TLS 1.3

draft-ietf-tls-tls13-21

Eric Rescorla
Mozilla

ekr@rtfm.com
Agenda

- Middlebox issues (PR#1091)
- close_notify and half-close (PR#1092)
- SNI and resumption (PR#1080)
Middlebox issues

• Some middleboxes appear to be sad when you negotiate TLS 1.3

• Error rates (Firefox Beta versus Cloudflare)
  – 2.2% for TLS 1.2
  – 3.9% for TLS 1.3

• This means you need fallback to deploy TLS 1.3

• Proposal: make TLS 1.3 look like TLS 1.2 resumption
Emulate TLS 1.2 resumption part 1: Always

- Move version negotiation entirely into supported_versions
  - ServerHello.version == 0x0303 (TLS 1.2)
- Restore the missing session_id and compression fields in ServerHello
- Change the post-ServerHello record layer version to 0x0303
- Merge HRR and ServerHello into a single message with the semantics distinguished by a special ServerHello.Random value.
- Implementations MUST ignore ChangeCipherSpec during handshake
Emulate TLS 1.2 resumption part 2: Compatibility Mode

- The client sends a fake session_id and the server echoes it
- The server sends ChangeCipherSpec messages after ServerHello/HelloRetryRequest (so that the middlebox ignores any "encrypted" data afterwards), and the client sends ChangeCipherSpec after ClientHello. ClientHello
  - Server’s ChangeCipherSpec SHOULD be sent when the client sends the fake session_id (not in PR#1091)
Issues Raised

• Should we only have compatibility mode?
  – We don’t need this for TLS 1.3/QUIC or DTLS
  – It’s not *entirely* clear we need the client-side CCS
  – At some point we may be able to stop sending server-side CCS

• Should we require the client to enforce CCS cardinality?
  – Require CCS be present
  – Require CCS to appear only once
  – This complicates the implementation of the receiver
Interlude: Chrome Data from David Benjamin

Firefox data hopefully coming soon
Chrome initial draft 18 deployment

• No evidence of TLS 1.3 ClientHello intolerance. supported_versions and GREASE did their job.

• TLS 1.3 ServerHello was a very different story.

• Successful handshakes to a TLS-1.3-capable service in Chrome beta:
  – TLS 1.2 - 98.3%
  – Draft 18 - 92.3%

• Middleboxes are intolerant to TLS 1.3 ServerHello. This violates TLS versioning rules: ClientHello is invariant, rest is version-specific.
Middleboxes

• TLS-terminating middleboxes generally work fine with TLS 1.3.
  – "Just" a server and client connected back-to-back. Server half negotiates TLS 1.2, client half only offers what it implements.

• Other middleboxes process TLS without terminating it. They then try to parse unknown version-specific messages and break.

• This is an oversimplified picture. A lot of middleboxes are a mix of the two strategies.
TLS 1.3 variants, round one

- "Experiment" → PR 1091 without the record-layer version change.

- We tested what we could locally, then performed A/B tests in the wild (1-RTT).

- Successful handshakes to a TLS-1.3-capable service in Chrome beta:
  - TLS 1.2 - 99.2%
  - Draft 18 - 95.8%
  - PR 1051 - 90.3%
  - Experiment - 98.2%
  - Experiment w/o client session ID - 95.4%

- Lots of user reports confirmed problems with each variant, including some for Experiment.
TLS 1.3 variants, round two

- Reproduced Experiment problems and changed record version for round 2.
- Successful handshakes to a TLS-1.3-capable service in Chrome beta:
  - TLS 1.2 - 98.6%
  - PR 1091 - 98.8%
- Corroborated by HTTP-level metrics.
- No user reports of problems thus far.
close_notify and half-close (PR#1092)

• Right now close_notify is sorta full-close
  – Receiver has to flush outstanding untransmitted data
  – And immediately send close_notify

• Not ideal
  – Lots of implementations don’t do this
  – Data may already be in flight
  – Reasons people may want half-close
  – Not clear why it’s there in the first place

• Proposal
  – Allow implementations to keep sending after receiving close_notify
  – Backward compatible with previous behavior
SNI and Resumption (PR#1080)

- RFC 6066 totally prohibits resuming with different SNIs
- Implementations aren’t good about following this
- Proposal
  - Client MUST only resume if SNI is in certificate
  - Client SHOULD only resume if the SNI is the same
    * No reason to think it will work anyway
  - Leaves the door open for the server to say that you can resume with different SNI
- Not entirely clear how to analyze this
  - But it looks like we already have these problems with existing implementations and HTTP coalescence
Next step

- Merge outstanding PRs (these and some editorial stuff)
- Issue -22
- Targeted WGLC?