tls   C. Wood
Internet-Draft   Apple, Inc.
Intended status: Experimental   October 22, 2018
Expires: April 25, 2019

                  Importing External PSKs for TLS 1.3
                 draft-wood-tls-external-psk-importer-00

Abstract

   This document describes an interface for importing external PSK (Pre-
   Shared Key) into TLS 1.3.

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 April 25, 2019.

Table of Contents

1. Introduction

   TLS 1.3 [RFC8446] supports pre-shared key (PSK) resumption, wherein
   PSKs can be established via session tickets from prior connections or
   externally via some out-of-band mechanism. The protocol mandates
   that each PSK only be used with a single hash function. This was
   done to simplify protocol analysis. TLS 1.2, in contrast, has no
   such requirement, as a PSK may be used with any hash algorithm and
   the TLS 1.2 PRF. This means that external PSKs could possibly be re-
   used in two different contexts with the same hash functions during
   key derivation. Moreover, it requires external PSKs to be
   provisioned for specific hash functions.

   To mitigate these problems, external PSKs can be bound to a specific
   hash function when used in TLS 1.3, even if they are associated with
   a different KDF (and hash function) when provisioned. This document
   specifies an interface by which external PSKs may be imported for use
   in a TLS 1.3 connection to achieve this goal. In particular, it
   describes how KDF-bound PSKs can be differentiated by different hash
   algorithms to produce a set of candidate PSKs, each of which are
   bound to a specific hash function. This expands what would normally
   have been a single PSK identity into a set of PSK identities.
   However, it requires no change to the TLS 1.3 key schedule.

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.

3. Overview

Intuitively, key importers mirror the concept of key exporters in TLS
in that they diversify a key based on some contextual information
before use in a connection. In contrast to key exporters, wherein
differentiation is done via an explicit label and context string, the
key importer defined herein uses a label and set of hash algorithms
to differentiate an external PSK into one or more PSKs for use.

Imported keys do not require negotiation for use, as a client and
server will not agree upon identities if not imported correctly.
Thus, importers induce no protocol changes with the exception of
expanding the set of PSK identities sent on the wire.

3.1. Terminology

o External PSK (EPSK): A PSK established or provisioned out-of-band,
   i.e., not from a TLS connection, which is a tuple of (Base Key,
   External Identity, KDF). The associated KDF (and hash function)
   may be undefined.

o Base Key: The secret value of an EPSK.

o External Identity: The identity of an EPSK.

o Imported Identity: The identity of a PSK as sent on the wire.

4. Key Import

A key importer takes as input an EPSK with external identity
'external_identity' and base key 'epsk', as defined in Section 3.1,
along with an optional label, and transforms it into a set of PSKs
and imported identities for use in a connection based on supported
HashAlgorithms. In particular, for each supported HashAlgorithm
'hash', the importer constructs an ImportedIdentity structure as
follows:

```
struct {
    opaque external_identity<1...2^16-1>;
    opaque label<0..2^8-1>;
    HashAlgorithm hash;
} ImportedIdentity;
```

A unique and imported PSK (IPSK) with base key 'ipskx' bound to this
identity is then computed as follows:

epskx = HKDF-Extract(0, epsk)
ipskx = HKDF-Expand-Label(epskx, "derived psk", Hash(ImportedIdentity),
Hash.length)

The hash function used for HKDF [RFC5869] is that which is associated with the external PSK. It is not bound to ImportedIdentity.hash. If no hash function is specified, SHA-256 MUST be used.

The resulting IPSK base key 'ipskx' is then used as the binder key in TLS 1.3 with identity ImportedIdentity.

With knowledge of the supported hash functions, one may import PSKs before the start of a connection.

EPSKs may be imported for early data use if they are bound to protocol settings and configurations that would otherwise be required for early data with normal (ticket-based PSK) resumption. Minimally, that means ALPN, QUIC transport settings, etc., must be provisioned alongside these EPSKs.

5. Deprecating Hash Functions

If a client or server wish to deprecate a hash function and no longer use it for TLS 1.3, they may remove this hash function from the set of hashes used during while importing keys. This does not affect the KDF operation used to derive concrete PSKs.

6. TLS 1.2 Compatibility

Key importers do not affect TLS 1.2 in any way. Recall that TLS 1.2 permits computing the TLS PRF with any hash algorithm and PSK. Thus, a PSK may be used with the same KDF (and underlying HMAC hash algorithm) as TLS 1.3 with importers. However, critically, the derived PSK will not be the same since the importer differentiates the PSK via the identity and hash function. Thus, TLS 1.3 imported PSKs are distinct from those used in TLS 1.2 and avoid cross-protocol collisions.

7. Security Considerations

This is a WIP draft and has not yet seen significant security analysis.

8. IANA Considerations

This document has no IANA requirements.

9. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

   [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>.

   [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand
              Key Derivation Function (HKDF)", RFC 5869,
              DOI 10.17487/RFC5869, May 2010, <https://www.rfc-
              editor.org/info/rfc5869>.

   [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms
              (SHA and SHA-based HMAC and HKDF)", RFC 6234,
              DOI 10.17487/RFC6234, May 2011, <https://www.rfc-
              editor.org/info/rfc6234>.

   [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>.

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

Appendix A. Acknowledgements

   The authors thank David Benjamin, Eric Rescorla, and Martin Thomson
   for discussions that led to the production of this document.

Author's Address

   Christopher A. Wood
   Apple, Inc.

   Email: cawood@apple.com