Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 13, 2011

P. Hoffman
VPN Consortium
J. Schlyter
Kirei AB

March 12, 2011

Using Secure DNS to Associate Certificates with Domain Names For TLS
draft-ietf-dane-protocol-06

Abstract

TLS and DTLS use 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. DNSSEC provides a mechanism for a zone operator to sign DNS information directly. This way, bindings of keys to domains 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 September 13, 2011.

Copyright Notice

Table of Contents

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 a 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.

This document applies to both TLS [RFC5246] and DTLS [4347bis]. 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 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") or on the certificate itself. For a fingerprint, a hash is taken of the binary, DER-encoded certificate, 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.

Certificate associations are made between a certificate or the hash of a certificate 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 (even if they are normally accessed with different host names), 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 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 in order to assure the proof of origin of the data.

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].

A note on terminology: Some people have said that this protocol is a form of "certificate exclusion". This is true, but only in the sense that a DNS reply that contains two of the certificate types defined here inherently excludes every other possible certificate in the universe (other than those found with a pre-image attack against on of those two). The certificate type defined here is better thought of as "enumeration" of a small number of certificate associations, not "exclusion" of a near-infinite number of other certificates.

Some of the terminology in this draft may not match with the terminology used in RFC 5280. This will be fixed in future versions of this draft, with help from the PKIX community. In specific, we need to say (in a PKIX-appropriate way) that when we say "valid up to" and "chains to", full RFC 5280 path processing including revocation status checking is intended.

2. Getting TLS Certificate Associations from the DNS

This document defines a new DNS resource record type, "TLSA". A

query on a prepared domain name for the TLSA RR can return one or more records of the type TLSA. The TLSA RRType is TBD.

2.1. Requested Domain Name

Domain names are prepared for requests 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.

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".

2.2. Format of the Resource Record

The format of the data in the resource record is a binary record with three values, which MUST be in the order defined here:

- A one-octet value, called "certificate type", 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 types in the future. The types defined in this document are:

  1 -- A certificate that identifies an end entity

  2 -- A certification authority's certificate

  Both types are structured using the RFC 5280 formatting rules and use the DER encoding. As described later in this document, type 1 certificates do not need to correctly use all PKIX semantics.

- A one-octet value, called "reference 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 -- Full certificate

  1 -- SHA-256 hash of the certificate

  2 -- SHA-512 hash of the certificate

  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.

- The "certificate for association". This is the bytes containing the full certificate or the hash of the associated certificate (that is, the certificate or the hash of the certificate itself, not of the TLS ASN.1Cert object).

Certificate types 1 and 2 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 certificate types.

2.3. Making Certificate Associations

The two certificate types for TLS have very different semantics. A TLS client conforming to this protocol receiving a certificate for association of type 1 MUST compare it, using the specified hash type, with the end entity certificate received in TLS. A TLS client conforming to this protocol receiving a certificate for association of type 2 MUST treat it as a trust anchor for that domain name.

Certificate type 1 (a certificate that identifies an end entity) is matched against the first certificate offered by the TLS server. The certificate for association is used only for exact matching, not for chained validation. With reference type 0, the certificate association is valid if the certificate in the TLSA data matches to the first certificate offered by TLS. With reference types other than 0, the certificate association is valid if the hash of the first certificate offered by the TLS server matches the value from the TLSA data.

Certificate type 2 (certification authority's certificate) can be used in one of two ways. With reference type 0, the certificate in the TLSA resource record is used in chaining from the end entity given in TLS. The certificate association is valid if the first certificate in the certificate bundle can be validly chained to the

trust anchor from the TLSA data. With reference types other than 0, if the hash of any certificate past the first in the certificate bundle from TLS matches the trust anchor from the TLSA data, and the chain in the certificate bundle is valid up to that TLSA trust anchor, then the certificate association is valid. Alternately, if the first certificate offered chains to an existing trust anchor in the TLS client's trust anchor repository, and the hash of that trust anchor matches the value from the TLSA data, then the certificate association is valid.

The end entity certificate from TLS, regardless of whether it was matched with a TLSA type 1 certificate or chained to a TLSA type 2 CA certificate, must have at least one identifier in the subject or subjectAltName field of the matched certificates matches the expected identifier for the TLS server. Further, 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 TSLA record.

2.3.1. Format of Certificates Used to Identify End Entities

When presented with a type 1 certificate, the TLS client MUST NOT verify the correct PKIX semantics for the keyCertSign bit of the keyUsage extension, nor of the the basicConstraints extension. This is because PKIX (RFC 5280) makes it clear that all self-signed certificates are CA certificates and cannot be end entity certificates. The last paragraph of section 3.2 of RFC 5280 says:

"This specification covers two classes of certificates: CA certificates and end entity certificates. CA certificates may be further divided into three classes: cross-certificates, self-issued certificates, and self-signed certificates. ... Self-issued certificates are CA certificates in which the issuer and subject are the same entity. ... Self-signed certificates are self-issued certificates where the digital signature may be verified by the public key bound into the certificate. Self-signed certificates are used to convey a public key for use to begin certification paths. End entity certificates are issued to subjects that are not authorized to issue certificates."

This means that a self-signed certificate (one where the subject and issuer are the same, and the public key in the certificate can be used to directly evaluate the signature on the certificate) must follow all the PKIX semantics rules for CAs, and probably need to follow all the policy rules as well. This is clearly not what people who want a simple way to associate their public signing key with their domain name in an end entity certificate that can be used in TLS.

Because of these PKIX requirements on end entity certificates, the processing rules for TLSA are very different for certificates that identify end entities directly and CA certificates that can be used to validate PKIX end entity certificates. The rules here allow self-signed certificates offered as type 1 certificates to not follow all the PKIX semantics rules.

2.4. Presentation Format

The RDATA of the presentation format of the TLSA resource record consists of two numbers (certificate and hash type) followed by the bytes containing the certificate or the hash of the associated certificate itself, presented in hex. An example of a SHA-256 hash (type 1) of an end entity certificate (type 1) would be:

```
_443._tcp.www.example.com. IN TLSA (
   1 1 5c1502a6549c423be0a0aa9d9a16904de5ef0f5c98
       c735fcca79f09230aa7141 )
```

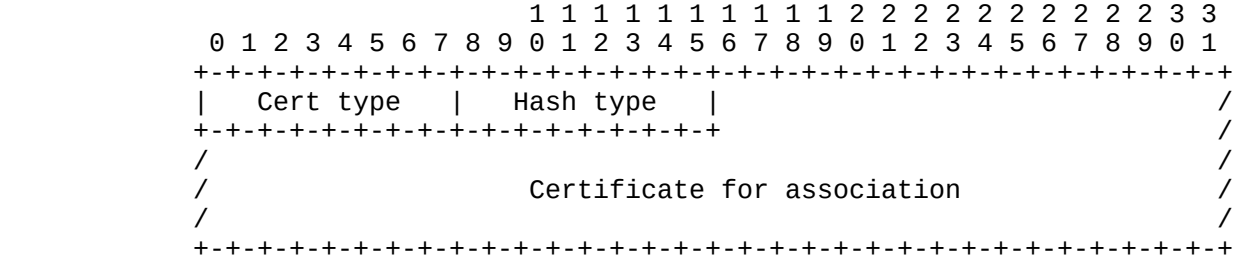
An example of an unhashed CA certificate (type 2) would be:

```
_443._tcp.www.example.com. IN TLSA (
   2 0 308202c5308201ada00302010202090... )
```

Because the length of hashes and certificates can be quite long, presentation format explicitly allows line breaks and white space in the hex values; those characters are removed when converting to the wire format.

2.5. Wire Format

The wire format is:



The wire format for the RDATA in the first example given above would be:

```
_443._tcp.www.example.com. IN TYPE65534 \# 34 ( 01015c1502a6549c42
              3be0a0aa9d9a16904de5ef0f5c98c735fcca79f09230aa7141 )
```

The wire format for the RDATA in the second example given above would be:

_443._tcp.www.example.com. IN TYPE65534 \# 715 0200308202c5308201a...

Note that in the preceding examples, "TYPE65534" is given as an example. That RR Type is in the IANA "private use" range; the real RR Type for TLSA will be issued by IANA, as described in the IANA Considerations section below.

3. Use of TLS Certificate Associations in TLS

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 nameserver to which one has a secured channel of communication.

If a certificate association contains a hash type that is not understood by the TLS client, that certificate association MUST be marked as unusable.

An application that requests TLS certificate associations using the method described in this document obtains zero or more usable certificate associations. If the application receives zero usable certificate associations, it processes TLS in the normal fashion.

If a match between one of the certificate association(s) and the server's end entity certificate in TLS is found, the TLS client continues the TLS handshake. If no match between the usable certificate association(s) and the server's end entity certificate in TLS is found, the TLS client MUST abort the handshake with an "access_denied" error.

4. Mandatory-to-Implement Algorithms

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

TLS clients conforming to this specification MUST be able to

correctly interpret TLSA records containing certificate types 1 and 2. TLS clients conforming to this specification MUST be able to compare a certificate for association with a certificate from TLS using hash type 0 (no hash used) and hash type 1 (SHA-256), and SHOULD be able to make such comparisons with hash 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.

5. IANA Considerations

5.1. TLSA RRtype

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

5.2. TLSA Certificate Types

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

Value	Short description	Reference
0	Reserved	[This]
1	Certificate to identify an end entity	[This]
2	CA's certificate	[This]
3-254	Unassigned	
255	Private use	

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

5.3. TLSA Hash Types

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

Value	Short description	Reference
0	No hash used	[This]
1	SHA-256	NIST FIPS 180-3
2	SHA-512	NIST FIPS 180-3
3-254	Unassigned	
255	Private use	

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

6. 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.

The values in the TLSA data will be normally entered in the DNS through the same system used to enter A/AAAA records, and other DNS information for the host name. If the authentication for changes to the host information is weak, an attacker can easily change any of this information. Given that the TLSA data is not easily human-readable, an attacker might change those records and A/AAAA records and not have the change be noticed if changes to a zone are only monitored visually.

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.

7. 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, and Ondrej Sury.

8. References

8.1. Normative References

[4347bis] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security version 1.2", draft-ietf-tls-rfc4347-bis (work in progress), July 2010.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

8.2. Informative References

[RFC4025] Richardson, M., "A Method for Storing IPsec Keying Material in DNS", RFC 4025, March 2005.

[RFC4255] Schlyter, J. and W. Griffin, "Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints", RFC 4255, January 2006.

Authors' Addresses

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org

Jakob Schlyter
Kirei AB

Email: jakob@kirei.se