Overview of TLS v1.3
What’s new, what’s removed and what’s changed?

OWASP
The Open Web Application Security Project
• Andy Brodie
  – Solution Architect / Principal Design Engineer.
  – On Worldpay eCommerce Payment Gateways.
  – Based in Cambridge, UK.

• I am neither a cryptographer nor a mathematician!
  – This means no maths in this presentation.
• History & Background.
• What’s Been Removed.
• What’s New & Changed.
  – Cipher Suites.
  – Handshake Changes.
  – Hashed-Key Derivation Function.
  – Session Resumption.
• Summary.
The Goals and Basics of TLS

HISTORY & BACKGROUND
# How SSL became TLS

<table>
<thead>
<tr>
<th>When</th>
<th>Who</th>
<th>What</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>Netscape</td>
<td>SSL 1.0 designed.</td>
<td>Never published as security flaws were found internally.</td>
</tr>
<tr>
<td>1995</td>
<td>Netscape</td>
<td>SSL v2.0 published.</td>
<td>Flaws found pretty quickly, which led to...</td>
</tr>
<tr>
<td>1996</td>
<td>Netscape</td>
<td>SSL v3.0 published.</td>
<td>SSL becomes ubiquitous.</td>
</tr>
<tr>
<td>1999</td>
<td>IETF</td>
<td>TLS v1.0 published (SSL v3.1)</td>
<td>Incremental fixes, political name change and IETF ownership.</td>
</tr>
<tr>
<td>2006</td>
<td>IETF</td>
<td>TLS v1.1 published (SSL v3.2)</td>
<td>Incremental fixes and capabilities.</td>
</tr>
<tr>
<td>2008</td>
<td>IETF</td>
<td>TLS v1.2 published (SSL v3.3)</td>
<td>What we should all be using!</td>
</tr>
<tr>
<td>2014</td>
<td>IETF</td>
<td>TLS v1.3 draft 1 (SSL v3.4)</td>
<td></td>
</tr>
</tbody>
</table>
A Client and Server can have a secure conversation over an insecure medium having never met before.
What is a secure conversation?

- **Confidentiality**
  - Conversation needs to be encrypted.
  - Stop interception of conversations.

- **Integrity**
  - Client & Server must be able to detect message tampering.
  - Prevent MITM attacks.

- **Authentication**
  - Client needs to trust they’re talking to the intended server.
  - Stop impersonation attacks.
TLS achieves this using various techniques...

- **Confidentiality**
  - Symmetric key encryption for application data.
  - Typically Advanced Encryption Standard (AES).

- **Integrity**
  - Authenticated Encryption with Additional Data (AEAD).
  - Usually AES-GCM (Galois/Counter Mode).

- **Authentication**
  - X509 certificates signed by a mutually trusted third party.
  - Typically server authenticated only.
Flow of messages in a TLS conversation

1. Open Socket
2. Handshake
3. Application Data
4. Alert
5. Close Socket
Flow of messages in a TLS conversation

- **Handshake**
  - Agree a cipher suite.
  - Agree a master secret.
  - Authentication using certificate(s).

- **Application Data**
  - Symmetric key encryption.
  - AEAD cipher modes.
  - Typically HTTP.

- **Alerts**
  - Graceful closure, or
  - Problem detected.
TLS V1.3

Key Goals of TLS v1.3:

- **Clean up** - Remove unsafe or unused features.
- **Security** - Improve security w/modern techniques.
- **Privacy** - Encrypt more of the protocol.
- **Performance** – 1-RTT and 0-RTT handshakes.
- **Continuity** – Backwards compatibility.
WHAT’S REMOVED IN TLS V1.3?
What’s removed in TLS v1.3

- Key Exchange
  - RSA

- Encryption algorithms:
  - RC4, 3DES, Camellia.

- Cryptographic Hash algorithms:
  - MD5, SHA-1.

- Cipher Modes:
  - AES-CBC.

- Other features:
  - TLS Compression & Session Renegotiation.
  - DSA Signatures (ECDSA ≥ 224 bit).
  - ChangeCipherSpec message type & “Export” strength ciphers.
  - Arbitrary/Custom (EC)DHE groups and curves.
This has mitigated quite a few attacks...

RC4
- Roos’s Bias 1995
- Fluhrer, Martin & Shamir 2001
- Klein 2005
- Combinatorial Problem 2001
- Royal Holloway 2013
- Bar-mitzvah 2015
- NOMORE 2015

RSA PKCS#1 v1.5
- Bleichenbacher 1998
- Jager 2015
- DROWN 2016

Renegotiation
- Marsh Ray Attack 2009
- Renegotiation DoS 2011
- Triple Handshake 2014

AES-CBC
- Vaudenay 2002
- Boneh/Brumley 2003
- BEAST 2011
- Lucky13 2013
- POODLE 2014
- Lucky Microseconds 2015

3DES
- Sweet32

Compression
- CRIME 2012

MD5 & SHA1
- SLOTH 2016
- SHAttered 2017
WHAT’S NEW AND CHANGED?
What’s New and Changed?

• Cipher Suite specification.
• Handshake Encryption.
• Post-Handshake Client Authentication.
• Key Schedule Generation
• Session Handling.
TLS v1.2 provides 37 Cipher Suites

- TLS 1.2 specifies **37** cipher suites.
  - Add previous versions in: **319** cipher suites.
• TLS v1.3 supports 5 cipher suites.
  – TLS_AES_128_GCM_SHA256
  – TLS_AES_256_GCM_SHA384
  – TLS_CHACHA20_POLY1305_SHA256
  – TLS_AES_128_CCM_SHA256
  – TLS_AES_128_CCM_8_SHA256
What happens to key exchange and authentication then?

• Key Exchange algorithms:
  – DHE & ECDHE
    • Only 5 ECDHE curve groups supported
    • Only 5 DHE finite field groups supported
  – Pre-Shared Key (PSK)
  – PSK with (EC)DHE

• Digital Signature (Authentication) algorithms:
  – RSA (PKCS#1 variants)
  – ECDSA / EdDSA
• The handshake has three goals:
  – Agree a cipher suite.
  – Agree a master secret.
  – Establish trust between Client & Server.

• Optimise for the most common use cases.
  – Everyone* wants a secure conversation.
  – Same cipher suites used across websites repeatedly.
  – Clients connect to the same sites repeatedly.

* ok, almost everyone!
Three Stages of a TLS 1.3 Handshake

Key Exchange

Server Parameters

Authentication
Client now makes assumptions about server support.

- **Client sends:**
  - Cipher Suite options.
  - List of supported groups/curves.
  - (EC)DHE Key Share(s).

- **Server sends:**
  - Cipher suite selection.
  - (EC)DHE Key Share

- **Client and Server now share a key.**
Client now makes assumptions about server support.

- Server sends:
  - Encrypted Extensions
    - Server Name
    - Message Length
    - ...and optionally many more
  - Certificate Request
    - Supported signature algorithms.
Client now makes assumptions about server support.

- Server sends:
  - Certificate.
  - Proof of private key possession.
  - Finished.
  - Application Data

- Client responds:
  - Certificate.
  - Proof of private key possession.
  - Finished.
GENERATING KEYS USING HKDF
• TLS <= v1.2 defines PRF algorithm.

• TLS v1.3 replaces this with HKDF.
  – HKDF encapsulates how TLS uses HMAC.
  – Re-used in other protocols.
  – Separate cryptographic analysis already done.

• Provides 2 functions:
  – Extract - create a pseudo-random key from inputs.
  – Expand - create more keys from the extract output.

• HMAC is integral to HKDF.
  – HMAC requires the Cryptographic Hash algorithm specified in the cipher suite (SHA256 or SHA384).
TLS <= v1.2 Creating Key Material from a master secret

Pre-master Secret

>= 46 bytes

Master Secret

48 bytes

Key Material

∞

PRF

Client MAC Key

Server MAC Key

Client Write Key

Server Write Key

Client Write IV

Server Write IV
TLS v1.3 Key Schedule Generation

PSK

Early Secret

Derive Secret

Binder Key
Client Early Traffic Secret
Early Exporter Master Secret

Handshake Secret

(EC)DHE

Derive Secret

Client Traffic Handshake Secret
Server Traffic Handshake Secret

Handshake Secret

Derive Secret

Client Application Traffic Secret 0
Server App Traffic Secret 0
Exporter Master Secret
Resumption Master Secret

Master Secret

Derive Secret

HKDF-Expand-Label
Derive-Secret Fixed
HKDF-Extract

Client Application Traffic Secret N
Server App Traffic Secret N
Nonce N
PSK Ticket N
What’s the difference?

PRE-SHARED KEYS AND SESSIONS
Why do we need sessions?

• Full handshakes are expensive.
  – Key generation.
  – Server (& Client) Authentication.

• Many HTTP clients need it.
  – Download web page resources (JS, CSS, images).
  – Dynamic web pages (XHR).
  – May not be feasible to keep connection open.
How do we establish a PSK?

• Out-of-band
  – Added to TLS in 2006 via RFC4279.

• During Handshake
  – Client announces it supports session resumption.
  – Provides a PSK *identity* during handshake.

• After handshake, Server sends “New Session Ticket”
  – Contains PSK identity, nonce and max age.
  – The PSK is derived from master secret.
  – Server can send multiple tickets.
So, TLS v1.3 supports PSK-based session resumption.
What about Zero Round Trip Time (0-RTT)?

• PSK means the key is known to both sides.
  – Does this mean Client can send data immediately?
  – Can we have a zero round trip time handshake?

Yes, we can!

• But...
  – No forward secrecy for the “early data” sent by client.
  – No guarantees of non-replay.
So, TLS v1.3 supports PSK-based session resumption.
• In initial session server sends `NewSessionTicket`.
  – Adds `max_early_data` extension.

• Client connects to resume:
  – Sends empty `early_data` extension in `ClientHello`.
  – Includes early application data in first flight.
  – Carries on passing early application data until...
  – ... server responds with `early_data`.
  – Client echoes `end_of_early_data` to acknowledge.
Extensions... Extensions everywhere!

BACKWARDS COMPATIBILITY
• Backwards compatibility is important
  – TLS v1.3 clients need to talk to TLS v1.2 servers.
  – TLS v1.2 clients need to talk to TLS v1.3 servers.
  – Structure of Hello messages is maintained.

• 21 extensions referenced in the RFC.
  – 12 in other RFCs!
All the extensions

<table>
<thead>
<tr>
<th>Extension</th>
<th>TLS 1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>server_name [RFC6066]</td>
<td></td>
</tr>
<tr>
<td>max_fragment_length [RFC6066]</td>
<td></td>
</tr>
<tr>
<td>status_request [RFC6066]</td>
<td></td>
</tr>
<tr>
<td>supported_groups [RFC7919]</td>
<td></td>
</tr>
<tr>
<td>signature_algorithms [RFC5246]</td>
<td></td>
</tr>
<tr>
<td>use_srtp [RFC5764]</td>
<td></td>
</tr>
<tr>
<td>heartbeat [RFC6520]</td>
<td></td>
</tr>
<tr>
<td>application_layer_protocol_negotiation [RFC7301]</td>
<td></td>
</tr>
<tr>
<td>signed_certificate_timestamp [RFC6962]</td>
<td></td>
</tr>
<tr>
<td>client_certificate_type [RFC7250]</td>
<td></td>
</tr>
<tr>
<td>server_certificate_type [RFC7250]</td>
<td></td>
</tr>
<tr>
<td>padding [RFC7685]</td>
<td></td>
</tr>
<tr>
<td>key_share</td>
<td></td>
</tr>
<tr>
<td>pre_shared_key</td>
<td></td>
</tr>
<tr>
<td>psk_key_exchange_modes</td>
<td></td>
</tr>
<tr>
<td>early_data</td>
<td></td>
</tr>
<tr>
<td>cookie</td>
<td></td>
</tr>
<tr>
<td>supported_versions</td>
<td></td>
</tr>
<tr>
<td>certificateAuthorities</td>
<td></td>
</tr>
<tr>
<td>oid_filters</td>
<td></td>
</tr>
<tr>
<td>post_handshake_auth</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH</td>
<td>Client Hello</td>
</tr>
<tr>
<td>SH</td>
<td>Server Hello</td>
</tr>
<tr>
<td>EE</td>
<td>Encrypted Extensions</td>
</tr>
<tr>
<td>CT</td>
<td>Certificate</td>
</tr>
<tr>
<td>CR</td>
<td>Certificate Request</td>
</tr>
<tr>
<td>NST</td>
<td>New Session Ticket</td>
</tr>
<tr>
<td>HRR</td>
<td>Hello Retry Request</td>
</tr>
</tbody>
</table>
• Protocol Version is mentioned in every message.
  – Now deprecated/fixed to old version values
  – Handshake claims 1.2, App Data claims 1.0.
  – New extension specifies list of supported versions.

• Fixed Values to prevent downgrade attacks.
  – Server “Random” has fixed last 8 bytes
    • DOWNGRD[0x01] for TLS 1.2 clients.
    • DOWNGRD[0x00] for <= TLS 1.1 clients.
• Removed
  – Anything that was unused, unsafe or didn’t offer value.
  – Mitigated lots of attacks.

• Added
  – Handshake encryption.
  – 1-RTT and 0-RTT PSK / Session Resumption.

• Changed
  – Cipher Suites.
  – PSK / Sessions.
  – Post-Handshake Client Authentication.
  – PRF now HKDF.
THANK YOU FOR LISTENING!