DTLS 1.2 Status

Eric Rescorla
RTFM, Inc.
ekr@rtfm.com

Current State

- New draft: `draft-ietf-tls-rfc4347-bis-02.txt`

- Requirement to handle data from old epochs
- Clarified Handling of data/handshake ordering during rehandshake
- Clarified handling of sequence numbers and epochs
- Clarified PMTU handling for stream-type transports
- Clarified behavior of cookies with key changes
- Are we done yet?

Handling data from old epochs

- Issue raised by Michael Tuexen
- Some implementations reject data from old epochs during rehandshake
  - This can create stalls
- New requirement:

Until the handshake has completed, implementations MUST accept packets from the old epoch.

Handling of sequence numbers

- Raised by Robin Seggelman
- Text isn't clear about sequence numbers and rehandshake
- Intention: each epoch has its own seqno
  - Separate replay checks for each epoch
- Changed text to clarify

PMTU Handling

- Issue raised by Michael Tuexen
- Previous text said to assume stream transports had "infinite PMTU"
  - This doesn't make any sense
- New text:

  For DTLS over TCP or SCTP, which automatically fragment and reassemble datagrams, there is no PMTU limitation. However, the upper layer protocol MUST NOT write any record that exceeds the maximum record size of 2^{14} bytes.

Data/handshake reordering

- Issue raised by ???
- What happens if handshake and data packets are reordered
  - Can happen from the final agent to speak
  - New handshakes and rehandshakes are different
- New text:

  Note that in the special case of a rehandshake on an existing association, it is safe to process a data packet immediately even if the CSS or Finished has not yet been received provided that either the rehandshake resumes the existing session or that it uses exactly the same security parameters as the existing association. In an other case, the implementation MUST wait for the receipt of the Finished to prevent downgrade attack.

Clarified cookie handling with key changes

- Issue raised by ???
- If the signing key changes, you can get multiple HelloVerifyRequests
- New text:

  If a server receives a ClientHello with an invalid cookie, it SHOULD treat it the same as a ClientHello with no cookie. This avoids race/deadlock conditions if the client somehow gets a bad cookie (e.g., because the server changes its cookie signing key). Note to implementors: this may results in clients receiving multiple HelloVerifyRequest messages with different cookies. Clients SHOULD handle this by sending a new HelloVerify in response to the new HelloVerifyRequest.

Next Steps

- Everything here was minor
- And mostly clarifications of minor issues
- Ready for WGLC?