TLS
Internet-Draft
Intended status: Standards Track
Expires: January 18, 2018

N. Sullivan
Cloudflare Inc.
July 17, 2017

Exported Authenticators in TLS
draft-ietf-tls-exported-authenticator-03

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

Copyright Notice

Table of Contents

1. Introduction 2
2. Authenticator 3
3. API considerations 5
4. Security Considerations 6
5. Acknowledgements 6
6. References 6
6.1. Normative References 6
6.2. Informative References 7
Author's Address 7

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 transmitted to the other party of the TLS connection at the application layer. The application layer protocol used to send the authenticator SHOULD use TLS as its underlying transport.

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 [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. These values are derived using an exporter as described in [RFC5705] (for TLS 1.2) or [TLS13] (for TLS 1.3). These values use different labels depending on the role of the sender:

- o The Handshake Context is an exporter value that is derived using the label "EXPORTER-client authenticator handshake context" or "EXPORTER-server authenticator handshake context" for authenticators sent by the client and server respectively.

- o The Finished MAC Key is an exporter value derived using the label "EXPORTER-server authenticator finished key" or "EXPORTER-client authenticator finished key" for authenticators sent by the client and server respectively.

The context_value used for the exporter is absent (length zero) for all four values. The length of the exported value 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. This structure is defined in [TLS13], Section 4.4.2.

The certificate message contains an opaque string called certificate_request_context. The format of certificate_request_context is defined by the application layer protocol and its value can be used to differentiate exported authenticators. For example, the application may use a sequence number used by the higher-level protocol during the transport of the authenticator to the other party. Using a unique and unpredictable value ties the authenticator to a given context, allowing the application to prevent authenticators from being replayed or precomputed by an attacker with temporary access to a private key.

CertificateVerify This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate.

```
 struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
 } CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see Section 4.2.3 of [TLS13] for the definition of this field). The signature is a digital signature using that algorithm. The signature scheme MUST be a valid signature scheme for TLS 1.3. This excludes all RSASSA-PKCS1-v1_5 algorithms and ECDSA algorithms that are not supported in TLS 1.3. For servers, this signature scheme must match one of the signature and hash algorithms advertised in the signature_algorithms extension of the ClientHello. The signature is computed using the over the concatenation of:

o A string that consists of octet 32 (0x20) repeated 64 times

o The context string "Exported Authenticator" (which is not NULL-terminated)

o A single 0 byte which serves as the separator

o 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 [TLS13] and Sections 7.4.2 and 7.4.6 of [RFC5246]. Alternative certificate formats such as [RFC7250] Raw Public Keys are not supported.

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

A given exported authenticator can be validated by checking the validity of the CertificateVerify message and recomputing the Finished message to see if it matches.

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 call an "authenticate" API which takes as input:

- certificate_request_context (from 0 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)

- signature scheme

The API returns the exported authenticator as output.

Given an established connection and an exported authenticator message, the application SHOULD be able to call a "validate" API that takes an exported authenticator as an input. If the Finished and CertificateVerify messages verify correctly, the API returns the following as output:

- certificate chain and extensions

- certificate_request_context

In order for the application layer to be able to choose the certificates and signature schemes to use when constructing an authenticator, a TLS server SHOULD expose an API that returns the content of the signature_algorithms extension of client's ClientHello 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 Section 4 were provided by Karthikeyan Bhargavan.

6. References

6.1. Normative References

[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>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <http://www.rfc-editor.org/info/rfc7250>.

[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>.

[TLS13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-21 (work in progress), July 2017.

6.2. Informative References

[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