Visual Spoofing of SSL Protected Web Sites and Effective Countermeasures*

Andre Adelsbach, Sebastian Gajek, and Jörg Schwenk

Horst Görtz Institute for IT Security, Ruhr Universität Bochum, Germany
{andre.adelsbach, sebastian.gajek, joerg.schwenk}@nds.rub.de

Abstract. Today the standard means for secure transactions in the World Wide Web (WWW) are the SSL/TLS protocols, which provide secure (i.e., private and authentic) channels between browsers and servers. As protocols SSL/TLS are considered secure. However, SSL/TLS's protection ends at the "transport/session layer" and it is up to the application (here web browsers) to preserve the security offered by SSL/TLS.
In this paper we provide evidence that most web browsers have severe weaknesses in the browser-to-user communication (graphical user interface), which attackers can exploit to fool users about the presence of a secure SSL/TLS connection and make them disclose secrets to attackers. These attacks, known as "Visual Spoofing", imitate certain parts of the browser's user interface, pretending that users communicate securely with the desired service, while actually communicating with the attacker. Therefore, most SSL/TLS protected web applications can not be considered secure, due to deficiencies in browser's user interfaces.
Furthermore, we characterise Visual Spoofing attacks and discuss why they still affect today's WWW browsers. Finally, we introduce practical remedies, which effectively prevent these attacks and which can easily be included in current browsers or (personal) firewalls to preserve SSL/TLS's security in web applications.

1 Introduction

The recent growth of the World Wide Web (WWW) and its broad acceptance even for security critical applications, such as banking and auctions, requires that web browsers provide means to establish secure, i.e., authentic and private, communication channels to servers. Today the Secure Socket Layer (SSL) [1] protocol and its successor, the Transport Layer Security (TLS) protocol [2], are the de-facto standard for secure communication on the Internet. SSL/TLS are publicly specified and have undergone a wide peer-review, which is the reason for them, as protocols, being believed to be secure [3]. Besides a secure connection, there must be a trustworthy and reliable way to inform users about the security

* Proceedings of the 1st Information Security Practice and Experience Conference (ISPEC2005), Singapore, 11-14 April, 2005, to appear in LNCS, Springer-Verlag

properties of a connection such that users are able to distinguish a secure connection from an insecure one. In general, a connection's properties are indicated by several features of a browser's user interface, e.g., the padlock icon in the status bar or the certificate dialog. In the following we will refer to the specific features of a web browser's user interface, which visualise information about the security status of a connection, as the browser's secure connection indicators (*BSCIs*).

Attacks, which tamper with these indicators to fool the user about the connection's real status, are called *Visual Spoofing* (*VS*) attacks. These attacks exploit the flexibility of browser's user interfaces to replace real BSCIs with fake ones[1] and imitate the look and feel of a trusted web site. By simulating a secure connection to a trusted service, the victim can be tricked to disclose any secret information[2], destined for the trusted web site, to the attacker. Consequently, VS attacks are a great threat for any secure web service, such as single-sign-on services, online banking or online shops.

VS attacks on their own, hosted on the attacker's web server, are harmless in general, because a victim would hardly direct his browser to that VS page by chance. Therefore, an attacker has to direct the unwitting user (or his browser) to the site hosting his VS attack. We refer to this auxiliary step as the *mounting attack*. Mounting attacks can be categorised into those operating on the network layer and those operating on the application layer. Prominent examples of the first category are ARP, IP or DNS spoofing attacks [4], whereas e-mail spoofing and URL spoofing are examples of the latter category. Due to the large number and diversity of mounting attacks, an attacker has many effective options for initiating VS attacks.

VS attacks have been studied for several years [5–8] in the research community, but with only little impact on today's browsers. Recent studies [9, 10] show a strongly increasing number of VS attacks in the Internet, because, in contrast to buffer-overflow attacks, VS attacks do not require sophisticated expert knowledge of operating systems and low-level programming; VS is applicable by moderately experienced attackers. Therefore, there is still the necessity of discussing these attacks and, even more important, to develop practical, i.e., effective and easy to use, countermeasures.

Our paper is structured as follows: In Section 2 we briefly review the basic idea of VS attacks, discuss existing BSCIs and point out vulnerabilities of deployed web browsers that allow an attacker to trick even experienced, security aware users about the status of a SSL/TLS connection. In Section 3 we review several ways how to exploit these vulnerabilities such that all BSCIs indicating

[1] More concretely, browser's BSCIs are deactivated and fake BSCIs are rendered in the display area of the browser, such that an user can hardly (if at all) distinguish fake from real BSCIs.

[2] Examples are payment information, such as credit card numbers and authorisation information (login, PIN and TAN) of online banking systems, or complete web identities administered by single-sign-on services (e.g., Microsoft's Passport) or attribute wallets.

a trustworthy secure connection can be perfectly faked in design as well as in functionality. Then, in Section 4 we discuss mounting attacks that can be used to route users to sites hosting the actual VS attack. We review related work, especially proposed countermeasures, in Section 5 and assess their suitability and effectiveness to counter VS attacks. In Section 6 we propose effective countermeasures against VS attacks, which can be easily integrated into common deployed browsers and which allow even average, naive users to detect VS attacks without restricting user's convenience. In Section 7 we discuss two possible ways of implementing these countermeasures. A demonstrator is available online [11]. Finally, in Section 8 we summarise our results and conclude.

2 Visual Spoofing (VS)

The SSL/TLS protocol has undergone intense peer review without finding severe vulnerabilities in the latest versions. However, to achieve overall security in real world applications, it is important to carefully integrate SSL/TLS, which, among other things, comprises the way security relevant information is displayed to the user.

VS attacks exploit vulnerabilities in the presentation of security relevant information. It is important to note that VS attacks are not limited to web browsers, but can, in principle, be applied to almost any application, which offers remote access to the user interface. However, as web browsers are widely deployed, used for various types of security critical applications and specifically designed to offer remote web designers extensive control over the rendering process (unfortunately, including the browser's user interface), they are perfect targets for VS attacks.

VS attacks on web browsers exploit the rich features offered to web designers to fake those parts of a browser's user interface, which display information about the connection's security status. As a result, users believe to communicate over a secure channel with the desired web server (as indicated by the browser's user interface), while actually communicating with a rogue service or over an insecure channel.

2.1 Browser Secure Connection Indicator (BSCI)

All current web browsers provide means to inform users about the status of a SSL connection, which we will refer to as *Browser's Secure Connection Indicators* (BSCIs). BSCIs allow users to distinguish a secure connection from an insecure one by displaying information, such as the server's certified identity and the cryptographic property of the connection. As BSCIs are the browser's only means for users to get information about the security status of a connection or retrieved document, their authenticity is crucial to the overall security. Common BSCIs (here exemplarily described for Internet Explorer) are:

- The most eye-catching indicator for a SSL-protected connection is a **padlock icon**, which is displayed in the status bar if the current web page has been retrieved over an SSL connection. A double click on this icon opens the so called certificate dialog (see below).

– The **certificate dialog** displays detailed information about SSL's current status, such as the server authentication information (including server name and certification authority) and the concrete cryptographic algorithms and key-lengths being used to protect the transmission of the rendered web page. For the user it is the prime means to evaluate the web site's authenticity.
– The **location bar** can be used to manually direct the browser to a certain web page, specified by a so called Uniform Resource Locator (URL). A URLs consists of the protocol's name followed by the address of the service. The prefix "https" in an URL indicates the use of SSL and is a further hint for a secure connection. After a web page has been retrieved and rendered the location bar displays the corresponding URL. Choosing an accurate address (mainly the domain name) for a web site can strengthen the trust in a document's source.
– The **menu bar** is an "indirect" indicator, as it contains no immediately visible information about the connection's status; it rather provides features, which may disclose a VS attack, and provides access to further BSCIs: Firstly, the menu bar ("View" menu) indicates the visibility of the browser's status and address bar. It is the only indicator for the real presence (and authenticity) of these BSCIs. Secondly, it provides access to the "document source" dialog, from which an experienced user can detect a VS attack, as well as access to the "document property" dialog, which contains details about cryptographic algorithms used to protect the retrieved document (e.g., cipher suite, key length).

2.2 Technical Preconditions for VS Attacks

For being susceptible to VS attacks browsers need to fulfil following preconditions: First, the browser's user interface has to be controllable by active web languages, e.g., Visual Basic Script or JavaScript, such that a retrieved web page can deactivate any BSCI when being rendered without the browser asking the user for permission or warning him. Second, the browser must have a standardised user interface. This allows an attacker to fake the SSL indicators without special knowledge about the user's user interface, as he can easily guess the browser's look and feel as expected by the user and fake it accordingly.[3]

Most currently available browsers, such as Microsoft's Internet Explorer 6, Netscape Navigator, and Firefox, fulfil these preconditions and are, as a matter of fact, susceptible to VS attacks. To our knowledge the Opera web browser does *not* fulfil these preconditions, as it does not allow an attacker to control the user interface by means of active web languages.

3 Proof of Concept Implementations of VS Attacks

The common principle of existing VS attacks is to deactivate the browser's BSCIs and display fake ones in the browser's rendering area by using design features of standard web languages.

[3] The latter condition holds for almost any standard application. Thus, it also holds for web browsers immediately.

An early proposed way to fake all BSCIs is to include images of BSCIs in a web page (see Felten et al. [5]). However, pure image-based VS attacks lack dynamic behaviour of the faked parts (e.g., certificate dialog or menu bar). Such static implementations are easily detectable by users and are only a minor threat in practice.

In [7] Li and Yongdong describe a VS attack containing a faked certificate dialog. A click-event on the padlock icon opens a fake certificate dialog, which displays wrong authentication information, while preserving the usual behaviour (response to mouse events). However, as this attack is based on Java Applets this leads to noticeable delays in rendering the fake certificate dialog.

Our Proof of Concept Implementation Our proof of concept VS attack uses DHTML, i.e., it renders static BSCIs components with standard HTML and implements dynamic behaviour with JavaScript and Cascading Style Sheets (CSS). All these techniques are standard means in web development, available in almost any browser. We mainly focused on Microsoft's Internet Explorer 6, as it is the most widely used browser today. The VS attack opens a new browser window with deactivated BSCIs (menu, button and status bar). The fake status bar (with lock icon) is included as an image. A double click event on the lock icon of the fake status bar opens a fake certificate dialog (browser window), which contains faked certificate information. The location bar is faked by using a HTML form, because this allows us to intercept user inputs and react accordingly to simulate the standard user interface behaviour, as expected by the victim; it can be used to analyse users' surf behaviour and to redirect the victim to further spoofed web sites. The buttons in the button bar change (onFocus-event) when the user moves the mouse pointer over a button as in the standard IE user interface. Furthermore, the button's functionality (e.g., back, refresh, stop) can be simulated by binding suitable JavaScript functions to the onClick-event of the corresponding button image.

A new finding of our proof of concept VS implementation is that even the whole menu structure of the menu bar can be spoofed by means of the layer feature of dynamic CSS. This comprises the "View" menu, which allows us to trick users about the actually activated bars. It is even possible to fake other BSCIs (see Section 2.1) like the source code or cryptographic properties of the rendered page by applying the same techniques. As a figure would hardly show any differences between the faked user interface and the original user interface, we provide a demonstrator of the VS attack for Internet Explorer 6 online [12]. With this demonstrator, we prove that IE's user interface including all BSCIs, can be nearly perfectly faked. There remain two "imperfections" in our proof of concept VS attack:

- The title bar of the fake certificate dialog contains "Microsoft Internet Explorer", because it is rendered in an Internet Explorer window instead of a local operating system (Windows) dialog.
- The certificate dialog does not open if a pop-up blocker is active.

To remove these remaining "problems" an attacker may use an alternative implementation of the fake certificate dialog based Macromedia's Flash. A demonstrator is also online [12]. Furthermore, we want to stress that it is also possible to implement the whole VS attack in Flash. However, tests in our department showed that nobody, although having a strong background in IT-Security, was able to distinguish the original browser's user interface from the one of our VS demonstrator based on these imperfections. Therefore, we believe, that most users will not be able to detect such an advanced VS attack as well. In the following section we will address complementary mounting attacks.

4 Mounting Attacks

To mount a VS attack in practice, the attacker has to direct his victim to a web server hosting the actual VS attack. In this section, we review three well-known preparative *mounting attacks* and discuss their efficiency to illustrate the ease of mounting visual spoofing in practice.

E-Mail Spoofing The attacker sends an e-mail to the user, which seems to origin from a trusted company that commonly contacts their clients with standardised mails, e.g., PayPal or eBay. The mail urges the user to follow a hyperlink referring to a malicious server that hosts the actual attack. Examples can be found in [9]. This mounting attack combined with a VS attack is known as *Phishing*. E-Mail spoofing is the most popular way of mounting VS, because it does not require sophisticated technical knowledge. Spoofed e-mails are commonly sent randomly to a large number of recipients in the hope, that at least some of the recipients are customers of the specific company forged by the attacker.

URL Attacks Here, the adversary hosts the implementation of the VS attack in a web domain, which has a name similar to (and easy to confuse with) the domain name of the spoofed web site. Now the attacker regularly publishes the fake domain in search engines, or includes links to this domain on other web sites (e.g. advertisements). Examples for URL attacks are:

1. "http://www.signin.ebai.com", which is can be easily confused with the real URL "http://www.signin.ebay.com".
2. URLs such as "http://www.paypal.com@the.attacker.com" exploit a rarely used feature, which allows to include a login name in an URL by prepending a string "login@" to the address part of an URL. Therefore, this URL refers to domain "the.attacker.com" instead of www.paypal.com, which is interpreted as a login name instead of an address.

Include or Refer to VS Attacks in Third Party Sites Another way to mount VS attacks is to include VS attacks in third party sites, which are used and trusted by many users (cross-site-scripting). A recent example includes VS code in an Ebay online auction to open a fake login page, which may send login and password to an attacker [13]. This attack may even be combined with elements to fake BSCIs. An attacker may also advertise wrong URLs in the name of some trusted company (e.g., a bank) which actually refers to a VS attack instead of the company's real web site.

Network based Attacks and Man-In-The-Middle Attacks In this type of mounting attack the attacker intercepts the communication between client and server. To this end the attacker may apply techniques such as ARP or DNS spoofing to push himself between the communication of client and server. This mounting attack is very powerful, because is allows the attacker to target selected users and it does not depend on users' interaction (e.g., by following some advertised link). On the other hand, such network based mounting attacks require more technical skills than sending fake phishing emails.

We want to stress that VS attacks enable successful man-in-the-middle attacks against SSL, because a man-in-the-middle attacker may visually spoof the authentication information (including the certificate dialog) of the server, such that the user does not notice the man-in-the-middle. Furthermore, this is the reason, why server-based countermeasures against VS attacks cannot be effective.

5 Existing Countermeasures

First of all, we want to note that visual spoofing can be countered indirectly, by countering mounting attacks. However, in this paper our focus is on direct countermeasures against VS, such that we do not go into the details of preventing the mounting attacks outlined above.

Felten et al. [5] proposed to deactivate all active web languages which facilitate spoofing (e.g. JavaScript, ActiveX or Java). From today's point of view this proposal seems to be impractical, because active web languages strongly improve the service offered by web sites – in fact, most popular web sites would not work anymore if a user would disable active web languages in his browser. As the WWW gained popularity through these languages, their restriction would severely decrease the web's usability, comfort and acceptance.

In [14] the authors introduced an idea for a new web browser as a consequence of lacking long-term solutions. The authors proposed to include unspoofable features in web browsers that reveal the presence of VS attacks. More concretely, they proposed to apply synchronised random dynamic boundaries (SRDs). The idea of SRD is to distinguish authentic parts of the browser GUI from rendered content received from a server by changing the boundary colours of the real GUI pseudo-randomly and unpredictable for remote attacker. Users have to compare the changing colours of browser windows with those of a reference window. Boundaries of an original browser window will be synchronised with the reference window, while spoofed browser windows won't be correctly synchronised. This proposal has practical drawbacks: Firstly, blinking features may disturb users. Secondly, the proposed implementation seems to be weak, because it is based on the XML User Interface Language (XUL). XUL allows also remote users to change the look and feel of a browser by applying means of CSS and JavaScript, which may allow an attacker to spoof this feature as well. In [15] the author describes how this vulnerability can be exploited. Therefore, we conclude that this proposal is not suitable to protect users against VS.

Li and Wu [7] proposed to prevent that the status bar is being deactivated by active web languages.[4] We consider this to be a good first step, but it is still not sufficient to completely counter VS attacks: an attacker can simply apply for a SSL certificate and host the spoofing attack on a SSL-enabled web server. A naive user will recognise the padlock icon, stemming from the attacker's certificate, and the spoofed URL displayed in the faked address bar. Therefore, he will believe to communicate securely to the requested service. Another proposal of Li and Wu is to improve the SRD concept by defining the reference point within the window (e.g. menu bar). This is also not an effective countermeasure, as such references can be spoofed even more easily.

In [8] the authors introduce the concept of *trusted credential area* (*TCA*). TCAs are solid, unspoofable areas in browsers' user interface, which visualise the authenticity of a web site by means of extended graphical credentials (e.g. brand logos, icons, seals). Thereby, certification occurs either by a trusted third party called *Logo Certification Authority* (*LCA*) or by self-certification. To reduce the involved overhead, the authors propose an extension of the TLS protocol. Our conclusion is that the idea of visualising trusted credentials is very good, especially for naive users. A brand logo is easier to understand than a list of cryptographic parameters. However, we see disadvantages in the LCA-proposal: Firstly, it either involves significant overhead or a new variant of the TLS protocol. Secondly, it is costly for a certification authority to verify that certified logos are sufficiently distinct, such that they cannot be confused by users. This problem already exists today in the context of brand imitations. This is why we highly appreciate the idea of self-certified logos. In independent work, we developed a countermeasure similar to the approach of Herzberg and Gbara: Whereas Herzberg and Gbara propose to use self-certified logos to authenticate web sites, we propose to authenticate the browser's user interface, which significantly reduces the "certification"-load put on users. In contrast to the approach of Herzberg and Gbara, the user needs only one personal logo to authenticate the user interface of his browser (see Section 6.1). In the following section we will discuss possible effective countermeasures in more detail.

6 Effective Countermeasures

The key-observation is that VS attacks aim at the browser's user interface and its perception by (naive) users. To counter VS effectively, two complementary types of measures are required:

1. **Improve User's Security Awareness:** The first and probably most important measure is to train users and improve their security awareness. Adequate security awareness of users is the ultimate prerequisite against any kind of VS attack. As long as a user does not care about security any technical countermeasure will be ineffective.
2. **Supporting Technical Measures:** As mentioned before, advanced VS attacks against standard browsers are hard to detect even for security specialists. Therefore, it is necessary to offer users reliable technical means to detect

[4] This has recently been implemented in Windows XP's Service Pack 2.

VS. Obviously, server-based countermeasures cannot completely prevent VS attacks, because an attacker can still circumvent such countermeasures by faking the user interface accordingly. On the one hand, personalisation of web sites (e.g., with logos selected by users) may improve the complexity of VS attacks, because each VS attack has to be adapted to the respective user and may only be possible for man-in-the-middle attackers. On the other hand, however, VS attacks by man-in-the-middle attackers cannot be completely prevented by server-based remedies. Instead, effective countermeasures have to eliminate weaknesses in the user interface. The fundamental weakness enabling VS attacks is the lack of BSCI authentication.

Due to the technical focus of this paper, we concentrate on the technical means to counter VS attacks. Our emphasis is on countermeasures that do not restrict users (e.g., by deactivating widely used features such as JavaScript), because this would strongly limit their acceptance by users. However, we want to stress that the effectiveness of these measures ultimately relies on the security awareness of users. In the following Sections we introduce several complementary concepts to counter VS attacks.

6.1 Personalisation

As discussed above, VS exploits vulnerabilities in browsers' SSL integration (displaying SSL meta-information) to simulate a secure connection to a trusted server by displaying fake meta-information. A straightforward method to prevent VS attacks is to implement BSCIs as tamper-resistant components of the user interface, which cannot be deactivated or changed by (active) web languages (at least for SSL connections).

We propose to authenticate BSCIs by applying the concept of personalisation with individually chosen background bitmaps (see Figure 1), as introduced by [16] in the context of trojan horses. We believe that the use of personalised bitmaps, e.g. the picture of the user's pet or friend, is more eye-catching than the proposal of blinking boundaries, while at the same time being less annoying. Compared to the proposal of self-certified logos (see Section 5), the configuration overhead for the user is significantly reduced, because the user only has to select one background image instead of self-certifying a logo for every SSL-enabled web site for which a user wants to counter VS attacks.

This has two advantages: first, the user has a stronger relation to his browser what makes changes more eye-catching. Furthermore, it is improbable that a *remote* attacker is able to determine (and fake) this individual modification. Due to security policies of web languages, the remote attacker is unable to neither figure out the position, nor the concrete background image chosen by the user. Therefore, an attacker can only act on the assumption that his victims use the standard user interface and a personalised user interface would immediately expose this VS attack. Figure 1 shows a screenshot of our demonstrator for Internet Explorer 6.

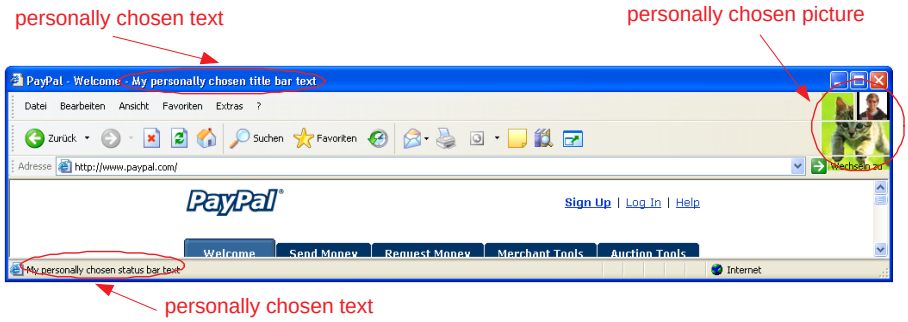

Fig. 1. Personalised User Interface. Marked regions are personalised BSCIs.

6.2 Independent Authenticated BSCIs

The idea of this concept is to introduce an additional helper application, which collects various relevant information from reliable sources and displays them in a trustworthy and authenticated way (see personalisation concept above). Our prototype implementation collects connection information (e.g., IP address of the server) from the operating system's network stack[5], and status information from the browser. The latter includes meta-information such as the status of activated bars (status bar, menu bar, etc.), URL and a brief summary of the SSL-related information. Based on this summary of security relevant information a user can verify whether the used web page is trustworthy or not. In case of a VS attack, the information displayed by the browser and the information displayed in the independent authenticated BSCI will differ and the VS attack will be noticed by the user.

6.3 Semantic and Syntactic Analyser

The main idea of this concept is to analyse the source code of a web page that is to be rendered in order to decide whether it contains VS code or not. The outcome of this analysis can be either used to warn the user, or to block suspicious content completely. However, we want to stress that the quality of syntactic and semantic analysis depends on the acquired knowledge base and that it might be hard to find suitable filtering rules in practice.

7 Implementation of Countermeasures

We propose two possible approaches of implementing our concepts. The first implementation combines all concepts in an adaptive web browser toolbar, which

[5] Note, that in case of network based mounting attacks, the information in the network stack will contain wrong information. However, this will be noticed, as this information will be different from the certificate information.

summarises all relevant information and allows the user to get this crucial information at a glance. As this toolbar is a local component of the user's system, a remote attacker cannot access it by means of active web languages.[6] The advantage of this implementation is that a user has a permanent and reliable overview about the status of his web connection. Once a user has personalised the browser's GUI (e.g. during installation), users achieve sufficient security against VS attacks. Users only have to verify the web browser's personalisation and the certificate information, which is always displayed. This is why this approach is suitable to protect even naive users. We have implemented a prototype of this toolbar as a Browser Helper Object (BHO)[7] for Microsoft's Internet Explorer Release 6. It can be downloaded at [11].[8]

A disadvantage of the toolbar described above is that its implementation depends on the underlying browser, as it must be integrated in the web browser's user interface. To overcome this drawback, we propose a proxy-based approach: a web proxy runs between the web browser and web site. Before the proxy forwards a requested web page to the browser, it embeds meta-information into the retrieved HTML-document. The web proxy may either operate on corporate gateways, e.g., as part of a firewall, or it may operate as a local software on each client, e.g., as part of a personal firewall. Its task is to summarise information about the connection's status, which is normally displayed by the BSCIs. This information may be displayed in a fixed, unspoofable HTML frame similar to an additional "status bar".

To make this idea even more practicable and secure, we propose to visualise the information as intuitive icons, e.g. a big lock icon, indicating a SSL connection. As the embedded information is encoded as HTML and rendered by the browser, care must be taken that the additional frame is resistant against manipulation by means of active web languages. This can be achieved by filtering selected instructions before forwarding the augmented web page to the browser and personalising the frame to authenticate it. The main advantage is that the HTML-encoded information can be displayed by any HTML web browser, whereas the main drawback of this proposal is that it breaks the end-to-end security of SSL/TLS-connections between the client and the server. However, as the proxy is operated by the user himself or by the company, we do not consider this to be a problem in practice. Our future work aims at analysing this approach in more detail and implementing this web proxy.

[6] At least not with reasonable security settings and without asking the user for permission. However, in these exceptional cases, no security is achievable anyway, as an attacker may completely corrupt the user's computing base.

[7] A BHO is a COM-component, which is automatically mounted during the start of the Internet Explorer application.

[8] Alternatively, one could use *Microsoft's Group Policy Editor* (*GPE*), which also allows to customise Internet Explorer's toolbars except the status bar. Thus, users are also able to personalise the browser's GUI. However, as the GPE does not provide means of personalising (authenticating) the status bar, we recommend the installation of Internet Explorer's Service Pack 2, which denies the status bar's deactivation by remote users.

8 Conclusion

In this paper, we analysed the character of Visual Spoofing (VS) attacks. Visual Spoofing exploits vulnerabilities in the web browser's integration of SSL/TLS, specifically the mechanism for indicating security relevant information, such as the use of SSL and the certified identity of the server, to users. The main reason why VS attacks succeed in today's web browsers is their ability to deactivate those parts of browser's user interface that visualise crucial meta-information and render faked ones instead.

We propose countermeasures, which authenticate those parts of the user interface that display security critical information and include an unspoofable summary of this information. Our countermeasure can be integrated directly in a web browser by means of a Browser Helper Object or by means of a proxy, which enriches retrieved HTML documents with security relevant information. These countermeasures can easily be included in existing web browsers, corporate firewalls or personal security suites.

References

1. Freier, A.O., Kariton, P., Kocher, P.C.: The SSL Protocol: Version 3.0. Internet draft, Netscape Communications (1996)
2. Dierks, T., Allen, C.: The TLS protocol version 1.0. Internet Request for Comment RFC 2246, Internet Engineering Task Force (1999) Proposed Standard.
3. Schneier, B., Wagner, D.: Analysis of the SSL 3.0 protocol. In: Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Oakland, USA, USENIX Press (1996)
4. Ornaghi, A., Valleri, M.: Man in the middle attacks Demos. In: BlackHat Conference, USA (2003)
5. Felten, E.W., Balfanz, D., Dean, D., Wallach, D.S.: Web Spoofing: An Internet Con Game. In: Proceedings of the 20th National Information Systems Security Conference, Baltimore, USA (1997)
6. Zishuang Eileen Ye, Y.Y., Smith, S.: Web Spoofing Revisited: SSL and Beyond. Technical report tr2002-417, Dartmouth PKI Lab (2002)
7. Li, T.Y., Yongdong, W.: Trust on Web Browser: Attack vs. Defense. In: Proceedings of the International Conference on Applied Cryptography and Network Security, Kunming, China (2003)
8. Herzberg, A., Gbara, A.: Protecting (even) Naive Web Users, or: Preventing Spoofing and Establishing Credentials of Web Sites. Internet draft, Bar Ilan University, Computer Science Department (2004)
9. Anti Phishing Working Group: Phishing Attack Trend Report – July (2004) http://www.antiphishing.org.
10. Litan, A.: Phishing Victims Likely Will Suffer Identity Theft Fraud. Gartner Research Note (May 14, 2004)
11. Adelsbach, A., Gajek, S., Schwenk, J.: Visual spoofing toolbar (2004) http://www.nds.rub.de/forschung/gebiete/UI/VS/download/visual spoofing toolbar.exe.
12. Adelsbach, A., Gajek, S., Schwenk, J.: Visual Spoofing Demonstrator based on DHTML (2004) To avoid misuse, please contact one of the authors for the URL.
13. Heise News Ticker: eBay konnte Passwortklau nicht verhindern (December 23, 2004) http://www.heise.de/security/news/meldung/print/54605.

14. Ye, Z.E., Smith, S.: Trusted Paths for Browsers. In: Proceedings of the 11th USENIX Security Symposium, San Francisco, USA (2002)
15. Mozilla.org: weak XUL security allows chrome UI spoofing (phishing attack) (2004) https://bugzilla.mozilla.org/show_bug.cgi?id=252198.
16. Tygar, J.D., Whitten, A.: WWW Electronic Commerce and Java Trojan Horses. In: Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Oakland, USA, USENIX Press (1996)