Dvorak: A Browser Credential Dumping Malware

José Areia[1,2][a], Bruno Santos[1,2] and Mário Antunes[1,3][b]

[1] *School of Management and Technology (ESTG), Polytechnic of Leiria, Leiria, Portugal*

[2] *Computer Science and Communication Research Centre (CIIC), Polytechnic of Leiria, Leiria, Portugal*

[3] *INESC TEC, Center for Research in Advanced Computing Systems, Porto, Portugal*

Keywords: Web Security, Browser Password Managers, Malware Development, Network Security, Security Analysis.

Abstract: Memorising passwords poses a significant challenge for individuals, leading to the increasing adoption of password managers, particularly browser password managers. Despite their benefits to users' daily routines, the use of these tools introduces new vulnerabilities to web and network security. This paper aims to investigate these vulnerabilities and analyse the security mechanisms of browser-based password managers integrated into Google Chrome, Microsoft Edge, Opera GX, Mozilla Firefox, and Brave. Through malware development and deployment, Dvorak is capable of extracting essential files from the browser's password manager for subsequent decryption. To assess Dvorak functionalities we conducted a controlled security analysis across all aforementioned browsers. Our findings reveal that the designed malware successfully retrieves all stored passwords from the tested browsers when no master password is used. However, the results differ depending on whether a master password is used. A comparison between browsers is made, based on the results of the malware. The paper ends with recommendations for potential strategies to mitigate these security concerns.

1 INTRODUCTION

Prioritising web security is crucial, and integrating password managers into browsers can significantly strengthen defence mechanisms. Despite the various challenges that passwords currently pose, they remain a widely used form of authentication on the web. Passwords are typically difficult for attackers to guess without any prior knowledge of the user (Dell'Amico et al., 2010), and 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 (Gabe Turner, 2023). In fact, a common behaviour among users is the tendency to reuse the same password on multiple websites Since this behaviour significantly compromises the web security 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 (Oesch and Ruoti, 2020), especially browser-based ones (Silver et al., 2014). In 2023, a report (Gabe Turner, 2023) was released, focusing on a survey regarding the utilisation of password managers. 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.

In this paper, we delve into browser-based password managers integrated into widely used platforms such as Google Chrome, Microsoft Edge, Opera GX, Mozilla Firefox, and Brave. This paper seeks to unearth vulnerabilities and explore the corresponding security mechanisms. By using specialised malware capable of extracting important files from each browser's password manager, we conduct controlled experiments. The created malware was able to successfully breach the defences of the tested browsers and extract all stored passwords in cases where a master password was not utilised. Interestingly, the results showed differences depending on whether a master password was used. To augment the comprehensiveness of our research, we compared the re-

[a] https://orcid.org/0009-0000-0595-0468

[b] https://orcid.org/0000-0003-3448-6726

Areia, J., Santos, B. and Antunes, M.
Dvorak: A Browser Credential Dumping Malware.
DOI: 10.5220/0012731300003767
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 434-441
ISBN: 978-989-758-709-2; ISSN: 2184-7711
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

sults of malware across different browsers. This examination revealed important differences in the security of each platform, providing valuable insights for users, developers, and security experts. Our investigation aimed to understand the security measures implemented in password managers beyond just identifying vulnerabilities. After uncovering these findings, our paper concludes with recommendations for potential remediation strategies that can be implemented to address the security concerns that have been identified. This paper also aims to serve as a guiding compass, presenting insights and suggestions to strengthen the security of browser-based password management.

Contributions of this paper are summarised as follows. To the best of our knowledge, this paper is the first attempt to introduce a malware capable of gathering the necessary files associated with the Browser-based Password Manager (BPM) of five different browsers and decrypting them, thereby gaining access to users' credentials. To promote reproducible research, the code is available on GitHub repository (not shown for blind review), as well as a video demonstrating the malware usage. In comparison to other studies (refer to Section 2), the paper stands out as the sole effort entirely focused on addressing the security concerns of BPM. In addition to highlighting these concerns, we employ a comprehensive toolkit to decrypt and gain access to passwords stored across five different browsers: Google Chrome, Microsoft Edge, Opera GX, Mozilla Firefox, and Brave. The paper also addresses open challenges and provides solutions to the security concerns identified, which can enhance the security of users utilising BPM. These solutions can be implemented by browser developers to improve user security.

The paper is organised as follows. Section 2 provides a detailed background and reviews related work in the literature. Section 3 presents all the work developed within this research. Section 4 discusses the results obtained within this research and provides a brief discussion of the findings. Section 5 focuses on the open challenges and future work inherent in this theme. Finally, Section 6 concludes the paper with pointers to future research directions.

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 aimed at stealing user's data.

Web browsers, such as Google Chrome, Mozilla Firefox, Microsoft Edge, Opera GX, and Brave, are applications designed to retrieve and display web pages, facilitating users' access to web content (Nelson et al., 2020). Occasionally, users need to log in with their credentials to view personal content. These browsers typically store website credentials, encompassing usernames and passwords, with the user's permission (Gasti and Rasmussen, 2012), thereby relieving users of the burden of memorising them for future use. This encapsulates the principle of any password manager (Yang et al., 2014): securely storing user passwords so that users need not remember them for subsequent use.

Another important concept when talking about password managers is the master/primary password, which serves as a single password granting access to all the credentials stored in the password manager (Vossaert et al., 2014). Therefore, the creation of the master password should adhere to secure standards. Firefox already gives the option of utilising a master password to protect your credentials. However, Firefox does not actively encourage users to use a master password, which may lead to lower adoption rates of this security feature (Gabe Turner, 2023). Chromium-based browsers do not have the option to set up a master password, but they use the computer as the master password. This means that the decryption of the credentials can only occur in the same computer as the encryption. This type of master password is weak if the computer is compromised.

Adversaries can obtain credentials from web browsers by accessing and decrypting files specific to the target browser (Luevanos et al., 2017). Google Chrome, Microsoft Edge, Opera GX, and Brave, are all Chromium-based browsers. Therefore, the file locations and the type of storage used for storing credential values are similar among all. The credential database location is the default profile folder of each browser, it is named `Login Data` and is in the SQLite 3 format (Muslim et al., 2020). This database can be opened with a simple database browser, but the stored passwords are encrypted. The username and password can be found in the table `logins` under `username_value` and `password_value` respectively. The URL for the respective website can be found in the same table under `origin_url` and `action_url`. A key to decrypt the passwords is required and is located in the parent directory of the database, in a JSON file named `Local State` (Mistele, 2021), which contains the base64-encoded key required to decrypt the browser-saved credentials (Deneut, 2022). This decryption key is inside `os_crypt` and with the name `encrypted_key`.

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` (Mistele, 2021). The username, password and the respective URL are stored in the JSON file inside `logins` with the names `encryptedUsername`, `encryptedPassword` and `formSubmitURL` respectively. The decryption key inside the database can be found in the table `metaData` under `password`. It is important to note that the Firefox versions prior to 75 do not have the file `key4.db`, instead it uses the file `key3.db` (Clévy, 2013).

Encryption is a way of permutation data so that only authorised parties can understand the information (Stallings, 2017). Encrypting and decrypting data requires cryptographic keys and a cryptographic algorithm (Menezes et al., 2018). Normally, only authorised parties should have the key to decrypt the data. If that key is compromised, the encrypted data may also be compromised. Encrypted data can be compromised if the key and the algorithm utilised, in the encryption, are known (Shabtai et al., 2012). In the context of browser-stored credentials, most browsers store encrypted data and the decryption key locally in the user profile directory (Yang et al., 2014).

Chromium-based browsers encrypt credentials using the symmetric algorithm AES (Lawrence, 2020; Dan Wesley, 2023) and they encrypt the decryption key using the `CryptUnprotectData` function. This function is part of the Win32 API, used in C++, the main programming language used to create the Chromium-based browsers (Ashcraft, 2022) (Qian et al., 2020). `CryptUnprotectData` was made available by Microsoft and is only utilised in Windows (Sajid et al., 2021). This function is characterised by the requirement that only a user with identical login credentials to the one who encrypted the data can decrypt it (Ashcraft, 2022). Furthermore, both the encryption and decryption processes must occur on the same computer. Firefox uses the AES algorithm for the encryption of the credentials and the 3DES algorithm for the encryption of the decryption key (Zhao and Yue, 2013).

To gain access to a machine, adversaries employ various initial access techniques. Examples of these attacks include phishing attachment (Alabdan, 2020) and replication through removable media (Muslim et al., 2020). A phishing attachment attack is a type of social engineering attack that employs intelligent emails with malware attachments. These emails exploit human behaviour to make the receiver execute the malware attachment. This malware attachment requires user execution for attacker access (Al-Mohannadi et al., 2016). 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 (Nissim et al., 2017). This technique targets untrained individuals in the field of cybersecurity (Tetmeyer and Saiedian, 2010).

For data exfiltration, which involves the unauthorised transfer of sensitive information from a system or network, breaching confidentiality and compromising data integrity (Alshamrani et al., 2019), adversaries can employ various techniques. These techniques may include traffic duplication (Al-Mhiqani et al., 2020) and exfiltration to cloud storage (Ullah et al., 2018). Another method used is through webhooks, which are HTTP-based callback functions that allow communication between two APIs (Biswas, 2021). Attackers can exploit these webhooks to exfiltrate and pilfer user data without leaving any trace of their identity (Alshamrani et al., 2019). A simple HTTP server listening to HTTP POST requests is also commonly used for exfiltration (Al-Bataineh and White, 2012).

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.

In 2013, knowledge-based, traditional and new password attacks were presented (Ciampa, 2013). Additionally, the author conducted a social study on user preferences for BPM. 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 (Ciampa, 2013). 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.

In the same year, research focusing on BPM was conducted (Zhao and Yue, 2013). The authors analysed four different browsers: Firefox, Google Chrome, Opera, and Safari. In their analysis, they described how browsers store credentials, including the files used for storage and the encryption methods employed, and highlighted the significant vulnerability in the ways browsers store credentials (Zhao and Yue, 2013). Consequently, they proposed a cloud-

Table 1: A Literature Work Comparison: Assessing Malware Development, Interoperability Analysis, Browser Analysis, Vulnerability Analysis, Proposed Countermeasures, and Mapping Countermeasure to Attack Vectors.

Works Reviewed	Malware Development	Interoperability Analysis*	Browser Analysis ($N \geqslant 3$)	Vulnerability Analysis	Proposed Countermeasures	Mapping Countermeasures To Attack Vectors
(Ciampa, 2013)	-	-	-	-	✓	-
(Zhao and Yue, 2013)	-	-	✓	✓	✓	✓
(Yang et al., 2014)	-	-	-	✓	✓	✓
(Oesch and Ruoti, 2020)	-	-	-	✓	✓	-
(Muslim et al., 2020)	✓	-	-	✓	✓	✓
Our Work	✓	✓	✓	✓	✓	✓

*By *interoperability*, we mean the ability to decrypt the password database on an external computer that is not the one that encrypted the passwords.

based storage-free BPM where protected data would be entirely stored in the cloud, eliminating the need for anything to be stored on a user's computer.

In 2014, literature advocating for cloud-based password managers utilising privacy-preserving biometrics was published (Yang et al., 2014). In their research, they exposed vulnerabilities in various browsers, particularly concerning the local storage of information. They argue that this method of storing credentials is highly insecure and proposed a cloud-based password manager that utilises two-factor authentication, comprising a master password and biometric authentication. Both authentication factors are stored locally on the client's computer, enhancing inherent privacy preservation.

In 2020 an analysis of the most common password managers, such as KeePassX, KeePassXC, BitWarden, and LastPass, along with some BPM was conducted (Oesch and Ruoti, 2020). They examined various aspects of these password managers, including password strength, randomness, storage, vault encryption, metadata privacy, and the auto-fill system. The authors 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 (Oesch and Ruoti, 2020).

In that same year, a paper presenting the implementation and analysis of a USB password stealer using PowerShell in Google Chrome and Firefox was introduced (Muslim et al., 2020). Similar to other studies, authors analysed and exposed vulnerabilities inherent in BPM and attempted to exploit them. With success, and aided by various Arduino hardware, they were able to steal credentials from both aforementioned browsers. The attack involved plugging the USB stick into the victim's computer, initialising the PowerShell script, downloading the necessary malware from their GitHub repository, gathering all the files, decrypting them on the victim's computer, and finally sending the files to their Gmail account (Muslim et al., 2020). The results indicated that this process took an average of 14 seconds before sending it to the author's email.

The studies presented in this literature review enhance the community's comprehension of password manager security and their adoption among users. These studies not only furnished crucial insights into the concepts but also proposed solutions for addressing future mitigation within the respective domains. Furthermore, to ascertain the novelty and distinctive contribution of our work, we conducted a thorough analysis of related reviews, comparing them with ours in terms of objectives and findings (Table 1).

3 MATERIALS AND METHODS

This section shows the implementation process of the Dvorak developed which tries to gather and decrypt 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 Dvorak will be made up of these two Python scripts. Figure 1 illustrates the attack scenario conducted in this research.

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. An attempt to decrypt credentials protected by the master password will also be made. Firefox protects the credentials with a master password set by the user, while Chromium-based browsers relies upon the computer as the master password. Dvorak 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 11 machine was utilised along with the programming language Python. It is worth noting that Windows Defender was always

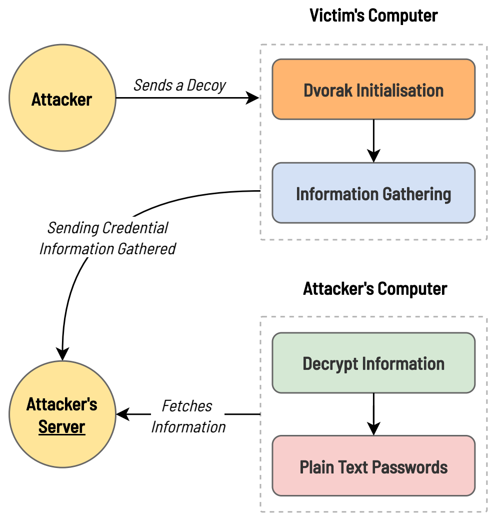

Figure 1: A Comprehensive Dvorak Attack Flow for Acquiring User's BPM Credentials.

active at the time of testing. Furthermore, administrator privileges were not needed to execute the Python script. The browsers used for testing were: Google Chrome version 121.0.6167.161, Microsoft Edge version 121.0.2277.112, Mozilla Firefox version 122.0.1, Opera GX version 106.0.4998.76 and Brave version 1.62.162. The versions used in the test were the most recent versions available at the time of writing this paper. The complete Dvorak workflow is shown in Figure 2, which is divided into five different phases: (I) initial access, (II) initialisation and information gathering, (III) compress and deliver information, (IV) receive and decrypt the information, and (V) gain access to information in plain text.

In the first phase (I) the Dvorak needs to be infiltrated into the victim's computer. The next step (II) for the malware would be to obtain the encrypted browser credentials. To gather that information, we developed a script that 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.

For transferring the files to the attacker's computer, a webhook could have been employed; however, for simplicity's sake, a basic local HTTP server implemented with Flask, in Python, was utilised. When exfiltrating the encrypted files, a potential issue arises as some files from Chromium-based browsers share the same name. Consequently, in the subsequent phase (III), the solution was to address this by creating a compressed archive containing the encrypted files for each browser. These compressed

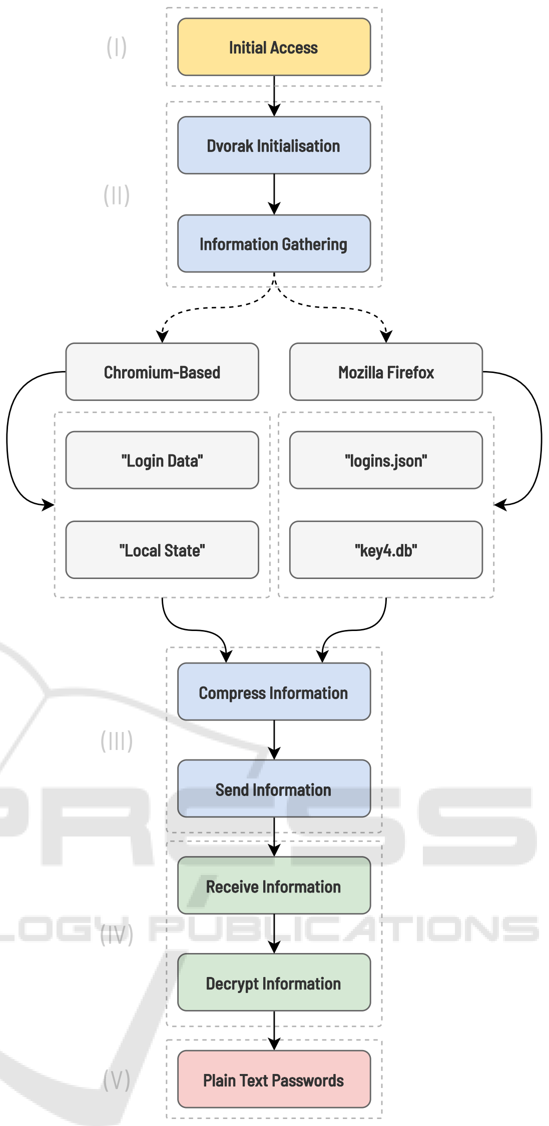

Figure 2: The Structured Dvorak Framework in Five Key Phases: (I) Initial Access, (II) Initialisation and Information Gathering, (III) Compress and Deliver Information, (IV) Receive and Decrypt the Information, and (V) Gain Access to Information in Plain Text.

archives were exfiltrated via a POST request to the local HTTP server.

With the encrypted files in hand, the next step (IV) is to decrypt the files to get the credentials in plain text. A script was also developed for this purpose, which was divided into Chromium-based decryption and Firefox decryption due to the variation in the decryption process for each browser.

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` function, (2) getting the

URL, the username and the encrypted password from the `Login Data` file, and (3) use the decrypted secret key to decrypt the password using the AES algorithm in Galois/Counter Mode (GCM) mode.

Firefox's 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) getting 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. Following the decryption process, Dvorak enters its final phase (V), during which the attacker gains access to the passwords in plain text, provided that the decryption process is successful.

The script was designed to operate on Windows, and additional development and research would be necessary to adapt it to other operating systems. The malware's development drew inspiration from two related works (Clévy, 2013; Yicong, 2020). By successfully decrypting the credentials, we demonstrate the feasibility of accessing browser-stored credentials.

4 RESULTS AND DISCUSSION

This section discusses the findings of the research regarding the performance of the developed malware. The goal was to evaluate the difficulty of obtaining and decrypting the credentials stored in the BPM while simultaneously assessing the security implications of storing credentials in browsers. For this purpose, a custom malware designed to steal saved user credentials was employed. Dvorak successfully retrieved saved login information from the aforementioned web browsers when no master password was utilised. When a master password is utilised in Firefox, the malware is incapable of decrypting the credentials. Additionally, because Chromium-based browsers use the computer as the master password, the results indicate that it is only possible to decrypt the credentials if the computer and the user are the one that encrypts the information. Table 2 presents the obtained results divided into four different topics: gathering information, the decryption process without a master password, system interoperability and decryption with a master password. A check mark indicates successful completion of the task.

The results of this research indicate that the malware successfully extracted usernames, passwords, and their respective URLs from all the tested browsers when no master password was used. The browser's encryption for saving credentials locally was compromised. This means that the encryption used for the secret key, which is used to encrypt the credentials, was broken. When a master password was used, Dvorak could not decrypt the Firefox credentials. However, Dvorak had success in decrypting Chromium-based credentials, meaning that using the computer as a master password has no security effect if the computer is invaded. 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. While saving credentials locally is not inherently insecure, it becomes so when there is no master password, set by the user, protecting the key and credentials.

During the development of this malware, we discovered that the Firefox-based decryption process was slightly more complex than the Chromium-based decryption process, as it was manually crafted without relying on any pre-existing package, such as AES, which is utilised within the Chromium-based decryption process. Furthermore, Chromium-based credentials can only be decrypted in the same computer, and by the same user, that encrypted them. However, this does not change the fact that the credentials were still gathered and decrypted when no master password was used. In terms of saving local credentials, Firefox is more secure than the other mentioned browsers, but only if a master password is set by the user.

5 OPEN CHALLENGES

Based on our analysis, we have identified the following open challenges and areas for future research:
Usage of Master Passwords: BPM should ensure the confidentiality of credentials, even in the event of a compromised computer. However, while fixes for this security issue are already available, browsers do not encourage or prioritise their use. One straightforward solution is to employ a master password, also known as a primary password, to access stored credentials. This master password should be strong and not stored anywhere on the computer, as it serves as the key to unlocking all stored credentials. This approach can help maintain the confidentiality of the user's credentials, even if the computer is compromised. Firefox already implements a master password system, however, this option is not widely advertised to users and may be unknown to many. If the master password is set, then the developed malware, Dvorak, cannot decrypt Firefox stored credentials. Chromium-

Table 2: Dvorak Benchmark Results: Assessing Performance in Information Gathering, Decryption Process, and System Interoperability across Google Chrome, Mozilla Firefox, Opera GX, Microsoft Edge, and Brave.

Browser	Information Gathering	Decryption W/o Master Password	System Interoperability*	Decryption W/ Master Password
Google Chrome	✓	✓	-	✓
Mozilla Firefox	✓	✓	✓	-
Microsoft Edge	✓	✓	-	✓
Opera GX	✓	✓	-	✓
Brave Browser	✓	✓	-	✓

*By *interoperability*, we mean the ability to decrypt the password database on an external computer that is not the one that encrypted the passwords.

based browsers use the computer as a master password, which is not a good security measure in case of malware invasion on the computer. It would be beneficial for browsers to prompt users to set up a master password by default before storing any credentials in the browser.

Usage of Two-Factor Authentication: In addition to the usage of a master password, considering another authentication mechanism for accessing the credentials is advisable. However, implementing two-factor authentication (2FA), such as SMS texts or voice-based tokens, while enhancing the security of the saved credentials, may significantly reduce the usability of the password manager. If implemented, we recommend that 2FA should not be required in all cases.

Cloud Migration: Browsers should consider saving the secret key used in the encryption/decryption process of credentials in the cloud. Access to this key should indeed only be possible using the master password mentioned earlier, and the 2FA for even better security. If users wish to save credentials in the browser, they would need to log into their profile and define the master password. Consequently, if a computer is compromised, the key would be inaccessible to malware attempting to decrypt the credentials. While this approach may slightly decrease usability, its implementation would greatly enhance security.

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, concerns persist regarding how users store their passwords securely. While password managers offer a solution, BPM face challenges in securely storing data locally. In this paper, we conducted a comprehensive review of how BPM store their credentials and proposed the potential implementation of various security measures to grant credential safeness. Additionally, we introduced a new malware, Dvorak, capable of gathering the necessary information of five different browsers to decrypt the credential databases of BPM and access the user's credentials in plain text. We also advocate for a thorough review of different methods for password encryption across all browsers. Finally, we conclude with the importance of ongoing research to evaluate the progress of both password managers and BPM, with a focus on enhancing security and usability aspects.

REFERENCES

Al-Bataineh, A. and White, G. (2012). Analysis and detection of malicious data exfiltration in web traffic. In *2012 7th International Conference on Malicious and Unwanted Software*, pages 26–31.

Al-Mhiqani, M. N., Ahmad, R., Zainal Abidin, Z., Yassin, W., Hassan, A., Abdulkareem, K. H., Ali, N. S., and Yunos, Z. (2020). A Review of Insider Threat Detection: Classification, Machine Learning Techniques, Datasets, Open Challenges, and Recommendations. *Applied Sciences*, 10(15):5208.

Al-Mohannadi, H., Mirza, Q., Namanya, A., Awan, I., Cullen, A., and Disso, J. (2016). Cyber-Attack Modeling Analysis Techniques: An Overview. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 69–76.

Alabdan, R. (2020). Phishing Attacks Survey: Types, Vectors, and Technical Approaches. *Future Internet*, 12(10):168.

Alshamrani, A., Myneni, S., Chowdhary, A., and Huang, D. (2019). A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities. *IEEE Communications Surveys & Tutorials*, 21(2):1851–1877.

Ashcraft, A. (2022). CryptUnprotectData function. Technical report, Microsoft Learn.

Biswas, N. (2021). Using Webhooks at the Site. In Biswas, N., editor, *Advanced Gatsby Projects*, pages 133–147. Apress, Berkeley, CA.

Ciampa, M. (2013). A Comparison of User Preferences for Browser Password Managers. *Journal of Applied Security Research*, 8(4):455–466.

Clévy, L. (2013). Firepwd: An open source tool to decrypt Mozilla protected passwords.

Dan Wesley, Jeff Borsecnik, S. C. (2023). Microsoft Edge password manager security.

Dell'Amico, M., Michiardi, P., and Roudier, Y. (2010). Password Strength: An Empirical Analysis. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. ISSN: 0743-166X.

Deneut, T. (2022). Security: General security scripts.

Gabe Turner, Aliza Vigderman, C. B. (2023). Password Manager Industry Report and Market Outlook in 2023.

Gasti, P. and Rasmussen, K. B. (2012). On the Security of Password Manager Database Formats. In Foresti, S., Yung, M., and Martinelli, F., editors, *Computer Security – ESORICS 2012*, Lecture Notes in Computer Science, pages 770–787, Berlin, Heidelberg. Springer.

Lawrence, E. (2020). Local Data Encryption in Chromium.

Luevanos, C., Elizarraras, J., Hirschi, K., and Yeh, J.-h. (2017). Analysis on the Security and Use of Password Managers. In *2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 17–24.

Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. (2018). Overview of cryptography. In *Handbook of Applied Cryptography*, pages 35–39. CRC press.

Mistele, K. (2021). Stealing Saved Browser Passwords: Your New Favorite Post-Exploitation Technique.

Muslim, A. A., Budiono, A., and Almaarif, A. (2020). Implementation and Analysis of USB based Password Stealer using PowerShell in Google Chrome and Mozilla Firefox. In *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, pages 421–426.

Nelson, R., Shukla, A., and Smith, C. (2020). Web Browser Forensics in Google Chrome, Mozilla Firefox, and the Tor Browser Bundle. In Zhang, X. and Choo, K.-K. R., editors, *Digital Forensic Education*, Studies in Big Data, pages 219–241. Springer International Publishing, Cham.

Nissim, N., Yahalom, R., and Elovici, Y. (2017). USB-based attacks. *Computers & Security*, 70:675–688.

Oesch, S. and Ruoti, S. (2020). 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)*, pages 2165–2182. USENIX Association.

Qian, C., Koo, H., Oh, C., Kim, T., and Lee, W. (2020). Slimium: Debloating the Chromium Browser with Feature Subsetting. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pages 461–476, New York, NY, USA. Association for Computing Machinery.

Sajid, M. S. I., Wei, J., Abdeen, B., Al-Shaer, E., Islam, M. M., Diong, W., and Khan, L. (2021). SODA: A System for Cyber Deception Orchestration and Automation. In *Proceedings of the 37th Annual Computer Security Applications Conference*, ACSAC '21, pages 675–689, New York, NY, USA. Association for Computing Machinery.

Shabtai, A., Elovici, Y., and Rokach, L. (2012). Introduction to Information Security. In Shabtai, A., Elovici, Y., and Rokach, L., editors, *A Survey of Data Leakage Detection and Prevention Solutions*, SpringerBriefs in Computer Science, pages 1–4. Springer US, Boston, MA.

Silver, D., Jana, S., Boneh, D., Chen, E., and Jackson, C. (2014). Password managers: Attacks and defenses. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 449–464, San Diego, CA. USENIX Association.

Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice*. Prentice Hall Press, USA, 7th edition.

Tetmeyer, A. and Saiedian, H. (2010). Security Threats and Mitigating Risk for USB Devices. *IEEE Technology and Society Magazine*, 29(4):44–49.

Ullah, F., Edwards, M., Ramdhany, R., Chitchyan, R., Babar, M. A., and Rashid, A. (2018). Data exfiltration: A review of external attack vectors and countermeasures. *Journal of Network and Computer Applications*, 101:18–54.

Vossaert, J., Lapon, J., and Naessens, V. (2014). Out-of-Band Password Based Authentication towards Web Services. In De Strycker, L., editor, *ECUMICT 2014*, Lecture Notes in Electrical Engineering, pages 181–191, Cham. Springer International Publishing.

Yang, B., Chu, H., Li, G., Petrovic, S., and Busch, C. (2014). Cloud Password Manager Using Privacy-Preserved Biometrics. In *2014 IEEE International Conference on Cloud Engineering*, pages 505–509.

Yicong, J. O. (2020). Decrypt chrome passwords.

Zhao, R. and Yue, C. (2013). 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*, CODASPY '13, pages 333–340, New York, NY, USA. Association for Computing Machinery.