TLS
Internet-Draft
Obsoletes: 6347 (if approved)
Intended status: Standards Track
Expires: June 30, 2018

E. Rescorla, Ed.
RTFM, Inc.
H. Tschofenig, Ed.
ARM Limited
T. Fossati
Nokia
T. Gondrom
Huawei
December 27, 2017

The Datagram Transport Layer Security (DTLS) Connection Identifier
draft-ietf-tls-dtls-connection-id-00

Abstract

This document specifies the "Connection ID" concept for the Datagram Transport Layer Security (DTLS) protocol, version 1.2 and version 1.3.

A Connection ID is an identifier carried in the record layer header that gives the recipient additional information for selecting the appropriate security association. In "classical" DTLS, selecting a security association of an incoming DTLS record is accomplished with the help of the 5-tuple. If the source IP address and/or source port changes during the lifetime of an ongoing DTLS session then the receiver will be unable to locate the correct security context.

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 June 30, 2018.

Copyright Notice

Copyright (c) 2017 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 (http://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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction 3
2. Conventions and Terminology 3
3. The "connection_id" Extension 3
4. Post-Handshake Messages 5
5. Record Layer Extensions 5
6. Examples 6
7. Security and Privacy Considerations 8
8. IANA Considerations 9
9. References 9
    9.1. Normative References 9
    9.2. Informative References 10
Appendix A. History 11
Appendix B. Working Group Information 11
Appendix C. Contributors 11
Authors' Addresses 12

1. Introduction

The Datagram Transport Layer Security (DTLS) protocol was designed for securing connection-less transports, like UDP. DTLS, like TLS, starts with a handshake, which can be computationally demanding (particularly when public key cryptography is used). After a successful handshake, symmetric key cryptography is used to apply data origin authentication, integrity and confidentiality protection. This two-step approach allows to amortize the cost of the initial handshake to subsequent application data protection. Ideally, the second phase where application data is protected lasts over a longer period of time since the established keys will only need to be updated once the key lifetime expires.

In the current version of DTLS, the IP address and port of the peer is used to identify the DTLS association. Unfortunately, in some cases, such as NAT rebinding, these values are insufficient. This is a particular issue in the Internet of Things when the device needs to enter extended sleep periods to increase the battery lifetime and is therefore subject to rebinding. This leads to connection failure, with the resulting cost of a new handshake.

This document defines an extension to DTLS to add a connection ID to each DTLS record. The presence of the connection ID is negotiated via a DTLS extension. It also defines a DTLS 1.3 post-handshake message to change connection ids.

2. Conventions and 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 RFC 2119 [RFC2119].

The reader is assumed to be familiar with the DTLS specifications since this document defines an extension to DTLS 1.2 and DTLS 1.3.

3. The "connection_id" Extension

This document defines a new extension type (connection_id(TBD)), which is used in ClientHello and ServerHello messages.

The extension type is specified as follows.

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

The extension_data field of this extension, when included in the ClientHello, MUST contain the CID structure, which contains the CID which the client wishes the server to use when sending messages towards it. A zero-length value indicates that the client is prepared to send with a connection ID but does not wish the server to use one when sending (alternately, this can be interpreted as the client wishes the server to use a zero-length CID; the result is the same).

```
struct {
    opaque cid<0..2^8-1>;
} ConnectionId;
```

A server which is willing to use CIDs will respond with its own "connection_id" extension, containing the CID which it wishes the client to use when sending messages towards it. A zero-length value indicates that the server will send with the client's CID but does not wish the client to use a CID (or again, alternately, to use a zero-length CID).

When a session is resumed, the "connection_id" extension is negotiated afresh, not retained from previous connections in the session.

This is effectively the simplest possible design that will work. Previous design ideas for using cryptographically generated session ids, either using hash chains or public key encryption, were dismissed due to their inefficient designs. Note that a client always has the chance to fall-back to a full handshake or more precisely to a handshake that uses session resumption (DTLS 1.2 language) or to a PSK-based handshake using the ticket-based approach.

Because each party sends in the extension_data the value that it will receive as a connection identifier in encrypted records, it is possible for an endpoint to use a globally constant length for such connection identifiers. This can in turn ease parsing and connection lookup, for example by having the length in question be a compile-time constant. Note that such implementations must still be able to send other length connection identifiers to other parties.

In DTLS 1.2, connection ids are exchanged at the beginning of the DTLS session only. There is no dedicated "connection id update" message that allows new connection ids to be established mid-session, because DTLS 1.2 in general does not allow post-handshake messages that do not themselves begin other handshakes. In DTLS 1.3, which does allow such messages, we use post-handshake message to update the connection ID Section 4 and to request new IDs.

DTLS 1.2 peers switch to the new record layer format when encryption is enabled. The same is true for DTLS 1.3 but since the DTLS 1.3 enables encryption early in the handshake phase the connection ID will be enabled earlier. For this reason, the connection ID needs to go in the DTLS 1.3 ServerHello.

4. Post-Handshake Messages

In DTLS 1.3, if the client and server have negotiated the "connection_id" extension, either side can send a new connection ID which it wishes the other side to use in a NewConnectionId message:

```
enum {
    cid_immediate(0), cid_spare(1), (255)
} ConnectionIdUsage;

struct {
    opaque cid<0..2^8-1>;
    ConnectionIdUsage usage;
} NewConnectionId;
```

cid Indicates the CID which the sender wishes the peer to use.

usage Indicates whether the new CID should be used immediately or is a spare. If usage is set to "cid_immediate", then the new CID MUST be used immediately for all future records. If it is set to "cid_spare", then either CID MAY be used, as described in Section 7.

If the client and server have negotiated the "connection_id" extension, either side can request a new CID using the RequestConnectionId message.

```
struct {
} RequestConnectionId;
```

Endpoints SHOULD respond to RequestConnectionId by sending a NewConnectionId with usage "cid_spare" as soon as possible. Note that an endpoint MAY ignore requests which it considers excessive (though they MUST be ACKed as usual).

5. Record Layer Extensions

This extension is applicable for use with DTLS 1.2 and DTLS 1.3. This extension can be used with the optimized DTLS 1.3 record layer format.

Figure 1 and Figure 2 illustrate the record formats of DTLS 1.2 and DTLS 1.3, respectively.

```
struct {
   ContentType type;
   ProtocolVersion version;
   uint16 epoch;
   uint48 sequence_number;
   opaque cid[cid_length];               // New field
   uint16 length;
   select (CipherSpec.cipher_type) {
      case block:  GenericBlockCipher;
      case aead:   GenericAEADCipher;
   } fragment;
} DTLSCiphertext;
```

Figure 1: DTLS 1.2 Record Format with Connection ID

```
struct {
   opaque content[DTLSPlaintext.length];
   ContentType type;
   uint8 zeros[length_of_padding];
} DTLSInnerPlaintext;

struct {
   ContentType opaque_type = 23; /* application_data */
   ProtocolVersion legacy_record_version = {254,253); // DTLSv1.2
   uint16 epoch;                         // DTLS-related field
   uint48 sequence_number;               // DTLS-related field
   opaque cid[cid_length];               // New field
   uint16 length;
   opaque encrypted_record[length];
} DTLSCiphertext;
```

Figure 2: DTLS 1.3 Record Format with Connection ID

Besides the "cid" field, all other fields are defined in the DTLS 1.2 and DTLS 1.3 specifications.

Note that for both record formats, it is not possible to parse the records without knowing if the connection ID is in use and how long it is.

6. Examples

Below is an example exchange for DTLS 1.3 using a single connection id in each direction.

```
   Client                                             Server
   ------                                             ------

   ClientHello
   (connection_id=5)
                               -------->


                               <--------       HelloRetryRequest
                                                        (cookie)

   ClientHello                 -------->
   (connection_id=5)
     +cookie

                               <--------             ServerHello
                                             (connection_id=100)
                                             EncryptedExtensions
                                                         (cid=5)
                                                     Certificate
                                                         (cid=5)
                                               CertificateVerify
                                                         (cid=5)
                                                        Finished
                                                         (cid=5)

   Certificate                -------->
   (cid=100)
   CertificateVerify
   (cid=100)
   Finished
   (cid=100)
                              <--------                      Ack
                                                         (cid=5)

   Application Data           ========>
   (cid=100)
                              <========         Application Data
                                                         (cid=5)
```

Figure 3: Example DTLS 1.3 Exchange with Connection IDs

Below is an example exchange for DTLS 1.2 using a connection id used uni-directionally from the client to the server.

```
   Client                                             Server
   ------                                             ------

   ClientHello
   (connection_id=empty)
                               -------->


                               <--------      HelloVerifyRequest
                                                        (cookie)

   ClientHello                 -------->
   (connection_id=empty)
     +cookie

                               <--------             ServerHello
                                             (connection_id=100)
                                                     Certificate
                                               ServerKeyExchange
                                              CertificateRequest
                                                 ServerHelloDone

   Certificate                 -------->
   ClientKeyExchange
   CertificateVerify
   [ChangeCipherSpec]
   Finished
   (cid=100)
                               <--------      [ChangeCipherSpec]
                                                        Finished

   Application Data           ========>
   (cid=100)
                              <========         Application Data
```

Figure 4: Example DTLS 1.2 Exchange with Connection IDs

7. Security and Privacy Considerations

The connection id replaces the previously used 5-tuple and, as such, introduces an identifier that remains persistent during the lifetime of a DTLS connection. Every identifier introduces the risk of linkability, as explained in [RFC6973].

In addition, endpoints can use the connection ID to attach arbitrary metadata to each record they receive. This may be used as a mechanism to communicate per-connection to on-path observers. There is no straightforward way to address this with connection IDs that

contain arbitrary values; implementations concerned about this SHOULD refuse to use connection ids.

An on-path adversary, who is able to observe the DTLS 1.2 protocol exchanges between the DTLS client and the DTLS server, is able to link the observed payloads to all subsequent payloads carrying the same connection id pair (for bi-directional communication). In DTLS 1.3, it is possible to provide new encrypted connection IDs, though of course those IDs are immediately used on the wire. Without multi-homing and mobility the use of the connection id is not different to the use of the 5-tuple.

With multi-homing, an adversary is able to correlate the communication interaction over the two paths, which adds further privacy concerns. In order to prevent this, implementations SHOULD attempt to use fresh connection IDs whenever they change local addresses or ports (though this is not always possible to detect). In DTLS 1.3, The RequestConnectionId message can be used to ask for new IDs in order to ensure that you have a pool of suitable IDs.

This document does not change the security properties of DTLS 1.2 [RFC6347] and DTLS 1.3 [I-D.ietf-tls-dtls13]. It merely provides a more robust mechanism for associating an incoming packet with a stored security context.

[[OPEN ISSUE: Sequence numbers leak connection IDs. We need to update the document to address this. One possibility would be the technique documented in https://quicwg.github.io/base-drafts/draft-ietf-quic-transport.html#packet-number-gap.]]

8. IANA Considerations

IANA is requested to allocate an entry to the existing TLS "ExtensionType Values" registry, defined in [RFC5246], for connection_id(TBD) defined in this document.

IANA is requested to allocate two values in the "TLS Handshake Type" registry, defined in [RFC5246], for request_connection_id (TBD), and new_connection_id (TBD), as defined in this document.

9. References

9.1. Normative References

[I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-22 (work in progress), November 2017.

[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>.

[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>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

9.2. Informative References

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

9.3. URIs

[1] mailto:tls@ietf.org

Appendix A. History

RFC EDITOR: PLEASE REMOVE THE THIS SECTION

draft-rescorla-tls-dtls-connection-id-00

- Initial version

Appendix B. Working Group Information

The discussion list for the IETF TLS working group is located at the e-mail address tls@ietf.org [1]. Information on the group and information on how to subscribe to the list is at https://www1.ietf.org/mailman/listinfo/tls

Archives of the list can be found at: https://www.ietf.org/mail-archive/web/tls/current/index.html

Appendix C. Contributors

Many people have contributed to this specification since the functionality has been highly desired by the IoT community. We would like to thank the following individuals for their contributions in earlier specifications:

* Nikos Mavrogiannopoulos
  RedHat
  nmav@redhat.com

Additionally, we would like to thank Yin Xinxing (Huawei), Tobias Gondrom (Huawei), and the Connection ID task force team members:

- Martin Thomson (Mozilla)

- Christian Huitema (Private Octopus Inc.)

- Jana Iyengar (Google)

- Daniel Kahn Gillmor (ACLU)

- Patrick McManus (Sole Proprietor)

- Ian Swett (Google)

- Mark Nottingham (Fastly)

Finally, we want to thank the IETF TLS working group chairs, Joseph Salowey and Sean Turner, for their patience, support and feedback.

Authors' Addresses

Eric Rescorla (editor)
RTFM, Inc.

EMail: ekr@rtfm.com

Hannes Tschofenig (editor)
ARM Limited

EMail: hannes.tschofenig@arm.com

Thomas Fossati
Nokia

EMail: thomas.fossati@nokia.com

Tobias Gondrom
Huawei

EMail: tobias.gondrom@gondrom.org