Network Working Group
Request for Comments: 2817
Updates: 2616
Category: Standards Track

R. Khare
4K Associates / UC Irvine
S. Lawrence
Agranat Systems, Inc.

May 2000

Upgrading to TLS Within HTTP/1.1

Status of this Memo

Copyright Notice

Abstract

This memo explains how to use the Upgrade mechanism in HTTP/1.1 to initiate Transport Layer Security (TLS) over an existing TCP connection. This allows unsecured and secured HTTP traffic to share the same well known port (in this case, http: at 80 rather than https: at 443). It also enables "virtual hosting", so a single HTTP + TLS server can disambiguate traffic intended for several hostnames at a single IP address.

Since HTTP/1.1 [1] defines Upgrade as a hop-by-hop mechanism, this memo also documents the HTTP CONNECT method for establishing end-to-end tunnels across HTTP proxies. Finally, this memo establishes new IANA registries for public HTTP status codes, as well as public or private Upgrade product tokens.

This memo does NOT affect the current definition of the ’https’ URI scheme, which already defines a separate namespace (http://example.org/ and https://example.org/ are not equivalent).

Table of Contents

1. Motivation

The historical practice of deploying HTTP over SSL3 [3] has distinguished the combination from HTTP alone by a unique URI scheme and the TCP port number. The scheme 'http' meant the HTTP protocol alone on port 80, while 'https' meant the HTTP protocol over SSL on port 443. Parallel well-known port numbers have similarly been requested -- and in some cases, granted -- to distinguish between secured and unsecured use of other application protocols (e.g. snews, ftps). This approach effectively halves the number of available well known ports.

At the Washington DC IETF meeting in December 1997, the Applications Area Directors and the IESG reaffirmed that the practice of issuing parallel "secure" port numbers should be deprecated. The HTTP/1.1 Upgrade mechanism can apply Transport Layer Security [6] to an open HTTP connection.

In the nearly two years since, there has been broad acceptance of the concept behind this proposal, but little interest in implementing alternatives to port 443 for generic Web browsing. In fact, nothing in this memo affects the current interpretation of https: URIs. However, new application protocols built atop HTTP, such as the Internet Printing Protocol [7], call for just such a mechanism in order to move ahead in the IETF standards process.

The Upgrade mechanism also solves the "virtual hosting" problem. Rather than allocating multiple IP addresses to a single host, an HTTP/1.1 server will use the Host: header to disambiguate the intended web service. As HTTP/1.1 usage has grown more prevalent, more ISPs are offering name-based virtual hosting, thus delaying IP address space exhaustion.

TLS (and SSL) have been hobbled by the same limitation as earlier versions of HTTP: the initial handshake does not specify the intended hostname, relying exclusively on the IP address. Using a cleartext HTTP/1.1 Upgrade: preamble to the TLS handshake -- choosing the certificates based on the initial Host: header -- will allow ISPs to provide secure name-based virtual hosting as well.

2. 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.

Either the client or server can use the HTTP/1.1 [1] Upgrade mechanism (Section 14.42) to indicate that a TLS-secured connection is desired or necessary. This memo defines the "TLS/1.0" Upgrade token, and a new HTTP Status Code, "426 Upgrade Required".

Section 3 and Section 4 describe the operation of a directly connected client and server. Intermediate proxies must establish an end-to-end tunnel before applying those operations, as explained in Section 5.

2.1 Requirements Terminology

Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in RFC 2119 [11].

3. Client Requested Upgrade to HTTP over TLS

When the client sends an HTTP/1.1 request with an Upgrade header field containing the token "TLS/1.0", it is requesting the server to complete the current HTTP/1.1 request after switching to TLS/1.0.

3.1 Optional Upgrade

A client MAY offer to switch to secured operation during any clear HTTP request when an unsecured response would be acceptable:

```
GET http://example.bank.com/acct_stat.html?749394889300 HTTP/1.1
Host: example.bank.com
Upgrade: TLS/1.0
Connection: Upgrade
```

In this case, the server MAY respond to the clear HTTP operation normally, OR switch to secured operation (as detailed in the next section).

Note that HTTP/1.1 [1] specifies "the upgrade keyword MUST be supplied within a Connection header field (section 14.10) whenever Upgrade is present in an HTTP/1.1 message".

3.2 Mandatory Upgrade

If an unsecured response would be unacceptable, a client MUST send an OPTIONS request first to complete the switch to TLS/1.0 (if possible).

```
OPTIONS * HTTP/1.1
Host: example.bank.com
Upgrade: TLS/1.0
Connection: Upgrade
```

3.3 Server Acceptance of Upgrade Request

As specified in HTTP/1.1 [1], if the server is prepared to initiate the TLS handshake, it MUST send the intermediate "101 Switching Protocol" and MUST include an Upgrade response header specifying the tokens of the protocol stack it is switching to:

```
HTTP/1.1 101 Switching Protocols
Upgrade: TLS/1.0, HTTP/1.1
Connection: Upgrade
```

Note that the protocol tokens listed in the Upgrade header of a 101 Switching Protocols response specify an ordered 'bottom-up' stack.

As specified in HTTP/1.1 [1], Section 10.1.2: "The server will switch protocols to those defined by the response's Upgrade header field immediately after the empty line which terminates the 101 response".

Once the TLS handshake completes successfully, the server MUST continue with the response to the original request. Any TLS handshake failure MUST lead to disconnection, per the TLS error alert specification.

4. Server Requested Upgrade to HTTP over TLS

The Upgrade response header field advertises possible protocol upgrades a server MAY accept. In conjunction with the "426 Upgrade Required" status code, a server can advertise the exact protocol upgrade(s) that a client MUST accept to complete the request.

4.1 Optional Advertisement

As specified in HTTP/1.1 [1], the server MAY include an Upgrade header in any response other than 101 or 426 to indicate a willingness to switch to any (combination) of the protocols listed.

4.2 Mandatory Advertisement

A server MAY indicate that a client request can not be completed without TLS using the "426 Upgrade Required" status code, which MUST include an an Upgrade header field specifying the token of the required TLS version.

```
HTTP/1.1 426 Upgrade Required
Upgrade: TLS/1.0, HTTP/1.1
Connection: Upgrade
```

The server SHOULD include a message body in the 426 response which indicates in human readable form the reason for the error and describes any alternative courses which may be available to the user.

Note that even if a client is willing to use TLS, it must use the operations in Section 3 to proceed; the TLS handshake cannot begin immediately after the 426 response.

5. Upgrade across Proxies

As a hop-by-hop header, Upgrade is negotiated between each pair of HTTP counterparties. If a User Agent sends a request with an Upgrade header to a proxy, it is requesting a change to the protocol between itself and the proxy, not an end-to-end change.

Since TLS, in particular, requires end-to-end connectivity to provide authentication and prevent man-in-the-middle attacks, this memo specifies the CONNECT method to establish a tunnel across proxies.

Once a tunnel is established, any of the operations in Section 3 can be used to establish a TLS connection.

5.1 Implications of Hop By Hop Upgrade

If an origin server receives an Upgrade header from a proxy and responds with a 101 Switching Protocols response, it is changing the protocol only on the connection between the proxy and itself. Similarly, a proxy might return a 101 response to its client to change the protocol on that connection independently of the protocols it is using to communicate toward the origin server.

These scenarios also complicate diagnosis of a 426 response. Since Upgrade is a hop-by-hop header, a proxy that does not recognize 426 might remove the accompanying Upgrade header and prevent the client from determining the required protocol switch. If a client receives a 426 status without an accompanying Upgrade header, it will need to request an end to end tunnel connection as described in Section 5.2 and repeat the request in order to obtain the required upgrade information.

This hop-by-hop definition of Upgrade was a deliberate choice. It allows for incremental deployment on either side of proxies, and for optimized protocols between cascaded proxies without the knowledge of the parties that are not a part of the change.

5.2 Requesting a Tunnel with CONNECT

A CONNECT method requests that a proxy establish a tunnel connection on its behalf. The Request-URI portion of the Request-Line is always an ’authority’ as defined by URI Generic Syntax [2], which is to say the host name and port number destination of the requested connection separated by a colon:

```
CONNECT server.example.com:80 HTTP/1.1
Host: server.example.com:80
```

Other HTTP mechanisms can be used normally with the CONNECT method -- except end-to-end protocol Upgrade requests, of course, since the tunnel must be established first.

For example, proxy authentication might be used to establish the authority to create a tunnel:

```
CONNECT server.example.com:80 HTTP/1.1
Host: server.example.com:80
Proxy-Authorization: basic aGVsbG86d29ybGQ=
```

Like any other pipelined HTTP/1.1 request, data to be tunneled may be sent immediately after the blank line. The usual caveats also apply: data may be discarded if the eventual response is negative, and the connection may be reset with no response if more than one TCP segment is outstanding.

5.3 Establishing a Tunnel with CONNECT

Any successful (2xx) response to a CONNECT request indicates that the proxy has established a connection to the requested host and port, and has switched to tunneling the current connection to that server connection.

It may be the case that the proxy itself can only reach the requested origin server through another proxy. In this case, the first proxy SHOULD make a CONNECT request of that next proxy, requesting a tunnel to the authority. A proxy MUST NOT respond with any 2xx status code unless it has either a direct or tunnel connection established to the authority.

An origin server which receives a CONNECT request for itself MAY respond with a 2xx status code to indicate that a connection is established.

If at any point either one of the peers gets disconnected, any outstanding data that came from that peer will be passed to the other one, and after that also the other connection will be terminated by the proxy. If there is outstanding data to that peer undelivered, that data will be discarded.

6. Rationale for the use of a 4xx (client error) Status Code

Reliable, interoperable negotiation of Upgrade features requires an unambiguous failure signal. The 426 Upgrade Required status code allows a server to definitively state the precise protocol extensions a given resource must be served with.

It might at first appear that the response should have been some form of redirection (a 3xx code), by analogy to an old-style redirection to an https: URI. User agents that do not understand Upgrade: preclude this.

Suppose that a 3xx code had been assigned for "Upgrade Required"; a user agent that did not recognize it would treat it as 300. It would then properly look for a "Location" header in the response and attempt to repeat the request at the URL in that header field. Since it did not know to Upgrade to incorporate the TLS layer, it would at best fail again at the new URL.

7. IANA Considerations

IANA shall create registries for two name spaces, as described in BCP 26 [10]:

- HTTP Status Codes
- HTTP Upgrade Tokens

7.1 HTTP Status Code Registry

The HTTP Status Code Registry defines the name space for the Status-Code token in the Status line of an HTTP response. The initial values for this name space are those specified by:

1. Draft Standard for HTTP/1.1 [1]
2. Web Distributed Authoring and Versioning [4] [defines 420-424]
3. WebDAV Advanced Collections [5] (Work in Progress) [defines 425]
4. Section 6 [defines 426]

Values to be added to this name space SHOULD be subject to review in the form of a standards track document within the IETF Applications Area. Any such document SHOULD be traceable through statuses of either 'Obsoletes' or 'Updates' to the Draft Standard for HTTP/1.1 [1].

7.2 HTTP Upgrade Token Registry

The HTTP Upgrade Token Registry defines the name space for product tokens used to identify protocols in the Upgrade HTTP header field. Each registered token should be associated with one or a set of specifications, and with contact information.

The Draft Standard for HTTP/1.1 [1] specifies that these tokens obey the production for 'product':

```
product         = token ["/" product-version]
product-version = token
```

Registrations should be allowed on a First Come First Served basis as described in BCP 26 [10]. These specifications need not be IETF documents or be subject to IESG review, but should obey the following rules:

1. A token, once registered, stays registered forever.
2. The registration MUST name a responsible party for the registration.
3. The registration MUST name a point of contact.
4. The registration MAY name the documentation required for the token.
5. The responsible party MAY change the registration at any time. The IANA will keep a record of all such changes, and make them available upon request.
6. The responsible party for the first registration of a "product" token MUST approve later registrations of a "version" token together with that "product" token before they can be registered.
7. If absolutely required, the IESG MAY reassign the responsibility for a token. This will normally only be used in the case when a responsible party cannot be contacted.

This specification defines the protocol token "TLS/1.0" as the identifier for the protocol specified by The TLS Protocol [6].

It is NOT required that specifications for upgrade tokens be made publicly available, but the contact information for the registration SHOULD be.

8. Security Considerations

The potential for a man-in-the-middle attack (deleting the Upgrade header) remains the same as current, mixed http/https practice:

- Removing the Upgrade header is similar to rewriting web pages to change https:// links to http:// links.
- The risk is only present if the server is willing to vend such information over both a secure and an insecure channel in the first place.
- If the client knows for a fact that a server is TLS-compliant, it can insist on it by only sending an Upgrade request with a no-op method like OPTIONS.
- Finally, as the https: specification warns, "users should carefully examine the certificate presented by the server to determine if it meets their expectations".

Furthermore, for clients that do not explicitly try to invoke TLS, servers can use the Upgrade header in any response other than 101 or 426 to advertise TLS compliance. 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.

8.1 Implications for the https: URI Scheme

While nothing in this memo affects the definition of the 'https' URI scheme, widespread adoption of this mechanism for HyperText content could use 'http' to identify both secure and non-secure resources.

The choice of what security characteristics are required on the connection is left to the client and server. This allows either party to use any information available in making this determination. For example, user agents may rely on user preference settings or information about the security of the network such as 'TLS required on all POST operations not on my local net', or servers may apply resource access rules such as 'the FORM on this page must be served and submitted using TLS'.

8.2 Security Considerations for CONNECT

A generic TCP tunnel is fraught with security risks. First, such authorization should be limited to a small number of known ports. The Upgrade: mechanism defined here only requires onward tunneling at port 80. Second, since tunneled data is opaque to the proxy, there are additional risks to tunneling to other well-known or reserved ports. A putative HTTP client CONNECTing to port 25 could relay spam via SMTP, for example.

References

[1] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[2] Berners-Lee, T., Fielding, R. and L. Masinter, "URI Generic Syntax", RFC 2396, August 1998.

[3] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[4] Goland, Y., Whitehead, E., Faizi, A., Carter, S. and D. Jensen, "Web Distributed Authoring and Versioning", RFC 2518, February 1999.

[5] Slein, J., Whitehead, E.J., et al., "WebDAV Advanced Collections Protocol", Work In Progress.

[6] Dierks, T. and C. Allen, "The TLS Protocol", RFC 2246, January 1999.

[7] Herriot, R., Butler, S., Moore, P. and R. Turner, "Internet Printing Protocol/1.0: Encoding and Transport", RFC 2565, April 1999.

[8] Luotonen, A., "Tunneling TCP based protocols through Web proxy servers", Work In Progress. (Also available in: Luotonen, Ari. Web Proxy Servers, Prentice-Hall, 1997 ISBN:0136806120.)

[9] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[10] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[11] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Rohit Khare
4K Associates / UC Irvine
3207 Palo Verde
Irvine, CA 92612
US

Phone: +1 626 806 7574
EMail: rohit@4K-associates.com
URI: http://www.4K-associates.com/

Scott Lawrence
Agranat Systems, Inc.
5 Clocktower Place
Suite 400
Maynard, MA 01754
US

Phone: +1 978 461 0888
EMail: lawrence@agranat.com
URI: http://www.agranat.com/

Appendix A. Acknowledgments

The CONNECT method was originally described in a Work in Progress titled, "Tunneling TCP based protocols through Web proxy servers", [8] by Ari Luotonen of Netscape Communications Corporation. It was widely implemented by HTTP proxies, but was never made a part of any IETF Standards Track document. The method name CONNECT was reserved, but not defined in [1].

The definition provided here is derived directly from that earlier memo, with some editorial changes and conformance to the stylistic conventions since established in other HTTP specifications.

Additional Thanks to:

- Paul Hoffman for his work on the STARTTLS command extension for ESMTP.
- Roy Fielding for assistance with the rationale behind Upgrade: and its interaction with OPTIONS.
- Eric Rescorla for his work on standardizing the existing https: practice to compare with.
- Marshall Rose, for the xml2rfc document type description and tools [9].
- Jim Whitehead, for sorting out the current range of available HTTP status codes.
- Henrik Frystyk Nielsen, whose work on the Mandatory extension mechanism pointed out a hop-by-hop Upgrade still requires tunneling.
- Harald Alvestrand for improvements to the token registration rules.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.