

TLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 17, 2017

N. Mavrogiannopoulos
RedHat
H. Tschofenig
ARM
T. Fossati
Nokia
May 16, 2017

Datagram Transport Transport Layer Security (DTLS) Transport-Agnostic
Security Association Extension
draft-mavrogiannopoulos-tls-cid-01

Abstract

This memo proposes a new Datagram Transport Transport Layer Security (DTLS) extension for DTLS 1.2 that provides the ability to negotiate, during handshake, a transport independent identifier that is unique per security association. This identifier effectively decouples the DTLS session from the underlying transport protocol, allowing the same security association to be migrated across different instances of the same transport, or to a completely different transport.

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 November 17, 2017.

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

Internet-Draft

DTLS ta_sa Extension

May 2017

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.	Conventions used in this document	3
3.	Transport Agnostic Security Association Extension	4
3.1.	Extended Client Hello	4
3.2.	Extended Server Hello	5
3.3.	Handling unknown CIDs	6
3.4.	Wire Format Changes	6
3.5.	De-duplication Algorithm	7
4.	Clashing HOTP CIDs	7
5.	Security Considerations	8
6.	IANA Considerations	8
7.	Acknowledgments	8
8.	References	8
8.1.	Normative References	9
8.2.	Informative References	9
	Authors' Addresses	10

[1.](#) Introduction

DTLS security context demultiplexing is done via the 5-tuple. Therefore, the security association needs to be re-negotiated from scratch whenever the transport identifiers change. For example, when moving the network attachment from WLAN to a cellular connection, or when the IP address of the IoT devices changes during a sleep cycle. A NAT device may also modify the source UDP port after a short idle period. In such cases, there is not enough information in the DTLS record header for a server that is handling multiple concurrent sessions to associate the new address to an existing client.

This memo proposes a new TLS extension [[RFC6066](#)] for DTLS 1.2 that provides the ability to negotiate, at handshake time, a transport independent identifier that is unique per security association. We call this identifier Connection ID (CID). Its function is to effectively decouple the DTLS session from the underlying transport

protocol, allowing the same DTLS security association to be migrated across different instances of the same transport, or even to a completely different transport - e.g., from UDP to GSM-SMS as showed in Figure 1.

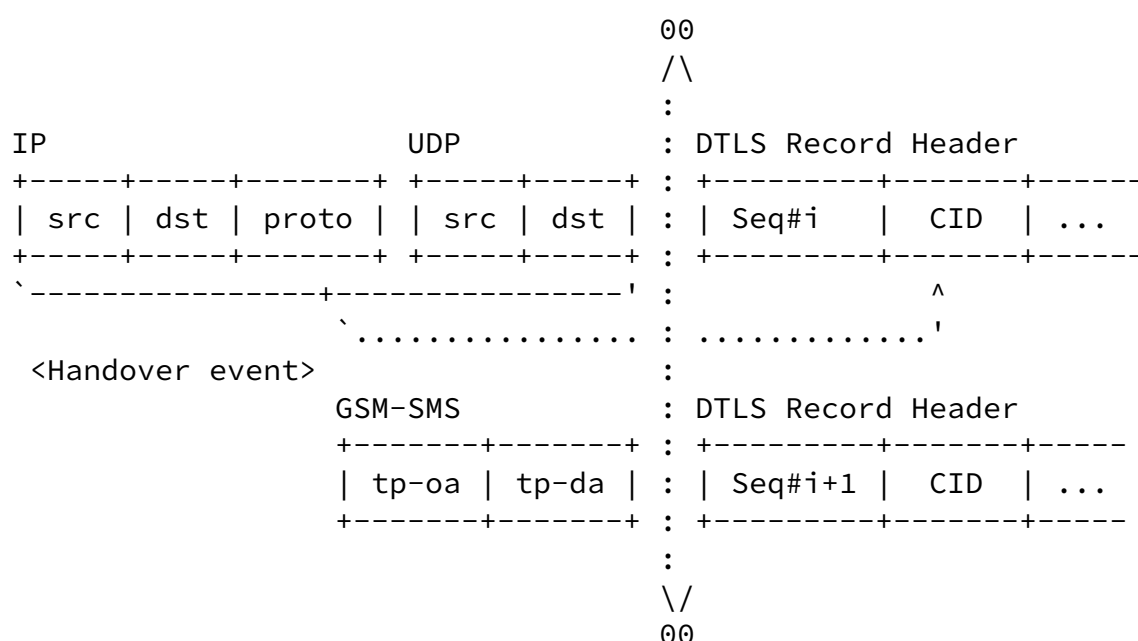


Figure 1: Transparent Handover of DTLS Session

We present two methods for producing the CID: the first uses a single value generated unilaterally by the server which is fixed throughout the session, whereas the second provides a sequence of identifiers that are created using a HMAC-based OTP algorithm [RFC4226] keyed with a per-session shared secret (see [Section 3.1](#) for details). The latter allows a client to shift to a new identifier, for example when switching networks, and is intended as a mechanism to counteract tracking by third party observers. However, it must be noted that this is not generally applicable as a tracking-protection measure: in fact, it becomes totally ineffective when the client is oblivious of changes in the underlying transport identifiers (e.g., on NAT rebind after timeout), and also does not guarantee unique identifiers (see [Section 4](#) for further details). Both methods generate a CID that is 32-bits in size, like the Security Parameter Index (SPI) in IPsec [RFC4301].

Similar approaches to support transparent handover of a DTLS session have been described in [[I-D.barrett-mobile-dtls](#)] and [[DTLSMOB](#)].

[2.](#) Conventions used in this document

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 [[RFC2119](#)].

[3.](#) Transport Agnostic Security Association Extension

In order to negotiate a Transport Agnostic Security Association, clients include an extension of type "ta_sa" in the extended client hello ([Section 3.1](#)). Servers that receive an extended hello containing a "ta_sa" extension MAY agree to use a Transport Agnostic Security Association by including an extension of type "ta_sa" in the extended server hello ([Section 3.2](#)).

If both server and client agree, the DTLSCiphertext format does change after the DTLS connection state is updated; i.e.: for the sending side, after the ChangeCipherSpec message is sent, for the receiving sides, after the ChangeCipherSpec is received.

The DTLSCiphertext format is changed for both the client and the server. However, only a client can initiate a switch to an unused 'cid' value; a server MUST utilize the same value seen on the last valid message received by the client.

```
struct {
    ContentType type;
    ProtocolVersion version;
    uint16 epoch;
    uint48 sequence_number;
    uint32 cid;                                // New field
    uint16 length;
    select (CipherSpec.cipher_type) {
        case block: GenericBlockCipher;
        case aead:  GenericAEADCipher;
```

```

    } fragment;
} DTLSCiphertext;

```

Figure 2: Modified DTLS Record Format

See [Section 3.3](#) for details on how to handle receipt of unknown or unexpected CIDs.

3.1. Extended Client Hello

In order to negotiate a Transport Agnostic Security Association, clients include an extension of type "ta_sa" in the extended client hello. The "extension_data" field of this extension SHALL contain the ClientSecAssocData structure in Figure 3.

In case the fixed(0) type has been negotiated, the 'cid' of the packets after ChangeCipherSpec is sent explicitly by the server.

In case the hotp(1) type has been negotiated, the initial 'cid' is calculated using the HOTP algorithm ([RFC4226](#)) as follows:

- o A 20-byte string is generated using a [RFC5705](#) exporter. The key material exporter uses the label "EXPORTER-ta-security-association-hotp" without the quotes, and without any context value.
- o The initial 'cid' equals to the first HOTP value (i.e., the 31-bit value of Sbits in [RFC4226](#) notation), generated by using the previously exported value as K.

Subsequent values of the HOTP algorithm can be used in place of the initial, as long as they fall into the negotiated window_size (see Figure 4).

```

enum {
    fixed(0), hotp(1), (255)
} SecAssocType;

struct {
    SecAssocType types<1..2^8-1>;
} ClientSecAssocData;

```

Figure 3: ta_sa extension, client

[3.2.](#) Extended Server Hello

Servers that receive an extended hello containing a "ta_sa" extension MAY agree to use a Transport Agnostic Security Association by including an extension of type "ta_sa", with "extension_data" being ServerSecAssocData in the extended server hello (Figure 4).

```
struct {
    SecAssocType type;
    select (type) {
        case fixed:
            struct {
                uint32 cid_value;
            };
        case hotp:
            struct {
                uint16 window_size;
            };
    };
} ServerSecAssocData;
```

Figure 4: ta_sa extension, server

In case the fixed(0) type is chosen, 'cid_value' contains the value to be used as 'cid'. In case hotp(1) type is chosen, 'window_size' must be greater or equal to 1, indicating the number of HOTP values that the server can recognize for this particular client.

[3.3.](#) Handling unknown CIDs

Server might need to deal with unknown or unexpected CIDs.

An unknown CID is one that is not found in the server's lookup table. This could happen because:

- o Server reboots and loses its state; or
- o There is a genuine bug in either client or server code (or even somewhere on the network path).

Either way, the server will not be able to locate a suitable context to use for sending an (encrypted) Alert back to the sender. Therefore, the server should simply discard records containing an unknown CID.

A CID might be unexpected if it existed but it's been already shifted. This situation may arise if the packet has been delayed by the network. Server MAY ignore a record carrying an unexpected CID.

Ignoring unknown or unexpected CIDs should also reduce the attack surface.

[3.4.](#) Wire Format Changes

How to signal the modified wire format to the receiving end is currently an open problem.

Note that moving the cid after the length field and computing the difference between the UDP datagram's and DTLS record's lengths is not an option because there is no guarantee that UDP datagrams carry one and one only DTLS record ([Section 4.1.1. of \[RFC6347\]](#)).

Ideally, we would just bump the version number, but there seems to be limited room for maneuver given the way TLS encodes version information in the record header, and also given that we want CID to work with DTLS 1.2 and later.

More discussion needed to sort out this point.

[3.5.](#) De-duplication Algorithm

The following algorithm assumes that receivers have an unambiguous way to tell that the wire format is the one described in [Section 3](#). As suggested in [Section 3.4](#), this could be signalled by a different version number { TBD, TBD }.

In order to enqueue an incoming record to the right security context,

a receiver SHALL extract the 4-tuple from the UDP packet and the ContentType of the TLS record. Then, if ContentType is change_cipher_spec or handshake, the receiver SHALL use the sender address to lookup or create (if ContentType is handshake and HandshakeType is one of ClientHello or ServerHello) the session context. If ContentType is one of application_data or alert, receiver SHALL inspect the ProtocolVersion field and: if it is { 3, 3 } (i.e., TLS v1.2), use the 4-tuple to lookup the session context, if it is { TBD, TBD }, extract the CID from the record and use it to lookup the session context.

4. Clashing HOTP CIDs

HOTP behaves like a PRF, thus uniformly distributing the produced CIDs across the 32-bit space. Table 1 presents the probability to end up with two separate sessions having the same HOTP CID when the number of concurrent sessions and/or the length of the CIDs sequence is increased.

Sessions x window_size	Collision probability
10	1.16415320717e-08, or about 1 in 85,899,347
100	1.16415254059e-06, or about 1 in 858,994
1000	0.000116408545826, or about 1 in 8,590
10000	0.011574031737, or about 1 in 86
100000	0.687813095694, or about 1 in 1
1000000	1.0, or about 1 in 1

Table 1

The takeaway is that 32-bits are probably too few for highly loaded servers that want to do HOTP as their primary CID allocation strategy. An alternative would be for the server to stop negotiating 'hotp' and fall back to 'fixed' when the number of active sessions crosses some threshold; another would be to increase the CID space to

40 or 48 bits when HOTP is used; yet another would be to allow the

length to be negotiable.

[5.](#) Security Considerations

CID does not affect the running protocol in any way other than adding an un-authenticated field to the record header. As such, this identifier has no effect on the overall security of the session with respect to authentication, confidentiality and integrity. On the other hand, since this identifier is not authenticated, it should not be used in any way that assumes it is, nor be assumed to be secret or unknown to an adversary. In general, this identifier should not be relied on more than the IP address or UDP port numbers are.

To address the privacy concerns of using a fixed identifier for the lifetime of a session which may roam through multiple networks, we have introduced the hotp identifier type. This type of identifier gives the client a chance to switch its ts_sa identity when also switching its transport identifiers or network attachment (assuming that client is made aware of the change before it sends a new DTLS record). The choice of which type of identifier to use is a trade-off between the request for privacy stated by the client and the ability of the server to control the identifiers in use at each point in time, as explained in [Section 4](#).

[6.](#) IANA Considerations

This document adds a new extension for DTLS: ts_sa(TODO). This extension MUST only be used with DTLS, and not with TLS. This extension is assigned from the TLS ExtensionType registry defined in [\[RFC5246\]](#).

[7.](#) Acknowledgments

Thanks to Achim Kraus, Carsten Bormann, Kai Hudalla, Simon Bernard, Stephen Farrell, for helpful comments and discussions that have shaped the document.

This work is partially supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI). This support does not imply endorsement.

[8.](#) References

8.1. Normative References

- [I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", [draft-ietf-tls-tls13-20](#) (work in progress), April 2017.
- [I-D.rescorla-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", [draft-rescorla-tls-dtls13-01](#) (work in progress), March 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4226] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", [RFC 4226](#), DOI 10.17487/RFC4226, December 2005, <<http://www.rfc-editor.org/info/rfc4226>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", [RFC 5705](#), DOI 10.17487/RFC5705, March 2010, <<http://www.rfc-editor.org/info/rfc5705>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", [RFC 6066](#), DOI 10.17487/RFC6066, January 2011, <<http://www.rfc-editor.org/info/rfc6066>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), DOI 10.17487/RFC6347, January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.

8.2. Informative References

- [DTLSMOB] Seggelmann, R., Tuexen, M., and E. Rathgeb, "DTLS Mobility", 2012.

Internet-Draft

DTLS ta_sa Extension

May 2017

[I-D.barrett-mobile-dtls]

Williams, M. and J. Barrett, "Mobile DTLS", [draft-barrett-mobile-dtls-00](#) (work in progress), March 2009.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), DOI 10.17487/RFC4301,

December 2005, <<http://www.rfc-editor.org/info/rfc4301>>.

Authors' Addresses

Nikos Mavrogiannopoulos
RedHat

EMail: nmav@redhat.com

Hannes Tschofenig
ARM

EMail: hannes.tschofenig@arm.com

Thomas Fossati
Nokia

EMail: thomas.fossati@nokia.com

