TLS Application-Layer Protocol Settings

draft-vvv-tls-alps

IETF 108 (virtual)

Half-RTT Problem

- TLS 1.3 supports “half-RTT data”, i.e. the data sent by the server immediately after ServerHello..Finished flight
- Can be used in protocols like HTTP/2 where the server sends SETTINGS first
- For TLS over TCP, a lot of widely deployed server TLS implementations do not support half-RTT data.
- A quick scan of popular HTTP/2 servers has revealed very few people actually sending SETTINGS in half-RTT.
- Introducing half-RTT changes the I/O pattern of the protocol, which can potentially lead to interoperability issues
- Even when a server *does* send half-RTT data, the client needs to know that for a fact. Otherwise, blocking on receiving half-RTT data would stall for an RTT.

ALPS extension

ALPS approach: put settings (of both client and server) into an extension.

Advantages:

1. Provides a guarantee to the application that it always has the peer’s settings
2. Solves the half-RTT problem, allowing the use of various HTTP/2 extensions during the first flight of requests, including:
    a. Extended CONNECT for WebSocket (RFC 8441)
    b. Client hint reliability (draft-davidben-http-client-hint-reliability)
    c. Opting out of header compression
3. Transparently deals with the problem of retaining peer’s settings for 0-RTT.

ALPS semantics

- Both client and server send an opaque blob with their settings (*ALPS values*)
- ALPS is declarative, and not a negotiation: client settings cannot depend on server settings and vice versa.
- Settings are encrypted with the handshake keys.
- 0-RTT is handled by storing both sets of settings inside the TLS ticket.
- If the settings provided by both the client and the server match, 0-RTT proceeds with settings from the previous sessions.

Full handshake with ALPS

ClientHello flight

ClientHello includes an empty application_settings extension.

ServerHello flight

EncryptedExtensions includes application_settings with the server settings.

Client’s Finished flight

ClientApplicationSettings message appears before client’s Certificate message (Finished message if no client certificate present).

Alternatives considered

The main alternative is fixing half-RTT data and using it instead.

This doesn’t solve the problem fully, as an explicit signal of half-RTT data support is still needed, and that signal would have to come at the TLS layer.

This also doesn’t provide semantics for retaining application settings for 0-RTT.

Open questions

- 0-RTT mismatch handling is currently simplified compared to how QUIC and HTTP/3 handle it.
  - Simplifies the API design
  - People might want to have the flexibility that QUIC provides
- No support for settings that are not retained across multiple connections.
  - HTTP currently does not have settings that change in that manner.
  - QUIC does have transport parameters of that nature, but QUIC wouldn’t be using ALPS

Discussion