TLS
Internet-Draft
Intended status: Standards Track
Expires: November 19, 2017

N. Sullivan
Cloudflare Inc.

May 18, 2017

Exported Authenticators in TLS
draft-ietf-tls-exported-authenticator-00

Abstract

This document describes a mechanism in Transport Layer Security (TLS) to provide an exportable proof of ownership of a certificate that can be transmitted out of band and verified by the other party.

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 http://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 November 19, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Authenticator 3
3. API considerations 4
4. Security Considerations 5
5. Acknowledgements 5
6. Normative References 5
Author's Address 6

1. Introduction

This document provides a way to authenticate one party of a Transport Layer Security (TLS) communication to another using a certificate after the session has been established. This allows both the client and server to prove ownership of additional identities at any time after the handshake has completed. This proof of authentication can be exported and transmitted out of band from one party to be validated by the other party.

This mechanism provides two advantages over the authentication that TLS natively provides:

multiple identities - Endpoints that are authoritative for multiple identities - but do not have a single certificate that includes all of the identities - can authenticate with those identities over a single connection.

spontaneous authentication - Endpoints can authenticate after a connection is established, in response to events in a higher-layer protocol, as well as integrating more context.

This document intends to replace much of the functionality of renegotiation in previous versions of TLS. It has the advantages over renegotiation of not requiring additional on-the-wire changes during a connection. For simplicity, only TLS 1.2 and later are supported.

Post-handshake authentication is defined in TLS 1.3, but it has the disadvantage of requiring additional state to be stored in the TLS state machine and it composes poorly with multiplexed connection protocols like HTTP/2. It is also only available for client authentication. This mechanism is intended to be used as part of a replacement for post-handshake authentication in applications.

2. Authenticator

The authenticator is a structured message that can be exported from either party of a TLS connection. It can be sent out-of-band to the other party of a TLS connection to be validated.

An authenticator message can be constructed by either the client or the server given an established TLS connection, a certificate, and a corresponding private key. This authenticator uses the message structures from section 4.4. of [I-D.ietf-tls-tls13], but different parameters. Also, unlike the Certificate and CertificateRequest messages in TLS 1.3, the messages described in this draft are not encrypted with a handshake key.

Each authenticator is computed using a Handshake Context and Finished MAC Key derived from the TLS session. The Handshake Context is identical for both parties of the TLS connection, the Finished MAC Key is dependent on whether the authenticator is created by the client or the server.

- The Handshake Context is an [RFC5705] (for TLS 1.2) or [I-D.ietf-tls-tls13] (for TLS 1.3) exporter value derived using the label "EXPORTER-authenticator handshake context" and length 64 bytes.

- The Finished MAC Key is an exporter value derived using the label "EXPORTER-server authenticator finished key" or "EXPORTER-client authenticator finished key", depending on the sender. The length of this key is equal to the length of the output of the hash function selected in TLS for the pseudorandom function (PRF); cipher suites that do not use the TLS PRF MUST define a hash function that can be used for this purpose or they cannot be used.

If the connection is TLS 1.2, the master secret MUST have been computed with the extended master secret [RFC7627] to avoid key synchronization attacks.

Certificate The certificate to be used for authentication and any supporting certificates in the chain.

The certificate message contains an opaque string called certificate_request_context which MUST be unique for a given connection. Its format should be defined by the application layer protocol and MUST be non-zero length. For example, it may be a sequence number used by the higher-level protocol during the transport of the authenticator to the other party.

CertificateVerify A signature over the value Hash(Handshake Context || Certificate)

Finished A HMAC over the value Hash(Handshake Context || Certificate || CertificateVerify) using the hash function from the handshake and the Finished MAC Key as a key.

The certificates used in the Certificate message MUST conform to the requirements of a Certificate message in the version of TLS negotiated. This is described in section 4.2.3. of [I-D.ietf-tls-tls13] and sections 7.4.2. and 7.4.6. of [RFC5246].

The exported authenticator message is the concatenation of messages: Certificate || CertificateVerify || Finished

3. API considerations

The creation and validation of exported authenticators SHOULD be implemented inside TLS library even if it is possible to implement it at the application layer. TLS implementations supporting the use of exported authenticators MUST provide application programming interfaces by which clients and servers may request and verify exported authenticator messages.

Given an established connection, the application should be able to obtain an authenticator by providing the following:

- certificate_request_context (from 1 to 255 bytes)

- valid certificate chain for the connection and associated extensions (OCSP, SCT, etc.)

- signer (either the private key associated with the certificate, or interface to perform private key operation)

Given an established connection and an exported authenticator message, the application should be able to provide the authenticator to the connection. If the Finished and CertificateVerify messages verify, the TLS library should return the following:

- certificate chain and extensions

- certificate_request_context

In order for the application layer to communicate which certificates it will accept, an API should be exposed that returns an array of TLS 1.3 SignatureScheme objects that corresponds to the signature

algorithms that the library is willing to validate in an exported authenticator message.

4. Security Considerations

The Certificate/Verify/Finished pattern intentionally looks like the TLS 1.3 pattern which now has been analyzed several times. In the case where the client presents an authenticator to a server, [SIGMAC] presents a relevant framework for analysis.

Authenticators are independent and unidirectional. There is no explicit state change inside TLS when an authenticator is either created or validated. * This property makes it difficult to formally prove that a server is jointly authoritative over multiple certificates, rather than individually authoritative over each. * There is no indication in the TLS layer about which point in time an authenticator was computed. Any feedback about the time of creation or validation of the authenticator should be tracked as part of the application layer semantics if required.

5. Acknowledgements

Comments on this proposal were provided by Martin Thomson. Suggestions for the security considerations section were provided by Karthikeyan Bhargavan.

6. Normative References

[I-D.ietf-tls-tls13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-20 (work in progress), April 2017.

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

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <http://www.rfc-editor.org/info/rfc5705>.

[RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <http://www.rfc-editor.org/info/rfc7627>.

[SIGMAC] Krawczyk, H., "A Unilateral-to-Mutual Authentication Compiler for Key Exchange (with Applications to Client Authentication in TLS 1.3)", 2016, <https://eprint.iacr.org/2016/711.pdf>.

Author's Address

Nick Sullivan
Cloudflare Inc.

Email: nick@cloudflare.com