Why Showing one TLS Certificate is not enough? — Towards a Browser Feedback for Multiple TLS Certificate Verifications

Henrich C. Pöhls*
University of Passau, Institute of IT-Security and Security Law
hp@sec.uni-passau.de

Abstract: Content reuse on the Web 2.0 is a common "phenomenon". However, it has now reached critical and sensitive areas, as for example online shopping's submission forms for credit card data. Browsers lack the ability to show anything else than the outer most's TLS certificate verification to the user. We show that there is a trend to embed security critical content from other site's into a website. We will use VISA's credit card submission form embedded in an `<iframe>` as example. We give detailed examples of existing tentatives to solve the problem. After analyzing them, we argue that a solution can only be at the web browser's core. Finally, we postulate five steps to be taken into consideration for to evaluate and structure future solutions.

1 Introduction

Since, "Web 2.0" Internet users are aware that the content they see in their web browser's window no longer comes from one single source. It can be created by others, re-distributed through several services, re-combined by one or more web application, before finally being displayed in the web browser to be viewed and used by the user. The combination of content from different sources to provide new or improved services for the typical Internet content is routinely done in "Mashups" [Pro07]. Just recently this phenomenon, especially that of embedded content from other locations, has begone to be applied to critical areas like banking or shopping. VISA's "Verified by Visa" is the most prominent example, it is also known by the name of MasterCard SecureCode or 3-D Secure [Vis06, Vis]. "Verified by Visa" is about to reach German customers in spring 2010 when more and more banks will introduce it [com10a].

Therefore, we took this prominent example to explore and analyze the before mentioned concerns. The screenshot in Figure 1 shows how a Verified by Visa enhanced web page looks like in the Safari web browser. As the content in section (a) of Figure 1 provides input fields for sensitive data, the question "Where does this content originate from?" is now paired with "Where do my credentials end up?". Of course, a sophisticated user would use browser functions to open the embedded content in a new window.

*is funded by BMBF (FKZ:13N10966) and ANR as part of the ReSCUe IT project

Figure 1: Credit card checkout of a shop: (a) shows the "Verified by Visa" content included via `<iframe>` (b) some logos (c) social network links (d) ads; Equal output for `https` and `http`

Figure 2: The "Verified by Visa" content's `<iframe>` loaded into a window and the certificate information dialog of the Safari web browser

Indeed, the content is embedded as an <iframe>, and once the <iframe>'s content was opened in a separate browser window the result of the browser's validation result can be seen as Figure 2 shows. The browser confirms that section (a) indeed came through an encrypted and authenticated Transport Layer Security (TLS) tunnel using HTTPS as the protocol [DR06].

However, the main goal of this work is not to criticize Visa's approach, Ross Andersen and Steven J. Murdoch have already given an exhaustive analysis of the system's flaws in [MA10]. Though, we will use "Verified by Visa" as an example to demonstrate a more general shortcoming and the resulting weaknesses: Today's browsers only provide the user with the certificate verification result of the outer most TLS certificate. In more general terms, every time the user is left unaware of the true origins or has to take additional and uncomfortable measures to verify the content's validity he is not able to make an informed decision. One strategy to judge the credibility of information is to find and verify its author [FSD+03]. A user can only do this for the TLS secured tunnel that the regular and outer HTML container was delivered. The embedded content, such as pictures, JavaScript and <iframe>s, can be dynamically, still securely, fetched from different sources via their own TLS secured tunnels. Though the regarding verification results are not available for the user of the website. This absence has been neglected too long. Furthermore we discuss the issues that arise from using the embedding techniques in sensitive applications like the input of credit card data. The above described shortcoming is still not widely known and discussed, but causes security problems when applied to sensitive applications.

The paper is organized as follows, we will start with existing approaches and then classify them. We then reason about the security of the existing approaches. Based on the most promising one, we will finally close with suggestions for future browser development.

2 Security Goal: GUI-Feedback of Authenticity and Confidentiality

First of all we will define the security goals that we would like to reach when showing content from mixed sources securely in one page. The first property is origin authenticity. Following the definition from Gollmann [Gol05], *data origin authentication* will "provide the means to verify the source and integrity of a message". Hence, the content's integrity is also protected. We will further distinguish two notions for authenticity of origin: *hop-to-hop* or *end-to-end*. *End-to-end authenticity* allows the origin of the content to be verified even if it has been copied, moved, or stored elsewhere, iff the content is not modified. Thus, it is persistant. *Hop-to-hop authenticity* allows to verify the origin of the sender of the content in an ongoing transmission. So for the point-to-point transmission of content, for example from the web server to a web browser, the sending party as the source can be verified. TLS offers exactly hop-to-hop authenticity, the authentication of origin is not preserved after the connection has ended, so it can not provide protection for the content after its transfer [Kah06].

Confidentiality is the second property. A usable confidentiality protection only makes sense if the communicating partners that keep their messages confidential can be authenti-

cated (before mentioned hop-to-hop authenticity). In the web confidentiality can be guaranteed for already downloaded content, but is also important when considering future sending of data. As in the example of "Verified by Visa" 'downloaded web content can contain a HTML `form`, the filled in form fields are submitted to a target which is totally unrelated to the source. The `forms` target, as specified in the `forms`'s `action`, could be a different domain or a different protocol (`https` or `http`) . Hence, we include confidentiality assurances for future actions because they are usual in the user web browser web server interactions. So in the following we will distinguish between *downstream confidentiality* of received data and *upstream confidentiality* protecting data that is send upstream in future transmissions.

As motivated before, we need a way to inform the user about: Confidentiality protection (up- or downstream) including the hop-to-hop authenticity verification result of the communicating endpoint (up- or downstream), and the authenticity and integrity (end-to-end) verification results for protected content. This includes information about failed verifications. The information must be conveyed to the user in such a way that they are hard to spoof by a maliciously crafted web page.

3 Discussion of Existing Approaches

Turning from the security goals to existing implementations or specifications, digital signatures and related technologies are in widespread use to securely mark a content's origin and allow for its verification. The idea of signing web content outside the scope of TLS secured communication tunnels was already mentioned in 2002 in a mailing list discussion [RM02]. In the same year the work of Chi and Wu discussed signed web content for caching [CW02]. More recently, new services were created[1] that allow authors to "claim" web content, thus providing the user with means to "verify" the origin. Our own analysis [BP08], and a recent review of likewise services have shown that while these services can provide end-to-end authenticity they do not provide any confidentiality protection, neither downstream nor upstream. Thus, the existing mechanisms included in TLS secured tunnels, for hop-to-hop authenticity and down- and upstream confidentiality seem a good starting point to look for a solution. We found four existing approaches that can be used to tackle the problem of verification of content from various trusted sources within one web browser's page. We will shortly describe each of them in the following and which of our security goals they reach.

3.1 Approach I: Why SSL is not enough

In their work "Why SSL is not enough", Matthias Quasthoff, Harald Sack, and Christoph Meinel [QSM07] content signed by XML signatures was embedded into XHTML web

[1] i.e. MicroID.org, ClaimID.com, FindMeOn.com, RegisteredCommons.org, Numly.com, or DulyNoted.co.uk

pages. For verification they used an extension to the Firefox web browser using an extension that shows the signed and verified content in a separate window as shown in Figure 3. This solution circumvents attacks from a malicious web page as the information inside the extension's window is not controllable by the web content. Therefore, it provides the correct information in case the malicious web page tries to change the rendering of signed content after the verification or visually highlights unsigned content as signed. In other words, the approach is suited to verify end-to-end authenticity. However, even with the use of XML encryption, which was not considered by the authors, a satisfying up- and downstream confidentiality protection would require a previously established session key or a previous established public key, under which sensitive content could be encrypted.

3.2 Approach II: ConCert

Our own solution named ConCert, as presented for example in [PÖ8], also signs content before it is embedded into HTML web pages. Rather than XML, ConCert uses Microformats [Mic06] to mark signed content and to embed a X.509 [HPFS02] compatible content certificate. Our choice for verification also was to extended the Firefox web browser. Differently from approach I the extension displays the signed and verified content in a sidebar. The sidebar, as is the separate window of approach I, is solely controlled by the extensions or the core browser and therefore out of control of regular and potentially malicious web content. On top of this feature set, which is comparable to the approach I, it allows to highlight the verified content inside the original rendered web page. Like approach I, our own solution offers end-to-end authenticity of content and additionally allow policy-conforming removal of signed content. It was not designed to offer confidentiality protection. If one would introduce confidentiality by including encryption, again keying material is needed.

I)  II) 

Figure 3: I) Infobox [QSM07]; II) Content in sidebar and CSS highlighting [PÖ8]; both Firefox.

3.3 Approach III: Comodo's Content Verification Certificates

The comercial TLS certificate issuer Comodo offers the service to register parts of certain web pages using so called Content Verification Certificates [Com10b]. We installed the external application for testing which communicates with the browsers using a plugin. Tests were run under Firefox and Internet Explorer. If a verifiable content is under the mouse pointer the verification application draws a green border around the whole screen, as shown in the screenshot in Figure 4. This is paired, by the possibility to right click on the object and see the certificate as issued by Comodo. The user guide [Com05] especially names trademarks, logos, and login boxes as verifiable content and they further claim that thy would detect what they called "Any overlap of content (even by a single pixel)" [Com10b]. This approach, again visually highlights the verification results, though does not re-render the content, but only display a green border around the screen. This is done in addition to the browser based TLS certificate verification and the browser's graphical user interface (GUI), and it also works for `http`, so non TLS secured, content. Only on Comodo's demo page[2] we were able to see that one could right click on a verified part, i.e. the logo, and were able to view the X509 certificate that Comodo was assigning this verifiable content. Generally speaking, approach III provides additional feedback on the authenticity for certain content by visual feedback. Further detailed hints of the source are available using Internet Explorer's windows dialog for showing the X509 certificate. The browser still signals only the outer most TLS connection to the user. The solution has two major drawbacks, first technical details are not well documented and exclusively works with Comodo's special certificates. Second, it tries to solve the problem to: "show the difference between a real website and a spoofed website, keeping a website verifiably authentic"[3]. We assume it ties content to domains, so moving a logo from one domain, where certified, to another should invalidate the certificate. With the little documentation at hand, this seems to offer hop-to-hop authenticity.

3.4 Approach IV: Local Browser User Scripts

The last approach to the problem is prototype we build ourselves. It uses the greasemonkey Firefox extension [gre10], which basically allows the user to run additional scripts on page load to modify a web page's content. With it, we wrote a proof-of-concept prototype. All `iframe`s with a `https` target automatically open new tabs with the `iframe`'s URL in the new tab. Hence, these windows have a full URL and browser bar, allowing the user to verify the certificates in the usual manner, as shown in Figure 2. This approach breaks the `iframe`'s embedding and thus allows the visualization of the otherwise hidden TLS certificate verification results. However, opening new windows is not in line with Visa's best practices for deployment of the "Verified by VISA" content: Visa recommends not to use popups [Vis05]. It has some usability issues, i.e. it is unclear, whether all web applications and shopping sites that embed content via an `iframe` are still usable

[2] www.contentverification.com
[3] www.comodo.com/e-commerce/ssl-certificates/content-verification.php

Figure 4: Visual of approach III: Border when mouse is over login form [Com10b]

when data is entered into the detached browser window instead of the `iframes` or not. However, this approach is appealing for several reasons: First, it is generic and works with each and every page that uses TLS-secured content in an `iframe`. Second, it works without depending on additional signatures. Third, it allows the user to clearly separate which content came from which site, thus offering hop-to-hop authenticity and up- and downstream confidentiality. Thus, approach IV solves the problem. Last but not least, it is secure as long as Firefox's TLS verification works securely, which is assumed. The drawback: It radically breaks the unity that probably was the reason for embedding the contents in the first place.

4 Further Classification and Discussion of Results

In the following we will shortly introduce the classification properties that we developed to compare and evaluate the existing approaches. We would like those properties to become the basis of future discussions and requirement for developing future solutions. We will shortly introduce the properties and then discuss how and to which extend each approach fulfills it, wrapping it up in a discussion also relating it to other work.

4.1 Classification Properties and Classification Results

4.1.1 Method to Distinguishably Mark the Verified (or Confidentiality Protected) Content within HTML (or HTTP)

A single web page as displayed in the browser can contain several embedded secured contents. The different contents have to be marked either within the HTML code or by ways of the HTTP protocol. This must be done in such a way that the verifier can avoid ambiguities. The approaches I and II discuss or use solutions that allow them to clearly state which content is signed and which signature is attached to which content. In the case of approach I this is done through XML signatures around the XHTML contents and in approach II we described how to mark signed content and embed X.509 [HPFS02] compliant signatures using Microformats. These inline approaches can be seen as steps in the direction of the Tim Berners-Lee's idea of a "Semantic Web" [BLHL01], but the marking can also be build on standards like RDFa [W3C06]. Comodo's approach (III) is not well documented, all we found is that the verification tool "will extract the credentials of the site from the IdAuthority service" [Com10b]. How the certified parts are distinguished from uncertified ones is not documented. Finally, approach IV uses `<iframe` and `src=''https://` as markers.

4.1.2 Way of Visual Highlighting the Verification Outcome

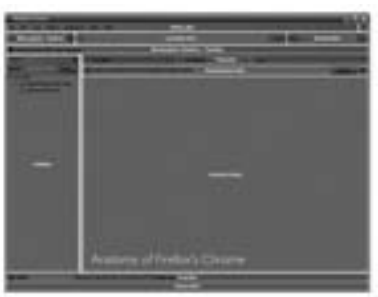
a)  b) 

Figure 5: a) Zones for visual feedback [Faa07a]; b) Markers to relate and access machine-readable content (design study) [Gla07]; both Firefox.

Browsers start to understand more and more semantic annotations that are embedded in XHTML (cp. Firefox 3.0 [Faa07b, Kap07]). Figure 5a taken from [Faa07a] shows the different zones one could possibly place GUI elements within the Firefox browser. This can be transferred to other browsers as well. Now we have seen that the approaches I and II add a copy of the content, to either the sidebar or some other part of the browser. These areas, unlike the Content Area, are not controllable by potentially malicious content. The approach IV also duplicates the content in the new window and leaves the existing certificate verification result notification as it is and thus outside the Content Area. Only approach III, implemented by Comodo, uses the mouse over event to interactively show

the user that the element under his mouse pointer has been verified by Comodo.

4.1.3 Browser Extension or Standalone

All four approaches need to extend the Browser using extensions or plug-ins, and most of them will need some additional components. While approach I uses Java, the second approach needed openssl [Ope06], and the third approach needed Comodo's Verification Engine to run in the background. While these approaches needed external applications, the last approach needed the greasemonkey browser extension. We assume in all cases that all the browser's extensions are trusted, thus an additional trusted extensions output can not be manipulated by the web site displayed. The same holds true for the browser in general and also assume this for the standalone application.

4.1.4 Academic or Commercial

The last classification distinguishes between commercial approaches and academic ones. To start with approach III, Comodo's is the only commercial approach. This is also the only approach that requires a content owner to pay for an additional content verification certificate from Comodo. However, it is the only one with a working interface (at least for Windows and Internet Explorer) which is not beta or research prototype, and has some certified real world content. For example, the Google logo on Google's index page or certain banks' logos such as the PayPal logo. However, it is unclear how many user's are actually using the verification tool and how many contents are verified today. The demoed pages do not contain secured content from multiple sources.

4.2 Discussion and Related Work

We showed that the idea of certified, signed or confidential content from different sources embedded into and alongside one and another is not a new concept. It has been discussed in one or another direction for the last 5 to 8 years. However, we pointed out that today's browsers lack support to securely use `https` to verifiably secure `iframes`. Without the help of additional tools or extensions their certificates are not accessible to the regular user of those web sites.

Apart from the four approaches presented in this paper in detail, it is important to mention two streams of work done in the field of web browsers. We will shortly look at one other extensions for the browser that deal with visually highlight privacy policy verification outcome. PrivacyBird [CMU02] is an existing extension for Internet Explorer. Secondly, we will discuss two design studies [Faa07b, Gla07] for Firefox. This helps to better understand the property, as they deal with the same problem of visual highlighting. The PrivacyBird [CMU02] extension for example. PrivacyBird automatically parses and then judges a web site's privacy policy. The policy must be expressed using the machine-readable P3P language [W3C02], a W3C standard to describe privacy policies. It is a security relevant

extension that adds a pictogram of a bird into the Status Bar (of Internet Explorer). If it displays a green bird with an extra red exclamation point this indicates that the site generally matches a users preferences but contains embedded content that does not match or does not have a privacy policy. A user study [CGA06] showed that this constant feedback is considered helpful and that users liked to be always informed. This kind of visual information bears one basic problem, the above mentioned "green bird with exclamation" does not need to be linked to a specific content. In other words, PrivacyBird's verification result is for a server or service, so its fine that the visual covers the actual domain the browser is showing. The problems start if parts of the shown web site come from different servers and have different privacy policies, PrivacyBird does not cater for this. Providing a link between verification outcome and verified content must be done most unobtrusively, but still securely and hard to spoof. This exact problem, is still left unresolved by browser vendors when regarding the security error messages related to non TLS content. For example, the browser displays an error message when a website, which is delivered through a TLS tunnel, tries to embed content, such as an image, from a non-https-secured source. This error message is not content specific, but generic. It does not tell the user which content and thus which part of the website is touched by the decision at hand.

The second example discusses several design studies that show how data, for example street adresses, shall be highlighted so that the user could activate other applications using the detected data. Kaply's Firefox extension Operator [Kap07] gives a good starting point, while Alex Faaborg's design studies [Faa07b], as mentioned above, show how hard visual highlighting (of not security critical) information already is. Another concept is given by Dimitri Glazkov [Gla07] is shown in Figure 5b. He provides markers on the left side of the browser right next to the Content Area, at the same hight as the regarding content. It should be pointed out, that both ideas are not considering security issues, like spoofing. This is rather concerning, but might change if use cases like the "Verified by Visa" are used broadly and highlight the need for a secure and usable solution. We postulate that such a security relevant functionality must be offered as a core browser functionality. We have learned from our own Firefox extension prototyping for other research projects that it is otherwise too cumbersome or even impossible to implement functions like certificate checking or certificate trust management in an extension. So, even though management is already done in the browser for the X509 certificates used to authenticate also the "hidden" TLS tunnels, additional components can not easily use them. We hope that concepts like the Operator Extension can be integrated and extended to the needed security functionality.

5 Conclusion and Future Work

Soon content reuse, already common for non-security sensitive content on the Web 2.0, will reach out into critical and sensitive areas, like credit card details during online shopping. Browsers still lack the ability to show the user anything else than the outer most's TLS certificate verification result. We argue that, based on the analysis of the detailed examples and existing tentatives to solve the problem, this must be done at the web browser's core. No solutions yet exist within the browsers themselves. We postulate that each

TLS session's certificate and its verification result shall be made differentiable and visible within the browser to allow users to verify the content that originated from that TLS session. Following our classification, we propose the following steps to reach the goal:
- Standardize the marking for content parts that are integer and authentic
- Provide visuals that one or several secured contents are within the actual content area.
- Highlight secured content visually within the content area; must be hard to spoof and visually unobtrusive.
- Integrate into the browser: no extensions, not external standalone.
- Provide an open solution: allowing public screening and academic or comercial development of new services.

This article hopes to further raise the awareness for this problem in web browser security and to stimulate the future integration of the above mentioned points into the development of web browsers and web content creation and management systems.

References

[BLHL01] T. Berners-Lee, J. Handler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[BP08] Bastian Braun and Henrich C. Pöhls. Authenticity: The missing link in the social semantic web. In *Digitale Soziale Netze - GI 2008*, 2008.

[CGA06] Lorrie Faith Cranor, Praveen Guduru, and Manjula Arjula. User interfaces for privacy agents. *ACM Trans. Comput.-Hum. Interact.*, 13(2):135–178, 2006.

[CMU02] CMU Usable Privacy and Security Laboratory at Carnegie Mellon University. Privacy Bird. `www.privacybird.org/`, 2002.

[Com05] Comodo Inc. COMODO Verification Engine User Guide. `www.vengine.com/pdfs/userguide.pdf`, 2005.

[com10a] comdirect. Ab Ende April kaufen Sie noch sicherer online ein. `http://www.comdirect.de/pbl/cms/cms/services/pages/s2/sicherheit/was_leisten/verified_by_visa/5663_se_verified_by_visa.html`, Mar. 2010.

[Com10b] Comodo Inc. What are CVCs? - Content Verification Certificates. `www.instantssl.com/ssl-certificate-products/content-verification.html`, 2010.

[CW02] C.-H. Chi and Y. Wu. An XML-Based Data Integrity Service Model for Web Intermediaries. In *Workshop on Web Content Caching and Distribution (WCW)*, 2002.

[DR06] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Updated by RFC 4366.

[Faa07a] Alex Faaborg. Anatomy of Firefox's Chrome. `http://people.mozilla.com/~faaborg/files/20070204-detectionUI/anatomyChrome.jpg_large.jpg`, 2007.

[Faa07b] Alex Faaborg. The User Interface of Microformat Detection. `blog.mozilla.com/faaborg/2007/02/04/`, February 2007.

[FSD+03] B. J. Fogg, Cathy Soohoo, David R. Danielson, Leslie Marable, Julianne Stanford, and Ellen R. Tauber. How do users evaluate the credibility of Web sites?: a study with over 2,500 participants. In *DUX '03: Proceedings of the 2003 conference on Designing for user experiences*, pages 1–15, New York, NY, USA, 2003. ACM Press.

[Gla07] Dimitri Glazkov. Margin Marks UI Concept. `http://glazkov.com/2007/09/04/margin-marks/`, Sept. 2007.

[Gol05] D. Gollmann. *Computer Security 2e*. John Wiley & Sons, 2005.

[gre10] Greasemonkey Firefox Extension. `www.greasespot.net`, Feb. 2010.

[HPFS02] R. Housley, W. Polk, W. Ford, and D. Solo. RFC 3280 - Internet X.509 PKI Certificate and Certificate Revocation List (CRL) Profile, Apr. 2002.

[Kah06] Usability and document authentication issues. `www.w3.org/2005/Security/usability-ws/papers/12-kahan-usability-and-doc-auth/`, March 2006.

[Kap07] Mike Kaply. Operator. `www.kaply.com/weblog/operator/`, Oct. 2007.

[MA10] Steven J. Murdoch and Ross Anderson. Verifed by Visa and MasterCard SecureCode: or, How Not to Design Authentication. In *Financial Cryptography and Data Security '10*, Jan. 2010.

[Mic06] Microformats. website. `www.microformats.org`, Aug. 2006.

[Ope06] OpenSSL Project. *OpenSSL*. OpenSSL, Dec 2006.

[PÖ8] H. C. Pöhls. ConCert: Content Revocation using Certificates. In *Sicherheit 2008*, volume 128 of *GI-Edition Lecture Notes in Informatics (LNI)*, pages 149–162, Saarbrücken, Germany, April 2008. GI.

[Pro07] ProgrammableWeb.com. Mashup Matrix. `programmableweb.com/matrix`, Aug 2007.

[QSM07] M. Quasthoff, H. Sack, and Ch. Meinel. Why HTTPS is Not Enough – A Signature-Based Architecture for Trusted Content on the Social Web. In *IEEE / WIC /ACM Int. Conf. on Web Intelligence*, 2007.

[RM02] Doug Ransom and Peter V. Mikhalenko. Discussion on "XHTML adoption curve". `lists.xml.org/archives/xml-dev/200204/msg00559.html`, Apr. 2002.

[Vis] Visa USA. Verified by Visa. `https://usa.visa.com/personal/security/vbv/index.html`.

[Vis05] Visa EUROPE. Verified by Visa: Merchant Deployment Best Practices Factsheet. `www.visaeurope.com/documents/vbv/verifiedbyvisa_merchantdeploymentbestpractices.pdf`, 2005.

[Vis06] Visa USA. Verified by Visa System Overview External Version 1.0.2. `https://partnernetwork.visa.com/vpn/global/retrieve_document.do?documentRetrievalId=119`, Dec. 2006.

[W3C02] W3C. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. `www.w3.org/TR/P3P/`, Apr. 2002.

[W3C06] W3C. RDFa Primer. `www.w3.org/TR/xhtml-rdfa-primer`, May 2006.