Using Secure DNS to Associate Certificates with Domain Names For TLS
draft-ietf-dane-protocol-12

Abstract

TLS and DTLS use PKIX certificates for authenticating the server. Users want their applications to verify that the certificate provided by the TLS server is in fact associated with the domain name they expect. TLSA provides bindings of keys to domains that are asserted not by external entities, but by the entities that operate the DNS. This document describes how to use secure DNS to associate the TLS server’s certificate with the intended domain name.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 30, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must...
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
  1.1. Certificate Associations ............................... 3
  1.2. Securing Certificate Associations ................... 4
  1.3. Terminology .............................................. 4
2. The TLSA Resource Record ..................................... 4
  2.1. TLSA RDATA Wire Format ................................. 5
      2.1.1. The Certificate Usage Field ...................... 5
      2.1.2. The Selector Field ............................... 6
      2.1.3. The Matching Type Field .......................... 6
      2.1.4. The Certificate for Association Field .......... 6
  2.2. TLSA RR Presentation Format ............................ 6
  2.3. TLSA RR Examples ....................................... 7
3. Domain Names for TLS Certificate Associations ................ 7
4. Semantics and Features of TLSA Certificate Usages .............. 8
  4.1. Pass PKIX Validation and Chain Through CA ............. 8
  4.2. Pass PKIX Validation and Match End Entity Certificate .. 8
  4.3. Pass PKIX Validation and Use Trust Anchor ............ 8
  4.4. Use of TLS Certificate Associations in TLS ........... 8
5. TLSA and DANE Use Cases and Requirements .................... 9
6. Mandatory-to-Implement Algorithms ........................... 10
7. IANA Considerations .......................................... 11
  7.1. TLSA RRtype ............................................. 11
  7.2. TLSA Usages ........................................... 11
  7.3. TLSA Selectors ......................................... 11
  7.4. TLSA Matching Types ................................... 12
8. Security Considerations ...................................... 12
9. Acknowledgements ........................................... 13
10. References .................................................. 13
  10.1. Normative References .................................. 13
  10.2. Informative References ................................. 14
Appendix A. Operational Considerations for Deploying TLSA
  Records ..................................................... 14
A.1. Provisioning TLSA Records with Aliases .................. 14
      A.1.1. Provisioning TLSA Records with CNAME Records .... 15
      A.1.2. Provisioning TLSA Records with DNAME Records ... 16
      A.1.3. Provisioning TLSA Records with Wildcards .......... 16
A.2. Provisioning Using NS Records ............................ 17
A.3. Securing the Last Hop ................................... 17
A.4. Handling Certificate Rollover ............................ 17
Appendix B. Pseudocode for Using TLSA .......................... 17
Authors’ Addresses .............................................. 19
1. Introduction

The first response from the server in TLS may contain a certificate. In order for the TLS client to authenticate that it is talking to the expected TLS server, the client must validate that this certificate is associated with the domain name used by the client to get to the server. Currently, the client must extract the domain name from the certificate, must trust a trust anchor upon which the server’s certificate is rooted, and must successfully validate the certificate.

Some people want a different way to authenticate the association of the server’s certificate with the intended domain name without trusting an external certificate authority (CA). Given that the DNS administrator for a domain name is authorized to give identifying information about the zone, it makes sense to allow that administrator to also make an authoritative binding between the domain name and a certificate that might be used by a host at that domain name. The easiest way to do this is to use the DNS.

There are many use cases for such functionality. [DANEUSECASES] lists the ones that the protocol in this document is meant to apply to. [DANEUSECASES] also lists many requirements, most of which the protocol in this document is believed to meet. Section 5 covers the applicability of this document to the use cases in detail.

This document applies to both TLS [RFC5246] and DTLS [RFC4347bis]. In order to make the document more readable, it mostly only talks about "TLS", but in all cases, it means "TLS or DTLS". This document only relates to securely associating certificates for TLS and DTLS with host names; other security protocols and other forms of identification of TLS servers (such as IP addresses) are handled in other documents. For example, keys for IPsec are covered in [RFC4025] and keys for SSH are covered in [RFC4255].

1.1. Certificate Associations

In this document, a certificate association is based on a cryptographic hash of a certificate (sometimes called a "fingerprint"), a public key, or on the certificate itself. For a fingerprint, a hash is taken of the binary, DER-encoded certificate or public key, and that hash is the certificate association; the type of hash function used can be chosen by the DNS administrator. When using the certificate itself in the certificate association, the entire certificate in the normal format is used. This document only applies to PKIX [RFC5280] certificates, not certificates of other formats. It also applies to public keys that are extracted from PKIX certificates, not just full certificates.
Certificate associations are made between a certificate or public key and a domain name. Server software that is running TLS that is found at that domain name would use a certificate that has a certificate association given in the DNS, as described in this document. A DNS query can return multiple certificate associations, such as in the case of different server software on a single host using different certificates, or in the case that a server is changing from one certificate to another.

1.2. Securing Certificate Associations

This document defines a secure method to associate the certificate that is obtained from the TLS server with a domain name using DNS; the DNS information may need to be protected by DNSSEC. Because the certificate association was retrieved based on a DNS query, the domain name in the query is by definition associated with the certificate.

DNSSEC, which is defined in RFCs 4033, 4034, and 4035 ([RFC4033], [RFC4034], and [RFC4035]), uses cryptographic keys and digital signatures to provide authentication of DNS data. Information retrieved from the DNS and that is validated using DNSSEC is thereby proved to be the authoritative data. The DNSSEC signature MUST be validated on all responses that use DNSSEC in order to assure the proof of origin of the data. This document does not specify how DNSSEC validation occurs because there are many different proposals for how a client might get validated DNSSEC results.

This document only relates to securely getting the DNS information for the certificate association using DNSSEC; other secure DNS mechanisms are out of scope.

1.3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document also makes use of standard PKIX, DNSSEC, and TLS terminology. See [RFC5280], [RFC4033], and [RFC5246] respectively, for these terms. In addition, terms related to TLS-protected application services and DNS names are taken from [RFC6125].

2. The TLSA Resource Record

The TLSA DNS resource record (RR) is used to associate a certificate with the domain name where the record is found. The semantics of how
The TLSA RR is interpreted are given later in this document.

The type value for the TLSA RR type is TBD.

The TLSA RR is class independent.

The TLSA RR has no special TTL requirements.

2.1. TLSA RDATA Wire Format

The RDATA for a TLSA RR consists of a one octet usage type field, a one octet selector field, a one octet matching type field and the certificate for association field.

```
+--------+---------+-----------+------------------+
| Usage  | Selector| Matching Type | Certificate for Association |
+--------+---------+-----------+------------------+
```

2.1.1. The Certificate Usage Field

A one-octet value, called "certificate usage" or just "usage", specifying the provided association that will be used to match the target certificate. This will be an IANA registry in order to make it easier to add additional certificate usages in the future. The usages defined in this document are:

- **0** -- Certificate MUST pass PKIX validation and MUST chain through a CA certificate matching the TLSA record
- **1** -- Certificate MUST pass PKIX validation and MUST match the TLSA record
- **2** -- Certificate MUST pass PKIX validation, with any certificate matching the TLSA record considered to be a trust anchor for this validation

The three certificate usages defined in this document explicitly only apply to PKIX-formatted certificates. If TLS allows other formats later, or if extensions to this protocol are made that accept other formats for certificates, those certificates will need their own certificate types.
2.1.2. The Selector Field

A one-octet value, called "selector", specifying what will be matched. This value is defined in a new IANA registry. The selectors defined in this document are:

0 -- Full certificate
1 -- SubjectPublicKeyInfo

2.1.3. The Matching Type Field

A one-octet value, called "matching type", specifying how the certificate association is presented. This value is defined in a new IANA registry. The types defined in this document are:

0 -- Exact match on selected content
1 -- SHA-256 hash of selected content
2 -- SHA-512 hash of selected content

Using the same hash algorithm as is used in the signature in the certificate will make it more likely that the TLS client will understand this TLSA data.

2.1.4. The Certificate for Association Field

The "certificate for association". This is the bytes containing the full certificate, SubjectPublicKeyInfo or the hash of the associated certificate or SubjectPublicKeyInfo. This is the certificate or the hash of the certificate itself, not of the TLS ASN.1 Certificate object.

2.2. TLSA RR Presentation Format

The presentation format of the RDATA portion is as follows:

- The certificate usage field MUST be represented as an unsigned decimal integer.
- The selector field MUST be represented as an unsigned decimal integer.
- The matching type field MUST be represented as an unsigned decimal integer.
The certificate for association field MUST be represented as a string of hexadecimal characters. Whitespace is allowed within the string of hexadecimal characters.

2.3. TLSA RR Examples

An example of a hashed (SHA-256) association of a PKIX CA certificate:

_443._tcp.www.example.com. IN TLSA ( 0 0 1 d2abde240d7cd3ee6b4b28c54df034b9 7983a1d16e8a410e4561cb106618e971 )

An example of a hashed (SHA-512) subject public key association of a PKIX end entity certificate:

_443._tcp.www.example.com. IN TLSA 1 1 2 92003ba34942dc74152e2f2c408d29ec a5a520e7f2e06bb944f4dca346baf63c 1b177615d466f6c4b71c216a50292bd5 8c9ebdd2f74e38fe51fffd48c43326cbc )

An example of a full certificate association of a PKIX trust anchor:

_443._tcp.www.example.com. IN TLSA 2 0 0 30820307308201efa003020102020... )

3. Domain Names for TLS Certificate Associations

TLSA resource records are stored at a prefixed DNS domain name. The prefix is prepared in the following manner:

1. The decimal representation of the port number on which a TLS-based service is assumed to exist is prepended with an underscore character ("_") to become the left-most label in the prepared domain name. This number has no leading zeros.

2. The protocol name of the transport on which a TLS-based service is assumed to exist is prepended with an underscore character ("_") to become the second left-most label in the prepared domain name. The transport names defined for this protocol are "tcp", "udp" and "sctp".

3. The domain name is appended to the result of step 2 to complete the prepared domain name.

For example, to request a TLSA resource record for an HTTP server
running TLS on port 443 at "www.example.com", you would use
"_443._tcp.www.example.com" in the request. To request a TLSA
resource record for an SMTP server running the STARTTLS protocol on
port 25 at "mail.example.com", you would use
"_25._tcp.mail.example.com".

4. Semantics and Features of TLSA Certificate Usages

The three certificate usages have very different semantics, but also
have features common to all three types.

4.1. Pass PKIX Validation and Chain Through CA

Certificate usage 0 is used to specify a CA certificate, or the
public key of such a certificate, that must be found in any of
the PKIX validation chains for the end entity certificate given by
the server in TLS. This usage is sometimes referred to as "CA
constraint" because it limits which CA can be used to issue
certificates for a given host name.

4.2. Pass PKIX Validation and Match End Entity Certificate

Certificate usage 1 is used to specify an end entity certificate, or
the public key of such a certificate, that must be matched with the
end entity certificate given by the server in TLS. This usage is
sometimes referred to as "service certificate constraints" because it
limits which end entity certificate may be used by a given host name.

4.3. Pass PKIX Validation and Use Trust Anchor

Certificate usage 2 is used to specify a certificate, or the public
key of such a certificate, that must be used as a trust anchor when
validating the end entity certificate given by the server in TLS.
This usage is sometimes referred to as "domain-issued certificate"
because it allows for a domain name administrator to issue
certificates for a domain without involving a third-party CA.

4.4. Use of TLS Certificate Associations in TLS

Section 2.1 of this document defines the mandatory matching rules for
the data from the TLS certificate associations and the certificates
received from the TLS server.

The TLS session that is to be set up MUST be for the specific port
number and transport name that was given in the TLSA query. The
matching or chaining MUST be done within the life of the TTL on the
TLSA record.
Some specifications for applications that run under TLS, such as [RFC2818] for HTTP, require the server’s certificate to have a domain name that matches the host name expected by the client. Some specifications such as at [RFC6125] detail how to match the identify given in a PKIX certificate with those expected by the user.

In order to use one or more TLS certificate associations described in this document obtained from the DNS, an application MUST assure that the certificates were obtained using DNS protected by DNSSEC. TLSA records must only be trusted if they were obtained from a trusted source. This could be a localhost DNS resolver answer with the AD bit set, an inline validating resolver library primed with the proper trust anchors, or obtained from a remote name server to which one has a secured channel of communication.

If a certificate association contains a certificate usage, selector, or matching type that is not understood by the TLS client, that certificate association MUST be marked as unusable. If the comparison data for a certificate is malformed, the certificate association MUST be marked as unusable.

An application that complies with this document first requests TLSA records in order to make certificate associations. If that application receives zero usable certificate associations, it processes TLS in the normal fashion; otherwise, that application attempts to match each certificate association with the TLS server’s end entity certificate. If such a match is found, that application continues the TLS handshake and ignores any remaining certificate associations; otherwise, that application MUST abort the TLS handshake with an "access_denied" error.

5. TLSA and DANE Use Cases and Requirements

The different types of certificates for association defined in TLSA are matched with various sections of [DANEUSECASES]. The three use cases from section 3 of [DANEUSECASES] are covered in this protocol as follows:

3.1 CA Constraints -- Implemented using certificate usage 0.

3.2 Certificate Constraints -- Implemented using certificate usage 1.

3.3 Domain-Issued Certificates -- Implemented using certificate usage 2.

The requirements from section 4 of [DANEUSECASES] are covered in this
protocol as follows (note that some of these might be excessively
glib):

Multiple Ports -- Covered in the TLSA request syntax.

No Downgrade -- Covered by DNSSEC itself.

Encapsulation -- Covered in the TLSA response semantics.

Predictability -- Covered by this spec.

Opportunistic Security -- Covered in the TLSA request syntax.

Combination -- Covered in the TLSA response semantics.

Roll-over -- Covered by the TTLs on the TLSA records.

Simple Key Management -- Implemented using the SubjectPublicKeyInfo
selector.

Minimal Dependencies -- Covered in the TLSA response semantics.

Minimal Options -- Covered in the TLSA response semantics.

Wild Cards -- Covered in a limited manner in the TLSA request
syntax; see Appendix A.

Redirection -- Covered in the TLSA request syntax; see Appendix A.

6. Mandatory-to-Implement Algorithms

DNS systems conforming to this specification MUST be able to create
TLSA records containing certificate usages 0, 1 and 2. DNS systems
conforming to this specification MUST be able to create TLSA records
with selectors 0 (full certificate) and 1 (SubjectPublicKeyInfo).
DNS systems conforming to this specification MUST be able to create
TLSA records using matching type 0 (no hash used) and matching type 1
(SHA-256), and SHOULD be able to create TLSA records using matching
type 2 (SHA-512).

TLS clients conforming to this specification MUST be able to
correctly interpret TLSA records with certificate usages 0, 1 and 2.
TLS clients conforming to this specification MUST be able to compare
a certificate for association with a certificate from TLS using
selectors type 0 and 1, and matching type 0 (no hash used) and
matching type 1 (SHA-256), and SHOULD be able to make such
comparisons with matching type 2 (SHA-512).
At the time this is written, it is expected that there will be a new family of hash algorithms called SHA-3 within the next few years. It is expected that some of the SHA-3 algorithms will be mandatory and/or recommended for TLSA records after the algorithms are fully defined. At that time, this specification will be updated.

7. IANA Considerations

7.1. TLSA RRtype

This document uses a new DNS RR type, TLSA, whose value is TBD. A separate request for the RR type will be submitted to the expert reviewer, and future versions of this document will have that value instead of TBD.

7.2. TLSA Usages

This document creates a new registry, "Certificate Usages for TLSA Resource Records". The registry policy is "RFC Required". The initial entries in the registry are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Short description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pass PKIX and chain through CA</td>
<td>[This]</td>
</tr>
<tr>
<td>1</td>
<td>Pass PKIX and match EE</td>
<td>[This]</td>
</tr>
<tr>
<td>2</td>
<td>Pass PKIX and trusted via certificate</td>
<td>[This]</td>
</tr>
<tr>
<td>4-254</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>Private use</td>
<td></td>
</tr>
</tbody>
</table>

Applications to the registry can request specific values that have yet to be assigned.

7.3. TLSA Selectors

This document creates a new registry, "Selectors for TLSA Resource Records". The registry policy is "Specification Required". The initial entries in the registry are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Short description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Full Certificate</td>
<td>[This]</td>
</tr>
<tr>
<td>1</td>
<td>SubjectPublickeyInfo</td>
<td>[This]</td>
</tr>
<tr>
<td>2-254</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>Private use</td>
<td></td>
</tr>
</tbody>
</table>
7.4. TLSA Matching Types

This document creates a new registry, "Matching Types for TLSA Resource Records". The registry policy is "Specification Required". The initial entries in the registry are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Short description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No hash used</td>
<td>[This]</td>
</tr>
<tr>
<td>1</td>
<td>SHA-256</td>
<td>NIST FIPS 180-3</td>
</tr>
<tr>
<td>2</td>
<td>SHA-512</td>
<td>NIST FIPS 180-3</td>
</tr>
<tr>
<td>3-254</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>Private use</td>
<td></td>
</tr>
</tbody>
</table>

Applications to the registry can request specific values that have yet to be assigned.

8. Security Considerations

The security of the protocols described in this document relies on the security of DNSSEC as used by the client requesting A/AAAA and TLSA records.

A DNS administrator who goes rogue and changes both the A/AAAA and TLSA records for a domain name can cause the user to go to an unauthorized server that will appear authorized, unless the client performs certificate validation and rejects the certificate. That administrator could probably get a certificate issued anyway, so this is not an additional threat.

If the authentication mechanism for adding or changing TLSA data in a zone is weaker than the authentication mechanism for changing the A/AAAA records, a man-in-the-middle who can redirect traffic to their site may be able to impersonate the attacked host in TLS if they can use the weaker authentication mechanism. A better design for authenticating DNS would be to have the same level of authentication used for all DNS additions and changes for a particular host.

SSL proxies can sometimes act as a man-in-the-middle for TLS clients. In these scenarios, the clients add a new trust anchor whose private key is kept on the SSL proxy; the proxy intercepts TLS requests, creates a new TLS session with the intended host, and sets up a TLS session with the client using a certificate that chains to the trust anchor installed in the client by the proxy. In such environments, the TLSA protocol will prevent the SSL proxy from functioning as expected because the TLS client will get a certificate association from the DNS that will not match the certificate that the SSL proxy
uses with the client. The client, seeing the proxy’s new certificate for the supposed destination will not set up a TLS session. Thus, such proxies might choose to aggressively block TLSA requests and/or responses.

Client treatment of any information included in the trust anchor is a matter of local policy. This specification does not mandate that such information be inspected or validated by the domain name administrator.

9. Acknowledgements

Many of the ideas in this document have been discussed over many years. More recently, the ideas have been discussed by the authors and others in a more focused fashion. In particular, some of the ideas here originated with Paul Vixie, Dan Kaminsky, Jeff Hodges, Phill Hallam-Baker, Simon Josefsson, Warren Kumari, Adam Langley, Ben Laurie, Ilari Liusvaara, Scott Schmit, Ondrej Sury, Richard Barnes, and Jim Schaad.

This document has also been greatly helped by many active participants of the DANE Working Group.

10. References

10.1. Normative References

[DANEUSECASES]
Barnes, R., "Use Cases and Requirements for DNS-based Authentication of Named Entities (DANE)", draft-ietf-dane-use-cases (work in progress), 2011.


10.2. Informative References


Appendix A. Operational Considerations for Deploying TLSA Records

A.1. Provisioning TLSA Records with Aliases

The TLSA resource record is not special in the DNS; it acts exactly like any other RRtype where the queried name has one or more labels prefixed to the base name, such as the SRV RRtype [RFC2782]. This affects the way that the TLSA resource record is used when aliasing in the DNS.

Note that the IETF sometimes adds new types of aliasing in the DNS. If that happens in the future, those aliases might affect TLSA records, hopefully in a good way.
A.1.1. Provisioning TLSA Records with CNAME Records

Using CNAME to alias in DNS only aliases from the exact name given, not any zones below the given name. For example, assume that a zone file has only the following:

```
sub1.example.com.       IN CNAME sub2.example.com.
```

In this case, a request for the A record at "bottom.sub1.example.com" would not return any records because the CNAME given only aliases the name given. Assume, instead, the zone file has the following:

```
sub3.example.com.       IN CNAME sub4.example.com.
```

In this case, a request for the A record at bombot.sub3.example.com would in fact return whatever value for the A record exists at bombot.sub4.example.com.

Application implementations and full-service resolvers request DNS records using libraries that automatically follow CNAME (and DNAME) aliasing. This allows hosts to put TLSA records in their own zones or to use CNAME to do redirection.

If the owner of the original domain wants a TLSA record for the same, they simply enter it under the defined prefix:

```
; No TLSA record in target domain

sub5.example.com.       IN CNAME sub6.example.com.
_443._tcp.sub5.example.com. IN TLSA 1 1 1 308202c5308201ab...
sub6.example.com.       IN A 10.0.0.0
```

If the owner of the orginal domain wants to have the target domain host the TLSA record, the original domain uses a CNAME record:

```
; TLSA record for original domain has CNAME to target domain

sub5.example.com.       IN CNAME sub6.example.com.
_443._tcp.sub5.example.com. IN CNAME _443._tcp.sub6.example.com.
sub6.example.com.       IN A 10.0.0.0
_443._tcp.sub6.example.com. IN TLSA 1 1 1 536a570ac49d9ba4...
```

Note that it is acceptable for both the original domain and the target domain to have TLSA records, but the two records are unrelated. Consider the following:
In this example, someone looking for the TLSA record for sub5.example.com would always get the record whose value starts "308202c5308201ab"; the TLSA record whose value starts "ac49d9ba4570ac49" would only be sought by someone who is looking for the TLSA record for sub6.example.com, and never for sub5.example.com.

Note that these methods use the normal method for DNS aliasing using CNAME: the DNS client requests the record type that they actually want.

A.1.2. Provisioning TLSA Records with DNAME Records

Using DNAME records allows a zone owner to alias an entire subtree of names below the name that has the DNAME. This allows the wholesale aliasing of prefixed records such as those used by TLSA, SRV, and so on without aliasing the name itself. However, because DNAME can only be used for subtrees of a base name, it is rarely used to alias individual hosts that might also be running TLS.

A.1.3. Provisioning TLSA Records with Wildcards

Wildcards are generally not terribly useful for RRtypes that require prefixing because you can only wildcard at a layer below the host name. For example, if you want to have the same TLSA record for every TCP port for www.example.com, you might have

\[ *._tcp.www.example.com. IN TLSA 1 1 1 5c1502a6549c423b... \]

This is possibly useful in some scenarios where the same service is offered on many ports.
A.2. Provisioning Using NS Records

[[ This was proposed, and questioned, and not yet followed through on. ]]

A.3. Securing the Last Hop

[[ Need to add text here about the various ways that a client who is pulling in TLSA records can be sure that they are protected by DNSSEC. ]]

A.4. Handling Certificate Rollover

[[ Need to add text here about how to handle a change in certificate. It would cover using two TLSA records at the same time, the TTL on the RRset, and coordinating that with the use of the certificates in the TLS server. ]]

Appendix B. Pseudocode for Using TLSA

[[ IMPORTANT NOTE FOR THE DANE WG: Please review this new appendix carefully. If you find differences between what is here and what is in the rest of the draft, by all means please send it to the WG mailing list. The ensuing discussion will hopefully help everyone. ]]

This appendix describes the interactions given earlier in this specification in pseudocode format. This appendix is non-normative. If the steps below disagree with the text earlier in the document, the steps earlier in the document should be considered correct and this text incorrect.

TLS connect using [transport] to [hostname] on [port] and receiving end entity cert C for the TLS server:

look up TLSA for _[port]._[transport].[hostname]

if (no secure TLSA record(s) received) {
    fall back to non-TLSA cert validation
}

// unusable records include unknown certUsage, unknown selectorType, unknown matchingType, and erroneous RDATA
strip unusable records
if (no usable TLSA record(s) received) {
    fall back to non-TLSA cert validation
}
// implement the selector function
function Select(S, X) = {
  if (S == 0) {
    return X
  }
  if (S == 1) {
    return X.SubjectPublicKey
  }
  return undef
}

// implement the matching function
function Match(M, X, Y) {
  if (M == 0) {
    return (X == Y)
  }
  if (M == 1) {
    return (SHA-256(X) == Y)
  }
  if (M == 2) {
    return (SHA-512(X) == Y)
  }
  return undef
}

// A TLS client might have multiple trust anchors that it might use
// when validating the TLS server’s end entity certificate. Also,
// there can be multiple PKIX validation chains for the
// certificates given by the server in TLS. Thus, there are
// possibly many chains that might need to be tested during
// PKIX validation.

for each TLSA record R {
  // pass PKIX validation and chain through CA cert from TLSA
  if (R.certUsage == 0) {
    for each PKIX validation chain H {
      if (C passes PKIX validation in H) {
        for each D in H {
          if (D is a CA certificate) and
            Match(matchingType, Select(selectorType, D), R) {
            accept the TLS connection
          }
        }
      }
    }
  }
  // pass PKIX validation and match EE cert from TLSA
if (R.certUsage == 1) {
    for each PKIX validation chain H {
        if (C passes PKIX validation in H) {
            if (Match(matchingType, Select(selectorType, C), R)) {
                accept the TLS connection
            }
        }
    }
}

// pass PKIX validation using TLSA record as trust anchor
if (R.certUsage == 2) {
    for each PKIX validation chain H that has R as the trust anchor {
        if (C passes PKIX validation in H) {
            accept the TLS connection
        }
    }
}

abort TLS handshake with "access_denied" error.

Authors' Addresses

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org

Jakob Schlyter
Kirei AB

Email: jakob@kirei.se