DTLS over DCCP
Internet Draft
Document: draft-ietf-dccp-dtls-04.txt
Expires: June 2008
Intended status: Proposed Standard

T. Phelan
Sonus Networks
December 21, 2007

Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP)

Status of this Memo

Abstract

This document specifies the use of Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP). DTLS provides communications privacy for datagram protocols and allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. DCCP is a transport protocol that provides a congestion-controlled unreliable datagram service.

Table of Contents

1. Introduction

This document specifies how to use Datagram Transport Layer Security (DTLS), as specified in [RFC4347], over the Datagram Congestion Control Protocol (DCCP), as specified in [RFC4340].

DTLS is an extension of Transport Layer Security (TLS, [RFC4346]) that modifies TLS for use with the unreliable transport protocol UDP. TLS is a protocol that allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering and message forgery. DTLS can be viewed as TLS-plus-adaptations-for-unreliability.

DCCP provides an unreliable transport service, similar to UDP, but with adaptive congestion control, similar to TCP and SCTP. DCCP can be viewed equally well as either UDP-plus-congestion-control or TCP-minus-reliability (although, unlike TCP, DCCP offers multiple congestion control algorithms).

The combination of DTLS and DCCP will offer transport security capabilities to DCCP users similar to those available for TCP, UDP and SCTP.

2. Terminology

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].

3. DTLS over DCCP

The approach here is very straightforward -- DTLS records are transmitted in the Application Data fields of DCCP-Data and DCCP-DataAck packets (in the rest of the document assume that "DCCP-Data packet" means "DCCP-Data or DCCP-DataAck packet"). Multiple DTLS records MAY be sent in one DCCP-Data packet, as long as the resulting packet is within the Path Maximum Transfer Unit (PMTU) currently in force, if the Don't Fragment (DF) bit is being used, or within the current DCCP maximum packet size if the DF bit is not being used (see section 3.5 for more information on PMTU Discovery). A single DTLS record MUST be fully contained in a single DCCP-Data packet; it MUST NOT be split over multiple packets.

3.1 DCCP and DTLS Sequence Numbers

Both DCCP and DTLS use sequence numbers in their packets/records. These sequence numbers serve somewhat, but not completely, overlapping functions. Consequently, there is no connection between

the sequence number of a DCCP packet and the sequence number in a DTLS record contained in that packet and no connection between sequence number-related features such as DCCP synchronization and DTLS anti-replay protection.

3.2 DCCP and DTLS Connection Handshakes

Unlike UDP, DCCP is connection-oriented, and has a connection handshake procedure that precedes the transmission of DCCP-Data and DCCP-DataAck packets. DTLS is also connection-oriented, and has a handshake procedure of its own that must precede the transmission of actual application information. Using the rule of mapping DTLS records to DCCP-Data and DCCP-DataAck packets in section 3, above, the two handshakes are forced to happen in series, with the DCCP handshake first, followed by the DTLS handshake. This is how TLS over TCP works.

However, the DCCP handshake packets DCCP-Request and DCCP-Response have Application Data fields and can carry user data during the DCCP handshake, and this creates the opportunity to perform the handshakes partially in parallel. DTLS client implementations MAY choose to transmit the ClientHello message in the DCCP-Request packet. DTLS server implementations MAY choose to respond to a ClientHello message received in a DCCP-Request packet with a HelloVerifyRequest message, if denial of service countermeasures are to be used, or a ServerHelloDone message otherwise, in the DCCP-Response packet. DTLS servers MAY also choose to delay the response until the handshake completes and then send the response in a DCCP-Data packet.

DTLS handshake messages can be quite large, theoretically up to 2^24-1 bytes and in practice often many kilobytes. Subsequently, unlike other DTLS messages, the handshake messages may be fragmented over multiple DTLS records. If the size of the ClientHello is too large to transmit in its entirety in a DCCP-Request packet the ClientHello MUST be sent in DCCP-Data packets after the DCCP handshake is complete. Similarly, if the server response to a ClientHello is too large to transmit in its entirety in a DCCP-Response packet, it MUST be sent in DCCP-Data packets after the DCCP handshake is complete.

Transmission of subsequent DTLS handshake messages MUST wait for the completion of the DCCP handshake and use DCCP-Data packets.

Note that even though the DCCP handshake is a reliable process (handshake messages are retransmitted as required if messages are lost), the transfer of Application Data in DCCP-Request and DCCP-Response packets is not necessarily reliable. For example, DCCP Server implementations are free to discard Application Data received in DCCP-Request packets. And if DCCP-Request or DCCP-Response packets need to be retransmitted, the DCCP implementation may choose to not include the Application Data present in the initial message.

Since the DTLS handshake is also a reliable process, it will interoperate across the data delivery unreliability of DCCP (after all, one of the basic functions of DTLS is to work over unreliable transport). If ClientHello messages or the HelloVerifyRequest or ServerHelloDone messages are lost, the ClientHello message will be retransmitted by DTLS.

This is regardless of whether the messages were sent in DCCP-Response/Request packets or DCCP-Data packets. However, the only way for DTLS to retransmit a ClientHello message that was originally transmitted in a DCCP-Request packet (and it or the response was lost somehow) is to wait for the DCCP handshake to complete and then send the ClientHello in a DCCP-Data packet. This is due to the characteristic of DCCP that the next opportunity to send data after sending data in a DCCP-Request is only after the connection handshake completes.

DCCP and DTLS use similar strategies for retransmitting handshake messages. If there is no response to the original request (DCCP-Request or ClientHello respectively) within normally 1 second, the message is retransmitted. The timer is then doubled and the process repeated until a response is received, or a maximum time is exceeded.

Therefore, if the ClientHello message is sent in a DCCP-Request packet, and the DCCP-Request or DCCP-Response message is lost, the DCCP and DTLS handshakes could be timing out on similar schedules. The DCCP-Request packets will be retransmitted on timeout, but the ClientHello packet cannot be retransmitted until the DCCP handshake completes (there is no possibility of adding new Application Data to a DCCP-Request retransmission). In order to avoid multiple retransmissions queuing up before the first retransmission can be sent, DTLS over DCCP MUST wait until the completion of the DCCP handshake before restarting its retransmission timer.

3.3 Effects of DCCP Congestion Control

Given the large potential sizes of the DTLS handshake messages, it is possible that DCCP congestion control could throttle the transmission of the DTLS handshake to the point that the transfer cannot complete before the DTLS timeout and retransmission procedures take effect. Adding retransmitted messages to a congested situation might only make matters worse and delay connection establishment.

Note that a DTLS over UDP application transmitting handshake data into this same network situation will not necessarily receive better throughput, and might actually see worse effective throughput. Without the pacing of slow-start and congestion control, a UDP application might be making congestion worse and lowering the effective throughput it receives.

As stated in [RFC4347], "mishandling of the [retransmission] timer can lead to serious congestion problems". This remains as true for DTLS over DCCP as it is for DTLS over UDP.

DTLS over DCCP implementations SHOULD take steps to avoid retransmitting a request that has been queued but not yet actually transmitted by DCCP, when the underlying DCCP implementation can provide this information. For example, DTLS could delay starting the retransmission timer until DCCP indicates the message has been transferred from DCCP to the IP layer.

In addition to the retransmission issues, if the throughput needs of the actual application data differ from the needs of the DTLS handshake, it is possible that the handshake transference could leave the DCCP congestion control in a state that is not immediately suitable for the application data that will follow. For example, DCCP CCID2 ([RFC4341]) congestion control uses an Additive Increase Multiplicative Decrease (AIMD) algorithm similar to TCP congestion control. If it is used then it is possible that transference of a large handshake could cause a multiplicative decrease that would not have happened with the application data. The application might then be throttled while waiting for additive increase to return throughput to acceptable levels.

Applications where this might be a problem should consider using DCCP CCID3 ([RFC4342]). CCID3 implements TCP-Friendly Rate Control (TFRC, [RFC3448])). TFRC varies the allowed throughput more slowly than AIMD and might avoid the discontinuities possible with CCID2.

3.4 DTLS Sessions and DCCP Connections

There is no necessary relationship between the life of a DTLS session and the life of a DCCP connection. Often the session and connection lives start and stop together (DCCP connection establishment immediately followed by DTLS session establishment, DTLS session termination immediately followed by DCCP connection termination), but this is not the only possibility.

A single DTLS session may span multiple DCCP connections using the DTLS session resumption features. The session resumption feature of DTLS is widely used and this situation is likely to occur frequently. It is even possible to resume a DTLS session over a different transport.

A DCCP connection has no knowledge of the type of application data it is transferring. It could conceivably contain multiple DTLS sessions, in series or even in parallel, while simultaneously transferring non-DTLS data. In practice this could be difficult to demultiplex at the application/DTLS level and support for this is likely to be rare to nonexistent. [RFC4347] does not specifically exclude multiple DTLS sessions simultaneously sharing the same underlying transport.

A special case of this DCCP connection flexibility is an application that starts up transferring non-DTLS data, and then switches to DTLS after some time. This is likely to be useful and has implications for the choice of DCCP Service Code. See section 3.6 for more information on this.

3.5 PMTU Discovery

Each DTLS record must fit within a single DCCP-Data packet. DCCP packets are normally transmitted with the DF (Don't Fragment) bit set for IPv4, and of course all IPv6 packets are unfragmentable in the network. Because of this, DCCP performs Path Maximum Transmission Unit (PMTU) Discovery.

DTLS also normally uses the DF bit and performs PMTU Discovery on its own, using an algorithm that is strongly similar to the one used by DCCP. A DTLS over DCCP implementation MAY use the DCCP-managed value for PMTU and not perform PMTU Discovery on its own. Alternatively, a DTLS over DCCP implementation MAY choose to use its own PMTU Discovery calculations, as specified in [RFC4347], but MUST NOT use a value greater than the value determined by DCCP.

DTLS implementations normally allow applications to reset the PMTU estimate back to the initial state. When that happens, DTLS over DCCP implementations SHOULD also reset the DCCP PMTU estimation.

DTLS implementations also sometimes allow applications to control the use of the DF bit (when running over IPv4). DTLS over DCCP implementations SHOULD control the use of the DF bit by DCCP in concert with the application's indications, when the DCCP implementation supports this. Note that DCCP implementations are not required to support sending packets with the DF bit not set.

Note that the DCCP Maximum Packet Size (MPS in [RFC4340]) is bounded by the current congestion control state (Congestion Control Maximum Packet Size, CCMPS in [RFC4340]). Even when the DF bit is not set and DCCP packets may then be fragmented, the MPS may be less than the 65,535 bytes normally used in UDP. It is also possible for the DCCP CMPS, and thus the MPS, to vary over time as congestion conditions change. DTLS over DCCP implementations MUST NOT use a DTLS record size that is greater than the DCCP MPS currently in force.

3.6 DCCP Service Codes

The DCCP connection handshake includes a field called Service Code that is intended to describe "the application-level service to which the client application wants to connect". Further, "Service Codes are intended to provide information about which application protocol a connection intends to use, thus aiding middleboxes and reducing reliance on globally well-known ports" [RFC4340].

It is expected that many middleboxes will give different privileges to applications running DTLS over DCCP versus just DCCP. Therefore, applications that use DTLS over DCCP sometimes and just DCCP other times MUST register and use different Service Codes for each mode of operation. Applications that use both DCCP and DTLS over DCCP MAY choose to listen for incoming connections on the same DCCP port and distinguish the mode of the request by the offered Service Code.

Some applications may start out using DCCP without DTLS, and then optionally switch to using DTLS over the same connection. Since there is no way to change the Service Code for a connection after it is established, these applications will use one Service Code.

3.7 New Versions of DTLS

As DTLS matures, revisions to and updates for [RFC4347] can be expected. DTLS includes mechanisms for identifying the version in use and presumably future versions will either include backward compatibility modes or at least not allow connections between dissimilar versions. Since DTLS over DCCP simply encapsulates the DTLS records transparently, these changes should not affect this document and the methods of this document should apply to future versions of DTLS.

Therefore, in the absence of a revision to this document, it is assumed to apply to all future versions of DTLS. This document will only be revised if a revision to DTLS makes a revision to the encapsulation necessary.

It is RECOMMENDED that an application migrating to a new version of DTLS keep the same DCCP Service Code used for the old version and allow DTLS to provide the version negotiation support. If the application developers feel that the new version of DTLS provides significant new capabilities to the application that will change the behavior of middleboxes, they MAY use a new Service Code.

4. Security Considerations

Security considerations for DTLS are specified in [RFC4347] and for DCCP in [RFC4340]. The combination of DTLS and DCCP introduces no new security considerations.

5. IANA Considerations

There are no IANA actions required for this document.

6. References

6.1 Normative References

[RFC4347] Rescorla, E., "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4340] Kohler, E., Handley, M., Floyd, S., "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

6.2 Informative References

[RFC4341] Floyd, S., Kohler, E., "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, March 2006.

[RFC4342] Floyd, S., Kohler, E., Padhye, J., " Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, March 2006.

[RFC3448] Handley, M., Floyd, S., Padhye, J., Widmer, J., " TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.

7. Author's Address

Tom Phelan
Sonus Networks
7 Technology Park Dr.
Westford, MA USA 01886
Phone: 978-614-8456
Email: tphelan@sonusnet.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.