TrustBar: Protecting (even Naïve) Web Users from Spoofing and Phishing Attacks

Amir Herzberg[1] **and Ahmad Gbara**
Computer Science Department
Bar Ilan University

Abstract

In spite of the use of standard web security measures (SSL/TLS), users often fail to detect `spoofed` web forms, and enter into them sensitive information such as passwords. Furthermore, users often access the spoofed sites by following a link sent to them in a (fraudulent) e-mail message; this is called `phishing`. Web spoofing and phishing attacks cause substantial damages to individuals and corporations. We analyze these attacks, and identify that most of them exploit the fact that users are not sufficiently aware of the secure site identification mechanisms in browsers. In fact, it appears that even web designers are often confused about the need to securely identify login forms; we show several sites that prompt users for passwords using unprotected forms.

We derive several secure user interface principles, and present TrustBar, a secure user interface add-on to browsers. For protected web pages, TrustBar identifies the site and the certificate authority, using logos or at least names (rather than URL). For unprotected pages, TrustBar displays highly visible warnings. Early experimental results indicate that these mechanisms provide substantial protection, even for naïve and off-guard web users, from spoofing/phishing attacks.

1 Introduction

The web is the medium for an increasing amount of business and other sensitive transactions, for example for online banking and brokerage. Virtually all browsers and servers deploy the SSL/TLS protocols to address concerns about security. However, the current usage of SSL/TLS by browsers, still allows web spoofing, i.e. misleading users by impersonation or misrepresentation of identity or of credentials.

Indeed, there is an alarming increase in the amount of real-life web-spoofing attacks, usually using simple techniques. Often, the swindlers lure the user to the spoofed web site, e.g. impersonating as financial institution, by sending her spoofed e-mail messages that link into the spoofed web-sites; this is often called a *phishing attack*. The goal of the attackers is often to obtain user-ID's, passwords/PINs and other personal and financial information, and abuse it e.g. for identity theft. A study by Gartner Research [L04] found that about two million users gave such information to spoofed web sites, and estimate 1.2B$ direct losses to U.S. banks and credit card issuers during 2003; other estimates of yearly damage due to phishing and spoofing attacks are between $400 million [L04a] to $1 billion [G04b]. For examples of phishing e-mail messages, see [Citi04]. Spoofing attacks, mostly using the phishing technique, are significant threats to secure e-commerce, see e.g. [APWG04, BBC03] and Figure 1.

[1] Contact author; addresses: herzbea@cs.biu.ca.il and http://amirherzberg.com.

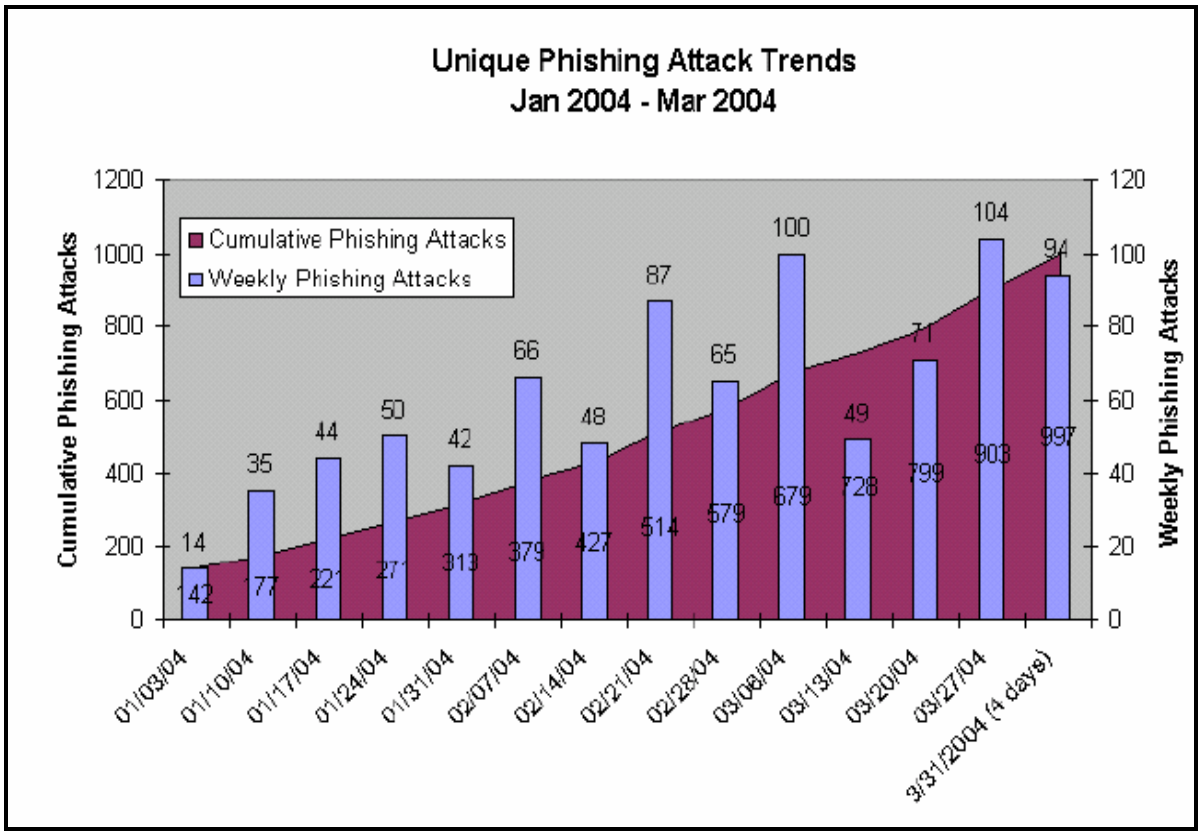

Figure 1: Phishing attack trends (source: [APWG04])

We investigate spoofing and phishing attacks and present countermeasures, focusing on solutions that protect naïve as well as expert users. Felten et al. first identified the web-spoofing threat in [FB*97]. However, in this work and all follow-up works [LN02, LY03, SF03, YS02, YYS02] so far, the focus was on attacks and countermeasures for knowledgeable and wary users, who check indications such as the URL (location) of the web site and the security lock (SSL/TLS) indicator. These attacks are not easy to deploy, as they require technical sophistication, and are sensitive to changes in browsers, operating systems and their configurations.

In fact, practical web-spoofing attacks deployed so far, do not use such techniques, or use just basic scripts and browser vulnerabilities e.g. to present fake location bar [APWG04, Citi04, SF9182]. Almost all of the many reported attacks left significant clues for the expert, attentive user, such as the lack of use of SSL/TLS (indicated by an open padlock icon, or by the lack of a padlock icon), and/or the use of a URL from a domain not owned by the victim web site. Such attacks will therefore succeed even using the countermeasures proposed in the existing literature [LN02, LY03, SF03, YS02, YYS02].

Still, these simple attacks are very effective [APWG04, BBC03, Citi04, L04]. We argue that this is due to several weaknesses in the user interface of the current popular browsers, and suggest simple extensions to browsers, that we believe will prevent many or most of these attacks. Our goal is to improve browser security-related user interface, and protect (even) naïve, inattentive web users. We defined and use the following secure user interface principles, based on previous works on secure usability [WT99, Y02, N93, JPH01, J00], but adapted and extended based on our analysis of web spoofing and phishing attacks, and on a survey we performed (described later).

The first principle establishes the importance of default settings related to security, such as the list of certification authorities `trusted` by browsers. This is a special case of the `unmotivated user` principle of [WT99] and of the `path of least resistance` principle of [Y02].

Secure UI Principle I: Security should be default, and defaults should be secure.
Default settings should provide adequate security, and only globally-trusted, obviously trustworthy parties may be trusted by default.

Even if defaults are secure, users may overrule them and disable security features, if they are overly disruptive to normal work and annoying, and in particular if their importance is not sufficiently clear to the users, and especially if disabling is easy. For example, many browsers, by default, warn users if an unprotected web page contains a form (whose contents will be sent in the clear). However, most web-forms, currently, are not protected; therefore this message pops up very frequently and almost all users disable it (`do not display this warning any more`). This leads us to the next principle:

Secure UI Principle II: Security must be usable to be used.
Users will avoid or disable security mechanisms which are hard to use, disruptive or annoying. Secure usage should be the most natural and easy usage.

The next secure UI principle follows from well-known user-interface design principles such as Nielsen's `recognition rather than recall` principle [N93]. Its relevance to security was noted e.g. by Grigg [G04], observing that users tend to `click thru` textual warning messages, e.g. the `unprotected form` warning mentioned above; see also [JPH01, section 3]. We also show that important sites such PayPal, Microsoft's Passport, Yahoo!, e-Bay and Chase, all ask users to enter passwords and/or other sensitive information in insecure web pages; this shows that not only users but also site designers and auditors did not notice the lack of protection (see Figure 5).

Secure UI Principle III: Alerts should wake-up.
Indicate security alerts, i.e. potential security exposures, using a clear, interruptive audio/visual signal.

Alerts should not only be noticeable and interruptive, they – and other security indicators, settings etc. – should also be clear and understandable to all (or at least most) users. This leads us to the third principle, which follows from usability `Blooper 35` in [J00]:

Secure UI Principle IV: `Cryptography` is in Greek.
Security and trust indicators and terms should be clear to all (naïve) users; avoid technical icons and jargon.

To prevent web spoofing, *TrustBar* uses a fixed are at the top of the browser window to display (validated) logos or names, of the web site owner and of the authority that identified the owner of the site. We recommend that commercial and organizational web sites present secure logo only in the TrustBar, and do so in *all* of their web pages. This will protect the integrity of all of the web pages, and increase the likelihood of users detecting a spoofed (sensitive) web page, by noticing the *lack* of the appropriate logo and/or credentials in the TrustBar.

We use cryptographic mechanisms to validate the logos and credentials in the TrustBar. Specifically, we bind the site's public key to the logo by signing both of them in a certificate, possibly as in [RFC3709], and use the (existing, widely available) SSL/TLS protocols [R00] to validate that the site has the private key corresponding to the public key.

Our solutions are simple and practical; we have implemented them as extensions of the Mozilla and FireFox open-source browsers [Mozilla]. Our code is open-source (and available from [MozDev]), and we hope that other browsers will have similar mechanisms soon, possibly using our code (one group already works on a version for Internet Explorer).

1.1 Related Works

Web Spoofing attacks were first presented in [FB*97]. There, and in most other publications on spoofing attacks, the adversary uses browser features to make it appear as if the browser displays the SSL-protected victim web page, while in fact it is displaying a cloned page. Some of these attacks are very simple, yet effective. For example, in one deployed attack [Citi04], the attacker opens two browser windows: a small one, which clones Citibank™ login screen and contains no status information or bars, inside a larger one, which is simply the regular Citibank™ web site. Such sites can be extremely convincing; we believe most users will not realize that they enter their password into a separate, insecure `pop-up` window, whose URL is not even displayed. In another deployed attack [APWG04, SF9182], a bug in many Internet Explorer™ browsers is exploited, allowing the attacker to cause false information to be displayed in the location bar. TrustBar can foil such attacks.

In Section 2, we show that users, and even designers, do not notice the lack of protection on sensitive web pages; we give several examples of sensitive yet unprotected web pages asking for passwords, including PayPal, Yahoo!, Chase, E-Bay, Amazon and Microsoft's Passport (see Figure 5). Attacks against Passport are also presented in [KR00].

Several works presented even more elaborate web spoofing attacks, using scripts and/or Java applets, to make it even harder for users to detect the cloning [LN02, LY03, SF03, YS02, YYS02]. These works also propose solutions, by disabling (important) browser functionalities, or using enhancements to the browser UI to make it hard or impossible for the attacker to display spoofed versions of important browser indicators [YS02, YYS02, LN02]. However, as noted by [YS02], these proposals rely on users' understanding their `signals`, which are (even) more elaborate than the existing location bar and SSL indicator. Therefore, these proposals are not appropriate for most (naïve, inattentive) users. In fact, as we argued above, we believe that the current indications of location and protection (SSL) are not sufficiently visible and significant to most users.

Our design strives for much more visible and easy to understand indications, allowing most users to detect when they reach a spoofed web site. Our approach is based on previous works on design of secure user interface for network applications, including TINGUIN (part of SEMPER, see [LPSW00]) and [WT99, HN98, H03, Y02, JP03].

We use `logo certificates`, possibly using the format in the draft standard `X.509 certificate extension for logotypes` [RFC3709], whose goals include identifying web sites and other purposes. In particular, Josang et al. [JPH01, JP03], and apparently[2] also Hoepman and Helme [HH99], suggested displaying the logo from such certificates in the browser interface, as protection against web spoofing. Our work extends their work substantially in functionality and security, in particular, displaying the name or logo of the certificate authority that identified the site and making TrustBar mandatory in all browser windows (which prevents attacks as in [LN02, LY03]). A significant difference is that our design is user-centric, and in particular we allow users to select the logo, image or text for protected sites. Close [C03] also proposed to allow users to select `pet-names` to identify web sites; that proposal was limited to textual names.

Adoption of TrustBar may provide incentive for protecting more web sites, compared to the current practice of protecting only very few web pages. As observed by [R00, BSR02] and others, protecting web pages with SSL/TLS involves substantial overhead; this may be a problem for some sites. This may motivate adoption of more efficient mechanisms to protect web pages, e.g. as in [BSR02].

This work focuses on preventing spoofed (fake) web sites, i.e. spoofing attacks; this includes the very common `phishing spoofing attacks`, where the user reaches the spoofed web site by following a link in a spoofed e-mail message he receives. See complementing countermeasures focusing on phishing attacks in [J05].

[2] We recently learned of [HH99]; since it is in Dutch, we could not read it.

1.2 Organization

In Section 2 we review web spoofing threat models, attacks and current defenses. From this, we derive a set of design criteria for protecting naïve and off-guard users against spoofing (and phishing) attacks, which we present in Section 3. In Section 4, we present the user interface design of TrustBar. In Section 5, we present results of an initial user survey, validating the deficiencies of the current browser security indicators and the improvements attained with TrustBar. We conclude with discussion of future work and recommendations for web-site owners, end users, and browser developers.

2 Web Spoofing: Threat Models, Attacks and Current Defenses

The initial design of Internet and Web protocols assumed benign environment, where servers, clients and routers cooperate and follow the standard protocols, except for unintentional errors. However, as the amount and sensitivity of usage increased, concerns about security, fraud and attacks became important. In particular, since currently Internet access is widely (and often freely) available, it is very easy for attackers to obtain many client and even host connections and addresses, and use them to launch different attacks on the network itself (routers and network services such as DNS) and on other hosts and clients. In particular, with the proliferation of commercial domain name registrars allowing automated, low-cost registration in most top level domains, it is currently very easy for attackers to acquire essentially any unallocated domain name, and place there malicious hosts and clients. We call this the *unallocated domain adversary*: an adversary who is able to issue and receive messages using many addresses in any domain name, excluding the finite list of already allocated domain names. This is probably the most basic and common type of adversary.

Unfortunately, we believe, as explained below, that currently, most (naïve) web users are vulnerable even against unallocated domain adversaries. This claim may be surprising, as sensitive web sites are usually protected using the SSL or TLS protocols, which, as we explain in the following subsection, securely authenticate web pages even in the presence of *intercepting adversaries* (often referred to as *Man In The Middle (MITM)* attackers). Intercepting adversaries are able to send and intercept (receive, eavesdrop) messages to and from *all* domains. Indeed, even without SSL/TLS, the HTTP protocol securely authenticates web pages against *spoofing adversaries,* which are able to send messages from all domains, but receive only messages sent to unallocated (adversary-controlled) domains. However, the security by SSL/TLS (against intercepting adversary; or by HTTP against spoofing adversary) is only with respect to the address (URL) and security mechanism (HTTPS, using SSL/TLS, or `plain` HTTP) requested by the application (usually browser). In a phishing attack (and most other spoofing attacks), the application specifies, in its request, the URL of the spoofed site. Namely, web spoofing attacks focus on the gap between the intentions and expectations of the (naïve) user, and the address and security mechanism specified by the browser to the transport layer.

In the next subsection, we give a brief description of the SSL/TLS protocols, focusing on their mechanisms for server authentication. We then review Web-spoofing and phishing attacks, showing how they are able to spoof even sensitive web sites protected by SSL/TLS. We also discuss some of the countermeasures against web spoofing proposed in previous works, and argue that they are appropriate for security savvy and alert users, but may not be sufficient for naïve or off-guard users. These will form the basis of the design criteria for defenses against web spoofing, which we present in the next section.

2.1 Server Authentication with SSL/TLS

Netscape Inc. developed the Secure Socket Layer (SSL) protocol, mainly to protect sensitive traffic, such as credit card numbers, sent by a consumer to web servers (e.g. merchant sites). Transport Layer Security (TLS) is the name of an IETF standard designed to provide SSL's functionality; most browsers enable by default both SSL and TLS. TLS has several improvements in cryptographic design, but they are beyond the scope of this work; therefore, we use, from here on, the name SSL, but refer also to TLS. For technical and other details see [R00].

We focus on SSL's core functionality and basic operations. Simplifying a bit, SSL operation is divided into two phases: a handshake phase and a data transfer phase. We illustrate this in Figure 2, for connection between a client and an imaginary bank site (*http://www.bank.com*). During the handshake phase, the browser confirms that the server has a domain name certificate, signed by a trusted Certificate Authority (*CA*), authorizing it to use the domain name *www.bank.com* contained in the specified web address (URL). The certificate is signed by *CA*; this proves to the browser that *CA* believes that the owner of the domain name *www.bank.com* is also the owner of the public key *PKserver*. Next, the browser chooses a random key *k*, and sends to the server $Encryt_{PKserver}(k)$, i.e. the key *k* encrypted using the public key *PKserver*. The brower also sends $MAC_k(messages)$, i.e. Message Authentication Code using key *k* computed over the previous messages. This proves to the server that an adversary didn't tamper with the messages to and from the client. The server returns $MAC_k(messages)$ (with the last message from the browser added to *messages*); this proves to the browser that the server was able to decrypt $Encryt_{PKserver}(k)$, and therefore owns *PKserver* (i.e. it has the corresponding public key). This concludes the handshake phase.

The data transfer phase uses the established shared secret key to authenticate and then encrypt requests and responses. Again simplifying, the browser computes $Encrypt_k(Request, MAC_k(Request))$ for each *Request*, and the server computes $Encrypt_k(Response, MAC_k(Response))$ for each *Response*. This protects the confidentiality and integrity of requests and responses.

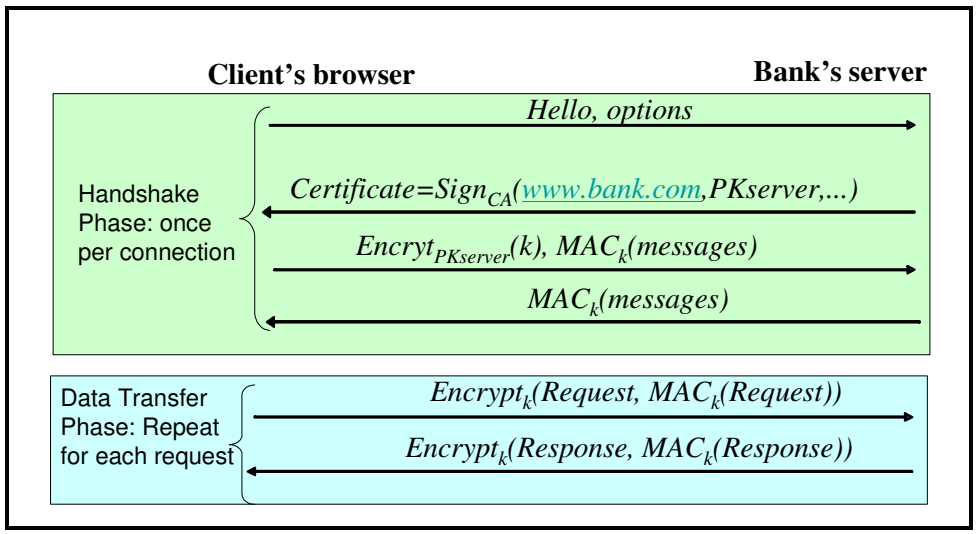

Figure 2: Simplified SSL/TLS Operation

To summarize, web security is based on the following basic security services of SSL:

1. Server (domain name) authentication[3]: SSL confirms that the server has the private key which can decrypt messages encrypted by the client using public key *PKserver*. The application using SSL, e.g. the browser, should confirm that this public key belongs to the `right server`. In particular, current browsers validate that the certificate is valid, signed by a trusted certificate authority, and contains the domain name of the site (www.bank.com in this example).
2. Confidentiality and authentication of the traffic between client and server, by using encryption and message authentication (MAC) using the shared secret `master key` established during handshake phase.

Unfortunately, most web pages are *not* protected by SSL. This includes most corporate and government web pages, and other sensitive web pages. One reason is performance; the SSL protocol, while fairly optimized, still

[3] SSL also supports client authentication, but very few web sites use this mechanism, possibly due to concerns about user acceptance and support costs; we therefore do not discuss it.

consumes substantial resources at both server and client, including at least four flows at the beginning of every connection, state in the server, and computationally-intensive public key cryptographic operations at the beginning of many connections. Efficiency may be substantially improved, e.g. see [BSR02].

2.2 Web Spoofing and Phishing Attacks

SSL is a mature cryptographic protocol; while few weaknesses were found in some early versions and implementations of it, the versions currently used, from version 3.0 of SSL and 1.0 of TLS, seem secure. (This refers to the full protocol; the simplified description above is not secure.) However, the security of a solution based on SSL/TLS depends on *how* the protocol is used, e.g. by browsers. There are two major vulnerabilities in the way browsers use SSL/TLS; in TrustBar design, we are addressing both of these vulnerabilities.

The first vulnerability is due to the validation that the server's public key, which SSL obtains from the server's certificate, belongs to the site with the given location (URL). This validation is the responsibility of the application (e.g. browser) and not part of the SSL/TLS specifications; SSL/TLS merely passes the server's certificate to the application.

Currently, browsers are vulnerable to the *false certificate attack*, where the adversary receives a certificate for the domain of the victim web page from a CA trusted by the browser, but containing a public key generated by the adversary. Therefore, the adversary has the matching private key and can pass SSL server authentication for the victim web page. We now explain how the false certificate attack works.

Most browsers are pre-installed or automatically updated [M04] with a long list of (over hundred) certification authorities which are trusted for server authentication by default; few users inspect this list and remove unknown or untrusted CA entries (e.g. Saunalahden[4]). In fact, the main criteria for inclusion in the list of trusted certificate authorities, at least for Mozilla [Mozilla] and Microsoft [M04], is a WebTrust for Certification Authorities audit or an equivalent third-party audit attestation. A CPA office obtains permission to grant WebTrust seals, by fulfilling modest educational and procedural requirements [A04].

Therefore, there is a competitive market for both validation of certification authorities (among auditors), and for issuing of certificates (among certification authorities). There is a risk that this competition may lead to lax certification; in fact, most certification authorities, and their auditors, give careful disclaimers of liability for damages from false certification. Furthermore, as we explain below, it is highly unlikely that users relying on a false certificate will be able to identify and sue the CA that issued this false certificate. Therefore, as observed by [ES00, FS*01, G04a], it is not too difficult to obtain valid yet false certificates from a CA listed in the default list of major browsers.

Furthermore, currently, browsers do not support revocation of certificates, e.g. when a CA detects that it was cheated and wants to cancel a false certificate, or is informed of private key exposure. In practice, certificates contain substantial validity periods, typically over a year. In practice, organizations would not tolerate having to wait more than few hours or at most a day or two to disable an unauthorized (e.g. spoofed) site; and in fact, such sites are normally blocked very quickly, but this is done by blocking their IP address or domain name, directly via the relevant ISP or domain name registrars. However, blocking IP address or domain name does not defend against an intercepting (MITM) adversary; recall that SSL goals include protection against intercepting adversary. However, this can be solved easily, by using a secure domain name server to validate the domain name of protected sites is not blocked. There are also many proposals for efficient revocation mechanisms, e.g. [M97, NN00].

[4] Saunalahden Serveri is included in the list of trusted root CA trusted by Microsoft [M04]. Unfortunately, the only information about it in [M04] is its name and URL, and the web site is apparently in Finnish, which means that the authors do not know anything about it. This definitely implies no criticism on our part of Saunalahden Serveri.

Browser designers are aware of the substantial cost of certificates, which may lead honest servers to use expired certificates, and want to support new certificate authorities. Therefore, browsers present a rather mild warning when the CA is unknown, certificate has expired, etc.; and many users approve the use of the certificate anyway. Grigg reports a `real` false certificate attack, using a certificate from a non-existent CA and relying on users to ignore and `click thru` the warning window [G04]. We believe this is the result of the fact that browsers violate `Secure UI Principle III: Alerts should wake-up` presented in the Introduction.

Therefore, the certificate management in current browsers is vulnerable. However, we only found a single report of a `real` false certificate attack [G04]. One explanation for the limited exploitation of this vulnerability, is the existence of an even easier to exploit vulnerability that we describe next, namely the dependency on the user to validate the identity and protection of web sites.

In the current design of browsers, the user is responsible to validate the authenticity of web sites, by noting relevant status areas in the browser user interface. The relevant status areas are the location bar, containing the URL (Universal Resource Locator), and the SSL indicator (typically, as open lock for insecure sites, closed lock for SSL/TLS protected sites). We are mostly interested in the *web spoofing attack,* which exploits this vulnerability, by directing the browser to an adversary-controlled *clone site* that resembles the original, *victim site*, which the user wanted to access. Web spoofing attacks are very common, and are the most severe threat to secure e-commerce currently. As we explain below, most web spoofing attackers simply rely on the fact that many users may not notice an incorrect URL or the lack of SSL indicator, when approaching their online banking site (or other sensitive site). Therefore, an attacker can circumvent the SSL site authentication trivially, by not using SSL and/or by using a URL belonging to a domain owned or controlled by the attacker, for which the attacker can obtain a certificate. More advanced attacks can mislead even users that validate the SSL indicator and location bar (containing URL).

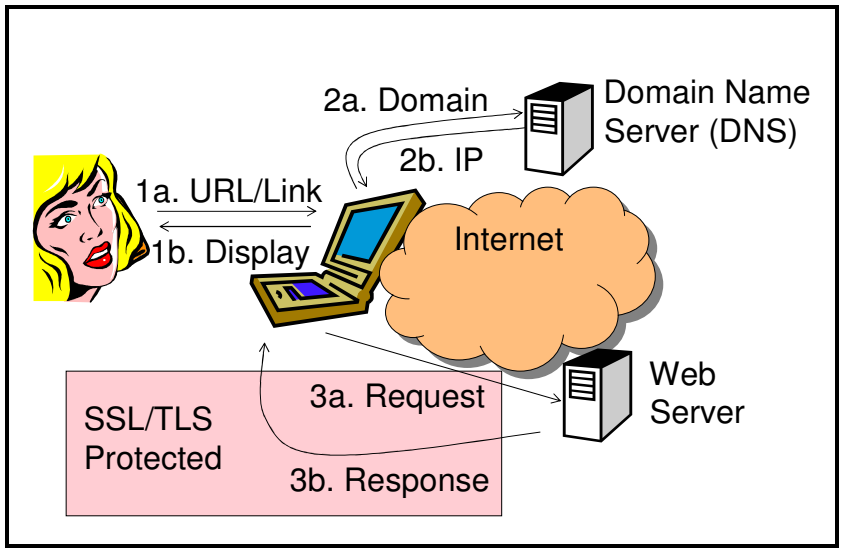

Figure 3: HTTP request/response process with SSL protection

The first challenge for a web spoofing attack is to cause the browser to receive the clone site, when the customer is really looking for the victim site. The attacker can exploit different parts of the process of receiving a (sensitive) web page. We illustrate the typical scenario of receiving a sensitive web page in Figure 3. The process begins when the user selects the web site, by entering its location (URL) or by invoking a bookmark or link, e.g. in an e-mail message (step 1a). The browser, or the underlying transport layer, then sends the name of the domain of the site, e.g. xxx.com, to a Domain Name Server (step 2a). The Domain Name Server returns the IP address of the site (step 2b). Now, the client sends an HTTP request to the site, using the IP address of the site (step 3a), and receives the HTTP response containing the web page (step 3b); these two steps are protected by SSL, if the URL

indicates the use of SSL (by using the *https* protocol in the URL). Finally, the browser presents the page to the user (step 1b).

If we did *not* use SSL, an intercepting adversary could attack all three pairs of steps in this process, as follows:

1. Trick the user into requesting the spoofed web site in step 1a, and/or into using *http* rather than *https*, i.e. not protect the request and response using SSL.

2. Return an incorrect IP address for the web server in step 2b. This can be done by exploiting one of the known weaknesses of the DNS protocol and/or of (many) DNS servers. A typical example is DNS cache poisoning (`pushing` false domain→IP mappings to the cache of DNS servers).

3. Intercept (capture) the request in step 3a (sent to the right IP address) and return a response in step 3b from the spoofed site.

The third attack requires the adversary to intercept messages, which is relatively hard (requires `man in the middle`, intercepting adversary). The second attack requires defeating DNS security, which is often possible, but may be difficult (except for an intercepting adversary). Hence, most spoofing attacks against SSL/TLS protected web sites focus on the first attack, i.e. tricking the user into requesting the spoofed web site and/or into using an insecure connection (without SSL) rather than an SSL-protected connection.

Unfortunately, swindlers often succeed in tricking users into using the wrong URL, or not using SSL (i.e. *http* rather than *https*). We believe this is due to two violations of the Secure UI principles by the current browser user interface. First, most (naïve) users are not aware of the structure of URL and domain names and their relation to ownership of the domain; by expecting them to understand such technical indications, browsers violate the *Secure UI Principle IV: `Cryptography` is in Greek.*. Second, users rarely type manually the address (URL) of the sensitive web page; instead, in most cases, users *link* into the sensitive web page from a *referring web page*, which is usually insecure, e.g. a homepage of the service provider or results from search engine, or from a *referring e-mail message*. Very few service providers and search engines use SSL to protect their results (links to sites). Therefore, an attacker that can intercept the requests in step 3a, or return incorrect IP addresses in step 2b, is usually able to trick the user into requesting the URL of the spoofed site. Security should not depend on users entering – or even knowing – the correct URL; see *Secure UI Principle II: Security must be usable to be used.*

Most web-spoofing attacks, however, use methods which do not require either interception of messages to `honest` web sites, or corruption of servers or of the DNS response; these methods work even for the weak `unallocated domain` adversary. One method is *URL redirection*, due to Felten et al. [FB*97]. This attack begins when the user accesses any `malicious` web site controlled by the attacker, e.g. containing some content; this is the parallel of a Trojan software, except that users are less cautious about approaching untrusted web sites, as browsers are supposed to remain secure. The attack works if the user continues surfing by following different links from this malicious site. The site provides modified versions of the requested pages, where all links invoke the malicious site, which redirects the queries to their intended target. This allows the malicious site to continue inspecting and modifying requests and responses without the user noticing, as long as the user follows links. However, this attack requires the attacker to attract the user to the malicious web site.

In practice, attackers usually use an even easier method to direct the user to the spoofed site: *phishing spoofing attacks,* usually using spam e-mail messages. In Figure 4 we describe the process of typical phishing attack used to lure the user into a spoofed web site[5]. The adversary first buys some unallocated domain name, often related to the name of the target, victim web site. Then, the adversary sends spam (unsolicited e-mail) to

[5] There are other forms of `phishing attacks`, e.g. tricking users into providing sensitive information by e-mail or phone, making them install malicious software, or luring them to the swindler's web site under some false promises (rather than impersonation as a trusted site).

many users; this spam contains a `phishing bait message`, luring the user to follow a link embedded in the bait message. The mail message is a forgery: its source address is of the victim entity, e.g. a bank that the user uses (or may use), and its contents attempt to coerce the user into following a link in the message, supposedly to the victim organization, but actually to the phishing site. If the victim entity signs all its e-mail, e.g. using S/MIME or PGP [Z95], then our techniques (described later on) could allow the user to detect this fraud. However, currently only a tiny fraction of the organizations signs outgoing e-mail, therefore, this is not an option, and many naïve users may click on the link in the message, supposedly to an important service from the victim entity. The link actually connects the users to the spoofed web site, emulating the site of the victim entity, where the user provides information useful to the attacker, such as credit card number, name, e-mail addresses, and other information. The attacker stores the information in some `stolen information` database; among other usages, he also uses the credit card number to purchase additional domains, and the e-mail addresses and name to create more convincing spam messages (e.g. to friends of this user).

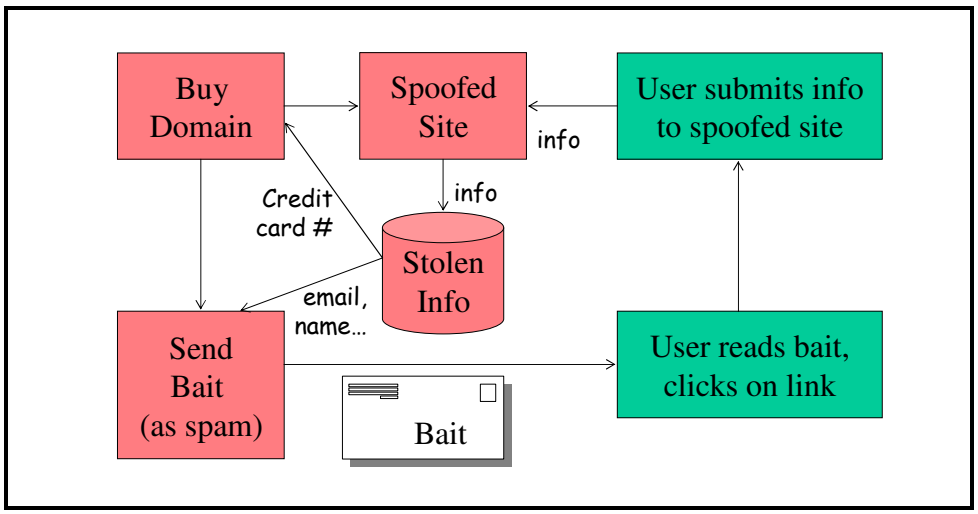

Figure 4: Process of Typical Phishing Spoofing Attack

Currently most phishing attacks lure the users by using spam (unsolicited, undesirable e-mail), as described above. However, we define *phishing spoofing attack* as (any method of) luring the user into directing his browser to approach a spoofed web site. For example, an attacker could use banner-ads or other ads to lure users to the spoofed site. We believe spam is the main phishing tool simply since currently spam is extremely cheap and hard to trace back to the attacker. Spamming is causing many other damages, in particular waste of human time and attention, and of computer resources. Currently, the most common protection against spam appears to be content based filtering; however, since phishing attacks emulate valid e-mail from (financial) service providers, we expect it to pass content-based filtering. Proposals for controlling and preventing spam, e.g. [CSRI04, He04], may also help to prevent or at least reduce spam-based phishing; another approach to prevent phishing is described in [J05].

Most phishing spoofing attacks require only an unallocated web address and server, but do not require intercepting (HTTP) requests of the user; therefore, even weak attackers can deploy them. This may explain their popularity, as shown in Figure 1. This means that the domain name used in the phishing attack is different from the domain name of the victim organization.

Many phishing spoofing attacks use deceptive domain names similar to the victim domain names, possibly motivating the change by some explanation in the e-mail message, e.g. see [BBC03]; often, they also display the anchor text with the correct victim URL, while linking to the spoofed web site. Unfortunately, most naïve users do not notice the attack or the change in the domain name, since:

- Users do not read the location bar at every transaction.
- Users do not always understand the structure of domain names (e.g., why accounts.citibank.com belongs to CitiBank™, while citibank-verify.4t.com was a spoofed site [Citi04]). Corporations using bad-formed domain names aggravate this, e.g. TD Waterhouse used the highly misleading address *http://tdwaterhouse.ip02.com* as a link in mail sent to their customers.
- Organizations often do not realize the importance of using a consistent domain name directly associated with their brand and use multiple domain names, often not incorporating their brand, e.g. CitiBank™ uses at least eight domains, often not incorporating their brand name, e.g. accountonline.com.

Since the spoofed web sites typically use deceptive domain names which they own or control, they usually could purchase a certificate from one of the (many) Certificate Authorities in the default list trusted by popular browsers; for example, why would a CA refuse to provide a certificate for the *4t.com* or *ip02.com* domains[6]? However, most spoofed web sites do not even bother to get a certificate (with the costs and overhead involved, and with some risk of identification). Instead, most spoofed web sites – and in particular all of the (many) examples in [Citi04] – simply *do not invoke SSL*. We observe that many, or probably even most, users did *not* notice the lack of use of SSL in these spoofed sites.

In fact, it turns out that many existing web sites require sensitive information such as user ID and passwords, in *unprotected web pages*. This includes some of the most important, well-known and sensitive web sites. In Figure 5 we show unprotected login forms of Chase™, PayPal™, Amazon™, Microsoft's .Net Passport™ and eBay™; we have informed all of these and other sites[7], and hope they will be protected by the time this article is published (eBay™ is already protected). The vulnerability of the Passport site may be most alarming, considering that it provides a `single-sign on` security service to other web sites, and therefore a weakness in it can impact many other sites; other weaknesses of Passport were reported in [KR00].

For readability, Figure 5 contains only the most relevant parts of the web pages, by reducing the size of the browser window; the reader can probably find other examples. Examples (a) (Chase™) and (b) (Amazon™) use the Mozilla browser, with our extension, described in the following section, which added a clear warning (on top) noting the fact that these pages are *not* protected by SSL. This warning would not appear in standard browsers (without our extension), and considering that many of these pages display padlocks and/or textual claims of security, it is quite probable the many (naïve?) users will think that these pages are secure; see results of survey in Section 5. In examples (c) (Microsoft .Net Passport™), (d) (eBay™) and (e) (PayPal™), we used the Microsoft Internet Explorer™ (without extension); the fact that these pages are unprotected is indicated here only by the *lack of the `lock`* icon representing secure sites. All of these pages prompt users to input passwords and account numbers, *implying or even explicitly claiming (incorrectly) that these sites are protected*. Most of these sites invoke SSL to protect the sensitive information (password) in transit. However, this is too late; an attacker could present a spoofed web-page, which would appear the same, and collect this sensitive information. This attack is often easier to deploy than eavesdropping on the communication between the user and the site; in most cases where the attacker is able to eavesdrop on the communication, she can also present a spoofed web page (and therefore get the information in the clear). We believe that this proves that users – and even serious web-site designers – do not notice the lack of SSL protection, even in sensitive web pages belonging to established organizations.

[6] The *ip02.com* domain is used by TD Waterhouse™, and the *4t.com* domain was used by a spoofed web site.
[7] Readers can check if these sites are still unprotected, as when we visited them on July-September 2004, by following the following links to Chase™, PayPal™, Amazon™, TD Waterhouse™, eBay™, Microsoft's .Net Passport™ and Yahoo!™.

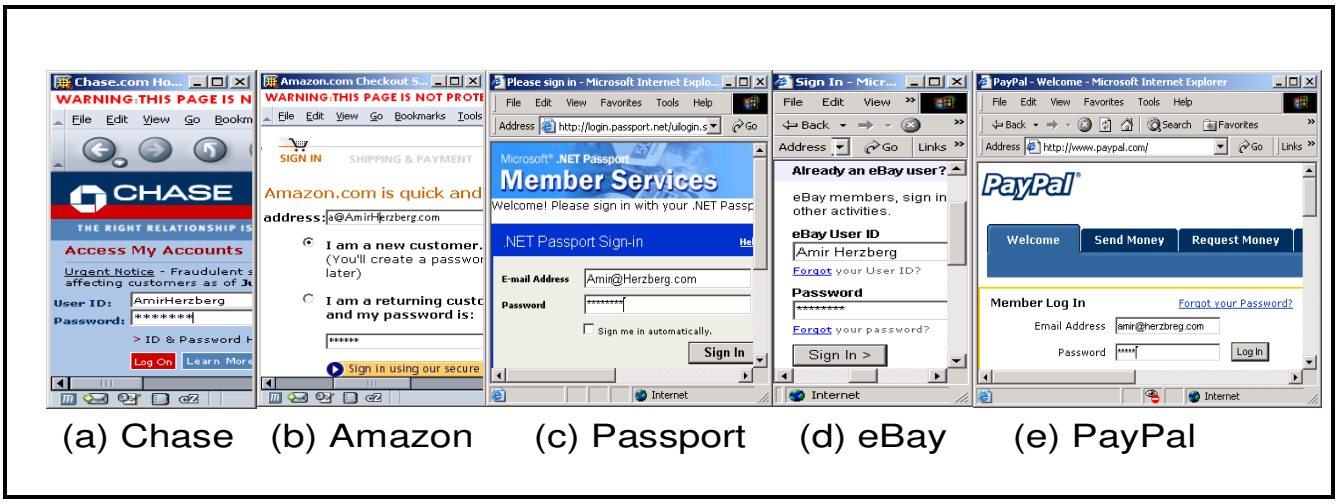

Figure 5: Unprotected Login to Important Sites.

3 Spoofing Prevention: Design Criteria

We now present design criteria for prevention of web and credential spoofing, extending the criteria presented in [YS02].

- **Provide branding, prevent spoofing.** Obviously, the most basic requirement is that credentials, logos or any other identification information should be secure against spoofing by adversaries. Namely, the adversary should not be able to emulate any such information presented by the anti-spoofing mechanism, when the user is viewing a web site unauthorized to present these credentials, logo, or address. In particular, the mechanism should allow organizations to use and reinforce brand identity, as a major mechanism to identify the organization and establish confidence in it and in the web site. Branding is also important for the certification authorities; currently, few users are aware of the identity of the CA certifying the owner of a protected page. By presenting the logo and/or name of the CA, we allow customers to base their trust on known identities (brands), increase the value of the brand and increase the motivations of the CA to diligently validate the site.

- **Effectiveness for naïve and off-guard users:** the credentials should be highly visible and simple to understand, which will ensure that even naïve, off-guard users, will detect the *lack of* necessary credentials when accessing a web site. In particular, as indicated by [YS02], graphical indicators are preferable to textual indicators, and dynamic indicators are preferable to static indicators. Furthermore, to facilitate recognition by naïve users, the credentials should use simple, familiar and consistent presentations. Finally, and again as indicated by [YS02], the (secure) browser should couple between the indicators and the content, rather than present them separately.

- **Support all kinds of credentials:** it should be possible to protect any credential, including information currently displayed in browser status areas (location, SSL indicator, etc.) and additional credentials such as logos, seals, certificates etc.

- **Minimize/avoid user work:** The solution should not require excessive efforts by the user, either to install or to use. In particular, we prefer to base credential validation on simple visual clues, without requiring any conscious user activity during validation. This is both to ensure acceptability of the mechanism, as well as to increase the likelihood of detection of the lack of proper credentials by naïve users.

- **Minimize intrusiveness:** the solution should have minimal or no impact on the creation of web sites and presentation of their content.

- **Customization:** the visual representation of the different credentials should be customizable by the user. Such customization may make it easier for users to validate credentials, e.g. by allowing users to use the same graphical element for categories of sites, for example for `my financial institutions`. Similarly, a customized policy could avoid cluttering the TrustBar with unnecessary, duplicate or less important logos; e.g., it may be enough to present one or two of the credit card brands used by the user (and that the site is authorized to accept), rather than present the logos for all of them. In addition, customization could allow users to assign easy to recognize graphical elements (`logos`) to sites that do not (yet) provide such graphical identification elements securely (i.e. that do not yet adopt our proposals). Finally, as argued in [YS02], by having customized visual clues, spoofing (of the TrustBar itself) becomes harder.

- **Migration and interoperability:** the solution should provide benefits to early adopting sites and consumers, and allow interoperability with existing (`legacy`) web clients and servers. In particular, it should be sufficient for a client to use the solution, to improve the security of identification of existing SSL protected web sites.

4 TrustBar UI Design: Security and Trust User Interface for Naïve Users

TrustBar is a new component to the user interface of the browser. The goal of TrustBar is to present highly visible, graphical interface, establishing securely the identity of the web site. We expect that the browser will present most identifiers via graphical elements such as logos, defined or at least approved by the user or somebody highly trusted by the user (see more below). We implemented the *TrustBar browser extension* for the open-source Mozilla™ and FireFox™ browsers; see screen-shots of two protected sites, with logos and other credentials presented in the TrustBar, in Figure 6. In Figure 6 (a) and (b) we show the login page of Gmail™; in (a) the site and the certificate (identifying) authority are identified by name, while in (b) they are identified by a logo. In Figure 6 (c) we show the form beginning the login process in UBS bank e-banking service.

These screen shots present our main design decision: TrustBar controls a significant area, located at the top of every browser window, and large enough to contain highly visible logos and other graphical icons for credentials. TrustBar must appear in *every* window opened by the browser, protected or unprotected, including windows used for helper applications and applets. This prevents attacks as in [LN02, LY03] where a spoofed site opens windows to hide browser indicators (e.g. padlock or location area) and `overwrite them` with misleading indicators.

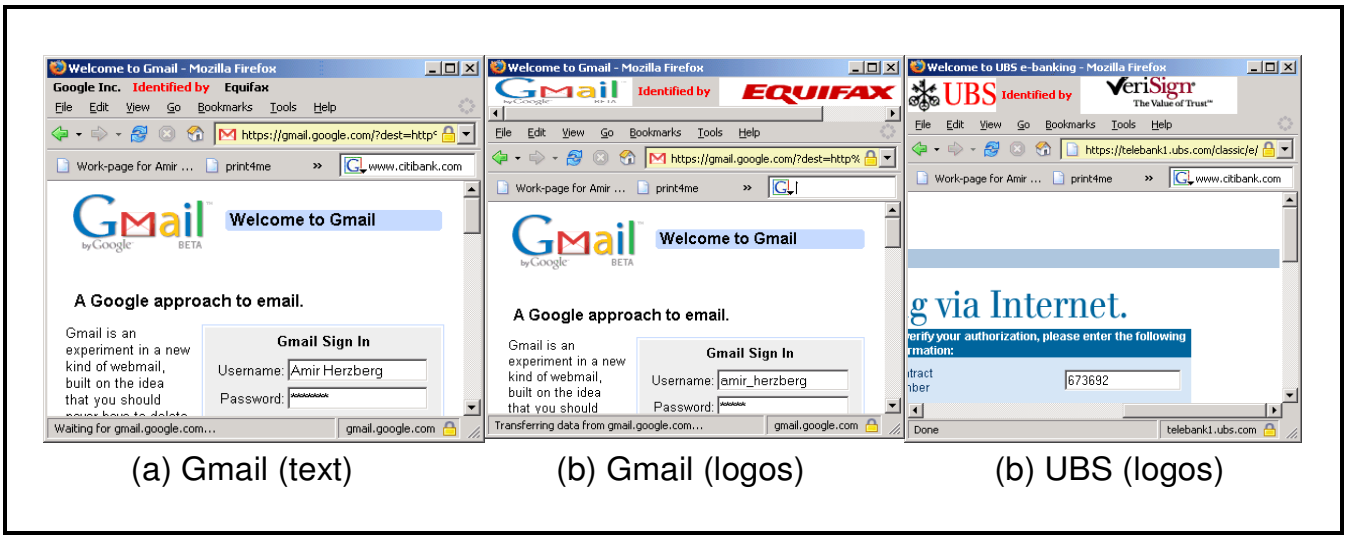

Figure 6: Screen-shots of secure sites with logo in TrustBar

Our implementation gives the browser (and code executed by it) access only to the window below the TrustBar. This implementation is easy in Mozilla, and seems to be secure against change by any content downloaded automatically from the Web[8]. Additional mechanisms to protect the TrustBar may include:

1. Helper and applet windows may be marked separately from the browser itself, and in particular from the trusted area, e.g. by enclosing all helper and applet windows by a special, highly visible `warning` border [YS02].
2. To make it harder to spoof the trusted area, even for a program that can write on arbitrary locations in the screen, it may be desirable that the background of the trusted area will be a graphical element selected randomly from a large collection, or selected by the user.
3. The browser may restrict opening of new (`pop-up`) windows, including for helper applications and applets. In particular, the browser may confirm that the content was approved by an appropriate authority. This solution can also assist in the prevention of spam web advertisements and pop-up windows, and is already supported by many new browsers.

4.1 Secure vs. Insecure Site Indication

Existing browsers indicate that a site is SSL-protected, by a small SSL-status icon, usually in the status area at the bottom of the page (see Figure 5). However, this indication is not very visible, and naïve or off-guard users may not notice its absence, when accessing a sensitive site (e.g. if visiting this site routinely, as for personal bank). Indeed, all of the real-life spoofing (and phishing) attacks we have seen, directly on the Web or described by others (e.g. [Citi04]), were on sites without SSL/TLS protection – even in cases where the attackers used domain names which do not appear related to any known trademark (e.g. *4t.com*, used in one of the phishing attacks on Citibank™). Furthermore, a web site can request that the browser avoid displaying the status area (simply by using the `window.open` JavaScript method), making the lack of SSL even harder to notice; as mentioned above, swindlers already exploited this method to spoof secure web sites.

To prevent these threats, whenever TrustBar detects that a web site is *not* SSL-protected, it displays a highly visible warning[9] message (see Figure 5 (a), (b)). We recommend that corporate and other serious web sites

[8] Of course, our implementation may be circumvented by malicious `Trojan horse` program running on the client machine, unless protected by the operating system. Considering that most operating systems and client computers are insecure, this is a serious threat, but outside the scope of this manuscript.

avoid this warning message, by protecting *all* of their web pages, and certainly all of their web forms[10], preferably presenting the corporate logo in the TrustBar. By protecting all of their pages, such sites will make it quite likely that their users will quickly notice the warning message in the trusted browser area, when the user receives a spoofed version of a web page of such sites. Furthermore, this ensures that all the organization's web pages will present the logo and credentials of the site (and organization) in the TrustBar, using and re-enforcing the brand of the organization.

4.2 Identification of Web Sites and Certification Authorities

Currently, browsers identify the provider of the web page by indicating the Universal Resource Locator (URL) of the web page in the location bar of the browser. This usually allows knowledgeable web users to identify the owner of the site, since the URL includes the domain name (which an authorized domain name registrar allocates to a specific organization; registrars are expected to deny potentially misleading domain names). However, the identity of the provider is not necessarily included (fully) in the URL, and the URL contains mostly irrelevant information such as protocol, file, and computer details. Furthermore, the URL is presented textually, which implies that the user must make a conscious decision to validate it. All this implies that this mechanism may allow a knowledgeable web user, when alert and on guard, to validate the owner of the site; but novice, naïve or off-guard users may not notice an incorrect domain, similarly to their lack of notice of whether the site is secure, as discussed in the previous subsection.

Furthermore, popular browsers are pre-configured with a list of many certification authorities, and the liabilities of certificate authorities are not well defined; also, the identity of the CA is displayed only if the user explicitly asks for it (which very few users do regularly, even for sensitive sites). As a result, it may not be very secure to use the URL or identity from the SSL certificate. Therefore, we prefer a more direct and secure means of identifying the provider of the web page, and – if relevant – of the CA, and not simply present the URL from the SSL certificate in the TrustBar.

TrustBar identifies both site and the certificate authority which identified the site, allowing users to decide if they trust the identification by that authority. The identification is based on the SSL server authentication, confirming that the site possesses the private key corresponding to a public key in a certificate signed by the given certificate authorities, which currently must be one of the certificate authorities whose keys are pre-programmed into the browser.

Preferably, TrustBar identifies the site and authority by logo (or some other image selected by the user, e.g. a `my banks` icon). However, since currently certificates do not contain a logo, TrustBar can also identify the site and authority by name. See Figure 6 for identifications by logo (in (b) and (c)) and by name (see (a)). TrustBar supports several methods of identifying the logo or name:

- Names are taken from the `organization name` field of the existing X.509 SSL certificates. However, TrustBar does not automatically trust all the (over hundred) certificate authorities trusted by the (popular) browsers. Instead, on the first time it receives a certificate from a given CA, it pops up a dialog, providing the user with the details of the CA and asking the user to determine if to trust site identifications by this CA (or just this particular identification). The user can also select a logo for the CA, to be presented in TrustBar for sites identified by this CA.

---

[9] The current implementation of the warning message should yet be improved, to include graphics, and to provide simple explanation clarifying that the site is not necessarily corrupt, just not protected, and how to ask for, and detect, protected site.

[10] Some people consider the warning on every unprotected web page as problematic, since it reduces the consumer's confidence in web sites. This may motivate warning only on `relevant` pages, e.g. only on web form (which may be used to input sensitive information). However, notice that web pages that contain active content such as scripts and helper applications, may also appear as `forms` to the user, but this is hard or impossible to detect by TrustBar.

- The user can identify a logo for a site, by `right-click` on an image of the logo (which usually appears on the same page). Whenever opening a page with the same public key, TrustBar automatically presents this logo.

- Dialog identification of site and CA by logo or name: users can right-click on the TrustBar to open a dialog, in which they can select any name, or image file to be used as the logo, for the site and for the CA. TrustBar may also automatically present one or more candidate logo images, taken from the page; simple heuristics based on the file name and the geometrics of the image will often succeed.

- The site may provide the logo in an appropriate (public key or attribute) certificate extension, e.g. as defined in [RFC3709]. This may be the same as the certificate used for the SSL connection, or another certificate (e.g. identified by a <META> tag in the page). The logo may be signed by entities that focus on validating logos, e.g. national and international trademark agencies, or by a certificate authority trusted by the user. As noted above, TrustBar does not automatically trust logos (or names) by certificate authorities; instead, it prompts the user who specifies whether identifications by this CA can be automatically trusted in the future.

By displaying the logo or name of the Certifying Authority (e.g. EquiFax or Verisign in Figure 6), we make use and re-enforce its brand at the same time. Furthermore, this creates an important linkage between the brand of the CA and the validity of the site; namely if a CA failed and issued a certificate for a spoofing web site, the fact that it failed would be very visible and it would face loss of credibility as well as potential legal liability.

Notice that most organizational web sites already use logos in their web pages, to ensure branding and to allow users to identity the organization. However, browsers display logos mostly in the main browser window, as part of the content of the web page; this allows a rogue, spoofing site to present false logos and impersonate as another site. One exception is the FavIcon, a small icon of the web site, displayed at the beginning of the location bar in most (new) browsers. Many browsers, e.g. [Mozilla], simply display any FavIcon identified in the web page. Other browsers, including Internet Explorer, display FavIcon only for web-pages included in the user's list of `Favorite` web pages, possibly to provide some level of validation. However, since browsers display FavIcon also in unprotected pages, and come with huge lists of predefined favorite links, this security is quite weak. We believe that the logo or icon presented in the FavIcon area should be considered a part of the TrustBar and protected in the same manner.

To validate the contents of the TrustBar, we first use SSL to ensure that the web site has the private key corresponding to a given public key. The browser – or TrustBar extension – then uses the site's public key to identify the name or logo. Notice this does *not* depend on the domain name or URL in the certificate.

4.2.1 User-certified Logo Identification and Peer identification (`Web of Trust`)

TrustBar generates, upon installation, a private signature key, which it uses later on to sign logo certificates, linking public keys and logos, if the user (manually) specifies the use of the logo for the public key. These `user certificate` can be stored in a file accessible via the network, so that other instances of TrustBar belonging to the same user, or to others trusting him, can automatically use the logos. TrustBar allows users to specify the location of one or more repositories from which it downloads logo certificates (when needed or periodically). TrustBar allows the user to input, or approve, logo certificate validation keys, e.g. of the same user on another machine. This allows a user to certify a logo in one machine (e.g. office) and use it automatically in other machines (e.g. home or mobile).

The user can also input or approve logo certificate validation keys of logo certification authorities, or of *other users he trusts*. This allows users to share the task of validating logos with trusted friends, similar to the PGP web-of-trust [Z95] model, essentially turning these friends into `tiny logo certificate authorities`. We believe this option will facilitate `grass-root` adoption of logo certificates, which may expedite the deployment of trustworthy, established logo certificate authorities.

4.3 `Last visit` and other historical indicators

Many operating systems, e.g. Unix, display the time and date of the last login as part of the login process. This allows users to detect unauthorized usage of their accounts, by noting usage after their last authorized login. The TrustBar (TrustBar) could provide such credentials, if desired, by maintaining record of previous access to each site (or using each public key). Each user can indicate in the `TrustBar preferences` what historical indicators to present (or maintain).

Notice, however, that the history is limited to access via this particular browser and computer. This may make the records of entrance to a particular site less valuable than the operating systems indicators; also notice that detection of spoofing using this mechanism requires users to notice wrong historical information. However, this mechanism may be useful to create differentiation between certificates issued by different certificate authorities. Recall that common browsers contain a pre-defined list of above a hundred certification authorities and keys; clearly, some of them are major, and widely used, and others may be minor, rarely used, and possibly less trusted. The browser could present some statistics on the number of certificates previously received from the CA of the current certificate; this could help make the user more cautious when accepting a certificate from a rarely used certificate authority.

5 Browser Secure UI Survey

In the previous works on secure usability, there has been very limited empirical study, and claims were mostly based on common sense and on the experience of the researchers. We believe that it is important to validate user interface principles and conjectures, using appropriate empirical studies. In this section, we present the validation of the secure UI principles and other assumptions regarding browser secure user interface, using a limited survey we conducted during [He04a]. We hope to extend this work in the future.

The survey was conducted by handing out questionnaires to the audience of [He04a] and asking them to fill them in; we received 42 filled-in surveys. The questionnaire is presented at the Appendix. Questions 1, 2 and 4 are self-contained, and were asked at the beginning of the presentation; question 3 was asked with respect to a series of screenshots, each presented briefly (details below). The survey was anonymous.

Almost all respondents reported using the web for more than one year, and having studied computer science or related area; we focused on the 37 out of the 42 who both reported studying computer science and using the web for more than one year. We believe that some of our claims, regarding naïve users, may hold even better for less experienced users.

The questionnaire focused on two issues: whether users understand the term `certificate authority`, which is essential to understand the existing browser security UI; and whether users can correctly distinguish between protected, unprotected and spoofed (fraudulent) sites, when using existing browser UI and when using TrustBar.

5.1 Usability of the `Certificate Authority` term

Currently, browsers extensively use the term `Certificate Authority` in their security UI and help screens (where they also explain it, of course). We consider this a technical term, and appropriate to test *Secure UI Principle IV: `Cryptography` is in Greek.*. The lack of understanding of what is a CA and what is its role, as well as the lack of awareness of the identity of the CA, motivates TrustBar's display of the CA identity (as name or logo), as well as the use of the term `identified by` rather than terms like `certified by`.

The first two questions checked whether users understand the term `certificate authority`; this (technical) term is used extensively in the security user interface of existing browsers. The first question simply asked if users know what is a certificate authority; 29 of the 37, or 78% of the (more experienced) respondents, indicated that they know the meaning of the term `certificate authority`. This already indicates that the use of this technical term

implied that at least 22% of the (relatively experienced, computer-science educated) users did not understand a critical security indicator – and this is assuming that every respondent that answered positively.

In the second question, we gave four names, and asked the participants which of the four are trustworthy certificate authorities; we also allowed `don't know` as an answer. The names were: VeriSlim, Visa, Saunalahden and VeriSign. The two last are certificate authorities in the default list of almost all browsers; Visa is a credit card company (and does not issue SSL certificates), and VeriSlim is an invented name (used as an example of a false CA).

It seems reasonable to expect that users that truly understand the meaning of a certificate authority, would at least identify VeriSign, which is probably the most well known and widely used certificate authority. However, only 20 of the responses identified VeriSign as a trusted CA, namely just a bit over half of the (experienced, CS-educated) users. We believe that this demonstrates the potential branding value that TrustBar can bring to the established, trust-worthy certificate authorities like VeriSign (additional study should confirm that after some exposure to sites `identified by` VeriSign in TrustBar, users trust in VeriSign will increase).

Most browsers would accept any SSL certificate from Saunalahden exactly in the same way they accept certificate from any of the other (over 100) `trusted` certificate authorities. However, we do not find it very surprising that {\em none} of the users identified Saunalahden as a trusted CA. We believe this shows that the current browser security UI is broken, since browsers do not expose the CA identity to the user and trust all of them equally.

Additional indications to this are the replies regarding Visa and VeriSlim. In fact, we were quite surprised to find that only 7 answers correctly identified VeriSlim as {\em not} being a trustworthy CA; and that only 6 answers identified Visa as {\em not} being a (trustworthy) CA.

To summarize, the replies we received seem to confirm our beliefs that the use of technical terms like `certificate` makes it harder for users to understand security indicators and controls, and therefore reduces the security (as per Principle IV).

5.2 Identification of protected, unprotected and spoofed sites

The main goal of the third question was mainly to validate our beliefs that the current browser security indicators are not clear and visible enough, to allow most (naïve) users to discern between unprotected, protected and spoofed sites. For this purpose, we explained briefly the current browser security indicators (padlock, location bar), and then presented in the survey three screen shots, presented in Appendix B. We presented each screen shot for 10 to 15 seconds; the screenshots were taken using Mozilla and in the Amazon web site. The first screen shot, in Figure 7, is unprotected login form; the second, in Figure 8, is a spoofed version of the (unprotected) Amazon site, in a different domain; and the third, in Figure 9, is a protected login form.

The results confirm our claims: most users are not able to discern between protected, unprotected and spoofed (fake) sites. In fact, only 10 of the 42 responses correctly identified the unprotected Amazon site, and only 11 correctly identified the protected Amazon site. For the fake site, we consider both the `fake` and the `unprotected` answers as valid (since the site is also unprotected), resulting in the slightly higher count of 15. Overall we find that although these users were clearly aware of the need to look for the security indicators, between 65% to 77% of them were not able to correctly identify the status of these pages. This clearly shows that the current browser security indicators are not sufficiently usable.

The second goal of the third question was to evaluate whether the use of TrustBar is likely to improve the ability of users to discern between unprotected sites, protected sites and spoofed (fake) sites. For this purpose, we gave users a very brief explanation on the TrustBar security indicators, and then presented three additional screen shots, this time using a browser equipped with TrustBar. Again, the screen shots are presented in Appendix B, and

each was presented for 10 to 15 seconds, taken using Mozilla in the Amazon web site. We leave it as a simple exercise to the reader to identify the protected, unprotected and spoofed (fake) among these three screen shots.

The results provide positive indication supporting out belief that the use of TrustBar improves the ability of (naïve) web users to discern between protected, unprotected and fake sites. Specifically, the number of user that correctly identified each of the three sites essentially *doubled* (to 21, 22 and 29).

We note that additional work is required to confirm these experiments, and validate the secure UI principles and the TrustBar design. In particular, an experiment involving users actually interacting with browsers would be very helpful. We are now working towards such additional experimentations.

6 Conclusions and Recommendations

As already shown in [FB*97] and in the developer community, currently web users, and in particular naïve users, are vulnerable to different web spoofing attacks; furthermore as shown in [APWG04, L04] and elsewhere, phishing and spoofing attacks are in fact increasingly common. In this paper, we describe browser and protocol extensions that we are designing and implementing, that will help prevent web-spoofing (and phishing) attacks. The main idea is to enhance browsers with a mandatory *TrustBar (TrustBar)*, with a fixed location at the top of every web page, as shown in Figure 6. The most important credential is probably the *Logo* of the organization, used to provide and re-enforce the *brand*; and, when some trusted authority certifies the logo or other credentials of the site, the logo of that trusted authority (e.g. certificate authority).

Our hope is that browser developers will incorporate the TrustBar as soon as possible, i.e. make TrustBar-enabled browsers. We hope to soon make available the source code of our implementation of the TrustBar (for the Mozilla browser), and we will be happy to cooperate with others on creating high-quality open source code available.

To conclude this paper, we present conclusions and recommendations for users and owners of sensitive web sites, such as e-commerce sites, for the period until browser are TrustBar-enabled; see additional recommendations in [TTV04]. We also note that even when using TrustBar-enabled browsers, viruses and other malicious software may still be able to create unauthorized transactions, due to operating system vulnerabilities. We recommend that highly sensitive web sites such as e-brokerage consider authorizing transactions using more secure hardware modules (see below).

6.1 Conclusions for Users of Sensitive Web-sites

The focus of this paper was on ensuring security even for naïve web users; however, even expert, cautious users can not be absolutely protected, unless browsers are extended with security measures as we propose or as proposed by [LY03, YS02, YS03]. However, cautious users can increase their security, even before the site incorporates enhanced security measures, by following the following guidelines:

1. Use an TrustBar-enhanced browser, using its `opportunistic logo identification` mechanism to establish logos for each of your sensitive web-pages. The authors developed and use a simple TrustBar extension to the Mozilla browser, and plan to make it available for download from their homepages soon (after some final touches).
2. Always contact sensitive web sites by typing their address in the location bar, using a bookmark or following a link from a secure site, preferably protected by SSL/TLS.
3. Never click on links from e-mail messages or from other non-trustworthy sources (such as shady or possibly insecure web sites). These could lead you to a `URL-forwarding` man-in-the-middle attack, which may be hard or impossible to detect, even if you follow guideline 1 above.
4. Be very careful to inspect the location bar and the SSL icon upon entering to sensitive web pages. Preferably, set up your browser to display the details of the certificate upon entering your most sensitive sites (most

browsers can do this); this will help you notice the use of SSL and avoid most attacks. Do not trust indications of security and of the use of SSL when they appear as part of the web page, even when this page belongs to trustworthy organizations; see the examples of insecure login pages in Figure 5, by respectable financial institutions and e-commerce sites.

5. If possible, restrict the damages due to spoofing by instructing your financial services to limit online transactions in your account to cover only what you really need. Furthermore, consider using sensitive online services that use additional protection mechanisms beyond SSL, as described below.

6.2 Conclusions for Owners of Sensitive Web-sites

Owners of sensitive web-sites are often financial institutions, with substantial interest in security and ability to influence their consumers and often even software developers. We believe that such entities should seriously consider one of the following solutions:

1. Provide your customers with a browser with security enhancements as described here, and encourage them to install and use it. We notice that the basic `TrustBar` enhancement, available in our site as of August 2004 for Mozilla, may suffice for most sites and customers. Many software integrators can perform such enhancements to Mozilla and other browsers easily, possibly taking advantage of the source code of our implementation.
2. Use means of authenticating transactions that are not vulnerable to web spoofing. In particular, `challenge-response` and similar one-time user authentication solutions can be effective against offline spoofing attacks (but may still fail against a determined attacker who is spoofing your web site actively in a `man in the middle` attack). Using SSL client authentication can be even more effective, and avoid the hardware token (but may be more complex and less convenient to the user).
3. Protect, using SSL/TLS, as many of your web pages as is feasible. In particular, be sure that every web form, i.e. web page requesting the user to enter (sensitive) information, is properly protected when it is sent to the user. Notice that many respectable companies (probably using respectable web-site designers) were not careful enough and have insecure web pages asking users to enter sensitive information, as shown in Figure 5; this is insecure (the site may invoke SSL to protect the information, but the user cannot know this is not a spoofing site – i.e. this practice allows a spoofing site to collect passwords).
4. Use cookies to personalize the main web page of each customer, e.g. include personal greeting by name and/or by a personalized mark/picture (e.g. see [PM04]). Also, warn users against using the page if the personal greeting is absent. This will foil many of the phishing attacks, which will be unable to present personalized pages.

   We also recommend that site owners are careful to educate consumers on the secure web and e-mail usage guidelines, including these mentioned above, as well as educate them on the structure of domain name and how to identify their corporate domains. This may include restricting corporate domains to only these that end with a clear corporate identity.

6.3 On the secure client requirement

Finally, we notice that even if our recommendations are all implemented, surfers using personal computers are still vulnerable to attacks by malicious software (`malware`) running on their computers, or by attackers who can use the same computer. This is the result of the weak security of existing operating systems, e.g. Microsoft™ issued 51 security advisories during 2003 alone (about one every week!). We therefore recommend, following [PPSW97, H03], to restrict the execution of sensitive transactions to trusted hardware, possibly in the form of a trusted personal device. Such a device can provide a truly high level of confidence in its TrustBar, allowing users to identify using user-name and passwords with relatively safety. Furthermore, such a device could support more secure forms of identification and authorization, such as using shared keys and one-time passwords. Finally, a mobile, personal trusted device is also the right mechanism to provide digital signatures with non-repudiation, i.e. allow the server as well as third party (e.g. judge) to validate a digital signature by the customer on submitted transactions and orders; see [H03] for details.

Acknowledgements

This work benefited from many fruitful discussions on the cryptography@metzdowd.com mailing list over the last few years, including different ideas and proposals related and similar to ours. We thank the owner and moderator, Perry Metzger, and the many participants. In particular, many thanks to Ian Grigg for his excellent, helpful comments and suggestions.

Thanks to Amos Fiat and Amos Israeli for their encouragement and helpful comments.

Thanks to the organizers of ENC04 for inviting me to deliver [He04a], and to the audience who filled in the review (see Section 5).

This work was support in part by National Science Foundation grant NSF CCR 03-14161 and by Israeli Science Foundation grant ISF 298/03-10.5.

References

[A04] Frequently Asked Questions about WebTrust, the American Institute of Certified Public Accountants, 2004.

[APWG04] Anti-Phishing Working Group, Phishing Attack Trends Report - March 2004, published April 2004, available online at http://www.antiphishing.org/resources.htm.

[BBC03] Virus tries to con PayPal users, BBC News, online at http://news.bbc.co.uk/2/hi/technology/3281307.stm, Wednesday, 19 November, 2003.

[BSR02] Client side caching for TLS. by D. Boneh, Hovav Shacham, and Eric Rescrola. In proceedings of the Internet Society's 2002 Symposium on Network and Distributed System Security (NDSS), pp. 195—202, 2002.

[C03] Tyler Close, Waterken™ YURL - Trust Management for Humans, July 2004 (first release July 6, 2003).

[CSRI04] The Coordinated Spam Reduction Initiative, Microsoft corporation, February 2004.

[Citi04] Citibank™ corp., Learn About or Report Fraudulent E-mails, at http://www.citibank.com/domain/spoof/report_abuse.htm, April 2004.

[E99] Carl Ellison, "The nature of a usable PKI", Computer Networks 31, pp. 823-830, 1999.

[ES00] Carl Ellison and Bruce Schneier**,** Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure. Computer Security Journal, v 16, n 1, 2000, pp. 1-7; online at http://www.schneier.com/paper-pki.html.

[FB*97] Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web Spoofing: An Internet Con Game. Proceedings of the Twentieth National Information Systems Security Conference, Baltimore, October 1997. Also Technical Report 540–96, Department of Computer Science, Princeton University.

[FS*01] Kevin Fu, Emil Sit, Kendra Smith, and Nick Feamster, Do's and Don'ts of Client Authentication on the Web, in the Proceedings of the 10th USENIX Security Symposium, Washington, D.C., August 2001.

[G00] Overview Of Certification Systems: X.509, PKIX, CA, PGP and SKIP, by Ed Gerck. THE BELL, ISSN 1530-048X, Vol. 1, No. 3, p. 8, July 2000.

[G04] Ian Grigg, personal communications, 2004.

[G04a] Ian Grigg, PKI considered harmful, online at http://iang.org/ssl/pki_considered_harmful.html, 2004.

[G04b] Ian Grigg, Phishing I - Penny Black leads to Billion Dollar Loss, online at http://www.financialcryptography.com/mt/archives/000159.html, June 2004.

[H03] Amir Herzberg, Payments and banking with mobile personal devices. CACM 46(5): 53-58 (2003).

[H04] Amy Harmon, Amazon Glitch Unmasks War Of Reviewers, February 14, 2004.

[He04] Amir Herzberg, Controlling Spam by Secure Internet Content Selection, to be presented in Fourth Conference on Security in Communication Networks '04, Amalfi, Sept. 2004.

[He04a] Amir Herzberg, Web Spoofing and Phishing Attacks and their Prevention, invited talk, Mexican International Conference in Computer Science 2004 (ENC-04), Colima, Mexico, Sept. 2004.

[HH99] J.-H. Hoepman and A. Helme, Secure Ecommerce: Voorwaarden voor veilig zakendoen over het Internet, MNET magazine 20/21, November 1999. Online at http://www.cs.kun.nl/~jhh/publications/MNet.html.

[HM04] Amir Herzberg, Yosi Mass: Relying Party Credentials Framework. Electronic Commerce Research, Vol. 4, No. 1-2, pp. 23-39, 2004.

[HM*00] Amir Herzberg, Yosi Mass, Joris Mihaeli, Dalit Naor and Yiftach Ravid: Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. IEEE Symposium on Security and Privacy, Oakland, California, May 2000, pp. 2-14.

[HN98] Amir Herzberg and Dalit Naor. Surf'N'Sign: Client Signatures on Web Documents. IBM Systems Journal, 37(1):61--71, 1998.

[J00] Jeff Johnson. GUI Bloopers: Dont's and Do's for Software Developers and Web Designers. Morgan Kaufmann Publishers, 2000.

[J05] Markus Jakobsson, Modeling and Preventing Phishing Attacks, submitted to Financial Cryptography 2005.

[JP03] Audun Jøsang and Mary Anne Patton, User interface requirements for authentication of communication, Proceedings of the Fourth Australian user interface conference on User interfaces, Volume 18, February 2003.

[JPH01] A. Jsang, M.A. Patton, and A. Ho. Authentication for Humans. In B. Gavish, editor, Proceedings of the 9th International Conference on Telecommunication Systems (ICTS2001). Cox School of Business, Southern Methodist University, Dallas, March 2001.

[KM00] Kohlas and U. Maurer, Reasoning about public-key certification - on bindings between entities and public keys, IEEE JSAC, vol. 18, no. 4, Apr, 2000.

[KR00] David P. Kormann and Aviel D. Rubin, Risks of the Passport Single Signon Protocol, Computer Networks, (July, 2000).

[L04] Avivah Litan, Phishing Attack Victims Likely Targets for Identity Theft, Gartner FirstTake, FT-22-8873, Gartner Research, 4 May 2004.

[L04a] Sophie Louvel, Fraudsters Go Phishing in a Million-Dollar Hole of Opportunity, Financial Insights – and IDC company, research report FIN1492, July 2004.

[LN02] Serge Lefranc and David Naccache, "Cut-&-Paste Attacks with Java". 5th International Conference on Information Security and Cryptology (ICISC 2002), LNCS 2587, pp.1-15, 2003.

[LPSW00] Lacoste, G., Pfitzmann, B., Steiner, M., and Waidner, M., ed., SEMPER -- Secure Electronic Marketplace for Europe. Berlin et al: Springer-Verlag, LNCS vol. 1854, 2000.

[LY03] Tieyan Li, Wu Yongdong. "Trust on Web Browser: Attack vs. Defense". International Conference on Applied Cryptography and Network Security (ACNS'03). Kunming China. Oct. 16-19, 2003. Springer LNCS.

[M97] Silvio Micali, "Efficient Certificate Revocation", Proceedings of RSA Data Security Conference, 1997.

[M04] Microsoft Root Certificate Program Members, Microsoft, April 2004.
[MR00] Patrick McDaniel and Aviel D. Rubin, A Response to "Can we Eliminate Certificate Revocation Lists?", (ps.gz, pdf), Financial Cryptography Conference, (February, 2000).

[MR04] N. Modadugu, and E. Rescorla. The Design and Implementation of Datagram TLS.
To appear in Proceedings of NDSS 2004.

[Mozilla] http://www.mozilla.org.

[MozDev] http://TrustBar.Mozdev.Org

[N93] Jacob Nielsen, Usability Engineering. Academic Press, Boston, ISBN 0-12-518405-0, 1993.

[NN00] Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. IEEE Journal on Selected Areas in Communications, 18(4):561-- 570, April 2000.

[PM04] PassMark Techniques, available by request from http://www.passmarksecurity.com/white.html, 2004.

[PPSW97] Andreas Pftizmann, Birgit Pfitzmann, Matthias Schunter and Michael Waidner, Trustworthy user devices. In Gunter Muller and Kai Rannenberg, editor, Multilateral Security in Communications, pages 137--156. Addison-Wesley, 1999. Earlier version: Trusting Mobile User Devices and Security Modules, IEEE Computer, 30/2, Feb, 1997, p. 61-68.

[R00] Eric Rescorla. SSL and TLS: Designing and Building Secure Systems. Addison-Wesley, 2000.
[RFC2693] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. RFC 2693, Internet Engineering Task Force, Sept. 1999.

[R95] Aviel D. Rubin, Trusted Distribution of Software Over the Internet, Proc. ISOC Symposium on Network and Distributed System Security, pp. 47-53, February, 1995.

[RFC3709] S. Santesson, R. Housley and T. Freeman, Internet X.509 Public Key Infrastructure: Logotypes in X.509 Certificates, Internet Engineering Task Force Request for Comments No. 3709, February 2004. URL: http://www.ietf.org/rfc/rfc3709.txt.

[SF9182] Multiple Browser URI Display Obfuscation Weakness, http://www.securityfocus.com/bid/9182/discussion/, Security Focus, December, 2003.

[TTV04] Anti-phishing: Best Practices for Institutions and Consumers, Gregg Tally, Roshan Thomas and Tom Van Vleck, McAfee Research, online at http://www.networkassociates.com/us/_tier2/products/_media/mcafee/wp_antiphishing.pdf, March 2004.

[WT99] Alma Whitten and J.D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In Proceedings of the 8th USENIX Security Symposium, August 1999.

[X.509 ITU-T recommendation X.509 | ISO/IEC 9594-8: "Information technology – open systems interconnection – the directory: public-key and attribute certificate frameworks".

[Y02] Yee, K.-P.. User Interaction Design for Secure Systems, University of California Berkeley Tech report, May 2002, Tech Report CSD-02-1184

[YS02] Zishuang (Eileen) Ye, Sean Smith: Trusted Paths for Browsers. USENIX Security Symposium 2002, pp. 263-279.

[YYS02] Eileen Zishuang Ye ,Yougu Yuan ,Sean Smith . Web Spoofing Revisited: SSL and Beyond . *Technical Report TR2002-417* February 1, 2002.

[Z95] Phil R. Zimmerman. The Official PGP User's Guide. MIT Press, Boston, 1995.

Appendix A: The TrustBar Web-Spoofing Survey (ENC'2004, Colima, Mexico, September 21, 2004)

1. Do you know what is a certificate authority?

Yes / No

If you answered `No`, skip the next question…

2. For each of the following, are they certificate authorities that you trust?

1. VeriSlim Yes / No / Don't know
2. Visa Yes / No / Don't know
3. Saunalahden Yes / No / Don't know
4. VeriSign Yes / No / Don't know

3. Which of the web sites on the screen is authentic, fake, or insecure?

1. Authentic / Fake / Insecure / Don't know
2. Authentic / Fake / Insecure / Don't know
3. Authentic / Fake / Insecure / Don't know
4. Authentic / Fake / Insecure / Don't know
5. Authentic / Fake / Insecure / Don't know
6. Authentic / Fake / Insecure / Don't know

4. Few statistical details about yourself…

1. Are you familiar with network security? No / Yes / Expert
2. For how long have you used the Web? Not at all / Year or less / More
3. Did you study computer science (or related area)?
   Not at all / under-grad student / graduate
   nada / licenciatura / maestria

Thank you !!! Muchas Gracias !!!

Appendix B: screen shots used in survey (question 3)

Figure 7: Question 3.1, unprotected login in Amazon site, without TrustBar

Figure 8: Question 3.2, spoofed Amazon login, without TrustBar

Figure 9: Question 3.3, protected Amazon login, without TrustBar

Figure 10: Question 3.4, spoofed Amazon login, with TrustBar

Figure 11: Question 3.5, unprotected login to Amazon's site, with TrustBar

Figure 12: Question 3.6, protected login to Amazon, with TrustBar