Internet Engineering Task Force
Internet-Draft
Updates: [[List TBD]] (if approved)
Intended status: Standards Track
Expires: December 20, 2018

K. Moriarty
Dell EMC
S. Farrell
Trinity College Dublin

June 18, 2018

Deprecating TLSv1.0 and TLSv1.1
draft-moriarty-tls-oldversions-diediedie-00

Abstract

This document [if approved] formally deprecates Transport Layer Security (TLS) versions 1.0 [RFC2246] and 1.1 [RFC4346] and moves these documents to the historic state. These versions lack support for current and recommended cipher suites, and various government and industry profiiles of applications using TLS now mandate avoiding these old TLS versions. TLSv1.2 has been the recommended version for IETF protocols since 2008, providing sufficient time to transition away from older versions. Products having to support older versions increase the attack surface unnecessarily and increase opportunities for misconfigurations. Supporting these older versions also requires additional effort for library and product maintenance.

This document updates the backward compatibility sections of TLS RFCs [[list TBD]] to prohibit fallback to TLSv1.0 and TLSv1.1. This document also updates RFC 7525.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[[Text in double-square brackets, like this, is commentary intended to be fixed as the draft evolves. You're already seen that we need to figure out the list of RFCs that this'd update in the abstract.]]

Transport Layer Security (TLS) versions 1.0 [RFC2246] and 1.1 [RFC4346] were superceded by TLSv1.2 [RFC5246] in 2008, which has now itself been superceded by TLSv1.3 [I-D.ietf-tls-tls13]. It is therefore timely to further deprecate these old versions. The expection is that TLSv1.2 will continue to be used for many years alongside TLSv1.3.

TLSv1.1 and TLSv1.0 are also actively being deprecated in accordance with guidance from government agencies (e.g. NIST SP 80052r2 [NIST800-52r2]) and industry consortia such as the Payment Card Industry Association (PCI) [PCI-TLS1].

The primary technical reasons for deprecating these versions include:

- They require implementation of older cipher suites that are no longer desirable for cryptographic reasons, e.g. TLSv1.0 makes TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA mandatory to implement

- Lack of support for current recommended cipher suites, especially using AEAD ciphers which are not supported prior to TLS 1.2

- Support for four protocol versions increases the likelihood of misconfiguration

- At least one widely-used library has plans to drop TLSv1.1 and TLSv1.0 support in upcoming releases; products using such libraries would need to use older versions of the libraries to support TLSv1.0 and TLSv1.1, which is clearly undesirable

Deprecation of these versions is intended to assist developers as additional justification to no longer support older TLS versions and to migrate to a minimum of TLSv1.2. Deprecation also assists product teams with phasing out support for the older versions to reduce the attack surface and the scope of maintenance for protocols in their offerings.

[[This draft is being written now so that the TLS WG chairs can just hit the "publication requested" button as soon as there is WG consensus to deprecate these ancient versions of TLS. The authors however think that deprecation now is timely.]]

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Support for Deprecation

Industry is actively following guidance provided by NIST and the PCI Council deprecating TLSv1.0 and TLSv1.1 by June 30, 2018. TLSv1.2 should remain a minimum baseline for TLS support at this time.

Specific details on attacks against TLSv1.0 and TLSv1.1 as well as their mitigations are provided in NIST SP800-52r2 [NIST800-52r2] and referenced RFCs. Although the attacks have been mitigated, if support is dropped for future library releases for these versions, it is unlikely attacks found going forward will be mitigated in older library releases.

They have provided the following rationale.

2.1. NIST 800-52r2

The following text is copied with permission from NIST SP800-52r2 [NIST800-52r2] section 1.2 History of TLS.

TLS 1.1, specified in [RFC4346], was developed to address weaknesses discovered in TLS 1.0, primarily in the areas of initialization vector selection and padding error processing. Initialization vectors were made explicit to prevent a certain class of attacks on the Cipher Block Chaining (CBC) mode of operation used by TLS. The handling of padding errors was altered to treat a padding error as a bad message authentication code, rather than a decryption failure. In addition, the TLS 1.1 RFC acknowledges attacks on CBC mode that rely on the time to compute the message authentication code (MAC). The TLS 1.1 specification states that to defend against such attacks, an implementation must process records in the same manner regardless of whether padding errors exist. Further implementation considerations for CBC modes (which were not included in RFC4346 [RFC4346]) are discussed in Section 3.3.2.

TLS 1.2, specified in RFC5246 [RFC5246], made several cryptographic enhancements, particularly in the area of hash functions, with the ability to use or specify the SHA-2 family algorithms for hash, MAC, and Pseudorandom Function (PRF) computations. TLS 1.2 also adds authenticated encryption with associated data (AEAD) cipher suites.

TLS 1.3, specified in TLSv1.3 [I-D.ietf-tls-tls13], represents a significant change to TLS that aims to address threats that have arisen over the years. Among the changes are a new handshake protocol, a new key derivation process that uses the HMAC-based Extract-and-Expand Key Derivation Function (HKDF), and the removal of cipher suites that use static RSA or DH key exchanges, the CBC mode of operation, or SHA-1. The list of extensions that can be used with TLS 1.3 has been reduced considerably.

3. Usage and Support

[[This section can be removed upon publication.]]

Usage statistics for TLSv1.0 and TLSv1.1 vary slightly, but are in general very low already and soon to decline further with the impending PCI deadline to migrate off of TLSv1.0 by June 30, 2018. As of January 2018, Stackexchange [StackExchange] quoted 4 percent of browsers using TLSv1.0.

The Alexa Top 1 Million Analysis [Alexa] from February 2018 shows that for the sites surveyed, the vast majority support TLSv1.2 (98.9 percent), with a mere 0.8 percent using TLSv1.0 and an even smaller percentage using TLSv1.1.

Support for TLSv1.0 has been removed or will be by July 2018 from the following standards, products, and services:

- 3GPP 5G

- [[Numerous web sites...]]

- CloudFare [CloudFlare]

- Amazon Elastic Load Balancing [Amazon]

- GitHub [GIT]

Many web sites have taken the action of including the deprecation of TLSv1.1 into their plans for deprecating TLSv1.0 for the PCI council deadline. Support for TLSv1.1 has been removed or will be by July 2018 from the following standards, products, and services:

- 3GPP 5G Release 16

- GitHub [GIT]

- Amazon Elastic Load Balancing [Amazon]

- CloudFare [CloudFlare]

- [[Numerous web sites...]]

4. Do Not Use TLSv1.0

TLSv1.0 MUST NOT be used. Negotiation of TLSv1.0 from any version of TLS MUST NOT be permitted.

Any version of TLS is more secure then TLSv1.0 and can be configured to prevent interception, though the highest version available is preferable.

Pragmatically, clients MUST NOT send a ClientHello with ClientHello.client_version set to {03,01}. Similarly, servers MUST NOT send a ServerHello with ServerHello.server_version set to {03,01}. Any party receiving a Hello message with the protocol version set to {03,01} MUST respond with a "protocol_version" alert message and close the connection.

Historically, TLS specifications were not clear on what the record layer version number (TLSPlaintext.version) could contain when sending ClientHello. Appendix E of [RFC5246] notes that TLSPlaintext.version could be selected to maximize interoperability, though no definitive value is identified as ideal. That guidance is still applicable; therefore, TLS servers MUST accept any value {03,XX} (including {03,00}) as the record layer version number for ClientHello, but they MUST NOT negotiate TLSv1.0.

5. Do Not Use TLSv1.1

TLSv1.1 MUST NOT be used. Negotiation of TLSv1.1 from any version of TLS MUST NOT be permitted.

Pragmatically, clients MUST NOT send a ClientHello with ClientHello.client_version set to {03,02}. Similarly, servers MUST NOT send a ServerHello with ServerHello.server_version set to {03,02}. Any party receiving a Hello message with the protocol version set to {03,02} MUST respond with a "protocol_version" alert message and close the connection.

Any newer version of TLS is more secure then TLSv1.1 and can be configured to prevent interception, though the highest version available is preferable. Support for TLSv1.1 is dwindling in libraries and will impact security going forward if mitagations for attacks cannot be easily addressed and supported in older libraries.

Historically, TLS specifications were not clear on what the record layer version number (TLSPlaintext.version) could contain when sending ClientHello. Appendix E of [RFC5246] notes that TLSPlaintext.version could be selected to maximize interoperability, though no definitive value is identified as ideal. That guidance is still applicable; therefore, TLS servers MUST accept any value {03,XX} (including {03,00}) as the record layer version number for ClientHello, but they MUST NOT negotiate TLSv1.1.

6. Updates to RFC7525

[[Since RFC7525 is BCP195, there'll probably be some process-fun to do an update of that. Formally, it may be that this document becomes a new part of BCP195 I guess, but we can figure that out with chairs and ADs.]]

This documents updates [RFC7525] Section 3.1.1 changing SHOULD NOT to MUST NOT as follows:

- Implementations MUST NOT negotiate TLS version 1.0 [RFC2246].

  Rationale: TLS 1.0 (published in 1999) does not support many modern, strong cipher suites. In addition, TLS 1.0 lacks a per-record Initialization Vector (IV) for CBC-based cipher suites and does not warn against common padding errors.

- Implementations MUST NOT negotiate TLS version 1.1 [RFC4346].

  Rationale: TLS 1.1 (published in 2006) is a security improvement over TLS 1.0 but still does not support certain stronger cipher suites.

This documents updates [RFC7525] Section 3.1.2 changing SHOULD NOT to MUST NOT as follows:

- Implementations MUST NOT negotiate DTLS version 1.0 [RFC4347].

  Version 1.0 of DTLS correlates to version 1.1 of TLS (see above).

7. Security Considerations

This document deprecates two older protocol versions for security reasons already described. The attack surface is reduced when there are a smaller number of supported protocols and fallback options are removed.

8. Acknowledgements

Thank you to those that reviewed and improved this document, including Yoav Nir, Russ Housley, and David Black.

9. IANA Considerations

[[This memo includes no request to IANA.]]

10. Contributors

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <https://www.rfc-editor.org/info/rfc2246>.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <https://www.rfc-editor.org/info/rfc4346>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

[Alexa] Will be deleted before publication, "The Alexa Top 1 Million Analysis https://scotthelme.co.uk/alexa-top-1-million-analysis-february-2018/", 2018.

[Amazon] CloudFlare, "Amazon Elastic Load Balancing Support Deprecated TLSv1.0 and TLSv1.1 https://aws.amazon.com/about-aws/whats-new/2017/02/elastic-load-balancing-support-for-tls-1-1-and-tls-1-2-pre-defined-security-policies/", 2017.

[CloudFlare] CloudFlare, "CloudFlare Deprecated TLSv1.0 and TLSv1.1 https://blog.cloudflare.com/deprecating-old-tls-versions-on-cloudflare-dashboard-and-api/", 2018.

[GIT] GitHub, "GitHub Deprecates TLSv1.0 and TLSv1.1 https://githubengineering.com/crypto-removal-notice/", 2018.

[I-D.ietf-tls-iana-registry-updates]
Salowey, J. and S. Turner, "IANA Registry Updates for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", draft-ietf-tls-iana-registry-updates-05 (work in progress), May 2018.

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-28 (work in progress), March 2018.

[NIST800-52r2]
National Institute of Standards and Technology, "NIST SP800-52r2 https://csrc.nist.gov/CSRC/media/Publications/sp/800-52/rev-2/draft/documents/sp800-52r2-draft.pdf", 2018.

[PCI-TLS1]
PCI Security Standards Council, "Migrating from SSL and Early TLS https://www.pcisecuritystandards.org/documents/Migrating-from-SSL-Early-TLS-Info-Supp-v1_1.pdf", 2016.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, DOI 10.17487/RFC4347, April 2006, <https://www.rfc-editor.org/info/rfc4347>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC7568] Barnes, R., Thomson, M., Pironti, A., and A. Langley, "Deprecating Secure Sockets Layer Version 3.0", RFC 7568, DOI 10.17487/RFC7568, June 2015, <https://www.rfc-editor.org/info/rfc7568>.

[StackExchange]
StackExchange - will be deleted before publication, "Stackexchange https://security.stackexchange.com/questions/177182/is-there-a-list-of-old-browsers-that-only-support-tls-1-0", 2018.

Appendix A. Change Log

Note to RFC Editor: if this document does not obsolete an existing RFC, please remove this appendix before publication as an RFC.

Authors' Addresses

Kathleen Moriarty
Dell EMC
176 South Street
Hopkinton
United States

EMail: Kathleen.Moriarty.ietf@gmail.com

Stephen Farrell
Trinity College Dublin
Dublin 2
Ireland

Phone: +353-1-896-2354
EMail: stephen.farrell@cs.tcd.ie