Dvorak: A Browser Credential Dumping Malware

José Areia[1,2*†] and Bruno Santos[1*†]

[1]School of Technology and Management, Polytechnic of Leiria, Portugal.
[2]Computer Science and Communication Research Centre, Polytechnic of Leiria, Portugal.

*Corresponding author(s). E-mail(s): 2230455@my.ipleiria.pt; 2230456@my.ipleiria.pt;
†These authors contributed equally to this work.

Abstract

Memorising passwords is undoubtedly a challenging task for humans. Consequently, the use of password managers, especially browser-based ones, has seen a noticeable increase in recent years. With that in mind, this paper aims to clarify and conduct an empirical analysis of the security features of password managers in Google Chrome, Microsoft Edge, Opera GX, Mozilla Firefox, and Brave. To corroborate the obtained results, we developed a malware capable of gathering the necessary files to decrypt passwords stored by the browsers and read them in plain text. The results indicate that it is possible to retrieve all the passwords stored in the mentioned browsers in plaintext. In this paper, readers can delve into the details of both the results and the discussion of the security analysis conducted, as well as the processes developed to create the malware. Additionally, relevant open challenges, future work, and conclusions will be presented.

Keywords: Browser Security, Malware, Password Managers, Security Analysis

1 Introduction

Despite the various challenges that passwords currently pose, they remain a widely used form of authentication on the web [1]. Passwords are typically difficult for attackers to guess without any prior knowledge of the user [2]. Paradoxically, using the same logic, it can be equally challenging for users to remember complex passwords. Consequently, users often resort to creating weak and short passwords [3]. In fact, a

common behaviour among users is the tendency to reuse the same password on multiple websites [4, 5]. Since this behaviour significantly compromises the security and safety of users, browsers have taken steps to address it by implementing their own password managers. These password managers can generate complex passwords, store them securely, and automatically fill in log-in forms. However, password managers are not immune to attacks [6], especially browser-based ones [7]. In 2023, Security.org [8] released a report where a survey on the utilisation of password managers was the focal point. The study, which received responses from 1500 U.S. residents, concluded that the number of people using password managers is on the rise. In 2022, the adoption rate was $\approx 21\%$, while in 2023, it increased to $\approx 34\%$, indicating a growth of $\approx 13\%$. Additionally, the article reports that $\approx 28\%$ of people stored their passwords in the browser in 2023, compared to $\approx 23\%$ in 2022. This trend appears to be consistently growing [9]. Therefore, we not only propose conducting a survey and a comparative analysis of various browser security measures in saving and managing user passwords, but we have also developed a malware capable of locally gathering files associated with saved passwords and decrypting them.

1.1 Contributions

Our results reveal that browser-based password managers store their information locally on users' computers, presenting a potential vector for attack. Moreover, our findings indicate that it was possible to collect all the files and decrypt them, gaining access to passwords in plain text, using the malware we developed. Based on the findings, summarily, our contributions include:

- Our research unveiled noteworthy security concerns associated with browser-based password managers, despite the reassurances conveyed in their advertisements, especially concerning the storage of saved user passwords. Addressing these security concerns is imperative for enhancing the overall security of user passwords.
- To the best of our knowledge, we are the first to introduce a malware capable of gathering the requisite files associated with browser-based password managers and decrypting them, thus gaining access to the passwords in plaintext.
- In comparison to other studies [6, 7, 10, 11], our work stands out as the sole effort entirely focused on addressing the security concerns of browser-based password managers. We not only expose these concerns but also deploy a comprehensive suite to decrypt and gain access to passwords stored in five different browsers: Google Chrome, Microsoft Edge, Opera GX, Mozilla Firefox, and Brave.

1.2 Paper organisation

The paper is organised as follows. Section 2 discusses a more detailed background and related work in the literature. Section 3 presents all the work developed within this research. In Section 4 the results obtained within this research are presented. Section 5, focuses on the discussion of the results, including a comparison with other works and an analysis of the best and worst results. Finally, in Section 6, the paper concludes with pointers to future work.

2 Background

We begin by reviewing essential background information related to browser credential storage, cryptography employed in this context, types of attacks, particularly initial access attacks aimed at gaining access to a user's computer, and exfiltration techniques.

Web browsers, including Google Chrome [12], Mozilla Firefox [13], Microsoft Edge [14], Opera GX [15], and Brave [16], typically store credentials, such as usernames and passwords of the website, with the user's consent [17, 18] to alleviate the need for users to remember them in the future. However, adversaries can obtain credentials from these web browsers by accessing files specific to the target browser [19]. Google Chrome, Microsoft Edge [20], Opera GX [21], and Brave [16], are all Chromium-based browsers. Therefore, the file locations and the type of storage used for storing credential values are similar among all the mentioned browsers. The credential database location is the default profile folder [1] [2] of each browser. It is named "Login Data" and is in the SQLite 3 format [19]. This database can be opened with a simple database browser, but the stored credentials are encrypted. A key to decrypt credentials is required and is located in the parent directory of the database, in a file named "Local State" [22]. This file is in JSON format and, as previously mentioned, it contains the base64-encoded key required to decrypt the browser-saved credentials [23]. On the other hand, Mozilla Firefox stores its credentials in a JSON file named "logins.json", and the key required to decrypt this information is stored in a SQLite 3 database named "key4.db" [22]. It is important to note that the Firefox versions prior to 75 do not have the file "key4.db"; instead, they use the file "key3.db" [24]. However, similar to the databases in Chromium-based browsers, retrieving the encryption key required to decrypt the credentials in Firefox is not feasible because it is also encrypted. Furthermore, these files, both the database and JSON file, are stored in the profile folder, located in a sub-directory of Mozilla Firefox [3] [4].

Encryption is a way of scrambling data so that only authorised parties can understand the information. Encrypting and decrypting data requires cryptographic keys and a cryptographic algorithm. Normally, only authorised parties should have the key to decrypt the data. If that key is compromised, the encrypted data may also be compromised. Knowing the decryption key and the algorithm used to encrypt the data, it can be decrypted by unauthorised parties [25]. In the context of browser-stored credentials, most browsers store encrypted data and the decryption key locally in the user profile directory [19]. Chromium-based browsers such as Google Chrome, Microsoft Edge, Opera GX and Brave encrypt data using AES [26, 27]. Firefox is not Chromium-based and uses the Public Key Cryptography Standard (PKCS) #11 [28]. Chromium-based browsers encrypt the decryption key using Chromium's OSCrypt [26, 27]. Firefox encrypts the decryption key using Network Security Services (NSS) [28].

To gain access to a computer, adversaries employ various initial access techniques [19]. Examples of these attacks include spear phishing attachment [29] and replication

[1] Windows: `%USERPROFILE%\appdata\local\BROWSER\user data\PROFILE\Login Data`
[2] `Linux: ~/.config/BROWSER/PROFILE/FILES`
[3] `Windows: %USERPROFILE%\appdata\roaming\mozilla\firefox\profiles\FILES`
[4] `Linux: ~/.mozilla/firefox/PROFILE/FILES`

through removable media [30]. A spear-phishing attachment attack is a type of social engineering attack that targets specific individuals, using intelligent emails with malware attachments [29, 31]. These emails exploit human behaviour to make the receiver execute the malware attachment. This malware attachment requires user execution for attacker access [29]. Replication through removable media works by relying on the physical introduction of a removable media, with malware, into a device. The mentioned media can be a USB flash drive or any other device that connects to the system. This technique targets untrained individuals in the field of cybersecurity [30].

For data exfiltration, adversaries can employ various techniques, including traffic duplication and exfiltration to cloud storage [32]. Another method used is through webhooks, which are HTTP-based callback functions that allow communication between two APIs [33]. Attackers can leverage these webhooks to exfiltrate and steal user data [34].

2.1 Literature review

Numerous researchers have dedicated their efforts to understanding and conducting security analyses of various password managers, encompassing both non-browser-based and a few browser-based options. This subsection aims to present some of the works related to this topic.

Oesch and Ruoti [6] presented their work, proposing an analysis of the most common password managers, such as KeyPass, BitWarden, and LastPass, along with some browser-based password managers. They examined various aspects of these password managers, including password strength, randomness, storage, vault encryption, metadata privacy, and the auto-fill system. They also offered recommendations based on their conclusions, advising against certain password managers, providing suggestions for improving existing password managers, and identifying areas for future research. These include exploring safer autofill systems, implementing improved master password policies, and developing filters to identify weak passwords.

In 2014, Ciampa [10] entered into knowledge-based, traditional and new password attacks. Additionally, the author conducted a social study on user preferences for browser password managers. The results led to the conclusion that within a group of students ($N = 166$), the majority preferred to use their web browser as a password manager over alternatives such as LastPass. However, users did not highly rate using web browser managers as an activity that improved the security of their accounts. Instead, they perceived themselves as using strong passwords and having sufficient memory to store all their passwords.

Both works presented in this literature review aimed to contribute to the community's understanding of the security of password managers and their usage among the masses. They not only provided essential information for comprehending the concepts, but also offered solutions for future mitigation's within the respective themes.

3 Materials & Methods

This section will show the implementation process of the malware developed [5] which will gather browser credentials to test the safeness of storing credentials in the most common browsers. To conduct the test, two scripts were created: one to gather encrypted credentials and exfiltrate them, and the other to attempt decryption. The malware will be made up of these two scripts. The objective of the test is to evaluate, for each browser, how hard it is to gather the encrypted credentials and, if possible, how hard it is to decrypt them. The malware was created in a controlled environment and was tested locally in compliance with law and ethical guidelines. For the creation and testing of the malware, a single Windows machine was utilised along with the programming language Python. The browsers used for testing were: Google Chrome, Microsoft Edge, Mozilla Firefox, Opera GX and Brave. The complete malware workflow is shown in Figure 1. In the workflow, the yellow processes will not be detailed, and the green processes are the main parts of the process.

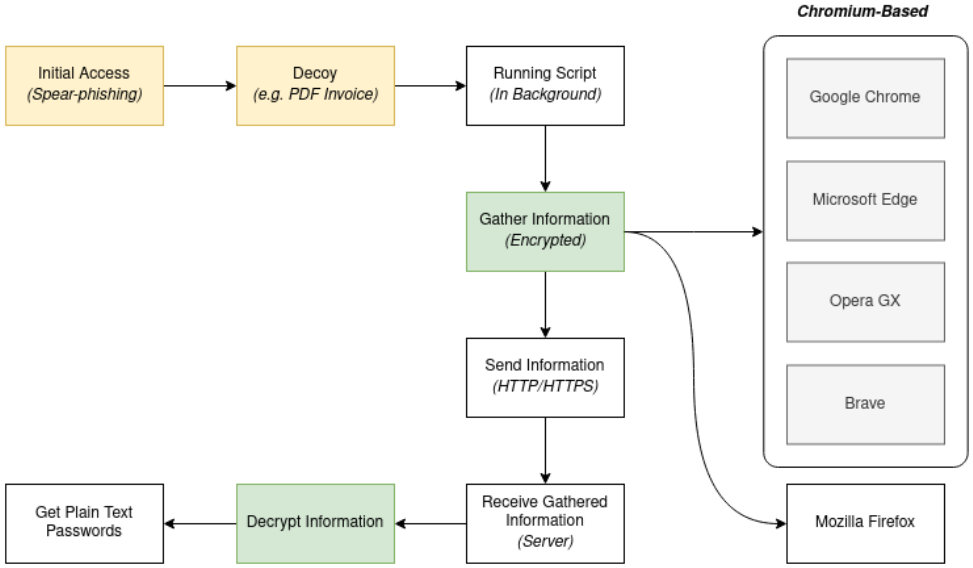

Fig. 1 Illustrating the comprehensive malware workflow: highlighting the initial phase access (not addressed in this paper) in yellow, and emphasising the significant phases investigated in this research with green highlights.

Initially, the malware needs to be infiltrated into the victim's computer. Examples of infiltration techniques include spear phishing and replication through removable media, such as a USB stick [30]. The next step for the malware would be to obtain the encrypted browser credentials. To gather that information, we developed a script that

[5]Project GitHub: https://github.com/brunobertolo/dvorak

collects this type of information from the aforementioned browsers. This information is scattered across multiple encrypted files (refer to Section 2). The script must know which operating system the victim is using so that the file locations can be adjusted accordingly. To exfiltrate the files over to the attacker's computer, we could have used a webhook, but for the sake of simplicity, a simple local HTTP server was used. When exfiltrating the encrypted files, it can be problematic that some of the files from Chromium-based browsers have the same name. The simple fix was to create a compressed archive, with the encrypted files, for each browser. These compressed archives were exfiltrated via a POST request to the local HTTP server. With the encrypted files in hand, the next step was to decrypt the files to get the credentials in plain text. A script was also created for this purpose. The script was divided into Chromium-based decryption and Firefox decryption because the process is different for each. The Chromium decryption process is divided into three different parts: (1) getting the encrypted secret key from the "Local State" file and decrypting it with the CryptUnprotectData algorithm, (2) gets the URL, the username and the encrypted password from the "Login Data" file, and (3) uses the decrypted secret key to decrypt the password using the AES algorithm in Galois/Counter Mode (GCM) mode.

The Firefox decryption process is similar but uses different algorithms to do the same. This process is also divided into three different parts: (1) getting the encrypted secret key from the "key4.db" file and decrypting it with the AES algorithm in Cipher Block Chaining (CBC) mode, (2) get the URL and the encrypted credentials from the "logins.json" file, and (3) use the decrypted secret key to decrypt the password and username with the DES3 algorithm in CBC mode.

The script was only created to work on Windows and further development and research would be needed to make it work on other operating systems as well. The malware was developed based on two related works [6] [7]. By decrypting the credentials, we show that obtaining the credentials saved in a browser is possible.

4 Results

This section addresses the findings and facts of the study based on the performance of the malware developed during the research. The objective is to assess the difficulty of obtaining and decrypting encrypted credentials saved in the browser and at the same time asses the security of saving credentials in browsers. For this purpose, a custom malware designed to steal saved user credentials was employed. The malware successfully retrieved saved login information from the aforementioned web browsers as detailed in this paper. Additionally, the results indicate that it is only possible to decrypt information regarding Chromium-based passwords if the computer is the one that encrypted the information.

5 Discussion

In this section, we will interpret and contextualise the results obtained, discuss their implications and outline some limitations of the study. The results of this case study

[6] Chromium-based: https://github.com/ohyicong/decrypt-chrome-passwords
[7] Firefox-based: https://github.com/lclevy/firepwd

are that the malware successfully extracted usernames, passwords, and URLs from all the sites tested. The browser's encryption for saving credentials locally was found to be breakable. This means that the encryption used for the secret key, which is used to encrypt the credentials, was also compromised. It should be noted that all of this testing was done locally, which means that all the components utilised are saved locally on the machine, facilitating the process. The combination of these results shows that credentials saved in browsers are susceptible to being stolen and decrypted. The malware developed for this purpose was based on two scripts found in GitHub, which means that scripts for stealing browser credentials are already available for public use. During the development of this malware, we found that the Firefox-based decryption process was slightly more complex than the Chromium-based decryption. However, this does not change the fact that the credentials were still gathered and decrypted. In terms of saving local credentials, no browser is more secure than another. Nevertheless, the fact that passwords stored in Chromium-based browsers cannot be decrypted if not on the computer that encrypted them poses a positive security implication compared to Firefox.

One improvement that browsers can and should make is to save the secret key used in the encryption/decryption of the credentials in the cloud. This key can only be accessed by using a master password associated with the user profile of each browser. If the user wants to save credentials in the browser, the user must log into their profile and define the master password. So that when a computer is compromised the key is unavailable for the malware to decrypt the credentials. Although it may slightly decrease usability, implementing it would significantly enhance security. The alternative to using browsers to save credentials is to use third-party password managers like BitWarden, Dashlane, 1Password or LastPass. This is the most secure and recommended way to store your credentials because they already use the cloud storage solution with better encryption algorithms.

The study was done locally without the infiltration of the malware; this is a limitation of the study because it does not take into account the first step of the process. For this malware to work, the victim's computer must be compromised first. Another limitation is the number of browsers tested. As mentioned in previous sections, the malware was only developed and tested for the Windows operating system, lacking analysis in other operating systems.

6 Conclusion

In the modern world, where information plays a crucial role, ensuring maximum security is imperative. Despite their inherent problems, passwords remain the most widely used method for achieving a trade-off between ease of use and ensuring a minimum level of security. This makes them the predominant form of authentication in various applications. However, how users store their passwords continues to be a concern for security. While password managers can be the answer, browser-based password managers, in particular, face challenges in how they store their data locally. We propose a comprehensive review of how browser-based password managers store their data, considering the potential implementation of a cloud-based solution. We also believe

that it is important to review different methods for password encryption across all browsers. Finally, we believe that researchers should continue to evaluate the progress of both password managers and browser-based password managers, focusing on both security and usability.

Acknowledgements

This research received support during the Computer Network Security course, instructed by Professor Mário Antunes, PhD (0000-0003-3448-6726) at the School of Management and Technology, Polytechnic of Leiria.

References

[1] Bonneau, J., Herley, C., Oorschot, P.C.v., Stajano, F.: The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In: 2012 IEEE Symposium on Security and Privacy, pp. 553–567 (2012). https://doi.org/10.1109/SP.2012.44 . ISSN: 2375-1207. https://ieeexplore.ieee.org/document/6234436 Accessed 2024-01-20

[2] Dell'Amico, M., Michiardi, P., Roudier, Y.: Password Strength: An Empirical Analysis. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9 (2010). https://doi.org/10.1109/INFCOM.2010.5461951 . ISSN: 0743-166X. https://ieeexplore.ieee.org/document/5461951 Accessed 2024-01-20

[3] Chaparro, B., Riley, S.: Password Security: What Users Know and What They Actually Do. (2006). https://www.semanticscholar.org/paper/Password-Security%3A-What-Users-Know-and-What-They-Do-Chaparro-Riley/15bb7d44b8f0d6e721f433c5456375442e83ae36 Accessed 2024-01-20

[4] Anupam Das, M.C. Joseph Bonneau: The tangled web of password reuse. NDSS **14**, 23–26 (2014)

[5] Florencio, D., Herley, C.: A large-scale study of web password habits. In: Proceedings of the 16th International Conference on World Wide Web. WWW '07, pp. 657–666. Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1242572.1242661 . https://dl.acm.org/doi/10.1145/1242572.1242661 Accessed 2024-01-20

[6] 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: 29th USENIX Security Symposium (USENIX Security 20), pp. 2165–2182. USENIX Association, ??? (2020). https://www.usenix.org/conference/usenixsecurity20/presentation/oesch

[7] Silver, D., Jana, S., Boneh, D., Chen, E., Jackson, C.: Password managers: Attacks and defenses. In: 23rd USENIX Security Symposium (USENIX Security 14), pp.

449–464. USENIX Association, San Diego, CA (2014). https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/silver

[8] Security.org Team: Password Manager Industry Report and Market Outlook in 2023 (2023). https://www.security.org/digital-safety/password-manager-annual-report/ Accessed 2024-01-20

[9] Koebert, J.: 84% of Internet Users Practice Dangerous Password Behaviors [Survey] (2023). https://allaboutcookies.org/password-users-behavior-survey Accessed 2024-01-20

[10] Ciampa, M.: A Comparison of User Preferences for Browser Password Managers. Journal of Applied Security Research **8**(4), 455–466 (2013) https://doi.org/10.1080/19361610.2013.825751 . Accessed 2024-01-20

[11] Gasti, P., Rasmussen, K.B.: On the Security of Password Manager Database Formats. In: Foresti, S., Yung, M., Martinelli, F. (eds.) Computer Security – ESORICS 2012. Lecture Notes in Computer Science, pp. 770–787. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_44

[12] Google: Google Chrome - The Fast & Secure Web Browser Built to be Yours (2023). https://www.google.com/chrome/ Accessed 2023-10-13

[13] Firefox, M.: Firefox - Protect your life online with privacy-first products (2023). https://www.mozilla.org/en-US/firefox/ Accessed 2023-10-13

[14] Microsoft: Get to Know Microsoft Edge (2023). https://www.microsoft.com/en-gb/edge/ Accessed 2023-10-13

[15] Opera: Opera GX | Gaming Browser | Opera (2023). https://www.opera.com/gx Accessed 2023-10-13

[16] Brave: Secure, Fast, & Private Web Browser with Adblocker (2023). https://brave.com/ Accessed 2023-10-13

[17] Google: Google Password Manager - Manage Your Password Safely & Easy (2023). https://passwords.google Accessed 2023-10-13

[18] Mozilla: Firefox: Free password manager (2023). https://www.mozilla.org/en-US/firefox/features/password-manager/ Accessed 2023-10-13

[19] MITRE: Credentials from Password Stores: Credentials from Web Browsers. Technical report, MITRE (2022). https://attack.mitre.org/techniques/T1555/003/

[20] Microsoft: Microsoft Edge and Chromium Open Source: Our IntentMicrosoft Edge and Chromium Open Source: Our Intent (2018). https://github.com/MicrosoftEdge/MSEdge/blob/7d69268e85e198cee1c2b452d888ac5b9e5995ca/

README.md Accessed 2023-10-13

[21] Opera: A First Peek at Opera 15 for Computers (2013). https://dev.opera.com/blog/a-first-peek-at-opera-15-for-computers/ Accessed 2023-10-13

[22] Mistele, K.: Stealing Saved Browser Passwords: Your New Favorite Post-Exploitation Technique (2021). https://kylemistele.medium.com Accessed 2023-10-13

[23] Deneut, T.: Security: General Security Scripts (2022). https://github.com/tijldeneut/Security Accessed 2023-10-13

[24] Clévy, L.: Firepwd.py, an open source tool to decrypt Mozilla protected passwords. original-date: 2013-08-27T17:58:11Z (2023). https://github.com/lclevy/firepwd Accessed 2023-10-13

[25] Cloudflare: What is encryption? (2023). https://www.cloudflare.com/learning/ssl/what-is-encryption/ Accessed 2023-10-13

[26] ericlaw: Local Data Encryption in Chromium (2020). https://textslashplain.com/2020/09/28/local-data-encryption-in-chromium/ Accessed 2023-10-13

[27] dan-wesley: Microsoft Edge password manager security (2023). https://learn.microsoft.com/en-us/deployedge/microsoft-edge-security-password-manager-security Accessed 2023-10-13

[28] Yicong: How to decrypt Firefox passwords with Python? (2023). https://medium.com/geekculture/how-to-hack-firefox-passwords-with-python-a394abf18016 Accessed 2023-10-13

[29] MITRE: Spearphishing attachment. Technical report, MITRE (2023). https://attack.mitre.org/techniques/T0865/

[30] MITRE: Replication through removable media. Technical report, MITRE (2023). https://attack.mitre.org/techniques/T0847/

[31] Harsh, K., Singh, T., Singh, P.K.: Emerging threats of cyber crimes

[32] MITRE: Exfiltration. Technical report, MITRE (2019). https://attack.mitre.org/tactics/TA0010/

[33] RedHat: What is a webhook? https://www.redhat.com/en/topics/automation/what-is-a-webhook Accessed 2024-01-09

[34] MITRE: Exfiltration over web service: Exfiltration over webhook. Technical report, MITRE (2023). https://attack.mitre.org/techniques/T1567/004/