Network Working Group   S. Hollenbeck
Internet-Draft   VeriSign, Inc.
Updates: 2246 (if approved)   October 7, 2002
Expires: April 7, 2003

Transport Layer Security Protocol Compression Methods
draft-ietf-tls-compression-02.txt

Status of this Memo

Copyright Notice

Abstract

The Transport Layer Security (TLS) protocol (RFC 2246) includes features to negotiate selection of a lossless data compression method as part of the TLS Handshake Protocol and to then apply the algorithm associated with the selected method as part of the TLS Record Protocol. TLS defines one standard compression method, CompressionMethod.null, which specifies that data exchanged via the record protocol will not be compressed. This document describes additional compression methods associated with lossless data compression algorithms for use with TLS.

Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

Table of Contents

1. Introduction

The Transport Layer Security (TLS) protocol (RFC 2246, [2]) includes features to negotiate selection of a lossless data compression method as part of the TLS Handshake Protocol and to then apply the algorithm associated with the selected method as part of the TLS Record Protocol. TLS defines one standard compression method, CompressionMethod.null, which specifies that data exchanged via the record protocol will not be compressed. While this single compression method helps ensure that TLS implementations are interoperable, the lack of additional standard compression methods has limited the ability of implementers to develop interoperable implementations that include data compression.

TLS is used extensively to secure client-server connections on the World Wide Web. While these connections can often be characterized as short-lived and exchanging relatively small amounts of data, TLS is also being used in environments where connections can be long-lived and the amount of data exchanged can extend into thousands or millions of octets. XML [4], for example, is increasingly being used as a data representation method on the Internet, and XML tends to be verbose. Compression within TLS is one way to help reduce the bandwidth and latency requirements associated with exchanging large amounts of data while preserving the security services provided by TLS.

This document describes additional compression methods associated with lossless data compression algorithms for use with TLS. Standardization of the compressed data formats and compression algorithms associated with the compression methods is beyond the scope of this document.

2. Compression Methods

TLS [2] includes the following compression method structure in sections 6.1 and 7.4.1.2 and Appendix sections A.4.1 and A.6:

```
enum { null(0), (255) } CompressionMethod;
```

which allows for later specification of up to 256 different compression methods. This definition is updated to segregate the range of allowable values into three zones:

1. Values from 0 (zero) through 63 decimal (0x3F) inclusive are reserved for future standardization efforts of the IETF TLS working group.

2. Values from 64 decimal (0x40) through 192 decimal (0xC0) are reserved for assignment by the IANA for specifications developed outside the TLS working group. Assignments from this range of values MUST be made by the IANA and MUST be associated with a formal reference that describes the compression method.

3. Values from 193 decimal (0xC1) through 255 decimal (0xFF) are reserved for private use.

Additional information describing the role of the IANA in the allocation of compression method identifiers is described in Section 5.

In addition, this definition is updated to include assignment of two additional compression methods:

```
enum { null(0), ZLIB(1), LZS(2), (255) } CompressionMethod;
```

These two compression methods are defined to provide implementers with alternatives based on compression performance, ease of implementation, and licensing requirements (see Section 3 for a description of intellectual property considerations). ZLIB is generally known as a freely-available, widely-deployed compression method, whereas LZS is generally known to provide memory footprint and performance advantages in stateful networking applications.

As described in section 6 of RFC 2246, TLS is a stateful protocol. Compression methods used with TLS can be either stateful (the compressor maintains it's state through all compressed records) or stateless (the compressor compresses each record independently), but there seems to be little known benefit in using a stateless compression method within TLS. Compression methods SHOULD be stateful to take advantage of the state management features offered

by TLS.

2.1 ZLIB Compression

The ZLIB compression method and encoding format is described in RFC 1950 [5] and RFC 1951 [6]. Examples of ZLIB use in IETF protocols can be found in RFC 1979 [7], RFC 2394 [8], and RFC 3274 [9].

ZLIB allows the sending compressor to select from among several options to provide varying compression ratios, processing speeds, and memory requirements. The receiving decompressor will automatically adjust to the parameters selected by the sender.

The sender MUST flush the compressor completely each time a compressed payload is produced. All data that was submitted for compression MUST be included in the compressed output, with no data retained to be included in a later output payload. Flushing ensures that each payload is complete so that compressed packet payloads can be decompressed independently.

2.2 LZS Compression

The Lempel Zif Stac (LZS) compression method and encoding format is described in ANSI publication X3.241 [10]. Examples of LZS use in IETF protocols can be found in RFC 1967 [11], RFC 1974 [12], and RFC 2395 [13].

LZS has the ability to maintain history information when compressing and decompressing packet payloads. When used with TLS, the compression history MUST be reset by the sender before compressing data and the decompression history MUST be reset by the receiver before decompressing data to ensure that compressed packet payloads can be decompressed independently.

The sender MUST flush the compressor completely each time a compressed payload is produced. All data that was submitted for compression MUST be included in the compressed output, with no data retained to be included in a later output payload. Flushing ensures that each payload is complete so that compressed packet payloads can be decompressed independently.

3. Intellectual Property Considerations

Many compression algorithms are subject to patent or other intellectual property rights claims. Implementers are encouraged to seek legal guidance to better understand the implications of developing implementations of the compression methods described in this document or other documents that describe compression methods for use with TLS.

4. Internationalization Considerations

The compression method identifiers specified in this document are machine-readable numbers. As such, issues of human internationalization and localization are not introduced.

5. IANA Considerations

This document does not have a direct impact on the IANA, but it does define ranges of compression method values for future assignment. Values from the range reserved for future standardization efforts of the TLS working group MUST be assigned according to the "Standards Action" policy described in RFC 2434 [3]. Values from the range reserved for private use MUST be used according to the "Private Use" policy described in RFC 2434. Values from the general IANA pool MUST be assigned according to the "IETF Consensus" policy described in RFC 2434.

6. Security Considerations

This document does not introduce any topics that alter the threat model addressed by TLS. The security considerations described throughout RFC 2246 [2] apply here as well.

Depending on the ciphersuite, symmetric encryption in TLS does not fully hide the length of symmetrically encrypted data. Use of TLS compression SHOULD take into account that the length of compressed data may leak more information than the length of the original uncompressed data.

7. Acknowledgements

The concepts described in this document were originally discussed on the IETF TLS working group mailing list in December, 2000. The author acknowledges the contributions to that discussion provided by Jeffrey Altman, Eric Rescorla, and Marc Van Heyningen. Later suggestions that have been incorporated into this document were provided by Tim Dierks, Pasi Eronen, Peter Gutmann, Nikos Mavroyanopoulos, and Bodo Moeller.

Normative References

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

[2] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A. and P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

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

Informative References

[4] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler, "Extensible Markup Language (XML) 1.0 (2nd ed)", W3C REC-xml, October 2000, <http://www.w3.org/TR/REC-xml>.

[5] Deutsch, L. and J-L. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.

[6] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[7] Woods, J., "PPP Deflate Protocol", RFC 1979, August 1996.

[8] Pereira, R., "IP Payload Compression Using DEFLATE", RFC 2394, December 1998.

[9] Gutmann, P., "Compressed Data Content Type for Cryptographic Message Syntax (CMS)", RFC 3274, June 2002.

[10] American National Standards Institute, "Data Compression Method, Adaptive Coding with Sliding Window of Information

Interchange", ANSI X3.241, 1994.

[11] Schneider, K., Friend, R. and K. Fox, "PPP LZS-DCP Compression Protocol (LZS-DCP)", RFC 1967, August 1996.

[12] Friend, R., Simpson, W. and K. Fox, "PPP Stac LZS Compression Protocol", RFC 1974, August 1996.

[13] Friend, R. and R. Monsour, "IP Payload Compression Using LZS", RFC 2395, December 1998.

Author's Address

Scott Hollenbeck
VeriSign, Inc.
21345 Ridgetop Circle
Dulles, VA 20166-6503
US

EMail: shollenbeck@verisign.com

Full Copyright Statement

Acknowledgement

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