INTERNET-DRAFT
<draft-ietf-tls-http-upgrade-00.txt>
Expires May 1999
November 16, 1998

Rohit Khare
UC Irvine

Upgrading to TLS Within HTTP/1.1

Status of this Memo

Abstract

This memo proposes a mechanism to upgrade HTTP/1.1 connections to use Transport Layer Security (TLS). Using an Upgrade: TLS/x.y request header would allow unsecured and secured traffic to share the same port (in this case, 80). A companion document describes the current practice of using a separate port for HTTP over TLS, <draft-ietf-tls-https-01.txt>.

0. Motivation

At the Washington DC IETF meeting in December 1997, the Applications Area Directors indicated they would like to see a mechanism for applying Transport Layer Security [TLS] within an HTTP connection, at the same port, instead of only being able to recommend a distinct port (443) and scheme (https). IANA has already issued ten new ports for application X over TLS/SSL to date.

The TLS working group has moved forward with an extensive draft on properly implementing https (draft-ietf-tls-https-00), but there is alternate precedent for "securing" a regularly opened connection for SMTP and other applications (draft-hoffman-smtp-ssl, draft-newman-tls-imappop-03, murray-auth-ftp-ssl-00, draft-ietf-ldap-ext-ldapv3-TLS-00.txt).

There has already been extensive debate on the http-wg , ietf-tls and ietf-apps-tls mailing lists about the advisability of permitting optional 'upgrades' to secure connections within the same channel, primarily focusing on the thread of man-in-the-middle attacks. Our intent here is not to engage in this debate, but merely to document a proposed mechanism for doing either with HTTP. Several applications being built upon HTTP might use this mechanism, such as the Internet Printing Protocol; we look to them for implementation guidance.

1. Introduction

TLS, a/k/a SSL (Secure Sockets Layer) establishes a private end-to-end connection, optionally including strong mutual authentication, using a variety of cryptosystems. Initially, a handshake phase uses three subprotocols to set up a record layer, authenticate endpoints, set parameters, as well as report errors. Then, there is an ongoing layered record protocol that handles encryption, compression, and reassembly for the remainder of the connection. The latter is intended to be completely transparent. For example, there is no dependency between TLS's record markers and or certificates and HTTP/1.1's chunked encoding or authentication.

The need to 'secure' running connections is not merely 'running SSL over port 80', an early challenge for firewall developers answered by Ari Luotonen's ssl-tunneling-02 draft in 1995 -- that scheme still requires a distinct port number to activate TLS.

The HTTP/1.1 spec reserves CONNECT for future use, deferring to the more recent draft-luotonen-web-proxy-tunneling-00 proposal. This technique perpetuates the concept that security is indicated by a magic port number -- CONNECT establishes a generic TCP tunnel, so port number is the only way to specify the layering of TLS with HTTP (https) or with NTTP (snews).

Instead, the preferred mechanism to initiate and insert TLS in an HTTP/1.1 session should be the Upgrade: header, as defined in section 14.42 of rev-03. Ideally, TLS-capable clients should add "Upgrade: TLS/1.0" to their initial request, and TLS-capable servers may reply with "101 Switching Protocol", complete the handshake, and continue with the "normal" response to the original request. However, the specification quoth:

> "The Upgrade header field only applies to switching
> application-layer protocols upon the existing transport-layer
> connection."

Aside from this minor semantic difference -- invoking TLS indeed changes the existing transport-layer connection -- this is an ideal application of Upgrade. This technique overlays the TLS-request on an HTTP method; requires client-initiation, and allows servers to choose whether or not to make the switch. Like the other examples of TLS-enabled application protocols, the original session is preserved across the TLS handshake; secured communications resumes with a servers' reply.

The potential for a man-in-the-middle attack (wherein the "TLS/1.0" upgrade token is stripped out) is precisely the same as for mixed http/https use:

1. Removing the token is similar to rewriting web pages to change https:// links to http:// links.
2. The risk is only present if the server is willing to vend that information over an insecure channel in the first place
3. If the client knows for a fact that a server is TLS-compliant, it can insist on it by only connecting as https:// or by only sending an upgrade request on a no-op method like OPTIONS.

Furthermore, for clients which do not actively try to invoke TLS, servers can use Upgrade: to advertise TLS compliance, too. Since TLS-compliance should be considered a feature of the server and not the resource at hand, it should be sufficient to send it once, and let clients cache that fact.

2. Potential Solution

Define "TLS/x.y" as a reference to the TLS specification (draft-ietf-tls-protocol-03), with x and y bound to its major and minor version numbers. Section 6.2.1 of the current draft explains why the TLS version would currently be defined as 1.0, not the actual parameters on the wire (which is "3.1" for backwards compatibility with SSL3).

An HTTP client may initiate an upgrade by sending "TLS/x.y" as one of the field-values of the Upgrade: header. The origin-server MAY respond with "101 Switching Protocols"; if so it MUST include the header "Upgrade: TLS/x.y" to indicate what it is switching to.

Servers which can upgrade to TLS MAY include the header "TLS/x.y" in an Upgrade response header to inform the client; servers SHOULD include such indication in response to any OPTIONS request.

Similarly, servers MAY require clients to switch to TLS first by responding with a new error code "418: Upgrade Required", which MUST specify the protocol to be supported. @@ This is a change to 'core' HTTP; if, processwise, it's too difficult to slip in a general-purpose error code, we may have to fall-back to "418: TLS Required".

Upgrade is a hop-by-hop header (Section 13.5.1), so each intervening proxy which supports TLS MUST also request the same version of TLS/x.y on its subsequent request. Furthermore, any caching proxy which supports TLS MUST NOT reply from its cache when TLS/x.y has been requested (although clients are still recommended to explicitly include "Cache-control: no-cache").

Note: proxy servers may be able to request or initiate a TLS-secured connection, e.g. the outgoing or incoming firewall of a trusted subnetwork.

3. Next Steps

While there is formal interest in promulgating a scheme for HTTP/TLS without allocating a new port number, implementations have been scarce. We cannot predict what might trigger adoption of this proposal.

Note: The Mandatory extension scheme for HTTP is another mechanism, though arguably less aprropriate, since TLS does not modify the semantics of HTTP itself. TLS would be using Upgrade for its stated purpose -- to switch to an entirely different protocol.

This document is available at http://www.ics.uci.edu/~rohit/http-tls.

3.1 Open Issues

There have been some questions about how to continue to resolve https: URLs with the scheme postulated here. There is a default assumption in many products that https and http:443 are equivalent.

Similarly, when resolving a mixture of secured and unsecured URLs from the same site, some might postulate the need to "downgrade" the connection. We suggest simply reopening the HTTP connection without TLS.

4. Acknowledgments

Thanks to Paul Hoffman for his work on the STARTTLS command extension for ESMTP. Thanks to Roy Fielding for assistance with the rationale behind Upgrade: and OPTIONS.

5. References

1. http://www.ics.uci.edu/pub/ietf/http/hypermail/1997q4/0495.html
2. http://www.w3.org/Protocols/HTTP/1.1/draft-ietf-http-v11-spec-rev-03.txt
3. http://www.ietf.org/internet-drafts/draft-ietf-tls-https-00.txt
4. http://www.imc.org/ietf-apps-tls/draft-hoffman-smtp-ssl
5. http://www.ietf.org/internet-drafts/draft-newman-tls-imappop-03.txt
6. http://www.consensus.com/ietf-tls/murray-auth-ftp-ssl-00.txt
7. http://www.ics.uci.edu/pub/ietf/http/
8. http://www.consensus.com/ietf-tls/
9. http://www.imc.org/ietf-apps-tls/
10. http://www.pwg.org/ipp/index.html
11. http://www.consensus.com/ietf-tls/ssl-tunneling-02.txt
12. http://www.ietf.org/internet-drafts/draft-luotonen-web-proxy-tunneling-00
.txt

13. http://www.consensus.com/ietf-tls/tls-protocol-03.txt
14. http://www.ics.uci.edu/~rohit/http-tls