Network Working Group   DeKok, Alan
INTERNET-DRAFT   FreeRADIUS
Updates: 5247, 5281, 7170
Category: Standards Track
<draft-dekok-emu-tls-eap-types-02.txt>
19 April 2020

TLS-based EAP types and TLS 1.3
draft-dekok-emu-tls-eap-types-02.txt

Abstract

EAP-TLS [RFC5216] is being updated for TLS 1.3 in [EAPTLS]. Many other EAP [RFC3748] and [RFC5247] types also depend on TLS, such as FAST [RFC4851], TTLS [RFC5281], TEAP [RFC7170], and possibly many vendor specific EAP methods. This document updates those methods in order to use the new key derivation methods available in TLS 1.3. Additional changes necessitated by TLS 1.3 are also discussed.

Table of Contents

1. Introduction

EAP-TLS is being updated for TLS 1.3 in [EAPTLS]. Many other EAP types also depend on TLS, such as FAST [RFC4851], TTLS [RFC5281], TEAP [RFC7170], and possibly many vendor specific EAP methods. All of these methods use key derivation functions that rely on the information which is no longer available in TLS 1.3. As such, all of those methods are incompatible with TLS 1.3.

We wish to enable the use of TLS 1.3 in the wider Internet community. As such, it is necessary to update the above EAP types. These changes involve defining new key derivation functions. We also discuss implementation issues in order to highlight differences between TLS 1.3 and earlier versions of TLS.

1.1. Requirements Language

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. Using TLS-based EAP methods with TLS 1.3

In general, all of the requirements of [EAPTLS] apply to other EAP methods that wish to use TLS 1.3. Implementations of other methods that wish to use TLS 1.3 MUST follow the guidelines in [EAPTLS].

There are, however a few key differences between EAP-TLS and other TLS-based EAP methods that necessitate this document. The simplest difference is that [EAPTLS] uses the EAP-TLS type ID (0x0D) in a number of calculations. That value should change for other method types.

More complex differences include derivation of additional keying material, as in FAST [RFC4851].

2.1. Key Derivation

The key derivation for TLS-based EAP methods depends on the value of the Type-Code as defined by [IANA]. The most important definition is of the Type-Code:

```
   Type-Code  = EAP Method type
```

The Type-Code is defined to be 1 octet for values smaller than 255. Where expanded EAP Type Codes are used, the Type-Code is defined to be the Expanded Type Code (including the Type, Vendor-Id (in network byte order) and Vendor-Type fields (in network byte order) defined in [RFC3748] Section 5.7).

```
   Type-Code  = 0xFE || Vendor-Id || Vendor-Type
```

Unless otherewise discussed below, the key derivation functions for all TLS-based EAP types are defined as follows:

```
   Key_Material = TLS-Exporter("EXPORTER_EAP_TLS_Key_Material",
                               Type-Code, 128)
   IV           = TLS-Exporter("EXPORTER_EAP_TLS_IV", Type-Code, 64)
   Method-Id    = TLS-Exporter("EXPORTER_EAP_TLS_Method-Id",
                               Type-Code, 64)
   Session-Id   = Type-Code || Method-Id
   MSK          = Key_Material(0, 63)
   EMSK         = Key_Material(64, 127)
   Enc-RECV-Key = MSK(0, 31)
   Enc-SEND-Key = MSK(32, 63)
   RECV-IV      = IV(0, 31)
   SEND-IV      = IV(32, 63)
```

We note that these definitions re-use the EAP-TLS exporter labels,

and change the derivation only by adding a dependency on Type-Code. The reason for this change is simplicity. There does not appear to be compelling reasons to make the labels method-specific, when we can just include the Type-Code in the key derivation.

These definitions apply in their entirety to TTLS [RFC5281] and PEAP as defined in [PEAP] and [MSPEAP]. Some definitions apply to FAST and TEAP, with exceptions as noted below.

It is RECOMMENDED that vendor-defined TLS-based EAP methods use the above definitions for TLS 1.3.

2.2. FAST and TEAP

EAP-FAST [RFC4851] and TEAP [RFC7170] cannot use the above derivation. Those methods use an inner tunnel EMSK to calculate the outer EMSK. As such, those key derivations cannot use the above derivation.

EAP-FAST previously used a PAC, which is a type of pre-shared key (PSK). Such uses are deprecated in TLS 1.3. As such, PAC provisioning is no longer part of EAP-FAST when TLS 1.3 is used.

TBD: Is this true? Comments from EAP-FAST people are useful here.

The key derivation for FAST and TEAP are similar enough that they gave be given together here. The only difference is the Type-Code. All derivations not given here are the same as given above in the previous section.

session_key_seed = TLS-Exporter("EXPORTER: session key seed", Type-Code, 40)

For FAST, the session_key_seed is also used as the key_block, as defined in [RFC4851] Section 5.1.

```
S-IMCK[0] = session_key_seed
  For j = 1 to n-1 do
       IMCK[j] = TLS-Exporter("EXPORTER: Inner Methods Compound
Keys", S-IMCK[j-1] | IMSK[j], 60)
       S-IMCK[j] = first 40 octets of IMCK[j]
       CMK[j] = last 20 octets of IMCK[j]
```

Where | denotes concatenation.

MSK = TLS-Exporter("EXPORTER: Session Key Generating Function", S-IMCK[j], 64) EMSK = TLS-Exporter("EXPORTER: Extended Session Key Generating Function", S-IMCK[j], 64)

3. Application Data

Unlike previous TLS version, TLS 1.3 continues negotiation after the TLS session has been initialized. Some implementations use the TLS "Finished" state as a signal that application data is now available, and an "inner tunnel" session can now be negotiated. As noted in [RFC8446], TLS 1.3 may include a "NewSessionTicket" after the "Finished" state. This change can cause many implementations to fail.

In order to correct this failure, implementations MUST also check if "Application Data" is available for a TLS connection. If the underlying TLS connection is still performing negotiations, then implementations MUST NOT send, or expect to receive application data in the TLS session.

We note that some TLS Application Programming Interfaces (APIs) signal the availability of application data by returning zero octets of application data, where they previously had returned an error which signalled that negotiation should continue. For those APIs, implementations SHOULD treat the combination of the "Finished" state and the availability of zero octets of application data as a signal that TLS negotiation has completed, and that the tunneled process can begin.

[EAPTLS] uses an empty application record to indicate that negotiation has finished. Methods which use "inner tunnel" methods should instead begin their "inner tunnel" negotiation by sending type-specific application data.

4. Security Considerations

[EAPTLS] Section 5 is included here by reference.

Updating the above EAP methods to use TLS 1.3 is of high importance for the Internet Community. Using the most recent security protocols can significantly improve security and privace of a network.

In some cases, client certificates are not used for TLS-based EAP methods. In those cases, the user is authenticated only after successful completion of the inner tunnel authentication. However, the TLS protocol sends a NewSessionTicket after receiving the TLS Finished message from the client, and therefore before the user is authenticated.

This separation of data allows for a "time of use, time of check" security issue. Malicious clients can begin a session and receive

the NewSessionTicket. Then prior to authentication, the malicious client can abort the authentication session. The malicious client can then use the obtained NewSessionTicket to "resume" the previous session.

As a result, EAP servers MUST NOT permit sessions to be resumed until after authentication has successfully completed. This requirement may be met in a number of ways. For example, by not caching the session ticket until after authentication has completed, or by marking up the cached session ticket with a flag stating whether or not authentication has completed.

5. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the TLS-based EAP methods for TLS 1.3 protocol in accordance with [RFC8126].

This memo requires IANA to add the following labels to the TLS Exporter Label Registry defined by [RFC5705]. These labels are used in derivation of Key_Material, IV and Method-Id as defined above in Section ?

The labels above need to be added to the "TLS Exporter Labels" registry.

* TBD

6. References

6.1. Normative References

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

[RFC3748]
Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC5216]
Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008

[RFC5247]
Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008,

[RFC5705]
Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, March 2010

[RFC7170]
Zhou, H., et al., "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, May 2014.

[RFC8126]
Cotton, M., et al, "Guidelines for Writing an IANA Considerations Section in RFCs", RC 8126, June 2017.

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

[RFC8446]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.

[EAPTLS]
Mattsson, J., and Sethi, M., "Using EAP-TLS with TLS 1.3", draft-ietf-emu-eap-tls13-03, November 2018.

[IANA]
https://www.iana.org/assignments/eap-numbers/eap-numbers.xhtml#eap-numbers-4

6.2. Informative References

[PEAP]
Palekar, A. et al, "Protected EAP Protocol (PEAP)", draft-josefsson-pppext-eap-tls-eap-06.txt, March 2003.

[MSPEAP]
https://msdn.microsoft.com/en-us/library/cc238354.aspx

[RFC4851]
Cam-Winget, N., et al, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", RFC 4851, May 2007.

[RFC5281]
Funk, P., and Blake-Wilson, S., "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, August 2008.

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org