Protecting Web Users from Malicious Content and Sites

Amir Herzberg
Dept. of Computer Science
Bar Ilan University

The web and its users are suffering from a growing amount of malicious, criminal abuses, of different forms. Some of the worst abuses are related to malicious content and web sites: spoofed web sites stealing passwords and other sensitive information, phishing email directing users to such spoofed sites, malicious scripts and other malware delivered via (often spoofed) web sites, and more.

We are aware of four different, possibly complementing approaches to improve the security of web users, against attacks by malicious, often misleading (fake, spoofed) content received from the web. These approaches are:

1. Improved security and authentication indicators.
2. Black lists identifying content suspect as malicious.
3. Trust management and accountability, based on digital signatures.
4. Improved user authentication mechanisms, to reduce risks due to password theft and Man-In-The-Middle attacks.

In this paper, we discuss the first three approaches in the given order. These three approaches deal with server and/or content authentication. The fourth approach deals with user authentication, which is an important and related yet separate issue; it is not covered in this paper.

Our discussion is based on our experience and conclusions from developing TrustBar, an improved security indicator extension to the FireFox browser, including feedback received from users, surveys and empirical data collected. We are extending TrustBar, to implement some of the new ideas described in this paper. These new ideas are mainly based on trust management and accountability based on digital signatures. Readers who are familiar with the area and TrustBar, and want to focus on our new ideas, may jump to section 3.

1. Improved security and authentication indicators.

Browser developers put a (justified) focus on making browsers as user-friendly as possible, and in particular make every effort to include only simple and crucial indicators and controls in the main menus. In particular, traditionally browsers offered only two indicators related to authentication and security in the default bars:

1. The location (address) bar, which displays the URL. The URL indicates the domain name, which provides an identification the organization owning the site. The URL also indicates the protocol used; in particular, when using an SSL or TLS protected site, the protocol is https.
2. The padlock or other symbol, usually in the status line, indicating the use of TLS/SSL.

These indicators are insufficient. Users do not understand domain names, and do not notice the lack of the padlock in the status bar. One reason for this is that sites confuse users, by presenting login forms which are not protected in an SSL/TLS connection. Such pages often include an image of a padlock as part of the page itself, and/or use an incorrect domain name. Such failures happen on some of the most sensitive, widely used login forms, e.g. of PayPal, Chase, Microsoft passport, and many more (see the I-NFL Hall of Shame). For example, Bank of America's homepage is unprotected (and contains an image of padlock to imply security), and their real-estate login is on a

separate domain reo.com. See figure below.

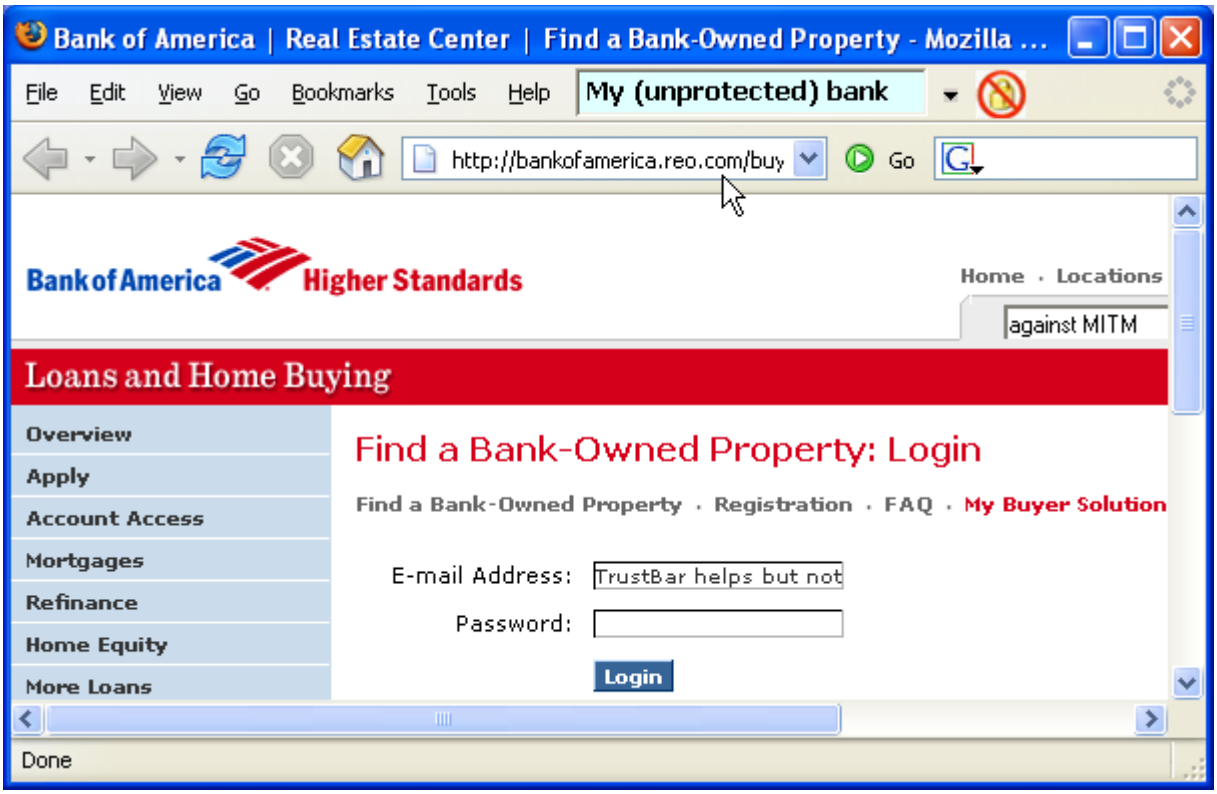

Figure 1: Unprotected BoA site, with wrong domain (reo.com).

We investigated alternative security and authentication indicators, and implemented several of these ideas in TrustBar. The main improved indicators are (see also figure below):

1. For SSL/TLS protected sites, TrustBar presents, by default, the name of the organization owning the site, as identified in the certificate. Trust bar also presents `Identified by:` and the logo (or name) of the certificate authority who have done this identification.
2. For unprotected sites, TrustBar presents, by default, the domain name. This makes it a bit easier for users to notice a wrong domain.
3. In both cases, users can assign their own name or logo, and TrustBar will then present this name/logo whenever presenting the site (see in the figure).
4. In addition, TrustBar presents a padlock for protected sites, and a warning icon (`do not enter` over padlock) for unprotected sites.

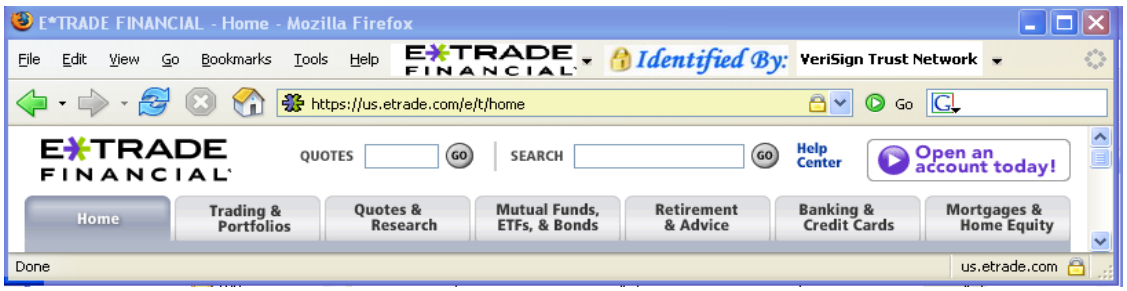

Figure 2: An SSL protected page with TrustBar

Displaying organization name from the certificate, and `Identified by: <CA>`, was recently adopted by developers of version 7 of Internet Explorer, and may become a standard mechanism in browsers. Similar mechanisms were adopted by Opera and Netscape.

Figure 1 also shows some improved security indicators adopted by the recent versions of FireFox: a padlock displayed in the location bar, domain name displayed next to the padlock in message area,

and a yellow background used for SSL/TLS protected sites.

Our experience and tests so far show that such improved indicators result in a significant improvement in the ability of users to detect spoofed sites. However, it is far from being a sufficient solution. We identified three main areas which require further improvement:

1. TrustBar offers very limited protection to users of unprotected login pages. Indeed, many of the current spoofing attacks direct the user to a spoofed page which is in a separate domain from the cloned (`victim') page, and TrustBar can help detect these attacks (esp. if the user assigned a name/logo to the page). However, TrustBar does not help against `Man In The Middle` attacks against unprotected pages, including a significant number of sensitive login pages. By extension, TrustBar does not protect the user against other malicious content, such as malware (worms, etc.) loaded by a (malicious) web page.
2. Users do not remember to validate the indicators. This seems to be a result of human psychology; while we still believe, based on different evidences, that users are concerned about security, many users fail to validate an indicator (which is normally OK). We tried to improve user awareness and understanding, by randomly changing TrustBar's indicators on a secure page, and expecting the user to click a warning button (`Hey!`). However, this technique, or at least our implementation of it, was too aggressive and annoyed users – resulting in a complete failure.
3. Users do not understand sufficiently the distinction between the web page displayed by the browser, which can present logos without authorization, and the name and/or logo displayed by TrustBar.

We can also identify some relevant secure usability guidelines; here are some:

1. **Defend, don't ask.** Users expect the security mechanisms to defend them, without interfering with their day to day operation and interrupting their work. In particular, users tend to ignore security indicators, and to click `yes` without really reading security dialogs.
2. **Prevent, don't teach.** The distinction between the web page and the browser's UI may be trivial – to us; it is not trivial or natural to users. We need to prevent abuse by misleading information in the web page itself, and not try to educate users on the difference. Similarly, we cannot depend on users objecting to dangerous requests from the web page, e.g. software install, by hitting `no` or `cancel` in the dialog opened by browsers. Users `just say yes`; and trying to educate them to `just say no` is futile.
3. **Cryptography is in Greek.** If security dialogs and indicators are to provide any help to the average user, then they must refrain from using obscure technical terms. Browsers and web sites explained certificates, CA and PKI concept to users – but they had no clue. Once we changed to the obvious `Identified by`, users at least understand.

2. Black lists identifying content suspect as malicious.

One natural approach to protect users from spoofed web sites, and potentially other malicious sites, is to maintain a `blacklist` of such sites. When the browser reaches such a page, it can warn the user and/or block access to the page.

Several extensions and few browsers now support such blacklists. Blacklists rely on databases of spoofed (or more generally malicious) web sites, maintained by dedicated organizations, and/or by feedback from users of security browser extensions. They can be very effective in blocking known malicious sites, and if they are sufficiently accurate, browsers could completely block access to the suspect sites – implementing the `defend, don't ask` principle.

Blacklists are already applied, for years, for other applications. Most notably, to block spam, many incoming mail servers rely on one or multiple blacklist of mail servers and domains which may be

sources of spam; servers usually query these lists using the DNS protocol.

However, blacklists have significant limitations and drawbacks; many of these problems are well known from the use of blacklists for spam prevention, and some are even more severe for web sites. As a result, we expect blacklists to be a very useful, attractive solution in the short term, but to have limited long term value and require complementing measures; this is also the common opinion by most experts woring on the spam problem. Some of the problems are:

1. Blacklists are reactive; they only list identified servers. Setting up a new site is very easy and inexpensive, and can be automated.
2. Current anti-spoofing blacklists seem to always operate based on the server's domain name. However, buying a new domain is really trivial for attackers, and almost cost free. Furthermore, listing domain names still allows for attacks by DNS poisoning, which is often still possible. This will require listing IP addresses and then IP address blocks – as done by anti-spam blacklists. However, this is likely to result – as for spam – in listing of larger and larger address blocks, cutting off innocent users as well as attackers.
3. The control and management of such blacklists require very large, manual, operations and expense, and on the other hand gives the operator control over availability. This may have undesirable social consequences.
4. Attackers often are able to exploit only a part of the services of a site, e.g. by buying site hosting services (but not controlling the entire web server). Blocking the entire server may not be a viable solution. Consider, for example, an attacker using a hosting service such as Akamai.

We conclude that blacklists of malicious domains can be an important defensive mechanism, but need to be complemented.

3. Trust management and accountability, based on digital signatures.

Finally, let us present a new direction we currently work on, for the next release of TrustBar. This idea seems very powerful, and may allow substantial improvement in security. It is easier to begin with a specific, simple problem: protecting users of unprotected login pages, such as Bank of America's sites, e.g. the one in Figure 1.

Let us first analyze the situation; is BoA really completely reckless and does not use SSL/TLS to protect the password? The situation is probably different. Most of the unprotected login pages do invoke SSL or TLS – but only once the user typed in the password and clicked `login`. As that point, a script on the page opens an SSL or TLS protected connection to the BoA's server, and sends an HTTP request over that encrypted connection. Most unprotected login sites mainly try to save the overhead of setting up the SSL/TLS connection, for the case when the user merely opens the page containing the login dialog, but does not actually login.

Of course, one way to protect the users of such unprotected login sites is to switch to a different site, or to change the site. Since we do not run such sites, we tried first to convince the site owners to change, by sending them letters and listing them in the Inter-Net Fraud League (I-NFL) Hall of Shame.

We next tried to help users of such unprotected login sites, by automatically loading an alternate protected site. It turns out that many of the unprotected login sites, also have an SSL/TLS protected alternate page, which is often using the same URL, except that using the HTTPS protocol (HTTP over SSL) rather than using plain HTTP. By the way, this does not seen to work for the BoA sites.

Therefore, upon reaching an unprotected login page which has an equivalent protected page, the browser (or TrustBar) can simply load the alternative, protected page. This seems easy enough, and

we implemented it in TrustBar. Currently this mechanism is disabled, but this is merely due to some bugs – we may return this feature later on. But we probably will not, since we believe we have a better – if harder to develop – solution.

The other solution allows the use of the non-SSL/TLS login page – but in a secure, protected manner. One advantage of this is that it will work even for pages where no SSL/TLS substitute exists or was found, e.g. the BoA sites. Another advantage is that we avoid problems such as the disappearance of the protected substitute page.

The crux of the solution is the observation, that the unprotected login form used by most of these sites, does invoke SSL / TLS to the right server; it only does this too late, i.e. after the user typed her password (to a page which could have been a fake, cloned version, without the user noticing). Therefore, the problem is only due to the fact that the user would not be able to notice, if it receives a modified version of that login page (sending an unencrypted copy of the password to an attacker).

To solve this, we added a function to TrustBar that will save a hash of such unprotected login pages, and inform the user when such sites change, or list `Same since: <date>` if they did not change. See screen shot in Figure 3 below. Notice that hashing has to be done not just for the HTML, but also for included objects, most notably different scripts and active objects.

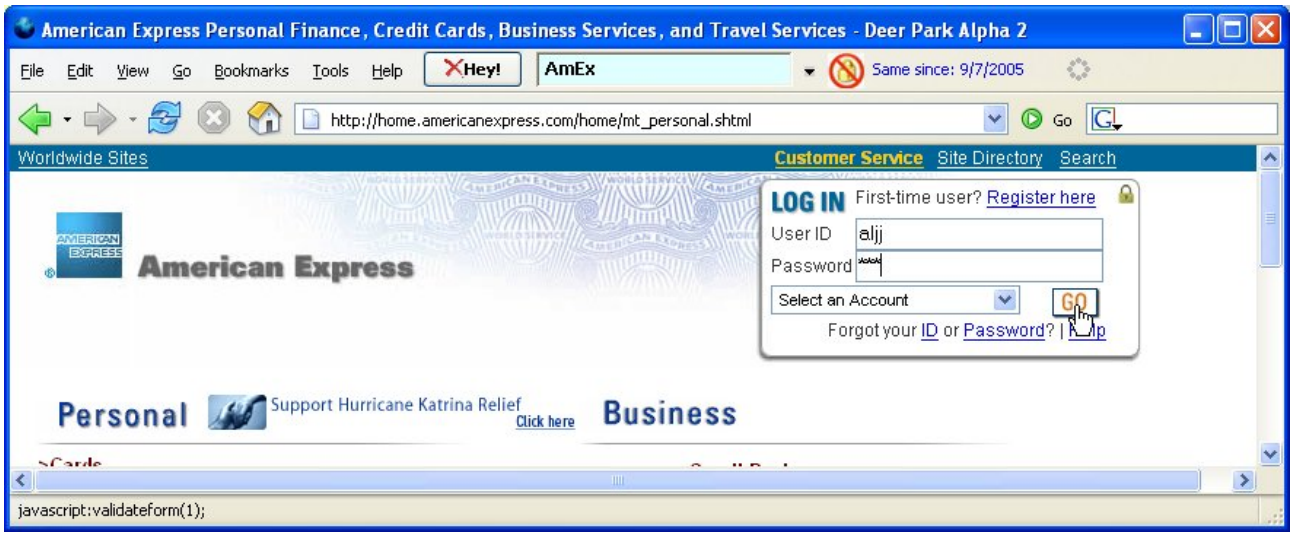

Figure 3: Unprotected Login site, but `Same since" <date>

However, this is limited – no protection for first time users, and no clear solution to what to do if/when the site does change. Now, if the site would cooperate, we could have the site digitally signing the new version. But what if the site is not willing to digitally sign its pages?

We now work on a solution, which will allow us to use signature of pages provided by third parties. This will include an efficient distribution mechanism for such signatures, possibly using DNS. We hope to be able to review and sign a fair number of important web pages and their elements (e.g. scripts). Our design also include mechanisms to allow signatures by multiple parties, with trust management mechanisms allowing recipients to decide which signatures to trust, and accountability mechanisms to punish or penalize `cheaters`.

With this mechanism, we can go far more than just identifying that a page did not change. In particular, the signed objects may include signed, accountable statements for such aspects as:

- This script (or: scripts in this page) does not contain malware.
- This page does not unauthorized logos (if it has logos – these are authorized – no spoofing!)
- This page or image does not contain pornographic materials inappropriate for minors.
- This page / image / object is not copyright protected by anyone