Upgrading to TLS within HTTP/1.1 (RFC 2817)

Jörg Uhl

University of Applied Sciences

GEORG-SIMON-OHM FACHHOCHSCHULE NÜRNBERG

Table of content

1 Introduction

1.1 Short summery

2 Client Requested Upgrade to HTTP over TLS

2.1 Optional Upgrade

2.2 Mandatory Upgrade

2.3 Server Acceptance of Upgrade Request

3 Server Requested Upgrade to HTTS over TLS

3.1 Optional Advertisement

3.2 Mandatory Advertisement

4 Upgrade across Proxies

4.1 Implications of Hop By Hop Upgrade

4.2 Requesting a Tunnel with CONNECT

4.3 Establishing a Tunnel with CONNECT

5 Error Situation

6 Security Considerations

6.1 Man-in-the-middle attack

6.2 Implication for the https: URI Schema

6.3 Security Considerations for CONNECT

1 Introduction

The RFC 2817 describes how to upgrade an existing unsecured TCP connection to a secured one. This upgrade mechanism uses the Hypertext Transfer Protocol (HTTP/1.1) and Transport Layer Security (TLS/1.0) Protocols.
This allows unsecured and secured HTTP traffic to share the same well known Port, in this case HTTP on Port 80 rather then HTTPS on Port 443.
It also describes how to establish end-to-end tunnels across HTTP proxies.
Note: The RFC 2817 does not affect the current definition of the HTTPS scheme.

1.1 Short summery

TLS establishes a private end-to-end connection, optionally including strong mutual authentication, using a variety of cryptosystems.
Either a client or a server can use the HTTP/1.1 Upgrade mechanism to indicate that a secured connection is desired or necessary. Therefore the RFC 2817 introduces a new HTTP/1.1 status code (426 Upgrade Required).

2 Client Requested Upgrade to HTTP over TLS

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

2.1 **Optional Upgrade**

A client can offer to switch to secured communication during any HTTP/1.1 request when an unsecured response from the server would be acceptable. Therefore he sends an HTTP request like the following.

```
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.
Note: The upgrade keyword must included within a connection header field whenever upgrade is present in an HTTP/1.1 message.

2.2 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. With this request the client checks out if an upgrade is possible or not.

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

2.3 **Server Acceptance of Upgrade Request**

As specified in HTTP/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
```

The server will switch protocols to those defined by the response’s Upgrade header field immediately after the empty line witch terminates the 101 message. 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.

3 Server Requested Upgrade to HTTS 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.

3.1 Optional Advertisement

As specified in HTTP/1.1, the server can include an Upgrade header in any response other than 101 or 426 to indicate a willingness to switch to any of the protocols listed.

3.2 Mandatory Advertisement

A server may indicate that a client request can not be completed without using TLS the "426 Upgrade Required" status code, which must include an Upgrade header field specifying the token of the necessary TLS version. Even if a client is willing to use TLS, it must use the operations, as mentioned before, to go on.
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.

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

Note: The TLS handshake cannot begin immediately after the 426 response**.**

4 Upgrade across Proxies

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 protocol change between itself and the proxy, not an end-to-end change.

Also a server which requests an upgrade, upgrades the connection between himself and the proxy which initiated the connection for the client.
Once a tunnel is established, any of the Client request operations can be used to establish a TLS connection

4.1 Implications of Hop By Hop Upgrade

If an server receives an Upgrade header from a proxy and responds with a 101 Switching Protocol response, it is changing the protocol only on the connection to the proxy.
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. A proxy that does not recognize 426 can remove the directly following Upgrade header and prevent the client from doing 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 next and repeat the request in order to obtain the required upgrade information.

4.2 Requesting a Tunnel with CONNECT

A CONNECT method requests that a proxy establish a tunnel connection on its behalf. The Request-URI part of the Request-Line is always an 'authority' as defined by URI Generic Syntax, 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=
```

4.3 Establishing a Tunnel with CONNECT

Like any other pipelined HTTP/1.1 request, data to be tunnelled 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. Successful (2xx), e.g. 200 OK, response indicates that the proxy has established a connection and has switched to tunnelling. If the connection may be through another proxy, the first proxy should request a tunnel to the requested server. A proxy is not allowed to respond with any 2xx status code unless it has either a direct or tunnel connection established to the server. An origin server which receives a CONNECT request for itself may respond with a 2xx status code to indicate that a connection is established.

5 Error Situation

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 (3xx code), by analogy to an old-style redirection to an https URI. User agent that did not recognize it would treat it as 300. The client would then look for a "Location" header in the response and attempt to repeat the request at the URL in that header field. Since the client did not know how to Upgrade to the TLS layer, it would at best fail again at the new URL.

6 Security Considerations

6.1 Man-in-the-middle attack

Removing the Upgrade header from a HTTP conversation is similar to rewriting web pages to change https:// links to http:// links. The request is than an unsecured instead a secured one. The same happened if the Upgrade header is removed, the client is not switching to TLS. This risk is only present if the server is willing to serve 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"**, which means: the amount of security depends on the awareness of the user in front of the client computer.

6.2 **Implication for the HTTPS URI Schema**

While nothing in the RFC 2817 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 the server. This allows either party to use any information available in making this determination.
E.g. On Client side, 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”.
On servers may apply resource access rules such as “the FORM on this page must be served and submitted using TLS".

6.3 **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 tunnelling at port 80.
Second, since tunnelled data is opaque to the proxy, there are additional risks to tunnelling to other well-known or reserved ports. A putative HTTP client CONNECTing to port 25 could relay spam via SMTP, for example.