

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2011

Ted. Hardie
Panasonic Wireless Research Lab
Jake. Khuon
October 18, 2010

Multipath DTLS Session Layer
draft-hardie-mdtls-session-00

Abstract

The Internet model has traditionally avoided the complication of an explicit session layer, in favor of having applications and transports divide the relevant work between them. While this has been a successful strategy when the packet flows from two hosts all take the same path, there may be advantages in using a session-layer strategy when a host initiates related flows from multiple interfaces in independent routing domains (e.g. a 3G or 4G interface and a WiFi interface). This draft discusses an approach re-using the security association mechanisms present in DTLS [[RFC4347](#)] to create a simple session layer.

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](#)].

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 April 21, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the

Internet-Draft

MDTLS

October 2010

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.

Table of Contents

1.	Introduction	3
2.	Why DTLS?	3
3.	Protocol Mechanics	3
3.1.	Datagram Transport Layer Security (DTLS) Client-Senders and Server-Receivers	3
3.2.	Sequence and Epoch Number Management	4
3.3.	Cookie Management	4
4.	Messaging and Message Types	4
4.1.	Session Setup	4
4.1.1.	The Prime Initiator	4
4.1.2.	The Reuse Initiator	4
5.	Traffic Profile	5
6.	DTLS receiver behavior	5
7.	IANA Considerations	5
8.	Security Considerations	5
9.	Acknowledgements	5
10.	Normative References	6
	Authors' Addresses	6

1. Introduction

The Internet model has traditionally avoided the complication of an explicit session layer, in favor of having applications and transports divide the relevant work between them. While this has been a successful strategy when the packet flows from two hosts all take the same path, there may be advantages in using a session-layer strategy when a host initiates related flows from multiple interfaces in independent routing domains (e.g. a 3G or 4G interface and a WiFi interface). This draft discusses an experimental approach re-using the security association mechanisms present in DTLS [[RFC4347](#)] to create a simple session layer.

2. Why DTLS?

DTLS uses a return routability check to avoid certain classes of Denial of Service attacks. Essentially, it passes a cookie to a host requesting a security association; only after it receives a request containing this cookie does it move forward with an association. In the current DTLS design, each cookie is designed to be used only once, and the method for checking a cookie is entirely stateless. But much the same design can be used to associate multiple inbound flows with the same session; in this case the cookie identifies new flows as being part of the same session.

Though the application data transmitted using DTLS is sent unreliably, DTLS uses a simple retransmission-based mechanism to create a reliable command channel. This command channel can be used to manage the state machine across flows. The experiment conducted locally explored some of those in detail, but that output will be described in detail in a later report.

3. Protocol Mechanics

[3.1.](#) Datagram Transport Layer Security (DTLS) Client-Senders and Server-Receivers

The transport mechanism is based on DTLS record flights. The Client/Sender sends traffic from a single flight out multiple physical interfaces, after a modified ClientHello confirms that multiple flows can be aggregated by the receiver. The Server/Receiver of the traffic creates a common queue for records received from all flows in the session, which are read out by applications just as any other DTLS data would be.

[3.2.](#) Sequence and Epoch Number Management

To enable reassembly of multiple transport packet streams at the receiver side, packets from all flows in the session must include a sequence and epoch number that originates from a single global number space.

[3.3.](#) Cookie Management

All packets received by the DTLS server must appear as part of the same session. For this reason, flows after the initial cookie assignment reuse the same cookie. In this case, that cookie re-use is a way of managing the association of multiple flows to a single session.

[4.](#) Messaging and Message Types

[4.1.](#) Session Setup

A session may be composed of several DTLS flows. Before any data traffic can be sent, the transport flows themselves must first be negotiated and set up. Once all DTLS flows are set up, traffic can be multiplexed across them. In order to set up these flows a series of messages will be used to communicate between the DTLS clients and servers.

The ClientHello message is used by the DTLS Client/Sender to begin the DTLS handshake with the DTLS Server/Receiver. There are two

variants required by this proposal.

[4.1.1.](#) The Prime Initiator

This is sent by the first DTLS Client/Sender in a multisender client array. This message is set to type [TBD] in the type field and has an epoch and sequence set to zero. A successful ClientHello negotiation of this type results in the sender receiving a cookie which may be reused by later Client/Senders.

[4.1.2.](#) The Reuse Initiator

This is sent by subsequent DTLS senders in the array. It is set to a type value of [TBD] and includes the session cookie received in the Prime Initiator exchange. Its epoch should be the same as that set at the end of the previous flow set-up and its sequence number should be subsequent to the last sequence number used by the previous flow. The DTLS Receiver/Server should not send back a new cookie in response to this variant of the ClientHello.

[5.](#) Traffic Profile

Packets between the DTLS clients and the DTLS server appear to the network as UDP traffic and could be marked by any QoS classes available for UDP.

[6.](#) DTLS receiver behavior

The DTLS Receiver/Server sees all packets from the flows connected by the initiating session cookie as part of the same session. All flows share a single sequence number space. When packets are received from any flow within the session, they can thus be correctly inserted into the receiver-side queue in order. This does require mid-queue insertion, with the resulting receiver-side costs.

[7.](#) IANA Considerations

This document makes no request of IANA. If the described mechanisms were standardized, they would require registrations in the Extension Type registry set up in [RFC 5246](#) [[RFC5246](#)]. We note that there is

no private use space in this registry.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

Since there are now multiple individual path carrying a portion of the overall session flow, there is a greater possibility of an on-path attacker disrupting the flow. The re-use of a return-routability cookie for multiple flows also re-opens the possibility of denial of service, though this can be handled in a variety of ways. The easiest would likely be to introduce a limitation in the number of flows eligible to re-use a cookie. More robust methods requiring a double cookie (session cookie and return routability check cookie) are also relatively easy to implement.

9. Acknowledgements

Some work in this area was funded by Panasonic's Next-Generation Mobile Development Center, and Iishi Hidenori, Takei Ichiro, and Sanda Takako of NMDC gave valuable feedback and encouragement. Girish Kumar coded most of the DTLS aspects of the test system. Eric Rescorla gave valuable early feedback.

10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", [RFC 4347](#), April 2006.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.

Authors' Addresses

Ted Hardie

Panasonic Wireless Research Lab
10900 Tantau Ave.
Cupertino, California 95014
USA

Phone: +1-408-628-5864
Email: ted.ietf@gmail.com

Jake Khuon

Email: khuon@neebu.net