Security and Identification Indicators for Browsers against Spoofing and Phishing Attacks[1]

Amir Herzberg[2] and Ahmad Jbara
Computer Science Department
Bar Ilan University

Abstract

In spite of the use of standard web security measures (SSL/TLS), users enter sensitive information such as passwords into scam web sites. Such scam sites cause substantial damages to individuals and corporations. In this work, we analyze these attacks, and find they often exploit usability failures of browsers.

We developed and describe TrustBar, a browser extension for improved secure identification indicators. Users can assign a name/logo to a secure site, presented by TrustBar when the browser presents that secure site; otherwise, TrustBar presents the certified site's owner name, and the name/logo of the Certificate Authority (CA) who identified the owner. Some of these ideas are already adopted by browsers, following our work.

We describe usability experiments, which measure, and prove the effectiveness, of TrustBar's improved security and identification indicators. We derive general secure-usability principles from our experiments and experience with TrustBar.

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

[1] Earlier version of this manuscript was titled `TrustBar: Protecting (even Naïve) Web Users from Spoofing and Phishing Attacks`. Until publication, the manuscript is available as ePrint Archive: Report 2004/155, at http://eprint.iacr.org/2004/155.
[2] Contact author; addresses: herzbea@cs.biu.ca.il and http://amirherzberg.com.

million [L04a] to $1 billion [G04b]. For examples of phishing e-mail messages, see the Anti-Phishing Working Group phishing archive [APWG]. Spoofing attacks, mostly using the phishing technique, are significant threats to secure e-commerce, see e.g. [APWG06, BBC03] and Figure 1.

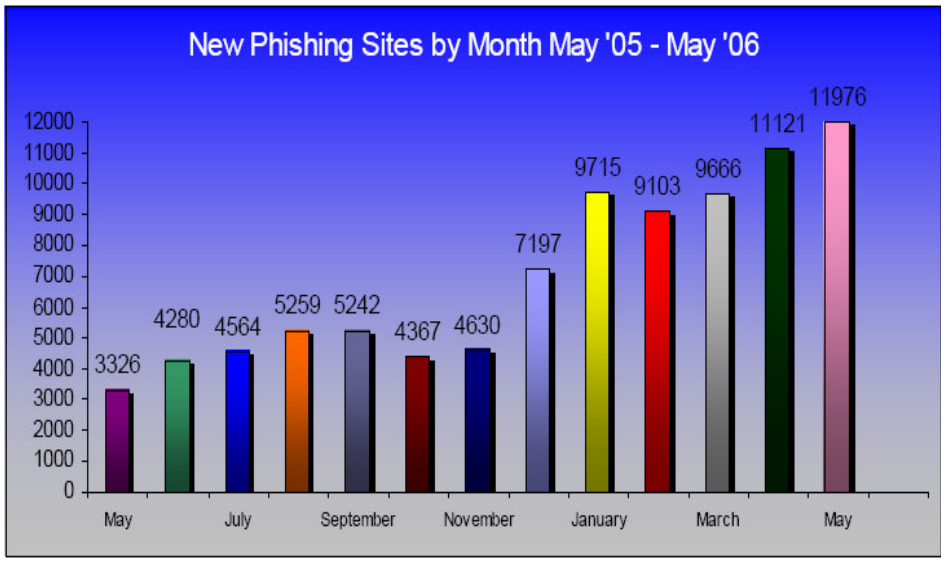

Figure 1: Phishing attack trends (source: [APWG06])

We investigate spoofing and phishing attacks and countermeasures, trying to protect naïve as well as expert users. We consider three main approaches to site identification indicators:

Standard/classical indicators: the indicators available in typical current browsers, consisting mainly of the location (address/URL) bar, and of indicators of the activation of SSL/TLS (a padlock and the use of the protocol name *https* rather than *http*).

Certificate-derived identification indicator: presenting an identifier (name or logo) for the site. If, as in current browsers, the identification is not always done by an entity trusted by the user (directly or by delegation), then we should also identify the entity responsible for the identification. Namely, in this case the identification indicator includes also a name or logo for the *Certificate Authority (CA)*, responsible for identifying the site.

User-customized identifiers: allowing users to choose a name or logo for a securely identified site, and later presenting this name/logo to identify this (SSL/TLS protected) site.

We implemented the last two approaches in a browser extension called *TrustBar*. We also conducted usability experiments, to measure and compare the effectiveness of the three approaches to sites identification

indicators. The results confirm the vulnerability of the standard indicators, as available in current browsers, and significant improvements in detection of spoofed sites, when using both of TrustBar's improved identification indicators, especially the user-customized identifiers.

1.1 Related Works

Felten et al. first identified the web-spoofing threat in [FB*97]. In this work, as well as follow-up works [LN02, LY03, SF03, YS02, YYS02] done prior to the first publication of the current manuscript [HG04], the focus was on attacks and countermeasures for knowledgeable and wary users. Specifically, the users were expected to correctly and carefully check indications such as the URL (location) of the web site and the security lock (SSL/TLS) indicator. These works showed clever web spoofing attacks, using scripts, Java applets or other `features' and bugs of common browsers, to fool even savvy and wary users. These attacks are not easy to deploy, as they require technical sophistication, and are sensitive to changes in browsers, operating systems and their configurations.

Several of these works also propose solutions, by disabling (important) browser functionalities, or using enhancements to the browser indicators 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' noticing and understanding their `signals`, which are (even) more elaborate than the existing location bar and SSL indicator. Therefore, these proposals are unlikely to be of significant value for most (naïve, inattentive) users. This intuitive conjecture, yet to be established or refuted experimentally, is based on our study and others [DTH06, WMG06], which show that most users are not able to properly use even the existing (easier) indicators.

In fact, practical web-spoofing attacks deployed so far, rarely if ever use advanced technical means; at the most, they use basic scripts and browser vulnerabilities, e.g. to present fake location bar [APWG06, Citi04, SF9182]. Essentially 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 are therefore mostly oblivious to the use of countermeasures such as proposed in [LN02, LY03, SF03, YS02, YYS02].

Still, these simple attacks are very effective [APWG06, BBC03, Citi04, L04]. We argue that this is due to several weaknesses in the user interface of the current browsers, and suggest simple extensions to browsers, that we believe will help users detect many or most of these attacks. In contrast to the above mentioned (and previous) works, our goal is specifically to provide reasonable, if not complete, protection even for the naïve user and inattentive user.

Our work is based on secure user interface principles, based on previous works on secure usability [WT99, Y02, N93, JPH01, J00]. Based on our experience and experiments, we formulated several additional secure usability principles, which are highly relevant to our particular concern but have implications to other secure-usability problems. Our approach is also based on previous works on design of secure user interface for network applications, including TINGUIN (part of SEMPER, see [LPSW00]) and [HN98, H03, JP03].

Following early versions of the current manuscript [HG04], several works evaluated the ability of typical web users to detect spoofed sites, using `classical` browser security and identification indicators, as well as new approaches. A study by Dhamija et al. [DTH06] shows that the classical indicators are not good enough to protect users against malicious websites; 90% of the participants were fooled by specific fake site. They concluded that a new approach must be examined as an alternative for the current standard. In their study, Dhamija et al developed 3 attack strategies (lack of knowledge, visual deception and bounded attention) based on the phishing archive maintained by the APWG. To summarize, participants made incorrect decisions because they did not know how computer systems work in particular security systems and indicators. Also, more experienced participants were fooled by visual deceptions.

Wu et al. [WMG06] tested several toolbars, including TrustBar, with certificate-derived site identification. They found high spoofing rates with all toolbars, and conclude that 'security toolbars' are ineffective. They conjecture that popup warnings, at the right time with the right message, may be more effective.

However, there are several problems with these conclusions. The most significant is that half of the sites tested in [WMG06], did not use SSL/TLS at all. For such sites, TrustBar cannot improve detection rates – it only presents an indicator that SSL/TLS is not active. Hence, TrustBar is at best irrelevant for these sites. Worse: new TrustBar users are alarmed by the `no SSL` indicator, especially in sites that they expect to be protected (more details within). As a result, it is not surprising that in [WMG06], users ignored TrustBar's indicators, even when reaching a spoofed version (without SSL) of an SSL protected site.

Furthermore, the only description of the toolbars in the tests of [WMG06] was by an email message at the middle of the test, or when users (rarely) clicked on a `what's this?` link in the toolbars. Results in the second part of the test, after receiving this email, were significantly better than in the first part. However, while both modes of TrustBar (certificate-derived and user-customized) are simple to use and explain, such identifiers are still non-trivial and an email message, in the middle of an experiment, is hardly sufficient. A realistic usability test should provide a reasonable level of introduction of any new mechanism, balancing this with the risk of causing users to be overly-aware of security. Notice also that [WMG06] only used certificate-derived identification with TrustBar, and did not use user-customized identification, which achieved better results in our experiments.

In our tests, we provided users with a few minutes describing the different relevant indicators and their usage, before presenting them with a time-intensive task of identifying fake sites. Our testing may, therefore, suffer from increased user attention and awareness to security, but it does allows us to fairly compare between different site and security identification mechanisms. Correct evaluation of the actual (and not just relative) effectiveness of security indicators, requires an experiment following appropriate user-education, done over long period of time to avoid favorable detection rates due to increased security awareness.

Our specific goal is to allow (most) users to distinguish between websites they trust for some sensitive information, e.g. a specific login page, vs. `spoofed` web sites which try to mimic the trusted sites.

Another approach to identify `spoofed` sites is by building a classifier, based on heuristics, expert-rules and/or on machine learning, to distinguish `spoofed' sites from `correct` sites, and block access to them or at least provide highly visible warning to the user. This approach was proposed by Chou et al. in *SpoofGuard* [CL*04]. SpoofGuard uses multiple sources of information including domain name, url, links and images in the page, as well as historical data, to evaluate the likelihood that a given page is spoofed; this is indicated to the user by a three-color code (red, yellow, green), with additional pop-up warning for especially suspect sites. However, once such a classifier becomes popular, the attackers can design sites carefully to avoid classification as `spoofed`. The SSL/TLS based identification mechanisms, evaluated in this manuscript, can complement such classifiers.

Yet another classifier approach, implemented in several commercial products, is to maintain lists of spoofed sites (`blacklist') by domain name and/or IP address, and again present the status of the site to the user by three color code (use yellow for sites on neither lists). This approach is currently very effective, with several efforts to create comprehensive black list services. However, the cost, to attacker, of changing IP addresses is small, and the cost of changing domain name is negligible. Therefore, there is a significant risk in relying on blacklists for blocking spoofed sites for the long or even medium term.

TrustBar's certificate-derived identifiers are the organization name (taken, by default, from existing site's SSL/TLS public key certificates), or – preferably - by logo. We believe that a logo is more visible, and by using it, users are more likely to notice an unprotected site (by having an appropriate warning and no logo) or an incorrect logo (also approved by the user or by an authority trusted by the user and sufficiently different to avoid confusion). Logos are preferable to automated, arbitrary assignment of `random` images to sites, as done by Tay [T04].

Currently, users have to manually select the logos, but we also allow authorities trusted by the user to certify logos, e.g. using the format in the draft standard `X.509 certificate extension for logotypes` [RFC3709], whose goals include identifying web sites. In particular, Josang et al. [JPH01, JP03], 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. Another 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. Earlier, Close [C06] also implemented a toolbar (`petname toolbar`), allowing users to select (textual) `pet-names` to identify web sites.

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. We briefly mention few other phishing attacks, which should be addressed by other mechanisms. See complementing countermeasures against phishing attacks in [J05, E04].

1.2 Organization

In Section 2 we review web spoofing threat models, attacks and current defenses. We also show that users, and even designers, do not notice the lack of protection on sensitive web pages. From this, we derive design criteria for protecting naïve and off-guard users against spoofing (and phishing) attacks, and secure usability principles; we present both in Section 3.

In Section 4, we present the specific user interface design of TrustBar, based on the design criteria and usability principle from Section 3.

In Section 5, we present results of an empirical user studies, evaluating the effectiveness of different security and identification indicators for browsers. 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.

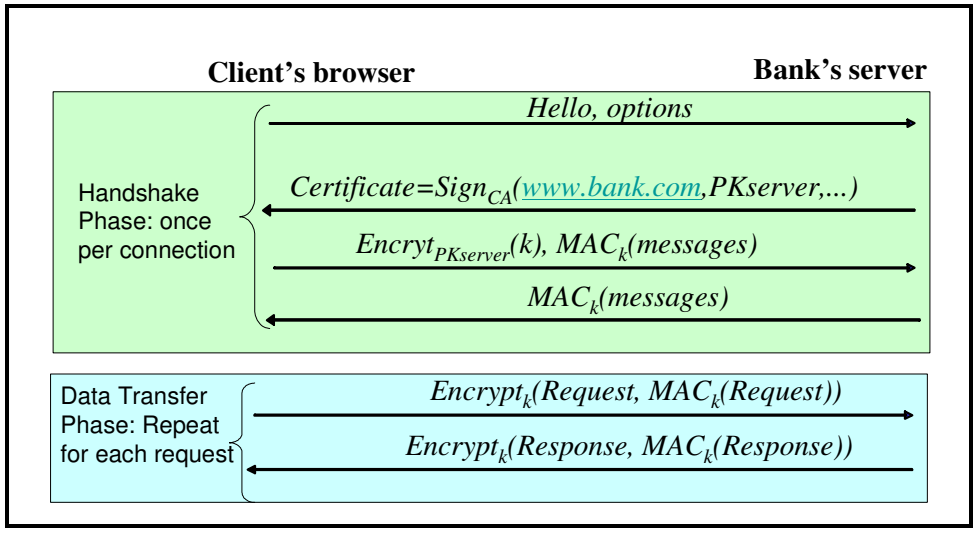

Figure 2: Simplified SSL/TLS Operation

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.

[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.

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 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

[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.

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 may not be too difficult to obtain valid yet false certificates from a CA listed in the default list of major browsers.

Furthermore, there are several certificate authorities who issue very low-cost certificates (e.g. for under 10$), by limiting the validation on an automated validation of the ownership of the domain name, by lookup at the *whois* database and performing e-mail request-response validation. Using current browsers, users are normally exposed only to the domain name, and not to other details in (more expensive) certificates; therefore such (low-cost and easily available) certificates are available to attackers. The best solution to this is to allow only certificate authorities which are trustworthy, and perform serious validation of identity before issuing a certificate, either by the browser vendor, or by a `security service provider` chosen by the user to make trust decisions.

Furthermore, currently, many 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].

Attackers can often also use a certificate from a non-trustworthy CA. 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].

Therefore, the certificate management in current browsers is vulnerable. However, there were few reports of a `real` false certificate attacks. 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).

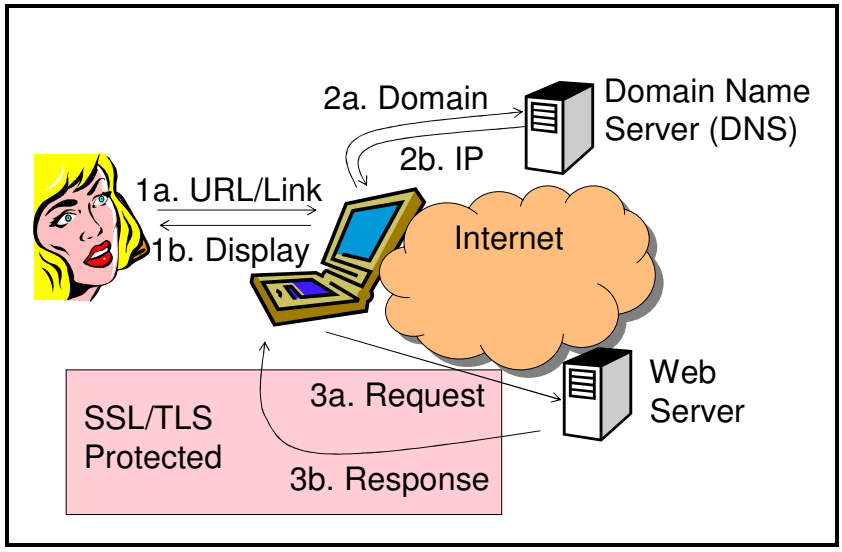

Figure 3: HTTP request/response process with SSL protection

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-to-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 one reason is that most (naïve) users are not aware of the structure of URL and domain names and their relation to ownership of the domain. Another reason is that 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.

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.

Attackers usually use an even easier method to direct the user to the spoofed site: *phishing 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 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. 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).

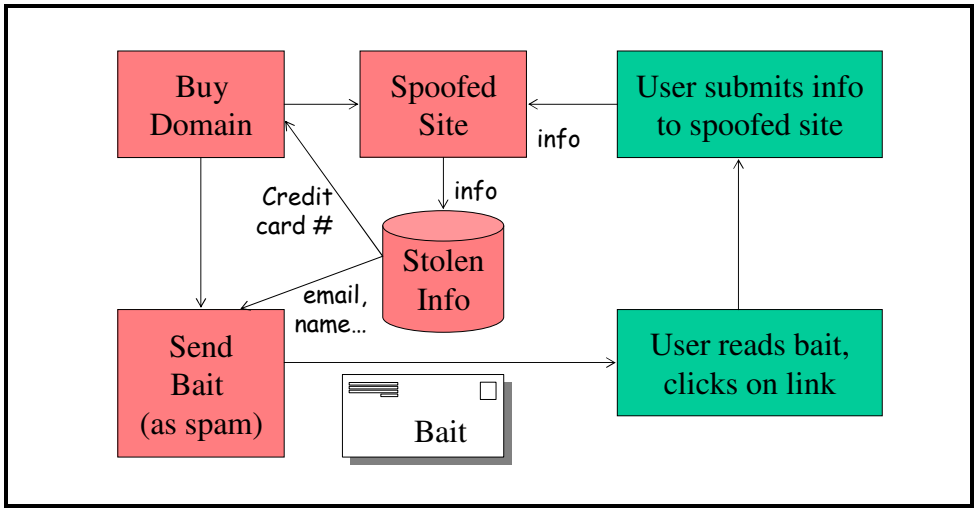

Figure 4: Process of Typical Phishing Spoofing Attack

Currently most phishing attacks lure the users by using spam (unsolicited, undesirable e-mail), as described above. TrustBar focuses on preventing web spoofing; this helps only against phishing/spoofing attacks, which lure the user into directing his browser to approach a spoofed web site. Such attacks do not always use e-mail, e.g. an attacker could use ads to lure users to the spoofed site. Other phishing attacks use the `sting`

[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).

approach, where they lure the user to their (malicious) site under some false promises, such as provision of passwords or credit card numbers. Proposals for controlling and preventing spam, e.g. [CSRI04, He04], may help to prevent or at least reduce spam-based phishing; however, content based filtering, the most common protection against spam currently, is not very effective against phishing since most phishing attacks emulate valid e-mail from (financial) service providers, and are likely to pass content-based filtering.

Many spoofed sites 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. Furthermore, users are not sufficiently aware of the distinction in security between browser-controlled fields, such as the location bar, and site-controlled fields and areas. This confusion is partially due to the flexibility of most modern browsers, which allow sites extensive controls over the content and appearance of web pages.

- 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

[6] The *ip02.com* domain is used by TD Waterhouse™, and the *4t.com* domain was used by a spoofed web site.

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, e.g. Microsoft's HotMail™, 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.*

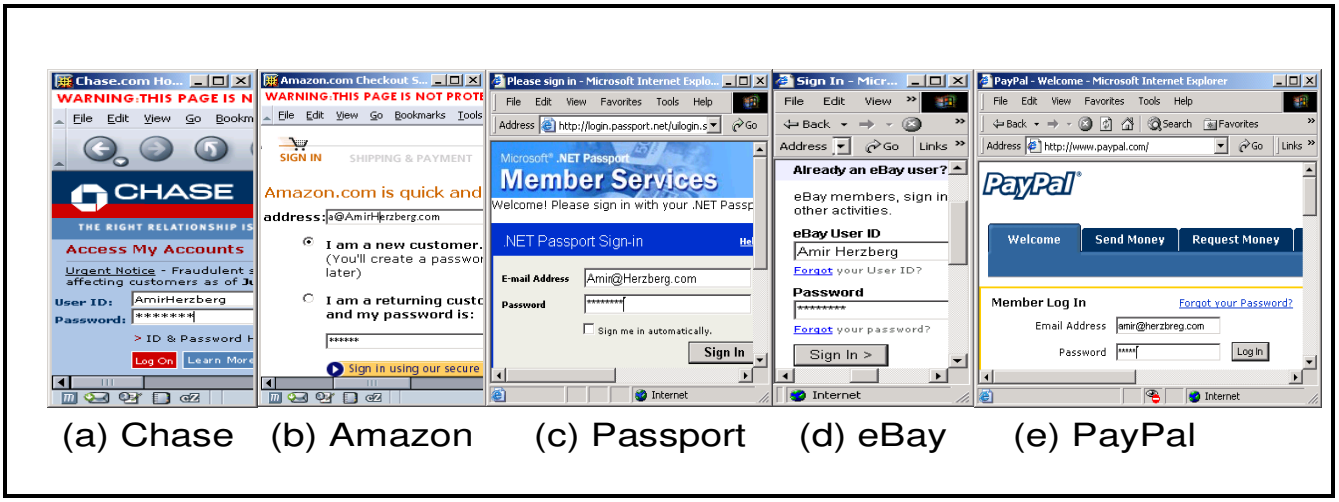

Figure 5: Unprotected Login to Important Sites.

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.

3 Design Criteria and Secure Usability Principles

We now present design criteria for prevention of web and credential spoofing, extending the criteria presented in [YS02], and then several secure usability principles, extending the principles in [WT99, Y02, J00].

- **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, since they are responsible for the identification of the sites; 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. Note, that instead of requiring users to develop trust in each CA, it would be better if users can trust some service that will validate trustworthy certificate authorities, in which case there is not need to present the CA to the user (such service could be done by the browser vendor or a security service provider, e.g. anti-virus).

- **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.

- **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 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 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.

3.1 Secure Usability Principles

We now present several secure-usability principles, which we applied in our design; these principles are based on previous works and our own experience and experiments (presented 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 fourth 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 area 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.

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 TrustBar as a 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 initial 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 `security indicators spoofing` 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.

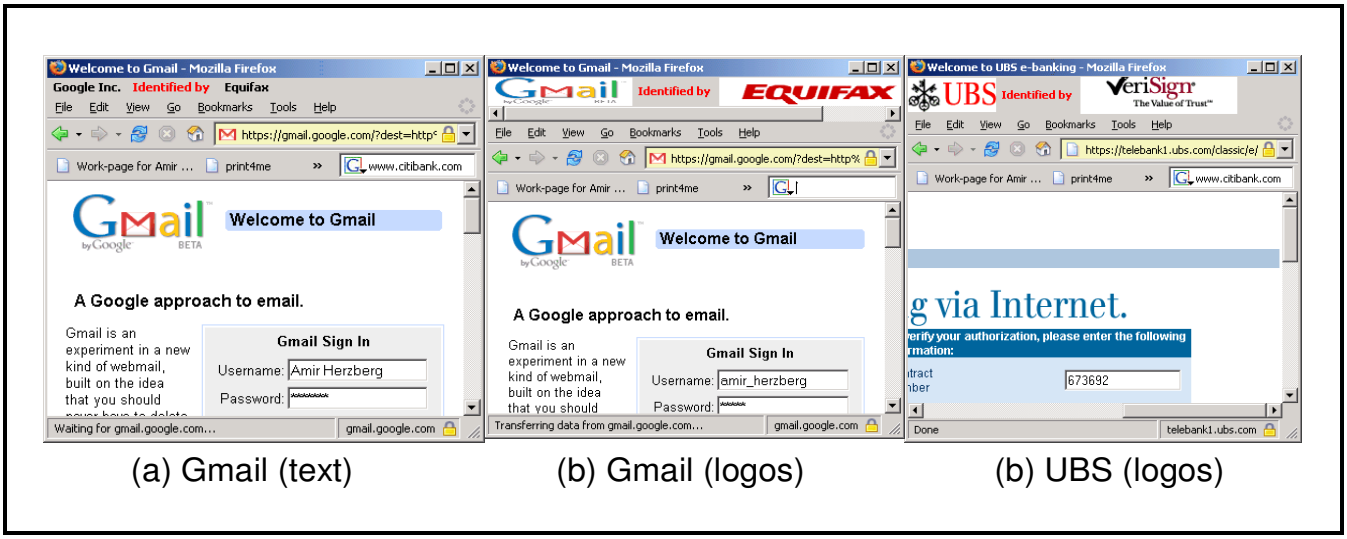

Figure 6: Screen-shots of secure sites with logo in TrustBar

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

[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.

bank). Indeed, most real-life spoofing (and phishing) attacks are 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 message (see Figure 5 (a), (b)) or icon. We recommend that corporate and other serious web sites avoid this warning message, by protecting *all* of their web pages, and certainly all of their web forms[9], preferably presenting the corporate logo in 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 TrustBar, using and re-enforcing the brand of the organization.

A possible concern is that users may be irritated by the TrustBar warning, which would appear on most web pages (since most are unprotected). As a result, more recent versions of TrustBar display a less obtrusive, graphical warning.

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

[9] 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.

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, by default, both site and the certificate authority (CA) which identified the site[10], 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 certificate-derived and user-customized identifiers for sites, by logo or name:

- *Certificate-derived identification:* Names are taken from the `organization name` field of the existing X.509 SSL certificates. Such names are presented together with the text `Identified by` and the name or logo of the Certificate Authority (CA) which identified this site. 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.

- *User-customized identification:* The user can identify a logo for a site, e.g. by `right-click` on an image of the logo (which usually appears on the same page). Users can also select a textual site identifier (a `petname'), presented by TrustBar to identify the site. Whenever opening a page with the same public key, TrustBar automatically presents this logo or petname for the site.

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.

[10] The identification of the CA can be omitted, if it is known to be trusted by the user, or if the user trusted a service or vendor who delegated that trust to the CA; unfortunately such mechanisms are not available in current browsers.

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.

4.3 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`. This may facilitate `grass-root` adoption of logo certificates, which may expedite the deployment of trustworthy, established logo certificate authorities.

4.4 `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. Secure identification indicators, such as 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.

4.5 Improve Security with 'Hey!'

In some versions of Trustbar, we experimentally added a module that randomly modified the site identifications, to test user awareness and ability to notice spoofed sites (with fake indicators), and to train users to detect spoofed sites. This was done by adding a 'Hey!' button, which users should use when the site appears suspect as possible spoofed site.

The `Hey!` indicators also allow us to identify spoofed sites, using reports generated by installations of TrustBar. Once a user suspects that a site is fraud, he just needs to click the 'Hey!' button to report that site. The Trustbar team will confirm the report and save it in central database. Each Trustbar-enabled browser will contact this database periodically to be updated with the last reports.

The more aggressive behavior of the 'Hey!' module, is that it simulates attacks, trying to enhance the users' awareness towards the real attacks. However, user response to the simulated attacks was very negative. Even reducing the frequency of the simulated attacks, and allowing users easy control over them, did not change the negative response. We therefore removed this mechanism.

4.6 Spoofing the Identification Indicator (TrustBar) itself?

If a new security and identification indicator such as TrustBar is widely deployed, it could become itself a target for spoofing. Some attacks may be prevented by appropriate support from the operating system and/or the browser, e.g. an attack that displays a browser window where the real indicator is removed and a fake one is presented instead, can easily be prevented by a browser which always presents the identification indicator in every window. While our implementation contains some defenses against such attacks, these details can change in different implementations and hence we do not elaborate on them.

However, such mechanisms may not be able to prevent all such attacks. For example, suppose the adversary presents a regular page, and within that page, runs an applet that emulates (fakes) the behavior of another browser window, including (fake) security and identification indicators.

The best defense against such attacks is probably to use customized indicators, so that the spoofed indicators will be different, and probably easily distinguished, from the customized indicators. This is another advantage of using the user-customizable indicators.

A similar solution is also possible for certificate-derived indicators, by cooperation of the sites or of a trusted third party. This solution requires each site to use one of a large set of different identifiers, e.g., logos in different colors, or logos combined with pictures from a large selection; let us call these different identifiers *skins*, and let $\{s_i | i=0,1,\ldots n\}$ be the set of skins for a particular site, whose public key is e. Each implementation of the identification indicator will use a secret key k and a pseudo-random function f to select a skin, e.g. when contacting site with public key e, we use skin number $f_k(e) \bmod n$. The skin number can be sent to the server (encrypted using e), or the server can send all skins to the client. This solution is more complicated technically but still very efficient, and can support the case where users are not willing to be involved in the choice of the indicator.

5 Secure Usability Experiments

In this section, we present an experimental user study that compares between three site identification approaches: classical identification, certificate-derived identification and user-customized identification. The results show a noticeable improvement in forgery detection when using both non-classical approaches, with some advantage to the customizable approach. The results also confirms our belief, that the classical identification approach is vulnerable even for non naïve users, and even when using only simple attacks, where the `classical identification' should perform at its best (compared to certificate-derived and user-customized identifiers). All experiments used the Firefox browser; for the certificate-derived and user-customized identifiers, we used the TrustBar extension.

Our experiments results confirmed the common belief that users (experiment participants) fail to identify fake sites, when using only the standard browser indicators of URL and SSL/TLS indicators (padlock, etc.). On the other hand, we found significant improvement in the detection rates when using both improved site identification mechanisms. These results may seem to contradict other studies [WMG06] of the effectiveness of `security bar` indicators; we later discuss some possible explanations for this seeming contradiction.

Specifically, our experiments compared fake site detection rates, for three site-indicator approaches:

Classical browser security and identification indicators – the location/URL and SSL/TLS indicators. Site identification indicators are available in typical browsers, and include the address bar and the SSL/TLS indicators (padlock in the status bar and at the end of the address bar, protocol name `https` in the address bar, yellow background). See Figure 7.

Certificate-derived indicators of organization and identifying-authority (CA). Identification of SSL/TLS protected sites, by display of the name or logo of the organization, and the name or logo of the certificate authority, as in Figures 8. This is the default identification of SSL/TLS protected pages, using the TrustBar extension (version 0.4) running in the FireFox browser. (Version 1.5.0.4).

User-Customized identification indicators. Identification of sites, by displaying a user selected logo or name for the site (Figure 9). The idea of name assigning was introduced independently by PetName [C03].

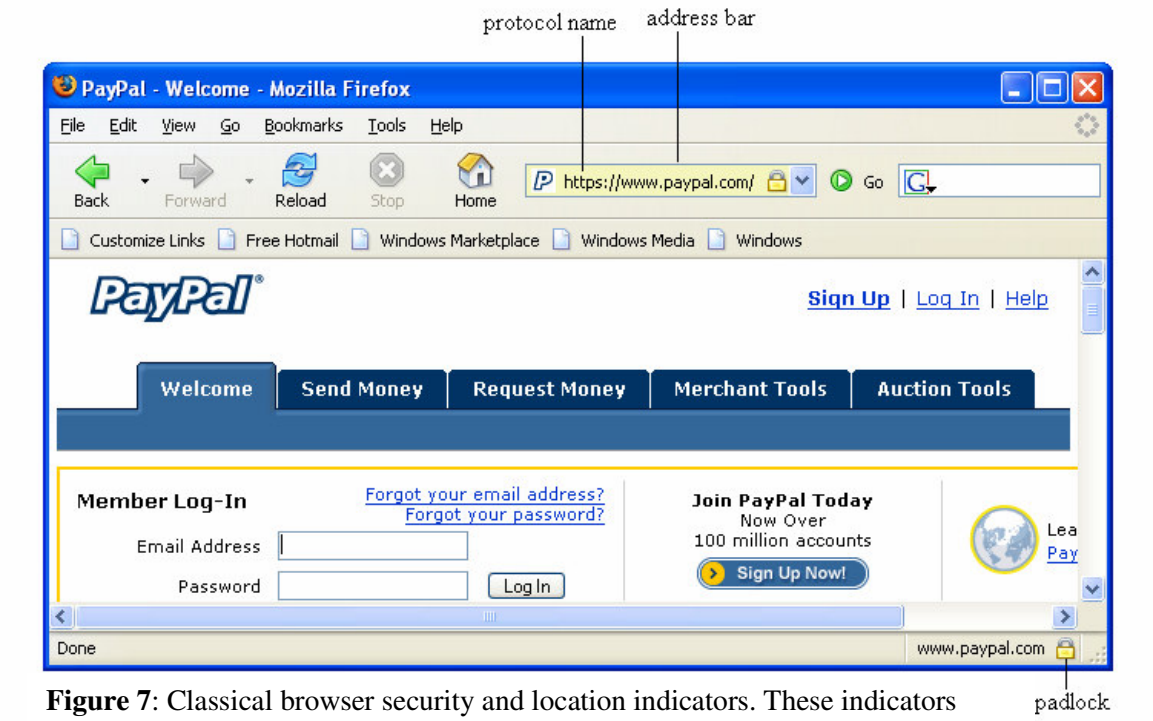

Figure 7: Classical browser security and location indicators. These indicators include address bar, status bar, padlock. They all considered vulnerable so attackers forge them to build malicious sites.

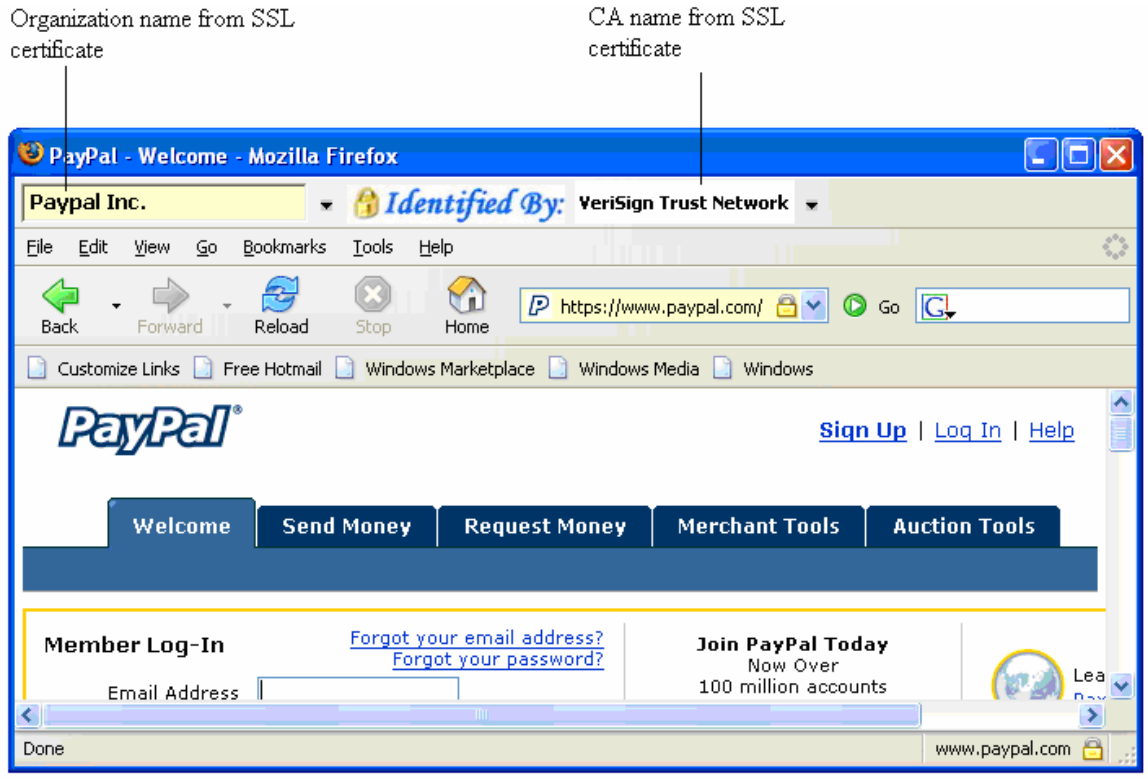

Figure 8: Site identification by 'Paypal Inc' identified by 'Verisign' using Trustbar. This is a default data obtained from the SSL certificate. It may be modified.

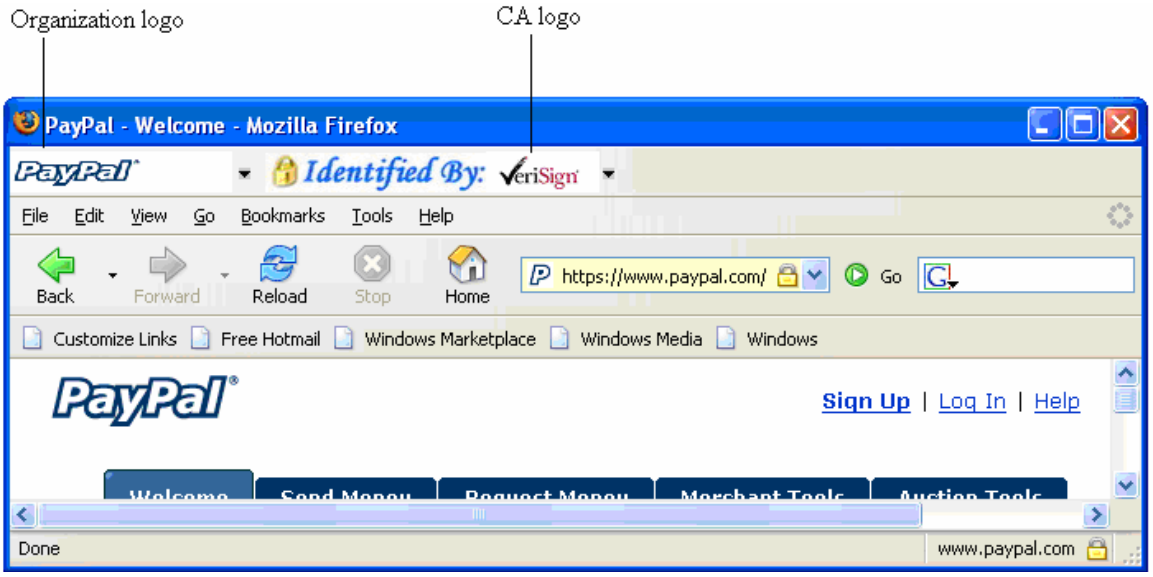

Figure 9: Site identification by 'Paypal logo' identified by 'Verisign logo' using Trustbar. The logo of Verisign presented automatically as VeriSign is widespread well-known CA. Nevertheless user may replace its logo with his/her own.

In this section, we describe three preliminary experiments. These experiments had different drawbacks and limitations, as we describe, which made their results not sufficiently trustworthy. We then describe our main experiment .

5.1 Real-use experiment

Initially, we thought that a long term experiment, integrated with training, would help users to be more aware and hence raise their detection rate over time, while also providing us with useful data for analysis. This was one of our motivations in developing the `Hey!' module, as an additional functionality for TrustBar

The `Hey!' module collects details about users upon Trustbar installations, and simulates attacks on sites that the user visiting during user's normal usage of the browser. Each time the user detects an attack, she should simply click on a `Hey!' (report fraud) button. If this was a simulated attack, we inform the user that she detected a training attack correctly; otherwise, we consider this a `real` spoofed-site detection report, and list it in a centralized database. If the user does not detect a simulated (training) attack, then TrustBar presents a `popup` alert, with some details about the undetected attack.

However, users were not willing to cooperate and use this mechanism; we received a lot of complaints, and worse – we found that almost all users disabled the `Hey!' training attacks, and many simply stopped using TrustBar to avoid them. We believe users were disturbed by the popup warnings, and many users felt less secure due to these warnings. The data we collected was not sufficient for meaningful analysis, so we aborted this experiment. We conclude that users are not very receptive or tolerant to `training attacks`; this approach, which initially sounded promising, seems hard to deploy in practice.

5.2 `Dry run` experiments

Our first usability testing was done by a written survey, filled by participants in two lectures given by the first author. The first lecture was on September 2004, during an invited talk to the Mexican International Conference in Computer Science (ENC-04); we received 37 responses from computer science students or graduates, with web-experience (we ignore here the few other responses). The second lecture was on January 2005, to third-year students in computer science in Bar Ilan University; we received 75 responses, all of which were of students with web-experience. We presented the participants with a sequence of questions, including several screen shots of web sites, which they had to categorize as protected, unprotected or fake. The results of the two groups were very similar, therefore we give only the overall percentage results.

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 `classical` browser indicators (location bar and padlock icon), and when using TrustBar's certificate-derived site identifiers. Due to the test method, it was obviously impossible to test the user-customized site identifiers.

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; 78% of the respondents, indicated that they know the meaning of the term `certificate authority` (CA).

However, do they *really* know what is a CA? 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 52% of the responses identified VeriSign as a trusted CA. 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 usage of an `identified by` field, identifying a brand, e.g. VeriSign, users trust will increase).

At the time of these tests, most browsers included Saunalahden in the list of `trusted` certificate authorities. However *none* of the users identified Saunalahden as a trusted CA. We believe the reason is that users expect the browser to make such trust decisions correctly; however browser vendors do not accept this responsibility, and expect users to make the ultimate trust decision. Additional indications to this are the replies regarding Visa and VeriSlim. Only 20% of the responses correctly identified VeriSlim as *not* being a trustworthy CA; and only 16% identified Visa as *not* being a (trustworthy) CA.

To summarize, the replies we received in this area, indicate that the use of technical terms like `certificate` and CA makes it *harder* for users to understand security indicators and controls, and therefore reduces the security.

We next tested the `classical` browser security and identification indicators. We first briefly explained the indicators (padlock, location bar). We then presented three screen shots, each for 10 to 15 seconds. The first screen shot was of an unprotected login form; the second, a spoofed version of the unprotected site, in a different domain; and the third, a protected login form.

The results confirm that most users are not able to discern between protected, unprotected and spoofed (fake) sites. In fact, only 18% correctly identified the unprotected site, only 20% correctly identified the protected site, and only 42% correctly detected the fake site as either `fake` or `unprotected`. Overall we find that although these users were clearly aware of the need to look for the security indicators, they failed in most identifications. This shows that the current browser indicators are not sufficient for security.

Finally, we tested certificate-derived site identification, using screen-shots of login forms using the FireFox browser with the TrustBar extension. We briefly explained the security indicators, and then presented three screen shots as before, but using a browser equipped with TrustBar.

The results indicate that the use of TrustBar's certificate-derived identifiers, 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 39%, 42% and 75%).

5.3 Unbounded detection time experiment

We next conducted an experiment to compare the detection rates with the different site and security indicators (classical, certificate-derived i.e. no customization, and user-customizable). We asked participants to distinguish between fake and authentic sites; we presented the same set of sites, each time with a different security and identification indicators approach. In this experiment, we allowed users to spend as much time as they liked in their efforts to identify the sites.

We aborted this experiment after doing it with seven participants, since we found that users spent absurd amounts of time trying to identify the fake sites, which was clearly disproportional to the amount of time users will actually spend in day to day web usage. Therefore, we aborted this experiment and changed to the time-limited experiment (described in the next subsection). However, there are still some interesting observations that can be made regarding the results, presented in Figure 10. Notice, however, that due to the small number of participants, these observations also require additional validation.

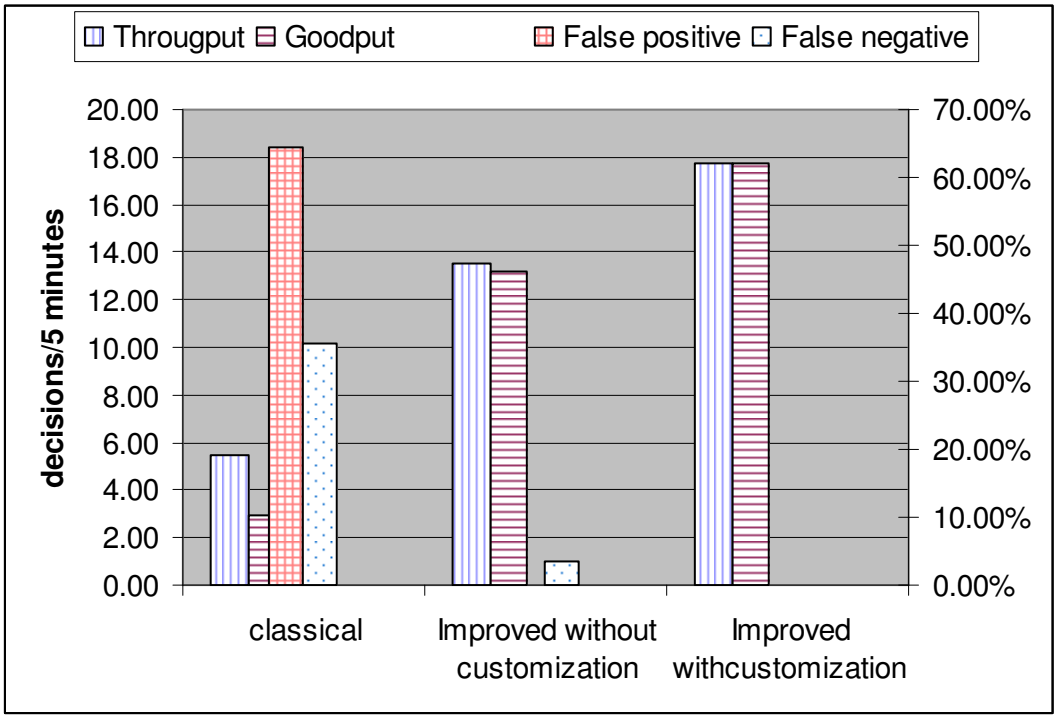

Figure 10: Results of the Unlimited-Time Experiment. Notice the significant difference between the classical site identification and the improved site identification approaches.

We first note the amazing amount of time that users dedicated to the effort to detect. We measure the rate in which users made their decisions by the average number of decisions per a five minute interval, and measure two values: the *throughput*, i.e. total number of decisions, and the *goodput*, i.e. number of correct decisions. Both throughput and goodput are very low when users were using the `classical' browser indicators (location bar, padlock, etc.): throughput was 5.5, i.e. roughly a decision per minute, and goodput was only 3 decisions per five minutes. Clearly, users made every effort to correctly detect the fake sites.

It is interesting to note, however, that even with such absurd effort to distinguish fake sites, there were still many failures when using the `classical` (URL and SSL/TLS icon) indicators. Specifically, we got an absurd rate of 63% false positives, i.e. users suspecting a valid site as being a fake site. This is another indication, that users, in this experiment, focused too much on security; in real life, very few users suspect the same sites!

However, it is interesting to note that in spite of all these efforts and focus on security, users *still failed to detect fake sites*. Specifically, when using `classical` browser identification and security indicators (URL, padlock), with unlimited time and enhanced security awareness, users still failed to detect 36% of the fake sites

(`false negative` errors). Considering that we have 63% false positives, we find that even in these favorable conditions, *the use of `classical` browser indicators did not help users at all to detect fake sites.*

The situation is dramatically different when using both improved site identification indicators supported by TrustBar, namely the certificate-derived indicators as well as the user-customizable indicators. We see dramatic decrease in the error rates: with both types of indicators, users had *no false positive errors*, and either *no false negative errors* (with user-customized indicators) or *under 5% false negative errors* (with certificate-derived indicators).

Furthermore, with both types of indicators we see significant improvement in the rates. Since error rates are so low, the throughput and the goodput are almost identical. However, we found significantly higher rates with the user-customized identifiers: 13 decisions per five minutes with certificate-derived indicators, and almost 18 decisions per five minutes with user-customizable indicators.

To conclude, this experiment gave a rather clear indication of the significant value of the improved identification indicators compared to the classical (URL and padlock) indicators, which were found to be of negligible value. However, we also found that if presented with only a security challenge, users often respond by investing disproportional efforts and time in detecting the fake sites or attack, which clearly results in skewed results. We therefore aborted the unlimited-time experiment, in favor of the time-limited experiment, described in the next subsection.

5.4 Time-Limited Experiment

We now describe our main and most detailed experiments, to measure the effectiveness of the three site identification approaches. Each participant was presented with random sequence of links, half of them to the real bank site (Figure 11) and the other half to fake sites, chosen randomly among the sites in Figures 12-14.

Participants were asked to identify whether each site was authentic or fake, but to focus on maximizing goodput, i.e. the number of correct decisions per minute, for a fixed time interval, of five minutes, for each of the three approaches. By putting the emphasis on goodput and limiting the time, we hoped to reduce the focus of users on the security, and have a decision making process which is closer to that of real users.

We note that due to the limited time of the experiment, we still expect that users were too much security-aware. Further research can reduce or eliminate this extra awareness to security, however we did not have the resources to conduct such experiments. We therefore limited our goal to confirm the improved spoofing detection ability using the improved identification and security indicators (as implemented in TrustBar), as we conjectured based on our experience and intuition, as well as based on the previous studies we conducted (see above).

We tried to be careful to avoid giving a favorable bias to the new, improved indicators, and made several choices to conservatively under-measure the advantage of the improved indicators over the classical indicators (URL and padlock). In particular, actual attacks are not perfect and may be detected by some `traces`, which may exist with `classical` indicators as well as with the new indicators. Therefore, to avoid a bias against the `classical` indicators, we used relatively simple and imperfect `site cloning` techniques, which made it relatively easy to detect the fake sites (even with `classical' browser indicators). We believe that users who lured under such attack will be lured under more sophisticated attacks, especially using the `classical' indicators.

We performed the experiment on a standard personal computer, running Windows XP, FireFox version 1.5.0.4, and with a fast Internet connection. Participants in this experiment were 23 undergraduate computer science students, in the Natanya College, Israel. The participants include 17.3% female and 82.7% male. The participants' age average was 25 and it ranges between 19 and 50. The average of participants experience using the web was 7.7 years.

We used the e-banking site of the Bank Hapoalim, the largest bank in Israel, as the site to be identified (and impersonated); 91% of the participants use e-banking, and 31% of those who use e-banking actually use the Hapoalim e-banking site. Again, our choice of a familiar banking site (and with imperfect cloning) should further assist users in detecting, even with the (weaker) `classical` identification and security indicators.

We created three fake versions for this e-banking site. All three fake versions completely duplicated the content of the original bank site; they differ in the attack (deception) technique. Specifically, we used the following three attack techniques:

- **Unprotected site with incorrect URL:** In this attack the fake site is located in an unprotected web page, with the same visible contents as the original site, placed in a domain controlled by the adversary (but often similar to the original address). Attackers hope that users will not notice the lack of SSL indicators or the different domain name supplied. For the experiment we removed one character (http://login.bankapoalim.co.il) from the original domain name (http://login.bankhapoalim.co.il) and configure the windows hosts file accordingly.
- **Protected site with incorrect URL:** The second fake site is a secure version of the previous copy; we duplicated the bank content, created a self signed certificate for incorrect URL (as the previous). This corresponds to an attacker which buys a (low-cost, low-assurance) certificate to their cloned site – an easy attack, although not very common yet. Our results show that attackers have a good reason not to bother to get such certificates; the chances of this attack to succeed are not significantly higher than of the `unprotected site` attack. Technically, to implement this attack we installed a fake server certificate to be installed on the localhost. We used the OpenSSL tool to create this certificate.

- **Fake address and status bar**: The third fake site was created by copying the bank source and saving it in a personal address. However, to trick the user, the real address and the status bars were replaced, not perfectly, by images captured from the original bank. To hide the original bars we used simple Javascript code.

The experiment participants have some advantages when compared with real users. All of them got an introduction about spoofing identification indicators, and they are all computer science students, with experience using the web. Due to these advantages we think that typical, unwary and naïve users will achieve worse results, especially in the classical approach (which requires more technical understanding, and is more familiar to the test participants).

Each participant did three tests, one for each of the identification and security indicators approaches. In the first test users used the classical approach, in the second they used browser with certificate-derived identification, and in the third test they used user-customizable site identification. Both of these improved identification indicators were provided using (different options of) TrustBar.

The experiment started with a five minutes, one-on-one introduction, about spoofing and the different identification indicators; for fairness this covered also the classical indicators, although the participants are supposed to be familiar with them. This introduction is given before the tests, so it may constitute an advantage to the classical approach that is tested first (and described fully in the introduction). On the other hand, one could suspect that the order of the tests may constitute an advantage to the improved identifiers due to learning effect; however we confirmed that this learning effect is small (see below).

After the tutorial the participants were required to fill a web form with statistical details, including email address, web seniority, e-Banking experience and security background.

Once this is done, the user is is presented during 5 minutes with a sequence of links chosen randomly, using the browser without any extension (classical security and identity indicators). The user clicks on each link, and as a result the clicked site is presented in a new window. The user freely checks the site, closes the window and decides whether it was real or fake. Automatically the system advances to the next link. After five minutes are done, the test is repeated for five more minutes using the certificate-derived indicators, and then for five more minutes using the user-customized indicators.

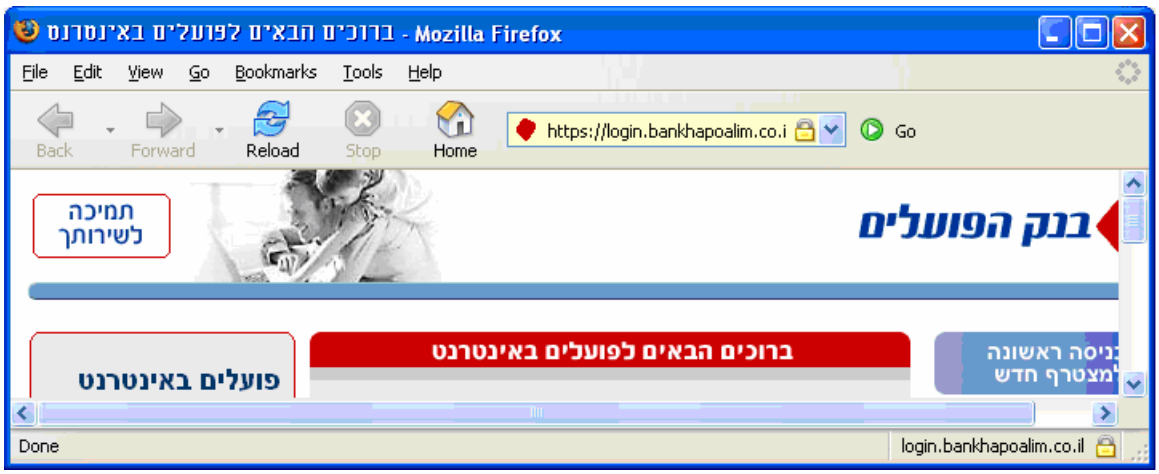

Figure 11: Real Bank Site. This is the legitimate site of 'Bank Hapoalim' visited with Firefox browser. You should notice here all the identification indicators such as the location bar, status bar and padlock.

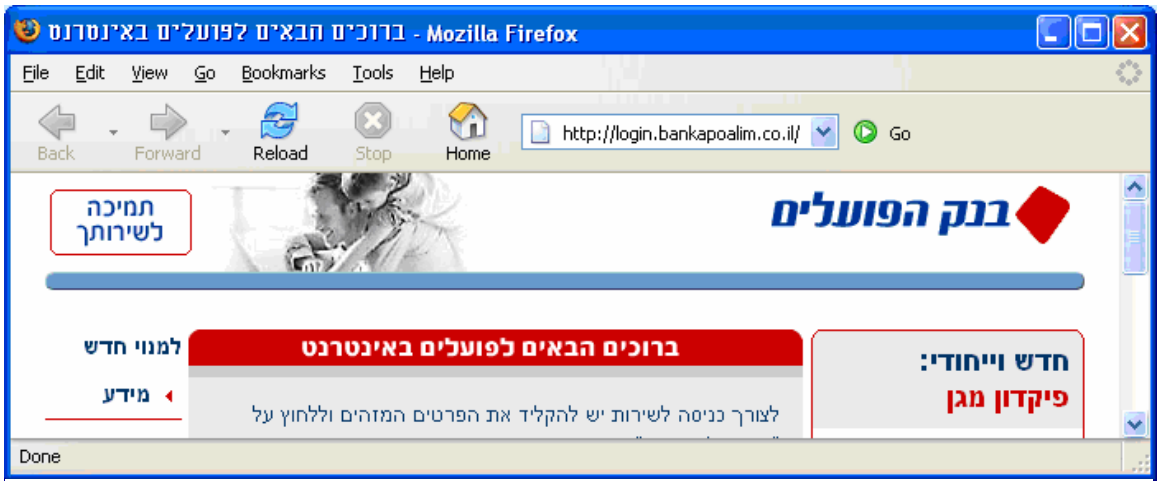

Figure 12: Fake Bank – Unprotected Version. This is a malicious site of the 'Bank Hapoalim'. It is presented as unprotected under different similar domain name. Notice the missing 'h' in the domain name. The real domain name is bankhapoalim.co.il.

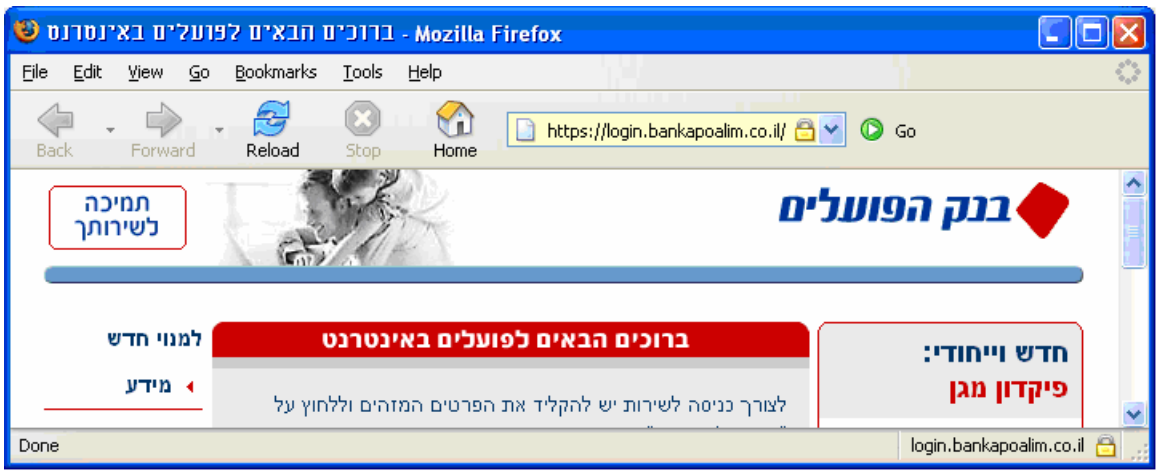

Figure 13: Fake Bank – Protected Version. This is a malicious site of 'Bank Hapoalim'. It is presented as protected, but using a different domain name and a self signed certificate.

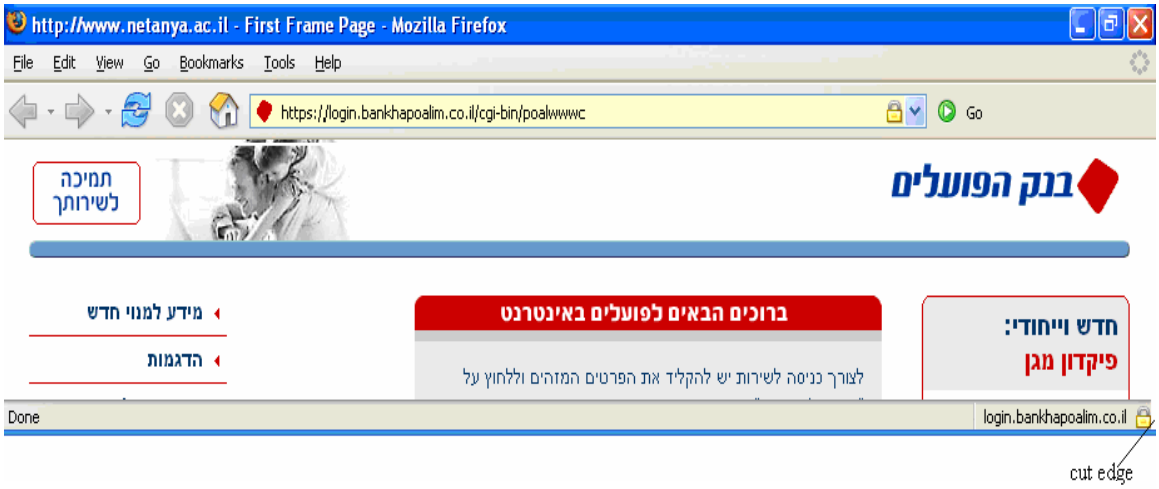

Figure 14: Fake Bank – security indicators replaced with images. The real size of this figure captures full screen so the left/right borders do not appear. Notice the relaxations we made; cut edge and thick border between the menu bar and the address bar. This is to reflect the fact that attackers often make imperfect forgeries, which may be detected by such `clues'.

Figure-15 shows a comparison between the participants' achievements, with each of the three identification and security indicators. The results show similar trends to the unlimited-time and `screen shots' experiments, namely: unacceptably high false negative and positive rates for the classical indicators, and much improved detection rates for the `improved` indicators – especially for the user-customized indicators. We also note significant improvements in throughput (and even more in goodput) from the `classical` indicators to the

certificate-derived indicators (and even further for the user-customized indicators). We conclude that users make more correct choices, and more quickly, using the improved indicators.

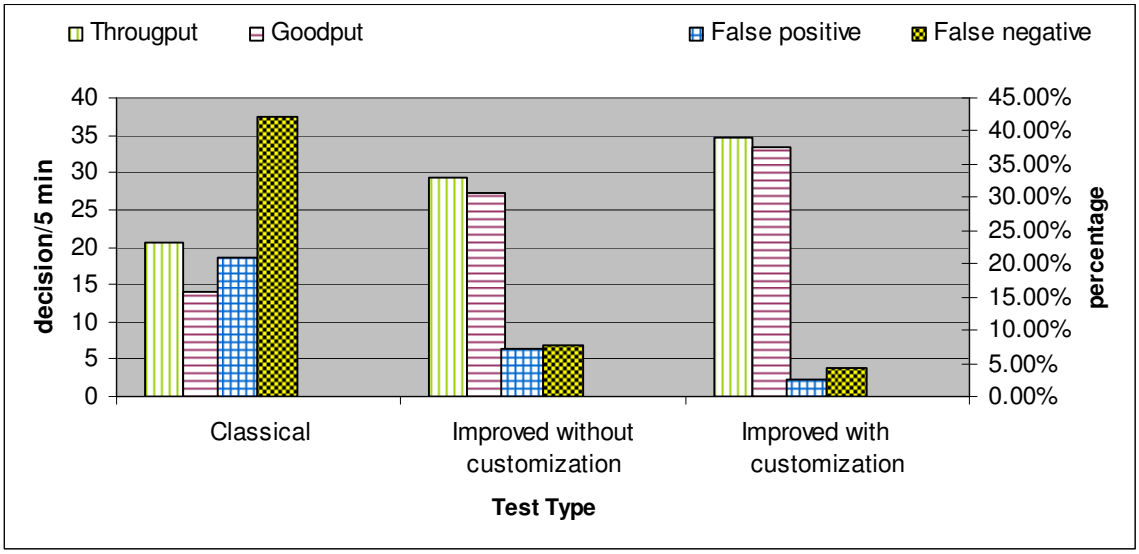

Figure 15: Results of the Limited-Time Experiment. Notice the significant difference between the classical site identification and the improved site identification approaches.

We also conducted a small additional experiment, to check whether our experiment design could have made an artificial impact on the results, due to a learning effect between the three tests (due to the fixed order of the three tests - which is a mistake we regret). In this additional experiment we had only 6 participants. Each of them made the first test three consecutive times.

The results, in Figure-16, show that repeating the first test (with `classical' indicators) three times had a modest effect of increasing throughput, however also with some increase in false positives. Therefore, the effect of `learning' is limited and mixed, and does not dramatically impact our findings.

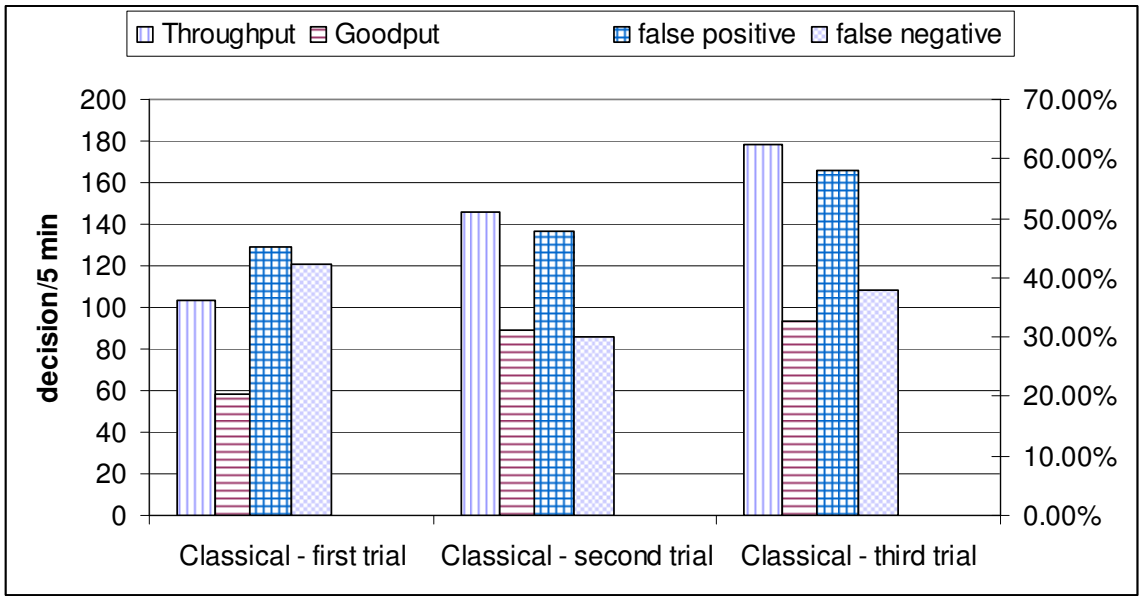

Figure 16: Repeated Classical Experiment Results.

Additional Statistics

We found no significant differences between different categories of participants, such as by gender, seniority, security background etc.

We also the detection rates for the different types of attack, as shown in Figure 17. For both improved indicators, we did not find any significant difference between the detection rates for different attacks. However, we found that detection raters using `classical` indicators are substantially higher for the attack using unprotected site with incorrect URL, compared to the attacks using an incorrect URL but with SSL protection, and the attack using a fake location bar. It appears that attacker can significantly improve their odds against users of current browsers by simply using SSL, but this is not effective against users of the improved identification indicators.

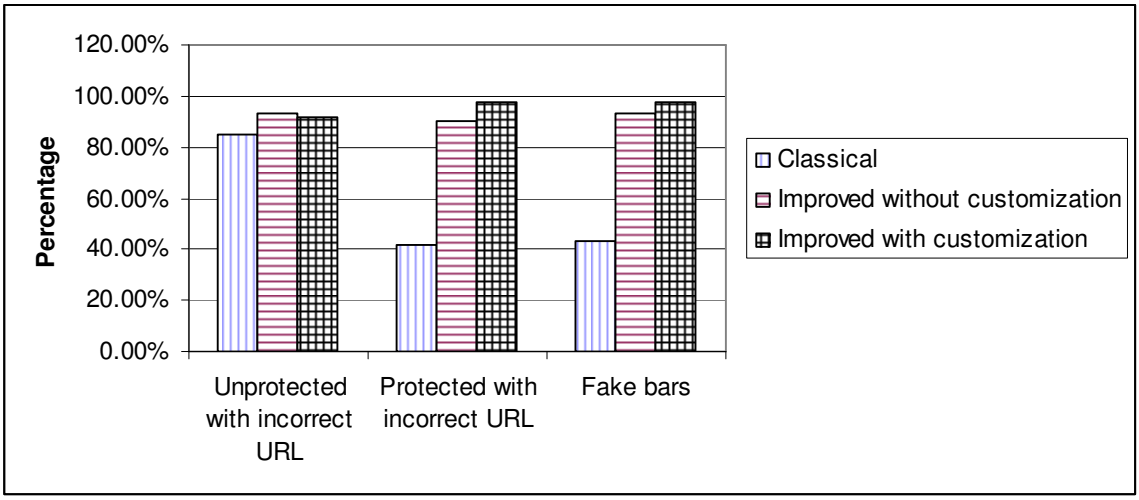

Figure 17: comparison between attack types in each identification approach.

6 Conclusions, Research Directions and Recommendations

As already shown in [FB*97], 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 designed, implemented and tested, that will help detect web-spoofing (and phishing) attacks. The main idea is to enhance browsers with a mandatory, improved security and identification indicator. Preferably, the indicator would be user-customizable, using information from the certificate by default.

Our experiments confirmed that improved indicators can significantly improve the detection rates and throughput of users. Notice that while our experiments show a conclusive advantage and impact of the improved indicators, they are not sufficient to actually estimate the expected real-life detection rates. Such measurement requires a much more extensive and longer-term experiments. In particular we need to check whether the detection rates reduce over long periods of time. Such experiments w2ill also need to use more sites, involve realistic scenarios and tasks, and involve a more broad and typical collection of attacks, and a reasonable ratio between real and fake sites.

Another open issue, which we did not test sufficiently, is the impact of using graphical indicators (e.g. logos) vs. textual indicators. In our experiments it was hard to compare between the two, because users were limited in time and motivated to achieve more clicks, so they prefered to minimally customize the system.

We believe that our results and experience justify the addition of improved indicators to new browsers; indeed, early releases of version 7 of the Internet Explorer (IE) browser, include a certificate-derived indicator

which is very similar to what we proposed and implemented in TrustBar. The IEv7 indicators identifies the CA, and restricts the `certificate-derived indicator' to sites whose identity is validated by a more trustworthy process, by certificate authorities trusted by the vendor. As we mention within, when validation is by a trustworthy CA, as determined by the vendor or by a security service chosen by the user, it seems reasonable to identify only the site (avoiding the `identified by` <CA> indicator, for better usability).

We presented several secure UI principles, based mostly on previous works, our own experience, experiments and common sense. Future research should establish a comprehensive set of secure UI principles and evaluation criteria, based on extensive, rigorous experimentation and validation.

To conclude this paper, we present conclusions and recommendations for users and owners of sensitive web sites, such as e-commerce sites; 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 browsers with improved security and identification indicators (possibly implemented by an extension such as TrustBar), and in particular customize the indicators (if possible) for your important sites.
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.
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.

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.
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 or similar client-identification mechanisms (e.g. IP address), 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.

We therefore recommend, following [PPSW97, H03], to restrict the execution of very 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, 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. We are also grateful for constructive suggestions from the anonymous referees.

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

Thanks to the organizers of ENC04 for inviting the first author to deliver a keynote lecture, and to the audience who filled in the review (see Section 5); also thanks to the other participants in our experiments, mostly from Natanya college and Bar Ilan university.

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.

[APWG] Anti-Phishing Working Group, Phishing Archive, at http://www.antiphishing.org/phishing_archive.html

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

[APWG06] Anti-Phishing Working Group, Phishing Activity Trends Report - May 2006, available online at http://www.antiphishing.org/reports/apwg_report_May2006.pdf.

[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.

[C06] Tyler Close, Petname Tool: Enabling web site recognition using the existing SSL infrastructure, presented in W3C Workshop on Transparency and Usability of Web Authentication, March 2006, New York City. Available from http://www.w3.org/2005/Security/usability-ws/papers/02-hp-petname/.

[CL*04] Neil Chou, Robert Ledesma, Yuka Terguchi and John C. Mitchell, Client-Side defense against web-based identity theft, NDSS, Feb. 2004.

[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.

[DT05] Rachna Dhamija and J. Doug Tygar. The battle against phishing: Dynamic security skins. In Proc. ACM Symposium on Usable Security and Privacy (SOUPS 2005), pages 77–88, 2005.

[DTH06] Rachna Dhamija, J.Doug Tygar, and Marti Hearst. Why Phishing Works. Proceedings of the Conference on Human Factors in Computing Systems (CHI2006), pp. 581-590, Montreal, Quebec, Canada, April 2006.

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

[E04] Aaron Emigh, Anti-Phishing Technology, Radix Partners, technical report, September 2004.

[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, in Fourth Conference on Security in Communication Networks '04, Amalfi, Sept. 2004; to be published in Springer-Verlag's LNCS series.

[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.

[HG04] Amir Herzberg, Ahmad Gbara, *TrustBar: Protecting (even Naive) Web Users from Spoofing and Phishing Attacks*. 2004: Cryptology ePrint Archive: Report 2004/155

[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.

[T04] Visual Validation of SSL Certificates in the Mozilla Browser using Hash Images, Hongxian E. Tay, CS Senior Honor Thesis, School of Computer Science, Carnegie Mellon University, Advisor: Adrian Perrig, May 2004.

[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.

[WMG06] Min Wu, Robert C. Miller and Simson L. Garfinkel, Do security toolbars actually prevent phishing attacks?, Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 601-610, Montréal, Québec, Canada, 2006.

[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.