Network Working Group
Internet-Draft
Expires: December 21, 1999

R. Khare
4K Associates / UC Irvine
S. Lawrence
Agranat Systems, Inc.
June 22, 1999

Upgrading to TLS Within HTTP/1.1
draft-ietf-tls-http-upgrade-01.txt

Status of this Memo

Abstract

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

This memo also clarifies how to exploit the HTTP/1.1 Upgrade mechanism in general. It creates new IANA registries for public HTTP status codes, and public or private Upgrade product tokens.

This memo also argues that 'https' is insufficient to discriminate between secure and non-secure URIs, and henceforth http: alone should be used. That is to say, both https: and port 443 could be safely deprecated upon deployment of this mechanism.

Status Notes

This memo is intended to proceed directly to Proposed Standard, since its functionality has been extensively debated, but not implemented, over the last two years. It is expected to update RFC 2616.

Copyright Notice

Table of Contents

1. Motivation 3
2. Introduction 3
3. Client Requested Upgrade to HTTP over TLS 4
3.1 Requesting Upgrade When Unsecured Is Not Acceptable 4
3.2 Requesting Upgrade When Unsecured Is Acceptable 4
3.3 Server Acceptance of Upgrade Request 5
4. Server Requested Upgrade to HTTP over TLS 5
4.1 Server Required Upgrade to HTTP over TLS 5
4.2 Server Advertised HTTP over TLS 6
5. HTTP Upgrade Usage Considerations 6
5.1 Upgrading across HTTP Proxies 6
6. Rationale for the use of a 4xx (client error) response code . 7
7. Rationale for the HTTP+TLS/1.0 Upgrade token 7
8. IANA Considerations 8
8.1 HTTP Status Code Registry 8
8.2 HTTP Upgrade Token Registry 8
9. Security Considerations 9
9.1 Implications for the https: URI Scheme 9
References 10
Authors' Addresses 10
A. Acknowledgments 11

1. Motivation

The historical practice for deploying HTTP over SSL3[2] 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. Other protocols have similarly requested (and in some cases were issued) a second well known port so that they can distinguish the secured and unsecured modes of operation in this way as well. Taken to its extreme, this approach in effect cuts in half the number of available well known ports.

At the Washington DC IETF meeting in December 1997, the Applications Area Directors, and the IESG broadly, reaffirmed that the practice of issuing parallel "secure" port numbers should be deprecated. The HTTP/1.1 Upgrade mechanism can indeed apply Transport Layer Security[5] to an HTTP connection, over the same port.

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. However, the Internet Printing Protocol[6], one of the first new application protocols built atop HTTP, has called for just such a mechanism in order to move forward 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

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 draft defines the "HTTP+TLS/1.0" Upgrade token and a new HTTP Reply Code, "426 Upgrade Required".

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. This specification provides a procedure for either a client or server to request that this TLS handshake phase begin on an existing HTTP/1.1 connection.

3. Client Requested Upgrade to HTTP over TLS

The client sends an HTTP/1.1 request with an Upgrade header field containing the token "HTTP+TLS/1.0".

3.1 Requesting Upgrade When Unsecured Is Not Acceptable

To complete the switch to secured operation before sending any clear HTTP traffic, the client MAY use a method such as "OPTIONS*".

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

The client MUST use the OPTIONS method if unsecured operation is unacceptable.

3.2 Requesting Upgrade When Unsecured Is Acceptable

The client MAY offer to switch to secured operation during a clear HTTP operation:

```
GET http://bank.example.com/acct_stat.html?749394889300 HTTP/1.1
Host: bank.example.com
Upgrade: HTTP+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).

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" response specifying the upgrade tokens it is switching to:

```
HTTP/1.1 101 Switching Protocols
Upgrade: HTTP+TLS/1.0
```

The TLS handshake bytes begin after the final CRNL of the HTTP response.

If the TLS handshake completes, 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.

In the 'required upgrade' case described in Section 3.1, the client will send the real request after the OPTIONS ("no-op") request has completed.

4. Server Requested Upgrade to HTTP over TLS

The Upgrade header field can be used in HTTP responses to advertise server policy.

4.1 Server Required Upgrade to HTTP over TLS

A server can indicate that a request can not be fulfilled without TLS secured operation using the "426 Upgrade Required" status code [see Section 6 for the rationale for why this is not a 3xx redirect response]. The 426 response MUST include an Upgrade header field specifying the token for the required TLS version.

```
HTTP/1.1 426 Upgrade Required
Upgrade: HTTP+TLS/1.0
...
```

The server cannot know whether or not the client is willing or able to Upgrade. The use of 426 means that the request has failed, as any 4xx code would. This has two important implications:

1. 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.
2. Neither the server nor the client can immediately begin the TLS handshake -- a new request must be made, whether over the same TCP connection or not.

If the client is capable of the protocol set specified by the server in the Upgrade header of a 426 response, it MAY begin a client-initiated sequence as specified in Section 3 to repeat the request.

[Since the original request was presumably sent in the clear, the Section 3.2 method reduce the number of round-trips in this case]

4.2 Server Advertised HTTP over TLS

As specified in [HTTP], the server MAY include an Upgrade header in any response to indicate a willingness to switch to any (combination) of the protocols listed. Only a 101 or 426 response lists Upgrade tokens that MUST be used to successfully complete the request.

5. HTTP Upgrade Usage Considerations

In the course of formalizing this mechanism, several principles of HTTP Upgrade usage have been clarified for future users.

- Servers MUST select at most one of the offered Upgrade tokens in the 101 Switching Protocols response.
- This implies that Upgrade tokens represent "bundles" of functionality. Skipping a sequential upgrade to X/1.0 then to Y/1.0 would require defining a joint XY/1.0 token, for example.
- This implies public Upgrade tokens should be managed by IANA, according to the process in [8].
- Reliable deployment of new protocol extensions requires a definitive failure error, "426 Upgrade Required" in this case. This is broadly useful for any Upgrade usage.

Note that since Upgrade was only defined in HTTP/1.1 (and above), upgraded protocols can assume persistent-connections by default.

5.1 Upgrading across HTTP Proxies

As a hop-by-hop header, Upgrade must be negotiated between each pair of HTTP counterparties. As an end-to-end protocol, HTTP+TLS/1.0 is only applicable across tunnels. The HTTP CONNECT method explicitly constructed a tunnel, but it requires unique port numbers to disambiguate services.

The following rules apply to relaying Upgrade requests:

1. Upon receipt of an Upgrade header field, a proxy server MUST either discard all the offers, or choose to forward only those it agrees to become a tunnel for.
2. Upon receipt of a "101 Switching Protocols" response, a proxy

server MUST become a tunnel, or report a more detailed proxy server error.

Furthermore a caching proxy SHOULD not reply to a request with Upgrade tokens from its cache. Clients are still advised to explicitly include "Cache-control: no-cache" in this case.

Note that these scenarios slightly complicate diagnosis of a 426-status response. Since Upgrade: is a hop-by-hop header, a proxy may have removed the client's original Upgrade request, while the origin server continues to insist no offer was received.

6. Rationale for the use of a 4xx (client error) response 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. Otherwise, there would be no solution in the Section 4.1 case.

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 the code 3YZ 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 HTTP+TLS/1.0, it would at best fail again at the new URL.

7. Rationale for the HTTP+TLS/1.0 Upgrade token

While TLS (and SSL) are properly ignorant of the syntax and semantics of encapsulated, encrypted traffic, it remains inappropriate to infer the protocol being secured by TCP port number. To reinforce the point that the upgraded protocol is now the composition of HTTP and TLS/1.0, we explicitly named the Upgrade token HTTP+TLS/1.0.

Note that the version number in the product token refers to the version of TLS employed; the version of HTTP to be used over TLS following the switch is calculated normally, viz. per the version compatibility rules of HTTP. [Note that while TLS is compatible with previous versions of SSL, they do not have TLS version numbers. If there were a backwards-compatible Upgrade, it might have specified HTTP+SSL/3.0 instead.]

Purely HTTP-compliant extensions such as IPP will reuse HTTP+TLS/1.0, while derivative works such as the Session Initiation Protocol are encouraged to define their own Upgrade mechanism and their own tokens.

8. IANA Considerations

IANA shall create registries for two name spaces, as described in BCP 26[8]:

- HTTP Status Codes
- HTTP Upgrade Tokens

8.1 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[3] [defines 420-424]
3. WebDAV Advanced Collections[4] (Work in Progress) [defines 425]
4. section Section 6 of this specification.[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].

8.2 HTTP Upgrade Token Registry

The HTTP Upgrade Token Registry defines the name space for product tokens used to identify protocols in the 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[8]. These specifications need not be IETF documents or be subject to IESG review, but should obey the following rules:

1. The registration for a given token MUST NOT be changed once registered.
2. The registry MUST NOT register a token whose 'product' component

is the same as that of an already registered token, unless the source of the authority for the registration is the same as the previous registry (if company XYZ, Inc. registered "XYZ/1.0", then no other entity should be allowed to register any token whose product component is "XYZ" without the consent of XYZ, Inc.

An initial value in this namespace is defined in Section Section 7 of this specification.

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

9. Security Considerations

The potential for a man-in-the-middle attack (deleting the HTTP+TLS/1.0 upgrade token) remains the same as current, mixed http/https practice:

- Removing the Upgrade token 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 that 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 connecting as https: (currently) or 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." -- there is no substitute for vigilance.

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.

9.1 Implications for the https: URI Scheme

This mechanism does not use the URI scheme name to indicate the protocol used. That is, any http: URI could be upgraded; and that https: URIs are no guarantee the server will upgrade.

Instead, 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 VPN', and servers may

resource access rules such as 'the form on this page must be served and submitted using TLS'.

This also implies both parties have the option of fallback to a less secure mode of operation if either party cannot shift to TLS and such unsecured operation is acceptable to both and to the human user; this is not possible with the 'https' scheme.

References

[1] Fielding, R.T., et. al, , "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[2] Rescorla, E.K., "HTTP Over TLS", Internet-Draft (Work In Progress), September 1998.

[3] Goland, Y.Y., Whitehead, E.J., et. al, , "Web Distributed Authoring and Versioning", RFC 2518, February 1999.

[4] Slein, J., Whitehead, E.J., et. al, , "WebDAV Advanced Collections Protocol", Internet-Draft (Work in Progress), June 1999.

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

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

[7] Rose, M.T., "Writing I-Ds and RFCs using XML", April 1999.

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

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

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.
- Jim Whitehead, for sorting out the current range of available HTTP status codes.

Full Copyright Statement