redhat.

TLS-DNSSEC-CHAIN

IETF 102, TLS Working Group

Resolving the downgrade attack without narrowing the scope

Paul Wouters
Senior Software Engineer, RHEL Security

The DANE TLSA record

_443._tcp.www.nohats.ca IN TLSA 3 1 1 <pkey sha256>

- Match resource (hostname + port) to a public key
- Live DNS data ensures freshness. Clients do not need data pinning, as everything is DNSSEC based and is updated within DNS and its TTL / RRSIG lifetimes.
- Resists stripping, Denial of Existence (DoE) require signed NSEC/NSEC3 proofs.
- Overlapping TLSA records handle key rollover
- No dependency on WebPKI
- There is no “testing” mode. Data is either live in DNS or not. Publishing in DNS means committing to the data

Last-mile obstacles

- DNSSEC adds latency (serialized round trips)
- Network path not always clean and usable for DNSSEC (accidental record mangling/ stripping)
- DNS record manipulation or stripping can lead to “bogus” or “indeterminate” state of DNSSEC validation. (“glomar response”)

TLS-DNSSEC-CHAIN extension

- Staple DNSSEC chain from root to TLSA record in a TLS extension using RFC 7901 dnssec chains
- The TLS server updates its own stapled data from live DNS (e.g. every hour)
- If no TLSA DNS record present in live DNS data, staple proof of Denial of Existence
  - Greenfield applications can require the extension, and yet also work with domains not using DANE / DNSSEC / TLSA
  - Optional extension is pointless because it can be stripped
- Client processes data as if it received directly from DNS. It's not different from receiving DNS data from recursive DNS server's cache
- **No pinning of DNS data involved:**
  - stapled data is "live DNS data" snapshot. **This is NOT HPKP**.

Defective Scope

Without extension pinning, the extension is vulnerable to a full downgrade attack

- The present draft fails to support incremental adoption by existing applications
- This means no chance of adoption in HTTPS, IMAP, NNTP, XMPP, SUBMIT, …
- With the extension subject to stripping:
  - No added security, the whole extension is moot
  - DANE-only clients have initially (and so forever) no clients. This actively prevents deploying a DANE based PKI
  - Clients that support using DANE or WebPKI get the **security of the weakest**
  - Clients that support using DANE pinning on top of WebPKI gain nothing
- Only DNS-over-TLS clients apparently(?) suffer no consequences?
  - Wouldn’t these fallback to plain DNS?

Anti-Downgrade options

1. Do nothing
2. Fix everything in new TLS extension
3. Two zero bytes in this RFC, specify non-zero semantics in a separate update RFC
4. Two byte TLS extension pin TTL (in hours)
5. Variable-length (0..255) reserved field (default empty) in this RFC, syntax and semantics in separate update RFC
6. Nested extension block (just like new TLS extension, but even more complicated)

1. Do Nothing

- Fails to address security considerations
- Leaves TLS client behaviour under-specified
- Some TLS clients will do TOFU and break
  - No method for TLS server to test TLS extension without risking commitment
- Prevents major use case of incremental DANE adoption by existing TLS clients
  - that (initially) mostly use just WebPKI
- Prevents major use case of enhancing WebPKI security with additional DANE security
- Will only be used by DNS-over-TLS clients that refuse fallback to plaintext DNS
  - or “greenfield” implementations mandating this extension, no vendor stepped forward
  - or “additive use case” clients, but no deployment path for security gain.

2. Fix everything in new TLS extension

- Leaves this RFC unable to support its most common use cases, and promotes insecure deployments that leave in all downgrade attacks
- Would require a second TLS extension to pin itself AND the tls-dnssec-chain extension or needs to obsolete this TLS extension.
- Needlessly complex committee design.
- A 2nd TLS extension will end in an identical discussion, except a name change of tls-dnssec-chain-bis while deploying a flawed specification.
- Buys us nothing over doing it properly in 1 TLS extension (whether or not specified in 1 or 2 RFCs)
- This solution is awful as demonstrated with:
  - https://tools.ietf.org/html/draft-asmithee-tls-dnssec-downprot-00

3. Two zero bytes now, specify in followup RFC

- Add two zero bytes. Zero means "TLS client MUST NOT pin extension”
- Supports TLS server experimentation without commitment (prevents clients interpreting unspecified as TOFU)
- Followup RFC documents non-zero semantics for these two bytes
- Can be discussed in TLS WG
- Allows current document to proceed

- This is STS-lite, other STS features
  - Test-mode make no sense here (DNS data is already live)
  - Subdomain scope drags in the public suffix list, too complex for general use. And TLSA records are scoped to just a single port!
- Commits TLS WG to define the two bytes in new document. Prevents stalling DANE PKI deployments

4. Two byte TLS extension pin TTL (in hours)

Add two bytes

- Upper bound in hours on how long client can require server to continue to send the extension.
- Denial of existence makes it possible for server to discontinue DANE or even DNSSEC signing of its zone. The pin only requires continued extension support. TLSA use can be abandoned at any time and extension pin set to 00 00.

- Allows TLS server experimentation without commitment
- Can be discussed in TLS WG, but 1h–65535h PIN time would be enough
  - Pinning less then 1h makes no real sense due to DNS TTLs.
  - 65535h is ~7.5 years.
- Holds document to get this done
- Commits TLS WG to define the two bytes in updated document ASAP to avoid needless delay

5. Variable-length (0..255) Reserved Field

- Reserved now, defined in a separate update RFC
- Zero (length) means "TLS client MUST NOT pin extension”.
  - Clients that implement only this RFC free to treat all values as “do not pin”
- Supports TLS server experimentation without commitment
- More flexibility in format of extension pinning in Update RFC (but ultimately, probably 0 or 2 bytes)
- One byte less for this RFC when update RFC not used, and one more when it is.

6. Extension block

- Complicated nested meta extension field
- Empty means "TLS client MUST NOT pin extension”
- Otherwise under-specified, unless client signals support for each nested extension, but then why nest???
- So this is just additional extensions in disguise, no obvious benefit from nesting
- Supports TLS server experimentation without commitment
- Supports gradually introduced tweaks, but that can be done later also (if proves necessary) to complement the minimal reserved fields
- Fancy way of saying “do nothing” or “fix everything in new TLS extension”

Summary notes

- One or two byte cosmetics **vs** accept known downgrade attack
- One or two byte cosmetics **vs** reduce scope to only DNS-over-TLS
- Possible (unlikely) replacing draft **vs** guaranteed replacing of draft with -bis
- Preventing DANE deployment vs freedom to choose WebPKI and/or DANE

- TLS != HTTPS
- Solution requested for planned support in openssl and postfix

Discussion

1. Do nothing
2. Fix everything in new TLS extension
3. Two zero bytes in this RFC, specify non-zero semantics in a separate update RFC
4. Two byte TLS extension pin TTL (in hours)
5. Variable-length (0..255) reserved field (default empty) in this RFC, syntax and semantics in separate update RFC
6. Nested extension block (just like new TLS extension, but even more complicated)