TLS
Internet-Draft
Intended status: Standards Track
Expires: May 17, 2015

S. Santesson
3xA Security AB
H. Tschofenig
ARM Ltd.

November 13, 2014

Transport Layer Security (TLS) Cached Information Extension
draft-ietf-tls-cached-info-17.txt

Abstract

Transport Layer Security (TLS) handshakes often include fairly static information, such as the server certificate and a list of trusted Certification Authorities (CAs). This information can be of considerable size, particularly if the server certificate is bundled with a complete certificate chain (i.e., the certificates of intermediate CAs up to the root CA).

This document defines an extension that allows a TLS client to inform a server of cached information, allowing the server to omit already available information.

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 May 17, 2015.

Table of Contents

1. Introduction

Reducing the amount of information exchanged during a Transport Layer Security handshake to a minimum helps to improve performance in environments where devices are connected to a network with a low bandwidth, and lossy radio technology. With Internet of Things such environments exist, for example, when smart objects are connected using a low power IEEE 802.15.4 radio or via Bluetooth Smart. For more information about the challenges with smart object deployments please see [RFC6574].

This specification defines a TLS extension that allows a client and a server to exclude transmission information cached in an earlier TLS handshake.

A typical example exchange may therefore look as follows. First, the client and the server executes the usual TLS handshake. The client may, for example, decide to cache the certificate provided by the server. When the TLS client connects to the TLS server some time in

the future, without using session resumption, it then attaches the cached_info extension defined in this document to the client hello message to indicate that it had cached the certificate, and it provides the fingerprint of it. If the server's certificate has not changed then the TLS server does not need to send its' certificate and the corresponding certificate list again. In case information has changed, which can be seen from the fingerprint provided by the client, the certificate payload is transmitted to the client to allow the client to update the cache.

2. Terminology

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

This document refers to the TLS protocol but the description is equally applicable to DTLS as well.

3. Cached Information Extension

This document defines a new extension type (cached_info(TBD)), which is used in client hello and server hello messages. The extension type is specified as follows.

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

The extension_data field of this extension, when included in the client hello, MUST contain the CachedInformation structure. The client MUST NOT send multiple CachedObjects with the same CachedInformationType.

```
enum {
     certificate_list(1), certificate_authorities(2) (255)
} CachedInformationType;

struct {
     select (type) {
       case client:
         CachedInformationType type;
         HashAlgorithm hash;
         opaque hash_value<1..255>;
       case server:
         CachedInformationType type;
     } body;
} CachedObject;

struct {
     CachedObject cached_info<1..2^16-1>;
} CachedInformation;
```

This document establishes a registry for CachedInformationType types; additional values can be added following the policy described in Section 7.

3.1. Certificate_list Fingerprint

When the CachedInformationType identifies a certificate_list, then the hash_value field MUST include the hash calculated over the certificate_list element of the Certificate payload provided by the TLS server in an earlier exchange, excluding the three length bytes of the certificate_list vector.

3.2. Certificate_authorities Fingerprint

When the CachedInformationType identifies a certificate_authorities, then the hash_value MUST include a hash calculated over CertificateRequest payload provided by the TLS server in an earlier exchange, excluding the msg_type and length field.

3.3. Fingerprint Hash Algorithm

The hash algorithm used to calculate hash values is conveyed in the 'hash' field of the CachedObject element. The list of registered hash algorithms can be found in the TLS HashAlgorithm Registry, which was created by RFC 5246 [RFC5246]. The value zero (0) for 'none' and one (1) for 'md5' is not an allowed choice for a hash algorithm and MUST NOT be used.

4. Exchange Specification

Clients supporting this extension MAY include the "cached_info" extension in the (extended) client hello. If the client includes the extension then it MUST contain one or more CachedObject attributes. Clients and servers MUST NOT include more than one CachedObject attribute per info type.

A server supporting this extension MAY include the "cached_info" extension in the (extended) server hello. By returning the "cached_info" extension the server indicates that it supports the cached info types. For each indicated cached info type the server MUST alters the transmission of respective payloads, as specified for each type. If the server includes the extension it MUST only include CachedObjects of a type also supported by the client (as expressed in the client hello).

Note that the client includes a fingerprint of the cached information to give the server enough information to determine whether cached information is stale. If the server supports this specification and notices a mismatch between the data cached by the client and its own information then the server MUST include the information in full and MUST NOT list the respective item in the "cached_info" extension.

Note: Clients may cache multiple data items for a single server if those servers are part of a hosting environment. To allow the client to select the appropriate information from the cached it is RECOMMENDED that the client uses information from the Server Name Indication [RFC6066].

Following a successful exchange of the "cached_info" extensions in the client and server hello, the server alters sending the corresponding handshake message. How information is altered from the handshake messages is defined per cached info type. Section 4.1 and Section 4.2 defines the syntax of the fingerprinted information.

The handshake protocol MUST proceed using the information as if it was provided in the handshake protocol. Since the Finished message is calculated over the exchanged data it will also include the hash of the cached data.

4.1. Omitting the Certificate List

When an object of type 'certificate_list' is provided in the client hello, the server MAY replace the list of certificates with an empty sequence with an actual length field of zero (=empty vector).

The original handshake message syntax is defined in RFC 5246 [RFC5246] and has the following structure:

```
opaque ASN.1Cert<1..2^24-1>;

struct {
    ASN.1Cert certificate_list<0..2^24-1>;
} Certificate;
```

Note that [RFC7250] allows the certificate payload to contain only the SubjectPublicKeyInfo instead of the full information typically found in a certificate. Hence, when this specification is used in combination with [RFC7250] and the negotiated certificate type is a raw public key then the TLS server omits sending a Certificate payload that contains an ASN.1Cert structure of the SubjectPublicKeyInfo.

4.2. Omitting the Trusted Certificate Authorities

When a fingerprint for an object of type 'certificate_authorities' is provided in the client hello, the server MAY replace the CertificateRequest message with an empty sequence with an actual length field of zero.

The original handshake message syntax is defined in RFC 5246 [RFC5246] and has the following structure:

```
opaque DistinguishedName<1..2^16-1>;

struct {
    ClientCertificateType certificate_types<1..2^8-1>;
    SignatureAndHashAlgorithm
      supported_signature_algorithms<2^16-1>;
    DistinguishedName certificate_authorities<0..2^16-1>;
} CertificateRequest;
```

5. Example

Figure 1 illustrates an example exchange using the TLS cached info extension. In the normal TLS handshake exchange shown in flow (A) the TLS server provides its certificate in the Certificate payload to the client, see step [1]. This allows the client to store the certificate for future use. After some time the TLS client again interacts with the same TLS server and makes use of the TLS cached info extension, as shown in flow (B). The TLS client indicates support for this specification via the "cached_info" extension, see

[2], and indicates that it has stored the 'certificate_list' from the earlier exchange. With [3] the TLS server acknowledges the supports of this specification and informs the client that it alterned the content of the certificate payload (see [4], as described in Section 4.1).

```
   (A) Initial (full) Exchange

   ClientHello  ->
                          <-  ServerHello
                              Certificate* // [1]
                              ServerKeyExchange*
                              CertificateRequest*
                              ServerHelloDone

   Certificate*
   ClientKeyExchange
   CertificateVerify*
   [ChangeCipherSpec]
   Finished                   ->

                          <- [ChangeCipherSpec]
                             Finished

   Application Data        <------->     Application Data


   (B) TLS Cached Extension Usage

   ClientHello
   cached_info=(certificate_list)   -> // [2]
                          <-  ServerHello
                              cached_info=
                              (certificate_list) // [3]
                              Certificate* // [4]
                              ServerKeyExchange*
                              CertificateRequest*
                              ServerHelloDone

   Certificate*
   ClientKeyExchange
   CertificateVerify*
   [ChangeCipherSpec]
   Finished                   ->

                          <- [ChangeCipherSpec]
                             Finished

   Application Data        <------->     Application Data
```

Figure 1: Example Message Exchange

6. Security Considerations

This specification defines a mechanism to reference stored state using a fingerprint. Sending a fingerprint of cached information in an unencrypted handshake, as the client and server hello is, may allow an attacker or observer to correlate independent TLS exchanges. While some information elements used in this specification, such as server certificates, are public objects and usually not sensitive in this regard, others may be. Those who implement and deploy this specification should therefore make an informed decision whether the cached information is inline with their security and privacy goals. In case of concerns, it is advised to avoid sending the fingerprint of the data objects in clear.

The hash algorithm used in this specification is required to have have strong collision resistance.

7. IANA Considerations

7.1. New Entry to the TLS ExtensionType Registry

IANA is requested to add an entry to the existing TLS ExtensionType registry, defined in RFC 5246 [RFC5246], for cached_info(TBD) defined in this document.

7.2. New Registry for CachedInformationType

IANA is requested to establish a registry for TLS CachedInformationType values. The first entries in the registry are

- certificate_list(1)
- certificate_authorities(2)

The policy for adding new values to this registry, following the terminology defined in RFC 5226 [RFC5226], is as follows:

- 0-63 (decimal): Standards Action
- 64-223 (decimal): Specification Required
- 224-255 (decimal): reserved for Private Use

8. Acknowledgments

We would like to thank the following persons for your detailed document reviews:

- Paul Wouters and Nikos Mavrogiannopoulos (December 2011)
- Rob Stradling (February 2012)
- Ondrej Mikle (in March 2012)
- Ilari Liusvaara, Adam Langley, and Eric Rescorla (in July 2014)

Additionally, we would like to thank the TLS working group chairs, Sean Turner and Joe Salowey, as well as the responsible security area director, Stephen Farrell, for his support.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3874] Housley, R., "A 224-bit One-way Hash Function: SHA-224", RFC 3874, September 2004.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

9.2. Informative References

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6574] Tschofenig, H. and J. Arkko, "Report from the Smart Object Workshop", RFC 6574, April 2012.

[RFC7250] Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, June 2014.

Authors' Addresses

Stefan Santesson
3xA Security AB
Scheelev. 17
Lund 223 70
Sweden

Email: sts@aaa-sec.com

Hannes Tschofenig
ARM Ltd.
Hall in Tirol 6060
Austria

Email: Hannes.tschofenig@gmx.net
URI: http://www.tschofenig.priv.at