A Usability Study and Critique of Two Password Managers *

Sonia Chiasson and P.C. van Oorschot
School of Computer Science, Carleton University, Ottawa, Canada
chiasson@scs.carleton.ca

Robert Biddle
Human Oriented Technology Lab, Carleton University, Ottawa, Canada

Abstract

We present a usability study of two recent password manager proposals: PwdHash (Ross et al., 2005) and Password Multiplier (Halderman et al., 2005). Both papers considered usability issues in greater than typical detail, the former briefly reporting on a small usability study; both also provided implementations for download. Our study involving 26 users found that both proposals suffer from major usability problems. Some of these are not "simply" usability issues, but rather lead directly to security exposures. Not surprisingly, we found the most significant problems arose from users having inaccurate or incomplete mental models of the software. Our study revealed many interesting misunderstandings – for example, users reporting a task as easy even when unsuccessful at completing that task; and believing their passwords were being strengthened when in fact they had failed to engage the appropriate protection mechanism. Our findings also suggested that ordinary users would be reluctant to opt-in to using these managers: users were uncomfortable with "relinquishing control" of their passwords to a manager, did not feel that they needed the password managers, or that the managers provided greater security.

1 Introduction

Several recent password managers, intended for protecting web accounts, generate strong passwords (i.e., more resistant to dictionary and/or social engineering attacks) from weaker but more easily remembered user-chosen passwords. They can also facilitate safer re-use of passwords across accounts by using various forms of salts such as those derived from target site domain names. The expectation is that these password managers increase security. Does this expectation materialize when real users are involved? Can ordinary users actually use these systems? Do they want to? These are some of the questions which we address in this paper, as we carry out a usability study of two password managers – PwdHash [24] and Password Multiplier [11].

Despite the inadequacies of passwords from a security viewpoint, they are becoming more common. This is causing additional challenges for ordinary users who accumulate password-protected accounts for a growing number of services and web sites. This also increases security risks for several reasons. One is that passwords are commonly reused across accounts; thus a password used for a low-security site, easily compromised by an attacker, may allow access to a higher-security site. Phishing attacks on passwords have increased dramatically over the past three years, complicating matters further. On top of these issues, it is well known that security is viewed as an obstacle by many users because security is rarely a user's primary goal. When security procedures impede users' end goals, users bypass security [2, 5, 26]. Thus usability is an increasingly important aspect of security in general, and password systems in particular.

Our objective was to examine if any progress had been made in addressing usability in security, seven years after "Why Johnny Can't Encrypt" [31]. Since novice users are often at increased risk from security vulnerabilities, we sought tools aimed at novice users as a logical starting point. Two password manager projects in particular piqued our interest: PwdHash and Password Multiplier. Both were published in 2005, had a significant focus on usability and positioned it as a major objective, and had publicly available implementations. Although the implementations that we tested are still in beta versions, their authors state that these should help novice computer users protect their passwords. We decided to carry out an independent study to test the effectiveness of the two systems from a usability standpoint – could they be successfully used, and did any security problems arise due to usability problems?

We found discrepancies between the usability claims

*This paper appears in the *Proceedings of the 15th USENIX Security Symposium*, Vancouver, Canada, August 2006.

of the published password manager papers and the results of our study. We uncovered numerous usability problems with the interfaces, some of which create security exposures. Many of the usability issues found are not particular to security, and are addressed in the existing Human-Computer Interaction (HCI) and usability literature. This suggests that the existing gap between the HCI and security communities hurts the latter due to a lack of awareness (or at least application) of the HCI usability literature, and leads only to rediscovery of well-known principles. Research effort would of course be better spent focusing on those usability issues that are unique to security interfaces.

The authors of both password managers state that security interfaces should be as transparent as possible; resulting in absolutely minimal or no change in user experience. Our study reveals that designing towards this belief can not only contribute to usability problems, but can lead to "dangerous errors" [31];[1] we conclude that this idea requires closer scrutiny and validation.

OUR CONTRIBUTIONS. We add to the relatively sparse, albeit growing, set of published usability studies in the security literature. We carry out an independent usability test of two proposed password mechanisms which have received significant attention within the security community: PwdHash [24] and Password Multiplier [11]. The results of our larger study directly contradict the findings reported for PwdHash (no user study was published for Password Multiplier), and suggest that both earlier papers (1) over claim the actual usability of their mechanisms as provided in their own publicly available implementations; and (2) result in "dangerous errors" with serious security implications. Our work reiterates the necessity of actual user studies before concluding that new security mechanisms are usable. It also raises the question of what the standard should be in the security community: should usability tests become a requirement for new authentication proposals, if a major claim is usability? Aside from usability and security considerations per se, we also provide interesting results on how users felt about these mechanisms: about giving up "control" of their passwords to these mechanisms, their perceived security, and their perceived necessity.

ORGANIZATION. Section 2 reviews the two password managers we evaluate. Section 3 first briefly provides context on usability testing, and then details our study methodology. Section 4 presents the data collected from interaction with participants, our interpretation and analysis of which is provided in Section 5. Section 6 provides further discussion and recommendations. Section 7 discusses related work. Section 8 gives concluding remarks.

2 Password Managers

Passwords are the most common form of authentication on computers today [23]; yet they are far from the most secure. Cognitive demands increase as people use more computer systems, leading to poorer password choices in an attempt to manage the load. Encouraging stronger passwords through password rules or advice on selecting good passwords does not help users remember the stronger passwords. Encouraging the use of passphrases similarly does not mitigate the problem of matching multiple passwords and multiple accounts.

One proposed solution to these password problems is *password managers*. One class of these managers maps low-entropy (easy to remember) user passwords into stronger passwords (more resistant to dictionary attacks), and may also generate site-specific passwords partially dependent on the domain name of the site (protecting against some phishing attacks). Several password managers exist in different formats: stand-alone applications (e.g., Site Password [13]), browser plug-ins (e.g., Password Maker [15]), browser scripts (e.g., Password Composer [14]), and bookmarklets (e.g., Password Generator [33]). PwdHash and Password Multiplier are two recent additions to the list. They are both browser plug-ins claiming to make it easier for users to have secure passwords without having to remember a number of complicated passwords.

2.1 PwdHash

PwdHash [24] is a browser plug-in that applies a cryptographic hash at the client machine to generate strong passwords based on the user's entered password and the site domain. The new stronger password is sent to the target site. No server-side changes are needed. Users activate the installed plug-in by adding the @@ prefix to passwords they want "protected", or by pressing the F2 key before entering their password. Once passwords are "protected", the users no longer know their effective passwords; thus PwdHash must always be used to log in to the web accounts. The authors [24] provide a web site where users can remotely generate their protected password for times when they are logging on to a site from a computer without the plug-in (see Figure 1). This generated password is copied and pasted by the users into the target password field.

Users must initially log on to each web account they want to protect, and change their password. They enter @@ in front of their new password, which activates PwdHash and generates a new "strong" password. Users later changing their protected password for a given site use that web site's Change Password interface, but with @@ as a prefix to both their current and new passwords.

PwdHash is also designed to prevent certain JavaScript attacks. It processes the input stream, scanning for the

Figure 1: Remote web site for generating PwdHash passwords

@@ character, processing so-designated passwords, and replacing the input, before JavaScript would have access. PwdHash also protects against phishing attacks by using a hash salt based on the domain name of the target web site. If users enter their password at a fraudulent site, that site's domain will be used as the salt.

An implementation is publicly available. The reported user study of five users found that participants experienced little difficulty using the plug-in and that the only usability problems were observed with the remote interface. The approach was positioned as unique in that it implements password hashing within the browser, without additional window pop-ups or having a visible software interface. The PwdHash authors believe that this improves its usability: *"We can reduce the threat of password attacks with no server changes and little or no change to the user experience. Since the users who fall victim to many common attacks are technically unsophisticated, our techniques are designed to transparently provide novice users with the benefits of password practices that are otherwise only feasible for security experts"* [24].

2.2 Password Multiplier

Password Multiplier (hereafter identified as P-Multiplier) is a second password manager intended to help users generate strong passwords for web accounts [11]. It is a plug-in for Mozilla's Firefox web browser, requiring no changes to the servers; all computation is done client-side. As with PwdHash, a cryptographic hash function is applied, generating strong passwords for the user's accounts. While a primary design goal of PwdHash is to protect web site passwords against phishing attacks, P-Multiplier is intended to generate and manage strong passwords from a single user-chosen password for arbitrary password-protected applications (i.e, not restricted to web site passwords); this difference does not affect our usability study, and the plug-in we tested is restricted to web site passwords. P-Multiplier uses a master username and master password so that users need only remember one password for all of their accounts. A protected password is generated based on the master username,

Figure 2: The dialog box for P-Multiplier is activated by double-clicking on the password field

master password, and the target site domain name. Users activate the plug-in by double-clicking on the password field or pressing Alt+P while the cursor is in that field. This opens a small dialog box. The master username is entered automatically (it is set upon installation). Users enter the master password (see Figure 2). When users click the OK button, the dialog box closes and the generated password is automatically placed into the web site's password field. The background colour of the web site's password field changes to signal the entry of the protected password.

Users must switch each account password over to a protected password generated by P-Multiplier, by using the Change Password interface of each web site. They double-click on the new password field to activate P-Multiplier and generate their new password. To update any password after it is protected, users must modify the "site name" argument used to generate the password (e.g., for a google.com password, change the site name in the P-Multiplier dialog box to google.com-two). Users thereafter must remember to enter their changed site name each time they log in to this particular site.

To help protect against dictionary attacks, P-Multiplier uses a two-stage process which in the first stage, massively iterates the hash function, with the intent to slow down any attack on the master password. This introduces a 100-second delay when first installing the plug-in, while it computes a first hash result which is cached on the local computer. Passwords later generated on this computer use the cached first hash, as well as the master secret again, as input to a second (less massively iterated) hash operation. Legitimate users experience the long delay only on new installations of the plug-in (not per-login). The idea is that attackers must compute the hash function from scratch each time they test a new master password, encountering the long delay with each trial.

For use from remote (secondary) computers, users must download and install the plug-in on the remote

computer, and on it enter the correct master password and username in order to generate their protected passwords (and experience the 100s first-stage delay). No alternative is provided for users unable to install software (this is positioned as a relatively rare occurrence by the P-Multiplier authors).

No usability testing was reported in publication [11] or on the P-Multiplier web site [12]. When discussing usability, they acknowledge that transportability is a necessary characteristic. They assert that PwdHash and P-Multiplier are equally transportable, however their software requires installation on a remote site (unlike PwdHash).

3 Study Methodology

To investigate the usability of these password managers, we conducted a study with participants who would be typical users of these systems. We first provide some background on usability studies, then describe our study methodology in detail.

3.1 Usability Testing

Since it is not a mainstream topic for most security researchers, we begin by briefly providing some background on usability studies. There are two general categories of methods for assessing the usability of a system: usability inspection methods and user studies. With usability inspection methods (such as cognitive walkthroughs and heuristic evaluations), evaluators inspect and evaluate usability-related aspects of a system. They are conducted without end users and require a certain level of expertise in usability [19]. These are useful in finding obvious usability problems but are no substitute for user studies with real users. Typically, usability inspection methods are used early on to guide the design process, then user studies are conducted to confirm the design decisions and find any problems that have been overlooked. User studies can range from closely controlled experimental studies testing specific hypotheses to field studies where the system is deployed for real usage and system logs and interviews are used to assess its usability. Most user studies fall somewhere in between, conducted in a lab, with pre-determined tasks, but also leaving room to observe users in a more ad hoc manner and uncover unexpected problems as they arise [27].

Usability tests are used to determine whether a system is suitable for the intended audience and for its intended purpose. Typically, the tests aim to uncover any difficulties encountered by the users as they go through a set of predetermined tasks. These tasks should be carefully chosen to reflect realistic usage scenarios. To preserve ecological validity, the environment should be set up to mimic reality as closely as possible in terms of technical details, but also in terms of instructions given. If a typical end-user is expected to be able to use the system without in-person training, then training on system use should not be provided during the test.

It is important to closely observe users as they perform these tasks as this is how most usability problems are revealed. The observer's role is mainly to observe and record what is happening. They need to be careful not to provide extra instructions or cues that may influence the user's actions. In fact, a script should be used to ensure that all participants receive the same information. It is important to emphasize that it is the system that is being tested and not the user; the participants should feel that they are helping with the development of the system rather than feel like their performance is being evaluated. A method called "think-aloud" is typically used, where users are asked to keep a running commentary as they perform the tasks. Pre/post questionnaires or interviews are also useful in gathering users' opinions, attitudes, and feedback about the system. This should be a secondary source of information, used in conjunction with observations and potentially system logs, because users' reported views often do not reflect their performance and often fail to reveal crucial usability problems.

Selecting the right participants for a usability study is important. The participants should accurately represent the users who would use the actual system, and be similar in terms of experience and knowledge. Improperly choosing participants will negatively affect the results of the tests, typically by missing critical usability problems.

The guideline stating that five users are enough to discover most usability problems [18, 29] has long been used to justify small usability studies. Recent work questions this assumption and highlights the fact that in most cases five users are not enough [9, 22, 28]. They found that some severe usability problems were only discovered after running a larger group of participants. The likelihood of finding usability problems is not evenly distributed. Some problems only arise under specific circumstances so using a small sample of users may not be sufficient to uncover them. The variability in the number of problems found by any one user also makes it unlikely that a sample of five users would discover most usability problems. Faulkner justifies that twenty users "can allow the practitioner to approach increasing levels of certainty that high percentages of existing usability problems have been found in the testing" [9].

3.2 Overview of Study

Our tests were conducted in Carleton University's Human-Oriented Technology Lab and the methodology was reviewed and approved by the university's ethics committee. Our study explicitly looks at the password managers as implemented rather than at the proposed additional implementations suggested by the systems' au-

thors.

The typical tasks that users would need to accomplish with password managers fall into four categories:

1. Migrate user accounts (passwords) to use the password manager
2. Log in to protected user accounts from a primary computer
3. Change passwords for user accounts
4. Access user accounts remotely, i.e., from a computer other than the primary computer, such as on a public or friend's machine.

Each participant completed a one-hour session, where they completed a set of five tasks designed to simulate the real tasks that users would accomplish with the password managers. The set of tasks was repeated so that each participant completed them with both PwdHash and P-Multiplier. The order in which the tasks and the programs were presented was balanced to avoid bias. Throughout the session, the experimenter observed the participant and recorded their actions. Additional user feedback was gathered through questionnaires.

3.3 Participants

Twenty-seven adults participated in the study. Most were students at our university, from various faculties and degree programs; none were students specializing in computer security. A few had technical backgrounds: four were from Computer Science, one studied Information Systems, and none were from Engineering. Data from one participant was eliminated as a language barrier coupled with very little computer experience hindered their ability to understand the tasks. Of the remaining 26 participants, 21 were between the ages of 18 and 30 and five were over 30 years old. Data from these 26 participants was used for all further analysis in this paper.[2]

The participants were familiar with using the web and logging on to web sites requiring a username and password. All but two reported visiting the web daily, and these two said they were online several times a week. The participants were fairly comfortable with using computers; 24 of the participants self-rated their general computer skill level at 6 or higher on a scale of 1 to 10.

We chose not to screen participants based on experience using Firefox. Typical Firefox users are more technically sophisticated than average users so pre-selecting on this criteria would have biased our pool of participants. Additionally, interaction with the browser's interface was minimal; participants simply had to enter URLs and navigate within web pages. These tasks are accomplished in the same manner in Firefox and Internet Explorer.

A pre-task questionnaire was used to gain insight into the participants' initial attitude towards web security and passwords. They reported using an average of six web sites requiring a password to log in. A summary of their responses is presented in Table 1.

Table 1: Participants' initial attitude towards web security and passwords. Results represent the number of participants (out of 26) responding yes to each question. *An additional 27% (7) responded "somewhat".

Question	**Number of Users**	
Do you sometimes reuse passwords on different sites?	96%	(25)
Are you concerned about the security of passwords?	*58%	(15)
Criteria for choosing passwords:		
Easy to remember	69%	(18)
Difficult for others to guess	54%	(14)
Suggested by the system	0%	(0)
Same as another password	62%	(16)
Other	12%	(3)
Participation in online activities requiring personal or financial details:		
Online purchases	62%	(16)
Online banking	73%	(19)
Online bill payments	73%	(19)
Other activities	27%	(7)

3.4 Tasks

Participants completed a set of tasks using two different computers during the session. Both computers were running Windows XP and Mozilla Firefox. One system had the PwdHash plug-in (version 1.0 for Mozilla Firefox) installed while the second computer included the P-Multiplier plug-in (version 0.3 for Windows, Linux, and Mac OS).

The tasks are described in the following list. The *Second Login* task is dependent on the *Update Pwd* task, i.e., users must have successfully changed their password before they are able to log on to the site a second time with their new protected password. All other tasks are independent of each other. We did not include a "delete password" task because neither system supports this functionality. The tasks are:

Log In: Logging on to a web site that already has its password protected by the plug-in. This simulates how users log on once their passwords have been converted to protected passwords.

Migrate Pwd: Logging on to a web site with an unprotected password then changing the password so that it becomes protected. This is required by users to initially migrate each of their passwords.

Remote Login: Logging on to a web site with a protected password from a remote computer that does not have the plug-in installed. This models how users would log on to their accounts from a computer other than their primary machine.

Update Pwd: Logging on to a web site with a protected

password then changing it to a new protected password. This situation would arise if users had to change their password once it is already protected.

Second Login: Logging on to a web site a second time, once the user has changed the password to a protected password. This task tests whether users understand how to log on to their account once they have changed to a protected password.

The tasks were set up using popular web sites (Hotmail, Google, Amazon, and Blogger) that users may encounter in real life. Test accounts were created so that participants did not use their personal accounts or passwords at any point during the experiment.

Participants completed the set of tasks with both plug-ins; the order was balanced so that each plug-in was seen first the same number of times. The order of the tasks within a set was also shuffled but an individual participant saw the tasks in the same order for both plug-ins. The *Update Pwd* and *Second Login* tasks were ordered so that they were always separated by exactly one task (for example, a participant completed the tasks in the order of *Log In*, *Remote Login*, *Update Pwd*, *Migrate Pwd*, *Second Login*). This ensured that participants changed their focus for a time before logging on to the web site a second time with their new protected password. One participant quit after completing the tasks only with PwdHash, but the remaining participants completed all tasks.

One of the difficulties with testing the usability of these plug-ins is that they initially have no visible interface. Even during the interaction, only P-Multiplier has a visible pop-up window. So simply giving the tasks to participants without instructions on how to use the plug-ins would have been futile. To preserve ecological validity, we tried to keep the instructions to a minimum; giving them written details of how to activate the plug-in, a brief explanation of how to change a password, and a short description of how to log on to a web site using a remote computer. The entire set of instructions was approximately half a page long for each plug-in (see Table 2). Users typically do not read manuals when they use software [3, 17, 31] so having participants follow detailed instructions would not have reflected a realistic scenario. Participants were also given a list of the usernames and passwords that they would require to complete the tasks. To minimize the effect of learning new passwords, a simple, one-word password was given for all tasks within a system ("alphabet" and "carleton"). These passwords were also written on a sheet in front of participants throughout the session.

Participants were given the instruction sheet for the particular plug-in and told that they could refer to it whenever necessary. They were directed to a computer with a Firefox browser window open and the appropriate plug-in pre-installed. They were instructed to pretend that this was their home computer and they should use Firefox as the browser for these tasks. Participants completed all tasks with a plug-in before switching computers to repeat the tasks with the second plug-in. No participant expressed any concern over using Firefox instead of the more popular Internet Explorer and no difficulties were observed due to using this alternative browser. Firefox was selected as the browser because the stable versions of the plug-ins were not available for Internet Explorer. Firefox was used in the original PwdHash usability study as well.

Table 2: Example instructions given to participants on how to use PwdHash

PwdHash Instructions:
Add @@ in front of passwords you want to be made secure, this will activate PwdHash. PwdHash will transform the password before sending it to the web site. For example, if your password is "bob", enter "@@bob".
You can also activate PwdHash by clicking on the password field and pressing the F2 key before entering your password.
To reset a password:
If your old password was not protected, enter the old password without activating PwdHash. When entering the new password, include the @@ at the front of the new password. This will activate PwdHash and transform this particular password.
If your old password was already protected by PwdHash, activate PwdHash for your old password. When entering your new password, activate PwdHash and enter a new password for the site.
To use remotely:
To log in to a web account from a computer that does not have PwdHash installed, visit:
http://crypto.stanford.edu/PwdHash/RemotePwdHash
to generate your protected password. Enter the address of the target site and your password. The protected password will be generated. It can be copied/pasted into the password field of the target site.

Each task was described on an index card (see Table 3 for an example). The card also included two questions asking participants to rate the difficulty of the task and their satisfaction with the software for this particular task. Participants could take as long as they needed to complete the task and were told that if they felt they had spent enough time on a task and could not complete it, they could quit. At the end of each task, they circled their responses to the two questions and were provided with the next index card.

When participants reached the task where they had to log on to a web account from a remote computer (*Remote Login* task), they were instructed to change computers and pretend that they were now at their friend's house where the software was not installed. This proved problematic for P-Multiplier since the authors' solution to remote access is to install the plug-in. Participants

Table 3: Example index card given to participants for the *Log In* task

Log on to www.google.com. Your password is protected by Password Multiplier.
This task was:

very easy	easy	neutral	difficult	very difficult

For this task, how satisfied are you with the software used to manage the password?

very dissatisfied	dissatisfied	neutral	satisfied	very satisfied

could not install the plug-in on the second computer because it had PwdHash installed and the combination of the two crashed the computer. The *Remote Login* task was therefore eliminated for P-Multiplier. Judging from participants' reactions as they read from the instruction sheet that they had to install software for remote access, they would not have been pleased with this solution even if they had been able to complete the task.[3]

After completing a set of tasks, participants answered a paper questionnaire about their experience with the particular plug-in. The entire process was repeated for the second plug-in. A final post-task questionnaire asked participants to compare the two plug-ins.

3.5 Data Collection

Data was collected in two ways: through observation and through questionnaires. An experimenter sat with each participant throughout the session, recording observations, noting any difficulties, any obvious misconceptions in the participant's mental model of the software, any comments made by the participant, and whether they successfully completed the task. Participants were asked at the beginning of the session to "think-aloud". Besides the standard instructions given to all participants, no further explanations were given even if a participant asked for more instructions. In these cases, the experimenter remained cordial, clarifying that we were testing the usability of the systems and needed to see if people could use them without explanations. Occasional prompts such as "what did you expect to happen there?" were used if participants forgot to think-aloud.

The users' goal was to successfully complete the tasks using the given password manager. They were given as much time as they wanted and the observer waited for the participants to signal that they had completed the task or that they had run out of ideas and could not complete it. The outcome of each task was recorded by the observer according to the following possibilities:

Successful: The participant completed the task without difficulty.

Dangerous success: The participant eventually completed the task after several attempts (i.e., had difficulty). The negative impact is that in some cases, the unsuccessful attempts prior to the eventual success expose the password to attack (see Section 5.4).

Failed: The participant gave up on the task without completing it.

False completion: The participant failed to complete the task but erroneously believed that they had in fact been successful.

Failed due to previous: The participant could not complete the task because they had incorrectly completed the preceding task. This only applies to the *Second Login* task, where the *Update Pwd* had to be successful in order to proceed.

The first outcome is considered most positive. The second is somewhat positive but users may have exposed their passwords to danger (e.g., to JavaScript attacks and phishing) as they floundered with the task. They may even have inadvertently exposed multiple passwords, since a typical reaction to being unable to log in is to try all of one's passwords to see if something will work. The fourth outcome is especially dangerous because it leads to a false sense of security on the part of users.

Secondary measures taken in the study consisted of several Likert-scale questions [16]. These ask respondents to choose their level of agreement with the given statement from a set of possible answers, usually ranging from strong agreement to strong disagreement. We used a 5-point scale (strongly disagree, disagree, neutral, agree, strongly agree). Participants answered two of these questions on the index cards after each individual task, then completed a 16-question questionnaire for each plug-in.

The questions from the questionnaire were a priori grouped into four sets that considered different aspects of the interaction: perceived security, comfort level with giving control of passwords to a program, perceived ease of use, and perceived necessity and acceptance. Each set contained four similar questions (see Table 4); the questions were randomly organized on the questionnaire so participants were not aware of the groupings. Participants circled their answer for each question among the five choices. Half of the questions were inverted to avoid bias.

4 Collected Results

Neither PwdHash nor P-Multiplier fared well in terms of usability. Both the quantitative and observational data point to major problems, as explained below.

Table 4: Sample questions for each question set (for PwdHash, the questionnaire for P-Multiplier was identical other than the name of the software).

Perceived Security
My passwords are secure when using PwdHash.
I do not trust PwdHash to protect my passwords from cyber criminals.
Comfort Level with Giving Control of Passwords to a Program
I am uncomfortable with not knowing my actual passwords for a web site.
Passwords are safer when users do not know their actual passwords.
Perceived Ease of Use
PwdHash is difficult to use.
I could easily log on to web sites and manage my passwords with PwdHash.
Perceived Necessity and Acceptance
I need to use PwdHash on my computer to protect my passwords.
My passwords are safe even without PwdHash.

Our first measure of usability was whether participants were able to successfully complete the given tasks with each password manager. Our goal was not to provide a measure of how much better one password manager is compared to the other but to investigate the usability of each. Looking at Tables 5[4] and 6, it appears that PwdHash outperformed P-Multiplier but still had a relatively high chance of potential security exposures, as many of PwdHash's successful outcomes were only realized after multiple attempts. These latter successful outcomes – labelled "dangerous successes" – can only be cautiously viewed as successes (see Section 5.4). The web sites used for the tasks were specifically chosen because they have a very high tolerance for incorrect login attempts. Participants frequently attempted to log in three to ten (or more) times before they were successful or gave up. With sites that limit the number of attempts, most users would have been locked out.

For the *Migrate Pwd*, *Update Pwd*, and *Second Login* tasks, a number of participants felt that they had successfully completed the task when in reality they had not. This was more common with P-Multiplier. This was mainly due to participants incorrectly believing they had successfully migrated their password from unprotected to protected and subsequently believing that they were logging on with a protected password when they were still using an unprotected password.

The Likert-scale responses from the questionnaires were converted to numeric values (1 = most negative, 3 = neutral, 5 = most positive). The responses were grouped according to their predefined sets to find the mean responses for each set. Means were calculated and differences between P-Multiplier and PwdHash were assessed by running t-tests. In a strict statistical sense, Likert-scale data should not be converted to numerical data. Since it is ordinal data, the differences between "strongly agree", "agree", and "neutral" are not necessarily the same. However, in practice this type of statistical analysis is the most common and accepted way of reporting Likert-scale data as the difference in results between parametric and non-parametric analysis are usually minimal.

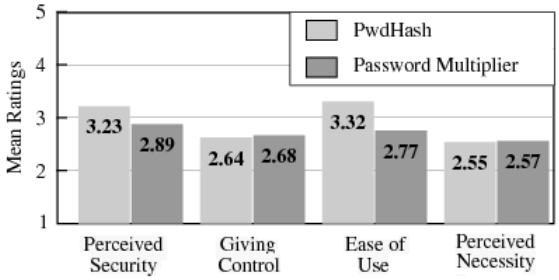

Figure 3: Mean questionnaire responses for each question group on scale of 1 to 5 (1 most -ve, 3 neutral, 5 most +ve)

We used t-tests to analyze the response distributions and determine the statistical significance of any differences. The t-tests can only be used to compare PwdHash and P-Multiplier against each other since we do not have an optimal system against which to compare the two. Examining the questionnaire data, the means for each group of questions reveal that neither systems fared very well; most values remained below neutral on the scale (see Figure 3). However, the t-test[5] showed that PwdHash was reported to be easier to use ($t(24) = 2.24$, $p < .05$) and perceived as more secure ($t(24) = 2.70$, $p < .05$) than P-Multiplier. The t-tests further revealed that the systems were similarly bad at making users feel comfortable with giving control to a password manager ($t(24) = -0.362$, $p = .721$) and that there was no difference between the two programs in how users felt regarding the perceived necessity of such systems ($t(24) = -0.207$, $p = .838$).

Examining their responses to the two questions from the index cards, we find that tasks completed with PwdHash were perceived as easier than those completed with P-Multiplier. Although participants reported higher satisfaction with PwdHash than P-Multiplier, in most cases the mean perceived difficulty and perceived satisfaction was below 4 for each. This means that participants initially reported positive reactions to the plug-ins. However these reported opinions need to be taken in context with user performance. In some cases, participants reported that the task was easy and that they were satisfied with the software even when they were unsuccessful at completing the task. In some of these instances participants were unaware that they had failed to complete the task. For example, they believed that they had generated

Table 5: Task Completion Results for PwdHash

	Success		Potentially causing security exposures							
			Dangerous Success		Failures					
					Failure		False Completion		Failed due to Previous	
Log In	48%	(12)	44%	(11)	8%	(2)	0%	(0)	N/A	N/A
Migrate Pwd	42%	(11)	35%	(9)	11%	(3)	11%	(3)	N/A	N/A
Remote Login	27%	(7)	42%	(11)	31%	(8)	0%	(0)	N/A	N/A
Update Pwd	19%	(5)	65%	(17)	8%	(2)	8%	(2)	N/A	N/A
Second Login	52%	(13)	28%	(7)	4%	(1)	0%	(0)	16%	(4)

Table 6: Task Completion Results for P-Multiplier

	Success		Potentially causing security exposures							
			Dangerous Success		Failures					
					Failure		False Completion		Failed due to Previous	
Log In	48%	(12)	44%	(11)	8%	(2)	0%	(0)	N/A	N/A
Migrate Pwd	16%	(4)	32%	(9)	28%	(7)	20%	(5)	N/A	N/A
Remote Login	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Update Pwd	16%	(4)	4%	(1)	44%	(11)	28%	(9)	N/A	N/A
Second Login	16%	(4)	4%	(1)	16%	(4)	0%	(0)	64%	(16)

a new secure password for a site when they had not even activated the plug-in – a potentially dangerous situation (see Section 5.4). In other cases, they said "well, this *should* have been easy, so I gave it a high rating". Obviously, relying solely on reported satisfaction and difficulty is misleading.

At the end of the session, participants were asked which of the two systems they preferred. Participants were nearly evenly distributed in terms of preference: 14 participants selected PwdHash and 11 chose P-Multiplier. The total number of responses is 25 because one participant only completed tasks with one system and could not compare the two.

5 Analysis and Interpretation of Results

Only one task in our study had a success rate of over 50%. This should concern the security community because when users cannot use a system correctly, they become vulnerable to attacks (see Section 5.4). It is important to examine the causes of failure in order to learn how to address these usability problems. The best source of information in this case is the observational data that recorded what happened as participants tried to use the systems.

Section 5.1 reports on usability problems common to both systems tested. Section 5.2 compares our findings with those of PwdHash's authors and section 5.3 reports on usability problems discovered specifically in P-Multiplier. Finally, section 5.4 discusses the particular security vulnerabilities exposed due to these usability problems.

5.1 Problems Common to Both Systems

Multiple issues arose because users' mental models did not match the reality of the system. They clearly were trying to make sense of what they saw and experienced during the interaction, but their understanding was incomplete or incorrect. Specifically, they had difficulty understanding when and how to activate each system, understanding how long it remains active once it is activated, determining to what fields the activation applied, and determining whether they had correctly accomplished a task.

Users were unsure about whether the systems were correctly activated. They often commented on "well, I *think* it did something" or "I guess that's what needed to be done". They perceived little feedback and were looking for some cue that they had been successful. One participant somehow decided that the "lock" icon on the browser that indicates whether the site is secure was the indicator of whether the password was protected. In each task, they looked at the icon to make sure it was closed then happily entered their password without activating the plug-in, fully believing that their password was protected. Another participant who could not figure out how to activate the plug-in reasoned "this password must be *really* secure – I can't even get in".

Another misconception was that they could activate the password manager once and it remained active throughout their computer session. They double clicked or pressed @@ with the very first password they entered and then assumed that all further passwords from this point onwards were protected without further action. This raises serious concerns because it gives users

a false sense of security. They believed they were protected while in fact their weak passwords continued to be used for their accounts. They were able to log in to their accounts because they never actually converted their passwords to "protected" passwords even though they believed that they had. In some cases this might possibly lead to even weaker password choices than normal because users believe they are being protected.

A second activation problem arose with each password managers' "alternative" trigger mechanism: pressing the F2 key for PwdHash and Alt+P for P-Multiplier. Both of these required that the users' cursor was already in the password field before triggering the program. Users would forget to click on the password field, then incorrectly assume that the program had been activated when they pressed F2 or Alt+P.

Several users erroneously believed that unique, random passwords were generated for them each time they activated the password manager; even for the same web site. For example, they believed that each time they logged on to Hotmail and used the password manager, a new, unique password was being generated. Of course this would not work since a web site expects the same password each time in order to authenticate the user. But this view was even held by participants who would be considered advanced or expert computer users such as Computer Science graduate students.

Not all usability problems encountered were a direct result of the password manager interfaces. Some problems were due to bad web site design. The sites used in the study were popular sites frequented by expert and novice users alike so the observed problems are likely to occur in real life as well. Participants had difficulty finding the login areas, had difficulty finding where to change their passwords, had difficulty changing their passwords, and had difficulty determining if they were correctly logged in to a site. These are valid usability issues that provide context and insight into the circumstances and environments where people will be using password managers. They must be taken into account even though they are not a direct result of the password manager interface. Another problem noticed by several participants was the inconsistency in designating the username field: it was called "username", "account name", or "email" on different web sites. The instruction sheet referred to it as "username"; this was sufficient to raise several questions.

Several participants gave up on tasks out of frustration, especially with P-Multiplier. Most said that in real life, they would have requested that their password be reset or would have created a new web account by this point. None mentioned that they would have looked for further documentation. Some assumed that something was wrong with the account and asked the observer to correct it, while others apologized for their "stupidity" and blamed themselves for the problem.

Another frustration shared by many participants was that they did not know their "actual" passwords. A second subset never even realized that this was in fact what was happening, revealing further mental model issues. Of those who understood this concept, they felt that the program should "trust" them with their own passwords: "why won't it tell me my password?" and "I wish it would show me my password when it first generates it, I won't lose it or share it!". Presumably, the tendency would be for users to write down their passwords for safekeeping; this has advantages and disadvantages, depending on the threat model.

Users also did not like relinquishing control to a computer program: "How do I know I can trust it?", "Now I have to trust two companies with my passwords, who guards the guards?" and "I felt like I had no control. Passwords are important but I have no idea what happened." Even though they know that password security is a problem, they feel that they are best equipped to care for their own passwords. This view is clearly contrary to that of security experts. The mismatch needs to be addressed if widespread adoption of security measures is to take place.

5.2 Comparison of findings with previous PwdHash findings

The authors of PwdHash briefly report the findings of their user study as follows [24]:

- *"The participants did not experience any major difficulties signing up for new accounts and logging in to them using the password prefix."*
- *"The user interface was so invisible that many participants did not observe the extension doing anything at all. They did not notice the lock icon, and their only clue that the extension was working was the fact that their password changed length when focus left the field, which they found confusing"*
- *"It was only once the users had to log in using a different browser that didn't have PwdHash installed that they encountered difficulties. They found the process of copying over the site address into the remote hashing page to be annoying, and if they did so incorrectly (for example, using gmail.com instead of google.com) the site that they were logging into did not provide useful feedback as to what went wrong."*
- *"When presented with a fake eBay site at a numeric IP address, most of the participants were willing to try logging in anyway, but their use of the password-prefix prevented the phishing page from reading their eBay passwords."*

Very little detail is given about the user study, its methodology, its participants, or the actual results. This makes it difficult to assess the validity of the study and its thoroughness, or to attempt a replication study. We address

each of their reported results by comparing them to our findings and report additional usability problems that we discovered. Our results contradict those found by the authors [24] with respect to the usability of PwdHash. We found that even though performing better than P-Multiplier, PwdHash still had major usability problems. While the success rates for our tasks with PwdHash were fairly high, more than half of these successful outcomes were only achieved after repeated attempts and errors by the users. Many successes were due to participants trying random actions until eventually something worked, not because participants had a clear understanding of how to use the software.

The majority of our participants did not comment on the changing length of their password. Many did, however, fail to realize that they needed to enter @@ in front their new password when re-typing it in for confirmation on a Change Password page. They assumed that entering the @@ once was enough. Visually, it was apparent that the two new passwords did not match due to the length difference, but users did not notice this cue. Many were confused by the lack of feedback and did not know if they had been successful in activating PwdHash. A similar problem occurred with P-Multiplier where users would activate the plug-in the first time they entered their password then assumed that the program remained active and proceeded to re-type their password in the password confirmation field without re-activating P-Multiplier.

Most of our users also had difficulty with the remote interface, but the problems encountered were different from those reported in the original study. First, users had difficulty reaching the remote web site due to its complicated URL. Several commented that they "would never find this site when they needed it". The second concern cited by several users was that the remote web site did not ask them for a username. They found this disturbing and could not understand how it was going to generate the correct password when it did not know who they were. "How will it know to generate *my* password?" and "How does it know who I am?" were common questions. One shrewd participant eventually concluded that "wait, it's going to give anyone who enters my regular password the same complicated password? That's not good!". Users obviously believe that the generated passwords are (and should be) somehow unique to them. This points to problems with users' mental model of the systems.

The users who successfully completed the remote login task did not have any trouble identifying the correct site URL to enter (granted, we were using www.amazon.com as the site, alternatives may not have been as obvious to users as with the Google and Gmail problem reported in the first study). Of those who failed to complete the task, most never even reached the remote web site. Although they were explicitly told that "you are now at your friend's house, they don't have the software installed, so you will need to log in remotely", they still attempted to log in using the @@. When this failed, they attempted to use their plain password and made random attempts at guessing the correct password. They did not refer to their instruction sheet that clearly had a section entitled "To log in remotely:". Even when users had only half a page of instructions, directly in front of them, they tended not to refer to it. On a similar note, only one user actually read and commented on the instructions posted on the remote web site about whether to enter @@ in front of their password on the remote site. Without reading the instructions, about half of the users entered their password with the @@ and half omitted it. Apparently, the software is capable of dealing with both situations because they were successful in generating their password in both cases. This may be good in terms of anticipating users' actions but breaks the mental model of what the @@ symbols represent.

We did not test if users would try to log in to a fraudulent site. Based on our observations, we predict that users would attempt to log on without realizing the danger. And based on users' behaviour at other sites when they had an incorrect password, it is our guess that users would divulge their plain text password when the first log in attempt failed and they tried alternatives. Thus even though PwdHash would protect their initial attempt, users would provide their clear text password on subsequent attempts thus creating a security vulnerability.

PwdHash alerts users when it thinks they are trying to enter a password into a non-password field. The alert message is long; most users simply dismissed it without reading it. Compounding the problem is that this alert sometimes appeared with legitimate password fields, usually when a user would start typing a password and then try to include the @@ as an afterthought.

PwdHash more closely mimics the interaction with which users are familiar. From a conceptual point of view, users simply have to remember that passwords starting with @@ are protected. They do not need to really understand that the @@ is invoking a plug-in that performs a cryptographic hash on their password. On the other hand, this familiarity also caused confusion. One participant voiced this concern: "Really, I don't see how my password is safer because of two @'s in front". This user obviously understood that the @@ were important but could not make the connection with typing in these symbols and triggering a program that will generate a stronger password. Some people were uncomfortable with the transparency, commenting that "at least with P-Multiplier I get a window, the password is going into the program".

5.3 Comments on usability of Password Multiplier

Although no user study was documented, we now comment on statements made in Halderman et al. [11] about the usability of P-Multiplier. They argue that their approach is *"both convenient for users and highly secure, a combination not offered by previous designs"*. In terms of convenience, they claim that their software can be run on different machines to access their password from anywhere and that the memory load is light because users only need to remember a username, the domain name for the site being accessed, and a single master password. They also highlight their browser integration, stating that this integration *"allows it to work as conveniently and transparently as possible, minimizing the burden imposed on the user and increasing the chances that the system will be used in practice. Second, browser integration allows our system to offer some protection against spoofing and phishing attacks, since by default the password program will use the name of the server that will actually receive the submitted password, even if an attacker has tricked the user into believing she is connecting to a different site"* [11].

We argue that P-Multiplier fails to meet the usability goals its authors set for themselves. P-Multiplier is not easily portable since it requires that the program be installed on a remote computer. Many users do not have privileges allowing them to install software on such machines and would likely find it inconvenient to install software in order to log in to a web site.

The memory load is not trivial. Users must remember the master username and master password for P-Multiplier. In addition, they must still remember their username for each web site. And finally, they must remember the modified domain names for all sites where they have changed their password.

Transparency is often a cited goal for security systems. However transparency is often translated to a lack of feedback and this leads to many usability problems as users are unable to form an accurate mental model of the system and its operation. Two participants never realized that they only needed one password with P-Multiplier. Each time, they entered a clear text password in the web site's password field, double clicked to activate the dialog box and then entered the master password. They never realized the generated password was overwriting their clear text password and that they did not have to manually enter anything into the web site's password field.

P-Multiplier's authors state that their plug-in helps to protect against phishing attacks. While this is true, it relies on the assumption that users correctly use the system. This is a problematic assumption, as from our observations, users frequently entered their plain text passwords and resorted to random guessing because they could not correctly use P-Multiplier. In these cases, users are no more protected against phishing than they would be without a password manager. (Some users might possibly be less diligent in scrutinizing sites because they assume that the software is protecting them.)

Users had high expectations of the password managers. They expected that any password generated by the software should be extremely strong, regardless of their own input. One web site offered a "strength-meter" when changing passwords, rating the strength of new passwords. When the password manager failed to generate a password considered "strong" by the web site, users expressed their concern and disapproval. Specifically, passwords generated by P-Multiplier were often rated as "medium" by Hotmail rather than "strong". Users felt that if they chose their own passwords, they would be able to produce a strong password.

P-Multiplier's solution for changing a password once it is protected is nearly impossible for users to understand. Of those who did understand it, they expressed frustration at now having to remember the modified domain and having to enter it each time they logged in to this web site. There also seems to be a bug in the program where users were able to change their password by entering a different master password instead of changing the domain. The plug-in accepted this incorrect master password and somehow generated a new password for the site. It was unclear whether the passwords so generated were secure, and it violated the user's mental model that all passwords in P-Multiplier are protected by a single master password.

5.4 Usability problems causing security exposures

Usability problems are of general concern in HCI and should also concern the security community because they may also lead users to bypass security mechanisms. Here, we comment on a separate matter: usability problems which may directly cause security exposures, even when the user has the intention of complying with the security mechanism.

The first concern is users failing to properly activate the respective mechanisms (e.g., failing to enter @@ or double-click the web site password field). For both password managers, when users enter their password but forget to invoke the mechanism, the (raw) password is sent on as if the manager software was not installed, naturally exposing it to JavaScript and phishing attacks – both of which PwdHash was designed to counter. In the case of P-Multiplier, this exposes what is the master password, sufficient to generate site passwords for many sites. This user error is a factor in three possible failure outcomes (Dangerous Success, Failure, False Completion).

The same problem exists when users change passwords. The interface requires entering the old password, the new password, and re-entering the latter. In some

cases, users activate the respective mechanism for the first new password field, but then re-enter their password the second time *without* activating it.

A different problem resulted from user confusion. When users failed to correctly activate the mechanism, some started guessing, entering all passwords they could think of, with and without the activation sequence. In this case, a phishing site (or JavaScript attack) could harvest a number of passwords of interest.

Another generic danger, not specifically related to the password managers in question, is that users who believe a password manager mechanism will be used may be more inclined to choose a simple (low-entropy) password, believing it will be strengthened; when it is not, it may be even more vulnerable to dictionary attacks.

6 Further Discussion and Recommendations

As discussed in Section 1, the problem of establishing and managing strong passwords for large numbers of accounts is unlikely to disappear soon. While password managers offer a solution, current implementations suffer from significant usability problems, which in our observation fall into two main categories: (1) users' mental models; and (2) users' views on the necessity of tools such as password managers, and acceptance (their willingness to hand over control of passwords to a computer program). We discuss these, in turn, in Sections 6.1 and 6.2. In Section 6.3, we consider additional criteria for security software to be usable.

6.1 Mental Model

Incorrect user mental models appears to be the biggest problem with the two password managers tested. While manifested in different ways, the common cause is that users do not even have a high-level understanding of what the manager programs are doing. The importance of a mental model for usability in security is a key issue identified by Whitten and Tygar [31], and as early as 1975 by Saltzer and Schroeder [25].[6] To be usable, a system must support a user's mental model, and must fit into their regular work patterns. This is not simply a matter of user satisfaction; failure to do so can result in security exposures. Through interactions with a system, users form a mental model as a mechanism to help in understanding, learning, and remembering. The primary means of doing so is by interpreting the actions and interface of the system as they are presented; an incoherent or inappropriate presentation leads to an erroneous, or incomplete model. A user's model serves as the basis for their interactions; thus improper models often lead to major usability problems. Norman's [21] Gulf of Evaluation is a measure of how much effort must be exerted to determine whether the system actions correspond to the user's intentions and expectations. Ideally this gulf should be small, indicating that the system provides accurate, understandable, and visible feedback to the user.

It is the responsibility of a good designer to ensure that users have the cues needed to form an accurate and complete mental model of a system [20]. Although the usability literature has long supported the need for accurate mental models, this seems not yet to be common knowledge – or perhaps more correctly, not common practice – for the security community. Users need not fully understand the details of complex security programs, but rather need a mental model that is consistent, and that will allow them to predict program behaviour and the results of their own actions.

In addition, the interfaces should provide better feedback. It is a well accepted usability principle that system feedback is necessary to narrow the Gulf of Evaluation and for users to develop accurate mental models. In our study, users were often left in states where they could not tell if their actions had been successful. In some cases, the systems provided feedback but it was not noticed by the users. P-Multiplier changed the background colour of the password fields containing generated passwords, but in our study not one user remarked on this subtle change or appeared to have noticed it. PwdHash alerted users when they tried to enter a password in a non-password field. However this alert sometimes appeared when users were in a valid field and its wording was confusing for users. Most users simply dismissed the warning. Participants were also confused when something had obviously gone wrong, but they received no feedback as to what may have caused the problem or how to recover from it. In their effort to be unobtrusive and subtle, these interfaces have reached a point where they are impossible to comprehend due to a lack of cues.

We make the following suggestions regarding feedback provided to users by the password managers tested:

1. It should be obvious when a password has been protected.
2. It should be obvious when the plug-in has been activated and is awaiting input. This directly contradicts the assumption that transparency is good for security interfaces. The lack of visual cues was problematic for both programs because it left users confused and unsure about how to proceed.
3. It should be clear how existing passwords are migrated (from pre-manager unprotected, to with-manager protection). Even with instructions on how to activate the program and an explanation that users must change their password, confusion arose. Several users attempted to "change" their password at the initial login prompt for a webpage rather than logging on then using the site's Change Password

interface. They felt that since the program was installed on the computer, "changing their password" meant that they could simply start using the plug-in to enter their password on a web site.

4. If something goes wrong, feedback should be short, understandable, and reveal how to address the problem. This is a standard usability principle. Unfortunately, it is unclear if this would be easy to implement since the problem may stem from the target web site rather than from the plug-in.
5. There should be a way for users to check which of their accounts are currently protected. Migrating to one of these systems is non-trivial since users will need to track which accounts have been migrated and which remain to be done. We suggest that the plug-in keeps a list of currently protected web accounts on the user's primary computer.

Better integration with the actual web pages may be technically difficult, but it would certainly help users. For example, when a password is incorrectly entered, users currently do not know if the problem is a typing error or an error in activating the manager. We observed users resorting to random guessing in hopes that something would work – with one security risk being that such passwords could all be exposed (see Section 5.4) and might include sensitive passwords (which the user resorts to trying) for unrelated accounts. Providing accurate error messages would reduce user frustration.

6.2 User Acceptance and View of Necessity

From a usability standpoint, user satisfaction and acceptance is always important (although sometimes users may have no choice, e.g., when it is required for a critical part of their job). When users must make an "opt-in" choice, it is particularly important that users accept the system – otherwise they may turn to an alternative (possibly insecure) service or find ways to bypass the security mechanisms [2, 5, 26]. Lack of user satisfaction and acceptance can lead to a lack of security.

We believe that helping users form a clearer mental model would significantly help with user acceptance of the studied password managers. Currently, users are uncomfortable with using the software (see Figure 3) and do not trust it because they do not understand it. They are worried about the safety of their accounts. They are worried that they will be unable to reach their accounts because the password manager will stand in their way. As an intermediary, the password manager needs to appear reliable, consistent, and predictable.

To increase user acceptance, we also recommend that along with the installation of password managers, users be educated about the importance of protecting passwords, and how password managers can achieve this goal. As with other security measures this is not a simple task since security is not the primary goal of most end-users; however, password managers have the advantage of potentially simplifying users' tasks (e.g., by requiring them to remember fewer or less-complicated passwords). Once users understand this, we would expect higher user acceptance.[7]

6.3 Criteria for Security Software to be Usable

Whitten and Tygar highlighted issues that arise when users have inaccurate or incomplete mental models, suggesting that for *security* software to be usable, users must [31]:

1. be reliably made aware of the security tasks they must perform;
2. be able to figure out how to successfully perform those tasks;
3. not make dangerous errors; and
4. be sufficiently comfortable with the interface to continue using it.

We suggest the following two additional criteria (closely related or supporting 2 and 3):

5. **be able to tell when their task has been completed**; and
6. **have sufficient feedback to accurately determine the current state of the system.**

The fifth concerns a usability problem seen in both the Whitten and Tygar study and our current study: users were unable to tell whether their task had been successfully completed and sometimes incorrectly assumed success. This can cause security vulnerabilities (e.g., as information believed to be secure can be left unprotected). The sixth draws on the well-known usability guideline of feedback, which is especially important for supporting accurate mental models in security interfaces. Transparency in this case can be dangerous because it leaves users free to make assumptions about the system that could lead to security exposures.

7 Related Work

Background on usability testing is given in Section 3.1. Section 2 mentions a few alternate password managers (see [11] and [24] for a good summary of other password managers). Here we focus on related work including usability tests for authentication mechanisms. Although the situation is now changing significantly, there have been surprisingly few such academic papers.

Growing interest is reflected by Cranor and Garfinkel [4], and the Symposium on Usable Privacy and Security (SOUPS). Zurko and Simon [34] introduced "user-centered security" in 1996. Prominent among past work is the case study of PGP 5.0 [31], which included a cognitive walkthrough inspection analysis and a lab user test involving 12 participants (see

Section 6). Another early authentication usability study is the Déjà Vu work [8], which included interviews with 30 people on password behaviour, and user testing with 20 participants; the focus of the user testing was on creation of password (image) portfolios, and memorability results. Prior to this, Adams and Sasse [1] explored password-related user behaviours and memorability issues through questionnaires and interviews, leading to a number of recommendations.

Recent papers involving user studies on graphical passwords include Davis et al. [6] with focus on security and poor user choices made by real users; and Wiedenbeck et al. [32] with focus on memorability in the PassPoints system, presenting results of a user study involving 32 undergraduates. Weinshall [30] introduces a challenge-response authentication protocol relying on recognition of images and presents results of a small user study.

Although not specifically on passwords, Garfinkel and Miller [10] carried out a 43-subject user test of a secure email prototype with focus on key continuity management features (automating certain key and certificate management activities related to signing email); they report increased protection against certain forms of social engineering, but not from attacks from new (unfamiliar) email addresses or from phishing.

Related to our observations that more visibility (vs. more transparency) would enhance usability of some security features, Depaula et al. [7] explore making relevant features of security mechanisms – including configurations, activities and implications of available security mechanisms – visible, to allow more informed user decisions.

8 Concluding Remarks

While the security community seeks to develop systems with stronger security, it is now commonly recognized that even the most technically secure system, if unusable, will fail in practice. However, without measurements on real users, we cannot evaluate usability. Usability tests with real users should be included in not only the development of security systems, but also in research which proposes new security tools and plug-ins. Both formative and summative usability tests are desirable. Formative tests are conducted throughout the development of the system to guide the development and find potential usability problems as they arise; these tests are typically less formal and their goal is to highlight any problems. Summative tests are used to gather performance data and provide measures of usability; they are more controlled and their goal is to validate the usability of a system or compare its performance for different groups of users.

We have refrained from making specific suggestions for changing the interfaces of the studied password managers, as any suggestions should themselves be tested for usability – and we have not done so. Thus until such time, we cannot authoritatively conclude that they would work. Instead, we have suggested further guidelines and requirements for the interfaces. Further work is needed to identify specific mechanisms to use in order to comply with these guidelines and address the requirements.

The goal of usability studies is to uncover problems so that they can be corrected. We have identified several usability problems with Password Multiplier and PwdHash which we believe are likely to exist in other similar password manager proposals. The next step is to identify mechanisms, if possible, by which these interface problems can be addressed. Ideally, we would then build a new interface that implements these mechanisms, and conduct further usability evaluation to test if this actually improves usability.

9 Acknowledgements

We thank the anonymous reviewers for their comments which helped improve this paper to its present form. We also thank the members of the Carleton's Digital Security Group and Mary Ellen Zurko for their feedback on earlier versions of this work. The first and third authors are supported in part by the "Legal and Policy Approaches to Identity Theft" project funded by the Ontario Research Network for E-Commerce. The second author is Canada Research Chair in Network and Software Security, and is supported in part by the Canada Research Chairs Program, and an NSERC Discovery Grant.

Notes

[1] Lack of feedback hindered users' ability to form accurate mental models, and to determine if passwords were being protected.

[2] One person did not try P-Multiplier; they quit after completing the tasks with PwdHash. Therefore only partial data is available for this participant. This left 25 participants for P-Multiplier.

[3] Using a third computer would have been a better experimental design, allowing participants to complete the task, but we do not expect that this would have led to different results.

[4] Technical problems caused one participant to miss the *Log In* and *Second Login* tasks with PwdHash

[5] A t-test is a ratio giving a measure of the difference between two means relative to the variability of each set. Larger ratios mean that the two groups are more distinct from each other. Significance p shows the likelihood that the results are due to chance.

[6] They note [25]: "Psychological acceptability: It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly. Also, to the extent that the user's mental image of his protection goals matches the mechanisms he must use, mistakes will be minimized."

[7] Ideally, this hypothesis would be verified by a separate study.

References

[1] A. Adams and M.A. Sasse. Users are not the enemy. *Comm. of the ACM*, 42(12):41–46, 1999.

[2] R. Anderson. Why cryptosystems fail. In *Proceedings of the 1st ACM Conference on Computer and Communications Security.*, December 1993.

[3] J.M. Carroll, P.L. Smith-Kerker, J.R. Ford, and S.A. Mazur-Rimetz. The minimal manual. *Human-Computer Interaction*, 3:123–153, 1987-1988.

[4] L.F. Cranor and S. Garfinkel. *Security and Usability: Designing Systems that People Can Use*. O'Reilly Media, edited collection edition, 2005.

[5] D. Davis. Compliance defects in public key cryptography. In *Proceedings of the 6th USENIX Security Symposium*, July 1996.

[6] D. Davis, F. Monrose, and M. Reiter. On user choice in graphical password schemes. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[7] R. DePaula, X. Ding, P. Dourish, K. Nies, B. Pillet, D. Redmiles, J. Ren, J. Rode, and R. Silva Filho. Two experiences designing for effective security. In *First Symposium on Usable Privacy and Security (SOUPS 2005)*, Pittsburgh, July 2005.

[8] R. Dhamija and A. Perrig. Déjà Vu: A User Study Using Images for Authentication. In *Proceedings of the 9th USENIX Security Symposium*, 2000.

[9] L. Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, 2003.

[10] S.L. Garfinkel and R.C. Miller. Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express. In *First Symposium on Usable Privacy and Security (SOUPS 2005)*, Pittsburgh, July 2005.

[11] J. Halderman, B. Waters, and E. Felten. A convenient method for securely managing passwords. In *Proceedings of the 14th International World Wide Web Conference*, 2005.

[12] J. Alex Halderman. Password Multiplier web site, www.cs.princeton.edu/~jhalderm/projects/password/, accessed January 2006.

[13] A. Karp. Site-specific passwords. Technical report, Hewlett-Packard Laboratories, January 2002.

[14] J. LaPoutre. Password Composer web site, http://www.xs4all.nl/~jlpoutre/BoT/Javascript/, accessed January 2006.

[15] Password Maker web site, http://passwordmaker.org/, accessed January 2006.

[16] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140, June 1932.

[17] B. Myers. Why are human-computer interfaces difficult to design and implement? Technical Report CMU-CS-93-183, Carnegie Mellon University, Department of Computer Science, 1993.

[18] J. Nielsen. *Usability Engineering*. Boston: AP Professional, 1993.

[19] J. Nielsen and R.L. Mack. *Usability Inspection Methods*. John Wiley & Sons, Inc, 1994.

[20] D.A. Norman. Cognitive engineering. In D.A. Norman and S.W. Draper, editors, *User Centered System Design: New Perspectives on Human-Computer Interaction*, chapter 3, pages 31–62. Lawrence Erlbaum Associates, Publishers: Hillsdale, NJ, 1986.

[21] D.A. Norman. *The Design of Everyday Things*. Basic Books, 1988.

[22] C. Perfetti and L. Landesman. Eight is not enough. *User Interface Engineering*, 2001.

[23] K. Renaud. Evaluating Authentication Mechanisms. In L.F Cranor and S. Garfinkel, editors, *Security and Usability*, chapter 6, pages 103–128. O'Reilly Media, 2005.

[24] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. Mitchell. Stronger password authentication using browser extensions. In *Proceedings of the 14th USENIX Security Symposium*, Baltimore, August 2005.

[25] J.H. Saltzer and M.D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[26] M.A Sasse and I. Flechais. Usable Security: Why do we need it? How do we get it? In L.F. Cranor and S. Garfinkel, editors, *Security and Usability*, chapter 2, pages 13–30. O'Reilly Media, 2005.

[27] B. Shneiderman. *Designing the User Interface*. Addison Wesley, 3rd edition, 1998.

[28] J. Spool and W. Schroeder. Testing web sites: Five users is nowhere near enough. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 2001)*, 2001.

[29] R.A. Virzi. Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors*, 34:457–468, 1992.

[30] D. Weinshall. Cognitive Authentication Schemes Safe Against Spyware (Short Paper). In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2006.

[31] A. Whitten and J.D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium*, Washington, D.C., August 1999.

[32] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Broditskiy, and N. Memon. Authentication Using Graphical Passwords: Effects of Tolerance and Image Choice. In *First Symposium on Usable Privacy and Security (SOUPS 2005)*, Pittsburgh, July 2005.

[33] N. Wolff. Password Generator web site, http://angel.net/~nic/, accessed January 2006.

[34] M.E. Zurko and Richard T. Simon. User-centered security. In *Proceedings of the 1996 New Security Paradigms Workshop*, pages 27–33, Lake Arrowhead, CA USA, 1996. ACM.