CoDTLS: DTLS handshakes over CoAP
draft-schmertmann-dice-codtls-00

Abstract

The Datagram Transport Layer Security protocol, DTLS, is usually transported over UDP. In Constrained Node Networks, there may be considerable limitations on the packet delivery rates and on practically usable packet sizes. Applications often can control the size and retransmission requirements of their data packets, but the DTLS handshake is out of scope for such application optimizations.

This specification defines how to perform a DTLS handshake on top of the CoAP protocol. The resulting DTLS connection may then be used for instance for transporting CoAP, or as a source of keying material. The latter case is particularly interesting if the CoAP exchanges transporting the DTLS handshake messages traverse CoAP proxies.

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 August 18, 2014.
1. Introduction

Constrained nodes in constrained node networks [I-D.ietf-lwig-terminology] often need robust security. The Constrained Application Protocol (CoAP), [I-D.ietf-core-coap], has chosen DTLS as the protocol to be used for communication security between CoAP nodes. DTLS was defined without special considerations for the capabilities of constrained nodes. The packets are relatively verbose, and the error control mechanisms and parameters work best in a typical Ethernet-like environment.
[I-D.hartke-core-codtls] proposes to mitigate these problems by running the DTLS handshake over CoAP. The present document discusses such a protocol in more detail, based on an initial implementation that was tested on MC13224-based constrained nodes (ARM7TDMI, 96 KiB RAM shared for both code and data filled from serial flash).

1.1. 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 RFC 2119 [RFC2119].

2. Stateless Compression

DTLS handshake messages are carried in CoAP bodies exchanged in CoAP requests and responses, possibly sliced up by the block protocol [I-D.ietf-core-block]. Each body is of a media type (Content-Format) that can contain multiple concatenated handshake messages. Along the lines of a compression scheme also defined in [I-D.hartke-core-codtls], the DTLS header is compressed as follows:

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

```
enum {
    change_cipher_spec(20), alert(21), handshake(22),
    application_data(23), (255)
} ContentType;
```

For T = 0, the initial header is followed by an 8-bit field for the "type". T = 1 compactly encodes the "type" value "alert" (21), T = 2 stands for "handshake" (22), T = 3 for "application_data" (23). (Not that "change_cipher_spec" is transported in a different way.)
For $V = 1$, this is followed by a two-byte field for the "version".
For $V = 0$, version is 254.255 (TLS 1.0), for $V = 2$ version is 254.253
(TLS 1.2), and $V = 3$ is reserved.

E encodes the epoch directly (0..4), 5 or 6 specifies that an 8 or
16-bit field followed. The value 7 is only allowed for handshake
packets following another handshake packet in a CoAP body, it means
the epoch is copied from the previous handshake packet in the same
body.

S encodes the sequence number. For values 1 to 6, the sequence
number is attached in 1 to 6 bytes, respectively. Value 0 means the
sequence number is not sent at all. Value 7 again refers to the
preceding handshake packet in the same body, adding one to the
sequence number used there.

L encodes the length. For values 1 and 2, the length is attached in
1 or 2 bytes. For value 3, the length is the remaining length of the
payload. Value 0 is reserved.

Within a handshake payload, multiple handshake messages are
concatenated, each preceded by a short header:

```
0 1 2 3 4 5 6 7
+---------------+
|     T      |  L |
+---------------+
```

T defines the handshake type (with two values special-cased: 32 for
change_cipher_spec and 33 for alert). L is the number of bytes that
follow and give the length; 0 stands for length 0, 1 and 2 for 1 or 2
bytes following giving the length, and 3 standing for the rest of the
handshake payload. Note that this format does not address the
fragmentation mechanism provided by DTLS, as the assumption is this
will not be required in DTLS handshakes performed by constrained
nodes.

3. Operation

To offer DTLS over CoAP, a CoAP server provides a resource that
accepts the media types defined in this section, identified by the
content-formats in Section 5.2. The specific URI of the resource is
up to the server. (In the examples, we are assuming it is at the
root of the server, i.e., no Uri-Path options are sent.) The client
learns the URI using the usual discovery processes, e.g., the CoRE
resource directory [I-D.ietf-core-resource-directory].
The client sends the client hello as an application/dtls-hello payload in a POST request to the DTLS URI of the server. The server MUST NOT accept Block1 options on requests carrying application/dtls-hello hello payloads unless it can already verify a cookie from the first block. (This means that both a cookieless ClientHello request, and the part of a cooked ClientHello that contains the cookie, need to fit into a single UDP packet of an appropriate size. The server needs to dimension its cookies accordingly.)

Once the client hello is accepted, the server builds state, as indicated in Location options in the 2.01 created response. The client switches to PATCHing that state using application/dtls-handshake messages. Instead of creating a separate resource for this, the client simply continues sending to the same DTLS resource. (Alternatively, the server could send back a URI for a new resource from the first successful POST exchange. This is not implemented in the current code, but will be required for operation through proxies.)

Figure 1 shows an example message exchange. The PATCH method is currently implemented as a POST, awaiting a PATCH method registration for CoAP.

Client                                Server
------                                ------
POST /                                 ---->
ClientHello               <-----   4.01 Unauthorized
                                 <-----   HelloVerifyRequest
POST /                                 ---->
ClientHello                                 2.01 Created /dCST0E
                                 <-----   ServerHello
                                 Certificate*
                                 ServerKeyExchange*
                                 CertificateRequest*
                                 <-----   ServerHelloDone
PATCH /dCST0E                 
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
Table 1 shows the implementation size of the current demonstrator implementation. Assuming that a CoAP library is already available, around 7.2 KiB are required for the entire CoDTLS implementation. (Note that the specific CoAP library in use required about 2134 bytes additional code to implement all the CoAP features required, including Block1, and some management code.) The implementation can operate with 2.0 KiB stack size.

<table>
<thead>
<tr>
<th>Size (KiB)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.41</td>
<td>ECC functions</td>
</tr>
<tr>
<td>0.95</td>
<td>AES modes (CCM + CMAC)</td>
</tr>
<tr>
<td>0.80</td>
<td>Storage management</td>
</tr>
<tr>
<td>0.79</td>
<td>Session management</td>
</tr>
<tr>
<td>0.15</td>
<td>PRF</td>
</tr>
<tr>
<td>1.78</td>
<td>CoAP Resource implementing handshake</td>
</tr>
<tr>
<td>0.32</td>
<td>Parse &amp; Send</td>
</tr>
</tbody>
</table>

Table 1: Code sizes in demonstrator implementation

4. Discussion

An alternative approach to the DTLS tunneling described here is to directly use the TLS handshake [RFC5246], as all prerequisites are already available in the reliability mechanisms provided by CoAP. However, this would require the addition of a DoS countermeasure, which in turn might be a useful component beyond the usage in this specification. Also, if it is desired to directly use the DTLS record layer after a CoAP-mediated handshake, the details of the transition from TLS to DTLS need to be specified.

5. IANA Considerations

This specification defines two new Internet media types [RFC6838]:

- application/dtls-hello
- application/dtls-handshake
This specification also defines the entries in the Content-Format registry for these new media types.

5.1. Media Types ("MIME Type")

The Internet media type [RFC6838] for a DTLS hello is application/dtls-hello.

Type name: application

Subtype name: dtls-hello

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: binary

Security considerations: See Section 6 of this document

Interoperability considerations: n/a

Published specification: This document

Applications that use this media type: Setup of DTLS sessions over CoAP

Additional information:
Magic number(s): n/a
File extension(s): n/a
Macintosh file type code(s): n/a

Person & email address to contact for further information:
Carsten Bormann
cabo@tzi.org

Intended usage: COMMON

Restrictions on usage: none

Author:
Carsten Bormann <cabo@tzi.org>

Change controller:
The IESG <iesg@ietf.org>
The Internet media type [RFC6838] for a DTLS handshake message is application/dtls-handshake.

Type name: application

Subtype name: dtls-hello

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: binary

Security considerations: See Section 6 of this document

Interoperability considerations: n/a

Published specification: This document

Applications that use this media type: Setup of DTLS sessions over CoAP

Additional information:
  Magic number(s): n/a
  File extension(s): n/a
  Macintosh file type code(s): n/a

Person & email address to contact for further information:
  Carsten Bormann
cabo@tzi.org

Intended usage: COMMON

Restrictions on usage: none

Author:
  Carsten Bormann <cabo@tzi.org>

Change controller:
  The IESG <iesg@ietf.org>

5.2. CoAP Content-Formats

  Media Type: application/dtls-hello

  Encoding: -
5.3. Link relation

TBD: There needs to be a way to find DTLS resources on a server, e.g., in a resource directory. This is usually done by defining an appropriate link relation.

6. Security Considerations

The security considerations of [RFC6347] and [I-D.ietf-core-coap] apply.

In its main part, this specification provides a way to carry around DTLS packets. Under the Internet Threat Model, those packets already traverse unsecured networks, so any attack that could be used to subvert DTLS packets sent over CoAP could already be used to subvert the DTLS packets when sent over traditional transports. Obviously implementers still need to implement the DTLS state machine fully. In addition, if the CoAP exchanges run over unsecured channels, those channels will need to be made robust to the usual attacks.

If the option is chosen to derive the Finished MACs from the CoAP representation, much more substantial security analysis is required, and this section will need to discuss its security considerations.

7. Acknowledgements

Olaf Bergmann is a co-author of the base specification this implementation has been derived from.

8. References
8.1. Normative References

[I-D.ietf-core-block]

[I-D.ietf-core-coap]


8.2. Informative References

[I-D.hartke-core-codtls]

[I-D.ietf-core-resource-directory]

[I-D.ietf-lwig-terminology]


Authors’ Addresses
Lars Schmertmann  
Universitaet Bremen Tzi  
Postfach 330440  
Bremen  D-28359  
Germany  
Email: lars@tzi.de

Klaus Hartke  
Universitaet Bremen Tzi  
Postfach 330440  
Bremen  D-28359  
Germany  
Phone: +49-421-218-63905  
Email: hartke@tzi.org

Carsten Bormann (editor)  
Universitaet Bremen Tzi  
Postfach 330440  
Bremen  D-28359  
Germany  
Phone: +49-421-218-63921  
Email: cabo@tzi.org