A DTLS 1.2 Profile for the Internet of Things

draft-hartke-dice-profile-03

Klaus Hartke, Hannes Tschofenig

Agenda

1. Communication Model
2. Scope: CoAP vs. non-CoAP
3. PSK Ciphersuite & PFS
4. Raw Public Key Mode
5. Certificate Mode
6. Error Handling
7. Session Resumption
8. Compression
9. Keep-Alive Extension
10. Downgrading Attack
11. Privacy
12. Random Numbers
13. RFC 6066: TLS Extensions

Communication Model

- Current focus:
  - IoT device to server (with server being unconstrained)
  - Unicast only

    (Multicast is covered in another document.)
- Is this a good focus of the document?
- Should we also cover the constrained server model?

Scope: CoAP vs. non-CoAP

- CoAP influences the choices since a number of ciphers are listed as mandatory-to-implement in the CoAP specification.

PSK Ciphersuite & PFS

- PFS: Compromise of long-term key does not compromise past session keys.
- CoAP specifies TLS_PSK_WITH_AES_128_CCM_8 as MTI.
  [I-D.sheffer-tls-bcp] recommends the use of PFS.
- Should we follow the recommendation?
- PSK Identities: RFC 4279 requires implementations to support PSK identities up to 128 octets and PSKs up to 64 octets.
  - Not useful in our context. Remarks?

Raw Public Key Mode

- Ciphersuite mismatch again:
  - CoAP ays TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
  - [I-D.sheffer-tls-bcp] says TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- RSA vs. ECC: ECC is more suitable for constrained devices but there may be an IPR challenge.

- Recommendation?

Certificate Mode

- What identifiers to use in the certificate?
  - FQDN (for the server-side)?
  - EUI-64 (for the client-side)?
- No CRLs / No OCSP / No TAMP ?
  - Use (not further elaborated) firmware update mechanism?
- What about time support (for certificate verification)?
- Cached Info extension assumed to lower the overhead.
- Depth of certificate chain?

Error Handling

- TLS allows error to be communicated using the Alert Protocol. Not all error messages are needed in all cases.
- Proposal for a sub-set of the error messages.
- Sometimes difficult to take meaningful actions due to the lack of user interface.
- Any assumptions about logging?

Session Resumption

- Session resumption reduces the number of messages and the computational overhead.
  - Drawback is the additional codebase.
- Suggestion is to make support for it mandatory.
- RFC 5077 is, however, not utilized.

TLS Compression

- [I-D.sheffer-tls-bcp] recommends to always disable DTLS-level compression due to attacks.
- Not used in IoT deployments → suggest to omit.

Keep-Alive Extension

- RFC 6520 [RFC6520] defines a heartbeat mechanism to test whether the other peer is still alive. The same mechanism can also be used to perform path MTU discovery.

- QUESTION: Do IoT deployments make use of this extension?

Downgrading Attack

- CoAP demands version 1.2 of DTLS.
- [I-D.bmoeller-tls-downgrade-scsv] is therefore also not applicable.
- TLS renegotiation attack [RFC5746]
  - Clients MUST respond to server-initiated renegotiation attempts with an Alert message (no_renegotiation)
  - Clients MUST NOT initiate them.

Privacy

- Mostly concerned about identifiers used in the TLS protocol (e.g., PSK identifier, certificate payloads).
- PFS is discussed elsewhere in the document.
- Authentication and the use of the same credential with different services obviously creates privacy problems.
- Anything else?

Random Numbers

- TLS requires random numbers.
- There have been problems with random number generation on embedded devices [Heninger].
- Is there anything that could be said?
- Is there a requirement for hardware support?

RFC 6066: TLS Extensions

- **Client Certificate URLs**: Allows avoiding to send client-side certificates. Send URLs instead.
- **Trusted CA Indication**: Allows client to indicate what trust anchor it supports.
- **Truncated MAC extension**: Reduces the size of the MAC at the Record Layer.
- **Server Name Indication**: Mechanism for a client to tell a server the name of the server it is contacting.
- **Maximum Fragment Length Negotiation**: Lowers the MFL for the Record Layer from 2^14 bytes to 2^9 bytes.