Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

T. Fossati
Nokia
H. Tschofenig
ARM
N. Mavrogiannopoulos
Red Hat

July 8, 2016

TLS/DTLS Optimizations for Internet of Things Deployments
draft-fossati-tls-iot-optimizations-00

Abstract

Internet protocols work well in a variety of environments, including Internet of Things (IoT) deployments. While there are some optimization possibilities to reduce code size, bandwidth utilization, and to improve battery lifetime, in general most Internet protocols are also applicable to constrained environments. TLS and DTLS are two such security protocols that can be used by many IoT devices since DTLS/TLS provide lot of flexiblity in terms credential choice, ciphersuite usage, etc. The DICE working group has developed a specification that profiles the use of TLS and DTLS for IoT environments, without changing the TLS/DTLS specifications.

This memo goes a step further and proposes changes to the DTLS/TLS protocol to introduce further optimizations. Since the ongoing work on TLS/DTLS 1.3 already offers several improvements (compared to previous versions) this document focuses on the use of version 1.3 and suggests further optimizations.

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 9, 2017.

Table of Contents

1. Introduction

Internet protocols work well in a variety of environments, including Internet of Things (IoT) deployments. While there are some optimization possibilities to reduce code size, bandwidth

utilization, and to improve battery lifetime, in general most Internet protocols are also applicable to constrained environments. TLS and DTLS are two such security protocols that can be used by many IoT devices since DTLS/TLS provide lot of flexiblity in terms credential choice, ciphersuite usage, etc. The DICE working group

has developed a specification that profiles the use of TLS and DTLS for IoT environments, without changing the TLS/DTLS specifications.

This memo goes a step further and proposes changes to the DTLS/TLS protocol to introduce further optimizations. Since the ongoing work on TLS/DTLS 1.3 [I-D.ietf-tls-tls13] already offers several improvements (compared to previous versions) this document focuses on the use of version 1.3 and suggests further optimizations.

This document discusses four extensions, namely:

Selective Fragment Retransmission: This extension improves retransmissions of lost handshake packets.

Transport Agnostic Security Associations: Changes to a connection (such as an IP address change) requires a new handshake. This extension introduces a transport independent identifier.

Reducing the DTLS Record Layer Header Overhead: This extension changes the record layer format to reduce the overhead.

Reducing Buffers: This extension allows a DTLS/TLS server running on a constrained node to indicate its buffer size.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Selective Fragment Retransmission

3.1. Problem Statement

The unit of retransmission used by the DTLS handshake is a whole flight (see Section 4.2.4 of [RFC6347]. If the underlying media is

inherently lossy, or shows high latency variance that might fire spurious retransmission, a single fragment that gets lost or is excessively delayed will force the whole flight to be retransmitted.

This is further exacerbated when the effective MTU is very low and therefore handshake messages have higher probability to be fragmented. For example, IEEE 802.15.4 networks have a 128-byte MTU size. In such an environment a very "ordinary" TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA negotiation can take up to 30 individual fragments, 2/3 of which are sent in flight 4. The loss of a single fragment in flight 4 implies a retransmission that is 20x the magnitude of the original loss.

The retransmission timer settings suggested in Section 11 of [I-D.ietf-dice-profile] offer mitigation for the spurious retransmit issue and, in general, help with congestion. However, they do not solve the retransmission of the entire flight.

3.2. Proposal

A potential solution is to add a NACK-based retransmission scheme to the DTLS handshake and the granularity of retransmission would be a message fragment. We note that each fragment in a DTLS handshake is effectively associated to a unique identifier defined by the tuple Handshake.{message_seq,fragment_offset,fragment_length} that can be used in the NACK report to identify the exact geometry of the missing data in the current flight, together with the right-most received byte.

4. Transport Agnostic Security Associations

4.1. Problem Statement

In DTLS, the security context demultiplexing is done via the 5-tuple. This implies that the associated DTLS context needs to be re-negotiated from scratch whenever the IP address changes. 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 cyle. A NAT device may also modify the source UDP port after an idle period. In such situations, there is not enough information in the DTLS record header for a DTLS server, which handles multiple clients, to associate the changed address to an existing client.

4.2. Proposal

A potential solution is to add the ability to negotiate, at handshake time, a transport independent identifier that is unique per security association. We call this identifier the 'Connection ID (CID)' in Figure 1. It decouples the DTLS session from the underlying transport protocol allowing the same DTLS security association to be migrated across different transport sessions.

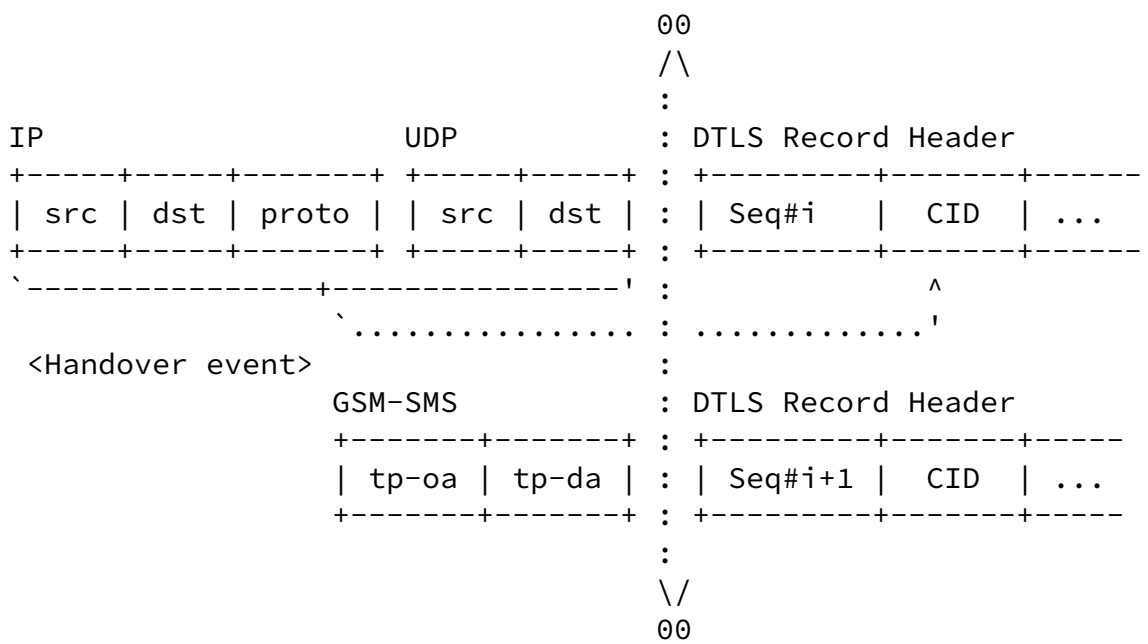

Figure 1: Transparent Handover of DTLS Session

That approach modifies the DTLS record layer header to the format

described in Figure 2.

```
struct {
     ContentType type;
     ProtocolVersion version;
     uint16 epoch;
     uint48 sequence_number;
     uint32 connection_id;                          // New field
     uint16 length;
     opaque fragment[DTLSPlaintext.length];
} DTLSPlaintext;
```

Figure 2: Modified DTLS Record Format

A similar approach to support transparent handover of a DTLS session has been described in [I-D.barrett-mobile-dtls] and [Seggelmann].

The privacy issue associated with the use of a long-term identifier must be taken into consideration. For example, client and server could use a hash chain [Lamport] derived from the shared secret and pick the next unused id on handover.

5. Reducing the DTLS Record Layer Header Overhead

5.1. Problem Statement

The DTLS record layer header adds 13 bytes of overhead, as described in Appendix B of [I-D.ietf-dice-profile]. While some of the information carried in the header is unavoidable, other parameters are redundant and included for backwards compatibility reasons. This burden becomes quite substantial in networks with small frame sizes (e.g., low power wide area networks).

Overhead that is not strictly needed could be removed to allow applications to transmit more data in a single packet or to make space for other DTLS features, such as the proposal described in Section 4.

5.2. Proposal

It is possible to at least remove the following parameters in the header:

- Protocol Version (2 bytes)
- The sequence number component of the nonce_explicit field at the AES-CCM layer is an exact copy of the sequence number in the record layer header field. This leads to a duplication of 8-bytes per record.

6. Reducing Buffers

6.1. Problem Statement

The Maximum Fragment Length extension [RFC6066] allows a client with limited buffer space to specify a different (smaller) maximum size for fragments that the server is allowed to send. The mechanism is not symmetrical: a server cannot state their buffer size. The assumption made in [RFC6066] is that the server is never going to be a constrained device, and therefore does not need such a capability. This may be true for many IoT deployments where the TLS client is implemented in an IoT device that connects to a server on the Internet that does not have memory limitations, such as a server in the cloud. However, with the desire to also deploy CoAPS and HTTPS-based servers in IoT devices, a constrained node may also need to run a DTLS/TLS server. In such a scenario, allowing a constrained server to advertise its Maximum Fragment Length helps to lower memory requirements.

6.2. Proposal

The semantics of the max_fragment_length extension could be modified to allow the server as well as the client to express their buffer sizes.

7. Acknowledgements

We would like to thank Stephen Farrell for suggesting the use of hash chains to implement a privacy-friendly connection id.

8. Security Considerations

This document suggests various extensions to DTLS/TLS and each of them comes with their own security and privacy considerations. Since this version of the document only suggests strawman proposals further discussions are needed to specify the details.

9. References

9.1. Normative References

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-13 (work in progress), May 2016.

[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>.

[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>.

9.2. Informative References

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

[I-D.ietf-dice-profile]

Tschofenig, H. and T. Fossati, "TLS/DTLS Profiles for the Internet of Things", draft-ietf-dice-profile-17 (work in progress), October 2015.

[Lamport] Lamport, L., "Password Authentication with Insecure Communication", Commun. ACM, 1981.

[Seggelmann] Seggelmann, R., Tuexen, M., and E. Rathgeb, "DTLS Mobility", ICDCN 12, 2012.

Authors' Addresses

Thomas Fossati
Nokia
Cambridge
UK

Email: thomas.fossati@nokia.com

Hannes Tschofenig
ARM
Cambridge
UK

Email: hannes.tschofenig@arm.com

Nikos Mavrogiannopoulos
Red Hat
Brno
Czech Republic

Email: nmav@redhat.com