of the problems in existing PMs are related to features and functionalities of PMs such as weak passwords and auto-fill that allow attackers to perform a variety of attacks including phishing, brute-force, cross-site scripting, and local decryption. In this research, we observed how critical the autofill feature is for password managers. In addition to the convenience it provides for the user, some researchers found that PMs that use clipboard without implementing auto-fill can be vulnerable to key loggers and clipboard attacks (Carlos Luevanos1 et al., 2017). On the other hand, the autofill feature can allow fishing, clipboard, iFrame sweep, and cross-site scripting attacks (Silver et al, 2014); (Carr and Shahandashti, 2020); (Aonzo et al, 2018); (Stock and Johns, 2014); (Oesch and Gautam and Ruoti, 2021). One of the ways Password managers can prevent that is by never auto-filling under certain conditions, and requiring user interaction through some form of trusted browser UI that cannot be affected by untrusted JavaScript. Password managers should also provide an option to clear the clipboard after a set amount of time (Carr and Shahandashti, 2020). It was observed that there was no strict enforcement of a strong master password in a number of PMs (Luevanos et al., 2017).

On the forensics side, the remanence of sensitive data such as passwords, in the main memory is still an issue, as a number of researchers were able to obtain authentication credentials from the volatile memory. Others were able to find the master password and other passwords in plain text. Additionally, most PMs and VAs do not protect the password with an extra encryption layer from forensic analysis. Many developers use obfuscation and implement native libraries to protect their code. Although that hindered reverse engineering these applications, it did not protect them from other types of exploitation performed by the examiners.

With respect to the secure design of password managers, the approaches we observed can be valuable in protecting against brute-force and credential-stealing attacks. The major theme among these studies is to solve the problem of storing passwords locally, on the app server, or in the cloud which can lead to massive data breaches. However, these approaches do not address other common vulnerabilities such as those related to auto-fill, clipboard, and keylogging.

It was observed that researchers make their selection of password managers/vault applications for security analysis based on popularity which is determined by the number of downloads from google play, and the vast majority of them are open-source. Although being an open source allowed people to find vanuralites, not everyone reports the vulnerabilities they discover. Thus, attackers can choose to keep these vulnerability secrets and exploit them in the future. In addition, a number of these open-source password managers lack many features that closed-source password managers have. Since such features strengthen the security of the applications, more effort is needed to investigate other closed-source password managers.

Some studies such as Christoforos Ntantogian et al., (2014) did not mention the apps they investigated. As a result, we were unable to verify the relevance of their results to the popular PMs often investigated in other studies. We also make similar observation to that of Chaudhary et al., (2019) as they mentioned that their selected papers often focused on the engineering perspective of PM design. They also mention that their study was not able to establish any particular importance of the security features and usability described, due to the different approaches of the password managers reviewed such as cloud-based and wallet based.

A limitation of this study is that the majority of the reviewed papers were conducted prior to 2018, which means that some of the vulnerabilities might have been patched in the newer versions of the PMs. Although Oesch & Ruoti, (2020) suggest that app-based and extension-based password managers have improved since the time they were analyzed in previous years, we did not find a comprehensive comparative analysis that covers all of the improvements in all of the analyzed apps over the years. But Oesch & Ruoti, (2020) also mentioned some remaining issues related to autofill in which the client is vulnerable to password harvesting attacks, which suggests that some security issues remain unresolved.

7 Conclusions

In this paper, we conducted a comprehensive literature review on password managers and vault apps, focusing on both security and forensics. We highlighted key security issues related to these applications based on the existing literature. We also reviewed studies that discussed user behavior towards these applications and the existing approaches for designing secure password managers. We hope that this work will inform future design of password manager and vault apps.

References

Alkaldi, N., Renaud, K.: MIGRANT: modeling smartphone password manager adoption using migration theory. ACM SIGMIS Database: DATABASE Adv. Inf. Syst. **53**(2), 63–95 (2022)

AlMuhanna, A., AlFaadhel, A., Ara, A.: Enhanced system for securing password manager using honey encryption. In: 2022 Fifth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU). IEEE (2022)

Aonzo, S., et al.: Phishing attacks on modern android. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (2018)

Apostolopoulos, D., Marinakis, G., Ntantogian, C., Xenakis, C.: Discovering authentication credentials in volatile memory of android mobile devices. In: Douligeris, C., Polemi, N., Karantjias, A., Lamersdorf, W. (eds.) I3E 2013. IFIP Advances in Information and Communication Technology, vol. 399, pp. 178–185. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37437-1_15

Barten, D.: Client-side attacks on the LastPass browser extension (2019)

Carr, M., Shahandashti, S.F.: Revisiting security vulnerabilities in commercial password managers. In: Hölbl, M., Rannenberg, K., Welzer, T. (eds.) SEC 2020. IFIP Advances in Information and Communication Technology, vol. 580, pp. 265–279. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58201-2_18

Chatterjee, R., et al.: Cracking-resistant password vaults using natural language encoders. In: 2015 IEEE Symposium on Security and Privacy. IEEE (2015)

Chaudhary, S., et al.: Usability, security and trust in password managers: a quest for user-centric properties and features. Comput. Sci. Rev. **33**, 69–90 (2019)

Dorai, G., et al.: Vide-vault app identification and extraction system for iOS devices. Forensic Sci. Int.: Digit. Invest. **33**, 301007 (2020)

Fagan, M., et al.: An investigation into users' considerations towards using password managers. Hum.-Cent. Comput. Inf. Sci. **7**(1), 1–20 (2017)

Fahl, S., Harbach, M., Oltrogge, M., Muders, T., Smith, M.: Hey, you, get off of my clipboard. In: Sadeghi, A.R. (ed.) FC 2013. Lecture Notes in Computer Science, vol. 7859, pp. 144–161. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_12

Gasti, P., Rasmussen, K.B.: On the security of password manager database formats. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 770–787. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_44

Gilbert, A., Seigfried-Spellar, K.C., Gilbert, A.K.: Forensic discoverability of iOS vault applications. J. Digit. Forensics Secur. Law **17**(1), 1 (2022)

Gonzalez, R., Chen, E.Y., Jackson, C.: Automated password extraction attack on modern password managers. arXiv preprint arXiv:1309.1416 (2013)

Gray, J., Franqueira, V.N.L., Yu, Y.: Forensically-sound analysis of security risks of using local password managers. In: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW). IEEE (2016)

He, Y., Wang, R., Shi, W.: Implementation of a TPM-based security enhanced browser password manager. Wuhan Univ. J. Nat. Sci. **21**(1), 56–62 (2016)

Huaman, N., et al.: They would do better if they worked together: the case of interaction problems between password managers and websites. In: 2021 IEEE Symposium on Security and Privacy (SP). IEEE (2021)

Li, Z., et al.: The {Emperor's} new password manager: security analysis of web-based password managers. In: 23rd USENIX Security Symposium (USENIX Security 2014) (2014)

Li, Y., Wang, H., Sun, K.: Bluepass: a secure hand-free password manager. In: Lin, X., Ghorbani, A., Ren, K., Zhu, S., Zhang, A. (eds.) SecureComm 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 238, pp. 185–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-78813-5_10

Luevanos, C., et al.: Analysis on the security and use of password managers. In: 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). IEEE (2017)

Martini, B., Do, Q., Choo, K.-K.R.: Mobile cloud forensics: an analysis of seven popular Android apps. arXiv preprint arXiv:1506.05533 (2015)

Ntantogian, C., et al.: Evaluating the privacy of Android mobile applications under forensic analysis. Comput. Secur. **42**, 66–76 (2014)

Oesch, S., et al.: "It basically started using me": an observational study of password manager usage. In: CHI Conference on Human Factors in Computing Systems (2022)

Oesch, S., Gautam, A., Ruoti, S.: The emperor's new autofill framework: a security analysis of autofill on iOS and Android. In: Annual Computer Security Applications Conference (2021)

Oesch, S., Ruoti, S.: That was then, this is now: a security evaluation of password generation, storage, and autofill in browser-based password managers. In: Proceedings of the 29th USENIX Conference on Security Symposium (2020)

Peng, M., et al.: DECADE-deep learning based content-hiding application detection system for Android. In: 2021 IEEE International Conference on Big Data (Big Data). IEEE (2021)

Sabev, P., Petrov, M.: Android password managers and vault applications: data storage security issues identification. J. Inf. Secur. Appl. **67**, 103152 (2022)

Sabev, P., Petrov, M.: Android password managers and vault applications: an investigation on data remanence in main memory (2021a)

Ruffin, M., et al.: Casing the vault: security analysis of vault applications. In: Proceedings of the 21st Workshop on Privacy in the Electronic Society (2022)

Sabev, P., Petrov, M.: Android password managers and vault applications: comparative security analysis. In: 2021 International Conference Automatics and Informatics (ICAI). IEEE (2021b)

Shirvanian, M., et al.: A hidden-password online password manager. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing (2021)

Silver, D., et al.: Password managers: attacks and defenses. In: 23rd USENIX Security Symposium (USENIX Security 2014) (2014)

Stobert, E., Biddle, R.: A password manager that doesn't remember passwords. In: Proceedings of the 2014 New Security Paradigms Workshop (2014)

Stock, B., Johns, M.: Protecting users against XSS-based password manager abuse. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications security (2014)

Walkup, E.: The password problem. No. SAND2016-5208T. Sandia National Lab. (SNL-NM), Albuquerque, NM (United States) (2016)

Yu, F., Yin, H.: A security analysis of the authentication mechanism of password managers. In: 2021 IEEE 21st International Conference on Communication Technology (ICCT). IEEE (20210

Zhang, X., Baggili, I., Breitinger, F.: Breaking into the vault: privacy, security and forensic analysis of Android vault applications. Comput. Secur. **70**, 516–531 (2017)

Zhao, R., Yue, C., Sun, K.: A security analysis of two commercial browser and cloud based password managers. In: 2013 International Conference on Social Computing. IEEE (2013)
Zhao, R., Yue, C.: All your browser-saved passwords could belong to us: a security analysis and a cloud-based new design. In: Proceedings of the third ACM Conference on Data and Application Security and Privacy (2013)
Zhao, R., Yue, C., Sun, K.: Vulnerability and risk analysis of two commercial browser and cloud based password managers. ASE Sci. J. **1**(4), 1–15 (2013)