

# iTrustPage: A User-Assisted Anti-Phishing Tool

Troy Ronda  
Dept. of Computer Science  
University of Toronto

Stefan Saroiu  
Dept. of Computer Science  
University of Toronto

Alec Wolman  
Microsoft Research

## ABSTRACT

Despite the many solutions proposed by industry and the research community to address phishing attacks, this problem continues to cause enormous damage. Because of our inability to deter phishing attacks, the research community needs to develop new approaches to anti-phishing solutions. Most of today's anti-phishing technologies focus on automatically detecting and preventing phishing attacks. While automation makes anti-phishing tools user-friendly, automation also makes them suffer from false positives, false negatives, and various practical hurdles. As a result, attackers often find simple ways to escape automatic detection.

This paper presents iTrustPage – an anti-phishing tool that does not rely completely on automation to detect phishing. Instead, iTrustPage relies on user input and external repositories of information to prevent users from filling out phishing Web forms. With iTrustPage, users help to decide whether or not a Web page is legitimate. Because iTrustPage is user-assisted, iTrustPage avoids the false positives and the false negatives associated with automatic phishing detection. We implemented iTrustPage as a downloadable extension to Firefox. After being featured on the Mozilla website for Firefox extensions, iTrustPage was downloaded by more than 5,000 users in a two week period. We present an analysis of our tool's effectiveness and ease of use based on our examination of usage logs collected from the 2,050 users who used iTrustPage for more than two weeks. Based on these logs, we find that iTrustPage disrupts users on fewer than 2% of the pages they visit, and the number of disruptions decreases over time.

## Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General—Security and protection

## General Terms

Security, Experimentation, Measurement

## Keywords

phishing, anti-phishing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EuroSys'08, April 1–4, 2008, Glasgow, Scotland, UK.  
Copyright 2008 ACM 978-1-60558-013-5/08/04 ...\$5.00.

## 1. INTRODUCTION

Today's anti-phishing tools have done little to stop the proliferation of phishing attacks. Relying on automatic ways to identify phishing attacks makes these tools suffer from both false positives and false negatives. For example, attackers can easily adapt to automatic spam and phishing email filters and find new ways to bypass them. Other approaches to defend against phishing, such as automatic password managers, face significant practical hurdles: they must allow users to temporarily deactivate them to handle backward compatibility issues or password resets. Many banks use visual cues such as customized login pages to prevent phishing, yet inattentive users can still be phished. While automation makes these anti-phishing tools more user-friendly, automation often introduces simple alternate ways for phishing attacks to continue to be viable in practice.

Despite the many automatic anti-phishing solutions, the phishing threat has continued to gain momentum. The following alarming trends indicate that our reliance on automatic tools to detect and prevent phishing attacks has had little effect. **Phishing toolkits have started to become available:** RSA recently uncovered a toolkit which displays the current version of a targeted Web page, yet copies any data entered to the phisher [6]. This "Universal Man-in-the-Middle Phishing Kit" sells for approximately \$1,000, and it allows phishers to get started by just entering a few parameters. **Phishing messages continue to be sent in great volume:** Symantec observed 166,248 unique phishing messages during the last six months of 2006; on average, this equates to 904 unique phishing messages per day [26]. Symantec's anti-phishing filters blocked an average of 8.48 million phishing messages per day over this time period. **The number of phishing Web sites continues to grow:** There were 27,221 new unique phishing Web sites reported in January, 2007 [1]; this is nearly three times the number of new unique phishing Web sites reported in January, 2006. Some in the banking industry claim that phishing is quickly becoming the single most important source of fraud they deal with [2]. **Phishing e-mails are becoming sophisticated and hard to filter:** Phishing messages have started to include personalized information, specific to the targeted user [17]. We believe that the research community must investigate new approaches for anti-phishing solutions to reverse the current trends.

In this paper, we present iTrustPage – a novel tool that does not completely rely on automation to detect phishing. Instead, iTrustPage relies on user input to decide on the legitimacy of a Web form. iTrustPage also uses external information repositories on the Internet to help the user with decision-making. A variety of sources of information on the Internet (e.g., a search engine such as Google) can be used to help establish the legitimacy of a particular Web page and/or Web form.

iTrustPage is a Web browser extension that prevents users from entering any information into suspicious Web forms. iTrustPage intercepts the user input and prompts the user for search terms that describe the Web page the user intended to visit. With these search terms, iTrustPage performs a Google search and validates the Web form only if it appears among the top search results<sup>1</sup>. If the suspicious Web form does not appear in the top search results, iTrustPage presents visual previews of those pages that do appear in the top search results, and asks the user whether any of them visually match the current form. If a match is found, the current form is likely to be a phishing attack. This step exemplifies one of the key benefits of user-assistance. The task of deciding whether the visual appearance of two Web pages is similar can be very hard for programs to get right, yet is a relatively easy task for a user. When the user indicates a visual match, iTrustPage redirects the user to the form selected from the search results. In this way, iTrustPage not only prevents users from filling out phishing forms, it also helps them locate the corresponding legitimate Web form.

In some cases, the search terms might not lead to any search results where the legitimate Web forms are visually similar to the current suspicious form. We believe that this type of limitation is fundamental to all anti-phishing tools – they all encounter ambiguity in certain cases when attempting to determine whether a site is legitimate. Unfortunately, there is no easy way to deal with this case. One conservative approach is to always block users from filling out such forms. This option might appeal to corporate system administrators. Another approach is to raise a warning before allowing the user to fill out the form. However, warnings have been shown to be ineffective: most users ignore them when they do not understand the implications [5, 27, 8]. While iTrustPage supports both options, the default option in our current implementation is to raise a warning.

iTrustPage uses two simple mechanisms to reduce how often users need to be involved in determining the legitimacy of a Web form. First, iTrustPage uses a whitelist of 467 domain names selected from the top 500 most popular Web sites as reported by Alexa.com (some of these Web sites belong to the same domain name). Second, iTrustPage uses a cache recording all the previously visited Web forms. In this way, users are never interrupted when filling out a form on a Web page whose domain appears in the whitelist or upon subsequent re-visits. Because these mechanisms introduce automation in our tool, we deliberately kept them simple and unsophisticated to minimize (and hopefully eliminate) the possibility of introducing false negatives when detecting whether a form is phishing. As our evaluation will show, we found that both the whitelist and the cache work surprisingly well in practice.

We implemented iTrustPage as a downloadable extension for the Firefox Web browser. iTrustPage is available for download from the Mozilla website for add-ons; once posted on this site, iTrustPage was downloaded by over 5,000 people in less than two weeks. We present an analysis of our tool’s effectiveness based on reported usability statistics collected from our deployed software. We show that iTrustPage disrupts users on less than 2% of the pages they visit and the number of disruptions decreases over time. We also find that iTrustPage resorts to its user-assisted validation scheme on 32.6% of the pages that users type into; the remaining pages are validated by iTrustPage’s whitelist and cache. Our current implementation collects these statistics and anonymizes them to protect our users identities; it also allows users to disable reporting entirely.

## 2. COMPARING AUTOMATION TO USER-ASSISTANCE FOR ANTI-PHISHING TOOLS

In this section, we illustrate the problems associated with automated anti-phishing tools with a case study: we investigate the effectiveness of the SpamAssassin e-mail filter. SpamAssassin is a very popular open-source spam and phishing e-mail filter. In our experiments, we use four different corpora of e-mail (including messages that are legitimate, phishing, and spam) collected from a public repository of e-mail suitable to use for testing in anti-spam and anti-phishing e-mail filters [21]. While e-mail filters are just one of the many tools in our anti-phishing arsenal, we believe that the problems they exhibit are representative of a wider range of issues associated with automated anti-phishing techniques. We then discuss why user-assisted tools have some inherent advantages over fully-automated tools. We delay our in-depth discussion of the previously proposed anti-phishing techniques and their benefits and drawbacks until Section 5.

### 2.1 Automation Introduces False Negatives and False Positives

To illustrate the presence of false negatives in automated phishing e-mail filters, we performed the following experiment. We ran a very recent version of SpamAssassin (version 3.1.8) over a relatively old corpus of 414 phishing e-mails collected between November 2004 and June 2005 [21]. We use the default configuration of SpamAssassin to identify phishing e-mails. Despite this corpus being more than two years old, SpamAssassin is still not very effective in classifying many of them as phishing. We find that 67 of the 414 phishing e-mail messages (16%) are labeled as legitimate by SpamAssassin.

We ran the same configuration of SpamAssassin over a corpus of 250 legitimate e-mails collected in 2004 from the same source [21]. Although each of these e-mail messages is legitimate, these messages are known to be hard to differentiate from spam and phishing e-mails because they contain unusual HTML markup, colored text, and phrases often found in illegitimate e-mail. Despite these e-mails being collected more than three years ago, SpamAssassin still labeled 8 of them (3.2%) as illegitimate.

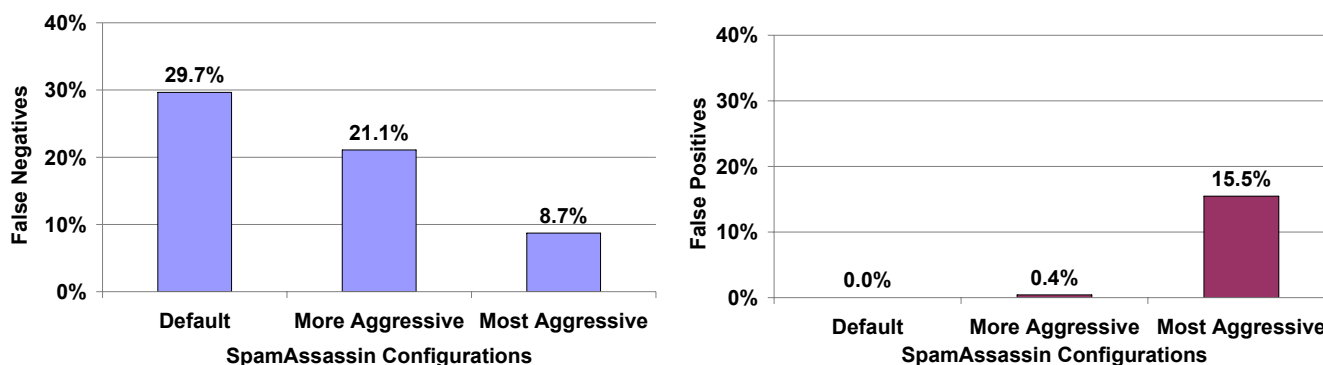
### 2.2 Coping With False Positives and False Negatives Simultaneously

When anti-phishing tools suffer from false negatives, they may not be effective in protecting their users. In such cases, they just decrease the likelihood that their users will become victims of phishing; however, users must remain vigilant and find other means to protect themselves from phishing attacks. On the other hand, false positives introduce serious usability concerns. For example, users are reluctant to continue using anti-phishing e-mail filters that classify even a small quantity of legitimate e-mail as phishing.

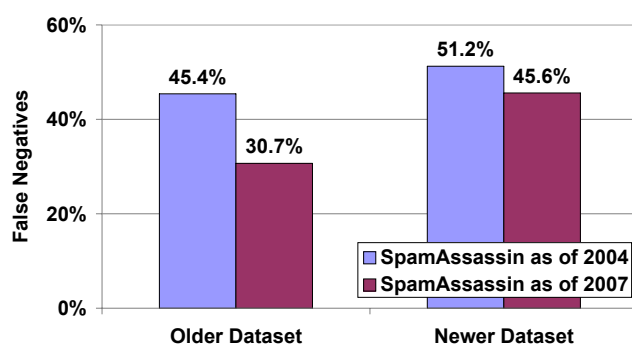
Unfortunately, it is very hard for an automated tool to eliminate *both* its false negatives and false positives. To reduce the rate of false negatives, the only option is to make the tool more aggressive. Unfortunately, becoming more aggressive usually leads to an increase in the rate of false positives. We illustrate this tension with the following experiment.

We ran SpamAssassin over two e-mail data sets: (1) a newer corpus of phishing e-mails from the same source as the one examined earlier [21] containing 1,423 e-mails dated between November 2005 and August 2006; and (2) a corpus of 478 legitimate e-mail messages sent by one of the authors over the past few months. We ran SpamAssassin with three different configurations: the default;

<sup>1</sup>Currently, iTrustPage uses the top 10 results returned by Google.



**Figure 1:** The trade-off between false positives and false negatives for SpamAssassin. We ran SpamAssassin with three configurations: the default one; a more aggressive configuration; and the most aggressive configuration. We increased SpamAssassin’s aggressiveness by decreasing its threshold for labeling an e-mail as phishing, from 5 to 3 to 1, respectively. On the left, we present the rate of false negatives in a corpus of 1,423 phishing e-mails collected between November 2005 and August 2006. On the right, we present the rate of false positives in a corpus of 478 legitimate e-mails sent by one of the authors over the past few months.



**Figure 2:** Phishing adapts to SpamAssassin. We ran SpamAssassin over two phishing data sets: an older data set collected between November 2004 and June 2005, and a newer data set collected between June 2005 and November 2005. We ran two versions of SpamAssassin – a version released in 2004 (version 3.0.1) and a very recent version released in 2007 (version 3.1.8). The rate of false negatives is higher in the newer phishing data set for both versions of SpamAssassin.

a more aggressive configuration; and the most aggressive configuration. We increased SpamAssassin’s aggressiveness by decreasing its threshold for labeling an e-mail as phishing. Figure 1 presents our results. Making SpamAssassin more aggressive has the expected effects: it reduces the rate of false negatives (as seen on the left) while simultaneously increasing the rate of false positives (as seen on the right).

### 2.3 Phishers Adapt to Automated Phishing Filters

Another problem with automation is that phishing attacks tend to evolve and adapt to the automated filters. We illustrate this problem with the following experiment. We ran SpamAssassin over the two phishing data sets used in the experiments above [21]: an older data set collected between November 2004 and June 2005, and a newer data set collected between June 2005 and November 2005. We ran two versions of SpamAssassin – an older version released in 2004 (version 3.0.1) and a recent version released in 2007 (version 3.1.8). We present the rate of false negatives in Figure 2.

For each of the two data sets, the most recent SpamAssassin has fewer false negatives. As expected, SpamAssassin is trying to adapt

to incoming phishing e-mail – its recent version is better than the older one. On the other hand, we also see that the rate of false negatives increases in the newer data sets for both versions of SpamAssassin. This illustrates that phishing attacks are adapting to bypass the SpamAssassin’s automated filters.

### 2.4 The Benefits of User-Assistance for Anti-Phishing Tools

We believe that tools (such as iTrustPage) that do not completely rely on automatic techniques for phishing detection have three key advantages over automated tools. First, we believe that false positives are rare and, if they do occur, they are less likely to irritate users. When a user is involved in the validation process, a false positive is a legitimate e-mail or Web page that the user helped to label as phishing. If they made a mistake, we believe that users are less likely to complain and stop using the tool. Because false positives are less of a concern, user-assisted anti-phishing tools can afford to make their automated detection very aggressive. If an e-mail (or a Web page) is deemed suspicious, such tools can then involve the users in a manual validation process.

Second, we believe it is more challenging for phishing attacks to adapt to user-assisted validation than to automated validation. Automatic phishing detection must examine human readable content and classify it as legitimate or suspicious. This is a very hard problem; as an example, there is debate in the machine learning community whether machine learning algorithms can be made secure against adversaries [4]. On the other hand, it is easier for people to interpret and understand human readable content. While still imperfect, people are much better at understanding the meaning of e-mails and Web pages than programs.

Third, user-assisted tools naturally support content in many languages whereas many automated anti-phishing tools have difficulty handling such content. For example, automated e-mail filters written to handle English e-mails cannot be directly ported to other languages. Instead, they must incorporate a whole new set of automatic rules and heuristics that correspond to the characteristics of phishing e-mails in those other languages.

## 3. THE DESIGN OF iTrustPage

In this section, we present the design and implementation of iTrustPage, our tool that prevents users from filling out phishing Web forms and then assists them in finding the corresponding legitimate Web form. When encountering a suspicious Web page,

iTrustPage does not simply warn the user – instead it offers them corrective action. The design of iTrustPage is based on three observations: (1) we can rely on users to assist with the process of deciding whether or not a Web page is legitimate, as there are certain tasks that are easy for people to do, yet are very difficult to automate reliably; (2) we can assist users by locating legitimate pages for them; and (3) we can use external information repositories, such as Internet search engine results, to assist with the process of deciding whether or not a given Web page is legitimate. User input is needed for two tasks: (1) describing search terms for the Web form they are filling in to see whether or not it is a well-known and established Web page; and (2) performing visual comparisons between the Web form they are filling in and the top Web forms related to their supplied search terms. The external information repositories used by iTrustPage are simply Google’s search index and the whitelist of known trustworthy web pages.

The remainder of this section presents a step-by-step description of how iTrustPage works, along with the intuition behind each step as well as presenting possible alternatives.

### 3.1 Automatic Classification

For first-time visits to a particular Web form, iTrustPage uses a built-in whitelist of popular legitimate sites. For subsequent re-visits, iTrustPage maintains a local cache of all previously validated Web forms visited by the user, as well as those forms manually approved by the user. In this way, iTrustPage *never* disrupts users when they visit a page on a popular, well-established domain or when they re-visit a page subsequently. Automatic validation makes iTrustPage easier to use: the tool remains transparent when these two heuristics determine that a form is legitimate.

### 3.2 Interactive Page Validation

When the automatic classification step fails to validate a particular Web form, iTrustPage involves the user in the validation process. Whenever users attempt to fill out an unvalidated Web form<sup>2</sup>, iTrustPage presents an overlay on the browser window that asks them to describe the form they *intend* to fill in, as if they are entering search terms for a search engine. iTrustPage uses the search terms to issue a search query using Google. If the top 10 search results contain the form’s Web site (i.e., its domain name), then iTrustPage infers that the Web site that serves this form is legitimate, and therefore the user is allowed to proceed and fill out the form. In some cases, the user reaches a suspicious Web form by navigating to it from the Google search page. For these forms, the user has already assisted iTrustPage, albeit unknowingly, to validate this page. Thus, iTrustPage does not ask the user to perform an additional Google search.

The fact that the site appears in the top 10 search results means that the Google crawler indexed the site, that many other sites link to it, and that the site is most likely not short-lived. Since the user selected the search terms that led to the search results, this means that the form presumably matches the user’s intent. Once a form is deemed legitimate, iTrustPage remembers that decision and will not intervene again. As future work, we could improve this mechanism by including the query results from other search engines, looking at other information repositories, such as the Whois domain registration database, and investigating whether “top 10” is the best number of search results to check (we used this number because it is the first page of results).

If the Web site that hosts the form does not appear in the top search results, iTrustPage then fetches the page contents of the top

<sup>2</sup>iTrustPage monitors keyboard and mouse input to detect when a user begins filling in a Web form.

three search results. iTrustPage presents the user with a visual preview of those pages and asks the user if any of the search result pages are visually similar to the Web form they intended to access. If the user detects a visual similarity, then the original form is probably a phishing form. Therefore, iTrustPage immediately redirects the user away from the original form to the legitimate form found in the search results. Figure 3 illustrates iTrustPage’s search interface overlay when visiting a questionable Web form (in this case <http://www.heinket.de>, a current phishing page spoofing PayPal). In fact, this Web form was phishing PayPal; once the user entered the search term “paypal”, iTrustPage previews the legitimate Web page, <http://www.paypal.com>.

### 3.3 Revising Search Terms

In some cases, the user will not find the intended Web site among the top 10 search results. When this happens, the user is asked to refine the search and the previous step is repeated. Nevertheless, it is possible (although not very common) that users may never find the desired site among the top search results. In this case, iTrustPage is unable to determine whether or not the form is legitimate.

While we make every effort to ensure that it only occurs rarely, the problem that iTrustPage faces, of being unable to determine whether or not a particular Web form is legitimate, is common to all detection tools, whether they try to detect phishing, spam e-mail, or malware. Sometimes there is simply not enough information to reliably make the correct decision. To address such situations, iTrustPage does allow the user to bypass the search mechanism when the user is unable to find a legitimate site using visual comparison.

### 3.4 Implementation Details

In this section, we present additional lower-level issues related to our tool’s implementation. iTrustPage is implemented as an extension to the Firefox Web browser. We have tested it on several commodity OSes, including Windows, Linux, and Mac OS X. Its source code is approximately 5,200 lines of code and it is freely available to download [3].

iTrustPage is configured to not block a user’s interaction with a Web form until they use the keyboard or bookmark the page. iTrustPage ignores common Firefox keyboard control characters such as the spacebar, the “Mac” key, and the control keys. iTrustPage can optionally be configured to block user interaction on page open; to only check pages with input boxes; and to only check when the user clicks on a form element. iTrustPage does not currently deal with embedded objects (e.g., Flash and ActiveX) because it does not receive their keyboard events. This is an implementation issue rather than a fundamental flaw in our design, and we intend to address this in a future version of iTrustPage.

### 3.5 Deployment

Over the past year, we have released several versions of iTrustPage. In the early releases, we incorporated several automatic heuristics to validate Web forms. The first heuristic was based on Google’s page rank. Google provides a Web service that takes as input a URL and returns the URL’s PageRank. The PageRank is a number between 0 (low) and 10 (high). Most sites have a PageRank of 0; only a few very popular sites have a PageRank higher than 8 (e.g., Google’s search page has a PageRank of 9). iTrustPage required a PageRank of at least 5 to automatically label a form as “validated”.

In addition to checking the PageRank of the actual form URL, iTrustPage also looked at the sequence of URLs accessed immediately preceding the form. If any previous URLs were served by the same site as the form, then the PageRank was calculated for each of



**Figure 3:** iTrustPage displaying its interface overlay. On the left, the user is visiting <http://www.heinket.de>, a well-known page phishing PayPal. Once the user enters their search terms (in this case “paypal”), iTrustPage previews the legitimate page, <http://www.paypal.com> (shown in the middle). On the right, iTrustPage informs the user about avoiding a suspicious page and being redirected to a legitimate one.

those previous URLs, and the maximum PageRank value was used to make the decision. The intuition behind this process was that often the URL of a form at a popular site has a very low PageRank, yet the site homepage that contains a link to that form has a very high PageRank.

To improve iTrustPage’s performance, once a user visited a page iTrustPage immediately prefetched the PageRank information, even if the user did not attempt to type anything on that page. In this way, iTrustPage could check whether the form had a high PageRank before the user started to fill it in. Without this prefetch, iTrustPage would have had to block the user from filling in the form briefly while retrieving its PageRank. We implemented this prefetching step asynchronously to avoid interfering with the user experience while loading the Web page.

However, after we examined the logs of our early releases, we decided to eliminate these automatic validation heuristics from iTrustPage for three reasons. First, we found that the automatic heuristics were not effective; Web forms with high page ranks were often found in iTrustPage’s whitelist or in its cache. Therefore these heuristics added little to iTrustPage’s overall effectiveness. Second, the automatic prefetch of the PageRank of every page accessed introduced serious privacy concerns. A small number of users pointed out that using iTrustPage meant transmitting to Google the URL of every single page browsed. Third, we worried about a “Google bomb” attack – influencing the ranking of a page by creating a large number of sites linking to it. By relying on an automatic test, iTrustPage could be subject to false negatives and thereby fail to protect its users against phishing attacks.

The heuristics based on PageRank would automatically validate any visited page with a high PageRank. To be successful, attackers would only need to setup their phishing pages on Web pages with high PageRank values. Instead, with the current version of iTrustPage a “Google bomb” attack can only be successful when the phishing page appears in the top 10 search results returned by Google based on the user’s search terms. For example, to phish PayPal attackers must make their phishing pages be one of the top 10 answers returned by Google for the search term PayPal. While not impossible, we believe this attack will be very difficult to mount in practice. In summary, by removing these automatic heuristics we did not significantly affect iTrustPage’s effectiveness, we improved iTrustPage’s users privacy, and we made “Google bomb” attacks much harder to mount in practice.



**Figure 4:** The number of daily installations of iTrustPage. While iTrustPage was released in early June, the FireFox website did not post a link to iTrustPage until June 27th. The number of fresh installs grew by two orders of magnitude the next day.

The most recent version of iTrustPage was released in early June 2007 and appeared on the FireFox website for third-party extensions on June 27th, 2007. All the results in this paper are based on logs received before August 9th 2007. During this period, we received logs from 5,184 users. Figure 4 shows the number of installations of iTrustPage since July 1st, 2007.

### 3.6 Circumventing iTrustPage

In this section, we describe how phishers might try to circumvent the detection algorithm implemented by iTrustPage. We anticipate three types of attacks.

One way to circumvent iTrustPage is to setup a phishing site in a domain listed in iTrustPage’s whitelist or in its cache. For example, an attacker can break into a popular Web site and replace one of their pages with a phishing form. While this attack is possible, most popular sites are typically well monitored, and such an attack would likely not go undetected for long. Another way to circumvent iTrustPage is using the “Google bomb” attack that we described earlier.

iTrustPage also relies on an implicit assumption: the user’s browser has not been compromised. If the browser is compromised then an attacker can easily disable iTrustPage’s functionality. From

# of Users with 2+ Weeks of Activity	2,050
Total # of Pages Browsed	796,038
Total # of Pages Typed Into	27,936
# of Hours Browsed in Total by All Users	42,928 (almost 5 years)

**Table 1: High-Level Statistics of Long-Lived Users.** *Our evaluation is based on users who used iTrustPage for at least two weeks.*

the beginning, we decided not to engineer against such attacks: if the browser is compromised, the user is subject to more harmful attacks, such as malware, viruses, spyware, or Trojan horses.

## 4. EVALUATION

The goal of this section is to evaluate how well iTrustPage works. For this, we attempt to answer the following six questions:

1. While most phishing attacks target passwords, divulging any kind of information to a malicious third party can raise serious privacy concerns. In addition to passwords, phishing attacks have been known to target social security numbers, addresses, birth dates, and other types of personal information. Thus, users risk becoming the victim of a phishing attack whenever they enter personal information into a Web page. *How often do users visit pages where they can input information?*

2. When filling out suspicious Web pages, iTrustPage interrupts users asking them to get involved in the validation process. If these disruptions occur too frequently, users might stop using iTrustPage. *How disruptive is iTrustPage to the users?*

3. iTrustPage uses a whitelist and a cache to minimize the number of disruptions; if a Web page appears in the whitelist or the cache, iTrustPage will not stop users from entering information into this Web page. *How effective are iTrustPage’s whitelist and cache?*

4. When a user starts to enter information on a Web page that does not appear in the whitelist or in the cache, iTrustPage deems the page suspicious. For suspicious Web pages, iTrustPage requires user assistance. Users must either validate this page by providing search terms to iTrustPage which then performs a Google search, or users choose to bypass iTrustPage’s validation process. There is one exception: if the way that the user arrived at the suspicious Web page in the first place was by performing a Google search, then the user has already provided assistance to iTrustPage and therefore iTrustPage does not ask the user to validate. *How often does iTrustPage validate pages?*

5. When validating the form with iTrustPage, users enter search terms describing the Web page they intended to type into. If the intended page is not found based on the search terms entered, users can continue to refine their searches until either the intended page is found or the users choose to bypass the validation process. *How many searches do users perform until they validate their pages?*

6. Ultimately, iTrustPage’s effectiveness is determined by whether it stops users from becoming victims of phishing attacks. *Did iTrustPage prevent any users from becoming phishing victims?*

We answer these questions by examining the logs returned by the users of iTrustPage. These users installed our tool by downloading it either from the FireFox website for third-party extensions or directly from our project Web site at the University of Toronto. After presenting a high-level characterization of these logs, we answer each of the six questions above in detail.

### 4.1 High-Level Characterization of iTrustPage’s Logs

Some of the installations of iTrustPage were short-lived – some users ended up uninstalling our tool after only a few hours or a few days. We removed these short-lived installations from our evaluation because we want to capture the tool’s effectiveness in steady-state, over the course of several weeks of browsing activity. In the remainder of this evaluation, all the results we present are based on the logs sent by users who used iTrustPage for at least two weeks after installation. Table 1 presents some high-level statistics about these long-lived installations.

### 4.2 How often do users visit Web pages which accept input?

Users enter information into Web pages by typing in HTML forms or in embedded scripts that display forms. While not all forms and embedded scripts are used for entering text in a Web page, measuring their presence on the Web can give us an upper bound on how often users visit Web pages which accept input. If few browsed pages include forms or embedded scripts, the risk of being phished is low. To determine the prevalence of forms and scripts on the Web, we examined how widespread the use of forms and embedded scripts is in the set of pages browsed by our users. We determine whether pages contain a form or embedded script by searching for the HTML “<form>” tag and the HTML “<script>” tag, respectively. In addition, we also record the presence of HTML input boxes of type “password”. Figure 5 presents our findings.

On the left, Figure 5 shows the fraction of all pages that include forms, password fields, and embedded scripts. While 7% of pages have password input boxes, almost two-thirds of all pages browsed contain forms. This suggests that users can enter information in many of the Web pages they visit. While many Web forms do not ask for personal information, forms have become highly prevalent on the Web. Embedded scripts are even more prevalent than forms, being found in over 80% of visited pages. These results suggest that being able to enter information in Web pages has now become the common case when users browse the Web.

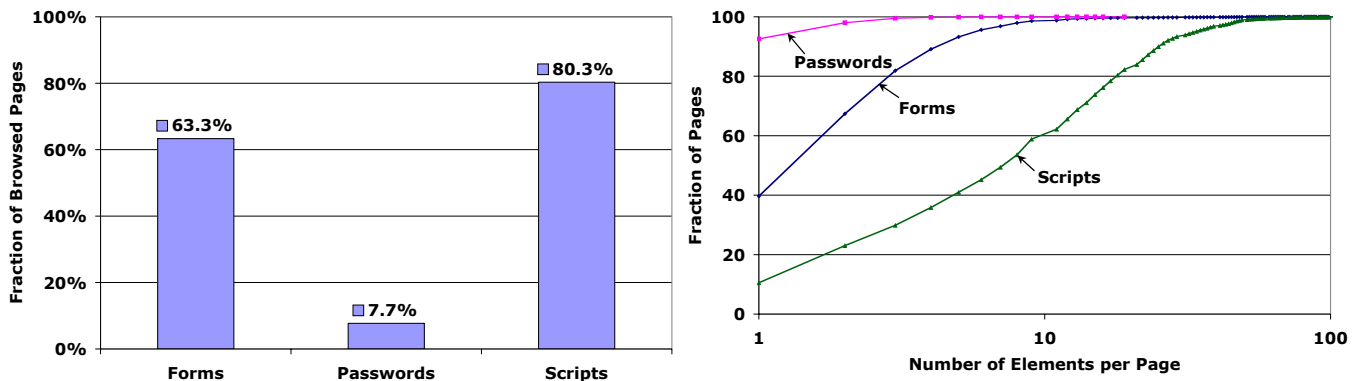
On the right, Figure 5 presents the cumulative distribution of the number of embedded scripts, forms, and passwords present in Web pages. For each distribution, we exclude the Web pages that do not contain any embedded scripts, forms, and passwords, respectively. Figure 5 shows that while most pages with password fields have only one such field (92.6%), 40% of all pages with embedded scripts include at least 10 such scripts. These results suggest that not only are forms and embedded scripts prevalent on the Web, but it is also common to have multiple forms and scripts present on a single Web page.

### 4.3 How disruptive is iTrustPage to the average user?

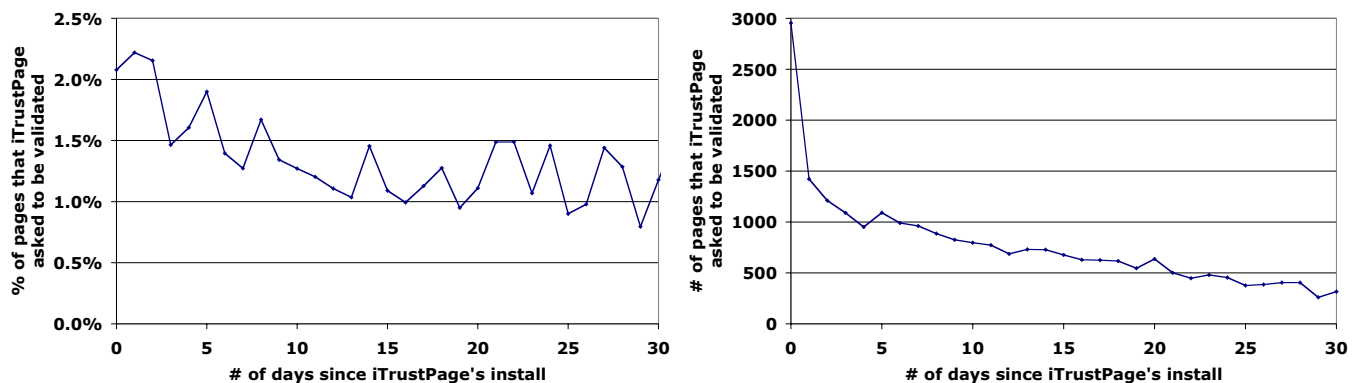
To validate suspicious Web forms, iTrustPage interrupts users and asks them for assistance in validating these Web pages. If these disruptions are too frequent, users might stop using iTrustPage. We believe that a high degree of user transparency is a key requirement of any Web security tool. iTrustPage must minimize the number of interruptions to its users.

Figure 6 illustrates how intrusive iTrustPage is. On the left, Figure 6 shows the fraction of Web pages that iTrustPage asked to be validated over time. On the right, Figure 6 presents the same data measured in the number of interruptions per day rather than the fraction of the total number of pages browsed. Our results indicate that iTrustPage is not excessively intrusive for its users. After one week of use, iTrustPage disrupts its users on less than 2% of all





**Figure 5:** The distribution of forms, passwords, and scripts in Web pages. (a) Fraction of pages browsed by iTrustPage’s users with form HTML tags, password HTML tags, and script HTML tags; (b) CDFs of the number of forms, passwords, and script tags in the browsed Web pages containing such tags. Note that in this graph we exclude the Web pages not containing any forms, passwords, or script tags, respectively.



**Figure 6:** How disruptive iTrustPage is. On the left, we present the ratio of the number of interruptions due to iTrustPage to the total number of pages visited each day. This graph presents the average of these ratios over time, relative to the number of days elapsed since the installation day. While iTrustPage stops users about 2% of the time in the first couple of days, after one week, the rate decreases to 1 to 1.5% of the time. On the right, we present the number of interruptions due to iTrustPage. The median number of interruptions per user is 0 every day, except than on the first day after the installation.

pages browsed. In fact, we found that fewer than half of iTrustPage’s users are disrupted daily (except during the day when users install their copy of iTrustPage).

#### 4.4 How effective are iTrustPage’s whitelist and cache?

iTrustPage uses a whitelist and a cache to minimize the number of disruptions. The whitelist contains 467 domain names selected from the top 500 most popular Web sites as reported by Alexa.com (some of these Web sites belong to the same domain name). Also, whenever a new Web page is validated or bypassed by a user, iTrustPage records this decision along with the page in the user cache. In this way, users are not interrupted upon re-visiting a Web page.

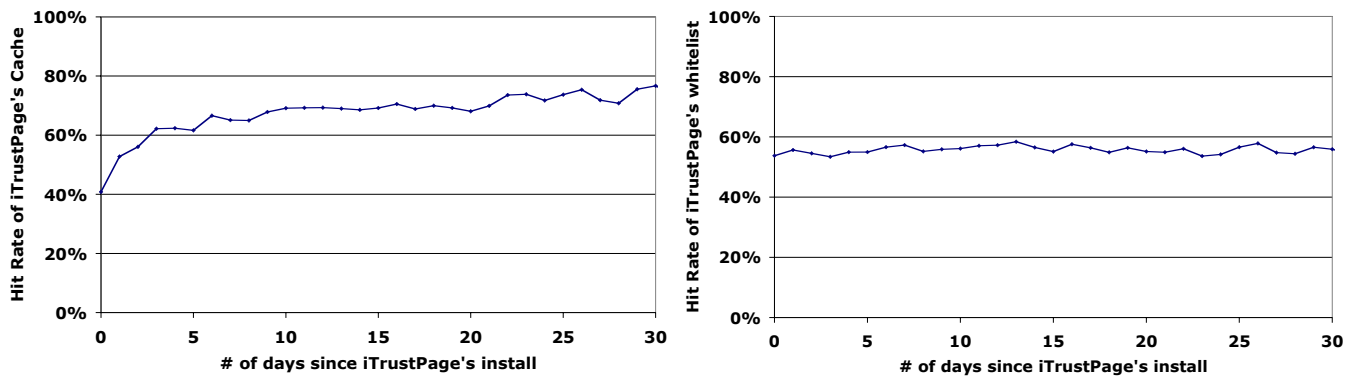
If these two techniques are effective, many of the pages users type into will appear either in the whitelist or in the cache. Figure 7 illustrates how effective the whitelist and the cache are by presenting their hit rates over time. On the left, we present the average hit rate of the caches of all users relative to the number of days since iTrustPage’s installation. While iTrustPage’s caches have an average of 40% hit rate on their first day, the hit rate grows to 65% after one week only. After only one week, two-thirds of the pages users type into appear in their caches (i.e., they were re-visited).

On the right, Figure 7 shows the hit rate of the whitelist for all users over time. The whitelist’s hit rate is analyzed independently of the cache’s hit rate. As expected, the hit rate of iTrustPage’s whitelist does not rise over time; instead, it remains flat at about 55%. Like the cache, the whitelist is also very effective – more than half the pages users type into appear in iTrustPage’s whitelist.

#### 4.5 How often does iTrustPage validate pages?

When a user types into a page, iTrustPage first checks whether the page’s domain appears in the whitelist. If not, iTrustPage next checks whether the page appears in iTrustPage’s cache. If the page appears neither in the whitelist nor in the cache, iTrustPage resorts on its user-assisted validation scheme. On the left, Figure 8 presents the breakdown of validating pages with iTrustPage. More than half of the pages that users type into (55.7%) appear in iTrustPage’s whitelist. Combining the whitelist and the cache, iTrustPage can validate 67.4% of pages typed into. Intuitively, this corresponds to the hit rate of a single cache servicing all iTrustPage’s users; this hit rate is different than the average hit rate of users’ caches and whitelists presented in Figure 7.

If a Web page does not appear either in the whitelist or in the cache, iTrustPage labels the page as suspicious. In this case, iTrust-



**Figure 7:** The hit rates of iTrustPage's cache and whitelist from a Google Search page. *The hit rates of each mechanism were analyzed in isolation from each other.*



**Figure 8:** Breakdown of validating the pages users type into. *On the left, we present how often pages are validated by the whitelist, the cache, and finally, by iTrustPage's user-assisted technique. On the right, we focus on the user-assisted validation only. With user-assistance, iTrustPage is successful at validating 66.4% of all suspicious pages. In the remaining cases, users choose to bypass the validation process.*

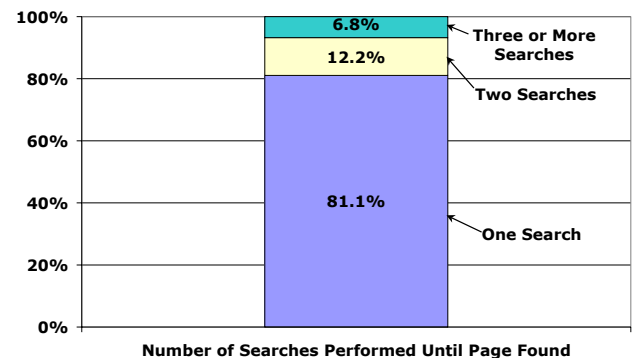
Page validates the page if the users can search and find the page on Google. If the page is not found, the user is presented with a set of alternate pages (i.e., the top three pages returned by Google) and the user is asked to choose a visually similar page.

On the right, Figure 8 presents the breakdown of how iTrustPage handles suspicious pages. Over 60% of suspicious pages are reached by navigating to them directly from a Google search; in these cases, iTrustPage validates the page without asking the user to perform an additional Google search. In almost 5% of cases, users validate their pages by searching for them and finding the answer in the top 10 results returned by Google. In a very small number of cases (0.2%), users choose to navigate to an alternate page instead. Finally, users bypass iTrustPage's validation process one-third of the time (33.6%).

While these results indicate that iTrustPage is successful at validating many of the suspicious Web forms that users type into, we also found that it is common for users to bypass iTrustPage's validation despite our efforts to make the tool unintrusive. One third of the time users avoid validating their Web pages, thereby running the risk of becoming the victim of a phishing attack.

#### 4.6 How many searches do users perform until they validate their page?

To validate their forms with iTrustPage, users must enter search terms describing the Web page they intended to type into. We examine how many times users revise their search terms to validate their Web forms. Figure 9 illustrates the number of attempts users



**Figure 9:** Number of times users must refine their search to validate a page. *Users can manually validate their page with only one search in over 81.1% of the cases.*

needed to revise their searches, for those searches that were eventually found in the top 10 search results. In 81.1% of the cases, the first search was successful, whereas 6.8% of the cases required at least three searches. These findings suggest that users can find their intended page with one search the vast majority of the time. Nevertheless, on occasion it is necessary to refine the search a few times in order to finish the validation process for some Web forms.



## 4.7 Did iTrustPage prevent any users from becoming phishing victims?

While measuring the iTrustPage’s non-intrusiveness and the effectiveness of assisting users with validating their forms is important, ultimately iTrustPage is useful only if it stops users from becoming victims of phishing attacks. Unfortunately, we cannot determine whether any browsed page is a phishing page without violating the privacy of iTrustPage’s users. Determining whether a page is phishing requires knowing what the page displays.

However, we can compute an upper bound on the number of prevented phishing attacks by investigating the cases when users chose to navigate to a page visually similar to the one they typed into. While we do not know whether all these cases correspond to prevented phishing attacks, in any phishing attack stopped by iTrustPage a user must choose to navigate to an alternate page.

We found 291 cases in our logs when a user chose a visually similar page. While 81 of these cases occurred within the first three days of installing iTrustPage (e.g., these might correspond to cases when users tested and explored iTrustPage’s functionality), 92 cases occur after the user ran iTrustPage for at least two weeks. We found HTML forms present in 240 of these pages.

## 4.8 Summary

Our evaluation shows that iTrustPage is effective and easy-to-use. Specifically, we found that:

- iTrustPage disrupts users on less than 2% of the pages they visit and the number of disruptions decreases over time.
- iTrustPage’s cache is effective – over two-thirds of the forms users type into appear in iTrustPage’s cache after one week of use.
- iTrustPage’s whitelist is effective – more than half of the forms users type into appear in iTrustPage’s whitelist.
- iTrustPage resorts to its user-assisted validation scheme on 32.6% of the pages users type into.
- iTrustPage requires user-assistance to validate suspicious Web forms two-thirds of the time; one-third of the time users choose to bypass iTrustPage’s validation.
- iTrustPage is easy to use: when searching to validate their Web forms, users can find their desired form in the top 10 search results 81.1% of the time.
- We cannot determine whether iTrustPage prevented any phishing attacks without violating the privacy of iTrustPage’s users. Nevertheless, we found 291 cases when users chose to navigate to a Web page other than the one they originally intended to visit. Of these cases, 92 occurred after the users were already familiar with running iTrustPage (i.e., at least two weeks after their installation date).

These findings show that iTrustPage’s approach to relying on user input and on external repositories of information shows promise. In the future, we intend to examine additional strategies to incorporate these insights into other Internet security tools.

## 5. RELATED WORK

In this section, we start by classifying current anti-phishing tools and techniques into four broad categories. After this categorization, we summarize the results of six previous studies on the behavior of Internet users when facing phishing Web pages, when using their

Web passwords, and when using different anti-phishing tools. We believe the results of these studies results portray the seriousness of the “phishing threat”.

### 5.1 Spam Filters and Blacklists

One approach relies on spam filters and blacklists to automatically prevent users from visiting a phishing Web page. Already there are many phishing-specific filters for popular email software (e.g., Exchange Server 2003 SP2 [19], Outlook [20], and SpamAssassin [25, 9]). Many Web browsers also incorporate blacklists to prevent users from visiting specific Web pages based on the domain or the IP addresses they are served from. Microsoft’s new IE7 browser, Mozilla’s FireFox 2, and Opera from version 9.1 all include lists of known phishing pages. If such a page is visited, the browser will either warn the user or block the page outright [12]. Very recently, a company has started to offer a special DNS service that filters out known phishing domains [16].

Some advantages of this general approach are its simplicity, transparency, and ease of deployment. Web browsers and spam filters are already highly popular. Automatically deploying filters and blacklists is easy to setup, and much of their functionality remains transparent to most users.

While effective, this class of solutions alone will not eliminate phishing attacks. Spam filters are not perfect. Phishing pages must be quickly discovered and added to blacklists, especially since the average uptime of a phishing page was only 4.5 days in August 2006 [1]. Also, the “freshness” of phishing pages can impact the effectiveness of some anti-phishing tools [30]. Studies have shown that many users ignore browser warnings when they do not understand their implications [27, 8, 5]. All these reasons lead us to believe that spam filters and blacklists will have only a marginal and temporary effect on the prevalence of phishing attacks.

### 5.2 New Web Authentication Tools

Another approach is to invent new Web authentication schemes that replace the current approach of users entering passwords directly into forms. Different techniques have been proposed to replace current authentication protocols between users and Web sites.

One such technique is out-of-band authentication, where users are asked to login through a different channel, more secure than the Web, such as a cell-phone [22] or a virtual machine [15]. Unfortunately, some of these techniques are subject to man-in-the-middle attacks [24]. Furthermore, out-of-band techniques can be logistically difficult to deploy.

Several research projects have proposed using password managers to protect users’ credentials [14, 23, 29]. Almost all these tools rely on a technique known as “password hashing” to ensure that one user never re-uses a password on more than one Web site. The idea is simple: these techniques use some site-specific information (for example, its SSL certificate or its domain name) combined with the user’s password as input to a secure hash function, whose output is forwarded to the server. In this way, the password manager ensures that even when phished, users do not divulge their passwords; instead, they divulge their passwords hashed with information specific to the phishing page. Phishers cannot reuse these passwords on the legitimate site.

The password hashing approach is both elegant and very effective. Unfortunately, password managers have not been quickly adopted by either users or Web sites. We believe several reasons are responsible for the moderate success of these tools. First, some tools have deployment issues. For example, when resetting a password, many Web sites today send an e-mail message containing a new, perhaps temporary, password. The user must enter this new

password to login, which requires disabling the password manager. The user then logs in with the temporary password, visits a form that allows users to change their password, and then re-enables the password manager to generate the permanent password. To handle this issue, many password managers use a special sequence of keys (e.g., typing the character '@' twice) or a toolbar button to activate or de-activate them [14, 23, 29]. De-activating a password manager is dangerous because the user becomes exposed to phishing attacks. Also, a recent study [5] found that many participants forget to activate (or re-activate) their password managers.

The same study [5] revealed a more subtle issue with password managers. The concept of password hashing is not easy to explain to many people, especially novice users. Some people's mental process of how online authentication works differs substantially from what password hashing does. As a result, users become frustrated, and they misunderstand when the tool is protecting them and when it is not. Ultimately, this leads to users still being exposed to phishing attacks. We believe that this is an important lesson: to be effective, a tool must be simple and intuitive, fitting most users' mental model of how Web browsing works.

### 5.3 New Web Interfaces

Another approach is to design Web interfaces that are less vulnerable to phishing. One such example is to require users to access important Web pages only through user-created labels [29]. In these cases, users have to go through an initial setup phase where they assign special labels to each of their important Web pages. From then on, as long as users visit their pages only by clicking on the appropriate label, they cannot be phished.

Another example allows users to create personalized visual clues and associate them with important Web pages. Many popular Web sites (e.g., Yahoo) have started to adopt this technique. Because a phishing page cannot know the correct visual clue, users can immediately detect when they are accessing a phishing page because their personalized clue is missing or incorrect.

The main disadvantage of both these approaches is that they place the burden on the user to notice the absence of personalized clues or to never forget to access the important page through their preset labels. These tools protect only the most diligent Web users, the ones who will always carefully check the authenticity of the Web forms they are about to fill in with their sensitive information. Unfortunately, studies [7, 27] have shown that many Internet users are not very careful when filling forms online.

Another approach relies on the content of a Web page to determine its legitimacy. CANTINA [31], an anti-phishing system, extracts keywords from a Web page, searches Google for these keywords, and determines whether the top Google search results contain the page's domain. The authors also evaluate additional heuristics combined with their keyword extraction technique. The pure keyword extraction correctly labels 97% of their phishing data set with 6% false positives; when combined with heuristics the system correctly labels 90% of their phishing dataset with 1% false positives. Unfortunately, this work doesn't address how to notify or block users from filling forms on suspicious Web pages. Whenever the false positive rate is not zero, it becomes problematic to simply notify users or block them from using a Web form. We believe this is a significant shortcoming in anti-phishing research and is an important reason for developing iTrustPage. Another problem with CANTINA is that an attacker can create a Web page that appears visually similar to an established page, while still misleading the tool into extracting incorrect search terms. For example, the phisher could hide the correct keywords as an image to protect them from being extracted. If the automated system extracts incor-

rect keywords, then the phishing web page may be automatically validated. Adding manual validation to an anti-phishing system is more robust – attackers cannot pick the keywords that users give to iTrustPage's manual validation task.

A different approach is to use automatic tools to fill in forms. Such tools remove the need for Internet users to type their credentials into online forms. These tools can then perform extra security checks to make sure that the form is legitimate. Web Wallet [28] is a tool that automatically fills in previously saved passwords. When the user is phished, Web Wallet detects that the current form has not been visited before. In this case, it presents the user with a list of previously filled-in forms; the user must review this list and either continue or choose one of the previous forms. If the user continues, Web Wallet issues a warning if the page has not been verified by TrustWatch [13], a site serving a list of verified Web forms.

On the surface, Web Wallet is similar to our tool: when filling out a new password field, the user is presented a list of previously filled-in forms (which are presumably legitimate). However, we believe iTrustPage is based on different insights than Web Wallet. Web Wallet relies on three mechanisms to prevent phishing: (1) automatic detection of password fields, (2) previously filled-in password fields, and (3) relying on users to notice and understand the description of a Web form (even when that description might not be available). We believe that systems with automatic detection of password fields can be "fooled" by clever Web forms (e.g., using Active-X controls or Flash scripts). Also, we think that a good security principle is to reduce the amount of confidential information stored and managed automatically by tools. Unfortunately, Web Wallet has to maintain a list of previously filled-in passwords.

Unlike the anti-phishing tools discussed above, iTrustPage is centered around three different observations. (1) users can describe the Web forms they are about to fill in; (2) these descriptions can be used to find more "established" forms on the Web; when such a form exists, the current form is most likely phishing; and (3) unlike many other solutions, iTrustPage does not simply warn the user when encountering a suspicious Web page – it offers them corrective action. iTrustPage's goal is to assist the user by taking them away from the phishing Web page and to help them find the corresponding legitimate page.

### 5.4 Centralized Approaches

A different approach uses a centralized server that tracks when users provide the same password to different sites [11]. The main observation behind the password-tracking approach is that when two different sites appear to have when users with the same password, it is likely that one site is phishing the other. Many issues have been raised regarding centralized approaches, including protecting users' privacy, filtering information introduced by phishers to poison the server's data, and whether the detection of phishing pages can be done sufficiently early to rescue any potential victims.

Another approach also uses the notion of visual similarity (i.e., page layouts and style) of Web pages [18]. Site owners submit their legitimate URLs and keywords to a central server. When the user receives an e-mail message with similar keywords, the system compares the visual similarity of the linked page in the e-mail message with the registered legitimate pages. Although this is an interesting approach, there are three key problems. First, it relies on site owners submitting their keywords and URLs, and phishers may attack this mechanism. Second, we believe it is possible for this type of heuristic to be misled by clever Web forms, which is why our tool relies on people to perform the visual comparisons. Third, the phisher may attempt to hide the relevant keywords in the phishing e-mail message.

## 5.5 User Studies Relevant to Phishing

In this section, we briefly summarize the results of six studies of how users behave when facing phishing pages, when using their Web passwords, and when using different anti-phishing tools. Overall, these studies paint a pessimistic picture of our progress against the “phishing threat”: it is very easy to mislead people into divulging their credentials with common phishing attacks.

Florêncio *et al.* [10] monitored the Web password habits of over half a million users over a period of three months. Based on these measurements, the authors extrapolate users’ Web password behaviors. We describe four of their findings relevant to the phishing problem. First, during a three week period, they observed passwords being typed into verified phishing sites 101 times. These are passwords which were previously used on other Web sites. Second, users type their passwords often. The authors estimate that users type passwords an average of eight times per day. Third, users have many Web accounts. On average, each user has 25 accounts requiring passwords. In addition, an average user has 6.5 passwords. Each password is shared between 3.9 different sites. Finally, users forget their passwords. The authors estimate that 4.3% of Yahoo users forgot their passwords during the three month study. They also found that Yahoo users changed their password 15 times out of every 100 logins.

Dhamija *et al.* [7] asked 22 participants to decide whether or not 20 Web pages are legitimate. They found that the participants made mistakes 40% of the time. Even worse, the best phishing Web page fooled 90% of the participants. Warnings are ineffective: 68% of the participants “proceeded without hesitation when presented with [certificate] warnings”.

Wu *et al.* [27] explored how well anti-phishing toolbars stop people from using a fraudulent Web page. They found that active warnings (e.g., pop-up warnings) are more effective than passive security cues. Even active warnings failed to prevent some of the participants from falling victim to the phishing pages. Moreover, some participants thought the warning given by the toolbar was invalid.

Downs *et al.* [8] performed a preliminary interview study including 20 participants with no computer security experience. They found that many users cannot distinguish legitimate e-mail from phishing e-mail. Many participants miss cues in the address bar, and they do not interpret pop-up messages in meaningful ways. They also found that security tools need to recommend a course of action instead of merely giving warnings.

Jagatic *et al.* [17] performed an actual phishing attack against 581 students at Indiana University in April 2005. The experiment used publicly available information to personalize their phishing e-mails. 72% of targeted students gave away their username and password when the e-mail message appeared to be from a friend. When the e-mail is sent from a fictitious person, the successful phishing rate drops to 16%. In the first 12 hours of their experiment, 70% of the total phishing responses occurred. Their findings illustrate that more sophisticated phishing attacks, such as sending personalized phishing e-mails, have very high rates of success.

Chiasson *et al.* [5] explored the usability of two password managers with 26 participants, as mentioned earlier. Their participants had difficulty building a mental model of the software – they do not have an understanding, even at a high-level, of what the software is doing. This led to frustration and misconception leading to dangerous security exposures. They also found that participants tend to dismiss alerts when the warning message is unclear.

## 5.6 Summary: Lessons Learned

In this section, we summarize the lessons learned from the above related work. In spite of all these anti-phishing efforts, the phishing threat has been gaining momentum as we discussed in Section 1. We believe that these lessons help us better identify different strategies for dealing with phishing.

- To be effective, an anti-phishing tool must be intuitive and simple-to-use. It can rely on users only to perform very simple tasks they normally perform when browsing the Web.
- Relying on users to be diligent and check for signs of suspect e-mails or Web pages can be only marginally successful.
- More sophisticated phishing attacks, such as the ones sending personalized e-mails, have high rates of success. It is difficult for spam filters to identify and eliminate personalized phishing e-mails.
- Many phishing pages are short-lived. To be effective, techniques based on detecting these pages and blocking users from visiting them must act quickly.

## 6. CONCLUSIONS

This paper presents iTrustPage, a tool for preventing users from filling out Web phishing forms. iTrustPage relies on two key observations: (1) user input can be used to disambiguate between legitimate and phishing sites, as long as the interaction with the user is simple and intuitive; and (2) Internet repositories of information can be used to assist the user with the decision making process.

Our results show that user-assisted anti-phishing strategies show promise while avoiding the problems associated with automatic tools. We believe that iTrustPage is just one example of a different approach to Internet security – having the user assist the security tool and helping it provide better protection. In the future, we intend to examine additional strategies to incorporate this insight into other Internet security tools.

## 7. ACKNOWLEDGMENTS

We would like to thank John Aycock, Fabian Monrose, Ratul Mahajan, Niels Provos, Anil Somayaji, Paul Van Oorschot and the anonymous reviewers for their helpful comments on earlier drafts of this paper.

## 8. REFERENCES

- [1] Anti-Phishing Working Group Website. <http://www.antiphishing.org/>.
- [2] Personal Communication, 2006. Confidential Source, Canadian Banking Sector. Toronto.
- [3] iTrustPage Tool, 2007. <http://www.cs.toronto.edu/~ronda/itrustpage/>.
- [4] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can Machine Learning Be Secure? In *Proceedings of the ACM Symposium on Information, Computer, and Communication Security (ASIACCS)*, Taipei, Taiwan, March 2006.
- [5] S. Chiasson and P. van Oorschot. A Usability Study and Critique of Two Password Managers. In *Proceedings of the USENIX Security Symposium*, August, 2006.
- [6] CNET News.com. New tool enables sophisticated phishing scams. [http://news.com.com/New+tool+enables+sophisticated+phishing+scams/2100-1029\\_3-6149090.html](http://news.com.com/New+tool+enables+sophisticated+phishing+scams/2100-1029_3-6149090.html).

- [7] R. Dhamija, J. D. Tygar, and M. Hearst. Why Phishing Works. In *Proceedings of Conference on Human Factors in Computing Systems (CHI)*, April 2006.
- [8] J. S. Downs, M. B. Holbrook, and L. F. Cranor. Decision strategies and susceptibility to phishing. In *Proceedings of the Symposium on Usable Privacy and Security*, July 2006.
- [9] I. Fette, N. Sadeh, and A. Tomasic. Learning to Detect Phishing Emails. In *Proceedings of the International World Wide Web Conference (WWW)*, Banff, Alberta, Canada, May 2007.
- [10] D. Florêncio and C. Herley. A Large-Scale Study of Web Password Habits. In *Proceedings of the International World Wide Web Conference (WWW)*, Banff, Alberta, Canada, May 2007.
- [11] D. Florêncio and C. Herley. Password Rescue: A New Approach to Phishing Prevention. In *Proceedings of USENIX Workshop on Hot Topics in Security*, July 2006.
- [12] R. Franco. Better Website Identification and Extended Validation Certificates in IE7 and Other Browsers, 2005. <http://blogs.msdn.com/ie/archive/2005/11/21/495507.aspx>.
- [13] GeoTrust. TrustWatch Search, 2006. <http://www.trustwatch.com/>.
- [14] J. Halderman, B. Waters, and E. Felten. A convenient method for securely managing passwords. In *Proceedings of the International Conference on World Wide Web*, May, 2005.
- [15] C. Jackson, D. Boneh, and J. C. Mitchell. Stronger Password Authentication Using Virtual Machines. 2006. <http://crypto.stanford.edu/SpyBlock/spyblock.pdf>.
- [16] K. Jackson. DNS Gets Anti-Phishing Hook, 2006. [http://www.darkreading.com/document.asp?doc\\_id=99089&WT.svl=news1\\_1](http://www.darkreading.com/document.asp?doc_id=99089&WT.svl=news1_1).
- [17] T. Jagatic, N. Johnson, M. Jakobsson, and F. Menczer. Social Phishing. *Communications of the ACM*. Vol. 50, No. 10., October, 2007.
- [18] W. Liu, X. Deng, G. Huang, and A. Fu. An Antiphishing Strategy Based on Visual Similarity Assessment. *IEEE Internet Computing*, Vol. 10, No.2. 58-65, March/April, 2005.
- [19] Microsoft. Exchange Server, 2006. <http://www.microsoft.com/exchange/default.mspx>.
- [20] Microsoft.com. Get anti-phishing and spam filters with Outlook SP2, 2005. [http://www.microsoft.com/athome/security/email/outlook\\_sp2\\_filters.mspx](http://www.microsoft.com/athome/security/email/outlook_sp2_filters.mspx).
- [21] J. Nazario. Phishingcorpus: phishing2. <http://monkey.org/~jose/phishing/phishing2.mbox>.
- [22] B. Parno, C. Kuo, and A. Perrig. Phoolproof Phishing Prevention. In *Proceedings of Financial Cryptography and Data Security (FC)*, 2006.
- [23] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. Mitchell. Stronger Password Authentication Using Browser Extensions. In *Proceedings of the Usenix Security Symposium*, April, 2005.
- [24] B. Schneier. Two-Factor Authentication: Too Little, Too Late. *Communications of the ACM*. Vol. 48, No. 4., April, 2005.
- [25] SURBL. Surbl lists, 2006. <http://www.surbl.org/lists.html>.
- [26] Symantec. Symantec Internet Security Threat Report: Trends for July - December 06. [http://eval.symantec.com/mktginfo/enterprise/white\\_papers/ent-whitepaper\\_internet\\_security\\_threat\\_report\\_xi\\_03\\_2007.en-us.pdf](http://eval.symantec.com/mktginfo/enterprise/white_papers/ent-whitepaper_internet_security_threat_report_xi_03_2007.en-us.pdf).
- [27] M. Wu, R. Miller, and S. Garfinkel. Do Security Toolbars Actually Prevent Phishing Attacks? In *Proceedings of Conference on Human Factors in Computing Systems (CHI)*, April 2006.
- [28] M. Wu, R. Miller, and G. Little. Web Wallet: Preventing Phishing Attacks by Revealing User Intentions. In *Proceedings of the Symposium on Usable Privacy and Security*, July 2006.
- [29] K. Yee and K. Sitaker. Passpet: convenient password management and phishing protection. In *Proceedings of the Symposium on Usable Privacy and Security*, July 2006.
- [30] Y. Zhang, S. Egelman, L. Cranor, and J. Hong. Phinding Phish: An Evaluation of Anti-Phishing Toolbars. In *Network and Distributed System Security Symposium (NDSS)*, San Diego, California, USA, February 2007.
- [31] Y. Zhang, J. Hong, and L. Cranor. CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. In *Proceedings of the International World Wide Web Conference (WWW)*, Banff, Alberta, Canada, May 2007.