Protecting the TLS Handshake

Tom Ritter and Daniel Kahn Gillmor

May 2014

Metadata is useful information

- Metadata is valuable information for censorship and surveillance regimes
  - Major surveillance programs gather and aggregate metadata.
  - Clear metadata makes it easy to censor unwanted traffic.

Metadata is useful information

- Metadata is valuable information for censorship and surveillance regimes
  - Major surveillance programs gather and aggregate metadata.
  - Clear metadata makes it easy to censor unwanted traffic.
  - “We kill people based on metadata” – General Michael Hayden

Metadata is useful information

- Metadata is valuable information for censorship and surveillance regimes
  - Major surveillance programs gather and aggregate metadata.
  - Clear metadata makes it easy to censor unwanted traffic.
  - “We kill people based on metadata” – General Michael Hayden
- Everything in the TLS handshake is this sort of metadata.
- The TLS handshake is in the clear.

Metadata is useful information

- Metadata is valuable information for censorship and surveillance regimes
  - Major surveillance programs gather and aggregate metadata.
  - Clear metadata makes it easy to censor unwanted traffic.
  - “We kill people based on metadata” – General Michael Hayden
- Everything in the TLS handshake is this sort of metadata.
- The TLS handshake is in the clear.

Pervasive monitoring is an attack.

TLS should not facilitate censorship or surveillance.

How can the situation be improved?

- Hiding metadata information requires mixing into a larger anonymity set.
- Other layers may *also* leak metadata; this is their responsibility. TLS should not leak more.
- If TLS reduces metadata leakage, other protocols have incentive to improve.

Optional or not?

- The anonymity sets provided will be larger if this is the only 1.3 handshake.
- Making this optional increases implementation complexity.
- If it is optional, then clients may try cleartext handshakes anyway.
- If we must make it optional, we should encourage implementations to default to on.

False sense of security?

- Without DNSSEC, we can't defend the handshake against active attacks.
- Opportunistic Security – defend against passive attackers anyway.
- With DNSSEC, we have the ability to detect or prevent active attacks.

False sense of security?

- Without DNSSEC, we can't defend the handshake against active attacks.
- Opportunistic Security – defend against passive attackers anyway.
- With DNSSEC, we have the ability to detect or prevent active attacks.

We don't have to be as good as the record layer –
But we should do better than cleartext.

Goals

- Handshake, including SNI, Encrypted against passive
- 1-RTT from ignorance, 0-RTT w/ History
- Algorithm Flexibility
  - Support I require NIST and I require Not-NIST
- Secondary
  - Aim for Forward Secrecy
  - Aim for Resisting Active MITM or detecting

1 RTT From Full Ignorance

- Client → Server
- Server → Client
- Client[HTTP Data] → Server

1 RTT From Full Ignorance

- Client → Server
- Server[Signed Symmetric Key, Cert] → Client
- Client[HTTP Data] → Server

1 RTT From Full Ignorance

- Client[Cert-Selecting Info] → Server
- Server[Signed Symmetric Key, Cert] → Client
- Client[HTTP Data] → Server

“Cert Selecting Info”

- Can’t be a SNI replacement
- Therefore, SNI data is encrypted
  - But how?
  - Need to get a pre-handshake key to the client, prior to ClientHello

DANISH

- Key in DNS
- Like DANE, but DANE is for x509 cert
- Thus, DANISH

Currently: 3 common CDN mechanisms

- cdn.example.com
  - CNAME to cdn.com
- cdn.com
  - A to w.x.y.z

- cdn.example.com
  - A to w.x.y.z

- cdn.example.com
  - zone cut from example.com, run by CDN

w/DANISH

- cdn.example.com
    - CNAME to cdn.com
- cdn.com
    - A to w.x.y.z
    - DANISH to [keydata]

- cdn.example.com
    - A to w.x.y.z
    - DANISH to [keydata]

- cdn.example.com
    - zone cut from example.com, run by CDN

Algorithm Requirement

- cdn.example.com
  - CNAME to **nist.**cdn.com
- cdn.com
  - A to w.x.y.z
  - DANISH to [**nist**-keydata]

- cdn.example.com
  - A to w.x.y.z
  - DANISH to [**nist**-keydata]

- cdn.example.com
  - zone cut from example.com, run by CDN

Doesn't require DNSSEC

- Resists passive adversary without DNSSEC
- Resists active adversary w/ DNSSEC*

Algorithm Flexibility

- All CDN servers can have uniform configuration
  - Answer for all keys, if desired
- ClientHello has opaque uint32 key ID
  - Not an SNI replacement

Failure Modes

- Client sends unknown key identifier or undecryptable input
  - DNS data stale, misconfigured, or malicious client
- Server responds “Use this pre-handshake key”
- Client restarts w/ ClientHello (1-RTT → 2-RTT)

Failure and Algo Flexibility

- Server responds “Use this pre-handshake key”
  - What key?!?! NIST? DJB?
- Two solutions for CDNs, outside of spec
  1. Opaque KeyID is not, top n bits indicate Algo
  2. CDN Servers answer to any key, but subsets have different defaults. nist.cdn.org A RRs → [Nist subset]

Active Attack

Client[Encrypted cdn.example.com] → Server

Client ← Attacker "Unknown Key, use this one"

Client can:

- Continue to 2-RTT handshake, vulnerable to active attack, which we detect at handshake end
- Not trust that, refresh DNSSEC information
- Choose their own destiny in the name of speed or security

Indicating TLS 1.3

Presence of a DANISH record, can indicate TLS 1.3 capability

- Same as DANE for SMTP
- But we're handwaving here

Handwave

- 0-RTT with History
- Forward Secrecy
    - Key Rotation is good, example.com-specified DANISH records hurt
- Fallback
    - DANISH implies TLS 1.3. If server barfs, browsers downgrade to TLS1.2, re-handshaking
    - Browsers pin TLS1.3 support per name via another mechanism

Other Ideas

- DNSNAME DNS Type
  - Like CNAME, but validates on target
  - *.cdn.org is used by CDN for every customer
- Server sends key in SYN/ACK, Server speaks first
  - Similar to TCP Fast Open
  - Like idea, requires massive overhaul

Even Faster!

Currently:

- DNS example.com
- TCP handshake
- TLS handshake
- DNS cdn.example.com
- (CNAME: DNS cdn.org)
- TCP Handshake
- TLS Handshake

Faster:

- DNS example.com
- TCP handshake
- TLS handshake
  - HTTP Headers w/ DNSSEC-signed DNS responses for cdn.example.com and cdn.org
- TCP Handshake