EXHIBIT 9

Internet Engineering Task Force (IETF)
Request for Comments: 6520
Category: Standards Track
ISSN: 2070-1721

R. Seggelmann
M. Tuexen
Muenster Univ. of Appl. Sciences
M. Williams
GWhiz Arts & Sciences

February 2012

Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension

Abstract

This document describes the Heartbeat Extension for the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols.

The Heartbeat Extension provides a new protocol for TLS/DTLS allowing the usage of keep-alive functionality without performing a renegotiation and a basis for path MTU (PMTU) discovery for DTLS.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6520.

Copyright Notice

Table of Contents

1. Introduction

1.1. Overview

This document describes the Heartbeat Extension for the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols, as defined in [RFC5246] and [RFC6347] and their adaptations to specific transport protocols described in [RFC3436], [RFC5238], and [RFC6083].

DTLS is designed to secure traffic running on top of unreliable transport protocols. Usually, such protocols have no session management. The only mechanism available at the DTLS layer to figure out if a peer is still alive is a costly renegotiation, particularly when the application uses unidirectional traffic. Furthermore, DTLS needs to perform path MTU (PMTU) discovery but has no specific message type to realize it without affecting the transfer of user messages.

TLS is based on reliable protocols, but there is not necessarily a feature available to keep the connection alive without continuous data transfer.

The Heartbeat Extension as described in this document overcomes these limitations. The user can use the new HeartbeatRequest message, which has to be answered by the peer with a HeartbeartResponse immediately. To perform PMTU discovery, HeartbeatRequest messages containing padding can be used as probe packets, as described in [RFC4821].

1.2. Conventions

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 [RFC2119].

2. Heartbeat Hello Extension

The support of Heartbeats is indicated with Hello Extensions. A peer cannot only indicate that its implementation supports Heartbeats, it can also choose whether it is willing to receive HeartbeatRequest messages and respond with HeartbeatResponse messages or only willing to send HeartbeatRequest messages. The former is indicated by using peer_allowed_to_send as the HeartbeatMode; the latter is indicated by using peer_not_allowed_to_send as the Heartbeat mode. This decision can be changed with every renegotiation. HeartbeatRequest messages MUST NOT be sent to a peer indicating peer_not_allowed_to_send. If an endpoint that has indicated peer_not_allowed_to_send receives a HeartbeatRequest message, the endpoint SHOULD drop the message silently and MAY send an unexpected_message Alert message.

The format of the Heartbeat Hello Extension is defined by:

```
enum {
   peer_allowed_to_send(1),
   peer_not_allowed_to_send(2),
   (255)
} HeartbeatMode;

struct {
   HeartbeatMode mode;
} HeartbeatExtension;
```

Upon reception of an unknown mode, an error Alert message using illegal_parameter as its AlertDescription MUST be sent in response.

3. Heartbeat Protocol

The Heartbeat protocol is a new protocol running on top of the Record Layer. The protocol itself consists of two message types: HeartbeatRequest and HeartbeatResponse.

```
enum {
   heartbeat_request(1),
   heartbeat_response(2),
   (255)
} HeartbeatMessageType;
```

A HeartbeatRequest message can arrive almost at any time during the lifetime of a connection. Whenever a HeartbeatRequest message is received, it SHOULD be answered with a corresponding HeartbeatResponse message.

However, a HeartbeatRequest message SHOULD NOT be sent during handshakes. If a handshake is initiated while a HeartbeatRequest is still in flight, the sending peer MUST stop the DTLS retransmission timer for it. The receiving peer SHOULD discard the message silently, if it arrives during the handshake. In case of DTLS, HeartbeatRequest messages from older epochs SHOULD be discarded.

There MUST NOT be more than one HeartbeatRequest message in flight at a time. A HeartbeatRequest message is considered to be in flight until the corresponding HeartbeatResponse message is received, or until the retransmit timer expires.

When using an unreliable transport protocol like the Datagram Congestion Control Protocol (DCCP) or UDP, HeartbeatRequest messages MUST be retransmitted using the simple timeout and retransmission scheme DTLS uses for flights as described in Section 4.2.4 of [RFC6347]. In particular, after a number of retransmissions without receiving a corresponding HeartbeatResponse message having the expected payload, the DTLS connection SHOULD be terminated. The threshold used for this SHOULD be the same as for DTLS handshake messages. Please note that after the timer supervising a HeartbeatRequest messages expires, this message is no longer considered in flight. Therefore, the HeartbeatRequest message is eligible for retransmission. The retransmission scheme, in combination with the restriction that only one HeartbeatRequest is allowed to be in flight, ensures that congestion control is handled appropriately in case of the transport protocol not providing one, like in the case of DTLS over UDP.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**