BogusBiter: A Transparent Protection Against Phishing Attacks

6

CHUAN YUE and HAINING WANG
College of William and Mary

Many anti-phishing mechanisms currently focus on helping users verify whether a Web site is genuine. However, usability studies have demonstrated that prevention-based approaches alone fail to effectively suppress phishing attacks and protect Internet users from revealing their credentials to phishing sites. In this paper, instead of preventing human users from "biting the bait," we propose a new approach to protect against phishing attacks with "bogus bites." We develop *BogusBiter*, a unique client-side anti-phishing tool, which transparently feeds a relatively large number of bogus credentials into a suspected phishing site. BogusBiter conceals a victim's real credential among bogus credentials, and moreover, it enables a legitimate Web site to identify stolen credentials in a timely manner. Leveraging the power of client-side automatic phishing detection techniques, BogusBiter is complementary to existing preventive anti-phishing approaches. We implemented BogusBiter as an extension to the Firefox 2 Web browser, and evaluated its efficacy through real experiments on both phishing and legitimate Web sites. Our experimental results indicate that it is promising to use BogusBiter to transparently protect against phishing attacks.

Categories and Subject Descriptors: H.4.3 [**Information Systems Applications**]: Communications Applications—*Information browsers*; K.4.4 [**Computers and Society**]: Electronic Commerce—*Security*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access*

General Terms: Design, Experimentation, Human Factors

Additional Key Words and Phrases: Phishing, web spoofing, credential theft, security, usability

ACM Reference Format:

Yue, C. and Wang, H. 2010. BogusBiter: A transparent protection against phishing attacks. ACM Trans. Internet Technol. 10, 2, Article 6 (May 2010), 31 pages.
DOI = 10.1145/1754393.1754395 http://doi.acm.org/10.1145/1754393.1754395

This work was partially supported by NSF grants ECCS-0901537 and CNS-0916022.
A short version of this article has appeared in *Proceedings of the Annual Computer Security Applications Conference* [Yue and Wang 2008].
Authors' address: C. Yue and H. Wang, Department of Computer Science, College of William and Mary, Williamsburg, VA 23187; email: {cyue,hnw}@cs.wm.edu.

DOI 10.1145/1754393.1754395 http://doi.acm.org/10.1145/1754393.1754395

1. INTRODUCTION

A phishing attack is typically carried out using an email or an instant message, in an attempt to lure recipients to a fake Web site to disclose personal credentials. Phishing attacks have seriously afflicted Internet users and financial institutions with identity thefts and brand reputation damage. According to recent Anti-Phishing Working Group (APWG) reports [APWG 2008] and Gartner surveys [GartnerSurvey 2006], the number of phishing sites, the number of phishing victims, and the amount of financial losses stemming from phishing attacks have all increased over the past few years.

To defend against phishing attacks, a number of countermeasures have been proposed and developed. Server-side defenses employ SSL certificates, user-selected site images, and other security indicators to help users verify the legitimacy of Web sites. Client-side defenses equip Web browsers with automatic phishing detection features or add-ons to warn users away from suspected phishing sites. However, recent usability studies have demonstrated that neither server-side security indicators nor client-side toolbars and warnings are successful in preventing vulnerable users from being deceived [Downs et al. 2006; Dhamija et al. 2006; Schechter et al. 2007; Whalen and Inkpen 2005; Wu et al. 2006a; Egelman et al. 2008]. This is mainly because (1) phishers can convincingly imitate the appearance of legitimate Web sites, (2) users tend to ignore security indicators or warnings, and (3) users do not necessarily interpret security cues appropriately. Educational defenses teach users to understand and avoid phishing attacks [Jagatic et al. 2007; Kumaraguru et al. 2007; Sheng et al. 2007]. But they cannot completely foil phishing attacks. Takedown defenses exploit spams and suspicious URLs to discover and shut down newly emerged phishing sites. However, the efficacy of this approach is limited, due to the ease of setting up and the short online time of phishing sites, as well as the application of takedown evasion methods by phishers [Jakobsson and Myers 2006; Moore and Clayton 2007; APWG 2008; KYE-Phishing 2008].

These different approaches are all preventive by nature. They endeavor to prevent users from being tricked into revealing their credentials to phishing sites. Nevertheless, these prevention-based approaches alone are insufficient to shield vulnerable users from "biting the bait" and defeat phishers, as human users are the weakest link in the security chain. The ever-increasing prevalence and severity of phishing attacks clearly indicate that anti-phishing is still a daunting challenge.

In response to this challenge, we have made two observations with respect to the acquisition of credentials by phishers and the automatic detection of phishing attacks on Web browsers. First, currently the majority of those who have "bitten the bait" and fallen victim to phishing attacks are real victims; thus, it is trivial for a phisher to verify the acquired credentials and trade them for money. However, if we can supply phishing sites with a large number of bogus credentials, we might be able to hide victims' real credentials among bogus credentials and make it harder for phishers to succeed.

Second, although remarkable advances in client-side automatic phishing detection have empowered Web browsers to identify the majority of phishing

sites [Chou et al. 2004; Garera et al. 2007; Ludl et al. 2007; Zhang et al. 2007b; FirefoxPhishingProtection 2008; MicrosoftPhishingFilter 2008], the possible false positives (legitimate Web sites misclassified as phishing sites) make it hard for Web browsers to directly block users' connections to suspected phishing sites. Thus, issuing warnings and expecting users to leave a suspected phishing site have become the most common actions employed by modern Web browsers. However, instead of just wishing vulnerable users could make correct decisions, if we can effectively transform the power of automatic phishing detection into the power of automatic fraud protection, we will take a big step forward towards winning the battle against phishing.

In this paper, we propose a new approach to protect against phishing attacks with "bogus bites" on the basis of the two observations we have mentioned. The key feature of this approach is to transparently feed a relatively large number of bogus credentials into a suspected phishing site, rather than attempt to prevent vulnerable users from "biting the bait." These "bogus bites" conceal victims' real credentials among bogus credentials, and enable legitimate Web sites to identify stolen credentials in a timely manner. Based on the concept of "bogus bites," we design and develop *BogusBiter*, a unique client-side anti-phishing tool that is complementary to existing prevention-based mechanisms. Seamlessly integrated with the phishing detection and warning mechanisms in modern Web browsers, BogusBiter is transparent to users.

At a user's Web browser, BogusBiter is turned on once a login Web page is classified as a phishing page by a Web browser's built-in phishing detection component or a third-party detection toolbar. For a victim who is beguiled into divulging a real credential, BogusBiter hides the real credential among a set of automatically generated bogus credentials, and then submits these credentials one by one to the phishing site. For a security-conscious user who does not reveal a real credential, BogusBiter also generates a set of bogus credentials, and then submits them to the phishing site in the same way as it does for a victim.

At the phishing site, a phisher will thus receive a much larger number of credentials than before, but the overwhelming majority are bogus credentials fed by BogusBiter. Elaborating bogus credential generation and submission mechanisms, BogusBiter makes it difficult for a phisher to distinguish who are real victims and which are real credentials. The only effective way for a phisher to sift out bogus credentials is to visit the legitimate Web site and verify whatever credentials have been collected from the phishing site.

At the legitimate Web site, if the phisher assumes the burden of verifying all the collected credentials to single out the real credentials, the unique design of bogus credential generation will enable the legitimate site to identify victims' stolen credentials in a timely manner and make it harder for a phisher to succeed. In other words, the bogus credential filtering process becomes the trigger for detecting stolen credentials at the legitimate Web site, and hence, ironically, the phisher's attempt to bypass BogusBiter helps us to achieve automatic fraud protection.

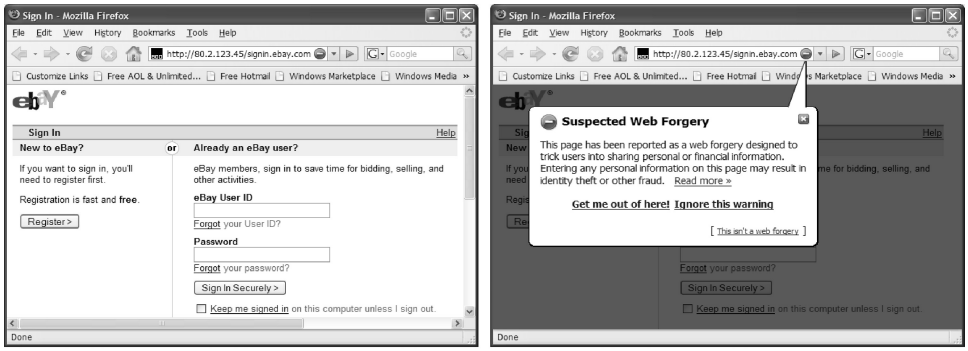

Fig. 1. (a) A phishing site designed to attack eBay users, (b) Firefox 2 phishing warning mechanism.

While leveraging the power of widely used client-side automatic phishing detection techniques, BogusBiter is not bound to any specific phishing detection scheme. Thus, BogusBiter can utilize the latest advances in phishing detection techniques such as blacklists and heuristics to protect against a wide range of phishing attacks. Moreover, BogusBiter is incrementally deployable over the Internet, and the fraud protection enabled at a legitimate Web site is independent of the deployment scale of BogusBiter. We implemented BogusBiter as a Firefox Web browser extension and evaluated its efficacy through real experiments over both phishing and legitimate Web sites. Our experimental results indicate that it is promising to use BogusBiter to transparently protect against phishing attacks.

The remainder of this article is structured as follows. Section 2 introduces the background of phishing attacks and the automatic phishing detection and warning mechanisms in modern Web browsers. Section 3 details the design of BogusBiter. Section 4 describes the implementation of BogusBiter. Section 5 evaluates the capability and performance of BogusBiter. Section 6 discusses the deployment of BogusBiter and potential evasions against BogusBiter. Section 7 surveys the related client-side anti-phishing research work, and finally, Section 8 concludes the article.

2. BACKGROUND

Figure 1(a) illustrates a phishing site designed to attack eBay users. In a typical scenario, a user receives a spoofed email that appears to be sent from the real eBay, luring the user to log into the phishing site. Once the user believes this site is the genuine eBay Web site and logs in, the user's username/password credential is stolen. Passwords have increasingly been targeted by harvesting attacks, as they protect online accounts with valuable assets [Florêncio and Herley 2007]. While some phishing attacks may steal other types of credentials such as credit card numbers and social security numbers, the most common type of phishing attack attempts to steal account numbers and passwords used for online banking [Jakobsson and Young 2005]. Therefore, protecting a

user's username/password credential is the primary focus of many client-side anti-phishing research work such as SpoofGuard [Chou et al. 2004], Dynamic Security Skins [Dhamija and Tygar 2005], AntiPhish [Kirda and Kruegel 2005], PwdHash [Ross et al. 2005], Web Wallet [Wu et al. 2006b], and Passpet [Yee and Sitaker 2006]. Our work also focuses on protecting a user's username/password credential. In the remainder of this article, we use the terms credential and username/password pair interchangeably.

The potential threat of phishing or Web spoofing attacks was first uncovered by Felten et al. [1997]. Today, phishing is not merely about Web site forgery and email spoofing, it has become a carefully planned and well structured multiphase effort to steal money. The life cycle of a phishing attack consists of six phases: *planning*, *setup*, *attack*, *collection*, *fraud* & *abuse*, and *post attack*, as defined by the Financial Services Technology Consortium [FSTC-Phishing 2005]. Various techniques can be applied in these six phases to combat phishing. However, since phishing sites can be easily set up and money laundering is still a difficult problem to curtail, current research and industry efforts focus mainly on the *attack* phase, with the objective of preventing users from submitting their credentials to phishing sites.

In contrast, the BogusBiter's protection against phishing attacks ranges from the *attack* phase to the *collection* phase, and then to the *fraud* & *abuse* phase, covering the flow of credentials. Specifically, BogusBiter retaliates against phishers with a large number of bogus credentials in the *attack* phase, makes it hard for them to identify real credentials in the *collection* phase, and detects their fraudulent activities in the *fraud* & *abuse* phase.

While distinct from preventive anti-phishing mechanisms, BogusBiter complements them in a natural way. In particular, BogusBiter leverages the power of client-side automatic phishing detection mechanisms and takes advantage of the state-of-practice phishing warning mechanisms in popular Web browsers to transparently protect vulnerable users.

Among automatic phishing detection mechanisms, two commonly used techniques are blacklists and heuristics. Blacklist-based techniques generate close-to-zero false positives and can detect most phishing attacks [Ludl et al. 2007; Zhang et al. 2007a; FirefoxPhishingTest 2006; Robichaux and Ganger 2006]. For example, Ludl et al. [2007] demonstrated that blacklists provided by Google (used by Firefox 2) can recognize almost 90% of live phishing sites. However, because some phishing sites may not be added into blacklists and the so-called *zero-day* attacks may occur, researchers have proposed various heuristic-based techniques to identify phishing sites in real time [Chou et al. 2004; Garera et al. 2007; Ludl et al. 2007; Zhang et al. 2007b]. These heuristic-based techniques have obtained very encouraging results. For example, CANTINA, a content-based detection tool proposed by Zhang et al. [2007b] can identify 90% of phishing pages with only 1% false positives. A URL-based classifier proposed by Garera et al. [2007] is another tool that can catch 95.8% of phishing pages with only 1.2% false positives.

Currently, Firefox 2 primarily employs blacklist-based techniques while Internet Explorer (IE) 7 uses both kinds of techniques [FirefoxPhishingProtection 2008; MicrosoftPhishingFilter 2008]. Because BogusBiter's design is

independent of any specific detection scheme, it can leverage advances in both blacklist-based techniques and heuristic-based techniques to combat the majority of phishing attacks.

Regarding phishing site warning mechanisms, the state of practice is to make it mandatory for a user to respond to the active warning of a suspected phishing site. Figure 1(b) illustrates the warning given by Firefox 2 [FirefoxPhishingProtection 2008] after correctly identifying the example Web site in Figure 1(a) as a phishing site. A user is unable to enter the username and password without first interacting with the warning page. If the user clicks the "Get me out of here!" link, the user is redirected to a default page and is protected. Otherwise, if the user clicks the "Ignore this warning" link, the warning page disappears and the user is exposed to the risk of credential theft. A similar warning mechanism is also used in IE 7 [MicrosoftPhishingFilter 2008].

Both Firefox 2 and IE 7 might choose such a active warning mechanism because: (1) issuing warnings simply through browser-based security indicators such as the address bar, the status bar, and various toolbars is ineffective [Downs et al. 2006; Dhamija et al. 2006; Schechter et al. 2007; Whalen and Inkpen 2005; Wu et al. 2006a; Egelman et al. 2008], and (2) directly blocking users' connections to suspected phishing sites is unacceptable, due to inevitable false positives. Although using an active warning page represents current best practice, a recent usability study conducted by Egelman et al. [2008] demonstrates that overall about 21% of participants still ignore the IE 7 and Firefox 2 active phishing warnings and fall for phishing attacks. Therefore, a crucial usability gap still exists in today's anti-phishing ecosystem, and many users who are most vulnerable to phishing still cannot be protected.

Phishing attacks are very insidious, and so far there is no single silver bullet for completely defeating phishers. Comprehensive, multifaceted, and integrated approaches are clearly needed in the anti-phishing ecosystem. BogusBiter fits into such an anti-phishing ecosystem especially by aiming to fill the aforementioned usability gap. Properly leveraging existing anti-phishing efforts, BogusBiter is capable of providing a transparent protection to those most vulnerable users.

3. DESIGN

In this section, we first give an overview on the design of BogusBiter, including the basic working mechanism, the main design assumption, and the two key design objectives. We then detail the offensive line and defensive line of BogusBiter.

3.1 Design Overview

BogusBiter is designed as either a new component or an extension to popular Web browsers such as Firefox 2 or IE 7. It integrates seamlessly with phishing detection and warning mechanisms of current Web browsers to protect vulnerable users against phishing attacks.

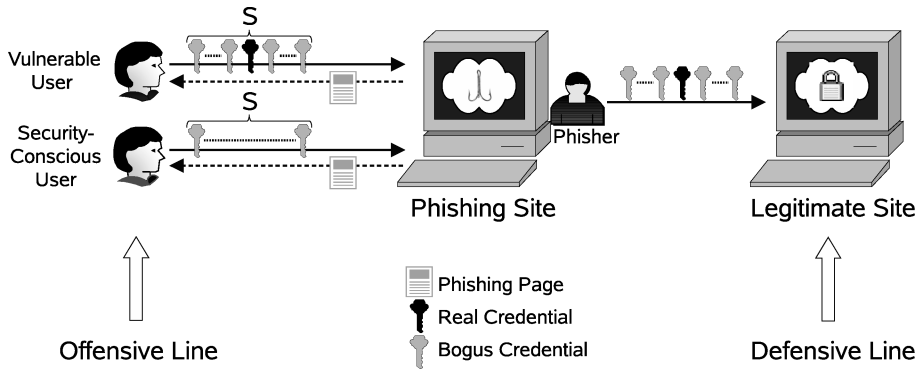

Fig. 2. Anti-phishing with BogusBiter.

3.1.1 *How It Works.* In the scenario without BogusBiter, when a phishing site is visited by users, only real credentials are submitted by vulnerable users, and a phisher can easily verify the collected credentials and trade them for money.

The basic idea of BogusBiter is very simple, as illustrated in Figure 2. When a login page is classified as a phishing page by a browser's built-in detection component or a third-party detection toolbar, BogusBiter is triggered. At this point, BogusBiter will perform differently based on a user's response to the browser's phishing warning page. For a vulnerable user who clicks the "Ignore this warning" link and submits a real credential, BogusBiter will intercept the victim's real credential, hide it among a set of $S-1$ generated bogus credentials, and then submit the S credentials one by one to the phishing site within a few milliseconds. For a security-conscious user who clicks the "Get me out of here!" link on the warning page, BogusBiter will generate a set of S bogus credentials, and then feed them one by one into the phishing site in the same way as it does for a vulnerable user. These actions are completely transparent to both vulnerable and security-conscious users. The BogusBiter extensions installed on users' browsers make up the offensive line. Later on, when a phisher verifies the collected credentials at the legitimate site, the defensive line enabled by BogusBiter will help a legitimate site to detect victims' stolen credentials in a timely manner.

3.1.2 *Design Assumption.* We assume that a phisher does not have a complete list of valid usernames for a targeted legitimate Web site, and cannot directly query a targeted legitimate Web site for the validity of a specific username. Although this assumption may not be strictly correct for email service Web sites and community Web sites, it is generally true for financial institutions, which are the main targets of phishing attacks. Financial institutions seldom have valid username lists publicly accessible. Meanwhile, for a failed login attempt, Web sites often try to hide whether the failure is due to an incorrect username or due to an incorrect password by returning the same error message [Bortz et al. 2007; Florêncio et al. 2007], making it very hard to test the validity of a given username.

Indeed, preventing the leakage of username validity information is necessary for protecting user privacy, guarding users from invasive advertising and phishing, and defending against password guessing attacks. To enhance such a protection, the recent work by Bortz et al. [2007] recommends that the response time of HTTP requests should be carefully controlled by some Web sites to remove timing vulnerabilities. Florêncio et al. [2007] further suggest that increasing username strength could be more beneficial than merely increasing password strength.

3.1.3 *Design Objectives.* To be effective, BogusBiter has two key design objectives:

—*Offensive objective*. BogusBiter should inject as many bogus credentials as possible into a phishing site, thus well hiding victims' real credentials among bogus credentials.

—*Defensive objective*. Given that a phisher is aware of BogusBiter and is willing to assume the heavy burden of sifting out bogus credentials, BogusBiter should enable a legitimate Web site to exploit the filtering process initiated by the phisher to detect victims' stolen credentials in a timely manner.

3.2 Offensive Line

To achieve its *offensive objective*, BogusBiter should strive to meet the following three requirements.

—*Massiveness*. The number of bogus credentials fed into a phishing site should be large so that the overwhelming majority of credentials received by a phisher are bogus.

—*Indiscernibility*. Without the credential verification at the legitimate Web site, it is extremely difficult for a phisher to deterministically discern, either at credential submission time or afterwards, who are real victims and what are real credentials.

—*Usability*. The usage of BogusBiter at the client-side should not incur undue overhead or unwanted side effects, nor should it produce any security or privacy concerns.

3.2.1 *Massiveness.* We use the *real-to-all* ratio—the ratio between the number of real credentials being collected and the total number of credentials being collected—to estimate how many bogus credentials should be fed into a phishing site to hide victims' real credentials. In the scenario without BogusBiter, most or perhaps all credentials collected by a phisher are real credentials submitted by victims, thus the *real-to-all* ratio is close to one. A phisher can easily verify these credentials at the legitimate Web site, assess their values, and ultimately use them to obtain money.

In the scenario of anti-phishing with BogusBiter (Figure 2), a phishing site receives both real credentials and bogus credentials. Real credentials came

from cheated users, that is, users who visited the phishing site and meanwhile became victims by revealing their credentials. The ratio between the number of cheated users and the total number of phishing site visitors can be denoted as *cheat-to-visit*. This ratio is often used by researchers to estimate the severity of phishing attacks. So if the total number of phishing site visitors is N, the number of real credentials being collected at the phishing site becomes "$N*$ *cheat-to-visit*." Meanwhile, because BogusBiter submits a set of S credentials in each phishing site visit (either by a cheated user or by a security-conscious user as explained in the design overview), the total number of credentials being collected at the phishing site becomes "$N * S$." Therefore, in the scenario of anti-phishing with BogusBiter, the *real-to-all* ratio can be computed as: $\frac{\textit{cheat-to-visit}}{S}$.

If all the phishing site visitors become victims, the *cheat-to-visit* ratio equals one. Therefore, the upper bound of the *real-to-all* ratio is $\frac{1}{S}$. However, the experiments conducted by Jakobsson and Ratkiewicz [2006] demonstrate that even with the effects of modern anti-phishing efforts, about $11 \pm 3\%$ of users will read a spoofed email, visit the phishing site, and enter their login credentials. In addition, Garera et al. [2007] found that on average, 8.24% of users become victims after visiting phishing sites. If we use 10% as a realistic value for the *cheat-to-visit* ratio, the *real-to-all* ratio becomes $\frac{1}{10S}$. Thus, if the value of the set size S is 10, a real credential will be hidden among 100 bogus credentials. Moreover, it is plausible to assume that the *cheat-to-visit* ratio will decrease in the long run due to technical advances and educational efforts—a trend that favors BogusBiter. Assuming that the indiscernibility requirement is achievable, we now analyze the probability and the expected number of tries for a phisher to single out a certain number of real credentials by verifying them at the legitimate Web site. Since each set of S credentials is submitted by BogusBiter from a user's browser within a few milliseconds, a phisher can easily group the collected credentials by sets and verify them. If a set of S credentials is submitted from a victim's browser, the real credential will be singled out by a phisher with an expected number of $\frac{S+1}{2}$ tries. However, because a phisher cannot discern which set includes a real credential, the phisher has to verify all sets of the collected credentials in order to single out as many real credentials as possible. Considering the very low *cheat-to-visit* ratio, without loss of generality, we simplify our analysis by mixing together all sets of the collected credentials. Let n be the total number of credentials collected at a phishing site, and m be the number of real credentials revealed by victims. Let X_k be the discrete random variable representing the number of tries performed by the phisher to single out k real credentials. Let $P_r(X_k = i)$ be the probability of "$X_k = i$" and $E[X_k]$ be the expectation of X_k. Intuitively, $P_r(X_k = i)$ is the probability that a phisher identifies the k^{th} real credential until the i^{th} try. That is, in the first $i-1$ tries, the phisher has identified $k-1$ real credentials; meanwhile, at the i^{th} try, the phisher also identifies a real credential. Therefore, based on the definition of the binomial coefficient, we can calculate $P_r(X_k = i)$ and $E[X_k]$ using Formula (1) and Formula (2), respectively, where $\sum_{i=k}^{n-m+k} P_r(X_k = i) = 1$

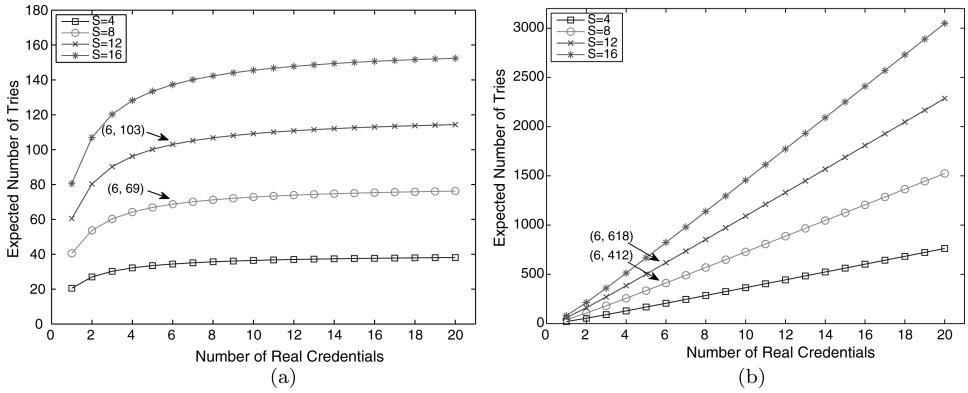

Fig. 3. Expected number of tries for a phisher to single out: (a) one real credential, (b) all real credentials.

and $k = 1, 2, \ldots, m$.

$$P_r(X_k = i) = \frac{\binom{n-m}{i-k}\binom{m}{k-1}}{\binom{n}{i-1}} \cdot \frac{m-(k-1)}{n-(i-1)} \tag{1}$$

$$E[X_k] = \sum_{i=k}^{n-m+k} i \cdot P_r(X_k = i) \tag{2}$$

$$= \sum_{i=k}^{n-m+k} i \cdot \frac{\binom{n-m}{i-k}\binom{m}{k-1}}{\binom{n}{i-1}} \cdot \frac{m-(k-1)}{n-(i-1)}.$$

With the *cheat-to-visit* ratio set to 10%, Figure 3(a) illustrates the expected number of tries for a phisher to single out one real credential, that is, $E[X_1]$. The four curves correspond to four different values of set size S. For example, if there are 6 real credentials hidden among all the collected credentials, to single out one real credential, the expected number of tries are 69 and 103, for set sizes 8 and 12, respectively. Figure 3(b) illustrates the expected number of tries for a phisher to single out all real credentials. Similarly, if there are 6 real credentials hidden among all the collected credentials, to single out these 6 real credentials, the expected number of tries are 412 and 618, for set sizes 8 and 12, respectively. From this example, we can see that a set size of 8 can already allow BogusBiter to feed a relatively large number of bogus credentials into a phishing site and well hide victims' real credentials among bogus credentials. However, we should note that such a hiding effect will never be enough to frustrate greedy phishers who intend to verify all the collected credentials. Therefore, a defensive line enabled by BogusBiter is highly desirable (see Section 3.3).

3.2.2 *Indiscernibility.* The indiscernibility requirement is essential for BogusBiter to work. It has two implications: (1) the submission actions initiated from victims' browsers should be very difficult to be differentiated from the submission actions initiated from security-conscious users' browsers, and

(2) victims' real credentials should be very difficult to be differentiated from bogus credentials generated by BogusBiter.

For a victim who ignores a browser's phishing warning, BogusBiter first intercepts the credential submission HTTP request before it is sent out. Next, BogusBiter creates $S-1$ bogus credentials based on the victim's real credential and spawns $S-1$ new HTTP requests based on the original HTTP request. Each of the $S-1$ spawned requests is exactly the same as the original request, except for carrying a bogus credential instead of a real one. Then, BogusBiter inserts the original HTTP request into the $S-1$ spawned requests and sends out all the S requests within a few milliseconds. Finally, BogusBiter interprets and properly processes the returned HTTP responses so that a phishing site cannot identify the differences between the S submissions.

For a security-conscious user who accepts a browser's phishing warning, BogusBiter first imitates a victim's behavior by entering a generated bogus credential into the phishing page and submitting it. Next, similar to the above case for a real victim, BogusBiter intercepts this original HTTP request, spawns $S-1$ new HTTP requests, and generates the corresponding $S-1$ bogus credentials as well. Finally, BogusBiter sends out the S requests and processes the returned responses in the same way as it does for a victim, thereby making it hard for a phisher to distinguish these submissions from those initiated from a victim's browser.

As for bogus credential generation, BogusBiter uses the original credential as the template to generate the $S-1$ bogus credentials. For a victim, the original credential is the victim's real credential and thus is ready to use. For a security-conscious user, the automatically generated original credential should be similar to a human's real credential. In current design, BogusBiter randomly generates a username/password pair as the original credential. For the remaining $S-1$ bogus credentials, a specific rule should be followed to generate them so that neither a human nor a computer can easily discern which is the original credential and which are the rest. We will present the rule used by BogusBiter in Section 3.3.

3.2.3 *Usability.* In terms of usability, the major advantage of BogusBiter is its transparency to users. Complementary to existing preventive anti-phishing approaches, BogusBiter automatically defends against phishing attacks without user involvement. Meanwhile, because BogusBiter only needs to submit some extra bogus credentials to a suspected phishing site and does not contact any third-party service, it will not cause any security or privacy problems.

The main usability concerns come from the scenario of a false positive (i.e., a legitimate Web site is wrongly classified as a phishing site). While the occurrence of false positives is rare for Firefox 2, IE 7, and recent detection techniques as mentioned in Section 2, BogusBiter should eliminate or reduce the possible side effects on users' access to misclassified legitimate Web sites.

The first side effect is that submitting a set of S login requests and waiting for responses will induce an additional delay to users. To reduce the delay, BogusBiter sends out all the S requests within a few milliseconds, so that the round-trip times of the S submissions can be overlapped as much as possible.

Accordingly, as long as the set size S is not too large, the additional delay incurred by BogusBiter should be minimal and unobtrusive. Our experimental results in Section 5 confirm that the additional delays are negligible.

The second side effect is that a user's real account may be locked because multiple login requests are submitted from the user's browser to a legitimate Web site within a few milliseconds. To defend against password guessing attacks, some Web sites may lock a user's account for a period of time after several failed login attempts. However, because all the usernames are different for the S login requests sent out by BogusBiter, the "account with many failed login attempts" alarm will not be triggered as discussed in [Pinkas and Sander 2002]. Our experiments on 20 legitimate Web sites confirm that account locking is not a concern for BogusBiter.

The third side effect is that a user may be asked to complete a CAPTCHA [Ahn et al. 2003] test, for the same reason that multiple login requests are submitted from the user's browser within a few milliseconds. Some Web sites may resort to this mechanism to counter password guessing attacks or denial of service attacks. However, in our legitimate site experiments where false positives are assumed to occur, no CAPTCHA test is triggered if the set size S is not greater than 10, and only two of the 20 Web sites ask a user to do a CAPTCHA test if the set size S is greater than 10.

3.3 Defensive Line

Simply requiring BogusBiter to meet the offensive objective is not sufficient. This is because even if victims' real credentials are well hidden among bogus credentials, a phisher can still visit the legitimate Web site to verify each of the collected credentials. Therefore, a defensive line is highly desirable, and BogusBiter should enable a legitimate Web site to exploit the verification process initiated (either manually or automatically) by the phisher to detect victims' stolen credentials in a timely manner.

3.3.1 *Working Mechanism.* BogusBiter makes such a defensive line feasible by imposing a *correlation requirement* upon the generation of the $S - 1$ bogus credentials. It is important that this correlation requirement should not violate the indiscernibility requirement of credential generation, that is, victims' real credentials should be very difficult to be differentiated from bogus credentials generated by BogusBiter.

—*Correlation Requirement*. Based on the original credential, a specific rule is applied to generate the $S - 1$ bogus credentials. This rule must guarantee that the S credentials in a set are correlated: given any one of them, we can reversely derive a small superset that includes all the S credentials.

BogusBiter attempts to meet both the correlation and indiscernibility requirements on credential generation by using a simple *substitution rule*. While there are other ways to meet the two requirements, we choose the substitution rule because of its simplicity and efficiency for verification. Due to our empirical experience that if the set size S is not greater than 10, no usability problem occurs and the delay overhead is small (see Section 5), the substitution rule

is tailored to have $S \leq 10$. Note that the exact value of S should be publicly known.

To generate the $S-1$ bogus username/password pairs, BogusBiter first computes an integer position i between 1 and S inclusively. This integer position determines which set of $S-1$ bogus credentials will be generated, and it also determines in which order the S credentials will be sent out to a phishing site. BogusBiter uses Formula (3) to deterministically compute this integer position i:

$$i = PRF(k,\ original\_username)\ \ mod\ \ S\ +\ 1, \tag{3}$$

where k is a master secret that is randomly chosen when a BogusBiter is installed or configured, and *PRF* is a secure pseudorandom function. The master secret k is securely stored and used by BogusBiter. A user does not need to memorize the master secret, but is allowed to export and use the same master secret on different computers. From a given original credential, the same $S-1$ bogus credentials will always be generated, and the S credentials will always be submitted to a phishing site in the same order. Therefore, even if a phisher can attack a victim multiple times, the phisher cannot find the real credential by observing which credential over time appears most often. Meanwhile, since this formula only securely hashes the original username, it is applicable both to Web sites that ask a user to submit username/password pair at the same time, and to Web sites that require a user to first submit a username and then submit a password.

Next, BogusBiter identifies the first digit in the original username as the username replacement character, denoted as *username-rc*; if the original username does not contain a digit, the first letter (upper or lower case) is identified as the username-rc. Using the same method, BogusBiter identifies the password replacement character in the original password, denoted as *password-rc*.

Then, for each integer position j from 1 to S inclusively where $j \neq i$, BogusBiter generates a bogus username/password pair by substituting both the username-rc character and the password-rc character in the original username/password pair using one of the following two replacement methods:

(1) For the case of $j - i > 0$: if username-rc (also for password-rc) is a letter, this lower (or upper) case letter is replaced by another lower (or upper) case letter $j - i$ places further down the alphabet, wrapped around if needed, i.e., 'z' is followed by 'a' (or 'Z' is followed by 'A'); if username-rc (also for password-rc) is a digit, this digit is replaced by another digit $j - i$ places further down the single digit sequence "0123456789," wrapped around if needed, that is, "9" is followed by "0."

(2) For the case of $j - i < 0$: if username-rc (also for password-rc) is a letter, this lower (or upper) case letter is replaced by another lower (or upper) case letter $i - j$ places further up the alphabet, wrapped around if needed, that is, "a" is followed by "z" (or "A" is followed by "Z"); if username-rc (also for password-rc) is a digit, this digit is replaced by another digit $i - j$ places further up the single digit sequence "0123456789", wrapped around if needed, i.e., "0" is followed by "9."

Table I.
Substitution from the original
username/password pair
(*mcsmith*/*Fuzzycat15*)

Position	Username/Password
j = 1	(kcsmith/Fuzzycat95)
j =2	(lcsmith/Fuzzycat05)
→**i =3**	**(mcsmith/Fuzzycat15)**
j =4	(ncsmith/Fuzzycat25)

Table II.
Derivation from the
username/password pair
(*lcsmith*/*Fuzzycat05*)

Username/Password
(icsmith/Fuzzycat75)
(jcsmith/Fuzzycat85)
(kcsmith/Fuzzycat95)
→(lcsmith/Fuzzycat05)
(mcsmith/Fuzzycat15)
(ncsmith/Fuzzycat25)
(ocsmith/Fuzzycat35)

Table I illustrates an example of applying the substitution rule to the original username/password pair (mcsmith/Fuzzycat15). In this example, the username replacement character username-rc is the first 'm' in the original username and the password replacement character password-rc is the digit "1" in the original password. These two alphanumeric characters will be replaced to generate $S - 1$ bogus credentials. If $S = 4$ and the computed integer position i is 3, three bogus username/password pairs are generated for $j = 1$, 2, and 4, respectively.

Finally, BogusBiter submits the S username/password pairs to a suspected phishing site following their corresponding position order. Using Formula (3) to compute the integer position i and using their position order to send out the S credentials, BogusBiter makes it hard for a phisher to narrow down a victim's real credential even if the victim visits a phishing site twice from the same browser and enters the real credential twice. However, we should note that the overall extent to which the indiscernibility requirement can be met still depends on the characteristics (such as meaningfulness) of a victim's real credential. We further discuss this limitation in Section 6.3.1.

Clearly the substitution rule above meets the correlation requirement. Given any one of the S credentials, we can derive at most $2 * (S - 1)$ variations based on the substitution rule, in which further down replacement produces $S - 1$ variations and further up replacement produces other $S - 1$ variations. These $2 * (S - 1) + 1$ credentials cover all the S credentials submitted to the phishing site. Table II lists an example derivation from the credential (lcsmith/Fuzzycat05).

Now let us see how a legitimate Web site can take advantage of the correlation requirement to identify the credentials stolen by phishing attacks. If a phisher is lucky enough (with $\frac{1}{S}$ probability) to choose a victim's real credential

```
Algorithm: SCI (f-uname/f-pword)
1.  Initialize the result set as empty : R = ∅;
2.  Construct the set : D = {(d-uname/d-pword) : (d-uname/d-pword)
                  is a credential derived from (f-uname,f-pword) };
3.  for  each (d-uname/d-pword) ∈ D do
4.      if d-uname matches a valid account's username then
5.          if d-pword matches the valid account's password then
6.              R=R ∪ {(d-uname/d-pword)};
7.          endif
8.      endif
9.  endfor
10. return the result set R;
```

Fig. 4. The Stolen Credential Identification (SCI) procedure.

as the first try to verify at the legitimate Web site, this login attempt will succeed and the legitimate Web site cannot detect the fact that a real credential has been stolen and verified. However, for any failed login attempt, the legitimate Web site will trigger the procedure of Stolen Credential Identification (SCI), which is illustrated in Figure 4. SCI takes the failed username/password pair (`f-uname/f-pword`) as its input. It constructs the set D of derived credentials (line 2), and seeks a match between a derived username/password pair and a valid account's username/password pair. Then, it adds any derived username/password pair (`d-uname/d-pword`) that matches a valid account's username/password pair to the result set R (line 6). SCI finally returns the result set R as its output.

If the failure of a login attempt is caused by a phisher who is verifying any one of the $S - 1$ bogus credentials generated from a victim's real credential, SCI must report a match since the derived credential set D contains the victim's real credential. The matched credential is the victim's real credential that has been revealed to the phisher, and is included in the result set R. However, if the failure of a login attempt is due to any other reasons, even if there is a chance that a derived username `d-uname` may match a valid account's username (line 4), the probability that the correspondingly derived password `d-pword` also happens to match this valid account's password (line 5) is extremely low. This probability is equivalent to that of randomly guessing a valid account's password. As an example, if a user accidentally mistypes the user's real password (or an attacker launches online password guessing attacks against a user), the login attempts will fail but SCI will not report a match.

Therefore, if the result set R is not empty, the username/password pair (the probability of having two or more credential pairs in the result set R is also extremely low) contained in R must have been stolen by a phisher. The legitimate Web site can take immediate actions to protect the victim even before the phisher figures out the victim's real credential. Because SCI is turned on only when a login attempt fails and it only needs a small number of verifications (at most $2 * (S - 1)$ for our substitution rule), the overhead is very small for a

legitimate Web site. If necessary, this identification task can even be delegated to a separate machine.

3.3.2 *Deployment of Defensive Line.* While BogusBiter is installed in a user's Web browser, the defensive line enabled by BogusBiter needs to be deployed only on those legitimate Web sites that are really targeted by phishers. These phishing-targeted legitimate Web sites listed in the APWG database [APWG-PSTC 2008] usually have properly registered domain names and well-designed Web pages, and may even be whitelisted by some phishing detection tools. None of their login pages will be misclassified as phishing pages by popular detection tools. The rare false positives [Zhang et al. 2007a; Robichaux and Ganger 2006] produced by phishing detection tools are mainly caused by some legitimate Web sites that are almost never targeted by phishing attacks. We do not need to deploy the defensive line of BogusBiter on them.

Moreover, the deployment work on phishing-targeted legitimate Web sites is very simple because SCI only uses these Web sites' existing authentication information and does not change their authentication mechanisms (no matter plaintext-equivalent mechanisms or verifier-based mechanisms). This server-side deployment cost is minimal compared to that of Dynamic Security Skins [Dhamija and Tygar 2005], which changes authentication mechanisms via the SRP protocol [Wu 1998], and to that of BeamAuth [Adida 2007], which demands an extra secret token for every user account.

3.3.3 *Scale-Independency Properties.* The defensive line enabled by BogusBiter also has two valuable scale-independency properties. First, the efficacy of the defensive line does not depend on the *cheat-to-visit* ratio, that is, it does not require a large percentage of users to properly respond to anti-phishing warnings. Second, the efficacy does not depend upon a massive installation of BogusBiter in users' browsers, that is, even a single vulnerable user who installs BogusBiter can benefit from a deployed defensive line. These two scale-independency properties are not only valuable by themselves, they also ensure that BogusBiter cannot be easily evaded by sophisticated phishers, as will be discussed in Section 6.

4. IMPLEMENTATION

We have implemented BogusBiter as a Firefox extension in approximately 1700 lines of JavaScript code and 100 lines of C++ code. Seamlessly integrated with the built-in phishing protection feature of Firefox 2 [FirefoxPhishingProtection 2008], BogusBiter consists of four main modules: Information Extraction, Bogus Credential Generation, Request Submission, and Response Process, as illustrated in Figure 5. We detail these four modules in the remainder of this section.

4.1 Information Extraction Module

The information extraction module extracts the username and password pair and its corresponding form element on a login page by analyzing Document

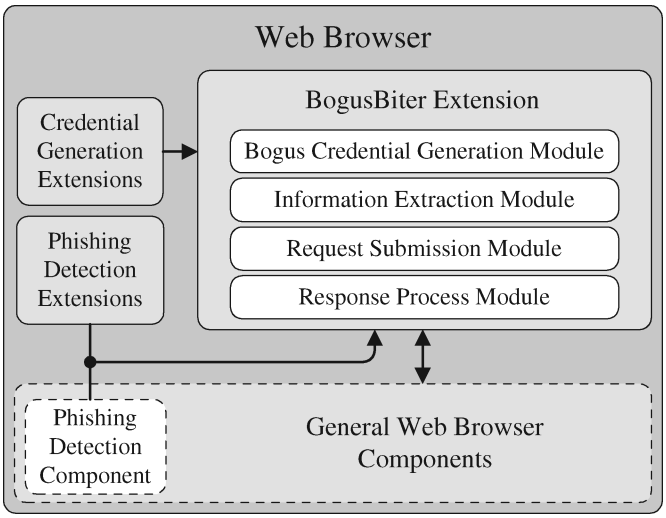

Fig. 5. Implementation of BogusBiter as a Firefox 2 browser extension.

Object Model (DOM) objects. First, all the HTMLInputElement objects within the HTMLDocument object of the login page are collected. Next, the password object is located by examining its special attribute *type="password."* A similar password locating method is also used in [Florêncio and Herley 2007; Ross et al. 2005]. Then, the HTMLFormElement object—the submission form object—associated with the password object is directly extracted. Finally, based on object attributes, the username object is extracted from other HTMLInputElement objects included in the submission form. Following this element extraction order combined with the attribute analysis of the visible input fields, BogusBiter can accurately identify username, password, and form elements on a login page. Note that phishers may use non-standard login pages to disrupt this automatic information extraction procedure and evade BogusBiter. We discuss some solutions to this kind of evasions in Section 6.3.2.

The information extraction module also implements a protection mechanism to defend against *input-stealing* attacks that use malicious JavaScript code on a phishing page to directly steal a victim's credential. Existing works such as AntiPhish [Kirda and Kruegel 2005] and PwdHash [Ross et al. 2005] provide good technical guidance for implementing such a protection mechanism. In BogusBiter, we adopt the *keystroke intercepting* technique introduced in PwdHash and create protectors on username and password elements. More precisely, username and password keystrokes are intercepted by the registered event handlers and are masked to hide from the JavaScript on a Web page. Therefore, a victim's real keystrokes are recorded by BogusBiter, but are blocked from being received by various JavaScript attacks [Ross et al. 2005]. We choose the *keystroke intercepting* technique because it is more generic than the *temporary deactivating* technique used in AntiPhish's Firefox version.

4.2 Bogus Credential Generation Module

The bogus credential generation module generates $S - 1$ bogus credentials based on an original credential. For a victim, the original credential is the victim's real credential. For a security-conscious user, in our current implementation, this module will randomly generate a username/password pair composed of upper/lower case letters and digits as the original credential. Advanced original credential generation methods can also be incorporated into BogusBiter, so that a randomly generated original credential will look more like a real user's credential. The substitution rule of BogusBiter is implemented in JavaScript. The open source HMAC_SHA256_MAC() JavaScript function implemented by Poettering [2008] is used as the secure pseudo-random function of Formula (3).

4.3 Request Submission Module

The request submission module is responsible for spawning and submitting multiple HTTP requests. Its implementation is guided by both the indiscernibility and usability requirements of BogusBiter. Since each HTTP request can only carry one credential, S requests are needed to submit a set of S credentials to a phishing site. For a victim, once a credential is entered and the submit button is clicked, the first HTTP request is initiated from the current browser window. For a security-conscious user, the action of accepting a phishing warning triggers BogusBiter to imitate a human's credential entering and button clicking actions and initiate the first HTTP request from the current browser window.

Next, just *before* the first HTTP request is actually sent out, BogusBiter is notified by Firefox's global notification service and intercepts this HTTP request. Then, BogusBiter quickly spawns the other $S - 1$ HTTP requests with each of them carrying a bogus credential. The main challenge here lies in how to efficiently spawn $S - 1$ new HTTP requests and schedule the submission of all the S requests. A few solutions are available, for example, using multiple submission windows, or reusing the submission form on one browser window to submit multiple times. However, they suffer from various usability drawbacks such as multiple Web page refreshing and long interaction time.

BogusBiter, instead, creates and uses internal HTTP channels to submit requests behind the screen. In order to make our extension code more portable, we choose to use XMLHttpRequest objects [XMLHttpRequest 2008] to create internal HTTP channels. XMLHttpRequest objects are supported by both Firefox 2 and IE 7, and they allow JavaScript to perform HTTP client functionalities such as submitting form data or loading data from a server. The first HTTP request is also associated with an HTTP channel, which is created by the browser. For this first HTTP request, all its contents, such as message header and message body [Fielding et al. 1999] can be extracted from its HTTP channel. Then, $S - 1$ XMLHttpRequest objects are created and their corresponding HTTP channels are established based on the contents extracted from the first HTTP channel. More specifically, BogusBiter executes the following four steps: request initialization, message body replacement, header fields setting, and header fields reordering:

(1) *Request Initialization:* For each of the $S - 1$ XMLHttpRequests, the same request type and URL as those in the first HTTP request are used. Asynchronous mode is used so that request sending is nonblocked and all the corresponding HTTP responses can be handled in a specified callback function. Since HTML forms, especially login forms, are in general submitted using POST instead of GET type of HTTP requests for security reasons, we only consider POST type of HTTP requests in the following discussion. Indeed, it is much simpler to process the GET type of HTTP requests.

(2) *Message Body Replacement:* For each of the $S - 1$ XMLHttpRequests, BogusBiter only needs to make a copy of the message body extracted from the first HTTP request, and then replace the original username/password pair with a bogus username/password pair. Nothing else needs to be changed in the message body. Because the bogus username/password pair and original username/password pair have the same length, the message body length does not change. Meanwhile, since the first HTTP request's message body is extracted before its HTTP channel is encrypted, this message body replacement also works correctly for secured (HTTPS) connections.

(3) *Header Fields Setting:* For each of the $S - 1$ XMLHttpRequests, the "Content-Type" request header field is set as "application/x-www-form-urlencoded" to mimic the case of submitting a form on a browser window. The "Content-Length" request header field is set to the same value as that of the first HTTP request, because the message body length is unchanged. The "Referer" request header field also needs to be set as the same value as that of the first HTTP request. The "Cookie" request header field is automatically set by the Firefox Web browser because the same URL has been specified. For each of the $S - 1$ XMLHttpRequests, BogusBiter also makes sure that "no-cache" is assigned to both the "Pragma" request header field and the "Cache-Control" request header field so that the form submissions will not be cached, and the "close" is assigned to the "Connection" request header field so that its TCP connection will not be persisted and shared with any other requests. The first HTTP request also needs to be adjusted to have the same values for these three request header fields.

(4) *Header Fields Reordering:* For each of the $S - 1$ XMLHttpRequests, the request header fields must be reordered so that the same order used in the first HTTP request will be used. Since the order of request header fields is not significant as defined in [Fielding et al. 1999], this reordering will not cause any problem. In our implementation, only the order of "Content-Type," "Content-Length," "Pragma," "Cache-Control," "Cookie," and "Referer" is adjusted by BogusBiter, due to some subtle implementation differences between an XMLHttpRequest and a regular HTTP request in Firefox. To support this reordering, we actually introduced a new function switchHeaderFieldsPosition(headerFieldA, headerFieldB) to Firefox's `nsIXMLHttpRequest` interface and `nsIHttpChannel` interface and implemented this new function in C++. This new function may also be useful for other applications that use XMLHttpRequest objects.

After the completion of these four steps, the $S-1$ XMLHttpRequests and the first HTTP request all have the same request type, URL, header fields, and header field order. Their message bodies are all the same except for carrying different username/password pairs. As previously described in BogusBiter's substitution rule, the submission order of these S requests is decided when the $S-1$ bogus credentials are generated. If the ith position ($1 \leq i \leq S$) is computed for the original credential, BogusBiter first asynchronously transmits $i-1$ XMLHttpRequests, which carry the first $i-1$ bogus credentials. Then, BogusBiter transmits the first initiated HTTP request, which carries the original credential. Finally, BogusBiter asynchronously transmits the remaining $S-i$ XMLHttpRequests, which carry the last $S-i$ bogus credentials. All the S requests are sent out within a few milliseconds, and no timing clue can be observed on a Web server or proxy.

4.4 Response Process Module

After receiving and interpreting an HTTP request, a Web site replies with an HTTP response message. For a legitimate Web site, if the credential carried in a request is valid, a successful login page is returned in the response message; otherwise, a failed login page is returned in the response message. Phishing sites may take different response actions after receiving credential submission requests (see Section 5.2). In every case, BogusBiter parses and renders the response message of the first HTTP request on the browser window, and processes the response messages of the $S-1$ XMLHttpRequests behind the screen using a callback function. Therefore, BogusBiter can always correctly match responses to their corresponding requests and work transparently to users.

Many times there are Web objects such as JavaScript and embedded images associated with each response message. These objects will be downloaded by the Web browser if the response message corresponds to an HTTP request initiated from the browser window, but by default will not be downloaded if the response message corresponds to an XMLHttpRequest. Future phishers may want to discern which are XMLHttpRequests by exploiting this fact and manipulating response contents. For example, a phisher may return a different HTML page to each submission, which includes a slightly different named image. Later on, by examining whether an image has ever been downloaded from the phishing site, the phisher can identify bogus credentials submitted by XMLHttpRequests.

To defend against such *rendering-based* attacks, BogusBiter utilizes a set of *hidden DOM windows* to render these asynchronously returned response pages for XMLHttpRequests, thus leaving no clue to phishers. Because the same Web objects cached by a browser will be directly used by different DOM windows, bandwidth-overhead incurred by this mechanism is negligible, especially for legitimate Web sites. For a phishing site that returns a different HTML page for each of BogusBiter's S submissions, the possible long delay due to downloading different Web objects will only annoy a victim and encourage the victim to leave the phishing site—a result that actually favors victims' interests.

5. EVALUATION

We conducted three sets of experiments to evaluate BogusBiter. In the first set of experiments, we built a testbed to verify the implementation correctness of BogusBiter with respect to indiscernibility. In the second and third sets of experiments, we ran BogusBiter against 50 phishing sites and 20 legitimate Web sites to validate its efficacy, in terms of attacking capability and usability.

5.1 Testbed Experiments

In the testbed experiments, we set up an Apache 2 Web server in a Linux machine and hosted over twenty various phishing Web pages on it. We used BogusBiter to send various login requests to these phishing Web pages either directly or through proxies. By examining both request logs and request contents at the Web server, we verified that all the S requests in a set are exactly the same, except for the credentials carried in the request bodies. In addition, we placed an open-source tool, Tcpmon [Tcpmon 2008], in between the Web browser and Web server to monitor TCP connections. We verified that the S submission requests are transmitted over S independent non-persistent TCP connections; therefore, it is hard for a phisher to differentiate these requests at the TCP connection level.

5.2 Phishing Site Experiments

In the phishing site experiments, we ran BogusBiter against 50 verified phishing sites chosen from PhishTank [PhishTank 2008]. PhishTank is a community based anti-phishing service and its data have been widely used for evaluating phishing detection techniques [Ludl et al. 2007; Zhang et al. 2007a, 2007b, FirefoxPhishingTest 2006]. These 50 chosen phishing sites are diverse in terms of their locations, design styles, and targeted brand names. For each phishing site, when it was online, we tested BogusBiter with four different set sizes of 4, 8, 12, and 16. Our major experiential findings are summarized as follows.

First, BogusBiter is capable of attacking all the 50 phishing sites. Acting as either a victim or a security-conscious user, BogusBiter always works correctly: it sends out all the S requests within 10 milliseconds, and then processes all the responses properly. In rare cases that phishing sites were not correctly detected by Firefox 2, we manually corrected the detection results to trigger BogusBiter.

Second, the delay caused by BogusBiter is minimal when the set size S is 4 or 8. Here the delay means the submission interaction time difference between using BogusBiter and not using BogusBiter. The submission interaction time is the time elapsed between the transmission of the first request and the reception of the last response. Figure 6(a) depicts the percentage of phishing sites versus the delay caused by BogusBiter under four different set sizes. We can see that if the set size S is 4 or 8, for over 85% of phishing sites, the delay is less than 4 seconds. This delay measure is common to either a security-conscious user or a victim, but the delay effect is different. A security-conscious user is unaware of such a delay because the user is actually redirected to a default Web page by Firefox. A victim may perceive this delay because the victim is waiting for

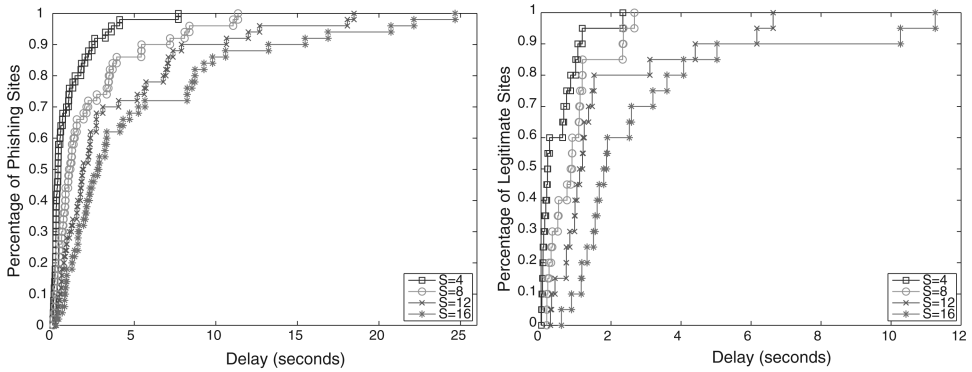

Fig. 6. Delay caused by BogusBiter on: (a) phishing sites, (b) legitimate sites, under different set size S.

the response from the phishing site. Nevertheless, it is definitely worthwhile adding a small delay on revealing a victim's credential, in order to make it less likely for phishers to succeed.

Third, phishing sites take three different response actions after receiving a user's credential submission request. Among 50 phishing sites, 38 of them simply redirect a user to the invalid login pages of the targeted legitimate Web sites; 11 of them keep a user at their local sites by using more faked Web pages; and the last phishing site is very tricky because it verifies the received credential in real time at the legitimate Web site and then sends back a response based on the verification result. If a user submits a valid credential, the phishing site steals the credential and then redirects the user to the legitimate Web site; otherwise, it lets the user re-login on the phishing site. All three types of response actions attempt to continue deceiving a victim and prevent the victim from realizing that an attack has happened. But the third type of response action not only obtains and verifies a credential in real time, it is also more deceptive to vulnerable users. The defensive line of BogusBiter indeed provides an excellent opportunity for a legitimate Web site to defend against such attacks in real time.

5.3 Legitimate Site Experiments

In the legitimate site experiments, we ran BogusBiter against 20 legitimate Web sites listed in Table III. None of these Web sites is classified as a phishing site by either Firefox 2 or IE 7. We intentionally set the detection results as phishing to simulate false positive cases, and used real accounts on these legitimate Web sites to evaluate the usability of BogusBiter. We summarize the major experimental results as follows.

First, as we expected, none of these legitimate Web sites lock a real account during our extensive tests. Second, if the set size S is 4 or 8, none of these legitimate Web sites require CAPTCHA tests. If the set size S is 12 or 16, only two Web sites ask a user to do a CAPTCHA test after receiving S credentials.

Table III. The 20 Legitimate Web Sites

paypal.com	amazon.com	gmail.com	cox.com	myspace.com
ebay.com	buy.com	yahoo.com	sprint.com	walmart.com
citibank.com	ecost.com	msn.com	geico.com	careerbuilder.com
53.com	ubid.com	aol.com	aaa.com	my.wm.edu

This test is a burden to a user but will not block a user's further interactions with a Web server. Third, the delay caused by BogusBiter is very small when the set size S is 4 or 8. Figure 6(b) depicts the percentage of legitimate sites versus the delay caused by BogusBiter under four different set sizes. We can see that if the set size S is 4 or 8, for all the 20 legitimate sites the delay is less than 3 seconds, and for over 85% of legitimate sites the delay is less than one second. Therefore, BogusBiter only induces a very small delay to users even if false positives really occur. The delay on legitimate Web sites is much smaller than that on phishing sites, since the request processing capability of legitimate Web sites is generally higher than that of phishing sites.

6. DISCUSSIONS

In this section, we discuss the deployment scale of BogusBiter, the preparations that may be needed for BogusBiter's massive deployment, and the limitations of BogusBiter.

6.1 Deployment Scale

As discussed in Section 3.3.2, the defensive line (the SCI procedure) enabled by BogusBiter needs to be deployed only on those legitimate Web sites that are really targeted by phishers. So, here we only discuss the deployment of the BogusBiter browser extension. Like most client-side protection mechanisms, BogusBiter protects only those users who install it. On one hand, due to its *scale-independency* properties, the defensive line enabled by BogusBiter can effectively identify the stolen credentials whose owners use BogusBiter, no matter how many users install BogusBiter and what percentage of them are real victims. On the other hand, the power of BogusBiter's offensive line against a phishing site is scaled to the number of users who install BogusBiter. With the increase of BogusBiter users, victims' real credentials can be better hidden among bogus credentials. Therefore, in order to protect as many users as possible, BogusBiter should be deployed as widely as possible. Ideally, if BogusBiter could be integrated into popular Web browsers as a built-in feature, a ubiquitous deployment will be easily achieved and the benefits brought by BogusBiter will be maximized.

6.2 Massive Deployment Preparation

When BogusBiter is integrated into popular Web browsers, it can be triggered with high confidence for blacklisted phishing login pages; it can also be triggered for suspicious (but not blacklisted) phishing login pages hosted on less popular Web sites. The main concern about such a massive deployment of BogusBiter is that if the login page of a legitimate site is wrongly flagged as a

phishing page, the load on the site's authentication servers will increase by a factor of S due to BogusBiter. However, the false positives produced by widely deployed phishing detection mechanisms such as used in IE 7 and Firefox 2 are rare, especially for popular Web sites that have a large number of users. This is because otherwise the false positives would have been noticed and corrected by these Web sites to prevent losing users. As reported in Zhang et al. [2007a], both IE 7 and Firefox 2 achieve a zero false positive rate for 516 representative legitimate Web sites. Thus, we expect that only few less popular and poorly designed legitimate Web sites need to prepare for a massive deployment of BogusBiter.

We suggest two simple solutions for these Web sites to prepare. Let us assume that BogusBiter's functionality is integrated into a new version of IE or Firefox Web browser. Using the browser, the operator of a legitimate Web site can easily verify whether the site's login page will be incorrectly classified as a phishing page. If a misclassification does occur, two simple solutions exist. One solution is to report this misclassification and request the Web browser vendor to either remove this legitimate site from the blacklist or add it to the whitelist. The other solution is to revise the login page of this site, for example by removing suspicious features, so that the page can pass heuristic-based tests [Chou et al. 2004; Garera et al. 2007; Ludl et al. 2007; Zhang et al. 2007b]. We suggest these preparations not merely for the need of BogusBiter's massive deployment. Indeed, legitimate Web sites may lose customers if they do not take active measures to reduce their chances of being misclassified.

6.3 Limitations of BogusBiter

Should BogusBiter become widely deployed, phishers may explore its limitations to circumvent it. In general, the potential evasions can be divided into offline evasions and online evasions.

6.3.1 *Offline Evasions.* In offline evasions, phishers analyze their collected credentials by using local username filtering techniques, meaningful credential filtering techniques, or statistical filtering techniques.

(1) *Local username filtering.* In BogusBiter's design, we assume that a phisher does not have a complete list of valid usernames for a targeted legitimate Web site, and cannot directly query a targeted legitimate Web site for the validity of a specific username. Otherwise, a phisher can simply conduct local username filtering without doing remote credential verification. Currently, this assumption may not be valid for some Web sites. For example, Bank of America's Web site can tell a user whether a login is valid before a password is entered. For these Web sites, we recommend them to hide their username validity information by using some protection methods such as suggested in Bortz et al. [2007] and Florêncio et al. [2007], thus not just to receive better protection from BogusBiter, but also to provide a necessary defense against privacy leaking, invasive advertising and phishing, password guessing, and even DoS attacks [Bortz et al. 2007; Florêncio et al. 2007].

(2) *Meaningful credential filtering.* Using current substitution rule, BogusBiter may generate meaningless bogus credentials from users'

meaningful credentials (e.g., credentials containing dictionary words or human names), especially if an original username or original password does not contain a digit. Thus, a phisher may only select meaningful credentials to verify, while discarding the rest. Although this kind of meaningful credential filtering is error-prone because a victim's real credential may be indeed meaningless and thus may be directly thrown away by a phisher, it can still be used by phishers to evade BogusBiter. Perhaps this is less of a concern for passwords, because the insecurity of low-entropy and guessable passwords has long been recognized [Halderman et al. 2005; Klein 1990; Monrose et al. 1999; Morris and Thompson 1979], and more and more high security Web sites require users to choose passwords that contain at least one letter and one number.

(3) *Statistical filtering.* A phisher may also analyze the variations of credentials and use statistical language models such as bigrams or trigrams to identify victims' real credentials. However, we argue that this type of statistical filtering is also error-prone for the same reasons as already mentioned in the meaningful credential filtering. Unfortunately, we cannot obtain representative credential datasets to further analyze and support this argument. In addition, we need to emphasize that other new rules (in addition to our substitution rule) could also be designed to generate bogus credentials from an original credential. Especially, if those new rules take into account the statistical characteristics of credentials in representative datasets, they could better hide victims' real credentials among generated bogus credentials. It is worthy to design and apply such kinds of new rules, even if they may incur the cost of increasing the derivable credentials (see Table II).

6.3.2 *Online Evasions.* Unlike offline evasions, in online evasions, phishers have to redesign their phishing sites and use special techniques to identify, in real time, which are real credentials submitted by victims. However, some inherent drawbacks limit the application and effectiveness of online evasion techniques. We now examine three representative classes of potential online evasion techniques.

(1) *JavaScript attacks.* A phisher may use two basic forms of JavaScript attacks to evade BogusBiter. One is an *input-stealing* attack that steals a user's credential using techniques such as keystroke monitoring, and then sends back the results to the phishing site at form submission time or in real time. The other is a *rendering-based* attack that manipulates response contents to discern which are bogus credentials submitted by XMLHttpRequests. As discussed in Section 4.1 and Section 4.4, BogusBiter defends against these two basic forms of JavaScript attacks by using the *keystroke intercepting* technique and the *hidden DOM windows* technique, respectively.

More sophisticated JavaScript attacks can be launched by phishers. For example, a phisher can first have the phishing site code pause for a second or two to wait for BogusBiter submitting all the S credentials. The phisher can then present all of the S credentials back to a user, along with lines of a message "To improve our security process and defend your account against automated attacks, please select your username/password from this list of credentials." If a user is fooled by such an attack, the phisher obtains the user's credential.

However, such attacks contain obvious hallmarks to distinguish themselves as malicious attacks that are specially fabricated to evade BogusBiter. Therefore, filtering functionalities can be added to BogusBiter to confidently detect and disable malicious JavaScript code. Note that detecting and filtering of malicious HTML content and JavaScript code is both desirable and feasible, and generic solutions can be found in recent research work such as SpyProxy [Moshchuk et al. 2007] and BrowserShield [Reis et al. 2006].

(2) *Nonstandard login page.* A phisher may use nonstandard login pages to evade BogusBiter. A phisher may use a login form without the *type="password"* HTML attribute, may write the entire phishing page in Flash, and may even display a virtual keyboard to users. For legitimate Web sites, using nonstandard login pages is not popular because it may cause some problems. For example, non-HTML login forms may create accessibility and usability problems [Wu 2006], and virtual keyboards are inconvenient to users and increase the risk of *shoulder surfing* attacks [VirtualKeyboard 2007; EBankingSecurity 2008]. Meanwhile, for phishing sites, using non-HTML login forms is also not popular because it makes a phishing attack more evident to users or phishing detection tools if its surface-level or deep-level characteristics become deviated from that of the targeted legitimate Web site. For these reasons, standard HTML pages remain the central focus of most anti-phishing research work [Chou et al. 2004; Kirda and Kruegel 2005; Ross et al. 2005; Wu et al. 2006b; Zhang et al. 2007b].

Indeed, BogusBiter can borrow some solutions proposed by other researchers to defend against these attacks. For example, one solution can be borrowed from Dynamic Security Skins [Dhamija and Tygar 2005]. More specifically, a customized "trusted window in the browser dedicated to username and password entry" [Dhamija and Tygar 2005] can also be used by BogusBiter. A user is required to copy the entered username and password from the trusted window and paste them to the user recognized username and password fields in a login form. Using this solution, BogusBiter can intercept an original credential before filling a phishing login form. Since BogusBiter only needs to use a user's paste actions to more accurately determine which are username and password fields, it can just paste a replaced bogus credential into a phishing login form and then do further replacements and submissions behind the screen. It is important to note that for BogusBiter, such a "trusted window" only needs to be triggered when a login page is classified as a phishing page and its username and password fields cannot be confidently identified. Also in such a case, BogusBiter will not be transparent to security-conscious users. After a security-conscious user clicks the "Get me out of here!" link on the phishing warning page, the user will be provided with the option to either really leave the site, or use the "trusted window" to help battle phishers by filling a bogus credential and identifying the username and password fields. The power of BogusBiter's offensive line may be reduced because some users may just choose to leave a phishing site, but perhaps some security-conscious users are willing to intentionally do some volunteering work to help strike back at the phishers.

(3) *CAPTCHA testing attack*. A phisher may use a CAPTCHA [Ahn et al. 2003] test to evade BogusBiter. CAPTCHA tests are mainly used to prevent automated registrations, but are seldom used in user authentication processes.

As shown in our legitimate site experiments, none of the legitimate sites asked a user to do a CAPTCHA test when the set size S is less than 10, and we actually assumed that false positives happened on all those Web sites. Introducing CAPTCHA testing attacks may decrease the number of phishing victims because the look and feel of the phishing site becomes quite different from that of the targeted legitimate Web site, and perhaps some users are unable or unwilling to solve CAPTCHAs [Kandula et al. 2005; InaccessibilityCAPTCHA 2008]. Ignoring these disadvantages, a phisher may still want to invoke a CAPTCHA testing attack at either a login page or a login response page.

These attacks may reduce the power of BogusBiter's offensive line, but will not affect the defensive line enabled by BogusBiter. If a phisher invokes the CAPTCHA testing at the login page, the $S-1$ requests generated by BogusBiter contain the same CAPTCHA answer as that of the original request; therefore, it is difficult for a phisher to tell which credential is entered by a human. If a phisher invokes the CAPTCHA testing at each of the S response pages, we recommend letting BogusBiter to make a replacement so that the CAPTCHA image on the first received response page is used on all the response pages. Therefore, it is still difficult for a phisher to identify the credential entered by a human. If there are legitimate Web sites that suffer from false positives and meanwhile want to use CAPTCHA testing on each of their response pages after seeing a small set of credential submissions from BogusBiter, they can simply send back the same CAPTCHA image on each of the S response pages thus will not be affected by this approach.

7. RELATED WORK

Basically the various client-side anti-phishing techniques can be classified into three different approaches. The first approach focuses on building tools or toolbars to enhance the security of a login process. Ye and Smith [2002] designed a prototype of "Trusted Path" to convey relevant trust signals from a Web browser to a human user. Dhamija and Tygar [2005] proposed "Dynamic Security Skins" to allow a legitimate Web site to prove its identity in a way that is easy for a user to verify but hard for a phisher to spoof. Ross et al. [2005] designed PwdHash to transparently produce different passwords for different domains, so that passwords stolen at a phishing site are not useful at a legitimate Web site. Wu et al. [2006b] introduced "Web Wallet" to direct an alternative safe path to a user if the user's intended Web site does not match the current Web site. Yee and Sitaker [2006] developed Passpet to combine the advantages of several previously devised techniques including petnames, password strengthening, and UI customization. Adida [2007] proposed BeamAuth to use a secret token in a URL fragment identifier as a second factor for Web-based authentication. These tools are very helpful, but users must be well trained to use them and must change some of their login habits. Usability is critical to the success of anti-phishing tools [Chiasson et al. 2006].

The second approach focuses on improving the accuracy of automatic phishing detection techniques. Chou et al. [2004] built SpoofGuard to compute spoof indexes using heuristics and to provide warnings for suspected phishing Web

sites. Recent work by Zhang et al. [2007b] and Garera et al. [2007] demonstrate that heuristic-based techniques can correctly identify over 90% of phishing pages with about 1% false positives. Fette et al. [2007] demonstrated that their machine-learning based techniques can correctly identify over 96% of phishing emails while mis-classifying only 0.1% of legitimate emails. Many other automatic phishing detection tools or toolbars have been developed, and both Firefox 2 and IE 7 have automatic phishing detection as a built-in feature. The evaluation of popular automatic phishing detection tools, toolbars, and Web browser features can be found in [Ludl et al. 2007; Zhang et al. 2007a; FirefoxPhishingTest 2006; Robichaux and Ganger 2006].

Researchers have also sought to develop nonpreventive anti-phishing approaches. Florêncio and Herley [2006] proposed a password rescue scheme that relies on client-side reporting and server-side aggregation to detect and protect stolen credentials. However, this scheme can only statistically make a detection decision after several users become victims, and it also raises privacy concerns by using an extra server to collect user activity information. Parno et al. [2006] proposed a Phoolproof anti-phishing mechanism. Although their mechanism eliminates reliance on perfect user behavior, a trusted mobile device must be used to perform mutual authentications. Birk et al. [2006] introduced an "active phishing tracing" method, which injects fingerprinted credentials into phishing sites to trace money laundering. Their method can support forensic analyses and enforce judicial prosecutions, but it cannot directly protect phishing victims. Anti-phishing companies such as Cyota (acquired by RSA Security) [RSA 2008] and Markmonitor [MarkMonitor 2008] have also experimented with injecting special credentials into a phishing site. However, these solutions are less effective than BogusBiter because they neither take the browser integration approach nor enable legitimate Web sites to detect victims' stolen credentials.

Finally, there is a related work in "spamming the spammers," and IBM actually offered a service to bounce unwanted email back to the computers that sent them [IBM-FairUCE 2005]. The objective of a spammer is to send junk emails, and IBM's approach intends to offend spammers by consuming their resources. In contrast, the objective of a phisher is to collect real credentials, and our approach intends to make it less likely for phishers to succeed by building both an offensive line and a defensive line.

8. CONCLUSION

We introduced BogusBiter, a new client-side anti-phishing tool to automatically protect vulnerable users by injecting a relatively large number of bogus credentials into phishing sites. These bogus credentials hide victims' real credentials, and force phishers to verify their collected credentials at legitimate Web sites. The credential verification actions initiated by phishers, in turn, create opportunities for legitimate Web sites to detect stolen credentials in a timely manner. BogusBiter is transparent to users and can be seamlessly integrated with current phishing detection and warning mechanisms on Web browsers. We implemented BogusBiter as a Firefox 2 extension and evaluated its effectiveness and usability.

Phishing is a serious security problem today, and phishers are smart, economically motivated, and adaptable. We must therefore actively pursue different approaches and promote the cooperation of different solutions. The effectiveness of BogusBiter depends on many factors, as we discussed in Section 6. But we believe that its unique approach will make a useful contribution to the anti-phishing research.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and the associate editor, Dr. Marco Maggini, for their careful and insightful comments. We also thank Barbara G. Monteith for her valuable suggestions to this article.

REFERENCES

ADIDA, B. 2007. BeamAuth: Two-factor Web authentication with a bookmark. In *Proceedings of the Conference on Computer and Communication Security (CCS)*. 48–57.

AHN, L., BLUM, M., HOPPER, N., AND LANGFORD, J. 2003. CAPTCHA: Using hard AI problems for security. In *Proceedings of Eurocrypt*. 294–311.

APWG. 2008. Anti-Phishing Working Group (APWG). http://www.antiphishing.org/.

APWG-PSTC. 2008. APWG: Phishing Scams by Targeted Company. http://www.millersmiles.co.uk/scams.php.

POETTERING, B. 2008. jssha256. http://point-at-infinity.org/jssha256/.

BIRK, D., DORNSEIF, M., GAJEK, S., AND GRÖBERT, F. 2006. Phishing phishers—tracing identity thieves and money launderer. Tech. rep. Horst-Görtz Institute of Ruhr-University of Bochum.

BORTZ, A., BONEH, D., AND NANDY, P. 2007. Exposing private information by timing Web applications. In *Proceedings of the International World Wide web Conference (WWW)*. 621–628.

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

CHOU, N., LEDESMA, R., TERAGUCHI, Y., AND MITCHELL, J. C. 2004. Client-side defense against web-based identity theft. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

DHAMIJA, R. AND TYGAR, J. D. 2005. The battle against phishing: Dynamic security skins. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*. 77–88.

DHAMIJA, R., TYGAR, J. D., AND HEARST, M. 2006. Why phishing works. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. 581–590.

DOWNS, J. S., HOLBROOK, M. B., AND CRANOR, L. F. 2006. Decision strategies and susceptibility to phishing. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*. 79–90.

EBANKINGSECURITY. 2008. eBanking Security. http://www.ebankingsecurity.com/ebanking_bad_for_your_bank_balance.pdf.

EGELMAN, S., CRANOR, L. F., AND HONG, J. 2008. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. 1065–1074.

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

FETTE, I., SADEH, N., AND TOMASIC, A. 2007. Learning to detect phishing emails. In *Proceedings of the International World Wide Web Conference (WWW)*. 649–656.

FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. 1999. RFC 2616, Hypertext Transfer Protocol – HTTP/1.1.

FIREFOXPHISHINGPROTECTION. 2008. Firefox Phishing Protection. http://www.mozilla.com/en-US/firefox/phishing-protection/.

FIREFOXPHISHINGTEST. 2006. Firefox 2 Phishing Protection Effectiveness Testing. http://www.mozilla.org/security/phishing-test.html.

FLORÊNCIO, D. AND HERLEY, C. 2006. Password rescue: A new approach to phishing prevention. In *Proceedings of the USENIX Workshop on Hot Topics in Security (HOTSEC)*.

FLORÊNCIO, D. AND HERLEY, C. 2007. A large-scale study of Web password habits. In *Proceedings of the International World Wide Web Conference (WWW)*. 657–666.

FLORÊNCIO, D., HERLEY, C., AND COSKUN, B. 2007. Do strong web passwords accomplish anything? In *Proceedings of the USENIX Workshop on Hot Topics in Security (HOTSEC)*.

FSTC-PHISHING. 2005. Understanding and countering the phishing threat. The Financial Services Technology Consortium (FSTC) Project White Paper, http://fstc.org/projects/counter_phishing_phase_1/.

GARERA, S., PROVOS, N., CHEW, M., AND RUBIN, A. D. 2007. A framework for detection and measurement of phishing attacks. In *Proceedings of the ACM Workshop On Recuring Malcode (WORM)*.

GARTNERSURVEY. 2006. Gartner, inc.,. http://www.gartner.com/it/page.jsp?id=498245.

HALDERMAN, J. A., WATERS, B., AND FELTEN, E. W. 2005. A convenient method for securely managing passwords. In *Proceedings of the International World Wide Web Conference (WWW)*. 471–479.

IBM-FAIRUCE. 2005. IBM set to use spam to attack spammer. http://money.cnn.com/2005/03/22/technology/ibm_spam/index.htm.

INACCESSIBILITYCAPTCHA. 2008. Inaccessibility of CAPTCHA. http://www.w3.org/TR/turingtest/.

JAGATIC, T. N., JOHNSON, N. A., JAKOBSSON, M., AND MENCZER, F. 2007. Social phishing. *Comm. ACM 50,* 10, 94–100.

JAKOBSSON, M. AND MYERS, S. 2006. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley-Interscience.

JAKOBSSON, M. AND RATKIEWICZ, J. 2006. Designing ethical phishing experiments: A study of (ROT13) rOnl query features. In *Proceedings of the International World Wide Web Conference (WWW)*. 513–522.

JAKOBSSON, M. AND YOUNG, A. 2005. Distributed phishing attacks. In *Proceedings of the Workshop on Resilient Financial Information Systems*.

KANDULA, S., KATABI, D., JACOB, M., AND BERGER, A. W. 2005. Botz-4-Sale: Surviving organized DDoS attacks that mimic flash crowds. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*. 287–300.

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

KLEIN, D. V. 1990. Foiling the cracker—A survey of, and improvements to, password security. In *Proceedings of the 2nd USENIX Workshop on Security*. 5–14.

KUMARAGURU, P., RHEE, Y., ACQUISTI, A., CRANOR, L. F., HONG, J., AND NUNG, E. 2007. Protecting people from phishing: The design and evaluation of an embedded training email system. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. 905–914.

KYE-PHISHING. 2008. Know Your Enemy: Phishing. http://www.honeynet.org/papers/phishing/.

LUDL, C., MCALLISTER, S., KIRDA, E., AND KRUEGEL, C. 2007. On the effectiveness of techniques to detect phishing sites. In *Proceedings of the International Conference on Detection of Instructions and Malware & Vulnerability Assessment (DIMVA)*.

MARKMONITOR. 2008. MarkMonitor: Internet Fraud Prevention and Brand Protection. http://www.markmonitor.com/.

MICROSOFTPHISHINGFILTER. 2008. Microsoft Phishing Filter. http://www.microsoft.com/protect/products/yourself/.

MONROSE, F., REITER, M. K., AND WETZEL, S. 1999. Password hardening based on keystroke dynamics. In *Proceedings of the Conference on Computer and Communication Security (CCS)*. 73–82.

MOORE, T. AND CLAYTON, R. 2007. Examining the impact of website take-down on phishing. In *Proceedings of the APWG eCrime Researchers Summit*.

MORRIS, R. AND THOMPSON, K. 1979. Password security: A case history. *Comm. ACM 22,* 11, 594–597.

MOSHCHUK, A., BRAGIN, T., DEVILLE, D., GRIBBLE, S. D., AND LEVY, H. M. 2007. Spyproxy: Execution-based detection of malicious web content. In *Proceedings of the USENIX Security Symposium*. 27–42.

PARNO, B., KUO, C., AND PERRIG, A. 2006. Phoolproof phishing prevention. In *Proceedings of the Financial Cryptography*. 1–19.

PHISHTANK. 2008. PhishTank. http://www.phishtank.com/.

PINKAS, B. AND SANDER, T. 2002. Securing passwords against dictionary attacks. In *Proceedings of the Conference on Computer and Communication Security (CCS)*. 161–170.

REIS, C., DUNAGAN, J., WANG, H. J., DUBROVSKY, O., AND ESMEIR, S. 2006. Browsershield: Vulnerability-driven filtering of dynamic html. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 61–74.

ROBICHAUX, P. AND GANGER, D. L. 2006. Gone phishing: Evaluating anti-phishing tools for Windows. http://www.3sharp.com/projects/antiphishing/gone-phishing.pdf.

ROSS, B., JACKSON, C., MIYAKE, N., BONEH, D., AND MITCHELL, J. C. 2005. Stronger password authentication using browser extensions. In *Proceedings of the USENIX Security Symposium*. 17–32.

RSA. 2008. Home - RSA, The Security Division of EMC. http://www.rsa.com/.

SCHECHTER, S. E., DHAMIJA, R., OZMENT, A., AND FISCHER, I. 2007. 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*. 51–65.

SHENG, S., MAGNIEN, B., KUMARAGURU, P., ACQUISTI, A., CRANOR, L. F., HONG, J., AND NUNGE, E. 2007. 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)*. 88–99.

TCPMON. 2008. tcpmon: An open-source utility to Monitor A TCP Connection. https://tcpmon.dev.java.net/.

VIRTUALKEYBOARD. 2007. Hacker demos how to defeat Citibanks virtual keyboard. http://blogs.zdnet.com/security/?p=195.

WHALEN, T. AND INKPEN, K. M. 2005. Gathering evidence: use of visual security cues in web browsers. In *Proceedings of the Conference on Graphics Interface*. 137–144.

WU, M. 2006. Fighting Phishing at the User Interface. Ph.D. thesis, MIT.

WU, M., MILLER, R. C., AND GARFINKEL, S. L. 2006a. Do security toolbars actually prevent phishing attacks? In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. 601–610.

WU, M., MILLER, R. C., AND LITTLE, G. 2006b. Web Wallet: Preventing phishing attacks by revealing user intentions. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*. 102–113.

WU, T. 1998. The secure remote password protocol. In *Proceedings of the Network and Distributed System. Security Symposium (NDSS)*.

XMLHTTPREQUEST. 2008. http://www.w3.org/TR/XMLHttpRequest/.

YE, Z. E. AND SMITH, S. 2002. Trusted paths for browsers. In *Proceedings of the USENIX Security Symposium*. 263–279.

YEE, K.-P. AND SITAKER, K. 2006. Passpet: Convenient password management and phishing protection. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*. 32–43.

YUE, C. AND WANG, H. 2008. Anti-phishing in offense and defense. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. 345–354.

ZHANG, Y., EGELMAN, S., CRANOR, L. F., AND HONG, J. 2007a. Phinding phish: Evaluating anti-phishing tools. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

ZHANG, Y., HONG, J., AND CRANOR, L. 2007b. CANTINA: A content-based approach to detecting phishing web sites. In *Proceedings of the International World Wide Web Conference (WWW)*. 639–648.

Received January 2009; revised October 2009; accepted November 2009