A Hybrid Password Manager

Alan H. Karp
alanhkarp@gmail.com

Abstract

The most popular password managers store passwords so you don't have to remember them. That choice imposes costs on the company providing the password manager and risks on its users. Another option is to calculate passwords as they are needed. SitePassword is such a calculator that provides the key features that users want in their password manager – remembering metadata, synchronizing across machines, and autofilling the login form. In addition, it can store passwords that you choose without the cost and with no more risk than password managers that only store your passwords.

1. Introduction

"Dealing with passwords is awful" is a statement few would disagree with. Password managers make the situation less awful, improving both security and usability by making it easier for you to have a different, strong password for every site. Of course, you probably care at least as much about ease of use, trust in the the password manager, and the ability to get your passwords from any device as you do the strength and uniqueness of your passwords.

There are two kinds of password managers. The most widely used of them remember passwords and other metadata, such as userids. They make the stored passwords available from any machine by using encrypted databases and cloud storage. In the following they're referred to a *password stores*.

Remembering your passwords adds cost for the company providing the password store to handle user accounts and to pay for the necessary cloud storage. It imposes risks on the user because the stored passwords can be stolen.[1] In addition, some companies have dropped support for their password databases, *e.g*, remembear.com, leaving their users in the lurch.

There is also a privacy issue with these password managers in that they know where and when you login.

The second type, denoted *password calculators* to distinguish them from tools that merely generate a password that is managed by other means, don't need to store passwords. They combine a super

[1] https://krebsonsecurity.com/2023/09/experts-fear-crooks-are-cracking-keys-stolen-in-last-pass-breach/

password with other data to calculate a strong password when you need to log in. They will be.

SitePassword is a password calculator designed for usability and security that supports the features users want in a password manager. There is no privacy issue, since SitePassword doesn't use the network. SitePassword can also store passwords the user has created making it a hybrid of the two types.

2. SitePassword Overview

SitePassword is a browser add-on that runs in Firefox and most Chromium browsers. It has three components. The popup, with the user interface shown in Figure 1, is where you provide the necessary metadata. A content script running on the page with the login form is responsible for finding the userid and password fields on the page and filling them in on request. A service worker manages the metadata and calculates the site password when the content script asks for it.

Unlike most password calculators, SitePassword provides features usually found only in password stores. In addition to remembering your settings and synchronizing them across machines, SitePassword finds and autofills both userid and password fields and provides a web page for use on devices where the extension is not installed.

When you first encounter a page with a login form at a given domain, the password field contains a placeholder *Click*

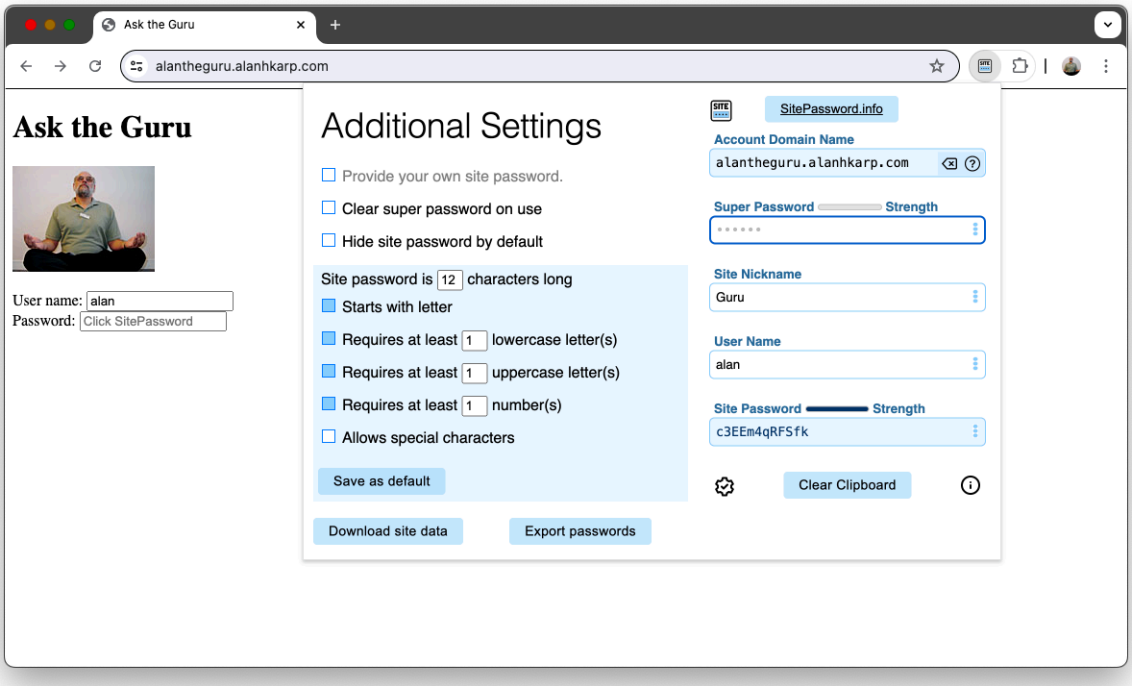

Figure 1: The SitePassword popup showing all available options.

SitePassword.[2] Clicking the SitePassword icon opens the popup. As you fill in the form the site password field updates on every keystroke, making it clear how uncorrelated the generated site passwords are.

As you mouse over to the login form, the placeholder changes to *Click here for password*. Click and your password gets filled in. When you return to that login page on any machine that synchronizes your bookmarks and has the extension installed, your userid is automatically filled in. You only need to click on the password field. The result is that using SitePassword is even easier than typing the same, weak password for every site.

There are times when a calculated password can't be used. Perhaps the password rules can't be satisfied by the available settings, as is the case for the five sites listed in Appendix 2. Perhaps you can't figure out how to change your password at a site, although the *Forgot Password* option should always suffice. Perhaps you are given a password that you are not allowed to change, as done by some companies.

To support such cases, SitePassword allows you to specify a password for a site. Simply fill in the form, check the *Provide Password* check box, and edit the site password field. Your password XOR'd with the computed site password is stored with the other settings for the domain. Once created in this way, SitePassword will insert the password you provided when you click on the password field.

SitePassword also provides a web page, shown in Figure 2, that you can use to get your passwords when the extension is not available, such as on your mobile devices. You don't even have to remember your settings if you synchronize bookmarks to the device. Just navigate to the corresponding bookmark. You'll be taken to sitepassword.info with your settings filled in.

You can also download a formatted table of your settings that you can print for times when your bookmarks are not available. Losing this piece of paper adds only a small amount of vulnerability as discussed in Section 4.

3. Usability Considerations

People who don't use a password manager often give usability as the main reason [1]. SitePassword was designed to be as usable as a simple game. Each step should be obvious; you should be able to experiment without worrying about breaking something; warnings and tooltips tell you what to do next. Additionally, SitePassword comes with extensive instructions, as shown in Figure 2.

SitePassword does the obvious things to make it easier to use. When filling out the form, the field you should be typing

[2] Most of the time. The placeholder text does not appear on some websites. A tooltip provides backup.

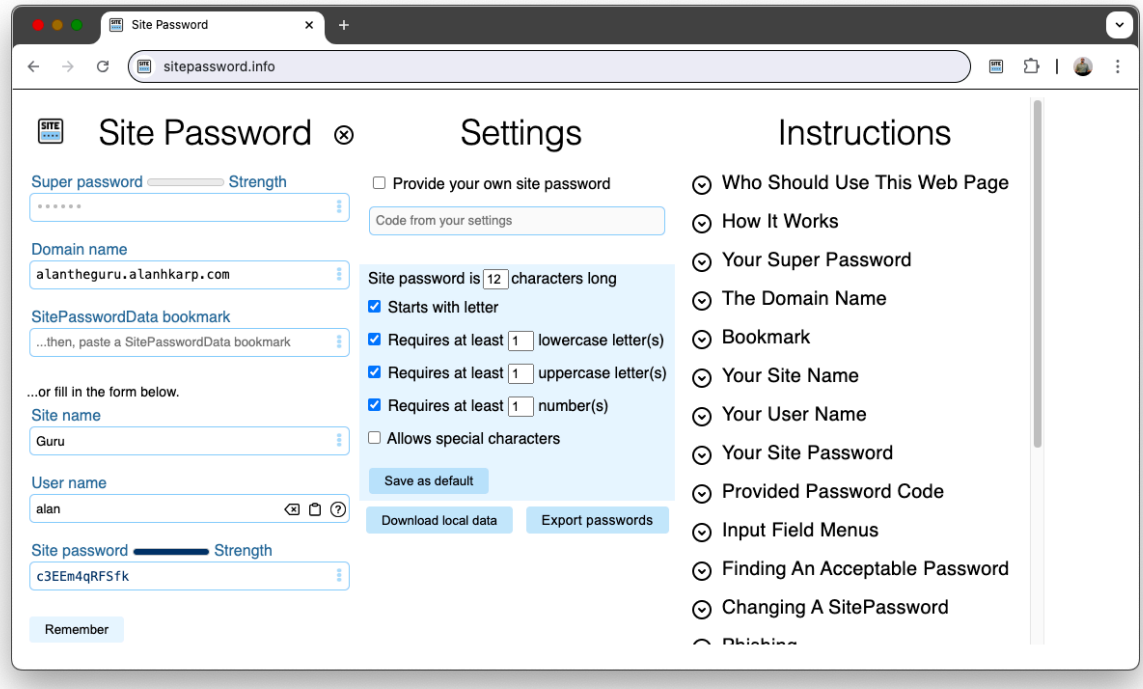

Figure 2: The SitePassword web page showing all available options and the table of contents of the instructions. Note that the calculated password is the same as in Figure 1.

into is in focus. It puts the information you need where you are looking, with tooltips providing additional information without introducing clutter. Affordances that make no sense in context, such as asking to copy the contents of an empty field, are disabled. Finally, reasonable defaults reduce the number of items you need to enter.

Websites often have rather specific password rules, such as how many lower/upper case letters, numbers, and special case characters are needed and even which special characters are accepted. Your only involvement in getting a valid site password is limited to filling out the form in the popup to match the rules. SitePassword automatically finds a suitable site password. .

There are times when you need to type the site password manually, such as when logging into an app on a smart TV. It is also possible that SitePassword won't find the password field on a page that has disabled pasting. Hence, SitePassword uses a font that doesn't make, for example, lower case l (ell) look like upper case I (eye).[3] The alternative of not using visually similar characters makes the site passwords easier to guess.

[3] An earlier version did not follow this advice, resulting in the author needing several tries to log into Netflix on his new smart TV.

One usability hazard is hard to avoid. SitePassword warns about potential phishing, but too many websites make the warning ambiguous by using multiple domain names for logging into the same account. Unfortunately, it is virtually impossible to use the domain name to distinguish phishing from a legitimate use of a different domain name. For example, are the domain names in Figures 1 and 3 for the same account? You can't tell from the trailing part of the domain name because, for example, alanhkarp.com might be providing these domain names as a service to independent parties. Perhaps allan.alanhkarp.com is attempting to phish users of alan.alanhkarp.com.

4. Security Considerations

A guiding principle is that a system cannot be secure unless users are aware of the implications of their actions. To that end, the SitePassword user experience is based on a set of usable security principles [2]. For one, SitePassword makes the easy way the secure way, as noted in Section 3. It also requires a click on the password field, both to avoid a family of attacks [3, p. 37] and to give you control over where your password goes. Finally, potential phishing results in the warning shown in Figure 3.

Of course, that's not enough. It's also

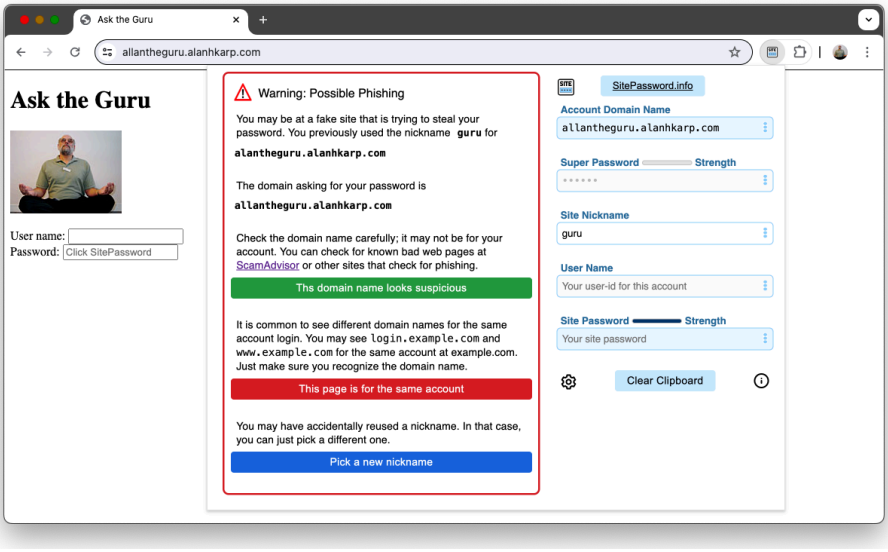

Figure 3: The SitePassword phishing warning.

The fact that most phishing warnings are false positives leaves two choices. Either ignore them or explain them. SitePassword made the latter choice in the hope that the explanation shown in Figure 3 suffices.

necessary to keep the user informed. To that end, SitePassword provides a strength meter for the super and site passwords. Leaving a password on the clipboard is not good practice, so Site-

Password indicates when that might be the case.

Other protections aren't visible to the user. Your super password is never stored. It is kept in session storage, a transient space only available to the extension for the duration of the browser session. Your site passwords are calculated in a way that protects your super password from off-line guessing, as described in Section 5.

In contrast to some other password managers, the callback that fills in the password is only registered if there is a single visible input element with type *password*, making clickjacking more difficult. If that element is in an iframe, the domain name of that iframe is used to select the metadata used to calculate the site password even though that choice leads to more false phishing warnings. Including your user name and site nickname in the computation provides salt to protect your super password from pre-image attacks.

Although most users won't notice, a code review will show that once loaded SitePassword does not use the network. Users aware of this fact know that their passwords are not being exfiltrated by their password manager.

Network independence goes further. SitePassword is self-contained as protection from supply chain attacks. Its two dependencies, zxcvbn [4], which is only used to estimate password strength, and a function to do UTF-8 encoding taken from a widely used SHA-256 library [5] are included in the distribution.

Sandboxed elements are untrusted, so a password manager must disable autofill for any password fields they contain.[4] However, that's not enough. SitePassword also disables input to prevent the user from being tricked to manually providing the password. A tooltip tells the user how to provide one if that's necessary.

Some decisions slightly weaken security to give a big gain in usability. For example, the site password is visible in the popup by default to increase the user's trust in the generated passwords.[5] Any risk is mitigated by the fact that you rarely open the popup, namely to enter your super password at the start of a browser session, to set up a new site password, or when you need to copy/paste your site password. You have the option to change the default.

Another decision was not using special characters and choosing a site password length of 12 as defaults. These decisions made it more likely that the default settings will generate an acceptable site password. You can choose other defaults to meet your specific needs.

---

[4] https://github.com/google/security-research/security/advisories/GHSA-mhhf-w9xw-pp9x

[5] Besides, it's fun watching your site password change as you type.

The default site password length warrants comment. Common wisdom is that a password should have more than 12 characters, but that guidance assumes the site password is something you can remember. The calculated string is effectively a random selection from a 62-character alphabet, giving a 12-character site password roughly 70 bits of entropy. Although it's possible that a random selection of characters is easy to guess, SitePassword only produces those that zxcvbn says require brute force guessing.

The main decision that weakens security is using unencrypted bookmarks to store your settings. There are many benefits.

- There's no need for SitePassword to deal with user accounts or cloud storage.
- Your settings are available on your mobile devices that can't install the extension.
- You won't lose your settings if you need to uninstall and reinstall the extension.
- You can import your bookmarks into other browsers.
- Support for personas is provided by browser profiles.

The risk is low; all an attacker with access to your bookmarks learns is the sites you log into, your userids at those sites, and the lengths of your passwords. That entails some risk, but that knowledge gives no hint of either your super or site passwords.

There are two reasons not to encrypt your settings. First, you select a bookmark by domain name when using the SitePassword web page, which means a key piece of data must be in the clear. Second, is the question of what to use for the encryption key. A key derived from your super password may not work, because the you might have different super passwords for different classes of web sites. It wasn't deemed worth the complication of dealing with that just to protect the only other pieces of information useful to an attacker, your user names, often your name or email address, and site nicknames, both of which are easy to guess. Additionally, an adversary can avoid some guesses by knowing the password length, but the number saved is modest.

There are also mitigating factors. It is hard to get to your bookmarks from a web page, so malware on your machine is the main threat. Such malware is likely to cause bigger problems than exposing your SitePassword settings.

Before being stored with the other settings in the bookmark for the domain, a user-provided password is XOR'd with the password calculated for the site. As a result, an attacker who learns your password for a site and has access to your bookmarks can calculate the computed password for the site but must still mount an offline attack to learn your super password.

The conversion of the hash function output to characters has security implementations. SitePassword starts by producing an array of valid characters for the site. For example, if the site disallows special characters, the array doesn't contain any. Each pair of bytes of the hash is used to select a character of the site password using $c_i = A[b_i \ mod \ |A|]$, where c_i is the i'th character of the site password, b_i is the i'th and $+1$'th bytes of the hash, and A is the alphabet with size $|A|$.

This algorithm introduces modulo bias that slightly weakens the site password. To see why construct an array of 256 characters by repeating the default alphabet of 62 characters and truncating to 256 characters. Each of the first eight characters appears five times while the rest only appear four times.

Fortunately, modulo bias reduces the number of tries needed to guess a site password by only a modest amount. Using two bytes per character makes the modulo bias negligible.

The last step is finding a password acceptable to the site's often arcane password rules. SitePassword leverages the fact that the hash output is effectively random to find one acceptable to the site by iterating a fast hash until one is found.

If an acceptable password is not found in 200 iterations, a value picked to meet the latency requirement, the deterministic algorithm described in Appendix 4 is used. There are no known websites that require that algorithm.

5. Offline Attacks

The biggest threat you face is an offline guessing attack against your super password, which is the same threat you face if a your password store is hacked.

Say that a password database is stolen from a site you use. An adversary then knows your userid, may be able to guess your site password from the stored hash, and can probably guess your nickname for the site. The adversary can then mount an offline guessing attack against your super password.

There are some simple mitigations. The first is choosing a strong super password, which you can verify with the strength meter. Also, nothing says you must have only one super password. You could have one for banking, a second for subscriptions, and a third for sketchy sites. This choice is not readily available to users of password stores.

SitePassword helps by using a slow hash function to compute the site password, 200,000 iterations of PBKDF2 with salt consisting of your user and site names, to produce a key with twice as many bytes as characters in your site password. The number of iterations was chosen to take as long as possible while still meeting the interactive latency requirement on a relatively slow machine.

This approach increases the computation needed for each try. Guessing a good 16-character super password will take a long time and a lot of resources. Specialized hardware can reduce that time by orders of magnitude, but you can make the attacker's problem harder by making your super password stronger than your site passwords, as explained in Appendix 3.

6. Related Work

There are a number of reviews [3] of password stores, so this section focuses on password calculators.

An early, very simple calculator [6] hashed the user's super password with a user selected nickname for the site to produce a password that the user would copy and paste into the password field. LessPass is a modern implementation of this idea.

That calculator was later adapted into the HP Antiphishing Toolbar for Internet Explorer, which added many of the features of a modern password manager. PassPet [7] is an extension of this work.

PwdHash [8] isn't really a password manager, but it is often compared to them. It hashes the user's password for a site with the domain name of the page. The user is still responsible for choosing and remembering a password for each site. Note that it doesn't work for sites that use different domain names for logging into the same account

Password managers often have an option to fill in login forms with no user action, making them vulnerable to a number of attacks [9]. Others fill in all password fields on the page when you click on the extension icon, which is better but still vulnerable to maliciously injected login forms. SitePassword requires a click on the password field you want filled in, avoiding this potential confusion.

There have been several proposals for "password-less" password managers. PALPAS [11] generates site passwords from a high entropy shared secret seed stored on each device and per site salt provided from a central server.

PALPAS protects your super password better than SitePassword, but PALPAS does not do as good a job supporting the use of someone else's machine, perhaps a public one. You must transfer the encrypted shared secret and some other data, which requires a compatible storage device. A SitePassword user needs only remembered information or data that can be copied from a piece of paper. Unlike SitePassword, PALPAS has no provision for user-provided passwords.

AutoPass [12] is like PALPAS in that the user-memorable super password is not part of the password calculation. It, too, relies on a backend server. Also like PALPAS, using AutoPass on someone else's machines is awkward. AutoPass makes one easily correctable mistake — it uses the URL of the login form page in the calculation, making it unusable at sites that use multiple domain names for

logging into the same account. Like PassPet, it caches the result of a slow computation to increase the work factor of a guessing attack. AutoPass uses rejection sampling when converting the hash to a set of characters, but that's less of an issue than for SitePassword because the AutoPass super password is not human memorable. Like SitePassword, AutoPass allows user-provided passwords. Because AutoPass relies on a login password to fetch metadata from the server, it is subject to phishing, but it does allow for account recovery in the event of a forgotten login password. Unlike PALPAS, getting your passwords from someone else's machine is easier than with SitePassword since the server provides all the necessary metadata.

Appendix 1 has notes on all the password calculators in the Chrome store as of March 2023 that provide at least one password manager feature.

7. Future Work

There is still work to be done. A user study would identify any glitches in the user experience. A vulnerability scan[6] reports only warnings, such as the use of === in non-cryptographic calculations. Still, a comprehensive security review of both the design and the code should also be done. Unfortunately, there is no funding for either a user study or security review.

In lieu of a user study, SitePassword is currently only available to those who know its URL in the Chrome Store. The hope is that early users will provide sufficient feedback. Informal discussions with a number of security experts has confirmed the safety of several aspects of the design.

The accuracy of the strength meters depends on the dictionary used by zxcvbn, which consists mostly of American English words. Giving you the ability to select a dictionary with words in your native tongue will give more accurate strength estimates.

There is an experimental Safari version that uses synchronized storage because that browser doesn't support the bookmarks API. As a result, your settings are only available if you download them to a file in the cloud. Automating the use of iCloud might address this shortcoming.

There are no plans to create apps for mobile devices because of existing security problems on those platforms [3, Chapter 4]. Should that problem be solved, there would still be the problem of making the metadata stored in browser bookmarks available to the app.

One way to protect your super password is to use a second factor [10]. The question is how to make sure you can get your passwords from any machine. This feature will be added if there is a suitable answer.

[6] https://app.shiftleft.io/apps

It may be possible to prevent offline guessing attacks by incorporating passkeys using something like the WebAuthN local client.[7] Doing that without forcing users to change all their passwords might prove challenging.

8. Conclusions

SitePassword is an open source, hybrid password manager that includes the features people want most – ease of use, trustworthiness, remembering of metadata, and availability of passwords across devices. In addition to calculating passwords when needed, it can also remember passwords you provide. Because it doesn't require user accounts or cloud storage, it is free.

Acknowledgements

The author would like to thank Michael Josefik for providing a professional look and feel to the user interface, Dale Schumacher for significant code improvements to the web version and generally useful discussions. Douglas Crockford provided significant feedback as an early adopter, and Jasvir Nagra provided guidance needed to make the content script work properly. Sjorerd Langkemper identified several algorithmic weaknesses in an earlier version. Dan Boneh's Stanford Security Lunch group provided security insights, particularly Rohit Nema, who did several sanity checks. Useful input came from the Friam security group.

References

1. Sarah Pearman and Shikun Aerin Zhang and Lujo Bauer and Nicolas Christin and Lorrie Faith Cranor, Why people (don't) use password managers effectively, Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019), pp 319–338, 2019
2. Yee, Ka-Ping, User Interaction Design for Secure Systems, Proceedings of the 4th International Conference on Information and Communications Security, pp. 278–290, http://zesty.ca/pubs/icics-2002-uidss.pdf, 2002
3. Oesch, Timothy, An Analysis of Modern Password Manager Security and Usage on Desktop and Mobile Devices, Ph.D. Thesis, University of Tennessee, https://trace.tennessee.edu/utk_graddiss/6670/ 2021
4. Wheeler, Daniel Lowe, zxcvbn: Low-Budget Password Strength Estimation, Proceedings of the 25th USENIX Conference on Security Symposium, pp. 157–173, 2016
5. Angel Marin and Paul Johnston, Secure Hash Algorithm (SHA256), http://www.webtoolkit.info/
6. Karp, Alan H., "Site-Specific Passwords", HPL-2002-39, https://shiftleft.com/mirrors/www.hpl.hp.com/techreports/2002/HPL-2002-39R1.pdf, 2002

[7] https://github.com/mylofi/webauthn-local-client

7. Yee, Ka-Ping and Sitaker, Kragen, Passpet: convenient password management and phishing protection, Second Symposium on Usable Privacy and Security (SOUPS 2006), pp 32-43, 2006

8. Blake Ross and Collin Jackson and Nick Miyake and Dan Boneh and John C Mitchell, Stronger Password Authentication Using Browser Extensions, 14th USENIX Security Symposium (USENIX Security 05), 2005

9. David Silver, Suman Jana, Dan Boneh, Eric Chen, and Collin Jackson. Password managers: attacks and defenses. In Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14). USENIX Association, USA, 449–464, 2014

10. Vivek Nair and Dawn Song, MFDPG: Multi-Factor Authenticated Password Management With Zero Stored Secrets, arxiv.org/abs/2306.14746, 2023

11. Moritz Horsch, Adreas Hulsing, Johannes Buchmann, PALPAS - PAsswordLess Synchronisation, https://arxiv.org/abs/1506.04549, 2016

12. F. Al Maqbali and C. J. Mitchell, "AutoPass: An automatic password generator," *2017 International Carnahan Conference on Security Technology (ICCST)*, Madrid, Spain, 2017, pp. 1-6, doi: 10.1109/CCST.2017.8167791

Appendix 1: Comparison of Password Calculators

There are a *lot* of password managers in the Chrome extension store. The majority store passwords, but there are also a lot that calculate them. I installed all of the password calculator extensions that claim to have at least one password manager feature and attempted to use each one to figure out what features it supports. Some of them I couldn't make work; others didn't work at all. Some have web pages for doing the calculation that no longer exist. A few have instructions not to use them!

The good news is that all of these password managers require a user action for autofill. The bad news is that user action is often in the popup, which risks putting the password in unintended fields, such as the answers to security questions or those injected by an adversary.

A surprising number are unusable, in contrast to those that store passwords. For example, some use the domain name as an input to the calculation, which generates different passwords for different domains, *eg*, client.schwab.com *vs*. www.schwab.com. Others have no option to generate passwords if the extension is not available or to put the password on the clipboard if autofill doesn't work. Most do not provide rich enough settings to guarantee generating a legal password.

The table summarizes properties of the all password calculators in the Chrome store as of March 2023 that I could make work and have at least one password manager feature. The column headings are:

Extension: The name of the extension to search in the Chrome store
Inputs: The values used to compute the password, settings include such things as the number of numbers, upper and lower case letters, and how many and which special characters to include.
M: Which values are remembered across invocations
S: If settings are synched across machines
F: Autofill (u = userid, p = password)
R: If there is a way to conform to the site's password rules (P = partial)
O: If you can get your passwords if the extension isn't available
P: If you are warned of potential phishing
A: If you need an account with the password manager's provider
D: If there is an app for mobile devices
E: If you can use your existing passwords
Notes: Things that don't fit the other categories.

Extension	Inputs	M	S	F	R	O	P	A	D	E	Notes
SitePassword	superpw userid nickname settings	All settings and user-provided passwords	Y	u p	Y	Y	Y	N	N	Y	Finds pw field on 120+ test sites
MindYour-Pass	pw per site part of domain, pw size, special chars	username	Y	u p	P	Y	N	Y	Y	Y	Autofill often falls
OnePass	nickname part of domain, pw size	superpw nickname	N	N	P	P	N	N	N	N	Recommends using any MD5 generator for if extension not available
Password Generator	nickname settings	all	Y	p	Y	Y	N	N	N	N	Autofill fails on iframe, uses sync storage
GenPass	superpw domain pw size	superpw nickname	Y	N	N	N	N	N	N	N	Different passwords for client.schwab and www.schwab
Password Maker	superpw nickname settings	superpw	N	?	P	N	N	N	N	N	Autofill doesn't work with iframe
Pure	superpw nickname grid	None	N	p	P	N	N	N	N	N	Only opens if it finds password field, fails on iframe
Entropass	superpw nickname hidden key pw size	nickname	Y	p	P	N	N	N	N	N	Can't tell how hidden key is synched
SynthPass	superpw part of domain, pw size	All	Y	u p	N	N	N	N	Y	N	Doesn't work if pw in iframe
PwdHash	password domain	None	N	na	N	Y	N	N	N	Y	Type into pw field with prefix
PawHash	superpw nickname settings	None	N	p	P	N	N	N	N	N	Doesn't work if pw in iframe
OfflinePass	superpw domain year, counter	superpw userid	N	N	N	Y	N	N	N	N	Different pw for client.schwab and www.schwab,
SimplePass-words	superpw nickname	superpw	N	p	N	N	N	N	N	N	Autofills on generating password
Just1Pass-word	superpw domain email	email	N	N	N	N	N	N	N	N	Different pw for client.schwab and www.schwab,

Appendix 2: Strange Password Rules

Many sites forbid using your name, user name, "common" words, keyboard sequences (qwerty, asdf) as part of your password, or a repeat of a previously used or similar password. The chance SitePassword will generate such a password is vanishingly small. The main requirement SitePassword doesn't consider is rules that disallow sequences and repeats. In these cases changing the site's nickname will generate an acceptable password with high probability.

There is a list of hundreds of sites that have particularly dubious password rules.[8] Given the ability to generate a new password by trying a different nickname, SitePassword should be able to generate a legal password in one or two tries for all but 5 sites.

- Easybank passwords must start with 5 numbers, which SitePassword is unlikely to generate.
- Intellink disallows passwords that have any three characters or three numbers in a row, *e.g.*, axqalkd3239x is not acceptable. SitePassword is unlikely to generate an acceptable password.
- ME Bank passwords consist of all numbers, but a given digit can appear no more than 5 times. There is only a modest probability that SitePassword will compute an acceptable 20-digit password.
- Onleihe: Your password is your birthday, and you can't change it. SitePassword can store that password but clearly isn't going to calculate it.
- Vancity Credit Union: Your password is all digits and must start with a 0. SitePassword will calculate a valid password for 10% of the nicknames you try.

It is likely that more sites with rules that are hard or impossible for SitePassword to satisfy will be identified. The ability to provide a password mitigates the risk to users.

Appendix 3: Forcing Online Tests

The password generation algorithm takes a super password s that is as guessable as a random set of S bits and a site specific, public[9] salt n_i to produce a password p_i for site i having P_i bits, where $p_i=F(h(s,n_i))$. Here h is an H-bit hash function, and F is a function that converts the bits output by the hash function into a password acceptable to the site.

Assume the adversary learns n_i and p_i for site i, perhaps from a breach. The adversary can then start guessing values s_j to see if $F(h(s_j,n_i))=p_i$. If there is only one s_j that produces p_i, the adversary can then generate passwords for all your sites. If more than

[8] https://dumbpasswordrules.com/sites/

[9] Not all the inputs may be public, *e.g.,* a user selected nickname for the site, but the salt is probably easy to guess.

one guess produces p_i, the adversary must test the guess online at least one other site you use. This candidate is either *s* or a collision. The guessing attack becomes impractical if the probability the guess is *s* is sufficiently small even if it doesn't take long to find each collision.

Collisions are generated when $S>P$, where *S* and *P* are the entropies in bits of *s* and p_i, respectively. The number of guesses for *s* that generate p_i is 2^{S-P} for $S-P>0$, and *1* otherwise. Unfortunately, making $S-P$ large is hard because *S* is limited by your ability to remember *s*, and *P* needs to be large enough to make p_i hard to guess. Nevertheless, there is a reasonable range of parameters for which a guessing attack is impractical.

An *L*-character p_i computed from a good hash function typically has entropy $|A|^L$, where $|A|$ is the number of characters in the alphabet Since *s* must be memorable, its entropy is usually less than that of a random string of the same length. If a user generated string has the entropy of a *K*-character random string, then the number of collisions is $2^{a(K-L)}$, where $a = \ln |A|$.

Consider some examples with a 64-character alphabet. With a good super password having entropy $K = 16$ characters (S = 96 bits) and a strong site password with $L = 12$ (P = 72 bits), there will be 2^{24} collisions, which is more than enough collisions for adequate guessing protection. Weaker super passwords don't get this much protection unless the site password is also weaker. A super password with K = *14* would be adequately safe from an offline guessing attack with a 10-character site password, which itself is moderately secure against guessing when stored using a slow hash function but not otherwise. A weaker super password would only be safe from a guessing attack when the site password is very weak.

SitePassword encourages you to pick a stronger super password than you might otherwise do by using a different strength metric for the super and site passwords. The weakest super password rated Strong has 16-bits more entropy than the weakest site password marked Strong. Users will probably only choose a super password rated Strong while not increasing the length of site passwords already rated Strong.

There is no need for the adversary to try all possible collisions online. Most of the collisions correspond to guesses that are effectively random strings. However, the adversary knows that human memorable strings typically contain one or more dictionary words or variants, such as capitalization, reversals, and L33T-speak substitutions. Measurements using zxcvbn show that about 7% (13%) of 12(24)-character random strings contain at least one substring that a person might choose. Therefore, we need to choose parameters that generate about 10 times as many collisions as it first appears.

So far the salt has been assumed to be public, but that's not entirely the case. In SitePassword, it consists of two terms, your userid for the site and an easy-to-remember nickname for the site with the tab character separating them. Changing the case of the letters in these fields would make them less guessable, but usability considerations re-

quire that the site nickname and userid be converted to lower case before the computation, reducing the entropy of the salt.

The nickname is supposed to be easy to remember, but appending the same, easy to remember string, such as your birth year, to every nickname increases the number of collisions enough to counter the adversary's ability to filter guesses. In other words, K-*L = 2* gives strong enough protection from offline guessing unless you are being individually targeted.

There is another source of collisions. The computed site password may not meet the rules for the site. For example, it may start with a number, which the site won't accept. If a guess generates a password that is not acceptable to the site, the adversary will have to got through the steps described in Appendix 4. If that result matches the known site password, the guess may be a collision. In other words, an initial hash result that looks nothing like the known site password was generated by a guess that is actually a collision. This protection is limited because more than 90% of candidate passwords computed with the default settings satisfy the rules for most sites.

An adversary who controls a site can reduce the number of required online tests by setting the password rules to require a very long site password, resulting in a small or even negative value for *S-P*. The risk to the adversary is that users might become suspicious if a site with no obvious security requirements asks for a very long password.

Note that an adversary can filter almost all guesses by learning the inputs and site password for a second site. The very fact that makes guessing attacks infeasible if the adversary has information for one site allows an adversary who has that information for more than one to almost completely avoid online tests. As a result, users with weaker super passwords should change them as soon as they find out that their login information at a site was compromised. Unfortunately, they will then need to change passwords everywhere they log in, which is what users must do if the a password manager that stores their passwords is compromised.[10]

Appendix 4: Computing an Acceptable Password

Sites have a variety of rules for what constitutes a password they will accept. There are often restrictions on length and on what characters are accepted, such as blanks, and requirements for various subsets of characters, such as no special characters. SitePassword gives you a form for describing these rules, but the algorithm that calculates your site password simply makes a random selection from the set of allowed characters. That choice might not satisfy a rule, such as "at least one digit" or "starts with a letter." Even so, SitePassword will find an acceptable password except in the most extreme cases, such as those sites listed in Appendix 2.

[10] They'll have to do the same thing if their encrypted password vault was stolen.

The process of finding an acceptable password starts with running 200,000 iterations of PBKDF2 to find a candidate password. If that password is not acceptable, SitePassword creates a new candidate by iterating PBKDF2 one more time. It repeats that step up to 200 times, a limit chosen to meet the latency requirement. In testing, this step always succeeds except in extreme configurations, such as requiring 6 special characters in a 12-character site password.

When the iterations do not generate an acceptable password, SitePassword could ask you to try again with a different site nickname. Instead, SitePassword reverts to a deterministic algorithm.

It starts by Inserting characters of the the required types at the beginning of the output and filing in the rest using bytes of the hash to select characters from the full alphabet. Then it shuffles the position of the characters based on bytes from the hash. This algorithm is guaranteed to find an acceptable password if one exists, *e.g.*, the total number of required characters doesn't exceed the password length.

There are two variants.

1. Use the same bytes for both selecting characters and for shuffling.
2. Use additional bytes from the hash for shuffling.

It would be simpler just to use the deterministic algorithm when the initial hash doesn't produce an acceptable password. Unfortunately, an adversary would need fewer tries to guess your site password. If your site password was generated with the second variant, an adversary who knows the site password needs to try fewer guesses of your super password online.

Consider how approach 1 works for the case where the password rules require one each of upper case, lower case, digit, and special character. These will appear in the first four positions before the shuffle. The adversary knows that before the shuffle the first character was selected from a set of 52 characters; the second, a set of 26; the third a set of 10; and the fourth from a set specified in the password rules, say 8. The result is that the adversary needs about 1% of the number guesses to find your site password.

Approach 2 uses additional bytes from the hash to do the shuffle, either one byte used as the seed for a pseudo-random number generator or one additional byte per character in the site password. In either case, the adversary needs fewer online tests because more bytes go into computing the site password.

To see this assume that a guess has all the characters that appear in a known site password but not necessarily in the right order. The adversary can un-shuffle the characters and reject the guess if the result doesn't match the known site password. As a result, the exponent of the number of collisions that must be tested online is reduced

from *S-P* to *S-P-6N*,[11] where *N=L* if a byte per character is used for the shuffle, or *N=1* if one byte is used as a random number seed.

Since the deterministic algorithm can make both site and super passwords easier to guess, that algorithm is used only when the iterative algorithm fails. As a practical matter, no site in some 300 used for testing has password rules that requires the deterministic algorithm.

[11] For a 64-character alphabet.