Workgroup: Transport Layer Security
Internet-Draft:
draft-thomson-tls-keylogfile-01
Published: 28 July 2023
Intended Status: Informational
Expires: 29 January 2024
Authors: M. Thomson
Mozilla

The SSLKEYLOGFILE Format for TLS

Abstract

A format that supports the logging information about the secrets used in a TLS connection is described. Recording secrets to a file in SSLKEYLOGFILE format allows diagnostic and logging tools that use this file to decrypt messages exchanged by TLS endpoints.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://martinthomson.github.io/sslkeylogfile/draft-thomson-tls-keylogfile.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-thomson-tls-keylogfile/.

Discussion of this document takes place on the Transport Layer Security Working Group mailing list (mailto:tls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tls/. Subscribe at https://www.ietf.org/mailman/listinfo/tls/.

Source for this draft and an issue tracker can be found at https://github.com/martinthomson/sslkeylogfile.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 29 January 2024.

Table of Contents

1. Introduction

Debugging or analyzing protocols can be challenging when TLS [TLS13] is used to protect the content of communications. Inspecting the content of encrypted messages in diagnostic tools can enable more thorough analysis.

Over time, multiple TLS implementations have informally adopted a file format that logging the secret values generated by the TLS key schedule. In many implementations, the file that the secrets are logged to is specified in an environment variable named "SSLKEYLOGFILE", hence the name of SSLKEYLOGFILE format. Note the use of "SSL" as this convention originally predates the adoption of TLS as the name of the protocol.

This document describes the SSLKEYLOGFILE format. This format can be used for TLS 1.2 [TLS12] and TLS 1.3 [TLS13]. The format also

supports earlier TLS versions, though use of earlier versions is forbidden [RFC8996]. This format can also be used with DTLS [DTLS13], QUIC [RFC9000][RFC9001], and protocols that also use the TLS key schedule. Use of this format could complement other protocol-specific logging such as QLOG [QLOG].

1.1. Applicability Statement

The artifact that this document describes - if made available to entities other than endpoints - completely undermines the core guarantees that TLS provides. This format is intended for use in systems where TLS only protects test data. While the access that this information provides to TLS connections can be useful for diagnosing problems while developing systems, this mechanism **MUST NOT** be used in a production system.

1.2. Conventions and Definitions

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. The SSLKEYLOGFILE Format

A file in SSLKEYLOGFILE format is a text file. This document specifies the character encoding as UTF-8 [RFC3629]. Though the format itself only includes ASCII characters [RFC0020], comments **MAY** contain other characters. Though Unicode is permitted in comments, the file **MUST NOT** contain a unicode byte order mark (U+FEFF).

Lines are terminated using the line ending convention of the platform on which the file is generated. Tools that process these files **MUST** accept CRLF (U+13 followed by U+10), CR (U+13), or LF (U+10) as a line terminator. Lines are ignored if they are empty or if the first character is an octothorp character ('#', U+23). Other lines of the file each contain a single secret.

Each secret is described using a single line composed of three values that are separated by a single space character (U+20). These values are:

label: The label identifies the type of secret that is being conveyed; see Section 2.1 for a description of the labels that are defined in this document.

client_random: The 32 byte value of the Random field from the ClientHello message that established the TLS connection. This value is encoded as 64 hexadecimal characters. Including this

field allows a single file to include secrets from multiple connections and for the secrets to be applied to the correct connection, though this depends on being able to match records to the correct ClientHello message.

secret: The value of the identified secret for the identified connection. This value is encoded in hexadecimal, with a length that depends on the size of the secret.

For the hexadecimal values of the client_random or secret, no convention exists for the case of characters 'a' through 'f' (or 'A' through 'F'). Files can be generated with either, so either form **MUST** be accepted when processing a file.

Diagnostic tools that accept files in this format might choose to ignore lines that do not conform to this format in the interest of ensuring that secrets can be obtained from corrupted files.

2.1. Secret Labels for TLS 1.3

An implementation of TLS 1.3 produces a number of values as part of the key schedule (see Section 7.1 of [TLS13]). Each of the following labels correspond to the equivalent secret produced by the key schedule:

CLIENT_EARLY_TRAFFIC_SECRET:
This secret is used to protect records sent by the client as early data, if early data is attempted by the client. Note that a server that rejects early data will not log this secret, though a client that attempts early data can do so unconditionally.

EARLY_EXPORTER_MASTER_SECRET:
This secret is used for early exporters. Like the CLIENT_EARLY_TRAFFIC_SECRET, this is only generated when early data is attempted and might not be logged by a server if early data is rejected.

CLIENT_HANDSHAKE_TRAFFIC_SECRET:
This secret is used to protect handshake records sent by the client.

SERVER_HANDSHAKE_TRAFFIC_SECRET:
This secret is used to protect handshake records sent by the server.

CLIENT_TRAFFIC_SECRET_0:
This secret is used to protect application_data records sent by the client immediately after the handshake completes. This secret is identified as client_application_traffic_secret_0 in the TLS 1.3 key schedule.

SERVER_TRAFFIC_SECRET_0:
This secret is used to protect application_data records sent by the server immediately after the handshake completes. This secret is identified as server_application_traffic_secret_0 in the TLS 1.3 key schedule.

EXPORTER_SECRET:
This secret is used in generating exporters Section 7.5 of [TLS13].

Each of the preceding labels are identified using the lowercase form of the label in [TLS13], except as noted. Note that the order that labels appear here corresponds to the order in which they are presented in [TLS13], but there is no guarantee that implementations will log secrets strictly in this order.

Key updates (Section 7.2 of [TLS13]) result in new secrets being generated for protecting application_data records. The label used for these secrets comprises a base label of "CLIENT_TRAFFIC_SECRET_" for a client or "SERVER_TRAFFIC_SECRET_" for a server, plus the decimal value of a counter. This counter identifies the number of key updates that occurred to produce this secret. This counter starts at 0, which produces the first application data traffic secret, as above.

2.2. Secret Labels for TLS 1.2

An implementation of TLS 1.2 [TLS12] (and also earlier versions) use the label "CLIENT_RANDOM" to identify the "master" secret for the connection.

3. Security Considerations

Access to the content of a file in SSLKEYLOGFILE format allows an attacker to break the confidentiality protection on any TLS connections that are included in the file. This includes both active connections and connections for which encrypted records were previously stored. Ensuring adequate access control on these files therefore becomes very important.

Implementations that support logging this data need to ensure that logging can only be enabled by those who are authorized. Allowing logging to be initiated by any entity that is not otherwise authorized to observe or modify the content of connections for which secrets are logged could represent a privilege escalation attack. Implementations that enable logging also need to ensure that access to logged secrets is limited, using appropriate file permissions or equivalent access control mechanisms.

In order to support decryption, the secrets necessary to remove record protection are logged. However, as the keys that can be derived from these secrets are symmetric, an adversary with access to these secrets is also able to encrypt data for an active connection. This might allow for injection or modification of application data on a connection that would otherwise be protected by TLS.

As some protocols that depend on TLS generate encryption keys, the SSLKEYLOGFILE format includes keys that identify the secret used in TLS exporters or early exporters (Section 7.5 of [TLS13]. Knowledge of these secrets can enable more than inspection or modification of encrypted data, depending on how an application protocol uses exporters. For instance, exporters might be used for session bindings (e.g., [RFC8471]), authentication (e.g., [RFC9261]), or other derived secrets that are used in application context. An adversary that obtains these secrets might be able to use this information to attack these applications. A TLS implementation might either choose to omit these secrets in contexts where the information might be abused or to require separate authorization to enable logging of exporter secrets.

Logging the TLS 1.2 "master" secret provides the recipient of a file in SSLKEYLOGFILE far greater access to an active connection. This can include the ability to resume the connection and impersonate either endpoint, insert records that result in renegotiation, or even forge Finished messages. Implementations might avoid these risks by not logging this secret value.

4. IANA Considerations

The "application/sslkeylogfile" media type can be used to describe content in the SSLKEYLOGFILE format. IANA [has added/is requested to add] the following information to the "Media Types" registry at https://www.iana.org/assignments/media-types:

Type name: application
Subtype name: sslkeylogfile
Required parameters: N/A
Optional parameters: N/A
Encoding considerations: 8bit (Unicode without BOM or ASCII only)
Security considerations: See Section 3.
Interoperability considerations: Line endings might differ from platform convention
Published specification: This document
Applications that use this media type: Diagnostic and analysis tools that need to decrypt data that is otherwise protected by TLS.
Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): N/A

Macintosh file type code(s): N/A

Person & email address to contact for further information: See the Authors' Addresses section.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See the Authors' Addresses section.

Change controller: IESG

5. References

5.1. 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/rfc/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

[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/rfc/rfc8174>.

[TLS12] 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/rfc/rfc5246>.

[TLS13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

5.2. Informative References

[DTLS13] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[QLOG] Marx, R., Niccolini, L., Seemann, M., and L. Pardue, "Main logging schema for qlog", Work in Progress, Internet-Draft, draft-ietf-quic-qlog-main-schema-06, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-qlog-main-schema-06>.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/rfc/rfc20>.

[RFC8471] Popov, A., Ed., Nystroem, M., Balfanz, D., and J. Hodges, "The Token Binding Protocol Version 1.0", RFC 8471, DOI 10.17487/RFC8471, October 2018, <https://www.rfc-editor.org/rfc/rfc8471>.

[RFC8996] Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <https://www.rfc-editor.org/rfc/rfc8996>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[RFC9261] Sullivan, N., "Exported Authenticators in TLS", RFC 9261, DOI 10.17487/RFC9261, July 2022, <https://www.rfc-editor.org/rfc/rfc9261>.

Acknowledgments

The SSLKEYLOGFILE format originated in the NSS project, but it has evolved over time as TLS has changed. Many people contributed to this evolution. The author is only documenting the format as it is used.

Author's Address

Martin Thomson
Mozilla

Email: mt@lowentropy.net