Connection Identifiers for DTLS 1.2
draft-ietf-tls-dtls-connection-id-02

Abstract

This document specifies the Connection ID construct for the Datagram Transport Layer Security (DTLS) protocol version 1.2.

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 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 April 25, 2019.
Table of Contents

1. Introduction ........................................... 3
2. Conventions and Terminology ................................. 3
3. The "connection_id" Extension ............................... 3
4. Record Layer Extensions .................................. 5
5. Record Payload Protection .................................. 5
6. Examples .................................................. 6
7. Security and Privacy Considerations ......................... 7
8. IANA Considerations ....................................... 8
9. References ............................................... 8
   9.1. Normative References .................................. 8
   9.2. Informative References ................................. 9
Appendix A. History .......................................... 10
Appendix B. Working Group Information .......................... 10
Appendix C. Contributors ...................................... 10
Authors’ Addresses ........................................... 11
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 endpoints to amortize the cost of the initial handshake across 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 are 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 devices enter extended sleep periods to increase their battery lifetime. The NAT rebinding leads to connection failure, with the resulting cost of a new handshake.

This document defines an extension to DTLS to add a connection ID (CID) to the DTLS record layer. The presence of the connection ID is negotiated via a DTLS extension.

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.

The reader is assumed to be familiar with DTLS [RFC6347].

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.

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

The extension_data field of this extension, when included in the ClientHello, MUST contain the ConnectionId structure, which carries
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 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.

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. Implementations which want to use variable-length
CIDs are responsible for constructing the CID in such a way that its
length can be determined on reception. Note that such
implementations must still be able to send other length connection
identifiers to other parties.

In DTLS, 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 in general does not allow TLS 1.3-style post-handshake messages
that do not themselves begin other handshakes. DTLS peers switch to
the new record layer format when encryption is enabled.
4. Record Layer Extensions

This extension is applicable for use with DTLS 1.2 and below. Figure 1 illustrates the record format. [I-D.ietf-tls-dtls13] specifies how to carry the CID in a DTLS 1.3 record.

```
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

Note that for both record formats, it is not possible to parse the records without knowing how long the Connection ID is.

In order to allow a receiver to determine whether a record has CID or not, connections which have negotiated this extension use a distinguished record type tls12_cid(25). Use of this content type has the following two implications:

- The CID field is present
- The true content type is inside the encryption envelope, as described below.

5. Record Payload Protection

When CID is being used, the DTLSCompressed value is first wrapped along with the true content type and padding into a DTLSWrappedCompressed value prior to encryption. The DTLSWrappedCompressed value is then encrypted.

```
struct {
    opaque compressed[TLSCompressed.length];
    ContentType type;
    uint8 zeros[length_of_padding];
} DTLSWrappedCompressed;
```

compressed  The value of DTLSCompressed.fragment
type The true content type.

zeroes Padding, as defined in [RFC8446].

In addition, the CID value is included in the MAC calculation for the DTLS record as shown below. The MAC algorithm described in Section 4.1.2.1 of [RFC6347] and Section 6.2.3.1 of [RFC5246] is extended as follows:

```
MAC(MAC_write_key, DTLSCompressed.epoch +
    DTLSCompressed.sequence_number +
    tls12_cid +
    DTLSCompressed.version +
    cid_length + // New input
    cid + // New input
    DTLSWrappedCompressed.length +
    DTLSWrappedCompressed.fragment);
```

where "+" denotes concatenation.

6. Examples

Figure 2 shows an example exchange where a connection id is used unidirectionally from the client to the server.
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 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). Without multi-homing or 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.

Importantly, the sequence number makes it possible for a passive attacker to correlate packets across CID changes. Thus, even if a client/server pair do a rehandshake to change CID, that does not provide much privacy benefit.

This document does not change the security properties of DTLS [RFC6347]. It merely provides a more robust mechanism for associating an incoming packet with a stored security context.

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 the following new values in the "TLS ContentType Registry":

- alert_with_cid(25)
- handshake_with_cid(26)
- application_data_with_cid(27)
- heartbeat_with_cid(28)

9. References

9.1. Normative References


9.2. Informative References


9.3. URIs

[1] mailto:tls@ietf.org


Appendix A. History

RFC EDITOR: PLEASE REMOVE THIS SECTION

draft-ietf-tls-dtls-connection-id-02

- Move to internal content types a la DTLS 1.3.
draft-ietf-tls-dtls-connection-id-01

- Remove 1.3 based on the WG consensus at IETF 101
draft-ietf-tls-dtls-connection-id-00

- Initial working group version (containing a solution for DTLS 1.2 and 1.3)
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 [2]

Archives of the list can be found at: https://www.ietf.org/mail-archive/web/tls/current/index.html [3]

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.)
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