Hardening Web Browsers Against Man-in-the-Middle and Eavesdropping Attacks

Haidong Xia and José Carlos Brustoloni
Department of Computer Science
University of Pittsburgh
210 S. Bouquet St. #6135
Pittsburgh, PA 15260 — USA
{hdxia,jcb}@cs.pitt.edu

ABSTRACT

Existing Web browsers handle security errors in a manner that often confuses users. In particular, when a user visits a secure site whose certificate the browser cannot verify, the browser typically allows the user to view and install the certificate and connect to the site despite the verification failure. However, few users understand the risk of man-in-the-middle attacks and the principles behind certificate-based authentication. We propose context-sensitive certificate verification (CSCV), whereby the browser interrogates the user about the context in which a certificate verification error occurs. Considering the context, the browser then guides the user in handling and possibly overcoming the security error. We also propose specific password warnings (SPW) when users are about to send passwords in a form vulnerable to eavesdropping. We performed user studies to evaluate CSCV and SPW. Our results suggest that CSCV and SPW can greatly improve Web browsing security and are easy to use even without training. Moreover, CSCV had greater impact than did staged security training.

Categories and Subject Descriptors

H.5.2 [**Information Systems**]: User Interfaces—*interaction styles, screen design, evaluation*; H.1.2 [**Information Systems**]: User/-Machine Systems—*human factors*; I.3.6 [**Computing Methodologies**]: Methodologies and Techniques—*interaction techniques*

General Terms

Security, Human Factors

Keywords

Web browser, HTTPS, SSL, certificate, password, man-in-the-middle attack, eavesdropping attack, just-in-time instruction, well-in-advance instruction, safe staging

1. INTRODUCTION

Users often find the Web's hypertext and browsing paradigms attractive, intuitive, and easy-to-use. Consequently, Web-based interfaces are now used in a wide variety of client-server applications. Among those applications, there are many that are security-sensitive, including access to bank and other financial accounts, e-commerce sites, and Web-based email. In such applications, security breaches could cause users major financial or privacy losses. However, many users employ Web browsers in such applications without ever receiving formal training on how to do so.

The technology necessary for securing Web applications is generally considered to be well-understood. Secure Web servers typically use the HTTPS protocol [34]. HTTPS layers HTTP (Hypertext Transfer Protocol) [9] over SSL (Secure Socket Layer) [11] (or its standard equivalent, TLS (Transport Layer Security) [7]). SSL offers communication confidentiality and integrity with certificate-based server authentication and protection against replay attacks. Using up-to-date cryptographic algorithms, such as SHA-1 [18] and AES [17], SSL can provide a high level of security.

However, the usability of this security technology has received surprisingly little attention in the literature [27]. User interfaces that are difficult to learn or prone to misuse can expose users to unacceptable risks, even if the underlying technology is secure [3, 29].

This paper addresses three questions on the usability of Web browser security. First, given a group of computer-literate users, how likely is it that an attack against them will succeed, in representative security-sensitive Web applications? Second, is it possible to foolproof Web browsers, such that they can be used securely even by untrained computer-literate users? Third, can education about the relevant security principles, attacks, and tools improve the security of how users browse the Web? We consider specifically attacks enabled by tools that are easily available on the Web: man-in-the-middle (MITM) attacks against HTTPS-based access to financial accounts and e-commerce sites [28], and eavesdropping attacks against HTTP-based email access [35, 8]. Given the ease with which such attacks can be deployed even by inexperienced attackers, the probability that they succeed should be very low.

We performed a user study to answer the first question. Our results show that, with current users and Web browsers, the mentioned attacks are alarmingly likely to succeed. More often than not, users' behavior defeats the existing Web security mechanisms.

In response to the second question, we contribute two novel user interface techniques for Web browsers, CSCV (Context-Sensitive Certificate Verification) and SPW (Specific Password Warnings). CSCV's goal is to thwart MITM attacks against HTTPS and other protocols that use certificates to authenticate servers. SPW cautions users against sending passwords in a form vulnerable to eavesdropping. We implemented CSCV and SPW in a Web browser and evaluated them in a second user study, involving the same users

WWW 2005, May 10–14, 2005, Chiba, Japan.
ACM 1-59593-046-9/05/0005.

and attacks as the first study. CSCV blocked MITM attacks against HTTPS-based applications completely. SPW greatly reduced the insecure transmission of passwords in an HTTP-based application. Although untrained, users had little trouble using CSCV and SPW. These results suggest that, at least for some security tasks, it is indeed possible to design user interfaces that are less error-prone for untrained users.

To answer the third question, we trained users from the first study on security principles, attacks, and tools. We then repeated the experiment using unmodified browsers. Our results show that education can indeed greatly improve how securely users behave. However, security education had significantly less impact than did CSCV and had about the same impact as did SPW.

The rest of this paper is organized as follows. Section 2 shows how easily available tools enable eavesdropping and MITM attacks against Web-based applications. Section 3 summarizes how HTTPS and server certificates would theoretically protect Web-based applications from such attacks, and discusses why current Web browsers allow users to defeat such protection. Section 4 analyzes the causes for certificate verification error, distinguishing contexts where errors are common and recoverable, from contexts where errors are clearly anomalous and suggestive of an attack. Sections 5 and 6 describe CSCV and SPW, respectively. Section 7 discusses the design principles underlying CSCV and SPW. Section 8 reviews an alternative approach for security training, and proposes a variant suitable for browsers, Staged PKI Client (SPKIC). Sections 9 and 10, respectively, report and discuss the results of user studies performed for evaluating CSCV, SPW, and SPKI. Sections 11 and 12 discuss related and future work, and Section 13 concludes.

2. EAVESDROPPING AND MAN-IN-THE-MIDDLE ATTACKS

This section explains how easily available tools enable eavesdropping and MITM attacks against Web applications.

Eavesdropping attacks are enabled by the use of shared media in networks such as Ethernet (with hubs) and Wi-Fi. In an eavesdropping attack, the attacker configures the respective network interface in promiscuous mode. In this mode, the attacker's computer receives any packets sent on the network, including packets destined to other nodes. If packets are unencrypted, the attacker can read packets' data, possibly including passwords and other credentials. Many easily available applications can be used for eavesdropping, including `tcpdump` [35] and `ethereal` [8].

In networks that do not use shared media (e.g., switched Ethernet) or where packets are encrypted, an attacker may be able to use a MITM attack to intercept communication between a client and a server. By impersonating the server, the attacker may be able to fool the client into connecting with the attacker rather than the server. The attacker can then capture the client's credentials (e.g., id and password), and use those credentials to connect to the server as the client. The attacker can relay packets between the client and the server, making the communication appear normal. However, the attacker can read, modify, inject, or drop any packet, even if client and server authenticate and encrypt all packets.

Tools for performing MITM attacks are freely available on the Web. For example, `arpspoof` and `dnsspoof` can be used when client and attacker are connected to the same network [28], as illustrated in Fig. 1. The first tool, `arpspoof`, sends to the client spoofed ARP packets that associate the attacker's MAC address with a local router's IP address. Consequently, the client sends to the attacker packets that would normally be forwarded through the router. The second tool, `dnsspoof`, sends to the client spoofed DNS packets that associate the attacker's IP address with a server's domain name. It causes the client to send to the attacker packets that are actually destined to the server. Another tool, `webmitm`, relays intercepted HTTP and HTTPS traffic between client and server [28]. It usually allows an attacker to capture client credentials effortlessly, even if communication with the server is SSL-secured.

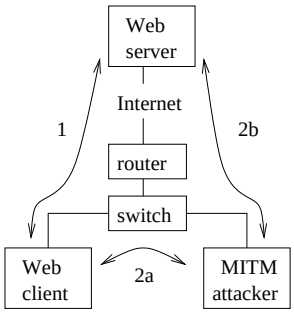

Figure 1: Before establishing a connection with a Web server (1), the client needs the MAC address of the router that connects it to the Internet (obtained using the ARP protocol) and the IP address of the server (obtained using the DNS protocol). By sending spoofed ARP or DNS responses to the client, a man-in-the-middle attacker (MITM) can cause the client to send to the MITM packets intended for the server (2a). The MITM can then eavesdrop, forge, or modify packets between client and server (2b), even if SSL is used.

On Wi-Fi networks, MITM attackers can use another easily available tool, `airsnarf` [25]. This tool includes a rogue access point utility with built-in DHCP, DNS, and HTTP spoofing. Attackers can cause clients to reassociate to the rogue access point by presenting a stronger signal than that of a legitimate access point (e.g., by being closer to the client or by using an amplifier or directional antenna). Attackers thus intercept all packets between client and the server, and can easily capture client credentials.

3. CONVENTIONAL CERTIFICATE VERIFICATION

This section discusses why, in theory, MITM attacks against secure Web servers would not be possible, and why, in practice, clients' Web browsers enable such attacks.

In principle, MITM attacks would not be possible in secure Web communication using HTTPS. HTTPS uses SSL. In SSL, the client authenticates the server using public-key cryptography and the server's certificate, as explained in the following.

A *principal* (e.g., Web server) can prove its identity to a *challenger* (e.g., client) by encrypting with the principal's *private key* and a public-key encryption algorithm (e.g., RSA [22]) nonces received from the challenger. (A nonce is a cryptographically random number that is never reused.) Each private key is known only by the respective principal. The private key corresponds to a *public key* that can be publicly known. Any challenger that knows a principal's public key can use it to decrypt the principal's response and verify the principal's identity. Challengers typically rely on *certificates* to determine a principal's public key. Certificates are data structures that associate a principal with a public key and are signed by a *certifying authority* (CA). The signature is a secure hash of the data structure, encrypted with the CA's private key. Any party that knows a CA's public key can validate the CA's signature and thus

Figure 2: Internet Explorer 6.0 dialog when server certificate cannot be verified because public key of issuing CA is unknown.

verify certificates issued by the CA. The public keys of major CAs (e.g., Verisign) are embedded in many client applications (e.g., Web browsers).

It is generally believed that it is infeasible for attackers to break public-key cryptographic algorithms. Consequently, when a server uses HTTPS, it would be infeasible for an attacker to impersonate the server, as necessary for a MITM attack.

However, the current state of Public Key Infrastructure (PKI) deployment is such that legitimate Web servers often present to clients certificates that clients cannot verify. To accommodate such servers, Web browsers typically allow users to override certificate verification errors and establish an HTTPS connection even when the server has not been authenticated. Fig. 2 shows a representative dialog of the Internet Explorer 6.0 browser in such a situation.

Unfortunately, the ability to override certificate verification errors exposes users to the risk of MITM attacks even when HTTPS is used. An attacker can easily forge a certificate that associates a server with a public key whose corresponding private key the attacker knows. The attacker cannot get a major CA to sign such a certificate (major CAs carefully verify identity out-of-band before issuing a certificate). However, the attacker can set up his or her own CA that signs the certificate. A client's browser will be unable to verify such a certificate and will likely display a warning such as the one shown in Fig. 2. If the user overrides the certificate verification error and accepts the attacker's forged certificate, the user will fall prey to a MITM attack, despite SSL protection.

4. WHY DOES CERTIFICATE VERIFICATION FAIL?

This section analyzes the causes of certificate verification errors, characterizes cases where such errors are common rather than anomalous, and critiques how such errors are presented to users in existing Web browsers.

There are three main causes of user-visible certificate verification errors in Web browsers. First, the browser may not know the public key of the CA that issued the server's certificate. Second, the issuer's or the server's certificate may be expired. Third, the server may have presented a certificate whose *common name* field does not match the server's fully qualified domain name. (Other certificate errors, such as syntax or signature errors, typically cause browsers to reject a connection without giving users override ability.)

The first cause occurs frequently in the case of an *internal* Web server, i.e., a Web server intended for use only by members of the organization that owns it (e.g., a university's students or a company's employees). Major CAs charge a significant annual fee for issuing a certificate for each server. Many organizations therefore choose to establish their own *private CAs*, i.e., CAs not certified by a major CA, for issuing certificates for internal servers. However, the public key of a private CA needs to be installed in client Web browsers to avoid verification errors. When the number of users is high or client computers are user-owned, this chore often ends up being neglected, and certificate verification errors commonly ensue. The existence of private CAs is a de facto situation that does not actually conform to existing PKI standards. Therefore, no firm guidelines can be found in the literature for how to handle them. Existing browsers simply leave the matter on the users' hands, by giving users override ability when such errors happen.

On the other hand, for *public* Web servers, i.e., Web servers open to the public, e.g., in bona fide financial institutions or e-merchants, errors due to the first cause are unexpected and suggestive of a MITM attack.

Certificate verification errors due to the second cause, expiration, could be caused by something as benign as inattentiveness of system administrators, and cannot be easily exploited by MITM attackers. It appears justified in this case to warn users and give them the ability to proceed, as is currently done.

Errors due to the third cause, name mismatch, can be very serious. Instead of using a self-signed certificate with a fake identity, a MITM attacker can use his or her real identity on a certificate from a major CA. Alternatively, a MITM attacker may use a major-CA certificate previously stolen from another server (along with the respective private key). In MITM attacks using certificates issued by major CAs, the certificate's only flaw may be that it belongs to a common name that differs from that of the Web server the user wishes to access. If the domains (last two components of the names) are distinct, the possibility of a MITM attack is high, and the user should not be allowed to proceed. On the other hand, name mismatch at the subdomain level (e.g., `s10.acme.com` vs. `s3.acme.com`) might be caused by sloppy management or server rearrangement at the target organization. It would also be difficult for a MITM attacker to get from a major CA a certificate with name mismatch only at the subdomain level. It therefore seems reasonable in the latter cases to warn users and give them the opportunity to proceed, as is currently done.

Certificate verification errors in existing Web browsers cause the browser to identify the cause(s) to the user and allow the user to establish a connection despite the error. Fig. 2 shows an example from Internet Explorer 6.0. Note that some of the information presented to the user is misleading. The alert says that "information you exchange with this site cannot be viewed or changed by others," which is incorrect in the case of a MITM attack. Moreover, the alert invites the user to view the certificate to decide whether to trust the CA. However, if the browser was unable to verify the issuer's signature on a certificate, viewing the certificate does not actually provide any basis for trust, since all the information in the failed certificate could be forged by a MITM attacker. Note also that the alert does not indicate the severity of the error or provide alternatives for overcoming it.

5. CONTEXT-SENSITIVE CERTIFICATE VERIFICATION

This section describes CSCV, a novel user interface technique that applies the previous section's analysis. CSCV guides users for

Figure 3: CSCV accepts private CA certificates on removable media.

Figure 4: CSCV dialog when user is internal member of the organization that owns the accessed server.

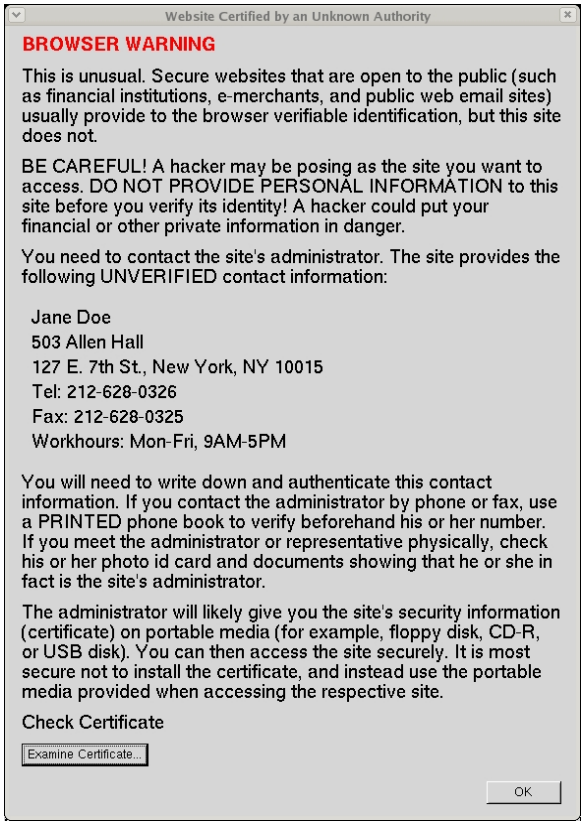

Figure 5: CSCV dialog when user is simply a client of the organization that owns the accessed server.

secure handling of certificate verification errors, without override mechanisms.

CSCV thwarts MITM attacks against HTTPS by clarifying the relationship between the user and the server whose certificate verification failed. As explained in the previous section, internal Web servers routinely cause certificate verification errors. To accommodate such servers without giving users override mechanisms, CSCV introduces tokens and modifies Web browsers and certificates issued by private CAs as explained in the following paragraphs.

CSCV-aware private CAs give to the respective organization's members tokens containing the CA's self-signed certificate. This certificate includes the CA's public key and is signed with the CA's private key. The tokens are distributed to members securely out-of-band on removable media, such as CDR, USB, or floppy disks. CSCV-aware Web browsers allow organization members to employ these tokens for verifying server certificates issued by the organization's private CA.

CSCV-aware private CAs also include the CA's *contact information* in the *issuer alternative name* field of server certificates they issue [14]. This field typically would otherwise be unused. A CA's contact information normally includes the CA administrator's name, address, telephone and fax numbers, and work hours. CSCV-aware Web browsers present this information to users to guide them in the recovery from certificate verification errors.

When certificate verification fails because the public key of the certificate's issuer is unknown, CSCV-aware Web browsers ask whether the user has the necessary security information (issuing CA's self-signed certificate) on removable media (e.g., floppy or USB disk), as shown in Fig. 3.

If the user does not have the CA's certificate, CSCV-aware browsers ask whether the user is an internal member (e.g., student or employee) of the organization that owns the server the user wishes to access. If so, the browser displays the CA's contact information and tells the user how to verify the contact information and the administrator, as illustrated in Fig. 4. Users thus learn how to obtain, securely and out-of-band, the private CA's self-signed certificate. After installing the certificate, users can authenticate the organization's servers securely, without resorting to overrides.

On the other hand, if the user is simply a client of the organization that owns the server, CSCV-aware browsers warn the user that the situation is unusual, because secure Web sites that are open to the public (such as those of financial institutions and e-merchants) typically can be verified by client software (since the respective certificates are usually issued by major CAs). The browser also explains in plain language, as illustrated in Fig. 5, that a MITM attack may be happening, and that an attacker could jeopardize the user's financial or other private information. The browser displays the CA's contact information and warns that the latter needs to be verified out-of-band. The browser then instructs the user how to (try to) obtain the CA's certificate. The user cannot connect to the server without first obtaining the CA's certificate and securely authenticating the server.

6. SPECIFIC PASSWORD WARNINGS

This section critiques how existing browsers treat transmission of unencrypted data, and introduces SPW, a new user-interface technique specifically for alerting users against transmitting passwords unencrypted.

Existing Web browsers can be configured to warn users when

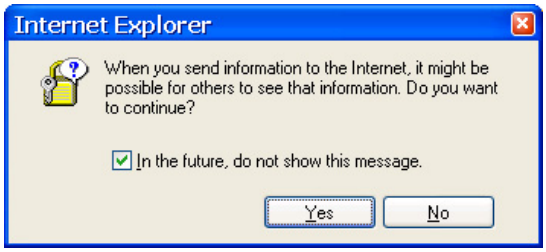

Figure 6: Internet Explorer 6.0 dialog when the user is about to send unencrypted data.

Figure 7: SPW dialog when user is about to send password unencrypted.

they are about to send unencrypted data. Fig. 6 shows a representative example from Internet Explorer 6.0. Because these warnings do not discriminate between passwords and other data, they happen quite often. Moreover, they do not explain to users the possible threats and consequences of sending a password unencrypted. Consequently, many users disable or ignore such warnings.

SPW-aware browsers detect when a user is about to transmit a password in plaintext (e.g., when accessing an insecure Web server) and ask the user whether the password protects an account that the user wouldn't want strangers to access (e.g., bank or email account), as shown in Fig. 7. If so, the browser strongly discourages the user from continuing. The browser informs that such accounts should be accessed securely and explains simply how the user can tell whether a site is being accessed securely. The browser also asks the user to consider whether the current server is an insecure replica of a server that the client normally accesses securely, and explains that such a replica could be used in a MITM attack. The browser cautions the user that an eavesdropper on the network may be able to capture the user's password and later impersonate the user, with possibly significant financial or privacy loss to the user, as shown in Fig. 8. Finally, the browser asks whether the user is willing to accept all the mentioned risks.

On the other hand, if the user wouldn't mind strangers having access to the contents of the account, an SPW-aware browser still cautions the user that an attacker might be able to capture the user's password and later impersonate the user. The browser then asks whether the user wishes to continue.

SPW-aware browsers can detect in one of two ways that a plaintext password is about to be sent. First, when a browser receives from a server an HTTP 401 (Unauthorized) response message, the browser would normally pop up a window where the user enters his or her id and password. If HTTP's Basic authentication scheme is the strongest authentication alternative allowed by the 401 response and understood by the browser, the browser would then retry the request that caused the 401 response, including the user's plaintext password [10]. Therefore, in such cases, before popping up the authentication window, SPW-aware browsers execute the SPW dialog described in the above paragraphs. Only if the user insists in continuing, the browser pops up the authentication window and, after user

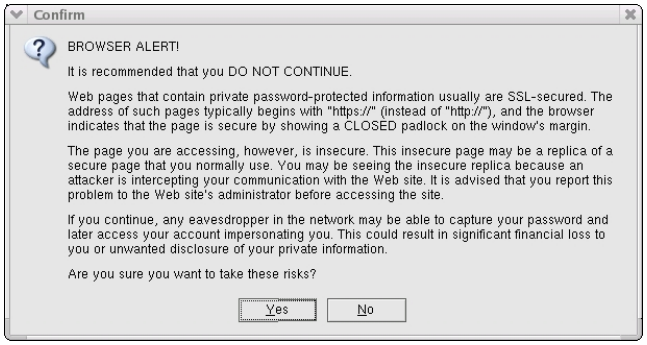

Figure 8: SPW follow-on dialog when user indicates that password protects an important account.

input, sends the id and password unencrypted to the server. The browser also associates the id and password with the *realm* specified in the 401 message, and caches them for the current browsing session. The cache makes it unnecessary for the user to enter the same information again when the browser receives a new 401 message in the same realm.

Second, the user may enter data into an HTML form with an input element of type *password*. (Form input elements of this type echo as asterisks the characters the user types.) When the user submits such a form, an SPW-aware browser checks the action associated with the form. If the action specifies a URL that uses HTTP rather than HTTPS, the browser executes the SPW dialog and starts the action only if the user insists in continuing.

Note that, unlike CSCV, SPW allows user overrides. This is because some legitimate password-protected Web servers, e.g., for Web-based email access, do not use HTTPS. In cases where passwords entered into an HTML form are processed by Javascript in the user's browser, servers may implement challenge/response schemes that actually protect user passwords reasonably well from eavesdropping and replay attacks [10]. Although such schemes do not protect the accessed content from eavesdropping, a user override in such cases may be justified.

7. JUST-IN-TIME INSTRUCTION

This section discusses the design principles underlying CSCV and SPW.

Currently, most personal computer software uses some form of *just-in-time instruction* (JITI) [6] to handle security errors. JITI gives the user information that the software writer deems important for the user to make a decision at a critical point of the processing of a task or error. At such a point, the software presents to the user a usually short message on a pop-up window. Interested users can learn more by following hyperlinks embedded in the message, or by searching the software's help facility. Users usually can also simply dismiss the message.

JITI can have many shortcomings. First, JITI messages often use concepts that nonspecialists do not understand. Second, JITI messages may not fully or correctly disclose possible consequences of users' decisions [16] (see, e.g., Fig. 2). Third, JITI messages usually do not tell users how they might overcome security errors: the software simply asks user permission to continue a task, despite lack of security. Fourth, because of the above problems, when JITI presents a security warning, time-pressed nonspecialists often simply click "continue," without fully investigating and understanding the risks involved. Over time, this behavior can become a hard-to-kick habit of dismissing JITI warnings.

We call *warn and continue* (WC) the class of user interfaces that implement JITI with all four of the above shortcomings. The current Internet Explorer (IE) 6.0 is a good example in this class.

CSCV is representative of another JITI class, which we call *guidance without override* (GWO). User interfaces in this class disclose risks in plain language and fully. They tell the user how to eliminate the conditions that caused a security warning, and do not allow the user to proceed without doing so. The user interface tailors the instructions for the specific situation the user is facing. In order to do so, the user interface may need to gather information from the user, from the server the user wishes to contact, or from other computers.

SPW belongs to yet another JITI class, which we call *guidance with override* (G+O). User interfaces in this class also disclose risks in plain language and fully, and may give users tailored instructions for eliminating the conditions that caused a warning. They may also diagnose the user's situation by inquiring the user. However, unlike GWO, G+O lets the user proceed without eliminating conditions that caused a warning. Because it does not need information or accommodations from servers that users access, G+O may be more broadly applicable than is GWO. On the other hand, G+O is also more vulnerable to user errors than is GWO.

8. WELL-IN-ADVANCE INSTRUCTION

This section discusses an alternative approach for achieving security via user training. In the next section, we experimentally compare an instance of this approach, SPKIC, with CSCV and SPW.

Well-in-Advance Instruction (WIAI) is a user interface design principle recently proposed by Whitten [31] for replacement of JITI. According to Whitten, JITI's main problem is that it leaves instruction to the last minute. Users are then typically focused on a task for which security is not a primary goal. Consequently, users are unlikely to try to learn security concepts at that time. Whitten argues that user interfaces should teach necessary security concepts *before* the user actually needs them.

A good way to implement WIAI, says Whitten, is via *safe staging* [30]. In safe staging, software varies the user interface according to the user's *stage*. Each stage should enable only functions and data that the user knows how to manipulate safely (e.g., the whole functionality on dummy data, or only safe functions on the user's own data). During each stage, software encourages the user to learn the concepts necessary to progress to the next stage. In *hard* staging, the software allows progression only when the user has demonstrated mastery of the relevant content. On the other hand, in *soft* staging there is no such enforcement.

Whitten implemented an email agent, Lime, that uses soft staging to teach users how to use public-key certificates under PGP's Web-of-Trust model. Such certificates are necessary for email authentication and confidentiality. In Lime's first stage, users are assumed to know how to trade keys on removable media. In this stage, the user can send secure email only to people the user has already personally met and traded keys with. In the second stage, the user may send email with the user's self-signed certificate and a few personal remarks that the recipient can use to identify the sender's identity. In this stage, the user can send fairly secure email to any acquaintances, even those with whom the user has not traded certificates; however, the data must not be very important, since security is not high. In the third stage, the user can send secure email to anybody certified by a person the user has previously met and traded keys with. In this stage, there are no functional or data restrictions. We call this methodology *staged web-of-trust* (SWOT).

SWOT is not directly applicable to browsers, among other reasons because SSL certificates are based on PKI, not Web of Trust. We therefore propose another similar soft staging methodology for teaching users how to deal with PKI-based certificates and unprotected passwords in Internet Explorer. We call this methodology *staged PKI client* (SPKIC). In SPKIC's first stage, users learn not to accept unverified certificates and to avoid sending unprotected passwords. Users also learn how to obtain the certificate of a private CA. (Similar instruction is not provided in Lime because the notion of meeting an intended email correspondent and trading keys is considered intuitive enough. However, in the case of a Web browser, it is not obvious to the user who, where, and when to meet in order to get a CA's certificate securely.) SPKIC-aware software enforce the mentioned restrictions so as to make this stage safe. In SPKIC's second stage, users learn about MITM and eavesdropping attacks and how to use OpenSSL to set up a private CA and issue bona-fide and bogus certificates. Users also learn how to modify an SSL client program so that it uses the OpenSSL library to verify certificates received from servers. (Whereas the first stage taught the "how" of dealing with certificates and passwords, the second stage focuses on the "why." SWOT's second stage achieves a similar goal using a simpler "social authentication" paradigm. Social authentication is well-suited to email, but unfortunately not applicable to Web browsing.) This stage is safe because only dummy data is used. Finally, in SPKIC's third stage, users employ unmodified Internet Explorer to visit a variety of sites. In this stage, there are no restrictions because the user is assumed to have the knowledge necessary to behave safely.

9. EXPERIMENTAL RESULTS

This section reports the results of three user studies performed to evaluate CSCV, SPW, and SPKIC.

The users in the first two studies were 17 male Computer Science undergraduates in their senior year. The first user study provides a baseline for security before any browser modifications or training about certificates and MITM and eavesdropping attacks. In it, users employed an unmodified Internet Explorer 6.0 browser. The second user study was performed back-to-back with the first study. In it, the same users employed a modified version of the Mozilla Firebird 0.6.1 browser with CSCV and SPW. No feedback or further information was given to students between the first two studies.

The second user study served also as the first stage of the third study. Twelve of the original seventeen students then received training on certificates and MITM and eavesdropping attacks, according to the SPKIC methodology described in the previous section. This stage lasted approximately one month. Students then progressed to SPKIC's third stage, where they again used the unmodified Internet Explorer 6.0 browser.

In each user study, we asked students to visit three fictional but realistic Web sites where students were assigned password-protected accounts. The first site is maintained by the students' university. It allows students to monitor the respective *reward points*. Students earn these points by doing well in exams, independent studies, or service to the university. The second site is maintained by a remote e-merchant that is not affiliated with the university and where students can spend their reward points, e.g., to buy books, CDs, or spring break vacations. The third site provides access to users' Web email accounts. We asked students to verify their balance on the first site, spend some of it by ordering something from the second site, and get an order confirmation message on the third site. We asked students to consider that reward points have real monetary value and to exercise regular care while browsing. We pointed out that a student could decide not to visit a site if he deemed the visit too risky. We asked students to "think aloud" while browsing and describe what they were doing and why.

		Study 1: 17 users			Study 2: 17 users			Study 3: 12 users		
Browser		Unmodified Internet Explorer			Modified Mozilla Firebird			Unmodified Internet Explorer		
UI methodology		Warn and continue (WC)			CSCV		SPW	Staged PKI client (SPKIC)		
Site type		UC/M	UC/C	NC	UC/M	UC/C	NC	UC/M	UC/C	NC
Score frequency	0	17	16	11	0	0	2	6	5	2
	50	0	–	–	1	–	–	2	–	–
	100	0	1	6	16	17	15	4	7	10
Average Score		0	6	35	97	100	88	42	58	83

Table 1: Users' security scores when accessing sites of various types. CSCV and SPW greatly improved security scores of untrained users on sites of all types. The effect of security training with SPKIC was smaller than that of CSCV, but similar to that of SPW. Please consult Table 2 for statistical significance of these results. On Tables 1 and 2, "UI" stands for user interface, "UC/M" represents site with unverified certificate and belonging to organization user is member of, "UC/C" is site with unverified certificate and belonging to organization user is simply a client of, and "NC" is site without certificate (no SSL).

All studies took place in a laboratory that the students had already been using for course assignments. By the student's computer, we provided a telephone and phone directories (phone company's local white and yellow pages and the university's directory). We configured the first two sites with HTTPS and server certificates issued by private CAs whose public key was unknown by client software. The first site's CA contact information was that of a real person listed on the university's directory, with office on a different floor in the same building as the laboratory. The second site's certificate was bogus. We configured the third site with HTTP only (no SSL).

We scored how securely users accessed the sites as follows. If a user accessed a site despite lack of security, the user got 0 points. In the first site, if a user simply did not visit the site insecurely, the user got 50 points. If the user also correctly obtained and installed the issuing CA's certificate and thus accessed the server after properly authenticating the server, the user got 100 points. Lack of security in the second and third sites could not be corrected. Thus, users who simply did not visit each site insecurely got 100 points. The students' security scores are represented on Table 1.

In the first study, users' think-aloud comments indicated that they were quite familiar with Internet Explorer's dialog for unverified server certificates (Fig. 2). A typical comment was "um, another of those pop-ups." Most users dismissed the warning with a comment like "I always just click yes when I see these pop-ups." Other users viewed the servers' certificates and, with only one exception, accepted them. A typical comment was, "looks legit to me." The one exception was a user who expressed concern about the e-merchant's certificate not being signed by a major CA. However, the same user accepted the certificate of his school's server, which also was unverified and could have been forged. Interestingly, about a third of the users found that sending their password unencrypted was too risky and decided not to visit the third site.

In the second study, users initially displayed some uneasiness about the modified browser's dialogs (Figs. 4 and 5). The dialogs ask the user to contact and verify the site's administrator. A typical user reaction was, "Am I really supposed to do this?" Many users tried to start again and give different answers so as to be able to access the site despite the certificate verification error. However, the modified browser does not provide any way to override the verification process, and users eventually followed the browser's directions for obtaining the private CA's certificate. The one exception was a user who decided not to visit the first site at all. The modified browser was very successful also at discouraging unencrypted transmission of passwords. A typical comment to the dialog in Fig. 8 was, "I guess I better not access this site."

In the third study, about half of the users were deliberate, examined the certificates, commented on the possibility of forgery and man-in-the-middle attack, and decided not to visit the sites. However, the other users continued to accept unverified server certificates, with comments similar to those in the first study. Most users decided not to send unencrypted passwords, with comments like "it's too risky."

Table 2 shows the statistical evaluation of the hypotheses tested by the user studies. Roughly, p-values estimate the likelihood that observed differences (shown in Table 1) are due to chance. Experimental evidence in favor of a hypothesis is by usual convention considered *highly significant* if p-value $\leq 1\%$, *significant* if $1\% < p$-value $\leq 5\%$, or *not significant* otherwise. We used the Wilcoxon signed-ranks (nonparametric) test to determine p-values. We considered Wilcoxon more suitable than a parametric test because the dependent variable in our experiments (i.e., security score) had only two or three possible values. Parametric tests, on the contrary, assume that the dependent variable has a fairly large number of possible values. We used Hommel's procedure to correct p-values for the fact that the user studies test multiple hypotheses [13, 24]. Because the procedure discarded as not significant only one of the hypotheses, the correction factor was 1.

10. DISCUSSION

Results for the first user study suggest that the actual security of existing browsers is appalling, when the "human in the loop" is considered. Because most users dismiss certificate verification error messages, SSL provides little real protection against MITM attacks. Users actually behaved less insecurely when interacting with the site that was *not* SSL-secured.

Results for the second study suggest that CSCV and SPW greatly improve browser security and are easy to use even without training. CSCV is used on the first two sites and resulted in higher scores than did SPW, which is used on the third site. All but one of the users were able to follow CSCV's online guidance for securely obtaining and installing the certificate of the private CA maintained by the user's organization. These users then accessed the organization's server securely. Observation of the users' behavior indicates that the absence of override ability was essential for achieving a high level of security and had only modest impact on usability. None of the users accessed the e-merchant's site insecurely, and most users refused insecure access to Web email.

Caution is needed in interpreting results of user studies performed in a laboratory, such as the ones reported in this paper. When users receive explicit instructions to perform certain tasks and know they are being observed, they may be more likely to complete the tasks than they would be in real life. Such a bias would tend to increase the security score for Study 2's first site, where the

Hypothesis	p-value	Significance
CSCV provides higher security scores than does IE's WC in UC/M sites	< 0.0005	highly significant
CSCV provides higher security scores than does IE's WC in UC/C sites	< 0.0005	highly significant
SPW provides higher security scores than does IE's WC in NC sites	0.003	highly significant
SPKIC provides higher security scores than does IE's WC in UC/M sites	0.023	significant
SPKIC provides higher security scores than does IE's WC in UC/C sites	0.014	significant
SPKIC provides higher security scores than does IE's WC in NC sites	0.008	highly significant
CSCV provides higher security scores than does SPKIC in UC/M sites	0.011	significant
CSCV provides higher security scores than does SPKIC in UC/C sites	0.025	significant
SPW provides higher security scores than does SPKIC in NC sites	not signif.	not significant

Table 2: Statistical evaluation of the hypotheses tested by the user studies. P-values were calculated using Wilcoxon's signed-ranks test, with familywise error rate controlled by Hommel's procedure.

user needs to go meet the CA contact for the score to reach 100. For the other sites and studies, the same bias would tend to depress security scores. Review of observations of user behavior and think-aloud comments suggests that this bias was significant for Study 2's first site, but not for the other sites and studies. We did not control this effect because, in real life, an organization's members usually do have compulsory reasons to access the organization's secure Web sites, and therefore would need to get the organization's private CA's certificate. For example, a student may need secure Web access to a university's servers to register for classes, reserve a book from the library, or check her grades; an employee may need secure Web access to check his email, get a parking permit, or file expense reports.

If a private CA issues certificates *only* for servers whose access is *optional*, it is possible that a significant number of users may not bother to get the CA's certificate or access the protected information. The likelihood of task completion is affected by the difficulty of such completion. In our experiments, the bona-fide CA contact was in the same building as the users. In the case of an organization distributed across several buildings or campuses, it may be advisable to maintain multiple certificates for secure servers. Servers can then use for each client a certificate with CA contact next to the client's location. If this is not possible, or users' motivation to access an organization's secure sites is low, system administrators need to distribute the private CA's certificate to users *proactively* (e.g., on removable media, as suggested in Section 5), rather than wait for users to come looking for them. Alternatively, system administrators may prefer to use a major CA in such applications.

Results of all three user studies could have been biased by users' age, gender, education, and ability. Additionally, results of the third user study could have been biased by the instructor's ability and particular methodology (SPKIC) used. It is unlikely that SPKIC could be used with a much more diverse user population, given the inclusion of programming tasks in SPKIC's second stage. In some sense, SPKIC's second stage may appear to approach an upper bound on the training that could realistically be achieved. However, interventions such as drill exercises using the actual final browser might actually be more effective. It is also possible that SPKIC would be more successful if the final browser were not one which the users had already acquired bad habits with, or if the final browser's user interface were modified, e.g., to end within the G+O class instead of the WC class.

Given the differences in methodology, results of our third user study cannot be readily compared with those of Lime. Lime's study had a more diverse user population, with ages ranging between 18 and 41 years, education ranging from some college to Master's degree, and various areas of expertise, including computer science, engineering, psychology, and history [31]. Users who had used PGP before were excluded from Lime's study. Additionally, Lime incorporated carefully designed messages and visual metaphors that would place its user interface within the G+O class. Nonetheless, at the beginning of Lime's third stage, 8 out of 10 users accepted unverified certificates. Users then received complaints about their own unverified certificates – a common event in PGP email that has no parallel in Web browsing, where clients typically do not have their own certificates. When then presented a correspondent's alleged new public key, 2 out of 9 users accepted the certificate without verification. Thus, at the end of Lime's last stage, a significant fraction of users were still vulnerable to MITM attacks. Although SWOT and SPKIC are quite different, neither staged methodology seems to achieve sufficiently usable security to deter the kinds of attacks we're concerned with in this paper. We suspect that, given human nature, the same is true of any user interface that gives users discretion to proceed without fixing serious security problems, such as unverified certificates (e.g., within JITI, any user interface outside of GWO).

11. RELATED WORK

Captive portals are Web servers used to authenticate users in many Wi-Fi hotspots. Although captive portals are SSL-secured, they are vulnerable to session hijacking, freeloading [36], and MITM attacks. In [36], we demonstrated new defenses against hijacking and freeloading that are easy to use because they are transparent to users and interoperate readily. In [37], we showed how such defenses can be integrated with new Wi-Fi security protocols and billing schemes. This paper's techniques complement that work by helping thwart MITM attacks in such environments.

Most previous work on security ignores human factors. The latter often make computer systems in practice insecure. An increasing awareness of this problem is turning user interfaces into an important area in the security research agenda [3, 39, 2, 26, 23, 27].

Our results are consistent with those of Whitten and Tygar's seminal work on the usability of public-key cryptography for email security [29]. Public-key cryptography is difficult for users to understand and use. Our results for CSCV suggest that, at least in applications of public-key cryptography to server authentication, it is possible to make user interfaces less error-prone.

Several systems secure the online retrieval of certificates by confirming a certificate's *fingerprint* out-of-band (e.g., by printing on the subject's personally delivered business card or by telephone) [21]. Introduction of such a method in CSCV could make CSCV easier to use, since it would allow users to get the private CA's certificate online and confirm it by telephone, instead of personally meeting the CA's contact. User tests need to be performed, however, to verify such a method's effectiveness against MITM attackers.

Identity-based cryptographic algorithms do not require certificates. They use as a party's public key the party's identifier (e.g., fully qualified domain name or IP address). Such algorithms are actively being researched [4] and, if they prove to be practical, they could provide an alternate solution for making Web browsers more usably secure.

Ackerman and Cranor have proposed *critics* that help Web browser users understand and negotiate privacy issues that may be involved in visiting certain sites [1]. Interestingly, critics automatically collect from trusted sites the latest intelligence on privacy threats. We believe that similar self-update capability may be able to significantly enhance the robustness and effectiveness of G+O interfaces, including SPW.

Ye and Smith have shown that an attacker can use Javascript to spoof a browser's usual signs that a site is being accessed securely ("https" on the address line and closed padlock) [38]. They propose "trusted path" mechanisms that allow users to detect such spoofing. However, our results show that, clear warnings notwithstanding, users often access sites insecurely if given the chance. We thus believe that CSCV and SPW would significantly further improve the usable security of browsers with trusted paths.

Whitten's safe staging can be interpreted as an amplification and application to security of two previous techniques: software *training wheels* [5] and tutoring *scaffolding* [12]. Training wheels disable in an initial stage functions that users tend to have problems with, whereas scaffolding gives novices help and examples that are gradually withdrawn as users become more proficient.

A large-scale study involving security training by Yan et al. at Cambridge University [33] obtained results similar to our own. That study attempted to determine the effectiveness of various interventions for reducing password vulnerability to dictionary attacks. The study found that teaching students how to choose more secure passwords can significantly reduce the incidence of vulnerable passwords. However, that incidence was still unsatisfactorily high (at least 10%) after the studied interventions. Yan et al. conjecture that to achieve acceptable compliance, computer systems need to proactively prevent users from choosing weak passwords, rather than simply rely on training.

12. FUTURE WORK

There are many other protocols, besides HTTPS, where clients use public key cryptography to authenticate servers. We are currently investigating how to adapt CSCV to improve the security of the SSH (Secure Shell) [15] and the usability of the PEAP (Protected Extensible Authentication Protocol) [20] user interfaces. Current SSH clients keep the public key of known hosts in a file. When a server presents an unknown or different public key to a client, the client software asks if the user wants to accept the key just for this session or all future sessions. This paper's results suggest that such an interface can effectively defeat SSH's security (note that MITM attack tools against SSH, such as `sshmitm`, are easily available on the Web [28]). PEAP is used in the new native Wi-Fi security scheme, WPA2/802.11i [32]. Current PEAP clients (e.g., Microsoft Windows XP's and Linux's xsupplicant [19]) simply refuse to connect to a network if the latter's authentication server's certificate cannot be verified. Such a design is secure, but difficult to use. We expect CSCV's guidance to improve PEAP usability significantly.

It would be interesting to investigate to what extent habituation may decrease the effectiveness of SPW (or, in general, any G+O interface): After seeing the same warning many times, users may pay less attention to it. On the other hand, CSCV (and, in general, any GWO interface) does not allow user override, and is thus not vulnerable to habituation. CSCV's usability may actually improve as users become familiar with CSCV's guidance.

13. CONCLUSIONS

Web browsers need to handle many security errors. The current practice favors delegating decisions to the user via pop-up windows that allow users to override security failures. Our results suggest that the current practice effectively defeats Web security.

Tools that can be freely downloaded from the Internet enable even novice hackers to perpetrate MITM attacks that cause significant loss to victims. Existing Web security mechanisms, such as server certificates and SSL, in theory protect users from such attacks. However, most users do not check or understand servers' certificates or ignore warnings that a certificate cannot be verified. Consequently, existing Web mechanisms provide little real security to most users. We proposed CSCV, a novel user interface technique for handling certificate verification errors. CSCV asks the user questions and receives from the server information that enable the browser to discriminate the context in which a certificate verification error occurs. Considering this context, the browser then guides the user in handling and possibly overcoming the error. We also proposed SPW for warning users about the specific threats and risks involved when they are about to send unencrypted passwords. We performed user studies to evaluate CSCV and SPW. Our results suggest that CSCV and SPW greatly improve the security of Web browsers and are easy to use even by untrained users.

Acknowledgements

This project was funded in part by the Pittsburgh Digital Greenhouse through a grant from the Commonwealth of Pennsylvania, Department of Community and Economic Development. We thank the anonymous referees for their valuable comments, and Elaine Rubinstein for her help in the statistical analysis of our results.

14. REFERENCES

[1] Ackerman, M. and Cranor, L.: Privacy Critics: UI Components to Safeguard Users' Privacy. In *Proc. Conf. Human Factors in Computing Systems (CHI'99)*, Extended Abstracts, ACM, 1999, pp. 258-259. [Online] `http://lorrie.cranor.org/pubs/privacy-critics.pdf`

[2] Adams, A. and Sasse, M.: Users are not the Enemy. In *Communications of the ACM*, 42(12):41-46, Dec. 1999.

[3] Anderson, R.: Why Cryptosystems Fail. In *Proc. 1st Conf. Computer and Communications Security (CCS'93)*, ACM, 1993, pp. 215-227. [Online] `http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/wcf.pdf`

[4] Boneh, D. and Franklin, M.: Identity Based Encryption from the Weil Pairing, in *J. of Computing*, SIAM, 32(3):586-615, 2003. [Online] `http://crypto.stanford.edu/~dabo/papers/ibe.pdf`

[5] Carroll, J. and Carrithers, C.: Training Wheels in a User Interface. In *Communications of the ACM*, 27(8):800-806, 1984.

[6] Collins, J., Greer, J., Kumar, V., McCalla, G., Meagher, P. and Tkatch, R.: Inspectable User Models for Just-In-Time Workplace Training. In *Proc. 6th International Conference on User Modeling (UM97)*. [Online] `http://www.cs.uni-sb.de/UM97/ps/CollinsJA.ps`

[7] Dierks, T. and Allen, C.: The TLS Protocol Version 1.0. RFC 2246, IETF, Jan. 1999. [Online] `ftp://ftp.rfc-editor.org/in-notes/rfc2246.txt`

[8] ethereal. [Online] http://www.ethereal.com/

[9] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, June 1999. [Online] ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt

[10] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and Stewart, L.: HTTP Authentication: Basic and Digest Access Authentication. RFC 2617, IETF, June 1999. [Online] ftp://ftp.rfc-editor.org/in-notes/rfc2617.txt

[11] Freier, A., Karlton, P., and Kocher, P.: The SSL Protocol Version 3.0. [Online] http://wp.netscape.com/eng/ssl3/draft302.txt

[12] Guzdial, M.: Software-Realized Scaffolding to Facilitate Programming for Science Learning. In *Interactive Learning Environments*, 4(1):1-44. [Online] http://guzdial.cc.gatech.edu/Emile-ILE.pdf

[13] Hommel, G.: A Comparison of Two Modified Bonferroni Procedures. In *Biometrika*, 76:624-625, 1989.

[14] Housley, R., Ford, W., Polk, W. and Solo, D.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459, IETF, Jan. 1999. [Online] ftp://ftp.rfc-editor.org/in-notes/pdfrfc/rfc2459.txt.pdf

[15] Lonvick, C.: SSH Protocol Architecture. Internet Draft, IETF, Oct. 2004. [Online] ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-ietf-secsh-architecture-17.txt

[16] Millett, L., Friedman, B. and Felten, E.: Cookies and Web Browser Design: Toward Realizing Informed Consent Online. In *Proc. Conference on Human Factors in Computing Systems (CHI'2001)*, ACM, Mar. 2001. [Online] ftp://ftp.cs.washington.edu/tr/2000/12/UW-CSE-00-12-03.pdf

[17] National Institute of Standards and Technology. Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197. Nov. 2001. [Online] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[18] National Institute of Standards and Technology. Specifications for Secure Hash Standard. Federal Information Processing Standards Publication 180-1. Apr. 1995. [Online] http://www.itl.nist.gov/fipspubs/fip180-1.htm

[19] Open1x. [Online] http://www.open1x.org/

[20] Palekar, A., Simon, D., Salowey, J., Zhou, H., Zorn, G. and Josefsson, S.: Protected EAP Protocol (PEAP) Version 2. Internet Draft, IETF, Oct. 2004. [Online] ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-josefsson-pppext-eap-tls-eap-10.txt

[21] Perrin, T.: Public Key Distribution Through "CryptoIDs". In *Proc. Workshop on New Security Paradigms*. ACM, 2003, pp. 87-102. [Online] http://trevp.net/cryptoID/cryptoID.pdf

[22] Rivest, R., Shamir, A., and Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the ACM* 21,2 (Feb. 1978), 120–126.

[23] Sandhu, R.: Good-Enough Security: Toward a Pragmatic Business-Driven Discipline. In *Internet Computing*, IEEE, vol. 7, no. 1, Jan. 2003. [Online] http://www.list.gmu.edu/journals/ic/03-sandhu-good.pdf

[24] Sankoh, A., Huque, M. and Dubey, S.: Some Comments on Frequently Used Multiple Endpoint Adjustment Methods in Clinical Trials. In *Statistics in Medicine*, 16:2529-2542, 1997.

[25] Shmoo. airsnarf. [Online] http://airsnarf.shmoo.com/

[26] Smetters, D.K. and Grinter, R.E.: Moving from the Design of Usable Security Technologies to the Design of Useful Secure Applications. In *Proc. Workshop on New Security Paradigms*, ACM, 2002.

[27] Smith, S.: Humans in the Loop: Human-Computer Interaction and Security. In *Security and Privacy*, IEEE, May/June 2003, 75–79. [Online] http://www.cs.dartmouth.edu/~sws/papers/humans.pdf

[28] Song, D. dsniff. [Online] http://naughty.monkey.org/~dugsong/dsniff/

[29] Whitten, A. and Tygar, J.D.: Why Johnny Can't Encrypt: A Case Study. In *Proceedings of Usenix Security Symposium*, Aug. 1999. [Online] http://www.usenix.org/publications/library/proceedings/sec99/full_papers/whitten/whitten.ps

[30] Whitten, A. and Tygar, J.D.: Safe Staging for Computer Security. In *Proceedings of the Workshop on Human-Computer Interaction and Security Systems*, CHI'2003, April 2003. [Online] http://www.andrewpatrick.ca/CHI2003/HCISEC/hcisec-workshop-whitten.pdf

[31] Whitten, A.: Making Security Usable. Tech Report CMU-CS-04-135 (Ph.D. dissertation), School of Computer Science, Carnegie Mellon University, May 2004.

[32] Wi-Fi Alliance. Wi-Fi Protected Access 2. [Online] http://www.weca.net/OpenSection/protected_access.asp?

[33] Yan, J., Blackwell, A., Anderson, R., and Grant, A.: The Memorability and Security of Passwords – Some Empirical Results. Tech. Report UCAM-CL-TR-500, University of Cambridge Computer Laboratory, Sept. 2000. [Online] http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/tr500.pdf

[34] Rescorla, E.: HTTP over TLS. RFC 2818, IETF, May 2000. [Online] ftp://ftp.rfc-editor.org/in-notes/rfc2818.txt

[35] tcpdump. [Online] http://www.tcpdump.org/

[36] Xia, H. and Brustoloni, J.: Detecting and Blocking Unauthorized Access in Wi-Fi Networks. In *Proc. Networking'2004*, IFIP, Lecture Notes in Computer Science, 3042:795-806, Springer-Verlag, May 2004. [Online] http://www.cs.pitt.edu/~jcb/papers/net2004.pdf

[37] Xia, H. and Brustoloni, J.: Virtual Prepaid Tokens for Wi-Fi Hotspot Access. In *Proc. 29th Intl. Conf. Local Computer Networks (LCN)*, IEEE, Nov. 2004, pp. 232-239. [Online] http://www.cs.pitt.edu/~jcb/papers/lcn2004.pdf

[38] Ye, E., and Smith, S.: Trusted Paths for Browsers. In *Proc. Usenix Security Symposium*, Aug. 2002. [Online] http://www.cs.dartmouth.edu/~sws/papers/usenix02.pdf

[39] Zurko, M.E. and Simon, R.: User-Centered Security. In *Proc. Workshop on New Security Paradigms*, ACM, 1996, pp. 27-33.