# Preventing the Revealing of Online Passwords to Inappropriate Websites with LoginInspector

Chuan Yue *
*University of Colorado at Colorado Springs*
*Department of Computer Science*
*Colorado Springs, CO 80918, USA*
*cyue@uccs.edu*

## Abstract

Modern Web browsers do not provide sufficient protection to prevent users from submitting their online passwords to inappropriate websites. As a result, users may accidentally reveal their passwords for high-security websites to inappropriate low-security websites or even phishing websites. In this paper, we address this limitation of modern browsers by proposing *LoginInspector*, a profiling-based warning mechanism. The key idea of LoginInspector is to continuously monitor a user's login actions and securely store hashed domain-specific successful login information to an in-browser database. Later on, whenever the user attempts to log into a website that does not have the corresponding successful login record, LoginInspector will warn and enable the user to make an informed decision on whether to really send this login information to the website. LoginInspector can also report users' insecure password practices to system administrators so that targeted training and technical assistance can be provided to vulnerable users. We implemented LoginInspector as a Firefox browser extension and evaluated it on 30 popular legitimate websites, 30 sample phishing websites, and one new phishing scam discovered by M86 Security Labs. Our evaluation and analysis indicate that LoginInspector is a secure and useful mechanism that can be easily integrated into modern Web browsers to complement their existing protection mechanisms. Security system administrators in our university commented that such a tool could be very helpful for them to strengthen campus IT security.

***K*eywords:** Web browser, password, security, phishing, JavaScript, Web, authentication

## 1 Introduction

Text passwords still occupy the dominant position in online user authentication [2, 16, 17, 23], and they provide an effective protection to our valuable online accounts. Password security heavily depends on creating strong passwords and protecting them from being stolen. However, in recent years, especially with the rampancy of Web-based malicious activities [31, 57], passwords have increasingly been targeted by various harvesting or stealing attacks. For example, one of the most severe threats to online users is the phishing attack [6, 8, 10, 11, 19, 28, 32, 34, 35, 40, 45, 47, 51], which uses spoofed websites to steal users' passwords and financial information.

To protect users' online passwords and accounts, modern Web browsers have already implemented many security features and mechanisms. For example, the five most popular browsers (Internet Explorer, Firefox, Google Chrome, Safari, and Opera) all support extended validation (EV) certificates to help users verify the authenticity of websites, and they also provide phishing detection and warning mechanisms to help users stay away from phishing websites. However, simply relying on a single layer of protection is insufficient in many cases. Providing modern browsers with defense-in-depth mechanisms to better protect users' online passwords is definitely important and necessary.

In this paper, we investigate modern browsers' limitation in preventing users from submitting online passwords to inappropriate websites. We highlight that at least in two cases, accidental online password revealing may happen. The first case is that if a browser fails to detect a phishing webpage, users may submit their online passwords to the phishing website and become victims. This case can happen because automatic phishing detection techniques are still not able to detect all the phish-

*This work is based on Jeff Hinson's master thesis titled "Preventing the Revealing of Online Account Information to Non-Relevant Websites" [18]. Jeff built the initial prototype of LoginInspector in his thesis, and Chuan extended that prototype and completed this work. Jeff also helped revise this final version of paper after acceptance. Jeff prefers to be simply acknowledged rather than be an author of the paper. Chuan respects Jeff's choice and most sincerely appreciates him for his important contributions to this work.

ing attacks in a timely manner and meanwhile maintain a very low false positive rate [4, 13, 29, 39, 48, 49]. The second case is that when users forget the passwords for a certain website, a common practice for them is to try the passwords for other websites that they do remember. However, this practice may reveal a user's login information for a high-security website such as a banking website to an inappropriate low-security website such as a gaming website. This second case can happen and we confirmed this possibility by conducting a user study (Section 2). For ease of presentation, in this paper we refer to the first case as *undetected phishing attacks*, and the second case as *risky password tries*.

In both cases, users' online passwords can be revealed to inappropriate websites that should not receive them. Unfortunately, modern browsers do not provide sufficient protection to users in these two cases; meanwhile, no previous research has been done to seriously address this limitation of modern browsers. Indeed, in both cases, the key problem is that browsers do not have the knowledge to inform a user that such types of login actions could reveal the user's password to an inappropriate website. More specifically, when a browser has no knowledge (i.e., fails to detect) that a user is visiting a phishing webpage, the browser is not able to warn the user to leave this webpage; when a browser has no knowledge that a user is trying some mismatching passwords (i.e., passwords for other websites), the browser is not able to prevent the user from actually submitting the passwords to the current website.

To address the limitation of modern browsers in preventing users from submitting online passwords to inappropriate websites, we propose *LoginInspector*, a profiling-based warning mechanism. The key idea of LoginInspector is to continuously monitor a user's login actions and securely store hashed domain-specific successful login information to an in-browser database. Later on, whenever the user attempts to log into a website that does not have the corresponding successful login record, LoginInspector will warn and enable the user to make an informed decision on whether to really send this login information to the website. LoginInspector can also report users' insecure password practices to system administrators of an enterprise or campus environment so that targeted training and technical assistance can be provided to vulnerable users.

LoginInspector is more like a whitelist-based approach. Its in-browser database is like containing a whitelist of a user's successful login records for different websites, and the whitelist is dynamically built up and securely maintained. Using this whitelist, LoginInspector can enable a user to make informed decisions in both the aforementioned cases. In particular, in the case of undetected phishing attacks, as long as the user had not already become a victim of this specific phishing website, a corresponding successful login record for this phishing website does not exist in the database. Similarly, in the case of risky password tries, as long as a tried username and password pair does not belong to a valid account of the user on the current website, a corresponding successful login record for this tried login information on the current website does not exist in the database. Therefore, in both cases, LoginInspector can accurately warn a user and allow a user to cancel the actual submission of the password to the website; it can also send the related information to system administrators and assist them to further protect and train users who cannot properly interpret the warning messages.

LoginInspector is a pure browser-side mechanism. No server-side deployment is needed, and no modification to a user's passwords is needed. LoginInspector is designed as an auxiliary tool to help a user enhance the security of the online passwords. Therefore, it will not incur any functionality problem (albeit without providing protection) to a user's login activities even if, for example, the user needs to use a computer that does not have LoginInspector installed.

We have implemented a prototype of LoginInspector as a Firefox browser extension and evaluated it on 30 popular legitimate websites, 30 sample phishing websites, and one new phishing scam discovered by M86 Security Labs. Our evaluation and analysis indicate that LoginInspector is a secure and useful mechanism that can help users prevent the accidental revealing of passwords to inappropriate websites. It is a simple mechanism that can be easily integrated into modern Web browsers to complement their existing protection mechanisms. Security system administrators in our university commented that such a tool could be very helpful for them to strengthen campus IT security.

The remainder of the paper is organized as follows: Section 2 motivates this work by reviewing related research and presenting a user study of online login practices. Section 3 details the design of LoginInspector. Section 4 analyzes the security, usability, and deployment of LoginInspector. Section 5 describes the implementation and evaluation of LoginInspector. Finally, Section 6 makes a conclusion and discusses the future work.

## 2 Motivation

Password security heavily depends on creating strong passwords and protecting them from being stolen. Weak passwords suffer from brute-force and dictionary attacks [30]; therefore, many online services require users to create and use strong passwords that are sufficiently long, random, and hard to crack by attackers. How-

ever, strong passwords are difficult to remember for users [1, 9, 30, 42]. As demonstrated by a recent large-scale usability study, many users write down or otherwise store their passwords and especially those with a higher entropy [25]. Furthermore, no matter how strong they are, passwords are also vulnerable to harvesting or stealing attacks.

In some cases, users may accidentally reveal the password for one website to another inappropriate website, making their sensitive login information for the original website at risk. We now justify that at least in the cases of *undetected phishing attacks* and *risky password tries*, accidental online password revealing may happen. The case of undetected phishing attacks is related to the ever-increasing prevalence of password harvesting attacks. The case of risky password tries can be attributed to the reality that Web users have more online accounts than ever before, and they are forced to create and remember more and more usernames and passwords probably using insecure practices such as sharing passwords across different websites [12, 36].

## 2.1 Undetected Phishing Attacks

This first case happens when browsers fail to detect a phishing attack and give a warning about it. In such a case, a vulnerable user may submit the password for the real website to the inappropriate phishing website. Phishers can directly use the obtained login information to break into the user's online account.

To protect Web users against phishing attacks [6, 8, 10, 11, 19, 28, 32, 34, 35, 40, 45, 47, 51], modern browsers often employ automatic phishing detection and warning mechanisms [54, 55, 56]. In terms of automatic phishing detection, two general types of techniques are blacklist-based techniques [29, 39, 48] and heuristic-based techniques [4, 13, 49]. No matter what techniques are used, browsers are still not able to detect all the phishing attacks in a timely manner and meanwhile maintain a very low false positive rate [4, 13, 29, 39, 48, 49]. Therefore, modern browsers cannot protect vulnerable users if those phishing attacks cannot be detected in the first place.

LoginInspector is more like a whitelist-based mechanism: even if a browser fails to detect a phishing attack and give a warning about it, our mechanism can still provide one more layer of protection by explictly giving a warning to the user and informing the user that he or she did not log into this website before. A related work, AntiPhish [24], can also generate a warning message whenever a user attempts to give away sensitive information to a phishing website. However, LoginInspector uses password hashing techniques while AntiPhish uses password encryption techniques. We believe hashing is more appropriate than encryption in this application because essentially what we need is an authenticator rather than a reversible mapping. Moreover, the detection and warning capability of LoginInspector is more refined than that of AntiPhish. In AntiPhish, whenever a mismatch on password happens, a phishing attempt is assumed (Section 3.2 of [24]). Such a decision criterion is not accurate. For example, if a user types a wrong password on a legitimate website, AntiPhish will assume the current website is a phishing website, but LoginInspector will display one of the two warnings (Figure 5) to provide accurate information to the user.

## 2.2 Risky Password Tries

This second case happens when users forget the password for a certain website. In such a case, a common practice for users is to try the passwords for other websites they do remember. However, if a password for a high-security website is tried on a low-security website, then this password may be revealed to the low-security website. For example, if a user attempts to use his or her Gmail password on a low-security gaming website, the Gmail password is revealed to the gaming website. The user may also try the corresponding Gmail username, may use the same username for Gmail and for the gaming website, or may use the Gmail address as the contact information on that gaming website. Therefore, both the Gmail password and username could be known to the gaming website. If this low-security gaming website is hacked or even if its authentication log is unintentionally released, the revealed Gmail login information could be further acquired by attackers.

The authors of this paper are also guilty of this practice of risky password tries. To further validate that this risky practice is indeed a common practice, we conducted a user study. This user study was pre-approved by the IRB (Institutional Review Board) of our university. Thirty adults, 15 females and 15 males, participated in our user study. They were voluntary students and faculty members randomly recruited in our campus library, bookstore, and cafeteria; they came from 17 departments of our university. Twenty-two participants were between ages of 18 and 30, and eight participants were over 30 years old. We did not collect any other demographic or sensitive information from participants. We did not screen participants based on their Web browsing experiences. We did not provide any incentive to the participants. We interviewed the participants when we met them on campus and asked each of them to answer a five-point Likert-scale (Strongly disagree, Disagree, Neither agree nor disagree, Agree, Strongly Agree) [58] questionnaire that consists of seven close-ended questions as listed in Table 1. Note that we randomized the sequence of the seven questions for each individual participant.

Table 1: The seven close-ended questions.

| |
|---|
| Q1: The security of online passwords is a concern. |
| Q2: Sometimes I forget my login username for a website. |
| Q3: Sometimes I forget my login password for a website. |
| Q4: Sometimes I type the username for one website to log into another website. |
| Q5: Sometimes I type the password for one website to log into another website. |
| Q6: When I type the username for one website to log into another website, I hope the Web browser can give me a warning. |
| Q7: When I type the password for one website to log into another website, I hope the Web browser can give me a warning. |

Table 2: Summary of the responses to the seven close-ended questions.

| | **Strongly disagree** | **Disagree** | **Neither agree nor disagree** | **Agree** | **Strongly Agree** |
|---|---|---|---|---|---|
| Q1 | 0(0.0%) | 3(10.0%) | 1(3.3%) | 17(56.7%) | 9(30.0%) |
| Q2 | 1(3.3%) | 5(16.7%) | 2(6.7%) | 20(66.7%) | 2(6.7%) |
| Q3 | 1(3.3%) | 3(10.0%) | 0(0.0%) | 24(80.0%) | 2(6.7%) |
| Q4 | 1(3.3%) | 1(3.3%) | 3(10.0%) | 20(66.7%) | 5(16.7%) |
| Q5 | 0(0.0%) | 3(10.0%) | 3(10.0%) | 18(60.0%) | 6(20.0%) |
| Q6 | 0(0.0%) | 3(10.0%) | 11(36.7%) | 12(40.0%) | 4(13.3%) |
| Q7 | 0(0.0%) | 2(6.7%) | 9(30.0%) | 13(43.3%) | 6(20.0%) |

A summary of the responses to the seven close-ended questions is presented in Table 2. Because the data collected are ordinal and do not necessarily have interval scales, we use the median and mode to summarize the data and use the percentages of responses to express the variability of the data. Overall, the median and mode responses are *Agree* for all the seven questions. Particularly in terms of passwords (Q3, Q5, and Q7), we can see that: (1) 86.7% of participants agree or strongly agree that sometimes they forget the password for a website; (2) 80.0% of participants agree or strongly agree that sometimes they try the password for one website on another website; and (3) 63.3% of participants agree or strongly agree that when they try the password for one website on another website, they hope the Web browser can give them a warning.

These results clearly indicate that users do sometimes forget passwords, and trying passwords for other websites is indeed a common practice. Answers to Q7 also indicate that most participants are aware of the risks of such type of password tries and they do expect browsers to give them a warning for their risky actions. Unfortunately, modern Web browsers do not provide protection to prevent risky password tries; meanwhile, no previous research has been done to seriously address this limitation of modern browsers.

## 2.3 Related Work on Password Management

To help Web users better manage their online accounts and enhance their password security, researchers have proposed a number of solutions such as password managers [50, 61], single sign-on systems [63], graphical passwords [7, 20], and password hashing systems [14, 33, 43]. These solutions have their own merits and advantages, but they also pose various reliability, security, and usability concerns. Browsers' built-in password managers as well as many third-party password managers [50, 61] must be able to recover the original passwords by decrypting the saved encrypted passwords. This requirement provides many opportunities for attackers to crack a password manager that is not well designed or implemented. LoginInspector does not need to recover the original passwords. Essentially, even if a user does not want to use any password manager, LoginInspector can still protect against undetected phishing attacks and risky password tries.

Web Wallet [41] is an anti-phishing solution, and essentially it is a password manager that can help users fill login forms using stored information. However, as pointed out by the authors, users have a strong tendency to use traditional Web forms for typing sensitive information instead of using the special browser sidebar user interface. Centralized single sign-on systems such as Microsoft Passport [63] may suffer from various attacks that could cause disastrous consequences [26]. Security limitations of graphical passwords are analyzed in [5, 38]. Security and usability limitations of password hashing systems such as Password Multiplier [14] and PwdHash [33] are analyzed in [3]. Both Password Multiplier and PwdHash require users to migrate their original passwords to hashed passwords, and this is a biggest usability limitation of those hashing-based password generation solutions as acknowledged in the Password Multiplier paper [14].

LoginInspector leverages the security advantages of password hashing techniques, but it does not inherit their usability disadvantages because it only uses hashing techniques to generate authenticators for determining warning types. Overall, even if some of existing solutions can to some extent help prevent accidental login information revealing, the majority of users who stick to the traditional way of using passwords (i.e., filling out a login form based on what they remember in the memory and submitting their original passwords to the remote website) still cannot be protected. What LoginInspector aims to accomplish is to prevent these users from accidentally revealing their sensitive login information.

Essentially, our key observation is that currently Web browsers do not have the knowledge to identify the afore-

mentioned accidental login information revealing cases. Therefore, they cannot help a user make informed decisions to avoid submitting sensitive login information to inappropriate websites. This observation motivated us to explore a simple profiling-based warning mechanism to provide an in-depth protection for Web users.

## 3 Design

In this section, we first give an overview on the design of LoginInspector. We then detail the architecture and components of LoginInspector.

### 3.1 Overview

The key idea of LoginInspector is to continuously monitor a user's login actions and securely store domain-specific successful login information to an in-browser database. Later on, whenever the user attempts to log into a website that does not have the corresponding successful login record, LoginInspector will warn and enable the user to make an informed decision on whether to really send this login information to the website.

We design LoginInspector as a browser extension that can be seamlessly integrated into modern Web browsers. As illustrated in Figure 1, the functioning of the LoginInspector browser extension consists of two main logical phases: the *profiling phase* and the *inspection and warning phase*. These two phases are centered around an in-browser *successful login profile database*.

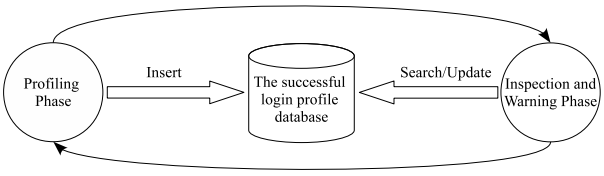


Figure 1: The functioning of the LoginInspector browser extension.

In the profiling phase, LoginInspector will build up the successful login profile for a user. Basically, whenever a user successfully logs into a website account for the first time, LoginInspector will insert a new record into the successful login profile database. Each record is uniquely determined by the domain name of the website and the username/password pair used in this successful login. For example, assume a user has two Twitter accounts A and B. The username/password pair is userA/pwdA for account A, and is userB/pwdB for account B. The first time the user successfully logs into twitter.com using account A, one new record will be created; the first time the user successfully logs into twitter.com using account B, another new record will be created. As will be elaborated in the next subsection, LoginInspector only stores hashed domain names and username/password pairs into the successful login profile database, thus minimizing security and privacy risks even if database records would be stolen by attackers.

In the inspection and warning phase, LoginInspector will leverage the information stored in the successful login profile database to enable a user to make informed login decisions. Basically, when a user types the username and password into a login form of a website, LoginInspector will first intercept this information. Second, it will inspect whether there is a corresponding successful login record for this website account in the database. Third, if there is a corresponding successful login record in the database, LoginInspector will submit the intercepted login information to the remote website; no warning will be given to the user and the user's login interaction is identical to that in the scenario of without using LoginInspector. LoginInspector can also update the corresponding database record with information such as the login timestamp.

However, if there is no corresponding successful login record in the database, LoginInspector will warn the user with accurate information regarding either the user did not log into this website before or the current login information does not match previous successful login records. It will ask the user to confirm whether this login information should really be submitted. If the user ignores the warning, LoginInspector will submit the intercepted login information to the website; if the user acknowledges the warning, LoginInspector will cancel the submission and allow the user to type in a new username and password.

LoginInspector is a pure browser-based solution – both the profiling phase and the inspection and warning phase happen inside of a user's browser. LoginInspector will ensure no password information will be transmitted to a remote website unless: (a) one corresponding successful login record exists in the database (indicating the current website is an appropriate website), or (b) the user ignores the warning and explictly confirms the transmission (indicating the user has made an informed decision based on the given warning).

### 3.2 Architecture and Components

Figure 2 illustrates the high-level architecture of LoginInspector, which consists of *the successful login profile database* and seven logical components: *login fields identification and protection*, *login profile inspection*, *warning generation*, *admin report*, *successful login detection*, *management*, and *import/export*.

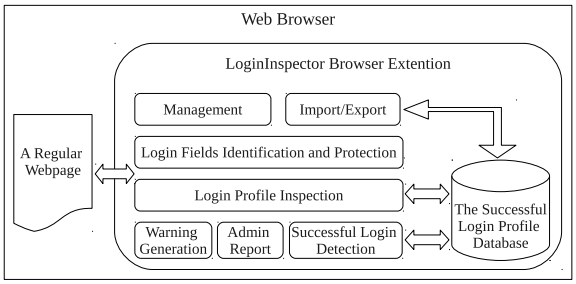


Figure 2: The architecture of the LoginInspector browser extension.

### 3.2.1 The successful login profile database

LoginInspector uses an in-browser database that can be implemented as an SQLite [62] database instance. SQLite has already been equipped in popular browsers such as Firefox and Google Chrome. Within the database, a *loginprofile* table is created to store all the successful login records. This table consists of six columns as shown in Table 3.

Table 3: The *loginprofile* table in the successful login profile database.

| id | domainHmac | recordHmac | timesUsed | firstUsed | lastUsed |
|---|---|---|---|---|---|

In the *loginprofile* table, each successful login record is uniquely identified by a recordHmac value. A unique *id* is generated by the database for each record. The values of domainHmac and recordHmac are calculated using Formula 1 and Formula 2, respectively:

$$domainHmac = HMAC(key,\ d) \quad (1)$$

$$recordHmac = HMAC(key,\ d \mid\mid u \mid\mid p) \quad (2)$$

where *key* is a secret key either randomly generated by the LoginInspector extension or directly specified by a user when LoginInspector is installed; *HMAC* is the Keyed-Hashing for Message Authentication [27] mechanism together with the SHA-256 [59] cryptographic hash function; $d$, $u$, and $p$ represent domain name, username, and password, respectively; "||" is the string concatenation operator. The secret key is securely stored in the password manager of a browser and transparently used by LoginInspector. We allow a user to specify the secret key when LoginInspector is installed so that the records in the successful login profile database can be conveniently exported and imported by LoginInspector. A user can also use the master password mechanism of a browser to further protect against the wrong people extracting the secret key.

The domain name $d$ is extracted from each login form's owner document, and it includes the full domain name prefixed with the protocol (e.g., https://www.amazon.com or http://en.wikipedia.org). In other words, we are interested in where exactly a login form comes from, instead of what the domain name of the top-level document loaded from a browser's URL address bar is. This design choice is reasonable because the owner document contains the most relevant information of a form. For example, on a mashup website (e.g., www.mashup.com), if a login form is submitted from a sub-frame document (e.g., specified by the *src* attribute of a *frame* element), the domain name of this sub-frame document will be extracted and used as the value for domain name $d$. Therefore, the saved successful login record can be matched no matter the owner document of the login form is included in mashup websites as a sub-frame document or is directly loaded as a top-level document. Similar to browsers' other features such as the password manager, LoginInspector uses more reliable and stable domain names instead of IP addresses and does not specifically consider pharming attacks which we believe should be addressed by some general solutions such as [21, 22].

The other three columns timesUsed, firstUsed, and lastUsed of the *loginprofile* table can keep track of the usage statistics of each record and provide a user with more detailed successful login information on a currently visited website. Users can configure whether they want to save these information to the database or not, by using the management component of LoginInspector.

### 3.2.2 Login fields identification and protection

When a new webpage is loaded in the browser, the login fields identification and protection component will identify the username and password login fields on the webpage; it will also intercept and protect a user's password keystrokes. It follows the strategy of first identifying the password field and then identifying the username field. To identify the password field, it uses two combined techniques: user-assisted identification and automatic identification. The user-assisted identification technique was proposed by Ross et al. in their PwdHash project [33]. This technique requires a user to either type the "@@" prefix (two consecutive "at" signs) or press the "F2" function key to explicitly indicate that an input field is a password field. The advantage of this technique is that it can effectively prevent malicious JavaScript on phishing webpages from stealing users' plaintext passwords. The disadvantage of this technique is that users need to remember to perform this additional action [3].

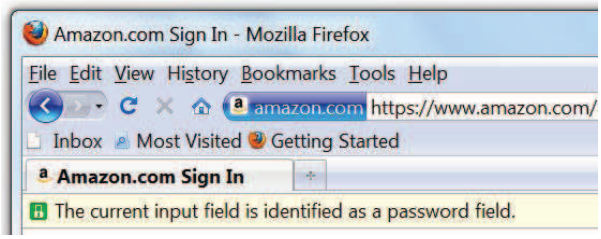


Figure 3: The password field indication message in a chrome notification box.

The automatic identification technique examines the special attribute *type="password"* in the DOM [53] tree to locate a password field. This technique can be evaded by phishers [33], but it is completely automatic.

The two identification techniques are combined in such a way that if a user indicates a password field, this field will be directly identified; otherwise, the automatic identification result will be used. Basically, once an input field is automatically identified as a password field, LoginInspector will mark that field as a protected field. Then, if and only if the input focus is on that field, LoginInspector will display a chrome type of notification box to a user below the browser's tab bar to make the user aware of that password field. Figure 3 illustrates a snapshot of the notification box. The notification box is of a chrome type, so that a LoginInspector user can customize it by putting special text or images into it. Therefore, a malicious webpage cannot easily spoof the notification box.

The user-assisted identification may be needed in two cases. One is when the automatic identification does not identify any password field. The other is when the automatically identified password field is inconsistent with the *should-be* password field as perceived by a user. In both cases, a user can set the input focus on the should-be password field and specify it as the password field using either the "@@" prefix or the "F2" function key [1]. LoginInspector will mark this user-specified password field as a protected field, and will similarly display the notification box if and only if the input focus is on this field. Note that usually this user-assisted identification may only need to be performed on phishing webpages, on which a password field could be deliberately made inaccurate [33].

If no password field is identified (either automatically or by the user), LoginInspector will do nothing anymore. Otherwise, the current webpage is regarded as a login webpage and LoginInspector will further identify the corresponding username field. Our experimental results (Section 5) indicate that the visible text input field immediately preceding the password field can be reliably identified as the username field. We found that Firefox also uses this username identification technique in its password manager. Even if the username field cannot be confidently identified on a login webpage, the functionality of LoginInspector will not be affected. In such a case, an empty string will be used as the username value for calculating the recordHmac. Therefore, the only effect is that the granularity of the successful login record becomes coarse if a user has multiple accounts sharing the same password on this website. Note that if a user has successfully logged into a website (Section 3.2.6), the username will be remembered by LoginInspector for this login session. Thus, if a user visits the change password webpage of this website, LoginInspector can still properly replace the old successful login record with the new calculated one.

After identifying the username and password fields of a login webpage, LoginInspector will monitor these two fields and will extract the login information typed by a user into these two fields. To prevent malicious JavaScript on a webpage such as a phishing webpage from recording a user's password keystrokes, LoginInspector will (1) intercept the password keystrokes as soon as a user begins to type, (2) prevent the real keypress events from propagating to the webpage, and (3) fire fake keypress events to generate the random fake password value. These three steps are basically the same as the ones developed by Ross et al. in their PwdHash project [33]. Later on, if the login form really needs to be submitted, LoginInspector will use the intercepted real password value to replace the generated fake password value, allowing the login process to proceed smoothly.

### 3.2.3 Login profile inspection

When a user submits a login form, the login profile inspection component will compare the intercepted user login information with the records stored in the successful login profile database. First, it will compute a *currentDomainHmac* and a *currentRecordHmac* using Formula 1 and Formula 2, respectively, based on the domain name extracted from the current login form's owner document and the username and password values intercepted from the current login form. Next, it will look up the database using the login profile inspection procedure illustrated in Figure 4. It passes the (currentDomainHmac, currentRecordHmac) pair to the procedure to get one of the three return results. The result *ExactMatch* means that there is an existing record with the recordHmac value equal to currentRecordHmac. The result *DomainMatch* means that there is no existing record with the recordHmac value equal to currentRecordHmac, but there is at least one record with the domainHmac

[1] To discard any identification result, a user can simply press the "F2" function key when the input focus in on that field.

```
Inspection (currentDomainHmac, currentRecordHmac)
1. if a record with recordHmac=currentRecordHmac exists
2.     return ExactMatch;
3. else
4.     if a record with domainHmac=currentDomainHmac exists
5.         return DomainMatch;
6.     else
7.         return NoMatch;
8.     endif
9. endif
```

Figure 4: The login profile inspection procedure.

value equal to currentDomainHmac. The result *NoMatch* means that there is no existing record with the domainHmac value equal to currentDomainHmac.

If the return result of the procedure is ExactMatch, the login profile inspection component will simply submit the login form using the intercepted real password, and no warning will be given to the user. The timesUsed and lastUsed information of the existing successful login record can also be updated. However, if the return result of the procedure is either DomainMatch or NoMatch, the login profile inspection component will further instruct the warning generation component to trigger a warning message.

#### 3.2.4 Warning generation

This component will generate two types of warning messages based on the instruction from the login profile inspection component. One type of message, referred to as *Initial Visit*, corresponds to the return result NoMatch. In this case, the warning message will remind a user that the user may not have previously logged into this website, and will ask whether the user really wants to proceed with the login action. The other type of message, referred to as *Credential Mismatch*, corresponds to the return result DomainMatch. In this case, the warning message will remind a user that the user may not have previously used this username and password pair to successfully log into the current website, and will also ask whether the user really wants to proceed with the login action. In both cases, the warning message will be displayed in a modal chrome dialog box. This dialog box is of a modal type, so that before continuing to perform any other browsing interactions, a user must respond to the warning by either clicking the "OK" button to *ignore* it or clicking the "Cancel" button to acknowledge it. Similar to the notification box illustrated in Figure 3, this dialog box is also of a chrome type; therefore, it can be customized and cannot be easily spoofed by a malicious webpage. Figure 5(a) and Figure 5(b) illustrate the snapshots of the dialog boxes for the two types of warning messages, respectively.

The *Initial Visit* warning message can be triggered when a user tries to log into a new legitimate website or a phishing website. The *Credential Mismatch* warning message can be triggered when a user tries to log into a website using the username and password information of a new account for the first time or using the username and password information for another website. These warning messages intend to help users make informed decisions, and we expect users can properly interpret these messages and make the correct decisions. If a user ignores a warning message, the login action continues and the username and password information will be sent to the website; if a user acknowledges a warning message, login action stops and nothing will be sent to the website.

A user can follow three basic principles to decide whether to ignore or acknowledge a warning message. First, at the beginning when LoginInspector is installed and used, the *Initial Visit* warning message will be given on each website because no existing record exists in the database. A user normally should ignore the warning. Second, after a user has successfully logged into most of his or her online accounts (e.g., after a couple of weeks), the occurrence of the *Initial Visit* warning message should be rare. Therefore, a user should be very cautious about this type of warning message and should carefully inspect whether the current website is a phishing website. The user should acknowledge the warning if the website is suspicious, and should otherwise ignore the warning. Third, the *Credential Mismatch* warning message should be rare all the time. If a user has two accounts on the same website, then ideally this type of message should only occur once when the user logs into the website using the second account for the first time. Therefore, a user should be very cautious about this type of message and should think about whether the tried login information is for another website (thus the warning should be acknowledged) or for another account on the same website (thus the warning should be ignored). These principles can be provided in the manual of LoginInspector to help users properly interpret the warning messages.

#### 3.2.5 Admin report

This component will generate and send reports to system administrators if it is enabled by either a system administrator or a user. In an enterprise or campus environment, system administrators can configure LoginInspector so that even if some users cannot properly interpret the above warning messages, the related information can be reported to an internal website of administrators through a POST type HTTP request.

The report does not contain passwords or their hash values; it only needs to contain the LoginInspector us-

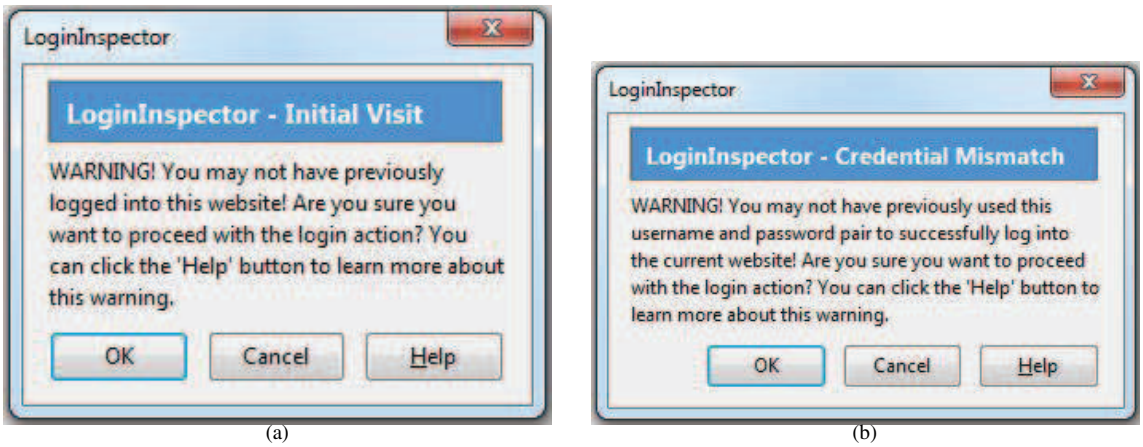


Figure 5: The modal chrome dialog boxes for (a) the *Initial Visit* warning message, (b) the *Credential Mismatch* warning message.

age information. For example, it can contain statistical information on a user's responses to the two types of warning messages in a session such as {"userid": "123456", "ignored *Initial Visit* warning": "10 times", "ignored *Credential Mismatch* warning": "6 times", "sessionStartTime": "1345846451434", "sessionEndTime": "1345846648635", .....}. If necessary, it can also contain the URLs of the websites on which warning messages were ignored. System administrators can then leverage the reported information to identify users' insecure password practices such as risky password tries or improper interpretation of warning messages, and they can further take actions to protect and train those vulnerable users. Administrators could also aggregate the reports received from different users to predict a phishing wave. For example, if suddenly a large number of *Initial Visit* messages are ignored by a bunch of users on some associated IP addresses or domain names, the likelihood of a new phishing scam is high. Security system administrators in our university commented that LoginInspector and its report capability could be very helpful for them to strengthen campus IT security.

#### 3.2.6 Successful login detection

This component will detect whether a user's login attempt is successful in the case when the user ignores a warning (Figure 5) and LoginInspector submits the login information to the website. The case that the login profile inspection procedure returns ExactMatch is directly considered as a successful login and is not handled by this component. Originally, we intended to make this detection process completely automatic. We found that one reliable heuristic for automatically detecting a successful login is to examine whether the login response webpage loaded on the browser also contains a visible password field. If so, this login action is considered unsuccessful because normally a failed login webpage asks a user to type in the username and password information again. Otherwise, the login action is considered successful. This technique works accurately on over 95% of websites that we tested (Section 5), but in rare cases the detection result was wrong. For example, the response webpage for a failed login may simply contain a link or button, which can take a user to the re-login webpage. In such a case, a failed login will be incorrectly identified as a successful login.

To overcome the limitation of the automatic detection, we introduced a user-assisted successful login detection method. Basically, once a login response webpage is fully loaded on the browser, LoginInspector will explictly ask a user to confirm whether this login attempt is successful. This confirmation message is also displayed in a modal chrome dialog box, and a user's response on the "Yes" or "No" button will be directly used as the detection result. This method is intuitive because a well-designed login response webpage often clearly manifests the status of the login action, which can be easily and accurately identified by a user. Moreover, this method will not impose too much burden on a user because the confirmation is needed only when a warning (Figure 5) is given and meanwhile the warning is ignored by the user. Once a successful login is confirmed by a user, a new record (Table 3) will be added to the *loginprofile* database table.

#### 3.2.7 Management and import/export

Finally, the management component will enable a user to perform tasks such as customizing warning messages, removing successful login records, and configuring whether to track the timesUsed, firstUsed, and lastUsed information. The import/export component will enable a user to export the successful login records to a

file and later to import those records from this file to a new browser on another computer. This import/export functionality is similar to that of the bookmark feature in Web browsers. We mentioned in Section 3.2.1 that the secret key used in Formula 1 and Formula 2 can be specified by a user. The advantage is that when a user imports the successful login records to a new browser on another computer, the user can directly type the specified key into LoginInspector through its management user interface. Otherwise, the randomly generated secret key also needs to be exported from the original browser and then supplied to the new browser, and such a functionality should only be accessed by authorized users. A user's profile should be synchronized between computers so that previously processed warning messages will not appear again. Currently, users can use this import/export functionality to achieve profile synchronization. In the future, we plan to enable users to take advantage of the data synchronization mechanism of browsers such as Firefox and Google Chrome to easily perform profile synchronization.

# 4 Analysis

We now analyze the security, usability, and deployment of LoginInspector.

## 4.1 Security

On the one hand, LoginInspector itself is designed to be secure. A user's login information for a website is hashed using the HMAC [27] message authentication mechanism together with the cryptographically strong SHA-256 [59] hash function, and only hashed values are stored in the successful login profile database. Therefore, even if attackers can manage to steal the complete database file or some successful login records, it is computationally infeasible for them to figure out the original plaintext username/password and domain name values that map to those HMAC values. Meanwhile, because this mechanism does not send the intercepted login information or hashed successful login records to any third-party server, it will not incur any new security or privacy problems.

On the other hand, LoginInspector can provide security benefits to a user by giving two types of warning messages (Figure 5) based on the previous successful login history of the user. This capability of LoginInspector is unique and is exactly what is lacking in existing Web browsers. Moreover, thanks to domain-based hashing, this capability is robust regardless of whether a user will reuse usernames and passwords across different websites or not. We now further analyze this capability for three categories of users based on the two accidental login information revealing cases defined in Section 2.

The first category of users are security conscious users who will never visit phishing websites and never perform risky password tries. These users do not need to use LoginInspector. If they do use, they will see very few warning messages once their successful login profiles become stable, and they can simply ignore those messages. The second category of users may accidentally visit phishing websites and become victims. Whenever a user in this category tries to log into a phishing website, regardless of the browser's ability to detect the attack, LoginInspector will display the *Initial Visit* warning message and explicitly inform the user that he or she may not have previously logged into the website. The third category of users may sometimes perform risky password tries. Whenever a user in this category performs a risky password try, LoginInspector will display the *Credential Mismatch* warning message and explicitly inform the user that he or she may not have previously used this username and password pair to successfully log into the current website. Note that there could be an overlap between the second category and the third category of users.

Modern Web browsers display "active" warnings to boost the effectiveness of their phishing and SSL error protection mechanisms [8, 37]. Because LoginInspector displays warning messages in a modal chrome dialog box, these warnings are also "active" and a user has to take an action. Therefore, it is reasonable to expect that by following the principles suggested in Section 3.2.4, users in the second and third categories can, to some extent, properly interpret the warning messages and protect themselves from accidentally revealing login information. Training users to read, understand, and (most importantly) pay serious attention to the warning messages is absolutely critical to the effectiveness of security warning mechanisms. Herley discussed that "users' rejection of the security advice they receive is entirely rational from an economic perspective" [15]. Following the recommendations provided in [15], we suggest that such a training should target at-risk population, that is, those who are vulnerable to phishing attacks and/or who have the practice of risky password tries, so that a better cost-benefit ratio can be achieved.

## 4.2 Usability

LoginInspector has two major usability advantages. One is that a user does not need to change the original passwords for any website. Some existing password management or phishing protection solutions such as Password Multiplier [14], PwdHash [33], and Passpet [43] all require users to visit the "change password" page of each individual website to migrate the original unmanaged password to a hashed password. However, such a

requirement imposes a big burden on a user. As pointed out in the Password Multiplier paper [14], "all the hash-based schemes have difficulty with the transition from unmanaged passwords."

The other usability advantage is that the login action of a LoginInspector user will not be affected at all, even if the user needs to use a browser on a computer that does not have LoginInspector installed. This is because LoginInspector is designed as an auxiliary tool to help a user enhance the security of the online passwords. The regular login functionality will be enhanced with one more layer of security protection, but it will not be degraded or disrupted when LoginInspector becomes unavailable. This usability advantage is lacking in those password hashing solutions [14, 33, 43]; it is also lacking in browsers' built-in or third-party password managers [50, 61].

LoginInspector also has two main usability disadvantages. One is that the two types of warning messages, especially the *Initial Visit*, will be frequently displayed during the profiling phase, and a user should ignore the warnings to build up the successful login profile. To better ensure the quality of this profiling phase, we suggest a user to perform it in a batch manner once LoginInspector is installed. For example, if a user has 30 online accounts on 20 websites, the user can log into those 30 accounts in about one hour to establish his or her successful login profile. During this process, if the user carefully submits the valid login information for all the 30 accounts, then 20 *Initial Visit* warning messages will be displayed and 10 *Credential Mismatch* warning messages will be displayed. The user simply needs to ignore all these 30 warnings to finish the profiling phase. Later on, the user will not see any warning message if the login information of those 30 accounts are used to log into the corresponding websites. The user only needs to be cautious about the two types of warning messages if they appear again. In an enterprise or campus environment, system administrators can also help regular users build up the profile, thus reducing the number of required actions from regular users and making the profiling stage of LoginInspector easier, faster, and less error-prone.

Note that the impact of webpage redirection to LoginInspector is very limited because similar to browsers' built-in password managers, LoginInspector only extracts URLs from the final login webpage instead of from an intermediate redirection webpage. It is very rare for the same website to use different URLs to host the login webpage for the same type of accounts[2]; if such a case happens, LoginInspector may display the *Initial Visit* warning message, and browsers' built-in password managers may ask a user to save another record to the database.

The other main usability disadvantage is that the established successful login profile is associated with a browser on one computer, and is not directly accessible on other computers. To address this issue, we suggest that if a user has multiple computers, the user can simply export the established successful login profile and import it to other computers. Therefore, the aforementioned profiling phase still only needs to be performed once in a batch manner, minimizing the burden on a user. However, if a user simply wants to temporarily use another computer such as a public computer [46], we do not suggest the user import his or her successful login profile to that computer. In other words, LoginInspector mainly focuses on protecting a user on his or her own computers.

[2] A bank website may use different login URLs, but normally they correspond to different types of accounts such as credit card accounts and saving accounts.

## 4.3 Deployment

LoginInspector can be incrementally deployed and the deployment is very simple. One reason is this mechanism is a pure browser-based solution and it can be seamlessly integrated into modern Web browsers as an extension. No server-side modification is needed. The other reason is that this mechanism simply provides one additional layer of protection to Web users. It only uses the existing login information of a user and it does not require any modification to the existing online user authentication mechanisms.

# 5 Evaluation

LoginInspector is designed to be implementable on different Web browsers. In-browser databases such as SQLite [62] are equipped in popular browsers such as Firefox and Google Chrome, making the implementation of the successful login profile database feasible. Meanwhile, the end-user extensibility of modern browsers such as Firefox, Internet Explorer, and Google Chrome also makes the implementation of other LoginInspector components feasible. We have implemented a prototype of LoginInspector as a Firefox browser extension. It works well as tested on Firefox versions from 3 to 9, which was the latest version at the time of this writing. We do not anticipate problems with newer versions. The extension is purely written in JavaScript with approximately 1600 lines of code. We believe LoginInspector can also be easily implemented as the extension for other popular browsers.

We have evaluated this LoginInspector extension on 30 popular legitimate websites, 30 sample phishing websites, and one new phishing scam discovered by M86 Security Labs [60]. We selected the 30 popular legitimate websites (as listed in Table 4) from two sources. One

source is the top 50 websites listed by Alexa.com; however, we removed non-English websites, gray content (e.g., adult) websites, and the websites that did not allow us to create an account. The other source is some of our frequently used websites. Websites such as paypal.com and wellsfargo.com set the *autocomplete="off"* property on their password fields or login forms; therefore, browsers' autocomplete feature [44] will not save users' form filling history to help speed up their future form filling process [44]. LoginInspector only stores hashed values to dramatically reduce the risk of having users' passwords cracked by attackers; thus its current version does not respect the *autocomplete* property and can work well on websites with the "autocomplete=false" property. Note that LoginInspector may not work well on websites that use one-time passwords because it could always raise the *Credential Mismatch* warning after the initial visit. What a user can do is to configure LoginInspector to ignore those sites based on their domain names.

We selected the 30 phishing websites from phishtank.com, which is a community based anti-phishing service widely used in research [29, 48, 49]. These phishing websites were randomly sampled with the criteria that they were online during our experiments, they were containing login webpages, and they were hosted on different domains or IP addresses. In our experiments, we mainly focused on evaluating the correctness and performance of this LoginInspector extension.

Table 4: The 30 popular legitimate websites.

| | | |
|---|---|---|
| mail.google.com | facebook.com | mail.yahoo.com |
| wikipedia.com | twitter.com | amazon.com |
| linkedin.com | wordpress.com | ebay.com |
| fc2.com | craigslist.org | imdb.org |
| aol.com | digg.com | careerbuilder.com |
| buy.com | aaa.com | newegg.com |
| tumblr.com | alibaba.com | 4shared.com |
| cnn.com | nytimes.com | foxnews.com |
| weather.com | groupon.com | photobucket.com |
| myspace.com | webmail.uccs.edu | portal.prod.cu.edu |

## 5.1 Correctness

We verified that this LoginInspector extension integrates seamlessly with the Firefox Web browser and works correctly on all of the 30 popular legitimate websites, the 30 sample phishing websites, and the new phishing scam discovered by M86 Security Labs [60].

### 5.1.1 Results on legitimate websites

On the 30 popular legitimate websites, LoginInspector can correctly and automatically identify the password field and the username field on each of the login webpages. It can correctly intercept password keystrokes and replace the intercepted password with a generated fake password; it can properly replace back the intercepted password when the login information really needs to be sent to the website. We observed the heuristic for automatic successful login detection that we originally planed to use (Section 3.2.6) works correctly on 29 websites except for aaa.com, which uses an extra response webpage that contains a link "Return to sign in page" for a failed login attempt. As discussed in Section 3.2.6, we switched to a user-assisted successful login detection method to overcome the limitation of the automatic detection. This method works correctly based on the user confirmation action.

Through logging all the operations and manually checking the content of the *loginprofile* table, we verified that all the database operations – including insert, update, and select – were correctly performed for the 30 websites. Meanwhile, the login profile inspection procedure illustrated in Figure 4 works correctly based on the existing records in the *loginprofile* database table, and the decisions on whether and what type of warning messages (Section 3.2.4) should be displayed were also precisely made. Note that in these correctness evaluations, whenever doable, we created at least two accounts on each website to test all the possible usage scenarios.

### 5.1.2 Results on phishing websites

On the 30 sample phishing websites, we observed that the password field and the username field can be correctly and automatically identified on 29 login webpages. Only on one login webpage the password field was not automatically identified by LoginInspector. We checked that the password field on that login webpage has the property *type="text"*, and every password character was displayed to a user in the input field. These results indicate that, overall, the sophisticated phishing attacks presented by Ross et al. [33] are not yet used in many phishing attacks. However, as analyzed in Section 3.2.2, if the automatic identification does not work well on phishing websites like in the above *type="text"* case, a user should specify the password field using either the "@@" prefix or the "F2" function key.

We also verified that the *Initial Visit* warning message was correctly displayed by LoginInspector on all the phishing login webpages. In our experiments, we acknowledged all those warnings because we do not want to save any phishing website record to the successful login profile database. In addition, among these 30 phishing websites, we observed that Firefox failed to detect seven of them and Google Chrome failed to detect eight

of them[3]. Therefore, on those phishing websites, no warning was displayed by these two popular browsers. This observation further justifies one of the motivations of our work, that is, *undetected phishing attacks* are commonplace.

#### 5.1.3 Results on one new phishing scam

A new phishing scam was discovered by M86 Security Labs in 2011 [60]. Basically in this scam, phishers attach an HTML file to the spam email, luring a user to open the attached HTML file and submit a form to perform some urgent tasks. Once a user submits the form, the stolen sensitive information will be transmitted through a POST type HTTP request to a hacked legitimate website. This new phishing scam is very stealthy because: (1) a browser simply loads the phishing webpage as a local file such as *file:///C:/Users/.../home.html*; (2) the form submission target is a legitimate, albeit hacked, website. Therefore, neither a browser nor a user can easily identify such a phishing attack. As reported by M86 Security Labs [60], popular browsers such as Firefox and Google Chrome did not detect any such malicious activity; meanwhile, there is an increase in these types of phishing spam campaigns over the last few months.

We decided to test whether LoginInspector can defend against this new phishing scam. We created emails to attach various login webpages, and then opened those attachments using latest versions of Firefox, Google Chrome, and Internet Explorer browsers. Obviously, no phishing warning was given by those Web browsers, and it seems those browsers currently do not use heuristic-based phishing detection techniques to inspect locally opened (i.e., *file:///...*) HTML webpages. As expected, LoginInspector also works correctly on the locally opened HTML webpages. It correctly displayed the *Initial Visit* warning message, thus enabling a user to make the informed decision to acknowledge warnings and cancel the submissions when those undetected phishing attacks occur.

### 5.2 Performance

We also measured the performance overhead of LoginInspector on the 30 popular legitimate websites. Firefox and the LoginInspector browser extension were installed on a laptop with a 2.67GHz CPU. Other JavaScript operations and HMAC calculations (Formula 1 and Formula 2) cause negligible overhead. For example, all the tested HMAC calculations were completed within 3 milliseconds. The overhead is mainly on the SQLite [62] database operations invoked by the JavaScript code. On each of the 30 legitimate websites, we measured the overhead of the database operations five times. We observed that all the select operations were completed within one millisecond. The average performance overhead for the insert operations is 140.6 milliseconds with a standard deviation of 47.2. The average performance overhead for the update operations is 70.2 milliseconds with a standard deviation of 13.1. We can see that, overall, the database operation overhead is still very low, and is only incurred when a login form is submitted. In addition, the insert operation overhead is only incurred when a new successful login record needs to be added to the database.

[3]Both Firefox and Chrome use the same blacklist provided by Google [52]; this slight difference in false negative rate could be caused by the blacklist download time difference between the two browsers.

## 6 Conclusion and Future Work

In this paper, we determined that modern Web browsers do not provide sufficient protection to prevent users from submitting their online passwords to inappropriate websites. We highlighted that in the cases of undetected phishing attacks and risky password tries, users may accidentally reveal their passwords for high-security websites to inappropriate low-security websites or even phishing websites. We proposed and presented *LoginInspector*, a profiling-based warning mechanism to address this limitation of modern browsers. LoginInspector establishes a successful login profile for a user and leverages this profile to enable a user to make informed login decisions and also enable system administrators to provide further protection or targeted training to vulnerable users. We analyzed the security, usability, and deployment of LoginInspector. We also evaluated the correctness and performance of the Firefox LoginInspector browser extension on legitimate and phishing websites. Our evaluation and analysis indicate that LoginInspector is a secure and useful mechanism, and it can complement Web browsers' existing mechanisms to provide an in-depth protection to a user's online login process.

In our future work, we plan to design a visual way (e.g., by using icons) to clearly differentiate the two types of warning messages (Figure 5). We want to evaluate whether a visually distinguishing factor could help users better understand what is happening without having to read those warning messages. We also plan to evaluate the usability of this standalone browser extension and then integrate it into the password managers of modern browsers. This integration can leverage the existing components of browsers' password managers. However, this integration will still allow users to independently enable either the LoginInspector or the password manager in case some users do not want to use both features.

## 7 Acknowledgments

We thank anonymous reviewers for their insightful comments and valuable suggestions. We thank our shepherd Mario Obejas for his great help in improving the final version of this paper. We also thank all the voluntary students and faculty members who participated in our user study. Jeff Hinson made important contributions to this work as highlighted on the first page of the paper. This work was partially supported by a UCCS 2011-2012 CRCW research grant.

## References

[1] ADAMS, A., AND SASSE, M. A. Users are not the enemy. *Commun. ACM 42*, 12 (1999), 40–46.

[2] BONNEAU, J., HERLEY, C., VAN OORSCHOT, P. C., AND STAJANO, F. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Proceedings of the IEEE Symposium on Security and Privacy* (2012), pp. 553–567.

[3] CHIASSON, S., VAN OORSCHOT, P. C., AND BIDDLE, R. A usability study and critique of two password managers. In *Proceedings of the USENIX Security Symposium* (2006), pp. 1–16.

[4] CHOU, N., LEDESMA, R., TERAGUCHI, Y., AND MITCHELL, J. C. Client-side defense against web-based identity theft. In *Proceedings of the NDSS* (2004).

[5] DAVIS, D., MONROSE, F., AND REITER, M. K. On user choice in graphical password schemes. In *Proceedings of the USENIX Security Symposium* (2004), pp. 151–164.

[6] DHAMIJA, R., AND J.D.TYGAR. The battle against phishing: Dynamic security skins. In *Proceedings of the SOUPS* (2005), pp. 77–88.

[7] DHAMIJA, R., AND PERRIG, A. Déjà vu: A user study using images for authentication. In *Proceedings of the USENIX Security Symposium* (2000), pp. 45–58.

[8] EGELMAN, S., CRANOR, L. F., AND HONG, J. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the CHI* (2008), pp. 1065–1074.

[9] FELDMEIER, D. C., AND KARN, P. R. Unix password security – ten years later. In *Proceedings of the Annual International Cryptology Conference (CRYPTO)* (1989), pp. 44–63.

[10] FELTEN, E. W., BALFANZ, D., DEAN, D., AND WALLACH, D. S. Web Spoofing: An Internet Con Game. In *Proceedings of the 20th National Information Systems Security Conference* (1997).

[11] FLORÊNCIO, D., AND HERLEY, C. Password rescue: A new approach to phishing prevention. In *Proceedings of the HotSEC* (2006).

[12] FLORÊNCIO, D., AND HERLEY, C. A large-scale study of web password habits. In *Proceedings of the WWW* (2007), pp. 657–666.

[13] GARERA, S., PROVOS, N., CHEW, M., AND RUBIN, A. D. A framework for detection and measurement of phishing attacks. In *Proceedings of the Workshop on Rapid Malcode (WORM)* (2007).

[14] HALDERMAN, J. A., WATERS, B., AND FELTEN, E. W. A convenient method for securely managing passwords. In *Proceedings of the WWW* (2005), pp. 471–479.

[15] HERLEY, C. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Proceedings of the New security Paradigms Workshop (NSPW)* (2009), pp. 133–144.

[16] HERLEY, C., AND VAN OORSCHOT, P. C. A research agenda acknowledging the persistence of passwords. *IEEE Security & Privacy 10*, 1 (2012), 28–36.

[17] HERLEY, C., VAN OORSCHOT, P. C., AND PATRICK, A. S. Passwords: If we're so smart, why are we still using them? In *Proceedings of the Financial Cryptography* (2009), pp. 230–237.

[18] HINSON, J. Preventing the Revealing of Online Account Information to Non-Relevant Websites. Master's Thesis (advised by Chuan Yue) at UCCS, `http://library.uccs.edu/search/o693952729`.

[19] JAKOBSSON, M., AND MYERS, S. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley-Interscience, ISBN 0-471-78245-9, 2006.

[20] JERMYN, I., MAYER, A., MONROSE, F., REITER, M. K., AND RUBIN, A. D. The design and analysis of graphical passwords. In *Proceeding of the USENIX Security Symposium* (1999), pp. 1–14.

[21] JUELS, A., JAKOBSSON, M., AND JAGATIC, T. N. Cache cookies for browser authentication (extended abstract). In *Proceedings of the IEEE Symposium on Security and Privacy* (2006), pp. 301–305.

[22] KARLOF, C., SHANKAR, U., TYGAR, J. D., AND WAGNER, D. Dynamic pharming attacks and locked same-origin policies for web browsers. In *Proceedings of the CCS* (2007), pp. 58–71.

[23] KELLEY, P. G., KOMANDURI, S., MAZUREK, M. L., SHAY, R., VIDAS, T., BAUER, L., CHRISTIN, N., CRANOR, L. F., AND LOPEZ, J. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *Proceedings of the IEEE Symposium on Security and Privacy* (2012), pp. 523–537.

[24] KIRDA, E., AND KRUEGEL, C. Protecting users against phishing attacks with antiphish. In *Proceedings of the Annual International Computer Software and Applications Conference (COMPSAC)* (2005), pp. 517–524.

[25] KOMANDURI, S., SHAY, R., KELLEY, P. G., MAZUREK, M. L., BAUER, L., CHRISTIN, N., CRANOR, L. F., AND EGELMAN, S. Of passwords and people: Measuring the effect of password-composition policies. In *Proceedings of the CHI* (2011), pp. 2595–2604.

[26] KORMANN, D. P., AND RUBIN, A. D. Risks of the passport single signon protocol. *Comput. Networks 33*, 1-6 (2000), 51–58.

[27] KRAWCZYK, H., BELLARE, M., AND CANETTI, R. RFC 2104, HMAC: Keyed-Hashing for Message Authentication, 1997. `http://www.ietf.org/rfc/rfc2104.txt`.

[28] KUMARAGURU, P., RHEE, Y., ACQUISTI, A., CRANOR, L. F., HONG, J., AND NUNG, E. Protecting people from phishing: The design and evaluation of an embedded training email system. In *Proceedings of the CHI* (2007), pp. 905–914.

[29] LUDL, C., MCALLISTER, S., KIRDA, E., AND KRUEGEL, C. On the effectiveness of techniques to detect phishing sites. In *Proceedings of the DIMVA* (2007).

[30] MORRIS, R., AND THOMPSON, K. Password security: a case history. *Commun. ACM 22*, 11 (1979), 594–597.

[31] PROVOS, N., RAJAB, M. A., AND MAVROMMATIS, P. Cybercrime 2.0: when the cloud turns dark. *Commun. ACM 52*, 4 (2009), 42–47.

[32] RACHNA DHAMIJA, J.D.TYGAR, AND MARTI HEARST. Why phishing works. In *Proceedings of the CHI* (2006), pp. 581–590.

[33] Ross, B., Jackson, C., Miyake, N., Boneh, D., and Mitchell, J. C. Stronger password authentication using browser extensions. In *Proceedings of the USENIX Security Symposium* (2005), pp. 17–32.

[34] Schechter, S. E., Dhamija, R., Ozment, A., and Fischer, I. The emperor's new security indicators: An evaluation of website authentication and the effect of role playing on usability studies. In *Proceedings of the IEEE Symposium on Security and Privacy* (2007), pp. 51–65.

[35] Sheng, S., Magnien, B., Kumaraguru, P., Acquisti, A., Cranor, L. F., Hong, J., and Nunge, E. Anti-Phishing Phil: the design and evaluation of a game that teaches people not to fall for phish. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)* (2007), pp. 88–99.

[36] Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R. A., Kruegel, C., and Vigna, G. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the CCS* (2009), pp. 635–647.

[37] Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., and Cranor, L. F. Crying wolf: an empirical study of ssl warning effectiveness. In *Proceedings of the 18th conference on USENIX security symposium* (2009), pp. 399–416.

[38] Thorpe, J., and van Oorschot, P. Human-seeded attacks and exploiting hot-spots in graphical passwords. In *Proceedings of the USENIX Security Symposium* (2007), pp. 103–118.

[39] Whittaker, C., Ryner, B., and Nazif, M. Large-scale automatic classification of phishing pages. In *Proceedings of the NDSS* (2010).

[40] Wu, M., Miller, R. C., and Garfinkel, S. L. Do security toolbars actually prevent phishing attacks? In *Proceedings of the CHI* (2006), pp. 601–610.

[41] Wu, M., Miller, R. C., and Little, G. Web wallet: preventing phishing attacks by revealing user intentions. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)* (2006), pp. 102–113.

[42] Yan, J., Blackwell, A., Anderson, R., and Grant, A. Password memorability and security: Empirical results. *IEEE Security and Privacy 2*, 5 (2004), 25–31.

[43] Yee, K.-P., and Sitaker, K. Passpet: convenient password management and phishing protection. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)* (2006), pp. 32–43.

[44] Yue, C. Mitigating cross-site form history spamming attacks with domain-based ranking. In *Proceedings of the DIMVA* (2011), pp. 104–123.

[45] Yue, C., and Wang, H. Anti-Phishing in Offense and Defense. In *Proceedings of the ACSAC* (2008), pp. 345–354.

[46] Yue, C., and Wang, H. SessionMagnifier: A Simple Approach to Secure and Convenient Kiosk Browsing. In *Proceedings of the Ubicomp* (2009), pp. 125–134.

[47] Yue, C., and Wang, H. BogusBiter: A Transparent Protection Against Phishing Attacks. *ACM Transactions on Internet Technology (TOIT) 10*, 2 (2010), 1–31.

[48] Zhang, Y., Egelman, S., Cranor, L. F., and Hong, J. Phinding phish: Evaluating anti-phishing tools. In *Proceedings of the NDSS* (2007).

[49] Zhang, Y., Hong, J., and Cranor, L. CANTINA: A content-based approach to detecting phishing web sites. In *Proceedings of the WWW* (2007), pp. 639–648.

[50] 1Password. `http://agilebits.com/products/1Password`.

[51] Anti-Phishing Working Group. `http://www.antiphishing.org`.

[52] Client specification for the Google Safe Browsing v2.2 protocol. `http://code.google.com/p/google-safe-browsing/wiki/Protocolv2Spec`.

[53] Document Object Model (DOM). `http://www.w3.org/DOM/`.

[54] Firefox Phishing and Malware Protection. `http://www.mozilla.com/en-US/firefox/phishing-protection/`.

[55] Google Chrome and Browser Security. `http://www.google.com/chrome/intl/en/more/security.html`.

[56] Internet Explorer 8 Readiness Toolkit. `http://www.microsoft.com/windows/internet-explorer/readiness/new-features.aspx`.

[57] Internet Security Threat Report, Security research and analysis — Symantec. `http://www.symantec.com/business/theme.jsp?themeid=threatreport`.

[58] Likert scale. `http://en.wikipedia.org/wiki/Likert_scale`.

[59] NIST: Secure Hashing. `http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html`.

[60] Phishing Scam in an HTML Attachment. `http://labs.m86security.com/2011/03/phishing-scam-in-an-html-attachment/`.

[61] RoboForm Password Manager. `http://www.roboform.com/`.

[62] SQLite Home Page. `http://www.sqlite.org`.

[63] Windows Live ID. `http://msdn.microsoft.com/en-us/library/bb288408.aspx`.