Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 29, 2011
January 25, 2011

P. Hoffman
VPN Consortium
J. Schlyter
Kirei AB

Using Secure DNS to Associate Certificates with Domain Names For TLS
draft-ietf-dane-protocol-03

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. Instead of trusting a certification authority to have made this association correctly, the user might instead trust the authoritative DNS server for the domain name to make that association. This document describes how to use secure DNS to associate the TLS server's certificate with the the intended domain name.

Status of this Memo

Copyright Notice

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 the 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 also 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 in a very unusual sense. That is, 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 one 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 domain name for the TLSA RR can return one or more records of the type TLSA. The TLSA RRType is TBD.

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 -- Hash of an end-entity certificate

  2 -- Full end-entity certificate in DER encoding

  3 -- Hash of an certification authority's certificate

  4 -- Full certification authority's certificate in DER encoding

- A one-octet value, called "hash type", specifying the type of hash algorithm used for the certificate association. This value is defined in a new IANA registry. When no hashing is used (that is, in the certificate types where the full certificate is given), the hash type MUST be 0. 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 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 through 4 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.

2.1. Making Certificate Associations

Items received in TLSA resource records can be treated like trust anchors by the TLS client. The trust anchor MUST NOT be loaded for longer than the TTL on the TSLA record.

The TLS client determines whether or not the certificate offered by the TLS server matches the certificate association in the TLSA resource record. If the certificate from the TLS server matches, the TLS client accepts the certificate association. Each certificate type has a different method for determining matching.

For types 1 and 3, the hash used in the comparison is the hash type from the TLSA data.

Types 1 (hash of an end-entity certificate) and 2 (full end-entity certificate) are matched against the first certificate offered by the TLS server. With these two types, the trust anchor is used only for exact matching, not for chained validation. For type 1, the certificate association is valid if the hash of the first certificate offered by the TLS server matches the value from the resource record. For type 2, the certificate association is valid if the certificate in the TLSA data matches to the first certificate offered by TLS.

Type 3 (hash of certification authority's certificate) can be used in one of two ways. 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 repositor, and the hash of that trust anchor matches the value from the TLSA data, then the certificate association is valid.

Type 4 (full certification authority's certificate) 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.

[[Need discussion of self-signed certificates being CA certificates. Need to be sure that this discussion uses correct PKIX terminology and is carefully explained.]]

2.2. 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 2) of an end-entity certificate (type 1) would be:

```
www.example.com. IN TLSA (
   1 2 5c1502a6549c423be0a0aa9d9a16904de5ef0f5c98
       c735fcca79f09230aa7141 )
```

An example of an unhashed (type 0) CA certificate (type 4) would be:

```
www.example.com. IN TLSA (
   4 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.

2.3. Wire Format

The wire format is:

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Cert type   |   Hash type   |                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               /
/                                                               /
/                    Certificate for association                /
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

```
www.example.com. IN TYPE65534 \# 34 ( 01025c1502a6549c423be0a0aa
                      9d9a16904de5ef0f5c98c735fcca79f09230aa7141 )
```

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

```
www.example.com. IN TYPE65534 \# 715 0400308202c5308201ada003020...
```

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 a match between the certificate association(s) and the server's end entity certificate in TLS is not found, the TLS client MUST abort the handshake with an "access_denied" error.

4. IANA Considerations

4.1. 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.

4.2. 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	Ref.
0	Reserved	[This]
1	Hash of an end-entity cert	[This]
2	Full end-entity cert in DER encoding	[This]
3	Hash of an CA's cert	[This]
4	Full CA's cert in DER encoding	[This]
5-254	Unassigned	

Applications to the registry can request specific values that have

yet to be assigned.

4.3. 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	Ref.
0	No hash used	[This]
1	SHA-1	NIST FIPS 180-2
2	SHA-256	NIST FIPS 180-2
3	SHA-384	NIST FIPS 180-2
4-254	Unassigned	

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

5. Security Considerations

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

A DNS administrator who goes rogue and changes both the A 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.

[[Add discussion of the idea that TLSA makes things worse if an intermediate CA is compromised. Need more from Stephen Farrell.]]

6. 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, Ilari Liusvaara, and Ondrej Sury.

7. References

7.1. 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.

7.2. 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