Anti-Phishing in Offense and Defense

Chuan Yue and Haining Wang
The College of William and Mary
{cyue,hnw}@cs.wm.edu

Abstract

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 implement BogusBiter as an extension to Firefox 2 web browser, and evaluate its efficacy through real experiments on both phishing and legitimate web sites.

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. 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 [6, 21, 23, 26, 28]. 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 [12, 16, 24] and takedown defenses [13, 18, 39] have also been studied. However, these defenses cannot completely foil phishing attacks and will take a long time to be effective on a large scale.

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 [4, 11, 17, 33, 36, 40], 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 mentioned above. 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.

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 implement BogusBiter as a Firefox web browser extension and evaluate its efficacy through real experiments over both phishing and legitimate web sites. Our experimental results indicate that BogusBiter is a promising anti-phishing approach.

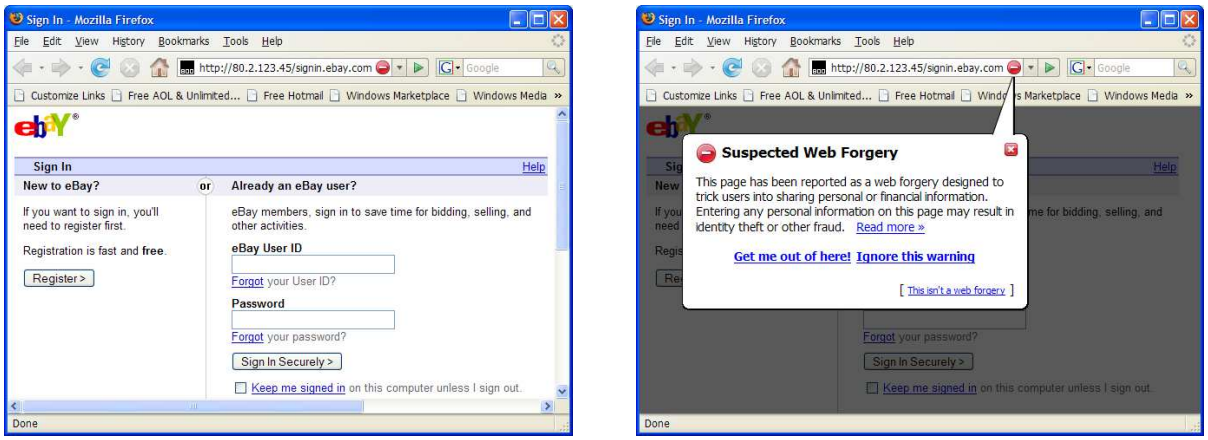
2. Background

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 [9]. 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 [15]. Therefore, protecting a user's username/password credential is the primary focus of many client-side anti-phishing research work such as SpoofGuard [4], Dynamic Security Skins [5], PwdHash [22], Web Wallet [29], and Passpet [31]. Our work also focuses on protecting a user's username/password credential. In the remainder of this paper, we use the terms credential and username/password pair interchangeably.

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 [17, 32, 35, 37]. For example, Ludl *et al.* 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 [4, 11, 17, 33]. These heuristic-based techniques have obtained very encouraging results. For example, CANTINA, a content-based detection tool proposed by Zhang *et al.* [33] can identify 90% of phishing pages with only 1% false positives. A URL-based classifier proposed by Garera *et al.* [11] is another tool which 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 [36, 40]. 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 warning of a suspected phishing site. Figure 1(b) illustrates the warning given by Firefox 2 [36] 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 on the link "Get me out of here!", the user is redirected to a default page and is protected. Otherwise, if the user clicks on the link "Ignore this warning", 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 [40]. Both Firefox 2 and IE 7 might choose such a strong 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 [6, 21, 23, 26, 28], and (2) directly blocking users' connections to suspected phishing sites is unacceptable, due to inevitable false positives. Although using strong warning pages represents current best practice, the usability study of the IE 7 warning page conducted by Schechter *et al.* [23] demonstrates that over 50% of participants still ignore the warning and enter their passwords, despite the overtness of the warning page and its strong wording. Another usability study conducted by Egelman *et al.* [7] shows that over 20% of participants ignore the strong warnings.



(a) (b)

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

3. Design

In this section, we first give an overview on the design of BogusBiter, and then we 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.

3.1.1. How It Works. When a login page is classified as a phishing page by a browser's built-in detection component, 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.

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 [3, 10], 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.* [3] recommends that the response time of HTTP requests should be carefully controlled by some web sites to remove timing vulnerabilities. Florêncio *et al.* [10] 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 hide 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 for detecting victims' stolen credentials in a timely manner.

3.2. Offensive Line

To achieve its *offensive objective*, BogusBiter needs 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 stolen and the total number of credentials being collected—to estimate how many bogus credentials could be fed into a phishing site to hide victims' real credentials. 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.

With BogusBiter equipped at each web browser, the *real-to-all* ratio will be determined by two factors. The first is the set size S, i.e., the number of credentials submitted by BogusBiter for each phishing site visit. The second is the *cheat-to-click* ratio, which is the ratio between the number of victims who reveal their credentials and the total number of users who visit the phishing site. The set size S is a parameter that we can configure, while the *cheat-to-click* ratio is related to the severity of phishing attacks. If all the phishing site visitors become victims, the *cheat-to-click* 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 [14] demonstrate that even with the effects of modern anti-phishing efforts, about $11 \pm 3\%$ of users will read a spoofed email, click the link it contains, and enter their login credentials. In addition, Garera *et al.* [11] 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-click* 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-click* ratio will decrease in the long run due to technical advances and educational efforts — a trend that favors BogusBiter.

Given the indiscernibility achieved by BogusBiter, 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 are 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-click* 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. The probability and expectation for X_k are described in Formula (1) and Formula (2), respectively, where $\sum_{i=k}^{n-m+k} P_r(X_k = i) = 1$ and $k = 1, 2, ..., m$.

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

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

For example, we use 10% as the *cheat-to-click* ratio and 10 as the value of the set size S. If there are six real credentials hidden among all the collected credentials, the expected number of tries for a phisher to single out one real credential, i.e. $E[X_1]$, is 86, and the expected number of tries for a phisher to single out all the six real credentials is 515. This example indicates that BogusBiter has the potential to feed a relatively large number of bogus credentials into a phishing site and well hide victims' real credentials among bogus credentials.

3.2.2. Indiscernibility. The indiscernibility requirement has two implications: the submission actions initiated from victims' browsers are indiscernible from the submission actions initiated from security-conscious users' browsers, and victims' real credentials are indiscernible 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. 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 mis-classified 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 [20]. 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 [25] 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

Given the indiscernibility of BogusBiter, phishers cannot single out real credentials without verifying the collected credentials one by one at legitimate web sites. Moreover, with the unique design of BogusBiter, the forced verification process, either manually or automatically conducted, will help legitimate sites to detect victims' stolen credentials and provide fraud protection in a timely manner.

3.3.1. Working Mechanism. BogusBiter makes such a defensive feature feasible by imposing a *correlation requirement* upon the generation of the $S-1$ bogus credentials, in addition to the indiscernibility requirement.

- ***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 uses a simple *substitution rule* to meet both the correlation and indiscernibility requirements. 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 uses Formula (3) to deterministically compute an integer position i between 1 and S inclusively:

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

where k is a master secret that is randomly chosen when a BogusBiter is installed or configured, and PRF is a secure

pseudo-random 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. 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 contains 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, i.e., '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, i.e., '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'.

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

Table 1: Substitution from the original username/password pair ($mcsmith/Fuzzycat15$).

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

Table 2: Derivation from the username/password pair ($lcsmith/Fuzzycat05$).

Table 1 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.

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 $2 * (S - 1)$ variations cover all the S credentials submitted to the phishing site. Table 2 lists an example derivation from the credential (lcsmith / Fuzzycat05).

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;

Figure 2: The Stolen Credential Identification (SCI) procedure.

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 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 2. 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 user-

name/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 [1] 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 [34] 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 mis-classified as phishing pages by popular detection tools. The rare false positives [32, 37] 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.

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-click* ratio, i.e., 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, i.e., even a single vulnerable user who installs BogusBiter can benefit from a deployed defensive line.

[1] The probability of having two or more credential pairs in the result set R is also extremely low.

4. Implementation

We implemented BogusBiter as a Firefox extension in JavaScript and C++, and seamlessly integrated it with the built-in phishing protection feature of Firefox 2 [36]. BogusBiter consists of four main modules. The information extraction module extracts the username and password pair and its corresponding form element on a login page by analyzing Document Object Model (DOM) objects. The bogus credential generation module generates bogus credentials based on an original credential. The request submission module spawns multiple HTTP requests and submits them to phishing sites. It uses XMLHttpRequest objects to create internal HTTP channels and submit HTTP requests behind the screen. By carefully performing request initialization, message body replacement, header fields setting, and header fields reordering, this module meets the indiscernibility and usability requirements of BogusBiter. Finally, the response process module correctly matches responses to their corresponding requests and properly processes them.

5. Evaluation

We conducted three sets of experiments to evaluate the potential efficacy of the proposed anti-phishing approach and our reference implementation.

5.1. Testbed Experiments

In the testbed experiments, we set up an Apache 2 web server in a Linux machine and hosted over twenty 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.

5.2. Phishing Site Experiments

In the phishing site experiments, we ran BogusBiter against 50 verified phishing sites chosen from PhishTank [41]. 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

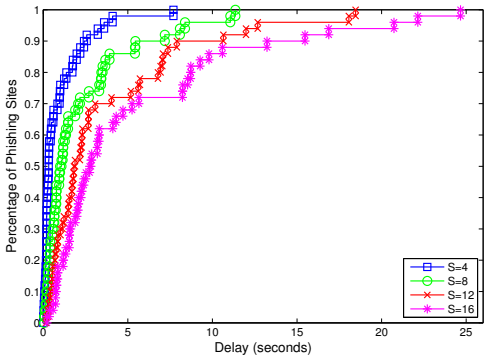


Figure 3: Delay on phishing sites.

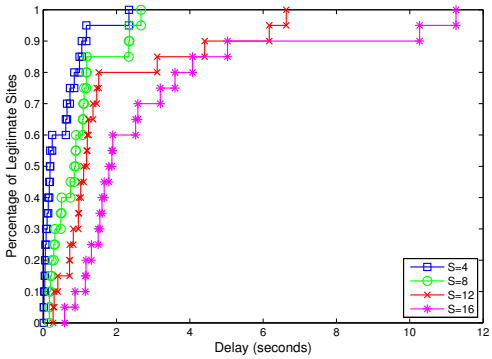


Figure 4: Delay on legitimate sites.

user, BogusBiter always works correctly. It sends out all the S requests within 10 milliseconds, and then properly processes their responses. 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 3 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 while waiting for the response from the phishing site. Nevertheless, it is 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. 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 is more deceptive. The defensive line of BogusBiter indeed provides a good 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 3. 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.

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

Table 3: The 20 legitimate web sites.

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. 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 4 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.

6. Discussions

In this section, we discuss deployment requirements, deployment preparations, and limitations of BogusBiter.

6.1. Deployment Requirements

At the client-side, users need to install BogusBiter. Vulnerable users can install BogusBiter to protect themselves, while security-conscious users can install BogusBiter to help protect others. Because BogusBiter is a browser extension, the client-side installation is straightforward. At the server-side, phishing-targeted legitimate web sites need to deploy the defensive line enabled by BogusBiter. However, this deployment work is very simple compared to that of Dynamic Security Skins [5] and BeamAuth [1], because our SCI only uses these sites' existing authentication information and does not change their authentication mechanisms.

6.2. Massive Deployment Preparations

The main concern regarding 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 server 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 [32], 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. The operators of these web sites can either revise their login pages or contact web browser vendors to fix this problem.

6.3. Limitations of BogusBiter

Phishers may use JavaScript attacks to evade BogusBiter. For example, phishers can directly steal a user's credential using keystroke monitoring techniques. Such attacks can be mitigated by adopting the *keystroke intercepting* technique introduced in PwdHash[22]. However, it is still possible for phishers to fabricate more sophisticated JavaScript attacks.

Phishers may also use non-standard login pages to evade BogusBiter. For example, phishers may use irregular HTML login forms, use CAPTCHA on login pages, or even write the entire login page in Flash. For legitimate web sites, using non-standard login pages is not popular because it may create accessibility and usability problems [27, 38]. 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 [4, 22, 29, 33].

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 [30] designed a prototype of "Trusted Path" to convey relevant trust signals from a web browser to a human user. Dhamija and Tygar [5] 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 $et\ al.$ [22] 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 $et\ al.$ [29] 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. Adida [1] 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.

The second approach focuses on improving the accuracy of automatic phishing detection techniques. Chou $et\ al.$ [4] built SpoofGuard to compute spoof indexes using heuristics and to provide warnings for suspected phishing web sites. Recent work by Zhang $et\ al.$ [33] and Garera $et\ al.$ [11] demonstrate that heuristic-based techniques can correctly identify over 90% of phishing pages with about 1% false positives. 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 [17, 32, 35, 37].

Researchers have also sought to develop non-preventive anti-phishing approaches. Florêncio and Herley [8] proposed a password rescue scheme which relies on client-side reporting and server-side aggregation to detect and protect stolen credentials. However, this scheme can only 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 $et\ al.$ [19] 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 $et\ al.$ [2] 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.

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, but we believe its unique approach will make a useful contribution to the anti-phishing research.

Acknowledgments: We thank anonymous reviewers for their insightful comments, and Barbara G. Monteith for her valuable suggestions. This work was partially supported by NSF grants CNS-0627339 and CNS-0627340.

References

[1] B. Adida. BeamAuth: Two-factor web authentication with a bookmark. In *Proceedings of the CCS*, pages 48–57, 2007.

[2] D. Birk, M. Dornseif, S. Gajek, and F. Gröbert. Phishing phishers - tracing identity thieves and money launderer. Technical Report, Horst-Görtz Institute of Ruhr-University of Bochum, 2006.

[3] A. Bortz, D. Boneh, and P. Nandy. Exposing private information by timing web applications. In *Proceedings of the WWW*, pages 621–628, 2007.

[4] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell. Client-side defense against web-based identity theft. In *Proceedings of the NDSS*, 2004.

[5] R. Dhamija and J.D.Tygar. The battle against phishing: Dynamic security skins. In *Proceedings of the SOUPS*, pages 77–88, 2005.

[6] J. S. Downs, M. B. Holbrook, and L. F. Cranor. Decision strategies and susceptibility to phishing. In *Proceedings of the SOUPS*, pages 79–90, 2006.

[7] S. Egelman, L. F. Cranor, and J. Hong. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the CHI*, pages 1065–1074, 2008.

[8] D. Florêncio and C. Herley. Password rescue: A new approach to phishing prevention. In *Proceedings of the HOTSEC*, 2006.

[9] D. Florêncio and C. Herley. A large-scale study of web password habits. In *Proceedings of the WWW*, pages 657–666, 2007.

[10] D. Florêncio, C. Herley, and B. Coskun. Do strong web passwords accomplish anything? In *Proceedings of the HOTSEC*, 2007.

[11] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the WORM*, 2007.

[12] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, 2007.

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

[14] M. Jakobsson and J. Ratkiewicz. Designing ethical phishing experiments: a study of (ROT13) rOnl query features. In *Proceedings of the WWW*, pages 513–522, 2006.

[15] M. Jakobsson and A. Young. Distributed phishing attacks. In *Proceedings of the workshop on Resilient Financial Information Systems*, 2005.

[16] P. Kumaraguru, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong, and E. Nung. Protecting people from phishing: The design and evaluation of an embedded training email system. In *Proceedings of the CHI*, pages 905–914, 2007.

[17] C. Ludl, S. McAllister, E. Kirda, and C. Kruegel. On the effectiveness of techniques to detect phishing sites. In *Proceedings of the DIMVA*, 2007.

[18] T. Moore and R. Clayton. Examining the impact of website take-down on phishing. In *Proceedings of the APWG eCrime Researchers Summit*, 2007.

[19] B. Parno, C. Kuo, and A. Perrig. Phoolproof phishing prevention. In *Proceedings of the Financial Cryptography*, pages 1–19, 2006.

[20] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *Proceedings of the CCS*, pages 161–170, 2002.

[21] Rachna Dhamija and J.D.Tygar and Marti Hearst. Why phishing works. In *Proceedings of the CHI*, pages 581–590, 2006.

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

[23] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer. 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*, pages 51–65, 2007.

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

[25] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of the Eurocrypt*, pages 294–311, 2003.

[26] T. Whalen and K. M. Inkpen. Gathering evidence: use of visual security cues in web browsers. In *Proceedings of the conference on Graphics interface*, pages 137–144, 2005.

[27] M. Wu. *Fighting Phishing at the User Interface*. PhD thesis, MIT, 2006.

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

[29] M. Wu, R. C. Miller, and G. Little. Web Wallet: preventing phishing attacks by revealing user intentions. In *Proceedings of the SOUPS*, pages 102–113, 2006.

[30] Z. E. Ye and S. Smith. Trusted paths for browsers. In *Proceedings of the USENIX Security Symposium*, pages 263–279, 2002.

[31] K.-P. Yee and K. Sitaker. Passpet: convenient password management and phishing protection. In *Proceedings of the SOUPS*, pages 32–43, 2006.

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

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

[34] APWG: Phishing Scams by Targeted Company. http://www.millersmiles.co.uk/scams.php.

[35] Firefox 2 Phishing Protection Effectiveness Testing. http://www.mozilla.org/security/phishing-test.html.

[36] Firefox Phishing Protection. http://www.mozilla.com/en-US/firefox/phishing-protection/.

[37] Gone Phishing: Evaluating Anti-Phishing Tools for Windows. http://www.3sharp.com/projects/antiphishing/gone-phishing.pdf.

[38] Inaccessibility of CAPTCHA. http://www.w3.org/TR/turingtest/.

[39] Know your Enemy: Phishing. http://www.honeynet.org/papers/phishing/.

[40] Microsoft Phishing Filter. http://www.microsoft.com/protect/products/yourself/.

[41] PhishTank. http://www.phishtank.com/.