Application Layer ATLS

Friel, Barnes, Pritikin
cisco.com
Related drafts

• draft-tschofenig-layered-tls-00
  • Layered DTLS/TLS

• draft-ietf-anima-bootstrapping-keyinfra-09
  • Bootstrapping Remote Secure Key Infrastructures (BRSKI)
Motivation – Device Bootstrap

• Installer unboxes and powers up a device that needs to establish a secure connection to a cloud service
• There is a TLS Terminating middlebox between the device and the cloud
• The device cannot do TLS certificate validation as it does not yet trust the private CA root used by the middlebox
• The device fails to connect to the cloud service
Potential Solutions

• Address via explicit, expensive configuration by operator or vendor
  1. Operator-specific configuration installed on device prior to shipping to customer – expensive and limited applicability
  2. Operator disables TLS interception on middlebox for specific domains – does not meet customer InfoSec requirements
  3. Operator manually installs private root on device – expensive and time consuming

• Device logic to work around infrastructure restrictions
  4. Device establishes an application layer encrypted channel with cloud service
    • Device could simply download middlebox trust information using this channel and then switch to network TLS
Application layer encrypted channel options

• Multiple potential solutions including but not limited to
  1. Define a handshake using JOSE/JWE/JWT
  2. Define an encapsulation for Noise
  3. Reuse TLS stack and exchange TLS Records

• Design goal: do this with as few lines of code as possible and minimise library dependencies

• Proposal rationale: device already calls TLS stack APIs at network layer, just get the device to call the APIs twice.
Transporting TLS in HTTP

- Transport (flights of) TLS Records in HTTP message bodies
- Lowest common denominator and greatest chance of traversing middleboxes
- ATLS server addressed with HTTPS URI vs. host/port
  - Could host application directly
  - Or do something more elaborate, like an HTTP reverse proxy
- Top-level applications suited to things that “look like” HTTP
  - Could alternatively run over a websocket
Architecture

Working proof-of-concept:
- OpenSSL C client talking ATLS over HTTPS to
- Java JSSE Spring Web Server
- Easy to consume OpenSSL and JSSE APIs summarised in draft Appendix
Discussion Points

• Turtles all the way down
  • Will middleboxes block encrypted application layer data?
  • Generally applicable for any application layer crypto: JOSE/JWE, Noise, etc.

• Use ATLS just for handshake or for application data too
  • Could just use ATLS for 2xRTTs and then use RFC5705 (Keying Material Exporters) and switch to RFC8188 (Encrypted Content-Encoding)

• HTTP Transport Reliability
  • Transport DTLS records in HTTP bodies
  • Just using ATLS for handshake 2xRTTs also mitigates somewhat

• Why not use HTTP CONNECT tunnels?
  • HTTP CONNECT and TLS Intercept are logically different functions
  • HTTP CONNECT tunnel establishment could succeed but there could still be a TLS middlebox behind the HTTP Proxy in the local network

• Server -> Client signalling
  • Upgrade to a websocket if necessary