A SECURE PASSWORD MANAGER

[1]Yasasw. Kumarakalva, [2]Vidya Shankar. G. S, [3]Shreekara. K. K, [4]Sridhar. P. H, [5]Asha. G. R
[1,2,3,4] IV year B.E. (2016-2017), [5] (Under the guidance of) Assistant Professor
[1,2,3,4,5] Department of Computer Science and Engineering (CSE)
[1,2,3,4,5] BMS College of Engineering, Bangalore, India.

Abstract— Nowadays, we find ourselves having numerous web accounts for email, banking, blogging, forums, online shopping, cloud storage and other services. The choice of passwords for these numerous web accounts presents us with a dilemma. By using a single password for all of these accounts, if the intruder gains access to any one of the passwords, he would be having access to the other user's accounts such as his banking passwords, email, online shopping passwords as well. On the other hand, by using different passwords for different systems, users may have the tendency to choose easy-to-remember or weak passwords and there is also a high chance of users forgetting their passwords, increasing the associated user support and operation overheads for password resets.
To solve this problem, we try to develop a secure password manager which is capable of locally storing multiple user accounts and passwords relieving the users from their burden and challenging task of remembering their multiple accounts and passwords. The data is encrypted using AES encryption algorithm ensuring maximum security to the user and the password manager is incorporated within the browser as an extension ensuring high usability and ease of use to the end users.

Index Terms— Password Manager, Sniffing, Spoofing, Password Overload,AES, IndexedDB, File System, Headless Browser, Automatic password change.

I. INTRODUCTION

Passwords are a basic fundamental necessity for information security and cannot be replaced in the near future because of its ease of use and deployment advantages. Passwords are used as a first-line of defense in securing almost all electronic information, networks, servers, devices, accounts, databases etc. Using text-based passwords is the most well-known method for authentication despite its security weaknesses and vulnerabilities. The common security risks are:

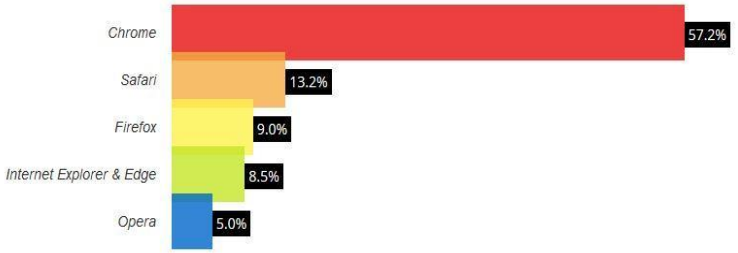
- Over the shoulder Attack: As the name suggests, a password can be obtained by looking over one's shoulder or by indirect monitoring using a camera.
- Brute Force Attack: As passwords are of finite length comprising of alphanumeric characters, an attacker can try out all the possible combinations with the help of a powerful computer until a valid password is found.
- Sniffing Attack: If a password is sent over a network unencrypted, it could be captured by network sniffing tools or the attacker can use a keylogger to capture the password while it is being typed.
- Login Spoofing Attack: The attacker sets up a fake login screen which looks and feels very similar to the real login screen. When a user types his credentials in such screen, the password would be transmitted to the attacker.

Password security heavily relies on the usage of strong passwords and protecting them from being guessed or stolen by an attacker. However, the downside to this is that strong passwords which are sufficiently long and random are often difficult to remember by the users.[6-9]. A research done shows that the passwords chosen by users are easy to compromise by attackers [1,2,3,4,5]. One more noticeable problem with the widespread application of passwords is re-usage of passwords. Users may make use of several services on the Internet, which requires them to create a new username/password tuple for the purpose of registration. Another research has showed that users deal with this password overload by re-using the same or passwords with very little changes for their multiple accounts. [10,11].

The solution to all the aforementioned problems is a Password Manager. A password manager basically stores all the user's online passwords in an encrypted form. A single master key password is needed to decrypt and access the password database. The password manager is often incorporated with a random password generator which helps the user in creating secure and unique passwords with ease. Another useful feature is the synchronization of password databases across multiple devices so that user can access his online accounts from any location and from any device provided that he is using the password manager on that device. The password manager works by capturing the user's credentials the first time they are entered and are later pre-filled on the user's behalf on further visits to the websites. Therefore, the main benefit observed here is that users need not remember any of their passwords for their various online accounts.

II. IMPLEMENTATION

Our initial approach was to create a platform specific application, but with the advent of Web API's, browsers have become far more powerful thereby eliminating the need of a platform specific application[17]. We have chosen Chrome to develop our extension because of the amount of development tools available with chrome and also because of the amount of users using chrome is higher than any other Web Browser as depicted in the graph[12]. We can use the same code to make the extension work on other browsers as well by using browser specific APIs.



For storing the user's credential details, we have used Indexed Database API which stores data inside the user's browser. Indexed Database API is a low-level asynchronous API for storing structured data, files and blobs at the client side which is based on HTML5 , thus making it browser independent. It is a transactional database system similar to SQL-based RDBMS but it uses a JavaScript-based Object oriented database in contrast to the fixed tables used in SQL-based RDBMS. IndexedDB allows us to store and retrieve objects that are indexed with a Key. It uses Indexing to enable fast searching of data. It uses DOM events to display the results. The applications using IndexedDB API works both online as well as offline regardless of network availability [13].

FileSystem API is a chrome-specific API based on HTML5 that can be used for storing the user's data in a sandboxed section of the user's local file system ensuring high security to the user. The FileSystem API is broken down into three major components. First, for reading and manipulating files it uses File/Blob,FileList, FileReader API's. Second, for creating and writing into files, it uses Blob() and FileWriter. Third, for Accessing Directories and File System it uses DirectoryReader, FileEntry/DirectoryEntry, and LocalFileSystem [14].

After storing the user's credentials, it is encrypted using a AES 256-bit encryption. The reason for choosing AES is because it is been proven to be unbreakable with its 256 bit key. A 256-bit key means that there are 2256 key possibilities which would require 234 years to find the right combination of the key even if all the supercomputers are involved in breaking the AES algorithm which is practically impossible [15].

After the encryption, the encrypted password is broken down into four different pieces and stored in different directory locations further adding to the security aspect of the password manager. This is accomplished by using the Jumbling Salting algorithm [16]. we use this Decentralised form of storage because it makes hacking attempts more difficult and even if anyone were to gain access to any one of the files, he/she cannot gain access to the database as we have divided the encrypted database in a nonlinear way. This is why we use the FileSystem API instead of just a database because using a custom form of storage is much more difficult to crack.

Backup of the stored passwords which is another important aspect of password managers in case the system crashes or all the data is lost during unexpected circumstances. Because we maintain a purely local footprint, this feature is very essential as the users depend on password managers for their daily needs.The stored passwords are backed up to the user's private cloud of their choice in case of emergency. This also allows synchronisation across multiple devices making it user-friendly.

One of the main functions of the browser extension is the Auto-Fill function. Many times users end up typing the same data fields again and again like address,name etc. We make this easier by adding this function using Chrome's Content Script Feature. This allows the extension to perform certain changes to the active page. We use this to manipulate the input fields of the page. We also use another feature provided by Chrome called Event Pages which allows us to continuously listen to changes made by the user in the input fields so that they can be "captured" and stored for further use, so that the user does not have to type it again if required. This feature is called Automatic Capture. Both these features go hand in hand to create a seamless Web Browsing experience.

We prefer using Progressive Web Apps over native to further support the extension with additional functionalities. Progressive Web Apps combines the best of the Web and also the best of the Apps. It does not require any installation and there is an option to make the app icon appear on the home screen, just like a normal app. A progressive web app works across all devices such as desktop, mobile, tablet irrespective of the form factor and operating system, thus making them responsive and independent of platform specific code like native apps. They are faster to load and require lesser resources. They do not have full access to local storage like native apps but they are mainly meant for apps and services that use a lot of cloud storage.

It uses HTTPS for communication ensuring the integrity of the data, and is designed to work offline as well as online eliminating network related constraints, and the application can be shared via a URL eliminating the need for installation thereby saving a lot of time and ensuring ease of use applications.We use this to provide special functionality such as two-factor authentication and fingerprint authentication, things which we cannot accomplish using just an extension because of restrictions put in place because of safety concerns.

Lastly, the main feature we aim to build is Automatic Password change. This is a tricky feature to implement as we need browser permission and write a code for each website fot changing its password; and the final hurdle is to make it usable so that a normal user without any coding knowledge can use it. This is the feature that makes the password manager 'platform dependant' as there is installation of certain softwares required. And there is also the limitation that only a few websites can be supported as code has to be written and updated for changing each website's password.

We use headless browsers to implement the above feature. we prefer Selenium due to its high usability and feature set. We use HTMLUnitDriver in Selenium to use the software as a headless browser. PhantomJS can also be used instead of it. The code is written in java as there is more support for it, and after successful execution of code (to successfully change the password of a website) the code is exported as a JAR executable file. This file is then embedded into an executable file (eg: .exe on windows) so that when the app is called upon, it executes a certain set of commands automatically without the user's intervention. An app can be called from a browser extension using messaging. For example, chrome provides a native messaging api to talk to external apps. It should be kept in mind that websites change and thus the code must be updated regularly in order to work properly.

III. CONCLUSION

Password managers are the most underrated piece of software these days. People need them now more than ever but do not understand its importance. Hacking attempts are growing day by day and are only going to increase in the future if nothing is done. Companies are doing all they can to protect their servers from these attacks but the vulnerable point is on the user's side when they use simple, easily crackable passwords. Many of the recent hacking attempts have been on the user's side. Since these users cannot remember complicated passwords, password managers are the only solution. We plan to spread awareness about this important software and provide open-source code so that the future of personal security improves by the slightest bit.

Although PRogressive web apps are great in theory and have been touted to be the future of apps, they have only been around since april 2016 when google introduced them to the world and thus there might be some early compatibility problems. Apple does not officially support progressive web apps yed, but they work on ios. Many developer conferences have been held and the future of this technology seems good.

IV. ACKNOWLEDGEMENTS

The work reported in this paper is supported by the college through the TECHNICAL EDUCATION QUALITY IMPROVEMENT PROGRAMME [TEQIP-II] of the MHRD, Government of India.

REFERENCES

[1] M. Bishop and D. V Klein. Improving system security via proactive password checking. Computers & Security, 14(3):233–249, 1995.

[2] J. Bonneau. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. Security and Privacy (SP), 2012 IEEE Symposium on, pages 538–552, 2012.

[3] M. Dell'Amico, P. Michiardi, and Y. Roudier. Password Strength: An Empirical Analysis. In INFOCOM, 2010 Proceedings IEEE, pages 1–9, 2010.

[4] D. Florencio and C. Herley. A large-scale study of web password habits. Proceedings of the 16th international conference on World Wide Web, pages 657–666, 2007.

[5] D. Malone and K. Maher. Investigating the distribution of password choices. In Proceedings of the 21st international conference on World Wide Web, WWW '12, pages 301–310, New York, NY, USA, 2012. ACM

[6] A. Adams and M. A. Sasse, "Users are not the enemy," Commun. ACM, vol. 42, no. 12, pp. 40–46, 1999.

[7] D. C. Feldmeier and P. R. Karn, "Unix password security – ten years later," in Proc. of CRYPTO, 1989.

[8] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of passwords and people: Measuring the effect of password-composition policies," in Proc. of CHI, 2011.

[9] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: Empirical results," IEEE Security and Privacy, vol. 2, no. 5, pp. 25–31, 2004.

[10] S. Gaw and E. W. Felten. Password management strategies for online accounts. In Proceedings of the second symposium on Usable privacy and security, SOUPS '06, pages 44–55, New York, NY, USA, 2006. ACM.

[11] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor. Encountering stronger password requirements: user attitudes and behaviors. In Proceedings of the Sixth Symposium on Usable Privacy and Security, SOUPS '10, pages 2:1–2:20, New York, NY, USA, 2010. ACM.

[12] https://www.w3counter.com/globalstats.php

[13] https://www.tutorialspoint.com/html5/html5_indexeddb.htm

[14] https://www.html5rocks.com/en/tutorials/file/filesystem/

[15] http://canadiancloudbackup.com/safe-safe-aes-256-encryption-data/

[16] Churi, Prathamesh P.; Ghate, Vaishali; Ghag, Kranti -- [IEEE 2015 International Conference on Science and Technology (TICST) - Pathum Thani, Thailand ""Jumbling- Salting: An Improvised Approach for Password Encryption

[17] Password Manager - Automatic Password Change Using Headless Browsers : A Survey