A Case for Avoiding Security-Enhanced HTTP Tools to Improve Security for Web-Based Applications

A Position Paper by Bradley Wood [1]

Abstract

This paper describes some of the general weaknesses of the current popular Hypertext Transmission Protocol (HTTP) security standards and products in an effort to show that these standards are not appealing for many applications. We will then show how we can treat HTTP browsers and servers as untrusted elements in our network so that we can rely on other mechanisms to achieve better overall security than can be attained through today's security-enhanced HTTP tools.

Introduction

The World Wide Web (WWW or the Web) has become the new popular computing paradigm for applications developers and decision makers. It is becoming increasingly popular to develop new applications and networks based on this model. There is also a lot of interest in migrating legacy applications to a Web-based infrastructure.

Unfortunately, we are finding that HTTP is not well suited to applications that have even the most basic security requirements. This limits the usefulness of the Web to applications that have few real security requirements. More importantly, decision makers are opting to develop Web-based applications to gain increased functionality at an admitted loss of security and control. Therefore, it is important that we develop the tools and techniques needed to satisfy our basic security requirements in a Web-based infrastructure.

A lot has been written in even the popular press about coming advances that promise secure WWW applications. Unfortunately, most of the current and emerging products and standards for adding security to HTTP are lacking; and, it is not clear that we will see satisfactory advancements in the foreseeable future. Therefore, applications developers are left to their own devices to create security-enhanced HTTP applications using currently available techniques and technologies.

In this discussion, we will examine some of the general weaknesses in the current security-enhanced HTTP products and standards. As an alternative, we will review techniques for satisfying security requirements without using any of the HTTP security enhancements.

Current Standards and Products

There has been a lot of activity recently in the area of security products and standards for the WWW. Two different standards are evolving as the dominant choices for adding HTTP security: the Secure Sockets Layer (SSL), and Secure HTTP (S-HTTP). It has also been reported that there are efforts underway to integrate SSL and S-HTTP into a universally accepted Web security solution[1]. Unfortunately, today's reality is quite different than some of the promised results.

One could argue that Web developers are faced with a difficult choice for adding standards-based security to their Web-based applications. One solution, epitomized by the *Netscape Navigator* and *Netscape Commerce Server,* provides relatively little security in favor of improved overall application robustness and a rich set of features. The other standard, S-HTTP, offers a robust set of security features on browsers and servers that are generally not as robust or full-featured as the Netscape product family.

---

[1] Bradley Wood is a Senior Member of Technical Staff in the Data Systems Security Department of Sandia National Laboratories, Albuquerque, NM 87185-0451. He can be reached by phone or fax at (505) 845-8461 or by electronic mail as Brad.Wood@Sandia.gov.

This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000.

Secure Sockets Layer

SSL[2] was originally developed by Netscape Communications [3] to enhance a Web browser and server to reduce the risks of exchanging sensitive information. The primary application for SSL is to allow a consumer to use a Web browser to purchase products and services using a credit card number for payment information. In this model, it is important to protect client information (like the credit card number) during the transaction. SSL is currently implemented in the *Netscape Commerce Server* [4] and the *Netscape Navigator* browser, as well as other products.

The primary security service offered by the *Netscape Commerce Server* is to establish a private (encrypted) communications channel between a server and a browser. This allows strangers to exchange information privately. This is useful if you are a merchant collecting orders from a variety of buyers on the Web, but it appears to have few other applications.

The *Netscape Commerce Server* also provides a relatively-strong mechanism for authenticating servers to browser users, provided the client checks the server certificate when a secure session is established, and provided that the server certificate is genuine. Client authentication is currently limited to a username / password technique.

Although there are relatively few advanced security features in the *Netscape Commerce Server*, there is still a lot of interest in using the Netscape product family. Some of the perceived Netscape advantages include:

- **Simple Key Management** - Server certificates are validated using public signature keys that are embedded in the *Netscape Navigator*. Browser users are not required to do anything to enable the SSL features in the browser. Therefore, every Netscape browser comes with these basic security features already enabled and ready to use.
- **Rich Feature Set** - The *Netscape Commerce Server* has an applications programming interface (API) and other features that allow content providers to create attractive and feature-rich Web sites. In a competitive environment, content providers are eager to leverage any feature that will distinguish their service among the multitude of sites on the Web.
- **Widely Distributed Browser** - The *Netscape Navigator* has been distributed as shareware, so it is readily available to anyone with even casual access to the Internet. Still, this browser is widely touted as being one of the most stable and feature-rich Web browsers in the industry. There are versions of the *Netscape Navigator* available for most major computing platforms including *Microsoft Windows*, Apple's *Macintosh*, and X-Windows under many different versions of UNIX. As a result, many industry sources report that the *Navigator* is the dominate Web browser on the market.

We are seeing a lot of interest in modifying the *Netscape Commerce Server* and *Netscape Navigator* to provide strong authentication of the browser user to the server. Some of these enhancements leverage Kerberos, DCE, and one-time password technologies.

We are also seeing a distressing number of successful attacks against Netscape's implementation of SSL and other security features in both the server and the browser [5] . This appears to be a logical result of the enormous pressure that the market has placed on Netscape to add features to their products as quickly as possible. Although Netscape has entered into an agreement with RSA Data Security to review their security implementations in the future, it is not clear that the market will ever demand fastidious security implementations at the expense of longer product or feature development cycle times.

Secure HTTP

Secure-HTTP [6] (S-HTTP) is the other major standard proposed for Web-based security enhancements. S-HTTP was originally developed by a team at CommerceNet and Enterprise Integration Technologies (EIT) [7] to provide a robust set of security services for a variety of applications, particularly robust commercial electronic commerce over the Internet using a Web-based infrastructure.

The primary strength of the S-HTTP specification is that it characterizes a rich set of robust, negotiable security features. S-HTTP has the potential to satisfy a variety of security requirements for both clients and servers using sophisticated cryptographic techniques. Indeed, S-HTTP could potentially solve most common Web-based security requirements.

Unfortunately, S-HTTP is not as widely deployed as SSL. Although we have had tool kits and prototype implementations for some time, there are relatively few production-quality products and applications using S-HTTP, and the S-HTTP community appears to be evolving more slowly than other product families (such as Netscape's).

In addition, the security features in an S-HTTP application must be fastidiously designed and implemented. There is often a complicated client enrollment process that must be performed in advance of establishing an S-HTTP session between a browser and a server. Most of these enrollment processes involve cryptographic key management and registration tasks.

We are also somewhat distressed by the poor quality of some of the products that implement S-HTTP. Many S-HTTP browsers and servers are built upon shareware or public domain products that themselves have some significant security problems. We have also noticed that most of the browsers that implement S-HTTP do not offer the features and overall robustness of the *Netscape Navigator*. There appears to be relatively little interest in widespread adoption of any S-HTTP browser in favor of the Netscape browser.

Other Approaches

We have also seen other approaches for adding security to a Web-based infrastructure that are not widely implemented but often mentioned in the some of the popular literature.

DCE - Proposals have been made to process standard HTTP transactions over an infrastructure that uses Distributed Computing Environment (DCE) security services [8]. Here, client workstations and servers use DCE security services to establish a trusted session or channel where standard HTTP transactions are supported. Although this approach requires that the user invest in a relatively-expensive DCE infrastructure, this approach may be appeal to enterprises that have already invested in DCE and who only need security enhancements for applications that run over their current DCE infrastructure. Another advantage of this approach is that you can use robust DCE security services without major modifications to the HTTP browser or server.

Kerberos - Enterprises that already use Kerberos security services are seeking to leverage that investment to improve their Web-based applications. We have seen some work at facilities like Sandia National Laboratories (New Mexico) where Kerberos is being integrated into the *Netscape Commerce Server* to provide strong user authentication. This approach uses an unmodified Netscape browser to securely pass a Kerberos username and password to the *Commerce Server* using SSL. The server then performs the Kerberos initialization function to verify the identity of the browser user and to obtain the access privileges (or tickets) for that user.

An Alternative Approach

A Question of Trust

An alternative approach to satisfying security requirements in a Web-based application is to simply treat the HTTP browser and server as untrusted elements in the computing network. We will introduce this approach by contrasting it with the approach that relies on the satisfying their security requirements using Web-based products or services.

Web Tools a Trusted Elements

In this approach, we want to rely on features in the Web browser and server to satisfy our security requirements. This approach is illustrated in Figure 1.

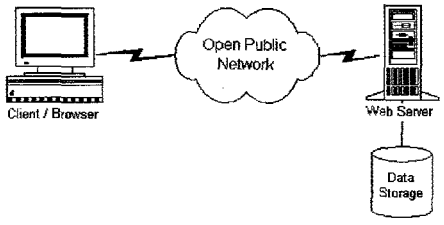

Figure 1 - Web Products as Trusted Network Elements

This approach is fairly common, and it is characterized by the following features:

- We rely on the Web browser and server to cooperatively authenticate each other and determine the identity of the browser user or client for the server.
- We rely on the Web browser and server to cooperatively protect the data exchanged over the open public network.
- We rely on the Web server software or operating system to enforce access controls to the stored data base.

This approach is popular, primarily because this model can be developed with a minimum investment in hardware, software, planning, and training. It leverages the advertised security features of the Web-based products.

Unfortunately, there are several potential problems with this approach:

a) **Server Processes with Vulnerabilities** - The common expectation is that if you install a Web server software package on a respectable computer, you will have a full-feature production Web site. In reality, to get most of the desired features, administrators must install a variety of network server processes on their computer. Some of these extra required server processes might provide file transfer, electronic mail, or database management services. Although this is technologically feasible, each server process has its own potential security weaknesses that an adversary could exploit to gain unauthorized access to the stored data. Therefore, the more products that we install on a single server, the more likely that server will become vulnerable or compromised.

b) **Access Controls** - It is not clear that we can trust the Web server software to provide fastidious access controls to the stored data. What we are seeing is that the current Web servers provide either few access control features, or they rely completely on the server operating system for data access control. Therefore, this approach may not be suitable for applications that need rigorous access control.

c) **Weak Authentication** - We are seeing many implementations that rely simply on traditional username / password pairs for authenticating parties. This is widely regarded as a weak technique. However, most Web servers do not offer stronger or more sophisticated authentication services. In addition, it is difficult to do rigorous access control based on weak authentication.

d) **Catastrophic Compromise** - In the likelihood that the single server is compromised through any number of common attacks, the entire information system is compromised. This event could be catastrophic if the stored data is sensitive in any way.

e) **Exposure** - Web servers are generally installed outside a traditional firewall or other security gateway. This is necessary because most Web functions are hindered by a firewall, proxy server, or other technique; and, the primary requirement for most Web servers is availability. Unfortunately, this makes the server highly vulnerable to a variety of potentially sophisticated adversaries.

Web Tools as Untrusted Elements

In the alternative approach, we treat the Web products as untrusted computing elements; and, we do not rely on these products to enforce our security policy. An example of this approach is shown in Figure 2.

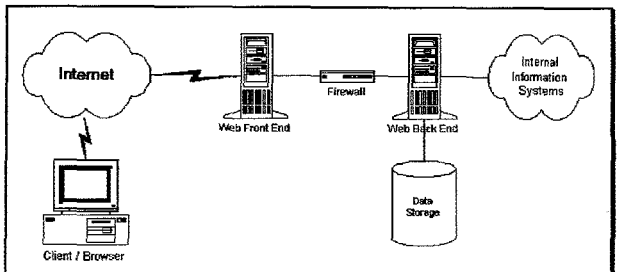

Figure 2 - Web Products as Untrusted Network Elements

This approach is characterized by the following features:

- There is a clear boundary between internal information systems or servers and external resources. The boundary is typically a firewall, proxy server, or some technique to limit the exposure of the internal network.
- The external Web server or Front End is used simply as a user interface. Most of the actual information processing is done on resources in the internal network.
- Access controls and other security requirements are usually satisfied by using database management systems or other products with rigorous access controls.
- Authentication is done between the actual browser user (the client) and internal information systems. The external Web Front End does not participate in the actual authentication process.

There are some distinct advantages to this technical approach:

a) Since the Web Front End is used as strictly a user interface, you can expect better performance than a system that must support many server process. In addition, this Web Front End can be optimized for its unique role.

b) This approach gives the network designer the ability to integrate a variety of well understood or mature security techniques into a Web-based infrastructure, such as security-enhanced messaging.

c) This technique also allows the designer to integrate traditional or legacy information systems such as database management systems into a Web-based infrastructure.

Unfortunately, this approach has one primary weakness. This approach is generally more complex than the traditional approach, leading to increased expense to procure and manage. This approach also requires that the information system be designed by experienced information systems security professionals.

Applications-Layer Security Protocols

We are also seeing a lot of activity in the WWW community on applications-layer security protocols. These protocols are really designed to work on top of or independent of a particular Web browser or server. Examples of these protocols include:

- Secure Transaction Technology (STT) developed by Visa and Microsoft [9]
- *Secure Courier* developed by MasterCard and Netscape [10]
- Secure Electronic Payment Protocol (SEPP) developed by IBM and others [11]

These protocols [12] all provide an independent means for developing strong security techniques at the Applications Layer. Therefore, these protocols could be added to Web browsers, servers, back end systems, and electronic messaging systems.

The point is that the WWW community has identified the benefit of moving their security mechanisms outside the WWW products, and this is just another approach to satisfying the security requirements for a family of applications.

New Developments

The market for advanced HTTP products is responding with new products at an amazing pace. Users are demanding improved HTTP security, and some vendors are responding with announcements of improved security features in their future products. For example, Netscape Communications has made several new product announcements.

- Netscape announced that a new version of their browser, *the Netscape Navigator v2.0,* will be generally available in January 1996 [13]. One intriguing feature of this product is the addition of a client-side digital certificate for public key applications. However, it is not clear how a user would actually take advantage of this digital certificate. We speculate that the primary purpose of this certificate is to support the security-enhanced messaging features that have also been added to the v2.0 browser. Ideally, we would like to use this certificate to provide a strong client-side authentication to the *Netscape Commerce Server*. However, it is not clear that this will be supported in the v2.0 browser.
- Netscape has also announced that they plan to release a new version of the *Netscape Commerce Server* in the first quarter of the 1996 calendar year [14]. Unfortunately, we have seen no information on what security enhancements might be in this server.
- Netscape has also announced that they plan to develop a family of security-enhanced Web products that incorporate the National Security Agency's FORTEZZA technology [15]. Current plans call for this product to be available sometime in late 1996, and it is not clear what features will actually be supported in any of the components.

Summary

There is a great deal of interest in adding security features to HTTP- or Web-based applications. Unfortunately, it is not clear that we can satisfy even our most basic security requirements with current security-enhanced HTTP products. There is no indication that planned product enhancements will fully rectify this situation.

Therefore, it is up to the applications developers to satisfy their security requirements using current technology. One technique that appears to satisfy this goal is to treat the WWW components in a network as untrusted elements, and use traditional techniques to enforce the security policy. This approach leads to networks that can be complex and expensive, but it appears to be the only way to implement a reasonable security policy on a Web-based infrastructure.

References

The majority of these references are Uniform Resource Listings (URL)s to hypertext documents on the World Wide Web.

[1] Press release on Terisa Systems Partnership: *http://dengue.terisa.com:80/new/pr/ 041095b.html*

[2] References on Secure Sockets Layer: *http://home.netscape.com/newsref/pr/ newsrelease17.html, http://home.netscape.com/newsref/std /sslref.html*

[3] Netscape's home page on the Word Wide Web: *http://home.netscape.com*

[4] *Netscape Commerce Server Reference Guide* and *Netscape Commerce Server Programming Guide*, 1995, Netscape Communications, Inc.

[5] References on attacks and vulnerabilities in Netscape products: *http://home.netscape.com/newsref/pr/ news-release68.html, http://home.netscape.com/newsref/pr/ newsrelease46.html, http://home.netscape.com/newsref/ std/random_seed_security.html; http://www.openmarket.com/press/ nssecurity.html*

[6] Secure HTTP specification: *http://www.eit.com:80/creations/s-http/; ftp://ds.internic.net/internet-drafts/ draft-ietf-wts-shttp-01.txt*

[7] Enterprise Integration Technologies home page on the World Wide Web: *http://www.eit.com/*

[8] Information on the Open Systems Foundation's DCE-Web project: *http://www.osf.org/www/dceweb/ index.html*

[9] References on the Secure Transaction Technology (STT) specification: *http://www.visa.com/cgi-bin/ vee/sf/commerce/ sttdownloads.html?2+0, http://www.windows.microsoft.com/windows/ie/stt.htm*

[10] References to the Secure Courier specification: *http://home.netscape.com/newsref/pr/ newsrelease33.html, http://home.netscape.com/newsref/pr/ newsrelease10.html, http://home.netscape.com/newsref/std/ credit.html*

[11] Reference on the Secure Electronic Payment Protocol: *http://www.mastercard.com/Sepp/sepptoc.htm*

[12] Announcement of intent to merge payment standards: *http://www.mastercard.com/Press/release-960201.htm*

[13] Information on *Netscape Navigator v2.0* features: *http://home.netscape.com/newsref/pr/ newsrelease43.html, http://home.netscape.com/newsref/pr/ newsrelease82.html*

[14] Information on new version of the *Netscape Commerce Server*: *http://home.netscape.com/newsref/pr/ newsrelease43.html*

[15] Netscape FORTEZZA announcement: *http://home.netscape.com/newsref/pr/ newsrelease49.html*