Breaking out of the Browser to Defend Against Phishing Attacks

D.K. Smetters and Paul Stewart
Palo Alto Research Center
3333 CoyoteHill Road
Palo Alto, CA 94304
{smetters,stewart}@parc.com

Abstract

Current approaches to phishing prevention are focused on the web browser and the user's interaction with it. We present a new approach to allowing users to interact reliably and securely with high-value and sensitive web sites, using *protected links* – a user-customizable set of secure bookmarks, that not only identify but also authenticate their targets. By combining visually distinctive display to prevent spoofing, digitally signing these bookmarks to protect them from tampering, and whitelisting of allowed bookmark providers to ensure correctness, we ensure users end up with an intuitive interface for accessing the sites they intend. We have implemented a prototype protected links system, and evaluated its usability with a small study of potential users with positive results.

1 Introduction

Phishing web sites attempt to extract sensitive information from users by masquerading as a site they trust. Defending against such attacks requires preventing the user from revealing sensitive data to the attacker that he can use.

The core problem posed by phishing attacks is that the user is sending sensitive data to someone other than *whom she intends*. We refer to this as the "mismatch problem". Unfortunately, current phishing defenses do not directly address the mismatch problem; instead they try to provide further feedback about web sites designed to enable users to solve it for themselves.

Blacklisting-style approaches try to help the user detect and avoid "bad" web sites (*e.g.,* (Microsoft Corporation, 2008; Mozilla, 2008a)) on the simple heuristic that such "bad" sites cannot possibly be who the user intends to communicate with. However, in an open universe of web sites, where users may legitimately wish to access almost any of them, it is fundamentally too hard to distinguish accurately between "obscure" or "new" and "bad". False negatives and positives are common (Zhang et al., 2007).

Browser cues designed to help users authenticate "good" web sites (*e.g.,* the SSL lock icon), and best-practice behavioral heuristics ("always type the URL directly into your web browser") aim to help users determine whether a site they have accessed is indeed the one they intended. Unfortunately, they do not succeed – browser cues are often ignored (Friedman et al., 2002; Dhamija et al., 2006; Whalen et al., 2006), and can be spoofed (Ye et al., 2005), and heuristics can fall to increasingly sophisticated attacks against DNS (*e.g.,* (Trend Micro, 2008; Wikipedia, 2008a)), where even a user who heeds advice and types in their own URLs can be misled.

1.1 Attacking the Mismatch Problem

We attack the mismatch problem directly. To do this, we need to allow users to intuitively indicate who they intend to communicate with in a way that can be securely interpreted, and hence ensured, by systems operating on their behalf.

Standard web browsers cannot do this; they are designed to communicate with as many different web sites and services as possible. When a user attempts to access her bank by clicking on a link or accessing a bookmark, the browser can only determine what site that (possibly incorrect or malicious) link actually refers to – not what site the user *believes* it refers to.

Instead, consider the following: for each high-value online site or service the user regularly accesses, he obtains what is in effect a secure bookmark. We refer to these as *protected links*. These links are represented to the user by well-known trademarks, logos, and standard (registered) business names, not URLs

which bear varying resemblances to the site of interest to the user.

Unlike traditional bookmarks, protected links identify their targets not only by URL, but also by the public keys of the servers involved. Accessing that service via a protected link authenticates the service on the user's behalf; protected links cannot be fooled, even in the face of attacks against the user's DNS. They ensure that the user only interacts with the service they intend.

A user obtains a protected link for each high-value service she regularly accesses. Links are obtained from a service provider's web site as part of the user's initial service enrollment, or from another entity with whom they have an established trust relationship (*e.g.,* their antivirus vendor). Link legitimacy is ensured by a form of *whitelisting*, preventing a user from installing a malicious link. Each protected link is signed, and any attempt (*e.g.,* by crimeware) to alter a protected link is detected and blocked. Finally, a user's own set of protected links is displayed together in a visually distinctive, personalized manner to prevent spoofing.

From the user's point of view, these protected links appear almost as a set of dedicated "applications", each intended to perform a particular task such as interacting with a particular bank. Users do not need to learn a complex set of rules to detect attacks against their high-value sites, only the simple heuristic that "to access my bank account online, I must use my online banking application". Actually installing special-purpose applications to connect with individual web sites would be ineffective, costly, and difficult. Protected links are designed to give the user the intuitive experience of interacting with such a special-purpose application, while preserving a simple web development and deployment model.

Protected links solve the mismatch problem, at least for high-value web sites. They are designed to be user-friendly and easily deployable. Results of an initial usability study support the idea that they are intuitive and easy for end users to understand.

In the remainder of the paper we analyze the security requirements for an effective secure bookmarking system in Section 2, and present our design for a system that meets them (Section 3), and a prototype implementation of that system (Section 4). Our user evaluation is described in Section 5. We then describe related work in Section 6, and conclude.

2 Security Requirements

Any secure bookmarking system attempting to solve the mismatch problem, *i.e.,* to ensure that the sites users access are always the ones they intend, must meet the following (informal) security requirements:

Identifiability: users should be able to easily determine the target of a protected link and match it to their intent.

Distinguishability: users should be able to recognize their protected links, even in the face of attempts by malware to provide visually similar alternatives.

Correctness: the actual target of a link must match the user's understanding; in particular, the process for obtaining new links must defend against attempts to insert malicious links.

Accuracy: accessing a service via a protected link must result in the user communicating only with its intended target. Attacks that prevent correct communication, such as manipulation of the user's DNS should result in an error.

Tamper Resistance: unauthorized modification of protected links must be detected and the link disabled.

In addition, it should be difficult to disable or delete protected links, as doing so might lead the user to default to less secure mechanisms of accessing her target.

3 System Design

We have designed a system architecture that meets the security requirements of Section 2. Our protected links system has three components:

- a set of *protected links* obtained by each user to access their high-value sites
- a *protected link toolbar application* which presents the available links to the user and enforces security policies over those links
- a *protected link browser*: a limited environment used to access the designated target of a protected link.

3.1 Protected Links

A *protected link*, or secure bookmark, securely identifies its intended target. It does so by registering to not only its target URL, but also the public keys or digital certificates of the (TLS-enabled) web servers authorized to provide that target content. This achieves our desired *accuracy* property; attempting to access a service via a protected link in the face of DNS manipulation will result in a connection to a malicious

Figure 1: Our prototype Protected Links toolbar.

server which does not possess the required private key, allowing the attack to be detected and the connection aborted. Every protected link is also digitally signed to provide *tamper resistance*.

Each protected link is identified to the user primarily by a logo, trademark or other graphic selected by the link provider; users do not have to interpret URLs, recognize domain names, or interpret digital certificates. This allows for easy *identifiability* of the target of a link. Corporations are typically much more mindful of their logos and other symbols of corporate identity than of the domains they use, which may be constrained by details of their network that users should not be asked to remember.

3.2 Protected Links Toolbar

A user's protected links are presented in the form of a *protected links toolbar*, an application which performs three critical roles.

Accuracy. First, it is the toolbar which ensures *accuracy*, by defending the user's protected links against alteration. It does so by verifying the signatures on each link prior to each use, supplemented by continual monitoring for attempts to change them or to add invalid links to the set. The user is prevented from visiting any link whose signature fails to verify. At the same time, because these links have user-interpretable semantics, we can warn the user about such attacks in intuitive terms ("An invalid web site is attempting to alter your protected link to Citibank to point to an unknown server. Operation not permitted.").

Distinguishability. Second, if a user is to access her high-value sites correctly via a protected link, she must first be able to *distinguish* her "real" protected links from attempts by malware to mislead her. This means that scattering protected links across a user's desktop is unlikely to provide good security; even if they are visually distinctive, small icons lost potentially in a crowd of distractors would be hard to tell from malicious look-alikes. Instead, we visually group all the user's protected links together in a bit of trusted screen real estate, in the form of a toolbar (an example can be seen in Figure 1), where only valid protected links are displayed.

That toolbar itself is more difficult to spoof in its entirety and can be monitored as part of, say, the antivirus software on the machine. Further protection can be provided by personalizing the toolbar to the user as part of installation (Dhamija & Tygar, 2005). For example, a user might install a certain background image or color scheme for her collection of protected links. Malware attempting to spoof the user's toolbar would need to also match that personalization.

Correctness. The final, and most critical role for the toolbar is to ensure *correctness* – to prevent the user from being misled into obtaining a malicious link that will lead them somewhere other than they intend.

Without analysis of what the user sees, such as the icon associated with a link, we cannot distinguish a malicious "mock Citibank" protected link from a legitimate one. To ensure that users only end up with correct links, we must limit who is allowed to publish such links. We do so by requiring such links be signed by an authorized entity, where "authorized entity" is determined by the vendor of the toolbar. It could be that vendor itself; or it could encompass a form of delegation similar to a certification hierarchy, where an organization would be authorized to sign protected links used to access its own sites.

In essence, we are *whitelisting* the high-value sites of the Internet. Whitelisting signed links, so that users must, say, obtain them from their service provider on enrollment rather than constructing them as with standard bookmarks, allows review by trained professionals to "vet" those small number of good sites for whom protected link status makes sense, and more careful construction of the links themselves to ensure they make sense to users.

Security of the protected link system therefore depends not only on the integrity and correct operation of the system components, but also on the set of public keys used to validate membership in the whitelist. This is a limited and defensible set of critical data,

much like that already relied on by a number of other security- critical systems (*e.g.,* root certificates for web browsers, virus definition files for antivirus products), which can be protected similarly.

3.3 Protected Link Browser

We use targeted and secured web links to interface with the secure areas of banks' and other institutions' standard web sites, via a standalone, stripped-down browser instance that is unconnected to the user's normal web browser (see Figure 2). Because it shares neither user data nor memory footprint with other browser instances (secure or insecure), protected links visited by the user are isolated both from the user's normal web browsing and from each other. Because it is at its heart a browser, it supports current web development models (and in fact can be deployed on top of existing SSL-protected web sites largely without change).

3.4 Deployment and Personalization

Our approach to protecting online interactions allows for incremental real-world deployment. In its simplest form, it can give value immediately, without any change required on the part of the institutions whose sites are the targets of protected links. As long as the secure portion of those sites supports navigation entirely via SSL, a protected link can easily be generated for each existing site by simply recording the combination of URLs and public keys used by that site's servers at a given point in time. Such links are *generic* – the same link information serves for any user attempting to access that site; as such they could be created and provided by the vendor of the protected link toolbar, rather than the sites themselves.

For fuller protection, a cooperating site could distribute its own protected links, and even *personalize* the links it distributes. Protected links can be personalized for the site they target, *e.g.,* by customizing the way they are rendered on the client to make them look more like a standalone application, or by having them target a special portion of the web site designed for such access.

A more interesting option is to personalize protected links for the user installing them. Such links could include credentials to enable that user to access the originating site – *e.g.,* simple cookies, or even client certificates and private keys.[1]

[1]As these credentials are to be used to access the site that distributed them, simplifying the system by having that site also generate the private key imposes little additional risk.

Personalizing protected links in this manner has a number of advantages: first, it forms a very simple (from the end user's point of view) means of distributing client certificates, usually an onerous process. Those client certificates allow further personalization, *e.g.,* by allowing the particular user/machine combination accessing a site to be recognized and authenticated prior to any user information entry; allowing early presentation of user-specific images or other customization to enable mutual authentication. These client certificates would normally not be sufficient to log the user into the target site on its own; a user name and password would still likely be required to protect the user from unauthorized access to her account by those in her immediate vicinity.

Finally, such personalized links would allow the target site to determine when a user came in via a protected link and and when they didn't. In its full form, online banks or other high-value sites simply would allow standard web browsers only access to the bank's public web site, or limited (*e.g., read-only*) access. Attempts to get at private account information would result in invitations to enroll in online banking and install a protected link.

3.5 Creating a New Link

The process of setting up a new protected link must be no more difficult than current online banking enrollment. This is not a high bar; enrollment in online banking or other high-value services has become a significant, high-effort process as providers attempt to protect themselves and their users. Users may also need to answer challenge questions or otherwise repeat an enrollment step each time they attempt to log in from a new machine.

We can easily use the same enrollment mechanism to set up protected links; instead of just establishing a user name and password, the end result of enrollment would also be a protected link file to install (*e.g.,* by double-clicking it).

3.6 Updating Protected Links

A final item to consider in deployability of such an infrastructure is evolution. Because protected links are simply portals into web sites, the "applications" they represent are infinitely evolvable. The only thing challenging to evolve are the links themselves - the URLs and the keys they contain. In the absence of key compromise, good coding and management practice would allow web sites to go almost indefinitely without having to update the contents of deployed protected links.

Figure 2: Opening a protected site in our prototype limited browser.

However, in case they do need at some point to update their information, it is simple to build an auto-update mechanism into protected links. Each link could contain information sufficient to access and authenticate a site responsible for updating it; the toolbar could check that site at regular intervals, automatically installing any (signed) updates that are available.

4 Implementation

We have implemented a prototype protected links system. Our initial toolbar application runs in Windows XP, but the remainder of the components of our system are either platform-independent (the protected links themselves), or available on any platform (the protected link browser and link generation tools).

4.1 Protected Link Toolbar

Our prototype toolbar can be seen in Figure 1; it simply displays the associated icon for each of the protected links the user has installed. A production toolbar might be visually more distinctive, and ideally be personalized by the user on installation (*e.g.,* with an image, or other "skin" (Dhamija & Tygar, 2005; Bank of America, 2006; Yahoo!, 2008)) to prevent spoofing.

On startup, the toolbar application looks for installed protected links in a user-specific directory. For each link found, it verifies that it is on the whitelist (*i.e.,* signed by an accepted provider) and has not been tampered with. For each verified link, it displays its associated icon, provided by the link issuer. For each link that fails to verify, the user is notified and offered the opportunity to delete (but not to use) the link.

When a user double-clicks a protected link, its contents are re-verified, and if valid, they are unpacked and its target accessed via the protected link browser.

A user can download a new protected link from an enrollment page on their provider's web site, or perhaps from a directory offered by the vendor of their toolbar software itself. In the former case, links can be *personalized* to that user (see Section 3.4), both in terms of appearance when executed and by containing credentials to authenticate that user/machine combination. A link is installed by double-clicking it, the extension given to link files (*.slink*) is associated with our toolbar application, which receives a message. The toolbar application then verifies the link and unpacks it into the user's link directory.

Our prototype implementation uses an extremely simple approach to bootstrapping trust – all protected links are signed by the same issuer, whose public key is embedded in the toolbar application. A production implementation would allow for more complex authentication hierarchies, and would store the system verification keys in a more secure location The integrity of those keys, and of the protected links system itself, would ideally be monitored (*e.g.,* by the user's antivirus software or the OS itself).

4.2 Protected Links

Our protected links (named *applicationname.slink*) are compressed archive files containing the components shown in Figure 3.

```
BankOfAmerica.slink
    |
    +-- main.png     (the application icon)
    +-- profile.zip  (the user profile)
    +-- xulapp.zip   (the XULRunner "app")
    +-- manifest.txt (the SHA1 digests of the
    |                 expected link contents)
    +-- manifest.sig (a digital signature
                      on manifest.txt)
```

Figure 3: The contents of a protected link archive. Descriptions to right.

Verification of a protected link requires first checking that *manifest.sig* contains a valid signature on *manifest.txt*, generated by a public whitelisting key known to the protected links infrastructure. Second, the *manifest.txt* file itself must be verified, by ensuring that the link archive contains all and only the files listed therein, and that the contents of those files match their `SHA-1` digests as listed (and signed).

Launching a protected link requires first reverifying it to ensure that it has not been tampered with, and then unpacking its various components into runnable form. Our current implementation first deletes any previously unpacked version of the link, except for any profile-specific cookies that might have been installed by previous executions of the protected link target, and then replaces them with a fresh copy of the contents of the protected link. This approach simplifies runtime verification, as we do not need to check that an existing unpacked file tree matches a link signature; however it does so at the cost of a slight delay on link execution (mostly due to unpacking of the link contents, not signature verification). The protected link browser is then invoked on the unpacked link.

This approach does open us up to a potential time-of-check *vs.* time-of-use vulnerability, where targeted malware could in theory modify our protected link contents after verification and extraction, but prior to execution. A production version of this software could address this risk by monitoring the unpacked files for modifications, and/or using a custom browser engine that verifies the files in the course of executing them.

Link Generation Our prototype links are generated by a simple `Perl` script, which takes a template set of protected link contents and customizes it for a specific site. The generator is given the target URL the link should open, and automatically retrieves the certificate for the web server serving that URL and stores it in the trusted certificate store for the new link. The generator can also be told about additional sites that it should trust (*e.g.,* secure caching proxies or advertising servers that provide some of the site's content), and insecure servers that the site relies on which it should silently ignore (but whose content will not be provided to the user). The generator is also given an icon file, which provides the user-visible "face" of the protected link, and can generate user-specific credentials, such as a digital certificate and private key, and/or an initial set of cookies. The generator then signs and packs the link content for distribution.

4.3 Protected Link Browser

To implement a protected link browser engine (see section 3), we use Mozilla XULRunner (Mozilla, 2008b), the stripped-down core of the FireFox browser designed to run standalone applications.[2]

Each of our "protected links" is in fact a full XULRunner "application". Such an application consists of a set of user profile data, including its own separate cookies file, browsing history, and its own certificate database (`cert.db` file) which tells that instance of XULRunner what certification authorities and web sites to trust (compressed into `profile.zip` above). The "application" (compressed into `xulapp.zip` above) also contains its own set of add-ons or extensions, and a separate set of browser "chrome". The resulting XULRunner instances execute separately from each other and from any running FireFox browser, and are not affected by anything in the user's standard FireFox environment or profile.. Such applications are quite small, about 40Kb in the compressed form in which we distribute them.

We configure each XULRunner "link" to connect to a particular URL, which causes that web site to open automatically when the XULRunner application is executed (*e.g.,* Figure 2). A setting in the profile information contained in each XULRunner "link" constrains these instances to only communicate using TLS; they will refuse to open any insecure links. We preload the certificate database of each "link" so that the corresponding XULRunner instance will only communicate with the set of web servers specified by that protected link – namely those associated with the web site of interest. Note that the certificate database allows connections only to servers with specified end-entity certificates; it does not have to indicate trust at the level of their signing certification authorities (which sign large numbers of certificates).

The fact that each link has its own certificate configuration that is downloaded with the link upon installation also gives us an easy mechanism for deploying and using client certificates; as noted in Section 3.4,

[2]FireFox 3 also includes this functionality, removing the need for a separate browser engine.

each protected link can be constructed specifically for its user, and could include a certificate and private key for that user to authenticate themselves to the site in question. This approach might be significantly simpler than traditional certificate enrollment mechanisms; as the certificate would be used only to authenticate to the target service, the fact that that service (the link issuer) would also issue the user's private key is not a particular concern.

Our default XULRunner application template has each link show only limited browser structure, or "chrome", allowing the user to navigate only within the site to which the protected link points; both the absence of a URL entry field and the underlying security configuration prevent the user from attempting to use it as a general purpose browser. Although the current implementation does not include forward and back buttons, relying on site-internal navigation only, such buttons could be added to either our default application template or individual protected links.

Our prototype currently uses the same browser "chrome" for all its protected links; however as this is a component of the link itself it could be customized. This would allow a site using a protected link to look more like a branded application than a standard browser window, although most of the content would still come over the web. Similarly, our prototype uses the protected link browser to access standard web sites, but a site could choose to provide specialized content tailored to this means of access. The result would still be considerably more flexible, updatable, and maintainable than a traditional "thick client" application.

Finally, because Firefox (and XULRunner's) add-on, or extension mechanism, installs extensions directly into the user profile. This means that extensions can be provided along with a protected link if the issuer of that link prefers, and the user's own extension selections cannot interfere with access to protected links. Because extensions can be installed by clicking on links to them, users could install them into a running protected link browser even with the limited chrome we provide. However, they would not persist across invocations of the link. More appealingly, link issuers could provide users with access to security-enhancing extensions to protect their online experience with that issuer (*e.g.,* (Rubenking, 2007)), without requiring the user to manually install additional software.

5 Evaluation

The design of our protected links system is based on the assumption that framing access to high-value web sites as secure bookmarks, or dedicated applications, would be significantly more intuitive to end users than asking them to monitor a plethora of browser-based security indicators. In order to test that assumption, and to assess whether users would be willing to "break out of their browsers" to interact with some web services, we performed a simple usability evaluation of our prototype.

Question	Response
Overall opinion	6 ± 0.6
Separate (non-browser) interface	5.7 ± 1.0
Would remember to use	6.3 ± 1.0
Confidence in security	5.6 ± 1.1

Table 1: User ratings of our protected links system (means and standard deviations). Scores are on a 7-point Likert scale, where 1 is most negative, and 7 is most positive.

Ten subjects were recruited from within our organization. They were selected to cover a range of knowledge of computers and security. We asked them a number of questions about their online behavior, and then explained to them the purpose and basic functions of our software.

Subjects were then asked to perform a set of simple tasks with the prototype: exploring the toolbar and opening one or more protected links to common phishing targets such as large banks, PayPal, or eBay; visiting a simulated enrollment page for a well-known bank and enrolling for a new protected link, which they were then to double-click to install, and open to access the "bank" site. Finally, we had them access a simulated phishing site, and attempt to install a "malicious" link, which was prevented by our software. At the end, they were asked for their opinions about the prototype. As our current prototype is not highly polished, we were interested more in their reaction to the idea as a whole than our specific implementation.

All but one of our subjects use or have used online banking, and most pay bills online and use PayPal and eBay. All reported familiarity with phishing attacks. Rating themselves on a seven-point Likert scale, where 1 is "no experience" and 7 is "expert", our subjects on average rate their experience with computers as 5.4 ± 0.9, and with security 3.4 ± 1.3. Their reported level of concern with online threats such as phishing and identity theft was 5.2 ± 1.4 (where 1 is "no concern" and 7 is "extremely concerned").

Subjects had a positive overall impression of the protected links approach (results shown in Table 1). While we were concerned that accessing high-value sites through a specialized interface instead of through the usual web browser might seem onerous, the ma-

jority of our subjects responded very positively to it. One subject reported that the use of a separate toolbar gave them an "extra level of confidence". Subjects also believe that they would be likely to remember to use a protected links toolbar, even if their services still offered less secure password-based login.

Confidence in the security of the toolbar mechanism itself depended critically on branding – if the user obtained the toolbar from a vendor they trusted and it was branded appropriately, they had high confidence that it would ensure their secure access to their intended targets (Table 1, "Confidence"). Without appropriate branding, confidence decreased. Similarly, subjects rated themselves as much more likely to install this software if it came from a trusted source, such as their bank, their firewall or antivirus vendor, or Google. Interestingly, subjects varied as to which of these sources they found most trustworthy.

These results suggest that our general approach is indeed sound, intuitive and appealing, but underscore the importance of branding in establishing initial trust. Subjects also reported to us that improving the feedback from the toolbar about its operation and security would also improve their confidence.

6 Related Work

In this section, we review other approaches to phishing defense, and discuss additional related work.

Anti-Phishing A large fraction of defenses against phishing fall into one of three classes. *User education:* train users learn to identify "bad" (phishing) sites, and to follow best practices designed to help ensure they end up at their intended web targets (Kumaraguru et al., 2007; Sheng et al., 2007; Srikwan & Jakobsson, 2008). *Blacklisting:* automatically recognize "bad" sites, and warn users about them or, if sufficiently confident, prevent them from visiting those sites entirely (Sutton, 2008; Microsoft Corporation, 2008; Mozilla, 2008a; Moore & Clayton, 2007; Zhang et al., 2007; Herzberg & Gbara, 2004; Close, 2006). *Improved User cues:* improve the technology of web browsers and/or web sites to help users distinguish "bad" sites from legitimate ones (Dhamija & Tygar, 2005; Bank of America, 2006; Chou et al., 2004).

Many of these approaches have significant limitations. Blacklisting approaches suffer from frequent false negatives and positives (Zhang et al., 2007), and their reports are often ignored by users (Wu et al., 2006a). Similarly, improved user cues are also often ignored or misunderstood (Schechter et al., 2007; Jackson et al., 2007), or depend on mechanisms that are insecurely implemented (Vamosi, 2007; SecuritySpace, 2008).

Secure Bookmarks Protected links are at their core *secure bookmarks*, by which we mean bookmarks that not only identify but also authenticate their targets. Though there exist a number of systems for synchronizing user bookmarks between machines (*e.g.,* (Google Labs, 2008; Project, 2008)), their interest in security is limiting unauthorized access to a user's bookmark collection, not protecting bookmark integrity, or assuring correct access.

In essence, establishing trust by linking directly to the public key or certifier of a web target is a simple mechanism for building a *localized web of trust* for the Internet. There are other systems which remember and authenticate the public keys of web servers on users' behalf, *e.g.,* to control the release of cookie information (Masone et al., 2007; Karlof et al., 2007a), to authenticate the *same origin policy* for controlling web site behavior (Karlof et al., 2007b), or to indicate to users whether they are interacting with a web site with whom they have a preexisting relationship (Close, 2006; Yee & Sitaker, 2006).

Identifiability and Distinguishability Much of the design of our protected links system centers around *identifiability* – ensuring that users can recognize the target of a link, by using logos and other well known marks, and *distinguishability* – visually separating, and personalizing a user's links to prevent spoofing.

Efforts to incorporate logos into digital certificates to increase users' ability to understand them (Santesson et al., 2004) adds support for our approach of identifying protected link targets to users by logo. However, we make that logo the primary interface element the user interacts with, rather than embedding it into the technical details of a connection she's already made.

The risk of spoofing of security indicators is well known (*e.g.,* (Ye et al., 2005; Wu et al., 2006b)). Personalization of sites and security components is one approach to make spoofing more difficult (Dhamija & Tygar, 2005; Bank of America, 2006; Yahoo!, 2008). However, attempts to personalize sites to allow users to authenticate them appears to fail in practice – users do not notice the absence of such security indicators even when they have been trained to expect them (Schechter et al., 2007). As our security components are *functional* – users click them to get access to the sites they need – rather than merely providing feedback, we think it more likely that users will notice their absence or warnings about their manipulation. This is particularly true if they were fully deployed as the only means of accessing one's high-value sites.

Internet Whitelisting Whitelisting of protected links – accepting such links only if they are signed by a

trusted third party – has a certain similarity to digital certificates, and in particular so-called "extended validation" (EV) certificates where the certificate signer "vets" the holder and contents prior to issuing the certificate (Wikipedia, 2008b). However, there are two major differences. First, what is being signed here is a bookmark, something a user can understand and interacts directly with in order to perform their intended task, rather than a digital certificate, something which is buried in the details of a security protocol. The result is that protected links are significantly more intuitive.

Second, by isolating our protected links in a separate toolbar, we can perform binary whitelisting – links that do not meet the whitelisting criteria simply do not appear. Extended validation (EV) or other SSL certificates are used to authenticate the only small percentage of connections a user makes with their web browser which are secure. The browser cannot prevent connections to sites without EV certificates. This leaves the user to figure out on their own what sites ought to have what sort of certificates, and what to do if they don't. Again, our policy is much simpler and easier to understand.

Browser Security From the point of view of the user, their protected links effectively act as *site-specific browsers* (SSBs) (Finkle, 2007), a convenient user interface fiction that presents a web site to a user as if it were a separate application. Standard SSBs run in separated browser instances to prevent errors in one site from crashing another; we also do so to protect sites from others' potential malicious behavior. A number of recent products have attempted to achieve the same end through *in-browser sandboxing* – accessing individual web sites within a protected "bubble" within the same browser, and removing all effects of that visit from the machine after it is complete (Green Border Technologies, 2008; Rubenking, 2007).

These and other tools designed to protect the user and their data from malicious web sites, *e.g.,* password managers (Yee & Sitaker, 2006; Wu et al., 2006b; Ross et al., 2005), and antikeylogger software (Rubenking, 2007) are complimentary to our approach. In fact, as many of them are packaged as FireFox extensions, they can be deployed transparently as part of our current prototype, by incorporating them directly into protected links.

7 Conclusion

In this paper, we have presented *protected links*, a new approach to phishing defense. Our approach combines secure bookmarks that authenticate their targets, with visually distinctive display to prevent spoofing of secure bookmarks, automatic verification of bookmark integrity to prevent tampering, and whitelisting of allowed bookmarks to ensure that users are not misled into bookmarking malicious sites. The result is a simple, intuitive mechanism for solving the "mismatch problem" and ensuring that users access the sites they intend. Results of an initial user evaluation suggest that our approach is indeed intuitive and appealing; though longer-running deployment studies are needed to truly evaluate its potential impact.

Acknowledgements

We would like to thank Victoria Bellotti and Brinda Dalal for help with our user evaluation, and Markus Jakobsson and Ignacio Solis for useful discussion.

References

Bank of America (2006). How bank of america sitekey works for online banking security. `http://www.bankofamerica.com/privacy/sitekey/`.

Chou, N., Ledesma, R., Teraguchi, Y., & Mitchell, J. C. (2004). Client-side defense against web-based identity theft. *NDSS '04: Proceedings of the 11th Annual 2004 Network and Distributed Systems Security Symposium.*

Close, T. (2006). Petname tool: Enabling web site recognition using the existing SSL infrastructure. *W3C Workshop on Transparency and Usability of Web Authentication.* New York City.

Dhamija, R., & Tygar, J. D. (2005). The battle against phishing: Dynamic security skins. *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security* (pp. 77–88). New York, NY, USA: ACM.

Dhamija, R., Tygar, J. D., & Hearst, M. (2006). Why phishing works. *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 581–590). New York, NY, USA: ACM.

Finkle, M. (2007). Site specific browser - webrunner. `http://starkravingfinkle.org/blog/2007/03/site-specific-browser-webrunner/`.

Friedman, B., Hurley, D., Howe, D. C., Felten, E., & Nissenbaum, H. (2002). Users' conceptions of web security: a comparative study. *CHI '02: CHI '02 extended abstracts on Human factors in computing systems* (pp. 746–747). New York, NY, USA: ACM.

Google Labs (2008). Google Browser Sync. `http://www.google.com/tools/firefox/browsersync/`.

Green Border Technologies, I. (2008). Greenborder. `http://www.greenborder.com`. Recently purchased by Google.

Herzberg, A., & Gbara, A. (2004). *Security and identification indicators for browsers against spoofing and phishing attacks* (Technical Report 2004/155). ePrint Archives.

Jackson, C., Simon, D., Tan, D., & Barth, A. (2007). An evaluation of extended validation and picture-in-picture phishing attacks. *Proceedings of Usable Security 2007 (USEC '07)*.

Karlof, C., Shankar, U., Tygar, D., & Wagner, D. (2007a). *Locked cookies: Web authentication security against phishing, pharming, and active attacks* (Technical Report UCB/EECS-2007-25). University of California at Berkeley.

Karlof, C., Shankar, U., Tygar, J. D., & Wagner, D. (2007b). Dynamic pharming attacks and locked same-origin policies for web browsers. *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security* (pp. 58–71). New York, NY, USA: ACM.

Kumaraguru, P., Rhee, Y., Acquisti, A., Cranor, L. F., Hong, J., & Nunge, E. (2007). Protecting people from phishing: the design and evaluation of an embedded training email system. *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 905–914). New York, NY, USA: ACM.

Masone, C., Baek, K.-H., & Smith, S. W. (2007). WSKE: Webserver-key-enabled cookies. *Proceedings of Usable Security 2007 (USEC '07)*.

Microsoft Corporation (2008). Windows internet explorer. `http://www.microsoft.com/windows/products/winfamily/ie/tax/default.mspx`.

Moore, T., & Clayton, R. (2007). Examining the impact of website take-down on phishing. *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit* (pp. 1–13). New York, NY, USA: ACM.

Mozilla (2008a). Phishing protection. `http://www.mozilla.com/en-US/firefox/phishing-protection/`.

Mozilla (2008b). XULRunner. `http://developer.mozilla.org/en/docs/XULRunner`.

Project, C. (2008). Chipmark. `https://www.chipmark.com/Main`.

Ross, B., Jackson, C., Miyake, N., Boneh, D., & Mitchell, J. C. (2005). Stronger password authentication using browser extensions. *Proceedings of the 14th USENIX Security Symposium* (pp. 2–2). Berkeley, CA, USA: USENIX Association.

Rubenking, N. J. (2007). Zonealarm ForceField beta. `http://www.pcmag.com/article2/0,2817,2186628,00.asp`.

Santesson, S., Housley, R., & Freeman, T. (2004). *Internet X.509 public key infrastructure: Logotypes in x.509 certificates*. IETF - Security Working Group, The Internet Society. RFC 3709.

Schechter, S. E., Dhamija, R., Ozment, A., & Fischer, I. (2007). The emperor's new security indicators. *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy* (pp. 51–65). Washington, DC, USA: IEEE Computer Society.

SecuritySpace (2008). Secure server survey. `http://www.securityspace.com/s\_survey/sdata/200712/certca.html`.

Sheng, S., Magnien, B., Kumaraguru, P., Acquisti, A., Cranor, L. F., Hong, J., & Nunge, E. (2007). Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish. *SOUPS '07: Proceedings of the 3rd symposium on Usable privacy and security* (pp. 88–99). New York, NY, USA: ACM.

Srikwan, S., & Jakobsson, M. (2008). Using cartoons to teach internet security. *Cryptologia*, *32*.

Sutton, M. (2008). A tour of the google blacklist. `http://portal.spidynamics.com/blogs/msutton/archive/2007/01/04/A-Tour-of-the-Google-Blacklist.aspx`.

Trend Micro (2008). Rogue domain name system servers. http://blog.trendmicro.com/rogue-domain-name-system-servers-5breposted5d/.

Vamosi, R. (2007). How phishers defeat online banking controls. `http://reviews.cnet.com/4520-3513_7-6762995-1.html`.

Whalen, T., Smetters, D., & Churchill, E. F. (2006). User experiences with sharing and access control. *CHI '06: CHI '06 extended abstracts on Human factors in computing systems* (pp. 1517–1522). New York, NY, USA: ACM.

Wikipedia (2008a). DNS cache poisoning. `http://en.wikipedia.org/wiki/DNS\_cache\_poisoning`.

Wikipedia (2008b). Extended validation certificates. `http://en.wikipedia.org/wiki/Extended_Validation_Certificate`.

Wu, M., Miller, R. C., & Garfinkel, S. L. (2006a). Do security toolbars actually prevent phishing attacks? *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 601–610). New York, NY, USA: ACM.

Wu, M., Miller, R. C., & Little, G. (2006b). Web wallet: preventing phishing attacks by revealing user intentions. *SOUPS '06: Proceedings of the second symposium on Usable privacy and security* (pp. 102–113). New York, NY, USA: ACM Press.

Yahoo! (2008). What is a sign-in seal? `http://security.yahoo.com/article.html?aid=2006102507`.

Ye, Z. E., Smith, S., & Anthony, D. (2005). Trusted paths for browsers. *ACM Trans. Inf. Syst. Secur.*, *8*, 153–186.

Yee, K.-P., & Sitaker, K. (2006). Passpet: convenient password management and phishing protection. *SOUPS '06: Proceedings of the second symposium on Usable privacy and security* (pp. 32–43). New York, NY, USA: ACM Press.

Zhang, Y., Egelman, S., Cranor, L., & Hong, J. (2007). Phinding phish: Evaluating anti-phishing tools. *NDSS '07: Proceedings of the 14th Annual Network and Distributed System Security Symposium*.