Network Working Group
Request for Comments: 2818
Category: Informational

E. Rescorla
RTFM, Inc.

May 2000

HTTP Over TLS

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

This memo describes how to use TLS to secure HTTP connections over the Internet. Current practice is to layer HTTP over SSL (the predecessor to TLS), distinguishing secured traffic from insecure traffic by the use of a different server port. This document documents that practice using TLS. A companion document describes a method for using HTTP/TLS over the same port as normal HTTP [RFC2817].

Table of Contents

1. Introduction 2
1.1. Requirements Terminology 2
2. HTTP Over TLS 2
2.1. Connection Initiation 2
2.2. Connection Closure 2
2.2.1. Client Behavior 3
2.2.2. Server Behavior 3
2.3. Port Number 4
2.4. URI Format 4
3. Endpoint Identification 4
3.1. Server Identity 4
3.2. Client Identity 5
References 6
Security Considerations 6
Author's Address 6
Full Copyright Statement 7

1. Introduction

HTTP [RFC2616] was originally used in the clear on the Internet. However, increased use of HTTP for sensitive applications has required security measures. SSL, and its successor TLS [RFC2246] were designed to provide channel-oriented security. This document describes how to use HTTP over TLS.

1.1. Requirements Terminology

Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in [RFC2119].

2. HTTP Over TLS

Conceptually, HTTP/TLS is very simple. Simply use HTTP over TLS precisely as you would use HTTP over TCP.

2.1. Connection Initiation

The agent acting as the HTTP client should also act as the TLS client. It should initiate a connection to the server on the appropriate port and then send the TLS ClientHello to begin the TLS handshake. When the TLS handshake has finished. The client may then initiate the first HTTP request. All HTTP data MUST be sent as TLS "application data". Normal HTTP behavior, including retained connections should be followed.

2.2. Connection Closure

TLS provides a facility for secure connection closure. When a valid closure alert is received, an implementation can be assured that no further data will be received on that connection. TLS implementations MUST initiate an exchange of closure alerts before closing a connection. A TLS implementation MAY, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an "incomplete close". Note that an implementation which does this MAY choose to reuse the session. This SHOULD only be done when the application knows (typically through detecting HTTP message boundaries) that it has received all the message data that it cares about.

As specified in [RFC2246], any implementation which receives a connection close without first receiving a valid closure alert (a "premature close") MUST NOT reuse that session. Note that a premature close does not call into question the security of the data already received, but simply indicates that subsequent data might have been truncated. Because TLS is oblivious to HTTP request/response boundaries, it is necessary to examine the HTTP data itself (specifically the Content-Length header) to determine whether the truncation occurred inside a message or between messages.

2.2.1. Client Behavior

Because HTTP uses connection closure to signal end of server data, client implementations MUST treat any premature closes as errors and the data received as potentially truncated. While in some cases the HTTP protocol allows the client to find out whether truncation took place so that, if it received the complete reply, it may tolerate such errors following the principle to "[be] strict when sending and tolerant when receiving" [RFC1958], often truncation does not show in the HTTP protocol data; two cases in particular deserve special note:

> A HTTP response without a Content-Length header. Since data length in this situation is signalled by connection close a premature close generated by the server cannot be distinguished from a spurious close generated by an attacker.
>
> A HTTP response with a valid Content-Length header closed before all data has been read. Because TLS does not provide document oriented protection, it is impossible to determine whether the server has miscomputed the Content-Length or an attacker has truncated the connection.

There is one exception to the above rule. When encountering a premature close, a client SHOULD treat as completed all requests for which it has received as much data as specified in the Content-Length header.

A client detecting an incomplete close SHOULD recover gracefully. It MAY resume a TLS session closed in this fashion.

Clients MUST send a closure alert before closing the connection. Clients which are unprepared to receive any more data MAY choose not to wait for the server’s closure alert and simply close the connection, thus generating an incomplete close on the server side.

2.2.2. Server Behavior

RFC 2616 permits an HTTP client to close the connection at any time, and requires servers to recover gracefully. In particular, servers SHOULD be prepared to receive an incomplete close from the client, since the client can often determine when the end of server data is. Servers SHOULD be willing to resume TLS sessions closed in this fashion.

Implementation note: In HTTP implementations which do not use persistent connections, the server ordinarily expects to be able to signal end of data by closing the connection. When Content-Length is used, however, the client may have already sent the closure alert and dropped the connection.

Servers MUST attempt to initiate an exchange of closure alerts with the client before closing the connection. Servers MAY close the connection after sending the closure alert, thus generating an incomplete close on the client side.

2.3. Port Number

The first data that an HTTP server expects to receive from the client is the Request-Line production. The first data that a TLS server (and hence an HTTP/TLS server) expects to receive is the ClientHello. Consequently, common practice has been to run HTTP/TLS over a separate port in order to distinguish which protocol is being used. When HTTP/TLS is being run over a TCP/IP connection, the default port is 443. This does not preclude HTTP/TLS from being run over another transport. TLS only presumes a reliable connection-oriented data stream.

2.4. URI Format

HTTP/TLS is differentiated from HTTP URIs by using the ’https’ protocol identifier in place of the ’http’ protocol identifier. An example URI specifying HTTP/TLS is:

```
https://www.example.com/~smith/home.html
```

3. Endpoint Identification

3.1. Server Identity

In general, HTTP/TLS requests are generated by dereferencing a URI. As a consequence, the hostname for the server is known to the client. If the hostname is available, the client MUST check it against the server’s identity as presented in the server’s Certificate message, in order to prevent man-in-the-middle attacks.

If the client has external information as to the expected identity of the server, the hostname check MAY be omitted. (For instance, a client may be connecting to a machine whose address and hostname are dynamic but the client knows the certificate that the server will present.) In such cases, it is important to narrow the scope of acceptable certificates as much as possible in order to prevent man in the middle attacks. In special cases, it may be appropriate for the client to simply ignore the server's identity, but it must be understood that this leaves the connection open to active attack.

If a subjectAltName extension of type dNSName is present, that MUST be used as the identity. Otherwise, the (most specific) Common Name field in the Subject field of the certificate MUST be used. Although the use of the Common Name is existing practice, it is deprecated and Certification Authorities are encouraged to use the dNSName instead.

Matching is performed using the matching rules specified by [RFC2459]. If more than one identity of a given type is present in the certificate (e.g., more than one dNSName name, a match in any one of the set is considered acceptable.) Names may contain the wildcard character * which is considered to match any single domain name component or component fragment. E.g., *.a.com matches foo.a.com but not bar.foo.a.com. f*.com matches foo.com but not bar.com.

In some cases, the URI is specified as an IP address rather than a hostname. In this case, the iPAddress subjectAltName must be present in the certificate and must exactly match the IP in the URI.

If the hostname does not match the identity in the certificate, user oriented clients MUST either notify the user (clients MAY give the user the opportunity to continue with the connection in any case) or terminate the connection with a bad certificate error. Automated clients MUST log the error to an appropriate audit log (if available) and SHOULD terminate the connection (with a bad certificate error). Automated clients MAY provide a configuration setting that disables this check, but MUST provide a setting which enables it.

Note that in many cases the URI itself comes from an untrusted source. The above-described check provides no protection against attacks where this source is compromised. For example, if the URI was obtained by clicking on an HTML page which was itself obtained without using HTTP/TLS, a man in the middle could have replaced the URI. In order to prevent this form of attack, users should carefully examine the certificate presented by the server to determine if it meets their expectations.

3.2. Client Identity

Typically, the server has no external knowledge of what the client's identity ought to be and so checks (other than that the client has a certificate chain rooted in an appropriate CA) are not possible. If a server has such knowledge (typically from some source external to HTTP or TLS) it SHOULD check the identity as described above.

References

[RFC2459] Housley, R., Ford, W., Polk, W. and D. Solo, "Internet Public Key Infrastructure: Part I: X.509 Certificate and CRL Profile", RFC 2459, January 1999.

[RFC2616] 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.

[RFC2119] Bradner, S., "Key Words for use in RFCs to indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol", RFC 2246, January 1999.

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

Security Considerations

This entire document is about security.

Author’s Address

Eric Rescorla
RTFM, Inc.
30 Newell Road, #16
East Palo Alto, CA 94303

Phone: (650) 328-8631
EMail: ekr@rtfm.com

Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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