TLS
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2013

S. Santesson
3xA Security AB
H. Tschofenig
Nokia Siemens Networks

July 15, 2012

Transport Layer Security (TLS) Cached Information Extension
draft-ietf-tls-cached-info-12.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 path (including all intermediary certificates up to the trust anchor public key).

This document defines an extension that omits the exchange of already available information. The TLS client informs a server of cached information, for example from a previous TLS handshake, allowing the server to omit the 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 January 16, 2013.

Table of Contents

1. Introduction

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 path (including all intermediary certificates up to the trust anchor public key).

Optimizing the exchange of information to a minimum helps to improve performance in environments where devices are connected to a network with characteristics like low bandwidth, high latency and high loss rate. These types of networks exist, for example, when smart objects are connected using a low power IEEE 802.15.4 radio. 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 of cached information from the TLS handshake.

A typical example exchange may therefore look as follows. First, the TLS exchange executes the usual TLS handshake. It may decide to store the certificate provided by the server for a future exchange. When the TLS client then connects to the TLS server some time in the future, without using session resumption, it then attaches the cached_information 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 had not changed then the TLS server does not need to send the full certificate to the client again. In case the information had changed, the certificate payload is transmitted to the client to allow the client to update it’s state information.

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].

3. Cached Information Extension

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

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

The extension_data field of this extension, when included in the client hello, MUST contain the CachedInformation structure.

```
enum {
     certificate_chain(1), trusted_cas(2) (255)
} CachedInformationType;

struct {
     CachedInformationType type;
     HashAlgorithm hash;
     opaque hash_value<1..255>;
} CachedObject;

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

When the CachedInformationType identifies a certificate_chain, 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.

When the CachedInformationType identifies a trusted_cas, then the hash_value MUST include a hash calculated over the certificate_authorities element of the CertificateRequest payload provided by the TLS server in an earlier exchange, excluding the two length bytes of the certificate_authorities vector.

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' is not an allowed choice for a hash algorithm and MUST NOT be used.

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

4. Exchange Specification

Clients supporting this extension MAY include the "cached_information" extension in the (extended) client hello, which MAY contain zero or more CachedObject attributes.

Server supporting this extension MAY include the "cached_information" extension in the (extended) server hello, which MAY contain one or more CachedObject attributes. By returning the "cached_information" extension the server indicates that it supports caching of each present CachedObject that matches the specified hash value. The server MAY support other cached objects that are not present in the extension.

Note: Clients may need the ability to cache different values depending on other information in the Client Hello that modify what values the server uses, in particular the Server Name Indication [RFC6066] value.

Following a successful exchange of "cached_information" extensions, the server MAY send fingerprints of the cached information in the handshake exchange as a replacement for the exchange of the full data. 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. The Finished message MUST be calculated over the actual data exchanged in the handshake protocol. That is, the Finished message will be calculated over the hash values of cached information objects and not over the cached information that were omitted from transmission.

The server MUST NOT include more than one fingerprint for a single information element, i.e., at maximum only one CachedObject structure per replaced information is provided.

4.1. Fingerprint of the Certificate Chain

When an object of type ’certificate_chain’ is provided in the client hello, the server MAY send a fingerprint instead of the complete certificate chain as shown below.

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;
```

By using the extension defined in this document the following information is sent:

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

The opaque ASN.1Cert structure is replaced with the CachedObject structure defined in this document.

Note: [I-D.ietf-tls-oob-pubkey] allows a PKIX certificate containing only the SubjectPublicKeyInfo instead of the full information typically found in a certificate. Hence, when this specification is used in combination with [I-D.ietf-tls-oob-pubkey] and the negotiated certificate type is a raw public key then the TLS server sends the hashed Certificate payload that contains a ASN.1Cert structure of the SubjectPublicKeyInfo.

4.2. Fingerprint for Trusted CAs

When a hash for an object of type 'trusted_cas' is provided in the client hello, the server MAY send a fingerprint instead of the complete certificate authorities information as shown below.

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;
```

By using the extension defined in this document the following information is sent:

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

The opaque DistinguishedName structure is replaced with the CachedObject structure defined in this document.

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_information extension, see [2], and indicates that it has stored the certificate_chain from the earlier exchange. With [3] the TLS server indicates that it also supports this specification and informs the client that it also supports caching of other objects beyond the 'certificate_chain', namely 'trusted_cas' (also defined in this document), and the 'foo-bar' extension (i.e., an imaginary extension that yet needs to be defined). With [4] the TLS server provides the fingerprint of the certificate chain as described in Section 4.1.

```
(A) Initial (full) Exchange

client_hello  ->
                        <-  server_hello,
                            certificate, // [1]
                            server_key_exchange,
                            server_hello_done

client_key_exchange,
change_cipher_spec,
finished                    ->

                        <- change_cipher_spec,
                           finished

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


(B) TLS Cached Extension Usage

client_hello,
cached_information=(certificate_chain)   -> // [2]
                        <-  server_hello,
                            cached_information= // [3]
                               (certificate_chain, trusted_cas, foo-bar)
                            certificate, // [4]
                            server_key_exchange,
                            server_hello_done

client_key_exchange,
change_cipher_spec,
finished                    ->

                        <- change_cipher_spec,
                           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. The hash algorithm used in this specification is required to have reasonable random properties in order to provide reasonably unique identifiers. There is no requirement that this hash algorithm must have strong collision resistance.

Caching information in an encrypted handshake (such as a renegotiated handshake) and sending a hash of that cached information in an unencrypted handshake might introduce integrity or data disclosure issues as it enables an attacker to identify if a known object (such as a known server certificate) has been used in previous encrypted handshakes. Information object types defined in this specification, such as server certificates, are public objects and usually not sensitive in this regard, but implementers should be aware if any cached information are subject to such security concerns and in such case SHOULD NOT send a hash over encrypted data in unencrypted handshake.

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_information(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_chain(1)
- trusted_cas(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)

Additionally, we would like to thank the TLS working group chairs, Eric Rescorla and Joe Salowey, as well as the security area directors, Sean Turner and Stephen Farrell, for their feedback and 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

[I-D.ietf-tls-oob-pubkey] Wouters, P., Gilmore, J., Weiler, S., Kivinen, T., and H. Tschofenig, "TLS Out-of-Band Public Key Validation", draft-ietf-tls-oob-pubkey-03 (work in progress), April 2012.

[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.

Authors' Addresses

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

Email: sts@aaa-sec.com

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at