Network Working Group
Internet-Draft
Expires: August 29, 2006

N. Modadugu
Stanford University
E. Rescorla
Network Resonance
February 25, 2006

Extensions for Datagram Transport Layer Security (TLS) in Low Bandwidth Environments

draft-modadugu-dtls-short-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on August 29, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document describes a series of extensions to Datagram Transport Layer Security (DTLS) which reduce the per-record bandwidth of the data channel. These extensions apply only to the on-the-wire representation of the protocol and do not affect cryptographic processing.

Table of Contents

1. Introduction 3
  1.1. Conventions Used In This Document 3
2. Background 3
3. Sequence Number Length 4
4. Version Field Elimination 5
5. Length Field Elimination 6
6. Implicit Application Data Header 7
7. Security Considerations 8
8. IANA Considerations 9
9. References 9
  9.1. Normative References 9
  9.2. Informative References 9
Authors' Addresses 10
Intellectual Property and Copyright Statements 11

1. Introduction

Datagram Transport Layer Security (DTLS) [5] is a protocol which provides channel-oriented communications security for datagram traffic. A communication channel that uses DTLS as security protocol incurs some bandwidth overhead that results from additional per-record headers and encryption overhead. Reducing this bandwidth overhead is desireable when DTLS is used in wireless environments or to secure real-time traffic. This document describes a series of extensions to DTLS which reduce the per-record bandwidth. These extensions apply only to the on-the-wire representation of the protocol and do not affect the data subject to cryptographic processing.

1.1. Conventions Used In This Document

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 [2].

2. Background

The DTLS record format (based on the TLS [3] record format) is shown below:

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

The major sources of per-record overhead in DTLS are:

Field	size(bytes)
type	1
version	2
epoch	2
sequence_number	6
length	2
MAC	10-20 bytes
encryption overhead	9-32

The largest performance improvement can be obtained by moving to a cipher suite with less overhead. DTLS-CTR [7] describes such a mode. This document describes how to reduce packet size further by reducing the size of some of the fields in the record.

All the optimizations described in this memo are implemented using the TLS Extensions mechanism [4].

3. Sequence Number Length

TLS, on which DTLS is based, uses a 64-bit sequence number. However, because TLS must run on a reliable protocol, the sequence number is implicit and does not take up space on the wire. In DTLS, the sequence number is explicit and broken up into a 16-bit "epoch" describing the number of handshakes that have happened on this association and a 48-bit per-epoch sequence number. This has the advantage of simplicity, but the disadvantage of taking up a fair amount of space in the packet.

The sequence_number_length extension shortens the on-the-wire representation of the sequence number without shortening the actual sequence number. This means that the high order bits are not present in the packet but rather MUST be deduced. Implementations SHOULD use the technique of Appendix A of [6] to compute the high order bits of the sequence number and epoch number.

When the client sends the sequence_number_length extension "extension_data" field must contain a SequenceNumberLengthValues field:

```
uint8 SequenceNumberLengthValue;
SequenceNumberLengthValue SequenceNumberLengthValues<1..7>;
```

This field contains the sequence number lengths which the client is willing to accept. These lengths are the combined length in bytes of the sequence number and epoch. Permissible values are 2,3,4,5,6,7,8, with the sequence number and epoch divided up according to the following table:

Total	Epoch	Sequence
2	2 bits	14 bits
3	4 bits	20 bits
4	6 bits	26 bits
5	1 byte	4 bytes
6	1 byte	5 bytes
7	1 byte	6 bytes
8	2 bytes	6 bytes

When the sequence number crosses a byte boundary, the high order bits of that byte SHALL be considered to be the epoch and the low order bits SHALL be considered to be the high order bits of the sequence number. Note that the 8-byte value is equivalent to the default DTLS behavior and is provided purely for completeness.

If the server receives a SequenceNumberLengthValue that is not one of the allowed values, it MUST abort the handshake with an "illegal_parameter" alert. If the server receives a sequence_number_length extension and does not wish to negotiate sequence_number_length it should ignore the sequence_number_length extension.

If the server wishes to negotiate sequence_number_length it responds with its own sequence_number_length extension. The server's "extension_data" field for this extension shall consist of a single SequenceNumberLengthValue value, which MUST be selected from the list provided by the client. If the client receives a SequenceNumberLengthValue that was not on its supplied list, it MUST abort the handshake with an "illegal_parameter" exception.

The new sequence number length takes effect following the change_cipher_spec for the new cipher suite. Because the epoch value is used to differentiate data from different cipher suite states (different negotiations) care must be taken when renegotiating sequence number length during active data transfer. In the worst case scenario, the receiver may need to try to parse/decrypt the packet under both potential state settings. Because only one produces a valid parse with a valid MAC, the correct choice is unambiguous.

4. Version Field Elimination

Every TLS/DTLS record contains a two-byte version field. This field is mostly redundant because the correct version is negotiated during

the TLS/DTLS handshake. The NoVersionField extension eliminates the version field from the wire representation.

In order to negotiate the non-use of the version field clients MAY include an extension of type "no_version_field" in the extended client hello. The "extension_data" field of this extension shall be empty.

Servers that receive an extended hello containing a "no_version_field" extension, MAY agree to omit the version field including an extension of type "no_version_field", with empty "extension_data", in the extended server hello.

Once the "no_version_field" extension is negotiated, packets in the newly negotiated association (i.e., after the change_cipher_spec) SHALL omit the version field. This does not affect the computation of the HMAC, which MUST include the version field as negotiated by the DTLS handshake, i.e., as it would have been included in the header.

5. Length Field Elimination

DTLS records contain a length field, which allows more than one record to be carried in a single datagram (though each record must fit inside a single datagram). However, if the peers agree to place only one record per datagram, the length field becomes superfluous. The "no_length_field" extension is used to make this agreement.

In order to negotiate the non-use of the length field clients MAY include an extension of type "no_length_field" in the extended client hello. The "extension_data" field of this extension shall be empty.

Servers that receive an extended hello containing a "no_length_field" extension, MAY agree to omit the length field including an extension of type "no_length_field", with empty "extension_data", in the extended server hello.

Once the "no_length_field" extension is negotiated, packets in the newly negotiated association (i.e., after the change_cipher_spec) SHALL omit the length field. This does not affect the computation of the HMAC, which MUST include the length field as negotiated by the DTLS handshake, i.e., as it would have been included in the header. When the "no_length_field" extension is in effect, implementations MUST NOT place more than one record per datagram.

6. Implicit Application Data Header

In principle, because all the data in the DTLS header is incorporated into the DTLS record MAC, the entire header can be omitted and reconstructed by trying all candidate headers. We propose a somewhat less radical approach: omitting the header for records of type "application_data". Because these records comprise the majority of the traffic on a DTLS connection, this extension provides a significant optimization while minimizing ambiguity.

In order to negotiate the implicit application data header, clients MAY include an extension of type "implicit_header" in the extended client hello. The "extension_data" field of this extension shall be empty.

Servers that receive an extended hello containing a "implicit_header" extension, MAY agree to this optimization extension of type "implicit_header", with empty "extension_data", in the extended server hello.

Once the "implicit_header" extension is negotiated, application data records in the newly negotiated association (i.e., after the change_cipher_spec) SHOULD omit the following values in the DTLS header:

- Content type
- Version
- Length
- Sequence Number

This does not affect the computation of the HMAC, which MUST include these values as if they were present. In addition, as with the "no_length_field" extension, there must only be one record per transport-level datagram.

The "implicit_header" extension introduces some ambiguity in record receipt processing. This ambiguity can, however, be resolved by trial decryption. Implementations MAY use the algorithm described below in order to properly receive a given record. The initial value of ESN (the expected sequence number) is set to 1 + (sequence number of the Finished message record), which should be the sequence number of the first application data record.

1. If the first byte does not match a known content type go to step 5.
2. If the version field does not match the current version go to step 5.
3. If the length does not match the rest of the record, go to step 5.

4. Attempt to decrypt the record as a record with header present. If the MAC verifies, set ESN to the record sequence number+1 and pass the record it to the next layer. Otherwise, proceed to step 5.
5. Prepend an application_data header with sequence number of ESN. Attempt to decrypt. If the MAC checks, set ESN to the record sequence number+1 and pass the record to the next layer.
6. Repeat step 5 with all ESN values in the current replay window.
7. If no valid ESN can be found, discard the record.

The above algorithm is generic. However, in applications where an application layer sequence number is present in plaintext in the record payload (see TODO) (e.g., RTP), it MAY be appropriate to maintain an offset between the two sequence numbers and use that to generate the initial ESN estimate in step 5. However, they MUST still use the sequence number of the last valid packet to set the replay--and hence resynchronization--window.

Note that although in principle this specification allows the inter-mixture of application_data records with and without the header, senders SHOULD generally send records without the header when the extension is in effect. The one reasonable exception would be if an application layer sequence number is present and makes a large jump. Implementations MAY add an explicit application_data header to several frames to effect a resynchronization.

7. Security Considerations

There are two security concerns introduced by these extensions. The first involves the security of the negotiation and the second the security of the transport protocol. Because the negotiation is protected by the TLS/DTLS handshake, attackers can neither force the use of these extensions nor block them while allowing the negotiation to succeed.

Although these extensions involve changing the bits on the wire, the transformations involved are made in authenticated but unencrypted data. This has two implications: (1) Any attacker who possesses the encrypted stream of an ordinary DTLS connection can generate a stream with any or all of these fields removed. Thus, if a connection uses these extensions and is weak, the underlying TLS connection must be weak as well. (2) Although the receiver needs to deduce certain values, this does not produce a security threat because the attacker could have replaced the real values on the wire with the values that the receiver deduces in the low bandwidth version. In both cases, what stops tampering is the use of a strong MAC.

8. IANA Considerations

This document defines four new extensions for DTLS, in accordance with [4]:

```
enum { sequence_number_length (??), no_version_field (??),
       no_length_field(??), implicit_header (??)} ExtensionType;
```

[[NOTE: These values need to be assigned by IANA]]

The "sequence_number_length", "no_length_field" and "implicit_header" extensions MAY only be used with DTLS and MUST NOT be used with TLS. The "no_version_field" extension MAY be used with either DTLS or TLS.

9. References

9.1. Normative References

[1] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Dierks, T. and E. Rescorla, "The TLS Protocol Version 1.1", draft-ietf-tls-rfc2246-bis-13 (work in progress), June 2005.

[4] Blake-Wilson, S., "Transport Layer Security (TLS) Extensions", draft-ietf-tls-rfc3546bis-02 (work in progress), October 2005.

[5] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", draft-rescorla-dtls-05 (work in progress), June 2005.

[6] Kent, S., "IP Encapsulating Security Payload (ESP)", draft-ietf-ipsec-esp-v3-10 (work in progress), March 2005.

9.2. Informative References

[7] Modadugu, N. and E. Rescorla, "AES Counter Mode Cipher Suites for TLS and DTLS", draft-modadugu-tls-ctr-00 (work in progress), October 2005.

Authors' Addresses

Nagendra Modadugu
Stanford University
353 Serra Mall
Stanford, CA 94305
USA

Email: nagendra@cs.stanford.edu

Eric Rescorla
Network Resonance
2483 E. Bayshore Rd., #212
Palo Alto, CA 94303
USA

Email: ekr@networkresonance.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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