Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 12, 2019

R. Housley
Vigil Security

November 8, 2018

TLS 1.3 Extension for Certificate-based Authentication with an External Pre-Shared Key

draft-housley-tls-tls13-cert-with-extern-psk-03

Abstract

This document specifies a TLS 1.3 extension that allows a server to authenticate with a combination of a certificate and an external pre-shared key (PSK).

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 May 12, 2019.

Copyright Notice

Copyright (c) 2018 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 (https://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.

1. Introduction

The TLS 1.3 [RFC8446] handshake protocol provides two mutually exclusive forms of server authentication. First, the server can be authenticated by providing a signature certificate and creating a valid digital signature to demonstrate that it possesses the corresponding private key. Second, the server can be authenticated by demonstrating that it possesses a pre-shared key (PSK) that was established by a previous handshake. A PSK that is established in this fashion is called a resumption PSK. A PSK that is established by any other means is called an external PSK. This document specifies a TLS 1.3 extension permitting certificate-based server authentication to be combined with an external PSK as an input to the TLS 1.3 key schedule.

2. Terminology

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. Motivation and Design Rationale

The invention of a large-scale quantum computer would pose a serious challenge for the cryptographic algorithms that are widely deployed today, including the digital signature algorithms that are used to authenticate the server in the TLS 1.3 handshake protocol. It is an open question whether or not it is feasible to build a large-scale quantum computer, and if so, when that might happen. However, if such a quantum computer is invented, many of the cryptographic algorithms and the security protocols that use them would become vulnerable.

The TLS 1.3 handshake protocol employs key agreement algorithms that could be broken by the invention of a large-scale quantum computer [I-D.hoffman-c2pq]. These algorithms include Diffie-Hellman (DH) [DH] and Elliptic Curve Diffie-Hellman (ECDH) [IEEE1363]. As a result, an adversary that stores a TLS 1.3 handshake protocol exchange today could decrypt the associated encrypted communications in the future when a large-scale quantum computer becomes available.

In the near-term, this document describes TLS 1.3 extension to protect today's communications from the future invention of a large-scale quantum computer by providing a strong external PSK as an input to the TLS 1.3 key schedule while preserving the authentication

provided by the existing certificate and digital signature mechanisms.

4. Extension Overview

This section provides a brief overview of the "tls_cert_with_extern_psk" extension.

The client includes the "tls_cert_with_extern_psk" extension in the ClientHello message. The "tls_cert_with_extern_psk" extension MUST accompanied by the "key_share", "psk_key_exchange_modes", and "pre_shared_key" extensions. The "pre_shared_key" extension MUST be the last extension in the ClientHello message, and it provides a list of external PSK identifiers that the client is willing to use with this server. Since tls_cert_with_extern_psk" extension is intended to be used only with initial handshakes, it MUST NOT be sent alongside the "early_data" extension. These extension are all described in Section 4.2 of [RFC8446].

If the server is willing to use one of the external PSKs listed in the "pre_shared_key" extension and perform certificate-based authentication, then the server includes the "tls_cert_with_extern_psk" extension in the ServerHello message. The "tls_cert_with_extern_psk" extension MUST accompanied by the "key_share" and "pre_shared_key" extensions. If none of the external PSKs in the list provided by the client is acceptable to the server, then the "tls_cert_with_extern_psk" extension is omitted from the ServerHello message.

The successful negotiation of the "tls_cert_with_extern_psk" extension requires the TLS 1.3 key schedule processing to include both the selected external PSK and the (EC)DHE shared secret value. As a result, the Early Secret, Handshake Secret, and Master Secret values all depend upon the value of the selected external PSK.

The authentication of the server and optional authentication of the client depend upon the ability to generate a signature that can be validated with the public key in their certificates. The authentication processing is not changed in any way by the selected external PSK.

Each external PSK is associated with a single Hash algorithm. The hash algorithm MUST be set when the PSK is established, with a default of SHA-256 if no hash algorithm is specified during establishment.

5. Certificate with External PSK Extension

This section specifies the "tls_cert_with_extern_psk" extension, which MAY appear in the ClientHello message and ServerHello message. It MUST NOT appear in any other messages. The "tls_cert_with_extern_psk" extension MUST NOT appear in the ServerHello message unless "tls_cert_with_extern_psk" extension appeared in the preceding ClientHello message. If an implementation recognizes the "tls_cert_with_extern_psk" extension and receives it in any other message, then the implementation MUST abort the handshake with an "illegal_parameter" alert.

The general extension mechanisms enable clients and servers to negotiate the use of specific extensions. Clients request extended functionality from servers with the extensions field in the ClientHello message. If the server responds with a HelloRetryRequest message, then the client sends another ClientHello message as described in Section 4.1.2 of [RFC8446], and it MUST include the same "tls_cert_with_extern_psk" extension as the original ClientHello message or abort the handshake.

Many server extensions are carried in the EncryptedExtensions message; however, the "tls_cert_with_extern_psk" extension is carried in the ServerHello message. It is only present in the ServerHello message if the server recognizes the "tls_cert_with_extern_psk" extension and the server possesses one of the external PSKs offered by the client in the "pre_shared_key" extension in the ClientHello message.

The Extension structure is defined in [RFC8446]; it is repeated here for convenience.

```
struct {
    ExtensionType extension_type;
    opaque extension_data<0..2^16-1>;
} Extension;
```

The "extension_type" identifies the particular extension type, and the "extension_data" contains information specific to the particular extension type.

This document specifies the "tls_cert_with_extern_psk" extension, adding one new type to ExtensionType:

```
enum {
    tls_cert_with_extern_psk(TBD), (65535)
} ExtensionType;
```

The "tls_cert_with_extern_psk" extension is relevant when the client and server possess an external PSK in common that can be used as an input to the TLS 1.3 key schedule.

To use an external PSK with certificates, clients MUST provide the "tls_cert_with_extern_psk" extension, and it MUST be accompanied by the "key_share", "psk_key_exchange_modes", and "pre_shared_key" extensions in the ClientHello. If clients offer a "tls_cert_with_extern_psk" extension without all of these other extensions, servers MUST abort the handshake. The client MAY also find it useful to include the the "supported_groups" extension. Note that Section 4.2 of [RFC8446] allows extensions to appear in any order, with the exception of the "pre_shared_key" extension, which MUST be the last extension in the ClientHello. Also, there MUST NOT be more than one instance of each extension in the ClientHello message.

The "key_share" extension is defined in Section 4.2.8 of [RFC8446].

The "psk_key_exchange_modes" extension is defined in Section 4.2.9 of [RFC8446]. The "psk_key_exchange_modes" extension restricts both the use of PSKs offered in this ClientHello and those which the server might supply via a subsequent NewSessionTicket. As a result, clients MUST include the psk_dhe_ke mode, and clients MAY also include the psk_ke mode to support a subsequent NewSessionTicket. Servers MUST select the psk_dhe_ke mode for the initial handshake. Servers MUST select a key exchange mode that is listed by the client for subsequent handshakes that include the resumption PSK from the initial handshake.

The "supported_groups" extension is defined in Section 4.2.7 of [RFC8446].

The "pre_shared_key" extension is defined in Section 4.2.11 of [RFC8446]. the syntax is repeated below for convenience. All of the listed PSKs MUST be external PSKs.

```
struct {
    opaque identity<1..2^16-1>;
    uint32 obfuscated_ticket_age;
} PskIdentity;

opaque PskBinderEntry<32..255>;

struct {
    PskIdentity identities<7..2^16-1>;
    PskBinderEntry binders<33..2^16-1>;
} OfferedPsks;

struct {
    select (Handshake.msg_type) {
        case client_hello: OfferedPsks;
        case server_hello: uint16 selected_identity;
    };
} PreSharedKeyExtension;
```

The OfferedPsks contains the list of PSK identities and associated binders for the external PSKs that the client is willing to use with the server.

The identities are a list of external PSK identities that the client is willing to negotiate with the server. Each external PSK has an associated identity that is known to the client and the server. (The identity is also referred to as an identifier or a label.)

The obfuscated_ticket_age is not used for external PSKs; clients SHOULD set this value to 0, and servers MUST ignore the value.

The binders are a series of HMAC values, one for each external PSK offered by the client, in the same order as the identities list. The HMAC value is computed using the binder_key, which is derived from the external PSK, and a partial transcript of the current handshake. Generation of the binder_key from the external PSK is described in Section 7.1 of [RFC8446]. The partial transcript of the current handshake includes a partial ClientHello up to and including the PreSharedKeyExtension.identities field as described in Section 4.2.11.2 of [RFC8446].

The selected_identity contains the external PSK identity that the server selected from the list offered by the client. If none of the offered external PSKs in the list provided by the client are acceptable to the server, then the "tls_cert_with_extern_psk" extension MUST be omitted from the ServerHello message. The server MUST validate the binder value that corresponds to the selected

external PSK as described in Section 4.2.11.2 of [RFC8446]. If the binder does not validate, the server MUST abort the handshake with an "illegal_parameter" alert. Servers SHOULD NOT attempt to validate multiple binders; rather they SHOULD select one of the offered external PSKs and validate only the binder that corresponds to that external PSK.

When the "tls_cert_with_extern_psk" extension is successfully negotiated, authentication of the server depends upon the ability to generate a signature that can be validated with the public key in the server's certificate. This is accomplished by the server sending the Certificate and CertificateVerify messages as described in Sections 4.4.2 and 4.4.3 of [RFC8446].

TLS 1.3 does not permit the server to send a CertificateRequest message when a PSK is being used. This restriction is removed when the "tls_cert_with_extern_psk" extension is negotiated, allowing the certificate-based authentication for both the client and the server. If certificate-based client authentication is desired, this is accomplished by the client sending the Certificate and CertificateVerify messages as described in Sections 4.4.2 and 4.4.3 of [RFC8446].

Section 7.1 of [RFC8446] specifies the TLS 1.3 Key Schedule. The successful negotiation of the "tls_cert_with_extern_psk" extension requires the key schedule processing to include both the external PSK and the (EC)DHE shared secret value.

If the client and the server have different values associated with the selected external PSK identifier, then the client and the server will compute different values for every entry in the key schedule, which will lead to the termination of the connection with a "decrypt_error" alert.

6. IANA Considerations

IANA is requested to update the TLS ExtensionType Registry to include "tls_cert_with_extern_psk" with a value of TBD and the list of messages "CH, SH" in which the "tls_cert_with_extern_psk" extension may appear.

7. Security Considerations

The Security Considerations in [RFC8446] remain relevant.

TLS 1.3 [RFC8446] does not permit the server to send a CertificateRequest message when a PSK is being used. This restriction is removed when the "tls_cert_with_extern_psk" extension

is offered by the client and accepted by the server. However, TLS 1.3 does not permit an external PSK to be used in the same fashion as a resumption PSK, and this extension does not alter those restrictions. Thus, a certificate MUST NOT be used with a resumption PSK.

Implementations must protect the external pre-shared key (PSK). Compromise of the external PSK will make the encrypted session content vulnerable to the future invention of a large-scale quantum computer.

Implementers should not transmit the same content on a connection that is protected with an external PSK and a connection that is not. Doing so may allow an eavesdropper to correlate the connections, making the content vulnerable to the future invention of a large-scale quantum computer.

Implementations must choose external PSKs with a secure key management technique, such as pseudo-random generation of the key or derivation of the key from one or more other secure keys. The use of inadequate pseudo-random number generators (PRNGs) to generate external PSKs can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the external PSKs and searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality random numbers is difficult. [RFC4086] offers important guidance in this area.

TLS 1.3 [RFC8446] takes a conservative approach to PSKs; they are bound to a specific hash function and KDF. By contrast, TLS 1.2 [RFC5246] allows PSKs to be used with any hash function and the TLS 1.2 PRF. Thus, the safest approach is to use a PSK with either TLS 1.2 or TLS 1.3. However, any PSK that might be used with both TLS 1.2 and TLS 1.3 must be used with only one hash function, which is the one that is bound for use in TLS 1.3. This restriction is less than optimal when users want to provision a single PSK. While the constructions used in TLS 1.2 and TLS 1.3 are both based on HMAC [RFC2104], the constructions are different, and there is no known way in which reuse of the same PSK in TLS 1.2 and TLS 1.3 that would produce related outputs.

8. Acknowledgments

Many thanks to Nikos Mavrogiannopoulos, Nick Sullivan, Martin Thomson, and Peter Yee for their review and comments; their efforts have improved this document.

9. References

9.1. 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/info/rfc2119>.

[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>.

9.2. Informative References

[DH] Diffie, W. and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory V.IT-22 n.6, June 1977.

[I-D.hoffman-c2pq] Hoffman, P., "The Transition from Classical to Post-Quantum Cryptography", draft-hoffman-c2pq-04 (work in progress), August 2018.

[IEEE1363] Institute of Electrical and Electronics Engineers, "IEEE Standard Specifications for Public-Key Cryptography", IEEE Std 1363-2000, 2000.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5246] 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/info/rfc5246>.

Author's Address

Russ Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA

Email: housley@vigilsec.com