International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 6, Issue 12, ISSN No. 2455-2143, Pages 93-97
Published Online April 2022 in IJEAST (http://www.ijeast.com)

SAFE VAULT: A PASSWORD MANAGER

Baibhav Singh and Dr. Elizabeth Jesi

Abstract: **Text passwords have long been the most popular user authentication method, with a huge number of Internet services using them. Users are faced with the almost insurmountable difficulty of generating and managing a huge number of site-unique and strong (i.e., non-guessable) passwords if they follow suggested practice. A password generator, i.e., a client-side technique that generates (and regenerates) site-specific strong passwords on demand with minimal user input, is one solution to handle this problem. Safe Vault, a password generating scheme previously defined as part of a wider review of such schemes, is detailed specified and analyzed in this work. Safe Vault was created to address flaws that had previously been recognized in password generators, and it uses unique approaches to do so. Safe Vault allows you to create passwords that meet critical real-world needs, such as required password updates, pre-specified passwords, and passwords that meet site-specific requirements.**

I. INTRODUCTION

Text password authentication is frequently used to authenticate users to online services, despite its well-publicized flaws. Biometrics, tokens, and multi-factor authentication have all been proposed as alternatives to simple password authentication [4]. Single-factor password authentication, on the other hand, is still commonly employed. Furthermore, the number of commonly used password-protected services has increased dramatically in recent years, resulting in an increase in the number of passwords users are expected to remember. The usage of text passwords has a number of drawbacks [2]. These problems can be divided into two categories: user-related and online service-related. User-related issues include:

- The high number of passwords required for Internet services is likely to overwhelm consumers, leading to the usage of the same password for several accounts.
- When forced to change a password, users will frequently use easily guessable passwords, such as their date of birth, pet's name, or anniversary date.
- When forced to change a password, users will frequently make minor changes, such as adding a serial number.

The following are examples of online service-related concerns that might make it nearly hard for consumers to remember all of their passwords:

- Many websites have a complicated password policy, such as demanding a minimum number of characters in passwords or requiring passwords to include or exclude specified characters.
- Some sites require users to update their passwords on a regular basis, such as once every 90 days.

A number of password generator techniques have been developed to address these difficulties [3, 6, 7, 12, 13, 14], which produce strong (i.e., tough to guess) and random-looking passwords and regenerate them as needed. Safe Vault is a password generator that creates site-specific passwords for internet services on demand.

The remainder of the paper is organized in the following manner. Section 2 explains what password generators are and provides a generic paradigm for them (based closely on [10]). Section 3 builds on the previous overview by providing a high-level description of Safe Vault. This is followed by a full description of how Safe Vault works in section 4. Section 5 examines the features of Safe Vault, focusing on how it addresses recognized flaws in similar methods. Finally, section 6 brings the paper to a close.

II. A GENERAL MODEL

a. Definition

This paper is about password generators, which are programmes that generate site-specific passwords to make password management easier for end users.

Passwords are generated on demand from a small set of easily remembered inputs. Note that the term has also been used to describe schemes for generating random or pseudorandom passwords that the user is then expected to remember; however, we use the term to describe a system that can generate the necessary passwords on demand and in a repeatable manner and is intended to be used whenever a user logs in. In recent years, a number of such systems have been proposed.

We propose a broad paradigm for such systems in this section, which we will use later in this study to describe our innovative approach, Safe Vault [10]. We note that McCarney [11] has previously discussed the general class of such schemes under the moniker generative password managers.

b. Model

The following are the components of a password generator:

- The password for a certain site is determined by a set of input values. So that the generated password is site-specific, some values must be site-specific. The values could be saved (locally or online) based on the authenticating site's criteria, or user-entered when necessary. These sorts of input can, and frequently do, be combined by systems.

- A password generator function takes the input values and combines them to create a suitable password. Depending on the requirements of the web site doing the authentication, this function could work in a variety of ways. For example, one website may prohibit the use of non-alphanumeric characters in passwords, while another may require the use of at least one of these characters. A password creation function must be customizable in order to be widely applicable.
- The created password can be sent to the authenticating site using a password output technique. For example, the created password could be displayed to the user, who must then type (or copy and paste) it into the right location.

On the user platform, all of this functionality must be implemented. Such an implementation can be done in a variety of ways, including as a standalone application or as a browser plug-in.

c. **Related Work**

Before we go any further, we'll go over some existing approaches for password generation techniques that follow the above model:

- Karp [6] presented the Site-Specific Passwords (SSP) technique in 2002/03, which is one of the earlier schemes of this sort. SSP creates a site-specific password by combining a long-term user master password with a memorable web site name selected by the user.
- Password Sitter [13], developed by Wolf and Schneider in 2006, generates a site-specific password based on a long-term user master password, the user identification, the application/service name, and a few adjustable criteria.
- PwdHash [12], created by Ross et al., produces a site-specific password by combining a long-term user master password, data linked with the web site, and (optionally) a second global password saved on the platform to create a site-specific password.
- ObPwd originally appeared in 2008, thanks to Mannan et al. [1, 7, 8, 9]. It adopts a different approach, producing a site-specific password based on a user-selected (site-specific) object (e.g., a file) and a number of optional factors, such as a long-term user password (referred to as a salt) and the web site URL.
- Finally, PALPAS [5] uses server-provided password policy data, a saved secret master password (the seed), and a user-specific secret value (the salt) that is synchronized across all user devices utilizing the server to generate passwords that conform with site-specific requirements.

There are also a number of widely available applications that follow the general idea, some of which are briefly discussed below. As browser extensions, the following schemes are available:

- Rnd Phrase is a web-based password generator and Firefox add-on. It creates site-specific passwords based on a specified salt (one per user), the host name, and the password given by the user. Only the password must be remembered by the user.
- Pwd Hash port is a Pwd Hash-based Opera add-on.

Apps for Android Phones: The two apps listed below are available for Android phones.

- Passwords generated by Advanced Password Generator can be copied and shared. The length and character(s) to include/exclude from a created password can be modified.
- Password generator produces passwords based on a set of parameters that can be customized, including a user salt. The system provides an assessment of the generated password's strength.

d. **Issues of existing schemes**

Before going into depth about how Safe Vault works, there are a few fundamental issues that affect all (or nearly all) of the preceding password generators. These concerns prompted the creation of Safe Vault, which has unique features aimed at resolving these challenges.

Password creation and maintenance: As previously stated, there is no problem if a user is already using the password generator when registering with a website - the user can just register whatever value the system generates. If the user has chosen and registered passwords with a variety of websites before using the password generator, all of these passwords must be changed to whatever the password generator generates. If a user has built relationships with multiple sites, this could be quite inconvenient, and it could be a significant barrier to system adoption.
When a user decides to change a website password, similar issues exist, for example, if the site requires users to change their passwords on a regular basis. The user's sole option will be to modify one of the password-generating inputs, such as the object (if a digital item is used as an input) or the user site name. If the user does not choose any of the inputs used to construct a password, changing it may be impossible.

Using different platforms If a user uses different platforms, such as a desktop and a smartphone, then any locally stored configuration data will cause complications.

Issues with password policies Another general issue is the necessity to automatically generate passwords in a site-specific format, which is a challenge that none of the previously proposed schemes, except PALPAS, have effectively handled. Some existing schemes allow users to customize a generated password, but the user must manually identify the website's needs and modify the parameters accordingly. Horsch and his co-authors [8, 9] have spent a lot of time looking into how to automatically generate a password that is tailored to a website's specific requirements.

III. SAFE VAULT: MODEL

e. Safe Vault Components

The Safe Vault server and the Safe Vault client software are the two major components of Safe Vault. The Safe Vault server stores user information such as the username and website-specific password regulations (i.e., the types of passwords that a given site will allow). The Safe Vault client programme has a graphical user interface and automatically generates site-specific user passwords based on a set of inputs.

f. Overview

Following that, based on the overall model presented in the previous part, we provide a high-level description of Safe Vault. We begin by describing the scheme's three key components: input values, password generation function, and output mechanism, as well as an initial implementation strategy.

- **Values to be entered.** We recommend using a variety of input types, as described below, as well as those used in prior schemes.
- ❖ **The URL** of the site for which the software is generating a password is the site name.
- ❖ **A username/email** which will be used for authenticating along with the password
- ❖ **A password policy** sets the set of site-specific password restrictions (for example, a length constraint and/or a minimum quantity of certain character classes). Many websites have very detailed policies that are the result of haphazard system design decisions.
- The input values are combined in the password creation stage to produce the appropriate site-specific password.
- The created password is automatically copied to the specified password field, allowing for password output and use.

Safe Vault will be implemented (at least initially) as a web application. We plan to investigate both stand-alone programmes and browser add-on capabilities in the future for use.

g. Stored Data

Safe Vault, as previously stated, requires access to a variety of configuration data in order to run automatically, and this data must be kept somewhere. Data can be stored in two places: the Safe Vault server and the Safe Vault client, and Safe Vault makes use of both. The configuration data on the server is kept long-term, i.e., for the duration of the user account on the server; the data on the client is kept either short-term, often for the duration of a session, or long-term, i.e., for the duration of the software installation on the client platform. The many types of stored data are summarized here.

i. Server-Stored Data

The Safe Vault server stores the following user-specific configuration data:

- The user’s account name
- The user’s email address
- The master password (hash)
- Generated passwords (encrypted)

ii. Client-Stored Data

The Safe Vault client stores the following information for a limited time:

- The session password
- A hash of the master password

IV. ANALYSIS

Following that, we'll look at the scheme's security features.

h. Trust Relationships

Clearly, the Safe Vault server must be trusted to some extent by the user because passwords cannot be produced correctly if it delivers wrong data. However, because it only has access to anhashed copy of the master password, which is used to encrypt all passwords, it cannot learn user passwords, and so can only be considered partially trusted.

i. Threat Model

Safe Vault's security and correct operation are predicated on a number of crucial assumptions, which we list.

i. The Safe Vault client is expected to be up to date and free of exploitable flaws.
ii. Because passwords are created and used by this device, it is considered that it is uncompromised. If the browser, for example, gets hacked, the generated passwords are obviously at risk.

j. Security Properties

We'll wrap up this topic by looking at whether the Safe Vault architecture achieves the needed features.

i. If an unauthorized party gains access to the Safe Vault server database, it will not be able to access any user credentials right away. If the unauthorized party obtains the encrypted master password and the session password used to encrypt the master password is poorly chosen, the attacker may be able to brute-force the session password and learn the master password, as stated under 2 above. This supports the idea of the Safe Vault server encrypting the database in addition to the database itself, offering protection against the compromise of stored user data.

ii. The assumption that all communications between the client and server are TLS-protected prevents the first sort of attack. A pinned certificate will be used to authenticate the server. The server will not be explicitly authenticated at the client end of the link, but the existence of the proper user will be confirmed by checking that the correct hash of the session password is delivered over the link.

V. CONCLUSION

We have detailed the design of the Safe Vault password generator in this paper. This server can only be trusted in part because it lacks the ability to recover individual user passwords. Of course, while the system appears to function in principle, it must be seen how well it performs in practice. Safe Vault is being built as a prototype, which will be used to conduct user trials to ensure that the intended high degree of usability can be attained. In the future, we intend to publish a paper about these trials.

VI. REFERENCES

[1]. Robert Biddle, Mohammad Mannan, Paul C van Oorschot, and Tara Whalen. User study, analysis, and usable security of passwords based on digital objects. IEEE Transactions on Information Forensics and Security, 6(3):970–979, 2011.

[2]. DineiFlorˆencio, Cormac Herley, and Paul C van Oorschot. Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts. In Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20–22, 2014, pages 575–590.USENIX Association.

[3]. J Alex Halderman, Brent Waters, and Edward W Felten. A convenient method for securely managing passwords. In Allan Ellis and Tatsuya Hagino, editors, Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005, pages471–479. ACM, 2005.

[4]. Cormac Herley and Paul C van Oorschot. A research agenda acknowledging the persistence of passwords. IEEE Security & Privacy,10(1):28–36, 2012.

[5]. Moritz Horsch, Andreas H¨ulsing, and Johannes A. Buchmann. PALPAS password less password synchronization. In 10th International Conference on Availability, Reliability and Security, ARES 2015, Toulouse, France, August 24-27, 2015, pages 30–39. IEEE Computer Society,2015.

[6]. Alan H Karp. Site-specific passwords. Technical Report HPL-2002-39(R.1), HP Laboratories, Palo Alto, May 2003. Available at https://www.labs.hpe.com/publications/HPL-2002-39(R.1).

[7]. Mohammad Mannan and P C van Oorschot. Passwords for both mobileand desktop computers: ObPwd for Firefox and Android. USENIX;login, 37(4):28–37, August 2012.

[8]. Mohammad Mannan and Paul C van Oorschot. Digital objects as passwords. In Niels Provos, editor, 3rd USENIX Workshop on Hot Topics inSecurity, HotSec'08, San Jose, CA, USA, July 29, 2008, Proceedings. USENIX Association, 2008.

[9]. Mohammad Mannan, Tara Whalen, Robert Biddle, and P C van Oorschot. The usable security of passwords based on digital objects: From design and analysis to user study. Technical ReportTR-10-02, School of Computer Science, Carleton University, February 2010.https://www.scs.carleton.ca/sites/default/files/tr/TR-10-02.pdf.

[10]. Fatma Al Maqbali and Chris J. Mitchell. Password generators: Oldideas and new. In Sara Foresti and Javier Lopez, editors, Information Security Theory and Practice — 10th IFIP WG 11.2InternationalConference, WISTP 2016, Heraklion, Crete, Greece, September 26-27,2016,Proceedings, volume 9895 of Lecture Notes in Computer Science,pages 245–253. Springer, 2016.

[11]. Daniel McCarney. Password managers: Comparative evaluation, design, implementation and empirical analysis. Master's thesis, CarletonUniversity, August 2013. Available at https://danielmccarney.ca/assets/pubs/McCarney.MCS.Archive.pdf.

[12]. Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John CMitchell. Stronger password authentication using browser extensions.In Patrick McDaniel, editor, Proceedings of the 14th USENIX SecuritySymposium, Baltimore, MD, USA, July 31 – August 5, 2005, pages17–32. USENIX Association, 2005.

[13]. Ruben Wolf and Markus Schneider. The passwordsitter. Technical report, Fraunhofer Institute for Secure Information Technology (SIT), May 2006. [19] Ka-Ping Yee and KragenSitaker. Passpet: Convenient password management and phishing protection. In Lorrie Faith Cranor, editor, Proceedings of the 2nd Symposium on Usable Privacy and Security, SOUPS 2006, Pittsburgh, Pennsylvania, USA, July 12-14, 2006, volume 149 of ACM International Conference Proceeding Series,

pages 32–43. ACM, 2006.

[14]. Ka-Ping Yee and KragenSitaker. Passpet: Convenient password management and phishing protection. In Lorrie Faith Cranor, editor, Proceedings of the 2nd Symposium on Usable Privacy and Security, SOUPS 2006, Pittsburgh, Pennsylvania, USA, July 12-14, 2006, volume 149 of ACM International Conference Proceeding Series, pages 32–43. ACM, 2006.