Pharewell to Phishing : Secure Direction and Redirection over the Web

Taehwan Choi Sooel Son Mohamed G. Gouda
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
{ctlight, samuel, gouda}@cs.utexas.edu

April 15, 2008

Abstract

The web pages over the World Wide Web are classified into two classes: insecure http pages and secure https pages. A user can tell whether a displayed page on his browser is insecure or secure by observing the background color of the URL box of the displayed page. If this background color is white (or yellow), then the displayed page is insecure (or secure, respectively). Henceforth, we refer to insecure web pages as white pages, and refer to secure web pages as yellow pages.

The conventional wisdom has always been that users should refrain from entering their sensitive data (such as passwords, credit card numbers, and social security numbers) into white pages, but they can enter these data into yellow pages. This conventional wisdom is based on the assumption that when the user requests a specific yellow page, the browser is *directed* to the right page and the browser does authenticate the called page before displaying it to the user.

Unfortunately, this assumption is not valid as it became clear recently that, through human mistakes or Pharming or Phishing attacks, a displayed yellow page may not be the same one that the user has requested in the first place. The net effect of this situation is that yellow pages are not as secure as they were once believed to be. Thus, users need to be careful when they enter their sensitive data into yellow pages.

In this paper, we propose to add a third class of secure web pages called brown pages. We show that brown pages are more secure than yellow pages especially in face of human mistakes and Pharming and Phishing attacks. Thus users can enter their sensitive data into brown pages without worry. We present a login protocol, called the Transport Login Protocol or TLP for short. An https web page that is displayed on the browser is classified brown by the browser if and only if this web page has been called into the browser either through TLP or from within another brown page that had been called earlier into the browser through TLP.

1 Introduction

When a user needs to display a web page on his browser, the user follows any one of four direction rules, described below, to request that his browser calls the page and displays it on the screen. If the requested page is an (insecure) http page, then the browser calls the page and displays it without any firm guarantee that the displayed page is the one that the user has requested. On the other hand, if the requested page is a (secure) https page, then the browser displays the page only after it has authenticated that the page is the one that the user has requested. Unfortunately, as described below, the authentication procedure is vulnerable to human mistakes, by the user, and to Pharming and Phishing attacks [37], by adversarial web sites. And so it is possible that the displayed page may not be the one requested by the user after all.

The user may not mind that the displayed page is different from the page that he has requested for two reasons. First, both the displayed page and the page that the user has requested have similar graphics and colors and the user may not notice that the displayed page is actually not the one that he has requested even in the presence of security indicators [13]. Second, the user may notice that the displayed page is not the one that he has requested, but he may believe that the displayed page is a legitimate redirection that was requested by the page that he has requested. In any case, the user may proceed to enter some sensitive data, such as his credit card number, into the displayed page which may happen to be an adversarial page.

This paper is dedicated to prevent these scenarios from occurring. Towards this end, we propose to introduce a new class of https web pages, which we refer to as brown pages. As discussed below, brown pages are secure against human mistakes and Pharming and Phishing attacks. Thus, when a user requests that his browser calls and displays an https page and then the browser displays the page and classifies it brown, the user knows that the displayed page is indeed the one that he has requested and so he can proceed to enter his sensitive data into it.

In order for the browser to be able to classify a called https page brown, the browser needs to call this page through a login protocol that is completely secure against human mistakes and Pharming and Phishing attacks. In this paper, we present and discuss the design and implementation of such a login protocol.

2 Direction Rules over the Web

Web browsers support four direction rules that can be followed by any user in order to call an https web page into the user's browser. In this section, we describe these four rules and show that they are prone to human mistakes and Pharming and Phishing attacks. In the next section, we describe how to augment these rules and make them more secure against these mistakes and attacks.

2.1 The Immediate Direction Rule:

In order for a user to call into his browser an https web page, the user can enter the URL of the page into the URL box of the browser.

The problem of immediate direction is that it is vulnerable to human mistakes since humans are not good at distinguishing different homographs [20]. For example, assume that a user wants to call the web page whose URL is `"https://www.mybank.com"`, but he enters by mistake the URL of a different page `"https://www.nybank.com"` into the URL box of his browser. In this case, if the graphics and colors of these two different web pages are similar, then the user may not notice that he has made a mistake and proceed to enter his password of one secure page into a different page.

2.2 The Redirection Rule:

For the convenience of its users, a web site may allow users to call its https web page as if the page is an http page. For example, the web site `"www.mybank.com"` may allow its users to call its https page using the URL `"http://www.mybank.com"`. In this case, when a user enters the URL `"http://www.mybank.com"` into the URL box of his browser, the web site `"www.mybank.com"` redirects the browser to the secure https page `"https://www.mybank.com"`.

The problem of redirection is that it is vulnerable to Pharming attacks, where DNS is manipulated so that the request for a web page is directed to a wrong web site [5]. In the above example, assume that the user enters the URL `"http://www.mybank.com"` into the URL box of his browser and, due to a manipulation of DNS, the request for this web page is directed to an adversarial web site that redirects the browser to the wrong secure web page `"https://www.nybank.com"`. In this case, as mentioned earlier, if the graphics and colors of these two different web pages are similar, then the user may not notice that his browser is displaying a wrong web page and proceed to enter his password of one secure page into a different page.

2.3 The Insecure Indirection Rule:

A user can call an http web page into his browser and find in this web page a link that is described as leading to the secure https web page `"https://www.mybank.com"`. By clicking on this link, the user calls into his browser the web page `"https://www.mybank.com"`.

The problem of insecure indirection is that it is vulnerable to Phishing attacks, where a user is lured to call into his browser an adversarial http page. This web page has a link that is described wrongly as leading to the secure https page `"https://www.mybank.com"`. In fact, this link leads to another web page `"https://www.nybank.com"`. Thus, by clicking on this link, the user thinks that he is calling `"https://www.mybank.com"`, but ends up with the page `"https://www.nybank.com"` displayed on his browser. The user does not notice the switch because the two web pages have similar graphics and colors (and similar URLs). The net effect is that the user enters his password for the web page `"https://www.mybank.com"` into the wrong page `"https://www.nybank.com"`.

2.4 The Secure Indirection Rule:

A user can call an https web page into his browser and find in this web page a link that is described as leading to the secure web site `"https://www.mybank.com"`. By clicking on this link, the user calls into his browser the web page `"https://www.mybank.com"`.

Secure indirection may or may not be secure depending on whether the first displayed page is adversarial. On one hand, if the first displayed page is not adversarial, then the user can click on the link that is described in this page as leading to `"https://www.mybank.com"` and indeed the next displayed web page will be `"https://www.mybank.com"`. On the other hand, if the first displayed page has been called into the browser using any one of the above three rules, then this page can in fact be adversarial and any link in it can lead to a web page different from the one being described. For example, the user can click on a link that is described as leading to the web page `"https://www.mybank.com"`, but then a different page `"https://www.nybank.com"` is called into his browser.

3 Securing the Direction Rules

In the previous section, we presented four direction rules that a user can follow in order to call an https page into his browser. We also illustrated, by way of examples, that these four rules are not as secure as they may seem at first. In each example, the user needs to call the web page `"https://www.mybank.com"` into his browser, but he ends up with a different displayed page `"http://www.nybank.com"`. In this section, we outline our proposal to modify the browser and some web sites in order that the above direction rules can become more secure in the face of human mistakes and Pharming and Phishing attacks. Our proposal consists of three parts.

1. **White, Yellow, and Brown Pages:**
   We propose to modify the browser so that the browser classifies each displayed http web page as white, and classifies each displayed https web page as either yellow or brown. As described below, a user should regard each white page as insecure, each yellow page as mildly secure (which means that the page is vulnerable to human mistakes and Pharming and Phishing attacks), and each brown page as highly secure (which means that the page is secure against human mistakes and Pharming and Phishing attacks).

2. **Classification of https Pages:**
   The modified browser assigns a classification, yellow or brown, to each displayed https page, depending on how this page has been called into the browser in the first place. Thus the same displayed https page can be assigned a yellow classification if it is called into the browser one way, and assigned a brown classification if it is called into the browser another way. In particular if an https page is called into the browser using any one of the first three direction rules, Rules 2.1 – 2.3 above, then this page

is assigned (by the browser) a yellow classification. Also if an https page is called into the browser by clicking on a link within a displayed yellow page, according to Rule 2.4 above, then this page is assigned a yellow classification. But if an https page is called into the browser by clicking on a link within a displayed brown page, according to Rule 2.4 above, then the called page is assigned (by the browser) a brown classification.

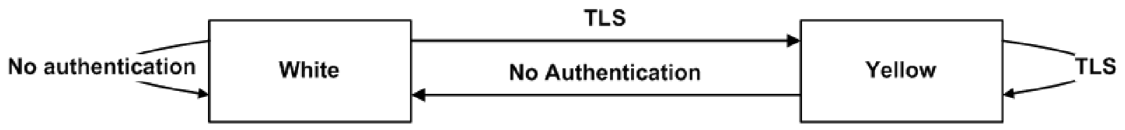

Figure 1: Classifications of the displayed page on current browser

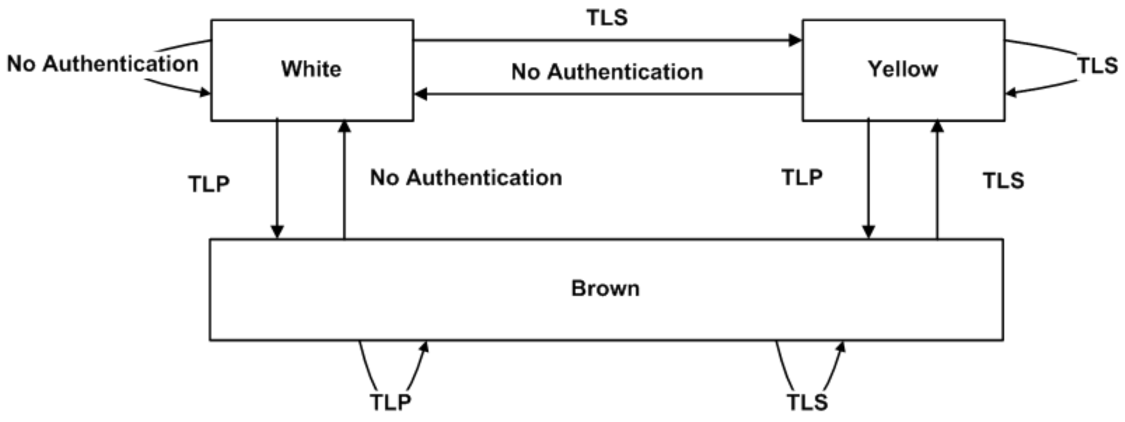

Figure 2: Classifications of the displayed page on modified browser

3. **A New Login Protocol:**
   We also propose to add a new login protocol to the browser and to some web sites that need to be (extra) secure against human mistakes and Pharming and Phishing attacks. We call this new login protocol the Transport Login Protocol or TLP for short. When a user invokes TLP on his browser and requests the browser to call a web page on a specified web site, the following three steps are executed. First, the browser and the specified web site use TLP to establish mutual authentication between each other. Second, if the mutual authentication between the browser and the web site succeeds, then the web site redirects the browser to an https web page. Third, the browser calls the secure web page and, upon receiving it, the browser assigns it a brown classification and displays it to the user. Moreover any https page, that is called into the browser using the protocol TLS from within a displayed brown page, is also assigned (by the browser) a brown classification.

When the browser displays a web page, the browser makes its classification of the displayed page clear to the user by choosing an appropriate background color for the URL box. If the displayed page is white (or yellow or brown respectively), then the background color for the URL box is white (or yellow or brown respectively). Note that the current browser already supports white and yellow classifications of web pages. So the main contributions of this project are merely the addition of brown classifications and the introduction of the new login protocol TLP which can be used in calling brown web pages into the browser.

(Recently, a green classification of https web pages has been introduced to distinguish those https pages that have extended validation certificates[18]. Clearly some green pages, like yellow pages, can still be adversarial, and can still be used in launching Pharming and Phishing attacks as described above. Henceforth, when we refer to yellow pages, we do mean yellow or green pages.)

Figure 1 shows the possible classifications, white or yellow, of the displayed page on the current browser (before our proposals are implemented), and the transitions between these possible classifications. For example the transition, labeled TLS, from the white classification to the yellow classification indicates that if the displayed page on the current browser is white then, using the protocol TLS, the next displayed page on the browser is yellow. Figure 2 shows the possible classifications, white, yellow, or brown, of the displayed page on the modified browser (after our proposals have been implemented) along with the transitions between these classifications.

The policy for entering sensitive data (such as usernames, passwords, credit card numbers, and social security numbers) into a displayed web page depends on the classification of the displayed page. This policy consists of the following three rules.

1. A user should never enter sensitive data into a white page.
2. A user can enter sensitive data into a brown page.
3. Before a user can enter sensitive data into a yellow page, the user should have prior knowledge that this data can be entered into this particular page, and the user should check that the URL box of the displayed page has indeed the URL of this particular page.

4 The Current Login Protocol

The current login protocol over the web consists of two protocols: the standard TLS protocol [9] (which is used to authenticate a secure web site by the browser), and a non-standard password protocol (which is sometimes used to authenticate the secure web site by the user and to authenticate the user by the secure web site). As described below, this current login protocol is vulnerable to human mistakes and Pharming and Phishing attacks. This vulnerability makes the current login protocol inappropriate for our purposes, of allowing the browser to classify as brown any https page that is called through the current login protocol, and compels us to seek another more secure login protocol.

The vulnerability of the current login protocol is due to the fact that this protocol is based on three techniques that are vulnerable to human mistakes and Phishing and Pharming attacks. These three techniques are as follows.

1. **Digital Certificates**:
   For a human user U to log in a web site S, the browser B of U first sends a Hello message to site S which replies by sending back a certificate. Browser B uses the received certificate to verify the identity of site S, but it cannot verify that S is the site that U wants to communicate with since the association between the certificate and S is not validated [16]. Therefore, an adversary can launch a Pharming or a Phishing attack, and end up directing the Hello message from B to a wrong web site S' which replies by sending back a certificate that browser B uses to verify the identity of site S' even though S' is not the site that U wants to communicate with. (To show that the problem of Phishing attacks is prevalent on the web, note that so far the web had as many as 28,015 sites that had launched Phishing attacks [22]. These attacks are estimated to have cost $2.8 Billions in the year 2006 alone [32].)

2. **Site Keys**:
   Once browser B of user U verifies the identity of a web site S, B allows S to display a custom image called the key of site S [44, 6]. This image had been agreed upon earlier by both the human user U and site S. Displaying this image is intended to convince U that it is indeed communicating with site S. But this may not be the case since it is possible that U is in fact communicating with an adversarial site S' that is launching a man-in-middle attack between U and S. In this attack, the adversarial site S' impersonates S as it communicates with U, and impersonates U as it communicates with S. Hence, S' can receive the site key from S and pass it on to U.

3. **Traditional Passwords**:
   Once user U examines the received site key of S and becomes convinced that it is indeed communicating with S, U replies by sending back his password to S. But if U happens to be in the middle of a man-in-middle attack, as described above, the password is actually sent to an adversarial site S'. If this happens, then S' can use the received password to impersonate U and log in site S. Even worse, if U had been using the same password to log in many other web sites, then the adversarial site S' can use the received password to impersonate U and log in all these other sites as well.

From this presentation, it follows that the three techniques of digital certificates, site keys, and traditional passwords are vulnerable to human mistakes and Pharming and Phishing attacks. Thus, our new login protocol TLP does not employ any of these techniques.

To be sure, we could have modified the login protocol SRP [52, 51] to make it secure against human mistakes and Pharming and Phishing attacks, and used the modified SRP in place of TLP. It turns out, however, that TLP has three features that make it more attractive for our purposes. These three features of TLP are as follows.

1. **TLP Universal Passwords**:
   Each user U needs only to memorize one password P, called the TLP universal password of U, that U uses to log in every web site S where U logs in using TLP. (It is important, though, that user U does not use his TLP universal password P to log in any web site S' where U logs in using a protocol other than TLP.)

2. **One-Time Login Data**:
   In each login protocol, a web site S needs to store some data related to the username and password for each user U that may need to log in site S. We refer to this data as the login data of user U in site S. In case of TLP, the login data for a user U in site S is updated after each successful login of U into S. Therefore, if an adversary somehow manages to steal the login data of some user U in site S, then the stolen data becomes useless in the next login of U into site S.

3. **Standard Cryptography**:
   Unlike SRP which uses exotic cryptography, TLP uses only standard symmetric cryptography and standard secure hash functions. Thus, every time the standards of symmetric cryptography or of hash functions are updated, the standards of TLP are updated accordingly.

5 The New Login Protocol

From the above discussion, we conclude that the current login protocol over the web is both complex and insecure and so we resolve to design the new login protocol TLP, that is simpler and more secure than the current protocol. Next, we describe our abstract design of TLP.

TLP is to be executed between browser B of user U and web site S. Prior to executing TLP, user U needs to have registered with site S by making its browser B store in S the following tuple of four data items:

$$(H(U), \quad n, \quad H(0,n,P,S), \quad H^2(1,n,P,S))$$

where

U	is the username of the user,
B	is the browser of user U,
n	is a nonce selected at random by browser B,
H	is a standard secure hash function,
0	is the character zero,
1	is the character one,
P	is the TLP universal password of user U, and
S	is the domain name of the web site.

Note that $H(0, n, P, S)$ denotes the application of the secure hash function H to the concatenation of the four data items 0, n, P, and S. Also, $H^2(1, n, P, S)$ denotes two consecutive applications of function H to the concatenation of the four data items 1, n, P, and S. After B stores this tuple in S, B forgets the tuple completely.

Executing TLP between browser B and site S is intended to achieve five objectives.

1. B checks that S is one of the sites where user U had previously registered (and stored the above tuple of four data items).

2. S checks that user U has entered his universal password P to browser B.

3. Both B and S agree on a symmetric session key that they can use to encrypt and decrypt their exchanged messages.

4. B selects a new random nonce n' and stores the following tuple of four data items in S in place of the above tuple:

   $$(H(U), \quad n', \quad H(0, n', P, S), \quad H^2(1, n', P, S))$$

   (Therefore, each successful login of browser B into site S causes the tuple of four data items that B had previously stored in S to be replaced by a new tuple of four data items also provided by B.)

5. S sends to B the URL of the next https page that B needs to call, using TLS, along with a cookie identifying user U and testifying that the login procedure between U's browser and S, has been successful. When the next https page is called into B, B assigns this page a brown classification. Moreover browser B assigns any other https page, that is called using TLS from within this brown page, a brown classification .

We adopt the following notation in describing a field in a message that is sent during the execution of TLP.

$$[expression1] < expression2 >$$

This notation means that the value of *expression1* is used as a symmetric key to encrypt the value of *expression2* before the message is sent.

To start executing TLP between B and S, user U enters three data items, namely U, P, and S, to a local web page named `httpl` stored in browser B. Then the execution of TLP proceeds with the following four message exchanges between B and S.

$$\begin{array}{ll}
B \rightarrow S: & \{\text{Hello Message}\} \\
 & U \\
 & \\
B \leftarrow S: & \{\text{Hello-Reply Message}\} \\
 & n, \quad [H(0, n, P, S)] < SN > \\
 & \\
B \rightarrow S: & \{\text{Login Message}\} \\
 & U, \\
 & [H(H^2(1, n, P, S), SN)] < H(1, n, P, S), \\
 & BN, H^2(1, n', P, S) >, \\
 & [H(BN, SN)] < n', H(0, n', P, S) > \\
 & \\
B \leftarrow S: & \{\text{Login-Reply Message}\} \\
 & [H(BN, SN)] < \text{URL of next https web page} >, \\
 & [H(BN, SN)] < \text{cookie} >
\end{array}$$

The hello message, from B to S, consists of the username of user U who wants to log in site S. On receiving this message, S fetches the tuple

$$(H(U), n, H(0, n, P, S), H^2(1, n, P, S))$$

that B had stored previously in S. Then S uses the data item $H(0, n, P, S)$ as a symmetric key to encrypt a new nonce SN that S selects at random. The result of the encryption is denoted $[H(0, n, P, S)] < SN >$ and is included in the hello-reply message that is sent from S to B.

After B receives the hello-reply message, it computes
$H(0, n, P, S)$ and uses it to obtain the nonce SN from the received message. Then, B selects at random two nonces BN and n', and uses the received SN and the computed BN and n' to construct the login message before sending it to site S.

After S receives the login message, it performs four tasks. First, it checks that user U has indeed entered its TLP universal password P into browser B. Second, S extracts the nonce BN from the received message, and now both B and S know BN and SN. Third, S stores the tuple: $(H(U), \quad n', \quad H(0, n', P, S), \quad H^2(1, n', P, S))$ in place of the earlier tuple. Fourth, S constructs the login-reply message and sends it to browser B.

After B receives the login-reply message, it concludes that S is one of the web sites where user U has previously registered. Moreover, B gets the URL of the https page that B needs to call next using TLS, along with a cookie that identifies user U and testifies to the fact that the login procedure between U's browser and S has been successful.

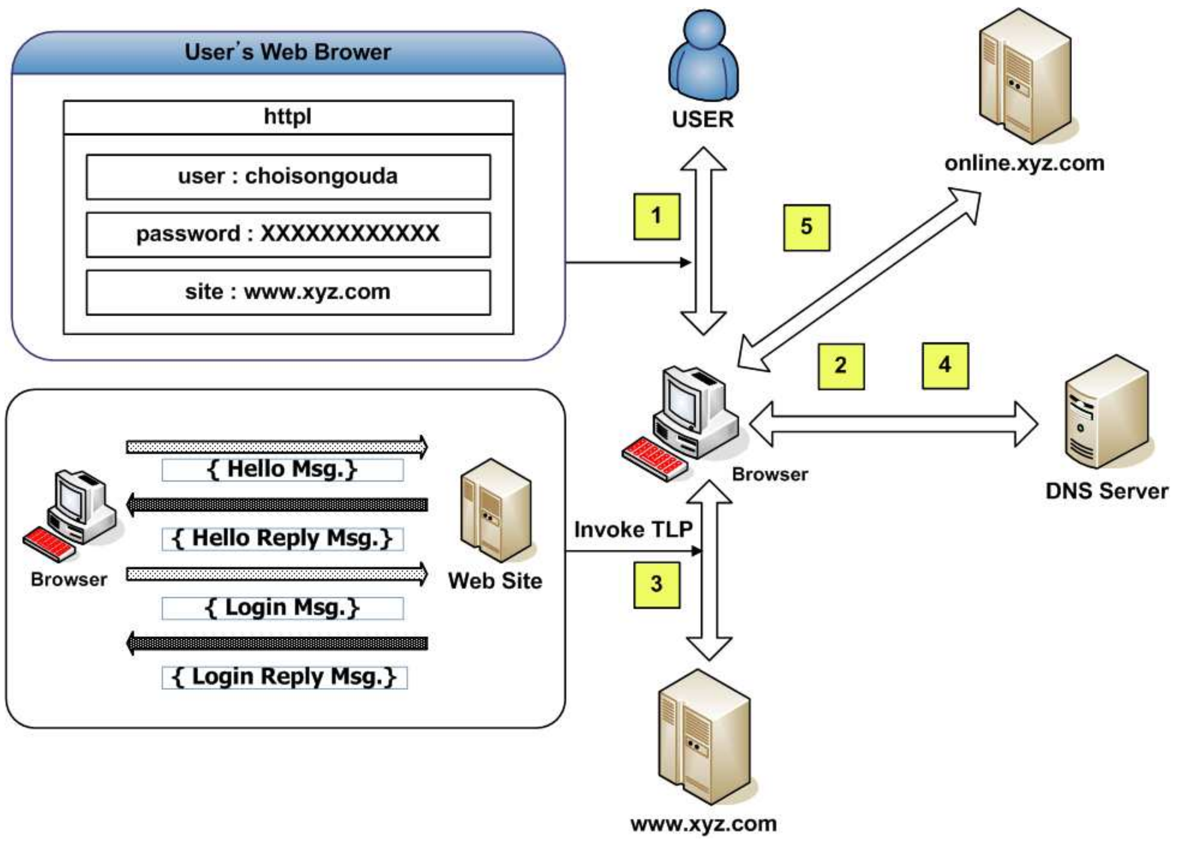

Figure 3: Using TLP

Figure 3 illustrates the five steps that are needed for a user to use TLP to log in a web site in a domain say `xyz.com`.

1. The user calls a local web page `httpl`, for convenience named the `httpl` page, on his browser and enters his username, his TLP universal password, and the site address `www.xyz.com` into this page.

2. The browser uses DNS to get the IP address of site `www.xyz.com`.

3. The browser and site `www.xyz.com` execute TLP. At the end, the browser receives the URL of a web page on site `online.xyz.com` and a cookie.

4. The browser uses DNS to get the IP address of site `online.xyz.com`.

5. The browser and site `online.xyz.com` execute TLS, and the browser gets the required https page at the end. The browser classifies this page, and any other https page that is called using TLS from within this page, brown.

6 Correctness of TLP

An adversary has no chance to succeed in attacking TLP between the browser of user U and web site S unless the adversary acquires a login tuple of U in S:

$$(H(U), n, H(0, n, P, S), H^2(1, n, P, S))$$

Moreover, an adversary cannot acquire such a login tuple unless the adversary succeeds in breaking into the secure database(s) where site S stores the login tuples of all its users – a hard task to perform!

Even if an adversary somehow succeeds in acquiring a login tuple of U in site S, this adversary cannot succeed in launching a user impersonation attack, a human mistake attack, or a Phishing attack against TLP between the browser of U and S.

1. **Security Against User Impersonation:**
   If an adversary acquires a login tuple of user U in site S, and if this adversary uses this tuple to impersonate a browser of user U and execute TLP in order to log in site S, then the adversary's attempt to log in site S will fail.

2. **Security Against Human Mistakes And Phishing Attacks:**
   If an adversary acquires a login tuple of user U in site S, and if this adversary uses this tuple to establish an adversarial web site S' whose domain is different from that of S, and if user U requests that his browser executes TLP and logs in site S', then the login attempt will fail.

Still, if it is possible for an adversary to acquire a login tuple of user U in site S, then this adversary can launch successful Pharming and eavesdropping attacks against TLP between the browser of U and S. To protect against this possibility, the login tuple of U in S is partitioned into two subtuples:

$$(H(U), n, H(0, n, P, S)) \text{ and } (H(U), H^2(1, n, P, S))$$

Each subtuple is stored in a different database. Thus, the first subtuple is stored in database 0 and the second subtuple is stored in database 1. When user U initially registers in site S, the browser of U generates the first login tuple and sends it to site S. Site S divides the received login tuple into two subtuples and stores the first subtuple in database 0 and stores the second subtuple in database 1.

Later when user U attempts to log in site S, site S forwards each of the messages that S receives from U's browser to the appropriate database so that this database can process the message and return a reply to S which forwards the reply back to U's browser. For example, when S receives the Hello(U) message from U's browser, S selects a nonce SN at random and sends both U and SN to database 0 which prepares a Hello-Reply(n, $[H(0, n, P, S)] < SN >$) message and returns it to S which forwards it back to U's browser. Thus no one, not even site S, gets to keep track of the subtuples stored in the two databases 0 and 1.

Because the two subtuples of user U in site S are continuously changing, it is reasonable then to assume that an adversary, who attempts to acquire the two subtuples of the same user U from the two databases 0 and 1, will do so at different times and will end up with two unsynchronized subtuples of the following form:

$$(H(U), n, H(0, n, P, S)) \text{ and } (H(U), H^2(1, n', P, S))$$

Fortunately, if an adversary who succeeds only in acquiring two unsynchronized subtuples of user U in site S, then this adversary cannot succeed in launching a Pharming or eavesdropping attack against TLP between the browser of U and S.

3. **Security Against Pharming Attacks:**
   If an adversary acquires two unsynchronized subtuples of user U in site S, and if this adversary uses these subtuples to establish an adversarial site S' whose domain is the same as that of site S, and if the DNS of U's browser is manipulated so that U's browser is directed to site S' instead of S, and if U requests that his browser executes TLP and logs in site S, then the login attempt will fail.

4. **Security Against Eavesdropping:**
   If an adversary acquires two unsynchronized subtuples of user U in site S, and if this adversary attempts to eavesdrop on the TLP communication between U's browser and S and obtain the cookie that is sent (encrypted) in the Login-Reply message of TLP, then the eavesdrop attempt will fail.

It is straightforward to prove that TLP satisfies the above four properties 1 through 4. To prove that TLP satisfies each of these properties, it is sufficient to identify a data item that the adversary will need, but will not be able to compute from its acquired data, in order to complete executing TLP successfully.

To prove that TLP satisfies Property 1, note that the adversary will not be able to compute the data item $H(1, n, P, S)$, even though it has acquired a login tuple of user U in site S, that is needed to compute the Login Message of TLP.

To prove that TLP satisfies Property 2, note that the adversary will not be able to compute the data item $H(0, n, P, S')$, even though it has acquired a login tuple of user U in site S, that is needed to compute the Hello-Reply Message of TLP.

To prove that TLP satisfies Property 3, note that the adversary will not be able to compute the data item $H^2(1, n, P, S)$, even though it has acquired two unsynchronized subtuples of user U in site S, that is needed to decrypt the Login Message of TLP and obtain BN.

To prove that TLP satisfies Property 4, note that the adversary will not be able to compute the data item BN, even though it has acquired two unsynchronized subtuples of user U in site S, that is needed to decrypt the cookie in the Login-Reply Message of TLP.

7 User Interface of TLP

As a proof of concept, we have developed a prototype of our Transport Login Protocol TLP. The browser side of our prototype is developed on the Firefox browser using the two technologies of Javascript and HTML. The web site side of our prototype is developed on the Tomcat web server using four technologies: Java, HTML, the JSP (Java Server Page) technology, and the MySQL database technology. Note that the MySQL database technology is used to manage the login tuples, of all users, that are stored in the web site.

We employed standard cryptography in our prototype. In particular, we employed the Secure Hash Algorithm SHA-1 for secure hash, and employed the Advanced Encryption Standard AES for symmetric key cryptography.

The guiding principle in our prototype is to ensure that the user never enters his TLP universal password into a web page that is supplied by a web site, but he can enter his password into a local web page that is supplied by his own browser. It turns out that this principle is hard to fulfill in our prototype in the light of the "Same Origin Policy" that is adopted by the Javascript technology. At the end, however, we were able to fulfill this principle by designing a novel user interface for our prototype. We discuss this user interface next.

Figure 4 details the four steps that need to be taken by a user to log in a web site `www.xyz.com`.

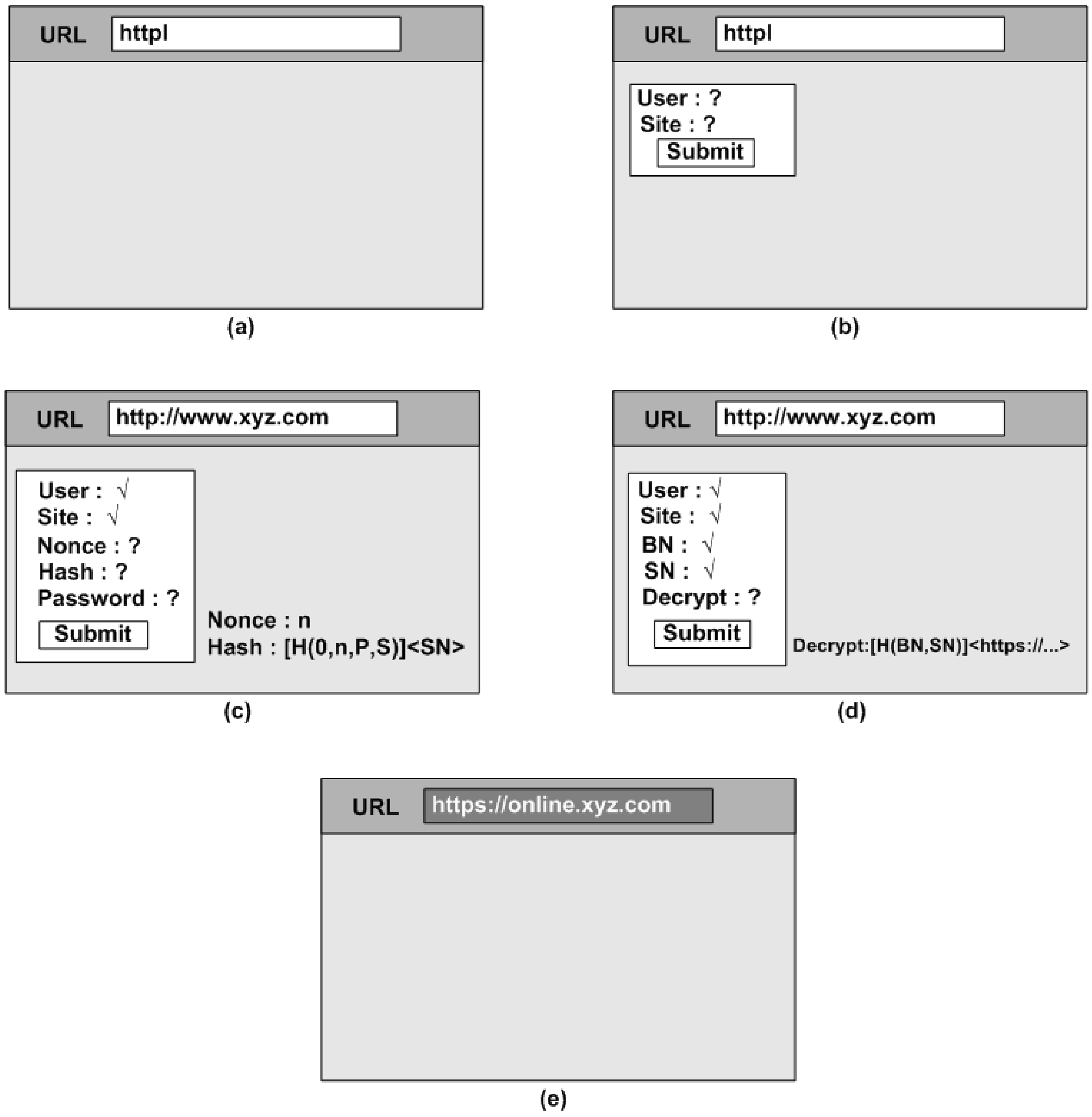

Figure 4: User Interface of TLP

1. The user first enters `httpl` into the URL box of his browser and pushes ¡return¿ ; see Figure 4a. This causes a display of the local page `httpl` to appear as a small window on the left site of the screen; see Figure 4b.

2. The user enters his username and the name of the site `www.xyz.com` into page `httpl` then clicks on the ¡submit¿ button in this page. This causes page `httpl` to execute, update its own display, and send a Hello message (the first message in TLP) to site `www.xyz.com` which replies by sending back the web page `http://www.xyz.com`. This page contains the two fields, named nonce and hash, of the Hello-Reply message (the second message in TLP); see Figure 4c.

3. The user copies the values of the two fields nonce and hash from the displayed page `http://www.xyz.com` and enters them into page `httpl`. The user then enters his password into page `httpl` and clicks on the ¡submit¿ button of this page. This causes page `httpl` to execute, update its own display, and send a Login message (the third message in TLP) to site `www.xyz.com` which replies by sending back a new web page `http://www.xyz.com`. This new page contains one field, named decrypt, of the Login-Reply message (the last message in TLP); see Figure 4d.

4. The user copies the value of field decrypt from the displayed page `http://www.xyz.com` and enters it into page `httpl`. The user then clicks on the ¡submit¿ button of page `httpl`. This causes page `httpl` to execute, compute the next https page, say page `https://online.xyz.com`, that needs to be called into the browser, and redirects the browser to call this page using TLS and display it on the screen. Note that in this case the browser assigns the displayed https page a brown classification, and so the background color of the URL box of the displayed page becomes brown as shown in Figure 4e.

Because the browser has classified the displayed page `https://online.xyz.com` brown, then if the user clicks on any link (of an https page) in page `https://online.xyz.com`, then the browser will classify the newly called page brown as well.

8 Related Work

Despite advances in security technologies, the Internet is plagued by vulnerabilities. These vulnerabilities exploit technical weaknesses as well as the propensity of humans to lowering their guards in routine transactions. In our daily web surfing experiences, homograph attacks [20] work very well for this reason. For the same reason, security indicators in browsers are not enough to prevent a user from accessing web pages hosted by malicious servers and alternative approaches are required [13]. While security mechanisms have been relatively successful in protecting systems from attacks that exploit technical loopholes or syntactic attacks [43], dealing with semantic attacks has proven much more difficult. As defined by Bruce Schneier [43], semantics attacks are attacks that depend on the way humans assign meaning to content. The difficulty of dealing with semantic attacks is due to the fact that most security systems depend on perfect human behavior that is always vigilant. In practice, security is not a user's primary concern and security checks and warnings can be considered by users as getting in the way of their use of applications. Thus, user errors result in security failures [48]. The total number of unique Phishing reports submitted to Anti-Phishing Working Group (APWG) in November 2007 was 28,074 [22]. Gartner says that Phishing attacks are estimated to have cost $2.8 Billions in the year 2006 alone [32].

8.1 Client-based anti-Phishing tools

In order to prevent Phishing in the client side, many anti-Phishing tools are developed. These anti-Phishing tools are based on blacklists, whitelists, and heuristics. Netcraft [35] mainly depends on blacklists and it can not detect a new Phishing site. Spoof Guard [11] is based on heuristics and it uses domain name,url, link, and image to evaluate the likelihood that a given page is part of a Phishing attack. SpoofGuard has high catch rate of 90% but also has high false positive rate of 42% [55]. CANTINA [56] is called a content-based approach since it is not only based on heuristics similar to Spoof Guard but also based on TF-IDF (Term Frequency and Inverse Document Frequency) information. It succeeds to reduce the false positives compared to Spoof Guard. Security toolbars such as SpoofStick [45], Netcraft Toolbar, Paypal Trustbar [23], eBay Account Guard [15], and SpoofGuard are designed for humans to use, but usability study found them all ineffective to prevent Phishing attacks [49].

8.2 Password-based approach

Password Hash [41] is a browser extension that allows users to log in multiple sites transparently with a universal password. The browser extension applies a cryptographic hash function to a combination of the plaintext password, domain name, and a salt. Thus, the break of one system does not lead to that of other sites. Usability study of Password Hash [10] involving 26 users show that password hash suffers from major usability problems. In addition to that, if password hashing is based on domain name, it is vulnerable to Pharming attacks [37, 46].

8.3 Server-based anti-Phishing

OpenDNS [38] is a way to filter Phishing sites at DNS levels. Thus, it is a blacklist based approach in DNS servers. Phishtank [39] is the online Phishing database which feeds this information to OpenDNS servers. PhishBouncer [4] uses HTTPS proxying and attribute-based checks to defend against Phishing attacks. PhishBouncer is based on whitelists, blacklists and heuristics. But it is still vulnerable to Pharming attacks and requires HTTPS proxy to be deployed in the Internet.

8.4 Cookie-based anti-Phishing

Though cookie-based approaches are effective in preventing Phishing attacks, they have limitations since cookies can be easily purged and cannot be a permanent solution for Phishing attacks. Temporary Internet Files (TIFs) are called cache cookies and are used as authenticators to protect from Phishing and Pharming attacks [27]. Active Cookies [28] rely on a new protocol that cookies are tagged with a specific, valid IP address and the channel redirected by a server and active cookies are fetched by the server to verify the client. But, it is vulnerable to IP based attacks. Adversaries can corrupt DNS [5] and BGP [36]. In order to resolve the vulnerabilities of the most vital function, DNS, in the Internet, DNS Security Extensions (DNSSEC) are proposed [1, 3, 2]. However, DNSSEC has obstacles to deploy in the Internet due to compatibility, scalability, and the ownership of root key. Locked Cookies [29] are cookies that are bound to the originating server's public key. Clients verify the server by comparing the public key in the locked cookie and the public key presented by the server. Web Server Key Enabled Cookie (WSKE) [31] bind cookies to domain names and public keys and similar to locked cookies.

8.5 Interface-based anti-Phishing

Interface-based anti-Phishing is the most popular approach to protect from Phishing attacks. PassMark [44] is a way to share a secret between a user and a web site and helps the user authenticate the web site. The user provides the web site with a shared secret such as image and/or passphrase in addition to his regular password. Some examples include site key [6] of Bank of America and Yahoo Sign-in Seal [53]. Site key consists of a unique image that users chose, an image title, and three challenge questions. By showing the unique image to users, users verify the server. However, site key is vulnerable to man-in-the-middle attack [54]. Moreover, the efficacy of security indicators such as HTTPS indicators and site-authentication images are ineffective such that users ignore HTTPS indicators and site-authentication images [42]. Yahoo Sign-in Seal is a secret message or image that users create to protect Yahoo accounts. It is similar to site key but it provides a way to create a customized text message with colors or images that users chose. Challenge questions are not completely free from security since common knowledge such as mother's maiden name can be collected in the Internet[21].

Dynamic Security Skins (DSS) [14] uses the trusted password window with a background image and SRP to authenticate a user and a server. Hash Visualization [12] is used to ensure that users are visiting authenticated web pages in DSS. Web Wallet [50] is a browser sidebar designed for users to submit their sensitive information. Users are required to press a dedicated security key on the keyboard to activate Web Wallet. However, Web Wallet itself can also be phished and users might not use Web Wallet to submit sensitive data.

8.6 Certificate-based authentication

Certificate seems as a panacea for security. However, what PKI can provide security is a popular falsehood [16]. PKI is more complex than what most people think. The problem arises when PKI is not well implemented and practiced. Certificate Authorities like VeriSign might issue erroneous certificates. VeriSign issued two VeriSign Class 3 code-signing digital certificates to an individual who fraudulently claimed to be a Microsoft employee [33]. Though VeriSign revoked those certificates, those certificates might be used if certificate validation is not implemented. The certification path validation algorithm is the algorithm which verifies that a given certificate path is valid under a given public key infrastructure [24]. This is complex assuming there are a number of intermediaries between the subject and the root. Delegated Path Validation (DPV) [40] and Server-based Certificate Validation Protocol (SCVP) [19] facilitates the validation of certificates.

OCSP (Online Certificate Status Protocol) [34] is an Internet protocol to obtain the revocation status of an X.509 digital certificate. Thus, clients must implement OCSP to check the status of certificates, but it was rarely deployed in the past.

To remedy Phishing problems, Extended Validation Certificate [18] was proposed. In order to highlight secure sites in IE7, it proposed to use different colors in URL address box. Unfortunately, this new technique is still vulnerable to picture-in-picture Phishing attacks and not promising to rectify Phishing problems [26].

8.7 Strong Password Protocols

TLS [9] provides privacy communication between two parties, but it does not ensure authentication. Strong password protocols provide authentication and prevents eavesdropping and impersonation. Encrypted Key Exchange (EKE) [7], Augmented EKE (AEKE) [8], Simple Password Exponential Key Exchange (SPEKE) [25], Password Derived Moduli (PDM) [30], and Secure Remote Password (SRP) [52, 51] are Strong password protocols. For mutual authentication, TLS-SRP [47] and TLS-PSK (Transport Layer Security Pre-shared Key) [17] must be used. Some of the objectives of TLP are achieved by the SRP, but not all the objectives of TLP are achieved by SRP. For example, TLP supports universal passwords but SRP does not. Also, TLP is simpler than SRP. For example, TLP is based on standard functions for performing secure hash and symmetric encryption and decryption, whereas SRP is based on a function for performing exponentiation.

9 Concluding Remarks

In this paper, we present a comprehensive proposal to counter human mistakes and Pharming and Phishing attacks that may occur when a user attempts to log in a secure web site. Our proposal is based on two ideas. First, we introduce a new classification, brown, of secure https web pages. When the browser of a user U classifies a displayed page brown, user U should conclude that the displayed page is secure and can enter his sensitive data into it. Second, we design a new login protocol, named TLP, that is secure against human mistakes and Pharming and Phishing attacks. The browser of a user U uses TLP to classify a displayed page brown according to two rules:

1. The displayed page is called into the browser using TLP.
2. The displayed page is called into the browser, using TLS, from within another brown page that was displayed earlier on the browser.

Note that TLP is not intended to replace TLS. On the contrary, our vision assigns complementary roles to be played by TLP and TLS: TLP can be used first to securely log in a web domain, then TLS can be used later to securely go from one web site to another within the logged in domain.

Note also that some mildly secure web domains may feel that they are in no danger of facing Pharming or Phishing attacks because adversaries have little incentive to launch such attacks against these domains. (Examples of such domains are those that host electronic reviewing and handling of submitted papers to conferences and journals.) These web domains can keep on employing TLS, as they do presently, both for logging in a domain and for going from one web site to another within this domain.

A nice feature of TLP is that a user can use the same username and same (TLP universal) password to securely log in any web site in the Internet. This means that the user needs only to memorize one username and one password for all web sites. Therefore it is reasonable to demand that each user chooses a long string, say of sixteen characters, to be his TLP universal password. And so TLP becomes naturally secure against online and offline dictionary attacks.

References

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), Mar. 2005.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard), Mar. 2005. Updated by RFC 4470.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard), Mar. 2005. Updated by RFC 4470.

[4] M. Atighetchi and P. Pal. Phishbouncer: An https proxy for attribute-based prevention of phishing attacks. In *submitted to The second APWG eCrime Researchers Summit*, 2007.

[5] D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833 (Informational), Aug. 2004.

[6] Bank of America. `http://www.bankofamerica.com/privacy/sitekey/`.

[7] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. pages 72–84.

[8] S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM Conference on Computer and Communications Security*, pages 244–250, 1993.

[9] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 4366 (Proposed Standard), Apr. 2006.

[10] S. Chiasson, P. C. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, pages 1–1, Berkeley, CA, USA, 2006. USENIX Association.

[11] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell. Client-side defense against web-based identity theft. In *11th Annual Network and Distributed System Security Symposium (NDSS '04)*, 2004.

[12] R. Dhamija. Hash visualization in user authentication. In *Proceedings of the Computer Human Interaction 2000 Conference.*, April 2000.

[13] R. Dhamija, J. Tygar, and M. Hearst. Why phishing works. In *the Proceedings of the Conference on Human Factors in Computing Systems (CHI2006)*, 2006.

[14] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*, pages 77–88, New York, NY, USA, 2005. ACM Press.

[15] eBay Toolbar and A. Guard. `http://pages.ebay.com/help/confidence/account-guard.html`.

[16] C. Ellison and B. Schneier. Ten risks of pki: What you're not being told about public key infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[17] P. Eronen and H. Tschofenig. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279 (Proposed Standard), Dec. 2005.

[18] R. Franco. Website identification and extended validation certificates in IE7 and other browsers. IEBlog, Nov 2005.

[19] T. Freeman, R. Housley, A. Malpani, D. Cooper, and W. Polk. Server-Based Certificate Validation Protocol (SCVP). RFC 5055 (Proposed Standard), Dec. 2007.

[20] E. Gabrilovich and A. Gontmakher. The homograph attack. *Commun. ACM*, 45(2):128, 2002.

[21] V. Griffith and M. Jakobsson. Messin' with texas, deriving mother's maiden names using public records. In M. Y. J. Ioannidis, A. Keromytis, editor, *Applied Cryptography and Network Security: Third International Conference, Lecture Notes in Computer Science*, volume 3531. Springer Berlin / Heidelberg, June 2005.

[22] A.-P. W. Group. Phising activity trends report. `http://www.antiphishing.org/reports/apwg_report_sept_2007.pdf`, Sept 2007.

[23] A. Herzberg. Trustbar: Re-establishing trust in the web. `http://www.cs.biu.ac.il/~herzbea/TrustBar/`, 2006.

[24] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280 (Proposed Standard), Apr. 2002. Updated by RFCs 4325, 4630.

[25] D. P. Jablon. Strong password-only authenticated key exchange. *Computer Communication Review*, 26(5):5–26, 1996.

[26] C. Jackson, D. R. Simon, D. S. Tan, and A. Barth. An evaluation of extended validation and picture-in-picture phishing attacks. In *Usable Security*, 2007.

[27] A. Juels, M. Jakobsson, and T. N. Jagatic. Cache cookies for browser authentication (extended abstract). In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 301–305, Washington, DC, USA, 2006. IEEE Computer Society.

[28] A. Juels, M. Jakobsson, and S. Stamm. Active cookies for browser authentication. In *14th Annual Network and Distributed System Security Symposium (NDSS '07)*, 2007.

[29] C. Karlof, U. Shankar, J. D. Tygar, and D. Wagner. Dynamic pharming attacks and locked same-origin policies for web browsers. In *ACM Conference on Computer and Communications Security*, pages 58–71, 2007.

[30] C. Kaufman and R. Perlman. PDM: a new strong password-based protocol. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pages 23–23, Berkeley, CA, USA, 2001. USENIX Association.

[31] C. Masone, K.-H. Baek, and S. Smith. WSKE: Web server key enabled cookies. In *Usable Security (USEC)*, 2007.

[32] R. McMillan. Gartner: Consumers to lose $2.8 billion to phishers in 2006. `http://www.networkworld.com/news/2006/110906-gartner-consumers-to-lose-28b.html`, 2006.

[33] Microsoft. Erroneous verisign issued digital certificates pose spoofing hazard. Technical Report Microsoft Security Bulletin MS01-017, 2001.

[34] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (Proposed Standard), June 1999.

[35] Netcraft. `http://www.netcraft.com`.

[36] O. Nordström and C. Dovrolis. Beware of BGP attacks. *SIGCOMM Comput. Commun. Rev.*, 34(2):1–8, 2004.

[37] G. Ollmann. The pharming guide. `http://www.ngssoftware.com/papers/ThePharmingGuide.pdf`.

[38] OpenDNS. `http://www.opendns.com`.

[39] PhishTank. `http://www.phishtank.com`, Nov 2006.

[40] D. Pinkas and R. Housley. Delegated Path Validation and Delegated Path Discovery Protocol Requirements. RFC 3379 (Informational), Sept. 2002.

[41] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*, pages 2–2, Berkeley, CA, USA, 2005. USENIX Association.

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

[43] B. Schneier. Semantic attacks: The third wave of network attacks. Crypto-Gram Newsletter, Sept 2006.

[44] P. Security. `http://www.passmarksecurity.com`.

[45] SpoofStick. `http://www.spoofstick.com`.

[46] S. Stamm, Z. Ramzan, and M. Jakobsson. Drive-by pharming. In *ICICS*, pages 495–506, 2007.

[47] D. Taylor, T. Wu, N. Mavroyanopoulos, and T. Perrin. Using srp for tls authentication draft-ietf-tls-srp-14. IETF TLS Working Group: http://www.ietf.org/internet-draft/draft-ietf-tls-srp-14.txt, December 2007.

[48] A. Whitten and J. D. Tygar. Why johnny can't encrypt: a usability evaluation of pgp 5.0. In *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.

[49] M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610, New York, NY, USA, 2006. ACM Press.

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

[51] T. Wu. Srp-6: Improvements and refinements to the secure remote password protocol. Submission to the IEEE P1363 Working Group.

[52] T. Wu. The secure remote password protocol. In *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, pages 97–111, 1998.

[53] Yahoo. `http://help.yahoo.com/l/us/yahoo/edit/privacy/edit-39.html`.

[54] J. Youll. Fraud vulnerabilities in sitekey security at bank of america. Technical report, Challenge/Response,LLC, July 2006.

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

[56] Y. Zhang, J. Hong, and L. F. Cranor. CANTINA: A content-based approach to detecting phishing web sites. In *Proceedings of the 16th International conference on World Wide Web*, 2007.