Client-side defense against web-based identity theft

Neil Chou Robert Ledesma Yuka Teraguchi Dan Boneh John C. Mitchell
Computer Science Department, Stanford University, Stanford CA 94305
{neilchou, led242, yukat, dabo, jcm}@stanford.edu

Abstract

Web spoofing is a significant problem involving fraudulent email and web sites that trick unsuspecting users into revealing private information. We discuss some aspects of common attacks and propose a framework for client-side defense: a browser plug-in that examines web pages and warns the user when requests for data may be part of a spoof attack. While the plug-in, SpoofGuard, has been tested using actual sites obtained through government agencies concerned about the problem, we expect that web spoofing and other forms of identity theft will be continuing problems in coming years.

1 Introduction

Web spoofing, also known as "phishing" or "carding" [CNN03, FBI03], is a significant form of Internet crime that is launched against hundreds or thousands of individuals each day. The US Secret Service and the San Francisco Electronic Crimes Task Force report that approximately 30 attack sites are detected each day. Each attack site may be used to defraud hundreds or thousands of victims, and it is likely that many attack sites are never detected. A typical web spoof attack begins with bulk email to a group of unsuspecting victims. Each is told that there is a problem with their account at a site such as E*Trade. Victims of the spoofing attack then follow a link in the email message to connect to a spoofed E*Trade site. Once a victim enters his or her user name and password on the spoof site, the criminal has the means to impersonate the victim, potentially withdrawing money from the victim's account or causing harm in other ways.

We describe some common characteristics of recent web spoofing attacks and propose a framework for client-side countermeasures. Like other inexact detection mechanisms, including virus detection and email spam filtering, the approach we explore involves looking for characteristics of previously detected attacks. We experiment with this approach using a browser plug-in called SpoofGuard. The plug-in monitors a user's Internet activity, computes a spoof index, and warns the user if the index exceeds a level selected by the user. While Internet-savvy users who watch the address bar, status bar, and other information carefully may not need SpoofGuard, the current level of accuracy and effectiveness may be sufficient to help many unsophisticated web users. If the methods we propose become widely deployed, through our plug-in or through other client-side defensive software, then phishers will certainly take steps to circumvent them. However, we expect further effort and study to produce correspondingly better defenses. Moreover, if synergistic server-side methods are deployed by concerned companies, it seems possible to thwart increasingly sophisticated attacks.

SpoofGuard uses domain name, url, link, and image checks to evaluate the likelihood that a given page is part of a spoof attack. For example, a page with a suspicious url such as `etrade-maintenance.suspicious.org` or `www.etrade.com@129.170.213.101/maintainance.asp` and an E*Trade logo will have a higher spoof index than a page with neither of these characteristics. SpoofGuard also uses history, such as whether the user has visited this domain before and whether the referring page was from an email site such as Hotmail or Yahoo!Mail. Most importantly, SpoofGuard intercepts and evaluates user posts in light of relevant history and the spoof index of a form page. SpoofGuard examines post data user name and password fields and compares posted data to previously entered passwords from different domains. This mechanism warns a user against sending her E*Trade password to a site with an E*Trade logo but outside the `etrade.com` domain, for example. Password comparisons are done using a cryptographically secure hash, so that plaintext passwords are never stored by SpoofGuard.

Stopping web spoofing bears some similarity to intrusion detection, spam filtering, and thwarting traditional social engineering attacks. Intrusion detection systems [Pax99, Sno03] typically monitor network and host ac-

tivity, compute statistical or other indices, and attempt to detect intrusions by comparing the index of current activity against previous statistics. While web spoofing may be regarded as a special case of intrusion detection, the browser seems like that appropriate place to combat web spoofing. A browser plug-in is relatively easy to install and has access to honest and spoof pages sent over https, giving SpoofGuard a better chance of catching an attack than a network proxy or other external http traffic monitors. While a plug-in alone does not have full information from email programs such as Outlook or Eudora that may contain the messages that launch an attack, the browser does provide an indication of the referring page or application, and it is possible to scan and parse pages from email sites such as Hotmail or Yahoo!Mail. Therefore, for non-expert users who read email through their browser, SpoofGuard has the potential to examine every step of a standard web spoof attack.

Like other intrusion detection efforts, it is appropriate to evaluate SpoofGuard by measuring its effective at preventing attacks, the false alarm rate (number of unnecessary warnings), and its performance impact. SpoofGuard will only be useful if it detects attacks without raising too many false alarms, since users will almost certainly reject any method that interferes with normal browsing activity. We have evaluated the false-alarm rate by using SpoofGuard ourselves over a period of time, and we have evaluated its effectiveness for preventing attacks using actual spoof sites brought to our attention by members of the San Francisco Electronic Crimes Task Force. While this is not an extensive enough test to draw broader conclusions, SpoofGuard does catch the sample attacks found in the wild and does not add any noticeable delay to ordinary web browsing.

Since web spoofing attacks begin with bulk email, a good general spam solution [Bri03, Din03] could reduce the incidence of web spoofing attacks. However, current spam solutions are only partly effective at blocking unwanted email, and we are not aware of any spam efforts aimed specifically at identify theft. While the browser-based techniques we explore in this paper are complementary and independent of spam filtering, there may be additional ways of combining email scanning with web page analysis that will lead to better spoof prevention in the future.

Previous efforts by the Princeton Secure Internet Programming group and others [FBDW97, EY01] have addressed another form of "web spoofing" in which an attacker causes all html page requests from a victim to pass through the attacker's site. This form of web spoofing allows the attacker to monitor all of the victim's activities, including posted passwords or account numbers. However, previous methods for countering this form of attack have focused on maintaining the integrity of browser indicators such as the url indicator in the status bar, not analyzing user behavior, web pages, and html post data to stop leakage of sensitive user information. While we considered using an alternate term such as "phishing" in this paper, we use "web spoofing" since this currently appears to be the term most commonly used by law enforcement and concerned companies.

The goals of this paper are to raise awareness of the web spoofing problem and propose a framework for client-side protection. While sophisticated and determined attackers will be able to circumvent our current tests (through simple techniques we explain later in the paper), there is plenty of room for improving specific tests and tuning the coefficients of our spoof index function. Furthermore, the web spoofing problem is important and we believe our SpoofGuard experience will be useful for developing more sophisticated defenses. We discuss the web spoofing problem in more detail in Section 2, and our solutions in Section 3. The SpoofGuard implementation and user interface are described in Section 4. Some SpoofGuard evaluation information appears in Section 5, followed by suggestions for server-side methods in Section 6, some more speculative client-side methods in Section 7, and concluding remarks in Section 8.

Throughout the paper we use the following terminology.

- *Spoof site* or *Spoof page*: the site or page that is a malicious copy of some legitimate web page.
- *Attacker*: the person or organization who sets up the spoof site.
- *Honest site* or *honest page*: the legitimate site or page that is being spoofed.
- *Spoof index*: a measure of the likelihood that a specific page is part of a spoof attack, described in Section 3.

A prototype version of SpoofGuard will be made publicly available shortly.

2 The problem

According to Agents of the U.S. Secret Service San Francisco Electronic Crimes Task Force [Von03], the U.S. Government's Internet Fraud Complaint Center received over 75,000 complaints in 2002. Of this number, 48,000 cases resulted in further action requests. This is a three-fold increase over 2001. The total dollar losses are estimated at more than $54 million compared to $17 million for 2001. A majority of these fraud complaints are intrusions, auction fraud, credit card/debit fraud, and computer intrusion. Agents of the U.S. Secret Service San Francisco Electronic Crimes Task Force report that web spoofing was first noticed in late 2001 and grew in

popularity in 2002, correlating with the large increase in Internet Fraud. Further, a majority of the $37 million increase in losses from 2001 to 2002 can be attributed to web spoofing. Agents working fraud cases in the Bay Area also report that a majority of their Internet cases involve web spoofing.

One factor that adds to the severity of web spoofing attacks is that many users use the same username and password at several sites. This allows a phisher who reels in a victim to use this information on more than one site. For this reason, companies that provide password-protected services are dependent on each other for their security. This is not only true with regard to web spoofing, but for other kinds of attacks as well. If passwords from one site can be stolen by attacking the site itself, these may also be used at other sites that protect their password database more effectively.

2.1 Sample attack

A recent attack described in a New York Times article [HF03] actually mentioned fraudulent email, indicating some level of public awareness of spoof attacks. On June 18, 2003, thousands of fraudulent e-mails with the subject "Fraud Alert" were sent out, hoping to reach Best Buy customers. The e-mails attempted to convince customers that Best Buy's fraud department required additional customer information, "in our effort to deter fraudulent transactions." To further lure unsuspecting victims, the e-mail provided a link that purported to reach a "special Fraud Department" at the Best Buy web site. Instead, the link actually pointed to a fraudulent page unrelated to Best Buy. The Best Buy attacker's page resembled an official Best Buy page, using the Best Buy logo, incorporating elements from an official Best Buy page, and providing links to other Best Buy resources. The page requested a customer's social security number and credit card information.

A web page from the Michigan Attorney General [Cox03] cites "a few giveaways to this particular scam:"

- The [email] message did not issue from an `@bestbuy.com` address,
- The link embedded in the message does not take the user to a "special Fraud Department page" on Best Buy's site, but to a page hosted under a completely different domain name (such as `digitalgamma.com` or `your-instant-credit-reporter.org`),
- The "National Credit Bureau" mentioned in the scam does not exist.

The Michigan Attorney General also points out that the Best Buy spoof is similar to spoofs imitating PayPal and eBay. A more recent Dow Jones Newswires story [Ber03] states that EarthLink, Citibank, Morgan Stanley's Discover unit, eBay Inc. and its PayPal unit, Wachovia Corp.'s First Union unit and the Massachusetts State Lottery reported phishing scams in recent months. Some general information about web spoofing, including additional news articles and records of actual attacks, may be found at `http://www.antiphishing.org/`, a web site provided by Tumbleweed Communications.

2.2 Properties of recent attacks

We describe common properties of ten spoof web sites recently found in the wild. Figure 4 gives an example of an Ebay spoof (partially obscured by a SpoofGuard pop-up warning the user).

- *Logos.* The spoof site uses logos found on the honest site to imitate its appearance.
- *Suspicious urls.* Spoof sites are located on servers that have no relationship with the honest site. The spoof site's url may contain the honest site's url as a substring (`http://www.ebaymode.com`), or may be similar to the honest url (`http://www.paypaI.com`). IP addresses are sometimes used to disguise the host name (`http://25255255255/top.htm`). Others use @ marks to obscure their host names (`http://ebay.com:top@255255255255/top.html`), or contain suspicious usernames in their urls (`http://middleman/http://www.ebay.com.`)
- *User input.* All spoof sites contain messages to fool the user into entering sensitive information, such as password, social security number, etc. Some successful spoofs have even been so bold as to ask for name, address, mother's maiden name, driver's license, and so on.
- *Short lived.* Most spoof sites are available for only a few hours or days – just enough time for the attacker to spoof a high enough number of users. The implication is that defensive methods that alert the user to a spoof site are more effective than reactive methods that attempt to shutdown the site.
- *Copies.* Attackers copy html from the honest site and make minimal changes. Two consequences are: (i) some spoof pages actually contain links to images (e.g. logos and buttons) on the honest site, rather than storing copies, (ii) the names of fields and html code remain as on the honest site. We note that when a spoof site refers to the honest site for embedded images it gives the honest site an opportunity to detect the spoof: the honest site detects an http request for an embedded image where the referral header is not the honest site. Such requests should not occur unless the honest site is being plagiarized.

- *Sloppiness or lack of familiarity with English.* Many spoof pages have silly misspellings, grammatical errors, and inconsistencies. In the Best Buy scam, the fake web page listed a telephone number with a Seattle area code for a Staten Island, NY, mailing address.
- HTTPS is uncommon. Most spoof web sites do not use https even if the honest site does. This simplifies setting up the spoof site.

3 Solutions

A number of tests can be used to distinguish spoof pages from honest pages. We present the tests we implemented and evaluated in three groups: stateless methods that determine whether a downloaded page is suspicious, stateful methods that evaluate a downloaded page in light of previous user activity, and methods that evaluate outgoing html post data. Our browser plug-in applies these tests to all downloaded pages and combines the results using a scoring mechanism described below. The total spoof index of a page determines whether the plug-in alerts the user and determines the severity and type of alert. Since pop-up warnings are intrusive and annoying, we attempt to warn the user through a passive toolbar indicator in most situations. A user checkbox can eliminate all pop-ups if desired.

We note that server-side methods, such as tracking server image requests, may also be effective in identifying spoof sites. However, the focus of this paper is on client-side browser solutions. In section 6, we comment on some ways that server-side modifications may make our client-side methods more reliable and effective.

3.1 Scoring

Given a downloaded web page and some browser state as input, our plug-in applies tests $T_1, \ldots, T_n$, with test T_i producing a number P_i in the range $[0, 1]$. By convention, $P_i = 1$ indicates that the page is likely to be a spoof and $P_i = 0$ indicates the opposite. Most of our tests return either 0 or 1, but some can return a value between 0 and 1.

We combine the test results into a total spoof score, TSS, using a standard aggregation function:

$$\begin{aligned} TSS(page) &= \sum_{i=1}^{n} w_i P_i \\ &+ \sum_{i,j=1}^{n} w_{i,j} P_i P_j \\ &+ \sum_{i,j,k=1}^{n} w_{i,j,k} P_i P_j P_k \\ &+ \ldots \end{aligned}$$

The w's are preset weights selected to minimize the false alarm rate. Note that most of the w's are set to zero so that the actual number of terms in the expression is relatively small. This approach of applying multiple tests and combining the results using a scoring mechanism is commonly used in intrusion detection systems and spam filters [Din03].

The scoring function not only sums individual tests, but also sums products of pairs, triples, and larger subsets of tests. The reason for product terms is that when certain combinations of events occur the likelihood of the page being a spoof increases dramatically. For example, if a company logo appears on an unauthorized page and the page contains password and creditcard fields, the page is very likely to be a spoof. Consequently, the term corresponding to the product of these three tests is given substantial weight.

3.2 Stateless page evaluation

We begin by describing a collection of tests that work by examining the current page only.

Url check There are various methods that attackers can use to produce misleading urls. For example, an @ in a url causes the string to the left to be disregarded, with the string on the right treated as the actual url for retrieving the page. Combined with the limited size of the browser address bar, this makes it possible to write urls that appear legitimate within the address bar, but actually cause the browser to retrieve a page from an arbitrary site.

Image check Spoof sites usually contain images taken from the honest site. For example, the eBay logo appears on spoofed eBay pages to give the user the impression that they are communicating with eBay. If the eBay logo appears on a login page unrelated to eBay, that page is suspicious. The same applies to other identifiably eBay-specific images such as banners and buttons. We note that corporate logos often legitimately appear on many e-commerce sites (e.g., the Amazon logo appears on sites that sell products through Amazon) and therefore we only count this test for pages that ask for private user input.

In order to apply this check in a stateless way, the SpoofGuard plug-in is supplied with a fixed database of images and their associated domains. Since attackers generally do not have email lists for customers of specific sites, they must try to spoof sites that are used by a significant fraction of web users. Thus SpoofGuard can be useful even if we only account for relatively small number of frequently spoofed domains such as eBay, PayPal, AOL, and so on. When the browser downloads a login page all images on the page are compared to images in the SpoofGuard database. The spoof-score for the page is increased if a match is found but the page's domain is not a valid domain for the image.

What if the spoof page contains a slight modification

of the real image? The image comparison test might fail to detect the spoof. Fortunately, as noted earlier, attackers often directly copy or link to images on the honest site. Nevertheless, we defend against small image modification by storing an image hash rather than the actual image. Image hashing refers to a hashing algorithm that produces the same hash for similar images. While present technology does not provide ideal image hashes, there has been some progress in this area [VKJM00]. In our case, image hashing can be strengthened by asking e-commerce sites to use images that are especially well suited for image hashing. For example, in many cases we could use optical character recognition (OCR) as the image hashing algorithm. An added benefit of image hashing is that storing an image hash rather than the full image reduces plug-in storage requirements. We discuss other aspects of this test in Section 5.3.

Link check The links contained within a page are examined. The link check fails for a page if at least one-fourth of the links fail the url check described above.

Password check Pages that request a password merit closer scrutiny than pages that do not. If a page requests a password (or other sensitive information), we also check whether https is used and, if so, whether the certificate check succeeded or failed.

3.3 Stateful page evaluation

In stateful page evaluation, the browser history file and additional history stored by SpoofGuard are used to evaluate the referring page. Since it is important to minimize the number of false alarms, SpoofGuard does not issue any warnings for visiting a site that is in the user's history file. The rationale for this is that if the user is warned the first time, and decides to proceed, the user is assumed to have sufficient reason to trust the site.

Domain check If the domain of a page closely resembles a standard or previously visited domain, the page may be part of a spoof. Although crude, we currently compare domains by Hamming (edit) distance. For example `efrade.com` will raise the domain check if `etrade.com` is in the file of commonly spoofed sites or in the user history. Clearly, it is possible to improve our comparison algorithm by studying the way people are fooled; this is a significant direction for future work.

A related issue is that some businesses outsource some of their web operations to contractors with different domain names. This poses an interesting challenge that we believe can be addressed. However, outsourced web activity leads to false alarms in the current version of SpoofGuard.

Referring page When a user follows a link, the browser maintains a record of the referring page. Since the typical web spoofing attack begins with an email message, a referring page from a web site where the user may have been reading email (such as Hotmail) raises the level of suspicion. One complication associated with Hotmail, for example, is that Hotmail uses numeric IP addresses instead of symbolic host names. Therefore, when a user clicks on a link in a Hotmail message, the browser provides a numeric IP address to SpoofGuard as the referring page. In this situation, SpoofGuard uses reverse DNS to find the domain name associated with a numeric address, allowing us to identify Hotmail as the referring site.

Image-domain associations The image check described above (in section 3.2) relies on a database associating images such as corporate logos with domains. The initial static database can be assembled using a web crawler or other tool, or it can be augmented using an individual's browsing history. An early version of SpoofGuard used a fixed database; the current SpoofGuard implementation uses a hashed image history file.

3.4 Evaluating post data

Evaluating post data is a critical part of any client-side defense against web-spoofing attacks, since the point of any defense is to prevent malicious sites from gaining confidential information from an honest web user. When a user fills in form data, SpoofGuard intercepts and checks the html post data, allowing the actual post to proceed only if the spoof index is below the user-specific threshold for posts. If a user confines pop-up warnings to posts only, then the user will never be interrupted when reading web pages, only (possibly) when filling in forms. Note, however, that even if no warnings are generated on pages leading up to the spoof form, the page checks described above are used, in combination with analysis of the post data, to determine the spoof index associated with an html post.

Outgoing password check SpoofGuard maintains a database of ⟨domain, user name, password⟩ triples. If the user reuses a password on a new domain, this trips the password check. To avoid the possibility of leaking sensitive information, the stored passwords are hashed using SHA-1 and the comparison is performed on hashed values.

Interaction with image check In our spoof index calculation, the image check interacts with the outgoing password check non-linearly. For example, if a user enters her E*Trade user name and password to a site that is

not at `etrade.com`, this raise the spoof index a certain amount (determined in part by user-selected weights). The spoof index is raised multiplicatively higher if the site also contains the E*Trade logo.

Check of all post data There are several ways that a web page might request a password. For example, a clever spoof site might use an image of the word "password" instead of html text to request the user's password. To protect against this form of spoof attack, all outgoing data in an html post can be hashed and checked against a database of passwords and other information deemed sensitive. In this way, we can still detect password leakage, even if the spoof page does not contain the text "password."

Exception for search engines Since a user may enter any data into a search engine, SpoofGuard is not suspicious of known search engines at known domains. It is also possible to ignore data posted into a "search" or "find" field in an arbitrary page. This allows for shopping sites, for example, that allow a customer to search the catalog by keyword or product name. Of course, good password practice would prevent an intelligent user from using a product name or other English word as an important password, rendering this exception unnecessary.

4 SpoofGuard architecture

SpoofGuard is an Internet Explorer *browser helper object*, or "plug-in." A browser helper object is a COM component that is loaded when IE starts up; it runs in the same memory context as the browser. In general, a browser helper object may perform any action on IE windows and modules, including manipulating the browser menu and toolbar, detecting and responding to browser events, and creating additional windows.

SpoofGuard accesses the Internet Explorer history file and uses three additional files stored in the user profile directory. One is a read-only file of host names of email sites such as Hotmail and Yahoo!Mail, used in the referring page check. The other two files are the file of hashed password history (domain, user name, and password) and the file of hashed image history. SpoofGuard can use reverse DNS to find domain names for numeric IP addresses, but does not otherwise send or receive any information on the network. The browser history file can be reset using a browser dialog box and the additional SpoofGuard histories can be reset using a button on the SpoofGuard configuration panel.

4.1 Plug-in interface and access to browser data

SpoofGuard consists of a COM component that extends IDeskband, an interface that causes IE to load SpoofGuard as a registered toolbar, and a few other modules that run in response to actions of the toolbar component. SpoofGuard is written in Visual C++, and uses both Windows Template Library (WTL) 7.0 and Microsoft Foundation Class Library (MFC). Two SpoofGuard window classes implement the CWindowImpl interface to define the appearance and user interaction of the toolbar. The interaction between the main modules, described below, is shown in Figure 1.

- WarnBar: This is a COM component that houses the SpoofGuard toolbar. All site evaluations and post data checks are carried out here.
- ReflectionWnd: This CWindowImpl class implements a transparent window that sits on top of the toolbar and reflects user messages (e.g. mouse clicks) to UWToolBar. WarnBar requests ReflectionWnd to pop-up a warning message when the user tries to send sensitive information to a suspicious server.
- UWToolBar: This CWindowImpl class defines the appearance of the toolbar. UWToolBar stores the user settings (e.g. check index, threshold, etc.) during runtime. WarnBar requests UWToolBar for these settings to determine the traffic lights color and the warning messages that appear in the Current Page Status dialog. User settings are stored in the registry when SpoofGuard closes.
- ConfigDlg opens an Options window when the user clicks the Options button. UWToolBar updates the user settings based on the result that ConfigDlg returns when the window terminates.
- DomainDlg opens the Current Page Status window when the user clicks on the traffic light icon. It contains the warning messages specific to the current page.

Browser data and events When Internet Explorer launches, it calls the SetSite method in the IObjectWithSite interface to initialize WarnBar. WarnBar receives a pointer to the web browser object, and passes the object to the ReflectionWnd and UWToolBar, allowing SpoofGuard to constantly check the browser contents during execution. Internet Explorer's DWebBrowserEvents2 class exports BeforeNavigate2 and DocumentComplete event handlers, which are both implemented in WarnBar class. A BeforeNavigate2 event occurs before navigation. This gives WarnBar the url that the browser is attempting to navigate to, the outgoing post, and a chance to cancel the navigation. The

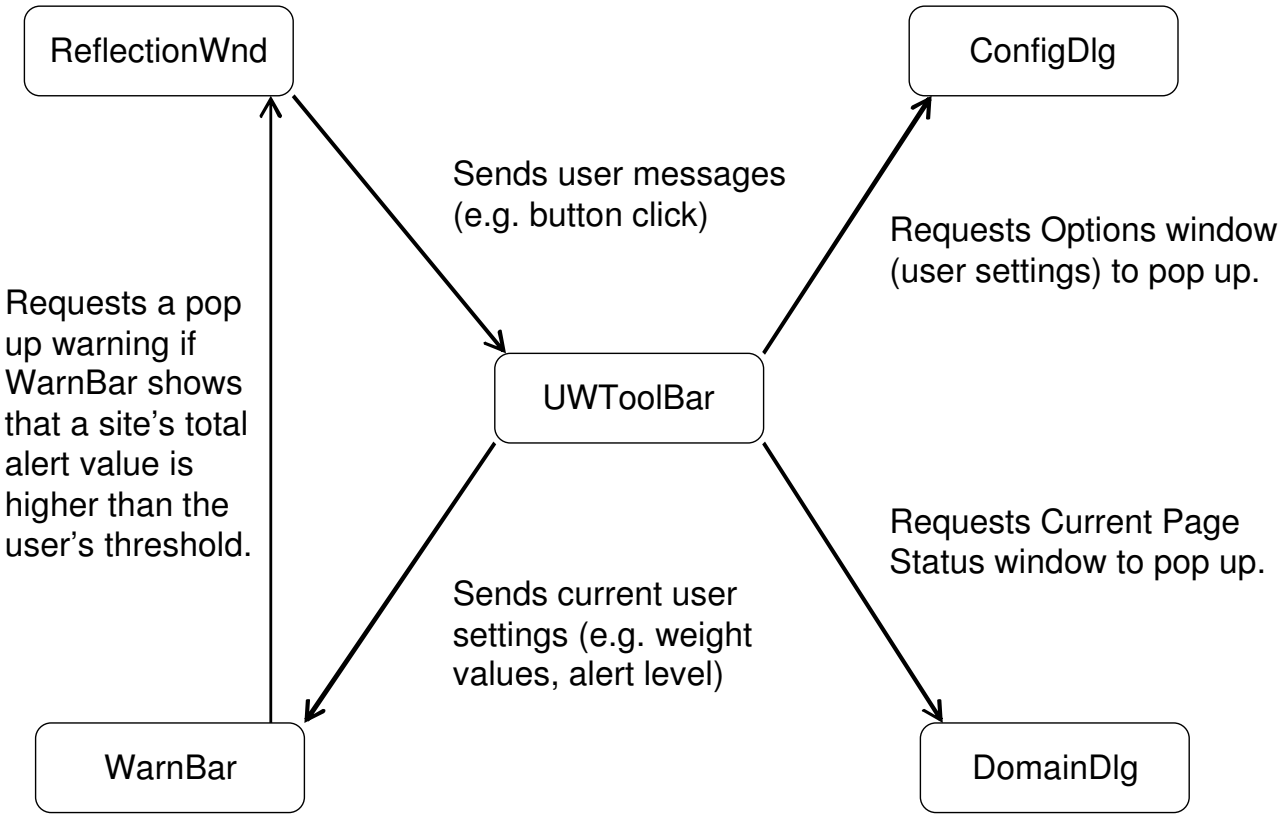

Figure 1. SpoofGuard architecture

Figure 2. SpoofGuard toolbar

url check using the history list, the domain name check, the email referring page check and post data check are carried out after a BeforeNavigate2 event. A DocumentComplete event occurs when a web site finishes loading completely. The image check, link check, and password check are carried out after a DocumentComplete event.

4.2 User interface

The SpoofGuard toolbar is shown in Figure 2. The options button can be used to configure the tool, while the traffic light (green, yellow, or red) provides an indication to the user about the current page. Clicking on the traffic light also pops up additional information about the current page. When the spoof rating is above the user-specified threshold, SpoofGuard will pop-up an additional warning window that requires user consent to send user web form input (or other http post data) out to a web site.

The configuration pop-up, shown in Figure 3, lets the user select a spoof rating threshold, and set independent weights and sensitivity levels for the domain name, url, link, password, and image checks. The user may also disable pop-ups, set history cache, enable image hash caching, and enable highlighting of suspicious links.

SpoofGuard has two methods to convey its analysis to the user. To keep SpoofGuard as unobtrusive as possible, we use a traffic light symbol on the browser bar to indicate the degree of spoof by color: red, yellow, and green. The actual colors displayed are determined by the user's threshold settings. Should the user want to read the details of SpoofGuard's analysis, he or she can click the traffic light, and more information appears. In extreme situations, SpoofGuard may also halt a post and ask the user if she wishes to continue. In combination, the traffic light and popup provide an effective means of alerting the user to suspicious web pages, while avoiding the annoyance of consistent popup windows.

4.3 Implementation difficulties and solutions

Detecting whether the user has clicked on an e-mail link A typical phisher sends unsolicited e-mails that contain links to the spoof site. SpoofGuard therefore attempts to determine whether the user was directed to a site from an e-mail message. One way is to check the referrer field against a list of host names associated with web-based e-mail providers. However, some providers (e.g., Hotmail) provide links to other parts of their service using numeric IP addresses instead of symbolic host names. Given a numeric address, SpoofGuard performs a reverse-DNS lookup to reveal the host name. Another way to estimate the likelihood that an email link was used is to see whether the referrer field is empty. The referring page field is empty with the browser is initially started, and when the user types in a url. Although an empty referrer field does not always imply that the user has clicked on an e-mail link, the field is empty if the user launches the browser through a link in his or her e-mail software. While it is conservative to treat an empty referrer field in the same way as a link from Hotmail, this may give false alarms since the referrer field is empty whenever a new Internet Explorer window is opened.

Different input names for usernames and passwords Since different sites have different input field names for usernames and passwords, twenty username and ten password variations are predefined in SpoofGuard, and they are used to identify sensitive information in the obtained post data structure. These predefined names are used by many online bank forms, commercial sites such as Amazon and eBay, and web-based e-mail sites. SpoofGuard currently does not recognize username and password combinations from sites that use other input field names.

Frames Frames have historically proven troublesome to both web browsers and users. For simplicity, SpoofGuard currently treats frames as independent pages, without parsing the frameset to determine its frames. For example, if a frameset includes frames located on different hosts, SpoofGuard may flag this situation as a possible malicious redirect. Related to this is a customizable security setting in Internet Explorer, "Navigate sub-frames across different domains", which gives the user a choice to allow or disallow this behavior. We expect to improve the handling of frames in a future version of SpoofGuard.

POST data vs AutoComplete SpoofGuard compares outgoing passwords with its database of <username, password, domain> triples rather than with values stored in Internet Explorer's AutoComplete repository. When a user submits a form, SpoofGuard obtains the post data as a SafeArray structure, which Internet Explorer passes to SpoofGuard via an event handler. SpoofGuard then scans it for sensitive information, and stores a resulting hash into a file. There are two advantages to using post data. First, post data checking is more secure, because SpoofWatch hashes the pass-

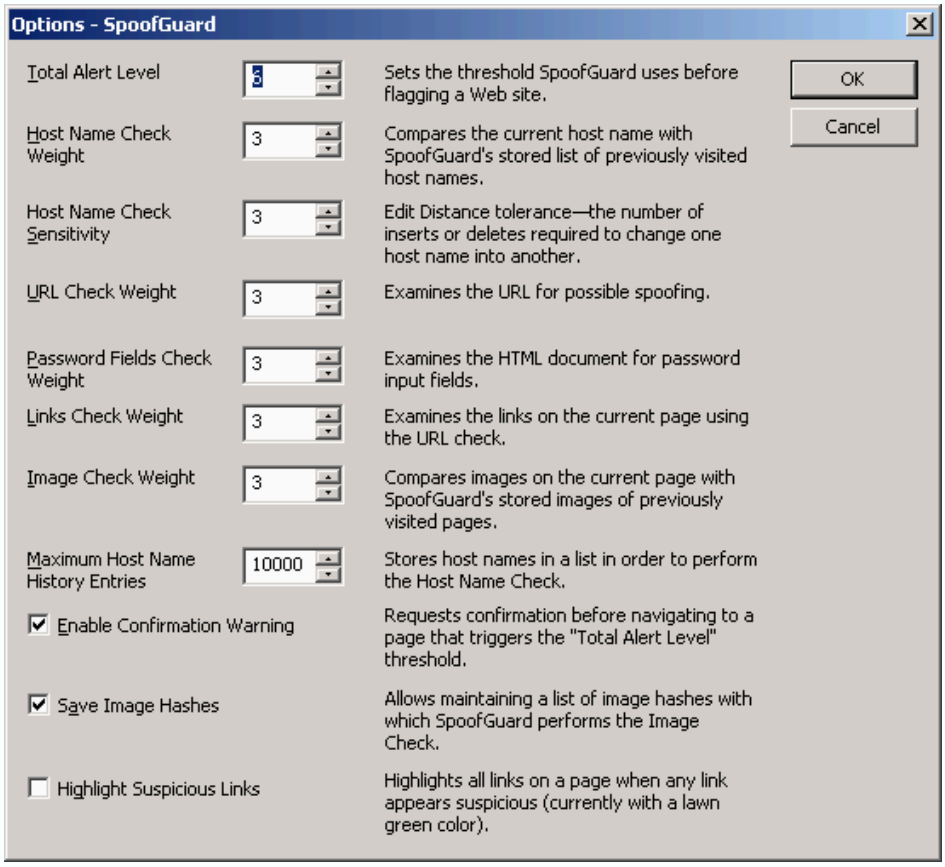

Figure 3. SpoofGuard configuration pop-up

words, whereas AutoComplete encrypts them with a known key that is stored on the user's computer. Second, many Internet Explorer users turn off AutoComplete either due to frequent pop-up windows that ask for the users permission before storing the data, or for privacy. Therefore, SpoofGuard is more secure while effectively serving a larger user base.

Redirects Currently, a page that redirects to another page may cause SpoofGuard to flash yellow or red or pop-up a small post data warning box, depending on user configuration. In the next version of SpoofGuard, we plan to recognize redirects in html pages more effectively and eliminate these spurious warnings.

5 Evaluation

We evaluated the effectiveness of our plug-in using several criteria. First, does the plug-in detect the sample spoofs found in the wild? Second, is the false alarm rate sufficiently small? Third, how difficult is it to write a spoof page that is not detected by our plug-in? Finally, how does our plug-in affect browser performance. We discuss each of these below.

5.1 Detection of spoof attacks

In addition to debugging tests to make sure each SpoofGuard measurement works properly, we evaluated SpoofGuard's overall effectiveness by testing it against fourteen actual spoof web pages sent to us by the U.S. Secret Service. Nine of the fourteen pages are spoofs of eBay's sign-in page. Two spoof pages purport to be "identity and billing verification" pages that request a large amount of personal information, such as eBay username and password, residence information, credit card information, ATM card PIN, bank account routing number, social security number, mother's maiden name, date of birth, and driver's license number and issuing state. One spoof site states that because of "regular maintenance of our security measures, your account has been randomly selected for this maintenance," and requests a username and password. The last two suggest that the user could win a car if a username and password are provided.

We tested SpoofGuard on all fourteen spoof pages using the default settings and recorded all SpoofGuard messages for each page. Since SpoofGuard does not analyze html files stored locally on a user's computer, we set up a web server that hosted the fourteen spoof pages. Since most spoof web sites do not use https, our server used ordinary insecure http. Each page was retrieved from our web server by entering the url directly into the address bar in Internet Explorer. In order to force SpoofGuard to analyze each page, we cleared Internet Explorer's history and restarted the browser before loading each spoof page; this kept the history of previous tests from biasing the analysis of another spoof page on the same spoof server (our test server). Finally, in order to provide SpoofGuard with some information about the honest site, we visited eBay's web site and navigated to the sign-in page before each eBay spoof. At the honest eBay sign-in, we performed a mock sign-in using 'hello' and 'test' as the username and password, respectively. Although eBay did not accept these as a legitimate user name and password pair, they were recorded by SpoofGuard, which was all that we needed for the test.

The results of our test were:

- All fourteen spoof pages have password input fields and SpoofGuard successfully noted this. SpoofGuard also noted that the form submissions were insecure because the pages were retrieved from our web server without using secure http (https).
- All fourteen pages include inlined images, such as the eBay logo, that are retrieved directly from eBay servers. These images were already in SpoofGuard's image file as a result of the initial navigation to the honest site. In the test, SpoofGuard correctly noted that the spoof pages with eBay images matched those images from the honest eBay site.
- We performed a mock sign-in on the spoof pages to test SpoofGuard's outgoing password check. For each page, we used 'hello' and 'test', the same pair used on the honest eBay site in the initialization part of the experiment. SpoofGuard successfully identified the user name and password from the honest site and popped up a warning to the user, as shown in Figure 4.

We believe that the SpoofGuard image check and outgoing password check are important strengths, since together these checks stop outgoing data and they are not redundant with information that may be found in the browser address bar, status bar, or rendered html. While the image checker's hashing algorithm can be improved to detect slight modifications to the images, the current checks successfully catch spoofs observed in the wild.

5.2 False alarm rate

The false alarm rate depends in part on how frequently the user establishes new accounts and how frequently the user clears the browser history cache. We have used SpoofGuard ourselves over several weeks. With default settings, there are occasional spurious yellow lights while browsing, and sometimes the first use of a legitimate site with user name and password input will trigger a false post warning. Many of the unnecessary warnings are the result of frame or redirection problems (noted in section 4.3) that we expect to resolve in the next version

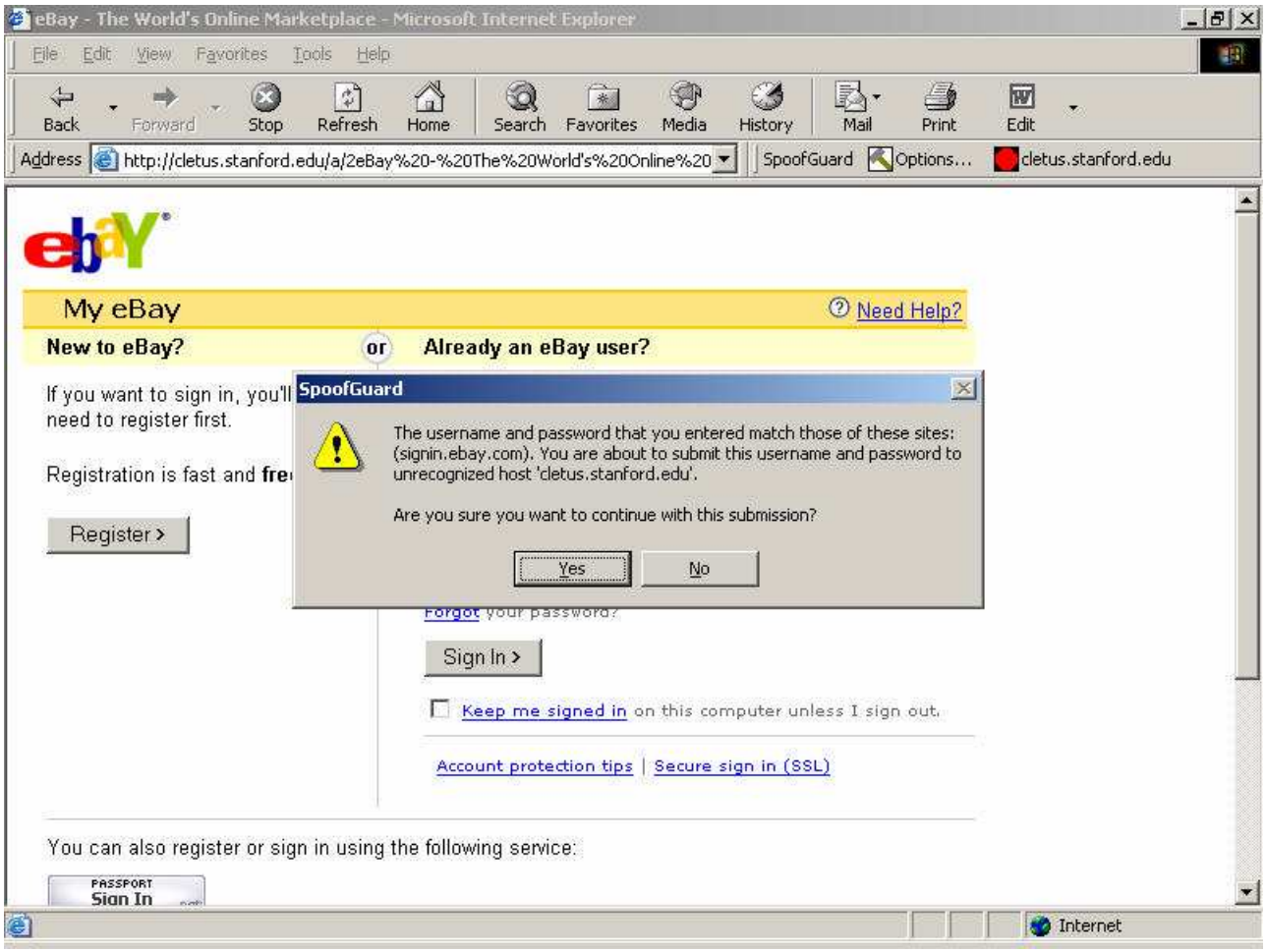

Figure 4. SpoofGuard detects honest user name and password on spoof site.

of SpoofGuard. If the user opens a new account, and intentionally uses the same password as another account, this will also produce an unwanted warning. However, second and subsequent visits (without clearing the history cache) do not lead to additional false alarms for this situation.

5.3 Security

The solutions described in Section 3 are certainly not fool-proof. An attacker with a reasonable understanding of web-site construction and a day or two of time can circumvent our current tests. For example, here are simple ways of fooling the SpoofGuard password and image checks:

- Some of our tests compare user input on a particular page to passwords that the user used at previous sites. An attacker could fool these tests by breaking the password input field on the spoof page into two adjacent fields that would look contiguous to the user, but would cause our password comparison tests to fail. Similarly, javascript on the spoof page could encode post data sent from the page so as to defeat our post data tests.
- Some of our checks compare images (logos) on a spoof page to images that appear on honest pages. An attacker could defeat these tests by slicing an image into adjacent vertical slices and presenting these slices one next to the other. None of the individual slices would match images in the plug-in's database, but to the user the complete image would look authentic. This would defeat some of our image tests.

These limitations notwithstanding, our methods clearly make it harder for attackers to setup effective spoof sites. Given the extremely low level of sophistication we've seen so far in actual spoof attacks, it is difficult to predict how quickly phishers would respond to deployment of SpoofGuard or related methods. In addition, should more sophisticated spoof sites appear, the framework we have adopted in SpoofGuard can be extended with more sophisticated checks and defensive password management.

5.4 Performance

As SpoofGuard users, we have not noticed any performance degradation as a result of the browser plug-in. We attempted to confirm this subjective impression by making rudimentary measurements. We inserted timing checks at the browser BeforeNavigate2 and DocumentComplete events, measuring CPU usage and navigation speed for retrieving and processing each page. We compared performance with SpoofGuard installed to performance with a dummy plug-in containing only the timing checks. To try to account for the fact that network latency shows up in the timing numbers, we also ran a second round of tests with all pages in the cache. Although the measurements do not capture SpoofGuard timing with great accuracy, the numbers we obtained seem to support our belief that there is no noticeable degradation of user browsing experience.

The performance tests were carried out on TrafficMarketplace's list of 30 most visited sites, using a 1GHz Pentium III with 128MB of RAM, connected through a 10 Mbps Ethernet card. Retrieving pages over the network, it took an average of 779 milliseconds to navigate from one page to another without SpoofGuard installed, and 911 milliseconds with SpoofGuard. With pages in the cache, these numbers dropped to 484 milliseconds and 601 milliseconds, respectively. These measurements suggest that the sequence of checks carried out by SpoofGuard take on the order of 100–250 milliseconds on an older processor. Furthermore, the CPU usage was 30% without SpoofGuard, and 40% with SpoofGuard, although the variance in CPU usage was high while the variance in timing numbers was low. Overall, however, it seems safe to conclude that SpoofGuard does not impose a significant performance penalty. The non-expert users for whom the plugin is designed are unlikely to notice the computation overhead; they are particularly unlikely to notice the SpoofGuard overhead if their network connections are slow.

6 Server-side assistance

The techniques we have implemented and tested are designed to detect web-spoofing attacks without any cooperation from web sites that are spoofed. However, we could do much more with the help of e-commerce web sites. For example, the two methods suggested below add simple tags to honest web pages. The additional information gathered from honest sites can be used detect spoofs more effectively.

6.1 Mark form fields with confidentiality tags

The outgoing password check compares outgoing data to stored (and hashed) sensitive data sent previously to honest login pages. E-commerce sites can help SpoofGuard identify sensitive fields by marking them with an additional html attribute.

We propose adding a CONFIDENTIALITY attribute to the <INPUT> html element. For a sensitive field (password, username, creditcard) the html element would look like

<INPUT NAME="username" TYPE="text"
CONFIDENTIALITY="username">

where CONFIDENTIALITY is one of *username, pass-*

word, creditcard, SSN and possibly other pre-specified strings. The confidentiality attribute helps SpoofGuard determine how to process the field. Note that we do not use the NAME attribute to infer confidentiality so as to give the site complete freedom over its field names.

Confidentiality tags could improve the detection rate and reduce the false alarm rate. Specifically, if data not currently tracked by SpoofGuard is marked confidential, SpoofGuard will be able to warn the user when this confidential data is exposed. If confidentiality tags become widely used, then SpoofGuard could become less likely to spuriously track information that is not confidential, reducing the likelihood of false alarms. The proposed html confidentiality tags may also have other uses beyond spoof identification. For example, a kiosk browser could close a window and flush short-term cookies after a certain idle time if entry to the site involved confidential data.

6.2 Image tagging

The image check described in Section 3.2 is useful in identifying spoof login pages since these pages need to reproduce the look-and-feel of the honest site. We already mentioned in Section 3.2 that e-commerce sites can help strengthen this mechanism by choosing images that can be hashed robustly. In addition, e-commerce sites can help make this test stateful rather than stateless. To do so we propose adding a new attribute to the IMG element in an html page. The attribute enables honest sites to identify images on their login page that are not supposed to appear on login pages outside the site. For example, the IMG element pointing to the eBay logo on eBay's login page would look like:

<IMG SRC=http://ebay.com/login-logo.gif SPOOFGUARD>

The SPOOFGUARD attribute indicates that if this image appears on a non-ebay web page requesting sensitive user input then it is likely the page is a spoof. Potentially, the SPOOFGUARD attribute could include a value (low, high) thus giving the site administrator some control over the score added to the total spoof score for the page. We note that this attribute should only be used on sensitive html pages such as a login page.

Next, we describe how the SpoofGuard plug-in would use this attribute. The difficulty is in ensuring that this attribute is not used for denial of service. We slightly reorganize the plug-in's image database. Each record in the database is as follows (one record per image):

image-hash	(d_1, f_1)	(d_2, f_2)	...	(d_s, f_s)

where d_i is a domain on which this image was found, and frequency f_i is the number of times the user visited the page. The plug-in either maintains the frequency value itself (adding one every time the page is loaded) or uses the browser's history file to compute it. Only images referenced with the SPOOFGUARD attribute are stored in the database.

When the browser downloads a login page containing an image whose hash is in the database it does the following:

- Check to see if the page's domain is in the list of domains associated with the image. If so, let F be the frequency for the domain. If not, set F to 0.
- Let M be the maximum frequency in the image's record. The test return the value $p = 1 - F/M$.

To see how this works, consider the eBay login page and consider an image marked with SPOOFGUARD on that page. Now, consider the record in the plug-in's image database corresponding to this image. Most likely, the eBay login page will have the highest frequency in the image record. Consequently, the eBay login page will have $p = 0$ indicating that it is not likely to be a spoof. Other pages containing this image with a SPOOFGUARD attribute (either set up by an attacker or by someone attempting to DoS eBay) will have a much lower frequency and therefore result in a p value close to 1. This technique prevents abuse of this test for denial of service.

6.3 Password hashing and site-specific salt

Users often use the same password at many different sites. For example, the same password may be used for an E*Trade account as for a newspaper site. We can combat this problem using site-specific password salt. Password salt, or other improvements of the standard password mechanism, also help with other security problems. In particular, when attackers break into a low security site they often try the recovered username/password combinations at various financial sites. As a result, a web site implementing proper security policies suffers when other sites do not apply recent patches and store passwords in the clear.

Passwords at one site can be made independent of passwords at other sites by adding a new SALT attribute to the html <INPUT> element. This attribute lets a site specify per-server salt; per-user salt is not possible since it is supplied before the user is identified. With this new attribute, password fields would look like:

<INPUT NAME="pwd" TYPE="password" SALT="E*Trade">

where the site developers ensure that their salt is unique to their site. For example, one could use the domain-name as a salt.

When processing a password field the browser first computes $E_{pwd}[salt]$, where E is block cipher, pwd is the password entered by the user, and $salt$ is the salt from input html element. The browser transmits the resulting value rather than the user's password. If the

salt attribute is not present in the html page, the browser uses 0 as the salt. The main point is that with the block cipher, it is hard to compute $E_{pwd}[X]$ from $E_{pwd}[Y]$ for $X \neq Y$. Consequently, a newspaper break-in will not compromise an E*Trade password. Note that this is similar to challenge-response authentication, except that each site uses a fixed and unique challenge rather than a random challenge. This way the site need only store in its database a hash of the submitted password value rather than the plaintext submitted password.

One difficulty in deploying site-specific salt is that all browsers must be simultaneously modified and web sites must re-authenticate their users after this mechanism is deployed. Another problem is that this mechanism itself is susceptible to spoofing. As presented, a spoof site need only contact the honest site to obtain the site-specific salt, then pass the same salt on to the victim. This will cause the victim's browser to send the spoof site the exact password needed to gain access to the honest site. Although we have not done a thorough study, site-specific salt may still be useful when the request comes over https and the certificate check establishes a reliable association between the salt and the requesting domain. With this limitation, site-specific salt will produce distinct passwords for distinct sites, and prevent a phisher who sets up an insecure (i.e., non-https) site from obtaining a password associated with a more secure (https) site.

7 More Speculative Techniques

We describe a few techniques that might be useful in combating spoof sites. We did not experiment with these since we consider them to be more speculative at the moment.

Collaborative Methods. Several projects [Bri03] use collaborative methods to identify spam email. A similar mechanism might apply to blocking spoof sites. Consider a user who uses our plug-in, but ignores the warning issued by the plug-in when visiting a site. The user enters his identifying information, submits the data, and then realizes that he just entered private information on a spoof site. At that point the user might want to alert the authorities as well as alert other users to avoid the site. By providing a "send alert" button in our browser plug-in the user could notify a central server that the current page is a spoof. If enough users identify the page as a spoof the server could alert all plug-ins to block the page. This might dramatically reduce the number of users who get duped by the spoof site.

We consider this method to be speculative for two reasons. First, user's who are duped by spoof sites are also likely to be unaware of the "send alert" button and its function. Second, this mechanism could potentially be used to launch a denial of service attack against an honest site.

Search engines to the rescue. Spoof sites are often direct copies of pages on the honest site. Therefore, when viewing a sensitive page (a page that requests a user password), our browser plug-in could do a Google search on some key phrases or links on the page. If a page similar to the current page is found at a different domain, the plug-in would increase the page's spoof-score.

We consider this method to be speculative since pages are often cached at various sites on the web and it would be difficult to distinguish a spoof from a cached page. In addition, if every browser in the world automatically issued a Google query for every password page it encounters, the resulting traffic would likely overwhelm Google. There is also no business incentive for Google to support such a service.

Forensics. Suppose internal tests at E*Trade indicate that a user's password has been compromised. E*Trade suspects that this is the result of a spoof E*Trade site. They wish to quickly determine where the site is. One option is to examine the user's history file since it contains all the sites the user visited recently, including the spoof site. However, a well-minded user would likely refuse to hand over the browser's history file due to privacy concerns. To reduce the user's exposure, SpoofGuard could keep track of sites where the user entered a password identical to his E*Trade password. Only those sites would be handed over to E*Trade.

We consider this method to be speculative since most spoof sites are active for only a few days. Most likely the process required for obtaining data from the user would take more time than that. Nevertheless, the problem of quickly locating spoof sites is important and deserves attention. We may experiment with using web crawlers for this task in the future.

8 Conclusion

Most of the $37 million increase in losses from Internet fraud observed between 2001 to 2002 has been attributed to web spoofing [Von03]. While web spoofing (or phishing) may become more sophisticated in the future, we propose a set of methods that appear effective for the kind of simple attacks observed by law enforcement and affected companies. SpoofGuard uses a combination of stateless page evaluation, stateful page evaluation, and examination of outgoing post data to compute a spoof index. When a user enters a username and password on a spoof site that contains some combination of

suspicious url, misleading domain name, images from an honest site, other measures discussed in section 3, and a username and password that have previously been used at an honest site, SpoofGuard will intercept the post and warn the user with a pop-up that foils the attack. We have tested SpoofGuard with actual attacks found in the wild and found the mechanisms generally unobtrusive and effective. While technically savvy Internet professionals probably do not need SpoofGuard themselves, there are many less sophisticated users who may benefit from this tool.

In order to effectively reduce the impact of Internet fraud based on web spoofing, SpoofGuard must be distributed and deployed, or the mechanisms tested here must be adopted by browser companies and integrated into standard browser security mechanisms. While the initial tests from our research effort are promising, we expect to continue to refine SpoofGuard and subject the components of our method to more rigorous statistical testing. Especially if some of the server-side methods described in section 6 are adopted by companies subject to web spoofing fraud, such as EarthLink, Citibank, Morgan Stanley's Discover unit, eBay, PayPal, banks and state lotteries [Ber03], we believe that SpoofGuard methods will reduce fraud. In addition to reducing the direct loss figure mentioned above, good protection against web spoofing would significantly reduce customer support costs.

A second consequence of deploying the methods described in this paper is that phishers will have to work harder to spoof web users into revealing sensitive information. As discussed in section 5.3 and elsewhere, many of our tests can be circumvented by relatively simple modifications to spoof pages. Like virus detection and spam filtering, we expect that any serious effort to combat web spoofing will lead to more sophisticated spoofs and the need for more sophisticated defenses. As mentioned throughout the paper, the methods currently implemented in SpoofGuard can be improved. Individual page tests can be improved, more page test can be added, and the formula for computing the spoof index can be refined. Furthermore, if e-commerce sites act on their concern about the problem, server-side techniques offer significant promise for combating web spoofing.

From a broader perspective, web spoofing takes advantage of the unauthenticated email and weak web-site authentication. As a Tumbleweed Communications web site regarding phishing [Tum03] points out, one countermeasure is "the use of digitally signed email to protect against phishing hacker attacks and spam email." While this is certainly true, digitally signed email has been technically feasible for many years, yet the adoption rate remains small. Strong web site authentication could also eliminate web spoofing. If challenge-response methods, for example, were widely deployed, then a spoof site authenticating a user would not have any way to impersonate the user on the honest site. In this sense, SpoofGuard helps patch over a weakness in current web practices that could be solved more effectively by stronger known technology. However, the history of the Internet suggests that once a convention is widely adopted, it is very difficult to introduce new standards.

Acknowledgments

Thanks to Alissa Cooper, Greg Crabb, Tom Pageler, Robert Rodriguez, and Chris Von Holt.

References

[Ber03] Tara Siegel Bernard. Citigroup's logo used in identity-theft attempt. SmartMoney.com, August 18, 2003. http://www.smartmoney.com/bn/ON/index.cfm?story=ON-20030818-000809-1407%.

[Bri03] Brightmail inc. http://www.brightmail.com, 2003.

[CNN03] 'phishing' scams reel in your identity. http://www3.cnn.com/2003/TECH/internet/07/21/phishing.scam/, July 22, 2003.

[Cox03] Mike Cox. Fraudulent emails - thieves intend to steal your personal information 6/2003, 2003. Posting from the Michigan Attorney General, http://www.michigan.gov/ag/0,1607,7-164--70494--,00.html.

[Din03] Theo Van Dinter. Spamassassin, 2003. http://useast.spamassassin.org/.

[EY01] S.W. Smith E.Z. Ye, Y. Yuan. Web spoofing revisited: Ssl and beyond, 2001. http://www.cs.dartmouth.edu/~pkilab/demos/spoofing/.

[FBDW97] Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web spoofing: An internet con game. In *Proceedings of 20th National Information Systems Security Conference*, 1997.

[FBI03] FBI web spoofing warning, 2003. http://www.fbi.gov/pressrel/pressrel03/spoofing072103.htm.

[HF03] Katie Hafner and Laurie J. Flynn. E-mail swindle uses false report about a swindle. NY Times, June 21, 2003.

[Pax99] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23–24):2435–2463, 1999.

[Sno03] Snort: The open source network intrusion detection system, 2003. `http://www.snort.org/`.

[Tum03] Tumbleweed Communications. Digitally signed email to protect against phishing hacker attacks, 2003. `http://www.tumbleweed.com/en/solutions/phishing.html`.

[VKJM00] R. Venkatesan, S.-M. Koon, M. H. Jakubowski, and P. Moulin. Robust image hashing. In *Proceedings of the International Conference on Image Processing*, 2000.

[Von03] C. T. Von Holt. Resident Agent In Charge, US Secret Service, San Jose, CA. Private communication, 2003.