Abstract

The pre-shared key mechanism available in TLS 1.3 is not suitable for usage with low-entropy keys, such as passwords entered by users. This document describes how the SPAKE password-authenticated key exchange can be used with 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 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 October 15, 2018.

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.
Table of Contents

1. Introduction ........................................ 2
2. Terminology ........................................... 3
3. Setup .................................................. 3
4. TLS Extensions ........................................ 4
5. Security Considerations ............................... 5
6. IANA Considerations ................................... 6
7. References ............................................. 6
   7.1. Normative References ............................... 6
   7.2. Informative References ............................. 7
8. Authors’ Addresses ..................................... 7

1. Introduction

DISCLAIMER: This is a work-in-progress draft of MLS and has not yet seen significant security analysis. It should not be used as a basis for building production systems.

In some applications, it is desirable to enable a client and server to authenticate to one another using a low-entropy pre-shared value, such as a user-entered password.

In prior versions of TLS, this functionality has been provided by the integration of the Secure Remote Password PAKE protocol (SRP) [RFC5054]. The specific SRP integration described in RFC 5054 does not immediately extend to TLS 1.3 because it relies on the Client Key Exchange and Server Key Exchange messages, which no longer exist in 1.3. At a more fundamental level, the messaging patterns required by SRP do not map cleanly to the standard TLS 1.3 handshake, which has fewer round-trips than prior versions.

TLS 1.3 itself provides a mechanism for authentication with pre-shared keys (PSKs). However, PSKs used with this protocol need to be "full-entropy", because the binder values used for authentication can be used to mount a dictionary attack on the PSK. So while the TLS 1.3 PSK mechanism is suitable for the session resumption cases for which it is specified, it cannot be used when the client and server share only a low-entropy secret.

Enabling TLS to address this use case effectively requires the TLS handshake to perform a password-authenticated key establishment (PAKE) protocol. This document describes an embedding of the SPAKE2 PAKE protocol in TLS 1.3 [I-D.irtf-cfrg-spake2] [I-D.ietf-tls-tls13]. This mechanism also applies to DTLS 1.3 [I-D.ietf-tls-dtls13], but for brevity, we will refer only to TLS throughout.
2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Setup

In order to use this protocol, a TLS client and server need to have pre-provisioned the values required to execute the SPAKE2 protocol (see Section 3.1 of [I-D.irtf-cfrg-spake2]):

- A DH group "G" of order "p*h", with "p" a large prime
- Fixed elements "M" and "N" for the group
- A hash function "H"
- A password "p"

Note that the hash function "H" might be different from the hash function associated with the ciphersuite negotiated by the two parties. The hash function "H" MUST be a hash function suitable for hashing passwords, e.g., Argon2 or scrypt [I-D.irtf-cfrg-argon2] [RFC7914].

The TLS client and server roles map to the "A" and "B" roles in the SPAKE specification, respectively. The identity of the server is the domain name sent in the "server_name" extension of the ClientHello message. The identity of the client is an opaque octet string, specified in the "spake2" ClientHello extension, defined below.

From the shared password, each party computes a shared integer "w" in the following way:

```
struct {
    opaque client_identity<0..255>;
    opaque server_name<0..255>;
    opaque password<0..255>;
} PasswordInput;

struct {
    opaque salt<0..255>;
    opaque idpass[H.length];
} PasswordWithContext;
```

- Encode the following values into a "PasswordInput" structure:
* "client_identity": The client’s identity, in the same form that is presented in the "identity" field of the "SPAKE2ClientHello" structure.

* "server_name": The server’s identity, in the same form presented in the "server_name" extension sent by the client.

* "password": The password itself

  o Use the hash function "H" with the encoded "PasswordInput" structure as input to derive an "n"-byte string, where "n" is the byte-length of "p".

  o Interpret the "n"-bit string as an integer in network byte order and let "w" be the result of reducing this integer mod "p".

Servers SHOULD store only the product "w*N" of "w" with the fixed element "N" of the group. Clients MAY compute "w" dynamically, based on the password and client and server identities for a given session.

4. TLS Extensions

A client offers to authenticate with SPAKE2 by including an "spake2" extension in its ClientHello. The content of this extension is an "SPAKE2ClientHello" value, specifying the client’s identity, where the identity matches that used in 'IdentifierAndPassword', and a key share "T". The value "T" is computed as specified in [I-D.irtf-cfrg-spake2], as "T = w*M + X", where "M" is a fixed value for the DH group and "X" is the public key of a fresh DH key pair. The format of the key share "T" is the same as for a "KeyShareEntry" value from the same group.

If a client sends the "spake2" extension, then it MAY also send the "key_share" and "pre_shared_key" extensions, to allow the server to choose an authentication mode. Unlike PSK-based authentication, however, authentication with SPAKE2 cannot be combined with the normal TLS ECDH mechanism.

```
struct {
    opaque identity<0..2^16-1>;
    opaque key_exchange<1..2^16-1>;
} SPAKE2Share;

struct {
    SPAKE2Share client_shares<0..2^16-1>;
} SPAKE2ClientHello;
```
A server that receives an "spake2" extension examines the list of client shares to see if there is one with an identity the server recognizes. If so, the server may indicate its use of SPAKE2 authentication by including an "spake2" extension in its ServerHello. The content of this extension is an "SPAKE2ServerHello" value, specifying the identity value for the password the server has selected, and the server’s key share "S". The value "S" is computed as specified in [I-D.irtf-cfrg-spake2], as "S = w*N + Y", where "N" is a fixed value for the DH group and "Y" is the public key of a fresh DH key pair. The format of the key share "S" is the same as for a "KeyShareEntry" value from the same group.

Use of SPAKE2 authentication is not inconsistent with standard certificate-based authentication of both clients and servers. Authentication are not mutually exclusive. If a server includes an "spake2" extension in its ServerHello, it may still send the Certificate and CertificateVerify messages, and/or send a CertificateRequest message to the client.

If a server uses SPAKE2 authentication, then it MUST NOT send an extension of type "key_share", "pre_shared_key", or "early_data".

struct {
    SPAKE2Share server_share;
} SPAKE2ServerHello;

Based on these messages, both the client and server can compute the shared key "K = x*(S-w*N) = y*(T-w*M)"", as specified in [I-D.irtf-cfrg-spake2]. The value "K" is then used as the "(EC)DHE" input to the TLS key schedule. The integer "w" is used as the PSK input, encoded as an integer in network byte order, using the smallest number of octets possible.

As with client authentication via certificates, the server has not authenticated the client until after it has received the client’s Finished message. When a server negotiates the use of this mechanism for authentication, it MUST NOT send application data before it has received the client’s Finished message.

5. Security Considerations

For the most part, the security properties of the password-based authentication described in this document are the same as those described in the Security Considerations of [I-D.irtf-cfrg-spake2]. The TLS Finished MAC provides the key confirmation required for the security of the protocol. Note that all of the elements covered by the example confirmation hash listed in that document are also covered by the Finished MAC:
o "A", "B", and "T" are included via the ClientHello

o "S" via the ServerHello

o "K", and "w" via the TLS key schedule

The "x" and "y" values used in the SPAKE2 protocol MUST have the same ephemerality properties as the key shares sent in the "key_shares" extension. This ensures that TLS sessions using SPAKE2 have the same forward secrecy properties as sessions using the normal TLS (EC)DH mechanism.

The mechanism described above does not provide protection for the client’s identity, in contrast to TLS client authentication with certificates.

[[ XXX(rlb@ipv.sx): Or maybe there’s some HRR dance we could do. For example: Server provides a key share in HRR, client does ECIES on identity. ]]

6. IANA Considerations

This document requests that IANA add a value to the TLS ExtensionType Registry with the following contents:

+-------+----------------+---------+-----------+
| Value | Extension Name | TLS 1.3 | Reference |
+-------+----------------+---------+-----------+
| TBD   | spake2         |  CH, SH |  RFC XXXX |
+-------+----------------+---------+-----------+

[[ RFC EDITOR: Please replace "TBD" in the above table with the value assigned by IANA, and replace "XXXX" with the RFC number assigned to this document. ]]

7. References

7.1. Normative References

[I-D.ietf-tls-dtls13]
7.2. Informative References


Authors' Addresses

Richard Barnes
Cisco
Email: rlb@ipv.sx

Owen Friel
Cisco
Email: ofriel@cisco.com