Securing IoT Applications with mbed TLS

Hannes Tschofenig

Part#4: Datagram Transport Layer Security (DTLS)
**Why DTLS?**

- Simple: Provide communication security for datagram protocols.
- For example: Constrained Application Protocol (CoAP) running over UDP
- Useful also for securing non-IP-based communication, such as low-power WAN protocols.
- We will talk about the DTLS internals and then do a hands-on with the Keil µVision 5 IDE using the PSK example from webinar #1 and the hardware-based RNG support, as developed in webinar #3, as a starting point. Then, DTLS support will be added.
DTLS
DTLS

• The design of DTLS is intentionally similar to TLS.

• DTLS 1.2 is documented in RFC 6347. Version 1.3 is in development.

• DTLS operates on top of an unreliable datagram transport
  • most importantly over UDP, but
  • works also over non-IP!
Transport Layer Comparison

Or: Why is UDP is “lightweight”?

<table>
<thead>
<tr>
<th>Features</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Connection establishment and teardown</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>2. Congestion Control</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>3. Fragmentation</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>4. Rate Control</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>5. Return-Routability Check</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Guidance for lightweight implementations of TCP in the Internet of Things devices is available.
Fragmentation and Maximum Transmission Unit (MTU)

Concepts

• UDP-based applications need to have awareness of the MTU supported by the path (the so-called Path MTU).

• Path MTU discovery
  • originally defined in RFC 1191
  • Later re-defined with Packetization Layer Path MTU Discovery (PLPMTUD), defined in RFC 4821

• Minimum MTU for use of IP:
  • 576 bytes for IPv4, and
  • 1280 bytes for IPv6

Mbed TLS APIs

mbedtls_ssl_set_mtu() – configure MTU by application

mbedtls_ssl_get_record_expansion() – determine overhead for DTLS headers and encryption

mbedtls_ssl_get_max_out_record_payload() – maximum application data size of a DTLS connection
Features added with Mbed TLS v.2.13

- Grouping outgoing handshake messages in a single datagram
- Support for fragmentation of outgoing handshake messages
- Reordering handshake packets that have been received out of order

Further reading:
- Release of Mbed TLS v.2.13
- UDP Usage Guidelines (RFC 8085)
- Mbed TLS over low-bandwidth, unreliable datagram networks
DTLS 1.2 Stack

- Four protocols use the record layer, namely
  - Handshake Protocol
  - Change Cipher Spec Protocol (not shown)
  - Alert Protocol (not shown)
  - Application Data Protocol
Record Layer

**DTLS**

*Sequence numbers* to allow decryption of individual records. Note: No stream ciphers supported. *Epoch* to recognize key changes.

**TLS**
Handshake Protocol

**DTLS**

*Sequence numbers* to detect reordering.

*Fragmentation Offset and Length* for avoiding IP (or lower layer) fragmentation.

**TLS**
Retransmission Timers
Concept and Implementation

Registering Timer Callbacks

• In mbed TLS a pair of callbacks is provided with `mbedtls_ssl_set_timer_cb`:
  • `void mbedtls_timing_set_delay(void *data, uint32_t int_ms, uint32_t fin_ms);`
  • `int mbedtls_timing_get_delay(void *data);`
• Note that the retransmissions happen at the levels of entire "flights" rather than individual messages.

DTLS Retransmission Mechanism
Retransmission Timers

How long should the sender wait till retransmission happens?

RFC 6347

• .... initial timer value of 1 second
• Double the value at each retransmission
• up to no less than 60 seconds.

RFC 7925

• ... initial timer value of 9 seconds
• Double the value at each retransmission
• up to no less then 60 seconds.

The minimum and maximum can be set using `mbedtls_ssl_conf_handshake_timeout()`.
Denial of Service
Handling integrity check failures

**TLS**

- The connection is immediately terminated when a record is received that does not pass the integrity check,
- Injecting bad records in TCP is just as hard as injecting a TCP RST to tear down the connection.
- This denies attackers an opportunity to do more than one guess at the message authentication key, while not introducing any new DoS vectors.

**DTLS**

- Injecting bad records is very easy; an attacker only needs to know the source and destination IP and port.
- [Section 4.1.2.7](#) of the DTLS 1.2 spec recommends not to tear down the connection.
Denial of Service, cont.

- It is possible to set a limit to the number of bad records before the connection is torn down.

- This is controlled by
  - the compile-time flag `MBEDTLS_SSL_DTLS_BADMAC_LIMIT` (enabled by default), and
  - the run-time setting `mbedtls_ssl_conf_dtls_badmac_limit()` (unlimited by default).
Denial of Service Settings

Avoiding Amplification Attacks

• A small one-shot message requires state allocation, computation and leads to a large response.

• Perform return-routability check with HelloVerifyRequest (HVR) message.
  • Message is stateless at the server
  • State encapsulated in cookie and must be returned by the client in 2nd ClientHello.
  • Adds one extra roundtrip

• HVR can be disabled via compile-time flag `MBEDTLS_SSL_DTLS_HELLO_VERIFY`
Replay Protection

• For application data protection DTLS provides limited additional properties besides authentication, integrity and confidentiality.

• Just like UDP, DTLS delivers datagrams.

• Unlike UDP, DTLS can detect and discard duplicated datagrams, if needed.

• In Mbed TLS, this feature is controlled by
  
  • the compile-time flag `MBEDTLS_SSL_DTLSANTI_REPLAY`,
  
  • and the run-time setting `mbedtls_ssl_conf_dtls_anti_replay()`,
  
  • both enabled by default.
DTLS Integration into the Application Code

1. Use `mbedtls_net_connect()` with `MBEDTLS_NET_PROTO_UDP` (instead of `MBEDTLS_NET_PROTO_TCP`).

   You may also need to give your stack extra time for setting up the networking interface.

2. Use `mbedtls_ssl_config_defaults()` with `MBEDTLS_SSL_TRANSPORT_DATAGRAM` (instead of `MBEDTLS_SSL_TRANSPORT_STREAM`).

3. Potentionally adjust application interaction with `mbedtls_ssl_read()` / `mbedtls_ssl_write()` interaction.
DTLS Configuration: Config.h

• Add `MBEDTLS_SSL_PROTO_DTLS` to enable DTLS support

• (Note: On the server-side you also have to enable cookie support and implement the appropriate callbacks.)
Hardware and Testbed Setup
Hands-on code now uses CMSIS-RTOS2 and Keil RTX5.
Hands-on
Conclusion
Summary

• DTLS is popular in IoT deployments. It is, for example, used to secure CoAP.

• The good news: DTLS is similar to TLS.

• Even better: It is easy to configure timer callbacks and other DTLS options.
The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks